**Jurnal Teknik Informatika (JUTIF)**
Vol. 3, No. 5, Oktober 2022, hlm. 1293-1300

DOI: https://doi.org/10.20884/1.jutif.2022.3.5.351
p-ISSN: 2723-3863
e-ISSN: 2723-3871

# *DEVELOPMENT OF PRICE DISTRIBUTION MODULE ON MERCHANDISE APPLICATION USING FLASK FRAMEWORK AT PT XYZ*

**Josua Ade Saputra[*1], Yerymia Alfa Susetyo, S.Kom., M.Cs.[2]**

[1] Universitas Kristen Satya Wacana; Jl. Dr. O. Notohamidjojo No.1, Salatiga 50715, telp/fax of (0298) 321212
[1,2] Program Studi Teknik Informatika, FTI UKSW, Salatiga
Email: [1]672018272@student.uksw.edu, [2] yeremia.alfa@uksw.edu

(Naskah masuk: 31 Mei 2022, Revisi: 8 Juni 2022, diterbitkan: 20 Oktober 2022)

***Abstract***

*PT XYZ is a company in Indonesia which is engaged in retail which has more than 14,300 stores and 32 branches or warehouses. The problem arises when the branch has determined the price of an item and the price of the item has to be distributed to each store. The price of an item will always change due to promos and discounts that have their own period, so it is necessary to spread prices to stores in real-time. In addition, with millions of transactions every day results in very fast data growth and requires large storage. The application module is used to quickly and precisely distribute predefined prices to each store. The development of this system uses the Flask Framework with the Python programming language. Flask facilitates system development because of its simple use. Developers can determine their own library to be used, so that the system is made lighter. Flask has the flask-sqlalchemy bind package which can easily distribute data to appropriate databases. The result of this research is a price distribution application module which is then implemented using the Python programming language and the Flask Framework.*

**Keywords**: *Flask, Database Distribution, Price Distribution, Python*

# PEMBANGUNAN MODUL DISTRIBUSI HARGA PADA APLIKASI MERCHANDISE MENGGUNAKAN *FRAMEWORK FLASK* DI PT XYZ

**Abstrak**

PT XYZ adalah perusahaan di Indonesia yang bergerak dalam bidang retail yang memiliki lebih dari 14.300 toko serta 32 cabang atau *warehouse*. Masalah muncul ketika cabang telah menentukan harga dari suatu barang dan harga barang harus disebarkan ke setiap toko. Harga dari suatu barang akan selalu berubah karena adanya promo dan diskon yang memiliki masa periode nya sendiri, sehingga diperlukan penyebaran harga ke toko secara *real-time*. Selain itu, dengan jutaan transaksi tiap harinya mengakibatkan pertumbuhan data yang sangat cepat dan memerlukan penyimpanan yang besar. Modul aplikasi digunakan untuk mendistribusikan harga yang telah ditetapkan ke setiap toko dengan cepat dan tepat. Pengembangan sistem ini menggunakan *Framework Flask* dengan bahasa pemrograman *Python*. *Flask* memudahkan dalam pengembangan sistem karena penggunaannya yang sederhana. Pengembang bisa menentukan sendiri *library* yang akan akan digunakan, sehingga sistem yang dibuat menjadi lebih ringan. *Flask* memiliki *package flask-sqlalchemy bind* yang dapat dengan mudah melakukan distribusi data ke *database* yang sesuai. Hasil dari penelitian ini adalah sebuah modul aplikasi distribusi harga yang kemudian diimplementasikan menggunakan bahasa pemrograman *Python* dan *Framework Flask*.

**Kata kunci**: *Flask, Distribusi Database, Distribusi Harga, Python*

## 1. PENDAHULUAN

PT XYZ adalah perusahaan Indonesia yang bergerak dalam bidang retail yang memiliki lebih dari 14.300 toko dan juga 32 cabang atau *warehouse*. Sebagai perusahaan retail besar yang dapat menghasilkan ribuan transaksi per harinya, PT XYZ memerlukan beragam sistem serta aplikasi yang dapat mendukung dan meringankan pemenuhan kebutuhan dari PT XYZ, kolega, maupun konsumen.

Salah satu aspek penting dari perusahaan retail adalah harga, di mana harga berpengaruh positif dan signifikan terhadap keputusan konsumen untuk

membeli suatu barang atau jasa [1]. Harga pokok penjualan merupakan aspek penting dalam dunia usaha, karena perusahaan menjadikan harga pokok penjualan sebagai dasar dalam pembuatan keputusan jual beli [2].

Setiap cabang di PT XYZ bertanggungjawab atas ratusan hingga ribuan toko dan cabang bertugas untuk mendistribusikan barang, melakukan pengawasan stok, dan juga menentukan harga jual barang di setiap toko. Masalah muncul ketika cabang telah menentukan harga dari suatu barang dan harga barang harus disebarkan ke setiap toko. Harga dari suatu barang akan selalu berubah karena adanya promo dan diskon yang memiliki masa periode nya sendiri, sehingga diperlukan penyebaran harga ke toko secara *real-time*.

PT XYZ menghasilkan jutaan transaksi per harinya sehingga terjadi pertumbuhan data yang sangat cepat dan memerlukan penyimpanan yang besar. *Database* terdistribusi merupakan salah satu jawaban dari permasalahan tersebut. Data yang ada akan dibagikan ke masing-masing cabang dengan wilayah geografis yang berbeda.

Solusi yang dapat dilakukan untuk mengatasi masalah yang ada adalah dengan membangun sistem atau aplikasi *web* untuk melakukan distribusi harga yang bertujuan untuk menyampaikan harga yang telah diatur ke toko. Pembangunan aplikasi *web* dibangun dengan bahasa pemrograman Python, serta menggunakan *framework Flask*.

Pemilihan *python* sebagai bahasa pemrograman dalam membangun aplikasi ini dikarenakan *python* merupakan bahasa pemrograman yang memiliki *library* yang besar dan lengkap [3]. Selain itu *python* juga memiiki sifat *multi-paradigma* yang membebaskan pengembang untuk memilih aplikasi akan bersifat *procedural* atau objek ataupun keduanya [4].

Dalam pembangunan aplikasi pengembang dapat dengan mudah mengatur *behaviour* dari suatu *web* serta membuat *web* menjadi lebih terstuktur dengan menggunakan *Flask*. *Flask* merupakan sebuah *microframework* karena fungsi, komponen, *library*, atau ekstensi yang tidak terpasang secara *default*, sehingga pengembang dapat menentukan sendiri dan hal tersebut akan membuat aplikasi yang dikembangkan menjadi lebih ringan [5]. Selain itu, *Flask* memiliki performa yang lebih ringan dan cepat karena inti *framework Flask* yang tergolong sederhana. Bahasa maupun arsitektur *Flask* juga relatif sederhana sehingga menjadikan *Flask* lebih mudah untuk dipahami [6]. Dalam *flask* juga terdapat package yaitu *flask-sqlalchemy bind* yang dapat dengan mudah mendistribusikan data ke *database* yang sesuai [7].

Pembangunan modul aplikasi ini menggunakan *PostgresSQL* sebagai basis datanya. Pemiihan *PostgresSQL* sebagai basis data karena *PostgresSQL* bersifat *open source* yang memudahkan pengembang menggunakannya sesuai dengan kebutuhan sistem yang akan dibangun [8]. Eksekusi *PostgresSQL* sendiri dapat dengan menggunakan beberapa *tool* seperti *pgAdmin* atau *DBeaver* [9]. Pada pembangunan aplikasi ini *tool* yang digunakan adalah *Dbeaver.*

Penelitian mengenai pembangunan aplikasi menggunakan *framework Flask* dengan arsitektur *REST* dan *ORM* pernah dilakukan oleh Brian Pratama Putra yang berjudul Implementasi *API Master Store* menggunakan *Flask*, *REST* Dan *ORM* di PT XYZ pada 2020. Penelitian tersebut bertujuan untuk membantu *programmer* dan pengembang untuk melakukan migrasi aplikasi-aplikasi yang ada di PT XYZ tanpa perlu melakukan duplikasi *database master store*. Pembangunan *API Master Store* dilakukan menggunakan metode *REST* dan *framework Flask*. Hasil dari penelitian ini adalah *API* dinamis yang dapat diatur data yang dibutuhkan untuk membangun aplikasi sehingga memudahkan para pengembang dan *programmer* dalam membuat aplikasi baru maupun melakukan migrasi aplikasi. Dari penelitian ini dapat menunjukkan bahwa *framework Flask* dapat menyelesaikan masalah pada peneliti, sehingga hal ini menjadi acuan pada penelitian ini untuk menggunakan *framework Flask* [10].

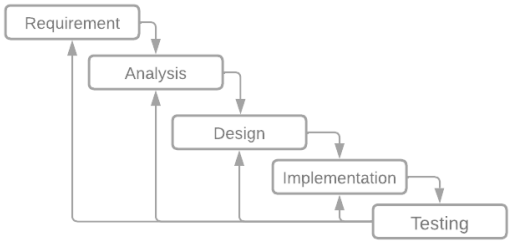
Penelitian membangun aplikasi berbasis *web* berjudul "Implementasi *Flask Framework* pada Pembangunan Aplikasi *Purchasing Approval Request*" oleh Dinda Fitri Ningtyas pada 2021. Dalam penelitian tersebut, *framework Flask* digunakan untuk tujuan mempermudah *user* dalam melakukan *purchasing* dan *approval request*, serta pemantauan posisi serta alur proses *request*. Pembangunan aplikasi dilakukan menggunakan metode *Rapid Application Development (RAD)* dan *framework Flask*. Penelitian ini menghasilkan sebuah aplikasi "*Purchasing Approval Request*" atau PAR yang dapat mengakomodasi kebutuhan perusahaan akan proses *purchasing* dan *approval request* serta meningkatkan efisiensi waktu dalam melakukan proses *purchasing* dan *approval request*. Melalui penelitian ini, peneliti berhasil mengimplementasikan *framework Flask* untuk menyelesaikan masalah yang diangkat oleh peneliti yaitu pembangunan aplikasi PAR, di mana hal tersebut memperkuat alasan penggunaan *framework Flask* pada penelitian ini [11].

Penggunaan *database* terdistribusi pernah dilakukan oleh Arif Setyo Nugroho tahun 2020 pada penelitiannya yang berjudul "Penerapan Basis Data Terdistribusi Untuk Transaksi Dan Kontrol Penjualan Antar Cabang". Penelitian tersebut dapat membuktikan bahwa Basis Data Terdistribusi mampu membangun sistem yang membantu pengolahan data perusahaan tersebut. Hasil dari penelitian ini menjadi acuan untuk menggunakan sistem *database* terdistribusi dalam pembangunan aplikasi [12].

Berdasarkan hal di atas, modul distribusi harga dibangun menggunakan *Framework Flask*, sehingga harga yang telah diatur oleh cabang dapat terdistribusi ke setiap toko dengan cepat dan tepat.

## 2. METODE PENELITIAN

Modul distribusi harga merupakan modul dari aplikasi penting perusahaan, sehingga metode penelitian yang paling tepat untuk digunakan menyesuaikan dengan kebutuhan perusahaan. Metode penelitian dapat dilihat pada Gambar 1.




Gambar 1. *Metode Penelitian*

Pada Gambar 1 menggambarkan setiap tahapan yang dilakukan dalam penelitian. Tahapan awal adalah *requirement* atau identifikasi masalah. Tahap ini berfungsi untuk memahami permasalahan yang ada dengan baik agar dapat menemukan sumber permasalahan yang akan diselesaikan. Dengan harga produk yang sering mengalami perubahan, PT XYZ membutuhkan sebuah sistem yang dapat melakukan distribusi harga yang telah ditetapkan dengan cepat ke toko-toko yang ada. Pertumbuhan data yang cepat dikarenakan banyaknya jumlah transaksi per hari juga membuat PT XYZ harus melakukan distribusi *database* agar data yang dihasilkan dapat diolah dan disimpan dengan baik.

Setelah sumber permasalahan teridentifikasi, langkah selanjutnya yaitu tahapan analisis. Tahap analisis bertujuan untuk menemukan solusi terbaik dari masalah yang ada melalui penelitian maupun eksperimen dan membandingkan dengan penelitian atau eksperimen serupa yang telah ada. Dalam pembangunan modul aplikasi, *Flask* dipilih karena dapat mengembangkan aplikasi *web* dengan cepat serta sederhana dan memiliki *library* yang mampu menjawab permasalahan. *Database* yang akan digunakan adalah *PostgreSQL* dengan *tools DBeaver*.

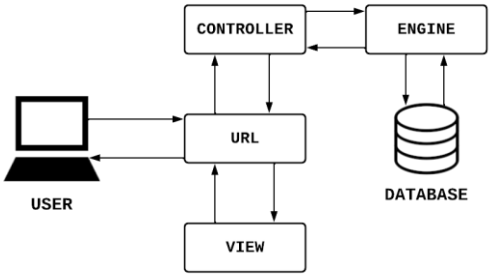
Tahapan selanjutnya adalah *design* di mana tahap ini bertujuan untuk menentukan perangkat keras yang *compatible*, merencanakan arsitektur sistem, *Use Case*, dan *Activity Diagram*. Tahap implementasi merupakan proses pembangunan modul distribusi harga meliputi pengkodean program atau *script* baik *Front-End* maupun *Back-End*. Proses implemetasi menggunakan *IDE Visual Studio Code* yang mendukung banyak bahasa pemrograman serta terdapat bermacam-macam ekstensi yang dapat membantu programmer [13]. Tahap terakhir adalah melakukan *testing* untuk memastikan sistem berjalan dengan baik. Metode yang digunakan untuk testing adalah metode *Black Box*. *Black Box* digunakan untuk menguji hasil dari proses *input* dan *output* sistem sesuai dengan harapan atau tidak [14].

## 3. HASIL DAN PEMBAHASAN

Bab ini dibagi ke dalam 3 sub bab yaitu perancangan sistem yang berisi penjelasan mengenai Arsitektur Sistem, *Use Case* dan *Activity Diagram* dari modul yang akan dibuat, kemudian implementasi, pembuatan *code* program menggunakan *framework Flask* dan *library flask-sqlalchemy*. Setelah tahap implementasi telah selesai, langkah selanjutnya yaitu melakukan *testing* dari modul yang telah dibuat menggunakan *Black Box*.

### 3.1. Perancangan Sistem

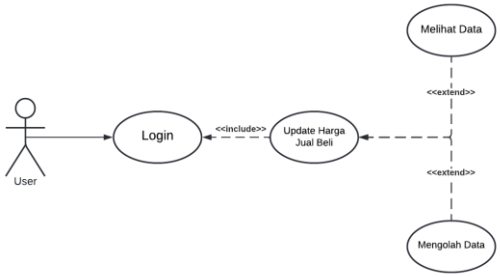
Perancangan sistem diawali dengan pembuatan arsitektur sistem. Arsitektur sistem berfungsi untuk menjelaskan alur data dari sistem. Arsitektur sistem dapat dilihat pada Gambar 2.




Gambar 2. Arsitektur Sistem

Gambar 2 menjelaskan bahwa *user* harus mengakses *URL* dimana *URL* tersebut berfungsi sebagai penghubung *view* dan *controller*. *Controller* memiliki 2 tugas, yaitu menciptakan *engine* yang bertugas untuk melakukan *query* ke *database* dan mengelola data. Hasil dari *query* adalah *response* berupa *JSON* yang nantinya akan dikelola sesuai dengan kebutuhan.

*Use case diagram* menjelaskan tentang apa saja yang bisa aktor lakukan terhadap sistem [15]. Adapun *Use case diagram* dari modul distribusi harga dapat dilihat pada gambar berikut.

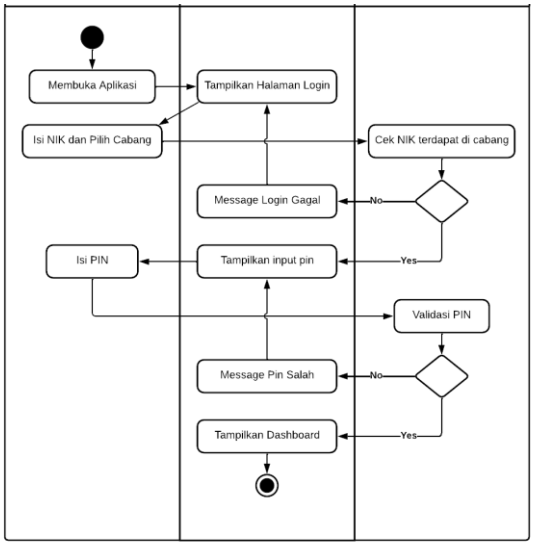



Gambar 3. *Use case diagram*

Gambar 3 merupakan *Use case diagram* yang digunakan dalam peneltian ini. Aktor yang disebut *User* merupakan karyawan yang akan menjalankan aplikasi ini. *User* dapat menjalankan fungsi yaitu *Update* Harga Jual Beli, tetapi sebelum itu, *user* diharuskan untuk melakukan *login* terlebih dahulu.

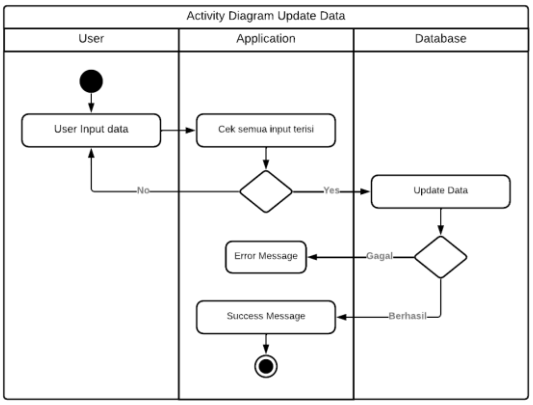
Jika *Use case diagram* menjelaskan aplikasi dengan sudut pandang aktor, *Activity diagram* menjelaskan aplikasi dengan sudut pandang sistem. *Activity diagram* menggambarkan proses dari suatu sistem berjalan. *Activity diagram* dapat dilihat pada gambar berikut.

Gambar 5. *Activity Diagram Update*




Gambar 4. *Activity Diagram Login*

Gambar 4 menjelaskan mengenai perancangan aktivitas login dimana *user* akan menginputkan NIK dan juga memilih asal cabang. Lalu *database* akan melakukan pengecekan apakah cabang yang dipilih memiliki NIK dari *user*. Apabila NIK dari *user* terdapat dalam *database* cabang tersebut, maka *input text* untuk memasukkan PIN akan muncul dan *user* memasukkan PIN nya. Apabila PIN benar, maka *user* akan dibawa ke tampilan utama yaitu *dashboard*. Setelah melakukan *login*, *user* juga dapat melakukan *update* data, *activity diagram update data* dapat dilihat pada Gambar 5.




Gambar 5 menjelaskan tentang cara *update* data, dimana *user* akan melakukan *input* data, lalu sistem akan melakukan validasi, dan data yang ada di *database* akan diperbarui.

### 3.2. Implementasi

Tahap awal implementasi *Flask* dalam pembuatan modul yaitu melakukan inisiasi dari *Flask* yang bisa dilihat pada Kode Program 1.

Kode Program 1. Wsgi.py

```
1  from application import create_app
2  app = create_app()
3  if __name__ == "__main__":
4      app.run(host='0.0.0.0',
5  port=5000, debug=True)
```

Di dalam *file* wsgi.py berisi mengenai pembuatan *instance* aplikasi *Flask* dengan nama *app* yang berisi kelas dari *create_app* yang akan dibahas pada Kode Program 2. Di baris terakhir, merupakan konfigurasi dari aplikasi *Flask* dijalankan di *localhost* dengan *port* 5000.

Kode Program 2. Config.py

```
1  from os import environ
2  from datetime import timedelta
3  # ini adalah kelas config untuk
4  setting atau connect database
5  class Config:
6      SECRET_KEY = 'inisecretkey'  #
7  environ.get('SECRET_KEY')
8      FLASK_APP =
9  environ.get('FLASK_APP')
10     FLASK_ENV =
11 environ.get('FLASK_ENV')
12     PERMANENT_SESSION_LIFETIME =
13 timedelta(minutes=60)
14     SQLALCHEMY_DATABASE_URI =
15 'postgresql+psycopg2://***:****@10
16 .4.30.134:5432/MCSC'
17     SQLALCHEMY_BINDS = {
18       'BZ01':
19 'postgresql+psycopg2//***:****@10.
20 4.30.134:5432/SATBDG',
21       'KZ01':
   'postgresql+psycopg2:
   //***:****@10.4.30.134:5432/MCS',
     }
```

Kode program 2 berisi tentang konfigurasi *database*. Baris 12 merupakan konfigurasi berapa lama *session* berlangsung. Lalu di baris 14 merupakan *database* yang pertama kali akan digunakan, dalam kasus ini, database tersebut berisi tentang data kode dan nama cabang. *SQLAlchemy* menggunakan *url* untuk melakukan koneksi. Format dari *url* tersebut yaitu:

‘**dialect+driver://username:password@host:port/database’**

Pada baris selanjutnya terdapat kode *SQLALCHEMY_BINDS*, *bind* dalam *flask* adalah sesuatu yang dapat mengeksekusi *query* yang biasanya berupa koneksi atau mesin yang terdiri dari *key* dan *url*. Hal ini berfungsi untuk menciptakan mesin atau koneksi dengan lebih mudah dan cepat. Setelah melakukan konfigurasi *database*, langkah selanjutnya yaitu melakukan konfigurasi *Flask* yang dapat dilihat pada Kode Program 3.

Kode Program 3. __init__.py

```
1  from flask import Flask
2  from flask_login import LoginManager
3  login_manager = LoginManager()
4
5  def create_app():
6      app = Flask(__name__)
7
8  app.config.from_object('config.Confi
9  g')
10     login_manager.init_app(app)
11     login_manager.login_view =
12 "signin"
13     login_manager.anonymous_user =
14 Anonymous
15     with app.app_context():
16       from .controller import
17 LoginController,HargaBeliJualControl
18 ler
19     return app
```

Kode Program 3 merupakan konfigurasi inti dari aplikasi *Flask* yang dibuat, berisi tentang menyatukan semua konfigurasi *library* ke dalam suatu *object* yaitu *app*. Konfigurasi tersebut adalah *flask-sqlalchemy* dan juga *flask-login*.

*LoginManager* yang ada di baris 3 berisi kode yang memungkinkan untuk *flask-login* dan aplikasi kita bekerja sama. Pada baris 8 merupakan pembuatan *object* dari *Flask* yang selanjutnya disatukan dengan *config* yang telah kita buat pada Kode Program 2. Setelah *object* dari aplikasi telah dibuat, konfigurasi *login* dapat dilakukan dengan cara yang ditunjukkan oleh baris 10. Setelah itu, menampilkan halaman *login* dengan cara menembak *url ‘sigin’* yang ada *di LoginController.py*. Bagian terakhir merupakan proses untuk mendaftarkan *Controller* agar dapat diakses oleh *Flask*. Untuk mengakses Controller yang telah didaftarkan dapat dilihat pada Kode Program 4.

Kode Program 1. LoginController.py

```
1  @app.route('/signin',
2  methods=['GET', 'POST'])
3  def signin():
4    if current_user.is_authenticated:
5        return
6  redirect(url_for('index'))
7    else:
8        if request.method == 'POST':
9            d = request.form.to_dict()
10           absen = d["username"]
11           password = d["password"]
12           kdStore = d["kd_store"]
13           status =
14 loginDao.getLogin(absen, password,
15 kd_store)
16           if status == True:
17               data =
18 loginDao.getUserByAbsen(absen,
19 kdStore)
20               login_user(data)
21           return
22 render_template('index.html')
23     dataBranch =
24 loginDao.getAllBranch()
25     return
26 render_template('signin/index.html',
27 branch=dataBranch)
28
29
```

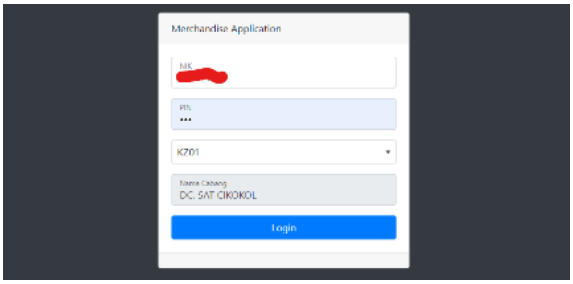
Kode Program 4 merupakan proses *user* melakukan *login*. Langkah awal yang dilakukan adalah mengecek apakah *session* dari *user* masih aktif atau tidak, jika *session* sudah selesai, maka akan dikembalikan ke halaman *login*. Lalu saat *user* telah memasukkan nik, *password*, dan cabang maka akan dilakukan proses validasi pada baris 14 yang akan dijabarkan pada Kode Program 5. Apabila berhasil, maka akan membentuk *session* yang berisi *object* dari *user* (baris 20) dan sistem akan menampilkan halaman *dashboard*.

Kode Program 2. LoginDao.py

```
1  Session = sessionmaker()
2  def getLogin(nik, pin,kd_store):
3      result{"status": False, "msg":
4  "Nik atau password salah."}
5      session =
6  Session(bind=db.get_engine(app,
7  kd_store))
8      try:
9       a = session.execute
10             ("select
11                app_user_security.g
12                et_valid_user(:nik,
13                :pin) as LOG_OK
14                from dual", {"nik":
15                nik, 'pin': pin}
16             ).fetchone()
17         if a[0] == 'T':
18             result = {"status":
19 True, "msg": "Login Sukses"}
20         elif a[0] == 'B':
21             result = {"status":
22 False, "msg": "Pin Blocked"}
23         elif a[0] == 'E':
24             result = {"status":
25 False, "msg": "Pin Expired"}
26         else:
27             result = {"status":
28 False, "msg": "Nik password
```

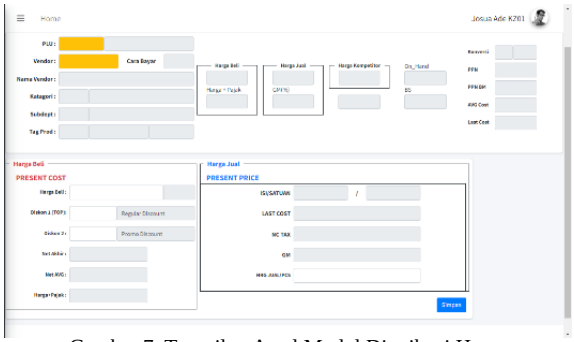
```
29      salah."}
30          except Exception as e:
31              print("Erro " + str(e))
32          session.close()
33          return result
34
35
```

Kode Program 5 merupakan penjabaran dari validasi *login*. Baris 6 menunjukkan cara untuk membuat mesin eksekusi *query* di mana *kd_store* berperan sebagai *bind_key* untuk koneksi *url database*. Fungsi dari mesin adalah mengeksekusi *query* ke *database* yang dituju. Hasil dari modul aplikasi yang sudah dibuat menggunakan *Flask* dapat dilihat pada Gambar di bawah.



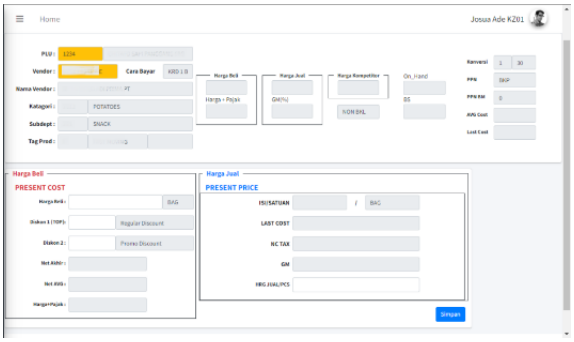

Gambar 6. Halaman Login

Pada Gambar 6 menampilkan halaman pertama yang ditampilkan saat menjalankan aplikasi. *User* akan mengisi *field* NIK, PIN, dan juga memilih cabang. Untuk *field* nama cabang, akan otomatis terisi setelah *user* memilih cabang, setelah itu *user* menekan tombol *Login* untuk masuk. Tampilan awal setelah user berhasil melakukan login dapat dilihat pada Gambar 7.



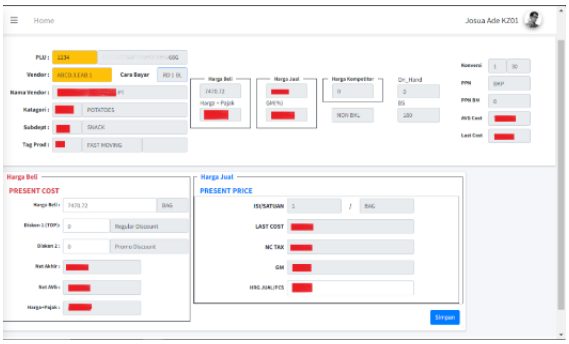

Gambar 7. Tampilan Awal Modul Distribusi Harga

Halaman distribusi harga berfungsi untuk mengatur harga dari suatu barang atau produk. Data dari modul distribusi ini akan dipakai oleh toko yang berada di bawah cabang. Langkah awal untuk mengatur harga dari suatu barang atau produk adalah dengan mengisi *field* PLU. Tampilan saat *user* mengisi *field* PLU dan menekan *enter* ada pada Gambar 8.




Gambar 8. Tampilan setelah enter PLU

Gambar 8 menggambarkan kondisi saat *user* mengisi PLU dan menekan *enter*. Sistem akan melakukan *query* untuk membaca data dan setelah sistem berhasil membaca data, user akan mengisi *field* vendor. Gambar 9 memperlihatkan tampilan setelah *user* mengisi *field vendor*.




Gambar 9. Tampilan Setelah Enter Vendor

Gambar 9 merupakan tampilan setelah *user* mengisi *field vendor*. Hampir sama seperti saat *enter* PLU, setelah mengisi *field vendor* dan menekan *enter*, sistem akan mengeksekusi *query* untuk membaca data.

Lalu untuk *update*, *user* hanya bisa mengganti harga beli, diskon 1, diskon 2, dan juga harga jual/*pcs*. Setelah menekan tombol simpan, sistem akan secara otomatis melakukan perhitungan berdasarkan data yang telah diinput oleh *user*.

### 3.3. Pengujian

Pengujian sistem pada penelitian ini menggunakan metode *Black Box*, berikut adalah hasil dari pengujian yang telah dilakukan

Table 1. Table Black Box Testing

| No | Skenario Pengujian | Hasil yang Diharapkan | Hasil Pengujian | Status Pengujian |
|---|---|---|---|---|
| 1 | Mencoba melakukan login dengan NIK yang sama, tetapi dengan cabang yang berbeda. (data nama di tiap cabang berbeda) | Nama yang dihasilkan di halaman *dashboard* nantinya akan berbeda, walaupun menggunakan NIK yang sama. | Sesuai yang diharapkan | *Valid* |
| 2 | Memasukkan NIK atau PIN secara asal dan tidak sesuai dengan data yang ada | Sistem menampilkan pesan *error* yang mengatakan bahwa NIK dan PIN salah dan melakukan *refresh page* | Sesuai yang diharapkan | *Valid* |
| 3 | Mengosongkan atau sengaja menginput data yang tidak sesuai di *field* PLU atau Vendor saat menekan enter di modul Distribusi Harga | Sistem menampilkan pesan *error* yang mengatakan bahwa plu atau vendor salah dan mengosongkan semua *field* yang ada | Sesuai yang diharapkan | *Valid* |
| 4 | Melakukan *update* data pada modul distribusi harga | Sistem melakukan *update* data pada *database* yang sesuai dan perhitungan rumus berjalan dengan baik | Sesuai yang diharapkan | *Valid* |
| 5 | Menekan tombol simpan tanpa melakukan *edit* pada *field* yang ada | Sistem tidak akan mengeksekusi *query update* karena tidak ada perubahan data | Sistem menjalankan *query update* walau tidak ada data yang berubah | *Invalid* |
| 6 | Menekan tombol simpan saat *field* PLU atau Vendor kosong | Sistem menampilkan pesan *error* yang mengatakan bahwa plu atau vendor tidak boleh kosong dan mengosongkan semua *field* yang ada | Sesuai yang diharapkan | *Valid* |

Dilihat dari hasil pengujian *Black Box* (Tabel 1) yang telah dilakukan, dapat disimpulkan bahwa 5 dari 6 fungsi sistem dapat berfungsi sebagaimana mestinya. Karena terdapat 1 fungsi sistem yang tidak sesuai, maka dilakukan pengkodean ulang fungsi agar dapat berjalan sesuai dengan yang diharapkan.

## 4. KESIMPULAN DAN SARAN

Berdasarkan pada penelitian yang dilakukan dapat ditarik kesimpulan bahwa *framework Flask* dapat digunakan membangun modul distribusi harga dengan baik. *Flask* membantu *programmer* dalam mengembangkan aplikasi dikarenakan *Flask* memiliki bahasa yang sederhana, penggunaan yang ringan, dan *libraries* yang mampu menjawab kebutuhan.

Penerapan sistem *database* terdistribusi yang berfungsi meningkatkan kecepatan *transfer* data dan mempermudah pengelolaan data, sudah dapat diselesaikan dengan penggunaan *library flask-sqlalchemy*. Dengan penggunaan *flask-sqlalchemy*, aplikasi yang dibuat dapat dengan mudah melakukan buka tutup koneksi ke suatu *database* dengan cepat dan akurat.

Adapun saran yang dapat diberikan dalam pengembangan implementasi *Flask* pada pembangunan aplikasi adalah menambahkan jejak log ketika login maupun logout sehingga history pemakai bisa dilihat dan dilacak dengan mudah.

## DAFTAR PUSTAKA


[1] T. Keputusan, P. Pada, P. T. Indomaret, and M. Unit, "Analisis Pengaruh Harga, Promosi, Lokasi Dan Kualitas Pelayanan Terhadap Keputusan Pembelian Pada Pt. Indomaret Manado Unit Jalan Sea," *J. EMBA J. Ris. Ekon. Manajemen, Bisnis dan Akunt.*, vol. 6, no. 4, pp. 3068–3077, 2018, doi: 10.35794/emba.v6i4.21224.

[2] S. Hubbulwathan Duri and D. Mustika, "PENENTUAN HARGA POKOK PENJUALAN PADA TOKO PUTRI TANI SEJAHTERA MENGGUNAKAN APLIKASI BERBASIS WEB," vol. 2, no. 1, pp. 9–16, 2020, [Online]. Available: http://ojs.politeknikjambi.ac.id/jaab.

[3] R. R. Saragih, "Pemrograman dan Bahasa Pemrograman," pp. 1–43, 2016.

[4] J. Enterprise, *Python Untuk Programmer Pemula*. Elex media komputindo, 2019.

[5] R. Irsyad, "Penggunaan Python Web Framework Flask Untuk Pemula," 2018, doi: 10.31219/osf.io/t7u5r.

[6] I. L. Mulyahati, "Implementasi Machine Learning Prediksi Harga Sewa Apartemen Menggunakan Algoritma Random Forestt Melalui Framework Webste Flask Python," vol. 52, no. 5, pp. 482–487, 2020.

[7] "Multiple Databases with Binds — Flask-SQLAlchemy Documentation (2.x)." https://flask-sqlalchemy.palletsprojects.com/en/2.x/binds/ (accessed Apr. 15, 2022).

[8] "PostgreSQL: Documentation: 9.5: Aggregate Functions." https://www.postgresql.org/docs/9.5/functions-aggregate.html (accessed Mar. 28, 2022).

[9] S. Munawaroh, "Mengeksplorasi Database PostgreSQL dengan PgAdmin III," *J. Teknol. Inf. Din.*, vol. X, no. 2, pp. 103–107, 2005.

[10] B. P. Putra and Y. A. Susetyo, "IMPLEMENTASI API MASTER STORE MENGGUNAKAN FLASK, REST DAN ORM DI PT XYZ," *SISTEMASI*, vol. 9, no. 3, p. 543, 2020, doi: 10.32520/stmsi.v9i3.899.

[11] D. F. Ningtyas and N. Setiyawati, "Implementasi Flask Framework pada Pembangunan Aplikasi Purchasing Approval Request Flask Framework Implementation in Development Purchasing Approval Request Application," *J. Janitra Inform. dan Sist. Inf.*, vol. 1, no. 1, pp. 19–34, 2021, doi: 10.25008/janitra.v1i1.120.

[12] A. S. Nugroho, "Penerapan Basis Data Terdistribusi Untuk Transaksi dan Kontrol Penjualan Antar Cabang (Studi Kasus : Mitra Sejahtera Plastik Kebumen Jawa Tengah)," *Dr. Diss. Univ. Technol. Yogyakarta*, 2020.

[13] "Documentation for Visual Studio Code." https://code.visualstudio.com/docs (accessed Apr. 19, 2022).

[14] M. M. Gultom and Maryam, "Sistem Informasi Penjualan Material Bangunan Pada Toko Bangunan Berkah Information System of Sales Building Material ( Case Study : Berkah Building Shop )," *J. Tek. Inform.*, vol. 1, no. 2, pp. 79–86, 2020.

[15] H. N. Putra, "Implementasi Diagram UML (Unified Modelling Language) dalam Perancangan Aplikasi Data Pasien Rawat Inap pada Puskesmas Lubuk Buaya," *Sink. J. dan Penelit. Tek. Inform.*, vol. 2, no. 2, pp. 67–77, 2018, [Online]. Available: https://jurnal.polgan.ac.id/index.php/sinkron/article/view/130.