New Social Platform

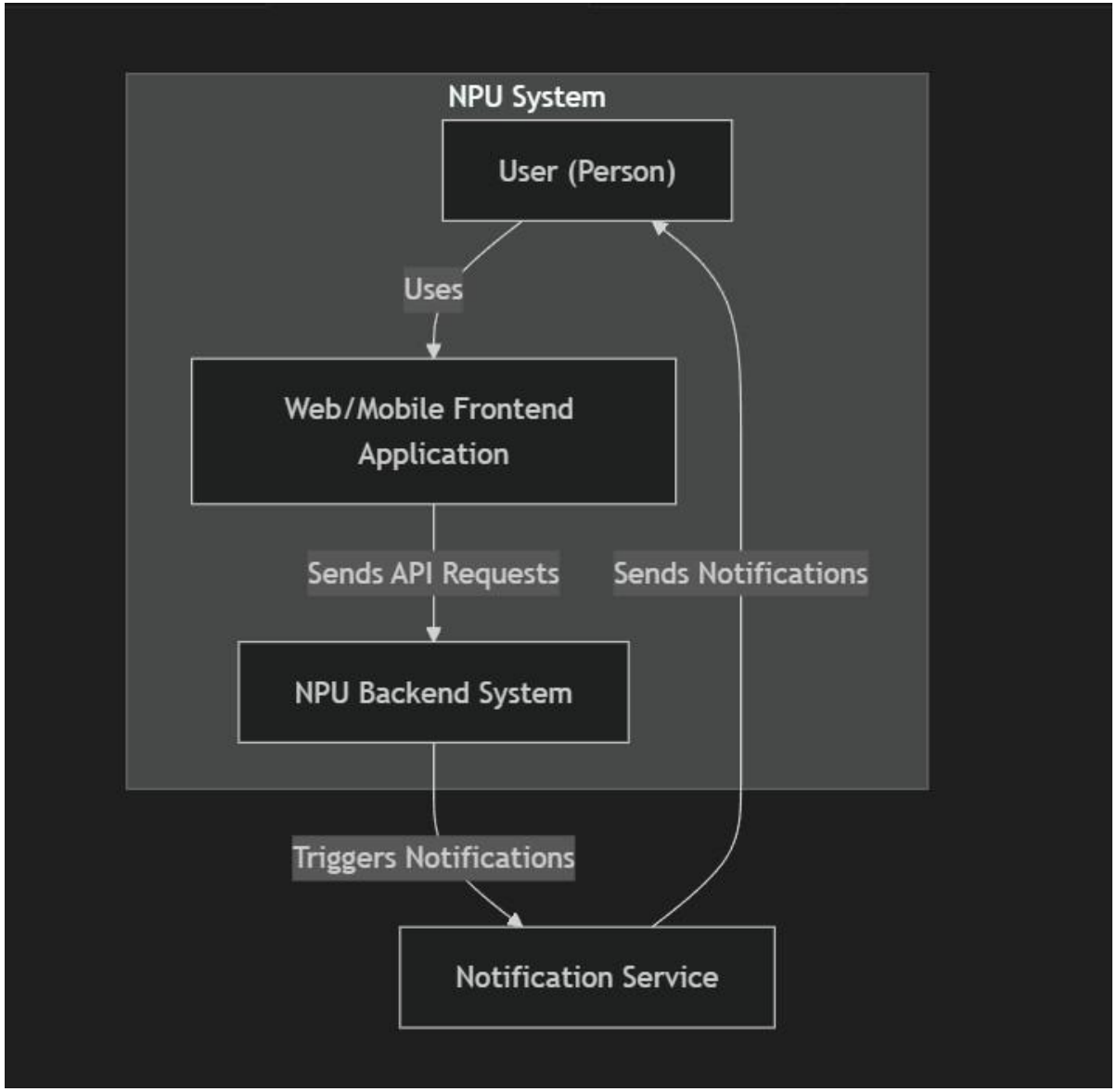
Agenda

1. Introduction
2. Architecture (C4) Overview
3. Technology Choices
4. Scalability & Resilience
5. Next Step / Q&A

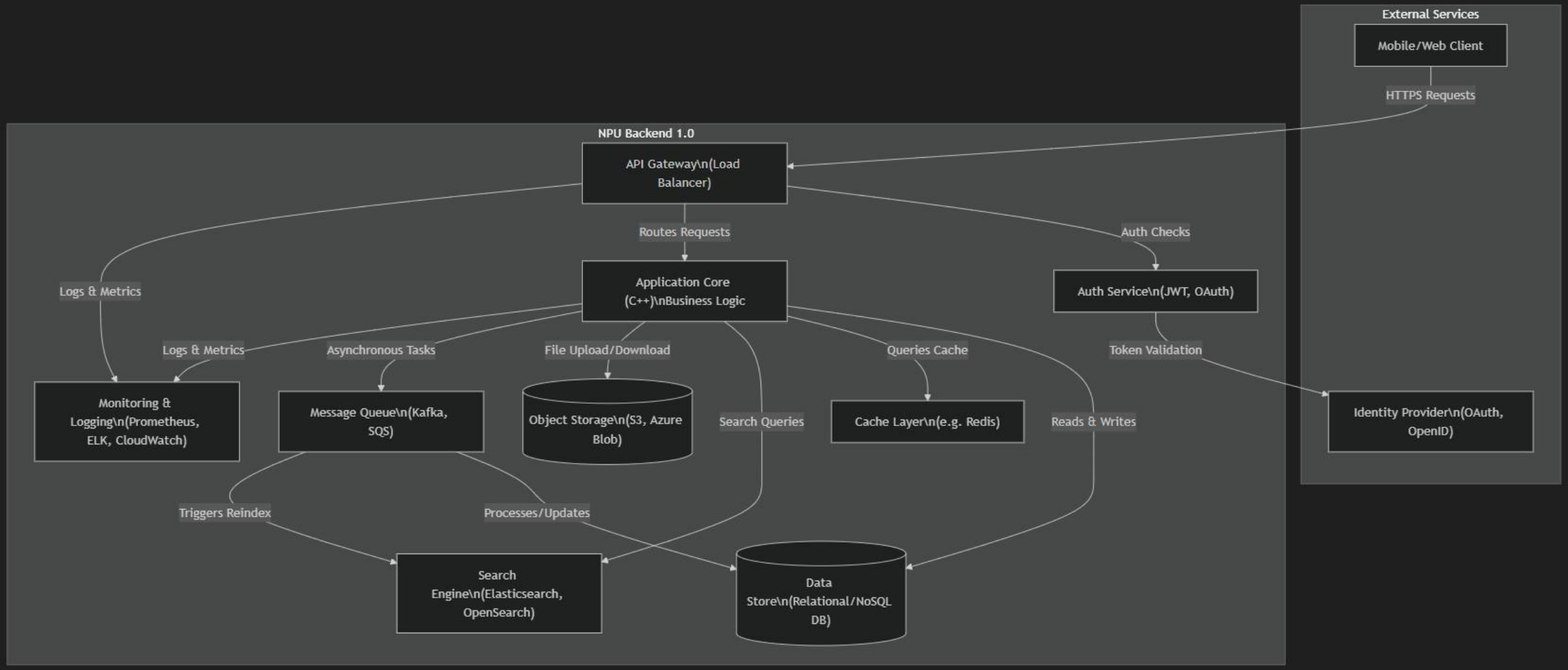
Introduction & Problem Statement

- **Goal**: Build a social platform where users upload pictures and descriptions of their “NPU” (Nice Part Usage) creations.
- **Key Features:**
  - Upload NPU images and descriptions
  - Rate other users’ creations
  - Search for ideas by element name or tags
- **Multi-Platform**: Supports both mobile and web clients.
- **Constraints**:
  - Written in **C++**
  - Runs on **cloud hosting** (AWS, Azure, etc.)
  - Present as a **C4 model** with a **Container diagram**

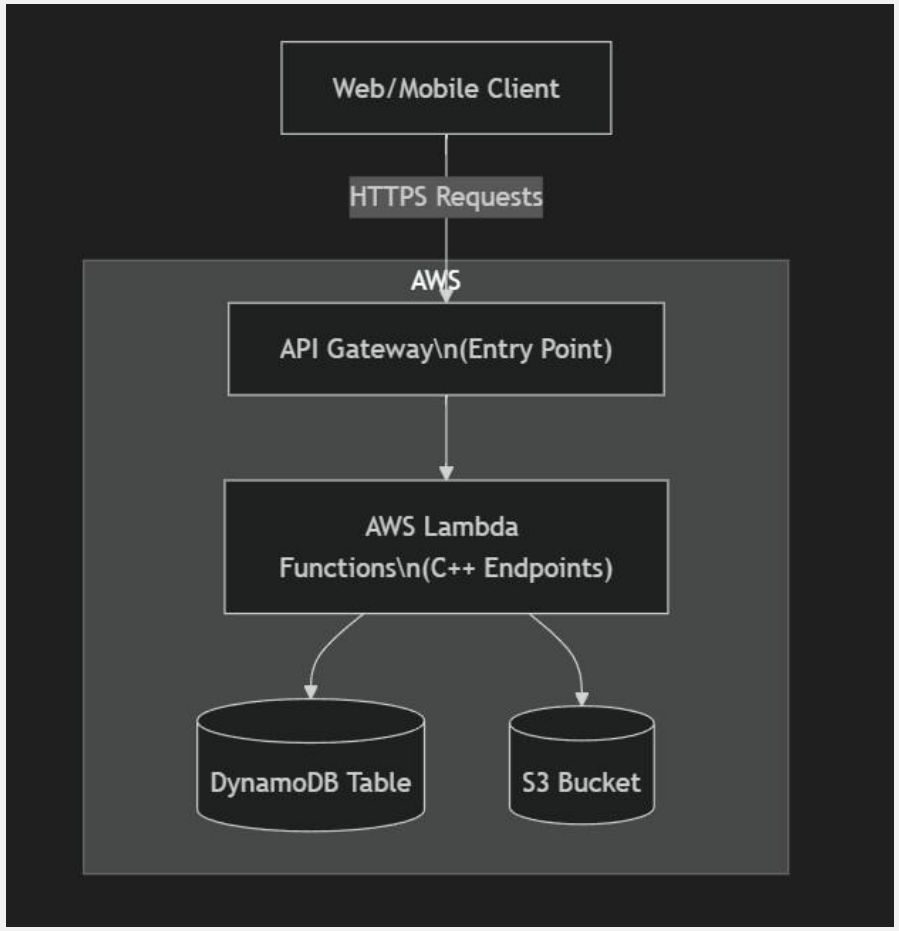
C4 Model - Context



C4 Model - Container



NPU Backend



AWS LAMBDA (C++): EACH ENDPOINT IS A SEPARATE LAMBDA FUNCTION.

API GATEWAY: ROUTES EXTERNAL HTTP REQUESTS TO THE APPROPRIATE LAMBDA FUNCTION.

DYNAMODB: STORES USER DATA, RATINGS, CREATION DESCRIPTIONS.

S3: STORES IMAGES (USER-UPLOADED NPU PICTURES).

Key Technology Choices

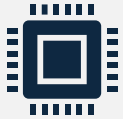
Language: C++

AWS Lambda

Serverless compute model.

Quick to deploy and scales automatically.

No need to manage servers for the short assignment timeline



DynamoDB

Integrates seamlessly with AWS Lambda and API Gateway

NoSQL database, fully managed by AWS.

Highly scalable for key-value operations.

Amazon S3

Object storage for images (NPU photos).

Durable and easy to manage.

Supports large file uploads without impacting the compute la

AWS Integration

All services (Lambda, DynamoDB, S3, API Gateway) work together seamlessly.

Minimizes operational overhead and setup time.

Scalability & Resilience

- **Lambda Auto-Scaling**: Spawns more function instances under load.
- **DynamoDB Auto Scaling**: Adjusts read/write throughput to handle spikes.
- **S3 Durability**: Replicates objects across multiple Availability Zones.
- **API Gateway Throttling**: Protects backend from excessive requests
- **Future Enhancements**: Could integrate a search engine (OpenSearch) or caching layer (Redis) if needed.

Conclusion & Q&A

- M**inimal Serverless Approach**: Using AWS Lambda + DynamoDB + S3 allows rapid development.
- **C++ in Lambda**: Meets the assignment's language requirement while staying cloud-native.
- T**rade-Offs**: DynamoDB is great for speed, but relational features are limited.
- Next Steps:
  - Introduce CI/CD pipeline for smooth deployments
  - Implement authentication (Cognito or Auth)
  - Add advanced search (OpenSearch)