

IF4051 Pengembangan Sistem IoT  
Laporan Tugas  
UTS



Ng Kyle  
13520040

Teknik Informatika  
Institut Teknologi Bandung

# Daftar Isi

<b>Daftar Isi</b>	<b>2</b>
<b>Deskripsi Fungsional Sistem</b>	<b>3</b>
Fungsional Sistem	3
User Journey	4
<b>Arsitektur IoT End-to-End System</b>	<b>8</b>
Arsitektur End-to-End System	8
Rancangan Hardware	9
Rancangan Software	11
Program ESP32	11
Aplikasi Backend	12
Aplikasi Merchant Reader	15
Aplikasi User App	17
Konfigurasi MQTT	20
<b>Pengujian Sistem</b>	<b>21</b>
Setup Pengujian	21
Pengujian Flow Registrasi	22
Pengujian Flow Pembayaran Berhasil	23
Pengujian Flow Pembayaran Gagal (Kurang Saldo)	24
Pengujian Flow Penambahan Saldo	26
Pengujian Flow Gagal Login (Akun Tidak Terdaftar dan Salah Credentials)	29
<b>Kesimpulan dan Saran</b>	<b>31</b>
Kesimpulan	31
Refleksi	31
Saran	32
<b>Daftar Referensi</b>	<b>33</b>
<b>Lampiran</b>	<b>34</b>

# Deskripsi Fungsional Sistem

## Fungsional Sistem

*Electronic Wallet* menjadi metode pembayaran yang umum digunakan pada era digital. Kemudahan dalam pemakaian yang efektif dan efisien menjadikan *Electronic Wallet* menjadi metode pembayaran yang digunakan dalam menggantikan uang dalam bentuk fisik. Umumnya, *Electronic Wallet* memiliki sebuah *hardware* yang berfungsi sebagai Input yang membaca informasi pengguna, yang dapat berupa masukan manual (pada sistem perbankan mobile apps) maupun secara fisik (berupa kartu e-money dan serupanya).

Dalam projek ini, dibuat sebuah sistem *Electronic Wallet* IoT End-To-End, yaitu dari akuisisi data melalui input (RFID), sistem komunikasi (MQTT), Data Processing (Backend), Merchant dan User Information (Frontend).

Fungsional Sistem yang dibentuk dapat dibagi menjadi beberapa bagian:

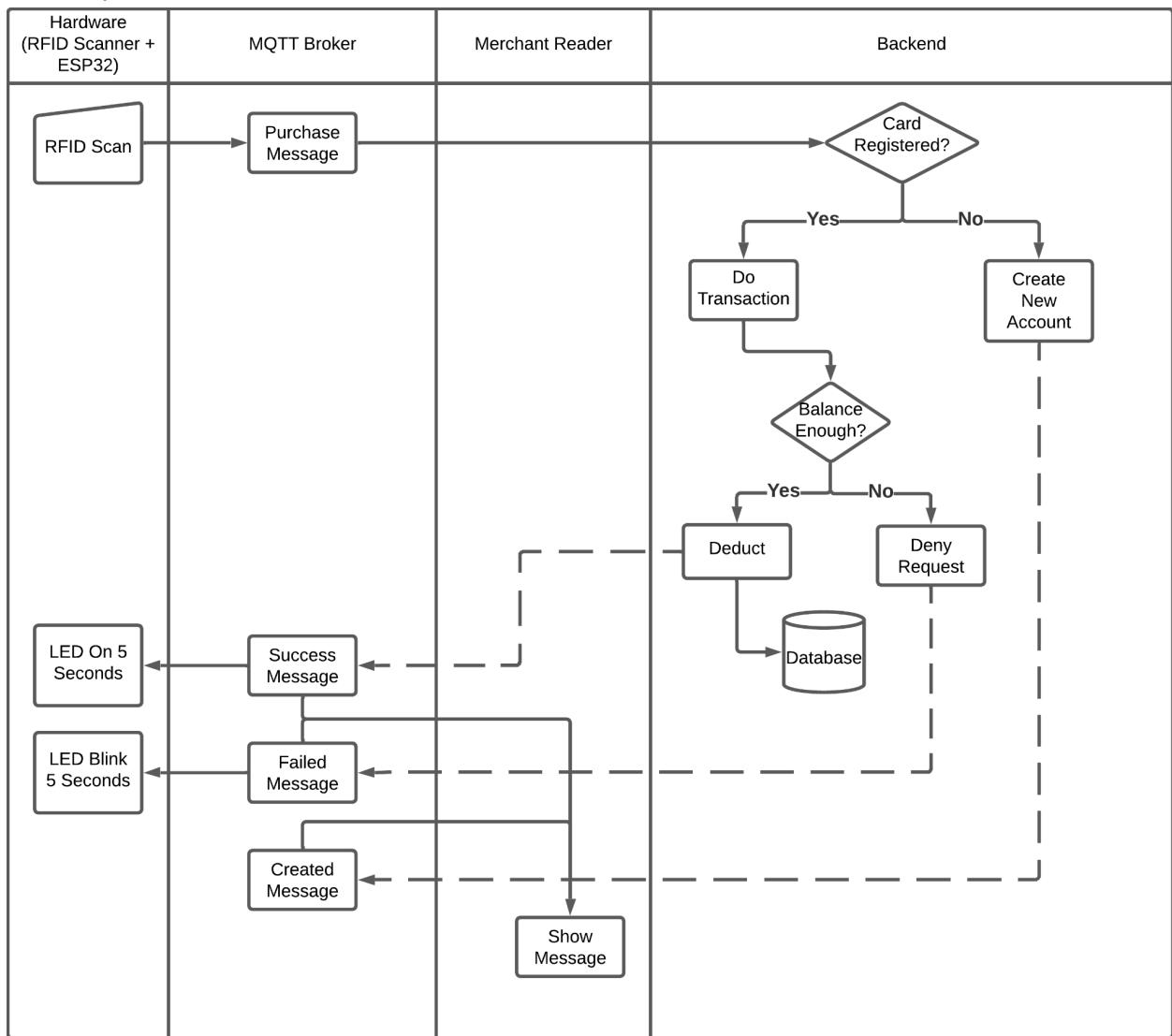
1. Perangkat RFID scanner bersama dengan ESP32 untuk membaca RFID dari pengguna (berupa kartu/tag).
2. LED Indicator pada ESP32 yang mengindikasikan kondisi dari pembayaran.
3. ESP32 sebagai perangkat yang memproses dan mengirimkan transaksi (bersama dengan RFID dan LED Indicator)
4. MQTT Message Broker dengan autentikasi client.
5. Sebuah Merchant Reader yang berfungsi menampilkan status pembayaran dalam bentuk teks dan informasi lainnya.
6. Sebuah User Application dengan autentikasi untuk mengakses informasi akun (saldo dan historis).
7. Integrasi fitur pembayaran dengan Payment Gateway (Midtrans) untuk Topup saldo pada User Application.

Setiap fungsional sistem akan dijelaskan lebih lanjut pada subbab Rancangan Hardware maupun Rancangan Software.

# User Journey

Secara umum berikut diuraikan beberapa user journey:

## A. User Payment

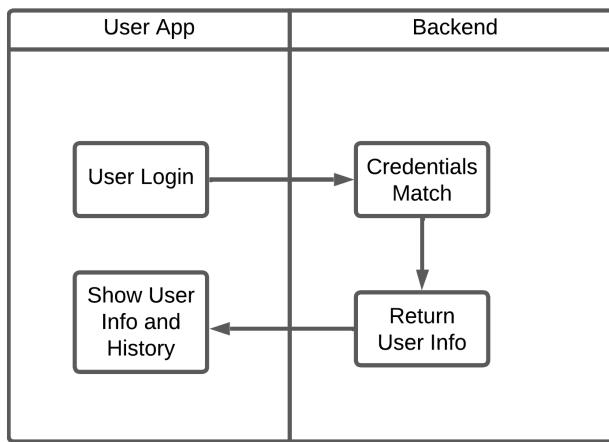


Gambar 1. User Journey Payment / Registrasi

1. User akan menginisiasi payment dengan menempelkan RFID pada RFID scanner.
2. Purchase Message dibangkitkan pada MQTT Broker
3. Backend membaca message purchase lalu memeriksa:
  - User memiliki kartu, maka akan proses transaksi

- User tidak memiliki kartu, maka akan dibuat akun baru, message created dikirim ke MQTT Broker dan ditampilkan pada Merchant Reader (terminate).
4. Bilamana akun ditemukan, akan dilakukan proses bersesuaian, yaitu:
    - Saldo user mencukupi, akan dideduct dan dikirimkan message sukses
    - Saldo tidak mencukupi, transaksi tidak dilakukan.
  5. ESP32 akan membaca message tersebut dan melakukan aksi pada LED:
    - Success Message, maka LED akan menyala selama 5 detik
    - Failed Message, maka LED akan blink selama LED
  6. Pesan Created, Failed, Success akan ditampilkan pada Merchant Reader selama 5 detik.

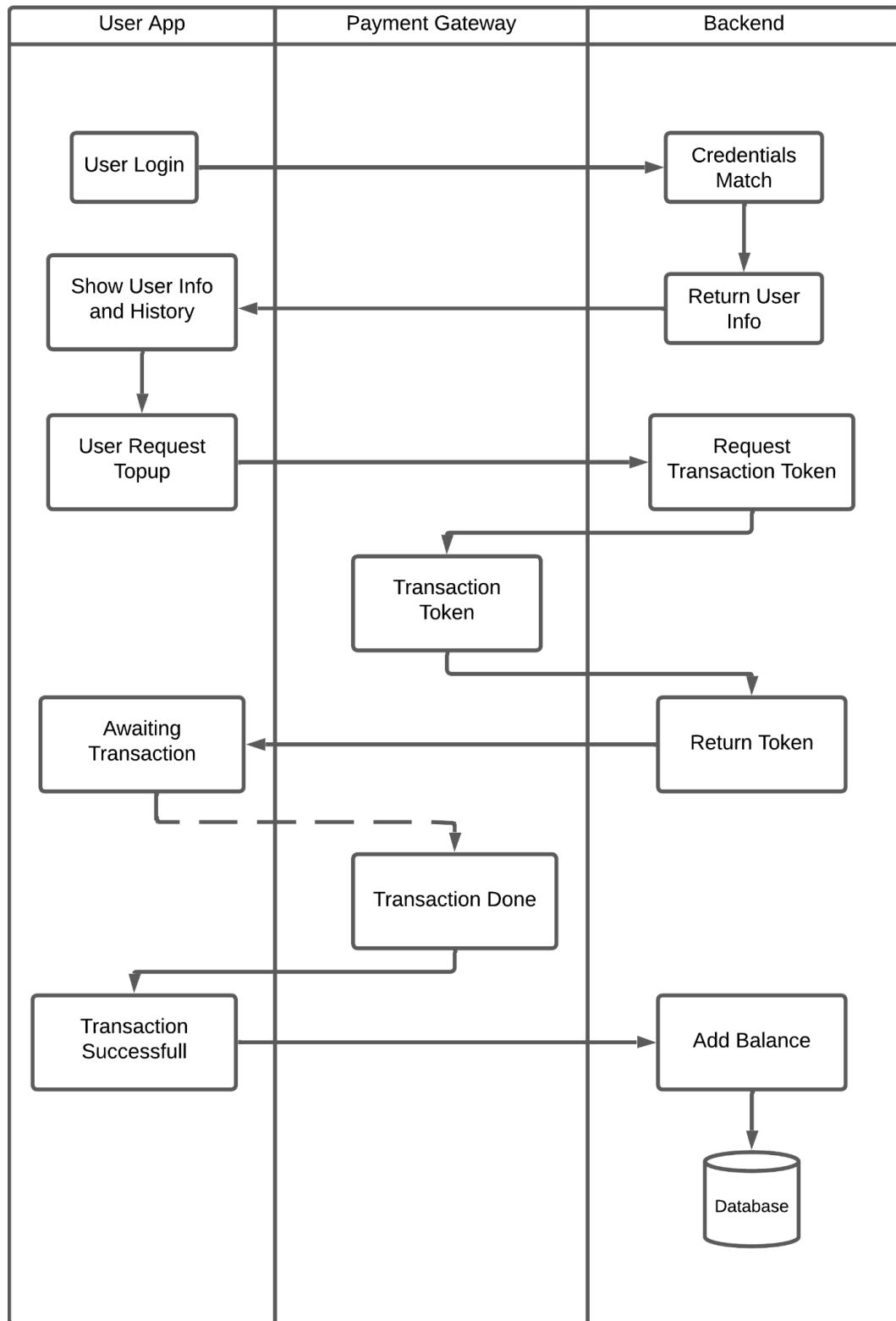
#### B. User Info



Gambar 2. User Info

1. User perlu login ke aplikasi User App
2. User App akan memeriksa apakah akun tersedia, lalu akan meminta data historis ke Backend aplikasi
3. Data akun dan historis ditampilkan

### C. User Topup

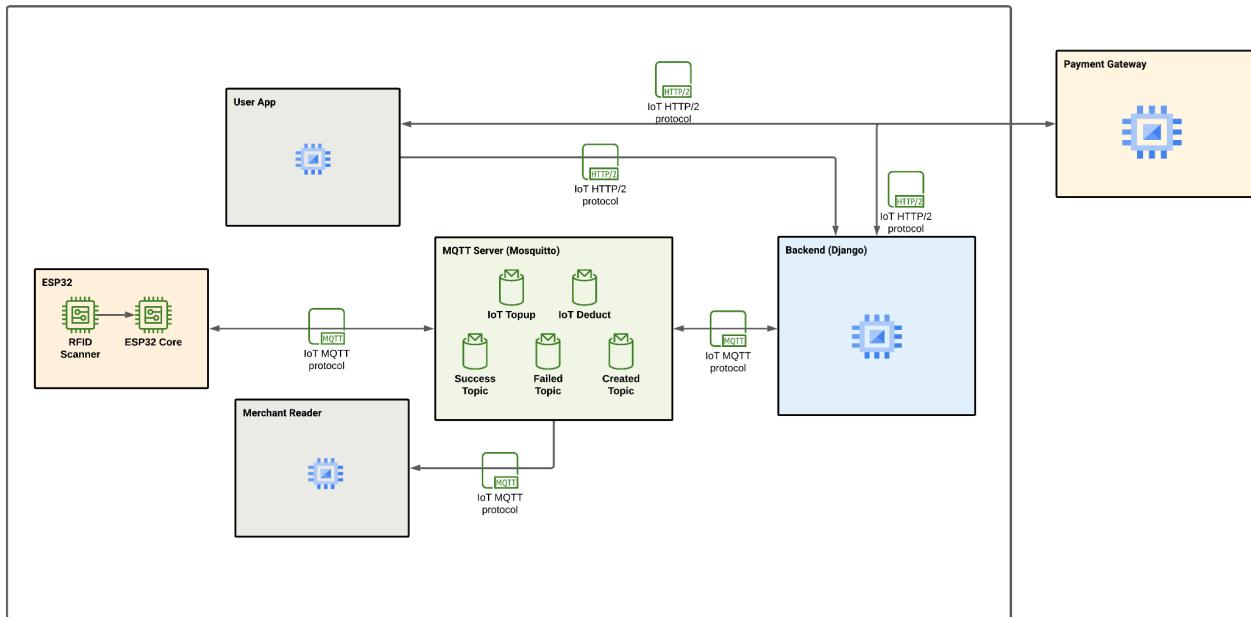


Gambar 3. User Topup

1. User Login ke aplikasi User App (User Journey B)
2. User Request Topup
3. Backend akan meminta token kepada Payment Gateway yang kemudian dikembalikan ke User App
4. User App akan menunggu transaksi berlangsung
5. Transaksi yang berhasil dilakukan akan ditambahkan ke balance user

# Arsitektur IoT End-to-End System

## Arsitektur End-to-End System



Gambar 4. End-to-End System

Sistem yang dibangun dapat dibagi menjadi beberapa layer:

- Layer Sensor (ESP32 dan RFID Reader)
- Layer User Interface (User App dan Merchant Reader)
- Layer Message (MQTT Broker)
- Layer Backend (Django)
- External System (Payment Gateway)

Setiap layer berkomunikasi dengan protokol yang sesuai

Tabel 1. Komunikasi Komponen

Komponen 1	Komponen 2	Protokol	Komunikasi
ESP32	MQTT Broker	MQTT	Topic Deduct, Success, Failed
Merchant Reader	MQTT Broker	MQTT (WS)	Topic Success, Failed, Created
Backend	MQTT Broker	MQTT	Topic Success, Failed, Created, Deduct

Backend	Payment Gateway	HTTP	Transaction
User App	Backend	HTTP	Transaction, User Info, Auth

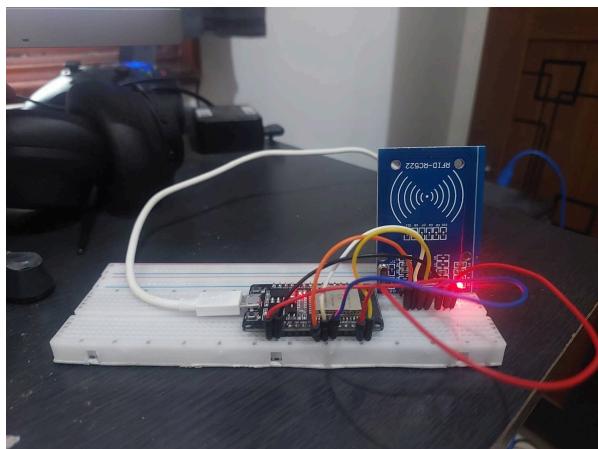
Setiap Topic memiliki tujuan tertentu (dengan format “iot/{Topic}”):

Tabel 2. Topik MQTT

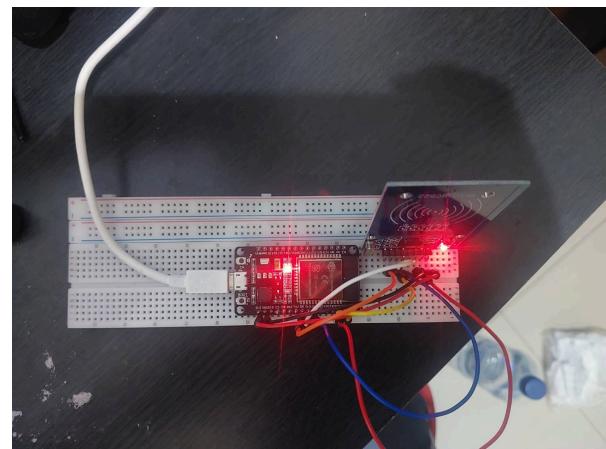
Topic	Deskripsi	Format Message / Isi
deduct	Transaksi terhadap sistem Electronic Wallet. Dibaca oleh Backend untuk memproses transaksi / registrasi akun baru.	Card UID
topup	Menambahkan saldo pada sistem Electronic Wallet. Tidak digunakan karena sistem memanfaatkan Payment Gateway, namun masih dapat dipanggil	Card UID
success	Informasi bahwa transaksi sukses	Pesan bebas
failed	Informasi bahwa transaksi gagal	Pesan bebas
created	Informasi bahwa akun terbentuk	Pesan bebas

## Rancangan Hardware

Hardware memanfaatkan ESP32 dan RC522.



Gambar 5a. Gambar Hardware (Horizontal)

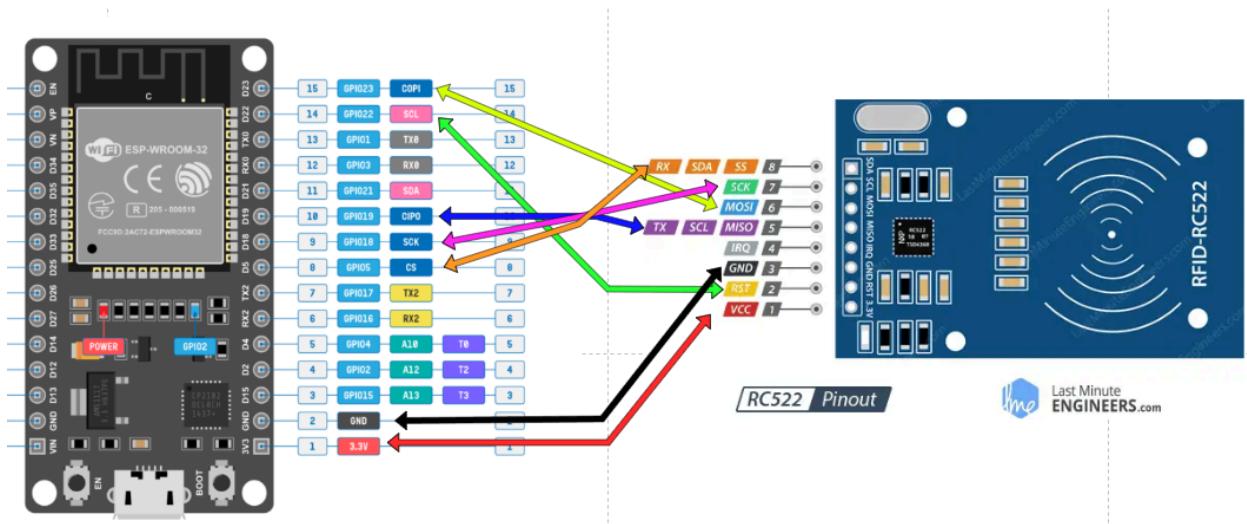


Gambar 5b. Gambar Hardware (Vertikal)

Skematik dari Hardware (koneksi ESP32 dengan RFID) diuraikan sebagai berikut:

Tabel 3. Skematik Hardware

RFID	ESP32
3.3V	3.3V
RST	22
GND	GND
MISO	19
MOSI	23
SCK	18
SDA	5



Gambar 6. Skematik Hardware

(Sumber dengan Perubahan: <https://www.circuitstate.com/> dan <https://lastminuteengineers.com/>)

IRQ tidak digunakan pada proyek ini sehingga tidak ditampilkan pada skematik.

# Rancangan Software

Seluruh dokumentasi penggerjaan dapat dilihat pada Lampiran Laporan Kerja. Berikut merupakan penjelasan rancangan sistem software.

## Program ESP32

Kode Sumber:

<https://github.com/Nk-Kyle/IoTPaymentSystem/blob/main/ESP/payment/payment.ino>

Pada Aplikasi ESP32, dimanfaatkan 2 Library tambahan untuk membantu penggerjaan yaitu:

- MFRC522 by GithubCommunity
- PubSubClient by Nick O'Leary

Program dibagi menjadi 7 prosedur berbeda selain daripada inisialisasi variable global.

Tabel 4. Prosedur Program ESP32

Prosedur	Penjelasan
setup()	Setup semua konfigurasi yang diperlukan agar device berjalan dengan baik: <ul style="list-style-type: none"><li>- PIN mode untuk LED</li><li>- Setup RFID (dan SPI)</li><li>- Setup dan koneksi WiFi</li><li>- Setup dan koneksi MQTT</li></ul>
loop()	Loop yang dilakukan terus menerus, dibagi menjadi beberapa bagian: <ol style="list-style-type: none"><li>1. Rekoneksi MQTT</li><li>2. MQTT client loop untuk callback message</li><li>3. Update waktu (millis)</li><li>4. Handle LED</li><li>5. Handle RFID</li></ol>
handleRFID()	Bagian kode yang dipanggil loop. Bertugas memeriksa adanya kartu yang baru ditempel dan mengirim request payment berdasarkan UID RFID yang dibaca.
handleLED()	Bagian kode yang dipanggil loop. Bertugas mengatur state LED berdasarkan kondisi payment (paymentState). <ul style="list-style-type: none"><li>- PAYMENT_SUCCESS : LED menyala selama 5 detik</li><li>- PAYMENT_FAILED : LED blinking selama 5 detik</li></ul>
sendPaymentRequest(St	Menerima String cardUID dan mengirimkan payment request ke

ring cardUID)	topik "iot/deduct"
callback(char *topic, byte *payload, unsigned int length)	Fungsi callback yang dipanggil bilamana adanya message baru pada topic-topic yang disubscribe. Berfungsi mengatur paymentState sesuai topic message baru
reconnect()	Rekoneksi dengan MQTT server

Pada ESP32 digunakan autentikasi terhadap MQTT server dengan user dan password. Selain itu perlu konfigurasi terhadap wifi. Berikut konfigurasi yang perlu diubah untuk menjalankan kode:

Tabel 5. Variable koneksi ESP32

Koneksi	Variable
MQTT	const char *mqtt_server const char *mqtt_topic const int mqtt_port const char *mqtt_user const char *mqtt_pwd
Wifi	const char *ssid const char *password

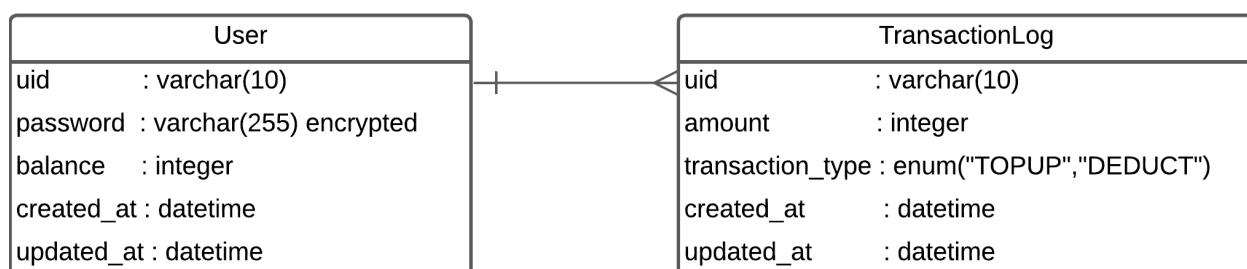
## Aplikasi Backend

Kode Sumber:

<https://github.com/Nk-Kyle/IoTPaymentSystem/tree/main/backend>

Backend menggunakan Django sebagai Framework utama. Database yang digunakan adalah SQLite dan dapat diubah menjadi server MySQL maupun PostgreSQL pada settings.py. Dengan memanfaatkan Django sebagai Framework, dengan mudah dapat mengkonfigurasi koneksi basis data, struktur basis data, serta admin panel.

Untuk persistensi data, digunakan skema basis data sebagai berikut:

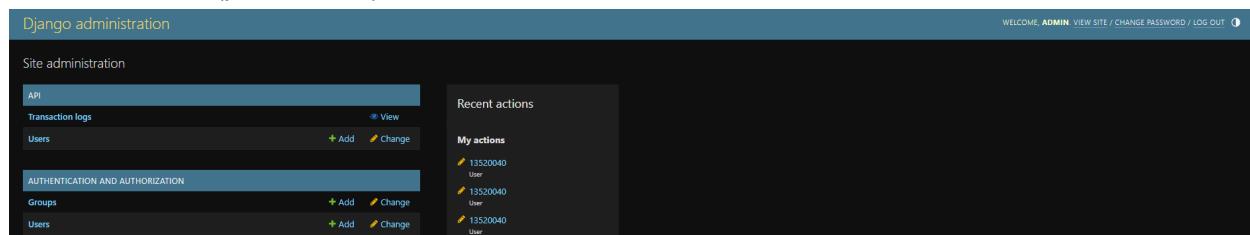


Gambar 6. ERD Basis Data

Untuk melakukan koneksi dengan MQTT, diperlukan metoda berbeda dari hal lazim pada Django karena umumnya memanfaatkan HTTP.

Konfigurasi client MQTT dilakukan paymentapp/mqtt.py dengan paho-mqtt, seluruh logic callback didefinisikan pada file tersebut. Client tersebut diinisialisasi pada paymentapp/\_\_init\_\_.py.

Untuk memudahkan tracing pada aplikasi, admin panel sudah didefinisikan. Berikut merupakan tampilan dari Admin Panel (pada /admin):



Gambar 7. Admin Panel /admin

Terdapat 2 halaman utama yang dapat dimanfaatkan:

- Transaction Logs ( /admin/api/transactionlog/ ) : melihat logs-logs yang tersimpan
- Users ( /admin/api/user/ ) : Melihat daftar user yang terdaftar

Select transaction log to view						
Action:	ID	CREATED AT	UPDATED AT	UID	AMOUNT	TRANSACTION TYPE
<input type="checkbox"/>	53	April 3, 2024, 3:23 p.m.	April 3, 2024, 3:23 p.m.	61FA7BE6	20000	Deduct
<input type="checkbox"/>	52	April 3, 2024, 3:23 p.m.	April 3, 2024, 3:23 p.m.	238872A9	20000	Deduct
<input type="checkbox"/>	51	April 3, 2024, 3:23 p.m.	April 3, 2024, 3:23 p.m.	238872A9	20000	Deduct
<input type="checkbox"/>	50	April 3, 2024, 1:41 p.m.	April 3, 2024, 1:41 p.m.	B3D8CSA6	20000	Topup
<input type="checkbox"/>	49	April 3, 2024, 12:22 p.m.	April 3, 2024, 12:22 p.m.	B3D8CSA6	20000	Deduct
<input type="checkbox"/>	48	April 3, 2024, 12:21 p.m.	April 3, 2024, 12:21 p.m.	238872A9	20000	Deduct

6 transaction logs

Gambar 8a. Admin Panel Transaction Log

Select user to change					
Action:	ID	CREATED AT	UPDATED AT	UID	BALANCE
<input type="checkbox"/>	7	April 3, 2024, 3:23 p.m.	April 3, 2024, 3:23 p.m.	61FA7BE6	80040
<input type="checkbox"/>	6	April 3, 2024, 12:22 p.m.	April 3, 2024, 1:41 p.m.	B3D8CSA6	100040
<input type="checkbox"/>	5	April 3, 2024, 12:21 p.m.	April 3, 2024, 3:23 p.m.	238872A9	40040

3 users

Gambar 8b. Admin Panel User

Terdapat 3 endpoint yang didefinisikan:

Tabel 6. Endpoint Backend

Endpoint	Method	Payload	Deskripsi
/api/auth/login/	POST	{ “uid”: “xxxxx”, “password”: “password” }	Login Aplikasi
/api/info/	GET	uid={uid}	Mendapatkan informasi pengguna (balance dan historis)
/api/topup/	POST	{ “uid”: “xxxxx”, “amount”: 20000 }	Melakukan topup untuk uid tertentu

## Aplikasi Merchant Reader

Kode Sumber:

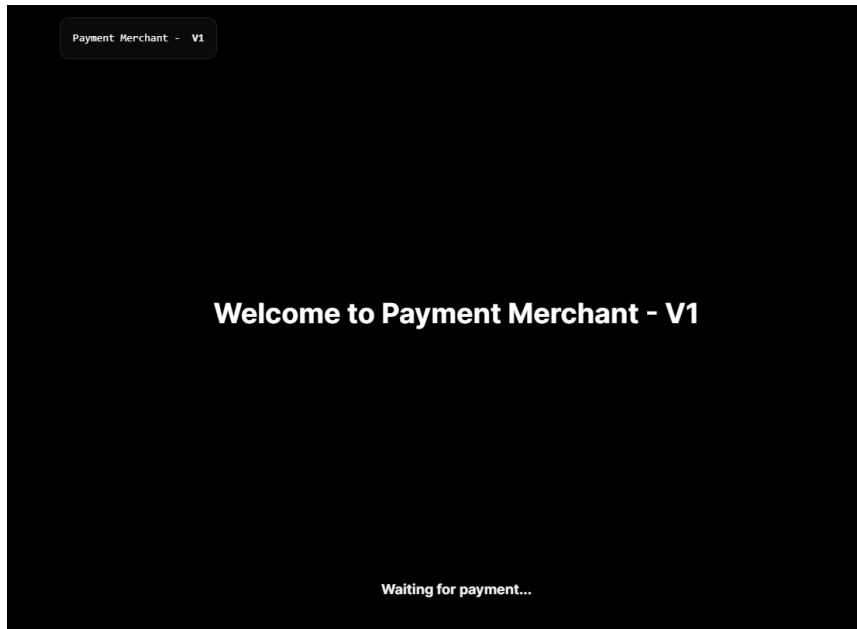
<https://github.com/Nk-Kyle/IoTPaymentSystem/tree/main/merchant-reader>

Aplikasi Merchant Reader berguna dalam menampilkan status pembayaran serta pembuatan akun baru. Aplikasi dibuat dengan framework Next.js dengan library tambahan mqtt untuk mengkoneksi dengan mqtt.

Untuk koneksi dengan mqtt, aplikasi hanya dapat berkomunikasi memanfaatkan protokol websocket (ws) sehingga server mqtt perlu membungkus komunikasi mqtt dengan protokol ws. Konfigurasi tersebut dilakukan pada merchant-reader/src/app/mqttListener.ts bertugas mengkoneksi dengan mqtt/websocket dan subscribe ke topic-topic relevan.

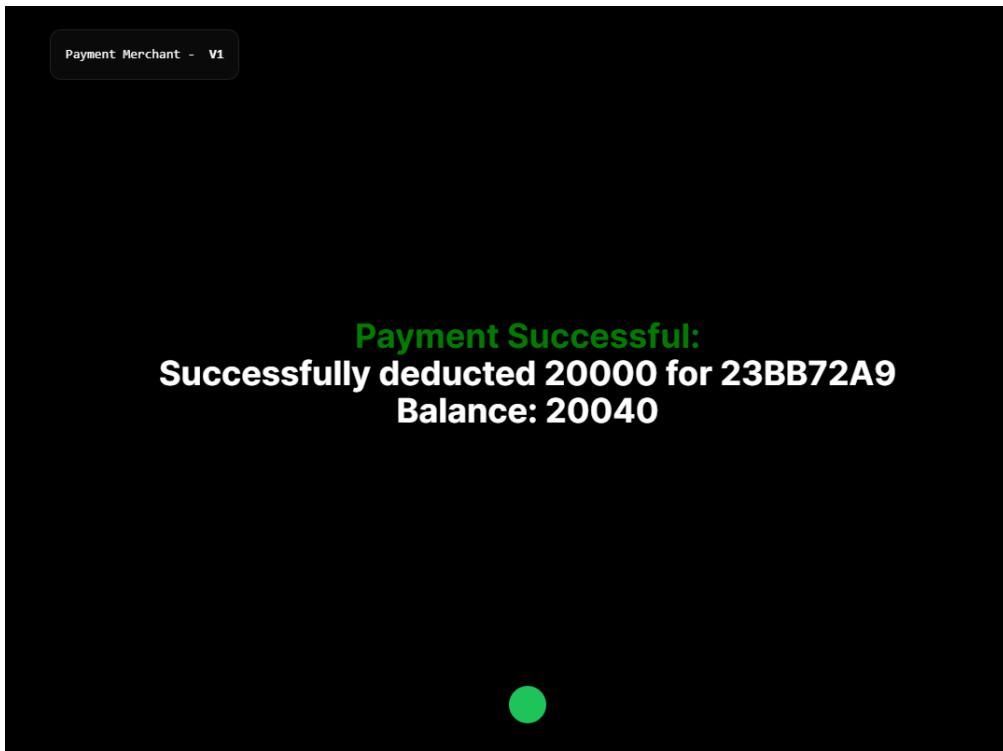
Setiap message pada topic dilakukan display. Berikut merupakan seluruh state yang mungkin ada pada aplikasi:

1. Aplikasi menunggu transaksi



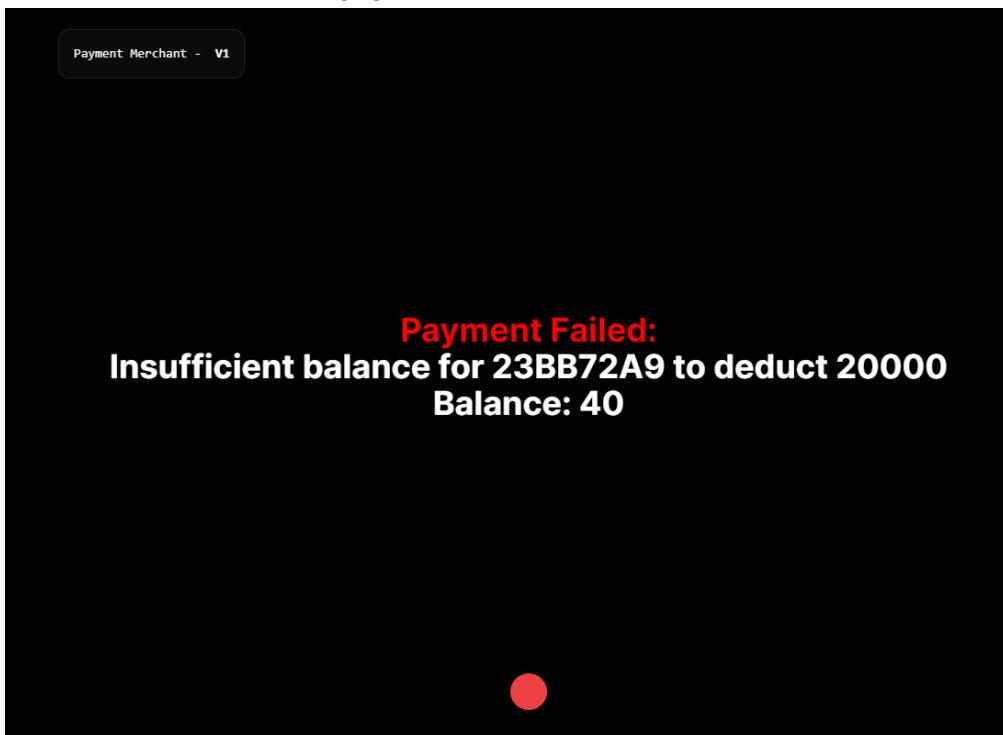
Gambar 9a. Merchant Reader Idle

2. Aplikasi mendapat pesan berhasil (iot/success)



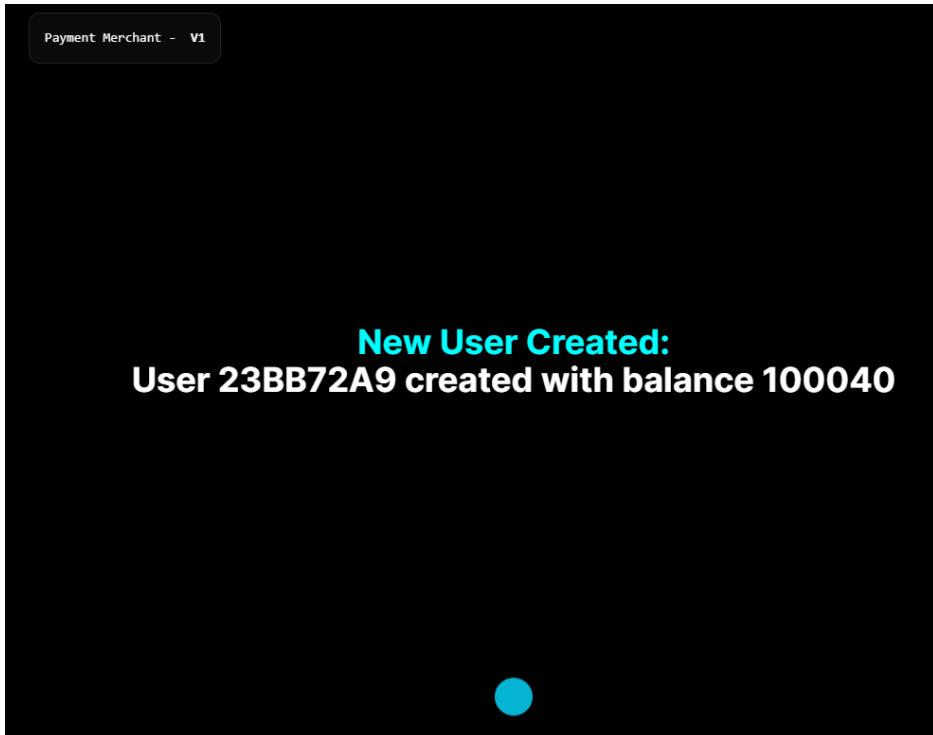
Gambar 9b. Merchant Reader Success

3. Aplikasi mendapat pesan gagal (iot/failed)



Gambar 9a. Merchant Reader Failed

4. Aplikasi mendapat registrasi (iot/created)



Gambar 9a. Merchant Reader Created

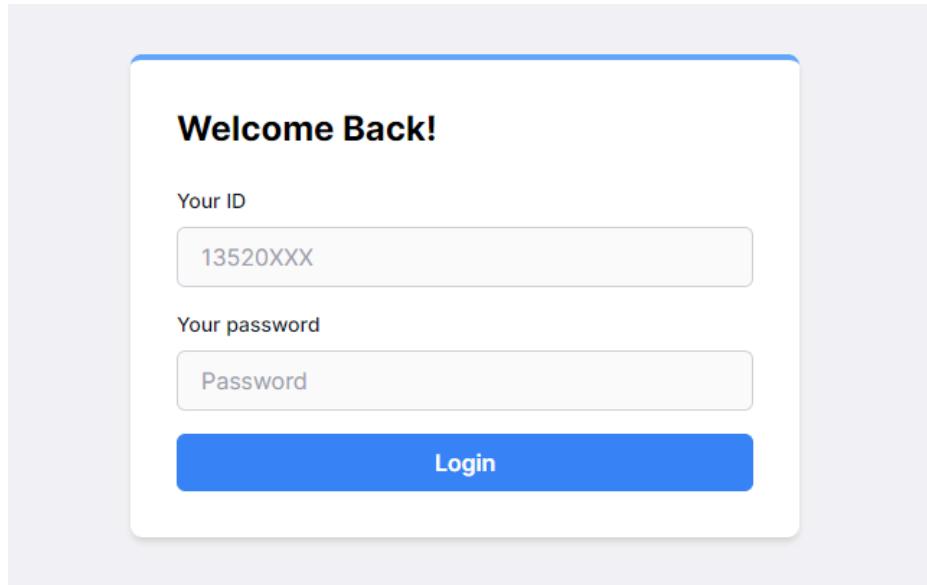
## Aplikasi User App

User App memiliki 2 halaman utama, yaitu Login Page (/) dan Dashboard Page (/dashboard). Seperti halnya dengan merchant reader, user app memanfaatkan Next.js sebagai frameworknya. Selain itu digunakan juga library Next-Auth dan Midtrans-Client yang masing-masing berfungsi untuk autentikasi dan transaksi payment gateway

Karena kebutuhan redirect oleh payment gateway midtrans setelah transaksi berhasil, maka diperlukan protokol https. Dalam hal ini digunakan ngrok.

A. Halaman utama (LoginForm.js)

Halaman ini menjadi halaman awal untuk user yang belum login. Terdapat 2 field yang perlu diisi user yaitu ID kartu serta password



Gambar 10. User App Login Page

Logika Autentikasi diatur dengan Next-Auth dan disimpan pada variable session pada sisi client. (Pada user-app/src/app/auth/[...nextauth]/route.js)

B. Halaman Dashboard (UserInfo.js)

Halaman ini berisikan detil user yaitu saldo dan historis transaksi user.

Terdapat juga fitur pembayaran yang diintegrasikan dan logout.

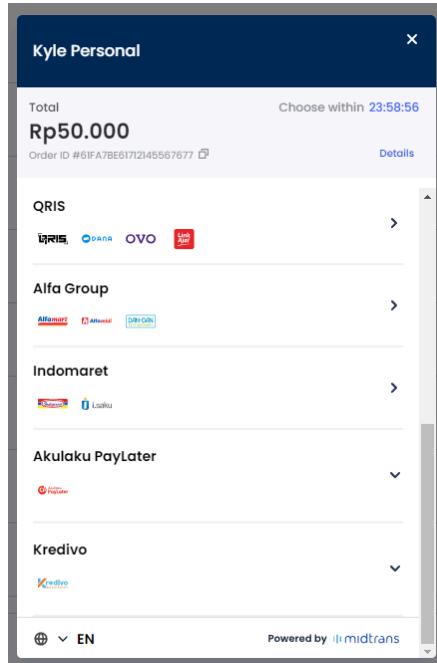
The screenshot shows a user dashboard with a header "Transaction History" and a "Logout" button. Below the header, it says "Welcome, 61FA7BE6" and "Balance: IDR 30040". There are three blue buttons for "Top-Up 20k", "Top-Up 50k", and "Top-Up 100k". The main area displays a list of transaction history entries:

- 03-04-2024, 18:54:02 IDR -20000
- 03-04-2024, 18:53:23 IDR 50000
- 03-04-2024, 18:49:49 IDR -20000
- 03-04-2024, 18:49:47 IDR -20000
- 03-04-2024, 18:49:40 IDR -20000
- 03-04-2024, 18:49:34 IDR -20000
- 03-04-2024, 18:49:26 IDR -20000

Gambar 11. User App Dashboard Page

#### C. Fitur Topup (api/tokenizer.js)

Topup dilakukan dengan klik salah satu opsi topup pada dashboard. Secara otomatis muncul popup berisi alternatif-alternatif pembayaran (bank, emoney, debit card, merchant, dan lainnya).



Gambar 12. User App Payment Window

Logic pada fitur ini diimplementasikan berdasarkan dokumentasi Payment Gateway Midtrans (<https://midtrans.com/en>).

## Konfigurasi MQTT

Mqtt Broker yang digunakan adalah mosquitto MQTT. Untuk aplikasi dapat berjalan, perlu konfigurasi khusus. Konfigurasi mosquitto yang digunakan dan didefinisikan pada mosquitto.conf adalah sebagai berikut

```
listener 1884
listener 8080

allow_anonymous false
password_file passwd.txt

protocol websockets
```

**listener 1884** dan **listener 8080** membuka port masing-masing untuk protokol mqtt dan websockets.

**allow\_anonymous false** dan **password\_file passwd.txt** mengkonfigurasikan bahwa perlu adanya autentikasi dalam koneksi terhadap server MQTT dengan credentials yang disimpan pada passwd.txt

**protocol websocket** membuka protokol websocket untuk mengakses server MQTT.

# Pengujian Sistem

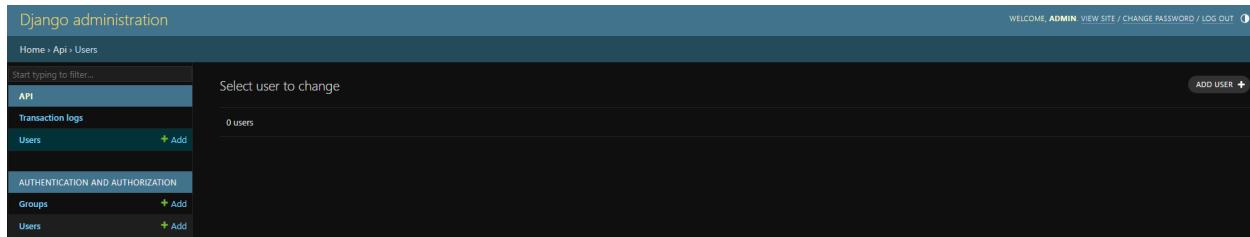
## Setup Pengujian

Sebagai basis pengujian, maka seluruh data dihapus dari basis data.

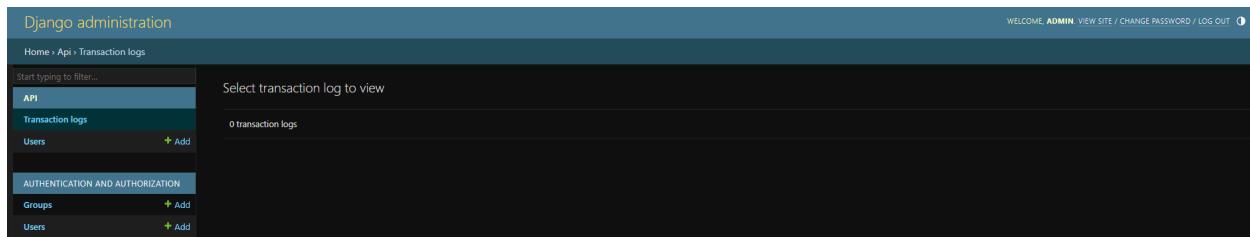
Setiap komponen perlu diaktivasi, yaitu:

1. MQTT Server (mosquitto)
2. ESP32 + RFID Scanner
3. Backend
4. Merchant Reader
5. User App

Berikut merupakan basis data setelah penghapusan semua data:



Gambar 13a. Basis data awal (User)



Gambar 13b. Basis data awal (Transaction Log)

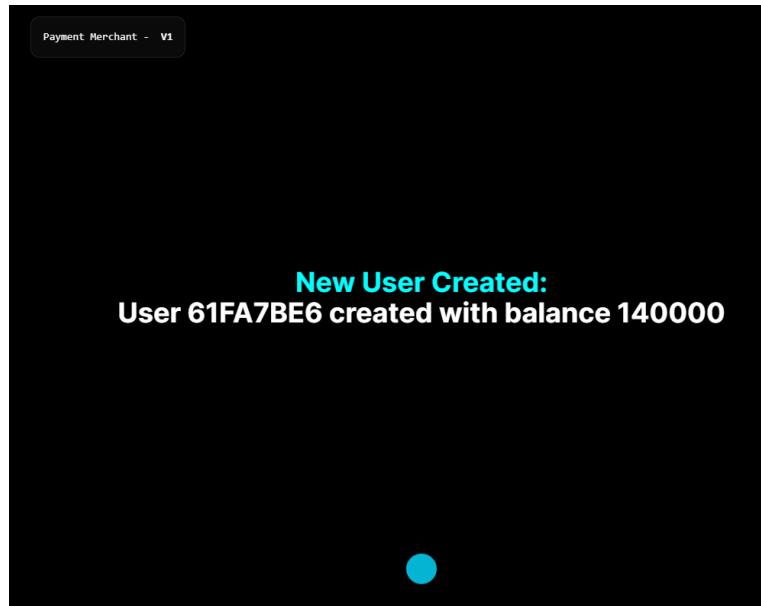
# Pengujian Flow Registrasi

Registrasi dilakukan dengan pertama-tama scan sebuah RFID pada RFID Scanner.

Ekspektasi:

1. Ditampilkan informasi Created pada Merchant Reader
2. Penambahan data User pada basis data dengan balance awal 140.000 (100.000 + 40.000)
3. Akses terhadap User App dengan UID RFID dan password default (password)

Hasil Uji:



Gambar 14a. Merchant Reader Registrasi User

ID	CREATED AT	UPDATED AT	UID	BALANCE
20	April 3, 2024, 7:44 p.m.	April 3, 2024, 7:44 p.m.	61FA7BE6	140000

Gambar 14b. Basis Data setelah Registrasi

## Transaction History

Logout

Welcome, 61FA7BE6

Balance: IDR 140000

Top-Up 20k Top-Up 50k Top-Up 100k

Gambar 14c. Tampilan User App setelah Login

#### Analisis:

- Merchant Reader berhasil membaca registrasi pengguna dan menampilkan UID kartu yaitu 61FA7BE6 dengan saldo awal 140.000
- Dapat terlihat pada basis data bertambah entri terbaru user UID 61FA7BE6 dengan saldo 140.000
- Pada User App dapat dilakukan login dengan UID 61FA7BE6 dan password password, menampilkan informasi user dan saldo user.

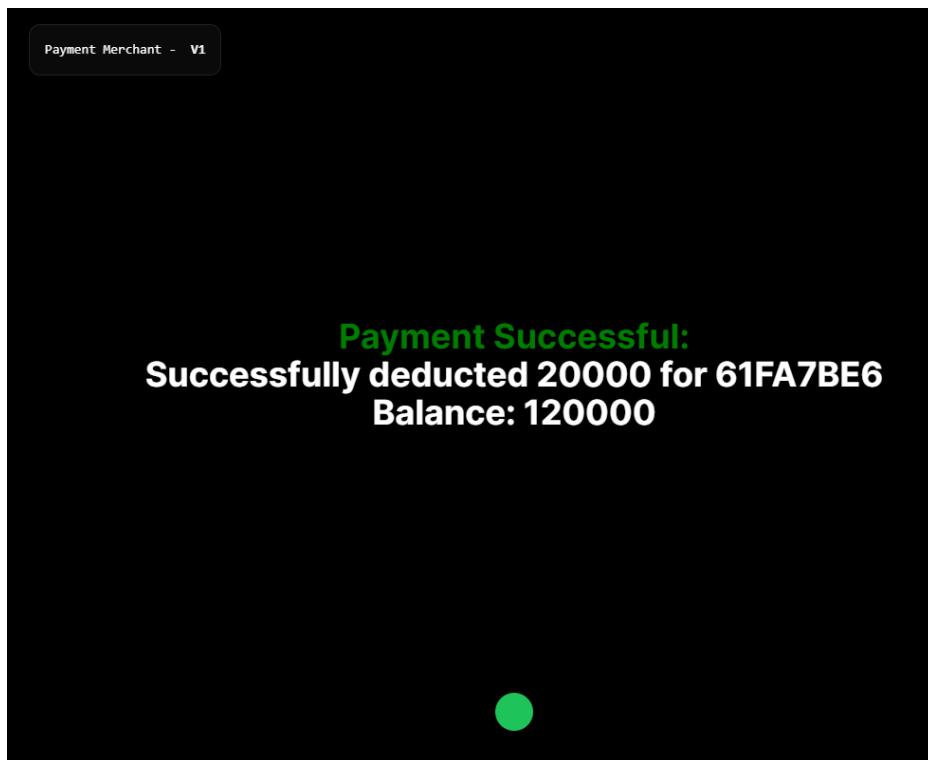
## Pengujian Flow Pembayaran Berhasil

Pembayaran dilakukan dengan scan terhadap RFID yang sudah teregistrasi sebelumnya.

#### Ekspektasi:

- Merchant Reader menampilkan pembayaran sukses dengan saldo berkurang sebesar 20.000 menjadi 120.000
- Tertambah Transaction Log pada basis data dan balance user berkurang
- User App menampilkan transaction log dan sisa balance (saldo)
- Lampu LED menyala selama 5 detik

#### Hasil Uji:



Gambar 15a. Merchant Reader Pembayaran Sukses

Select transaction log to view						
Action:	ID	CREATED AT	UPDATED AT	UID	AMOUNT	TRANSACTION TYPE
<input type="checkbox"/> 72	72	April 3, 2024, 7:45 p.m.	April 3, 2024, 7:45 p.m.	61FA7BE6	20000	Deduct
1 transaction log						

Gambar 15b. Basis Data Penambahan Transaction Log Pembayaran

Select user to change						
Action:	ID	CREATED AT	UPDATED AT	UID	BALANCE	ADD USER +
<input type="checkbox"/> 20	20	April 3, 2024, 7:44 p.m.	April 3, 2024, 7:45 p.m.	61FA7BE6	120000	
1 user						

Gambar 15c. Basis Data Pengurangan Saldo Pengguna

### Transaction History

Welcome, 61FA7BE6  
Balance: IDR 120000

[Top-Up 20k](#) [Top-Up 50k](#) [Top-Up 100k](#)

03-04-2024, 19:45:19  
IDR -20000

Gambar 15d. User App menampilkan data setelah pembayaran

#### Analisis:

- Merchant Reader berhasil membaca pembayaran yang sukses serta menampilkan UID kartu dan sisa balance 120.000.
- Dapat terlihat pada basis data bertambah entri terbaru pada Transaction Logs dengan Amount 20.000, Transaction Type Deduct, dan UID 61FA7BE6.
- Dapat terlihat pada basis data User dengan UID 61FA7BE6 memiliki saldo yang sudah berkurang menjadi 120.000
- Lampu LED berhasil menyala selama 5 detik

## Pengujian Flow Pembayaran Gagal (Kurang Saldo)

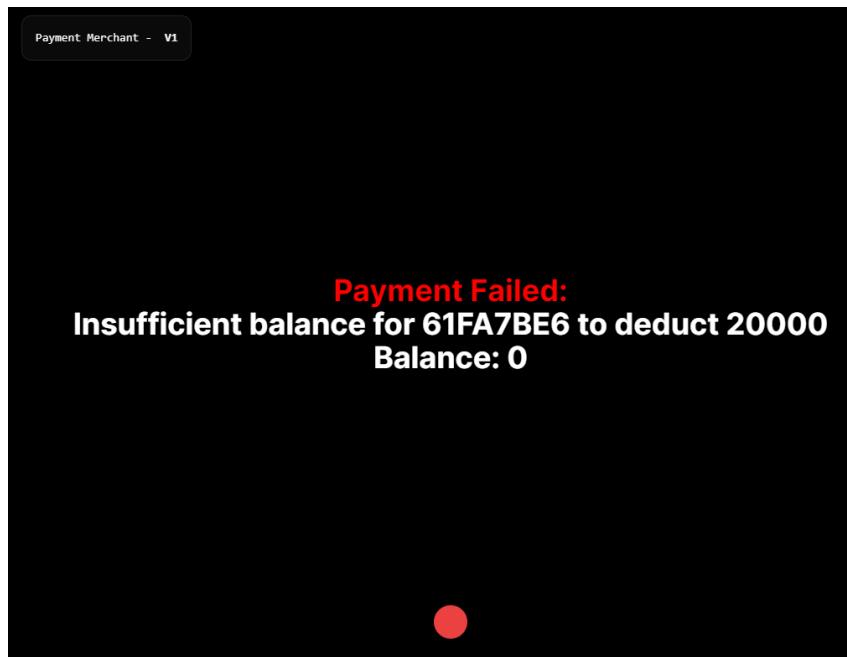
Pembayaran dilakukan dengan scan terhadap RFID dan dilakukan berulang hingga saldo habis/kurang

#### Ekspektasi:

- Merchant Reader gagal pada transaksi ke-5 karena saldo menjadi 40.000 yang kurang dari biaya transaksi 20.000

- Tertambah Transaction Log pada basis data sebanyak 6 kali dan balance user berkurang menjadi 0, total 7 entri
- User App menampilkan transaction log dan sisa balance (saldo) dengan total 7 entri pembayaran
- Lampu LED blink selama 5 detik setelah saldo habis

Hasil Uji:



Gambar 16a. Merchant Reader Pembayaran Gagal

The screenshot shows the Django Admin interface for the "Transaction logs" model. The title bar says "Django administration" and "Home > Api > Transaction logs". On the left, there's a sidebar with "Start typing to filter..." and links for "API", "Transaction logs", "Users", and "Groups". The main area has a heading "Select transaction log to view" and a table with columns: Action, ID, CREATED AT, UPDATED AT, UID, AMOUNT, and TRANSACTION TYPE. The table lists 7 rows of transaction logs, all of which are "Deduct" type and have a UID of "61FA7BE6" and an amount of "20000". The last row is dated April 3, 2024, 7:45 p.m.

Action:	ID	CREATED AT	UPDATED AT	UID	AMOUNT	TRANSACTION TYPE
<input type="checkbox"/>	78	April 3, 2024, 7:46 p.m.	April 3, 2024, 7:46 p.m.	61FA7BE6	20000	Deduct
<input type="checkbox"/>	77	April 3, 2024, 7:46 p.m.	April 3, 2024, 7:46 p.m.	61FA7BE6	20000	Deduct
<input type="checkbox"/>	76	April 3, 2024, 7:46 p.m.	April 3, 2024, 7:46 p.m.	61FA7BE6	20000	Deduct
<input type="checkbox"/>	75	April 3, 2024, 7:46 p.m.	April 3, 2024, 7:46 p.m.	61FA7BE6	20000	Deduct
<input type="checkbox"/>	74	April 3, 2024, 7:46 p.m.	April 3, 2024, 7:46 p.m.	61FA7BE6	20000	Deduct
<input type="checkbox"/>	73	April 3, 2024, 7:46 p.m.	April 3, 2024, 7:46 p.m.	61FA7BE6	20000	Deduct
<input type="checkbox"/>	72	April 3, 2024, 7:45 p.m.	April 3, 2024, 7:45 p.m.	61FA7BE6	20000	Deduct

7 transaction logs

Gambar 16b. Basis Data Transaction Log Setelah Kegagalan Transaksi

The screenshot shows the Django Admin interface for the "Users" model. The title bar says "Django administration" and "Home > Api > Users". On the left, there's a sidebar with "Start typing to filter..." and links for "API", "Transaction logs", "Users", and "Groups". The main area has a heading "Select user to change" and a table with columns: Action, ID, CREATED AT, UPDATED AT, UID, and BALANCE. The table lists 1 row of users, with ID 20, a UID of "61FA7BE6", and a balance of "0".

Action:	ID	CREATED AT	UPDATED AT	UID	BALANCE
<input type="checkbox"/>	20	April 3, 2024, 7:44 p.m.	April 3, 2024, 7:46 p.m.	61FA7BE6	0

1 user

Gambar 16c. Basis Data User setelah Kegagalan Transaksi

**Transaction History**

Welcome, 61FA7BE6  
Balance: IDR 0

Top-Up 20k   Top-Up 50k   Top-Up 100k

Date	Time	Amount	Status
03-04-2024	19:46:28	IDR -20000	Success
03-04-2024	19:46:24	IDR -20000	Success
03-04-2024	19:46:21	IDR -20000	Success
03-04-2024	19:46:19	IDR -20000	Success
03-04-2024	19:46:17	IDR -20000	Success
03-04-2024	19:46:15	IDR -20000	Success
03-04-2024	19:45:19	IDR -20000	Failure

Gambar 16d. Tampilan User App setelah Kegagalan Transaksi

Analisis:

- Merchant Reader berhasil membaca pembayaran hingga saldo kurang dari biaya transaksi
- Dapat terlihat pada basis data bertambah 6 entri baru untuk UID 61FA7BE6
- Dapat terlihat pada basis data User dengan UID 61FA7BE6 memiliki saldo yang sudah berkurang menjadi 0
- User App menampilkan 7 transaksi yang berhasil sebelum kurang saldo
- Lampu LED berhasil blink selama 5 detik

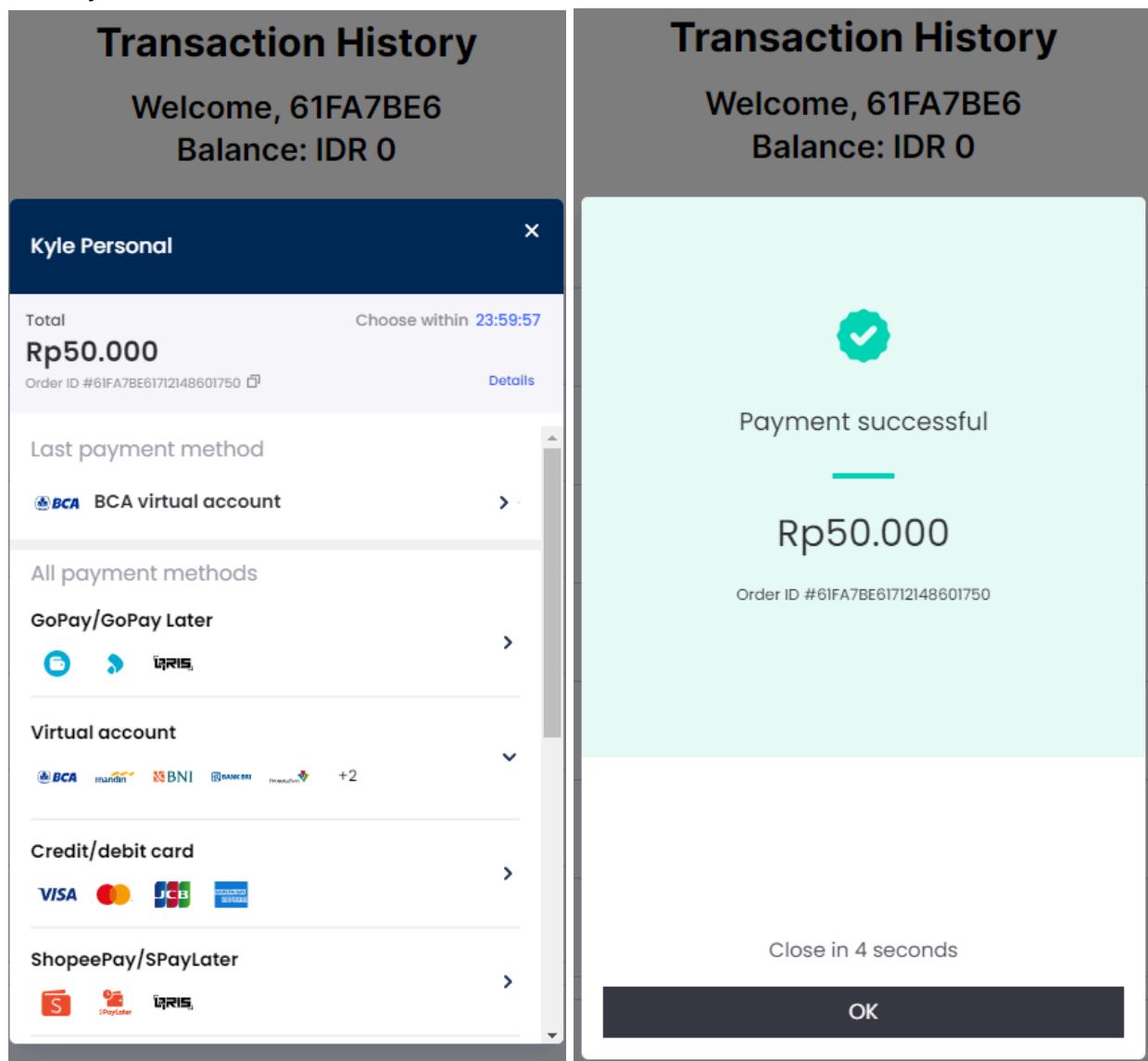
## Pengujian Flow Penambahan Saldo

Topup saldo dilakukan pada halaman dashboard, dengan Topup sebesar 50.000 melalui Payment Gateway.

Ekspektasi:

- Tertampil window pembayaran dan informasi pembayaran sebesar 50.000
- Pembayaran dilakukan dan dialihkan ke halaman terima kasih
- Penambahan pada admin Transaction Log berupa Topup 50.000
- Penambahan saldo User
- Penambahan saldo dan historis pada User App

Hasil Uji:



Gambar 17a. Pembayaran melalui Payment Gateway

# Thank you for your purchase!

Your order has been placed successfully.

[Go to Dashboard](#)

Gambar 17b. Halaman Terima Kasih setelah Pembayaran

The screenshot shows the Django Admin interface for the 'Transaction logs' model. The page title is 'Django administration' and the URL is 'Home > Api > Transaction logs'. On the left, there's a sidebar with links for API, Transaction logs, and Users. The main content area is titled 'Select transaction log to view' and contains a table with 8 rows of transaction data. The columns are: ID, CREATED AT, UPDATED AT, UID, AMOUNT, and TRANSACTION TYPE. The data shows various transactions, mostly 'Deduct' types, with one 'Topup' entry at the top. The 'AMOUNT' column shows values like 50000, 20000, etc.

Action:	ID	CREATED AT	UPDATED AT	UID	AMOUNT	TRANSACTION TYPE
<input type="checkbox"/>	79	April 3, 2024, 7:50 p.m.	April 3, 2024, 7:50 p.m.	61FA7BE6	50000	Topup
<input type="checkbox"/>	78	April 3, 2024, 7:46 p.m.	April 3, 2024, 7:46 p.m.	61FA7BE6	20000	Deduct
<input type="checkbox"/>	77	April 3, 2024, 7:46 p.m.	April 3, 2024, 7:46 p.m.	61FA7BE6	20000	Deduct
<input type="checkbox"/>	76	April 3, 2024, 7:46 p.m.	April 3, 2024, 7:46 p.m.	61FA7BE6	20000	Deduct
<input type="checkbox"/>	75	April 3, 2024, 7:46 p.m.	April 3, 2024, 7:46 p.m.	61FA7BE6	20000	Deduct
<input type="checkbox"/>	74	April 3, 2024, 7:46 p.m.	April 3, 2024, 7:46 p.m.	61FA7BE6	20000	Deduct
<input type="checkbox"/>	73	April 3, 2024, 7:46 p.m.	April 3, 2024, 7:46 p.m.	61FA7BE6	20000	Deduct
<input type="checkbox"/>	72	April 3, 2024, 7:45 p.m.	April 3, 2024, 7:45 p.m.	61FA7BE6	20000	Deduct

Gambar 17c. Basis Data Transaction Log setelah Topup

The screenshot shows the Django Admin interface for the 'Users' model. The page title is 'Django administration' and the URL is 'Home > Api > Users'. On the left, there's a sidebar with links for API, Transaction logs, and Users. The main content area is titled 'Select user to change' and contains a table with 1 row of user data. The columns are: ID, CREATED AT, UPDATED AT, UID, and BALANCE. The data shows a single user record with ID 20, created and updated at April 3, 2024, 7:44 p.m., and an updated balance of 50000. There's also a note '1 user' below the table.

Action:	ID	CREATED AT	UPDATED AT	UID	BALANCE
	20	April 3, 2024, 7:44 p.m.	April 3, 2024, 7:50 p.m.	61FA7BE6	50000

Gambar 17d. Basis Data User setelah Topup

[Logout](#)

### Transaction History

Welcome, 61FA7BE6  
Balance: IDR 50000

[Top-Up 20k](#) [Top-Up 50k](#) [Top-Up 100k](#)

03-04-2024, 19:50:48 IDR 50000
03-04-2024, 19:46:28 IDR -20000
03-04-2024, 19:46:24 IDR -20000
03-04-2024, 19:46:21 IDR -20000
03-04-2024, 19:46:19 IDR -20000
03-04-2024, 19:46:17 IDR -20000
03-04-2024, 19:46:15 IDR -20000
03-04-2024, 19:45:19 IDR -20000

Gambar 17e. Tampilan User App setelah Topup

Analisis:

- Payment Gateway berhasil memproses pesanan topup
- Dapat terlihat pada basis data bertambah entri baru dengan Transaction Type Topup
- Dapat terlihat pada basis data User dengan UID 61FA7BE6 memiliki saldo 50.000
- User App menampilkan transaksi Topup dengan warna hijau

## Pengujian Flow Gagal Login (Akun Tidak Terdaftar dan Salah Credentials)

Login pada aplikasi dapat dilakukan jika dan hanya jika:

1. Akun terdaftar pada aplikasi
2. Password yang disediakan benar

Ekspektasi:

- Gagal login ketika akun yang dimasukkan tidak terdaftar
- Gagal login ketika password yang dimasukkan salah

Hasil Uji:

Welcome Back!

Your ID

Your password

**Login**

Invalid Credentials

Gambar 18a. Tampilan Login Gagal (Akun Tidak Terdaftar)

Welcome Back!

Your ID

Your password

**Login**

Invalid Credentials

Gambar 18b. Tampilan Login Gagal (Salah Password)

**Analisis:**

Kedua kasus, akun tidak terdaftar dan kesalahan password, tertangani dengan baik dan menampilkan informasi Invalid Credentials.

# Kesimpulan dan Saran

## Kesimpulan

Aplikasi yang dibuat memiliki fitur layaknya *electronic wallet* yang umumnya ada dengan fitur utama pembayaran. Untuk menyokong sistem *electronic wallet*, dibuat fitur-fitur pendukung yaitu Merchant Reader sebagai tampilan status transaksi, aplikasi untuk melihat historis dan detail user serta melakukan topup.

Sistem dibuat dengan sistem IoT, khususnya hardware memanfaatkan ESP32 dengan modul RC522 untuk membaca RFID. MQTT dimanfaatkan sebagai message broker pada sistem serta pemakaian protokol websocket untuk menyokong komunikasi dengan *web browser*. Selain itu, HTTP digunakan sebagai protokol komunikasi.

Pemanfaatan Backend Django dalam memproses transaksi dan persistensi data yang disimpan pada basis data, dalam hal ini SQLite. Data yang dikomunikasikan dibaca oleh Merchant Reader. Sebuah aplikasi User dikembangkan dengan berkomunikasi dengan Backend untuk mendapatkan detail User serta melakukan transaksi melalui Payment Gateway Midtrans yang memperbolehkan topup melalui berbagai vendor.

## Refleksi

Dalam pengembangan beberapa kendala dan teknis baru didapati. Konfigurasi terhadap server MQTT memiliki berbagai opsi konfigurasi, baik autentikasi, port, hingga protokol.

Pemanfaatan modul pada ESP32 memiliki kesulitan tersendiri khususnya penyesuaian dan pemilihan PIN yang sesuai antara modul dan board ESP32. Selain kendala software, kendala hardware menjadi hal yang perlu diperhatikan (koneksi kabel serta lainnya). Koneksi antara ESP32 dengan WiFi dan MQTT Broker terkadang menjadi permasalahan, khususnya selama pengembangan dengan IP yang dinamis (sesuai assignment DHCP).

Pengembangan aplikasi penyokong sistem IoT perlu memperhatikan protokol yang digunakan dan akses terhadap data sistem. Informasi yang dimanfaatkan, salah satunya pada RFID menjadi sebuah titik kerentanan karena sifatnya yang publik dan dapat diubah sewaktu-waktu.

## Saran

1. Pemanfaatan sistem enkripsi dan metoda lainnya dalam autentikasi pesan antara device dapat dikembangkan untuk meningkatkan keamanan tukar menukar pesan pada sistem.
2. UI yang dikembangkan masih sederhana, dapat dilakukan peningkatan baik aspek UI (User Interface) maupun UX (User Experience).
3. Penanganan kendala komunikasi yang lebih baik (pengiriman pesan yang gagal) agar transaksi dapat dijamin diproses.
4. Desain komunikasi khususnya topik pada sistem MQTT yang lebih baik sehingga tidak di-broadcast ke seluruh device, misal mengembangkan sistem menjadi multiple RFID Reader dan Merchant Reader, maka perlu agar data informasi transaksi ditampilkan terhadap pasangan RFID Reader dan Merchant Reader yang bersesuaian.
5. Rangkaian hardware yang lebih rigid sehingga kabel tidak terexpose yang memungkinkan mati/rusaknya sistem.

# Daftar Referensi

- [1] *Docs: Next.js.* Docs | Next.js. (n.d.). <https://nextjs.org/docs>
- [2] Finkenzeller, K., & Müller, D. (2019). *RFID Handbook: Fundamentals and Applications in contactless smart cards, radio frequency identification and near-field communication.* Wiley.
- [3] Hillar, G. C. (2017). *Mqtt Essentials: A lightweight IOT protocol: The preferred IOT publish-subscribe Lightweight Messaging Protocol.* Packt Publishing.
- [4] Miguelbalboa. (n.d.). *Miguelbalboa/RFID: Arduino Rfid Library for MFRC522.* GitHub. <https://github.com/miguelbalboa/rfid>
- [5] *Mosquitto.conf man Page.* Eclipse Mosquitto. (2023, August 24). <https://mosquitto.org/man/mosquitto-conf-5.html>
- [6] *Welcome to Midtrans documentations.* Midtrans Documentation. (n.d.). <https://docs.midtrans.com/>

# Lampiran

Link Github:

<https://github.com/Nk-Kyle/IoTPaymentSystem/>

Link Video:

<https://youtu.be/juDvmqOd0MY>

Link Dokumentasi Teknis/Kerja:

[https://docs.google.com/document/d/1thT-XZ5QS8DILmi1KBX1nLI8fyUmTsYSdCIJCJjml10/edit  
?usp=sharing](https://docs.google.com/document/d/1thT-XZ5QS8DILmi1KBX1nLI8fyUmTsYSdCIJCJjml10/edit?usp=sharing)