GCP PROJECT (PART 2)

By: David Durães Valadares

CLOSO

Contents

Architecture Description

This project implements an AI-enhanced document processing system on Google Cloud Platform that automatically extracts structured data from PDF invoices. The architecture is designed to be simple, scalable, and fully serverless while demonstrating effective use of core Google Cloud services.

When an invoice is uploaded, it triggers an automated pipeline that uses machine learning to extract relevant data and store it in a data warehouse for analysis. No manual intervention is required after the upload.

Overall Architecture

The solution is built using Cloud Storage, Cloud Functions, Document AI, and BigQuery, with a Cloud Run–based web dashboard for user interaction and result visualization. Cloud Storage serves as the entry point to the workflow. Uploading a PDF invoice automatically triggers a Cloud Function, which manages the entire processing logic. This includes sending the document-to-Document AI, transforming the extracted data, and storing the final result in BigQuery. The web dashboard allows users to upload invoices and view processed data through a simple interface.

This serverless, event-driven design reduces infrastructure management and allows the system to scale automatically.

Component Description

Cloud Storage

Cloud Storage is used to store uploaded PDF invoices. The storage bucket is configured to trigger an event whenever a new file is added. This event is what starts the processing pipeline.

Cloud Functions

A second-generation Cloud Function is triggered when a new invoice is uploaded. The function retrieves the PDF from Cloud Storage, sends it to Document AI for processing, and then prepares the extracted data for storage. It also includes basic error handling and logging to ensure that failures can be identified and debugged.

Document AI

Document AI's pre-trained Invoice Parser is used to extract structured information from invoices. It identifies key fields such as invoice numbers, dates, supplier details, line items, and totals. Using Document AI avoids the need to build a custom OCR or parsing solution and provides reliable results for standard invoice formats.

BigQuery

BigQuery is used to store the extracted invoice data. The data is saved in a structured table that supports nested fields for line items. This makes it easy to run queries, analyze invoice data, and display results in the dashboard.

Cloud Run (Web Dashboard)

The web dashboard is deployed on Cloud Run and provides a simple way for users to upload invoices and view processed results. It connects directly to Cloud Storage for uploads and BigQuery for querying extracted data.

Data Flow

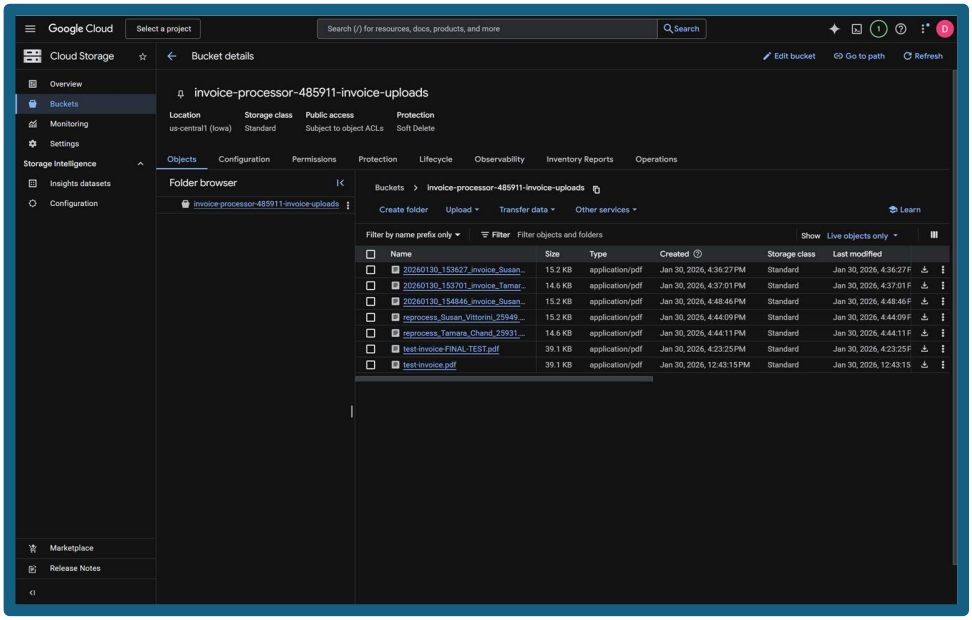
1. A user uploads a PDF invoice using the web dashboard or directly into the Cloud Storage bucket.
2. Once the upload is complete, Cloud Storage generates an event.
3. This event triggers the Cloud Function automatically.
4. The Cloud Function sends the invoice to Document AI for processing.
5. Document AI extracts structured data from the document and returns the results.
6. The Cloud Function formats the data and stores it in BigQuery.
7. The processed invoice data becomes available for viewing and analysis through the dashboard.

Key Characteristics

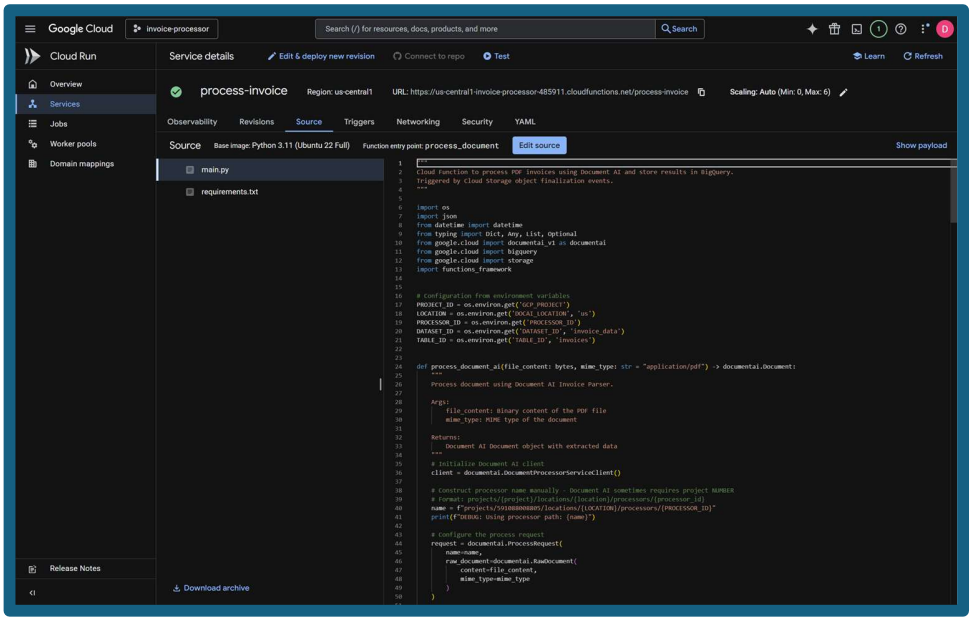
- The system runs automatically after a file is uploaded.
- All components are serverless and scale as needed.
- There is no manual processing once the invoice is submitted.
- The architecture is secure and cost-effective due to managed cloud services.
- This architecture demonstrates how multiple Google Cloud services can be combined to build a practical, real-world document processing solution using AI.

Implementation Evidence

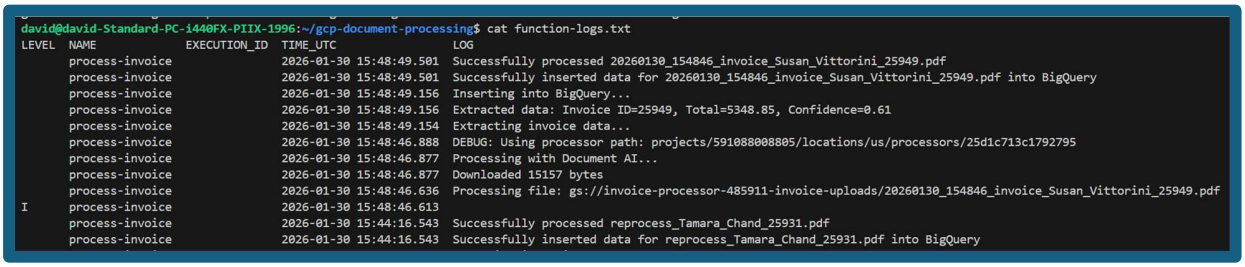
1. Cloud Storage



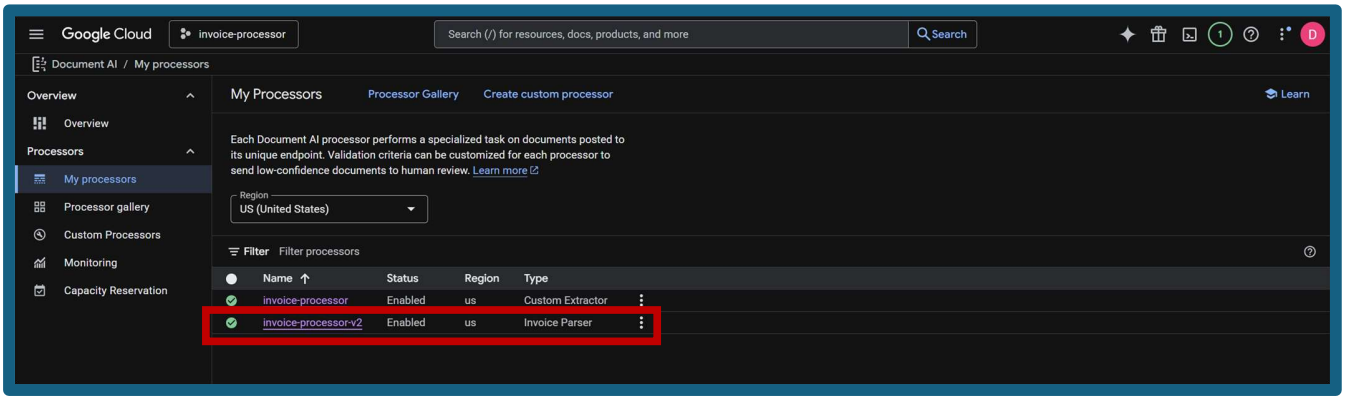
2. Cloud Functions



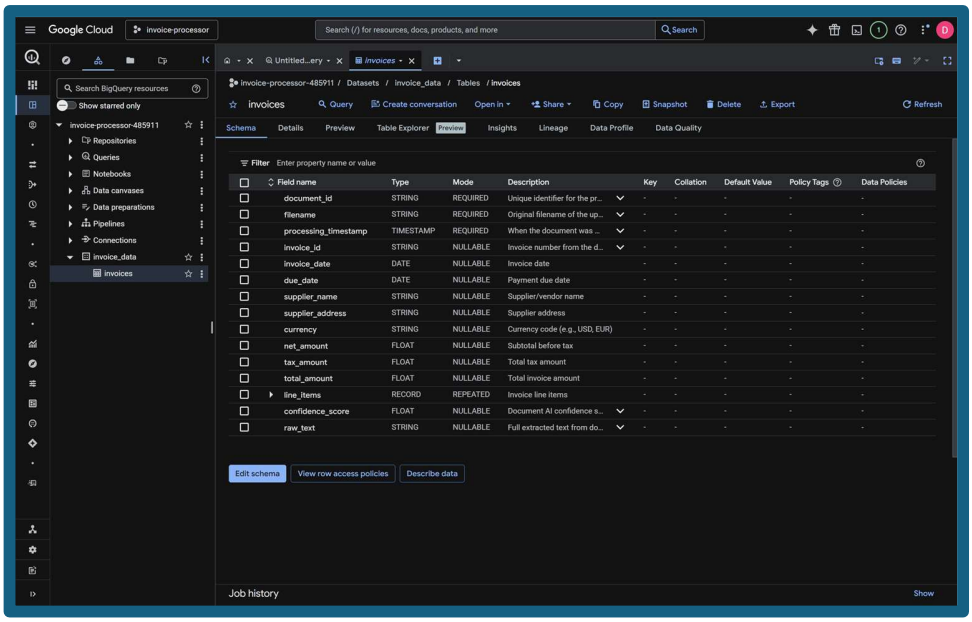
3. Cloud Function Logs



4. Document AI processor



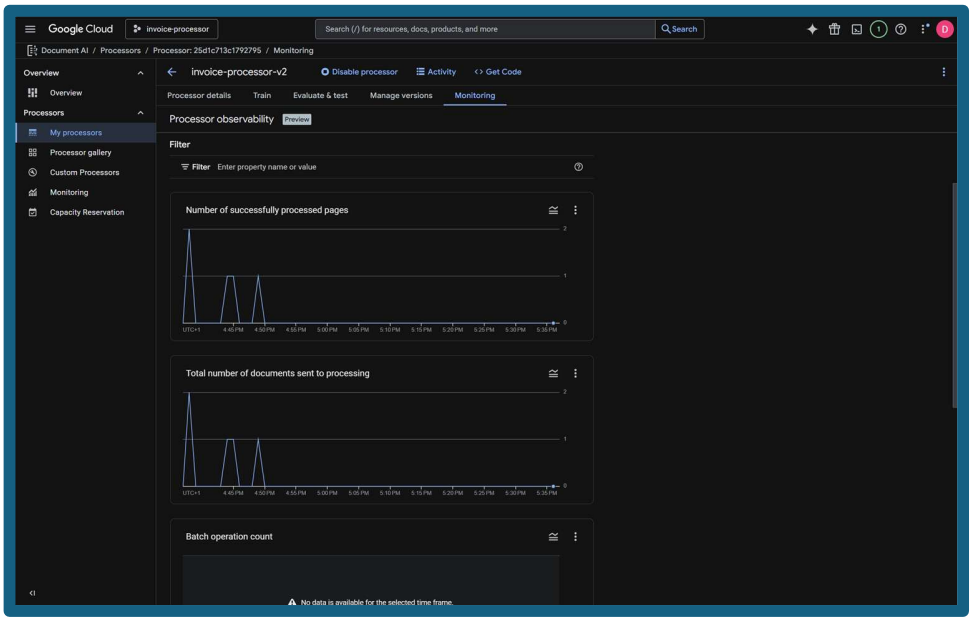
5. BigQuery



6. BigQuery - Query Results

```
david@david-Standard-PC-i440FX-PIIX-1996:~/gcp-document-processing$ export PATH="$HOME/google-cloud-sdk/bin:$PATH"
david@david-Standard-PC-i440FX-PIIX-1996:~/gcp-document-processing$ bq query --use_legacy_sql=false '
SELECT
    filename,
    invoice_id,
    invoice_date,
    supplier_name,
    total_amount,
    currency,
    confidence_score,
    processing_timestamp
FROM `invoice-processor-485911.invoice_data.invoices`
ORDER BY processing_timestamp DESC
LIMIT 10'
+-----------------------------------------------+------------+--------------+-----------------+--------------+----------+--------------------+----------------------+
|                   filename                    | invoice_id | invoice_date |  supplier_name  | total_amount | currency |  confidence_score  | processing_timestamp |
+-----------------------------------------------+------------+--------------+-----------------+--------------+----------+--------------------+----------------------+
| 20260130_154846_invoice_Susan_Vittorini_25949.pdf | 25949      |   2012-06-02 | NULL            |      5348.85 | $        | 0.6128489577677101 |  2026-01-30 15:48:49 |
| reprocess_Tamara_Chand_25931.pdf              | 25931      |   2012-05-05 | NULL            |      3469.66 | $        | 0.7337410241598263 |  2026-01-30 15:44:16 |
| reprocess_Susan_Vittorini_25949.pdf           | 25949      |   2012-06-02 | NULL            |      5348.85 | $        | 0.6128489577677101 |  2026-01-30 15:44:13 |
| test-invoice-FINAL-TEST.pdf                   | US-001     |   2019-11-02 | East Repair Inc. |       154.06 | $        | 0.9604797955047388 |  2026-01-30 15:23:30 |
+-----------------------------------------------+------------+--------------+-----------------+--------------+----------+--------------------+----------------------+
```

7. Processor Monitoring

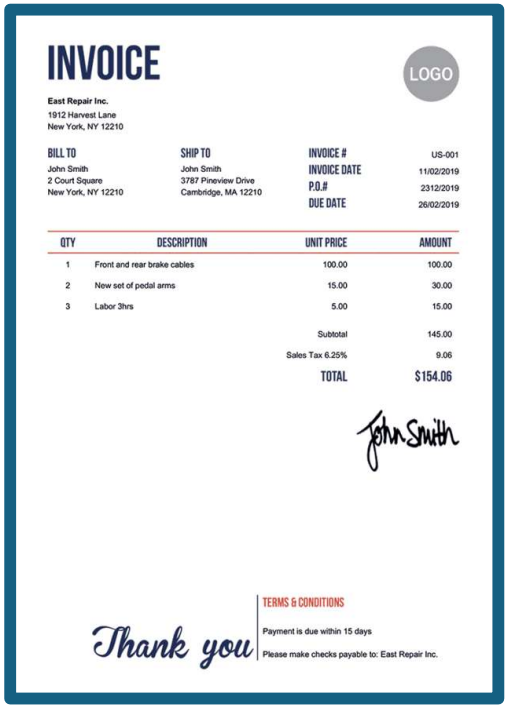


Validation Results

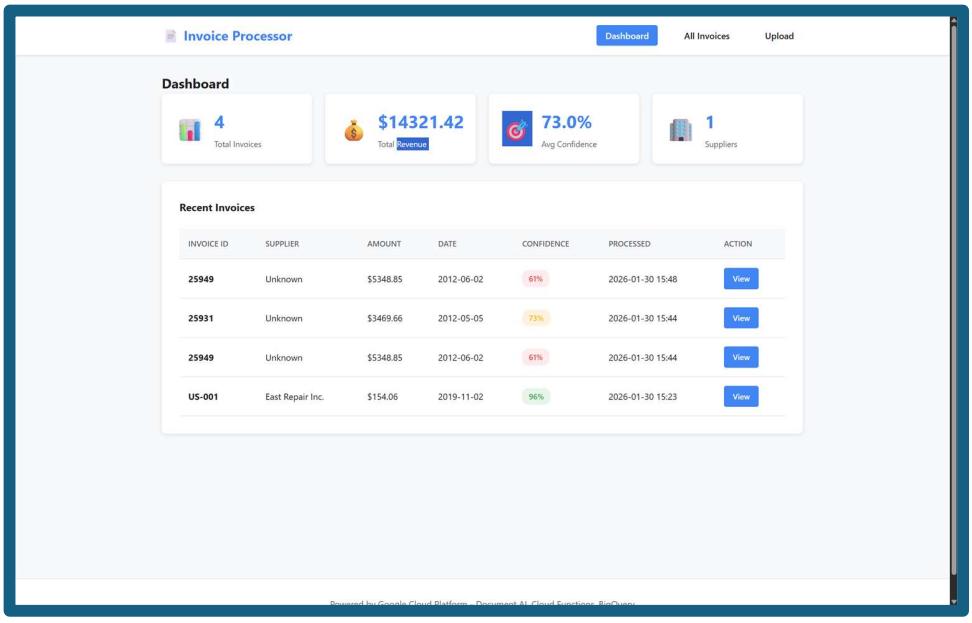
You can test the application using the link below:

Link: https://invoice-dashboard-ewknksf4wa-uc.a.run.app/

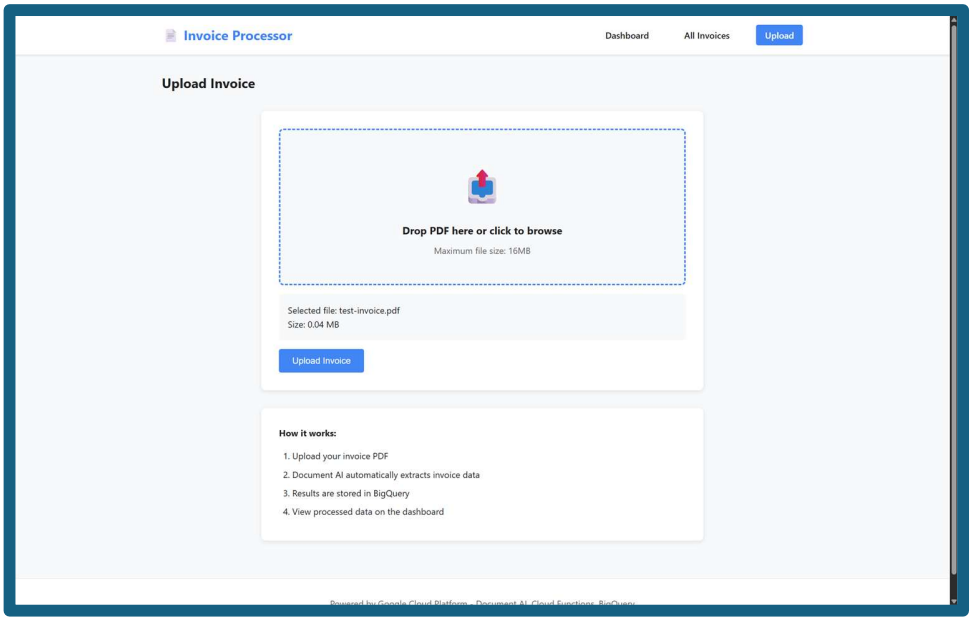
Example input:



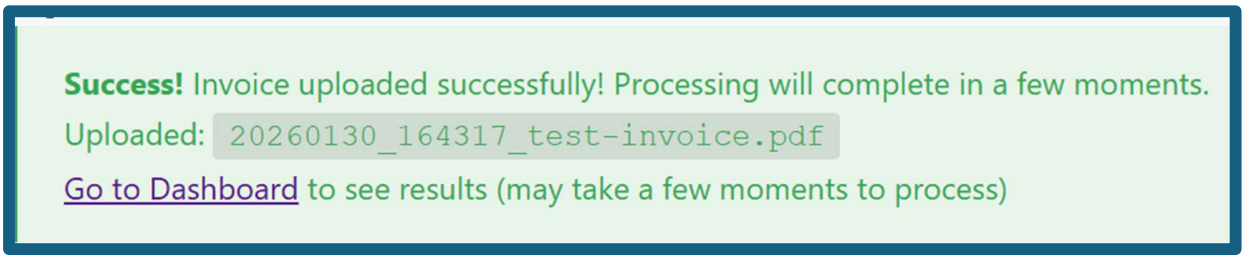
1. Open the dashboard and click **Upload**.



2. Drag and drop an invoice PDF, then click **Upload Invoice**.

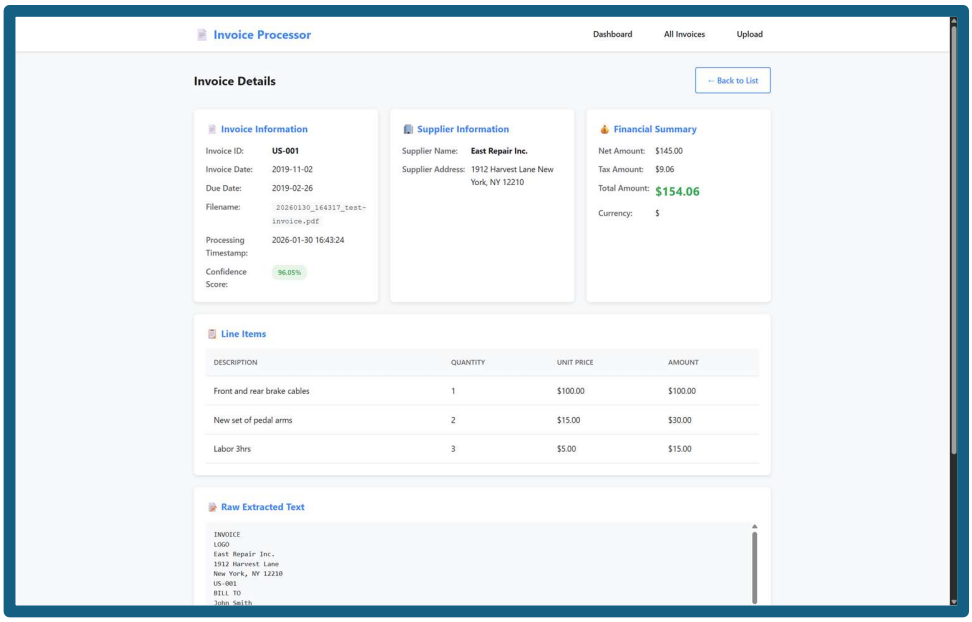


3. After a success message appears, navigate to **All Invoices** or return to the dashboard.



Output:

4. The uploaded invoice details will be displayed for review.



Reflection

The main challenges in this project were not related to coding, but to configuration details within Google Cloud that could completely block a working solution. Early on, I selected the wrong Document AI processor by creating a Custom Extractor instead of using the Invoice Parser, which caused confusing errors until I realized the issue was the processor configuration rather than my code. Another challenge was getting Cloud Functions to communicate with Document AI, as the processor path required the project number instead of the project ID. This small detail stopped the pipeline from working until it was corrected. I also encountered repeated BigQuery insertion failures because invoice dates were returned in unexpected formats such as “May 05 2012,” which required extending my date parsing logic to handle real-world variations.

I also faced deployment issues with Cloud Run, where new revisions were created but did not receive traffic. This made it appear as though my fixes were not working until I explicitly routed traffic to the latest revision.

I overcame these issues by slowing down and relying heavily on Cloud Logging instead of guessing. Carefully reading error messages, verifying configurations in the console, and testing changes incrementally helped me understand how the services actually behave rather than how I expected them to behave.

This project taught me that cloud development is less about complex code and more about how managed services interact. Small configuration or deployment mistakes can have a much larger impact than logic errors. If extended further, I would focus on improving monitoring, adding retry mechanisms, and making the system more resilient to inconsistent input data.

I also chose not to use the Pluralsight sandbox environment after hearing from classmates about its service and networking limitations. Deploying directly on Google Cloud allowed me to host a publicly accessible web application, making it easier for the instructor to fully test and validate the complete solution.