TASK 5.2D – SIT225

TRUONG KHANG THINH NGUYEN

For the task 5.2D, I used MongoDB for cloud-based database and CouchDB for local-based database as two alternatives to Firebase since I want to have different perspectives in terms of storing in local or cloud.

COMPARISON OF DIFFERENT DATABASES

-Firebase

Firebase offers a faster setup method and a fully managed backend. Configuration is really simple, especially when integrating with mobile and web applications, thanks to the full SDKs and detailed documentation.

Firebase offers two main data storage solutions: Realtime Database and Firestore. The Realtime Database is designed for realtime data synchronization, whereas Firestore supports more complex querying, including hierarchical data structures.

In terms of ease of use, Firebase is specifically intended for straightforward and simple usage with an emphasis on rapid development and deployment. Its real-time capabilities and built- in services, such as authentication and cloud operations, make it ideal for developer use, particularly in mobile app development.

-MongoDB

MongoDB, whether self-hosted or managed through MongoDB Atlas, provides greater flexibility and control over configuration. It allows a wide number of deployment options and configurations, however this flexibility may result in increased setup and administrative complexity. But for this task, I chose to implement it in cloud.

MongoDB stores data in BSON documents, which allow for sophisticated searching and indexing capabilities. It allows complicated searches, aggregations, and indexing, making it ideal for applications that require extensive and diversified data operations.

Regarding the difficulty of using it, MongoDB is versatile, but it may require more setup than Firebase, particularly for unique settings. Its broad driver support and documentation make integration easier, but the learning curve is harder for beginners compared to Firebase's more user-friendly approach.

-CouchDB

CouchDB's RESTful HTTP API makes configuration relatively simple. Its emphasis on a schema-free document architecture and ease of replication across distributed contexts make it ideal for applications that require **offline**-first functionality.

CouchDB stores data in JSON documents and queries via MapReduce. Its RESTful interface allows for easy interaction and integration with other online services. However, its querying capabilities are less advanced than MongoDB's.

CouchDB's offline-first architecture and straightforward REST API make it simple to deal with, particularly for dispersed applications. However, its querying and indexing are less sophisticated than MongoDB's, which may limit its ability to handle more complex data operations.

To conclude, Firebase is really good at in terms of usability and integration, making it perfect for rapid development and real-time apps. MongoDB provides strong querying capabilities and flexibility, making it ideal for complicated data requirements. CouchDB offers simplicity and offline capabilities, but only basic querying functions. Each offers advantages depending on the specific requirements of the application.

COMPARISON BETWEEN MQTT AND SERIAL COMMUNICATION

With respect to data transfer, Serial communication is ideal for direct, low-speed data transfer over short distances, which is commonly employed in embedded systems or connecting microcontrollers to sensors. MQTT, on the other hand, is intended for scalable, efficient messaging across networks, making it ideal for IoT and cloud-based applications.

Talking about the parsing and management, Serial communication requires manual parsing and manipulation of data frames, which can be difficult and error prone. MQTT abstracts these issues by providing built-in support for message routing, delivery guarantees, and topic-based subscription, making it easier for developers to conduct their tasks.

And for applications, Serial communication is suitable for direct device-to-device communication in limited settings. MQTT is appropriate for distributed systems, where devices must communicate over a network under changing conditions and message dependability and scalability are critical.

In conclusion, MQTT is generally more efficient for networked applications due to its lightweight design and support for several QoS levels, whereas serial communication's performance is restricted by its simplicity and physical limits. It really depends on the specific requirements of the current applications.