Implementasi *Backend System* Untuk Integrasi *Payment Gateway* Pada Sistem Pembayaran Kost Menggunakan Express.js

Juan Angela Alma[1], Agus Prihanto[2]

[1,3] Teknik Informatika, Fakultas Teknik, Universitas Negeri Surabaya
[1]juan.20058@mhs.unesa.ac.id
[2]agusprihanto@unesa.ac.id

***Abstrak*— Rumah kost menjadi pilihan populer bagi mahasiswa, pekerja, dan pelancong karena lokasinya yang strategis, fasilitas lengkap, dan harga terjangkau. Meski sistem informasi kost telah ada untuk manajemen penyewa dan pembayaran, banyak yang masih menggunakan metode pembayaran semi-tradisional. Oleh karena itu, diperlukan teknologi modern seperti payment gateway Midtrans. Integrasi langsung dengan *payment gateway* melalui *frontend* memiliki kelemahan dalam keamanan data. Penelitian ini bertujuan untuk merancang dan membangun Backend System untuk menambahkan mekanisme keamanan pada sistem pembayaran kost. Mekanisme keamanan yang dimaksud dalam penelitian ini adalah kerahasiaan Client Key Midtrans. Penelitian ini menggunakan metode yang meliputi beberapa tahapan penting, yaitu: identifikasi masalah, studi literatur, analisis kebutuhan, perancangan, implementasi, dan pengujian. Hasil pengujian menunjukkan bahwa integrasi secara langsung melalui *frontend* tanpa *backend* kerahasiaan *client key* tidak sepenuhnya terjaga. Sebaliknya, integrasi dengan *backend system*, *client key* disimpan sepenuhnya di sisi *server*, sehingga tidak terekspos ke *frontend*. Selain itu, integrasi dengan *backend* tidak memerlukan pengiriman *client key* ke sisi *backend* maupun Midtrans. Penelitian ini membuktikan bahwa *backend system* mampu menambahkan mekanisme keamanan dengan mengamankan *client key* di sisi *server* dan mengurangi risiko akses tidak sah.**

Kata Kunci— **Rumah Kost, Sistem Pembayaran, *Payment Gateway*, Keamanan, *Backend System, Client Key*.**

I. PENDAHULUAN

Rumah kost merupakan pilihan tempat tinggal yang banyak disukai oleh mahasiswa, pekerja, dan pelancong. Penyediaan fasilitas yang menguntungkan, seperti lokasi yang strategis, fasilitas yang lengkap, dan harga yang terjangkau membuat rumah kost tidak hanya sekedar tempat tinggal, namun juga menciptakan lingkungan yang dinamis dan ramah.

Banyak sistem informasi kost diimplementasikan untuk memfasilitasi kebutuhan pengelola dan penyewa kost. Sistem-sistem ini bertujuan untuk mempermudah proses manajemen kost, mulai dari pencatatan data penyewa, pengelolaan pembayaran, hingga payment ketersediaan kamar.

Referensi [1], [2], [3] menunjukkan bahwa masih ada penerapan metode pembayaran yang semi tradisional dalam beberapa sistem informasi kost. Hal ini menyoroti bahwa walaupun ada kemajuan dalam penggunaan teknologi informasi, tetap terdapat beberapa praktik yang belum sepenuhnya mengadopsi solusi modern.

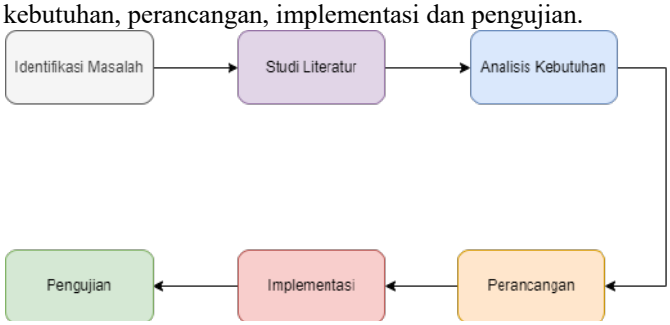
Oleh karena itu, diperlukan solusi untuk mengembangkan sistem pembayaran kost dengan menerapkan teknologi yang lebih modern. Salah satu cara penerapan teknologi di bidang pembayaran adalah menggunakan *payment gateway* [4]. Salah satu penyedia layanan pembayaran yang populer adalah Midtrans [9].

Integrasi dengan Midtrans dapat dilakukan langsung melalui *frontend*, namun integrasi langsung melalui *frontend* memiliki beberapa kekurangan (i) keamanan data, (ii) menyulitkan pengembang aplikasi *frontend*, (iii) fleksibilitas yang terbatas, (iv) distribusi tugas yang rumit. Permasalahan tersebut memerlukan *server* yang menjadi perantara antara *frontend* dan *payment gateway*. Kehadiran *server* dapat meningkatkan keamanan tambahan agar fungsi aplikasi jadi lebih baik. *Backend system* adalah salah satu contoh *server* penghubung antara *frontend* dengan *payment gateway* [5].

Penelitian ini bertujuan untuk membuat *backend system* dari sistem pembayaran kost yang dapat meningkatkan keamanan dan fleksibilitas integrasi dengan *payment gateway* pihak ketiga. Perkembangan teknologi dalam bidang akomodasi dapat memperbaiki proses manajemen dan pembayaran untuk kepuasan pengguna yang lebih baik.

II. METODOLOGI PENELITIAN

Metodologi penelitian yang digunakan pada penelitian ini meliputi identifikasi masalah, studi literatur, analisis kebutuhan, perancangan, implementasi dan pengujian.



Gbr. 1 Metodologi Penelitian.

A. *Identifikasi Masalah*

Pada tahap ini penulis melakukan penelitian untuk menentukan dasar dari penelitian yang akan dilakukan. Dalam penelitian ini diambil masalah terkait sistem informasi kost yang masih menggunakan cara yang semi tradisional atau bahkan sepenuhnya tradisional dan penerapan *Backend System* pada *payment gateway* Midtrans yang bertujuan untuk menambahkan mekanisme keamanan pembayaran pada Sistem Pembayaran Kost. Mekanisme keamanan yang dimaksud pada penelitian ini adalah kerahasiaan *client key* sehingga *client key* tidak disimpan dan terekspos di sisi *frontend*.

B. Studi Literatur

Pada tahap ini penulis melakukan eksplorasi terhadap konsep pembayaran modern yang menggunakan *payment gateway* Midtrans. Selain itu, penulis juga melakukan eksplorasi penerapan *payment gateway* menggunakan RESTful API dengan framework Express JS.

C. Analisis Kebutuhan

Analisis kebutuhan merupakan tahapan peneliti menetapkan persyaratan untuk sistem pembayaran kos berdasarkan permasalahan yang akan diatasi. Setiap persyaratan yang telah diidentifikasi kemudian dicatat secara terperinci, termasuk deskripsi yang jelas, sumber persyaratan, dan kriteria keberhasilan yang dapat diukur. Pada penelitian ini, teknologi yang akan digunakan adalah Express.js sebagai *backend* dan MySQL sebagai penyimpanan datanya. Berikut adalah hasil identifikasi persyaratan yang telah dilakukan oleh peneliti.

TABEL I
KEBUTUHAN FUNGSIONAL

No	Kebutuhan Fungsional
1	Sistem harus dapat membuat transaksi pembayaran melalui Midtrans
2	Sistem dapat menerima webhook dari midtrans untuk merubah status pembayaran
3	Sistem harus dapat menampilkan tagihan yang belum dibayar
4	Sistem harus dapat menampilkan rincian pembayaran

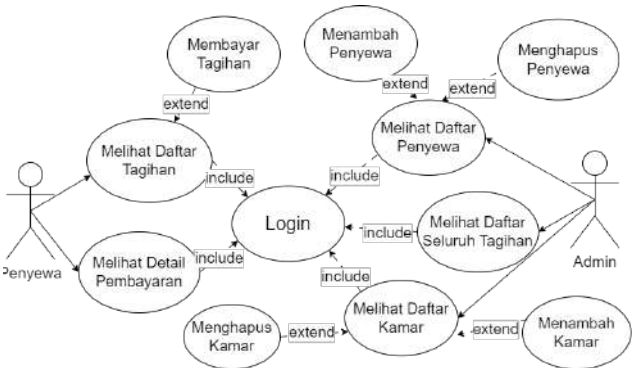
TABEL II
KEBUTUHAN NON-FUNGSIONAL

No	Kebutuhan Non-Fungsional
1	Sistem berjalan dengan lancar
2	Sistem harus mempunyai performa yang baik

D. Perancangan Sistem

1) Use Case

Use case diagram adalah jenis diagram UML(Unified Modeling Language) yang menggambarkan fungsi, ruang lingkup, dan interaksi pengguna dengan sistem tersebut [6]. Diagram use case memvisualisasikan interaksi antara pengguna (aktor) dan sistem (use case), serta tindakan apa saja yang dapat dilakukan aktor terhadap use case secara rinci [7]. Berikut use case dari Sistem Informasi Pembayaran Kost.



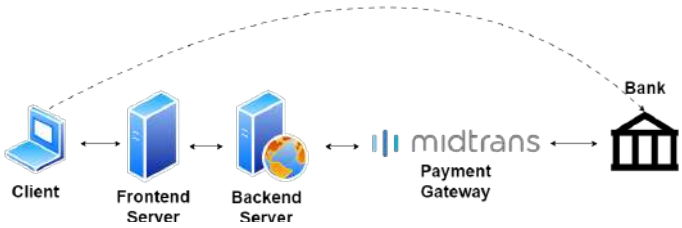
Gbr. 2 *Use Case* Diagram Sistem Pembayaran Kost.

Berikut penjelasan dari use case:

1. *Login*: Penyewa dan Admin dapat melakukan login ke dalam sistem. Use case ini diimplementasikan di *backend* dalam API *Login* Pengguna dengan *endpoint* `/api/login`.
2. Melihat Daftar Tagihan: Penyewa dapat melihat daftar tagihan dan membayar tagihan. Use case ini diimplementasikan di *backend* dalam API Tagihan Penyewa dan API Pembayaran yang memiliki *endpoint* `/api/bills` dan `/api/bills/:id/pay`.
3. Melihat Detail Pembayaran: Penyewa dapat melihat detail pembayaran yang telah dilakukan. Use case ini diimplementasikan di *backend* dalam API Rincian Pembayaran yang memiliki *endpoint* `/api/payments/:invoice`.
4. Melihat Daftar Seluruh Tagihan: Admin dapat melihat seluruh daftar tagihan yang belum dibayar oleh penyewa. Use case ini diimplementasikan di *backend* dalam API Tagihan Penyewa yang diakses oleh Admin dengan *endpoint* /api/admin/bills.
5. Melihat Daftar Penyewa: Admin dapat melihat, menghapus dan menambah penyewa. Use case ini diimplementasikan *backend* ke dalam API Manajemen Pengguna yang memiliki *endpoint* /api/admin/tenants(GET,POST), api/admin/tenants/:id(DELETE).
6. Melihat Daftar Kamar: Admin dapat melihat daftar kamar beserta status ketersediaannya. Admin juga dapat menghapus dan menambah kamar baru. Use case ini diimplementasikan *backend* ke dalam API Manajemen Kamar yang memiliki *endpoint* /api/admin/rooms(GET,POST), /api/admin/tenants/:id(DELETE).

2) Arsitektur Sistem Pembayaran Kost

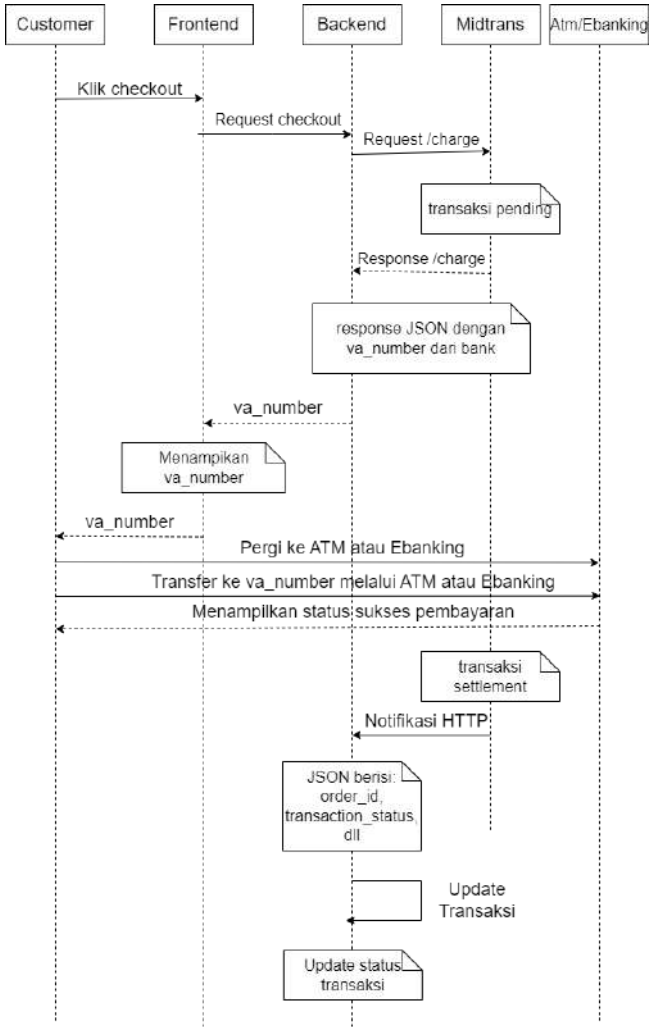
Arsitektur pembayaran kost disusun untuk visualisasi alur kerja dari sistem dan dapat digunakan untuk membuat pandangan yang terstruktur tentang bagaimana sistem terintegrasi. Berikut adalah grafik arsitektur sistem pembayaran kost.



Gbr. 3 Arsitektur Sistem Pembayaran Kost.

3) Sequence Diagram Proses Pembayaran dengan Payment Gateway Midtrans Menggunakan RESTful API

Sequence diagram menunjukkan proses interaksi yang disusun dalam urutan berdasarkan waktu [8]. Diagram ini menggambarkan proses dan objek yang terlibat dan urutan komunikasi antar objek sistem. Berikut adalah sequence diagram dari proses pembayaran dengan *payment gateway* Midtrans.



Gbr. 4 Sequence Diagram Alur Pembayaran dengan *Payment Gateway*.

E. Implementasi

Pada tahap ini, peneliti akan melakukan pengembangan sistem berdasarkan rancangan yang telah ditentukan. Instalasi dan konfigurasi sistem akan dilakukan sesuai dengan dependensi yang dibutuhkan. Peneliti akan menulis kode, menyiapkan database, mengintegrasi sistem dengan *payment gateway* dan mengimplementasikan setiap API sesuai dengan perencanaan sebelumnya.

F. Pengujian

Fokus pengujian ini adalah melakukan uji coba mekanisme keamanan yang akan membandingkan kerahasiaan *client key* pada integrasi langsung melalui *frontend* dan dengan *backend system*.

III. Hasil dan Pembahasan

Hasil dari penelitian yang dilakukan berupa data-data yang diambil setelah peneliti membangun *backend*. Peneliti membandingkan data-data yang ada pada web yang menerapkan integrasi langsung tanpa *backend* dari *frontend* menggunakan Snap.js dengan data-data yang setelah integrasi menggunakan *backend* sebagai jembatannya.

A. Implementasi API Sistem Pembayaran Kost

Penulis membuat API yang dibutuhkan *frontend* dan mendokumentasikan menggunakan OpenAPI Specification. OpenAPI Spesification adalah framework dokumentasi API yang populer saat ini [10]. API Sistem Pembayaran Kost adalah API yang diproses oleh *backend system* tanpa integrasi dengan *payment gateway*.

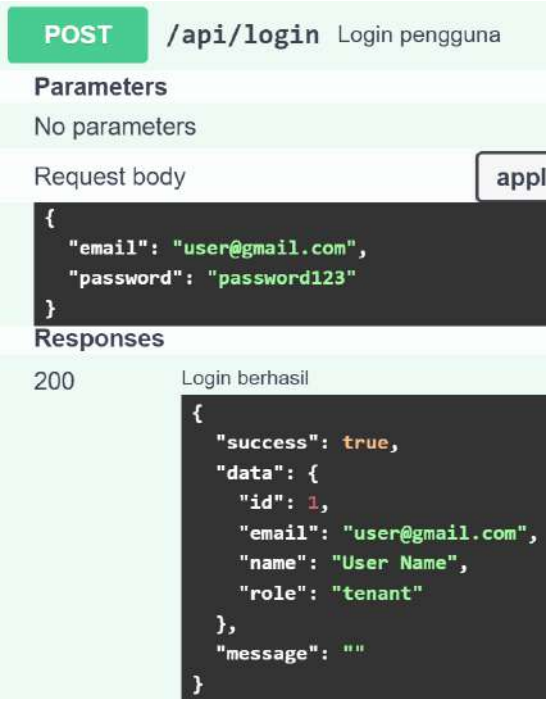
1) API Login Pengguna

API *Login* Pengguna digunakan untuk mengirim informasi *Login* pengguna dan *backend system* mendeteksi hak akses dari pengguna yang dikirimkan. Tabel berikut menunjukkan spesifikasi dari API *Login* pengguna dengan *endpoint* `/api/login`.

Tabel III
Spesifikasi API *Login* Pengguna

Endpoint	Method	Deskripsi	Parameter	Response
/api/login	POST	Melakukan *login* pengguna	Body: { email, password }	200 OK: { id, email, name, role }

Pada gambar di bawah ini menunjukkan dokumentasi API *Login* pengguna. Gambar ini menampilkan parameter yang harus dikirimkan dalam *request body* dan contoh respon dari *backend*.



Gbr. 5 Dokumentasi API *Login* Pengguna.

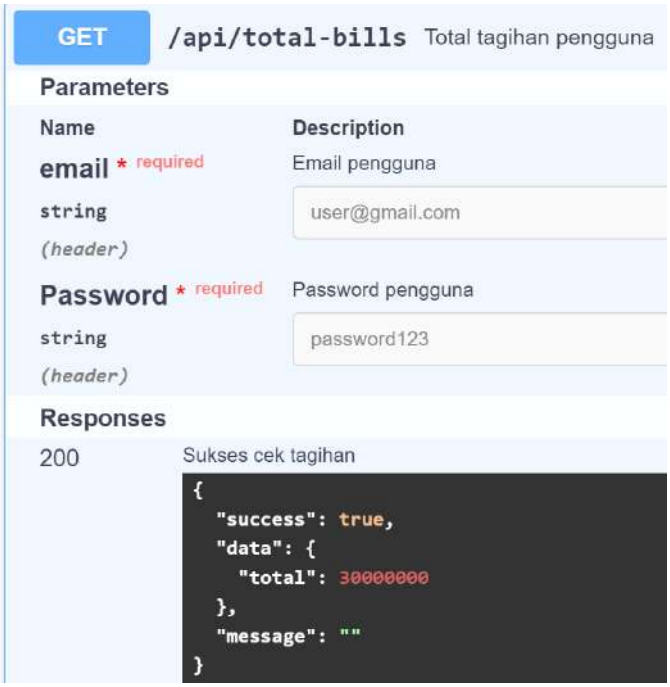
2) API Tagihan Penyewa

API tagihan penyewa digunakan untuk mendapatkan informasi tagihan dari penyewa. Terdapat dua API tagihan dengan *endpoint* `/api/total-bills` dan `/api/bills`. API tagihan dengan *endpoint* `/api/total-bills` memiliki informasi total tagihan yang harus dibayar penyewa. API tagihan dengan *endpoint* `/api/bills` memiliki informasi riwayat tagihan penyewa. Berikut adalah tabel yang menunjukkan spesifikasi dari API Tagihan Penyewa.

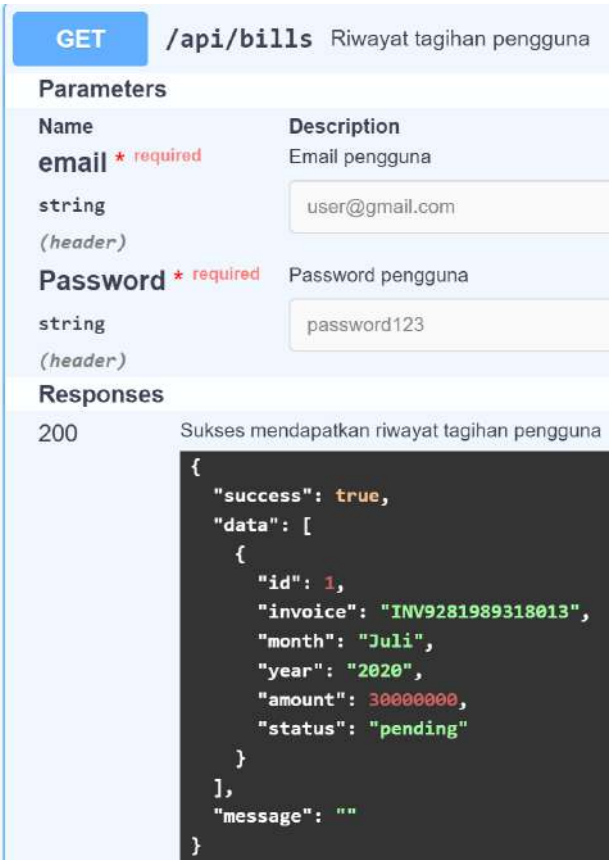
TABEL IV
SPESIFIKASI API TAGIHAN PENYEWA

Endpoint	Method	Deskripsi	Parameter	Response
/api/total-bills	GET	Mendapatkan total tagihan yang harus dibayar	Header: { email, password }	200 OK: { total }
/api/bills	GET	Mendapatkan daftar riwayat tagihan	Header: { email, password }	200 OK: [{ id, month, year, amount, status }]

Dokumentasi API Tagihan juga disertakan. Kedua *endpoint* tagihan memiliki dokumentasi API yang dapat dilihat pada gambar OpenAPI Specification berikut.



Gbr. 6 Dokumentasi API Total Tagihan Penyewa.



Gbr. 7 Dokumentasi API Daftar Tagihan Penyewa.

3) API Rincian Pembayaran

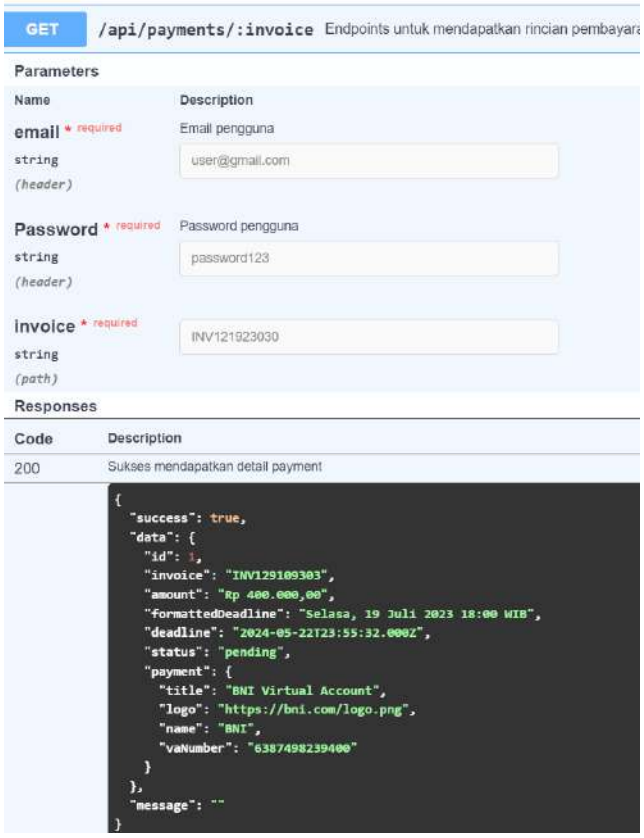
Penulis membuat API yang dapat menampilkan rincian pembayaran dari suatu tagihan. API ini digunakan oleh

frontend untuk mengetahui data pembayaran seperti total yang harus dibayar, nomor rekening pembayaran, batas akhir pembayaran dan status pembayaran. Berikut adalah spesifikasi API rincian pembayaran.

TABEL V
SPESIFIKASI API RINCIAN PEMBAYARAN

Endpoint	Method	Deskripsi	Parameter	Response
/api/payment/:invoice	GET	Mendapatkan rincian pembayaran berdasarkan *invoice*	Header: { email, password } Path: { invoice }	200 OK: { id, invoice, amount, formattedDeadline, deadline, status, payment }

API Rincian Pembayaran mengambil data pembayaran berdasarkan *invoice* yang dikirimkan melalui path. Berikut merupakan dokumentasi dari API Rincian Pembayaran.



Gbr. 8 Dokumentasi API Rincian Pembayaran.

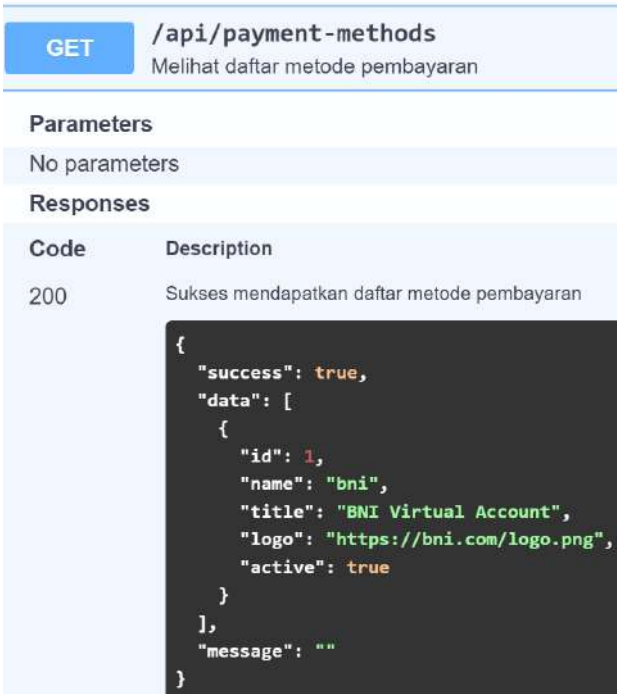
4) API Metode Pembayaran

Metode pembayaran yang digunakan dalam *backend system* dapat diakses oleh *frontend* dengan API ini. API Metode Pembayaran dapat digunakan oleh *frontend* agar daftar metode pembayaran bersifat dinamis yang dapat diubah sesuai dengan kebutuhan atau kebijakan sistem pembayaran. Berikut merupakan spesifikasi API Metode Pembayaran.

TABEL VI
SPESIFIKASI API METODE PEMBAYARAN

Endpoint	Method	Deskripsi	Parameter	Response
/api/payment-methods	GET	Mendapatkan daftar metode pembayaran		200 OK: [{ id, name, title, logo, active }]

API Metode Pembayaran mempunyai *endpoint* `/api/payment-methods` dan memiliki respon status 200 dengan *body* id, name, title, logo, active. Berikut merupakan dokumentasi dari API Metode Pembayaran.



Gbr. 9 Dokumentasi API Metode Pembayaran.

5) API Tagihan yang diakses oleh Admin

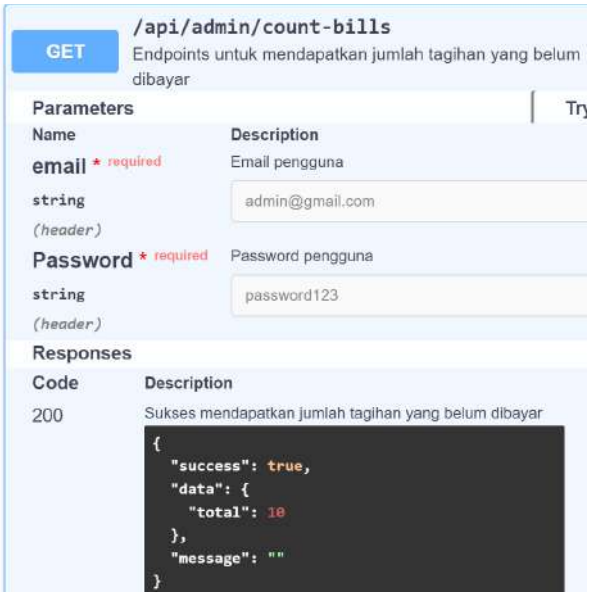
Penulis telah membuat API yang hanya dapat diakses oleh pengguna dengan hak akses admin. Salah satunya adalah API tagihan. Terdapat dua API tagihan yang diakses oleh admin dengan *endpoint* `/api/admin/count-bills` dan `/api/admin/bills`.

TABEL VII
SPESIFIKASI API TAGIHAN YANG DIAKSES ADMIN

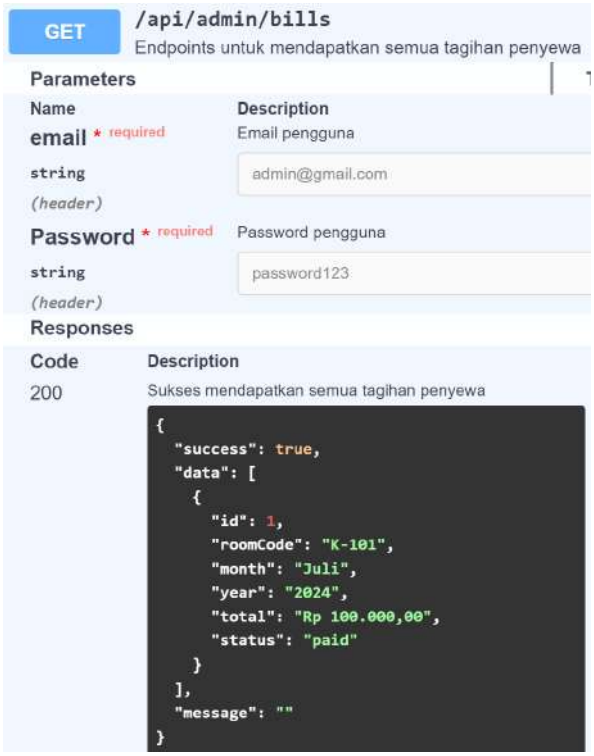
Endpoint	Method	Deskripsi	Parameter	Response
/api/admin/count-bills	GET	Mendapatkan jumlah tagihan	Header: { email, password	200 OK: { total }

		yang belum dibayar	}	
/api/admin /bills	GET	Mendapatkan daftar tagihan penyewa	Header: { email, password }	200 OK: [{ id, roomCode, month, year, total, status }]

Berikut adalah dokumentasi spesifikasi API tagihan yang hanya bisa diakses oleh admin:



Gbr. 10 Dokumentasi API Jumlah Tagihan.



Gbr. 11 Dokumentasi API Daftar Tagihan yang diakses Admin.

6) API Manajemen Penyewa yang diakses oleh Admin

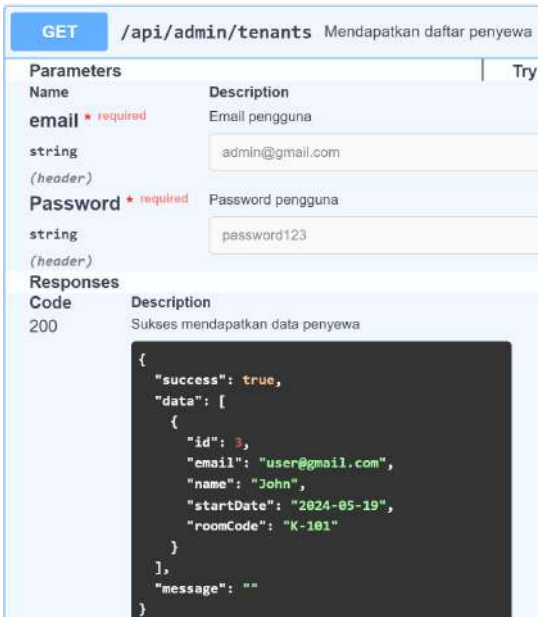
Penulis juga membuat API yang hanya dapat diakses oleh pengguna yang mempunyai hak akses admin yaitu API Manajemen Penyewa. API ini memungkinkan admin untuk mengelola data penyewa. Terdapat tiga *endpoint* yang digunakan untuk melihat, menambah dan menghapus data penyewa. Berikut ini adalah tabel yang menunjukkan detail API Manajemen Penyewa khusus admin.

TABEL VIII
SPESIFIKASI API MANAJEMEN PENYEWA

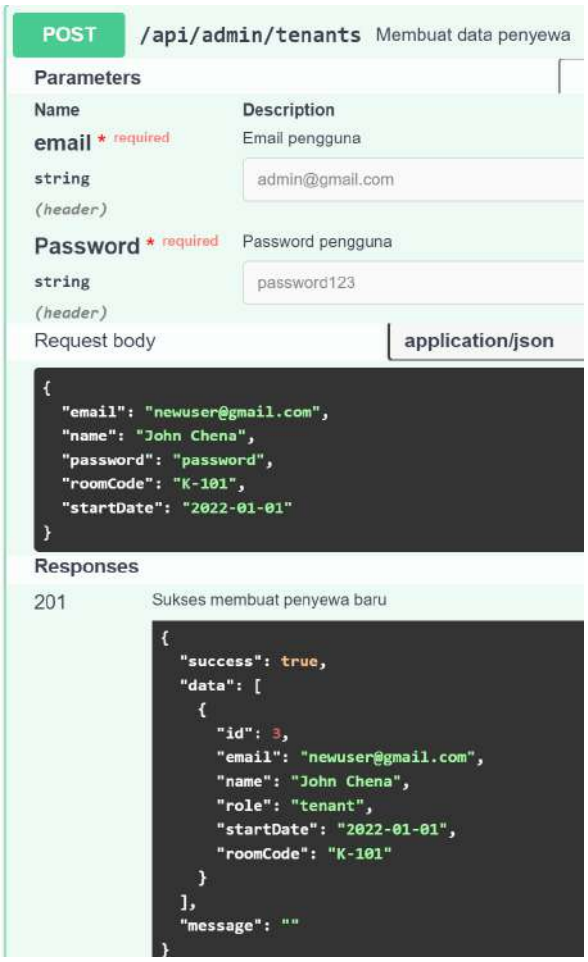
Endpoint	Method	Deskripsi	Parameter	Response
/api/admin /tenants	GET	Mendapatkan daftar penyewa	Header: { email, password }	200 OK: [{ id, email, name, startDate, roomCode }]
/api/admin /tenants	POST	Menambah data penyewa	Header: { email, password } Body: { email, name, startDate, roomCode }	201 OK: { id, email, name, role startDate, roomCode }

Endpoint	Method	Deskripsi	Parameter	Response
/api/admin /tenants/:id	DELETE	Menghapus data penyewa	Header: { email, password } Path: { id }	200 OK: {}

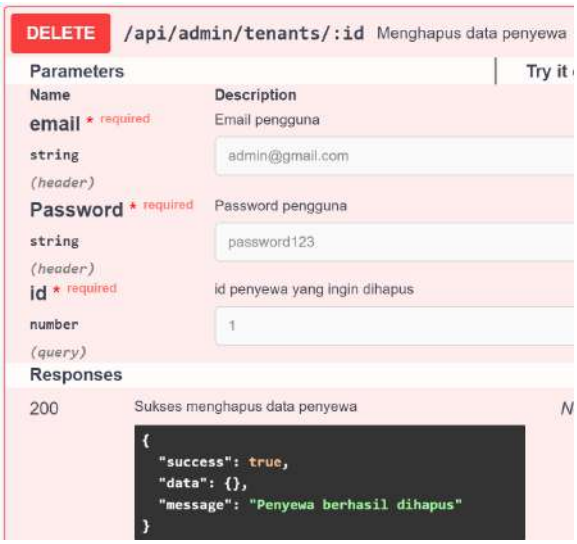
API untuk menghapus data penyewa membutuhkan id penyewa yang harus dikirimkan di path ketika melakukan permintaan API. Id penyewa digunakan oleh *backend system* untuk mengidentifikasi penyewa yang akan dihapus. Dokumentasi spesifikasi API Manajemen Penyewa yang hanya bisa diakses oleh admin dapat dilihat pada gambar berikut.



Gbr. 12 Dokumentasi API Daftar Penyewa.



Gbr. 13 Dokumentasi API Menambahkan Data Penyewa.



Gbr. 14 Dokumentasi API Menghapus Data Penyewa.

7) API Manajemen Kamar yang diakses oleh Admin

Penulis telah membuat API khusus yang hanya dapat diakses oleh pengguna dengan hak akses admin untuk mengelola kamar. API ini memungkinkan admin untuk melihat daftar

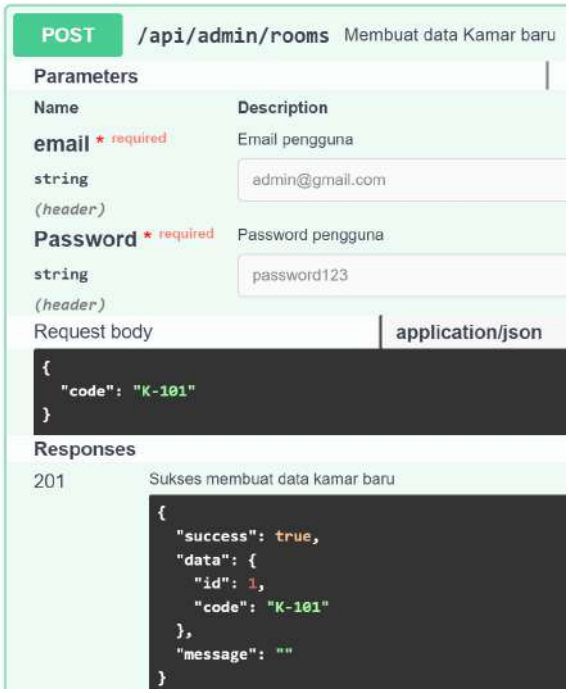
kamar, menambah kamar baru, dan menghapus kamar yang ada. Berikut adalah tabel yang menunjukkan detail API Manajemen Kamar khusus admin.

TABEL IX
SPESIFIKASI API MANAJEMEN PENYEWA

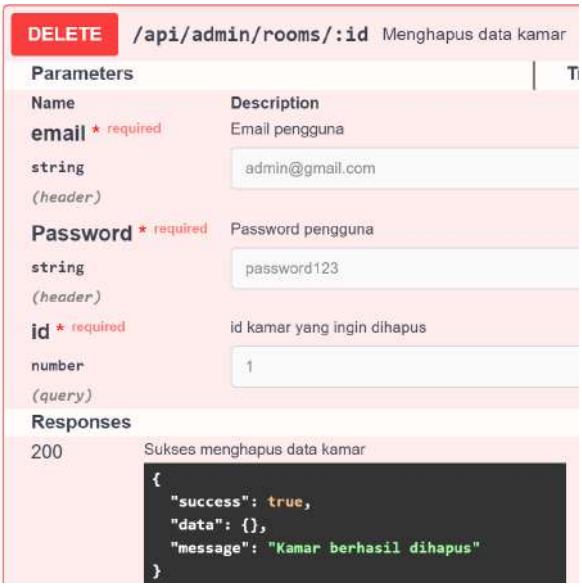
Endpoint	Method	Deskripsi	Parameter	Response
/api/admin /rooms	GET	Mendapatkan daftar kamar	Header: { email, password }	200 OK: [{ id, code, available }]
/api/admin /rooms	POST	Menambah data kamar	Header: { email, password } Body: { code }	201 OK: { id, code }
/api/admin /rooms/:id	DELETE	Menghapus data kamar	Header: { email, password } Path: { id }	200 OK: {}

Permintaan untuk menghapus data kamar mengharuskan pengiriman id kamar. Id kamar digunakan oleh *backend system* untuk menentukan kamar yang akan dihapus dari database. Dokumentasi spesifikasi API Manajemen Kamar yang hanya dapat diakses oleh admin dapat dilihat pada gambar berikut.

Gbr. 15 Dokumentasi API Daftar Kamar.



Gbr. 16 Dokumentasi API Menambah Data Kamar.



Gbr. 17 Dokumentasi API Menghapus Data Kamar.

B. *Implementasi API Pembayaran*

Penulis membuat API pembayaran yang terintegrasi dengan midtrans, sebuah penyedia layanan pembayaran online. API Pembayaran mencakup beberapa *endpoint* untuk menangani proses pembayaran dan notifikasi dari Midtrans.

1) API Permintaan Charge

Dalam penerapan pembayaran Midtrans, API *Charge* digunakan untuk memproses permintaan pembayaran dari midtrans. *Endpoint* ini memungkinkan sistem untuk mengirimkan informasi pembayaran ke Midtrans dan menerima respon terkait status pembayaran. Proses ini

memastikan bahwa transaksi dilakukan dengan aman dan sesuai dengan standar yang ditetapkan oleh Midtrans. Berikut adalah tabel yang menunjukkan detail dari permintaan *Charge* API.

TABEL X
SPESIFIKASI API API *CHARGE*

Endpoint	Method	Deskripsi	Parameter	Response
/api/bills/:id/pay	POST	Memproses pembayaran tagihan berdasarkan ID tagihan	Header: { email, password, x-api-key } Path: id Body: { payment }	201 OK: [{ invoice, status }]

Permintaan API *Charge* harus dikirimkan oleh *frontend* dengan menyertakan id tagihan melalui path. Id tagihan digunakan untuk menentukan tagihan yang akan dibuatkan pembayaran Midtrans oleh *backend system*. Dokumentasi permintaan *Charge* API terdapat pada gambar di bawah ini menunjukkan contoh *request* dan *response* yang lebih detail dari API tersebut.

Gbr. 18 Dokumentasi API *Charge*.

API Permintaan *Charge* ketika diakses akan mengirimkan informasi pembayaran kepada Midtrans dengan data diantaranya:

1. server_key: kunci *server* unik yang diberikan oleh midtrans untuk setiap *merchant*.
2. payment_type: jenis metode pembayaran yang digunakan pada transaksi.
3. order_id: id unik untuk setiap pembayaran yang dihasilkan oleh *backend system*.
4. gross_amount: jumlah total yang harus dibayar oleh penyewa.
5. bank: nama bank yang akan digunakan untuk pembayaran jika metode pembayaran yang dipilih adalah bank_transfer.

Setelah Midtrans memberikan respon informasi pembayaran, maka *backend system* akan menyimpan informasi yang dibutuhkan ke dalam database.

2) API Penerima Notifikasi Pembayaran Midtrans

Proses pembayaran Midtrans mencakup notifikasi yang dikirimkan ke *backend system*, termasuk status pembayaran dan id pembayaran. API ini dirancang untuk menerima dan memproses notifikasi dari Midtrans, memastikan bahwa status setiap pembayaran diperbarui dengan tepat dalam database *backend system*.

TABEL XI
SPESIFIKASI API PENERIMA NOTIFIKASI MIDTRANS

Endpoint	Method	Deskripsi	Response
/api/midtrans-notification	POST	Menerima notifikasi status pembayaran dari Midtrans	201 OK

Data yang dikirimkan oleh midtrans ke API notifikasi akan digunakan oleh *backend system* untuk memperbarui data pembayaran yang ada pada database. *Backend system* juga menerapkan transformasi status pembayaran seperti pada tabel berikut.

TABEL XII
SPESIFIKASI API PENERIMA NOTIFIKASI MIDTRANS

Status Midtrans	Status *Backend*	Keterangan
capture	paid	Pembayaran berhasil dan diterima (jika fraud_status nya bernilai accept)
settlement	paid	Pembayaran berhasil dan diterima
cancel	failed	Pembayaran dibatalkan
failure	failed	Pembayaran dibatalkan

expire	failed	Pembayaran dibatalkan
deny	failed	Pembayaran dibatalkan
pending	pending	Pembayaran sedang menunggu konfirmasi

Penerapan transformasi ini memungkinkan status pembayaran disederhanakan tanpa mengubah esensi dari masing-masing status. Pendekatan ini memperoleh fleksibilitas yang signifikan dalam manajemen pembayaran, memungkinkan penyesuaian yang lebih mudah terhadap perubahan kebijakan atau kebutuhan bisnis di masa depan.

C. *Konfigurasi Autentikasi Pembayaran Midtrans*

Konfigurasi autentikasi pembayaran Midtrans memerlukan kunci akses yang dikirimkan dari *backend system*. *Merchant* Midtrans akan diberikan tiga key yaitu id *merchant*, *client key*, dan *server key* yang dapat diakses melalui dashboard Midtrans seperti pada gambar berikut.

Gbr. 19 Dashboard API Keys Midtrans.

Server key dan *client key* dikirim pada setiap permintaan *Charge* API yang dilakukan oleh *backend system*. Midtrans akan memverifikasi permintaan *Charge* API yang datang dengan *server key* dan *client key*. Berikut merupakan penerapan *server key* dan *client key* pada *backend system* pembayaran kost.

```
const midtransClient = require('midtrans-client');
const core = new midtransClient.CoreApi({
    isProduction: false,
    serverKey: midtransServerKey,
    clientKey: midtransClientKey
})
```

Konfigurasi autentikasi pembayaran Midtrans memerlukan kunci akses yang dikirimkan dari *backend system*. Merchant Midtrans akan diberikan tiga key yaitu id *merchant*, *client key*, dan *server key* yang dapat diakses melalui dashboard Midtrans seperti pada gambar berikut.

D. *Autentikasi Client*

Backend system menerapkan autentikasi menggunakan API Key agar terhindar dari penggunaan oleh client yang tidak dikenal. Permintaan pada API *Charge* akan diverifikasi terlebih dahulu menggunakan middleware dengan mencocokkan data API Key yang dibawa dan yang ada pada *backend*. *Frontend* diharuskan mengirim API Key yang dimilikinya pada *header* dengan format sebagai berikut.

```
Header: {
"x-api-key": "q_EaTiX+xbBXLyO05.+zDXjI+Qi_X0v"
}
```

E. *Pengujian Kerahasiaan Client Key pada Integrasi tanpa Backend System*

Pengujian penyimpanan *client key* pada integrasi langsung dapat dilakukan dengan beberapa langkah sebagai berikut.

1. Menyiapkan lingkungan pengujian.

   Lingkungan pengujian akan disiapkan dengan menggunakan contoh website dari Midtrans, yang dapat diakses melalui https://demo.midtrans.com sebagai bagian dari pengujian kerahasiaan *client key*.

2. Analisis kode sumber.

   Penulis menganalisis kode sumber dan menemukan bahwa terdapat integrasi menggunakan pustaka snap.js. Berikut merupakan kode hasil analisis penulis.

```
<script type="text/javascript" src="/javascripts/app.js"></scr
<script type="text/javascript" src="https://app.sandbox.midtra
        data-client-key="VT-client-yrHf-c8Sxr-ck8tx"></script>
```

Gbr. 20 Skrip Integrasi *Payment Gateway* pada *Frontend*.

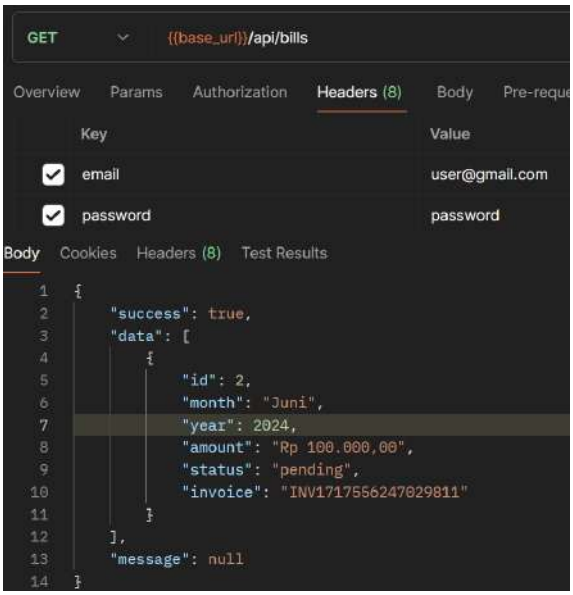
Analisis ini menemukan bahwa di bagian <head> dari halaman HTML terdapat skrip yang memuat pustaka snap.js dari Midtrans, yang digunakan untuk menangani proses pembayaran di *frontend*. *Client key* disertakan langsung dalam atribut `data-client-key` pada elemen <script>, yang menunjukkan bahwa *client key* disimpan dan digunakan di sisi *frontend*.

F. *Pengujian Kerahasiaan pada Integrasi dengan Backend System*

Penulis melakukan pengujian kerahasiaan *client key* pada penggunaan API untuk memastikan tidak ada penggunaan *client key* ketika melakukan permintaan ke API tersebut. Untuk melakukan itu, penulis menggunakan *tool* Postman untuk melakukan pengujian terhadap API.

1. Pengujian API Tagihan Penyewa.

   Pengujian API Tagihan dapat dilakukan dengan mengirimkan permintaan HTTP dengan method GET, menggunakan *header* yang berisi *email* dan *password* dari penyewa tersebut tanpa mengirimkan *client key*. Berikut adalah permintaan HTTP API Tagihan Penyewa menggunakan Postman.

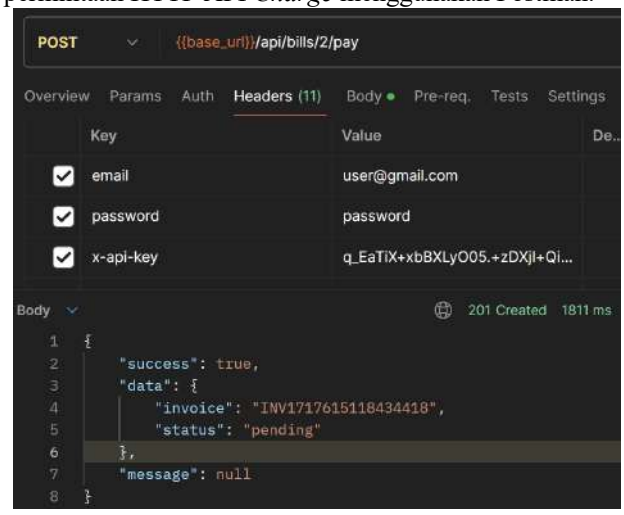


Gbr. 21 Pengujian API Tagihan Penyewa.

Gambar di atas menunjukkan permintaan HTTP ke API Tagihan Penyewa berhasil dilakukan dan mendapatkan respon sesuai yang diharapkan dengan menyertakan *header* yang berisi *email* dan *password* tanpa adanya *client key* yang dikirim.

2. Pengujian API *Charge*.

Pengujian API *Charge* dilakukan pada *endpoint* /api/bills/:id/pay. Id pada parameter didapatkan dari id tagihan. Untuk melakukan pengujian, permintaan dikirimkan dengan method POST dengan *header* yang berisi *email*, *password* dan x-api-key. Berikut merupakan permintaan HTTP API *Charge* menggunakan Postman.

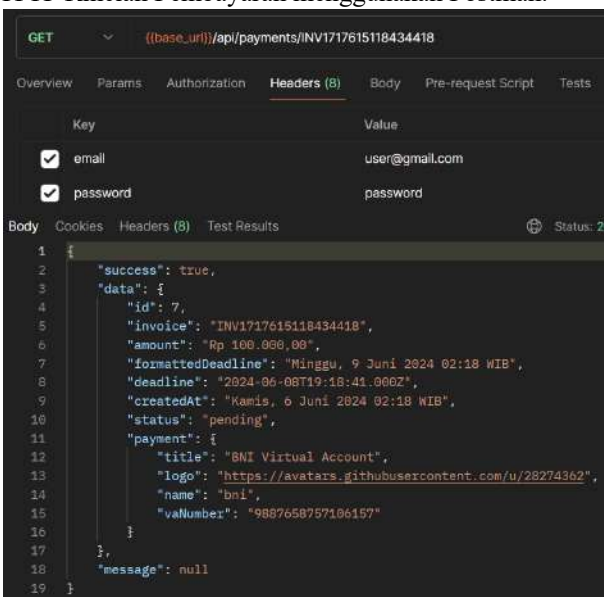


Gbr. 22 Pengujian API *Charge*.

Gambar di atas terlihat permintaan berhasil dilakukan dan mendapatkan status respon 201 Created, dengan respon *body* yang berisi data *invoice* dan status sesuai dengan yang diharapkan.

3. Pengujian API Rincian Pembayaran.

Pengujian API Rincian Pembayaran dilakukan pada *endpoint* /api/payment/:invoice. Invoice pada parameter didapatkan dari data pembayaran setelah melakukan *Charge*. Untuk melakukan pengujian, permintaan dikirimkan dengan method GET dengan *header* yang berisi *email* dan *password*. Berikut merupakan permintaan HTTP Rincian Pembayaran menggunakan Postman.



Gbr. 23 Pengujian API *Charge*.

Gambar di atas menunjukkan permintaan HTTP pada API Detail Pembayaran berhasil dilakukan dan mendapatkan data detail pembayaran sesuai dengan yang diharapkan.

G. Perbandingan Kerahasiaan Client Key pada Integrasi Tanpa Backend System dan dengan Backend System

TABEL XIII
PERBANDINGAN KERAHASIAAN CLIENT KEY

Aspek	**Integrasi tanpa *Backend System***	**Integrasi dengan *Backend System***
Kerahasiaan *client key*	Tidak, rentan terhadap inspeksi elemen dan jaringan.	Ya, *client key* disimpan di sisi *server*, tidak terekspos ke *frontend*.
Risiko Pencurian *client key* Tinggi	Ya, *client key* dapat diakses oleh pihak tidak berwenang.	Tidak, *client key* terlindungi dalam lingkungan *server* yang terkontrol.
Frontend mengirimkan *client key*	Ya, *frontend* mengirimkan *client key* ke *server* Midtrans.	Tidak, *frontend* tidak perlu mengirimkan *client key* ke *backend system* ataupun Midtrans.

IV. KESIMPULAN

Penelitian ini berhasil mengimplementasikan *backend system* dengan berbagai API untuk sistem pembayaran kost,

diantaranya API *Login* pengguna, API tagihan penyewa, API rincian pembayaran, API metode pembayaran, serta API yang diakses oleh Admin seperti API manajemen penyewa dan kamar. Autentikasi pembayaran menggunakan *client key* dan *server key* yang disimpan sepenuhnya di sisi *server*. Autentikasi Client juga diterapkan pada penelitian ini dengan menggunakan x-api-key.

Pengujian menunjukkan bahwa pada integrasi tanpa *backend system*, *client key* disimpan di *frontend*, sehingga kerahasiaan *client key* tidak sepenuhnya terjaga. Sebaliknya, pada integrasi dengan *backend system*, *client key* disimpan pada sisi *server*, sehingga tidak terekspos ke *frontend* sehingga lebih rahasia. Integrasi tanpa *backend system* mengharuskan *frontend* untuk mengirimkan *client key* ke *server* Midtrans, sedangkan dengan *backend system*, *frontend* tidak perlu mengirimkan *client key*.

V. SARAN

Berdasarkan hasil penelitian dan pembahasan yang telah dilakukan, terdapat beberapa saran yang dapat diberikan untuk pengembangan lebih lanjut:

1. Penguatan keamanan *backend system* dengan menerapkan keamanan tambahan seperti penggunaan enkripsi data, serta implementasi mekanisme autentikasi dan otorisasi yang lebih kuat (Misalnya OAuth, JWT) akan lebih memperkuat keamanan sistem.
2. Penerapan integrasi dengan *payment gateway* membuat *backend system* sangat tergantung dengan *server payment gateway*, ada kalanya *server payment gateway* mati dan tidak bisa mengirim notifikasi pembayaran ke *backend system*. Untuk mengatasi ini, diperlukan suatu metode pencocokan data pembayaran antara *backend system* dan *payment gateway* secara berkala sehingga transaksi yang terlewat dapat terdeteksi dan disinkronkan. Pencocokan ini bisa dilakukan dengan menggunakan API yang disediakan oleh *payment gateway* untuk memeriksa status transaksi secara periodik dan memastikan bahwa setiap pembayaran yang dilakukan tercatat dengan benar di *backend system*. Hal ini akan mengurangi risiko kehilangan data transaksi.

REFERENSI

[1] Melyani, N. A., Iqrom, M., & Amrullah, A. (2023, August). Sistem Informasi Penyewaan Kost Kita Berbasis Web Menggunakan Metode Object Oriented Analysis and Design: Our Web-Based Boarding Rental Information System Using Object Oriented Analysis and Design Method. In SENTIMAS: Seminar Nasional Penelitian dan Pengabdian Masyarakat (pp. 308-318).

[2] Satria, S., Gusman, D., & Azrialdi, E. (2022). Rancang Bangun Sistem Informasi Kost Berbasis Web di Kecamatan Tampan: Design and Build of Web-Based Boarding Information System In Tampan District. MALCOM: Indonesian Journal of Machine Learning and Computer Science, 2(1), 28-36.

[3] Prayogi, A. A., Niswar, M., & Rijal, M. (2020, June). Design and implementation of REST API for academic information system. In IOP Conference Series: Materials Science and Engineering (Vol. 875, No. 1, p. 012047). IOP Publishing.

[4] Ramadhan, A. W., Susanto, A., & Saraswati, G. W. (2023). Implementasi Digital Payment Gateway Midtrans Pada Sistem Agribisnis Di Temanggung (SIADIT). J-SAKTI (Jurnal Sains Komputer dan Informatika), 7(1), 95-107.

[5] Syahputra, M. D. A., & Mulya, M. F. (2023, January). Analisis dan Perancangan E-ticket Metaverse Event Berbasis Midtrans Payment gateway (Studi Kasus: PT Semesta Realitas Indonesia). In Prosiding TAU SNARS-TEK Seminar Nasional Rekayasa dan Teknologi (Vol. 2, No. 1, pp. 37-49).

[6] Dharwiyanti, S., & Wahono, R. S. (2003). Pengantar unified modeling language (uml). IlmuKomputer. com, 11(1), 1-13.

[7] Destriana, R., Kom, M., Husain, S. M., Kom, S., Handayani, N., Kom, M., ... & Kom, S. (2021). Diagram UML Dalam Membuat Aplikasi Android Firebase" Studi Kasus Aplikasi Bank Sampah". Deepublish.

[8] Sulistyorini, P. (2009). Pemodelan visual dengan menggunakan uml dan rational rose. Dinamik, 14(1).

[9] Rahardika, P. (2020). Implementasi Sistem Pembayaran Dengan Payment Gateway Pada Pemesanan Tour & Transport (Studi Kasus PT. Hanoman Pandu Wisata) (Doctoral dissertation, University of Technology Yogyakarta).

[10] Dos Santos, J. S., Azevedo, L. G., Soares, E. F., Thiago, R. M., & da Silva, V. T. (2020). Analysis of Tools for REST Contract Specification in Swagger/OpenAPI. In ICEIS (2) (pp. 201-208).