KESATRIA: Jurnal Penerapan Sistem Informasi (Komputer & Manajemen)
Terakreditasi Nomor 204/E/KPT/2022 | Vol. 5, No. 2, April (2024), pp. 669-676

Implementasi Python API dengan *Framework Flask* sebagai *Cloud Run Service* Untuk Proses *Update* di PT. XYZ

Rizky Nandang Pratama[1] , Yeremia Alfa Susetyo[2]*
[1,2]Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Indonesia
E-mail: 672020226@student.uksw.edu[1], yeremia.alfa@uksw.edu[2]

Abstract

This study focuses on implementing an API using the Flask framework in the Python programming language, which is then deployed as a Cloud Run service to facilitate data updating processes in XYZ company. By utilizing Flask as a lightweight and user-friendly API development framework, we successfully integrated it into the Google Cloud Platform's (GCP) Cloud Run environment to leverage its scalability and high performance. Our testing indicates that the Cloud Run service provides benefits in terms of automatic scalability, easy infrastructure management, and high reliability in efficiently processing data updates. Thus, the findings of this research affirm that this approach offers an effective solution for enhancing data updating processes in XYZ company, leveraging the advantages of Flask technology and the Cloud Run service..

Keywords: *Flask Framework, Python, API, Google Cloud Platform (GCP), Google Cloud Run*

Abstrak

Studi ini memfokuskan pada implementasi sebuah API menggunakan framework Flask dalam bahasa pemrograman Python yang kemudian dibungkus sebagai layanan Cloud Run untuk memfasilitasi proses pembaruan data di perusahaan XYZ. Dengan menggunakan Flask sebagai kerangka kerja pembangunan API yang ringan dan mudah digunakan, kami berhasil mengintegrasikannya ke dalam lingkungan layanan Cloud Run milik Google Cloud Platform (GCP) untuk memanfaatkan skalabilitas dan kinerja yang tinggi. Pengujian yang kami lakukan menunjukkan bahwa layanan Cloud Run memberikan manfaat dalam hal skalabilitas otomatis, pengelolaan infrastruktur yang mudah, dan keandalan tinggi dalam memproses pembaruan data secara efisien. Dengan demikian, hasil dari penelitian ini menegaskan bahwa pendekatan ini memberikan solusi yang efektif untuk meningkatkan proses pembaruan data di perusahaan XYZ, dengan memanfaatkan keunggulan teknologi Flask dan layanan Cloud Run.

Kata Kunci: *Flask Framework, Python, API, Google Cloud Platform (GCP), Google Cloud Run*

1. Pendahuluan

PT XYZ adalah salah satu perusahaan terbesar di Indonesia yang berfokus pada sektor ritel. Sebagai perusahaan ritel dengan jumlah transaksi harian yang sangat besar, PT XYZ harus memiliki beragam sistem dan aplikasi untuk mendukung transaksi serta memenuhi kebutuhan konsumen, mitra kerja, dan perusahaan itu sendiri[1]. Di samping itu perlu adanya sebuah layanan yang dapat membantu perusahaan untuk berhubungan dengan mitra kerja sebagai pihak penyuplai untuk dapat mengakses database stok yang dimiliki [2].

PT XYZ menghadapi permasalahan yang kompleks terkait manajemen stoknya, terutama terkait kurangnya keterhubungan antara sistem internal perusahaan dan aplikasi mitra kerja. Dengan transaksi harian yang besar, PT XYZ menghadapi kesulitan dalam menjaga akurasi dan ketersediaan stok produk. Kurangnya integrasi antar aplikasi

ISSN: 2720-992X
Copyright © KESATRIA: Jurnal Penerapan Sistem Informasi (Komputer & Manajemen)

menyebabkan keterlambatan dalam pembaruan informasi stok, yang pada gilirannya dapat berpotensi menyebabkan ketidakakuratan dan keterlambatan respons terhadap permintaan konsumen[3]. Oleh karena itu, penting bagi PT XYZ untuk meningkatkan keterhubungan antar sistem dan aplikasi dengan guna mengatasi permasalahan manajemen stok yang dihadapi[4].

Solusi untuk mengatasi masalah manajemen stok di PT XYZ adalah dengan menggunakan Flask API. Flask API merupakan alat yang mempermudah pembuatan layanan web untuk menghubungkan sistem PT XYZ dengan aplikasi mitra. Dengan menggunakan Flask API, data stok dapat diperbarui dengan cepat dan akurat antara kedua perusahaan. Penggunaan Cloud Run sebagai platform juga membantu menjalankan solusi ini secara efisien. Dengan demikian, PT XYZ dapat meningkatkan kepuasan pelanggan dan lebih baik dalam mengelola stok produknya di pasar ritel.

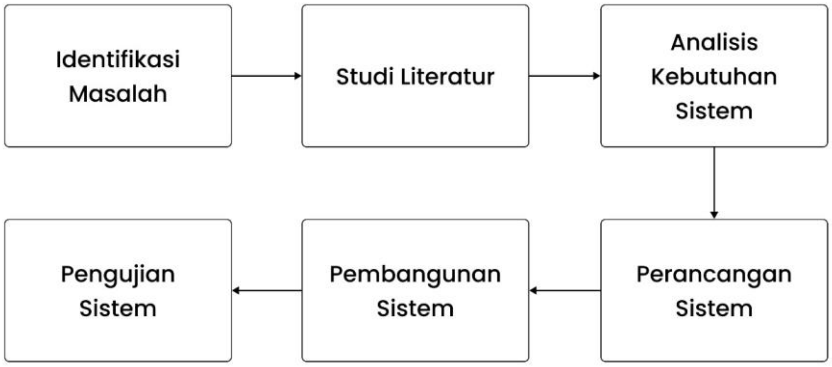
Flask API adalah sebuah kerangka kerja (framework) pengembangan aplikasi web yang menggunakan bahasa pemrograman Python. Flask API memungkinkan pengembang untuk membuat layanan web yang dapat diakses oleh aplikasi atau sistem lain melalui HTTP, memberikan kemampuan untuk berbagi data dan berinteraksi dengan aplikasi lain dengan mudah menggunakan protokol web standar[5].

Google Cloud Run adalah layanan runtime container tanpa keadaan yang menjalankan aplikasi dalam kontainer di platform Google Cloud. Proses deploy aplikasi sangat sederhana dengan menggunakan image Docker, dan layanan ini secara otomatis menyesuaikan skala aplikasi sesuai dengan kebutuhan sumber daya. Keunggulan utama Cloud Run adalah kemampuannya untuk secara otomatis menyesuaikan skala aplikasi sesuai dengan kebutuhan sumber daya yang lebih besar. Salah satu fitur paling inovatif dari Cloud Run adalah model tanpa server, yang membebaskan pengguna dari upaya pengelolaan infrastruktur, menyediakan solusi efisien dan mudah digunakan untuk menjalankan aplikasi di lingkungan kontainer tanpa perlu khawatir tentang infrastruktur dan skalabilitas[6].

2. Metodologi Penelitian

Tahapan penelitian dilakukan seperti yang disajikan pada gambar 1. Penelitian ini melibatkan sejumlah langkah yang mencakup identifikasi masalah, studi literatur, analisis kebutuhan sistem, perancangan sistem, pengembangan sistem, dan pengujian sistem.Tahap identifikasi masalah merupakan tahapan awal yang dilakukan untuk mengidentifikasi permasalahan yang akan dibahas dalam penelitian ini. Fokus penelitian ini adalah pada integrasi antara dua sistem atau aplikasi yang berbeda agar dapat saling berkomunikasi[7]. Tahap studi literatur melibatkan pengumpulan data dari berbagai sumber studi kepustakaan, seperti jurnal penelitian, artikel, dan situs internet yang relevan dengan topik penelitian. Data yang dikumpulkan meliputi implementasi, pengujian, dan informasi terkait teknologi Google Cloud Run, Python API, serta Python Flask. Tahap analisis kebutuhan dari sistem yaitu mempertimbangkan hal yang perlu dipersiapkan dan diperlukan dalam pembentukan sistem. Pada penelitian ini sistem memerlukan Python beserta dengan library dan framework-nya dalam pengembangan API serta Google Cloud sebagai penyedia layanan cloud computing. Tahap perancangan sistem dalam penelitian ini memanfaatkan Cloud Run service sebagai platform untuk menjalankan sistem yang mengelola pembaruan status pengiriman. Cloud Run Service dipilih karena kemampuannya dalam mengelola aplikasi dalam lingkungan kontainer yang dapat diubah ukurannya sesuai kebutuhan, serta otomatisasi manajemen sumber daya dan penskalaan. Sistem dijalankan dalam lingkungan Cloud Run service di Google Cloud Platform. Ini memungkinkan aplikasi untuk dijalankan dalam kontainer Docker yang dapat disesuaikan ukurannya sesuai kebutuhan, serta otomatisasi manajemen sumber daya dan penskalaan.

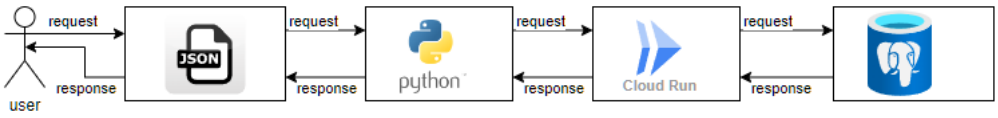
Tahap Pembangunan Sistem pada penelitian ini dilakukan dalam dua tahapan. Tahapan yang pertama adalah pembentukan Python API dengan framework flask. Tahapan yang kedua yaitu proses deploy ke Google Cloud Run. Pada tahapan awal dibuat fungsi fungsi python seperti fungsi hashing, validasi JSON, dan juga fungsi update lalu akan dijalankan dengan file main dari python. Setelah semua projek python selesai di lokal kemudian perlu dilakukan beberapa konfigurasi pada Google Cloud untuk persiapan deploy menjadi sebuah Cloud Run service seperti perlunya pepmbuatan Artifact Registry dan konfigurasi Cloud Triggers untuk membangun cloud run service nya. Tahap terakhir yaitu pengujian sistem. Pengujian sistem pada penelitian ini menggunakan aplikasi pihak ketiga yaitu Postman untuk memfasilitasi pelaksanaan permintaan metode HTPP yaitu method POST.



Gambar 1. Tahapan Penelitian

3. Hasil dan Pembahasan

Hasil dari penelitian ini adalah sebuah python API yang dijalankan melalui Google Cloud Run Service untuk proses *update* dalam kasus ini adalah memperbarui status pengiriman.



Gambar 2. Arsitektur Sistem

Diagram alur ini dimulai ketika pengguna mengirimkan permintaan (request) kepada sistem. Permintaan ini kemudian diterima dan diproses melalui beberapa tahapan proses. Tahapan pertama adalah interaksi dengan format data JSON Pada tahap ini, permintaan yang diterima di *parsing* dan diproses dalam format JSON untuk ekstraksi informasi yang relevan. Data yang telah diproses melalui format JSON dikirim ke lingkungan pemrograman Python, di tahap ini, proses logika bisnis dan manipulasi data dilakukan menggunakan bahasa pemrograman Python. Cloud Run digunakan untuk menjalankan dan menangani proses lebih lanjut dari aplikasi yang telah diproses dalam lingkungan Python. Akhirnya, respons atau hasil dari proses yang dilakukan dikirim kembali ke pengguna dari *database*. Hal ini menunjukkan bahwa sistem telah berhasil memproses permintaan pengguna dan mengembalikan respons yang sesuai berdasarkan data yang tersimpan dalam *database*. Database yang digunakan pada penelitian ini adalah PostgreSQL. PostgreSQL merupakan *Relational Database Management System* (RDBMS) yang bersifat *open source*. PostgreSQL menekankan kompabilitas, ekstensibilitas dan kreativitas. PostgreSQL memiliki kemampuan seperti *subqueries*,

transactions, *triggers* karena mendukung *Structured Query Languange* (SQL). PostgreSQL banyak digunakan pada aplikasi berbasis *website* dan *mobile* serta *analytics app* sehingga PostgreSQL sangatlah cocok digunakan pada aplikasi yang membutuhkan pengolahan data yang lebih kompleks[8].

Kode Program 1. Fungsi *Hashing*

```
1  sha256_data = sha256(value_sha256.encode("utf-8")).hexdigest()
2  value = sha256_data + token)
3  reqData = [param, {"key": value}]
4  body_message = {"datas": reqData}
5  print(json.dumps(body_message))
```

Kode program 1 adalah adalah kode yang digunakan untuk menghitung *hash* SHA-256 dari data yang akan diperbarui dan membuat pesan data JSON yang berisi *hash* data tersebut juga kunci yang akan digunakan untuk pengecekan integritas data nantinya[9]. Setelah fungsi *hashing* dibuat perlu dibuat fungsi untuk memvalidasi data yang telah di enkripsi dengan algoritma SHA-256. Kode dari fungsi validasi data dapat dilihat pada kode program 2.

Keunggulan dari kode tersebut terletak pada penggunaan SHA-256 hashing, yang merupakan salah satu algoritma hashing kriptografi yang sangat aman dan diakui secara luas. Dengan mengenkripsi nilai menggunakan algoritma ini sebelum menambahkan token, kode tersebut memastikan bahwa data yang dikirimkan aman dan tidak dapat dibaca oleh pihak yang tidak sah. Selain itu, penggunaan format JSON untuk membungkus data yang dikirimkan memudahkan dalam pertukaran data antara sistem-sistem yang berbeda. Dengan demikian, pendekatan yang digunakan dalam kode tersebut mencerminkan praktik keamanan yang baik dalam pertukaran data dan memastikan integritas serta keamanan informasi yang dikirimkan.

Kode Program 2. Fungsi Validasi

```
1  par1 = data[0]
2  par2 = data[1]
3  par1_withtoken = sha256_data + token
4  if par1_withtoken == par2["key"]:
5     Log.info(message=par1)
6     return par1
7  else:
8     par1 = {"status": False, "msg": "Param tidak diketahui"}
9     Log.error(message=par1)
10    return par1
```

Kode program 2 adalah implementasi dari fungsi validasi JSON yang mengoperasikan data dalam bentuk JSON sebagai argumen. Fungsi ini menawarkan keunggulan dalam validasi data dengan memeriksa nilai hash SHA-256 dari parameter data sesuai dengan *key* yang telah ditetapkan. Fungsi ini mengekstrak nilai dari JSON dan mengidentifikasi parameter pertama dan kedua. Dengan menggabungkan beberapa nilai dan menerapkannya pada hash SHA-256, fungsi memastikan integritas data dengan metode enkripsi yang kuat. Dengan menambahkan token pada hasil enkripsi, fungsi memeriksa apakah hasilnya sesuai dengan *key* yang diharapkan**.**

Perlu ditambahkan beberapa file yang berfungsi sebagai konfigurasi yang diperlukan dalam proses deploy Python API tersebut agar menjadi sebuah layanan Cloud Run. Dalam proyek ini, diperlukan konfigurasi untuk *deploy* secara otomatis. Karena program API yang dibangun membutuhkan Docker, maka perlu dibuat Dockerfile di dalam direktori program. Dockerfile ini digunakan untuk menentukan langkah-langkah yang diperlukan untuk membuat kontainer Docker yang akan menjalankan aplikasi. Dengan adanya Dockerfile, Cloud Run dapat mengambil kode dari repositori kode sumber, membangun gambar Docker, dan melakukan *deploy* gambar tersebut ke lingkungan Cloud Run secara otomatis setiap kali ada perubahan di repositori kode sumber. Hal ini memastikan bahwa

versi terbaru dari aplikasi selalu tersedia dan dijalankan dengan cepat dalam lingkungan yang terisolasi dan aman. Isi kode dari dockerfile seperti pada Kode Program 3.

Kode Program 3. Dockerfile

```
1  FROM python:3-slim-bullseye As builder
2  RUN python -m venv /opt/venv
3  COPY --from=builder /opt/venv /opt/venv
4  WORKDIR PYTHONPROJECT
5  ENV PATH="/opt/venv/bin:$PATH"
6  ENV CLOUD_APPS CLOUD_RUN
7  EXPOSE 8080
8  COPY . .
9  RUN /opt/venv/bin/pip install gunicorn
10 CMD . /opt/venv/bin/activate && exec gunicorn --bind 0.0.0.0:8080 --workers 1 --threads 8 --timeout 0
   main:app
```

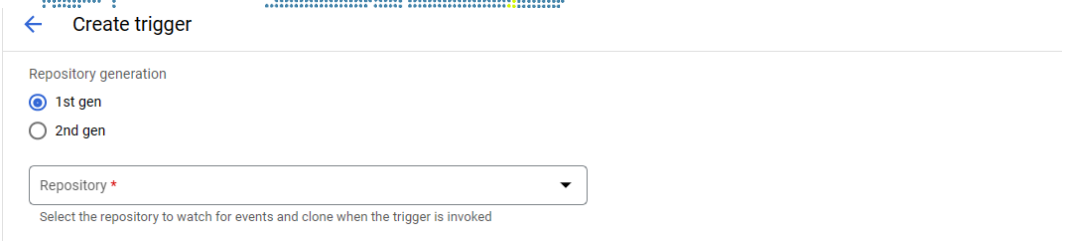
Perlu ditambahkan beberapa file yang berfungsi sebagai konfigurasi yang diperlukan dalam proses deploy Python API tersebut agar menjadi sebuah layanan Cloud Run. Dalam proyek ini, diperlukan konfigurasi untuk *deploy* secara otomatis. Karena program API yang dibangun membutuhkan Docker, maka perlu dibuat Dockerfile di dalam direktori program. Dockerfile ini digunakan untuk menentukan langkah-langkah yang diperlukan untuk membuat kontainer Docker yang akan menjalankan aplikasi. Dengan adanya Dockerfile, Cloud Run dapat mengambil kode dari repositori kode sumber, membangun gambar Docker, dan melakukan *deploy* gambar tersebut ke lingkungan Cloud Run secara otomatis setiap kali ada perubahan di repositori kode sumber. Hal ini memastikan bahwa versi terbaru dari aplikasi selalu tersedia dan dijalankan dengan cepat dalam lingkungan yang terisolasi dan aman. Isi kode dari dockerfile seperti pada Kode Program 3.

File cloudbuild.yaml juga perlu ditambahkan ke direktori *root* sebagai konfigurasi untuk membangun lalu mendorong *image* ke Artifact Registry untuk selanjutnya di *deploy* ke Cloud Run. File cloudbuild.yaml dimodelkan menggunakan sumber daya Build dari API Cloud Build. File konfigurasi ini berisi Langkah-langkah seperti build, push, dan deploy yang nantinya akan dijalankan oleh Cloud Build.

Kode Program 4. Cloudbuild.yaml

```
1  steps:
2  - name: gcr.io/cloud-builders/docker
3    args: ['build', '-t', 'gcr.io/PROJECT_ID/IMAGE', '.']
4  - name: 'gcr.io/cloud-builders/docker
5    args: ['push', 'gcr.io/PROJECT_ID/IMAGE']
6  - name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'
     entrypoint: gcloud
7    args: ['run', 'deploy', 'SERVICE-NAME', '--image', 'gcr.io/PROJECT_ID/IMAGE', '--region', 'REGION']
   images:
9  - gcr.io/PROJECT_ID/IMAGE
```

Dalam membuat Cloud Build melakukan pekerjaannya secara otomatis perlu dibuat Cloud Build Trigger yang merupakan pemicu agar memulai pembangunan setiap kali terdapat *push* baru ke repositori. Untuk itu ketika proses pembuatan *trigger* ini perlu beberapa penyetelan seperti definisi lokasi penyimpanan file konfigurasi (cloudbuild.yaml) yang akan dikerjakan oleh Cloud Build serta menentukan repositori mana yang akan dibangun secara otomatis.

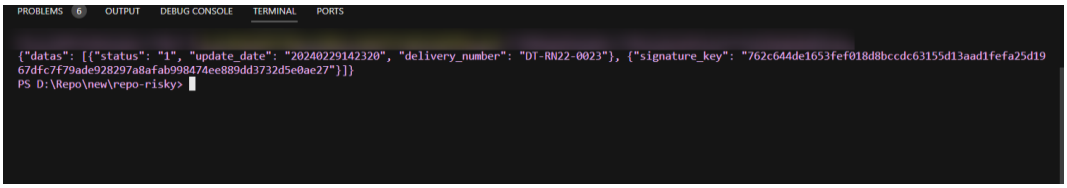


Gambar 3. Konfigurasi Repositori Cloud Trigger

Gambar 4. Inisialisasi File Konfigurasi Cloud Trigger

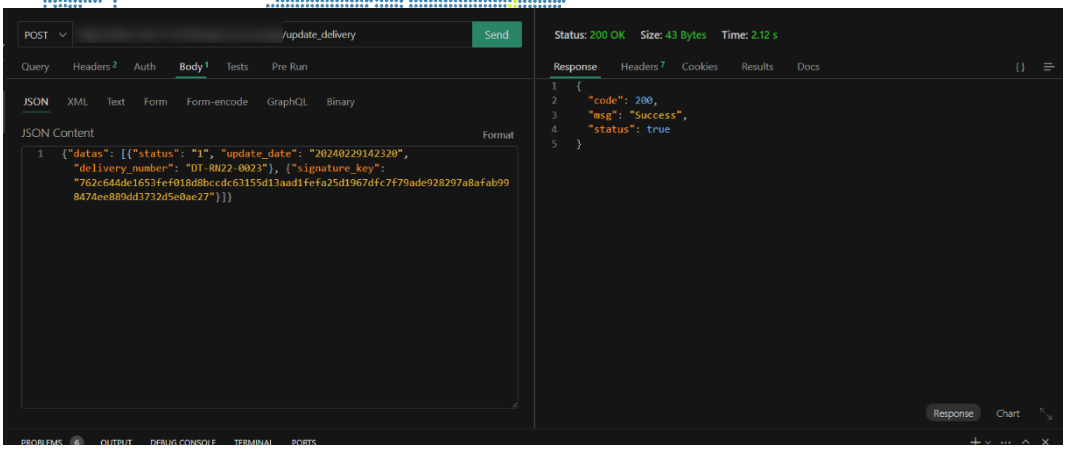
Menggunakan teknologi Cloud Build dan Cloud Trigger untuk *deploy* layanan Cloud Run memberikan sejumlah kelebihan yang signifikan. Pertama-tama, otomatisasi proses deploy menjadi lebih mudah dengan adanya Cloud Build, yang memungkinkan pengguna untuk mengotomatiskan langkah-langkah build dan deploy aplikasi ke Cloud Run. Dengan menetapkan aturan pada Cloud Trigger, seperti mendeteksi perubahan kode atau repositori, proses deploy dapat diaktifkan secara otomatis, mengurangi keterlibatan manual dalam proses tersebut. Selain itu, integrasi yang baik antara Cloud Build dan Cloud Trigger dengan layanan lain di Google Cloud Platform (GCP) mempermudah sinkronisasi aliran kerja pengembangan dan *deploy* secara keseluruhan.

Setelah proses *deploy* berhasil akan mendapatkan URL dari Cloud Run Service yang telah kita buat. URL tersebut yang dapat kita gunakan untuk melakukan *testing* API. Untuk tahapan *testing* yang pertama adalah melakukan enkripsi dengan file hashing tadi. Kita harus memasukan data yang ingin kita perbarui ke dalam fungsi tersebut.



Gambar 5. JSON hasil *hashing*

Pada gambar 5 diperlihatkan respon dari fungsi *hashing* yaitu berupa data JSON yang akan kita gunakan untuk melakukan *request method* HTTP ke layanan Cloud Run. Kemudian data JSON tersebut akan dikirim dalam body seperti pada gambar 6 dengan *method POST*. Data akan divalidasi terlebih dahulu. Apabila data telah tervalidasi dan berhasil maka data tersebut akan terbaharui dan *response* akan memberikan balikan pesan *success*.



Gambar 6. Pesan *Success*

Pengujian dilakukan dengan menggunakan metode *black box* untuk memverifikasi kinerja perangkat lunak dan memastikan bahwa semuanya berjalan dengan baik.

Tabel 1. Pengujian API

No	Uraian	Hasil yang diharapkan	Status
1.	Enkripsi data yang akan di *update*	Data akan berubah dalam format JSON	Valid
2.	Melakukan validasi data yang di *update*	Sistem akan melakukan validasi data sesuai dengan *key* dari JSON	Valid
3.	Melakukan *update* sesuai dengan hasil validasi	Sistem akan melakukan update ketika hasil validasi = *true*	Valid
4.	Memberikan hasil kembalian json	Ketika berhasil melakukan *update* maka *status = true*	Valid
5.	Data di *insert* ke tabel yang sesuai	Ketika *update* berhasil data tabel akan otomatis membuat *row* baru untuk menambahkan data yang telah ter *update*	Valid

Berdasarkan hasil pengujian menggunakan metode *black box*, dapat disimpulkan bahwa sistem telah berhasil melewati semua tahapan pengujian yang telah ditetapkan dengan status valid[10]. Enkripsi data yang akan diperbarui telah dilakukan dengan benar, dan data telah diubah ke dalam format JSON secara tepat. Validasi data yang diperbarui juga telah dilakukan dengan benar sesuai dengan kunci (*key*) yang ada dalam JSON, dan sistem hanya melakukan pembaruan ketika hasil validasi adalah benar (*true*). Selain itu, hasil kembalian JSON menunjukkan bahwa sistem telah berhasil melakukan pembaruan dengan status true. Data yang diperbarui juga telah dimasukkan ke dalam tabel yang sesuai, di mana sistem secara otomatis membuat baris baru untuk menambahkan data yang telah diperbarui. Dengan demikian seluruh percobaan telah mencapai tingkat keberhasilan sebesar 100% sehingga dapat dinyatakan perangkat lunak sudah berjalan dengan baik serta sesuai dengan kebutuhan awal.

4. Kesimpulan

Penelitian ini mengimplementasikan Python API dengan menggunakan *framework* Flask melalui layanan Cloud Run di Google Cloud Platform. Tahapan penelitian mencakup identifikasi masalah, studi literatur, analisis kebutuhan sistem, perancangan sistem, pengembangan sistem, dan pengujian sistem. Fokus penelitian adalah integrasi antara dua sistem atau aplikasi agar dapat saling berkomunikasi.

Pembangunan sistem dilakukan dalam dua tahap, yaitu pembentukan Python API dengan Flask dan proses *deploy* ke Google Cloud Run. Pengujian sistem menggunakan aplikasi pihak ketiga, yaitu Postman, untuk memfasilitasi permintaan HTTP, khususnya *method* POST. Penelitian ini memberikan kontribusi dalam implementasi dan pengujian sistem yang efisien dan andal untuk komunikasi antara aplikasi mitra dengan aplikasi yang dimiliki oleh PT. XYZ.

Masih terdapat kekurangan dalam penelitian yang dilakukan karena terdapat beberapa fitur yang masih dikembangkan. Sebagai saran untuk meningkatkan proyek ini adalah dengan menambahkan mekanisme pub/sub, yang akan memungkinkan komunikasi asinkron antara aplikasi mitra dan aplikasi PT. XYZ, meningkatkan skalabilitas dan fleksibilitas sistem, memfasilitasi manajemen pesan yang lebih efisien, dan tetap menjaga keamanan data. Dengan menerapkan pub/sub, sistem akan lebih siap mengatasi lonjakan beban dan beradaptasi dengan perubahan kebutuhan dalam pengembangan masa depan.

Daftar Pustaka

[1] J. Alexander and N. Husufa, "Implementasi point of sales berbasis web pada usaha Olive Cafe," *Juli*, vol. 2, pp. 1–14, 2020, [Online]. Available: https://jurnal.ikhafi.or.id/index.php/jusibi/452

[2] S. Sardjono, "Perancangan Sistem Inventori Stock PT. XYZ Dengan Metoda Array Accessing," *SisInfo–Jurnal Sist. Inf. dan Inform.*, vol. 1, no. 02, 2019, [Online]. Available: http://repository.unibi.ac.id/187/%0Ahttp://repository.unibi.ac.id/187/1/PERANCANGAN SISTEM INVENTORI STOCK PT. XYZ DENGAN METODA ARRAY ACCESSING.pdf

[3] M. Riza Baihaqie, P. Komputer Muda, P. Kabupaten Pekalongan, and J. Tengah, "Analysis and Design for Non-Cash Payments of Regional Taxes and Levies using Payment Gateway Services," *Semnasif (Seminar Nas. Inform.*, pp. 101–122, 2021.

[4] T. Y. B. Cici Wiarsih, "Universitas Muhammadiyah Purwokerto 124," vol. 8, no. 0281, pp. 124–134, 2013.

[5] M. D. Anwar and I. A. Kautsar, "Arsitektur Perangkat Lunak Berbasis Layanan Mikro pada Sistem Manajemen Informasi Kantin," *Phys. Sci. Life Sci. and Engineering*, vol. 1, no. 2, p. 13, 2024, doi: 10.47134/pslse.v1i2.196.

[6] S. Ifrah, *Getting Started with Containers in Google Cloud Platform*. Melbourne: VIC, 2021. doi: https://doi.org/10.1007/978-1-4842-6470-6.

[7] U. Integrasi and D. Dan, "Abstrak: Industri (usaha) di bidang kuliner di Indonesia merupakan usaha yang mengalami pertumbuhan yang sangat pesat. Peningkatan kualitas pelayanan pada suatu usaha akan berimplikasi dengan maksimalnya profit yang di dapatkan. Teknologi yang tepat untuk," vol. 4, no. 1, pp. 36–41, 2022.

[8] A. D. Praba and M. Safitri, "Studi Perbandingan Performansi Antara Mysql Dan Postgresql," *J. Khatulistiwa Inform.*, vol. 8, no. 2, pp. 88–93, 2020, doi: 10.31294/jki.v8i2.8851.

[9] F. P. Utama, G. Wijaya, R. Faurina, and A. Vatresia, "Implementasi Algoritma AES 256 CBC, BASE 64, Dan SHA 256 dalam Pengamanan dan Validasi Data Ujian Online," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 10, no. 5, pp. 945–954, 2023, doi: 10.25126/jtiik.20231056558.

[10] A. Fahrezi, F. N. Salam, G. M. Ibrahim, R. R. Syaiful, and A. Saifudin, "Pengujian Black Box Testing pada Aplikasi Inventori Barang Berbasis Web di PT. AINO Indonesia," *J. Ilmu Komput. dan Pendidik.*, vol. 1, no. 1, pp. 1–5, 2022, [Online]. Available: https://journal.mediapublikasi.id/index.php/logic