Game Recommendation Service

Project By: Ayush Kumar

Contents

1. Introduction

2. System Overview

3. Architectural Strategies

4. Dataset and Database used

5. The Recommendation Engine

6. The Backend

7. The Frontend

Introduction:

A Recommendation System that uses machine learning for suggesting a list of games based on user's selection/

Built using Python Flask, React.JS and completely containerized using docker.

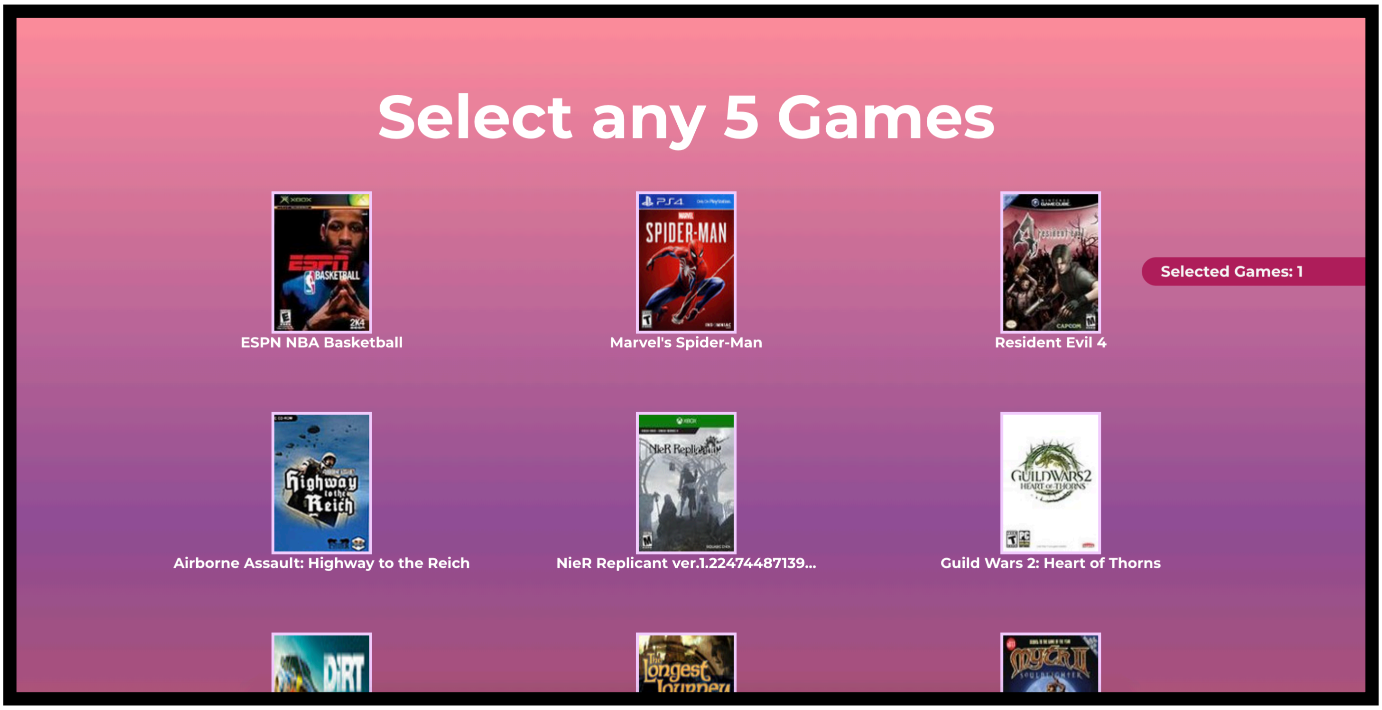
- The Gaming Industry by the year 2022 has emerged as a giant that seems to grow to reach unprecedented levels.
- The Industry sees more revenue than even the film and TV industry combined.
- As a still growing venture with profitable future many companies have foraged into the gaming business, in which a new popular model of online gaming service is provided.

Goals:

- This Recommendation WebApp aims to provide a recommendation of games from a catalog, for the user based on their selections.
- This will allow the user to get access to a list games matching their preferences.

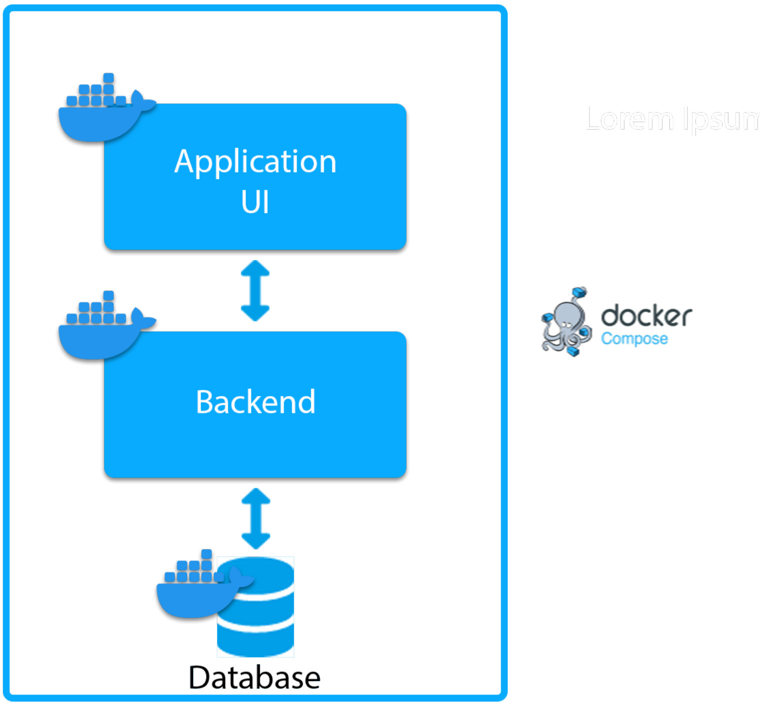
To get Recommendations:

- Users will be able to select 5 games from the provided random catalog of games.
- The Recommendation System will generate a list of similar games, that will be presented to the user.


Select any 5 Games
ESPN NBA Basketball
Marvel's Spider-Man
Resident Evil 4
Selected Games: 1
Airborne Assault: Highway to the Reich
NieR Replicant ver.1.22474487139...
Guild Wars 2: Heart of Thorns

Game List Section

System Overview:



High Level Diagram

Components of the Micro-services:

The Docker Containers used here are:

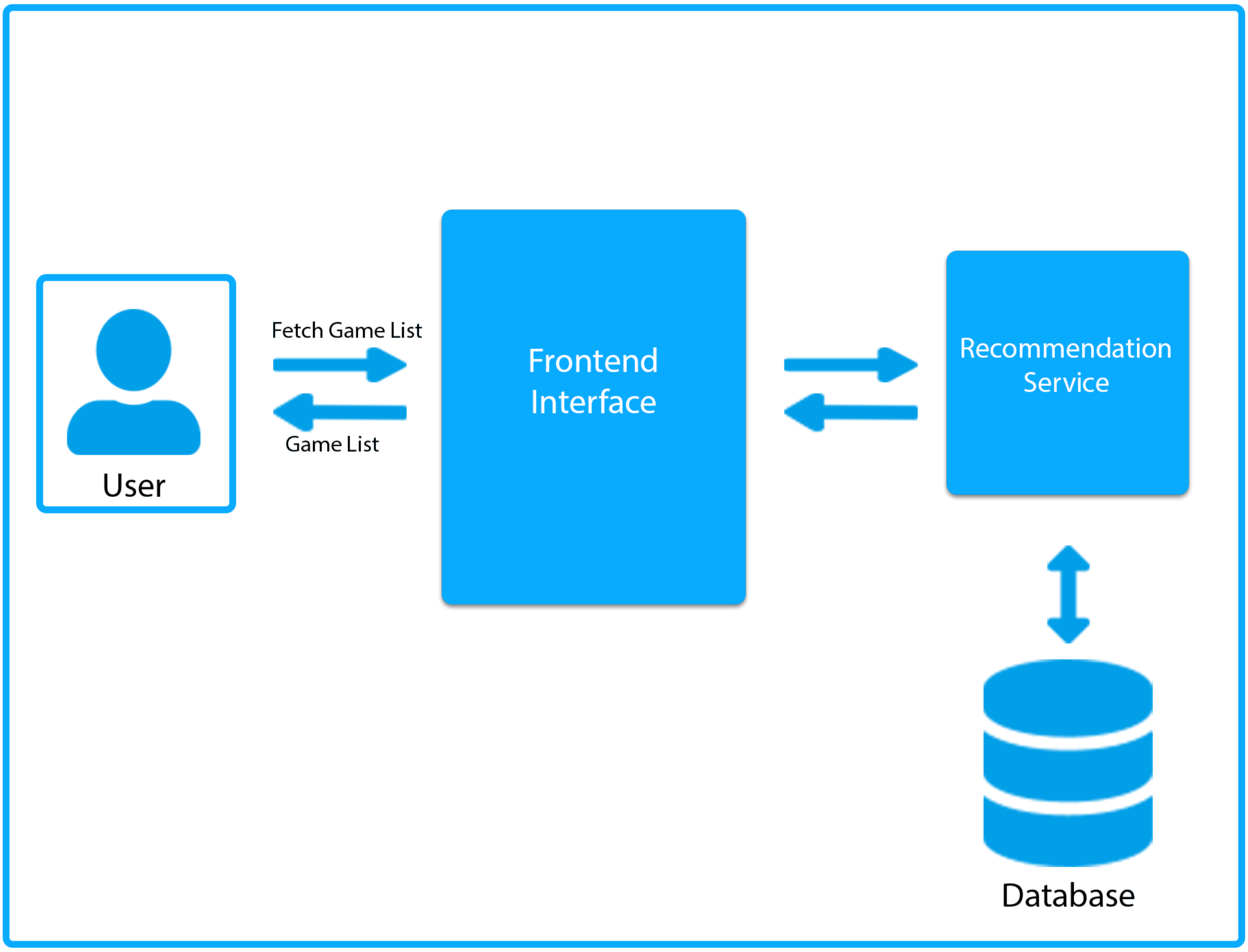
Database Container: mysql:latest
Backend Container: python:3.9.0
Frontend Container: React JS

The Database used is MySQL.
The Backend is created using Python.
The Frontend is created using React.Js

Architectural Strategies

The game recommendation system design has been divided into 3 micro services that run with the help of docker compose, where the containerised instances of the database, the backend and the frontend are run.
This allows for better application deployment and business agility.

System Architecture:



The User can make get requests to fetch the initial game list from the database.
The initial game list is a random collection of 12 games, that have score above 80 out of 100.

Upon Selection of 5 games, the Post request is sent to the backend service to generate 3 recommendations for each of the selected games.
Returning list of 15 games.

Dataset and Database Used:

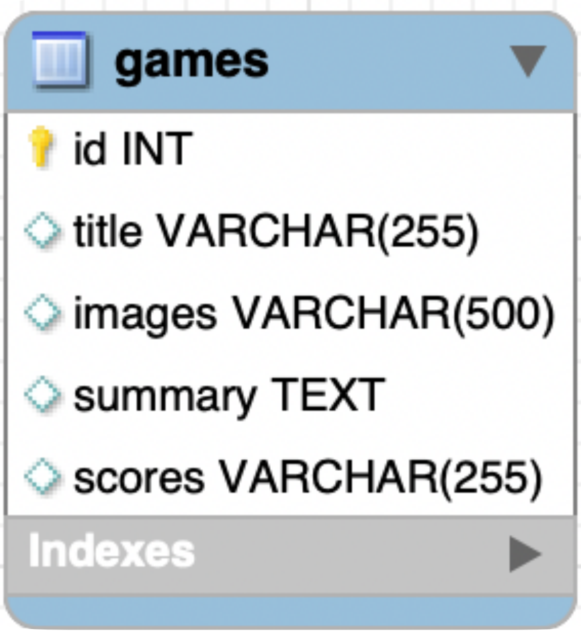
The data is generated from scraping 200 pages of the top games of all times list from meteoritic website, which is a popular site that hosts and catalogs game reviews publicly.

The Link from where the dataset is created from:

https://www.metacritic.com/browse/games/score/metascore/all/all/filtered?page=

The scraped data is then loaded into the database container.

THE DATABASE SCHEMA:



games

id INT

title VARCHAR(255)

images VARCHAR(500)

summary TEXT

scores VARCHAR(255)

Indexes

Table Name: games
Column Names: id, title, images, summary, scores
In total there are **19945** entries that have been scraped from the source website and stored in the database.

The Recommendation Engine:

For generating recommendations from the user’s choices, we have chosen to use word to vector approach for recommendation systems.

In this we generate vectors for the tokenised summary of the games for all the games provided and then to get the cosine similarity, we generate data for the cosine distance for each word vector generated.

Using this, we calculate the top 5 closest similar games for each of the game chosen by the user.

The names from the user are received from the payload from the POST request.

The rows from the database for each recommendations are then exported back to UI as response.

The Backend:

The Backend is made is using Python, where Flask is used to generate the api endpoints and handle the database connection.

THE API ENDPOINTS USED ARE:

- **GAME-LIST:**
  Endpoint: /game-list;
  Method: [GET]
  Parameters:

Parameter	Type	Description
id	Number	ID Number of the Game Entry from the Database
Title	String	Title of the Game
Images	String	URL for the Image Icon of the Game
Summary	String	Summary of the Game
Scores	String	Metracritic Score Value for the Game

- **RECOMMENDATIONS:**
  Endpoint: /api/recommendations
  Method: [POST]
  Parameters: The Post Request will send an array of titles of the games, selected to the server, If successful, the server will send the response in the JSON format with same as above parameters for each recommended game.

The User Interface:

The User Interface is made using React.JS

The Main Components are:

- **Hero:**
  Landing page for the web app with 3D UI that can viewed by dragging the mouse on the screen
- **GameList**:
  List of Games for User's Selection of Games that is generated randomly from the database.
  The User needs to select any 5 from the choices provided. As soon as the number of images selected equals to 5, post request is made to get the recommendations

- **GameRecs**:
  Pop-up for the list of Recommended Games generated by the Model from the backend.

- **GameCard**:
  Pop-up card for information when each game is selected

- **GameRender:**
  Components for Game Icon, Game Back and Game Card

Further Details for the frontend components can be availed from the Frontend Docs folder, via the index.html