

**DOKUMEN CD - 5**



**SISTEM SIMULASI JARINGAN 5G BERBASIS  
*DASHBOARD* TERPUSAT DALAM LINGKUNGAN  
*CLOUD NATIVE* UNTUK *MANAGED TELECOM*  
*LABORATORY***

Oleh:

**Ari Erginta Ginting/1101204178**

**Bagus Dwi Prasetyo/1101204109**

**Ima Dewi Arofani/1101204375**

**Mochamad Rafli Hadiana/1101202426**

**PRODI S1 TEKNIK TELEKOMUNIKASI  
FAKULTAS TEKNIK ELEKTRO  
UNIVERSITAS TELKOM  
BANDUNG**

**2024**

### Lembar Pengesahan Dokumen

Judul Capstone Design : Sistem Emulasi Jaringan 5G Berbasis *Dashboard* Terpusat dalam Lingkungan *Cloud Native* untuk *Managed Telecom Laboratory*.

Jenis Dokumen : Usulan Gagasan dan Pemilihan Topik

Nomor Dokumen : FTE-CD-4





Nomor Revisi : 1

Tanggal Pengesahan : 23/12/2023

Fakultas : Fakultas Teknik Elektro

Program Studi : S1 Teknik Telekomunikasi

Jumlah Halaman : 23 hal

Data Pemeriksaan dan Persetujuan			
Ditulis Oleh	Nama : Ari Erginta Ginting	Jabatan : Mahasiswa	
	NIM : 1101204178	Tanda Tangan	
	Nama : Bagus Dwi Prasetyo	Jabatan : Mahasiswa	
	NIM : 1101204109	Tanda Tangan	
	Nama : Ima Dewi Arofani	Jabatan : Mahasiswa	
	NIM : 1101204375	Tanda Tangan	
	Nama : Mochamad Rafli Hadiana	Jabatan : Mahasiswa	
	NIM : 1101202426	Tanda Tangan	
Disetujui Oleh	Nama : Ridha Muldhina Negara, S.T., M.T.	Jabatan : Pembimbing 1	
	Tanggal : 23/12/2023	Tanda Tangan	
	Nama : Prananto Bayu H.	Jabatan : Pembimbing 2	
	Tanggal :	Tanda Tangan	

### Timeline Revisi Dokumen

<b>Versi, Tanggal</b>	<b>Revisi</b>	<b>Perbaikan yang dilakukan</b>	<b>Halaman Revisi</b>
1,21/12/2023	Penambahan Gambaran mengenai koneksi end-to-end 5G Network	Menambahkan Ilustrasi mengenai koneksi end-to-end 5G Network	8,10
	Penambahan Ilustrasi Arsitektur Aplikasi Frontend dan Backed lebih jelas	Menambahkan Ilustrasi Arsitektur Aplikasi Frontend dan Backed secara merinci	13,14

## 5.1 Skenario Umum Pengujian

Dalam skenario umum pengujian aplikasi, dilakukan empat jenis pengujian utama. Pertama, pengujian API untuk memastikan bahwa setiap API berfungsi dengan baik dan benar. Kedua, pengujian *End-to-End* (E2E) yang bertujuan untuk memverifikasi seluruh alur kerja aplikasi dari perspektif pengguna. Ketiga, pengujian performa untuk mengukur batas kapasitas sistem aplikasi, dengan menilai seberapa banyak pengguna yang dapat menggunakan aplikasi hingga mencapai kapasitas maksimum. Terakhir, *User Acceptance Testing* (UAT) bertujuan menilai kepuasan pengguna terhadap aplikasi. UAT melibatkan pengguna akhir, admin, dan calon pembeli untuk memberikan *feedback* terhadap fungsionalitas dan pengalaman pengguna. Pengujian ini dilakukan di lingkungan pra-produksi setelah semua pengujian teknis selesai, guna memastikan aplikasi memenuhi ekspektasi pengguna sebelum diluncurkan ke produksi.

## 5.2 Detil Pengujian

Pengujian aplikasi kami melibatkan empat jenis utama pengujian yang dirancang untuk memastikan fungsionalitas, alur, kinerja, dan kepuasan pengguna terhadap aplikasi. Keempat jenis pengujian ini adalah *API Testing*, *End-to-End (E2E) Testing*, *Performance Testing*, dan *User Acceptance Testing* (UAT). Setiap jenis pengujian memiliki tujuan dan metodologi spesifik yang membantu memastikan bahwa aplikasi berfungsi sesuai dengan harapan pengguna dan standar kualitas yang telah ditetapkan. Berikut ini adalah penjelasan lebih rinci tentang skema masing-masing pengujian.

### 5.2.1 API Testing

*API testing* bertujuan untuk memastikan bahwa seluruh WebSocket API dan REST API berjalan sesuai dengan yang diharapkan menggunakan metode *Whitebox Testing*. Menggunakan metode ini, penguji berinteraksi dengan API yang telah tersedia untuk memastikan fungsionalitasnya bekerja dengan baik dan benar. Selain itu, penguji juga memiliki akses ke dalam *source code* yang apabila terdapat error selama testing, penguji dapat langsung memperbaiki kode yang error tersebut. Pada aplikasi Postman, penguji hanya memberikan input, dan memeriksa output terhadap spesifikasi yang diharapkan. Pengujian untuk setiap API akan dilakukan sebanyak 30x percobaan agar semakin meyakinkan fungsionalitasnya bekerja tanpa kendala. Berikut adalah tabel yang mendeskripsikan alur dari *API testing* yang akan dilakukan.

**Tabel 0.1 Langkah API Testing**

Langkah	Deskripsi
Membuat dan Menyimpan Request	Buka Postman, buat request baru, masukkan URL endpoint, pilih metode HTTP, dan tambahkan header serta body yang diperlukan. Simpan request dalam collection.
Mengirim Request dan Memeriksa Respons	Kirim request dengan mengklik tombol <b>Send</b> dan periksa status code serta body respons untuk memastikan sesuai dengan yang diharapkan.
Menganalisis Hasil Test	Analisis hasil pengujian yang ditampilkan, periksa test yang lulus atau gagal, dan identifikasi serta laporkan masalah jika ada.

Pengujian dilakukan menggunakan Postman dan Visual Studio Code dengan hasil yang diharapkan dari pengujian ini adalah semua API berfungsi sesuai spesifikasi yang diharapkan, interaksi antar *libraries* tidak menghasilkan kesalahan, dan sistem mampu menangani input yang tidak valid dengan tepat.

Pada tahap ini, *testing* untuk tipe REST API yang tersedia akan dilakukan menggunakan aplikasi Postman untuk melihat respons dari API tersebut beserta dengan HTTP *status code* yang dikeluarkan. Sedangkan *testing* untuk tipe WebSocket API yang tersedia akan dilakukan menggunakan bantuan aplikasi *code editor* untuk menjalankan *script* yang telah dibuat menggunakan Django *libraries* bernama Websockets untuk melakukan testing koneksi WebSocket secara lokal. Seluruh API akan dikelompokkan menjadi lima bagian berdasarkan jenis dan fungsionalitasnya untuk memudahkan pelaksanaan *testing* dan membuatnya lebih terstruktur.

#### 5.2.1.1 User Management APIs

API terkait *user management* memiliki fungsi krusial dalam pembuatan, pengubahan, dan penghapusan akun *user* yang akan digunakan dalam aplikasi. API ini hanya dapat diakses oleh akun admin terkecuali satu, yang dapat digunakan oleh user untuk melihat informasi dari akun miliknya. Berikut adalah daftar API *endpoint* terkait *user management*.

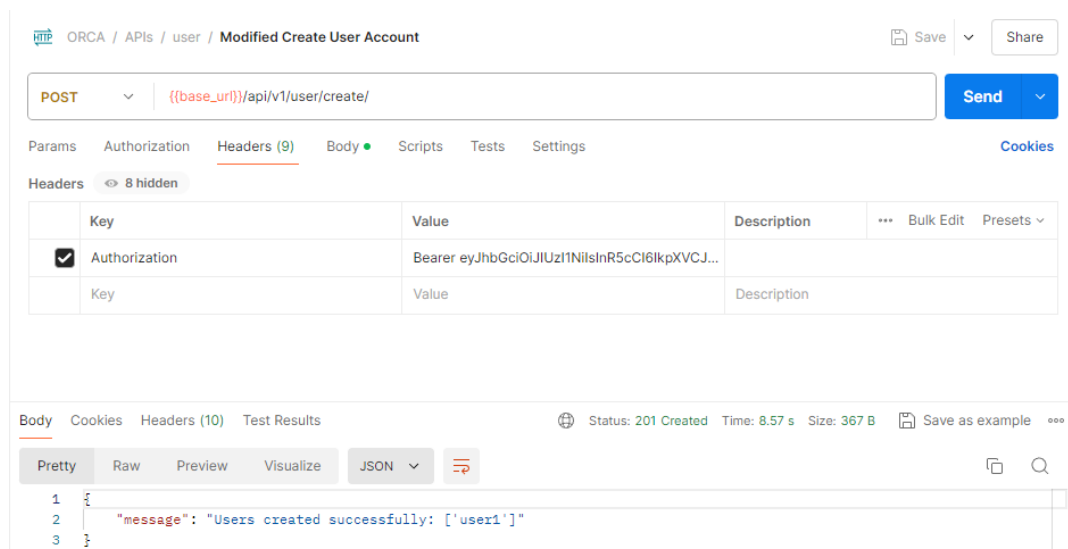
**Tabel 0.2 List API User Management**

<i>Category</i>	<i>Method</i>	<i>API Endpoint</i>
-----------------	---------------	---------------------

Create user	POST	{{base_url}}/api/v1/user/create/
List all users	GET	{{base_url}}/api/v1/user/list/
Update user's password	PATCH	{{base_url}}/api/v1/user/update/{{users_id}}/
Delete user	DELETE	{{base_url}}/api/v1/user/delete/{{users_id}}/
Get requested user information	GET	{{base_url}}/api/v1/user/information/

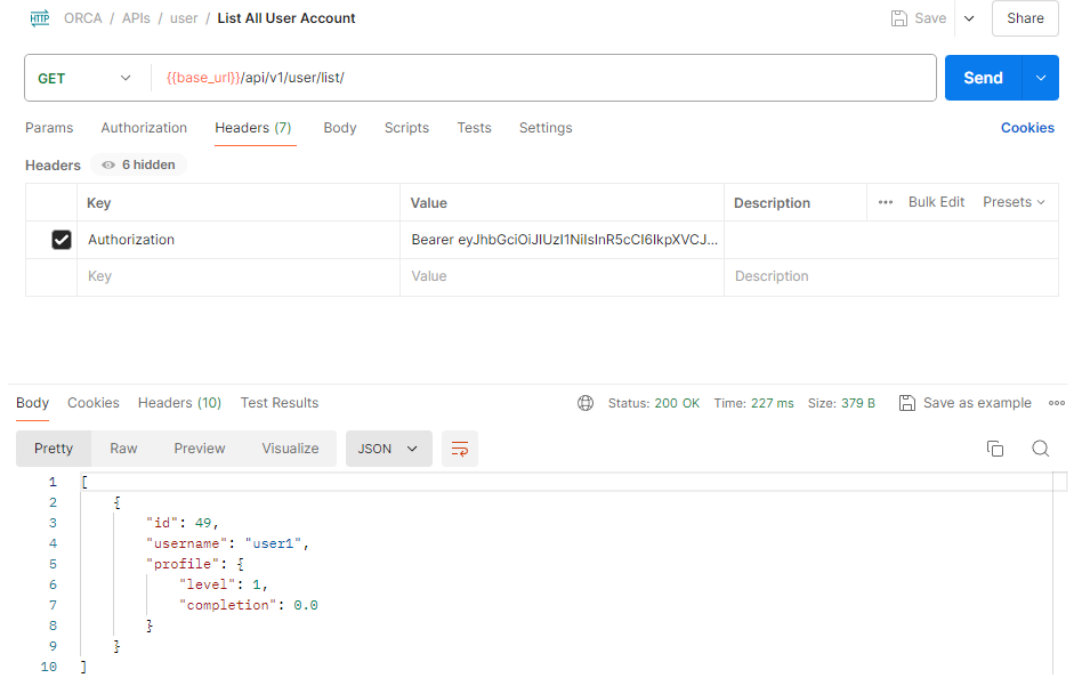
Tabel di atas berisi daftar REST API *endpoint* yang tersedia terkait dengan *user management*. Setiap API di atas memiliki fungsionalitasnya masing-masing, berikut akan disajikan proses testing seluruh fungsionalitas untuk role admin dan user melalui User Management APIs yang dilakukan menggunakan aplikasi Postman.

#### 1. Admin Berhasil Membuat Akun User Baru



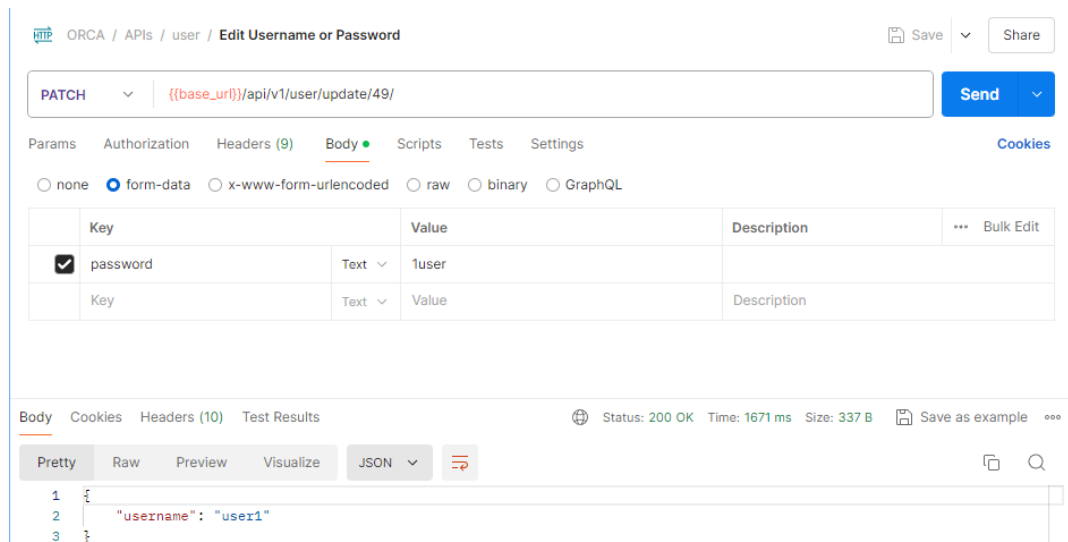
**Gambar 0.1 Response API Create user**

#### 2. Admin Berhasil Melihat Daftar Akun User



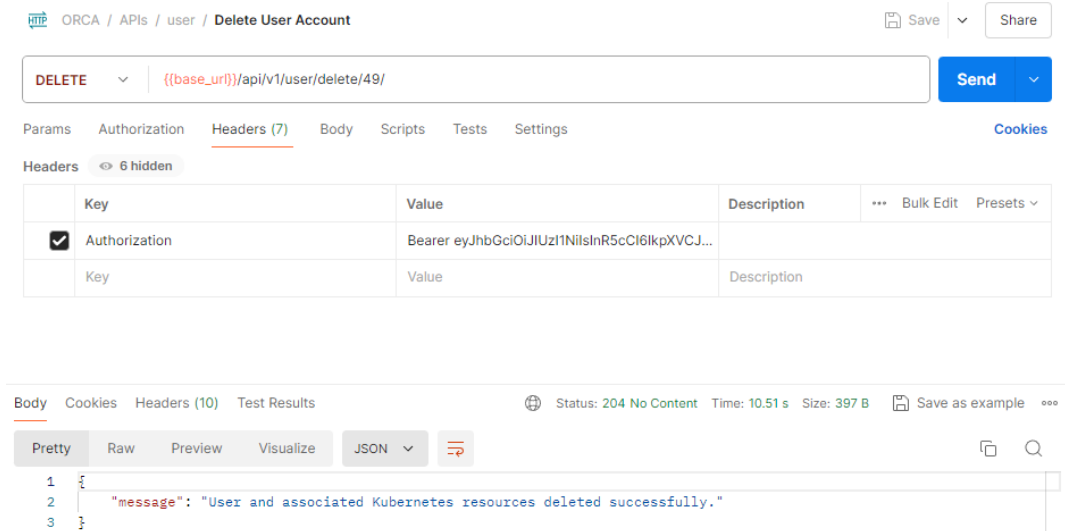
**Gambar 0.2 Response API Read All User**

### 3. Admin Berhasil Mengubah *Password* Akun User



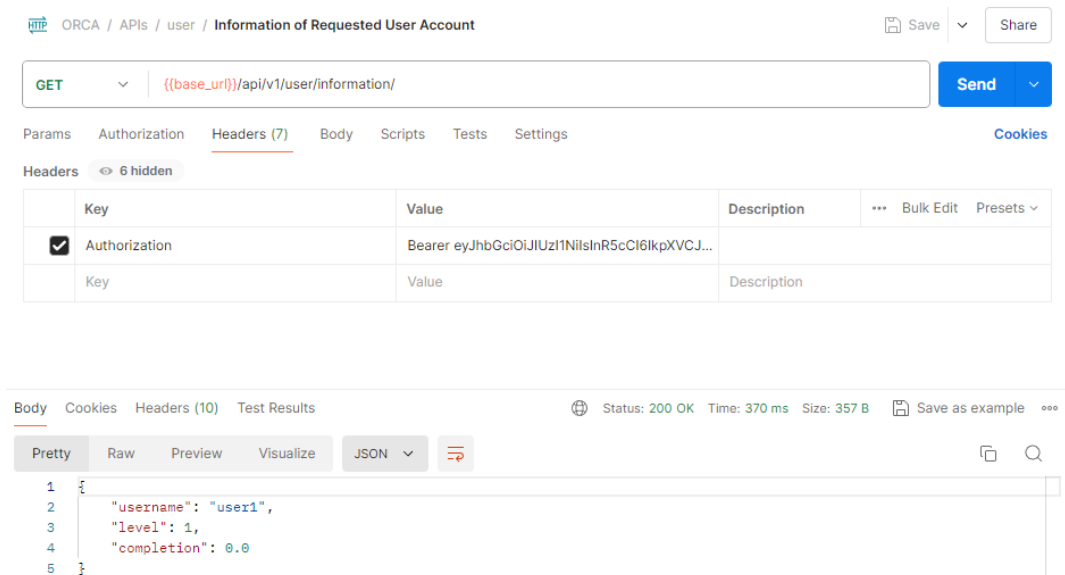
**Gambar 0.3 Response API Update Password User**

### 4. Admin Berhasil Menghapus Akun User



**Gambar 0.4 Response API Delete User**

## 5. User Berhasil Melihat Informasi Akunnya



**Gambar 0.5 Response API Get User Info**

### 5.2.1.2 Token APIs

Terdiri dari API yang berkaitan dengan JSON Web Token (JWT) untuk layanan otentikasi dan otorisasi yang digunakan dalam pengembangan ini. Berikut adalah daftar API *endpoint* terkait manajemen token.

**Tabel 0.3 List API Token**

<i>Category</i>	<i>Method</i>	<i>API Endpoint</i>
<i>Login</i>	<i>POST</i>	<code>{{base_url}}/api/v1/token/access/</code>



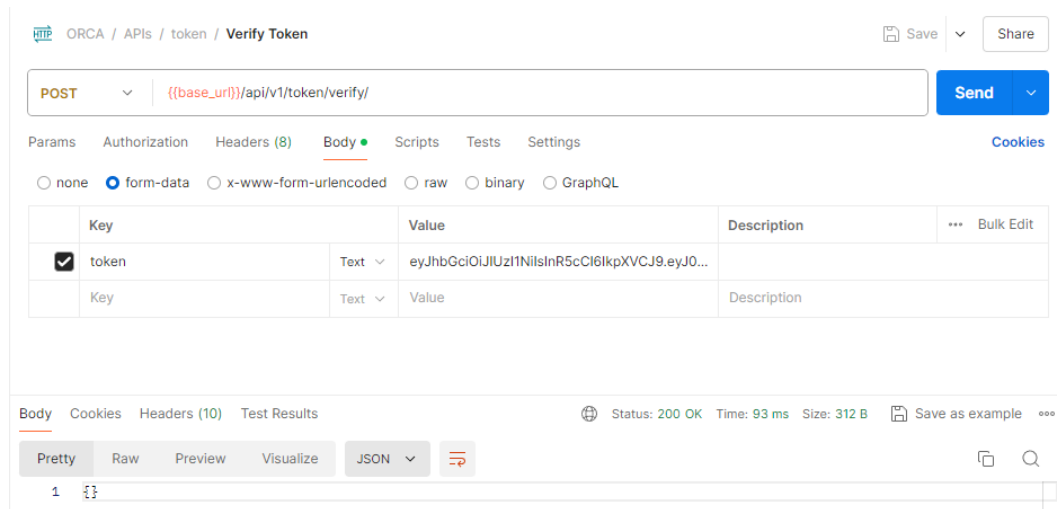




- [illegible]

### Gambar 0.8 Response API Refresh Token

4. Admin/User berhasil memverifikasi validitas token yang sedang digunakan untuk memutuskan apakah user masih memiliki otoritas mengakses *authenticated* API dengan input data yang diperlukan dari sisi user adalah *access* token.



**Gambar 0.9 Response API Verify Token**

### 5.2.1.3 Kubernetes APIs

API terkait kluster Kubernetes terdiri dari beberapa tipe *endpoint* yang memiliki pola dan fungsi yang mirip, sehingga daftar REST API akan dipersingkat hanya untuk tiap-tiap pola yang berbeda. Fungsi API ini meliputi manajemen *resources* di dalam kluster Kubernetes termasuk *deployment*, *Pods*, dan *namespace*. Selain itu, terdapat Websocket API yang digunakan untuk mengintegrasikan data-data penting yang bersifat *real-time*. Berikut adalah daftar API-nya.

**Tabel 0.4 List Kubernetes API**

<i>Category</i>	<i>Method</i>	<i>API Endpoint</i>
<i>Get Requested Pod List</i>	<i>GET</i>	<code>{{base_url}}/api/v1/kube/pods/</code>
<i>Get Requested Deployment List</i>	<i>GET</i>	<code>{{base_url}}/api/v1/kube/deployments/</code>
<i>Get Requested Pod Log</i>	<i>GET</i>	<code>{{base_url}}/api/v1/kube/pods/{{pods_name}}/logs/</code>
<i>Restart 5G Components</i>	<i>POST</i>	<code>{{base_url}}/api/v1/kube/restart_{{component_name}}/</code>
<i>Get Core (AMF &amp; UPF) Log</i>	<i>GET</i>	<code>{{base_url}}/api/v1/kube/get_amf_logs/</code> & <code>{{base_url}}/api/v1/kube/get_upf_logs/</code>

<i>Get Core (AMF &amp; UPF) Deployment Status</i>	<i>GET</i>	<i>{{base_url}}/api/v1/kube/get_amf_deployments/ &amp; {{base_url}}/api/v1/kube/get_upf_deployments/</i>
---	------------	--

**Tabel 0.5 List Kubernetes Websocket API**

<i>Caterogy</i>	<i>Endpoint</i>
<i>Ping and cURL command</i>	<i>ws://{{base_url}}/ws/shell/</i>
<i>Get UE Monitoring (Key Performance Indicator)</i>	<i>ws://{{base_url}}/ws/monitoring/</i>
<i>Get SCTP Protocol Information</i>	<i>ws://{{base_url}}/ws/protocolstack/</i>

Kedua tabel di atas berisi daftar REST API dan Websocket API *endpoint* yang tersedia terkait dengan manajemen kluster Kubernetes. Setiap API di atas memiliki fungsionalitasnya masing-masing, berikut akan disajikan proses testing seluruh fungsionalitas Kubernetes APIs yang dilakukan menggunakan aplikasi Postman untuk tipe REST API dan juga *code editor* dengan Django *libraries* untuk tipe Websocket API.

1. User berhasil mendapatkan daftar informasi pod Kubernetes miliknya. Informasi ini akan digunakan untuk manajemen pod yang dilakukan oleh user.

ORCA / APIs / kube / Get Requested Pods

GET `{{base_url}}/api/v1/kube/pods/` Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Headers 6 hidden

	Key	Value	Description	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ...				
	Key	Value	Description			

Body Cookies Headers (10) Test Results Status: 200 OK Time: 2.37 s Size: 479 B Save as example

Pretty Raw Preview Visualize JSON

```

1 {
2   "pods": [
3     {
4       "name": "oai-nr-ue-level1-user1-76b96bd65-dwdsh",
5       "ip": null,
6       "network_status": "Not available",
7       "state": "Pending",
8       "namespace": "user1",
9       "node": "node3"
10    }
  ]
}

```

**Gambar 0.10 API Response List Pod**

2. User berhasil mendapatkan daftar informasi deployment Kubernetes miliknya. Informasi ini akan digunakan untuk manajemen siklus hidup dari komponen 5G.

ORCA / APIs / kube / Get Requested Deployments

GET `127.0.0.1:8001/api/v1/kube/deployments/` Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Headers 6 hidden

	Key	Value	Description	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ...				
	Key	Value	Description			

Body Cookies Headers (10) Test Results Status: 200 OK Time: 531 ms Size: 942 B Save as example

Pretty Raw Preview Visualize JSON

```

1 [
2   {
3     "deployment_name": "oai-cu-level1-user1",
4     "namespace": "user1",
5     "replicas": 0,
6     "available_replicas": null,
7     "ready_replicas": null,
8     "updated_replicas": null,
9     "strategy": "Recreate",
10    "conditions": [

```

**Gambar 0.11 API Response List Deployment**

3. User berhasil mendapatkan informasi log dari pod Kubernetes miliknya. Log ini akan berguna ketika user akan melakukan konfigurasi pada komponen 5G.

ORCA / APIs / kube / Get Requested Pod's Logs

GET `{{base_url}}/api/v1/kube/pods/oai-nr-ue-level1-user1-ccd7864b7-gd2d2/logs/` Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Headers 6 hidden

Key	Value	Description
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ...	
Key	Value	Description

Body Cookies Headers (10) Test Results Status: 200 OK Time: 588 ms Size: 1.31 KB Save as example

Pretty Raw Preview Visualize JSON

```

1 [
2   {
3     "timestamp": "00:04:20",
4     "log": "1687914.367127 [HW] I connect() to 192.168.1.2:4043 failed, errno(113)"
5   },
6   {
7     "timestamp": "00:04:21",
8     "log": "1687915.367814 [HW] I Trying to connect to 192.168.1.2:4043"
9   },
10  ]

```

**Gambar 0.12 API Response Get Log Pod**

4. User berhasil me-*restart* deployment Kubernetes (komponen 5G). Informasi terkait deployment didapatkan dari daftar informasi deployment pada API sebelumnya.

ORCA / APIs / kube / Restart Single DU

POST `{{base_url}}/api/v1/kube/restart_single_du/` Send

Params Authorization Headers (8) Body Scripts Tests Settings Cookies

Headers 7 hidden

Key	Value	Description
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ...	
Key	Value	Description

Body Cookies Headers (10) Test Results Status: 200 OK Time: 666 ms Size: 420 B Save as example

Pretty Raw Preview Visualize JSON

```

1 {
2   "message": "Deployment restarted successfully.",
3   "details": "deployment.apps/oai-du-level1-user1 restarted\n"
4 }

```

**Gambar 0.13 API Response Restart Deployment**

5. User berhasil melakukan perintah ping dan melihat hasilnya. Perintah ini dijalankan menggunakan sebuah *script* pada Django untuk melakukan tes websocket.

```

test_shell.py M X
shell > test_shell.py
1 import asyncio
2 import websockets
3 import json
4
5 async def test_websocket():
6     uri = "ws://10.30.1.221:8002/ws/shell/"
7     async with websockets.connect(uri) as websocket:
8         # Send initial data to start the command
9         message = json.dumps({
10             'pod_name': 'oai-nr-ue-level1-user1-59f86bc6db-gx5fx', # Replace with your actual pod name
11             'namespace': 'user1', # Replace with your actual namespace
12             'command': 'ping google.com' # User-defined command
13         })
14         await websocket.send(message)
15
16 # Listen for messages from the server
17 while True:
18     # Receive data from the server
19     data = await websocket.recv()
20     print('Received: ', data)
21
22 if __name__ == '__main__':
23     asyncio.run(test_websocket())
24
25 (venv_orca_backend_ws) bagus@cs5g-backend:~/ORCA/orca_backend_ws/shell$ python test_shell.py
Received: {'command_output': 'PING google.com (172.253.118.101) 56(84) bytes of data.\n'}
Received: {'command_output': '64 bytes from sl-in-f101.1e100.net (172.253.118.101): icmp_seq=1 ttl=105 time=42.1 ms\n'}
Received: {'command_output': '64 bytes from sl-in-f101.1e100.net (172.253.118.101): icmp_seq=2 ttl=105 time=42.4 ms\n'}
Received: {'command_output': '64 bytes from sl-in-f101.1e100.net (172.253.118.101): icmp_seq=3 ttl=105 time=48.1 ms\n'}
Received: {'command_output': '64 bytes from sl-in-f101.1e100.net (172.253.118.101): icmp_seq=4 ttl=105 time=45.5 ms\n'}
Received: {'command_output': '\n'}
Received: {'command_output': '--- google.com ping statistics ---\n'}
Received: {'command_output': '4 packets transmitted, 4 received, 0% packet loss, time 3004ms\n'}
Received: {'command_output': 'rtt min/avg/max/mdev = 42.085/44.492/48.054/2.442 ms\n'}
^Z
[3]+  Stopped                  python test_shell.py
(venv_orca_backend_ws) bagus@cs5g-backend:~/ORCA/orca_backend_ws/shell$

```

**Gambar 0.14 Proses Testing API Websocket User PING**

6. User berhasil melakukan perintah cURL dan melihat hasilnya. Perintah ini dijalankan menggunakan sebuah *script* pada Django untuk melakukan tes websocket.

```

test_shell.py M X
shell > test_shell.py
5 async def test_websocket():
6     uri = "ws://10.30.1.221:8002/ws/shell/"
7     async with websockets.connect(uri) as websocket:
8         message = json.dumps({
9             'pod_name': 'oai-nr-ue-level1-user1-59f86bc6db-gx5fx', # Replace with your actual pod name
10            'namespace': 'user1', # Replace with your actual namespace
11            'command': 'curl google.com' # User-defined command
12        })
13        await websocket.send(message)
14
15 # Listen for messages from the server
16 while True:
17     # Receive data from the server
18     data = await websocket.recv()
19     print('Received: ', data)
20
21 if __name__ == '__main__':
22     asyncio.run(test_websocket())
23
24 (venv_orca_backend_ws) bagus@cs5g-backend:~/ORCA/orca_backend_ws/shell$ python test_shell.py
Received: {'command_output': '<HTML><HEAD><meta http-equiv="content-type" content="text/html; charset=utf-8">\n'}
Received: {'command_output': '<TITLE>301 Moved</TITLE></HEAD><BODY>\n'}
Received: {'command_output': '<H1>301 Moved</H1>\n'}
Received: {'command_output': 'The document has moved\n'}
Received: {'command_output': '<A HREF="http://www.google.com/">here</A>.\n'}
Received: {'command_output': '</BODY></HTML>\n'}
^Z
[4]+  Stopped                  python test_shell.py
(venv_orca_backend_ws) bagus@cs5g-backend:~/ORCA/orca_backend_ws/shell$

```

**Gambar 0.15 Proses Testing API Websocket User CURL**

7. User berhasil mendapatkan informasi Key Performance Indicator (KPI) dari komponen UE. Perintah ini dijalankan menggunakan sebuah *script* pada Django untuk melakukan tes websocket.







HTTP ORCA / APIs / kube / **Get AMF Log** Save Share

GET ▼ {{base\_url}}/api/v1/kube/get\_amf\_logs/ Send ▼

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Headers 6 hidden

	Key	Value	Description	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ...				
	Key	Value	Description			

Body Cookies Headers (10) Test Results Status: 200 OK Time: 856 ms Size: 1.4 KB Save as example ...

Pretty Raw Preview Visualize JSON ▼ ≡

```

1 [
2   {
3     "timestamp": "00:12:46",
4     "log": "* getaddrinfo() thread failed to start"
5   },
6   {
7     "timestamp": "00:12:46",
8     "log": "* Could not resolve host: oai-nrf"
9   },
10  ]

```

Gambar 0.18 Response API AMF Log

HTTP ORCA / APIs / kube / **Get UPF Log** Save Share

GET ▼ {{base\_url}}/api/v1/kube/get\_upf\_logs/ Send ▼

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Headers 6 hidden

	Key	Value	Description	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ...				
	Key	Value	Description			

Body Cookies Headers (10) Test Results Status: 200 OK Time: 604 ms Size: 1.35 KB Save as example ...

Pretty Raw Preview Visualize JSON ▼ ≡

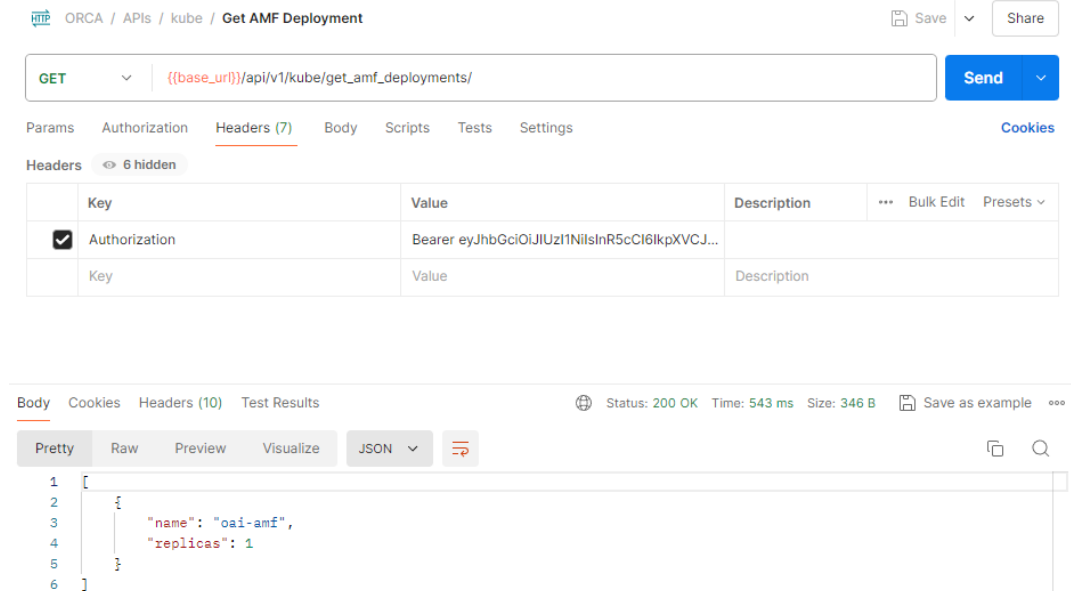
```

1 [
2   {
3     "timestamp": "13:40:53",
4     "log": "[2024-06-04 06:40:53.994] [upf_n4] [info] handle_receive(16 bytes)"
5   },
6   {
7     "timestamp": "13:40:53",
8     "log": "[2024-06-04 06:40:53.994] [upf_n4] [info] Received SX HEARTBEAT REQUEST"
9   },
10  ]

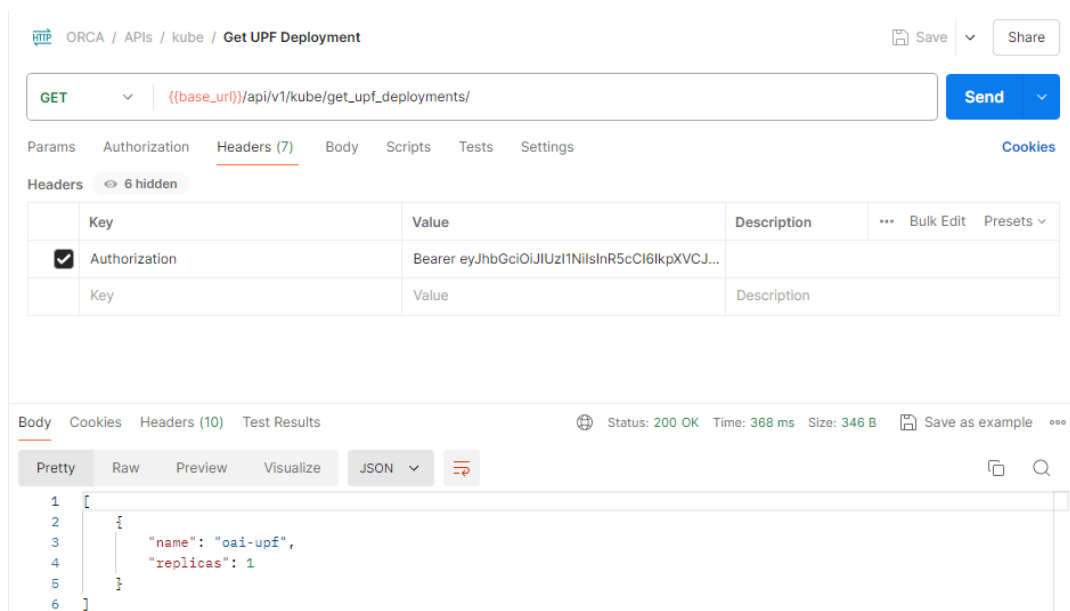
```

Gambar 0.19 Response API UPF Log

10. User Berhasil Mendapatkan Informasi Status Deployment AMF dan UPF. Informasi ini digunakan untuk mengindikasikan tersedia atau tidaknya komponen Core.



**Gambar 0.20 Response API AMF Deployment**



**Gambar 0.21 Response API UPF Deployment**

#### 5.2.1.4 Helm Chart APIs

API yang berkaitan dengan Helm Chart memiliki peran utama untuk instalasi seluruh komponen 5G yang diperlukan oleh *user* ke dalam kluster Kubernetes berdasarkan *namespace* per masing-masing *user*. Selain daripada itu, fungsi API ini juga berkaitan dengan manajemen *values* yang terdapat pada setiap komponen 5G yang berhasil terinstal. Dan fungsi terakhirnya adalah termasuk untuk manajemen siklus hidup dari setiap

komponen. Seperti halnya dengan Kubernetes APIs sebelumnya, API ini juga memiliki berbagai pola dan fungsi yang mirip. Berikut adalah daftar API terkait dengan Helm Chart.

**Tabel 0.6 List Helm Chart API**

<i>Category</i>	<i>Method</i>	<i>API Endpoint</i>
<i>Get Configuration Values</i>	<i>GET</i>	<i>{{base_url}}/api/v1/oai/values_{{component_name}}/</i>
<i>Config Component Values</i>	<i>POST</i>	<i>{{base_url}}/api/v1/oai/config_{{component_name}}/</i>
<i>Start Component</i>	<i>POST</i>	<i>{{base_url}}/api/v1/oai/start_{{component_name}}/</i>
<i>Stop Component</i>	<i>POST</i>	<i>{{base_url}}/api/v1/oai/stop_{{component_name}}/</i>

API yang berkaitan dengan Helm Chart memiliki peran utama untuk instalasi seluruh komponen 5G yang diperlukan oleh *user* ke dalam kluster Kubernetes berdasarkan *namespace* per masing-masing *user*.

Selain daripada itu, fungsi API ini juga berkaitan dengan manajemen *values* yang terdapat pada setiap komponen 5G yang berhasil terinstal. Dan fungsi terakhirnya adalah termasuk untuk manajemen siklus hidup dari setiap komponen. Seperti halnya dengan Kubernetes APIs sebelumnya, API ini juga memiliki berbagai pola dan fungsi yang mirip. Berikut adalah daftar API terkait dengan Helm Chart.

1. User berhasil mendapatkan informasi nilai konfigurasi pada setiap komponen 5G. Informasi ini digunakan untuk memvalidasi benar atau tidaknya hasil konfigurasi yang dilakukan oleh user.

HTTP ORCA / APIs / oai / Values Single CU

GET `{{base_url}}/api/v1/oai/values_single_cu/` Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Headers 6 hidden

Key	Value	Description
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ...	
Key	Value	Description

Body Cookies Headers (10) Test Results Status: 200 OK Time: 315 ms Size: 520 B Save as example

Pretty Raw Preview Visualize JSON Copy Search

```

1 {
2   "values": {
3     "cuId": "",
4     "cellId": "",
5     "f1InterfaceIPadd": "",
6     "f1cuPort": "2152",
7     "f1duPort": "2152",
8     "n2InterfaceIPadd": "",
9     "n3InterfaceIPadd": "",
10    "mcc": "",
  
```

**Gambar 0.22 Response API Get Configuration Value**

2. User berhasil mengubah nilai konfigurasi pada setiap komponen 5G. Nilai ini akan berpengaruh terhadap keberhasilan konfigurasi setiap komponen.

HTTP ORCA / APIs / oai / Config Single CU

POST `{{base_url}}/api/v1/oai/config_single_cu/` Send

Params Authorization Headers (9) Body Scripts Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

```

1 {"f1_cuport": "2152", "f1_duport": "2152"}
  
```

Body Cookies Headers (10) Test Results Status: 200 OK Time: 2.37 s Size: 359 B Save as example

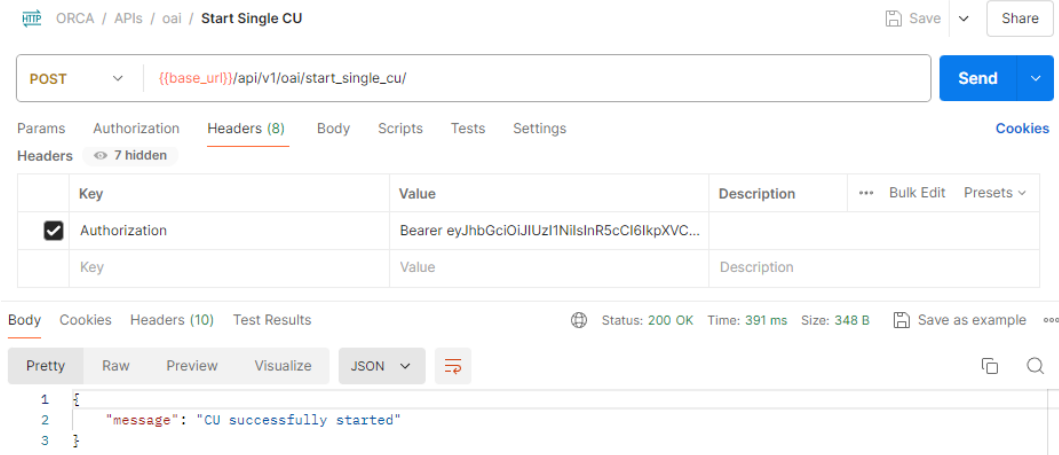
Pretty Raw Preview Visualize JSON Copy Search

```

1 {
2   "message": "Configuration Updated Successfully"
  
```

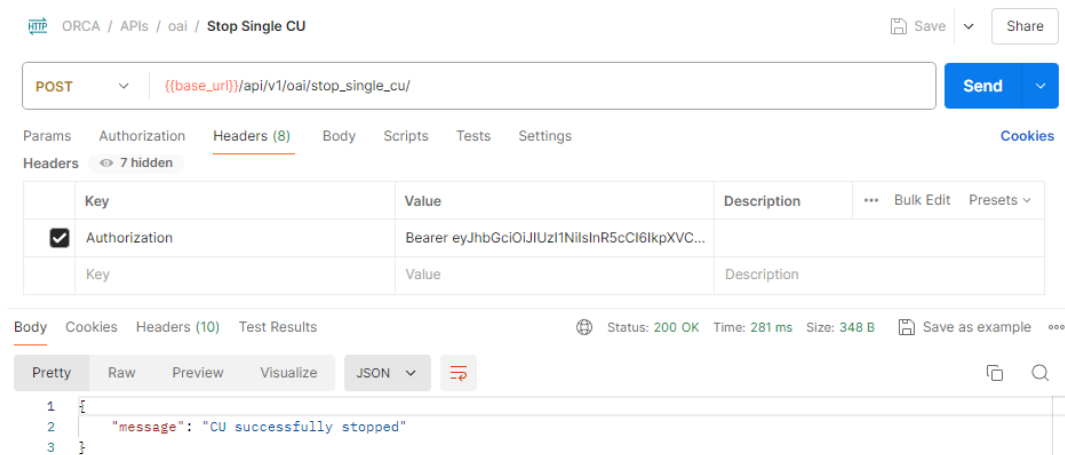
**Gambar 0.23 Response API Post Configuration Value**

3. User berhasil menjalankan setiap komponen 5G (Deployment Kubernetes). Ini berkaitan dengan manajemen siklus hidup dari komponen 5G.



**Gambar 0.24 Response API Start Component**

4. User berhasil menghentikan setiap komponen 5G (Deployment Kubernetes). Ini berkaitan dengan manajemen siklus hidup dari komponen 5G.



**Gambar 0.25 Response API Stop Component**

### 5.2.1.5 Wireshark APIs

Terakhir ini merupakan REST API yang berkaitan dengan fungsi *sniffing* layaknya aplikasi Wireshark. API ini akan membantu *user* dalam *sniffing* paket data pada setiap komponen 5G dan menyimpan riwayat *sniffing* ke dalam sebuah *file pcap* yang disimpan di dalam *database*. Pada API ini menggunakan tipe Websocket sebagai API utama yang digunakan untuk mengintegrasikan data-data penting yang bersifat *real-time* yang dalam kasus ini adalah data hasil proses *sniffing* paket data. Berikut adalah daftar API terkait fungsi *sniffing*.

Tabel 0.7 List Wireshark API

<i>Category</i>	<i>Method</i>	<i>API Endpoint</i>
<i>List PCAP Files</i>	<i>GET</i>	<i>{{base_url}}/api/v1/shark/pcap_files/</i>
<i>Download PCAP Files</i>	<i>GET</i>	<i>{{base_url}}/api/v1/shark/pcap_files/{{files_id}}/download/</i>
<i>Delete PCAP Files</i>	<i>DELETE</i>	<i>{{base_url}}/api/v1/shark/pcap_files/{{files_id}}/remove/</i>

Tabel 0.8 List Wireshark Websocket API

<i>Category</i>	<i>Endpoint</i>
<i>Sniffing Process and Save PCAP Files into Database</i>	<i>ws://{{base_url}}/ws/sniff/</i>

Kedua tabel di atas berisi daftar REST API dan Websocket API *endpoint* yang tersedia terkait dengan fungsionalitas *sniffing*. Setiap API di atas memiliki fungsionalitasnya masing-masing, berikut akan disajikan proses testing seluruh fungsionalitas Wireshark APIs yang dilakukan menggunakan aplikasi Postman untuk tipe REST API dan juga *code editor* dengan Django *libraries* untuk tipe Websocket API.

1. User berhasil mendapatkan informasi paket data dari hasil *sniffing* pada pod (komponen 5G) dan berhasil menyimpan *file* PCAP pada *database*.

```

test_sniff.py M x
sniff > test_sniff.py
4
5 async def test_websocket():
6     uri = "ws://10.30.1.221:8002/ws/sniff/"
7     async with websockets.connect(uri) as websocket:
8         # Send initial data to start the tcpdump command
9         message = json.dumps({
10             'pod_name': 'oai-cu-level1-user1-cb4b4dcb6-kdw1j', # Replace with your actual pod name
11             'namespace': 'user1', # Replace with your actual namespace
12             'interface': 'n2' # Uncomment and replace with actual interface if needed (e.g., 'f1', 'n2', 'n3', 'oa
13         })
14
15 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS EXPOSED PORTS
16 (venv_orca_backend_ws) bagus@cs5g-backend:~/ORCA/orca_backend_ws/sniff$ python test_sniff.py
17 1 0.000000000 192.168.1.1 -> 192.168.1.2 SCTP 106 HEARTBEAT
18 ^Z
19 [8]+ Stopped python test_sniff.py
20 (venv_orca_backend_ws) bagus@cs5g-backend:~/ORCA/orca_backend_ws/sniff$ python test_sniff.py
21 1 0.000000000 172.21.6.1 -> 172.21.6.94 SCTP 106 HEARTBEAT
22 ^Z
23 [9]+ Stopped python test_sniff.py
24 (venv_orca_backend_ws) bagus@cs5g-backend:~/ORCA/orca_backend_ws/sniff$

```

**Gambar 0.26 Proses Testing API Websocket Sniffing Pod**

2. User berhasil mendapatkan informasi daftar *file* PCAP yang telah tersedia pada *database*.

ORCA / APIs / shark / List Pcap Files

GET {{base\_url}}/api/v1/shark/pcap\_files/ Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Headers 6 hidden

Key	Value	Description	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVC...		
Key	Value	Description		

Body Cookies Headers (10) Test Results Status: 200 OK Time: 274 ms Size: 648 B Save as example

Pretty Raw Preview Visualize JSON

```

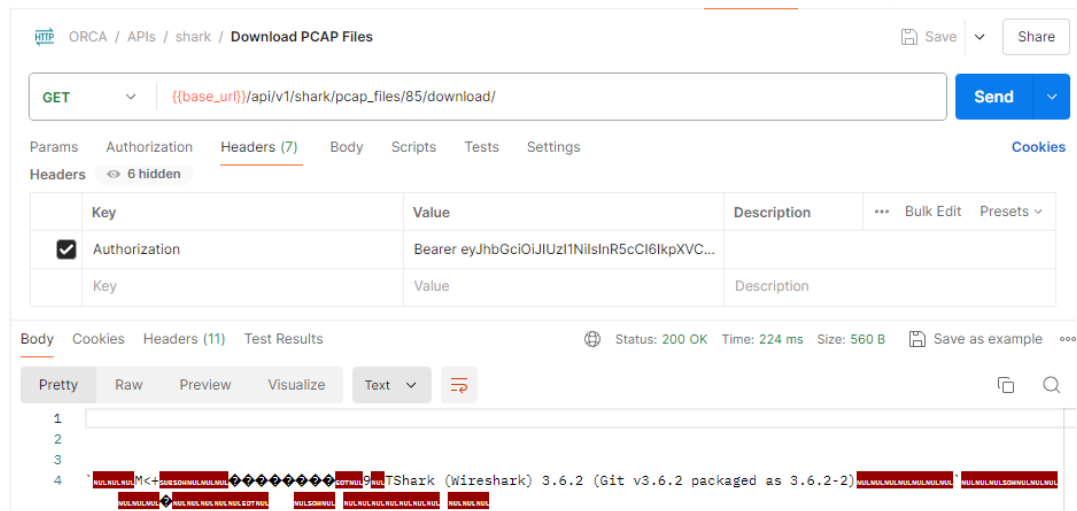
1 [
2   {
3     "id": 85,
4     "user": "user1",
5     "filename": "user1_oai-du-level1-user1-764665b6c9-p5nrg_20240623_172812.pcap",
6     "file_size": 128,
7     "created_at": "2024-06-24T00:28:12.744195+07:00"
8   },
9   {
10    "id": 86,
11    "user": "user1",
12    "filename": "user1_oai-nr-ue-level1-user1-cdd7864b7-gd2d2_20240623_172845.pcap",
13    "file_size": 13628,
14    "created_at": "2024-06-24T00:29:03.165935+07:00"
15  }
16 ]

```

**Gambar 0.27 Response API List PCAP File**

3. User berhasil men-*download* *file* PCAP yang diinginkan dari *database*







pengujian unit atau integrasi untuk dapat diperbaiki. Hal ini membantu memastikan aplikasi memenuhi semua persyaratan bisnis dan teknis, serta mengurangi risiko terkait rilis produk ke pengguna akhir. Berikut adalah tabel yang mendeskripsikan alur dari *E2E testing* yang akan dilakukan.

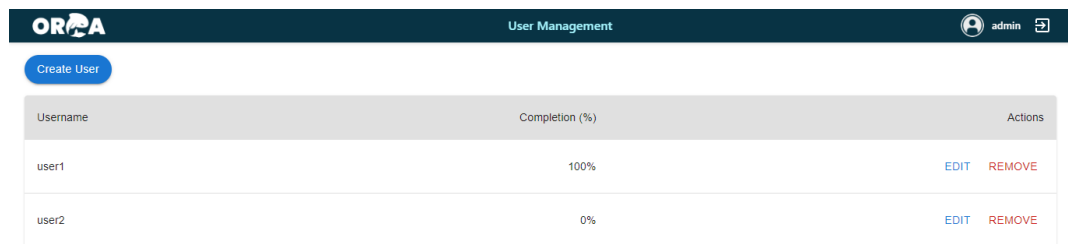
**Tabel 0.9 Langkah *E2E Testing***

Langkah	Deskripsi
Setup Proyek	Siapkan lingkungan pengujian yang mencakup URL akses aplikasi, kredensial login, dan data pengujian yang diperlukan.
Penyusunan Rencana Pengujian	Buat rencana pengujian yang mencakup skenario E2E, seperti membuka halaman, mengisi formulir, dan mengklik tombol, serta hasil yang diharapkan untuk setiap langkah.
Menjalankan Tes Manual	Ikuti rencana pengujian secara manual dengan menjalankan langkah-langkah yang telah ditentukan pada UI aplikasi di browser, mencatat setiap tindakan dan hasil yang diamati.
Pengamatan Hasil	Catat hasil pengujian untuk melihat apakah ada kegagalan atau kesalahan yang terjadi selama pengujian, termasuk tangkapan layar jika diperlukan.
Debugging	Jika ditemukan masalah, analisis dan perbaiki masalah tersebut, kemudian ulangi pengujian untuk memastikan perbaikan.
Dokumentasi Tes	Dokumentasikan hasil pengujian secara rinci, termasuk langkah-langkah yang dilakukan, hasil yang diamati, dan masalah yang ditemukan serta perbaikannya.

Pada tahap ini, proses *E2E testing* yang dilakukan akan memiliki pola berulang sama dengan sebelumnya pada *functional testing*. Hanya saja, kali ini proses *testing* akan dilakukan melalui browser Google Chrome dengan menggunakan tampilan UI yang telah dikembangkan sebagai sisi *frontend*. Berikut adalah poin-poin yang akan dijabarkan dalam bentuk fungsionalitas mewakili setiap APIs yang telah dijelaskan pada *functional testing*.

### 5.2.2.1 User Management Functions

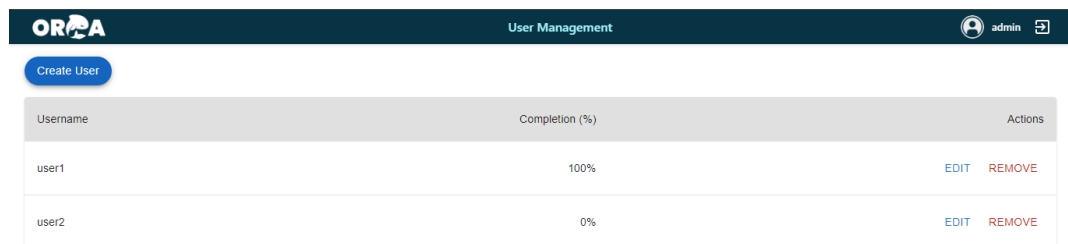
1. Admin berhasil membuat akun user baru



Username	Completion (%)	Actions
user1	100%	<a href="#">EDIT</a> <a href="#">REMOVE</a>
user2	0%	<a href="#">EDIT</a> <a href="#">REMOVE</a>

**Gambar 0.30 Tampilan Admin Membuat User**

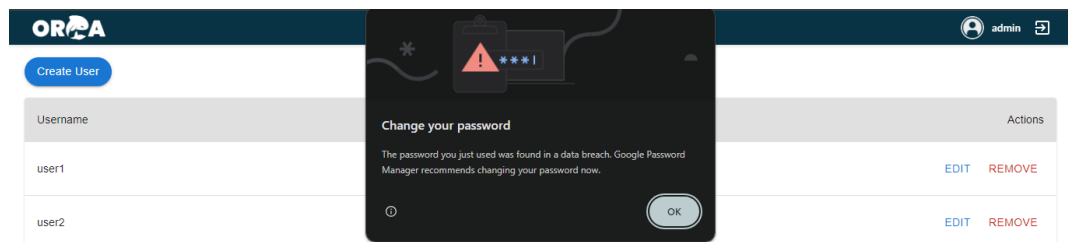
2. Admin berhasil melihat daftar akun user



Username	Completion (%)	Actions
user1	100%	<a href="#">EDIT</a> <a href="#">REMOVE</a>
user2	0%	<a href="#">EDIT</a> <a href="#">REMOVE</a>

**Gambar 0.31 Tampilan Admin Melihat User**

3. Admin berhasil mengubah password akun user

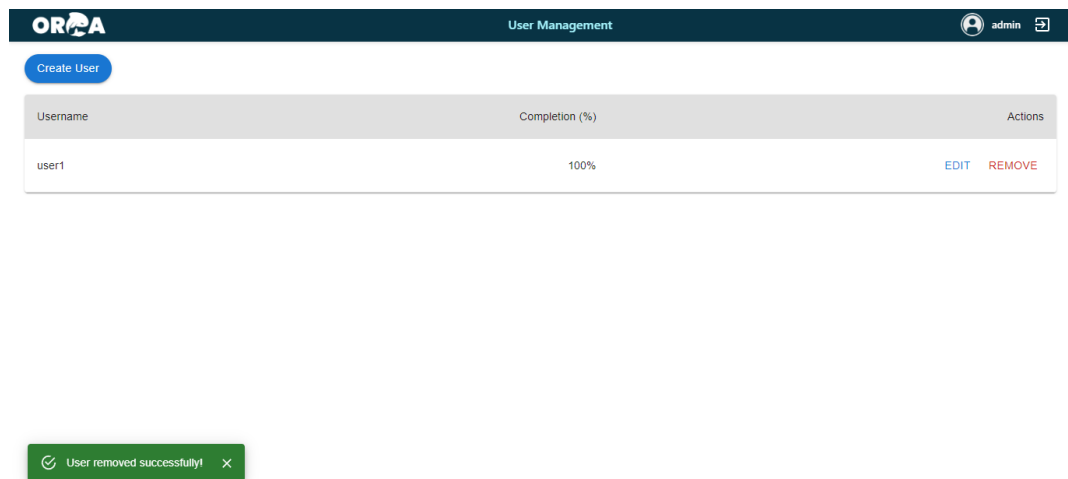


Username	Completion (%)	Actions
user1		<a href="#">EDIT</a> <a href="#">REMOVE</a>
user2		<a href="#">EDIT</a> <a href="#">REMOVE</a>

✔ User updated successfully! ✕

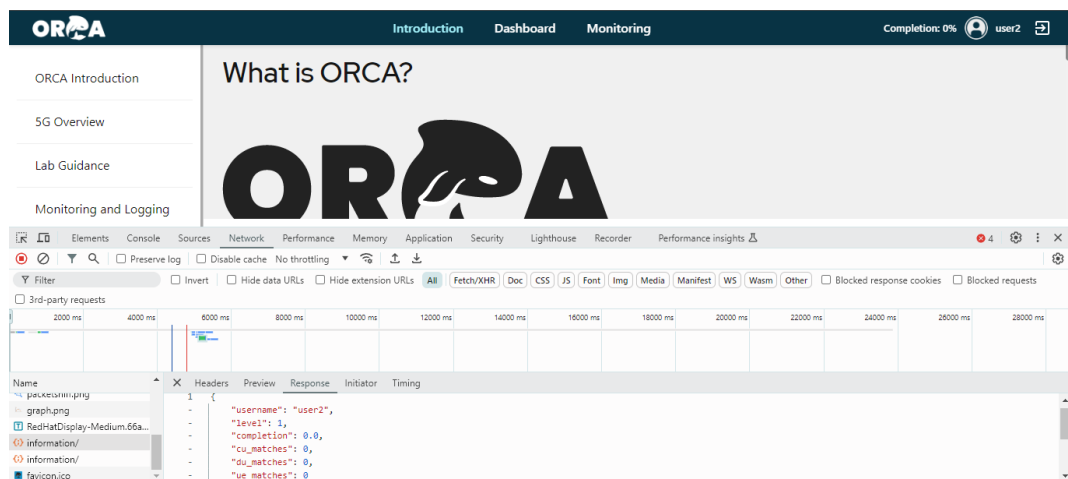
**Gambar 0.32 Tampilan Admin Mengubah Password User**

4. Admin berhasil menghapus akun user



**Gambar 0.33 Tampilan Admin Menghapus User**

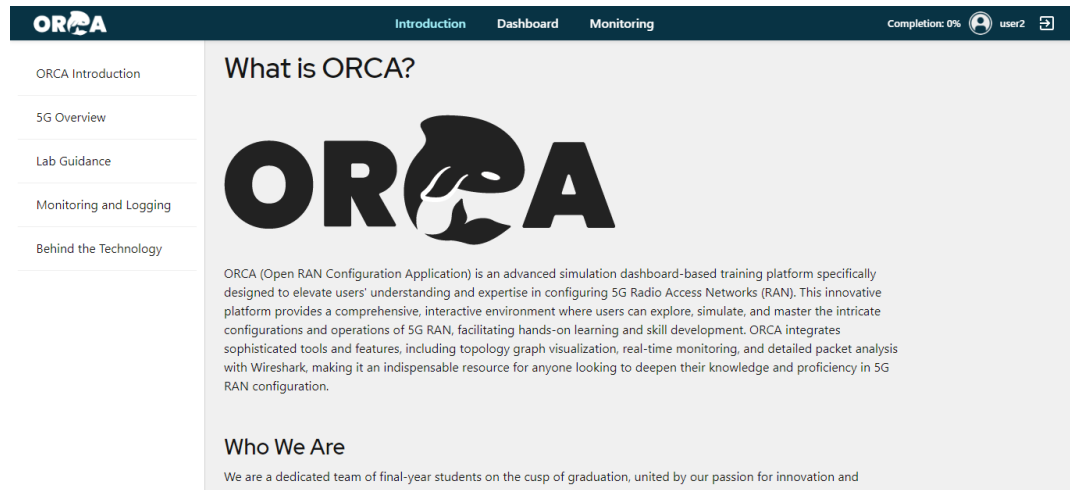
5. User berhasil melihat informasi akunnya



**Gambar 0.34 Verifikasi Informasi User dari Browser**

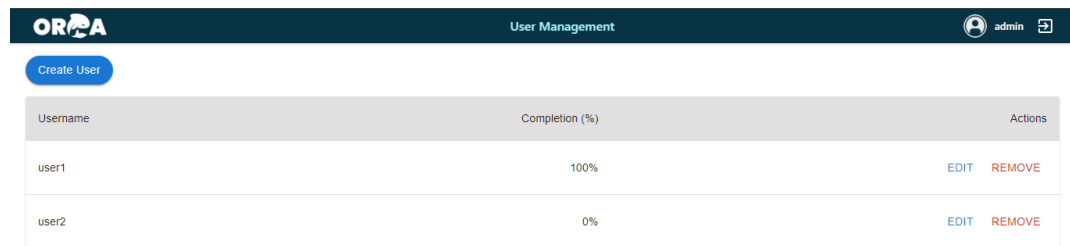
### 5.2.2.2 Token Functions

1. User berhasil masuk ke dalam halaman introduction website (*login*)



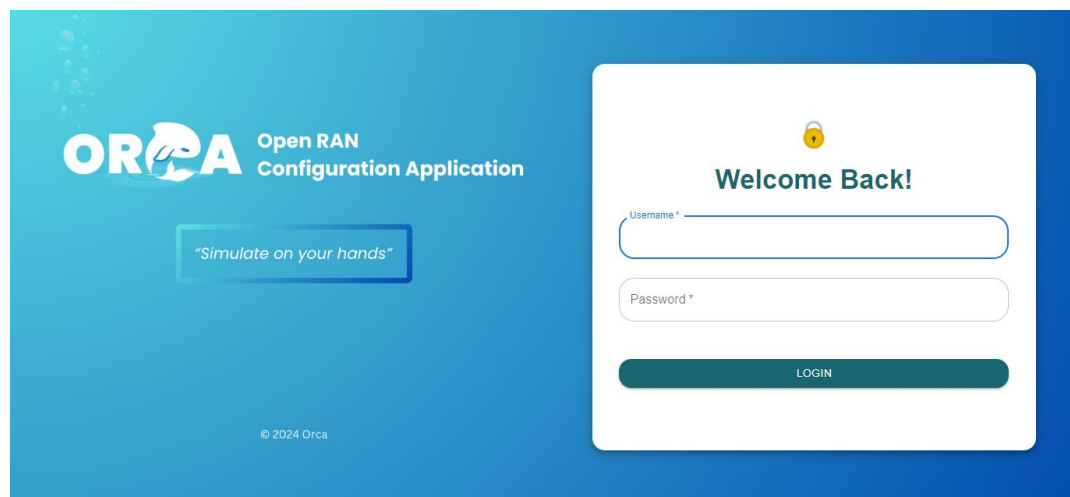
**Gambar 0.35 Verifikasi Informasi User Berhasil Login**

2. Admin berhasil masuk ke dalam halaman user management (*login*)



**Gambar 0.36 Verifikasi Informasi Admin Berhasil Login**

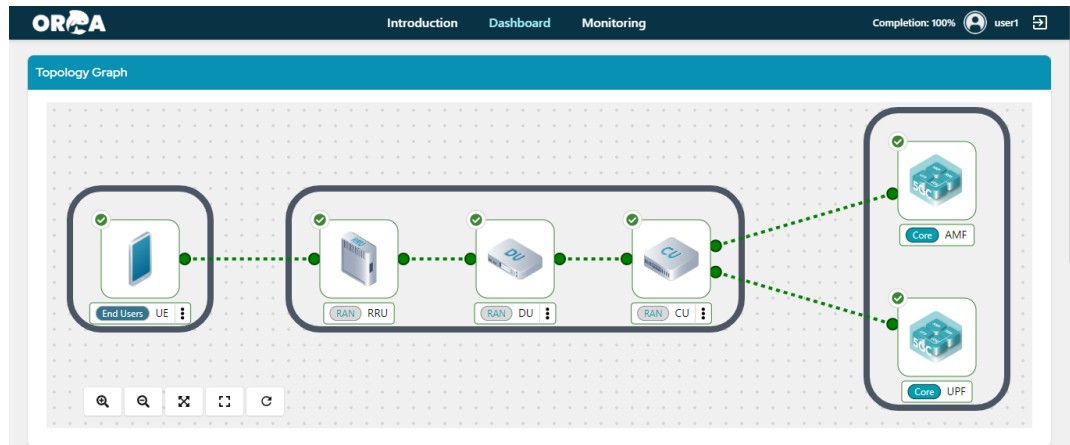
3. Admin/User berhasil keluar dari website dan kembali ke halaman *login* (*logout*)



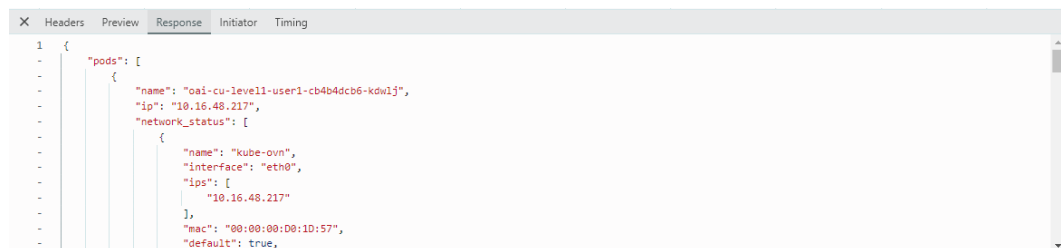
**Gambar 0.37 Verifikasi User dan Admin Berhasil Logout**

### 5.2.2.3 Kubernetes Functions

1. User berhasil mendapatkan daftar informasi pod Kubernetes miliknya ditandai dengan munculnya gambar topologi pada laman dashboard dengan status komponen stopped/running (frame berwarna kuning/hijau).

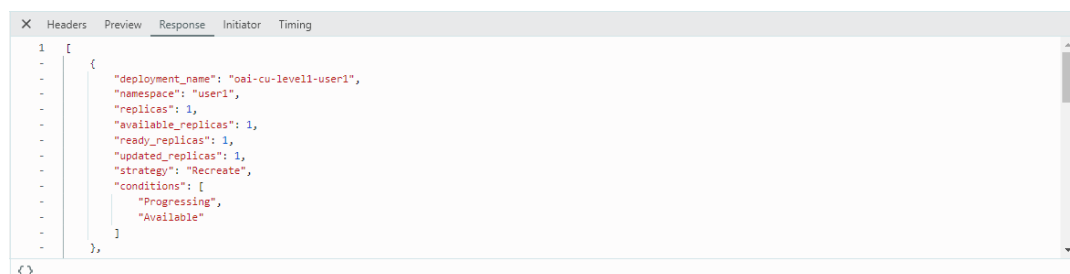


**Gambar 0.38 Koneksi Topologi Terhubung**



**Gambar 0.39 Response API Pod Saat Topologi Terhubung**

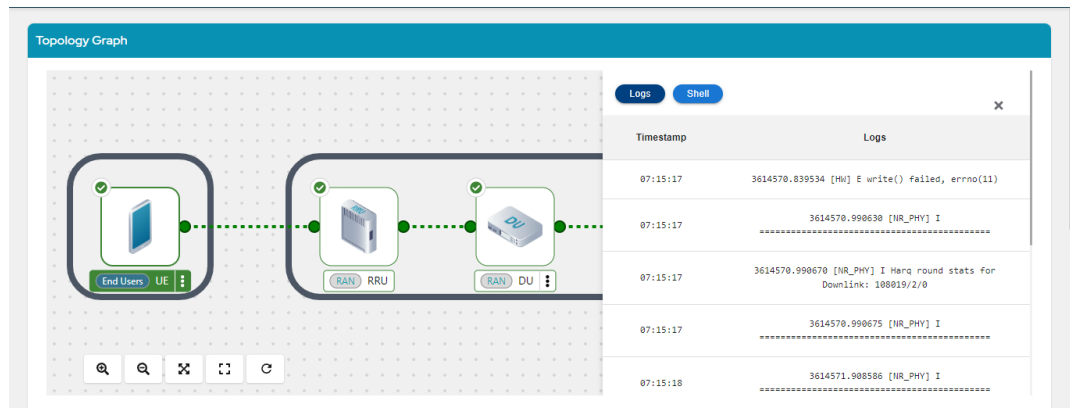
2. User berhasil mendapatkan daftar informasi deployment Kubernetes miliknya ditandai dengan munculnya gambar topologi pada laman dashboard dengan status komponen stopped/running (frame berwarna kuning/hijau).



**Gambar 0.40 Response API Deployment Saat Topologi Terhubung**

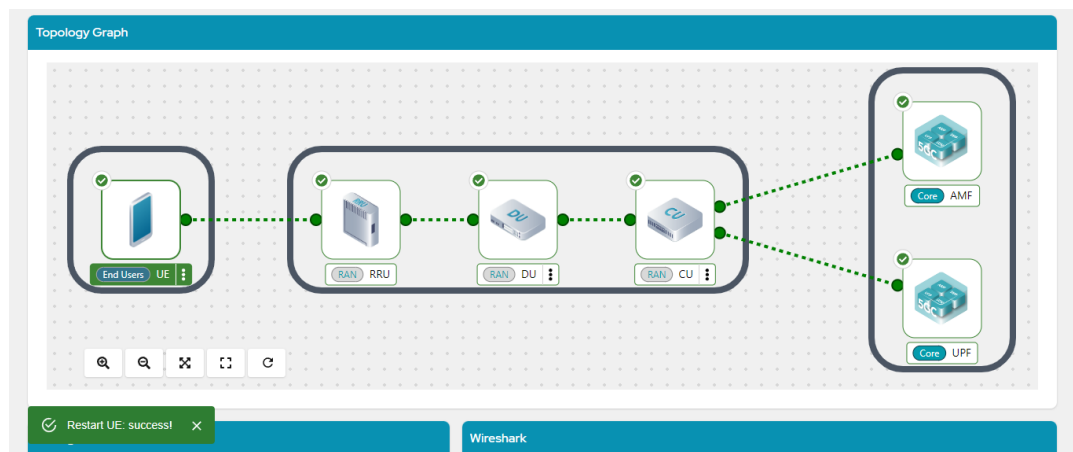
3. User berhasil mendapatkan informasi log dari pod Kubernetes miliknya ditandai dengan munculnya informasi log pada setiap komponen ketika menekan ikon

komponen dan mengarahkan pada tab Logs yang terdapat pada sidebar yang muncul ketika ikon komponen ditekan.



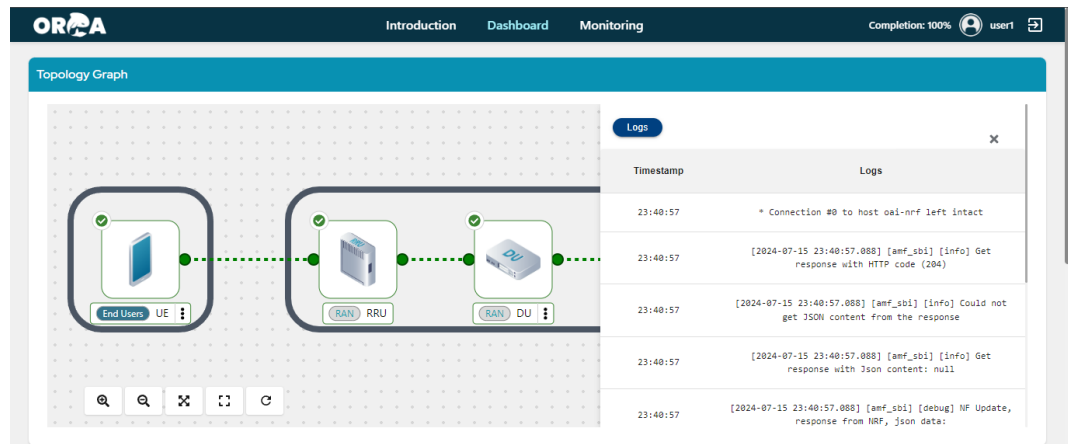
**Gambar 0.41 UE Berhasil Menampilkan Log**

4. User berhasil *me-restart* deployment Kubernetes (komponen 5G) sebagai manajemen siklus hidup komponen, ditandai dengan notifikasi sukses proses restart yang dilakukan pada setiap komponen. Proses restart dapat dilakukan dengan menekan kebab button pada setiap komponen ataupun dengan klik kanan pada mouse ataupun touchpad untuk memunculkan pop up pilihan tombol restart.



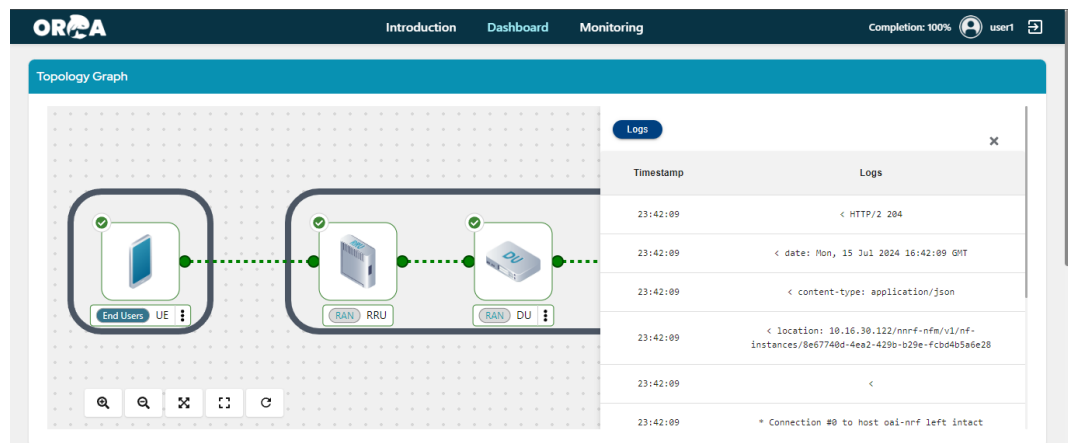
**Gambar 0.42 UE Berhasil Di Restart**

5. User berhasil mendapatkan informasi log dari komponen AMF ditandai dengan munculnya informasi log pada komponen AMF ketika menekan ikon komponen dan mengarahkan pada tab Logs yang terdapat pada sidebar yang muncul ketika ikon komponen ditekan.



**Gambar 0.43 AMF Berhasil Menampilkan Log**

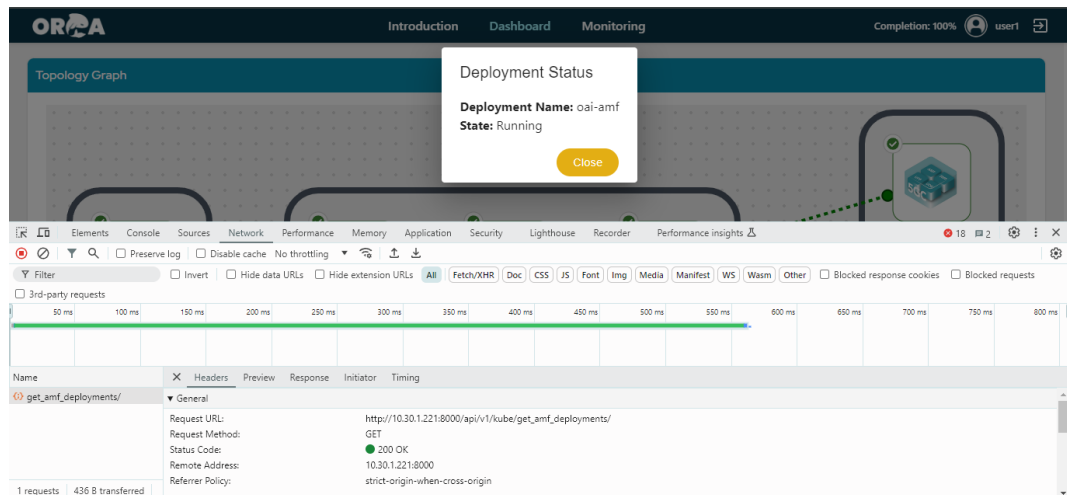
6. User berhasil mendapatkan informasi log dari komponen UPF ditandai dengan munculnya informasi log pada komponen UPF ketika menekan ikon komponen dan mengarahkan pada tab Logs yang terdapat pada sidebar yang muncul ketika ikon komponen ditekan.



**Gambar 0.44 UPF Berhasil Menampilkan Log**

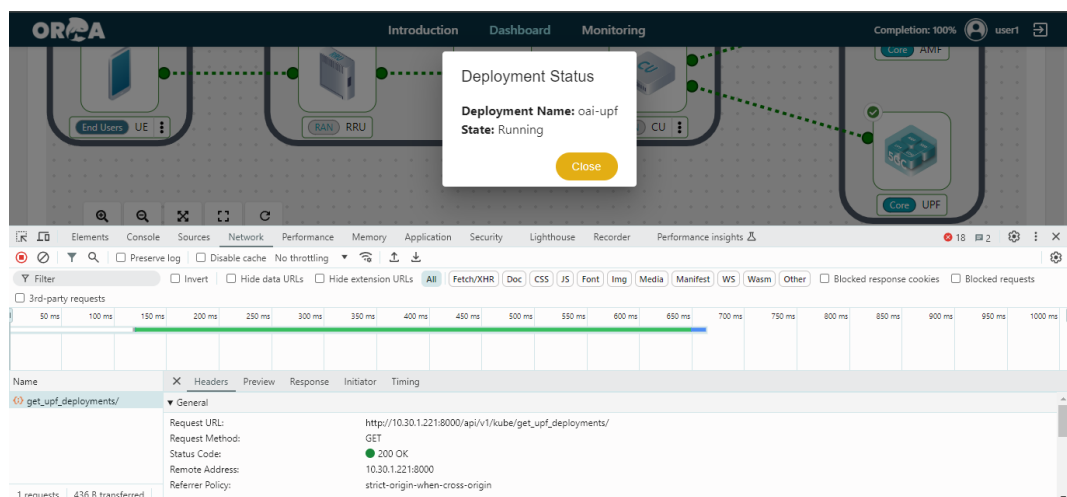
7. User berhasil mendapatkan informasi data deployment komponen AMF ditandai dengan munculnya informasi deployment status pada komponen AMF ketika menekan ikon checklist atau silang yang terdapat pada frame komponen AMF.





**Gambar 0.45 AMF Berhasil Menampilkan State Pod**

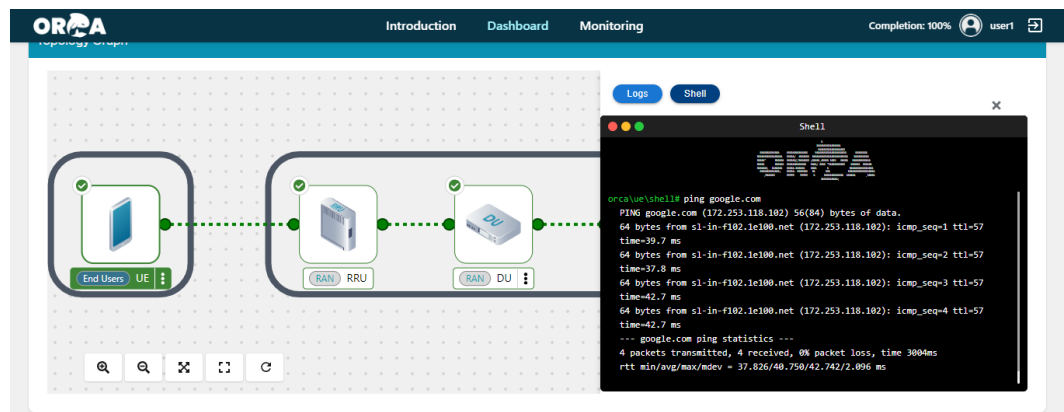
8. User berhasil mendapatkan informasi data deployment komponen UPF ditandai dengan munculnya informasi deployment status pada komponen UPF ketika menekan ikon checklist atau silang yang terdapat pada frame komponen UPF.



**Gambar 0.46 UPF Berhasil Menampilkan State Pod**

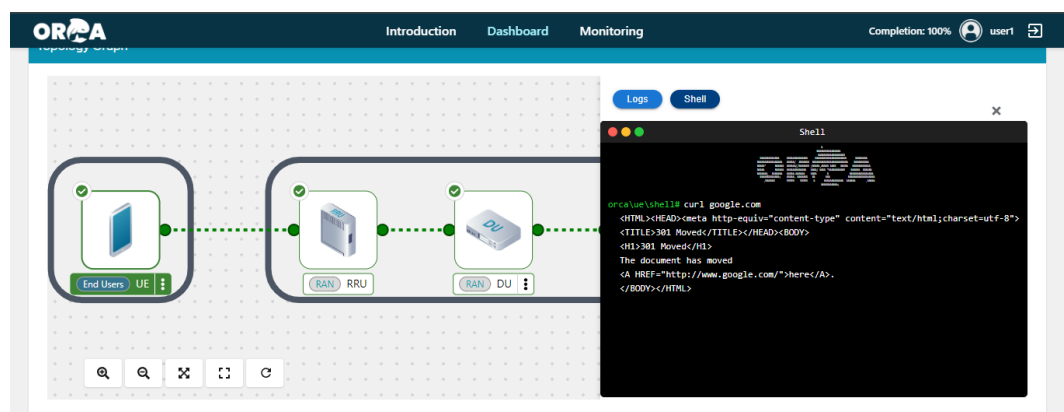
9. User berhasil melakukan perintah ping dan melihat hasilnya pada komponen UE. Proses ping dapat dilakukan dengan menekan ikon komponen UE, kemudian mengarahkan ke tab Shell pada sidebar yang muncul setelah ikon komponen ditekan. Seperti contoh di bawah, perintah ping google.com menghasilkan output yang menandakan proses ping berjalan dengan baik dan benar. Proses ping ini bertujuan untuk mengecek status koneksi *end-to-end* (E2E) dari komponen UE hingga ke komponen Core dan ke internet berjalan dengan baik atau tidak.





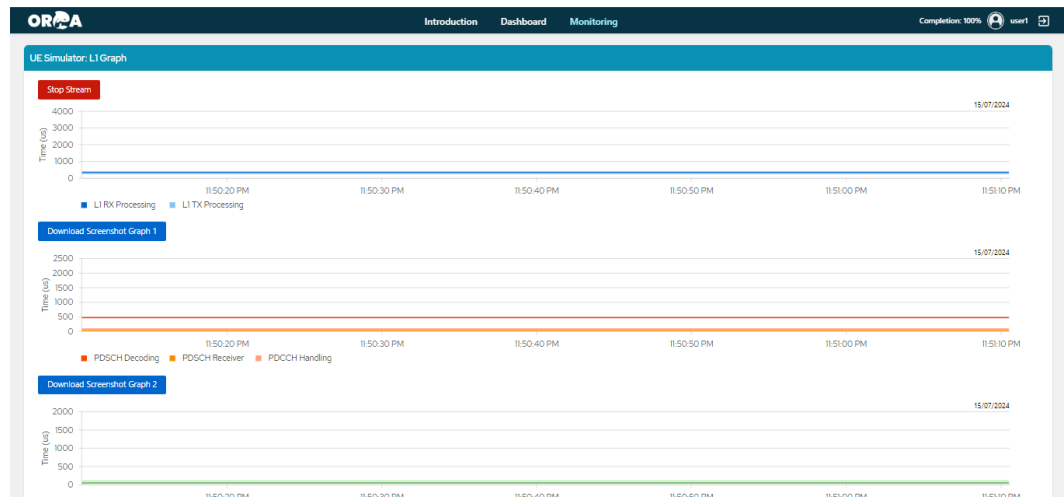
**Gambar 0.47 UE Berhasil Menampilkan Hasil Ping**

10. User berhasil melakukan perintah cURL dan melihat hasilnya pada komponen UE. Proses cURL dapat dilakukan dengan menekan ikon komponen UE, kemudian mengarahkan ke tab Shell pada sidebar yang muncul setelah ikon komponen ditekan. Seperti contoh di bawah, perintah curl google.com menghasilkan output yang menandakan proses cURL berjalan dengan baik dan benar. Proses cURL ini bertujuan untuk mensimulasikan smartphone (UE) dapat mengakses laman website pada internet dengan cara yang simple melalui command-line interface (CLI).



**Gambar 0.48 UE Berhasil Menampilkan Hasil CURL**

11. User berhasil mendapatkan informasi *Key Performance Indicator* (KPI) dari komponen UE, ditandai dengan munculnya nilai-nilai KPI pada table value beserta dengan munculnya grafik monitoring yang bekerja secara real-time untuk memvisualisasikan nilai-nilai KPI tersebut. Informasi ini bisa didapatkan di laman monitoring pada website dashboard.

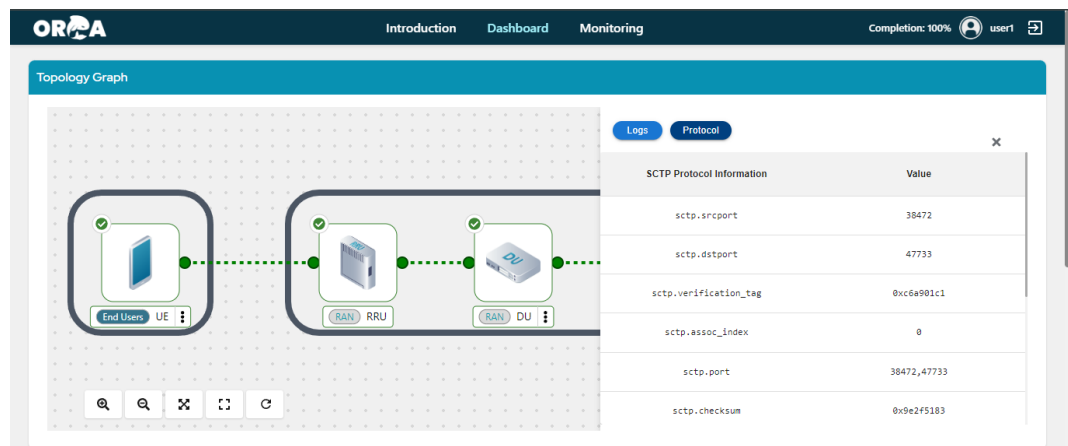


**Gambar 0.49 User Berhasil Mendapatkan Hasil Monitoring**

Table Value			
Key Performance Indicator	Value (us)	Count	Total Time (us)
L1 TX processing	272.929	3287403	1863.028
ULSCH encoding	48.802	54870	464.716
L1 RX processing	355.457	8706141	3509.706
UL Indication	98.995	3287403	802.546
PDSCH receiver	40.317	54851	833.121
PDSCH decoding	474.975	54850	2956.438
-> Deinterleave	4.837	54851	175.097
-> Rate Unmatch	9.718	54851	1156.099
-> LDPC Decode	294.934	54851	1964.035
PDSCH unscrambling	3.656	54851	75.794

**Gambar 0.50 User Berhasil Mendapatkan Key Table Value**

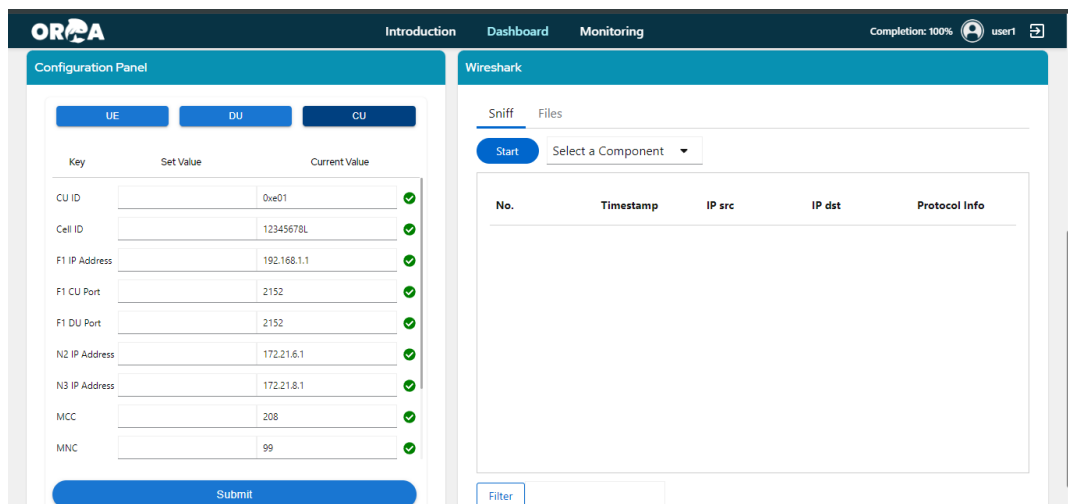
12. User berhasil mendapatkan informasi protokol SCTP dari komponen RAN 5G (CU dan DU), ditandai dengan munculnya informasi terkait protocol SCTP ketika menekan ikon komponen CU ataupun DU dan mengarahkan pada tab Protocol yang terdapat pada sidebar yang muncul ketika ikon komponen ditekan.



**Gambar 0.51 User Berhasil Mendapatkan Hasil Protocol SCTP**

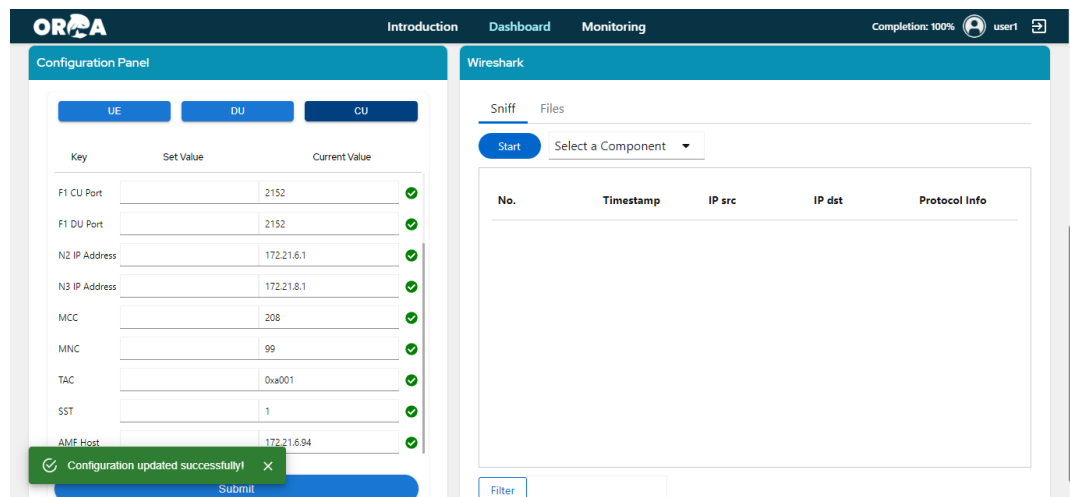
#### 5.2.2.4 Helm Chart Functions

1. User berhasil mendapatkan informasi value konfigurasi pada setiap komponen 5G (CU, DU, dan UE) yang terletak di menu Configuration Panel pada lama dashboard. Hal tersebut ditandai dengan munculnya value pada kolom Current Value setelah user melakukan konfigurasi pada kolom Set Value dan menekan tombol submit untuk mengkonfirmasi perubahan value pada komponen yang dikonfigurasi.



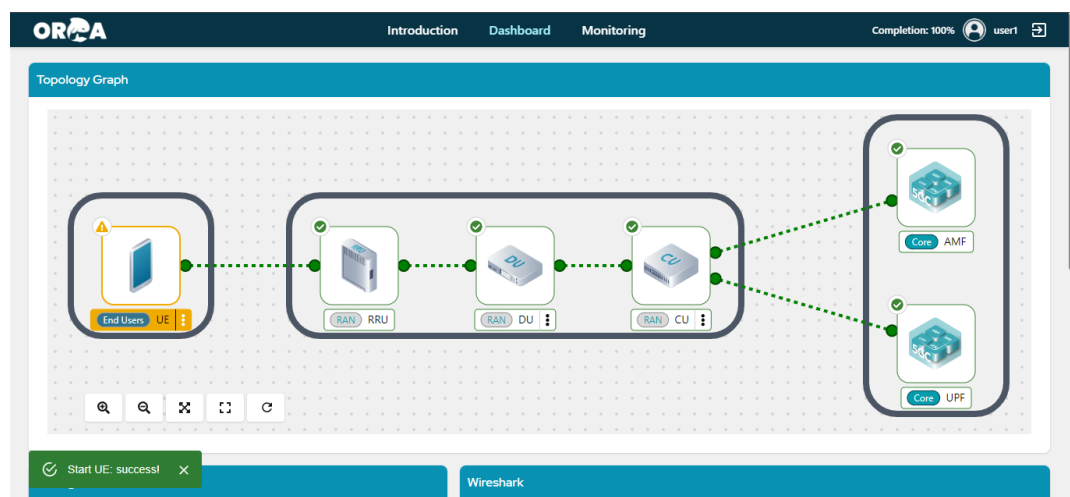
**Gambar 0.52 User Berhasil Mendapatkan Hasil Value Config**

2. User berhasil mengubah value konfigurasi pada setiap komponen 5G (CU, DU, dan UE) yang terletak di menu Configuration Panel pada lama dashboard. Hal tersebut ditandai dengan munculnya notifikasi status berhasil konfigurasi sesaat setelah user mengkonfirmasi perubahan value pada komponen yang dikonfigurasi.



**Gambar 0.53 User Berhasil Mengubah Hasil Value Config**

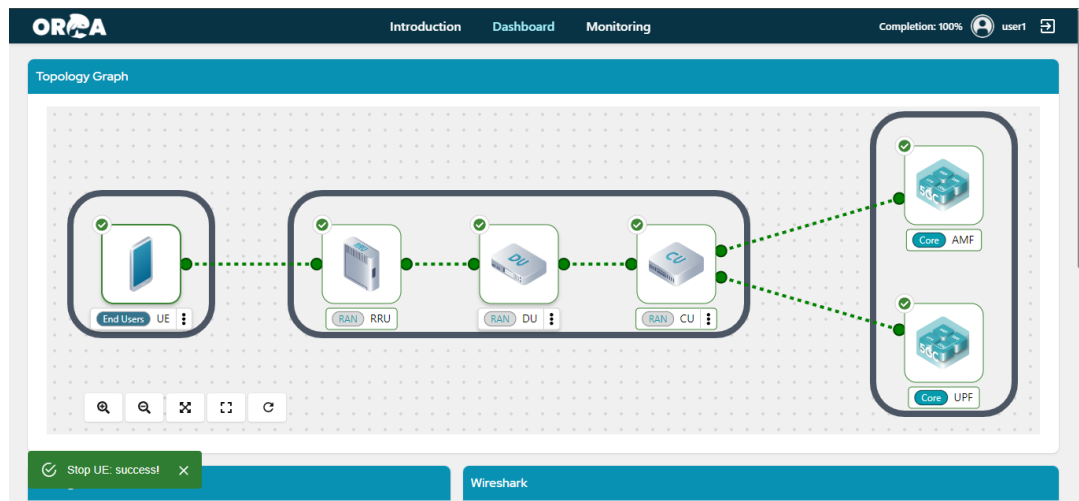
3. User berhasil menjalankan setiap komponen 5G (CU, DU, dan UE) pada manajemen siklus hidup komponen. Hal tersebut ditandai dengan notifikasi sukses pada proses start yang dilakukan pada setiap komponen beserta berubahnya warna frame menjadi hijau pada komponen tersebut sesaat setelah notifikasi sukses menghilang. Proses ini dapat dilakukan dengan menekan kebab button pada setiap komponen ataupun dengan klik kanan pada mouse ataupun touchpad untuk memunculkan pop up pilihan tombol start.



**Gambar 0.54 User Berhasil Menghubah State UE**

4. User berhasil menghentikan setiap komponen 5G (CU, DU, dan UE) pada manajemen siklus hidup komponen. Hal tersebut ditandai dengan notifikasi sukses pada proses stop yang dilakukan pada setiap komponen beserta berubahnya warna frame menjadi kuning pada komponen tersebut sesaat setelah notifikasi sukses

menghilang. Proses ini dapat dilakukan dengan menekan kebab button pada setiap komponen ataupun dengan klik kanan pada mouse ataupun touchpad untuk memunculkan pop up pilihan tombol stop.



**Gambar 0.55 User Berhasil Menghentikan UE**

#### 5.2.2.5 Wireshark Functions

1. User berhasil mendapatkan informasi paket data dari hasil *sniffing* pada setiap komponen CU, DU, dan UE serta berhasil menyimpan *file* PCAP pada *database*. Hal tersebut ditandai dengan munculnya informasi paket data hasil sniffing di tab Sniff pada menu Wireshark yang terdapat pada laman dashboard, tepat disebelah menu Configuration Panel. Informasi yang ditampilkan meliputi nomor, timestamp, IP src, IP dst, dan protocol info.

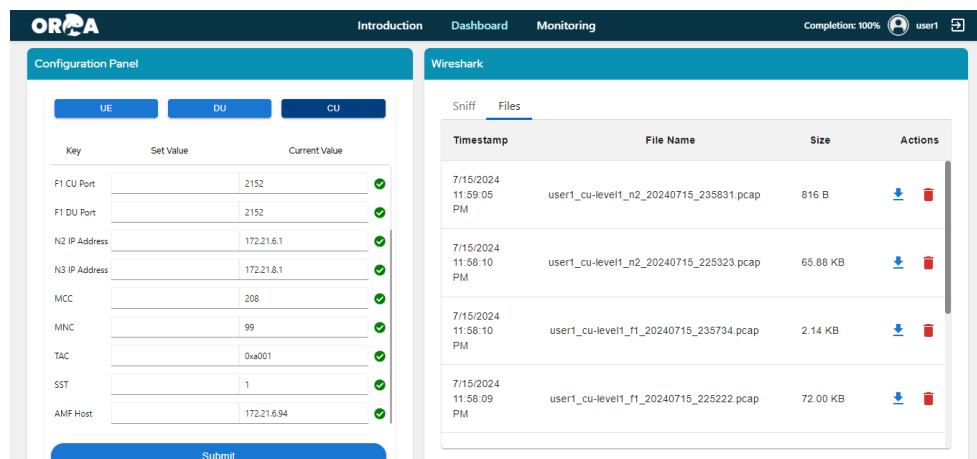
No.	Timestamp	IP src	IP dst	Protocol Info
13	6.008903318	12.1.1.3	172.253.118.100	ICMP 84 Echo (ping) request id=0x9580, seq=7/1792, ttl=64
14	6.054319579	172.253.118.100	12.1.1.3	ICMP 84 Echo (ping) reply id=0x9580, seq=7/1792, ttl=105 (request in 13)
15	7.010556931	12.1.1.3	172.253.118.100	ICMP 84 Echo (ping) request id=0x9580, seq=8/2048, ttl=64
16	7.051404914	172.253.118.100	12.1.1.3	ICMP 84 Echo (ping) reply id=0x9580, seq=8/2048, ttl=105 (request in 15)
17	8.011636594	12.1.1.3	172.253.118.100	ICMP 84 Echo (ping) request id=0x9580, seq=9/2304, ttl=64
				ICMP 84 Echo (ping) reply id=0x9580, seq=9/2304, ttl=105 (request in 17)

**Gambar 0.56 User Berhasil Mendapatkan Hasil Sniffing UE**

Selain itu, notifikasi sukses akan muncul ketika file PCAP hasil dari proses sniffing yang telah dilakukan berhasil disimpan pada database sesaat setelah tombol stop

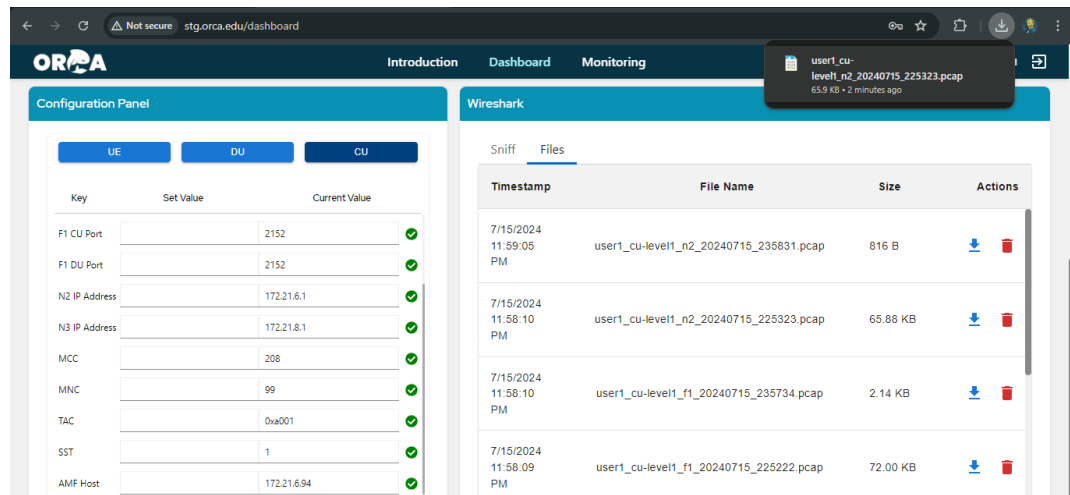
sniffing ditekan. File PCAP ini berisi informasi paket data hasil sniffing namun dalam bentuk yang lebih rinci sehingga user dapat melakukan olah data yang lebih rinci melalui file PCAP tersebut yang dapat dibuka menggunakan aplikasi Wireshark.

2. User berhasil mendapatkan informasi daftar *file* PCAP yang telah tersedia pada *database*. Hal tersebut ditandai dengan munculnya daftar file PCAP di tab Files pada menu Wireshark yang terdapat pada laman dashboard. Beberapa informasi terkait file tersebut seperti contohnya kapan file tersebut dibuat, nama file, ukuran file, dan juga menu actions yang dapat digunakan untuk mendownload ataupun menghapus file PCAP yang diinginkan dari database.



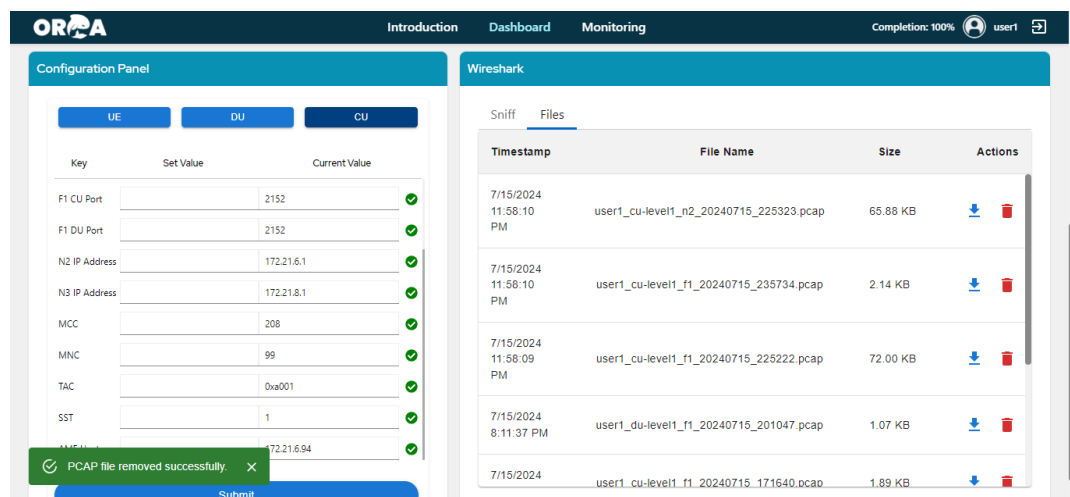
**Gambar 0.57 User Berhasil Mendapatkan Hasil File Sniffing**

3. User berhasil men-*download* *file* PCAP yang diinginkan. Hal tersebut ditandai dengan munculnya pop up pada browser setelah menekan tombol download. File PCAP dapat disimpan pada device user dan dapat digunakan untuk keperluan olah data yang lebih rinci.



**Gambar 0.58 User Berhasil Mengunduh Hasil File Sniffing**

4. User berhasil menghapus *file* PCAP yang diinginkan dari *database*. Hal tersebut ditandai dengan munculnya notifikasi sukses sesaat setelah tombol konfirmasi penghapusan file PCAP yang diinginkan ditekan.



**Gambar 0.59 User Berhasil Menghapus Hasil File Sniffing**

### 5.2.3 Performance Testing

Pengujian *performance testing* adalah langkah penting dalam mengukur dan menilai batas kapasitas keseluruhan aplikasi serta performanya di bawah *load* tinggi. Pengujian ini mencakup pengujian *system load capacity* dan koneksi E2E yang dirancang untuk mengidentifikasi batas maksimum resource, serta memastikan aplikasi tetap stabil dan responsif dalam penggunaan.



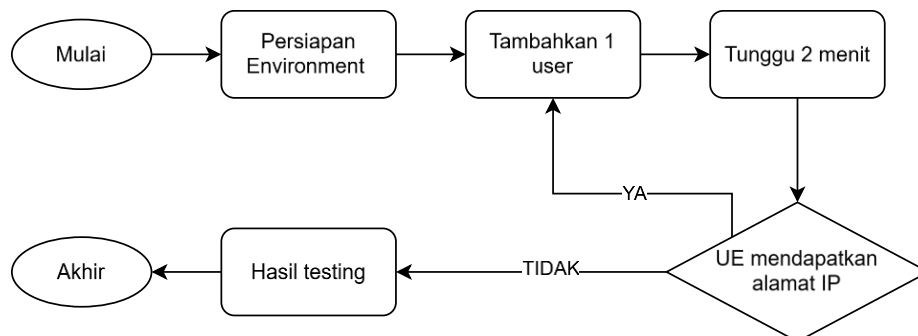
**Tabel 0.10 Langkah *Performance Testing***

Langkah	Deskripsi
Setup Testing Environment	Mempersiapkan lingkungan testing dan memastikan semua komponen klaster berjalan dan siap digunakan.
Inisialisasi Alat Pengujian	Menyiapkan alat pengujian seperti Grafana untuk memonitor dan menguji beban serta kinerja aplikasi.
Pengujian Utilisasi Resource User	Menguji bagaimana aplikasi menggunakan resource selama kondisi normal dan di bawah beban tinggi untuk mengidentifikasi penggunaan rata-rata dan potensi bottleneck.
Pengujian Kapasitas Infra	Menilai batas maksimum kapasitas infrastruktur aplikasi dan memastikan aplikasi tetap responsif saat berada di load maksimum.

Pengujian ini menggunakan helm untuk menguji beban dan kinerja aplikasi, serta Grafana untuk visualisasi dan pemantauan performa. akan membantu memonitor penggunaan CPU dan memori dari berbagai komponen aplikasi secara real-time dan Grafana akan menampilkan data performa secara mendetail.

#### 5.2.3.1 *System Load Capacity*

Pengujian kapasitas beban sistem bertujuan untuk mengidentifikasi batas maksimum jumlah user yang dapat ditampung oleh aplikasi berdasarkan hasil dari pengujian utilisasi sumber daya. Skenario pengujian menggunakan metode *Ramp Up Time* yang meliputi pengujian batas kapasitas maksimum pengguna yang dapat ditampung oleh aplikasi dan pengamatan perubahan beban pada sistem ketika aplikasi digunakan oleh jumlah pengguna yang meningkat hingga mencapai batas maksimum.

**Gambar 0.60 Diagram Alir Pengujian *System Load Capacity***



Gambar 5.60 memperlihatkan diagram alir yang digunakan dalam proses *system load capacity*. Pada inisiasi awal, persiapan *environment* meliputi persiapan konfigurasi beberapa *user* dan memastikan tidak ada *resource* lain yang berjalan pada saat pengujian.

NAME↑	PF	READY	STATUS
oai-cu-level1-user1-cb4b4dcb6-kdwlj	●	2/2	Running
oai-du-level1-user1-79fc6bc764-qkw5s	●	2/2	Running
oai-nr-ue-level1-user1-6d868fcd4-b7xnj	●	2/2	Running

**Gambar 0.61 Daftar Pod dan Status Pod per User**

Selanjutnya satu akun *user* ditambahkan dengan konfigurasi lengkap dan pada Gambar 5.61 memperlihatkan setiap akun *user* memiliki 3 komponen 5G RAN (CU, DU, dan UE) yang harus dipastikan sudah dalam state *Running*. Setelah 2 menit dapat memeriksa kembali apakah akun *user* dapat terkoneksi 5G secara E2E. Jika iya, maka akun *user* baru akan ditambahkan dan proses ini terus berjalan hingga *user* terbaru menunjukkan penurunan performa yang ditandai dengan tidak terkoneksi secara E2E.

**Tabel 0.11 Pengujian System Load Capacity**

Menit ke-	Utilisasi CPU	Utilisasi RAM	Keterangan
0	7,2%	23,9%	Idle sistem
1	18,2%	28,5%	Koneksi E2E berhasil pada user ke 1
2	27,9%	32,6%	Koneksi E2E berhasil pada user ke 2
3	37,2%	36,6%	Koneksi E2E berhasil pada user ke 3
4	47,1%	40,6%	Koneksi E2E berhasil pada user ke 4
5	54,4%	44,5%	Koneksi E2E berhasil pada user ke 5
6	62,4%	48,8%	Koneksi E2E berhasil pada user ke 6
7	71,2%	52,7%	Koneksi E2E berhasil pada user ke 7
8	80,2%	56,5%	Koneksi E2E berhasil pada user ke 8
9	86,5%	60,4%	Koneksi E2E berhasil pada user ke 9
10	96,1%	64,9%	Koneksi E2E berhasil pada user ke 10
11	null	null	Koneksi E2E tidak berhasil pada user ke 11

Tabel 5.11 memperlihatkan hasil dari system load capacity dengan metode Ramp Up testing yaitu dengan peningkatan 1 user setiap 2 menit. Pada tabel dapat dilihat total akun user yang telah dibuat berjumlah 11 akun dengan maksimum utilisasi CPU 100% dan utilisasi RAM 75,70%.

Kemudian, pengukuran penggunaan resource per user juga dilakukan untuk mengetahui perkiraan pemakaian resource yang digunakan oleh satu user. Pengukuran dilakukan menggunakan tools K9S dengan parameter pengukuran yang diambil adalah CPU dan RAM.

NAME↑	PF	READY	STATUS	RESTARTS	CPU	MEM
oai-cu-level1-user1-cb4b4dcb6-kdwlj	●	2/2	Running	0	1	116
oai-du-level1-user1-79fc6bc764-qkw5s	●	2/2	Running	0	1641	1493
oai-nr-ue-level1-user1-6d868fcd4-b7xnj	●	2/2	Running	0	1252	632

**Gambar 0.62 Monitoring Resource pada User**

Hasil screenshot menggunakan *tool* k9s pada Gambar 5.62, terdapat pod CU, DU, dan juga UE yang sedang dalam state *Running*. Pod CU menggunakan 1mCPU dan 116MiB RAM, pod DU menggunakan 1641mCPU dan 1493MiB RAM, sedangkan pod UE menggunakan 1252mCPU dan 632MiB RAM. Data tersebut diambil beberapa kali sehingga dapat ditentukan total penggunaan *resource* per satu *user* yang ditampilkan pada tabel 5.12.

**Tabel 0.12 Alokasi CPU dan RAM Satu User**

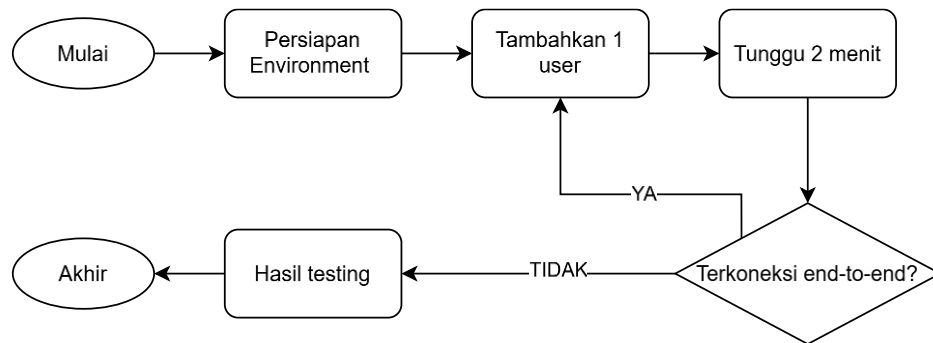
Komponen	CPU (mCPU)	RAM (MiB)
CU	1	119
DU	1569	1304
UE	1211	542
<b>Jumlah</b>	<b>2781</b>	<b>1965</b>

Berdasarkan tabel 5.12 yang menunjukkan penggunaan sumber daya CPU dan RAM, satu akun *user* memerlukan sekitar 2781mCPU dan 1965MiB RAM. Penggunaan sumber daya ini dihitung setelah semua konfigurasi komponen benar dan UE berhasil terhubung ke 5G core. Data ini penting untuk memahami alokasi sumber daya yang diperlukan untuk memastikan koneksi E2E yang stabil.

### 5.2.3.2 End-to-end Connection

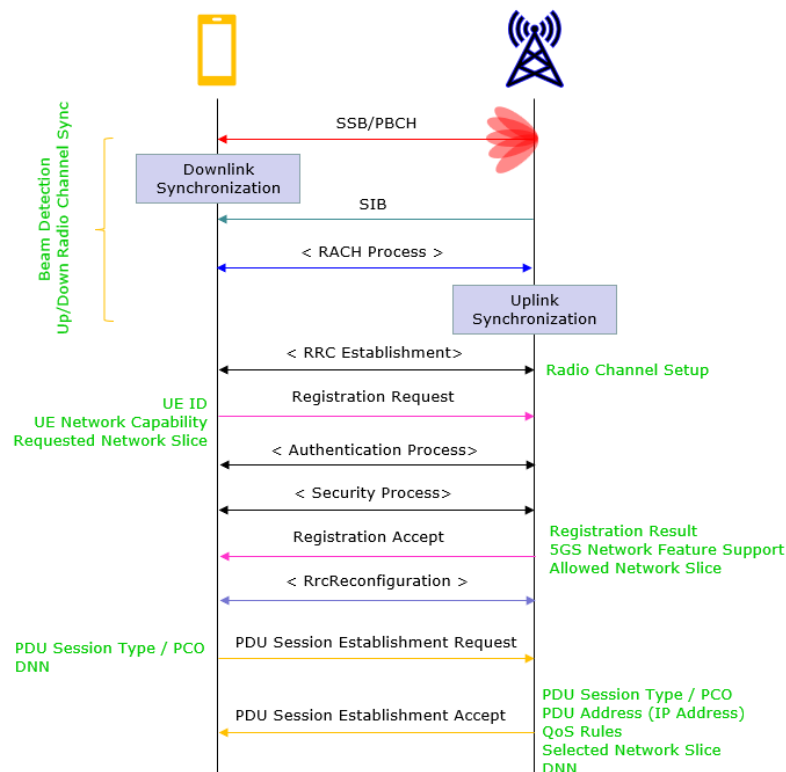
Pengujian E2E merupakan proses yang digunakan untuk memastikan bahwa seluruh komponen 5G dapat terhubung dengan baik dari CN hingga ke RAN. Gambar 5.63

memperlihatkan alur skenario pengujian E2E yang dilakukan pada komponen RAN khususnya pada UE.



**Gambar 0.63 Diagram Alir Pengujian E2E**

Pengujian koneksi E2E dilakukan pada komponen UE untuk memastikan bahwa pengguna akhir dapat terhubung ke jaringan 5G dan mendapatkan alamat IP yang valid. UE mendapatkan alamat IP dan terhubung ke jaringan core 5G dengan melewati berbagai proses dan menggunakan berbagai macam protokol.



**Gambar 0.64 Proses UE Mencapai PDU Establishment**

Gambar 5.64 memperlihatkan serangkaian proses autentikasi yang dilakukan dari UE ke gNB agar dapat terhubung ke core. Pada akhir proses, terdapat PDU Session Establishment Accept yang menandakan UE sudah terhubung ke core dan mendapatkan

alamat IP sehingga UE dapat terhubung ke internet. Ketika dilakukan *sniffing* packet menggunakan wireshark, proses autentikasi menggunakan berbagai macam protokol seperti F1AP, NR RRC, SCTP, dan NAS seperti terlampir pada Gambar 5.65.

No.	Time	Source	Destination	Protocol	Length	Info
9	34.120141	192.168.1.2	192.168.1.1	FIAP/NR RRC	254	InitialULRRCTransfer, RRC Setup Request
10	34.121393	192.168.1.1	192.168.1.2	FIAP/NR RRC	250	SACK (Ack=0, Arwnd=106496), DLRRCTransfer, RRC Setup
11	34.323962	192.168.1.2	192.168.1.1	SCTP	62	SACK (Ack=0, Arwnd=106496)
12	34.764284	192.168.1.2	192.168.1.1	FIAP/NR RRC/NAS-SGS	138	ULRRCTransfer, RRC Setup Complete, Registration request MAC=0x00000000
13	34.968816	192.168.1.1	192.168.1.2	SCTP	62	SACK (Ack=1, Arwnd=106496)
14	35.043675	192.168.1.1	192.168.1.2	FIAP/NR RRC/NAS-SGS	190	DLRRCTransfer, DL Information Transfer, Authentication request MAC=0x00000000
15	35.243978	192.168.1.2	192.168.1.1	SCTP	62	SACK (Ack=1, Arwnd=106496)
16	35.274389	192.168.1.2	192.168.1.1	FIAP/NR RRC/NAS-SGS	122	ULRRCTransfer, UL Information Transfer, Authentication response MAC=0x00000000
17	35.476021	192.168.1.1	192.168.1.2	SCTP	62	SACK (Ack=2, Arwnd=106496)
18	35.484712	192.168.1.1	192.168.1.2	FIAP/NR RRC/NAS-SGS	290	DLRRCTransfer, DL Information Transfer, Security mode command MAC=0x00000000
19	35.691947	192.168.1.2	192.168.1.1	SCTP	62	SACK (Ack=2, Arwnd=106496)
20	35.712826	192.168.1.2	192.168.1.1	FIAP/NR RRC/NAS-SGS	158	ULRRCTransfer, UL Information Transfer MAC=0x00000000
21	35.782959	192.168.1.1	192.168.1.2	FIAP	174	SACK (Ack=3, Arwnd=106496), UEContextSetupRequest
22	35.784050	192.168.1.2	192.168.1.1	FIAP	242	SACK (Ack=3, Arwnd=106496), UEContextSetupResponse
23	35.987983	192.168.1.1	192.168.1.2	SCTP	62	SACK (Ack=4, Arwnd=106496)
24	35.994828	192.168.1.2	192.168.1.1	FIAP/NR RRC	102	ULRRCTransfer, Security Mode Complete MAC=0x3e94b687
25	35.995786	192.168.1.1	192.168.1.2	FIAP/NR RRC	122	SACK (Ack=5, Arwnd=106496), DLRRCTransfer, UE Capability Enquiry MAC=0x90f17d95
26	36.200867	192.168.1.2	192.168.1.1	SCTP	62	SACK (Ack=4, Arwnd=106496)
27	36.469231	192.168.1.2	192.168.1.1	FIAP/NR RRC	114	ULRRCTransfer, UE Capability Information MAC=0x6e20989d
28	36.470364	192.168.1.1	192.168.1.2	FIAP/NR RRC/NAS-SGS	342	SACK (Ack=6, Arwnd=106496), DLRRCTransfer, RRC Reconfiguration MAC=0x597b7406
29	36.671976	192.168.1.2	192.168.1.1	SCTP	62	SACK (Ack=5, Arwnd=106496)
30	36.750100	192.168.1.2	192.168.1.1	FIAP/NR RRC	102	ULRRCTransfer, RRC Reconfiguration Complete MAC=0x3361b24e
31	36.751646	192.168.1.1	192.168.1.2	FIAP	118	SACK (Ack=7, Arwnd=106496), UEContextModificationRequest
32	36.752177	192.168.1.2	192.168.1.1	FIAP	98	SACK (Ack=6, Arwnd=106496), UEContextModificationResponse
33	36.960010	192.168.1.1	192.168.1.2	SCTP	62	SACK (Ack=8, Arwnd=106496)
34	37.716499	192.168.1.2	192.168.1.1	FIAP/NR RRC/NAS-SGS	114	ULRRCTransfer, UL Information Transfer MAC=0xefc5d2ba

**Gambar 0.65 Hasil Wireshark Saat Proses PDU Establishment**

Langkah terakhir adalah verifikasi pada UE yang meliputi pengecekan alamat IP dan pengujian konektivitas ke internet. Interface `oaitun_ue1` merupakan interface yang digunakan untuk mengirimkan trafik data-plane. Sehingga, ketika interface ini sudah mendapatkan alamat IP menandakan sudah bisa terhubung ke internet. Gambar 5.66 merupakan verifikasi alamat IP pada interface `oaitun_ue1` dan verifikasi koneksi ke internet.

```

root@oai-nr-ue-level1-user1-6d868fcd4-b7xnj:/opt/oai-nr-ue# ip addr show oaitun_ue1
3: oaitun_ue1: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
    link/none
    inet 12.1.1.2/24 brd 12.1.1.255 scope global oaitun_ue1
        valid_lft forever preferred_lft forever
    inet6 fe80::ea6f:225b:44c9:8029/64 scope link stable-privacy
        valid_lft forever preferred_lft forever
root@oai-nr-ue-level1-user1-6d868fcd4-b7xnj:/opt/oai-nr-ue# ping www.google.com
PING www.google.com (64.233.170.103) 56(84) bytes of data:
64 bytes from sg-in-f103.1e100.net (64.233.170.103): icmp_seq=1 ttl=57 time=36.4 ms
64 bytes from sg-in-f103.1e100.net (64.233.170.103): icmp_seq=2 ttl=57 time=35.5 ms
64 bytes from sg-in-f103.1e100.net (64.233.170.103): icmp_seq=3 ttl=57 time=34.9 ms
64 bytes from sg-in-f103.1e100.net (64.233.170.103): icmp_seq=4 ttl=57 time=36.5 ms
--- www.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 34.871/35.829/36.513/0.681 ms

```

**Gambar 0.66 Pod UE Mendapatkan Interface `oaitun_ue1`**

Tabel 5.13 memperlihatkan hasil pengujian end-to-end dengan menggunakan metode yang sama pada System Load Capacity. Total akun user yang sudah dibuat berjumlah 11 akun dengan total *success rate* 100% sebanyak 10 akun user dan *success rate* 0% sebanyak 2 akun user.

**Tabel 0.13 Hasil Pengujian *Success Rate***

User ke-	Success Rate	Keterangan
0	100%	UE mendapatkan alamat IP

1	100%	UE mendapatkan alamat IP
2	100%	UE mendapatkan alamat IP
3	100%	UE mendapatkan alamat IP
4	100%	UE mendapatkan alamat IP
5	100%	UE mendapatkan alamat IP
6	100%	UE mendapatkan alamat IP
7	100%	UE mendapatkan alamat IP
8	100%	UE mendapatkan alamat IP
9	100%	UE mendapatkan alamat IP
10	100%	UE mendapatkan alamat IP
11	0%	UE tidak mendapatkan alamat IP

#### 5.2.4 User Acceptance Testing

*User Acceptance Testing* (UAT) bertujuan untuk menilai kepuasan pengguna terhadap aplikasi dan memastikan aplikasi memenuhi kebutuhan mereka. UAT melibatkan *end users*, admin, dan calon pembeli aplikasi untuk memberikan *feedback* terhadap fungsionalitas dan pengalaman pengguna.

Pengujian UAT dilakukan dengan menggunakan server milik Telecom Infra Project (TIP). Jumlah *end user* yang melakukan pengujian ini berjumlah 30 orang dari berbagai *background* seperti mahasiswa, *staff* TIP, dan *engineer*. Untuk dokumentasi pelaksanaan UAT ini dapat dilihat pada lampiran 5.1 – 5.4.

Pelaksanaan UAT dilakukan dengan sesi uji coba yang terstruktur. Pada pengujian ini, user mendapatkan kesempatan untuk melakukan konfigurasi dan mensimulasikan 5G secara langsung pada *dashboard*. Di akhir sesi uji coba, *user* akan mengisi kuisioner *feedback* mengenai pengalaman mereka ketika menggunakan aplikasi. Pengujian UAT memungkinkan *user* maupun *developer* mengidentifikasi kekurangan atau *bug* pada fitur-fitur yang ada. Selain itu juga agar aplikasi lebih sesuai dengan kebutuhan dan ekspektasi *end user*.



Pengujian UAT ini dibagi menjadi dua. Yang pertama *functional testing* bertujuan untuk mengetahui fitur-fitur pada aplikasi berhasil dijalankan oleh *user*. Yang kedua yaitu *usability testing* untuk mendapatkan *feedback* berharga mengenai aplikasi dari *end user*.

**Tabel 0.14 Langkah Pengujian UAT**

Langkah	Deskripsi
Penyusunan Kuesioner	Menyusun pertanyaan dengan metode System Usability Scale (SUS) untuk mendapatkan feedback dari pengguna
Penyusunan Tabel Functional Testing	Menyusun tabel untuk <i>functional testing</i> tiap fitur
Pelaksanaan UAT	Mengundang 30 pengguna dari berbagai background untuk menguji dashboard. UAT dibagi kedalam enam sesi, di mana tiap sesinya berisi lima orang.
Pengumpulan Kuesioner	Mengumpulkan dan menganalisis feedback dari pengguna yang melakukan pengujian.
Perbaikan berdasarkan feedback dari pengguna	Melakukan perbaikan pada dashboard berdasarkan feedback yang diterima

#### 5.2.4.1 Functional Testing

*Functional testing* adalah jenis pengujian yang memverifikasi bahwa setiap fitur pada aplikasi beroperasi sesuai dengan spesifikasi yang diinginkan. Pengujian ini menggunakan metode *black box* untuk menguji fitur pada tiap halamannya.

Metode *Black Box* adalah metode pengujian aplikasi di mana struktur internal seperti *source code* aplikasi tidak perlu diketahui oleh penguji sehingga pengujian ini dilakukan dari perspektif *end user*. Penguji hanya fokus pada apa yang dilakukan aplikasi.

*Functional testing* dengan metode *black box* adalah teknik yang ampuh untuk memastikan bahwa fitur aplikasi memenuhi persyaratan fungsionalnya tanpa menggali struktur internalnya. Hal ini membantu mengidentifikasi perbedaan antara hasil yang diharapkan dengan hasil sebenarnya dan memastikan aplikasi *user-centric*.

Tabel 0.15 Hasil *Functional Testing* pada Halaman Admin

Admin Page						
No	Halaman	Fitur	Langkah Pengujian	Hasil yang Diharapkan	Hasil Aktual	Komentar
1	User Management	Create	Membuat user baru sesuai dengan jumlah yang diinginkan	Akun user baru berhasil ditambahkan	Berhasil	
		Edit	Mengubah password baru pada user	Password baru pada user berhasil diubah	Berhasil	
		Remove	Menghapus akun user	Akun user berhasil dihapus dari sistem	Berhasil	

Tabel 0.16 Hasil *Functional Testing* pada Halaman Pengguna

User Page						
No	Halaman	Fitur	Langkah Pengujian	Hasil yang Diharapkan	Hasil Aktual	Komentar
1	Login	Login	Memasukkan username dan password lalu klik login	User berhasil login dan diarahkan ke halaman Introduction	Berhasil	
2	Introduction	Introduction	Menekan setiap menu pada sidebar di halaman Introduction	Berhasil menampilkan semua menu pada sidebar	Berhasil	



3	Dashboard	Topology Graph	Menekan tombol decorator status pada setiap komponen	Berhasil menampilkan pop up yang menunjukkan "deployment status"	Berhasil	
			Menekan tombol gambar setiap komponen	Berhasil menampilkan sidebar dan user dapat melihat Protocol Stack dan Log pada sidebar tersebut	Berhasil	
			Menekan tombol "Protocol Stack" dan "Logs" pada sidebar komponen UE	Berhasil menampilkan list protocol dan logs pada komponen UE	Berhasil, tapi belum sempurna	Protocol dan logs terkadang berhasil ditampilkan pada sidebar, terkadang tidak muncul pada sidebar.
			Menekan tombol "Protocol Stack" dan "Logs" pada sidebar komponen DU	Berhasil menampilkan list protocol dan logs pada komponen DU	Berhasil, tapi belum sempurna	Protocol dan logs terkadang berhasil ditampilkan pada sidebar,

				terkadang tidak muncul pada sidebar.	
		Menekan tombol "Protocol Stack" dan "Logs" pada sidebar komponen CU	Berhasil menampilkan list protocol dan logs pada komponen DU	Berhasil, tapi belum sempurna	Protocol dan logs terkadang berhasil ditampilkan pada sidebar, terkadang tidak muncul pada sidebar.
		Melakukan ping dan curl pada terminal di sidebar komponen UE	Berhasil melakukan ping dan curl	Berhasil	
		Menekan tombol kebab menu pada komponen UE. Lalu menekan tombol "Stop", "Start" dan "Restart"	Komponen UE berhasil reload jika terdapat perubahan konfigurasi ketika menekan tombol "Restart". Komponen UE berhenti running	Berhasil	

				ketika menekan tombol "Stop" dan running kembali setelah menekan tombol "Start".		
			Menekan tombol kebab menu pada komponen DU. Lalu menekan tombol "Stop", "Start" dan "Restart"	<p>Komponen DU berhasil reload jika terdapat perubahan konfigurasi ketika menekan tombol "Restart".</p> <p>Komponen DU berhenti running ketika menekan tombol "Stop" dan running kembali setelah menekan tombol "Start".</p>	Berhasil	

			<p>Menekan tombol kebab menu pada komponen CU. Lalu menekan tombol "Stop", "Start" dan "Restart"</p>	<p>Komponen CU berhasil reload jika terdapat perubahan konfigurasi ketika menekan tombol "Restart". Komponen CU berhenti running ketika menekan tombol "Stop" dan running kembali setelah menekan tombol "Start".</p>	Berhasil	
			<p>Menekan tombol link tiap interface</p>	<p>Berhasil menampilkan pop up nama interface</p>	Berhasil	
			<p>Menggeser tombol tiap komponen</p>	<p>Tiap komponen dapat digeser sesuai keinginan User</p>	Berhasil	
			<p>Menggeser topology</p>	<p>Topologi dapat digeser sesuai keinginan User</p>	Berhasil	
			<p>Menekan tombol control bar (zoom in, zoom out, fit to screen, reset view)</p>	<p>Bentuk topologi berhasil zoom in, zoom out, fit to screen, dan</p>	Berhasil	

			kembali ke bentuk semula		
		Configuration Panel	Mengisi value pada setiap kolom yang tersedia	User dapat menginput value pada setiap kolom saat mengkonfigurasi	Berhasil
			Menekan tombol submit setelah mengisi value	Value yang telah terinput berhasil di-submit	Berhasil
			Dapat melihat value yang sudah terupdate setelah proses submit	User dapat melihat value yang telah ter-submit pada kolom "Current Value"	Berhasil
			Mengkonfigurasi semua value pada komponen UE	Komponen UE berhasil terkonfigurasi, persentase completion bertambah, terdapat checklist pada config panel, warna pada topologi dan link berubah warna menjadi hijau	Berhasil

			Mengkonfigurasi semua value pada komponen DU	Komponen DU berhasil terkonfigurasi, persentase completion bertambah, terdapat checklist pada config panel, warna pada topologi dan link berubah warna menjadi hijau	Berhasil	
			Mengkonfigurasi semua value pada komponen CU	Komponen CU berhasil terkonfigurasi, persentase completion bertambah, terdapat checklist pada config panel, warna pada topologi dan link berubah warna menjadi hijau	Berhasil	
		Wireshark	Menekan tombol dropdown untuk memilih komponen yang akan di sniff	User dapat memilih komponen CU, DU, dan UE	Berhasil	

					Wireshark berhasil meng-capture paket data pada komponen UE jika pengguna memilih komponen UE terlebih dahulu untuk di-sniffing. Jika komponen lain dipilih terlebih dahulu, wireshark tidak dapat meng-capture paket data pada komponen UE.
			Menekan tombol start untuk memulai proses sniffing paket data pada komponen UE	Wireshark berhasil mengcapture paket data dan menampilkan protokol info pada komponen UE	Berhasil, tapi belum sempurna
			Menekan tombol start untuk memulai proses sniffing paket	Wireshark berhasil mengcapture paket data dan menampilkan	Berhasil, tapi belum sempurna
					Wireshark berhasil meng-capture paket data



			data pada komponen DU	protokol info pada komponen DU		pada komponen DU jika pengguna memilih komponen DU terlebih dahulu untuk di-sniffing. Jika komponen lain dipilih terlebih dahulu, wireshark tidak dapat meng-capture paket data pada komponen DU.
			Menekan tombol start untuk memulai proses sniffing paket data pada komponen CU	Wireshark berhasil mengcapture paket data dan menampilkan protokol info pada komponen CU	Berhasil, tapi belum sempurna	Wireshark berhasil meng-capture paket data pada komponen CU, namun tidak bisa dilakukan

				lebih dari 1 user di waktu yang sama.	
			Menekan tombol stop untuk menghentikan proses sniffing paket data	<p>Wireshark berhasil menghentikan proses sniffing paket data pada semua komponen</p> <p>Berhasil</p>	
			Meng-input paket data yang ingin di-filter dan menekan tombol "Filter"	<p>Berhasil memfilter paket data yang diinginkan</p> <p>Berhasil</p>	
			Menekan tab Files untuk melihat daftar file pcap yang telah di capture sebelumnya	<p>Berhasil menampilkan tab Files pada fitur Wireshark</p> <p>Berhasil</p>	
			Menekan tombol download untuk mendownload file pcap yang diinginkan	<p>File pcap berhasil diunduh di device user dan file tidak corrupt</p> <p>Berhasil</p>	
			Menekan tombol remove untuk menghapus file	<p>File pcap berhasil dihapus pada tab Files</p> <p>Berhasil</p>	

			pcap yang diinginkan			
4	Monitoring	L1 UE Graph	Melihat informasi grafik dari komponen UE secara real-time	Grafik berhasil ditampilkan secara real time	Berhasil	
		UE Protocol Table	Melihat informasi protocol dari komponen UE dalam bentuk tabel secara real-time	Berhasil menampilkan informasi protocol pada komponen UE secara real-time	Berhasil	

#### 5.2.4.2 Usability Testing

Usability testing adalah metode pengujian untuk mengamati dan mengevaluasi bagaimana pengalaman pengguna saat berinteraksi dengan sistem. Usability testing yang digunakan yaitu System Usability Scale (SUS) yang dikembangkan oleh John Brooke di tahun 1996. SUS adalah standar kuesioner yang paling banyak digunakan untuk menilai usability yang dirasakan oleh pengguna. SUS bertujuan untuk mengevaluasi kegunaan berbagai produk dan sistem, seperti situs web, aplikasi *software*, dan *device*. SUS terdiri dari 10 pertanyaan dengan lima pilihan jawaban responden, mulai dari “Sangat Tidak Setuju” hingga “Sangat Setuju”.

10 Pertanyaan SUS yaitu di antaranya:

1. Saya berpikir akan menggunakan sistem ini lagi
2. Saya merasa sistem ini rumit untuk digunakan
3. Saya merasa sistem ini mudah digunakan
4. Saya membutuhkan bantuan dari orang lain atau teknisi dalam menggunakan sistem ini

5. Saya merasa fitur-fitur sistem ini berjalan dengan semestinya
6. Saya merasa ada banyak hal yang tidak konsisten (tidak serasi pada sistem ini)
7. Saya merasa orang lain akan memahami cara menggunakan sistem ini dengan cepat
8. Saya merasa sistem ini membingungkan
9. Saya merasa tidak ada hambatan dalam menggunakan sistem ini
10. Saya perlu membiasakan diri terlebih dahulu sebelum menggunakan sistem ini

Setelah responden mengisi kuesioner, maka perlu melakukan perhitungan terhadap data yang didapatkan. SUS memiliki aturan dalam menghitung skor yang didapat. Berikut adalah aturan skoring pada metode SUS:

- Untuk pertanyaan bernomor ganjil: nilai dari setiap pertanyaan yang didapat dari pengguna akan dikurangi 1 nilai. Contoh nilai yang didapat yaitu 4 maka nilai akhirnya (4-1=3)
- Untuk pertanyaan bernomor genap: nilai 5 dikurangi nilai dari setiap pertanyaan yang didapat dari pengguna. Contoh nilai yang didapat yaitu 4 maka nilai akhirnya (5-4=1)
- Hal ini bertujuan untuk menskalakan semua nilai dari 0 hingga 4
- Jumlahkan semua nilai, lalu kalikan dengan 2,5 untuk mengkonversi rentang skor menjadi 0 hingga 100

Setiap pertanyaan bernomor ganjil, nilai yang diharapkan adalah 5 (Sangat Setuju) dan nilai yang tidak diharapkan adalah 1 (Sangat Tidak Setuju). Lalu untuk pertanyaan bernomor genap, nilai yang diharapkan adalah 1 (Sangat Tidak Setuju) dan nilai yang tidak diharapkan adalah 5 (Sangat Setuju). Cara di atas adalah cara untuk menentukan skor akhir dari tiap responden. Selanjutnya adalah menghitung data dari keseluruhan responden dengan mencari rata-ratanya. Jadi menjumlahkan skor akhir dari semua responden lalu dibagi dengan jumlah responden.

Rumus untuk menghitung skor SUS sebagai berikut:

$$\bar{x} = \frac{\sum x}{n}$$

( 0.1 )

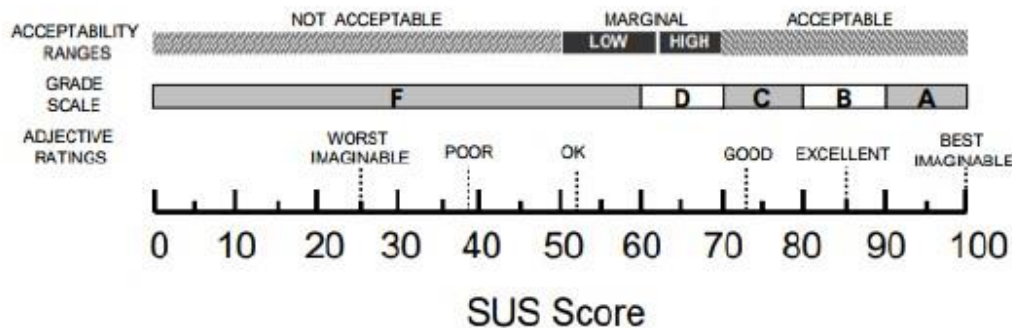
Keterangan:

$\bar{x}$  : skor rata-rata

$\sum x$ : jumlah skor SUS

$n$ : jumlah responden

Dari hasil nilai rata-rata skor SUS yang didapat, langkah selanjutnya yaitu membanding nilai skor SUS yang didapat dengan kategori penilaian SUS.



**Gambar 0.67 Index Parameter Penilaian System Usability Scale**

Nilai rata-rata SUS dari berbagai penelitian adalah 68. Sederhananya, jika di bawah 68 maka nilai SUS dianggap rendah dan aplikasi perlu diperbaiki. Sedangkan jika di atas 68, maka nilai SUS dianggap tinggi. Namun jika mengacu pada gambar, maka skor di atas 80 dianggap “*excellent*”, antara 70 dan 80 dianggap “*good*”, antara 50 dan 69 “*OK*” atau dapat diterima, dan di bawah 50 dianggap “*poor*”.

Selain 10 pertanyaan SUS pada kuesioner, kami menambahkan enam pertanyaan untuk mengetahui apakah dashboard kami meningkatkan *skill* dan menambah pengetahuan pengguna mengenai 5G RAN dan apakah pengguna akan merekomendasikan aplikasi ini ke orang lain. Enam pertanyaan tersebut adalah:

1. Saya merasa fitur *Graphical User Interface* (GUI) pada ORCA *dashboard website* memudahkan saya dalam konfigurasi komponen RAN 5G (CU, DU, dan UE)
2. *Lab guidance* dan materi Open RAN 5G pada halaman Introduction membantu saya dalam memahami Open RAN 5G dan parameter-parameter konfigurasi 5G RAN
3. Visualisasi *topology graph* membantu saya memahami arsitektur 5G
4. Saya dapat mempelajari protokol apa saja pada jaringan 5G dan dapat menganalisis paket data jaringan melalui fitur Wireshark
5. Saya dapat memahami *Key Performance Indicator* (KPI) apa saja yang terdapat pada komponen UE pada fitur monitoring

6. Seberapa besar anda akan merekomendasikan ORCA *dashboard website* ini kepada orang lain?

Kelemahan dari kuesioner SUS adalah kita tidak dapat mengetahui masalah pada aplikasi jika terdapat pertanyaan dengan nilai yang rendah atau nilai akhirnya rendah. Oleh karena itu, terdapat kolom kesan dan saran untuk dapat mengetahui masalah yang dialami pengguna dan mendapatkan masukan dari pengguna terkait aplikasi.

Berikut adalah hasil analisis perhitungan *usability testing* dari kuesioner *feedback* pengguna. Nilai akhir SUS dari UAT 1 adalah 65,42. Untuk perhitungan lebih lengkapnya dapat dilihat pada lampiran 5.5. Lalu untuk hasil akhir SUS dari UAT 2 yaitu 77,1 dan perhitungan lengkapnya dapat dilihat pada lampiran 5.6.

**Tabel 0.17 Hasil Perhitungan SUS dari UAT 1**

No	Respondent	User Persona	Total	Final Score (Total x 2.5)
1	Respondent 1	Staff TIP	22	55
2	Respondent 2	Staff TIP	30	75
3	Respondent 3	Student	23	57,5
4	Respondent 4	Student	30	75
5	Respondent 5	Student	18	45
6	Respondent 6	Student	23	57,5
7	Respondent 7	Student	23	57,5
8	Respondent 8	Student	20	50
9	Respondent 9	Staff TIP	26	65
10	Respondent 10	Student	21	52,5
11	Respondent 11	Staff TIP	28	70
12	Respondent 12	Student	27	67,5
13	Respondent 13	Student	32	80
14	Respondent 14	Student	22	55
15	Respondent 15	Student	35	87,5
16	Respondent 16	Student	23	57,5
17	Respondent 17	Student	32	80
18	Respondent 18	Student	32	80

19	Respondent 19	Staff TIP	28	70
20	Respondent 20	Student	26	65
21	Respondent 21	Student	26	65
22	Respondent 22	Engineer	22	55
23	Respondent 23	Engineer	31	77,5
24	Respondent 24	Engineer	29	72,5
25	Respondent 25	Engineer	28	70
26	Respondent 26	Engineer	25	62,5
27	Respondent 27	Engineer	18	45
28	Respondent 28	Engineer	32	80
29	Respondent 29	Engineer	22	55
30	Respondent 30	Student	31	77,5
Average				65,42

**Tabel 0.18 Hasil Perhitungan SUS dari UAT 2**

No	Respondent	User Persona	Total	Final Score (Total x 2.5)
1	Respondent 1	Staff TIP	39	97,5
2	Respondent 2	Staff TIP	34	85
3	Respondent 3	Student	27	67,5
4	Respondent 4	Student	36	90
5	Respondent 5	Student	29	72,5
6	Respondent 6	Student	27	67,5
7	Respondent 7	Student	32	80
8	Respondent 8	Student	33	82,5
9	Respondent 9	Staff TIP	32	80
10	Respondent 10	Student	34	85
11	Respondent 11	Staff TIP	33	82,5
12	Respondent 12	Student	37	92,5
13	Respondent 13	Student	29	72,5
14	Respondent 14	Student	34	85



15	Respondent 15	Student	35	87,5
16	Respondent 16	Student	28	70
17	Respondent 17	Student	30	75
18	Respondent 18	Student	29	72,5
19	Respondent 19	Staff TIP	30	75
20	Respondent 20	Student	27	67,5
21	Respondent 21	Student	26	65
22	Respondent 22	Engineer	22	55
23	Respondent 23	Engineer	33	82,5
24	Respondent 24	Engineer	29	72,5
25	Respondent 25	Engineer	35	87,5
26	Respondent 26	Engineer	25	62,5
27	Respondent 27	Engineer	29	72,5
28	Respondent 28	Engineer	29	72,5
29	Respondent 29	Engineer	30	75
30	Respondent 30	Student	32	80
Average				77,1

### 5.3 Analisa Hasil Pengujian

#### 5.3.1 Hasil API Testing

Hasil API *testing* dilihat berdasarkan akumulasi keberhasilan seluruh API yang tersedia pada setiap fungsionalitas sesuai dengan spesifikasi yang telah ditentukan. Hasil ini mengacu pada fungsionalitas API yang dites menggunakan aplikasi Postman untuk tipe REST API dan juga *code editor* dengan Django *libraries* untuk tipe WebSocket API. Hasil ini akan disajikan dalam bentuk tabel yang dibagi menjadi lima bagian.

##### 5.3.1.1 Hasil User Management APIs

**Tabel 0.19 Hasil User Management APIs**

Kategori Fungsi	Hasil yang diharapkan	Status		Kendala
		Sukses	Gagal	

Create user	HTTP status code: 201 Created	✓		-
List all users	HTTP status code: 200 OK	✓		-
Update user's password	HTTP status code: 200 OK	✓		-
Delete user	HTTP status code: 204 No Content	✓		-
Get requested user information	HTTP status code: 200 OK	✓		-

Berdasarkan tabel di atas, seluruh kategori fungsi berstatus sukses, yang berarti tingkat keberhasilannya sebesar 100%. Fungsi untuk manajemen user tidak memerlukan olah data yang rumit pada *database*, sehingga kemungkinan adanya kendala ataupun factor penghambat sangatlah kecil. Dengan begitu, dapat disimpulkan fungsionalitas API ini bekerja dengan baik.

### 5.3.1.2 Hasil Token APIs

**Tabel 0.20 Hasil Token APIs**

Kategori Fungsi	Hasil yang diharapkan	Status		Kendala
		Sukses	Gagal	
Login	HTTP status code: 200 OK	✓		-
Logout	HTTP status code: 200 OK	✓		-
Refresh Token	HTTP status code: 200 OK	✓		-
Verify Token	HTTP status code: 200 OK	✓		-

Berdasarkan tabel di atas, seluruh kategori fungsi berstatus sukses, yang berarti tingkat keberhasilannya sebesar 100%. Fungsi untuk manajemen JSON Web Token (JWT)

yang dikembangkan sudah mencakup segala hal yang dibutuhkan untuk mendukung proses otentikasi dan otorisasi. Selain itu, tidak ada factor penghambat dalam proses pengembangan fungsionalitas API ini yang menandakan API bekerja dengan baik.

### 5.3.1.3 Hasil Kubernetes APIs

**Tabel 0.21 Kubernetes APIs**

Kategori Fungsi	Hasil yang diharapkan	Status		Kendala
		Sukses	Gagal	
<i>Get Requested Pod List</i>	<i>HTTP status code: 200 OK</i>	✓		-
<i>Get Requested Deployment List</i>	<i>HTTP status code: 200 OK</i>	✓		-
<i>Get Requested Pod Log</i>	<i>HTTP status code: 200 OK</i>	✓		-
<i>Restart 5G Components</i>	<i>HTTP status code: 200 OK</i>	✓		-
<i>Get Core (AMF &amp; UPF) Log</i>	<i>HTTP status code: 200 OK</i>	✓		-
<i>Get Core (AMF &amp; UPF) Deployment Status</i>	<i>HTTP status code: 200 OK</i>	✓		-
<i>Ping and cURL command</i>	<i>Hasil perintah ping dan cURL dapat ditampilkan dengan benar pada terminal code editor</i>	✓		-
<i>Get UE Monitoring (Key Performance Indicator)</i>	<i>KPI dapat ditampilkan dengan benar pada terminal code editor</i>	✓		-

<i>Get SCTP Protocol Information</i>	<i>Informasi protocol SCTP dapat ditampilkan dengan benar pada terminal code editor</i>	✓		-
--------------------------------------	---	---	--	---

Berdasarkan tabel di atas, seluruh kategori fungsi berstatus sukses, yang berarti tingkat keberhasilannya sebesar 100%. Terlepas dari itu, fungsi untuk manajemen kluster Kubernetes yang telah dikembangkan memiliki beberapa faktor penghambat yang sempat muncul meliputi penempatan file config yang tidak tepat, sehingga fungsi manajemen tidak dapat dijalankan dengan benar.

Namun hal itu dapat diatasi dengan menempatkan file config pada direktori yang tepat beserta pemberian hak akses yang benar agar system backend dapat mengaksesnya tanpa kendala. Dan juga, sebagai factor pendukung dalam pengembangan fungsi Kubernetes ini, tidak hanya dibutuhkan REST API melainkan juga Websocket API. Websocket API sangat membantu dalam hal integrasi dengan data-data yang bersifat real-time dan dibutuhkan untuk diolah secara real-time. Dengan begitu, dapat disimpulkan jika kendala yang sempat muncul dapat diatasi dengan baik dan keseluruhan fungsi bekerja dengan benar.

Rencana untuk pengembangan lebih lanjut dari fungsionalitas ini adalah dengan memanfaatkan Websocket API dengan maksimal agar pengolahan data yang membutuhkan koneksi real-time dapat dilakukan dengan baik. Meskipun saat ini penggunaan websocket sudah dilakukan, namun dalam implementasinya belum dimanfaatkan dengan maksimal dikarenakan keterbatasan fungsi yang diperlukan hingga saat ini.

#### 5.3.1.4 Hasil Helm Chart APIs

**Tabel 0.22 Hasil Helm Chart APIs**

Kategori Fungsi	Hasil yang diharapkan	Status		Kendala
		Sukses	Gagal	
<i>Get Configuration Values</i>	<i>HTTP status code: 200 OK</i>	✓		-
<i>Config Component Values</i>	<i>HTTP status code: 200 OK</i>	✓		-

<i>Start Component</i>	<i>HTTP status code: 200 OK</i>	✓		-
<i>Stop Component</i>	<i>HTTP status code: 200 OK</i>	✓		-

Berdasarkan tabel di atas, seluruh kategori fungsi berstatus sukses, yang berarti tingkat keberhasilannya sebesar 100%. Terlepas dari itu, fungsi untuk manajemen Helm Chart sempat memiliki factor penghambat dalam proses pengembangannya. Faktor tersebut sama halnya dengan fungsi manajemen kluster Kubernetes yang terletak pada kurang tepatnya pemberian hak akses untuk melakukan baris perintah, yang mana dalam kasus ini adalah baris perintah Helm. Dengan begitu, dapat disimpulkan tidak ada kendala pada fungsionalitasnya.

### 5.3.1.5 Hasil Wireshark APIs

**Tabel 0.23 Hasil Wireshark APIs**

Kategori Fungsi	Hasil yang diharapkan	Status		Kendala
		Sukses	Gagal	
<i>List PCAP Files</i>	<i>HTTP status code: 200 OK</i>	✓		-
<i>Download PCAP Files</i>	<i>HTTP status code: 200 OK</i>	✓		-
<i>Delete PCAP Files</i>	<i>HTTP status code: 204 No Content</i>	✓		-
<i>Sniffing Process and Save PCAP Files into Database</i>	<i>Hasil sniffing paket data dapat tertampil dengan benar pada terminal code editor</i>	✓		-

Berdasarkan tabel di atas, seluruh kategori fungsi berstatus sukses, yang berarti tingkat keberhasilannya sebesar 100%. Fungsi untuk Wireshark merupakan fungsi penting yang memerlukan koneksi real-time untuk proses pengolahan paket data hasil dari proses sniffing yang dilakukan. Faktor penghambat yang sempat ditemui adalah penggunaan REST API yang kurang efektif dalam proses pengolahan data sniffing secara real-time.

Dengan adanya permasalahan itu, solusi yang digunakan adalah pengimplementasian Websocket API seperti halnya dengan kasus yang terjadi pada fungsi manajemen kluster Kubernetes. Dengan begitu, pengolahan paket data hasil proses sniffing dapat dilakukan secara langsung tanpa harus menunggu data yang dihasilkan dikumpulkan semuanya terlebih dahulu. Rencana kedepannya dalam pengembangan ini adalah untuk mengimplementasikan websocket dengan maksimal untuk fungsi sniffing.

Kendala yang ditemui setelah mengimplementasikan websocket pada fungsi sniffing terdapat pada proses koneksi untuk websocket terkadang memakan waktu yang sedikit lama. Sehingga, terdapat sedikit jeda pada sisi frontend ketika sebuah tombol berfungsi untuk memicu koneksi websocket tersebut dan mengirimkan datanya ke sisi frontend.

### 5.3.2 Hasil *End-to-End* (E2E) Testing

Hasil ini mengacu pada keberhasilan seluruh alur pada aplikasi yang mengindikasikan sistem frontend dan backend bekerja harmonis pada semua fungsinya. Proses testing dilakukan menggunakan website dashboard yang telah dikembangkan.

#### 5.3.2.1 Hasil User Management Functions

**Tabel 0.24 Hasil User Management Functions**

Kategori Fungsi	Hasil yang diharapkan	Status		Kendala
		Sukses	Gagal	
Membuat Akun User	Admin berhasil membuat akun user baru	✓		-
Melihat Daftar Akun User	Admin berhasil melihat daftar akun user	✓		-
Mengupdate Password Akun User	Admin berhasil mengubah password akun user	✓		-
Menghapus Akun User	Admin berhasil menghapus akun user	✓		-
Melihat Informasi Akun User	User berhasil melihat informasi akunnya	✓		-

Berdasarkan tabel di atas, seluruh kategori fungsi berstatus sukses, yang berarti tingkat keberhasilannya sebesar 100%. Hal ini menandakan bahwa proses integrasi frontend

hingga ke backend pada fungsi manajemen user berjalan dengan benar dan telah memenuhi ekspektasi dari fungsi yang tersedia pada website dashboard yang dikembangkan.

### 5.3.2.2 Hasil Token Functions

**Tabel 0.25 Hasil Token Functions**

Kategori Fungsi	Hasil yang diharapkan	Status		Kendala
		Berhasil	Gagal	
Login	User berhasil masuk ke dalam laman introduction, Admin berhasil masuk ke dalam laman user management	✓		-
Logout	Admin/User berhasil keluar dari website dan kembali ke halaman login	✓		-
Refresh Token	Admin/User tetap berhasil mengakses website dashboard ketika masa aktif token sudah usang	✓		-
Verifikasi Token	Admin/User tetap berhasil mengakses website dashboard meskipun telah keluar dari laman dalam kurun waktu masa aktif token	✓		-

Berdasarkan tabel di atas, seluruh kategori fungsi berstatus sukses, yang berarti tingkat keberhasilannya sebesar 100%. Hal ini menandakan bahwa proses integrasi frontend hingga ke backend pada fungsi token terkait otentikasi dan otorisasi berjalan dengan benar dan telah memenuhi ekspektasi dari fungsi yang tersedia pada website dashboard yang dikembangkan.

### 5.3.2.3 Hasil Kubernetes Functions

**Tabel 0.26 Hasil Kubernetes Functions**

Kategori Fungsi	Hasil yang diharapkan	Status	Kendala
-----------------	-----------------------	--------	---------



		Berhasil	Gagal	
User berhasil mendapatkan daftar informasi pod Kubernetes miliknya	Muncul gambar topologi pada laman dashboard dengan status komponen stopped/running	✓		-
User berhasil mendapatkan daftar informasi deployment Kubernetes miliknya	Muncul gambar topologi pada laman dashboard dengan status komponen stopped/running	✓		-
User berhasil mendapatkan informasi log dari pod Kubernetes miliknya	Muncul informasi log pada setiap komponen pada sidebar bagian tab Logs	✓		-
User berhasil me-restart deployment Kubernetes (komponen 5G) sebagai manajemen siklus hidup komponen	Notifikasi sukses proses restart yang dilakukan pada setiap komponen	✓		-
User berhasil mendapatkan informasi log dari komponen AMF	Muncul informasi log komponen AMF pada sidebar bagian tab Logs	✓		-
User berhasil mendapatkan informasi log dari komponen UPF	Muncul informasi log komponen UPF pada sidebar bagian tab Logs	✓		-
User berhasil mendapatkan informasi data	Muncul informasi deployment status	✓		-

deployment komponen AMF	komponen AMF pada tombol decorator status			
User berhasil mendapatkan informasi data deployment komponen UPF	Muncul informasi deployment status komponen UPF pada tombol decorator status	✓		-
User berhasil melakukan perintah ping dan melihat hasilnya pada komponen UE	Perintah ping menghasilkan output berupa paket diterima dari server dituju	✓		-
User berhasil melakukan perintah cURL dan melihat hasilnya pada komponen UE	Perintah cURL menghasilkan output berupa paket ditampilkan dalam bentuk HTML code dari server dituju	✓		-
User berhasil mendapatkan informasi Key Performance Indicator (KPI) dari komponen UE	Muncul nilai-nilai KPI pada table value beserta dengan munculnya grafik monitoring yang bekerja secara real-time untuk memvisualisasikan nilai- nilai KPI tersebut	✓		-
User berhasil mendapatkan informasi protokol SCTP dari komponen RAN 5G (CU dan DU)	Munculnya informasi terkait protocol SCTP pada sidebar bagian tab Protocol	✓		-

Berdasarkan tabel di atas, seluruh kategori fungsi berstatus sukses, yang berarti tingkat keberhasilannya sebesar 100%. Hal ini menandakan bahwa proses integrasi frontend hingga ke backend pada fungsi manajemen kluster Kubernetes berjalan dengan benar dan

telah memenuhi ekspektasi dari fungsi yang tersedia pada website dashboard yang dikembangkan.

Terlepas itu, adapun sedikit anomali yang terkadang muncul ketika menjalankan fungsi ini grafik topologi terkadang tidak tertampil dengan benar pada saat user pertama kali mengakses laman dashboard, dan hal itu dapat teratasi dengan menekan tombol refresh yang terletak di dalam menu Topology Graph yang tersedia pada lama dashboard.

#### 5.3.2.4 Hasil Helm Chart Functions

**Tabel 0.27 Hasil Helm Chart Function**

Kategori Fungsi	Hasil yang diharapkan	Status		Kendala
		Berhasil	Gagal	
User berhasil mendapatkan informasi value konfigurasi pada setiap komponen 5G (CU, DU, dan UE)	Muncul value pada kolom Current Value setelah user melakukan konfigurasi pada kolom Set Value	✓		-
User berhasil mengubah value konfigurasi pada setiap komponen 5G (CU, DU, dan UE)	Muncul notifikasi status berhasil konfigurasi sesaat setelah user mengkonfirmasi pengubahan value pada komponen yang dikonfigurasi	✓		-
User berhasil menjalankan setiap komponen 5G (CU, DU, dan UE) pada manajemen siklus hidup komponen	Notifikasi sukses pada proses start yang dilakukan pada setiap komponen beserta berubahnya warna frame menjadi hijau pada komponen tersebut	✓		-
User berhasil menghentikan setiap	Notifikasi sukses pada proses stop yang	✓		-

komponen 5G (CU, DU, dan UE) pada manajemen siklus hidup komponen	dilakukan pada setiap komponen beserta berubahnya warna frame menjadi kuning pada komponen tersebut			
---	---	--	--	--

Berdasarkan tabel di atas, seluruh kategori fungsi berstatus sukses, yang berarti tingkat keberhasilannya sebesar 100%. Hal ini menandakan bahwa proses integrasi frontend hingga ke backend pada fungsi manajemen Helm Chart berjalan dengan benar dan telah memenuhi ekspektasi dari fungsi yang tersedia pada website dashboard yang dikembangkan.

### 5.3.2.5 Hasil Wireshark Functions

**Tabel 0.28 Hasil Wireshark Functions**

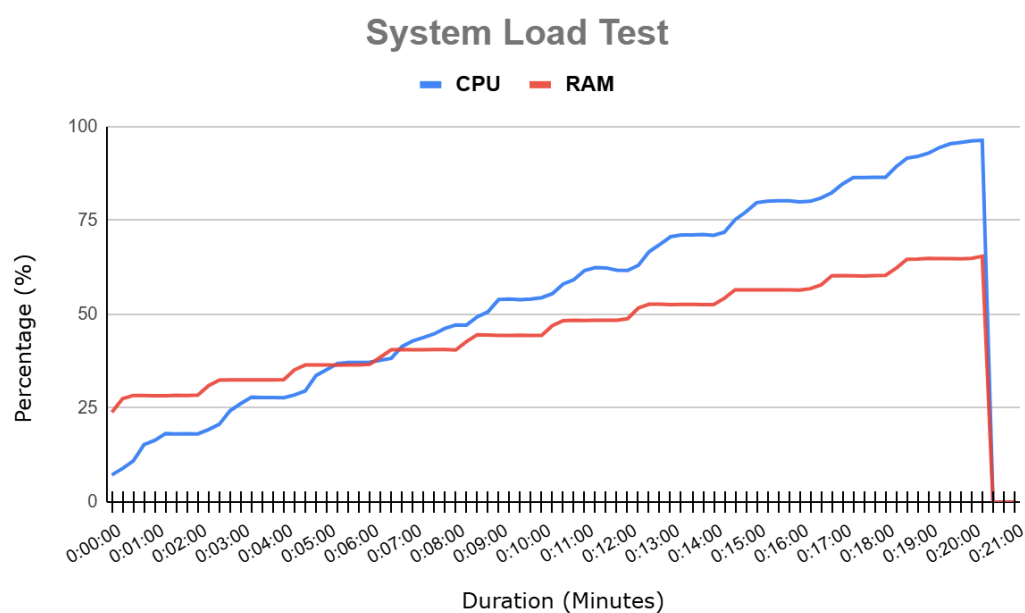
Kategori Fungsi	Hasil yang diharapkan	Status		Kendala
		Berhasil	Gagal	
User berhasil mendapatkan informasi paket data dari hasil <i>sniffing</i> pada setiap komponen CU, DU, dan UE serta berhasil menyimpan <i>file</i> PCAP pada <i>database</i>	Muncul informasi paket data hasil sniffing di tab Sniff pada menu Wireshark, Notifikasi sukses ketika file PCAP berhasil disimpan pada database	✓		-
User berhasil mendapatkan informasi daftar file PCAP yang telah tersedia pada database	Muncul daftar file PCAP di tab Files pada menu Wireshark	✓		-

User berhasil men-download file PCAP yang diinginkan	Muncul pop up hasil download pada browser setelah menekan tombol download	✓		-
User berhasil menghapus file PCAP yang diinginkan dari database	Muncul notifikasi sukses untuk penghapusan file PCAP yang diinginkan	✓		-

Berdasarkan tabel di atas, seluruh kategori fungsi berstatus sukses, yang berarti tingkat keberhasilannya sebesar 100%. Hal ini menandakan bahwa proses integrasi frontend hingga ke backend pada fungsi Wireshark berjalan dengan benar dan telah memenuhi ekspektasi dari fungsi yang tersedia pada website dashboard yang dikembangkan.

Terlepas itu, adapun sedikit anomali yang terkadang muncul ketika menjalankan fungsi ini berkaitan dengan tampilan data yang bersifat real-time. Data tersebut terkadang tidak tertampil dengan benar pada saat proses sniffing berjalan, dan hal itu akan menjadi rencana improvisasi pengembangan untuk kedepannya.

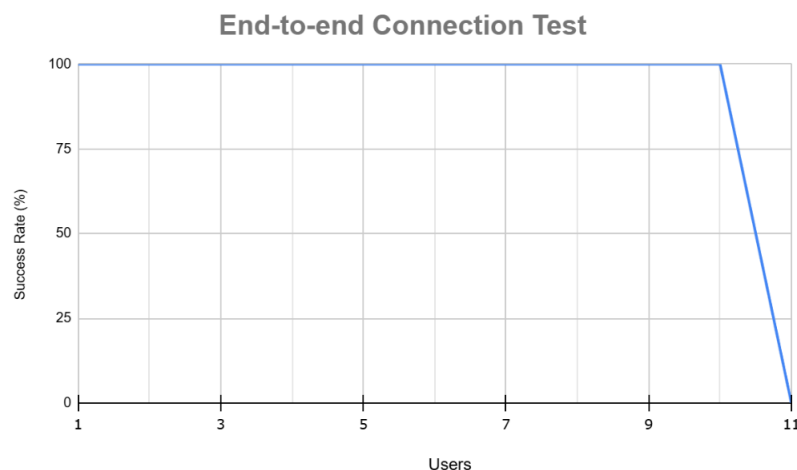
### 5.3.3 Hasil *Performance Testing*



**Gambar 0.68 Grafik System Load Capacity**

Berdasarkan grafik pada Gambar 5.68 ditunjukkan pengaruh peningkatan waktu terhadap persentase beban pada CPU dan RAM. Dengan meningkatnya durasi waktu, maka persentase beban yang diperoleh CPU dan RAM juga meningkat. Hal ini ditunjukkan dengan meningkatnya beban setiap menit karena menggunakan metode *Ramp Up Time testing* yang akan meningkatkan 1 *user* per menit ketika kondisi masih terpenuhi.

Pada menit ke-11, CPU dan RAM mencapai beban tertingginya sehingga dimulai *user* 11 sudah tidak dapat terkoneksi ke 5G core. Sedangkan pada menit ke 10, *user* masih dapat terkoneksi ke 5G core dan mendapatkan alamat IP serta mengakses internet. Referensi grafik dapat dilihat pada Gambar 5.69.



**Gambar 0.69 Grafik End-to-End Connection**

Berdasarkan grafik pada Gambar 5.69 ditunjukkan pengaruh jumlah *user* terhadap *success rate end-to-end connection*. Dapat dilihat pada *user* 1 hingga 10 memiliki *success rate* 100%, artinya UE sudah mendapatkan alamat IP dan sudah bisa terhubung ke internet. Sedangkan dimulai pada *user* 11, UE sudah tidak bisa terkoneksi *end-to-end* yang dipengaruhi oleh System Load Resource. Akibatnya, dimulai dari *user* 11 UE sudah tidak mendapatkan alamat IP sehingga tidak dapat mengakses internet.

Pada pengujian *performance testing* yang dilakukan seperti *System Load Test* dan *End-to-End Connection*, dapat dianalisa bahwa klaster mampu membuat lebih dari 1 koneksi 5G *end-to-end*. Berdasarkan grafik, klaster mampu menangani hingga 10

*user*. Artinya, klaster dapat *handle* 10 *end-to-end connection* secara bersamaan dengan parameter-parameter RAN yang berbeda.

Berdasarkan Tabel 5.12, dapat diambil data bahwa penggunaan *resource* per satu *user* adalah 2781mCPU dan 1965MiB RAM. Dengan mempertimbangkan toleransi 5%, kebutuhan CPU dan RAM per *user* meningkat menjadi 2920mCPU dan 2063MiB RAM. Penambahan toleransi ini memastikan bahwa sistem memiliki kapasitas yang cukup untuk menangani lonjakan beban sementara. Kemudian, angka-angka tersebut dibulatkan agar lebih mudah digunakan dalam perencanaan sumber daya di masa depan. Hasil akhir dapat dilihat pada Tabel 5.29.

**Tabel 0.29 Total Kebutuhan Per User**

Resource	Kebutuhan Per User (KPU)	Toleransi 5%	Pembulatan KPU
CPU	2781 mCPU	2920 mCPU	3000 mCPU
RAM	1965 MiB	2063 MiB	2100 MiB

Tabel 5.29 menunjukkan hasil angka akhir setelah dilakukan pembulatan. Berdasarkan angka ini, pengguna dapat menentukan alokasi *resource* yang digunakan dengan menggunakan rumus berikut ini:

$$Total\ CPU = (Total\ user \times KPU) \quad (0.2)$$

$$Total\ RAM = (Total\ user \times KPU) \quad (0.3)$$

Jika ingin menentukan jumlah *user* yang dapat digunakan oleh sebuah *resource*, dapat menggunakan rumus berikut ini:

$$Total\ user = \frac{(Total\ resource - Idle\ resource)}{KPU} \quad (0.4)$$



Rumus dapat di kalkulasikan pada *resource* CPU ataupun RAM. Agar sistem dapat berjalan dengan baik, hasil dari rumus *total user* sebaiknya dilakukan pembulatan ke bawah dan ambil nilai terkecil dari hasil perhitungan total user berdasarkan KPU CPU atau KPU RAM.

Berdasarkan spesifikasi yang diimplementasikan pada Tabel 4.1, total penggunaan sumber daya dari klaster Kubernetes yang digunakan adalah 40 CPU dan 56 GiB RAM. Berdasarkan hasil pengujian kapasitas beban sistem dan koneksi *end-to-end*, klaster kami mampu menangani hingga 10 *user*. Dengan demikian, jika dihitung menggunakan rumus yang telah dijelaskan dan dikurangi dengan *resource idle system*, hasil perhitungan ini telah terbukti akurat. Rencana pengembangan berkelanjutan memerlukan optimasi *resource* untuk setiap komponen yang telah dijalankan, terutama untuk komponen dengan kebutuhan *resource* tinggi seperti DU dan UE. Optimasi ini sangat penting untuk memastikan penggunaan CPU dan RAM yang lebih efisien, sehingga dapat mendukung lebih banyak *user* dan meningkatkan skalabilitas aplikasi.

#### 5.3.4 Hasil User Acceptance Testing

##### 5.3.4.1 Hasil Functional Testing

Pada pelaksanaan UAT ini, *user* mencoba semua fitur pada *dashboard* seperti *configuration panel* untuk mengkonfigurasi tiap komponen 5G, *topology graph* untuk melihat apakah tiap komponen 5G sudah berhasil terkoneksi, dan fitur-fitur lainnya. Pengujian ini berfungsi untuk mengetahui apakah fitur sudah berjalan sempurna atau belum dari sisi *user*. Dapat dilihat kembali pada tabel 5.14, semua fitur pada halaman *dashboard* admin berhasil berfungsi seperti yang diharapkan. Admin dapat membuat akun *user* baru, mengganti *password* akun *user*, dan menghapus akun *user* tanpa ada kendala ataupun *error*.

Pada halaman *user*, hampir semua fitur berhasil berfungsi secara sempurna. Masih ada beberapa fitur yang terdapat *bug* atau dijalankan secara bersamaan mengalami *error*. Fitur tersebut yaitu fitur *protocol stack* dan *logs* pada *sidebar* tiap komponen di *topology graph*. Fitur ini berhasil berfungsi menampilkan *protocol stack* dan *logs* tiap komponen, tapi terkadang hasil yang ditampilkan hanya ‘null’. Selain itu, ada fitur *sniffing wireshark* yang hanya bisa melakukan *sniffing* paket data satu komponen saja. Hal ini disebabkan oleh *load* yang tinggi di VM di mana kami men-develop *dashboard* tersebut. Semua fitur dapat

berjalan sempurna ketika hanya diakses oleh satu *user* saja. Namun ketika banyak *user* mengakses *dashboard* secara bersamaan, beberapa fitur berjalan tidak sempurna.

Fitur *sniffing* pada wireshark membutuhkan waktu yang lama dikarenakan proses *sniffing* berpusat pada *backend*. Proses *sniffing* ini lama dikarenakan memproses data yang telah difilter. Untuk mengatasi masalah ini, proses *sniffing* dipindah ke pod tiap pengguna. Lalu metode *sniffing* yang digunakan pun berbeda. Sebelumnya metode yang digunakan yaitu semua data hasil *sniffing* diambil terlebih dahulu, baru setelah itu data tersebut difilter sesuai dengan protokol yang diinginkan. Metode yang digunakan sekarang yaitu data difilter terlebih dahulu, baru setelah itu data hasil *sniffing* ditampilkan.

#### 5.3.4.2 Hasil Usability Testing

*Usability testing* digunakan untuk dapat mengetahui dan mengevaluasi pengalaman pengguna ketika menggunakan *dashboard*. Oleh karena itu, pengujian ini menggunakan kuesioner SUS yang diisi oleh pengguna setelah mencoba semua fitur pada *dashboard*. Dari data yang didapat, dilakukan analisis perhitungan sesuai dengan ketentuan analisis data SUS. Hasil perhitungan data SUS dari UAT 1 mendapatkan nilai rata-rata 65,42. Nilai ini masih di bawah nilai minimal SUS yaitu 68. Namun masih dalam kategori “OK” atau dapat diterima, meskipun perlu perbaikan hingga mencapai nilai minimal.

Terdapat beberapa pertanyaan yang mendapatkan nilai rendah pada kuesioner SUS ini. Dari pertanyaan bernilai rendah ini, kita dapat mengetahui hal-hal yang perlu ditingkatkan yaitu *user* merasa masih banyak hal yang tidak konsisten, *user* masih mengalami hambatan, *user* butuh panduan dan perlu membiasakan diri terlebih dahulu dalam menggunakan *dashboard*.

Pada kuesioner SUS juga terdapat kolom kesan dan saran yang diisi oleh *user*. Berikut adalah kesimpulan dari pendapat dan saran *user* terkait aplikasi:

1. Perlu ditingkatkan dan dioptimalkan fitur yang masih belum berfungsi dengan baik
2. Tampilan UI masih belum responsive terhadap perubahan ukuran layar

Sehingga dapat disimpulkan pada pengujian UAT 1 ini masih terdapat beberapa fitur yang belum berjalan dengan sempurna seperti yang dijelaskan pada sub bab Hasil *Functional Testing*. Selain itu juga tampilan UI masih belum responsive. *Feedback* dari pengguna membantu kami untuk mengetahui apa saja yang perlu diperbaiki dan ditingkatkan pada *dashboard*.

Setelah melakukan beberapa perbaikan pada *dashboard* berdasarkan *feedback* dan pengalaman pengguna pada UAT 1, selanjutnya kami melaksanakan UAT 2 untuk

mengetahui *feedback* pengguna terhadap aplikasi kami yang sudah diperbaiki. Nilai akhir dari data perhitungan SUS di UAT 2 ini yaitu 77.1. Skor SUS yang didapat mengalami peningkatan dan berhasil di atas nilai minimal 68. Skor SUS pada UAT 2 ini juga masuk ke kategori *good to excellent*.

Dari enam pertanyaan tambahan, dapat disimpulkan bahwa fitur-fitur pada dashboard kami dapat menambah pengetahuan dan *skill* pengguna mengenai 5G RAN. Nilai rata-rata tiap pertanyaan adalah  $\geq 4$  dari skala 1-5 di mana nilai 1 berarti “Sangat Tidak Setuju” dan nilai 5 berarti “Sangat Setuju”. Pengguna rata-rata setuju dengan enam pertanyaan tersebut. Analisis perhitungan untuk enam pertanyaan tambahan ada di lampiran 5.7

**Tabel 0.30 Nilai Rata-Rata Tiap Pertanyaan**

No	Pertanyaan	Nilai Rata-Rata
1	Saya merasa fitur <i>Graphical User Interface</i> (GUI) pada ORCA <i>dashboard website</i> memudahkan saya dalam konfigurasi komponen RAN 5G (CU, DU, dan UE)	4,4
2	<i>Lab guidance</i> dan materi Open RAN 5G pada halaman Introduction membantu saya dalam memahami Open RAN 5G dan parameter-parameter konfigurasi 5G RAN	4,4
3	Visualisasi <i>topology graph</i> membantu saya memahami arsitektur 5G	4,5
4	Saya dapat mempelajari protokol apa saja pada jaringan 5G dan dapat menganalisis paket data jaringan melalui fitur Wireshark	4,2
5	Saya dapat memahami <i>Key Performance Indicator</i> (KPI) apa saja yang terdapat pada komponen UE pada fitur monitoring	4,1
6	Seberapa besar anda akan merekomendasikan ORCA <i>dashboard website</i> ini kepada orang lain?	4,7

## 5.4 Kesimpulan

Penelitian ini menyoroti beberapa tantangan utama dalam infrastruktur teknologi jaringan 5G konvensional yang mempengaruhi efisiensi, fleksibilitas, dan reliabilitas sistem. Dari aspek manajemen, pengelolaan perangkat secara terpisah meningkatkan risiko *human error* dan kesalahan konfigurasi sehingga mengurangi efisiensi dan reliabilitas. Pada aspek kontrol, keterbatasan dalam pengumpulan metrik dan pemantauan anomali perangkat secara *real-time* menimbulkan kendala signifikan terhadap efisiensi sistem secara keseluruhan. Selain itu, baik perusahaan maupun institusi pendidikan menghadapi keterbatasan sumber daya yang menghambat upaya untuk mengadopsi dan memahami teknologi 5G secara efektif dalam skala kecil untuk keperluan penelitian dan pendidikan.

Untuk mengatasi permasalahan ini, ORCA (Open RAN Configuration Application) diperkenalkan sebagai platform pelatihan berbasis *dashboard* simulasi yang membantu pengguna dalam memahami dan mengkonfigurasi komponen 5G RAN secara sistematis dengan berbagai tingkatan yang meningkatkan pemahaman kasus penggunaan. ORCA menyediakan fitur tambahan seperti monitoring dan *sniffing* yang membantu dalam pemahaman dan konfigurasi komponen-komponen pada 5G RAN. Sistem ORCA terdiri dari empat subsistem, yaitu *frontend*, *backend*, Infrastruktur, dan 5G, dengan fitur-fitur utama seperti grafik topologi, fitur *sniffing*, konfigurasi komponen 5G, dan *user management*.

Pengujian sistem yang dilakukan terdiri dari *API testing*, *End-to-End (E2E) testing*, *performance testing*, dan *user acceptance testing*. *API testing* bertujuan untuk memastikan bahwa setiap WebSocket API dan REST API berjalan sesuai dengan yang diharapkan. *End-to-End testing* bertujuan untuk memastikan bahwa seluruh alur aplikasi, mulai dari *frontend* hingga *backend*, bekerja secara harmonis dan sesuai dengan kebutuhan pengguna akhir. Lalu, *performance testing* adalah langkah penting dalam mengukur dan menilai batas kapasitas keseluruhan aplikasi serta performanya di bawah *load* tinggi. Sedangkan UAT dilakukan untuk menilai kepuasan pengguna terhadap aplikasi dan memastikan aplikasi memenuhi kebutuhan mereka.

Berdasarkan *API testing*, pengujian dilakukan menggunakan Postman dan Visual Studio Code dengan hasil yang diharapkan dari pengujian ini adalah semua API berfungsi sesuai spesifikasi, interaksi antar modul tidak menghasilkan kesalahan, dan sistem mampu menangani input yang tidak valid dengan tepat. Hasil yang didapatkan setelah pengujian

API *testing* adalah semua API pada *backend* dapat berfungsi sesuai dengan respon yang diharapkan.

Selanjutnya untuk *End-to-End testing* dilakukan menggunakan tampilan UI yang telah dikembangkan menggunakan web browser Google Chrome. Hasil yang diharapkan adalah keseluruhan alur aplikasi bekerja dengan harmonis sesuai dengan spesifikasi yang diinginkan. Dan hasil yang didapatkan adalah seluruh alur dapat berfungsi dengan baik dan benar sesuai dengan spesifikasi.

Kemudian dari sisi *performance testing* sesuai dengan implementasi spesifikasi yang sudah disediakan, dapat disimpulkan bahwa spesifikasi sumber daya yang telah disediakan mampu menampung lebih dari 1 koneksi E2E. Berdasarkan hasil pengujian, total user atau koneksi E2E yang dapat dibangun pada infrastruktur mampu mencapai 10 koneksi. Hasil benchmark menunjukkan rekomendasi penggunaan sumber daya yang dibutuhkan 1 *user* adalah 3000CPU dan 2400MiB RAM.

Pada pengujian *user acceptance testing*, terdiri dari *functional testing* and *usability testing*. Hasil *functional testing* pada UAT 1 menunjukkan hampir semua fitur berhasil berfungsi secara sempurna. Masih ada beberapa fitur yang terdapat *bug* atau dijalankan secara bersamaan mengalami *error*. Sedangkan *usability testing* pada UAT 1 memiliki nilai akhir 65,42 di mana nilai ini masih di bawah nilai standar SUS yaitu 68. Nilai ini masuk kategori “OK”, namun masih memerlukan perbaikan. Setelah melakukan perbaikan pada aplikasi dan melaksanakan UAT 2, nilai *usability testing* meningkat menjadi 77.1 dan masuk kategori *good to excellent*. Secara keseluruhan, *dashboard website* untuk mensimulasikan komponen 5G ini berguna untuk pengguna lebih memahami jaringan 5G.

Dari hasil beberapa pengujian yang telah dilakukan, dapat disimpulkan bahwa dashboard yang dikembangkan dari sisi *frontend* dan *backend* sudah berfungsi dengan baik. Dashboard ini mampu menangani hingga 10 user dengan artian mampu menangani hingga 10 koneksi 5G E2E yang berbeda dengan akumulasi. Kemudian berdasarkan pengujian UAT yang langsung dihadapkan kepada user, hampir semua fitur sudah berjalan dengan baik. Dapat disimpulkan bahwa *dashboard* yang telah dibuat memenuhi segala pengujian dan mampu menjawab permasalahan yang ada.

Dashboard sistem simulasi yang telah dikembangkan tentunya masih memerlukan rencana pengembangan lanjutan. Hal ini ditinjau dari beberapa pengujian yang masih dapat ditingkatkan dari segala sisi. Sehingga diharapkan dari pengembangan *dashboard* ini dapat menjadi referensi dan bisa meningkatkan hasil pengujian yang telah dilakukan. Kemudian

diharapkan hasil pengembangan *dashboard* ini dapat memajukan dunia akademik khususnya pada bidang telekomunikasi.

## Daftar Pustaka