New Project Template: OCR Receipt Reader

1. Introduction

1.1 Purpose

This document outlines the architectural design plan for a Receipt Reader program. This document is intended for the project development team. This will help guide the initial development of the project.

1.2 Scope

This architecture covers a receipt reader program that will read information from a receipt picture and store information of the purchase.

1.3 Definitions, Acronyms, and Abbreviations

- OCR: Optical Character Recognition
- API: Application Programming Interface
- DB: Database
- S3: AWS Simple Storage Service
- DynamoDB: AWS Database Service
- IMAP: tool used to interact with email
- GSI: Global Secondary Index

1.4 References

- IMAP documentation
- Azure Vision documentation
- AWS Documentation

1.5 Overview

This document will proceed to inform architectural style, possible concerns, high-level system overviews, architecture design, architecture strategies and key decisions for implementation.

2. Architectural Representation

2.1 Architectural Style and Rationale

This project will focus on a Monolith Architectural Design to simplify development and deployment. This is a small project that focuses on implementing designed functionality and not a high quantity of user input.

3. Concerns

3.1 System Concerns

- Performance: Ensure that different types of receipts can be read, and proper data can be extracted.
- Scalability: Ability to store a number of receipts and data long term.

- Security: Ability to maintain user information/email secure.

4. System Overview

4.1 High-Level Description

OCR receipt reader allows users to get information about a receipt by sending it to an email adress and storing the receipt and the purchasing information in a database as well as providing a table in Notion for the user.

5. Architectural Strategies

5.1 Key Strategies

Given the current experience of the development team and the requirements for this project, the following technologies will be used:

- IMAP Script for email interaction, this will allow the receipts attached to the email to be retrieved.
- AWS S3 for storing receipt image, by keeping the original image of the receipt store it allows some features to be implemented if necessary:
    - OCR re-runs
    - Receipt display to user
    - Debugging or manual verification
    - Full trail audit (original source of “truth”)
- AWS DynamoDB for storing receipt data. Database will hold receipt information. A NoSQL database is great for lightweight, structured data, like receipts.

5.2 Error Handling Strategies

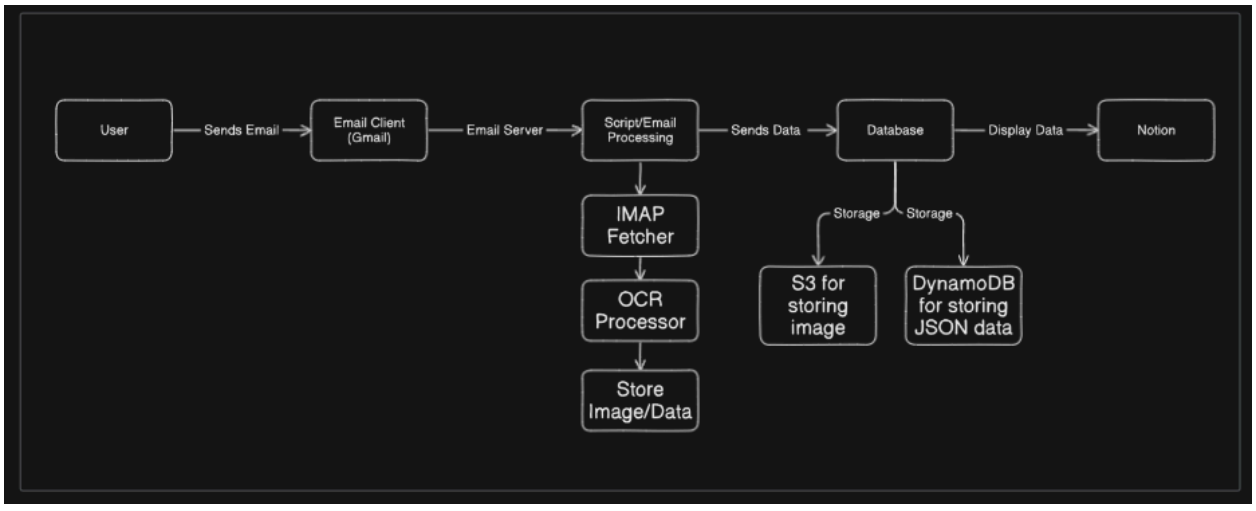
- Implementing retry logic in case of network issues when connecting to external APIs (AWS S3/DynamoDB, Azure Vision, Notion).
- Log errors for manual review at a later time.
- Emails that fail processing should be flagged if they fail repeatedly.
- Script should still run other emails even if one fails.

6. System Architecture

6.1 Overview of Layers/Modules

- Front-End: notion tables for displaying information retrieved from receipts.
- Back-End: Manages storage of images, retrieved data, and application logic.
- Database: Stores data and images for situations where data/images need to be re-used.

6.2 Component Diagrams



- Link: https://app.eraser.io/workspace/sznWy0US7n90bJeXkCh4?origin=share&elements=luzOnasf2GseXs5zhVNVRw

6.3 Database Design

```
{
  "user_email": “user_email",
  “upload_timestamp_id”: “date_time_string + # + UUID”,
  “receipt_id”: UUID,
  "store": "store_name",
  “receipt_timestamp”: “date_time_string”,
  "total": purchase_total,
  "items": [
        {description: “item_description”, quantity: 1, cost: 1.99},
        {description: “item_description”, quantity:1, cost: 1.99},
            ],
  "receipt_url": "aws_s3_url"
}
```

Attribute	Type	Role/Key	Description
user_email	S (String)	Partition Key (PK)	email address of the user
upload_timestamp_id	S (String)	Sort Key (SK)	Upload date and time + # +UUID
receipt_id	S (String)	Attribute	UUID
store	S (String)	Attribute	Name of store
receipt_timestamp	S (String)	Attribute	Date and time on receipt
total	N (Number)	Attribute	Total value of purchase
items	L (List)	Attribute	List of items purchased
receipt_url	S (String)	Attribute	S3 url receipt image

Inside Items(attribute):

Attribute	Type	Role/Key	Description
Description	S (String)	Attribute	Description of product
Quantity	N (Number)	Attribute	Quantity of Product bought
Cost	N (Number)	Attribute	Cost of 1 Unit of product

7. Key Architectural Decisions

7.1 Decision Log

- Monolith Architecture: Chosen due its simplicity for development and deployment.
- Tech Stack: based on budget, functionality and team experience.

8. Quality Attributes

8.1 Performance

The initial performance goal for this OCR Receipt reader is to retrieve data from a receipt, store it in a database and display the information to the user.
Since this is a monolith script process the actual time is dependent on the responsiveness of external APIs and requires testing.
Most of the technologies being used for this application are high performance and low latency.
Also, this version of the application is relying on local/manual execution, once moved to a cloud base architecture (AWS Lambda) it can potentially scale to a high-volume operation.

8.2 Scalability

- This project initially won't scale automatically if the number of emails increases, since it will be run locally when needed.
- The script which allows this program to work will be run locally and manually.

8.3 Security

- Secure handling and storage of credentials for IMAP login, AWS and Azure keys, and Notion API tokens.

8.4 Maintainability

Following best code practices and good documentation for maintenance.

9. Risks and Technical Debt

9.1 Identified Risks

- Technical Challenges:
    - OCR ability to read different receipts, and variable quality will affect relevancy and accuracy of data stored.
    - Integrating different technologies.
    - Different receipts having all the information needed by the database.

9.2 Technical Debt

Potential need for Global Secondary Indexes (GSI) for complex queries later.

10. Appendices

10.1 Glossary

Provide a glossary of terms used throughout the document.

10.2 Index

Include an index of terms and sections for easy navigation.

10.3 Revision History

Document the revision history of this document.