

# Game Review

---

**KELOMPOK 5**

SBD-02

# MEET THE TEAM

---



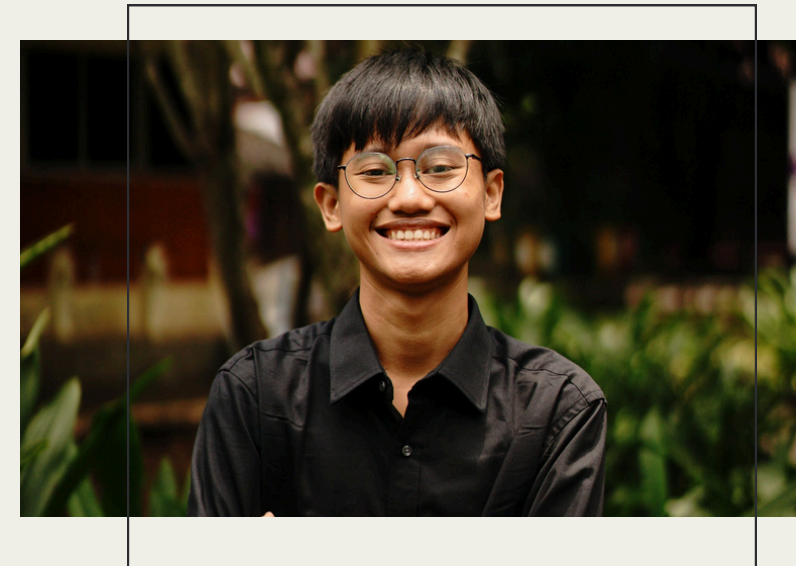
**Ivan Yuantama**

Pembuat backend & dockerize



**Ryan Safa Tjendana**

Pembuat backend & dockerize



**Surya Dharmasaputra S.**

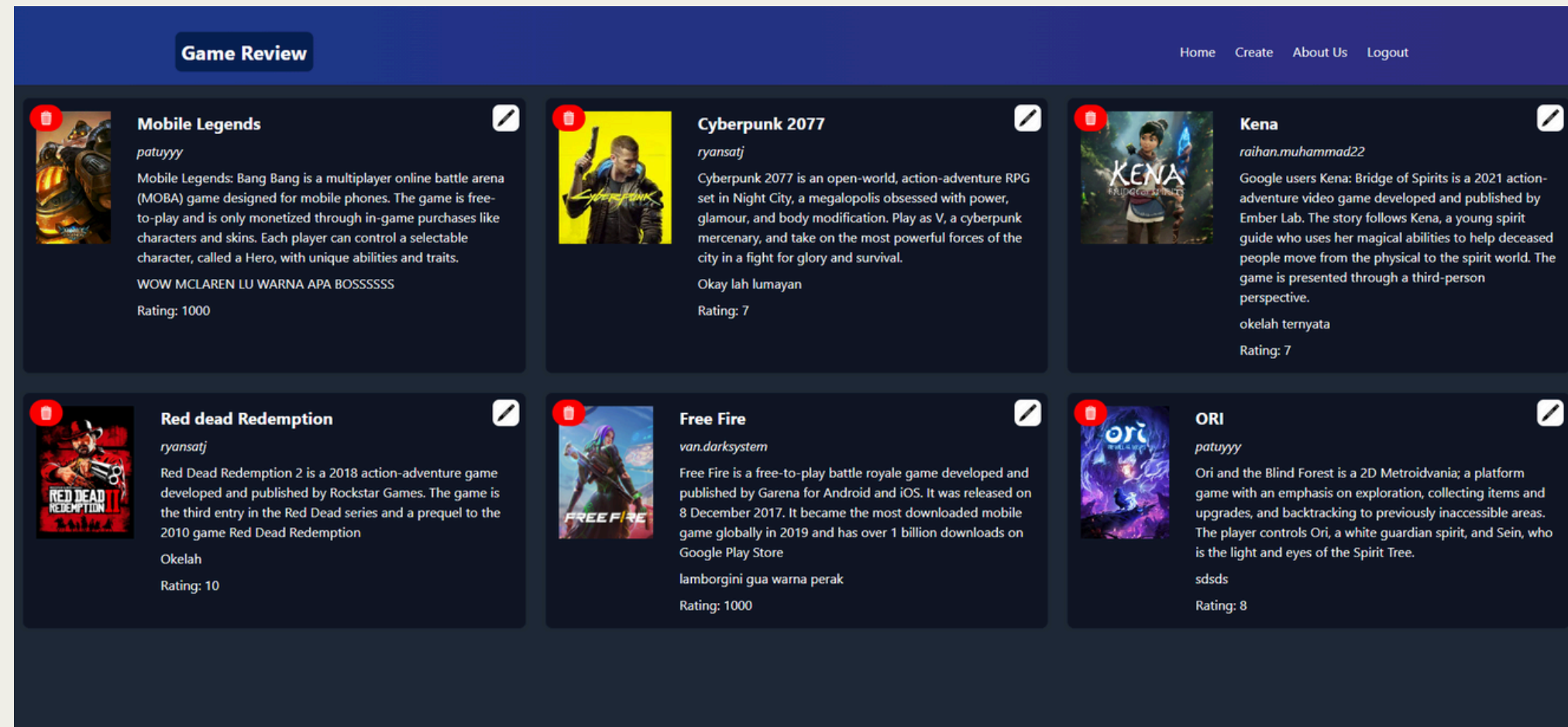
Pembuat PPT & Frontend



**M. Daffa Rizkyandri**

Pembuat PPT & Frontend

# PENJELASAN APLIKASI



Game Review merupakan sebuah aplikasi berbasis web yang digunakan oleh user untuk menambahkan ulasan mengenai permainan yang dimainkan olehnya. Dengan aplikasi ini, user dapat menambahkan ulasan, melihat ulasan yang telah dibuatnya dan dibuat oleh orang lain, dan menghapus ulasan yang pernah dibuatnya.

Aplikasi ini dibuat dengan menggunakan MongoDB sebagai database, Node.js sebagai back end, dan React.js untuk front end. Program juga dikontainerisasi dengan menggunakan aplikasi Docker dan disimulasikan menggunakan Vite.

# PENGIMPLEMENTASIAN CRUD

---

## Create

User dapat membuat sebuah akun dan juga ulasan game yang nantinya akan disimpan dalam database

## Update

User dapat memperbarui ulasan game yang telah dibuatnya

## Read

User dapat membaca ulasan game dari user lainnya, entah itu secara singkat maupun secara detail.

## Delete

User dapat menghapus ulasan yang telah dibuatnya.

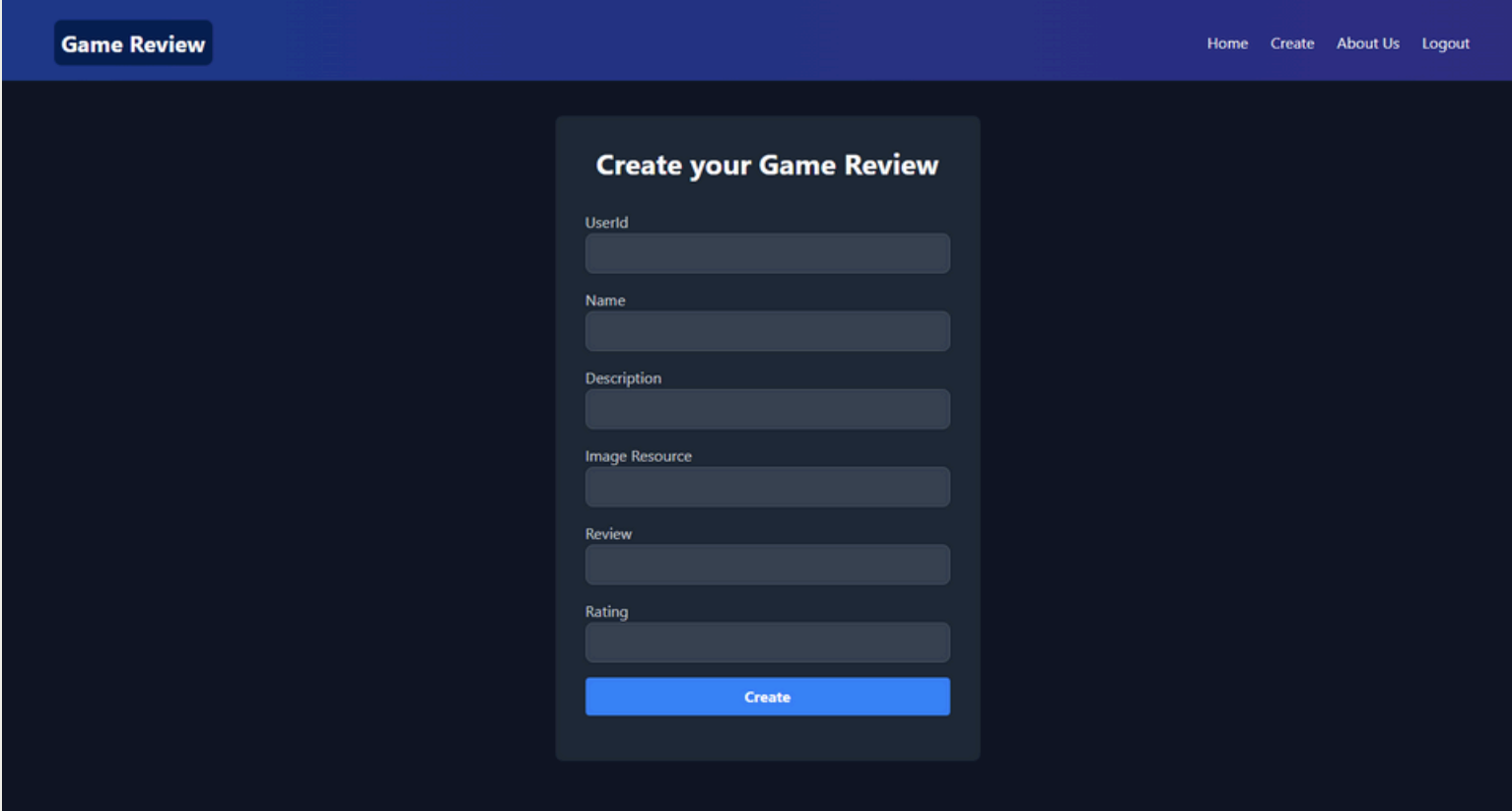
# CREATE SECARA DETAIL

---

- Pada gameController.js

```
makeReview = async(req, res) =>{
  try{
    const{userid, name, description, resources, review, rating} = req.body;
    const game = new Game ({
      userid,
      name,
      description,
      resources,
      review,
      rating,
    })
    await game.save();
    res.status(201).json(game);
  } catch(err){
    console.error(err);
    res.status(500).send(err);
  }
}
```

Kode disamping merupakan kode implementasi create pada gameController.js. Kode tersebut digunakan untuk membuat sebuah ulasan baru. Kode ini akan menyimpan param berupa userid yang terbaca otomatis, name, description, resources, review, dan juga rating yang dimasukkan oleh user.



The screenshot displays a web application interface for creating a game review. At the top, there is a dark blue navigation bar with the text 'Game Review' on the left and links for 'Home', 'Create', 'About Us', and 'Logout' on the right. The main content area is dark blue and features a central white card titled 'Create your Game Review'. This card contains several input fields: 'UserId', 'Name', 'Description', 'Image Resource', 'Review', and 'Rating'. Each field is represented by a light blue rectangular box. Below these fields is a prominent blue button labeled 'Create'.



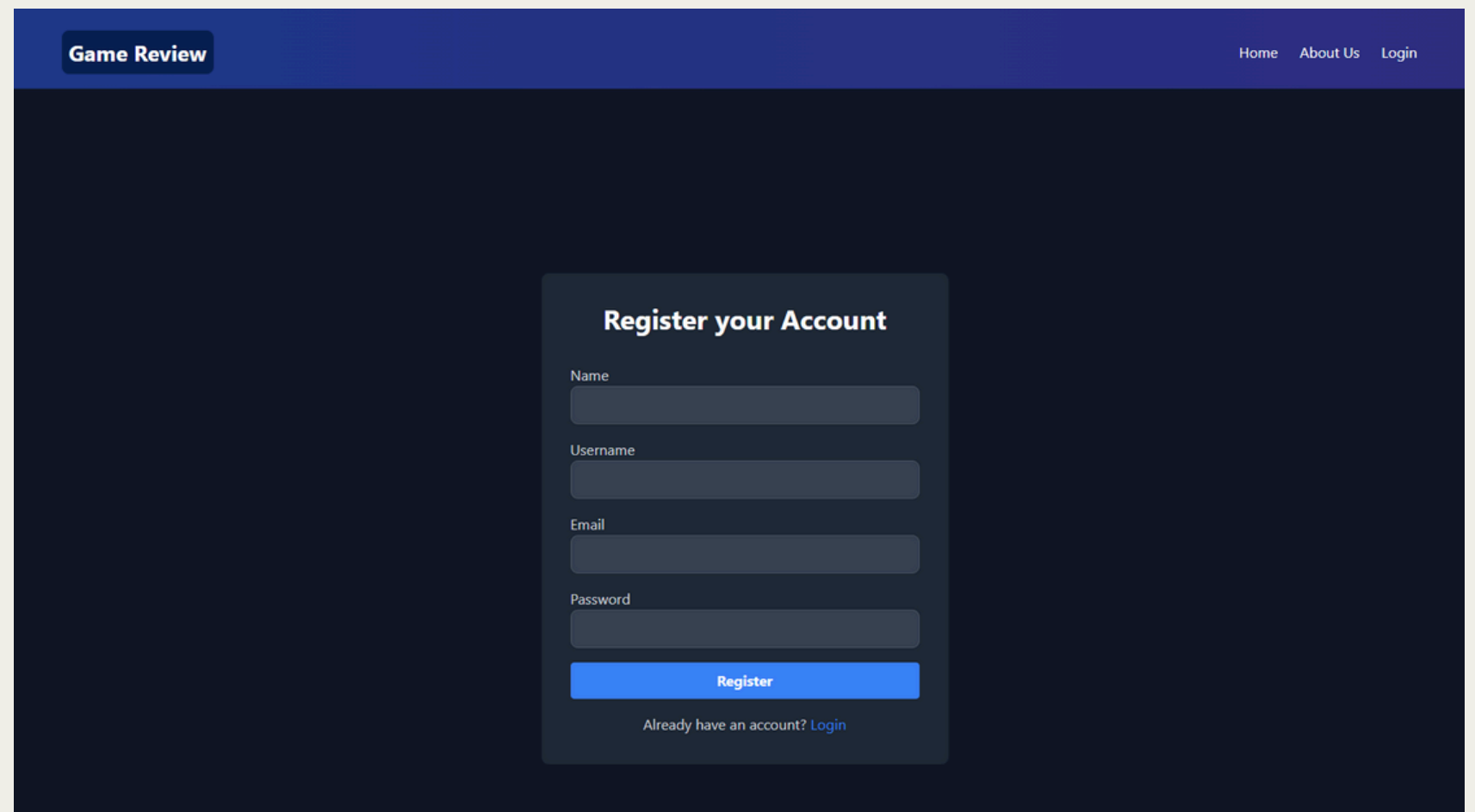
# CREATE SECARA DETAIL

---

- Pada userController.js

```
signUp = async(req, res) =>{
  const{name, email, password, username} = req.body;
  const user = new User({
    username, name, email, password
  })
  try{
    await user.save();
    res.status(201).json(user);
  } catch(err){
    console.error(err);
    res.status(500).send(err);
  }
}
```

Kode disamping merupakan kode implementasi create pada userController.js dimana program akan membuat sebuah akun yang terdapat name, email, password, dan juga username yang diinput oleh user pada program dan akan disimpan pada database.



The screenshot shows a web application interface with a dark blue header. On the left, there is a 'Game Review' button. On the right, there are links for 'Home', 'About Us', and 'Login'. The main content area is dark blue and features a 'Register your Account' form. The form has four input fields: 'Name', 'Username', 'Email', and 'Password'. Below these fields is a blue 'Register' button. At the bottom of the form, there is a link that says 'Already have an account? Login'.

# READ SECARA DETAIL

- Pada userController.js

```
login = async(req, res) => {  
  const{username, password} = req.body;  
  try{  
    const user = await User.findOne({  
      username, password  
    });  
    res.status(201).json(user);  
  } catch(err){  
    console.error(err);  
    res.status(500).send(err);  
  }  
}
```

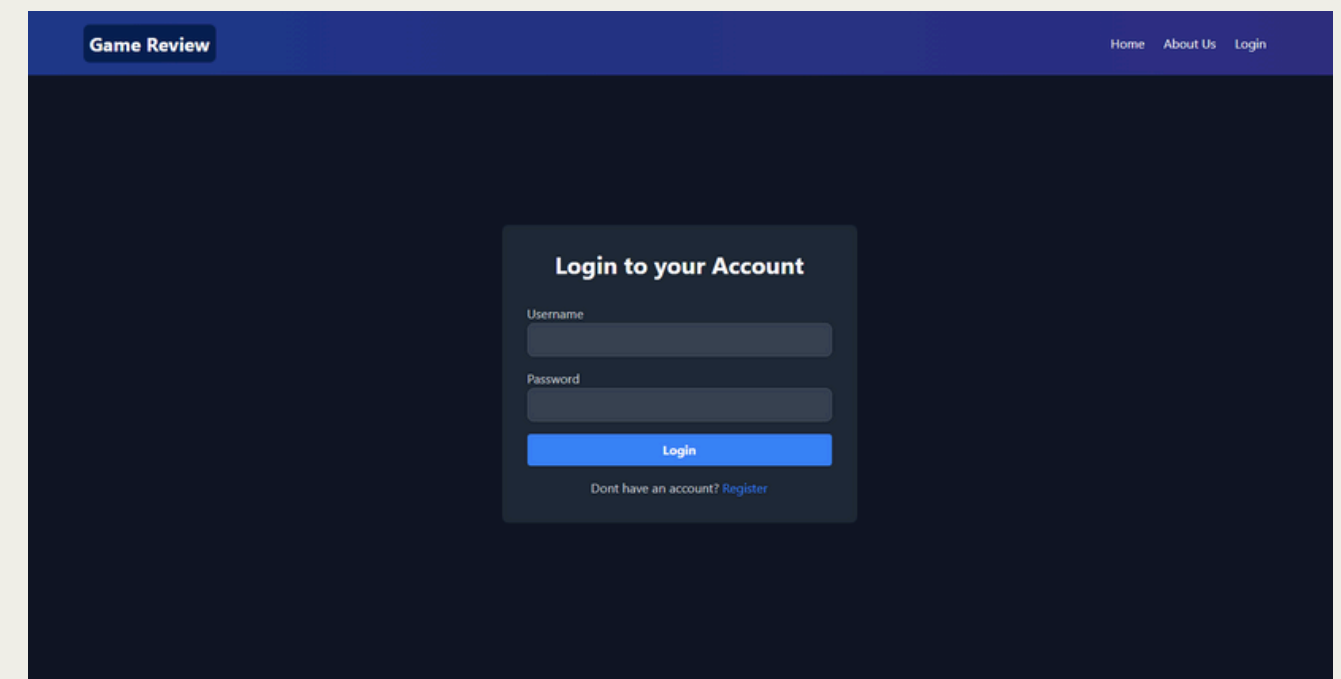
```
getUserbyId = async(req, res) => {  
  const{id} = req.body;  
  try{  
    const user = await User.findById(id);  
    res.status(200).json(user);  
  } catch(err){  
    console.error(err);  
    res.status(500).send(err);  
  }  
}
```

```
getAllUser = async(req, res) => {  
  try{  
    const user = await User.find();  
    res.status(200).json(user);  
  }  
  catch(err){  
    res.status(500).send(err);  
  }  
}
```

Tiga buah kode disamping adalah kode yang terdapat pada userController.js. Fungsi login merupakan fungsi yang berguna untuk membaca inputan user pada page login. Apabila username dan password yang dimasukkan terdapat pada database maka user akan dapat masuk ke website utama.

Fungsi getUserbyId merupakan sebuah fungsi dimana program akan membaca user id dari user yang melakukan login pada website.

Fungsi getAllUser akan mendapatkan seluruh user yang terdapat pada database



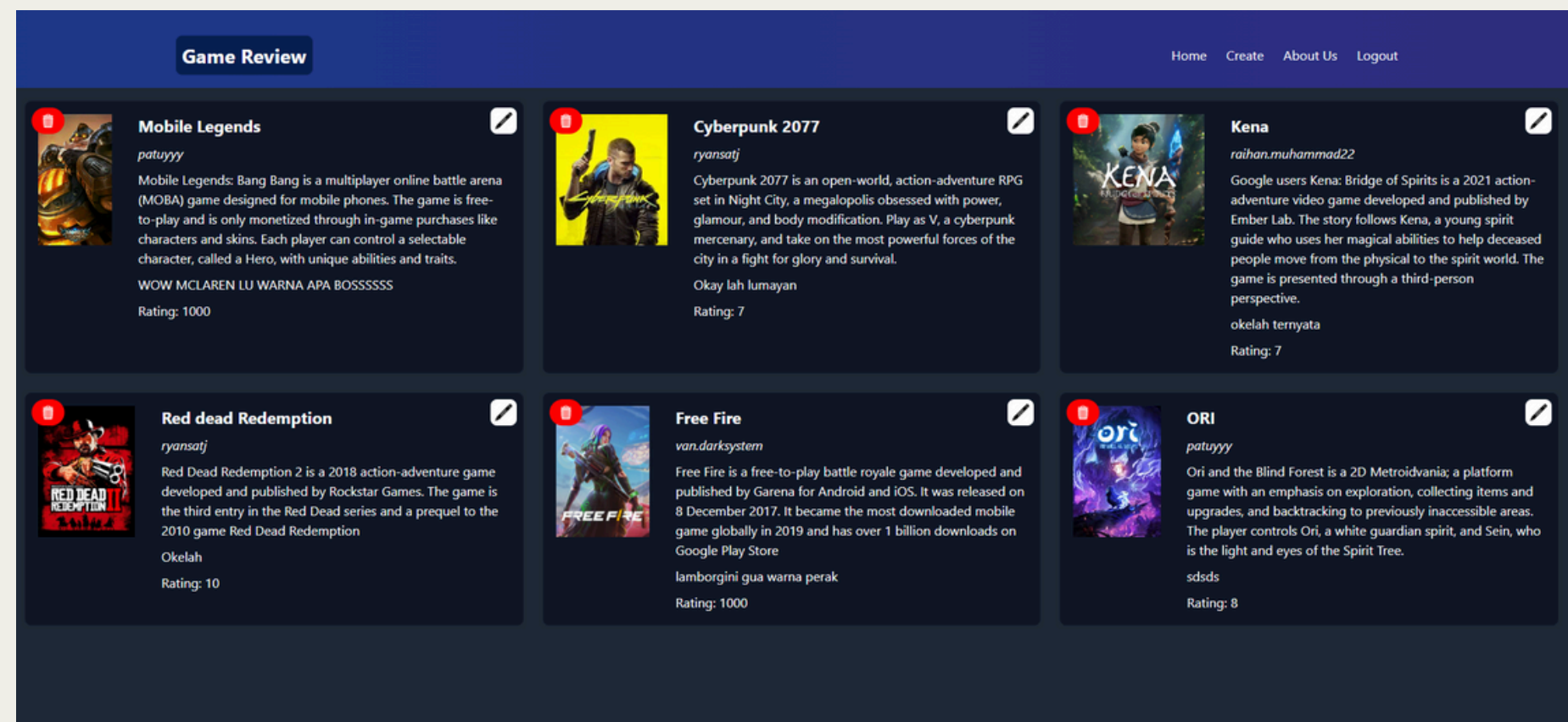
# READ SECARA DETAIL

- Pada gameController.js

```
const getAllReview = async (req, res) => {
  try {
    const result = await Game.aggregate([
      {
        $lookup: {
          from: 'users',
          localField: 'userid',
          foreignField: '_id',
          as: 'userDetails'
        }
      },
      {
        $unwind: '$userDetails'
      },
    ],
  );
}
```

```
{
  $project: {
    name: 1,
    description: 1,
    resources: 1,
    review: 1,
    rating: 1,
    username: '$userDetails.username'
  }
};
```

Kode disamping merupakan kode implementasi read pada gameController.js. Kode tersebut digunakan untuk menampilkan keseluruhan review yang telah dibuat oleh seluruh user. Data yang ditampilkan adalah name, description, resources, review, rating, dan username.





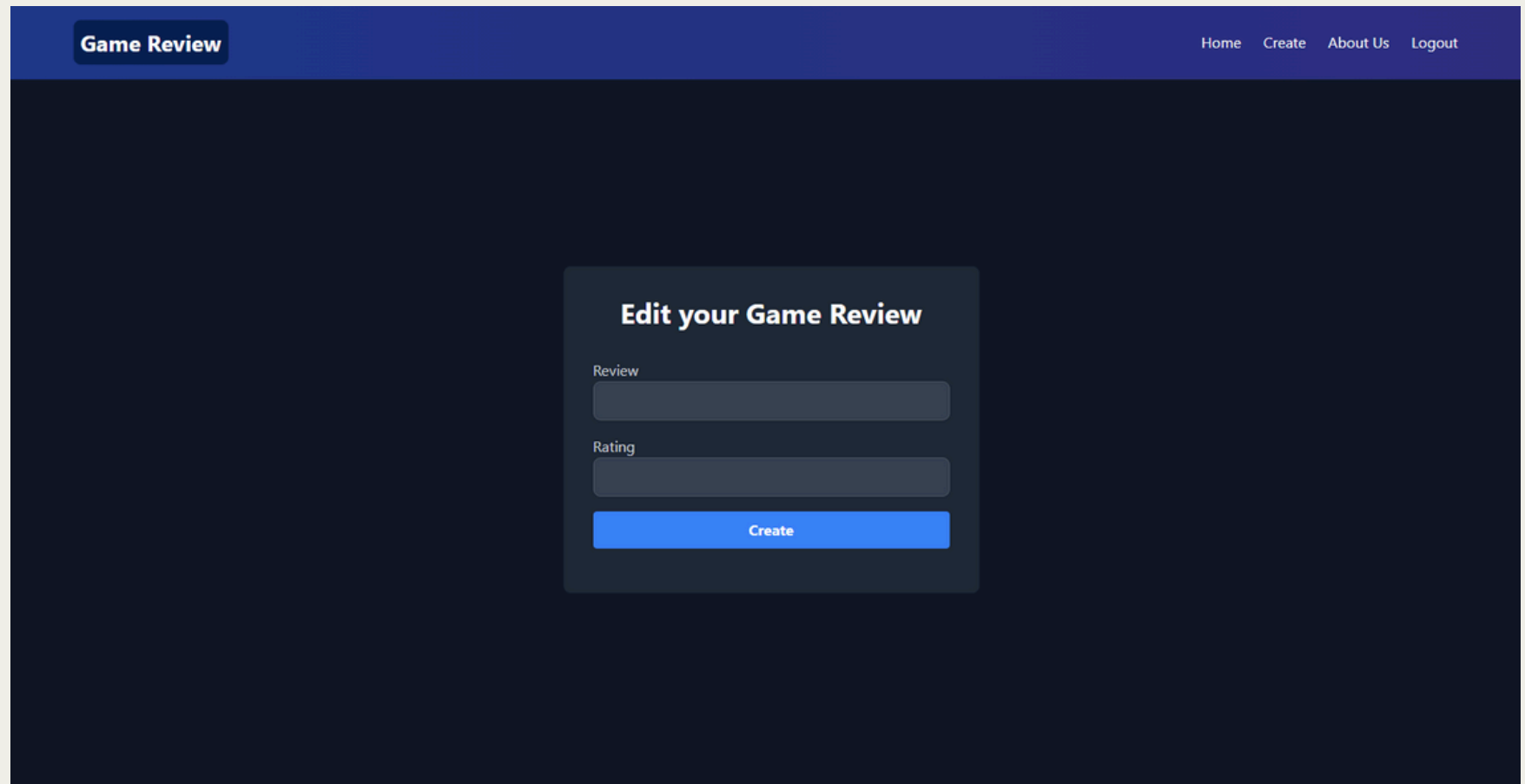
# UPDATE SECARA DETAIL

---

- Update pada gameController.js

```
const updateReview = async (req, res) => {
  try {
    const { id } = req.params;
    const { review, rating } = req.body;
    const result = await Game.findByIdAndUpdate(
      id,
      { review, rating },
      { new: true }
    );
    res.status(200).json(result);
  } catch (err) {
    console.error(err);
    res.status(500).send(err);
  }
};
```

Pengimplementasian update terdapat pada gameController.js. Fungsi updateReview akan mengupdate ulasan yang telah ada ketika user yang berkaitan ingin mengedit salah satu field pada review yang telah dibuatnya, entah itu deskripsinya maupun ratingnya.



The screenshot displays a web application interface for editing a game review. At the top, there is a dark blue header with the text 'Game Review' on the left and navigation links 'Home', 'Create', 'About Us', and 'Logout' on the right. The main content area is dark blue and features a central white box titled 'Edit your Game Review'. Inside this box, there are two input fields: 'Review' and 'Rating'. Below these fields is a blue button labeled 'Create'.

# DELETE SECARA DETAIL

- Delete pada gameController.js

```
const deleteReview = async (req, res) => {  
  try {  
    const { id } = req.params;  
    const result = await Game.findByIdAndDelete(id);  
    res.status(200).json(result);  
  } catch (err) {  
    console.error(err);  
    res.status(500).send(err);  
  }  
};
```

Pengimplementasian delete terdapat pada gameController.js juga. Fungsi deleteReview merupakan fungsi yang akan menghapus ulasan yang telah dibuat oleh user dengan mencari review yang berkaitan pada database berdasarkan id yang ada.



# IMPLEMENTASI DOCKER CONTAINERIZATION

---

## MongoDB Docker Container

MongoDB dijalankan dalam container Docker yang terisolasi. Ini memastikan bahwa database terpisah dari aplikasi lainnya dan mudah untuk dikelola serta dipindahkan. Data MongoDB disimpan di volume yang dipetakan ke direktori ./mongo\_db pada host. Ini memastikan data tetap persisten meskipun container MongoDB dihentikan atau dihapus.

## API Interaction Docker Container

API backend yang dibangun menggunakan Node.js berkomunikasi dengan MongoDB untuk melakukan operasi CRUD pada data review game. Dengan Docker Compose, API backend dapat diatur untuk bergantung pada container MongoDB, memastikan bahwa database berjalan sebelum backend mencoba mengaksesnya.

## FrontEnd Docker Container

Container frontend digunakan untuk menyediakan antarmuka pengguna dari aplikasi review game. Frontend ini berinteraksi dengan API untuk mengambil dan menampilkan data review game. Frontend berjalan dalam container terpisah, memastikan development environment frontend terisolasi dari komponen lain.

# IMPLEMENTASI DOCKER CONTAINERIZATION

---

## MongoDB Docker Container

MongoDB dijalankan dalam sebuah container yang didefinisikan dalam docker-compose.yml. Container ini menggunakan image mongo:latest dari Docker Hub dan diberi nama dbcontainer. Port 2717 pada host diarahkan ke port 2717 pada container. Untuk memastikan data tetap persisten, direktori ./mongo\_db pada host di-mapping ke /data/db di dalam container. Restart diatur ke always, yang berarti container akan secara otomatis restart jika terjadi kegagalan. Dengan setup ini, database MongoDB terisolasi dan dapat dijalankan secara konsisten di berbagai environment.

```
mongo_db:
  container_name: dbcontainer
  ports:
    - 2717:2717
  image: mongo:latest
  restart: always
  volumes:
    - ./mongo_db:/data/db
```

docker-compose.yml

# IMPLEMENTASI DOCKER CONTAINERIZATION

---

## API Interaction Docker Container

API backend dijalankan dalam container Docker. Konteks build-nya diatur ke direktori ./Backend\_MONGO, dan image dibangun menggunakan Dockerfile yang ada di direktori tersebut. Container ini diberi nama myrestapi dan port 4001 pada host diarahkan ke port 4001 pada container. Dockerfile untuk API backend menggunakan image dasar node:alpine, sebuah image yang ringan berbasis Alpine Linux. Setelah menyalin file package.json dan package-lock.json ke dalam container, perintah npm install dijalankan untuk menginstal dependensi. Kemudian, semua file proyek disalin ke dalam container, dan aplikasi dijalankan dengan perintah npm start. Seperti MongoDB, kebijakan restart diatur ke always.

```
# Node API services
api:
  build: ./Backend_MONGO
  container_name: myrestapi
  ports:
    - 4001:4001
  restart: always
```

docker-compose.yml

```
FROM node:alpine
WORKDIR /usr/src/app
COPY ./package.json ./
COPY ./package-lock.json ./
RUN npm install
COPY . .
EXPOSE 4001
CMD ["npm", "start"]
```

Dockerfile



# IMPLEMENTASI DOCKER CONTAINERIZATION

---

## FrontEnd Docker Container

Frontend aplikasi juga dijalankan dalam container terpisah yang didefinisikan dalam docker-compose.yml. Konteks build untuk frontend diatur ke direktori ./Frontend, dan image dibangun menggunakan Dockerfile yang ada di direktori tersebut. Container ini diberi nama myfrontend dan port 5173 pada host diarahkan ke port 5173 pada container. Dockerfile frontend juga menggunakan image dasar node:alpine. File package.json dan package-lock.json disalin ke dalam container dan perintah npm install dijalankan untuk menginstal dependensi. Setelah itu, semua file proyek disalin ke dalam container dan aplikasi dijalankan dengan perintah npm run dev --host. Kebijakan restart juga diatur ke always untuk memastikan container restart secara otomatis jika terjadi kegagalan.

```
# Frontend
react-ui:
  build: ./Frontend
  container_name: myfrontend
  ports:
    - 5173:5173
  restart: always
```

docker-compose.yml

```
FROM node:alpine
WORKDIR /app
COPY ./package.json ./
COPY ./package-lock.json ./
RUN npm install
COPY ..
EXPOSE 5173
CMD ["npm", "run", "dev", "--host"]
```

Dockerfile

# BUKTI DOCKER SUDAH BERHASIL BERJALAN

```
dbcontainer | {"t":{"$date":"2024-05-26T04:34:07.183+00:00"},"s":"I", "c":"STORAGE", "id":22315, "ctx":"initandlisten","msg":"Opening WiredTiger","attr":{"config":"create,cache_size=7351M,session_max=33000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,remove=true,path=journal,compressor=snappy),builtin_extension_config=(zstd=(compression_level=6)),file_manager=(close_idle_time=600,close_scan_interval=10,close_handle_minimum=2000),statistics_log=(wait=0),json_output=(error,message),verbose=[recovery_progress:1,checkpoint_progress:1,compact_progress:1,backup:0,checkpoint:0,compact:0,evict:0,history_store:0,recovery:0,rts:0,salvage:0,tiered:0,timestamp:0,transaction:0,verify:0,log:0]"},"}
myrestapi |
myrestapi | > backend@1.0.0 start
myrestapi | > nodemon index.js
myrestapi |
myrestapi | [nodemon] 3.1.0
myrestapi | [nodemon] to restart at any time, enter `rs`
myrestapi | [nodemon] watching path(s): *.*
myrestapi | [nodemon] watching extensions: js,mjs,cjs,json
myrestapi | [nodemon] starting `node index.js`
myfrontend |
myfrontend | VITE v5.2.11 ready in 207 ms
myfrontend |
myfrontend | → Local: http://localhost:5173/
myfrontend | → Network: http://172.18.0.2:5173/
myrestapi | Server is running on port 4001
dbcontainer | {"t":{"$date":"2024-05-26T04:34:07.763+00:00"},"s":"I", "c":"WTRECOV", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":{"ts_sec":1716
698047,"ts_usec":763787,"thread":"1:0x7fec779fcc80","session_name":"txn-recover","category":"WT_VERB_RECOVERY_PROGRESS","category_id":30,"verbose_level":"DEBUG_1","verbose_level_id":1
,"msg":"Recovering log 22 through 23"}}}
dbcontainer | {"t":{"$date":"2024-05-26T04:34:07.795+00:00"},"s":"I", "c":"WTRECOV", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":{"ts_sec":1716
698047,"ts_usec":794977,"thread":"1:0x7fec779fcc80","session_name":"txn-recover","category":"WT_VERB_RECOVERY_PROGRESS","category_id":30,"verbose_level":"DEBUG_1","verbose_level_id":1
,"msg":"Recovering log 23 through 23"}}}
dbcontainer | {"t":{"$date":"2024-05-26T04:34:07.843+00:00"},"s":"I", "c":"WTRECOV", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":{"ts_sec":1716
698047,"ts_usec":843169,"thread":"1:0x7fec779fcc80","session_name":"txn-recover","category":"WT_VERB_RECOVERY_PROGRESS","category_id":30,"verbose_level":"DEBUG_1","verbose_level_id":1
,"msg":"Main recovery loop: starting at 22/38528 to 23/256"}}}
dbcontainer | {"t":{"$date":"2024-05-26T04:34:07.905+00:00"},"s":"I", "c":"WTRECOV", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":{"ts_sec":1716
698047,"ts_usec":905540,"thread":"1:0x7fec779fcc80","session_name":"txn-recover","category":"WT_VERB_RECOVERY_PROGRESS","category_id":30,"verbose_level":"DEBUG_1","verbose_level_id":1
,"msg":"Recovering log 22 through 23"}}}
dbcontainer | {"t":{"$date":"2024-05-26T04:34:07.954+00:00"},"s":"I", "c":"WTRECOV", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":{"ts_sec":1716
698047,"ts_usec":954217,"thread":"1:0x7fec779fcc80","session_name":"txn-recover","category":"WT_VERB_RECOVERY_PROGRESS","category_id":30,"verbose_level":"DEBUG_1","verbose_level_id":1
,"msg":"Recovering log 23 through 23"}}}
dbcontainer | {"t":{"$date":"2024-05-26T04:34:07.986+00:00"},"s":"I", "c":"WTRECOV", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":{"ts_sec":1716
698047,"ts_usec":986775,"thread":"1:0x7fec779fcc80","session_name":"txn-recover","category":"WT_VERB_RECOVERY_PROGRESS","category_id":30,"verbose_level":"DEBUG_1","verbose_level_id":1
,"msg":"recovery log replay has successfully finished and ran for 223 milliseconds"}}}
dbcontainer | {"t":{"$date":"2024-05-26T04:34:07.986+00:00"},"s":"I", "c":"WTRECOV", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":{"ts_sec":1716
698047,"ts_usec":986852,"thread":"1:0x7fec779fcc80","session_name":"txn-recover","category":"WT_VERB_RECOVERY_PROGRESS","category_id":30,"verbose_level":"DEBUG_1","verbose_level_id":1
,"msg":"Set global recovery timestamp: (0, 0)}}}
dbcontainer | {"t":{"$date":"2024-05-26T04:34:07.986+00:00"},"s":"I", "c":"WTRECOV", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":{"ts_sec":1716
698047,"ts_usec":986863,"thread":"1:0x7fec779fcc80","session_name":"txn-recover","category":"WT_VERB_RECOVERY_PROGRESS","category_id":30,"verbose_level":"DEBUG_1","verbose_level_id":1
,"msg":"Set global oldest timestamp: (0, 0)}}}
dbcontainer | {"t":{"$date":"2024-05-26T04:34:07.987+00:00"},"s":"I", "c":"WTRECOV", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":{"ts_sec":1716
698047,"ts_usec":987185,"thread":"1:0x7fec779fcc80","session_name":"txn-recover","category":"WT_VERB_RECOVERY_PROGRESS","category_id":30,"verbose_level":"DEBUG_1","verbose_level_id":1
,"msg":"recovery rollback to stable has successfully finished and ran for 0 milliseconds"}}}
dbcontainer | {"t":{"$date":"2024-05-26T04:34:07.988+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":{"ts_sec":1716
698047,"ts_usec":988229,"thread":"1:0x7fec779fcc80","session_name":"WT_SESSION.checkpoint","category":"WT_VERB_CHECKPOINT_PROGRESS","category_id":6,"verbose_level":"DEBUG_1","verbose_
level_id":1,"msg":"saving checkpoint snapshot min: 1, snapshot max: 1 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 343"}}}
dbcontainer | {"t":{"$date":"2024-05-26T04:34:07.991+00:00"},"s":"I", "c":"WTRECOV", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":{"ts_sec":1716
```

CHECK OUT OUR WEB !!!

---

<https://game-review-full.vercel.app/>

# G I T H U B   R E P O S I T O R Y

---

<https://github.com/IvanYuantama/Dockerize-web>

**THANK YOU**