

II3160 Teknologi Sistem
Terintegrasi

**Layanan Preferensi dan
Customization Request
Clothing dengan Integrasi
Sistem Fashion Upcycling
Service**

Oncar Awwalu Rozaqy
18221068

Daftar Isi

Daftar Isi	1
Daftar Tabel	3
Daftar Gambar	4
Deskripsi Singkat Layanan	5
Tautan Layanan	5
Business capability	6
Core Subdomain & Bounded-Context	9
Arsitektur Sistem	11
Deskripsi Teknis Layanan	13
Layanan Awal (Customized Request Clothing)	13
Functional Requirements	13
Non Functional Requirements	13
Class Attribute	14
Class Operations	15
Use Case Diagram	16
Sequence Diagram	16
Layanan Hasil Integrasi	19
Use Case Diagram	20
Sequence Diagram	20
Implementasi Layanan	21
Website Application	22
Containerization	23
Implementasi	24
Interface	25
Dokumentasi API Endpoints	28
Layanan Awal (Customized Request Clothing)	28
Layanan Hasil Integrasi	29
Analisis	37



Daftar Tabel

Tabel 3.1 Daftar Kebutuhan Fungsional Sistem	13
Tabel 3.2 Daftar Kebutuhan Non Fungsional Sistem	13
Tabel 3.3 Daftar Atribut Kelas	14
Tabel 3.4 Daftar Operasi Kelas	15
Tabel 4.1 Dokumentasi API Endpoints layanan awal	28
Tabel 4.2 Dokumentasi API Endpoints layanan hasil integrasi	29

Daftar Gambar

Gambar 1.1 Value Streams dan Business Capability Customized Request Clothing	7
Gambar 1.2 Subdomain Business Capability Order Management	7
Gambar 1.3 Value Streams dan Business Capability Integrasi	8
Gambar 1.4 Core Subdomain dan Bounded-Context	9
Gambar 2.1 Arsitektur Sistem Integrasi Layanan	11
Gambar 3.1 Use Case Diagram Customized Request Clothing	16
Gambar 3.2 Sequence Diagram Register	17
Gambar 3.3 Sequence Diagram Login	18
Gambar 3.4 Sequence Diagram Cloth Preference	18
Gambar 3.5 Sequence Diagram Customization	19
Gambar 3.6 Sequence Diagram FashUp	20
Gambar 3.7 Sequence Diagram FashUp	21
Gambar 3.8 Router Integrasi FashUp	21
Gambar 3.9 Fungsi yang digunakan dalam integrasi	22
Gambar 3.10 Dockerfile	24
Gambar 3.11 Penggunaan Axios pada React	25
Gambar 3.12 Halaman Login	25
Gambar 3.13 Halaman Register	26
Gambar 3.14 Halaman Customization (Utama)	26
Gambar 3.15 Halaman Customization (Customize Form)	27
Gambar 3.16 Halaman Customization (FashUp)	27
Gambar 5.1 Modul Koneksi MySQL	37

Deskripsi Singkat Layanan

Customized Request Clothing merupakan bisnis yang menyediakan produk berupa pakaian yang didesain sesuai dengan permintaan pengguna dan dibuat langsung dengan menggunakan tangan atau tenaga manusia. Desain yang diinginkan oleh pengguna akan diproduksi oleh tim produksi sehingga produk yang dihasilkan akan sesuai dengan keinginan pengguna. Oleh karena itu, dipilih sebuah *core service* yaitu "Customized Request". Layanan ini memungkinkan pengguna untuk memberikan preferensi baju sesuai dengan keinginannya seperti *font*, *size*, *color*, dan tipe pakaian. Setelah itu, sistem akan menampilkan pakaian-pakaian yang sesuai dengan keinginan pengguna tersebut. Apabila telah sesuai, pengguna dapat melakukan *request order* dengan memberikan beberapa instruksi khusus terkait baju yang diinginkan. Layanan yang akan dibangun juga menerapkan beberapa fungsionalitas lain yang mendukung terciptanya layanan ini.

Layanan ini akan diintegrasikan dengan layanan *Fashion UpCycling* yaitu FashUp. FashUp adalah layanan yang memungkinkan penggunanya untuk melakukan upcycling produk tekstil lama menjadi sebuah produk baru yang memiliki nilai lebih. FashUp akan membantu layanan ini untuk melakukan beberapa fungsionalitas yaitu rekomendasi produk dan estimasi kuantitas produk yang dapat dihasilkan berdasarkan material yang dimiliki.

Tautan Layanan

1. Layanan Utama Customized Request Clothing

Berikut merupakan tautan layanan hasil implementasi beserta dokumentasi API dan source code.


Github: <https://github.com/oncarrozagy/TST-Customized-Request-Clothing.git>

API : [Dokumentasi API](#)

Web : [Customized Request Clothing Website](#)

Akun Dummy:

Customer:



Username : ongkarr

Password : 12345678

Admin

Username : adminuser

Password : adminpass

2. Layanan Integrasi FashUp (18221090)

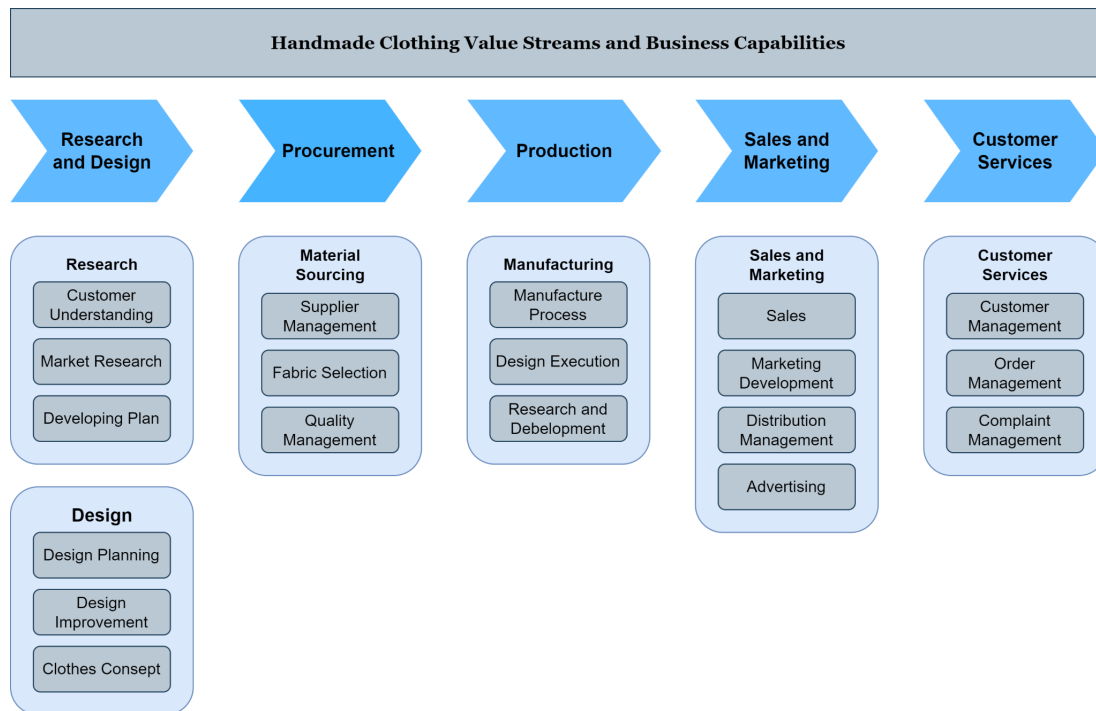
Berikut merupakan tautan layanan FashUp yang digunakan untuk integrasi pada layanan utama

Github: <https://github.com/marchelinefanni/Final-Project-TST.git>

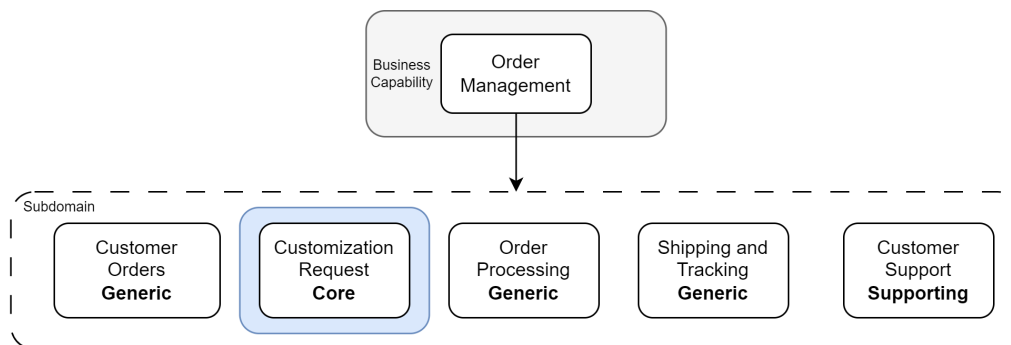
API : <http://fashupauth.hactd6f8f6brg2ca.southeastasia.azurecontainer.io>

Business capability

Berikut merupakan *value streams* dan *business capabilities* dari *Customized Request Clothing*. *Business capability* yang digunakan adalah "Order Management" yang memiliki beberapa subdomain yang dijelaskan pada gambar berikutnya.



Gambar 1.1 Value Streams dan Business Capability Customized Request Clothing



Gambar 1.2 Subdomain Business Capability Order Management

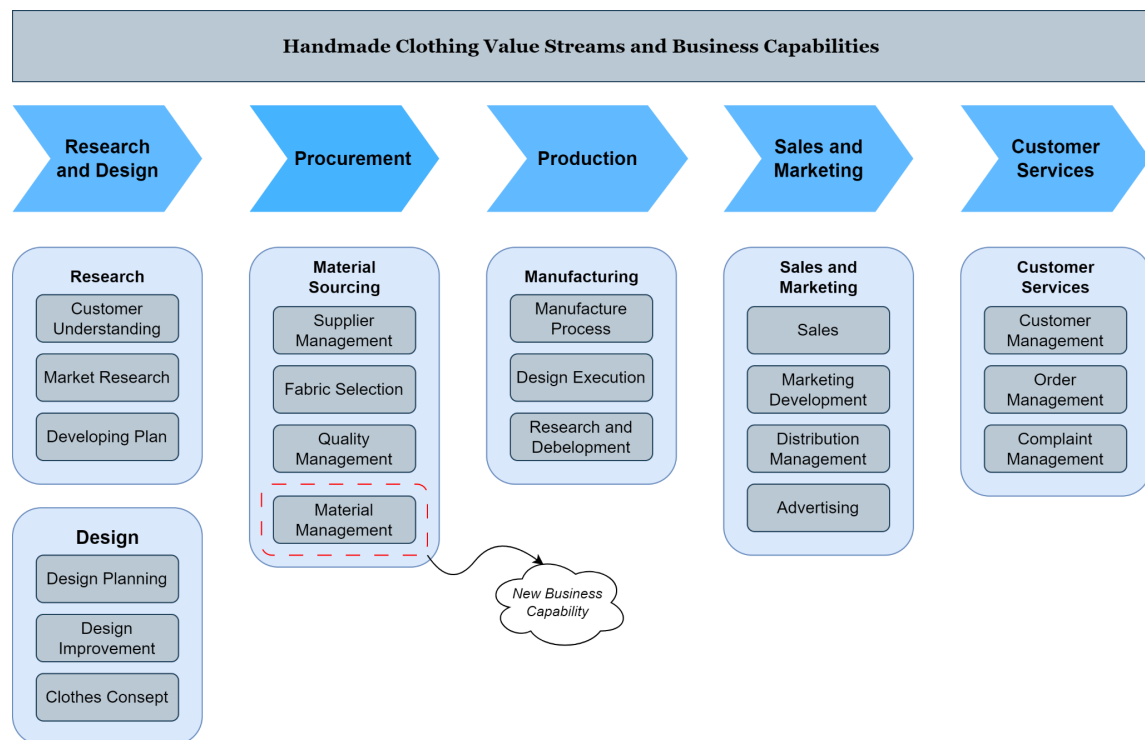
Business Capability "Order Management" memiliki beberapa subdomain yang memiliki fungsi atau layanan berbeda. Setiap subdomain memiliki beberapa kategori, diantaranya adalah sebagai berikut.

1. **Core**: Merupakan subdomain yang bersifat unik, menjadi diferensiasi dibanding kompetitor, *competitive edge*, dan sebagai pembeda dibandingkan dengan bisnis lain.

2. **Generic:** Masalah umum yang dapat diselesaikan dengan solusi *off-the-shelf* dan juga dilakukan oleh pihak lain sehingga bukan sesuatu yang menjadi faktor kompetitif dengan yang lain dan umumnya sudah tersedia.
3. **Supporting:** Tidak memberikan keuntungan kompetitif, namun diperlukan untuk berjalannya sebuah organisasi. Sifatnya sederhana dan *low-entry barrier*.

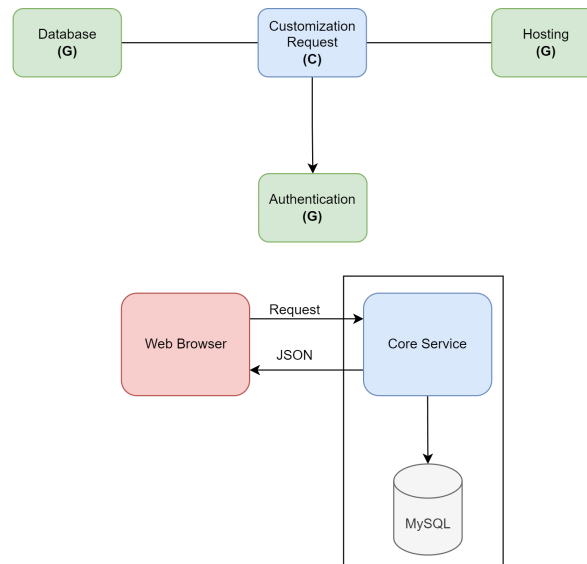
Pada bisnis ini, subdomain yang menjadi *core service* adalah *customization request* karena dapat menjadi sebuah keunggulan ataupun *competitive edge* sekaligus pembeda dibandingkan dengan bisnis lain yang bergerak pada industri yang sama.

Bisnis ini melakukan integrasi dengan FashUp yang akan menghasilkan sebuah *business capability* baru. *Business capability* tersebut adalah *material management* dan akan masuk ke *business capability parent* (Level 1) yaitu *Material Sourcing*. Sehingga, berikut merupakan *value streams* dan *business capability* bisnis ini setelah dilakukan integrasi.



Gambar 1.3 Value Streams dan Business Capability Integrasi

Core Subdomain & Bounded-Context



Gambar 1.4 Core Subdomain dan Bounded-Context

Core subdomain dari bisnis *Customized Request Clothing* memiliki 3 *bounded-context* yaitu Database, Authentication, dan Hosting. Ketiga *bounded-context* ini dikategorikan sebagai generik dan implementasinya menerapkan layanan *off-the-shelf* dengan rincian sebagai berikut.


1. Database

Basis data yang akan digunakan pada layanan ini adalah jenis basis data relasional. Data akan disimpan dalam bentuk tabel-tabel yang saling berelasi atau terkait sehingga skemanya sudah tetap dan lebih terstruktur. Model basis data ini akan memudahkan penelusuran data sehingga cocok untuk digunakan pada layanan ini.

DBMS yang dipilih untuk menyimpan data adalah MySQL. Pemilihan DBMS tersebut didasarkan karena kemudahan untuk digunakan dan diimplementasikan. MySQL cocok digunakan untuk proyek dalam skala kecil dan tidak membutuhkan kompleksitas yang tinggi. Selain itu, MySQL sudah banyak digunakan dan memiliki kinerja yang cukup mapan. Basis data yang dibuat akan diimplementasikan dan disimpan pada cloud yang disediakan oleh Azure yang terkoneksi dengan MySQL.

2. Authentication

Autentikasi pada layanan ini dilakukan dengan OAuth2 dan JWT (JSON Web Tokens). Autentikasi dilakukan memberikan akses pada pihak tertentu yang



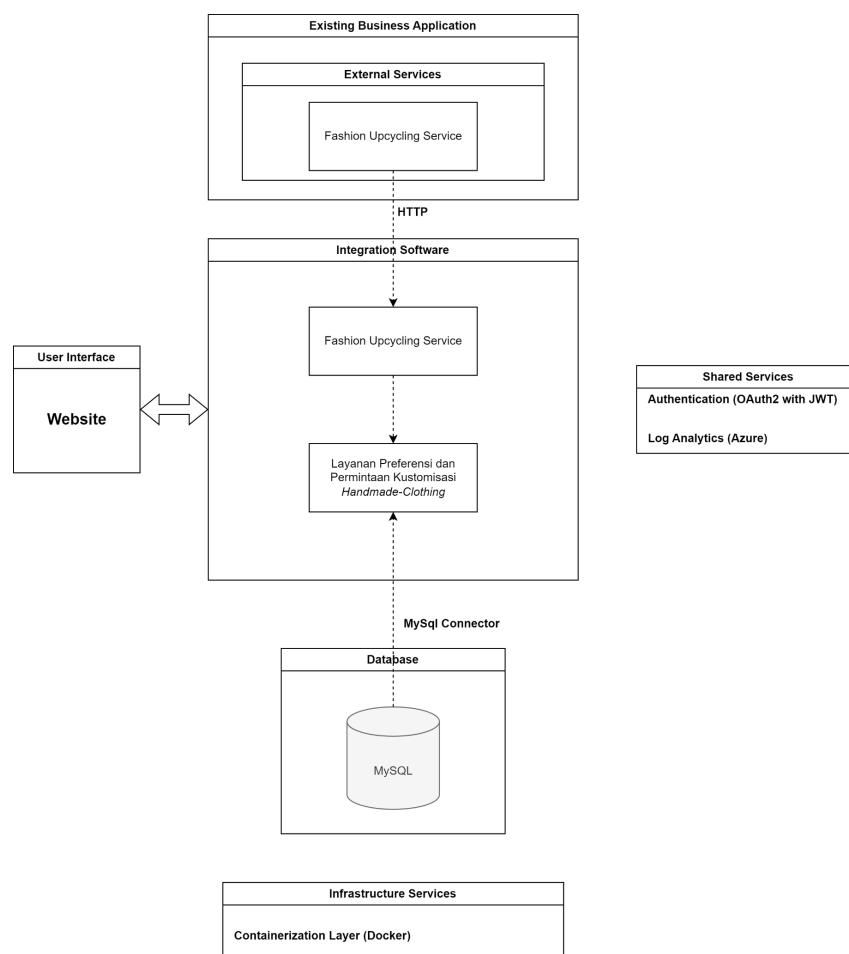
memiliki kepentingan pada suatu *endpoints* yang dimiliki layanan. Oleh karena itulah pada layanan ini diterapkan *role* untuk membatasi akses pada beberapa *endpoint* untuk mencegah wewenang pengaksesan yang salah.

3. Hosting

Hosting dari *endpoints* layanan ini diimplementasi menggunakan Azure Container App. Alasannya adalah karena penggunaannya cukup mudah dan terdapat kredit gratis bagi mahasiswa. Layanan ini nantinya juga akan memiliki *user interface* berupa *website* untuk mempermudah pelanggan dalam menggunakan atau menerima layanan dari bisnis ini.


Arsitektur Sistem

Bisnis *Customized Request Clothing* memerlukan integrasi dengan sebuah layanan eksternal berupa *UpCycling system* yang digunakan untuk menerapkan beberapa fungsionalitas berupa rekomendasi produk berdasarkan material yang dimiliki dan jumlah produk yang dapat dihasilkan berdasarkan jumlah material yang dimiliki. Oleh karena itu, berikut merupakan gambaran rancangan arsitektur layanan hasil integrasi antara layanan preferensi dan permintaan kustomisasi *Customized Request Clothing* dengan *Fashion Upcycling Service*.



Gambar 2.1 Arsitektur Sistem Integrasi Layanan

Gambar di atas merupakan arsitektur perancangan integrasi antara Layanan *Customized Request Clothing* dengan *Fashion Upcycling Service*. Layanan integrasi dapat dilakukan dengan memanggil *endpoint* yang dimiliki oleh *external services* yaitu *Fashion Upcycling Service*. Kedua layanan ini saling terhubung melalui protokol HTTP.



Database yang digunakan adalah MySQL yang terhubung dengan Layanan *Customized Request Clothing* melalui MySQL connector. Dalam integrasi ini, *user interface* yang digunakan adalah *website* dengan *shared services* yaitu OAuth2 dan JWT untuk autentikasi serta Azure untuk *log analytics*. Terakhir, pada infrastruktur layanan, integrasi ini menerapkan *containerization layer* berupa Docker.

Deskripsi Teknis Layanan

Layanan Awal (Customized Request Clothing)

Functional Requirements

Berikut merupakan daftar kebutuhan fungsional layanan awal *Customized Request Clothing*.

Tabel 3.1 Daftar Kebutuhan Fungsional Sistem

ID	Kebutuhan	Penjelasan
F01	Sistem harus mampu menampilkan daftar produk pakaian yang tersedia	Sistem dapat menampilkan data menu pakaian yang tersedia dan dapat dipesan oleh pengguna dengan rincian informasi berupa nama, harga, <i>font</i> , ukuran, dan warna.
F02	Sistem mampu menyediakan laman autentikasi untuk memastikan pengguna yang berhak saja yang dapat mengakses layanan	Sistem dapat menyediakan fitur autentikasi berupa <i>login</i> dan <i>signup</i> untuk memastikan pengguna dapat mengakses laman dengan hak akses yang benar.
F03	Sistem harus mampu menyediakan fitur preferensi pakaian untuk membantu pengguna dalam memilih pakaian yang sesuai	Sistem dapat menampilkan formulir yang dapat diisi pengguna untuk menyesuaikan preferensi pakaian yang dipilih pengguna dan menampilkan produk-produk yang sesuai dengan preferensinya.
F04	Sistem harus mampu menyediakan fitur kustomisasi pakaian bagi pelanggan yaitu berupa formulir kustomisasi	Sistem dapat menampilkan formulir kustomisasi yang dapat diisi pengguna sesuai dengan keinginannya terkait produk yang dipilih.

Non Functional Requirements

Berikut merupakan daftar kebutuhan non fungsional yang harus dapat dipenuhi oleh layanan awal *customized request*.

Tabel 3.2 Daftar Kebutuhan Non Fungsional Sistem

ID	Parameter	Kebutuhan
NF01	<i>Availability</i>	Sistem dapat diakses selama 24 jam dalam 7 hari seminggu.
NF02	<i>Reliability</i>	Sistem harus memiliki tingkat <i>error</i> di bawah 5 %.
NF03	<i>Memory</i>	Sistem harus memiliki memori yang cukup untuk menyimpan seluruh data dan juga operasi sistem.
NF04	<i>Response time</i>	Sistem harus memiliki <i>Response time</i> kurang dari 5 detik dalam memproses instruksi pengguna.
NF05	<i>Security</i>	Sistem harus dapat menjaga keamanan data pengguna
NF06		Sistem hanya dapat diakses oleh pengguna yang mempunyai akses dengan melalui proses autentikasi

Class Attribute

Berikut merupakan daftar atribut kelas yang terdapat pada layanan *customized request*.

Tabel 3.3 Daftar Atribut Kelas

No	Nama Kelas	Atribut
1	User	<u>userID</u> : int
		firstName: string
		lastName : string
		email : string
		phoneNumber : string
		shippingAddress: string
		username: string
		password: string
		role : string
2	Products	<u>productID</u> : int
		description: string
		price: int
		stock: int
		default_font: string
		default_color: string
		size: string
		productType: string
		imageurl : string
3	customizationRequests	<u>customizationID</u> : int

		customerID: int
		productID: int
		specialInstructions: string
		orderDate: datetime

Class Operations

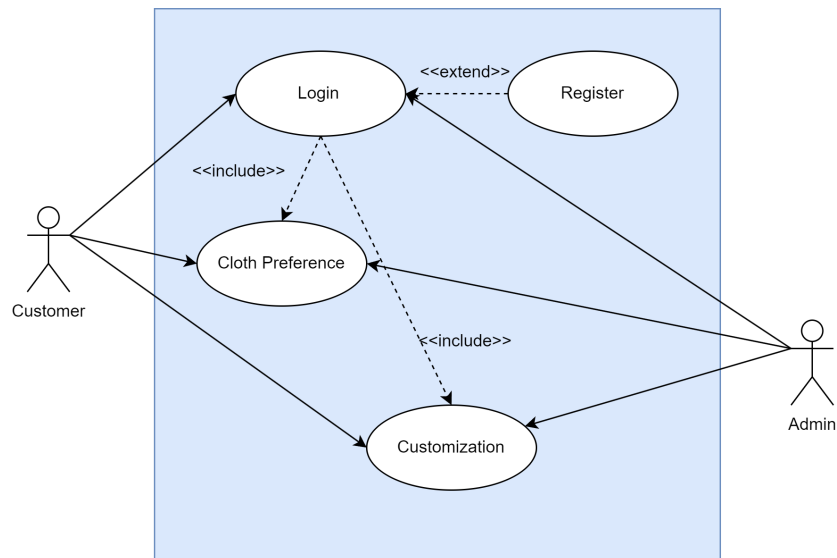
Berikut merupakan daftar operasi kelas yang terdapat pada layanan *customized request*.

Tabel 3.4 Daftar Operasi Kelas

No	Nama Kelas	Operasi
1	User	getAllUsers
		deleteUser
		createUser
		login
2	Products	getAllProducts
		createProduct
		getProductById
		updateProduct
		deleteProduct
		getRecommendationByPreference
3	customizationRequests	getAllRequests
		createRequest
		getRequestByUserID
		deleteRequest

Use Case Diagram

Berikut merupakan *use case diagram* dari layanan yang akan diimplementasi. Pada *use case* ini, aktor yang terlibat adalah *customer* dan *admin*. Setiap *use case* akan diberikan rincian skenarionya berdasarkan *sequence diagram* yang akan dijelaskan pada bagian berikutnya.



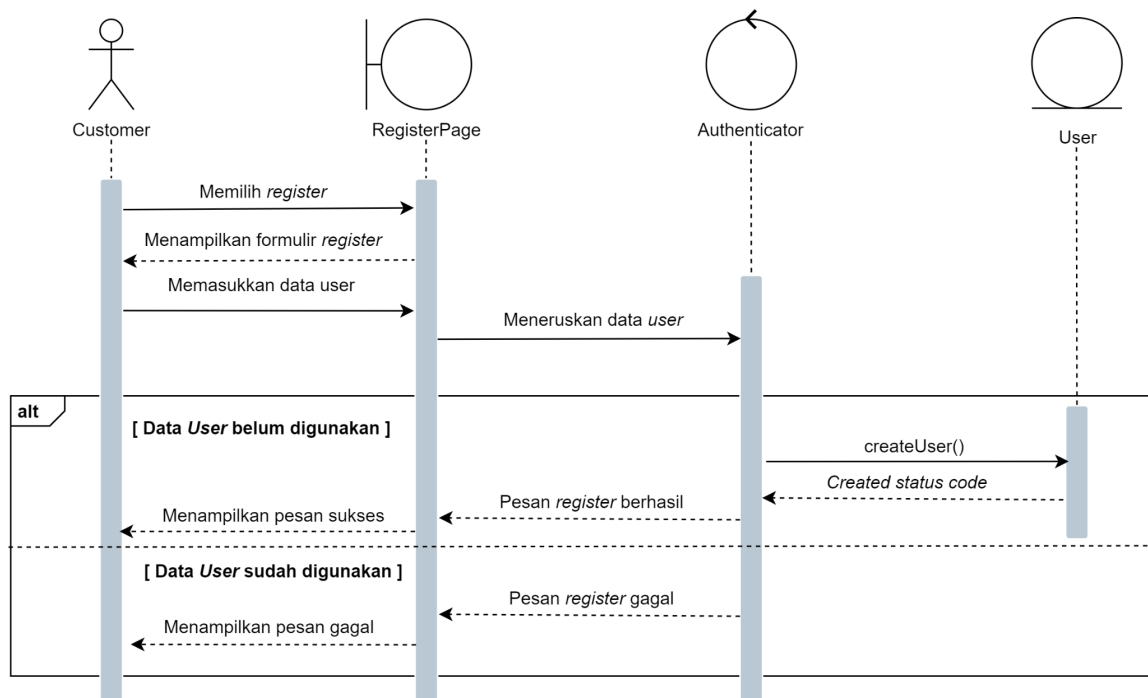
Gambar 3.1 Use Case Diagram Customized Request Clothing

Pada *use case diagram* tersebut, kedua aktor harus melakukan login untuk dapat mengakses *use case* yang lainnya. Pada layanan ini, *use case* yang akan diimplementasikan adalah **Login**, **Register**, **Cloth Preference**, dan **Customization**. Kedua *role* berhak atas keseluruhan *use case* yang ada. Bagi admin, fungsionalitas-fungsionalitas yang tidak diimplementasi dapat dilakukan melalui API layanan.

Sequence Diagram

1. Register

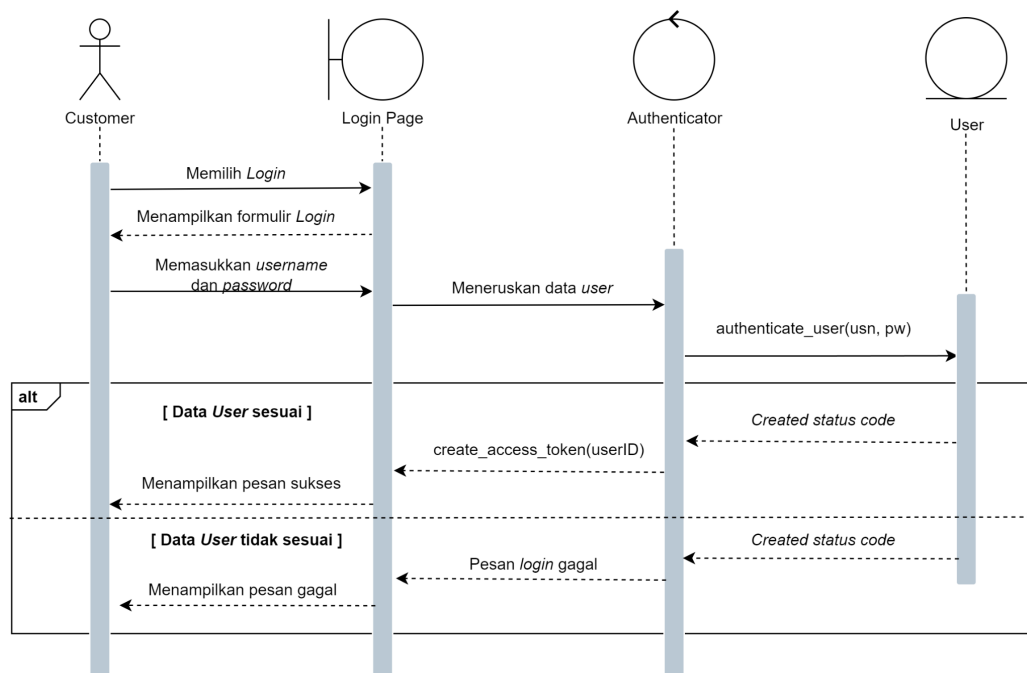
Berikut merupakan *sequence diagram* dari *register* pada layanan *customized request*.



Gambar 3.2 Sequence Diagram Register

2. Login

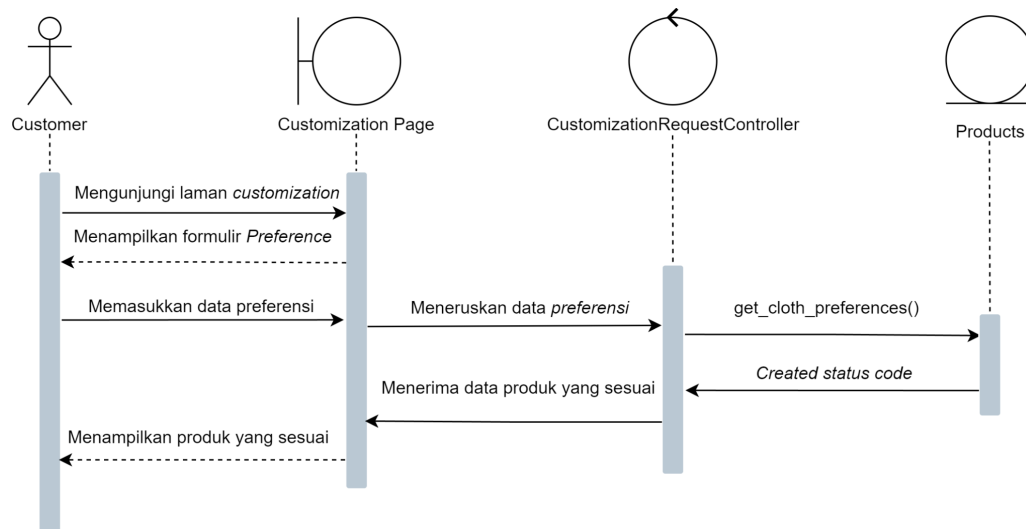
Berikut merupakan *sequence diagram* dari *login* pada layanan *customized request*.



Gambar 3.3 Sequence Diagram Login

3. Cloth Preference

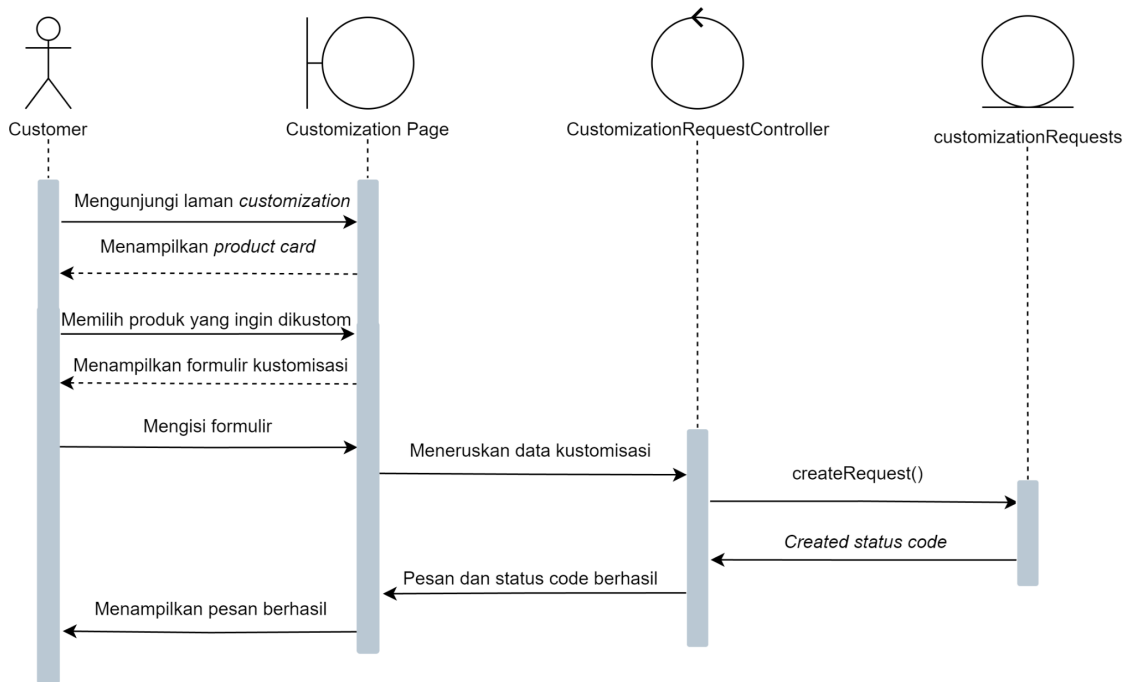
Berikut merupakan *sequence diagram* dari *cloth preference* pada layanan *customized request*.



Gambar 3.4 Sequence Diagram Cloth Preference

4. Customization

Berikut merupakan *sequence diagram* dari *customization* pada layanan *customized request*.



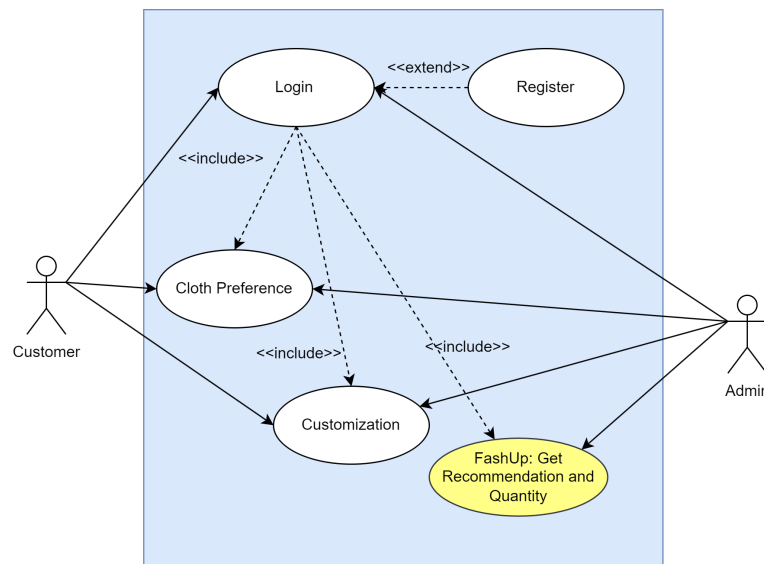
Gambar 3.5 Sequence Diagram Customization

Layanan Hasil Integrasi

Layanan FashUp akan integrasi dan diimplementasi pada bisnis *Customized Request Clothing* sebagai *business capability* baru pada *Material Sourcing*. Pihak *administrator* atau admin dari bisnis ini dapat menggunakan layanan FashUp untuk melakukan pengelolaan material. Pihak admin dapat mendapatkan rekomendasi dan jumlah produk pakaian yang dapat dihasilkan berdasarkan material (beserta beratnya) yang dimiliki. Layanan hasil integrasi ini dapat diakses pada antarmuka yang sama, yaitu *website*, dengan melakukan login sebagai admin.

Use Case Diagram

Integrasi dengan layanan FashUp akan menghasilkan sebuah layanan baru. Berikut merupakan *use case* hasil integrasi. Layanan FashUp hanya akan bisa diakses oleh *admin*.

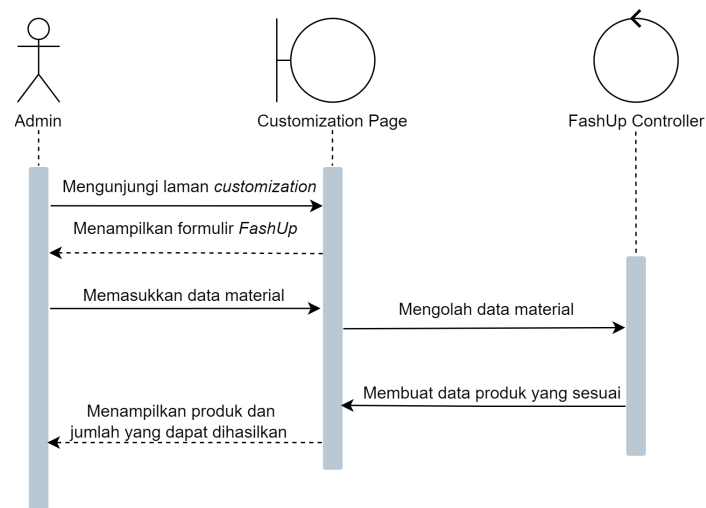


Gambar 3.6 Sequence Diagram FashUp

Admin akan bertugas untuk mengelola material yang dimiliki menggunakan layanan FashUp. Melalui data material dan jumlah material yang dimiliki, admin dapat mengetahui apa saja produk pakaian yang dapat dibuat dan diperoleh beserta jumlahnya melalui layanan FashUp. *Use Case* ini nantinya akan diimplementasi pada antarmuka yang sama dengan *Customization* dan *Cloth Preference*, namun *customer* tidak dapat mengaksesnya.

Sequence Diagram

Berikut merupakan *sequence diagram* layanan FashUp yang diintegrasikan pada layanan awal.



Gambar 3.7 Sequence Diagram FashUp

Implementasi Layanan

Integrasi layanan dilakukan pada API dan diimplementasi pada *website* layanan nantinya. Oleh karena itu, sebelum melakukan implementasi, perlu dilakukan integrasi layanan FashUp ke layanan utama *Customized Request Clothing*. Integrasi dilakukan dengan menggunakan bahasa Python, memanfaatkan *library* bawaan yaitu **Requests** untuk melakukan *hit* pada endpoint milik FashUp. *Endpoint* yang berhasil di-*hit* akan dibuat pada *router* baru yaitu FashUp Router.

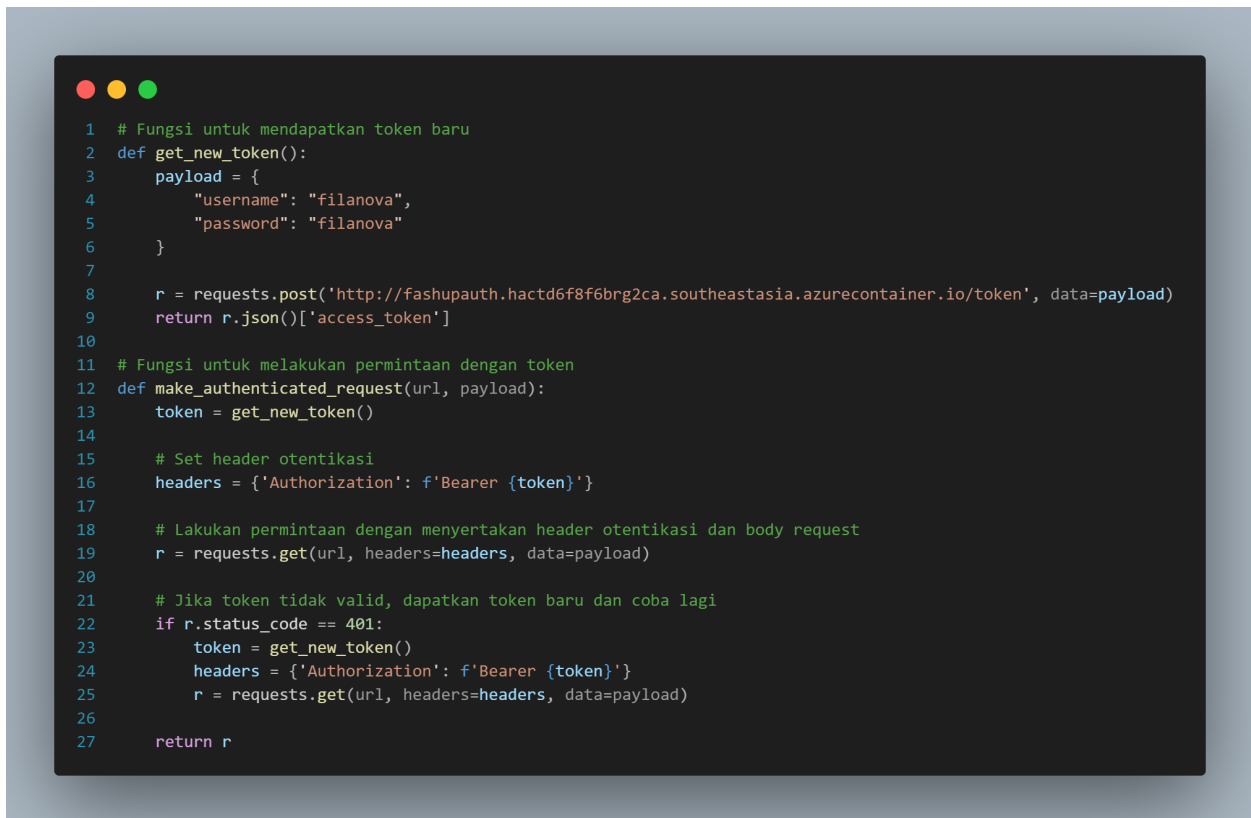
```

1 fashup_router = APIRouter(
2     tags=['fashup']
3 )
4
5 fashup = {}
6
7
8 @fashup_router.get('/productrecommendations')
9 async def get_product_based_on_material(material_input: str, user: Annotated[Users, Depends(get_current_user)]):
10     if user[8] != "admin":
11         raise HTTPException(status_code=status.HTTP_401_UNAUTHORIZED, detail="You are not an admin.")
12     response = make_authenticated_request(f'http://fashupauth.hactd6f8f6b9g2ca.southeastasia.azurecontainer.io/recommendation?material_input={material_input}', {})
13     return response.json()
14
15 @fashup_router.get('/quantity')
16 async def get_quantity(material_input: str, weight_input: int, user: Annotated[Users, Depends(get_current_user)]):
17     if user[8] != "admin":
18         raise HTTPException(status_code=status.HTTP_401_UNAUTHORIZED, detail="You are not an admin.")
19     response = make_authenticated_request(f'http://fashupauth.hactd6f8f6b9g2ca.southeastasia.azurecontainer.io/quantity?material_input={material_input}&weight_input={weight_input}', {})
20     return response.json()
  
```

Gambar 3.8 Router Integrasi FashUp

Pada router tersebut, digunakan juga dua buah fungsi yang berfungsi untuk melakukan *hit* dengan menggunakan *access_token* yang diperoleh pada layanan

FashUp. Hal ini dilakukan karena *endpoint productrecommendations* dan *quantity* pada FashUp memerlukan autentikasi.



```
1 # Fungsi untuk mendapatkan token baru
2 def get_new_token():
3     payload = {
4         "username": "filanova",
5         "password": "filanova"
6     }
7
8     r = requests.post('http://fashupauth.hactd6f8f6brg2ca.southeastasia.azurecontainer.io/token', data=payload)
9     return r.json()['access_token']
10
11 # Fungsi untuk melakukan permintaan dengan token
12 def make_authenticated_request(url, payload):
13     token = get_new_token()
14
15     # Set header otentikasi
16     headers = {'Authorization': f'Bearer {token}'}
17
18     # Lakukan permintaan dengan menyertakan header otentikasi dan body request
19     r = requests.get(url, headers=headers, data=payload)
20
21     # Jika token tidak valid, dapatkan token baru dan coba lagi
22     if r.status_code == 401:
23         token = get_new_token()
24         headers = {'Authorization': f'Bearer {token}'}
25         r = requests.get(url, headers=headers, data=payload)
26
27     return r
```


Gambar 3.9 Fungsi yang digunakan dalam integrasi

Website Application

Customized Request Clothing dapat diakses oleh pengguna melalui aplikasi web dengan lingkungan pengembangan sebagai berikut.

- Operating System : MacOS, Windows
- DBMS : MySQL
- Development Tools : Vite React Js
- Programming Language : Typescript
- Filling System : Github
- Deployment : Vercel

Customized Request Clothing adalah aplikasi pakaian yang dapat diakses melalui aplikasi web. Aplikasi ini dikembangkan dengan menggunakan beberapa teknologi dan lingkungan pengembangan. Pengguna dapat mengakses aplikasi ini baik melalui sistem

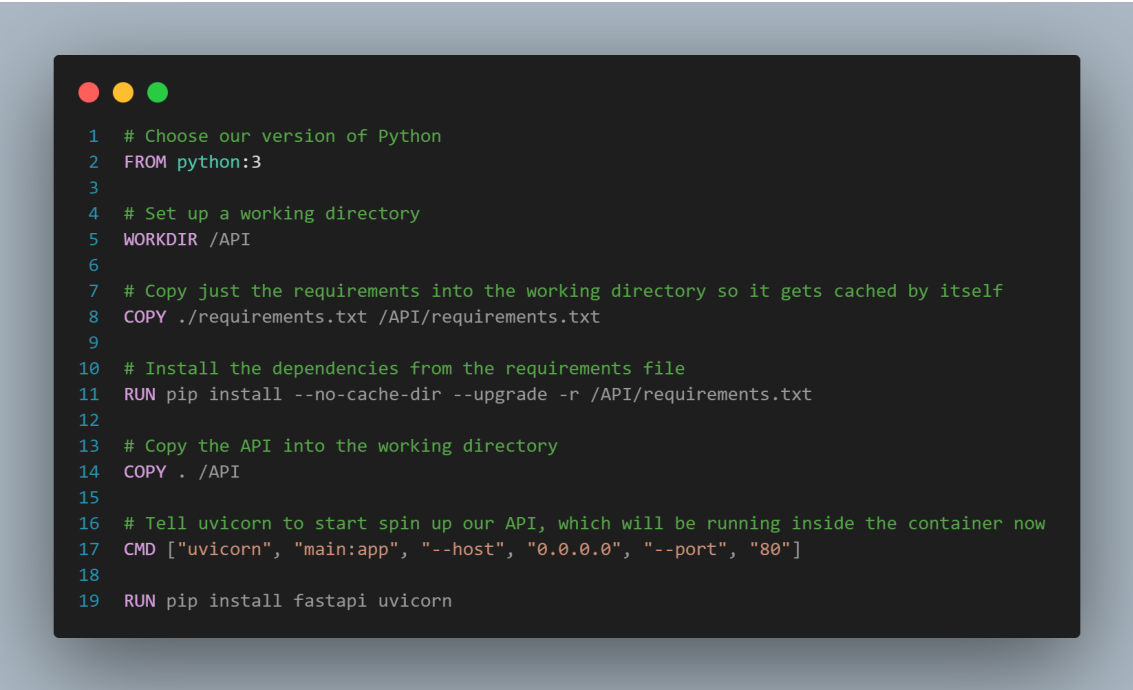


operasi MacOS maupun Windows. *Database Management System (DBMS)* yang digunakan adalah MySQL, memberikan kehandalan dalam penyimpanan dan pengelolaan data. Lingkungan pengembangan dibangun menggunakan Vite React Js, sebuah *framework* React yang memungkinkan pengembangan cepat dan efisien. Bahasa pemrograman yang digunakan dalam pengembangan aplikasi ini adalah Typescript, yang membawa fitur *strong-typing* dan konsep pemrograman OOP untuk meminimalkan kesalahan terkait tipe dan struktur data. Sumber kontrol proyek dilakukan melalui Github untuk mempermudah proses kolaborasi dan *deployment*. Terakhir, untuk *deployment*, aplikasi ini di-host menggunakan Vercel, memastikan ketersediaan dan skalabilitas yang optimal dan dipilih karena kemudahan penggunaannya. Dengan kombinasi dari berbagai teknologi ini, *Customized Request Clothing* memberikan pengalaman pengguna yang baik melalui antarmuka yang akan dilampirkan pada bagian berikutnya.

Containerization

Dalam menjalankan aplikasi layanan ini, *containerization* dilakukan dengan Docker. Docker dipilih sebagai kontainer aplikasi ini karena kemampuannya untuk menyederhanakan dan mengoptimalkan siklus hidup aplikasi. Docker adalah platform perangkat lunak yang memungkinkan aplikasi untuk dikemas, didistribusikan, dan dijalankan beserta seluruh dependensinya dalam lingkungan terisolasi yang dikenal sebagai "kontainer." Alasan lain mengapa Docker digunakan adalah fleksibilitasnya dalam menjalankan kontainer di berbagai sistem operasi. Penggunaan Docker juga menjadi pilihan karena mampu mengatasi masalah yang sering muncul akibat perbedaan konfigurasi dan dependensi antara lingkungan pengembangan dan produksi. Dengan begitu, aplikasi yang akan dikembangkan dapat berjalan dengan baik serta konsisten mulai dari tahap *development* hingga *deployment*.

Dalam mengatur *containerization* dari layanan ini, akan dilakukan sebuah konfigurasi *dockerfile*. *Dockerfile* memungkinkan aplikasi dengan bahasa pemrograman Python dengan menggunakan FastAPI dan Uvicorn ke dalam sebuah kontainer Docker. Pilihan menggunakan Docker sebagai platform deployment memberikan keuntungan dalam hal portabilitas dan konsistensi lingkungan aplikasi di berbagai tahap pengembangan dan produksi. Berikut merupakan *dockerfile* yang digunakan pada API layanan *customized request clothing*.



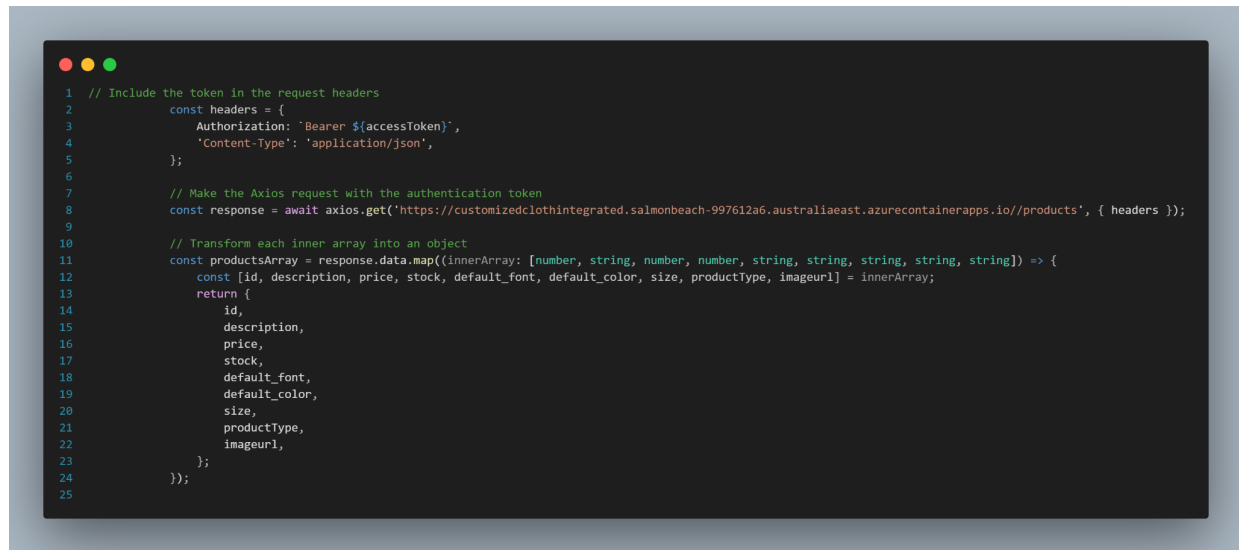
```
1 # Choose our version of Python
2 FROM python:3
3
4 # Set up a working directory
5 WORKDIR /API
6
7 # Copy just the requirements into the working directory so it gets cached by itself
8 COPY ./requirements.txt /API/requirements.txt
9
10 # Install the dependencies from the requirements file
11 RUN pip install --no-cache-dir --upgrade -r /API/requirements.txt
12
13 # Copy the API into the working directory
14 COPY . /API
15
16 # Tell uvicorn to start spin up our API, which will be running inside the container now
17 CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "80"]
18
19 RUN pip install fastapi uvicorn
```

Gambar 3.10 Dockerfile

Implementasi

Setelah melakukan kontainerisasi, layanan API dapat di-*deploy* agar dapat digunakan sebagai *backend service* dari *website* nantinya. *Deployment* API digunakan melalui Microsoft Azure menggunakan **Container App**. Langkah-langkah *deployment* sudah dijelaskan pada bagian laporan atau tugas-tugas sebelumnya.

Setelah berhasil melakukan *deploy* API sebagai *backend*, akan dibuat *frontend* yang disambungkan dengan *backend service*. Penyambungan dilakukan dengan menggunakan modul *axios* pada *React*. Hal ini dilakukan untuk memperoleh data dari *backend* API yang didapat dari database sehingga dapat ditampilkan pada *website*. Berikut merupakan contoh implementasi *axios* untuk mendapatkan hal tersebut.



Gambar 3.11 Penggunaan Axios pada React

Interface

Berikut merupakan implementasi antarmuka yang digunakan pada layanan *Customized Request Clothing* yang telah diintegrasikan dengan FashUp.

1. Login

A screenshot of a login form interface. The form is centered on a white background and has a light gray border. At the top, the word "LOGIN" is written in bold red text. Below it, there are two input fields: "Username" and "Password". The "Username" field contains the text "example123". The "Password" field is masked with dots. Below the input fields is a red button with the text "Login" in white. At the bottom of the form, there is a link that says "Don't have an account yet? Register".

Gambar 3.12 Halaman Login

2. Register

Register

First Name

John

Last Name

Doe

Phonenumber

081234567890

Address

Jl. Ganesha No. 10 Bandung

Email

Email must be unique

Username

Username must be at least 4 characters and unique

Password

Password must be at least 6 characters

Sign Up

Already have an account? [Login](#)

Gambar 3.13 Halaman Register

3. Customization

Welcome to Customize Clothing

You can set your preference here:

Font:
Example: Arial


Color:
Example: Red

Size:
Select Size


Product Type:
Example: T-Shirt

Apply Reset


Here are some products we recommends




Cotton T-shirt with various colors
Price: Rp 200000
Default Font: Arial
Default Color: Red
Size: L
[Customize This](#)




Warm hoodie for cold weather
Price: Rp 350000
Default Font: Times New Roman
Default Color: Black
Size: M
[Customize This](#)




Casual Jeans with Logo Embroidery
Price: Rp 450000
Default Font: Sans Serif
Default Color: Blue
Size: M
[Customize This](#)




Summer Dress with Floral Print
Price: Rp 300000
Default Font: Script
Default Color: Yellow
Size: XL
[Customize This](#)




Sports Shorts with Logo
Price: Rp 250000
[Customize This](#)



Graphic Print T-shirt
Price: Rp 220000
[Customize This](#)



Denim Jacket with Custom Patch
Price: Rp 420000
[Customize This](#)



Printed Sweatshirt
Price: Rp 300000
[Customize This](#)

Gambar 3.14 Halaman Customization (Utama)

Dokumentasi API Endpoints

Layanan Awal (Customized Request Clothing)

Berikut merupakan dokumentasi API layanan *Customized Request Clothing* yang terdiri dari Authentications, Customizations, Products, dan Users. Pada tabel berikut juga terdapat penjelasan apakah *endpoint* tersebut memerlukan autentikasi dan siapa saja *role* yang berhak mengaksesnya.

Tabel 4.1 Dokumentasi API Endpoints layanan awal

Method	Endpoint	Usage	Auth	Role
Default				
GET	/	Menampilkan pesan "Welcome to Customization API!"	False	All
Authentications				
POST	/authentications/login	Melakukan login bagi pengguna yang telah terdaftar untuk mendapatkan akses token	False	All
POST	/authentications/register	Melakukan pendaftaran pengguna dengan mengisi beberapa parameter data (firstname, lastname, phonenumber, email, address, username, password, dan role)	False	All
Customizations				
GET	/customizationRequests	Menampilkan seluruh data <i>request</i> yang pernah dibuat pada layanan	True	Admin
POST	/customizationRequests	Membuat <i>request</i> baru dengan beberapa parameter (productID dan specialInstructions)	True	All
GET	/customizationRequests/{userID}	Menampilkan data <i>request</i> yang pernah dibuat oleh pengguna dengan userID tertentu	True	All
GET	/customizationRequests/{font}/{color}/{size}/{productType}	Menampilkan data produk yang memiliki kecocokan dengan data <i>clothing preference</i> (font, color, size, productType) yang dimasukkan <i>user</i>	True	All
DELETE	/customizationRequests/{customizationID}	Menghapus data <i>request</i> dengan customizationID tertentu	True	Admin

Users				
GET	/users	Menampilkan seluruh data pengguna yang terdaftar pada layanan	True	Admin
DELETE	/users	Menghapus data pengguna dengan userID tertentu	True	Admin
Products				
GET	/products	Menampilkan seluruh data produk (productID, description, price, stock, default_font, default_color, size, productType, imageurl) yang terdaftar pada basis data layanan	True	All
POST	/products	Membuat data produk baru ke dalam basis data layanan <i>Customized Request Clothing</i> berdasarkan data produk yang dimasukkan oleh pengguna melalui <i>endpoint</i> ini	True	Admin
GET	/products/{productID}	Menampilkan data produk (productID, description, price, stock, default_font, default_color, size, productType, imageurl) dengan productID tertentu	True	All
PUT	/products/{productID}	Mengubah data produk dengan productID tertentu pada basis data	True	Admin
DELETE	/products/{productID}	Menghapus data produk dengan productID tertentu pada basis data	True	Admin

Layanan Hasil Integrasi

Berikut merupakan dokumentasi layanan FashUp yang telah diintegrasikan dengan layanan awal *Customized Request Clothing*.

Tabel 4.2 Dokumentasi API Endpoints layanan hasil integrasi

Method	Endpoint	Usage	Auth	Role
FashUp				
GET	/productrecommendations	Menampilkan rekomendasi produk fashion upcycling berdasarkan material_input (jenis bahan) yang dimasukkan oleh pengguna	True	Admin
GET	/quantity	Menampilkan kuantitas dari produk	True	Admin

		katalog yang dapat diperoleh berdasarkan data material_input (jenis bahan) dan weight_input (berat bahan) yang dimasukkan oleh pengguna		
--	--	---	--	--

Penjelasan Endpoints

Berdasarkan daftar *endpoint* beserta dengan kegunaannya di atas, untuk mengaksesnya diperlukan beberapa jenis permintaan (*request*) dan informasi *header* tertentu. Selain itu, *response* dari API ini adalah berupa format JSON. Berikut merupakan dokumentasi rinci setiap *endpoint* yang melibatkan detail mengenai jenis permintaan yang dapat digunakan dan respons yang dihasilkan dari setiap *endpoint* yang di implementasi atau digunakan pada *website*.

1. Login

Method : POST

Endpoint : /authentications/login

Usage : Melakukan login bagi pengguna yang telah terdaftar untuk mendapatkan akses token

Parameters :

- username: string
- password: string

Request

Request Sample (cURL):

```
curl -X 'POST' \
'https://customizedclothintegrated.salmonbeach-997612a6.australiaeast.azurecontainerapps.io/authentications/login' \
-H 'accept: application/json' \
-H 'Content-Type: application/x-www-form-urlencoded' \
-d
'grant_type=&username=oncarrozaqy&password=12345678&scope=&client_id=&client_secret='
```

Response

Response Example (JSON)

– Successful Response (200)

```
{
  "access_token":
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoib25jYXJyb3phcXkiLCJleHAiOjE3MDI2NDEyMj19.yvdaV3wcaGB-bLhbmfdKb-apqGQSMcFPyVAVl6OlQho",
  "token_type": "bearer",
  "role": "admin"
}
```

2. Register

Method: POST

Endpoint: /authentications/register

Usage: Melakukan pendaftaran pengguna dengan mengisi beberapa parameter data (firstname, lastname, phonenumber, email, address, username, password, dan role)

Parameters:

- firstName: string
- lastName: string
- email: string
- phoneNumber: string
- shippingAddress: string
- username: string
- password: string
- role: string

Request

Request Sample (cURL):

```
curl -X 'POST' \
'https://customizedclothintegrated.salmonbeach-997612a6.australiaeast.azurecontainerapps.io/authentications/register?firstname=ongkar&lastname=awalu&phonenu
mber=1928471284&address=jl%20bandung&email=test%40mail.com&password=12345678&u
sername=apajadeh&role=admin' \
-H 'accept: application/json' \
-d ''
```

Response

Response Example (JSON)

– Successful Response (200)

```
"Registration Completed!"
```

3. Create Request

Method : POST
Endpoint : /customizationRequests
Usage : Membuat request baru dengan beberapa parameter (productID dan specialInstructions)
Parameters :

- productID: int
- specialInstructions

Request

Request Sample (cURL):

```
curl -X 'POST' \  
  'https://customizedclothintegrated.salmonbeach-997612a6.australiaeast.azurecon-  
tainerapps.io/customizationRequests?productID=1&specialInstructions=test' \  
  -H 'accept: application/json' \  
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoib25jYXJyY3phcXkiLCJleHAiOjE3MDI2NDE1ODR9.Si_KCVOg521faxrfyk0cJl4kGI2BR9YvjPtUv84lQqM' \  
  -d ''
```

Response

Response Example (JSON)

– Successful Response (200)

```
"Request Created"
```

4. Clothes Preferences

Method : GET
Endpoint : /customizationRequests/{font}/{color}/{size}/{productType}
Usage : Menampilkan data produk yang memiliki kecocokan dengan data clothing preference (font, color, size, productType) yang dimasukkan user
Parameters :

- font: string
- color: string
- size: string
- productType: string

Request

Request Sample (cURL):


```
curl -X 'GET' \
'https://customizedclothintegrated.salmonbeach-997612a6.australiaeast.azurecon
tainerapps.io/customizationRequests/arial/red/l/t-shirt' \
-H 'accept: application/json' \
-H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoib25jYXJyb3phcXkiLCJleHAiOjE3M
DI2NDIxMjR9.eankVHqwJN-uLmnYWnrMLeBLbB7UbxFukq75oHHNZXc'
```

Response

Response Example (JSON)

– Successful Response (200)

```
[
  [
    1,
    "Cotton T-shirt with various colors",
    200000,
    100,
    "Arial",
    "Red",
    "L",
    "T-shirt",
    "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQiZT5I377eG3HlyRwjOwf1_
12ha2QTJm6TKg&usqp=CAU"
  ],
  [
    30,
    "Red T-shirt",
    20000,
    76,
    "Arial",
    "Red",
    "XXL",
    "T-shirt",
    "google.com"
  ]
]
```

5. Read All Products

Method : GET
Endpoint : /products
Usage : Menampilkan seluruh data produk (productID, description, price, stock, default_font, default_color, size, productType, imageurl) yang terdaftar pada basis data layanan
Parameters : No Parameters

Request

Request Sample (cURL):

```
curl -X 'GET' \

'https://customizedclothintegrated.salmonbeach-997612a6.australiaeast.azurecon
tainerapps.io/products' \
-H 'accept: application/json' \
-H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoib25jYXJyb3phcXkiLCJleHAiOjE3M
DI2NDIxMjR9.eankVHqwjN-uLmnYWnrMLeBLbB7Ubx-fukq75oHHNZXc'
```

Response

Response Example (JSON)

– Successful Response (200)

```
[
  [
    1,
    "Cotton T-shirt with various colors",
    200000,
    100,
    "Arial",
    "Red",
    "L",
    "T-shirt",
    "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQiZT5I377eG3HlyRwjOwf1_
12ha2QTJm6TKg&usqp=CAU"
  ],
  [
    2,
    "Warm hoodie for cold weather",
    350000,
    50,
    "Times New Roman",
    "Black",
    "M",
    "Hoodie",
    "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcT_ZfRq50_X1BkWS9PdPziLY
Sd_QTs_Xg7niAnVeRbgf5MkU_lpWQkGlmLO4MMirjrBGnM&usqp=CAU"
  ],
]
```

6. Get Product Based On Material

Method : GET
Endpoint : /productrecommendations
Usage : Menampilkan rekomendasi produk fashion upcycling berdasarkan material_input (jenis bahan) yang dimasukkan oleh pengguna
Parameters :

- material_input: string

Request

Request Sample (cURL):

```
curl -X 'GET' \
'https://customizedclothintegrated.salmonbeach-997612a6.australiaeast.azurecon
tainerapps.io/productrecommendations?material_input=Denim' \
-H 'accept: application/json' \
-H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoib25jYXJyb3phcXkiLCJleHAiOjE3M
DI2NDIxMjR9.eankVHqwJN-uLmnYWnrMLeBLbB7Ubx-fukq75oHHNZXc'
```

Response

Response Example (JSON)

– Successful Response (200)

```
[
  "Here's your product recommendation(s): Jacket, Tote bag"
]
```

7. Get Quantity

Method : GET

Endpoint : /quantity

Usage : Menampilkan kuantitas dari produk katalog yang dapat diperoleh berdasarkan data material_input (jenis bahan) dan weight_input (berat bahan) yang dimasukkan oleh pengguna

Parameters :

- material_input: string
- weight_input: int

Request

Request Sample (cURL):

```
curl -X 'GET' \
'https://customizedclothintegrated.salmonbeach-997612a6.australiaeast.azurecon
tainerapps.io/quantity?material_input=Denim&weight_input=1000' \
-H 'accept: application/json' \
-H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoib25jYXJyb3phcXkiLCJleHAiOjE3M
DI2NDIxMjR9.eankVHqwJN-uLmnYWnrMLeBLbB7Ubx-fukq75oHHNZXc'
```

Response

Response Example (JSON)

– Successful Response (200)

```
[
  "With that amount of material, you can get:",
  {
    "headband": 9,
    "scarf": 2,
    "tote_bag": 2,
    "wallet": 19,
    "blanket": 1,
    "dress": 1,
    "jacket": 1
  }
]
```

Analisis

Layanan *Customized Request Clothing* diintegrasikan dengan layanan FashUp milik Marchelline Fanni (18221090) untuk menambah sebuah *business capability* baru mengenai *material management*. Proses pengembangan layanan dilakukan dengan melalui beberapa langkah yang telah dilakukan diantaranya sebagai berikut.


1. Melakukan konfigurasi *database* melalui MySQL. Server yang digunakan adalah *flexible server* dari Microsoft Azure, memanfaatkan kredit akun sebagai mahasiswa.
2. Membangun *microservice* utama *Customized Request Clothing*. Hal ini dilakukan untuk menyediakan API layanan yang mendukung keberjalanan *core service* layanan. Setiap *endpoints* yang dibangun terdapat pada dokumentasi API layanan awal. API ini selanjutnya dihubungkan dengan *database* menggunakan MySQL Connector.

A screenshot of a code editor showing a Python script for connecting to a MySQL database. The script is numbered from 1 to 24. It imports the mysql.connector module and defines a configuration dictionary with host, user, password, database, client_flags, and ssl_ca. A try block attempts to connect to the database using the configuration. If an error occurs, it checks for specific error codes (ER_ACCESS_DENIED_ERROR and ER_BAD_DB_ERROR) and prints corresponding messages. If the connection is successful, it creates a cursor object.

```
1 import mysql.connector
2 from mysql.connector import errorcode
3
4 config = {
5     'host' : 'customized-handmade-clothing.mysql.database.azure.com',
6     'user' : 'oncarrozaqyy',
7     'password' : '*****',
8     'database' : 'customized_clothing',
9     'client_flags' : [mysql.connector.ClientFlag.SSL],
10    'ssl_ca' : './ssl/DigiCertGlobalRootG2.crt.pem'
11 }
12
13 try :
14     conn = mysql.connector.connect(**config)
15     print("Connection established")
16 except mysql.connector.Error as err :
17     if err.errno == errorcode.ER_ACCESS_DENIED_ERROR :
18         print("Something is wrong with the user name or password")
19     elif err.errno == errorcode.ER_BAD_DB_ERROR :
20         print("Database does not exist")
21     else :
22         print(err)
23 else :
24     cursor = conn.cursor()
```

Gambar 5.1 Modul Koneksi MySQL

3. Menerapkan autentikasi menggunakan OAuth2 dan JWT (JSON Web Token). Hal ini ditujukan untuk memberikan akses yang benar kepada pengguna tertentu dan mencegah pengaksesan yang salah dari pihak yang tidak berwenang.
4. Melakukan integrasi dengan layanan FashUp menggunakan *library* requests yang dimiliki oleh Python seperti yang sudah dijelaskan pada bagian sebelumnya.

- 
5. Melakukan kontainerisasi menggunakan Docker dan melakukan *deploy* API ke Container App di Microsoft Azure.
 6. Membangun frontend layanan untuk *website* aplikasi menggunakan lingkungan pengembangan yang sudah disebutkan sebelumnya.
 7. Melakukan *deploy* aplikasi menggunakan vercel dan melakukan pengujian aplikasi. Hasil yang didapat adalah seluruh fungsionalitas berjalan dengan benar, meskipun terkadang *response time* yang dibutuhkan untuk mengakses beberapa fungsionalitas cukup lambat.

Dalam melakukan langkah-langkah tersebut, terdapat beberapa hal yang menarik dan memerlukan perhatian lebih untuk ditangani. Salah satunya adalah ketika melakukan *request* ke API milik FashUp. Layanan FashUp juga menerapkan autentikasi yang memerlukan akses token untuk mengakses *endpoint* yang dimilikinya. Oleh karena itu, diperlukan sebuah fungsi yang akan menyimpan akses token yang diperoleh dari FashUp. Selanjutnya, akses token tersebut akan digunakan sebagai *header* ketika akan melakukan *hit* terhadap *endpoint* FashUp yang digunakan pada layanan *Customization Request Clothing*. Detail dari implementasi tersebut sudah dijelaskan pada bagian implementasi.

Hasil integrasi kedua layanan ini tentunya memberikan *value proposition* baru bagi bisnis. Hadirnya FashUp tentu sangat membantu keberjalanan bisnis terutama dalam proses pengelolaan material pakaian. Hasilnya adalah produksi pakaian dapat dioptimalkan dan *customer* dapat menerima layanan kustomisasi pakaian dengan pengalaman yang berkesan, tidak hanya sekedar memesan pakaian yang diinginkan.