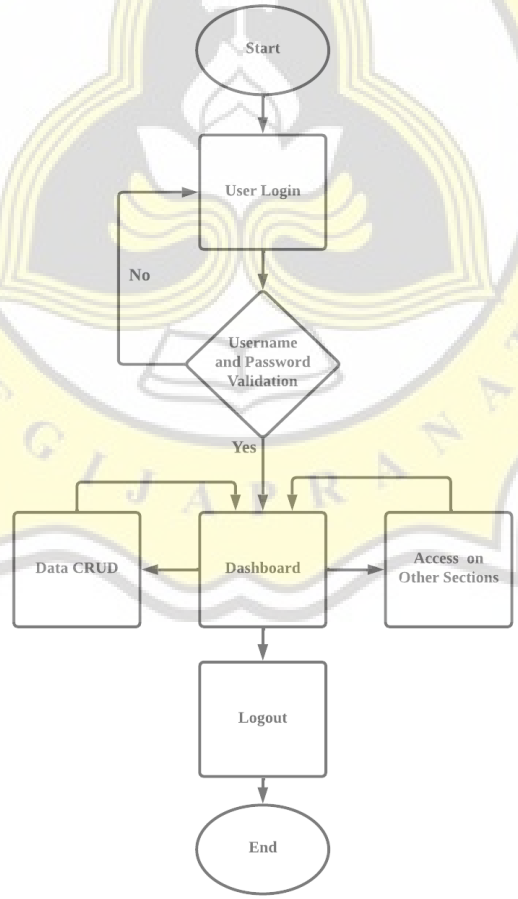
BAB 4
HASIL PENGEMBANGAN DAN PENGUJIAN

4.1 *Flowchart* aplikasi web *dashboard*

Gambar 4.1 menampilkan alur proses dalam aplikasi web *dashboard* Apps2pay dimulai dengan langkah *login* pengguna ke dalam aplikasi. Setelah *login* berhasil, proses akan berlanjut ke akses menuju *dashboard* yang menyajikan informasi terkini dan fitur-fitur penting bagi pengguna. Di dalam *dashboard*, pengguna dapat melakukan operasi CRUD *(Create, Read, Update, Delete)* terhadap data, mengakses bagian-bagian lain dalam aplikasi seperti pengaturan dan detail data transaksi, serta melakukan langkah *logout* untuk mengakhiri proses. *Flowchart* ini mencakup langkah-langkah *login*, akses *dashboard*, operasi CRUD data, akses bagian-bagian lain dalam aplikasi, dan langkah *logout* sebagai tahapan akhir dalam penggunaan aplikasi.



Gambar 4.1 *Flowchart* aplikasi web *dashboard*

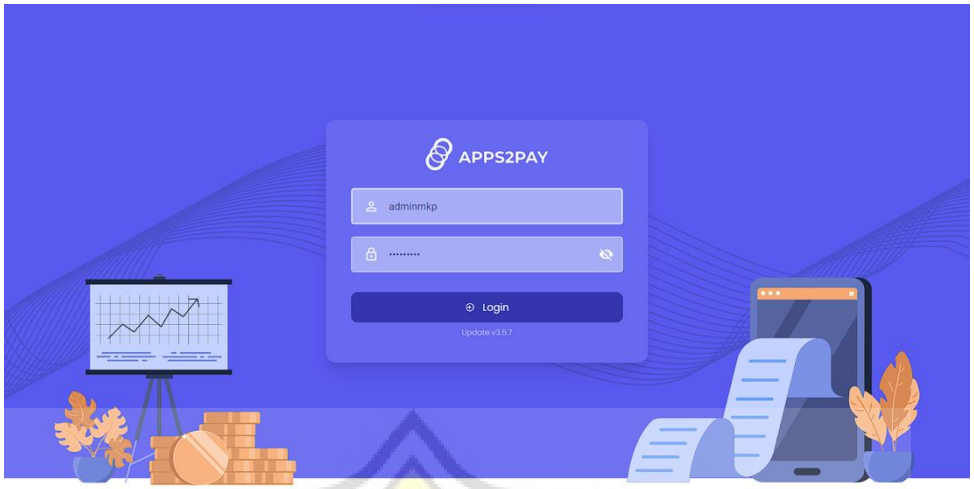
4.2 Rancangan aplikasi web *dashboard*

Dalam merancang aplikasi web *dashboard* Apps2pay, pendekatan yang diambil adalah dengan memanfaatkan *dashboard* yang sudah ada sebagai referensi untuk perancangan tampilan (UI/UX). Pendekatan ini memungkinkan penggunaan *dashboard* yang telah teruji dan terbukti efektif sebagai acuan dalam mengembangkan tampilan baru. Dengan menggunakan *dashboard* yang telah ada sebagai referensi, proses perancangan desain dapat menjadi lebih mudah karena adanya wawasan tentang desain yang baik, tata letak yang efisien, serta fitur-fitur yang relevan dan diperlukan dalam aplikasi *dashboard* yang sedang dibangun.

Pendekatan ini diterapkan untuk menghemat waktu dan upaya dalam merancang tampilan dari awal, sambil tetap memastikan bahwa desain aplikasi yang dihasilkan tetap terlihat profesional dan menarik bagi pengguna. Selain itu, penggunaan *dashboard* yang telah ada sebagai acuan juga membantu mencapai konsistensi antara aplikasi *dashboard* yang baru dikembangkan dengan aplikasi *dashboard* yang sudah ada, sehingga pengguna dapat merasa familiar dan mudah beradaptasi dengan antarmuka yang baru. Dan berikut merupakan hasil dari perancangan dari aplikasi web *dashboard* yang telah dibangun:

4.2.1 *Login* Pengguna

Dalam alur proses *login*, pengguna diminta untuk memasukkan informasi dari akun, seperti *username* dan *password*, untuk melakukan *login* ke dalam aplikasi dengan tampilan antarmuka pada gambar 4.2. Jika informasi yang dimasukkan tidak sesuai dengan data yang terdaftar, sistem akan menganggap *login* gagal dan pengguna akan kembali ke langkah *login* untuk mencoba lagi. Namun, jika informasi yang dimasukkan benar, sistem akan memverifikasi data tersebut dan *login* akan berhasil.



Gambar 4.2 Halaman *login*

Proses *login* dalam aplikasi web menggunakan API pada gambar 4.3 dimulai dengan pengguna mengirimkan data berupa *username* dan *password* melalui antarmuka pengguna. Data ini akan dikirim ke server yang mengelola API untuk proses otentikasi. Server akan memverifikasi data yang diterima dengan data yang terdaftar dalam sistem.

```
const loginFunction = () => {
  const inputData = {
    username: username,
    password: password,
  };
  fetch(BASE_URL + "/login/dummylogin", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      Accept: "application/json",
    },
    body: JSON.stringify(inputData),
  })
    .then((response) => {
      return response.json();
    })
    .then((json) => {
      if (json.success === true) {
        notify(json.message, json.result.token, json.result.nama);
        setAuth(json.result.token, json.result.nama);

        if (json.result.privilege != null) {
          setPrivilegeRole(json.result.privilege);
        } else {
          setPrivilegeRole([]);
        }

        setHirarkiLogin(json.result.hirarkiId);
      } else if (json.success === false) {
        notifyError(json.message);
      }
    })
    .catch((err) => {
      notifyError("Failed To Login");
    });
};
```

Gambar 4.3 Fungsi API *login*

Jika data tersebut valid, server akan menghasilkan respon berupa token, *username*, dan *user role* seperti yang dicontohkan pada gambar 4.4. Token adalah sebuah kode unik yang digunakan untuk mengidentifikasi pengguna yang telah berhasil *login*. Token ini akan digunakan dalam setiap permintaan API selanjutnya untuk otentikasi pengguna dalam mengakses API lain yang ada dalam *dashboard*. Token akan disimpan oleh aplikasi klien (aplikasi web) sebagai referensi otentikasi pengguna.

```
{
    "statusCode": "200",
    "success": true,
    "responseDatetime": "2023-04-12T12:41:42.594346697+07:00",
    "result": {
        "id": 222,
        "nama": "username",
        "token": "f7h4wunq9gwagj9rjg.eyJleHAiOjE2ODEzMjEzMDIsInBhc3N3b3JkIjoiZ3dCTm.tNOGxwcnp3SOlmL2hBY0Vtb3ZYK3crSzRVeGFOT2lteTV0RERoSm5uYV8.
           ta1pIRy9uVUdDbHRqbmxtc3BCMnYwSjh3Um82ZmVibWFzR3J5RFE9PSIsInVzZXJuYW1lIjoidXNlcm5ldGEifQ.mFeJbfMnABxqpkDpTMrWNKDmnbMVhyQhn5lyn4FWHn4",
        "role": "IT_MASTER",
        "privilege": [
            {
                "id": 9,
                "group": "DASHBOARD_MENU",
                "userRole": "VIEW_CHART_REVENUE",
                "status": true
            },
            {
                "id": 12,
                "group": "AQUIRING",
                "userRole": "VIEW_AQUIRING",
                "status": true
            },
            {
                "id": 13,
                "group": "AQUIRING",
                "userRole": "ADD_AQUIRING",
                "status": true
            },
            {
                "id": 16,
                "group": "PAYMENT",
                "userRole": "VIEW_PAYMENT",
                "status": true
            }
        ]
    },
    "message": "Welcome, username"
}
```

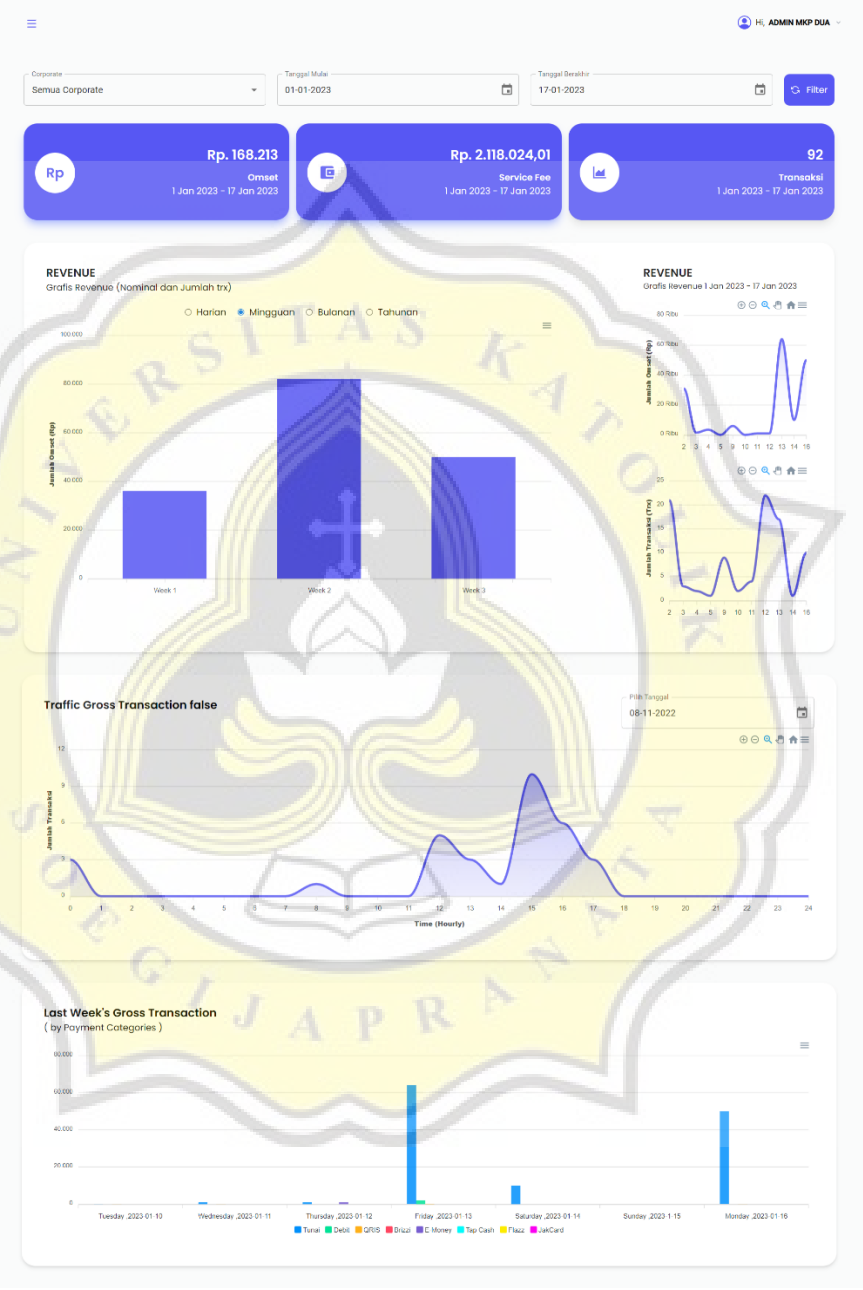

Gambar 4.4 Respon dari API *login*

Selain itu, *username* juga akan disimpan oleh aplikasi klien. *Username* ini akan digunakan untuk menampilkan informasi pengguna pada bagian profil dalam *dashboard*. Informasi tersebut dapat berupa nama pengguna, foto profil, atau informasi lain yang relevan untuk tampilan profil pengguna. *User role* juga akan disimpan sebagai data autentikasi dalam aplikasi klien. *User role* merupakan informasi mengenai peran atau hak akses pengguna dalam sistem. Informasi ini akan digunakan untuk mengatur akses fitur atau bagian-bagian tertentu dalam *dashboard* yang dapat diakses oleh pengguna sesuai dengan perannya.

4.2.2 Akses *dashboard*

Setelah proses *login* berhasil, pengguna akan diarahkan menuju halaman *dashboard* yang menjadi halaman utama dalam aplikasi web *dashboard* yang

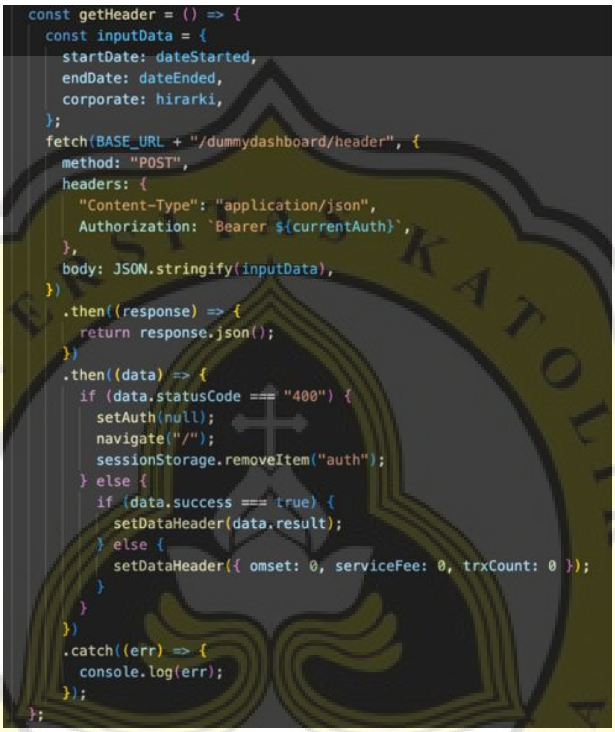
terlihat pada gambar 4.5. Halaman ini berisi ringkasan data, grafik, atau informasi lain yang relevan dengan sistem atau aplikasi yang digunakan. Untuk mengakses informasi tersebut, aplikasi web akan menggunakan API yang akan dieksekusi menggunakan beberapa fungsi pada saat halaman *dashboard* diakses.



Gambar 4.5 Halaman *dashboard*

Fungsi yang dijalankan terlihat pada gambar 4.6 merupakan fungsi yang digunakan untuk melakukan pemanggilan API guna mengambil data untuk header

dashboard. Fungsi ini mungkin menggunakan metode HTTP dalam mengirim permintaan ke API untuk *header dashboard*. Setelah permintaan berhasil, respon dari API akan diterima dan variabel yang telah dideklarasikan sebelumnya akan digunakan untuk menyimpan data yang diperoleh dari respon API. Data yang telah diproses kemudian dapat digunakan dalam komponen *header dashboard* untuk menampilkan informasi yang relevan kepada pengguna.



```
const getHeader = () => {
  const inputData = {
    startDate: dateStarted,
    endDate: dateEnded,
    corporate: hirarki,
  };
  fetch(BASE_URL + "/dummydashboard/header", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      Authorization: `Bearer ${currentAuth}`,
    },
    body: JSON.stringify(inputData),
  })
    .then((response) => {
      return response.json();
    })
    .then((data) => {
      if (data.statusCode === "400") {
        setAuth(null);
        navigate("/");
        sessionStorage.removeItem("auth");
      } else {
        if (data.success === true) {
          setDataHeader(data.result);
        } else {
          setDataHeader({ omset: 0, serviceFee: 0, trxCount: 0 });
        }
      }
    })
    .catch((err) => {
      console.log(err);
    });
};
```

Gambar 4.6 Fungsi API data *header*

Respon yang diberikan dari API *revenue* berupa data yang mencakup ringkasan transaksi yang dikelompokkan dalam jumlah omset, *service fee*, dan jumlah transaksi yang berjalan. Gambar 4.7 memberikan contoh respon yang diberikan, dimana data yang dibutuhkan terringkas dalam data objek *result.*

```
{
    "statusCode": "200",
    "success": true,
    "responseDatetime": "2023-04-13T10:27:02.942976801+07:00",
    "result": {
        "omset": 2166023,
        "serviceFee": 965,
        "trxCount": 131
    },
    "message": "Successfully Fetching Record"
}
```


Gambar 4.7 Respon API *header*

Pada gambar 4.8, terdapat fungsi yang digunakan untuk melakukan pemanggilan API guna mendapatkan data dari *revenue* untuk *dashboard*. Fungsi tersebut mungkin menggunakan metode HTTP untuk mengirim permintaan ke API data *revenue*. Setelah permintaan berhasil, respon dari API akan diterima dan disimpan dalam sebuah variabel yang berfungsi sebagai data yang akan ditampilkan dalam grafik *dashboard*.

```
const getRevenueData = () => {
  modalOpenClose(true);
  const bodyData = {
    startDate: dateStarted,
    endDate: dateEnded,
    corporate: hirarki,
  };
  fetch(BASE_URL + "/dummydashboard/revenue", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      Accept: "application/json",
      Authorization: `Bearer ${currentAuth}`,
    },
    body: JSON.stringify(bodyData),
  })
    .then((response) => {
      return response.json();
    })
    .then((json) => {
      if (json.statusCode === "401") {
        setAuth(null);
        navigate("/");
        modalOpenClose(false);
      } else {
        console.log("Revenue Data Samping", json.result);
        if (json.result === null) {
          setRevenueData([]);
          modalOpenClose(false);
        } else {
          setRevenueData(json.result);
          modalOpenClose(false);
        }
      }
    })
    .catch((err) => {
      console.log(err);
      modalOpenClose(false);
    });
```

Gambar 4.8 Fungsi API data *revenue*

Respon yang diberikan dari API *revenue* pada gambar 4.9 berupa data dalam bentuk *array*. Data tersebut kemudian disimpan dan diolah melalui logika javaScript untuk mendapatkan hasil sesuai dengan kebutuhan tampilan *dashboard*. Proses pengolahan data ini dapat melibatkan manipulasi *array*, penghitungan matematika, pengelompokan data, atau transformasi lainnya sesuai dengan kebutuhan tampilan yang diinginkan.

```
{
  "statusCode": "200",
  "success": true,
  "responseDatetime": "2023-04-12T13:13:09.574317142+07:00",
  "result": [
    {
      "week": "1",
      "count": 1,
      "sum": 1232
    },
    {
      "week": "2",
      "count": 20,
      "sum": 12330
    },
    {
      "week": "3",
      "count": 220,
      "sum": 213120
    },
    {
      "week": "4",
      "count": 330,
      "sum": 3934330
    },
    {
      "week": "5",
      "count": 12,
      "sum": 849333
    }
  ],
  "message": "Successfully Fetching Record"
}
```


Gambar 4.9 Respon API *revenue dashboard*

Gambar 4.10 memperlihatkan fungsi yang digunakan dalam mendapatkan data dari daftar *corporate* yang tersedia melalui pemanggilan API. Fungsi ini mungkin melibatkan penggunaan metode HTTP untuk mengambil data dari API yang ditentukan, dan mengirimkan permintaan ke server untuk mengambil data *corporate* dalam bentuk *array*. Setelah data diterima dari API, data tersebut kemudian disimpan dalam variabel atau struktur data yang sesuai di dalam logika javaScript aplikasi. Data dalam bentuk array ini bisa digunakan untuk berbagai tujuan, seperti mengisi komponen tampilan dalam *dashboard* seperti tabel, grafik, atau komponen UI lainnya yang membutuhkan data *corporate* tersebut.

```
const getDataCorp = () => {
  const bodyData = {
    order: "ASC",
    limit: 90,
    page: 1,
    keyword: "",
    hirarkiId: hirarkiUser,
  };
  fetch(BASE_URL + "/corporate/list", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      Accept: "application/json",
      Authorization: `Bearer ${currentAuth}`,
    },
    body: JSON.stringify(bodyData),
  })
    .then((response) => {
      return response.json();
    })
    .then((json) => {
      if (json.statusCode === "401") {
        setAuth(null);
        navigate("/");
      } else {
        if (json.result != null) {
          setDataCorp(json.result);
        } else {
          setDataCorp([]);
        }
      }
    })
    .catch((err) => {
      console.log(err);
    });
};
```

Gambar 4.10 Fungsi API *corporate list*

Respon yang diberikan dalam API *list corporate* pada gambar 4.11 berupa data *array* yang kemudian akan disimpan dan diolah melalui logika javaScript. Data tersebut digunakan dalam *dashboard* dalam memperlengkapi informasi yang diperlukan oleh komponen yang sesuai dengan kebutuhan tampilan.

Gambar 4.11 Respon dari API *list* data

Pada implementasinya, dalam menggunakan API untuk mengakses data pada halaman *dashboard*, digunakan fungsi useEffect seperti pada gambar 4.12 dimana fungsi ini merupakan salah satu fitur dalam React JS. Fungsi useEffect digunakan untuk menjalankan kode yang terkait dengan API ketika komponen halaman *dashboard* telah dimuat dan tampil dalam DOM. Dalam penggunaan useEffect, kode untuk mengakses API akan ditempatkan dalam fungsi callback yang akan dieksekusi setelah komponen halaman *dashboard* dimuat. Fungsi tersebut akan mengirimkan permintaan API ke server untuk mengambil data yang diperlukan untuk ditampilkan dalam halaman *dashboard*, seperti data ringkasan, data grafik, atau data lain yang relevan. Hasil dari permintaan API akan diolah dan ditampilkan dalam halaman *dashboard* menggunakan logika dari javascript yang telah ditentukan.

```
useEffect(() => {
  getHeader();
  getRevenueData();
  getDataCorp();
}, []);
```

Gambar 4.12 Fungsi useEffect

4.2.3 CRUD Data

Proses CRUD *(Create, Read, Update, Delete)* data dalam aplikasi web *dashboard* terjadi setelah pengguna berhasil *login* dan mengakses *dashboard*. Proses ini memerlukan *user role* yang telah disimpan saat proses *login* sebagai salah satu faktor autentikasi untuk mengakses tampilan aplikasi. Proses pengambilan data dilakukan seperti pada contoh gambar 4.13 dimana data *user role* diambil melalui fungsi dari useStateContext. Variabel kemudian diolah menjadi data boolean sebagai validasi akses bagi pengguna dalam melakukan operasi CRUD terhadap data.

```
90   const [privilege] = useStateContext()
91   const editCorp = privilege.some((item) => {
92     return (
93       item.role === "CORPORATE" &&
94       item.task === "EDIT_CORPORATE"
95     );
96   });
97
98   const deleteCorp = privilege.some((item) => {
99     return (
100      item.role === "CORPORATE" &&
101      item.task === "DELETE_CORPORATE"
102    );
103  });
104
105  const insertCorp = privilege.some((item) => {
106    return (
107      item.role === "CORPORATE" &&
108      item.task === "ADD_CORPORATE"
109    );
110  });
```

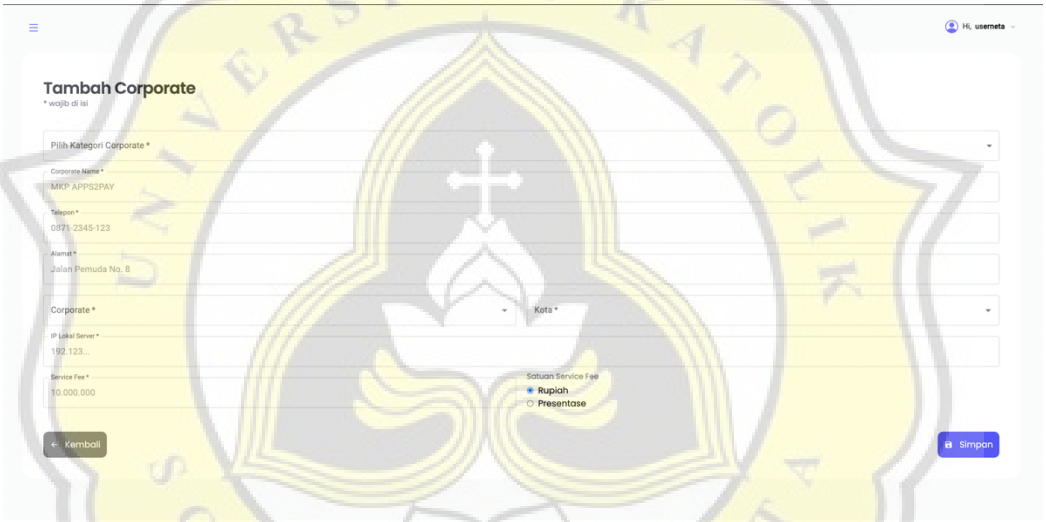
Gambar 4.13 Fungsi pengambilan *user role* untuk validasi

Proses CRUD data dilakukan melalui interaksi pengguna dengan antarmuka aplikasi yang telah dirancang sesuai dengan *user interface* (UI) dan *user experience* (UX). Pengguna hanya dapat mengakses tampilan aplikasi sesuai dengan *user role* yang telah ditentukan saat proses *login*, sehingga pengaturan hak akses dapat

memastikan hanya pengguna yang memiliki otoritas yang dapat melakukan operasi CRUD terhadap data.

a. *Create data*

Pada bagian ini, pengguna dapat menambahkan data baru ke dalam sistem melalui form yang diisi dengan informasi yang diperlukan, seperti yang terlihat pada gambar 4.14. Setelah pengguna mengisi *form*, data tersebut dikirimkan ke sistem untuk diolah sesuai dengan logika bisnis atau proses yang telah ditentukan. Data baru yang dimasukkan oleh pengguna melalui *form* tersebut dapat digunakan untuk menyimpan data ke dalam database, memperbarui tampilan halaman, atau melakukan proses bisnis tertentu sesuai kebutuhan sistem.



Gambar 4.14 *Form* tambah data

Data baru dari form disimpan dalam database aplikasi web *dashboard* melalui API dengan menggunakan fungsi javaScript seperti yang ditunjukkan pada gambar 4.15. Data tersebut dikirim ke API menuju dalam database. Setelah data berhasil disimpan, sistem dapat menggunakannya untuk menampilkan data pada tampilan *dashboard*, melakukan analisis data, atau memproses data sesuai logika bisnis yang ditentukan. Penggunaan API dalam penyimpanan data ke dalam

database memungkinkan interaksi yang efisien antara pengguna dan *database* melalui permintaan dan respon API.

```
const dataInput = (e) => {
  const inputData = {
    uraian: uraian,
    alamat: alamat,
    telepon: telepon,
    servicefee: parseInt(servicefee),
    corporatecategory: parseInt(corpCategory),
    iplocalserver: ipLocal,
    idkota: parseInt(idkota),
    ispercentage: isPercentage,
    parentCID: parentCID,
  };

  fetch(BASE_URL + "/dashboarddummy/add", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      Accept: "application/json",
      Authorization: `Bearer ${currentAuth}`,
    },
    body: JSON.stringify(inputData),
  })
    .then((response) => response.json())
    .then((json) => {
      if (json.statusCode === "401") {
        setAuth(null);
        navigate("/");
        sessionStorage.removeItem("auth");
      } else {
        setStatus(json.success);
        let message = json.message;
        if (json.success === true) {
          notify(message);
        } else if (json.success === false) {
          notifyErr(message);
        }
      }
    })
    .catch((error) => {
      console.log(error);
    });
};
```

Gambar 4.15 Fungsi menambahkan data

Respon dari API untuk menambahkan data berisi pesan dan *status code* yang mengindikasikan hasil dari permintaan yang diberikan. Gambar 4.16 menunjukkan contoh respon yang berisi pesan, *status code*, dan catatan waktu dari permintaan tersebut. Respon ini dapat digunakan untuk memberikan informasi lebih lanjut kepada pengguna atau digunakan dalam proses pemantauan atau

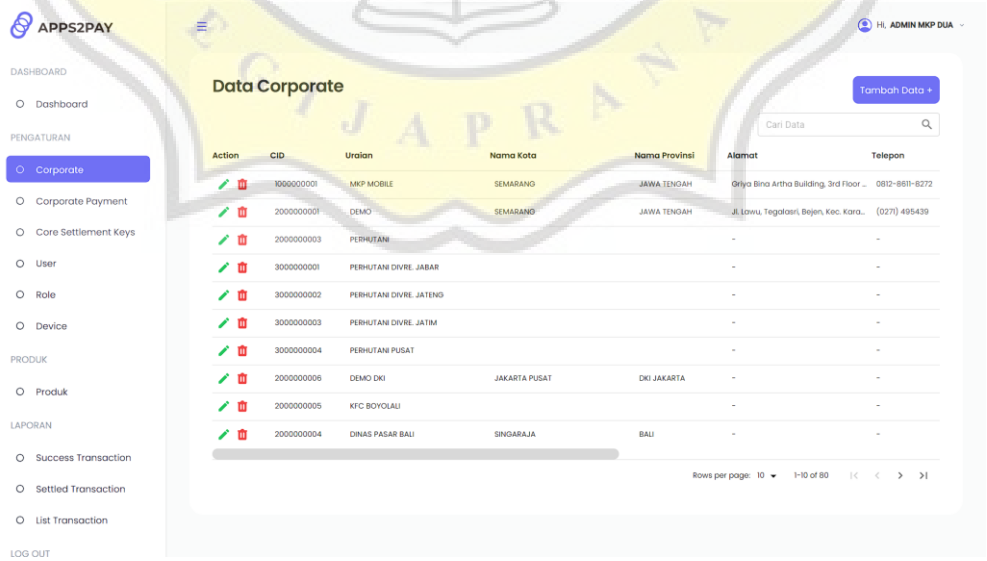
pelacakan operasi yang dilakukan pada data baru yang ditambahkan ke dalam database melalui API.


```
{
    "statusCode": "200",
    "success": true,
    "responseDatetime": "2023-04-13T09:38:47.128816831+07:00",
    "result": 142,
    "message": "Successfully Creating New Record"
}
```


Gambar 4.16 Respon dari API untuk menambahkan data

b. Read data

Pada tahap ini, pengguna dapat melihat data yang sudah ada dalam sistem melalui tampilan yang ditampilkan pada gambar 4.17. Data tersebut dapat ditampilkan dalam berbagai bentuk, seperti tabel, grafik, atau tampilan visual lainnya yang relevan dengan konteks aplikasi web *dashboard* yang sedang digunakan. Tampilan data yang sudah ada ini memungkinkan pengguna untuk mengakses informasi yang relevan, melakukan analisis data, dan membuat keputusan berdasarkan data yang ditampilkan. Tampilan data dalam bentuk yang sesuai juga dapat membantu pengguna untuk memahami data dengan lebih baik dan mengambil tindakan yang diperlukan berdasarkan informasi yang ditampilkan dalam aplikasi web *dashboard*.



Gambar 4.17 Halaman tampilan tabel data

Data yang tampil merupakan data yang didapatkan melalui respon dari API yang kemudian diolah seperti pada gambar 4.18. Pengguna dapat membaca informasi yang telah tersimpan dalam *database*, dan menampilkan data tersebut dalam antarmuka aplikasi yang telah dirancang dengan *user interface (UI)* dan *user experience (UX)* yang baik.

```
const getDataCorp = () => {
  const bodyData = {
    order: order,
    limit: limit,
    page: pagination,
    keyword: search,
    orderBy: header,
    hirarkiId: hirarkiUser,
  };
  fetch(BASE_URL + "/dashboarddummy/list", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      Authorization: `Bearer ${currentAuth}`,
    },
    body: JSON.stringify(bodyData),
  })
    .then((response) => {
      return response.json();
    })
    .then((data) => {
      if (data.statusCode === "401") {
        setAuth(null);
        navigate("/");
        sessionStorage.removeItem("auth");
        modalOpenClose(false);
      } else {
        if (data.result != null) {
          setDataCorp(data.result);
          setCount(data.countData);
        } else {
          setDataCorp([]);
          setCount(0);
        }
      }
    })
    .catch((err) => {
      console.log("err =>", err);
    });
};
```

Gambar 4.18 Fungsi pengambilan data untuk tabel

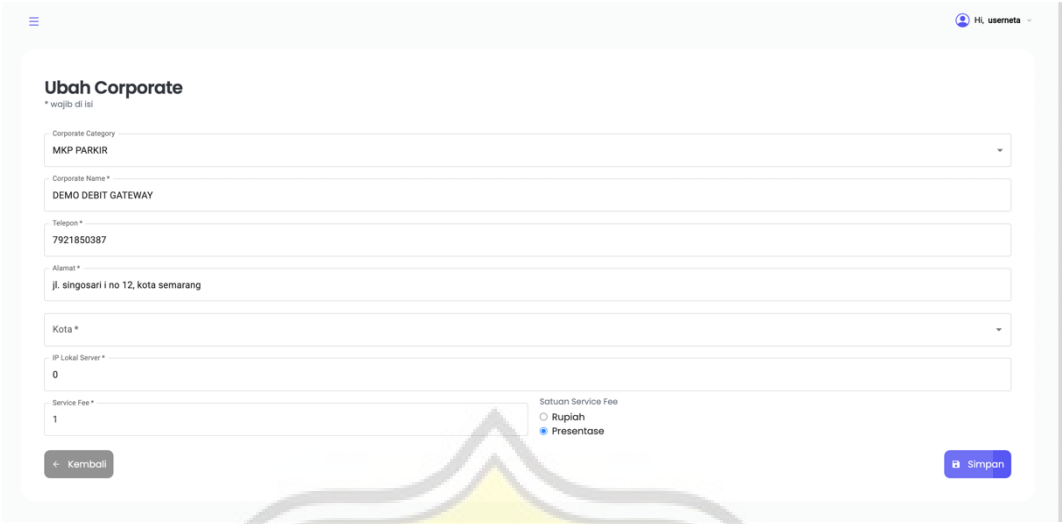
Data yang diterima sebagai respon dari API list corporate pada gambar 4.19 berbentuk array, dan akan diolah menggunakan logika javaScript sebelum disimpan. Data tersebut akan digunakan dalam *dashboard* untuk melengkapkan informasi yang diperlukan oleh komponen tampilan sesuai kebutuhan.

```
{
    "statusCode": "200",
    "success": true,
    "responseDatetime": "2023-04-13T09:40:59.546048251+07:00",
    "countData": 99,
    "result": [
        {
            "id": 1,
            "cid": "5000000005",
            "uraian": "CORPORATE",
            "idKota": 0,
            "alamat": "ALAMAT",
            "telepon": "0812345678",
            "level": 1,
            "gambar": "-",
            "hirarkiId": "1000000001",
            "ipLocalServer": "-",
            "serviceFee": 10,
            "isPercentage": false,
            "corporateCategory": 5,
            "namaCorporateCategory": "MKP",
            "namaKota": "SEMARANG",
            "namaProvinsi": "JAWA TENGAH",
            "latitude": "-6.981352",
            "longitude": "110.413392"
        },
        {
            "id": 2,
            "cid": "2000000005",
            "uraian": "CORPORATE NEW",
            "idKota": 0,
            "alamat": "ALAMAT",
            "telepon": "123456789",
            "level": 2,
            "gambar": "-",
            "hirarkiId": "1000000001/2000000001",
            "ipLocalServer": "-",
            "serviceFee": 0,
            "isPercentage": false,
            "corporateCategory": 5,
            "namaCorporateCategory": "MKP",
            "namaKota": "SEMARANG",
            "namaProvinsi": "JAWA TENGAH",
            "latitude": "-7.603726",
            "longitude": "110.971345"
        }
    ],
    "message": "Successfully Fetching Record"
}
```


Gambar 4.19 Respon API *read data*

c. Update data

Proses ini memungkinkan pengguna untuk memperbarui data yang sudah ada dalam sistem. Pengguna dapat mengedit atau memperbaiki informasi yang sudah tersimpan dalam *database*. Proses ini melibatkan interaksi pengguna dengan antarmuka aplikasi berupa *form* seperti yang ditunjukan pada gambar 4.20 untuk mengubah nilai-nilai data yang ada.



Gambar 4.20 *Form* edit data

Data yang telah diperbarui oleh pengguna akan disimpan kembali ke dalam database melalui API yang dijalankan, seperti yang terlihat pada gambar 4.21. Proses ini melibatkan pengiriman data yang diperbarui ke API yang sesuai, hingga kemudian diolah dan disimpan kembali ke dalam *database* sesuai dengan operasi pembaruan data. Setelah data berhasil disimpan kembali ke dalam database, sistem dapat mengakses data yang telah diperbarui untuk keperluan selanjutnya, seperti menampilkan data yang diperbarui pada tampilan *dashboard* atau melakukan operasi lain yang relevan dengan data yang telah diperbarui.

```
const dataUpdate = () => {
  const inputData = {
    uraian: uraian,
    alamat: alamat,
    telepon: telepon,
    servicefee: parseInt(servicefee),
    corporatecategory: parseInt(corpCategory),
    iplocalserver: ipLocal,
    idkota: parseInt(idkota),
    ispercentage: isPercentage,
    id: parseInt(initialId),
  };

  fetch(BASE_URL + "/dashboarddummy/edit", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      Accept: "application/json",
      Authorization: `Bearer ${currentAuth}`,
    },
    body: JSON.stringify(inputData),
  })
    .then((response) => response.json())
    .then((json) => {
      setStatus(json.success);
      let message = json.message;
      if (json.success === true) {
        notify(message);
      } else if (json.success === false) {
        notifyErr(message);
      }
    })
    .catch((error) => {
      console.log(error);
    });
};
```

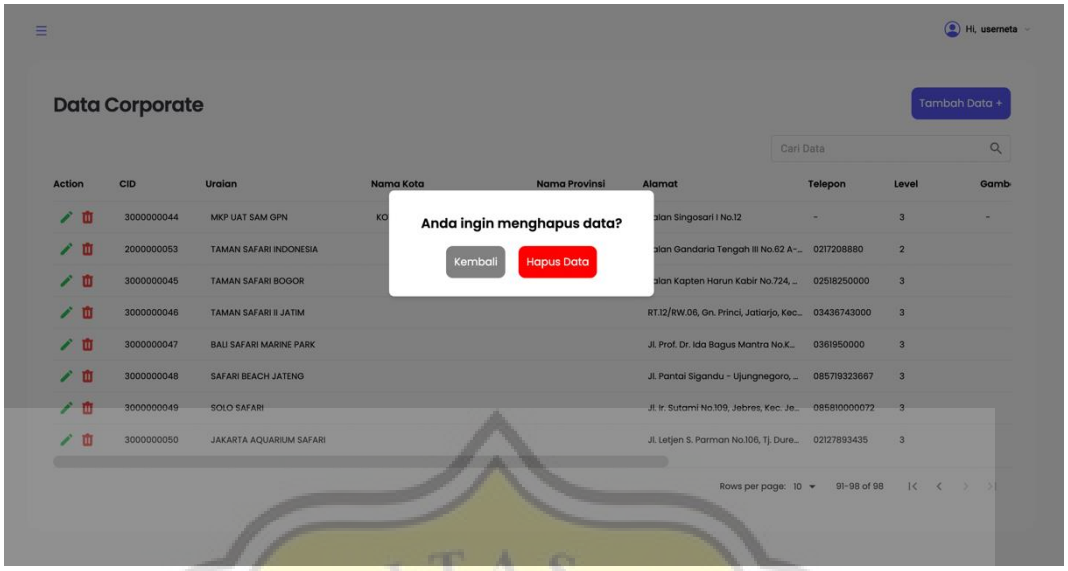
Gambar 4.21 Fungsi API edit data

Respon dari API untuk pembaruan data berisi pesan dan status code yang memberikan informasi mengenai hasil dari permintaan yang telah dilakukan. Contoh respon ini dapat ditemukan pada gambar 4.22 yang mencakup pesan, status code, catatan waktu, dan informasi dari data yang telah diubah dari permintaan tersebut. Respon ini dapat digunakan untuk memberikan informasi tambahan kepada pengguna, atau dapat digunakan dalam proses pemantauan atau pelacakan operasi yang dilakukan pada data baru yang ditambahkan ke dalam database melalui API.

Gambar 4.22 Respon API pembaruan data

d. *Delete data*

Pada proses ini, pengguna dapat menghapus data yang tidak diperlukan dalam sistem. Pengguna dapat memilih data yang ingin dihapus melalui antarmuka aplikasi, dan data yang dihapus akan dihapus dari *database*. Proses ini memerlukan konfirmasi dari pengguna sebelum data benar-benar dihapus untuk menghindari penghapusan data yang tidak disengaja atau salah. Pada proses ini validasi dilakukan dengan bantuan antarmuka aplikasi yang telah dirancang dengan *user interface (UI)* dan *user experience (UX)* seperti pada gambar 4.23.



Gambar 4.23 Validasi penghapusan data

Proses berikutnya dalam menghapus data melibatkan eksekusi fungsi yang memanggil API penghapusan, sebagaimana terlihat pada gambar 4.24. Fungsi ini akan mengirimkan permintaan pada API yang ditentukan untuk menghapus data yang diinginkan dari *database*.

```
2071  const deleteData = (id) => {
2072    const inputData = {
2073      idRole: parseInt(id),
2074    };
2075    fetch(BASE_URL + "/role/delete", {
2076      method: "POST",
2077      headers: {
2078        "Content-Type": "application/json",
2079        Accept: "application/json",
2080        Authorization: `Bearer ${currentAuth}`,
2081      },
2082      body: JSON.stringify(inputData),
2083    })
2084      .then((response) => {
2085        return response.json();
2086      })
2087      .then((json) => {
2088        if (json.success === true) {
2089          notify(json.message);
2090        } else if (json.success === false) {
2091          notifyErr(json.message);
2092        }
2093      })
2094      .catch((err) => {
2095        alert(err);
2096      });
2097  };
```

Gambar 4.24 Fungsi API penghapusan data

Setelah permintaan berhasil dieksekusi, sistem akan menerima respon dari API seperti pada gambar 4.25 yang berisi pesan dan status code yang mengindikasikan hasil dari permintaan penghapusan. Respon ini dapat digunakan untuk memberikan informasi kepada pengguna atau dalam proses pemantauan operasi penghapusan data.

Gambar 4.25 Respon API penghapusan data

4.2.4 Akses fitur lainnya

Setelah pengguna berhasil mengakses *dashboard* dan melewati proses *login* yang melibatkan penyimpanan *user role* sebagai autentikasi akses, pengguna dapat melakukan akses pada fitur-fitur lain dalam aplikasi web *dashboard*. *User role* ini akan digunakan untuk memverifikasi hak akses pengguna terhadap fitur-fitur tertentu dalam aplikasi, sehingga memungkinkan pengguna untuk mengakses fitur-fitur tersebut. Selain fitur CRUD data, terdapat juga fitur lain dalam aplikasi *dashboard* yang memungkinkan pengguna untuk mengekspor data menjadi file PDF seperti yang dicontohkan pada gambar 4.26.

Gambar 4.26 Tampilan halaman menu *export to pdf*

Dan pada gambar 4.27 ditampilkan halaman yang memungkinkan pengguna untuk melakukan ekspor data dalam bentuk file excel. Kedua fitur tersebut memberikan akses bagi pengguna untuk mengunduh data yang ditampilkan dalam aplikasi dalam bentuk file PDF atau Excel. Pengguna dapat memilih data yang ingin diekspor berdasarkan filtrasi data kemudian menjalankan fungsi melalui tombol *filter* yang disediakan dalam antarmuka aplikasi.

Gambar 4.27 Tampilan halaman menu *export excel*

Selain fitur CRUD data, terdapat pula fitur tambahan dalam aplikasi *dashboard* yang memungkinkan pengguna untuk mengkonversi data menjadi file PDF dan Excel. Fitur ini umumnya memberikan akses bagi pengguna untuk mengunduh data yang ditampilkan dalam aplikasi dalam *form*at file PDF atau Excel. Pengguna dapat memilih data yang ingin diekspor berdasarkan filtrasi data yang telah ditentukan, dan melanjutkan proses filtrasi melalui tombol yang tersedia dalam antarmuka aplikasi. Hingga kemudian file akan dihasilkan dan dapat diunduh oleh pengguna sebagai hasil dari proses filtrasi data.

4.2.5 *Logout*

Proses *logout* merupakan langkah yang diambil ketika pengguna ingin mengakhiri sesi penggunaan dan keluar dari akun. Pada proses ini, dilakukan fungsi penghapusan data token yang sebelumnya telah disimpan saat pengguna berhasil melakukan *login*. Langkah ini bertujuan untuk membersihkan data token yang telah

disimpan dalam aplikasi, sehingga tidak dapat diakses oleh pengguna lain. Proses *logout* dilakukan melalui tombol yang tersedia dalam antarmuka aplikasi yang kemudian fungsi penghapusan token dari sistem pada gambar 4.28 akan dijalankan dan sesi penggunaan akan berakhir.

```
152   const { setAuth } = useStateContext();
153   const logOut = () => {
154     setAuth(null);
155     sessionStorage.removeItem("auth");
156   };
```

Gambar 4.28 Fungsi penghapusan token

4.3 Hasil Pengujian Aplikasi

Metode pengujian validitas data dengan menggunakan tabel r hitung sebagai media penilaian adalah cara umum untuk mengukur hubungan antara dua variabel dalam analisis statistik. Tabel r hitung digunakan dalam pengujian validitas Pearson, di mana r hitung dibandingkan dengan nilai kritis dalam tabel untuk menentukan signifikansi hubungan antara variabel[16]. Hasil pengujian dapat membantu peneliti dalam menginterpretasi hasil analisis data.

Hasil pengujian pertama dari validitas Pearson dari 31 data responden, ditemukan beberapa variabel yang tidak memenuhi kriteria validitas. Oleh karena itu, dilakukan eliminasi variabel agar hanya variabel yang valid yang digunakan dalam analisis selanjutnya. Tabel 4.1 menampilkan hasil pengujian validitas yang menunjukkan bahwa terdapat validitas positif yang signifikan antara variabel X dan variabel Y. Dari hasil tersebut dapat disimpulkan bahwa variabel X berpengaruh terhadap variabel Y. Selain itu, hasil pengujian ini juga menunjukkan bahwa nilai signifikansi (p-value) sebesar 0,005 lebih kecil dari alpha level (α) yang telah ditentukan sebelumnya[16].

Table 4.1 Tabel hasil uji validitas

Correlations

		SI1	SI2	PE1	PE2	PE3	EE1	EE2	FC1	FC2	BI1	BI2	Total
SI1	Pearson Correlation	1	0,133	0,106	0,260	0,227	.439*	.361*	0,196	0,269	0,202	0,162	.545**
SI2	Pearson Correlation	0,133	1	0,290	0,158	0,221	0,306	0,124	0,019	0,239	.405*	0,044	.456**
PE1	Pearson Correlation	0,106	0,290	1	0,270	-0,083	0,213	0,248	0,274	0,102	0,324	0,053	.435*

PE2	Pearson Correlation	0,260	0,158	0,270	1	0,303	.437*	.486**	.430*	-0,026	0,093	.380*	.612**
PE3	Pearson Correlation	0,227	0,221	-0,083	0,303	1	.623**	0,235	.356*	-0,078	0,330	0,303	.571**
EE1	Pearson Correlation	.439*	0,306	0,213	.437*	.623**	1	0,244	0,270	0,193	0,324	0,246	.700**
EE2	Pearson Correlation	.361*	0,124	0,248	.486**	0,235	0,244	1	.507**	.358*	0,261	.582**	.707**
FC1	Pearson Correlation	0,196	0,019	0,274	.430*	.356*	0,270	.507**	1	-0,081	0,289	.536**	.613**
FC2	Pearson Correlation	0,269	0,239	0,102	-0,026	-0,078	0,193	.358*	-0,081	1	0,151	0,321	.375*
BI1	Pearson Correlation	0,202	.405*	0,324	0,093	0,330	0,324	0,261	0,289	0,151	1	0,293	.588**
BI2	Pearson Correlation	0,162	0,044	0,053	.380*	0,303	0,246	.582**	.536**	0,321	0,293	1	.628**
Total	Pearson Correlation	.545**	.456**	.435*	.612**	.571**	.700**	.707**	.613**	.375*	.588**	.628**	1

Pada pengujian validitas menggunakan Pearson correlation, nilai korelasi pearson dapat digunakan untuk mengevaluasi validitas data. Caranya dengan membandingkan nilai korelasi pearson yang diperoleh dengan nilai tabel r pada taraf signifikansi 0,05 dan derajat kebebasan n-2[17]. Jika nilai korelasi pearson lebih besar dari nilai tabel r yang sesuai, maka data dianggap valid. Dengan nilai n adalah 31 yang berupa jumlah responden yang didapatkan, maka didapatkan nilai derajat kebebasan yakni 29 sehingga didapatkan nilai tabel r 0,355. Pada tabel 4.2 ditampilkan rangkuman data dari pengujian validitas yang dilakukan dan dapat disimpulkan bahwa terdapat hubungan yang cukup kuat, baik itu positif atau negatif, antara kedua variabel yang diuji, dimana perubahan pada salah satu variabel akan berdampak pada perubahan pada variabel lainnya.

Table 4.2 Tabel rangkuman hasil uji validitas

Variabel	R Hitung	R tabel	Keterangan
SI1	0,545	0,355	Valid
SI2	0,456	0,355	Valid
PE1	0,435	0,355	Valid
PE2	0,612	0,355	Valid
PE3	0,571	0,355	Valid
EE1	0,700	0,355	Valid

EE2	0,707	0,355	Valid
FC1	0,613	0,355	Valid
FC2	0,375	0,355	Valid
BI1	0,588	0,355	Valid
BI2	0,628	0,355	Valid

Pengujian reliabilitas dalam penelitian dapat dipastikan dengan menggunakan Cronbach's alpha. Jika nilai alpha lebih dari 0,6, data dianggap reliabel dan dapat dipercaya. Pengujian ini biasanya dilakukan pada data yang dikumpulkan menggunakan instrumen yang sama, seperti kuesioner atau tes. Dengan nilai alpha yang baik, data dianggap konsisten dan dapat dipercaya untuk analisis data selanjutnya[17]. Berikut merupakan rumus Cronbach's alpha:

$$\alpha = (n / (n-1)) * (1 - (\Sigma s\text{^}2 / St\text{^}2))$$

Keterangan:

α = Hasil reliabilitas yang dicari

n = Jumlah item pertanyaan

Σs^2 = Variansi item

st^2 = Variansi total dari item

Pengujian reliabilitas data yang menggunakan metode Cronbach's alpha dengan hasil nilai 0,792 dan *N of items* sebanyak 11 pada tabel 4.3 menunjukkan tingkat keandalan yang baik. Nilai alpha melebihi 0,6 menunjukkan bahwa data memiliki konsistensi yang tinggi antara item-item dalam instrumen yang digunakan, sehingga data dapat dipercaya sebagai basis analisis data. Dengan *N of items* sebanyak 11, pengujian reliabilitas dilakukan pada sejumlah besar item dalam instrumen, dan hasilnya menunjukkan bahwa data tersebut dapat diandalkan dan valid.

Table 4.3 Tabel hasil uji reliabilitas

Reliability Statistics

Cronbach's Alpha	N of Items
.792	11