DALHOUSIE UNIVERSITY

CSCI 5411 – Mid Term Project

Quiz UP

Study Notes & Quiz Generator with RAG on AWS

Name: Shray Moza
ID: B00987463

1) Domain Selection & Justification

App Name: Quiz up

Domain: Academic productivity with the help of **note generation** and **quiz creation** from lecture PDFs or notes.

Problem Statement: Students spend most of their time summarizing lecture material and creating practice questions to get a better understanding. Manual effort is inefficient and prone to missing key concepts. We want to create a system that ingests PDFs/notes, extracts core ideas, generates concise study notes, and produces formative assessments.

Business Case: - Increases student productivity and retention by converting raw materials into structured notes and quizzes. Supports retrieval of relevant content on demand with a Retrieval-Augmented Generation (RAG) workflow.

For the implementation I will be using Amplify, Cognito, SageMaker, S3, Lambda, and API Gateway and Cloudwatch. These services are sufficient to support hosting, inference, and secure access.small open-source models and a pragmatic vector search, enabling a fully in-house pipeline.

---

2) AI Approach & Techniques

1. RAG pipeline
2. Embeddings with MiniLM
3. FAISS vector store

---

3) Functional Requirements

1. **Upload & Storage:**
   - ❖ Users upload PDFs.
   - ❖ System stores originals and outputs in S3.
2. **Preprocessing:** Extract, clean, and chunk text.
3. **Embedding & Indexing:**
   - ❖ Compute embeddings for chunks.
   - ❖ Build FAISS index.
   - ❖ Persist to S3.
4. **Query / Task Types:**
   - ❖ Study notes for a document/topic.
   - ❖ A 5-question quiz (MCQ) with an answer key.
5. **RAG Pipeline:**
   - ❖ Retrieve top-k relevant chunks.
   - ❖ Compose prompt with context.

- Call model for generation.

6. **Delivery:** Return generated notes/quiz to user and store artifacts in S3.
7. **Monitoring:** Basic metrics and logs via CloudWatch (Lambda + SageMaker endpoint/notebook).
8. **Auth:** Cognito user pool for authenticated access.

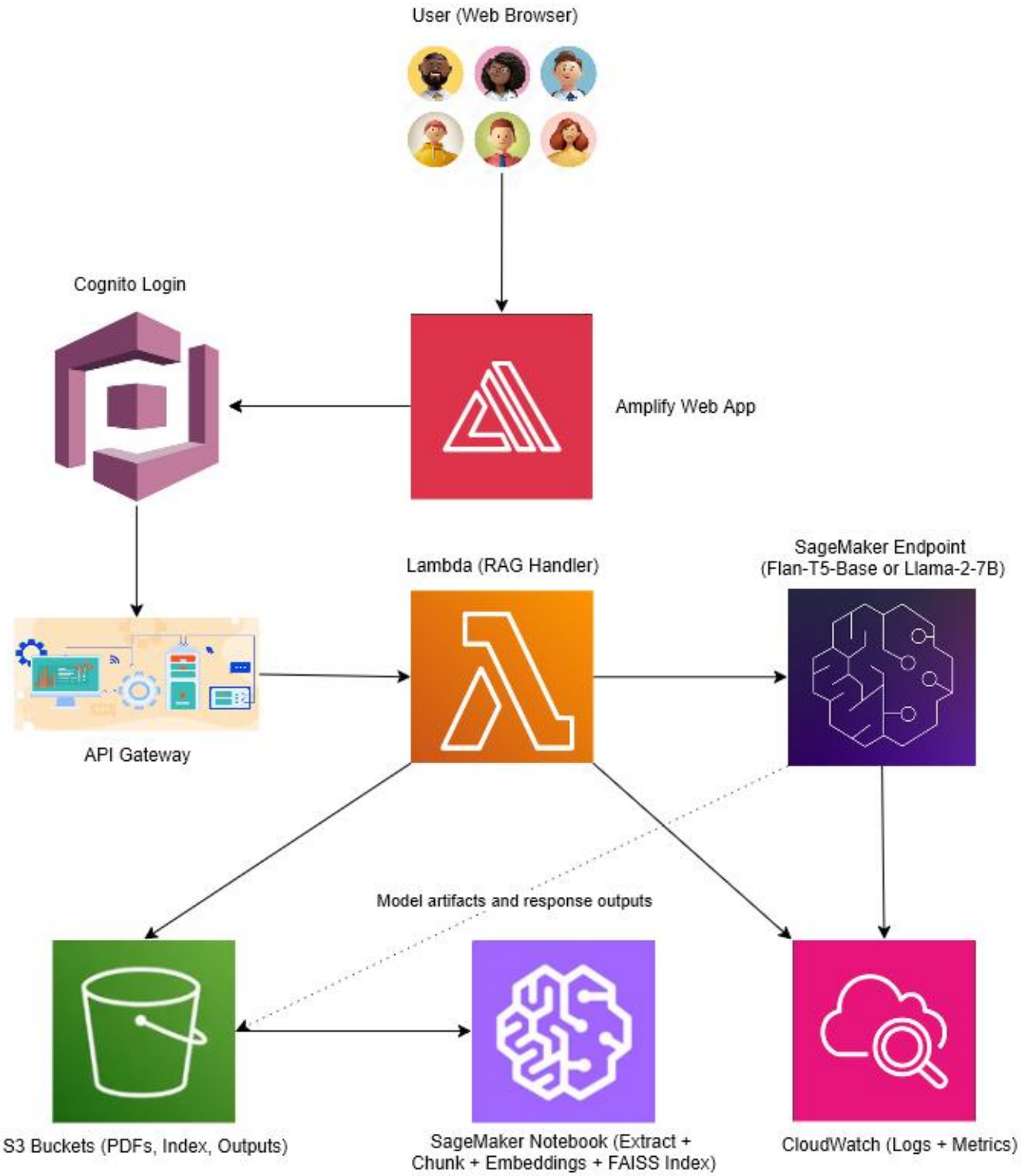
---

4) Non-Functional Requirements

- **Security:**
  - IAM least privilege.
  - Private S3 buckets.
  - VPC for SageMaker endpoint.
  - HTTPS via API Gateway.
  - Cognito for authenticated calls.
- **Scalability:**
  - Serverless API (Lambda/API Gateway).
  - SageMaker endpoint scales by instance size.
  - Notebook-only dev operations for batch jobs.
- **Reliability:**
  - S3 durability for inputs/outputs.
  - Index stored redundantly in S3.
  - Retriable Lambda steps.
- **Performance:**
  - End-to-end generation target < 5–8s for small inputs.
  - Batch builds for large PDFs.
  - Caching common prompts.
- **Cost Efficiency:**
  - Small instance classes (ml.t3.medium, ml.c5.large).
  - Endpoint up only during demos.
  - Notebooks stopped when idle.
  - CloudWatch log retention controls.
- **Observability:**
  - CloudWatch metrics/alarms for Lambda errors and SageMaker invocation latency.
  - S3 access logs enabled.

---

5) Initial Architecture Design (made with draw.io)



5.1 Services

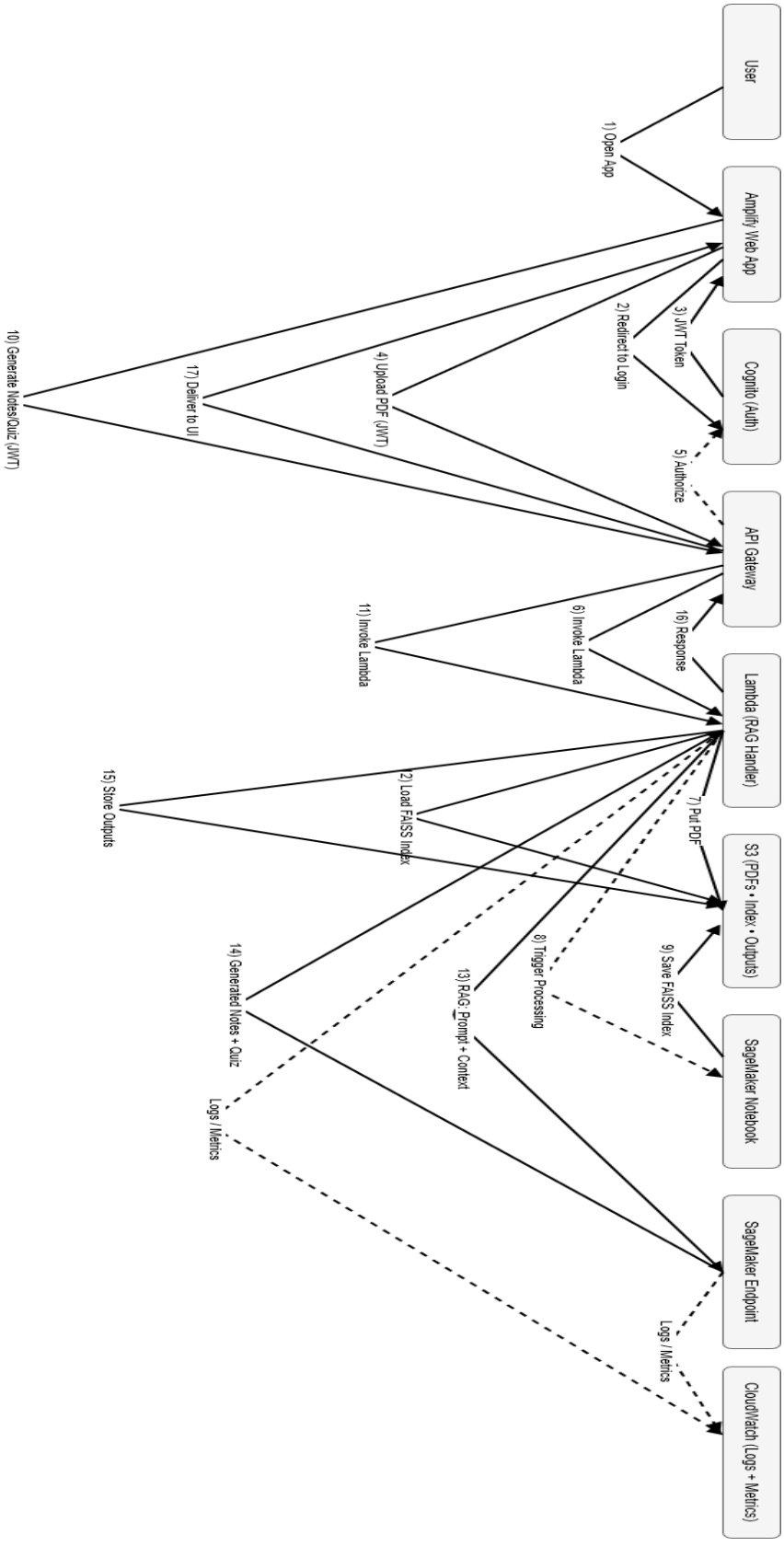
1. **S3** – durable storage for PDFs, indexes, and generated outputs.
2. **Amplify** – Hosting the web app, easy auth integration, CI/CD
3. **SageMaker Notebook** – orchestrates data prep, embedding, and FAISS index build.
4. **FAISS (in Notebook/Code)** – vector search (no managed vector DB required).
5. **SageMaker Realtime Endpoint** – hosts a small open-source text-generation model for consistent inference via HTTPS.
6. **Lambda** – RAG API: loads FAISS index from S3, retrieves chunks, builds prompts, calls SageMaker endpoint.
7. **API Gateway** – HTTPS interface for UI/clients.
8. **Cognito** – authentication and JWT validation at API Gateway.
9. **CloudWatch** – logs/metrics/alarms.

5.2 Architecture Description

1. User opens webapp hosted on **Amplify.**
2. User authenticates using **Cognito**.
3. User uploads PDF via UI → **API Gateway** → **Lambda** → stores in **S3**.
4. Processing job triggered: **SageMaker Notebook** reads PDF from **S3**, extracts & chunks text, computes embeddings, builds **FAISS** index, and saves index.faiss + docstore.pkl to **S3**.
5. On generation request: **Lambda** loads FAISS index from **S3**, retrieves top-k chunks for the prompt, calls **SageMaker Endpoint** with prompt+context, returns generated notes/quiz to UI, and stores outputs in **S3**.
6. **CloudWatch** collects logs/metrics.

---

6) Data Sequence Diagram (made with draw.io)



7) AWS Services & Tech Stack

1. **Compute/ML:** SageMaker Notebook, SageMaker Realtime Endpoint (Flan-T5-Base default)
2. **Vector Search:** FAISS (Python), artifacts in S3
3. **Embeddings:** sentence-transformers all-MiniLM-L6-v2
4. **API:** API Gateway, Lambda
5. **UI:** Amplify
6. **Storage:** S3 with separate prefixes for raw, index, outputs
7. **Auth:** Cognito user pool & authorizer
8. **Monitoring:** CloudWatch Logs, basic alarms on error rates/latency.

---

8) Security Analysis

1. **IAM Least-Privilege:** Separate roles for Notebook, Endpoint, Lambda. Each has scoped permission:
   - ❖ Lambda can GetObject from indexes/ and raw/ prefixes and PutObject to outputs/ only.
   - ❖ Notebooks can read raw/, write indexes/.
   - ❖ Endpoint can read models from ECR/S3 as needed.
2. **S3 Hardening:**
   - ❖ Block public access.
   - ❖ Bucket policies restricted specific IAM roles.
   - ❖ Server-side encryption (SSE-S3).
   - ❖ Enable access logs.
3. **Network:** Security groups to limit access.
4. **API:**
   - ❖ HTTPS via API Gateway.
   - ❖ Cognito authorizer.
   - ❖ Request/response size limits.
5. **Secrets:**
   - ❖ No hard-coded keys.
   - ❖ Rely on IAM roles.
6. **Data Handling:**
   - ❖ PDFs and outputs stored under per-user prefixes.
   - ❖ Redact PII in logs.

---

9) Identification of Potential Architectural Challenges

1. **Instance Limits:**
   - ❖ Learner Lab restricts instance families/sizes.
   - ❖ Large models won't deploy → use small instruction-tuned models.

2. **Service Availability:** OpenSearch/others may be unavailable → FAISS-on-S3 fallback is primary.
3. **Endpoint Cost & Warmup:**
    - ❖ Keep endpoints off when not in use.
    - ❖ Pre-warm before demos.
4. **Latency:** Loading FAISS from S3 per request may add latency → cache index in memory via provisioned Lambda or warm start pattern.
5. **Token Limits:** Use chunking and concise prompts to return summaries section-wise for long inputs.

---

10) Initial Cost Estimation (conservative, 1–2 weeks of development/use)

Service	**Assumption**	**Est. Cost**
SageMaker Notebook	ml.t3.medium,few hours/day, stopped when idle	$3–$8
SageMaker Endpoint	ml.t3.medium, used 4-5 hrs/day for 5-7 days	< $15
Lambda + API Gateway	Low traffic dev/test	<$3
S3 (data in GBs) + Transfer	PDFs + indexes + outputs	<$2
CloudWatch Logs	Basic retention, low volume	$1–$2
Total		**< 40**
