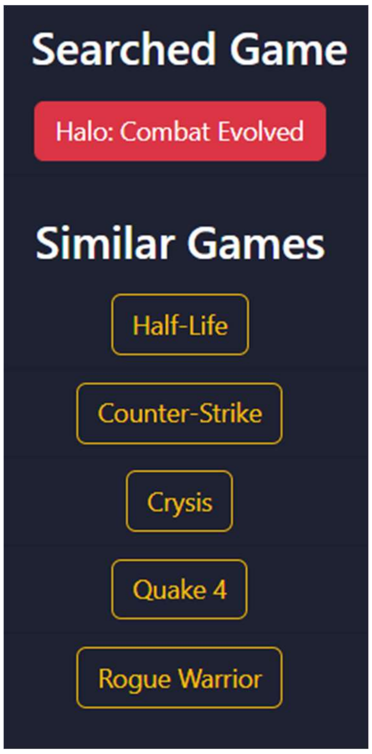
2022

NextGame



CAB432 Assignment 1

Connor McSweeney

n8844488

21/09/2022

Contents

Introduction

Mashup Purpose & description

This application allows users to search for and discover information relating to a particular video game, as well as games considered to be similar. A user can search for a specific game by title and is presented with details pertaining to that game - namely a rating, summary, video review, video trailer, story summary, and the beginning of a playthrough. In addition to this, a user is presented with a list of five games similar to the one searched, and has the option of viewing the above mentioned details for each of those games. In essence, it is an application that allows users to discover and learn about games that they are likely to enjoy.

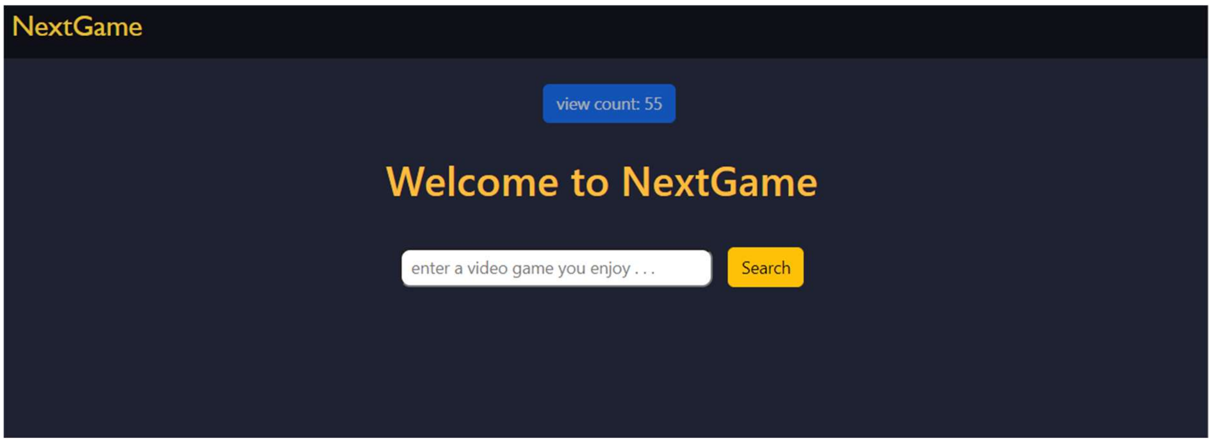


Figure 1

Services used

IGDB Games API (v4)

Returns a list of information related to one or more games - specific fields can be listed. Can be used with a *search* field if a game's *Id* is not known. The *Id* of a game can be used to find more information about that game. Requires authorisation and a client Id. Returns JSON format.

Endpoint: https://api.igdb.com/v4/games

Docs: https://api-docs.igdb.com/#about

YouTube Data API - Search

Returns a list of data objects relating to a video that would be listed using YouTube's standard public search function. The *videoId* can be retrieved from this response and used within a HTML <*iframe*> tag. Requires an API key and returns JSON format.

Endpoint: https://youtube.googleapis.com/youtube/v3/search?

Docs: https://developers.google.com/youtube/v3/docs

Persistence Service

The persistence service used for the page view counter is DynamoDB, hosted on the Amazon Web Services platform.

- https://aws.amazon.com/dynamodb/
- Item name: n8844488_assignment1

The application itself has been packaged into a Docker image and uploaded to DockerHub. The link to is listed below.

- https://hub.docker.com/repository/docker/connoraus/cab432-assignment1

Mashup Use Cases and Services

US 1: Find similar games

As a	gamer
I want	to find games similar to those I already know and enjoy
So that	I might know what to buy/play in the future

Using the search bar present at the top of the page, a user will search for a game that they enjoy (**Figure 2**). The middleware will then make two requests to the IGDB API. The first request returns a short list of the closest matching games based on the user's input. For each game in that list, the *Id* of game and the *Id* of *similar_games* are listed. The second request takes the *Id* of the closest matching game in the search result, as well as the *Id* of its listed *similar_games* to query further information relating to each game. The resulting names for the matching and similar games are presented to the user (**Figure 3**).

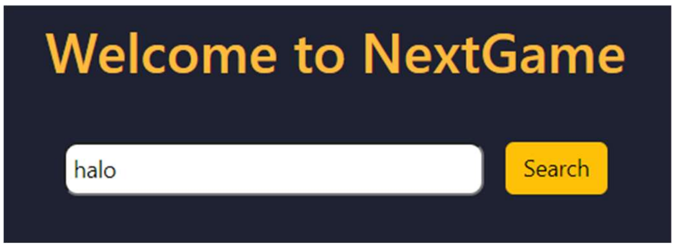


Figure 2

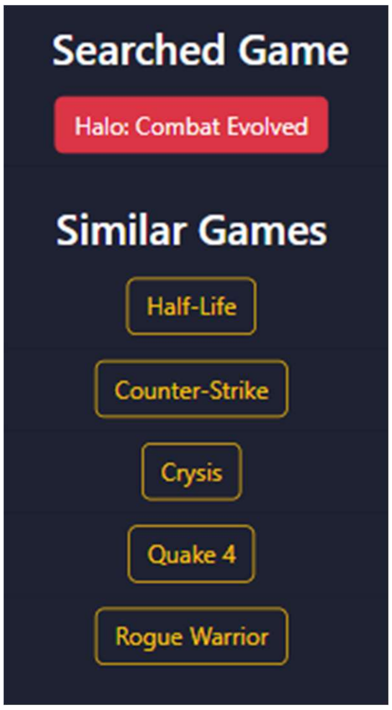
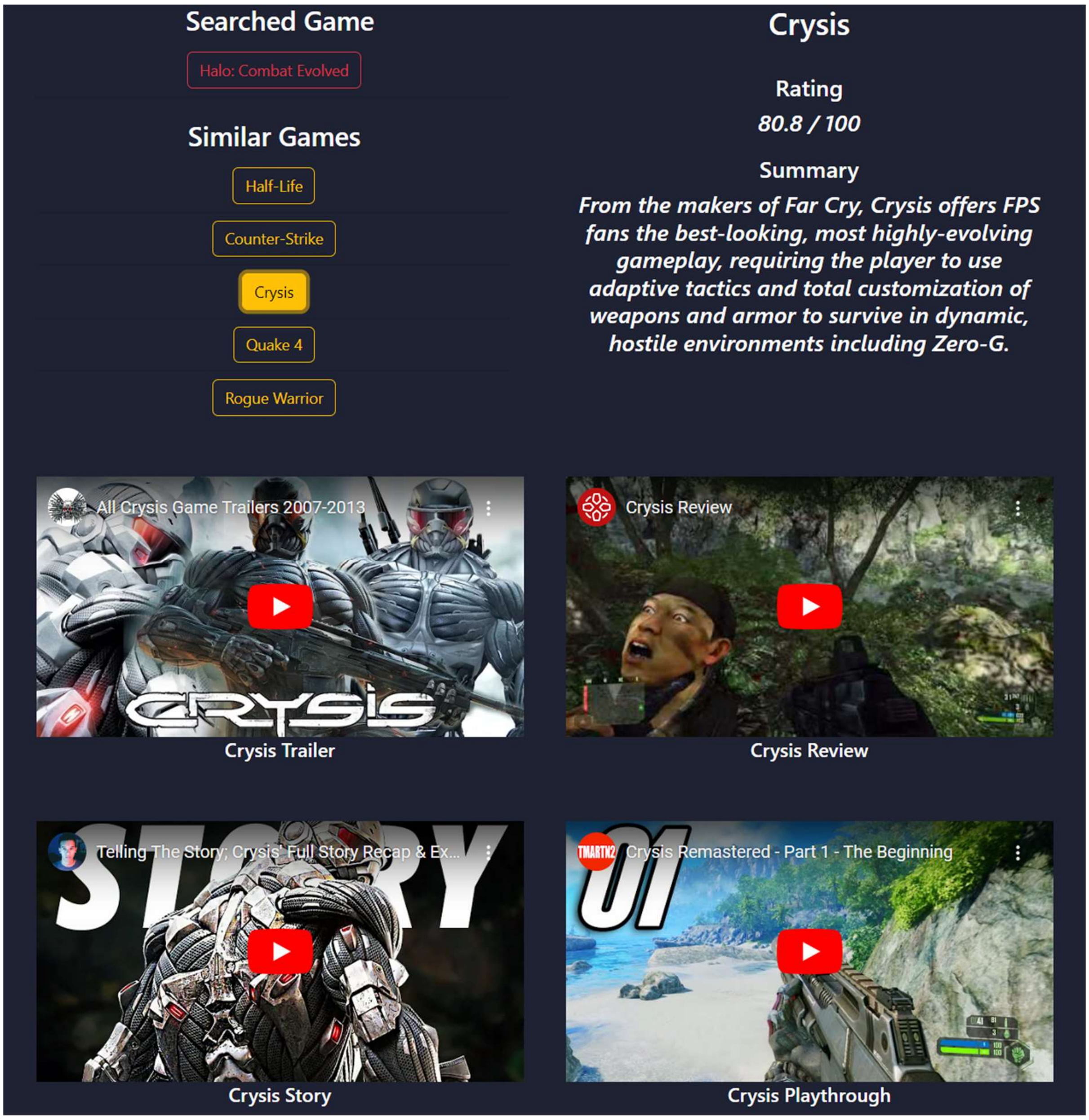


Figure 3

US 2: Learn more about a game

As a	gamer
I want	find information relating to a game
So that	I have an idea of whether I might like it

Each of the games presented (**Figure 3** above) can be selected by the user to view further information about the game. Selecting a game will display the *name, rating,* and *summary* (returned during the second IGDB request). Additionally, the middleware will be instructed to make four calls to the YouTube API each with a different search query related to the game in question. The resulting *videoId*'s are used to embed videos within the webpage for the user to view. Each video has a different topic, allowing the user to view a trailer, review, story summary and playthrough of that game (**Figure 4**).



From the makers of Far Cry, Crysis offers FPS fans the best-looking, most highly-evolving gameplay, requiring the player to use adaptive tactics and total customization of weapons and armor to survive in dynamic, hostile environments including Zero-G.

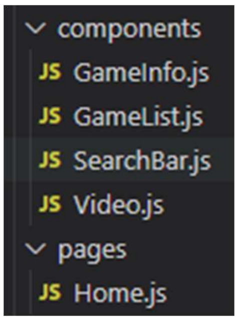
Figure 4

Technical breakdown

Architecture and Data Flow

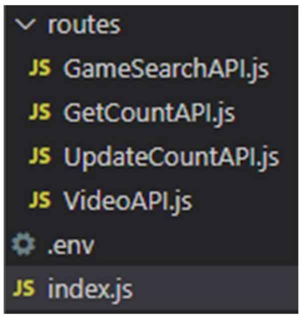
The application includes a single page front-end using React.js and a middleware service using node.js and express. The front-end renders a *Home* page that imports four component types to populate the page (**Figure 5**). *Home* also handles all requests and responses made to the middleware.

The middleware service receives each request at the *index.js* file and implements a router to pass each request to one of four routes (**Figure 6**). Each of these routes then handles the request and response data between the middleware and application front-end, as well as any external APIs. All responses to the front-end are packaged and sent in JSON format.



```
components
  GameInfo.js
  GameList.js
  SearchBar.js
  Video.js
pages
  Home.js
```

Figure 5



```
routes
  GameSearchAPI.js
  GetCountAPI.js
  UpdateCountAPI.js
  VideoAPI.js
.env
index.js
```

Figure 6

Upon opening the webpage, the application will render a title, a search bar, and a view count using a combination of the *SearchBar* component and standard Javascript. Passed to the *SearchBar* component is a callback function that sets the value of the *search* variable when the user enters an input and clicks the *Search* button. Any change to the *search* variable triggers a useEffect function that invokes a fetch request to the middleware.

This fetch request is handled by the *GameSearchAPI*. This route invokes two post - necessary due to authorisation requirements - requests to the IGDB API. These requests are handled using a single route as they are always executed in conjunction and utilize very similar request configurations. The first request uses the input parameter (*search*) to find a matching game. The result is parsed and the first item in the list is used to repeat the process using the *Id* of that game and the *Id* of listed *similar_games*. This second request receives information specific to each game and the result is returned to the front-end. If the initial request returns no matching results (an empty array), a value of '0' is returned to the front-end, triggering the display of a custom error message.

The resulting data is passed to the *GameList* and *GameInfo* components in the front-end and are rendered on screen. The searched game is set as the default *selectedGame* variable, triggering the useeffect used to fetch a collection of YouTube *videoId*s. As described above, each of these requests is handled by a middleware route, in this case, the *VideoAPI*. This API invokes a get request and receives the top item (video) matching the search input. For each item, only the *videoId* is returned to the front-end. Each *videoId* is then used to generate an embedded video within the webpage using the *Video* component.

The view count seen at the top of the screen is updated and rendered using a useEffect function triggered upon first opening the page. This use effect first requests the value then updates the value using separate queries. Separate API's were considered best practice due to the possibility of extending the application to multiple pages that may display but not update the view count upon loading.

The view count API's use *get* and *update* requests to respectively get and update the value of the variable *count* housed in a DynamoDB database hosted on Amazon Web Service infrastructure. Each request/response simply passes through and/or manipulates the value of *count*.

Throughout the application, global variables are used by the middleware service to authorize each API request. Additionally, many front-end fetch requests embed parameters into the fetch URL for use by the middleware. Lastly, logging is implemented throughout the application front-end and middleware for debugging purposes.

Figure 7 displays a sequence diagram highlighting the significant interactions and flow of data between the entities involved in and with the application.

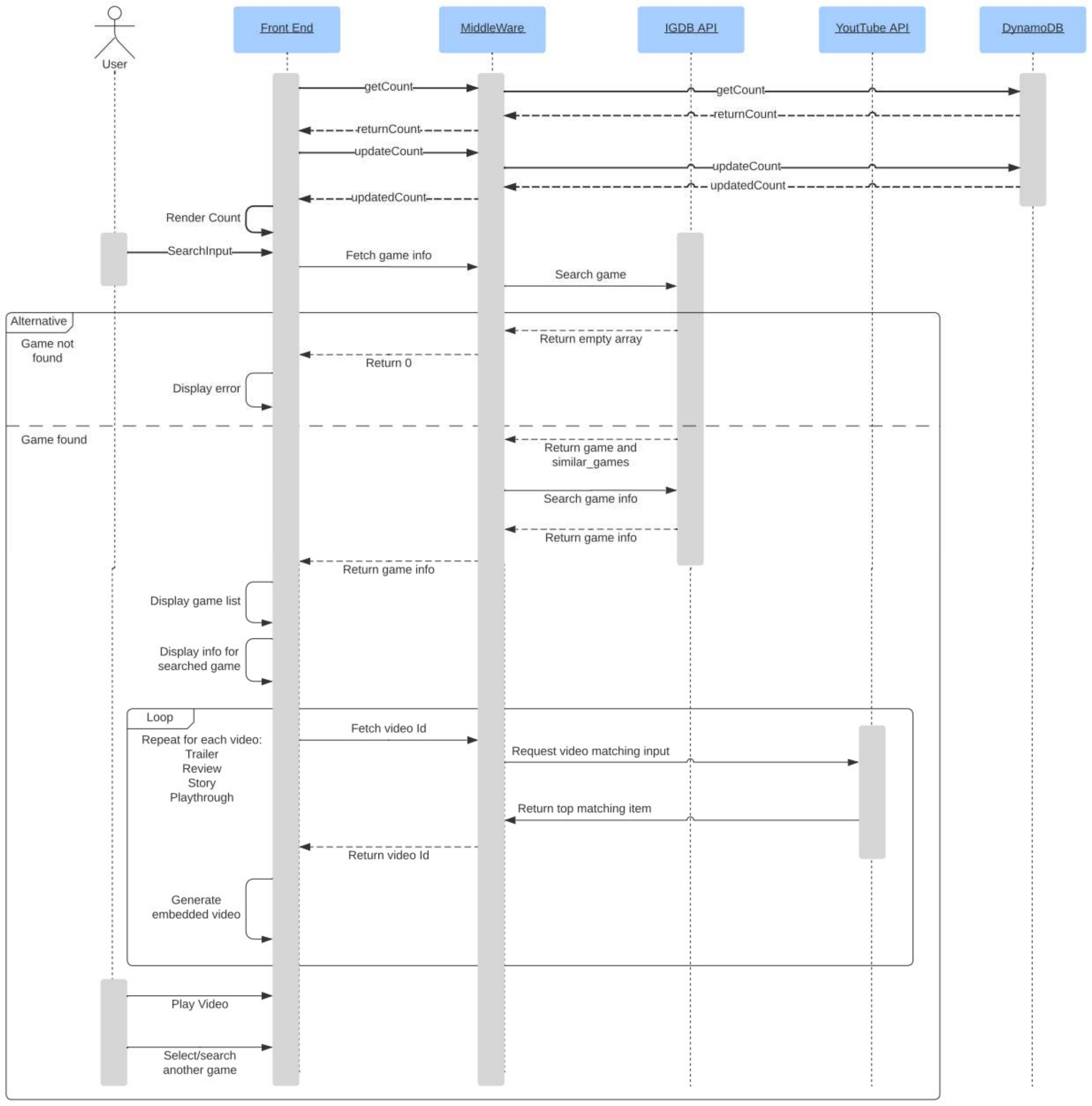


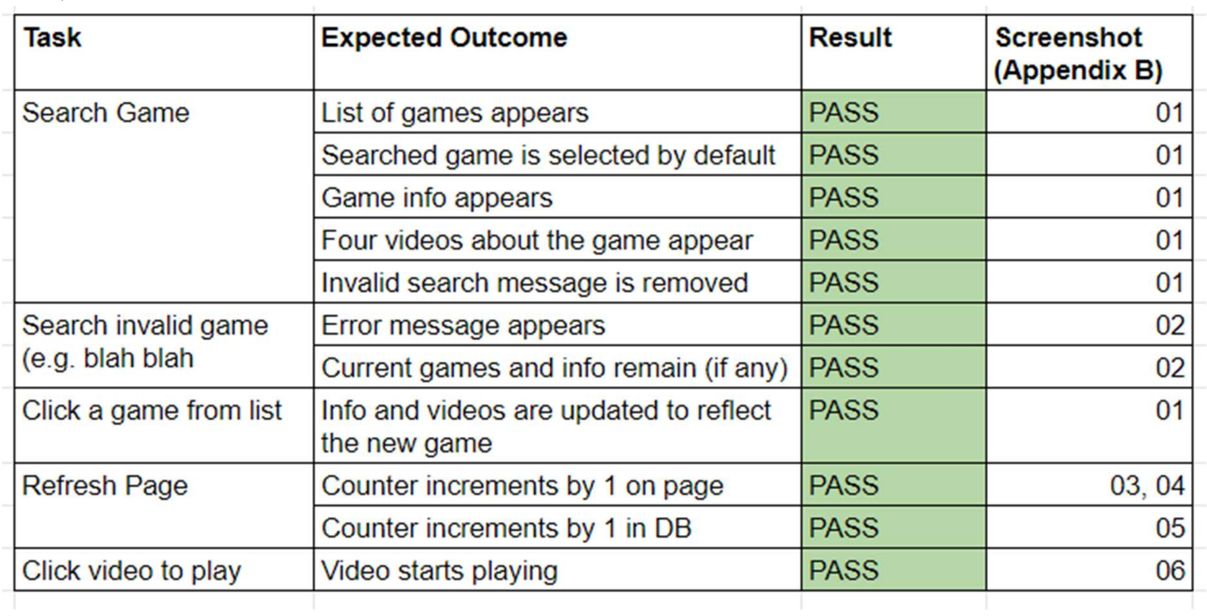
Figure 7

Deployment and the Use of Docker

Docker was used to build and deploy this application using a container hosted on an AWS virtual machine. A *Docker* file was configured to ensure that the image used node:erbium, copied the pertinent application files, declared environment variables, installed application modules, built the front-end, exposed a port, and ran the server. Additionally, a *.dockerignore* was used to ensure that the *node_modules* and *.env* file was excluded from the Docker image. All specific environment variables and port connections are declared during the running of a Docker container.

A screenshot of the *Docker* file can be seen in **Appendix 1 - Figure 01**. Furthermore, the Docker image has been pushed to DockerHub, and can be found at *https://hub.docker.com/repository/docker/connoraus/cab432-assignment1*.

Test plan



Task	Expected Outcome	Result	Screenshot (Appendix B)
Search Game	List of games appears	PASS	01
	Searched game is selected by default	PASS	01
	Game info appears	PASS	01
	Four videos about the game appear	PASS	01
	Invalid search message is removed	PASS	01
Search invalid game (e.g. blah blah	Error message appears	PASS	02
	Current games and info remain (if any)	PASS	02
Click a game from list	Info and videos are updated to reflect the new game	PASS	01
Refresh Page	Counter increments by 1 on page	PASS	03, 04
	Counter increments by 1 in DB	PASS	05
Click video to play	Video starts playing	PASS	06

Difficulties / Exclusions / unresolved & persistent errors /

The application implements all functionality specified within the assessment outline. However, certain challenges identified during the development of the application have been listed below.

- A continuing issue faced in the deployment of this application is the presence of an occasional Cross-Origin Resource Sharing (CORS) error linked to the embedding of YouTube videos into HTML using the *<iframe>* tag. It appears that these errors - present in the browser console - are domain specific and cannot be overcome without configuration of the browser used to host the application. Despite these errors, the videos behave as expected. Nevertheless, the *cors* module has been installed on the middleware service.
- Querying the IGDB API when searching for a particular game returns a list of games, each with a list of *similar_games*. While the search returns a list of games ordered by how closely they match the search input, the list of *similar_games* is simply provided in ascending order. Therefore, limiting the number of *similar_games* returned to the user will may exclude games that are most closely related to the searched game. Furthermore, as each request only returns a maximum of ten results, the last game in the list of *similar_games* must be excluded to ensure that the searched game is not excluded when querying for further game data (i.e. not

excluding a similar game would result in a query for eleven games, thus truncating the game with highest game *Id* from the response).

- When searching for videos using the YouTube API, there is no way of knowing whether a video has an age restriction. This, however, does not present a major issue, as any embedded videos requiring age verification provide links to the video hosted on the YouTube domain. This allows users to verify their age if they wish to view the video.
- Screen reading has not implemented in this application due to the limited focus on UI aspects

Extensions

Possible extensions have been identified below.

- The addition of written reviews or links to written reviews were considered for this application. This presented difficulties as professional reviews do not exist for all games listed in the IGDB database.
- A search bar linked to the YouTube API was considered in place of the fourth embedded video. Entering keywords into the search bar would be combined with the game name and used to query the YouTube video database.

User guide

Upon opening the webpage, a user will be presented with a *View Count*, page title, and a search bar (**Figure 8**). Placeholder text prompts the user to enter the name of a video game.

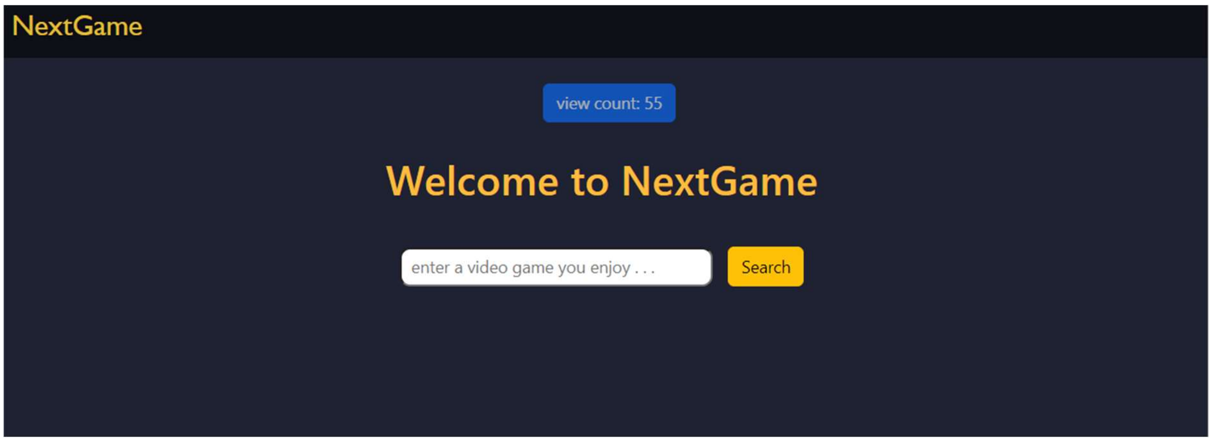
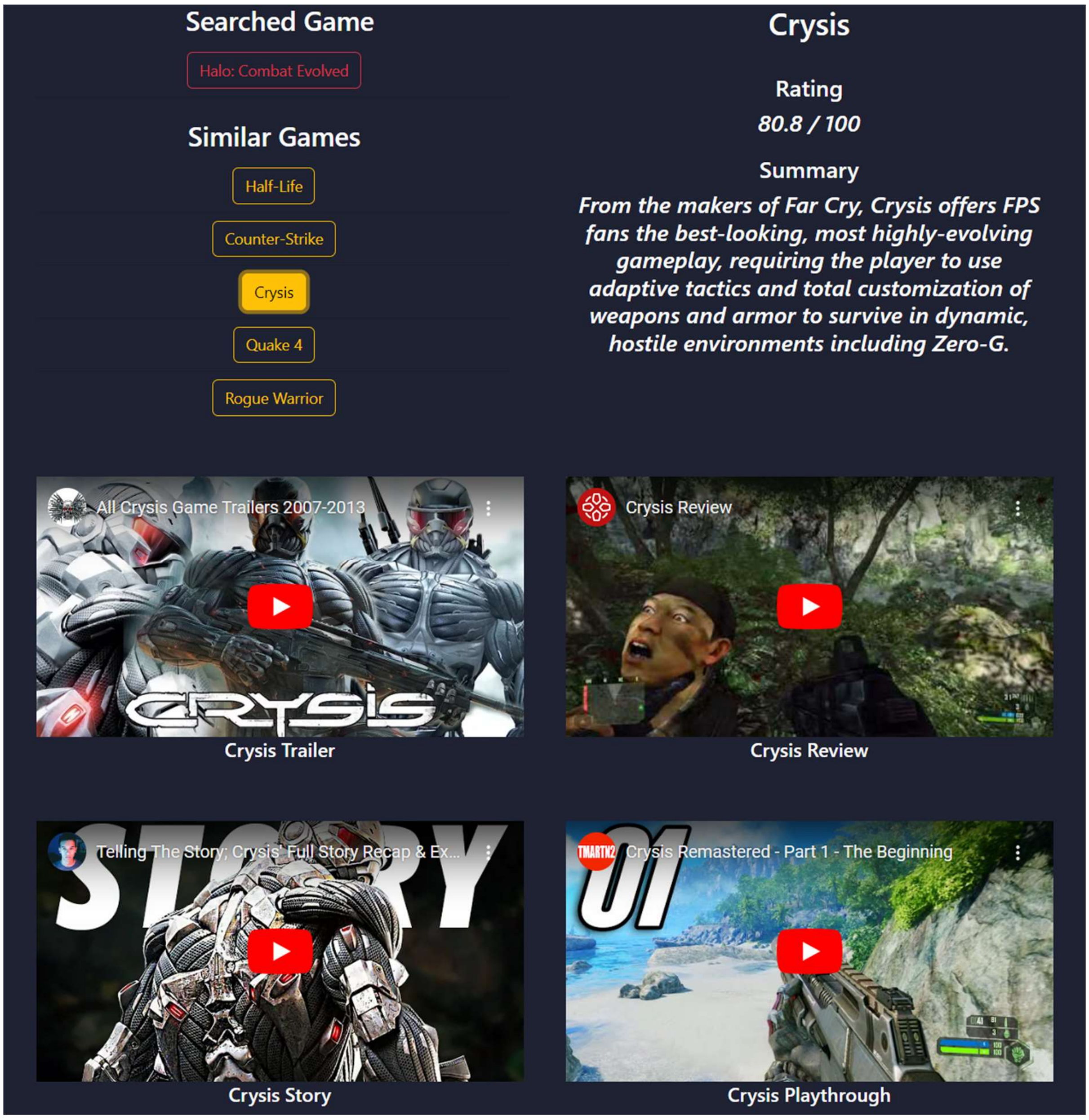


Figure 8

Entering an input and clicking the *Search* button will present the user with a list of games. The topmost game highlighted in red is the game most closely matching the search input. The games highlighted in yellow are games deemed similar to the searched game. Each game is a clickable button. Selecting a game will fill the button image, denoting the current selection, and present the user with the *name, rating, summary,* and four videos relating to that game (**Figure 9**). The searched game is selected by default.


Searched Game

Halo: Combat Evolved

Similar Games

Half-Life

Counter-Strike

Crysis

Quake 4

Rogue Warrior

Crysis

Rating

80.8 / 100

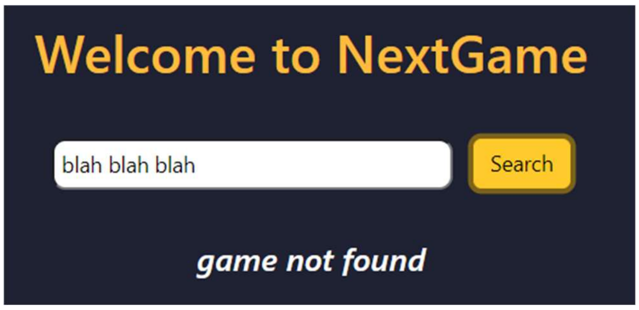
Summary

From the makers of Far Cry, Crysis offers FPS fans the best-looking, most highly-evolving gameplay, requiring the player to use adaptive tactics and total customization of weapons and armor to survive in dynamic, hostile environments including Zero-G.

All Crysis Game Trailers 2007-2013
Crysis Trailer
Crysis Review
Crysis Review
Telling The Story; Crysis' Full Story Recap & Ex...
Crysis Story
Crysis Remastered - Part 1 - The Beginning
Crysis Playthrough

Figure 9

If the input for the game search does not match any games within the IGDB database, a message will appear stating `game not found` (**Figure 10**). Importantly, this does not remove the list of current games and information present (if any).


Welcome to NextGame

blah blah blah

Search

game not found

Figure 10

Analysis

Question 1: Vendor Lock-in

The application includes two external APIs excluding that of the persistent service. In the event that the IDGB API became unavailable, it could be replaced with a similar video game database such as RAWG or TheGamesDB. Both options house a comparable number of games and game information, however, each API is queried in a different manner, and neither returns a list of similar games. Lastly, if the YouTube API became unavailable, a competitor such as DailyMotion could be used as it contains a search API and it is possible to embed DailyMotion videos within a web application. However, DailyMotion is unlikely to have the breadth of videos for selection which may present issues, particularly in the case of niche games.

In the event the DynamoDB API was removed, other AWS database services (e.g S3) could be used as a replacement with little hassle as the authentication details could be accessed via the same method. Replacing the vendor entirely, however, would require the use of new authentication details linked to an account with that vendor. Examples include an Azure SQL Database or Azure Database for PostgreSQL. Alternatively, self-managed databases such as MySQL could be used. In such a case, it would be possible to run such a database via a Docker image and deploy it alongside the web application.

Question 2: Container Deployment

This application is best deployed and scaled using a container-based service. This is because the application is not tied to any specific operating system infrastructure. Therefore, the application can benefit from the efficiencies of running a container-based service. Additionally, as the application is quite small and is not demanding of the system CPU or memory, the majority of request wait time can be associated with message transfer time. Thus, there is a greater need for scaling out rather than scaling up. However, deployment of the application directly to a virtual machine would provide a greater amount of isolation in the event the user accounts were implemented and security became a prioritization.

If the application were to be deployed at scale, a persistent service would be utilised to house queried game data. This level of persistence would also be used in the case of user/account information, however, would not be used to house video information (at least not for all video types) due to the constantly adapting nature of YouTube and videos relating to video games. Additionally, load balancing would be used to house recently searched game data and video data in order to reduce request time, database load, and load on system resources.

References

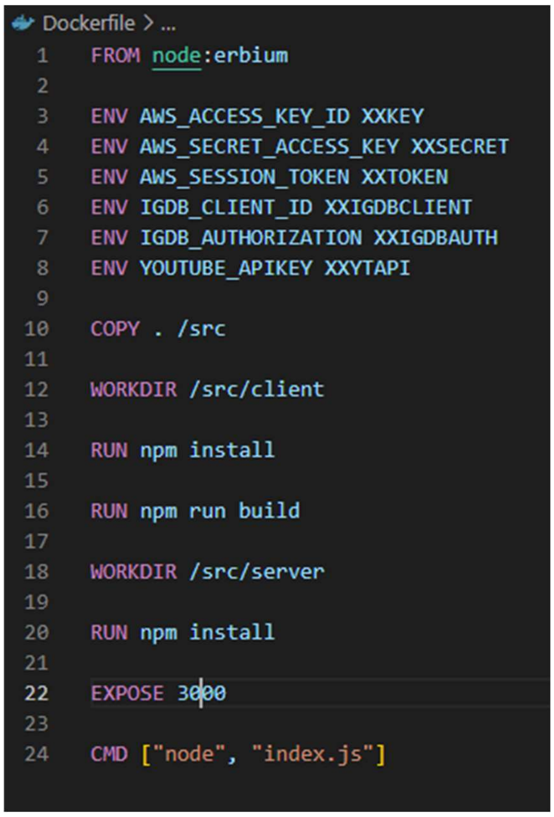
aws. (n.d.). *DynamoDB API - Amazon DynamoDB*. aws. Retrieved September 17, 2022, from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.API.html

Google. (n.d.). *YouTube Data API | google developers*. Google. Retrieved September 10, 2022, from https://developers.google.com/youtube/v3

IGDB. (n.d.). *IGDB API - About*. IGDB API docs. Retrieved September 10, 2022, from https://api-docs.igdb.com/#about

Appendices

Appendix A: Deployment and the Use of Docker

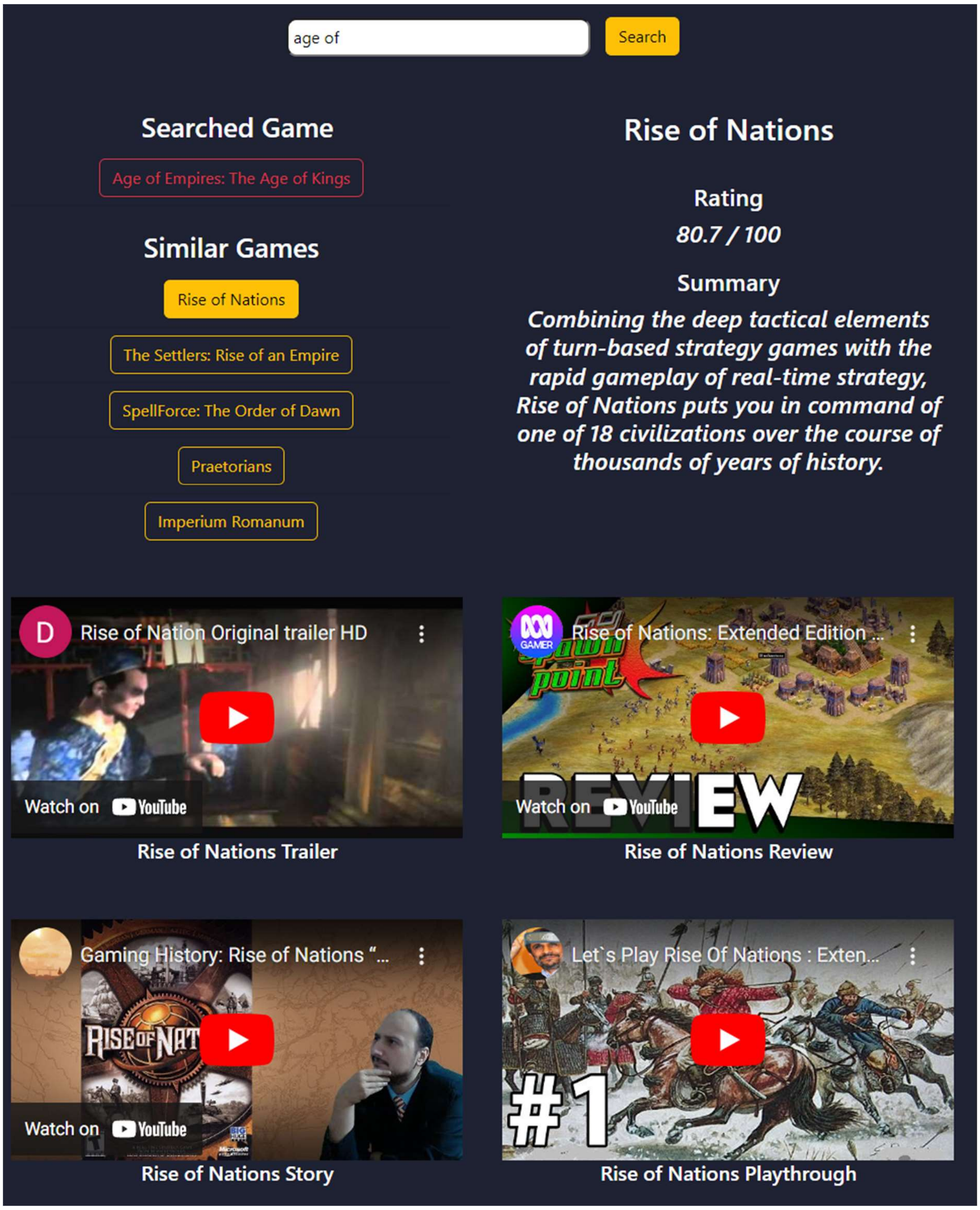


Dockerfile > ...

```
FROM node:erbium

ENV AWS_ACCESS_KEY_ID XXKEY
ENV AWS_SECRET_ACCESS_KEY XXSECRET
ENV AWS_SESSION_TOKEN XXTOKEN
ENV IGDB_CLIENT_ID XXIGDBCLIENT
ENV IGDB_AUTHORIZATION XXIGDBAUTH
ENV YOUTUBE_APIKEY XXYTAPI

COPY . /src

WORKDIR /src/client

RUN npm install

RUN npm run build

WORKDIR /src/server

RUN npm install

EXPOSE 3000

CMD ["node", "index.js"]
```

Figure 1

Appendix B: Test Plan



age of

Search

Searched Game

Age of Empires: The Age of Kings

Similar Games

Rise of Nations

The Settlers: Rise of an Empire

SellForce: The Order of Dawn

Praetorians

Imperium Romanum

Rise of Nations

Rating

80.7 / 100

Summary

Combining the deep tactical elements of turn-based strategy games with the rapid gameplay of real-time strategy, Rise of Nations puts you in command of one of 18 civilizations over the course of thousands of years of history.

Rise of Nation Original trailer HD

Rise of Nations Trailer

Rise of Nations: Extended Edition ...

Rise of Nations Review

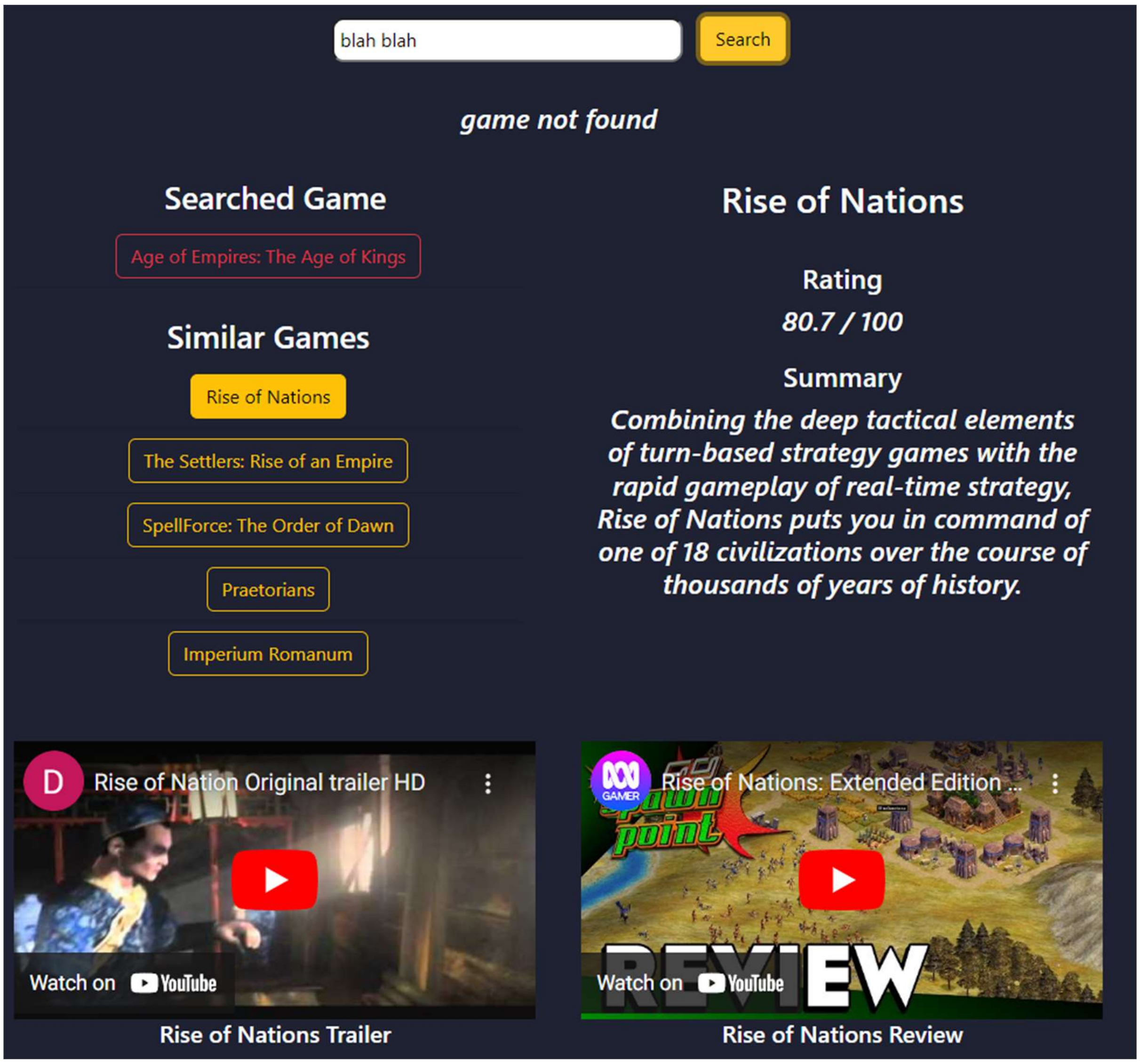
Gaming History: Rise of Nations "...

Rise of Nations Story

Let`s Play Rise Of Nations : Exten...

Rise of Nations Playthrough

Figure 1


blah blah

Search

game not found

Searched Game

Age of Empires: The Age of Kings

Similar Games

Rise of Nations

The Settlers: Rise of an Empire

SpellForce: The Order of Dawn

Praetorians

Imperium Romanum

Rise of Nations

Rating

80.7 / 100

Summary

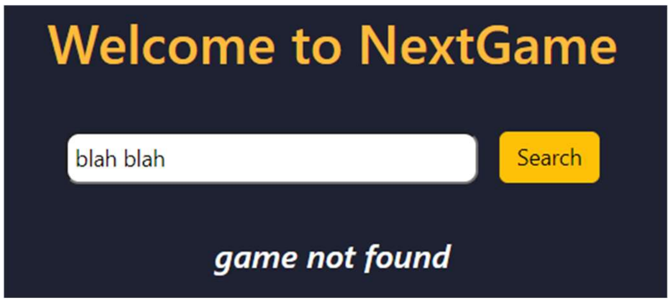
Combining the deep tactical elements of turn-based strategy games with the rapid gameplay of real-time strategy, Rise of Nations puts you in command of one of 18 civilizations over the course of thousands of years of history.

Rise of Nation Original trailer HD
Watch on YouTube
Rise of Nations: Extended Edition ...
Watch on YouTube

Rise of Nations Trailer

Rise of Nations Review

Figure 2


Welcome to NextGame

blah blah

Search

game not found

Figure 3

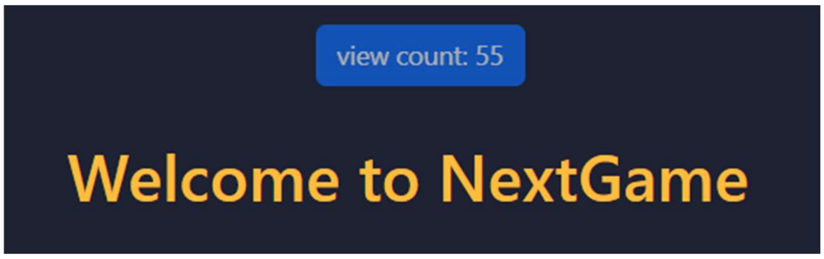


Figure 4

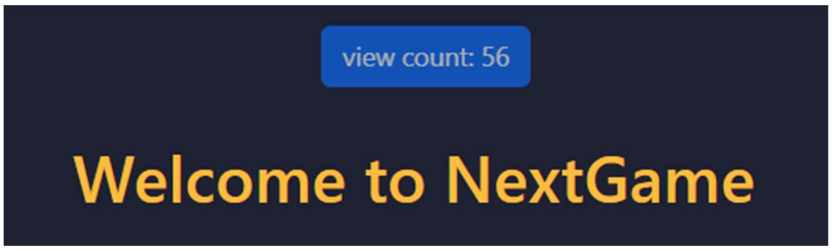
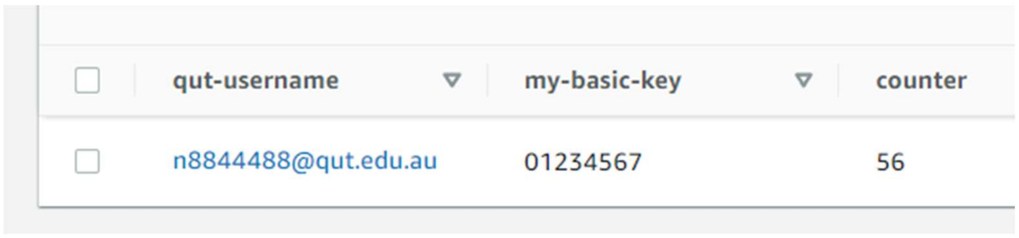


Figure 5



qut-username	my-basic-key	counter
n8844488@qut.edu.au	01234567	56

Figure 6

D
Rise of Nation Original trailer HD
Watch Later
Share
0:31 / 2:36
YouTube
Rise of Nations Trailer

Figure 7