Serverless architecture because

* Scales automatically (good for my app due to unpredictable usage patterns – intermittent traffic)
* Good for functions that tale little execution time (good for my app, cause I don’t have complex data processing)
* Short-lived event driven workloads

Java Spring Boot because

* Widely used for scalable applications (MVC and REST APIs)
* Easily deployed in a cloud native environment
* Supports microservices
* Provides authentication, authorization, encryption, and other security features
* Highly compatible with CI/CD tools like Jenkins, GitLab CI, and AWS CodePipeline
* Able to connect to cloud-native databases like Amazon RDS (for relational databases like MySQL or PostgreSQL) or Amazon DynamoDB (for NoSQL).
* Has cloud-native features like serverless computing, containers, and cloud storage
* Works with AWS services (S3 for file storage, DynamoDB or RDS for databases, Lambda for serverless processing)

Message bus vs REST

Need immediate response? → REST API

Need background processing & event-driven behavior? → Message Bus

System has many microservices that should work independently? → Message Bus

Simple CRUD operations? → REST API