CS480 – Course project

Summer 2021

Database: epic_price_tracking

Ajay, Sanjay and Sampath

Description:

The store has games which have a unique name, publisher, genre, and other details associated with the game. Each publisher had published multiple games over the years. Each game might have addons like the DLC but require the base game to function. Our web scraping tool will scrape the Epic Game Store for prices weekly and record the prices for all the games and addons. Users can register for notifications from the web tool on price drops for specific games. For every time we run our web scraping tool, we compare the current price to the previous price, and notify all users who subscribed to be notified for that particular game. Our GUI tool, a web application for users to select games by publisher, genre, and view price history of the game. It will also allow users to select add-ons associated with a game and view the price history. Users will have full freedom to choose any permutation of selections for notifications on price drop.

Part 2 – CRUD (Create, read, update,and delete)

List of strong entities:

1. game
2. user
3. genre

List of weak entities:

1. user_tracks_game
2. game_genre
3. price_record

We will implement a web scraping script capable of fetching the required information about the games from the Epic Games Store www.epicgames.com.

This Script will be exposed via an API that will be available to the admin to trigger a fetch from the website or can also be scheduled to perform a web scraping operation on a timely basis.

We will implement the following functionality using Java and SQL with necessary GUIinterfaces.

1. Insert/delete/update/read a game (all attributes except the game_id). The game_id should be generated by the system automatically using MySQL autoincrement.

2. Insert/delete/update/read a price_record (all attributes except the record_id). The record_id should be generated by the system automatically using MySQL autoincrement.

3. Insert/delete/update/read a user (all attributes except the user_id). The user_id should be generated by the system automatically using MySQL autoincrement.

4. Insert/delete/update/read a user_tracks_game table which contains the games that users subscribe to receive discount information.

5. Insert/delete/update/read a genre table (all attributes except the genre_id). The genre_id should be generated by the system automatically using MySQL autoincrement.

6. Insert/delete/update/read a game_genre table which is the junction table for genre and game tables.

Part 3 – Queries

Based on the Demo, we will implement the following functionality using Java and SQL with necessary GUI interfaces.

Trivial Queries:

1. List all the games under a particular genre.
2. List all the games published by a particular publisher.
3. List all matching games for a search term.
4. List a game's lowest price since release.
5. Get the current logged in user information along with first name, last name, email etc.

Non-trivial Queries:

1. List all the ancestral genres of a particular genre.
2. List a game's lowest prices from every week since release to the current week.
3. List a game's lowest prices from every month since release to the current month.
4. List all matching games for a search term and add-ons for those games with their lowest prices.
5. Get a list of all user email addresses that have subscribed to receive price change notifications for a particular discounted game.