

**PENGUNAAN SISTEM INFORMASI KOMUNITAS GAME - UPNVJT**

**LAPORAN**



**Oleh :**

**ALFIAN DORIF MURTADLO      20081010251**

**RICO PUTRA ANUGRAH      20081010024**

**PROGRAM STUDI**

**INFORMATIKA FAKULTAS**

**ILMU KOMPUTER**

**UNIVERSITAS PEMBANGUNAN NASIONAL**

**“VETERAN” JAWA TIMUR**

**2023**

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Berlakang**

Dalam memperkaya pengalaman komunitas game di Universitas Pembangunan Nasional (UPN), kami dengan bangga mempersembahkan sistem website terbaru yang dirancang khusus untuk memonitor progress permainan para pemain. Dengan menggunakan API Express untuk backend dan Front End Next.js untuk antarmuka pengguna,

Sistem ini tidak hanya memberikan fasilitas pengamatan progress, tetapi juga menghadirkan fitur tambahan yang memungkinkan interaksi yang lebih dalam di antara anggota komunitas. Sistem ini memperkenalkan dua peran utama: peran Teacher dan peran Player. Peran Teacher diberikan kepada mereka yang bertanggung jawab untuk mengelola dan mengubah konten game. Dengan peran ini, para pengajar dapat memperbarui informasi, menambahkan tantangan baru, dan membuat perubahan yang diperlukan untuk meningkatkan pengalaman bermain bagi seluruh komunitas. Keberadaan peran Teacher memastikan fleksibilitas dan keberlanjutan konten game yang terus berkembang.

Sementara itu, peran Player memberikan anggota komunitas kemampuan untuk melihat pencapaian mereka dalam permainan. Dengan akses yang terfokus pada pemain, setiap anggota dapat memantau progress pribadi mereka, menciptakan kompetisi yang sehat, dan meningkatkan motivasi untuk mencapai target tertinggi. Peran Player memberikan akses eksklusif kepada informasi permainan pribadi masing-masing anggota, menciptakan pengalaman yang lebih personal dan berfokus pada pencapaian individu.

Melalui integrasi peran Teacher dan Player, sistem ini tidak hanya memberikan kesempatan untuk mengamati progress permainan, tetapi juga menciptakan lingkungan yang memungkinkan pengajaran dan pembelajaran secara kolaboratif di dalam komunitas game UPN. Kami yakin bahwa penggunaan API Express, Front End Next.js, dan peran yang terstruktur ini akan meningkatkan daya tarik dan interaktivitas dalam komunitas game, menciptakan pengalaman bermain yang lebih mendalam dan bermakna bagi semua anggota.

## **BAB II**

### **METODOLOGI**

#### **2.1 Pemilihan Teknologi**

##### **2.1.1 Firestore:**

Firestore dipilih sebagai basis data karena keunggulannya dalam menyediakan database NoSQL yang dapat diakses secara real-time dan mudah diintegrasikan dengan aplikasi web. Model data yang fleksibel dan kemampuan sinkronisasi otomatis membuat Firestore menjadi pilihan yang tepat untuk menyimpan dan memperbarui informasi permainan secara efisien.

##### **2.1.2 Express:**

Express diadopsi sebagai framework backend karena kemudahannya dalam membuat API yang dapat diandalkan dan efisien. Dengan desain yang minimalis dan fleksibel, Express memungkinkan pengembang untuk dengan cepat membangun dan mengelola endpoint API yang diperlukan untuk berkomunikasi dengan frontend dan mobile apps, memberikan kontrol penuh terhadap logika server.

##### **2.1.3 Next.js:**

Pemilihan Next.js sebagai kerangka pengembangan frontend didasarkan pada kemampuannya untuk menyediakan antarmuka pengguna yang responsif dan dinamis. Dengan fitur pre-rendering dan kemampuan untuk membangun aplikasi dengan cepat, Next.js memberikan pengalaman pengguna yang lebih baik. Selain itu, Next.js menyederhanakan proses routing dan manajemen state di sisi klien, membuatnya ideal untuk pengembangan aplikasi web yang kompleks.

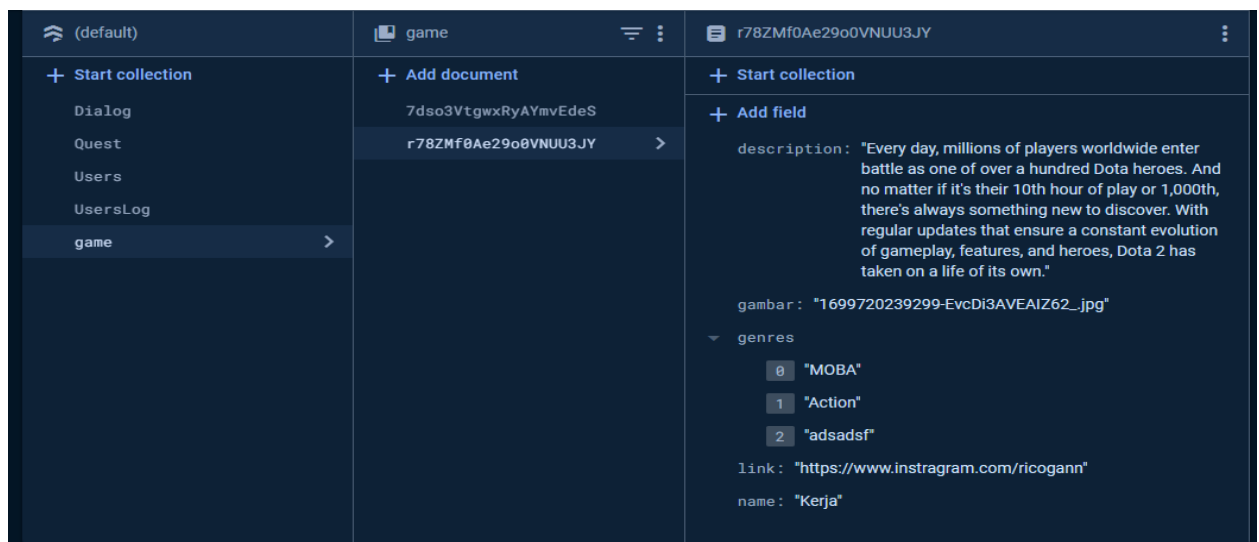
#### **2.2 Struktur Database di Firestore**

Dalam struktur basis data ini, kami menetapkan beberapa koleksi utama yang mencerminkan entitas-entitas kunci dalam konteks komunitas game. Setiap koleksi ini memiliki kaitan dan ketergantungan yang ditentukan untuk mendukung fungsionalitas Website secara menyeluruh. Collection yang dibuat atau biasa dikenal dengan tabel ini

mencakup informasi tentang Userslog, Dialog, Users, Quest, Game, dan entitas lainnya yang relevan. Selain itu, hubungan antar koleksi dijelaskan melalui identifikasi dokumen dan subkoleksi yang terstruktur. Ini menciptakan sistem penataan data yang logis dan memudahkan akses informasi dengan cepat. Basis data ini juga dirancang untuk mendukung pembaruan data secara real-time, memastikan bahwa informasi yang disajikan kepada pengguna selalu terkini.

Dengan pemahaman tentang kerangka kerja struktur database secara umum, selanjutnya kita akan mengeksplorasi setiap dokumen dan kunci informasi yang dihasilkan oleh basis data Firestore untuk memenuhi kebutuhan aplikasi komunitas game yang dinamis dan terkoneksi ini.

### 2.2.1 Collection/Tabel game



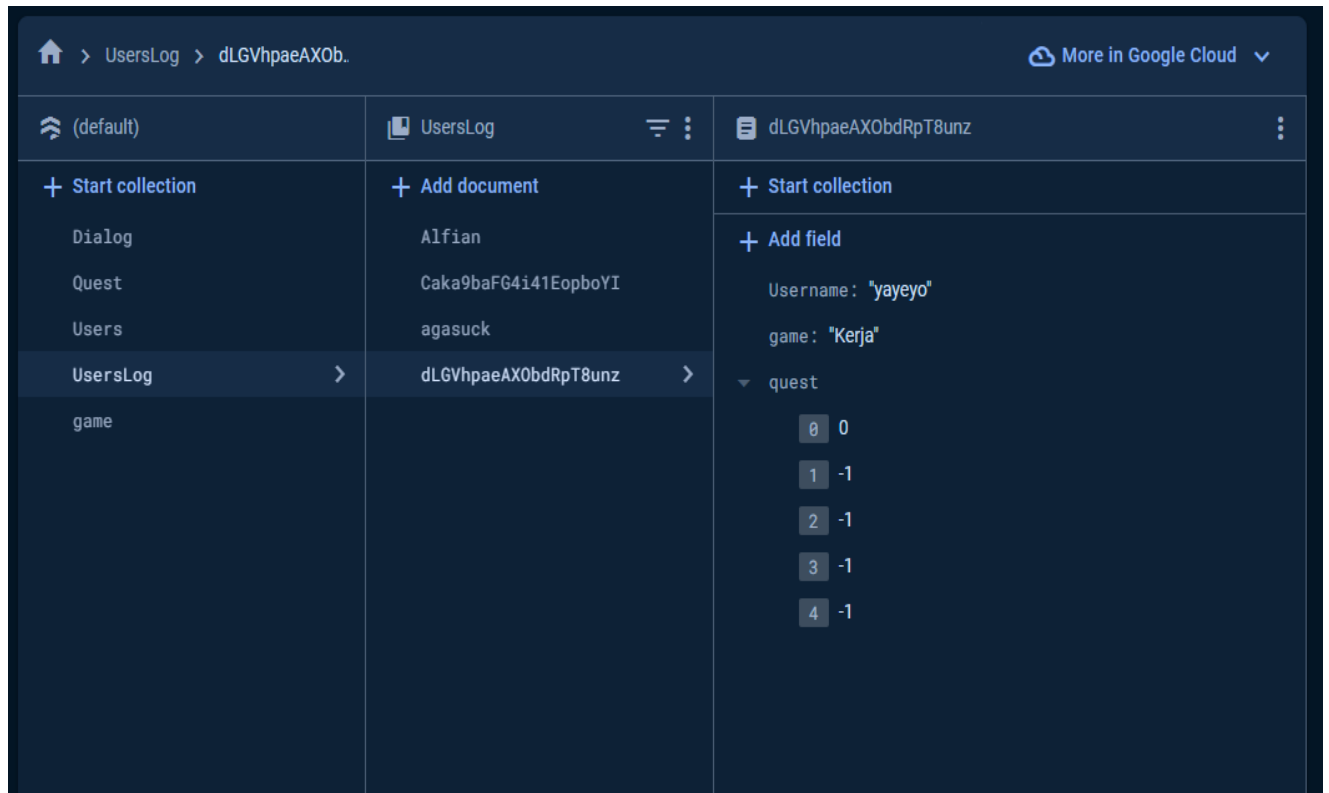
**Gambar 2.1 Collection Tabel**

Collection/Tabel game disini merepresentasikan semua data game yang terdaftar pada website sinarpadu.com dimana akan menyimpan beberapa field diantara lain sebagai berikut :

```
- Collection: game
- Document: <game_document_id>
- Field: description (String)
- Field: gambar (String)
- Field: genres (Array)
  - Element: "MOBA" (String)
  - Element: "Action" (String)
  - Element: "adsadsf" (String)
- Field: link (String)
- Field: name (String)
```

- Collection : Table game
- description: Deskripsi permainan
- gambar: Nama file atau URL gambar permainan.
- genres: Array yang berisi genre-genre permainan.
- link: Tautan ke halaman Download game di Itch.io
- name: Nama permainan (PK)

### 2.2.2 Collection/Tabel Userslog



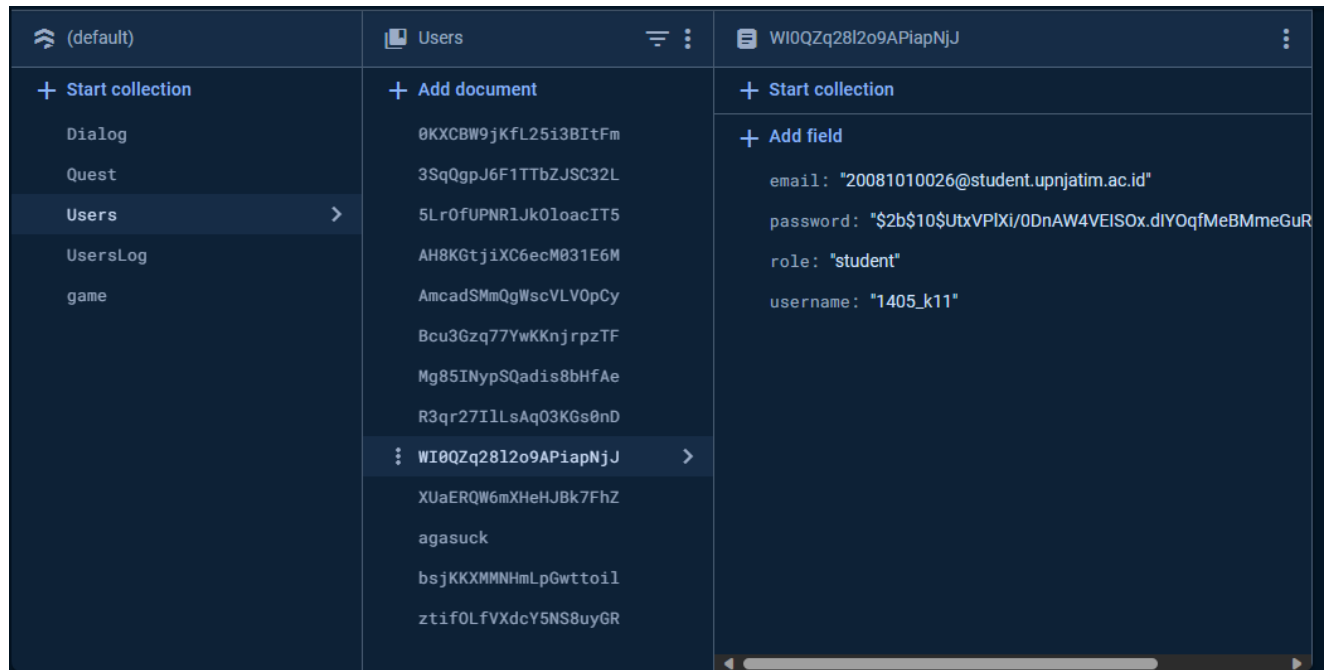
**Gambar 2.2 Collection UsersLog**

Collection/Tabel Userslog disini merepresentasikan semua data log game permainan pemain pada game tersebut yang nantinya dapat diakses pada website sinarpadu.com serta nama game yang bersangkutan dimana akan menyimpan beberapa field diantara lain sebagai berikut :

```
- Collection: Userslog
- Document: <user_document_id>
  - Field: Username (String)
  - Field: game (String)
  - Field: quest (Array)
    - Element: 0 (Number)
    - Element: -1 (Number)
    - Element: -1 (Number)
    - Element: -1 (Number)
    - Element: -1 (Number)
```

- user\_document\_id: Identifikasi unik (PK)
- Username: Nama pengguna dalam bentuk string.(FK dari Collection Users)
- game: Nama permainan yang dihubungkan ke pemain (FK dari Collection game)
- quest: Array yang berisi keadaan quest untuk pengguna tertentu. apabila index 0 bernilai -1 maka Quest1 belum dimulai ,value 0 tahap 1,value 1 tahap 2,value 2 tahap 3, dst sampai value 99 apabila quest selesai. tahapan didalam Quest bisa saja hanya memiliki panjang array 3 tetapi bisa memiliki 99 untuk menandakan jika Quest tersebut selesai.

### 2.2.3 Collection/Tabel Users



Gambar 2.3 Collection Users

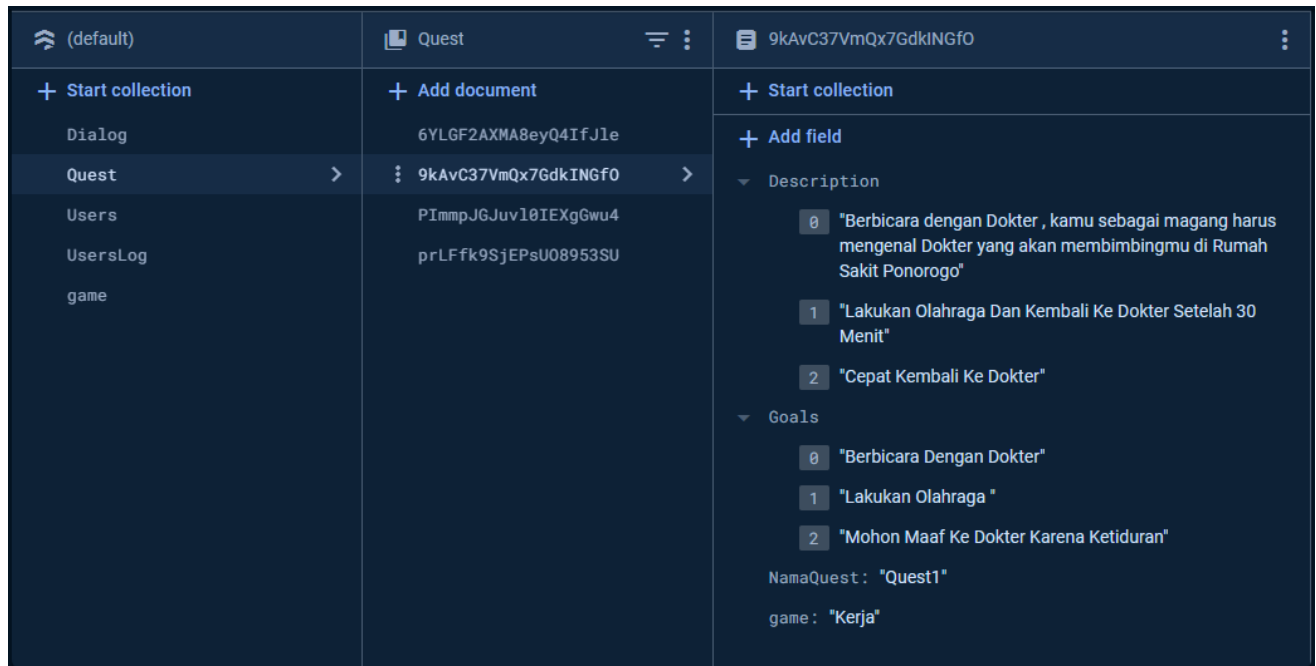
Collection/Tabel Users disini merepresentasikan semua data pemain yang telah mendaftar pada website sinarpadu.com tersebut yang nantinya dapat diakses pada website sinarpadu.com. dimana collection Users akan menyimpan beberapa field diantara lain sebagai berikut :

```
- Collection: users
- Document: <user_document_id>
  - Field: email (String)
  - Field: password (String)
  - Field: role (String)
  - Field: username (String)
```

- user\_document\_id: Identifikasi unik
- email: Alamat email pengguna dalam bentuk string.
- password: Kata sandi yang di-hash, dalam bentuk string.
- role: Peran pengguna, dalam bentuk string. dimana digunakan untuk login apabila role “student” maka akan redirect ke menu Student. apabila “teacher” maka akan redirect ke menu Teacher.
- username: Nama pengguna dalam bentuk string. (PK)

Dalam pengimplementasiannya Users disini tak hanya digunakan untuk login pada website namun juga login pada game yang sudah terdaftar pada website Sinarpadu.com. pada login Game nantinya apabila Users Belum pernah memainkan game tersebut maka UsersLogs data pemain akan langsung diinisialisasi pada level awal dimana nantinya akan mengambil Field username disini untuk digunakan melakukan identifikasi data pada Userslog.

## 2.2.4 Collection/Tabel Quest



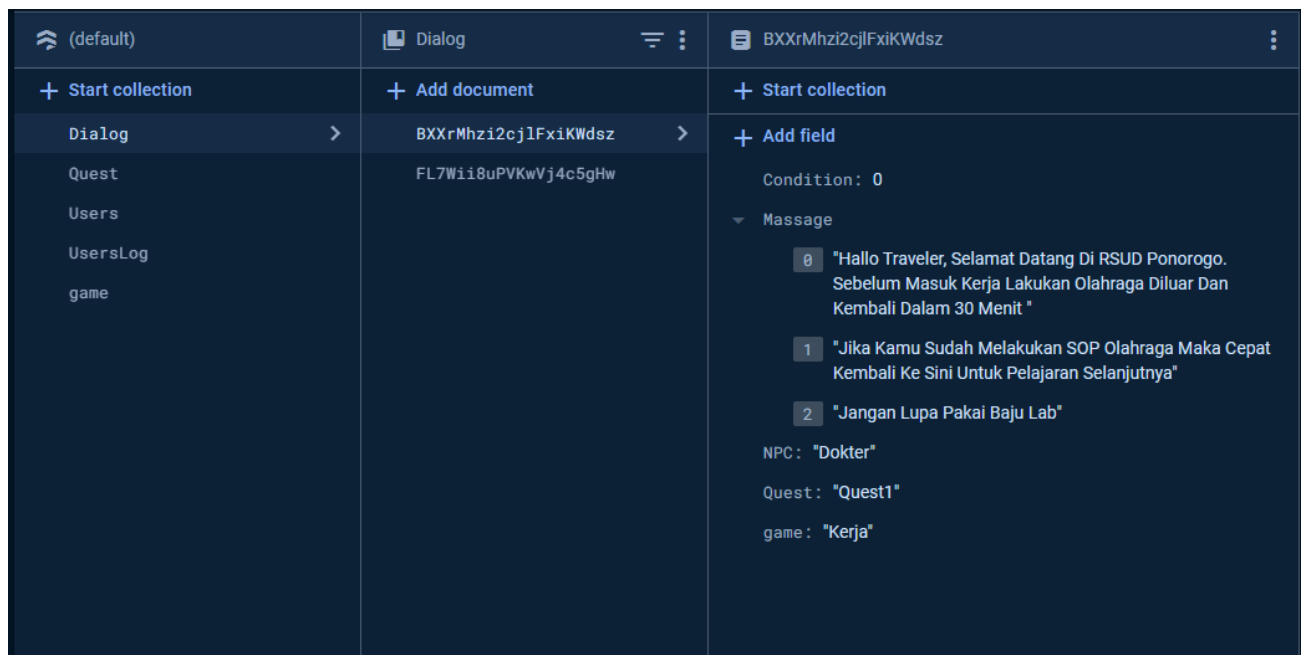
Gambar 2.4 Collection Quest

Collection/Tabel Quest disini merepresentasikan semua data Quest yang telah diinputkan pada website sinarpadu.com tersebut yang nantinya dapat diakses pada website sinarpadu.com. dimana collection Quest akan menyimpan beberapa field diantara lain sebagai berikut :

```
- Collection: Quest
- Document: <quest_document_id>
- Field: Description (Array)
  - Element: "Berbicara dengan Dokter, kamu sebagai magang harus mengena
  - Element: "Lakukan Olahraga Dan Kembali Ke Dokter Setelah 30 Menit" (
  - Element: "Cepat Kembali Ke Dokter" (String)
- Field: Goals (Array)
  - Element: "Berbicara Dengan Dokter" (String)
  - Element: "Lakukan Olahraga" (String)
  - Element: "Mohon Maaf Ke Dokter Karena Ketiduran" (String)
- Field: NamaQuest (String)
- Field: game (String)
```

- quest\_document\_id: Identifikasi unik untuk setiap dokumen dalam koleksi "Quest."
- Description: Array yang berisi deskripsi setiap tugas quest.
- Goals: Array yang berisi tujuan-tujuan dari quest.
- NamaQuest: Nama quest dalam bentuk string. (PK)
- game: Nama permainan yang dihubungkan dengan quest.(FK) dari Collection game

## 2.2.5 Collection/Tabel Dialog



Gambar 2.5 Collection Dialog

Collection/Tabel Dialog disini merepresentasikan semua data Dialog yang telah diinputkan pada website sinarpadu.com tersebut yang nantinya dapat diakses pada website sinarpadu.com. dimana collection Dialog akan menyimpan beberapa field di antara lain sebagai berikut :

```
- Collection: Dialog
- Document: <dialog.document_id>
  - Field: Condition (Number)
  - Field: Message (Array)
    - Element: "Hallo Traveler, Selamat Datang Di RSUD Ponorogo. Sebelum Masuk Kerja Lakukan Olahraga Diluar Dan Kembali Dalam 30 Menit "
    - Element: "Jika Kamu Sudah Melakukan SOP Olahraga Maka Cepat Kembali Ke Sini Untuk Pelajaran Selanjutnya"
    - Element: "Jangan Lupa Pakai Baju Lab" (String)
  - Field: NPC (String)
  - Field: Quest (String)
  - Field: game (String)
```

- dialog\_document\_id: ID\_Unik
- Condition: Nilai numerik yang menentukan kondisi atau tahapan Quest(FK dari Collection Quest (Length array dari Field Goals))
  - Message: Array yang berisi pesan atau dialog yang akan ditampilkan.
  - NPC: Karakter non-pemain yang terlibat dalam dialog.
  - Quest: Nama quest yang terkait dengan dialog. (FK dari Collection Quest)
  - game: Nama permainan yang dihubungkan dengan dialog. (FK dari Collection game)



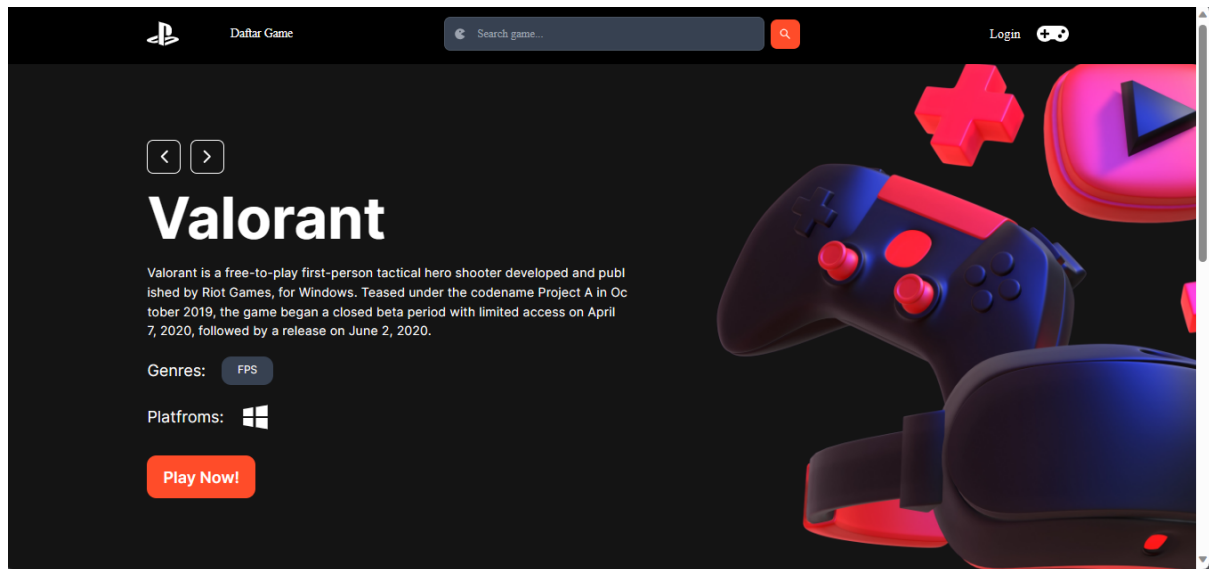
## BAB III

### PENGGUNAAN SYSTEM & SOURCE CODE

#### 3.1 Register

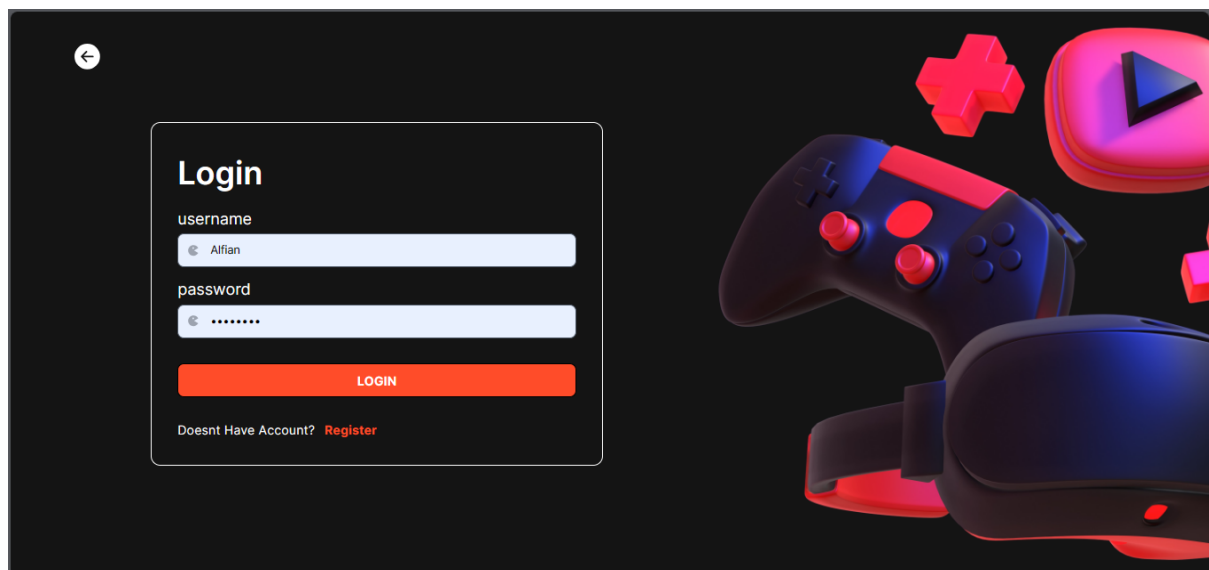
##### 3.1.1 Tata Cara Register

1) Click Tombol Login Pada Pojok Kanan Atas Seperti Pada Gambar 3.1



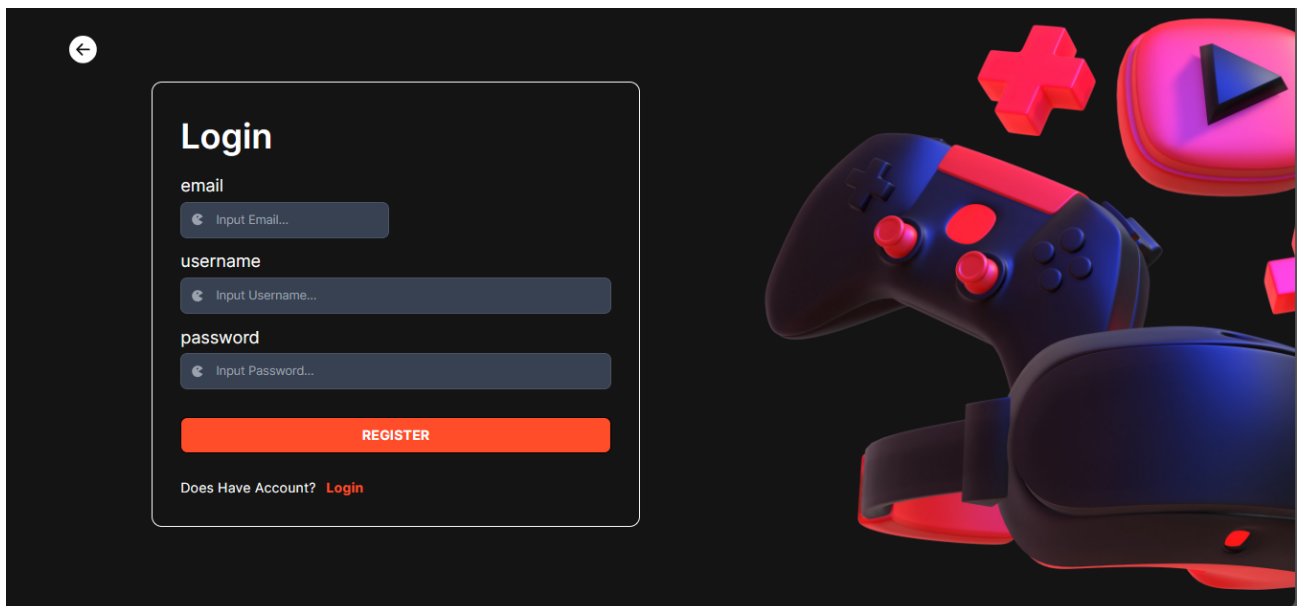
(Gambar 3.1 Home)

2) Click Tombol Register Pada Gambar 3.1



(Gambar 3.2 Login)

- 3) Lakukan Isi Form Pada Registrasi. Jika Username / Email belum terdaftar maka registrasi akan berhasil



(Gambar 3.3 Form Registrasi)

### 3.1.2 Source Code Front End

#### 3.1.2.1 API Register Handling

```
async register(body: AuthRegister): Promise<ReturnAuthRegister> {  
  const register = await fetch(`${this.getBaseUrl()}/api/auth/register`, {  
    method: "POST",  
    headers: {  
      "Content-Type": "application/json",  
    },  
    body: JSON.stringify(body),  
  });  
  
  const response: ReturnAuthRegister = await register.json();  
  if (response.status === false) {  
    throw new Error(response.message);  
  } else {  
    return response;  
  }  
}
```

Function ini digunakan untuk melakukan pemanggilan API dan mengolah data Respon yang nantinya akan dilempar pada button Submit Register.

### 3.1.2.2 Button Register

```
<button
  className="border border-black w-full bg-[#FF4C29] p-2 rounded-lg text-white font-bold uppercase"
  onClick={handleRegister}
>
  REGISTER
</button>
```

```
const handleRegister = async () => {
  const authRegister: AuthRegister = {
    email,
    username,
    password,
  };
  const res = await auth.register(authRegister);
  if (res.status === true) setIsRegister(false);
};
```

Function `handleRegister` digunakan untuk memanggil function API yang sebelumnya sudah dijabarkan pada 3.1.2.1 yang terconnect di FE button Register.

### 3.1.3 Source Code Back End

#### 3.1.3.1 Route Register

```
authController.post("/register", async (req, res) => {
  const result = await m$auth.register(req.body);
  return response.sendResponse(res, result);
});
```

Route Register yang nantinya akan menggunakan Method POST yang berguna untuk melakukan INSERT data dengan routing `"/register"`. nantinya akan mengeksekusi response yang diolah di module `m$auth.register`.

#### 3.1.3.2 Module Register

```
register = async (body) => {
  try {
    //cek jumlah karakter password
    if (body.password.length < 5) {
      return {
        status: false,
        message: "Password harus melebihi 6 karakter!",
      };
    }
  }
}
```

```

const password_hash = bcrypt.hashSync(body.password, 10);

const users = {
  email: body.email,
  password: password_hash,
  username: body.username,
  role: "student",
};

users.password = password_hash;

//cek email sudah terdaftar atau belum
const postDoc = db
  .collection(usersCollection)
  .where("email", "=", users.email);
const docSnapshot = await postDoc.get();

const account = [];

docSnapshot.forEach((doc) => {
  const userData = doc.data();
  account.push(userData);
});

if (account.length > 0) {
  return {
    status: false,
    message: "Akun sudah ada, daftar dengan email yang berbeda",
  };
}

const postDocUsername = db
  .collection(usersCollection)
  .where("username", "=", users.username);
const docSnapshotUsername = await postDoc.get();

const accountUsername = [];

docSnapshotUsername.forEach((doc) => {
  const userData = doc.data();
  accountUsername.push(userData);
});

if (accountUsername.length > 0) {
  return {
    status: false,
    message:
      "Akun sudah ada, daftar dengan username yang berbeda",
  };
}

//input data ke db
const register = await db
  .collection(usersCollection)
  .add(users)

```

```

        .then((docRef) => {
            console.log("Document added with ID:", docRef.id);
        })
        .catch((error) => {
            console.error("Error adding document:", error);
        });

        return {
            status: true,
            code: 201,
            message: "Register success",
        };
    } catch (error) {
        console.error("register auth module Error: ", error);
        return {
            status: false,
            message:
                "Error, check the console log of the backend for what happened",
        };
    }
};

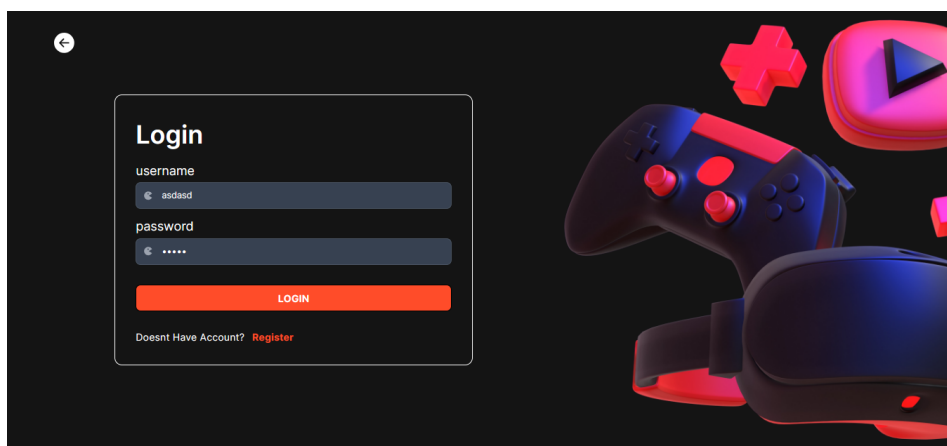
```

Pada Module Register telah dilakukan beberapa tahapan yakni pemberlakuan minimal character pada password yang nantinya akan dilakukan hash, melakukan pencocokan pada data yang diinputkan dengan database dimana apabila terdapat email atau username yang sama maka register tidak dapat dilakukan serta sudah terdapat penerapan error Handling terhadap respon.

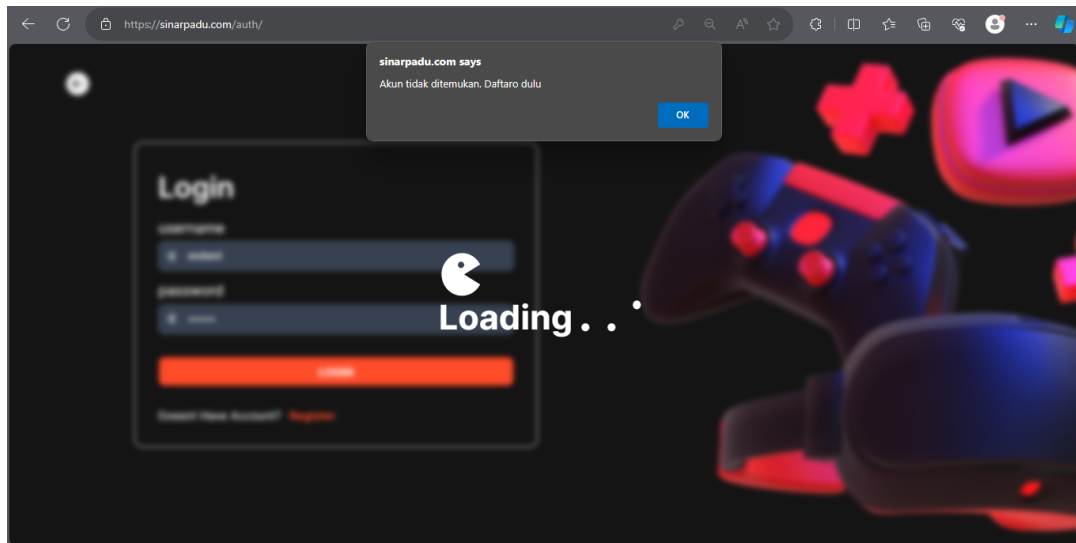
## 3.2 Login

### 3.2.1 Tata Cara Login

- 1) Click Tombol Login Pada Pojok Kanan Atas Seperti Pada Gambar Home 3.1
- 2) Lakukan Input pada Form Login lalu tekan login. Jika Akun terdaftar maka user akan diredirect ke Menu Home. dan Jika Salah Maka Akan Muncul Notifikasi Pada Gambar 3.5



**Gambar 3.4 Form Login**



**Gambar 3.5 Notifikasi Gagal Login**

### 3.2.2 Source Code Front End

#### 3.2.2.1 API Login Handling

```

async login(body: AuthLogin): Promise<ReturnAuthLogin> {
  const login = await fetch(`${this.getBaseUrl()}/api/auth/login`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(body),
  });

  const response: ReturnAuthLogin = await login.json();

  if (response.status === false) {
    return response;
  } else {
    this.setCookie("CERT", response.token, 1);
    const auth = new libAuth();
    if (auth.parse()?.role === "teacher") {
      window.location.href = "/admin";
    } else {
      window.location.href = "/";
    }
    return response;
  }
}

```

Function ini digunakan untuk melakukan pemanggilan API login dan mengolah data Respon yang nantinya akan dilempar pada button Submit login. dimana terdapat Cookies yang nantinya digunakan sebagai akses login yang nantinya disimpan di browser pengguna serta perubahan redirect halaman berdasarkan role yang dimiliki User.

### 3.2.2.2 Button Login

```
<button
  className="border border-black w-full bg-[#FF4C29] p-2 rounded-lg text-white font-bold uppercase"
  onClick={handleLogin}
>
  LOGIN
</button>
```

```
const handleLogin = async () => {
  setIsLoading(true);
  const authLogin: AuthLogin = {
    username,
    password,
  };
  const res = await auth.login(authLogin);
  if (res.status === false) {
    setIsLoading(false);
    alert(res.message);
  }
};
```

Function handleLogin digunakan untuk memanggil function API yang sebelumnya sudah dijabarkan pada 3.2.2.1 yang terconnect di FE button Login. disitu dilakukan pengiriman data username dan password.

### 3.2.3 Source Code Back End

#### 3.2.3.1 Route Login

```
authController.post("/login", async (req, res) => {
  const result = await m$auth.login(req.body);

  return response.sendResponse(res, result);
});
```

Route Register yang nantinya akan menggunakan Method POST yang berguna untuk melakukan POST data dengan routing “/login”. nantinya akan mengeksekusi response yang diolah di module m\$auth.login.

### 3.2.3.2 Module Login

```
login = async (body) => {
  try {
    const users = {
      username: body.username,
      password: body.password,
    };

    console.log(users);

    const postDoc = db
      .collection(usersCollection)
      .where("username", "==", users.username);
    const docSnapshot = await postDoc.get();

    const account = [];

    docSnapshot.forEach((doc) => {
      const userData = doc.data();
      const userId = doc.id;
      account.push({ id: userId, data: userData });
    });

    if (account.length == 0) {
      return {
        status: false,
        message: "Akun tidak ditemukan. Daftaro dulu",
      };
    }

    if (!bcrypt.compareSync(body.password, account[0].data.password)) {
      return {
        status: false,
        code: 401,
        error: "Wrong password",
      };
    }

    const payload = {
      id: account[0].id,
      email: account[0].data.email,
      username: account[0].data.username,
      role: account[0].data.role,
    };

    const token = jwt.sign(payload, "jwt-secret-code", {
      expiresIn: "8h",
    });

    return {
      status: true,
      message: "Login success",
      code: 200,
      token: token,
    };
  } catch (error) {
```



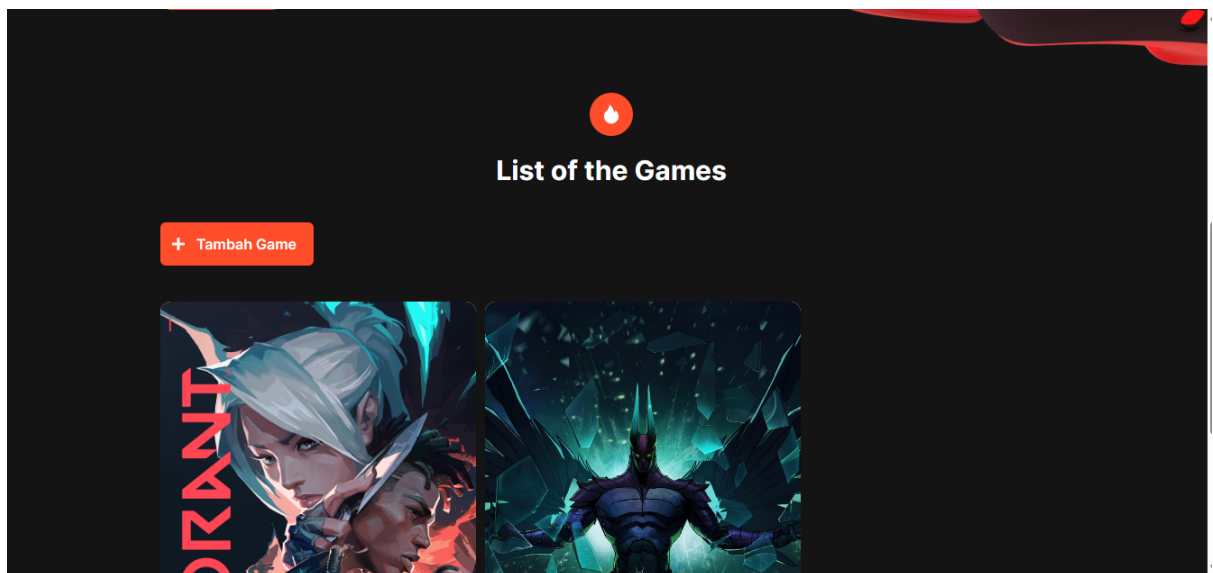
```
console.error("register auth module Error: ", error);
return {
  status: false,
  message:
    "Error, check the console log of the backend for what happened",
};
}
};
```

Pada Module Login telah dilakukan beberapa tahapan yakni digunakan untuk proses login pengguna. Pertama, ia ambil informasi login dari input, cari pengguna di Firestore berdasarkan username, dan periksa kecocokan password dengan bcrypt. Jika berhasil, hasilkan token JWT dengan info pengguna dan kirim respons sukses dengan token. Jika tidak, kirim pesan kesalahan yang sesuai. Kode ini memanfaatkan async/await, query Firestore, bcrypt, dan JWT untuk autentikasi, serta menangani kesalahan.

### 3.3 Game (Create)

#### 3.3.1 Tata Cara Create Game

- 1) Lakukan login terlebih dahulu menggunakan akun admin
- 2) Click Tambah Game Pada Menu Home Lihat Gambar 3.6



Gambar 3.6 Menu Home Setelah Login Sebagai Admin

- 3) Isi Form Sesuai inputan yang ada. Apabila game tersebut sudah terdapat pada database maka penambahan game gagal dan sebaliknya.

The screenshot shows a web browser window with the URL <https://sinarpadu.com/admin/create/>. The page is titled "Create Game Page". It contains the following form fields:

- Judul Game:** A text input field with the placeholder "Input Judul Game ...".
- Deskripsi:** A text input field with the placeholder "Input Deskripsi ...".
- Genre:** A text input field with the placeholder "Action, Adventure, RPG ...".
- Link Game:** A text input field containing the URL "https://www.instagram.com/ricogann".
- Poster:** A file upload section with a "Choose file" button and the text "No file chosen".

At the bottom of the form is a red button labeled "Simpan". The background of the page features a dark theme with a game controller and VR headset illustration.

**Gambar 3.7 Form Create Game**

### 3.3.2 Source Code Front End

#### 3.3.2.1 API Create Game Handling

```

async create(data: FormData) {
  const game = await fetch(`${this.getBaseUrl()}/api/game/add`, {
    method: "POST",
    body: data,
  });

  const response = await game.json();

  if (response.status === false) {
    throw new Error("Failed to create game data");
  } else {
    window.location.href = "/admin";
  }
}

```

Function ini digunakan untuk melakukan pemanggilan API create game dan mengolah data Respon yang nantinya akan dilempar pada button simpan. disini menggunakan Method Post karena akan melakukan POST data.

#### 3.3.2.2 Button Create Game (Simpan)

```

<button
  className="py-2 px-5 mt-5 rounded-xl bg-red-500"
  onClick={handleCreate}
>
  <h1 className="text-[15px] font-semibold">Simpan</h1>
</button>

```

```

const handleCreate = async () => {
  setIsLoading(true);
  const formData = new FormData();
  if (
    data.name === "" ||
    data.description === "" ||
    data.genres.length === 0 ||
    data.link === "" ||
    gambarName === ""
  ) {
    setIsLoading(false);
    alert("Data tidak boleh kosong");
    return;
  } else {
    formData.append("name", data.name);
    formData.append("description", data.description);
    formData.append("genres", data.genres.join(","));
    formData.append("link", data.link);
    formData.append("gambar", data.gambar as Blob);

    const game = new Game();
    const response = await game.create(formData);
  }
};

```

Function `handleCreate` digunakan untuk memanggil function API yang sebelumnya sudah dijabarkan pada 3.3.2.1 yang terconnect di FE button Login. disitu dilakukan pengiriman data `name`, `description`, `genres`, `link`, dan `gambar`.

### 3.3.3 Source Code Back End

#### 3.3.3.1 Route Create Game

```

gameController.post("/add", upload.single("gambar"), async (req, res) => {
  const result = await m$game.addGame(req.body, req.file);

  return response.sendResponse(res, result);
});

```

Route Register yang nantinya akan menggunakan Method POST yang berguna untuk melakukan POST data dengan routing `"/add"`. nantinya akan mengeksekusi response yang diolah di module `m$game.addGame`.

### 3.3.3.2 Module addGame

```
addGame = async (body, file) => {
  const game = {
    name: body.name,
    description: body.description,
    gambar: file ? file.filename : null,
    genres: body.genres.split(","),
    link: body.link,
  };

  const postDoc = db
    .collection(gameCollection)
    .where("name", "=", game.name);
  const docSnapshot = await postDoc.get();

  const dbGame = [];

  docSnapshot.forEach((doc) => {
    const userData = doc.data();
    const userId = doc.id;
    dbGame.push({ id: userId, data: userData });
  });

  if (dbGame > 0) {
    return {
      status: false,
      message: "Game sudah terdaftar",
    };
  }

  const register = await db
    .collection(gameCollection)
    .add(game)
    .then((docRef) => {
      console.log("Document added with ID:", docRef.id);
    })
    .catch((error) => {
      console.error("Error adding document:", error);
    });

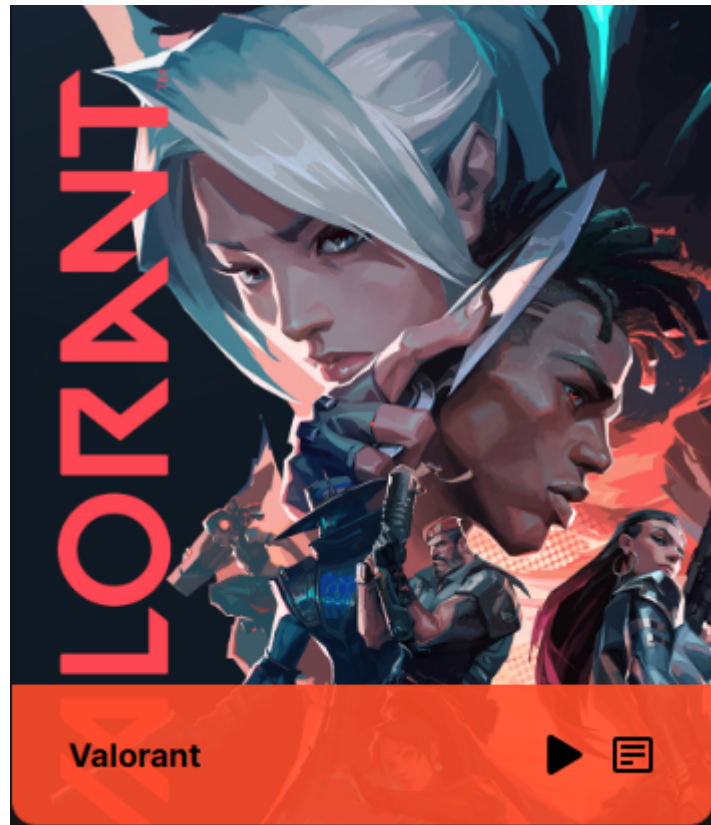
  return {
    status: true,
    code: 200,
    message: "Add game success",
  };
};
```

Fungsi addGame digunakan untuk menambahkan game baru ke database. Informasi game diambil dari input, dan dilakukan pengecekan apakah game dengan nama yang sama sudah terdaftar. Jika sudah, mengembalikan pesan kesalahan; jika belum, menambahkan game baru ke database dan memberikan respons sukses

### 3.4 Game (Update)

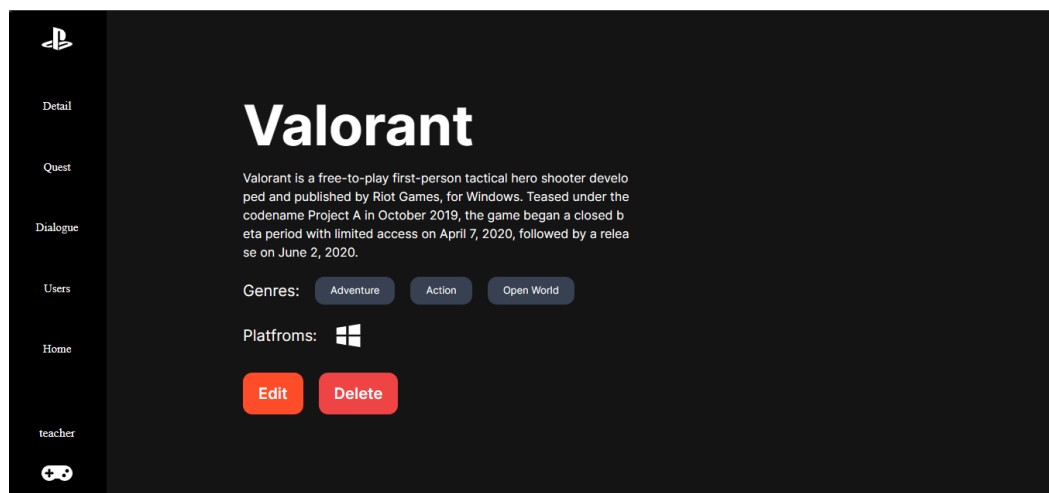
#### 3.4.1 Tata Cara Update Game

- 1) Lakukan login terlebih dahulu menggunakan akun admin
- 2) Click icons detail pada menu game yang diinginkan di home lihat Gambar 3.8



Gambar 3.8 Icon Details

- 4) Tekan Tombol Edit untuk mengarah ke form edit seperti pada Gambar 3.9



Gambar 3.9 Page Detail Game

- 5) Ubah data game yang diinginkan sesuai dengan inputan form pada gambar 3.10 kemudian klik tombol Simpan.

**Edit Page**

**Judul Game**  
Valorant

**Deskripsi**  
Valorant is a free-to-play first-person tactical hero shooter develop

**Genre**  
FPS

**Link Game**  
https://instagram.com/ricogann

**Poster**  
Choose file 1699724060643-MV5BNmNhM2NjMTgtNmlyZC00ZmVjLTk4YWVhZmZjNGY2NThtNDhkXkEyXkFqcGdeQXVyODU4MDU1NjU...

**Simpan**

Gambar 3.10 Form Edit Game

### 3.4.2 Source Code Front End

#### 3.4.2.1 API Handling Edit Game

```
async update(name: string, data: FormData) {
  const game = await fetch(
    `${this.getBaseUrl()}/api/game/update/${name}`,
    {
      method: "PUT",
      body: data,
    }
  );

  const response = await game.json();

  if (response.status === false) {
    throw new Error("Failed to update game data");
  } else {
    window.location.href = "/admin";
  }
}
```

Function ini digunakan untuk melakukan pemanggilan API Edit dan mengolah data Respon yang nantinya akan dilempar pada button Simpan di form Edit. dengan Method PUT untuk melakukan Edit / perubahan data yang telah ada.

### 3.4.2.2 Button Simpan Untuk Form Edit

```
<button
  className="py-2 px-5 mt-5 rounded-xl bg-red-500"
  onClick={handleEdit}
>
  <h1 className="text-[15px] font-semibold">Simpan</h1>
</button>
```

```
const handleEdit = async () => {
  setIsLoading(true);
  const formData = new FormData();
  const game = new Game();
  if (
    data.name === "" ||
    data.description === "" ||
    data.genres.length === 0 ||
    data.link === "" ||
    data.gambar === ""
  ) {
    alert("Data tidak boleh kosong");
    setIsLoading(false);
    return;
  } else if (file === undefined) {
    formData.append("name", data.name);
    formData.append("description", data.description);
    formData.append("genres", data.genres as unknown as string);
    formData.append("link", data.link);
    formData.append("gambar", "");
    formData.append("oldGambar", data.gambar as string);
    if (!name) return;
    await game.update(name, formData);
  } else {
    formData.append("name", data.name);
    formData.append("description", data.description);
    formData.append("genres", data.genres as unknown as string);
    formData.append("link", data.link);
    formData.append("gambar", file);
    formData.append("oldGambar", data.gambar as string);
    if (!name) return;
    await game.update(name, formData);
  }
};
```

Function `handleEdit` digunakan untuk memanggil function API yang sebelumnya sudah dijabarkan pada 3.4.2.1 yang terconnect di FE button Edit Game “Simpan”. disitu dilakukan Pembaharuan data name,description,genres,link, dan gambar. serta penghapusan gambar yang lama di directory server.

### 3.4.3 Source Code Back End

#### 3.4.3.1 Route Edit Game

```
gameController.put(  
  "/update/:name",  
  upload.single("gambar"),  
  async (req, res) => {  
    const result = await m$game.updateGame(  
      req.params.name,  
      req.body,  
      req.file  
    );  
    return response.sendResponse(res, result);  
  }  
);
```

Route Update yang menggunakan Method PUT yang berguna untuk melakukan Pembaharuan data dengan routing “/update/:name”. nantinya akan mengeksekusi response yang diolah di module m\$game.updateGame. disini juga menggunakan multer gambar untuk mengolah inputan gambar dari data string yang di parsing.

#### 3.4.3.2 Module UpdateGame

```
updateGame = async (name, body, file) => {  
  const updateData = await db  
    .collection(gameCollection)  
    .where("name", "=", name)  
    .get();  
  
  const gameData = [];  
  
  updateData.forEach((doc) => {  
    const data = doc.data();  
    const id = doc.id;  
    gameData.push({ id, data });  
  });  
  
  if (!file) {  
    const game = {  
      name: body.name,  
      description: body.description,  
      gambar: body.oldGambar,  
      genres: body.genres.split(","),  
      link: body.link,  
    };  
  
    await db  
      .collection(gameCollection)  
      .doc(gameData[0].id)
```



```

        .update(game);
    } else {
        fs.unlinkSync(`./src/public/${body.oldGambar}`);
        const game = {
            name: body.name,
            description: body.description,
            gambar: file ? file.filename : gameData.gambar,
            genres: body.genres.split(","),
            link: body.link,
        };

        await db
            .collection(gameCollection)
            .doc(gameData[0].id)
            .update(game);
    }

    return {
        status: true,
        code: 200,
        message: "data updated successfully",
    };
};

```

Fungsi updateGame digunakan untuk memperbarui data game dalam database. Pertama, melakukan pencarian data game berdasarkan nama, kemudian memperbarui informasi game sesuai dengan input yang diberikan, termasuk mengelola file gambar baru dan menghapus file gambar lama jika ada. Akhirnya, mengembalikan respons sukses.

### 3.5 Game (Delete)

#### 3.5.1 Tata Cara Update Game

- 1) Lakukan login terlebih dahulu menggunakan akun admin
- 2) Click icons detail pada menu game yang diinginkan di home lihat Gambar 3.8
- 3) Tekan Tombol Delete untuk menghapus game seperti pada Gambar 3.9

#### 3.5.2 Source Code Front End

##### 3.5.2.1 API Handling Delete Game

```

async delete(name: string) {
    const game = await fetch(
        `${this.getBaseUrl()}/api/game/delete/${name}`,
        {
            method: "DELETE",
        }
    );

    const response = await game.json();

    if (response.status === false) {
        throw new Error("Failed to delete game data");
    } else {
        window.location.href = "/admin";
    }
}

```

Function ini digunakan untuk melakukan pemanggilan API Delete dan mengolah data Respon yang nantinya akan dilempar pada button Delete. dengan Method DELETE untuk melakukan delete data yang telah ada.

### 3.5.2.2 Button Delete Game

```
<button
  className="py-3 px-5 mt-2 rounded-xl bg-red-500"
  onClick={handleDelete}
>
  <h1 className="text-[20px] font-semibold">
    Delete
  </h1>
</button>
```

```
const handleDelete = async () => {
  setIsLoading(true);
  const game = new Game();
  if (name) {
    await game.delete(name);
  }
  setIsLoading(false);
};
```

Function handleDelete digunakan untuk memanggil function API yang sebelumnya sudah dijabarkan pada 3.5.2.1 yang terconnect di FE button Delete Game “Delete”. disitu dilakukan Penghapusan data.

## 3.5.3 Source Code Back End

### 3.5.3.1 Route Delete Game

```
gameController.delete("/delete/:name", async (req, res) => {
  const result = await m$game.deleteGame(req.params.name);

  return response.sendResponse(res, result);
});
```

Route Delete yang menggunakan Method DELETE yang berguna untuk melakukan penghapusan data dengan routing “/delete/:name”. nantinya akan mengeksekusi response yang diolah di module m\$game.deleteGame.

### 3.5.3.2 Module Delete Game

```
deleteGame = async (name) => {
  const deleteData = await db
    .collection(gameCollection)
    .where("name", "=", name)
```

```

        .get();

const deleteUsersLogData = await db
    .collection("UsersLog")
    .where("game", "==", name)
    .get();

const deleteQuestData = await db
    .collection("Quest")
    .where("game", "==", name)
    .get();

const deleteDialogueData = await db
    .collection("Dialog")
    .where("game", "==", name)
    .get();

const gameData = [];
const usersLogData = [];
const questData = [];
const dialogueData = [];

deleteUsersLogData.forEach((doc) => {
    const data = doc.data();
    const id = doc.id;
    usersLogData.push({ id, data });
});

deleteQuestData.forEach((doc) => {
    const data = doc.data();
    const id = doc.id;
    questData.push({ id, data });
});

deleteDialogueData.forEach((doc) => {
    const data = doc.data();
    const id = doc.id;
    dialogueData.push({ id, data });
});

deleteData.forEach((doc) => {
    const data = doc.data();
    const id = doc.id;
    gameData.push({ id, data });
});

fs.unlinkSync(`./src/public/${gameData[0].data.gambar}`);

await db.collection(gameCollection).doc(gameData[0].id).delete();

usersLogData.forEach(async (doc) => {
    await db.collection("UsersLog").doc(doc.id).delete();
});
questData.forEach(async (doc) => {
    await db.collection("Quest").doc(doc.id).delete();
});

```

```

dialogueData.forEach(async (doc) => {
    await db.collection("Dialog").doc(doc.id).delete();
});

return {
    status: true,
    code: 200,
    message: "data deleted successfully",
};
};

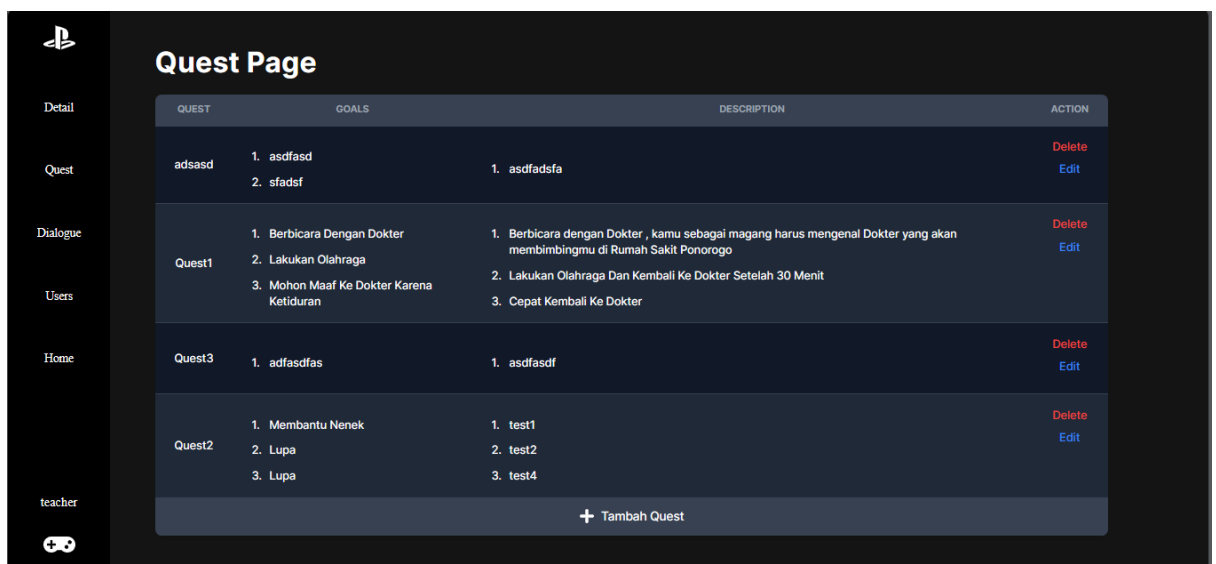
```

Fungsi deleteGame digunakan untuk menghapus data game beserta data terkait (UsersLog, Quest, Dialogue) dari database. Pertama, melakukan pencarian data berdasarkan nama game, dan kemudian menghapus gambar terkait dari direktori. Selanjutnya, menghapus data game dan data terkait lainnya. Akhirnya, mengembalikan respons sukses.

### 3.6 Quest (Add/Insert)

#### 3.6.1 Tata Cara Insert Quest

- 1) Lakukan login terlebih dahulu menggunakan akun admin
- 2) Click icons detail pada menu game yang diinginkan di home lihat Gambar 3.8
- 3) Pilih “Quest” pada sidebar kemudian tekan Tambah Quest seperti pada Gambar 3.11 kemudian akan melakukan direct ke form tambah quest.



The screenshot shows a web application interface for managing quests. On the left is a sidebar menu with icons and labels for 'Detail', 'Quest', 'Dialogue', 'Users', 'Home', and 'teacher'. The main area is titled 'Quest Page' and contains a table with the following data:

QUEST	GOALS	DESCRIPTION	ACTION
adsasd	1. asdfasd 2. sfadsf	1. asdfadsfa	Delete Edit
Quest1	1. Berbicara Dengan Dokter 2. Lakukan Olahraga 3. Mohon Maaf Ke Dokter Karena Ketiduran	1. Berbicara dengan Dokter , kamu sebagai magang harus mengenal Dokter yang akan membimbingmu di Rumah Sakit Ponorogo 2. Lakukan Olahraga Dan Kembali Ke Dokter Setelah 30 Menit 3. Cepat Kembali Ke Dokter	Delete Edit
Quest3	1. adfasdfas	1. asdfasdf	Delete Edit
Quest2	1. Membantu Nenek 2. Lupa 3. Lupa	1. test1 2. test2 3. test4	Delete Edit

At the bottom of the table is a button labeled '+ Tambah Quest'.

Gambar 3.11 Table Quest

- 4) Isi Inputan berdasarkan urutan kolom jika sudah tekan save seperti pada gambar 3.12 yang nantinya berfungsi untuk menyimpan data Quest.

**Quest Page**

QUEST	GOALS	DESCRIPTION	ACTION
adsasd	1. asdfasd 2. sfadsf	1. asdfadsfa	Delete Edit
Quest1	1. Berbicara Dengan Dokter 2. Lakukan Olahraga 3. Mohon Maaf Ke Dokter Karena Ketiduran	1. Berbicara dengan Dokter , kamu sebagai magang harus mengenal Dokter yang akan membimbingmu di Rumah Sakit Ponorogo 2. Lakukan Olahraga Dan Kembali Ke Dokter Setelah 30 Menit 3. Cepat Kembali Ke Dokter	Delete Edit
Quest3	1. adsfasfas	1. asdfasdfs	Delete Edit
Quest2	1. Membantu Nenek 2. Lupa 3. Lupa	1. test1 2. test2 3. test4	Delete Edit

Form at the bottom:

QUEST:  GOALS:  + Tambah Goals DESCRIPTION:  + Tambah Deskripsi

+ Save

Gambar 3.12 Table Insert Quest

### 3.6.2 Source Code Front End

#### 3.6.2.1 API Handling Add/Insert Quest

```

async add(quest: Quests) {
  const add = await fetch(`${this.getBaseUrl()}/api/quest/add`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(quest),
  });

  const response = await add.json();

  if (response.status === false) {
    throw new Error("Failed to add quest");
  } else {
    window.location.reload();
  }
}

```

Function add digunakan untuk melakukan pemanggilan API Create dan mengolah data Respon yang nantinya akan dilempar pada button Save. dengan Method Create untuk melakukan penambahan data yang baru.

#### 3.6.2.2 Button Save

```

<button
  className={` ${
    isEdit &&
    editIndex ===
      index
      ? "block"
      : "hidden"
  } font-medium text-blue-600 dark:text-blue-500 hover:underline`}
  onClick={() => {
    handleSave();
  }}
>
  Save
</button>

```

```

const handleSave = async () => {
  const quest = new Quest();
  if (tempData.length > 0) {
    tempData.map(async (item) => {
      await quest.add(item);
    });

    window.location.reload();
  } else {
    await quest.update(data[editIndex].id, data[editIndex].data);
  }
};

```

Function `handleSave` digunakan untuk memanggil function API yang sebelumnya sudah dijabarkan pada 3.6.2.1 yang terconnect di FE button Quesst “Save”. disitu dilakukan Penyimpanan data Quest.

### 3.6.3 Source Code Back End

#### 3.6.3.1 Route Add Quest

```

questController.post("/add", async (req, res) => {
  const result = await m$quest.addQuest(req.body);

  return response.sendResponse(res, result);
});

```

Route Add yang menggunakan Method Add yang berguna untuk melakukan penghapusan data dengan routing “/add”. nantinya akan mengeksekusi response yang diolah di module `m$quest.addQuest`.

### 3.6.3.2 Module Add Quest

```
addQuest = async (body) => {
  try {
    const add = await db
      .collection(questCollection)
      .add(body)
      .then((docRef) => {
        console.log("Document added with ID:", docRef.id);
      })
      .catch((error) => {
        console.error("Error adding document:", error);
      });

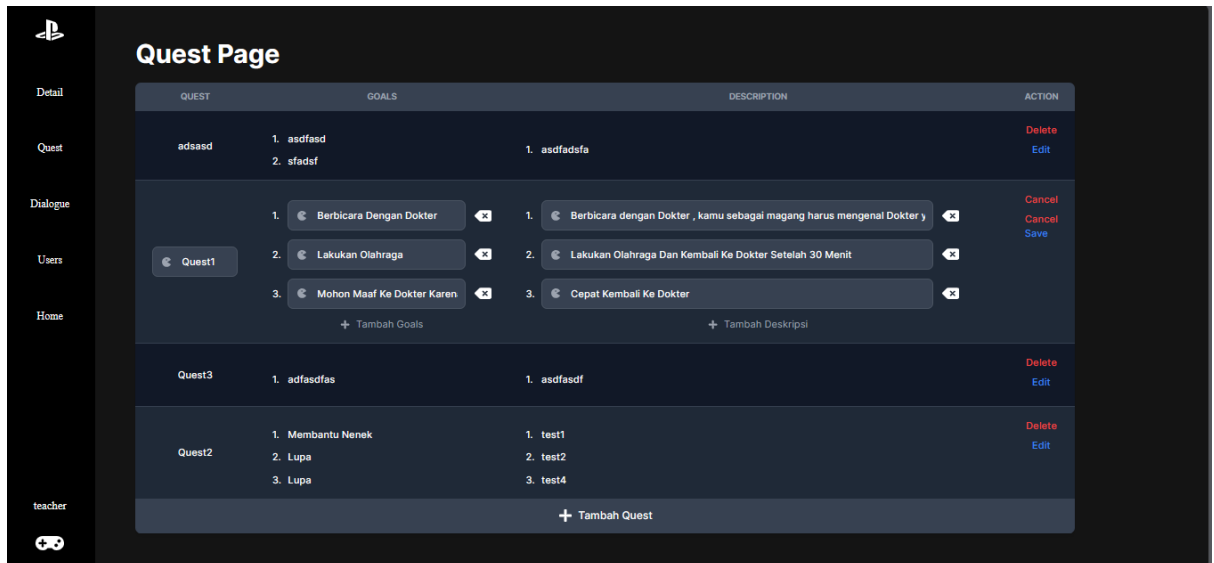
    return {
      status: true,
      code: 200,
      message: "data added successfully",
    };
  } catch (error) {
    console.error("request addData module Error: ", error);
    return {
      status: false,
      message:
        "Error, check the console log of the backend for what happened",
    };
  }
};
```

Fungsi addQuest merupakan operasi asinkron untuk menambahkan data ke sebuah koleksi dalam database, mungkin menggunakan Firebase. Fungsi ini menerima parameter body yang berisi data yang akan ditambahkan. Setelah mencoba menambahkan data, jika berhasil, mencetak ID dokumen baru ke konsol dan mengembalikan respons positif. Jika terjadi error, menangkap error, mencetak pesan error, dan mengembalikan respons negatif dengan pesan yang sesuai.

## 3.7 Quest (Edit)

### 3.7.1 Tata Cara Edit Quest

- 1) Lakukan login terlebih dahulu menggunakan akun admin
- 2) Click icons detail pada menu game yang diinginkan di home lihat Gambar 3.8
- 3) Pilih “Quest” pada sidebar kemudian tekan Edit seperti pada Gambar 3.11 kemudian akan melakukan direct ke form edit quest.
- 5) Isi Inputan berdasarkan urutan kolom jika sudah tekan save seperti pada gambar 3.12 yang nantinya berfungsi untuk menyimpan data Quest.



Gambar 3.13 Table Edit Quest

### 3.7.2 Source Code Front End

#### 3.7.2.1 API Handling Edit Quest

```

async update(id: string, quest: Quests) {
  const update = await fetch(
    `${this.getBaseUrl()}/api/quest/update/${id}`,
    {
      method: "PUT",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(quest),
    }
  );

  const response = await update.json();

  if (response.status === false) {
    throw new Error("Failed to update quest");
  } else {
    window.location.reload();
  }
}

```

Function update digunakan untuk melakukan pemanggilan API update Quest dan mengolah data Respon yang nantinya akan dilempar pada button Save di form edit. dengan Method PUT untuk melakukan pembaharuan data yang baru.



### 3.7.2.2 Button Save

```
<button
  className={` ${
    isEdit &&
    editIndex === index
      ? "hidden"
      : "block"
  } font-medium text-blue-600 dark:text-blue-500 hover:underline`}
  onClick={() =>
    handleEdit(
      index,
      true
    )
  }
>
  Edit
</button>
```

```
const handleEdit = async (index: number, isEdit: boolean) => {
  setEditIndex(index);
  setIsEdit(isEdit);
};
```

Function handle edit digunakan untuk memanggil function API yang sebelumnya sudah dijabarkan pada 3.7.2.1 yang terconnect di FE button Quest “edit”. disitu dilakukan akan mengubah tampilan seperti pada gambar 3.13 serta melakukan save dengan method yang sama dengan code sebelumnya di 3.6.

### 3.7.3 Source Code Back End

#### 3.7.3.1 Route Add Quest

```
questController.put("/update/:id", async (req, res) => {
  const result = await m$quest.updateQuest(req.params.id, req.body);

  return response.sendResponse(res, result);
});
```

Route Add yang menggunakan Method Add yang berguna untuk melakukan pembaharuan data dengan routing “/update/:id”. nantinya akan mengeksekusi response yang diolah di module m\$quest.updateQuest.

#### 3.7.3.2 Module Update Quest

```
updateQuest = async (id, body) => {
```

```

const updateData = await db.collection(questCollection).doc(id).get();

const questData = updateData.data();

await db.collection(questCollection).doc(id).update(body);

return {
  status: true,
  code: 200,
  message: "data updated successfully",
};
};

```

Fungsi updateQuest adalah operasi asinkron yang digunakan untuk mengupdate data dalam koleksi database (mungkin Firebase) berdasarkan ID yang diberikan. Fungsi ini mengambil data terkini berdasarkan ID, kemudian melakukan pembaruan menggunakan data yang diberikan pada parameter body. Setelah itu, fungsi mengembalikan respons positif yang menyatakan bahwa pembaruan data telah berhasil.

### 3.8 Quest (Delete)

#### 3.8.1 Tata Cara delete Quest

- 1) Lakukan login terlebih dahulu menggunakan akun admin
- 2) Click icons detail pada menu game yang diinginkan di home lihat Gambar 3.8
- 3) Pilih “Quest” pada sidebar kemudian tekan delete seperti pada Gambar 3.11 kemudian data akan langsung Quest akan langsung terhapus.

#### 3.8.2 Source Code Front End

##### 3.8.2.1 API Handling Delete Quest

```

async delete(id: string) {
  const del = await fetch(
    `${this.getBaseUrl()}/api/quest/destroy/${id}`,
    {
      method: "DELETE",
    }
  );

  const response = await del.json();

  if (response.status === false) {
    throw new Error("Failed to delete quest");
  } else {
    window.location.reload();
  }
}
}

```

Function Delete digunakan untuk melakukan pemanggilan API Delete Quest dengan tujuan menghapus data. Respon yang nantinya akan dilempar pada button delete di form gambar 3.11. dengan Method DELETE untuk melakukan penghapusan data.

### 3.8.2.2 Button Delete

```
<button
  className={` ${
    isEdit &&
    editIndex === index
      ? "hidden"
      : "block"
  } font-medium text-red-500 dark:text-red-500 hover:underline`}
  onClick={() =>
    handleDelete(
      item.id
    )
  }
  >
  Delete
</button>
```

```
const handleDelete = async (id: string) => {
  const quest = new Quest();
  await quest.delete(id);
};
```

Function handle Delete digunakan untuk memanggil function API yang sebelumnya sudah dijabarkan pada 3.8.2.1 yang terconnect di FE button Quest “Delete”. disitu dilakukan akan menghapus data Quest yang dipilih.

### 3.8.3 Source Code Back End

#### 3.8.3.1 Route Delete Quest

```
questController.delete("/destroy/:id", async (req, res) => {
  const result = await m$quest.destroyQuest(req.params.id);

  return response.sendResponse(res, result);
});
```

Route delete yang menggunakan Method DELETE yang berguna untuk melakukan penghapusan data dengan routing “destroy/:id”. nantinya akan mengeksekusi response yang diolah di module m\$quest.destroyQuest.

#### 3.8.3.2 Module Destroy Quest

```

destroyQuest = async (id) => {
  try {
    await db.collection(questCollection).doc(id).delete();

    return {
      status: true,
      code: 200,
      message: "data deleted successfully",
    };
  } catch (error) {
    console.error("request deleteData module Error: ", error);
    return {
      status: false,
      message:
        "Error, check the console log of the backend for what happened",
    };
  }
};

```

Fungsi destroyQuest adalah operasi asinkron yang digunakan untuk menghapus data dalam koleksi database (mungkin Firebase) berdasarkan ID yang diberikan. Fungsi ini mencoba menghapus data dengan ID tersebut, dan jika berhasil, mengembalikan respons positif. Jika terjadi error selama proses, menangkap error, mencetak pesan error ke konsol, dan mengembalikan respons negatif dengan pesan yang sesuai.

### 3.9 Dialogue (Create) With ChatGPT

#### 3.9.1 Tata Cara Create Dialog

- 1) Lakukan login terlebih dahulu menggunakan akun admin
- 2) Click icons detail pada menu game yang diinginkan di home lihat Gambar 3.8
- 3) Pilih “Dialogue” pada sidebar kemudian tekan Tambah Dialog seperti pada Gambar 3.14 kemudian akan melakukan direct ke form tambahDialog.

**Dialogue Page**

QUEST	NPC	MESSAGE	CONDITION	ACTION
Quest1	Dokter	1. Hallo Traveler, Selamat Datang Di RSUD Ponorogo. Sebelum Masuk Kerja Lakukan Olahraga Dluur Dan Kembali Dalam 30 Menit 2. Jika Kamu Sudah Melakukan SOP Olahraga Maka Cepat Kembali Ke Sini Untuk Pelajaran Selanjutnya 3. Jangan Lupa Pakai Baju Lab	0	Delete Edit
Quest1	Dokter	1. Kamu Ini Bagaimana Hari Pertama Sudah Ketiduran ? Bagaimana Hari Kerja Selanjutnya Ill 2. Hush, Pergi Dari Klinik Kami. Kami Tidak Butuh Pekerja Malas Seperti Mu	2	Delete Edit

+ Save

JUMLAH PARAGRAF	TOPIK ACUAN	SUB-TOPIK ACUAN (PER-PARAGRAF)	CONDITION
3	buka lowo	1. menyapa traveler 2. menguji traveler di SPBU 3. perintah pemain apa ujian yang akan dilaksanakan	

+ Save

**Gambar 3.14 Form Tambah Dialog (Generate Chat GPT)**

- 4) Isi Inputan yang ada dan tunggu Loading pemanggilan Chat GPT rekomendasi. jika sudah maka dialog message akan langsung terisi dan bisa anda ubah jika merasa kurang seperti pada Gambar 3.15

**Dialogue Page**

QUEST	NPC	MESSAGE	CONDITION	ACTION
Quest1	Dokter	1. Hallo Traveler, Selamat Datang Di RSUD Ponorogo. Sebelum Masuk Kerja Lakukan Olahraga Dluur Dan Kembali Dalam 30 Menit 2. Jika Kamu Sudah Melakukan SOP Olahraga Maka Cepat Kembali Ke Sini Untuk Pelajaran Selanjutnya 3. Jangan Lupa Pakai Baju Lab	0	Delete Edit
Quest1	Dokter	1. Kamu Ini Bagaimana Hari Pertama Sudah Ketiduran ? Bagaimana Hari Kerja Selanjutnya Ill 2. Hush, Pergi Dari Klinik Kami. Kami Tidak Butuh Pekerja Malas Seperti Mu	2	Delete Edit

+ Tambah Dialogue

JUMLAH PARAGRAF	TOPIK ACUAN	SUB-TOPIK ACUAN (PER-PARAGRAF)	CONDITION
3		1. "Selamat datang, traveler! Kami adalah perusahaan yang sedang membu 2. "Hai traveler! Apakah Anda pernah mengalami situasi darurat di perjalan 3. "Pemain, tugas Anda adalah untuk menyusun daftar prioritas dalam men	

+ Tambah Dialogue

**Gambar 3.15 Hasil ChatGPT**

## 3.9.2 Source Code Front End

### 3.9.2.1 API Create Dialogue & Chat GPT Handling

```

async add(dialog: gameData) {
  const add = await fetch(`${this.getBaseUrl()}/api/dialog/add`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(dialog),
  });

  const response = await add.json();

  if (response.status === false) {
    throw new Error("Failed to add quest");
  } else {
    window.location.reload();
  }
}

```

```

async generateOpenAi(prompt: Prompt) {
  const generate = await fetch(
    `${this.getBaseUrl()}/api/dialog/openai/generate`,
    {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(prompt),
    }
  );

  const response = await generate.json();

  if (response.status === false) {
    throw new Error("Failed to generate quest");
  } else {
    return response;
  }
}

```

Function ini digunakan untuk melakukan pemanggilan API create dialog dan generate dialog dengan ChatGPT. Respon yang nantinya akan dilempar pada button save. disini menggunakan Method Post karena akan melakukan POST data.

### 3.9.2.2 Button Create Game (Simpan)

```

<button
  className={` ${
    isEdit &&
    editIndex ===
      index &&
      item.id !== ""
      ? "block"
      : "hidden"
  } font-medium text-blue-600 dark:text-blue-500 hover:underline`}
  onClick={() => {
    handleSave();
  }}
>
  Save
</button>

```

```

<button
  className={` border-b bg-gray-700 dark:border-gray-700 text-center w-full`}
  onClick={() => handleSendPropmt()}
>
  <div className="flex items-center justify-center gap-2 px-6 py-3 font-medium"
    <FaPlus className="w-5 h-5" />
    Save
  </div>
</button>

```

Function `handleSave` digunakan untuk memanggil function API yang sebelumnya sudah dijabarkan pada 3.9.2.1 yang terconnect di FE button save. disitu dilakukan permintaan ChatGpt dengan memanggil `handleSendPropmt` untuk mendapatkan response dari inputan user dalam mengisi dialog.

### 3.9.3 Source Code Back End

#### 3.9.3.1 Route Create Dialog & Generate By ChatGPT

```

dialogController.post("/add", async (req, res) => {
  const result = await m$dialog.addDialog(req.body);

  return response.sendResponse(res, result);
});

dialogController.post("/openai/generate", async (req, res) => {
  const result = await m$dialog.openAiForDialogue(req.body);

  return response.sendResponse(res, result);
});

```

Route Dialog dan ChatGPT ini akan menggunakan Method POST yang berguna untuk melakukan POST data dengan routing “/add” untuk melakukan save data dan “/openai/generate” untuk mendapatkan response dialog yang diinginkan. nantinya akan mengeksekusi response yang diolah di module m\$dialog.addDialog dan m\$dialog.openAiForDialogue..

### 3.9.3.2 Module addDialog dan ChatGPT

```
addDialog = async (body) => {
  try {
    const add = await db
      .collection(dialogCollection)
      .add(body)
      .then((docRef) => {
        console.log("Document added with ID:", docRef.id);
      })
      .catch((error) => {
        console.error("Error adding document:", error);
      });

    return {
      status: true,
      code: 200,
      message: "data added successfully",
    };
  } catch (error) {
    console.error("request addData module Error: ", error);
    return {
      status: false,
      message:
        "Error, check the console log of the backend for what happened",
    };
  }
};

openAiForDialogue = async (body) => {
  const prompt = `
    Saya sedang membuat game dengan game berbasis cerita, berikan saya 1 pesan
    dengan maksimal 100 kata per-paragrafnya, dengan jumlah paragraf ${
      body.jumlahParagraf
    } dengan topik acuannya adalah ${body.topikAcuan}
    serta untuk sub-topik acuannya per paragraf yang memiliki kalimat aktif
    untuk digunakan sebagai dialog adalah ${body.subTopikAcuan.map(
      (element, index) => {
        return `${
          index + 1
        } dengan sub topik acuan ${element}`;
      }
    )}
  `;
}
```



```

    })
  };
  const openai = new OpenAI({
    apiKey: "sk-z3wa0opb7kXuKR1UQTyAT3B1bkFJnhLvr736CgYsKgP4n10d",
  });
  const completion = await openai.chat.completions.create({
    messages: [
      {
        role: "system",
        content: prompt,
      },
    ],
    model: "gpt-3.5-turbo",
  });
  return {
    status: true,
    code: 200,
    message: completion.choices[0].message.content,
  };
};

```

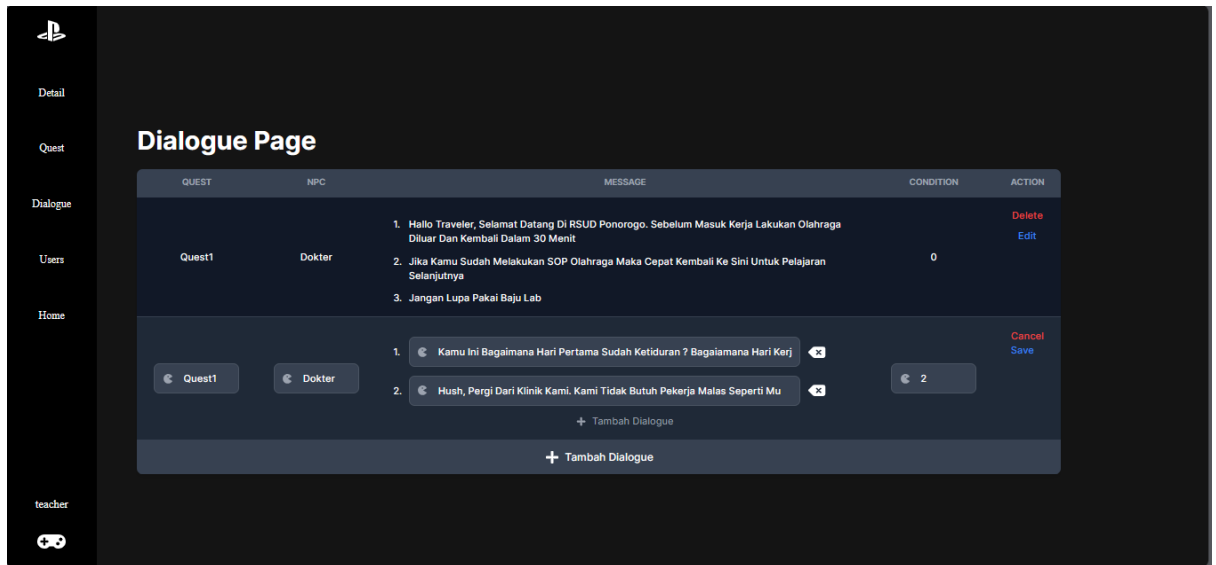
Fungsi `addDialog` adalah operasi asinkron yang digunakan untuk menambahkan data dialog ke koleksi database (mungkin Firebase). Fungsi ini mengirim data dialog ke koleksi, mencetak ID dokumen baru jika berhasil, dan mengembalikan respons positif. Jika terjadi error selama proses, menangkap error, mencetak pesan error ke konsol, dan mengembalikan respons negatif dengan pesan yang sesuai.

Fungsi `openAiForDialogue` mengirim permintaan ke OpenAI menggunakan model GPT-3.5-turbo untuk menghasilkan dialog berdasarkan prompt yang dibuat dari parameter body. Prompt ini dirancang untuk meminta dialog dengan jumlah paragraf, topik acuan, dan sub-topik acuan tertentu. Hasilnya diperoleh melalui OpenAI, dan respons positif berisi hasil dialog disertakan dalam respons tersebut. Jika terjadi error selama proses, mengembalikan respons negatif dengan pesan error.

### 3.10 Dialogue (Edit)

#### 3.10.1 Tata Cara Edit Dialogue

- 1) Lakukan login terlebih dahulu menggunakan akun admin
- 2) Click icons detail pada menu game yang diinginkan di home lihat Gambar 3.8
- 3) Pilih “Dialogue” pada sidebar kemudian tekan Edit seperti pada Gambar 3.16 kemudian akan melakukan direct ke form edit quest lalu isi berdasarkan kolom yang ada.



Gambar 3.16 Table Edit Dialogue

### 3.10.2 Source Code Front End

#### 3.10.2.1 API Handling Edit Dialogue

```

async update(id: string, dialog: gameData) {
  const update = await fetch(
    `${this.getBaseUrl()}/api/dialog/update/${id}`,
    {
      method: "PUT",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(dialog),
    }
  );

  const response = await update.json();

  if (response.status === false) {
    throw new Error("Failed to update quest");
  } else {
    window.location.reload();
  }
}

```

Function update digunakan untuk melakukan pemanggilan API update Dialogue dan mengolah data Respon yang nantinya akan dilempar pada button Save di form edit. dengan Method PUT untuk melakukan pembaharuan data yang baru.

### 3.10.2.2 Button Save pada edit

```
<button
  className={` ${
    isEdit &&
    editIndex ===
      index
      ? "block"
      : "hidden"
  } font-medium text-red-600 dark:text-red-500 hover:underline`}
  onClick={() =>
    handleEdit(
      index,
      false
    )
  }
>
  Cancel
</button>
```

```
const handleEdit = async (index: number, isEdit: boolean) => {
  setEditIndex(index);
  setIsEdit(isEdit);
};
```

Function handle edit digunakan untuk memanggil function API yang sebelumnya sudah dijabarkan pada 3.10.2.1 yang terconnect di FE button Quest “edit”. disitu dilakukan akan mengubah tampilan serta melakukan save dengan method yang sama dengan code sebelumnya.

### 3.10.3 Source Code Back End

#### 3.10.3.1 Route Add Quest

```
dialogController.put("/update/:id", async (req, res) => {
  const result = await m$dialog.updateDialog(req.params.id, req.body);

  return response.sendResponse(res, result);
});
```

Route Update yang menggunakan Method PUT yang berguna untuk melakukan pembaharuan data dengan routing “/update/:id”. nantinya akan mengeksekusi response yang diolah di module m\$dialog.updateDialog..

#### 3.7.3.2 Module Update Dialog

```
updateDialog = async (id, body) => {
```

```

const updateData = await db.collection(dialogCollection).doc(id).get();

const questData = updateData.data();

await db.collection(dialogCollection).doc(id).update(body);

return {
  status: true,
  code: 200,
  message: "data updated successfully",
};
};

```

Fungsi updateDialog adalah operasi asinkron yang digunakan untuk mengupdate data dialog dalam koleksi database (mungkin Firebase) berdasarkan ID yang diberikan. Fungsi ini mengambil data terkini berdasarkan ID, kemudian melakukan pembaruan menggunakan data yang diberikan pada parameter body. Setelah itu, fungsi mengembalikan respons positif yang menyatakan bahwa pembaruan data telah berhasil.

### 3.11 Dialogue (Delete)

#### 3.11.1 Tata Cara delete Dialogue

- 1) Lakukan login terlebih dahulu menggunakan akun admin
- 2) Click icons detail pada menu game yang diinginkan di home lihat Gambar 3.8
- 3) Pilih “Dialogue” pada sidebar kemudian tekan delete seperti pada Gambar 3.11 kemudian data dialogue akan langsung terhapus.

#### 3.11.2 Source Code Front End

##### 3.11.2.1 API Handling Delete Dialogue

```

async delete(id: string) {
  const del = await fetch(
    `${this.getBaseUrl()}/api/dialog/delete/${id}`,
    {
      method: "DELETE",
    }
  );

  const response = await del.json();

  if (response.status === false) {
    throw new Error("Failed to delete quest");
  } else {
    window.location.reload();
  }
}

```

Function Delete digunakan untuk melakukan pemanggilan API Delete dialogue dengan tujuan menghapus data. Respon yang nantinya akan dilempar pada button delete di form gambar 3.11. dengan Method DELETE untuk melakukan penghapusan data.

### 3.11.2.2 Button Delete

```
<button
  className={` ${
    isEdit &&
    editIndex === index
      ? "hidden"
      : "block"
  } font-medium text-red-500 dark:text-red-500 hover:underline`}
  onClick={() =>
    handleDelete(
      item.id
    )
  }
  >
  Delete
</button>
```

```
const handleDelete = async (id: string) => {
  const dialog = new Dialog();
  await dialog.delete(id);
};
```

Function handle Delete digunakan untuk memanggil function API yang sebelumnya sudah dijabarkan pada 3.11.2.1 yang terconnect di FE button Dialog “Delete”. disitu dilakukan akan menghapus data Dialog yang dipilih.

### 3.11.3 Source Code Back End

#### 3.11.3.1 Route Delete Dialog

```
dialogController.delete("/delete/:id", async (req, res) => {
  const result = await m$dialog.deleteDialogue(req.params.id);

  return response.sendResponse(res, result);
});
```

Route delete yang menggunakan Method DELETE yang berguna untuk melakukan penghapusan data dengan routing “delete/:id”. nantinya akan mengeksekusi response yang diolah di module m\$dialog.deleteDialogue.

#### 3.11.3.2 Module Delete Dialogue

```

deleteDialogue = async (id) => {
  try {
    await db.collection(dialogCollection).doc(id).delete();

    return {
      status: true,
      code: 200,
      message: "data deleted successfully",
    };
  } catch (error) {
    console.error("request deleteData module Error: ", error);
    return {
      status: false,
      message:
        "Error, check the console log of the backend for what happened",
    };
  }
};


```

Fungsi deleteDialogue adalah operasi asinkron yang digunakan untuk menghapus data dialog dalam koleksi database (mungkin Firebase) berdasarkan ID yang diberikan. Fungsi ini mencoba menghapus data dengan ID tersebut, dan jika berhasil, mengembalikan respons positif. Jika terjadi error selama proses, menangkap error, mencetak pesan error ke konsol, dan mengembalikan respons negatif dengan pesan yang sesuai.

### 3.12 Melihat Data Log Pemain Dalam Gim

#### 3.12.1 Tata Cara Melihat Log Pemain

- 1) Lakukan login terlebih dahulu menggunakan akun admin
- 2) Click icons detail pada menu game yang diinginkan di home lihat Gambar 3.8
- 3) Pilih “Users” pada sidebar pada Gambar 3.17 maka akan terlihat data userslog semua pemain.



USERNAME	QUEST 1	QUEST 2	QUEST 3	QUEST 4	QUEST 5
test12	0	-1	-1	-1	-1
agasuck	0	-1	-1	-1	-1
yayeyo	0	-1	-1	-1	-1

Gambar 3.17 Users Page

### 3.12.2 Source Code Front End

#### 3.12.2.1 API Handling Get UsersLog

```

    async getByUsername(username: string): Promise<DataUsers[]> {
        const quest = await fetch(
            `${this.getBaseUrl()}/api/users/log/${username}`
        );

        const response: ReturnUsers = await quest.json();

        if (response.status === false) {
            throw new Error("Failed to get quest data");
        } else {
            return response.data;
        }
    }

```

Function `getByUsername` digunakan untuk melakukan pemanggilan API `get Userslog` dan mengolah data Respon setelah user mengakses page `Users` di page `Teacher`. dengan Method `GET` untuk mendapatkan data..

### 3.12.3 Source Code Back End

#### 3.12.3.1 Route Add Get UsersLog

```
userController.get("/usersLog/:game", async (req, res) => {  
  ⚡ const result = await m$user.getDataBasedGame(req.params.game);  
  
  return response.sendResponse(res, result);  
});
```

Route Get UsersLog yang menggunakan Method GET yang berguna untuk melakukan Pengambilan data dengan routing “/userslog/:game”. nantinya akan mengeksekusi response yang diolah di module m\$user.getDataBasedGame..

#### 3.7.3.2 Module Update Dialog

```
getDataBasedGame = async (game) => {  
  try {  
    const getData = await db  
      .collection(usersLogCollection)  
      .where("game", "=", game)  
      .get();  
  
    const data = [];  
  
    getData.forEach((doc) => {  
      const dataUser = doc.data();  
      data.push({ id: doc.id, dataUser });  
    });  
  
    return {  
      status: true,  
      code: 200,  
      data,  
    };  
  } catch (error) {  
    console.error("request deleteData module Error: ", error);  
    return {  
      status: false,  
      message:  
        "Error, check the console log of the backend for what happened",  
    };  
  }  
};
```



Fungsi `getDataBasedGame` adalah operasi asinkron yang digunakan untuk mengambil data dari koleksi database (mungkin Firebase) berdasarkan game yang diidentifikasi. Fungsi ini menggunakan metode `where` untuk memfilter data berdasarkan nilai properti "game" yang sesuai. Data yang ditemukan kemudian diambil, diproses, dan disusun dalam bentuk array yang berisi objek dengan ID dokumen dan data pengguna terkait. Jika operasi berhasil, fungsi mengembalikan respons positif dengan data yang diambil. Jika terjadi error selama proses, fungsi menangkap error, mencetak pesan error ke konsol, dan mengembalikan respons negatif dengan pesan yang sesuai.

## **BAB IV**

### **DOKUMENTASI/FILE SOURCE CODE**

#### **4.1 Source Code Front End**

Source Code FrontEnd menggunakan Typescript dengan framework Next.Js 13.

Link : [IndyJones123/website-game-upn \(github.com\)](https://github.com/IndyJones123/website-game-upn). Tata cara penggunaan telah dilampirkan pada readme.md

#### **4.2 Source Code Back End**

Source Code BackEnd menggunakan Javascript dengan framework React.

Link : [IndyJones123/api-web-game-upn \(github.com\)](https://github.com/IndyJones123/api-web-game-upn). Tata cara penggunaan telah dilampirkan pada readme.md

#### **4.3 Dokumentasi API**

Dokumentasi API ini berisi record testing API yang telah dibuat pada source code Backend

Link : [AliceWeb \(getpostman.com\)](https://getpostman.com/AliceWeb)

#### **4.4 Source Code Game Kerja**

Source Code game kerja menggunakan C# dengan tools Unity

Link : [IndyJones123/AliceGameKerja \(github.com\)](https://github.com/IndyJones123/AliceGameKerja)

#### **4.5 Website Sinarpadu**

Website live yang telah di deploy dengan Link sebagai berikut :

Link : [Testing \(sinarpadu.com\)](https://sinarpadu.com/Testing)

#### **4.6 Link Download Game Prototype “Kerja”**

Game telah dibuild dan dideploy di website itch.io

Link : <https://agasuck.itch.io/why-is-it-so-hard-to-apply-a-job>