Phase 3

Group Members

● Michael Dillahunty (mjdqw5)

● Justin Agudah (joagnp)

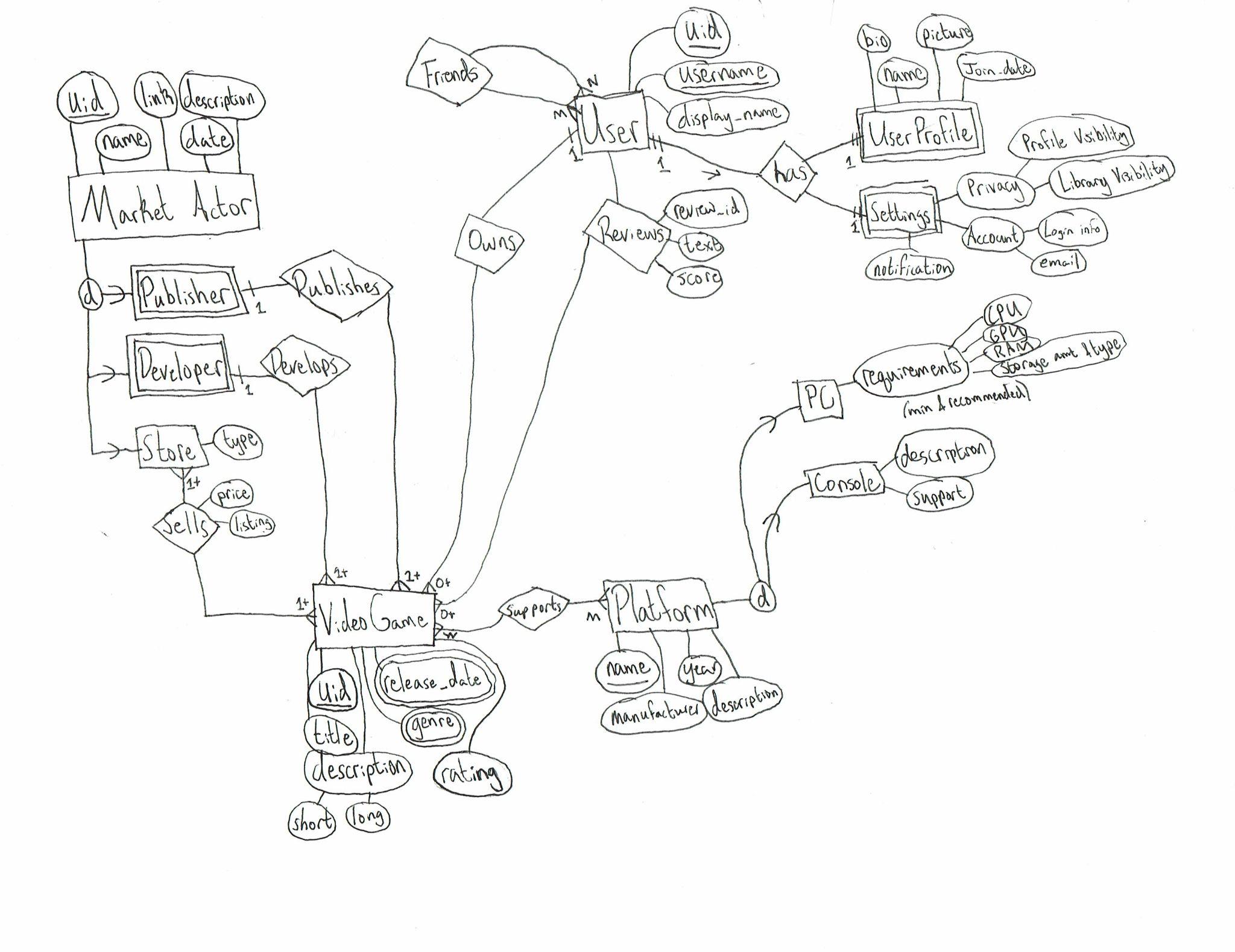
● Ryan Edwards (rje7hp)

## Problem Statement

The application we plan to build is a video game catalog for users to track and review games they own. There are so many platforms (Steam, Epic, Battle.Net, Windows Store, physical media, etc) that tracking your collection of games is often unwieldy. This application aims to cut down on the ‘digital sprawl’ created by online game retailers. The database is core to the functionality of this kind of service - users, games, and everything else will need clear models and relationships.

Our design will aim to catalog video games and serve as the definitive platform for users to track what games they own, have owned, or have played. It provides for reviewing games, searching for games by a number of attributes (genre, year, etc), and finding where the game is available, if it is. It could also be expanded to have basic user-driven marketplace functionality for physical game releases, such as most older games or special edition releases that people may sell as collectibles.

## Revised Conceptual Database Design



The site is primarily a catalog of games and so collects information relevant to them. The user owns a profile that can be seen publicly (depending on their settings) and settings which deal with the user’s interaction with the site.

## 

## Revised Application Program Design

**User Manual**

1. First, the user will be directed to the login page, where they’ll input their username and password to access their GameWorld account.

## 

1. Then the user will be directed to their own game library, where they can see all the games they’ve owned and reviewed.
2. Moreover, the user is able to modify/interact with their game library in various ways, such as inserting, removing, or modifying a review for a particular game, filtering the games by year (ascending), and much more.
3. Finally, when the user is done using their account, they can logout, redirecting them back to the login page.

## Entities

\* == required (optional if absent)

**Bold** == primary key

*Italics* == foreign key

* **VideoGame**

| **Key** | **Type** | **Meaning** |
| --- | --- | --- |
| release\_date | date | Date game released |
| \*title | varchar(50) | Game’s title |
| descr\_s | text | Short description of the game for a blurb |
| desc\_l | text | Long description for the store page |
| **\*id** | Bigint unsigned (Serial) | Serial ID of game |
| genre | varchar(50) | Genre tags for the game |
| rating | decimal(2, 1) | Game review score |

* **Requirement**

| **Key** | **Type** | **Meaning** |
| --- | --- | --- |
| game\_id | Bigint unsigned | Foreign key of VideoGame(id) |
| pc\_id | Bigint unsigned nullable | Foreign key of PC(id) |
| console\_id | Bigint unsigned nullable | Foreign key of Console(id) |

* **PC**

| **Key** | **Type** | **Meaning** |
| --- | --- | --- |
| **\*id** | Bigint unsigned (serial) | Unique id |
| op\_sys | varchar(100) | Operating system required |
| gpu\_min | varchar(50) | Min gpu |
| cpu\_min | varchar(50) | Min cpu |
| ram\_min | varchar(50) | Min ram |
| gpu\_max | varchar(50) | Recommended gpu |
| cpu\_max | varchar(50) | Rec. cpu |
| ram\_max | varchar(50) | Rec. ram |
| disk\_size | varchar(50) | Disk space usage |

* **Console**

| **Key** | **Type** | **Meaning** |
| --- | --- | --- |
| \***id** | Bigint unsigned (serial) | Unique id |
| release\_date | date | Release date |
| title | varchar(50) | Name of console |
| manufacturer | varchar(50) | Name of console maker |
| descr | varchar(250) | Short description |

* **User**

| **Key** | **Type** | **Meaning** |
| --- | --- | --- |
| **\*id** | Bigint unsigned (serial) | Unique ID of the user (serial) |
| \*username | varchar(50) | Name used to login |
| display\_name | varchar(50) | Name displayed on profile and to other users (defaults to username if not given) |
| join\_date | datetime | User join date |
| email | varchar(50) | Email used to register |

* **UserGameLibrary**

| **Key** | **Type** | **Meaning** |
| --- | --- | --- |
| user\_id | Bigint unsigned | Foreign key of User(id) |
| game\_id | Bigint unsigned | Foreign key of VideoGame(id) |

* **UserGameReview**

| **Key** | **Type** | **Meaning** |
| --- | --- | --- |
| user\_id | Bigint unsigned | Foreign key of User(id) |
| game\_id | Bigint unsigned | Foreign key of VideoGame(id) |
| review | Text(1000) | Review text |
| score | Tinyint unsigned <= 10 | Review score |

* **UserProfile**

| **Key** | **Type** | **Meaning** |
| --- | --- | --- |
| user\_id | Bigint unsigned | Foreign key of User(id) |
| realname | varchar(50) | User’s real name |
| realname\_visibility | Enum(‘public’, ‘friends’, ‘private’) | Visibility of this aspect |
| picture | BLOB | Picture (bytes object) |
| picture\_visibility | Enum(‘public’, ‘friends’, ‘private’) | Visibility of this aspect |

* **MarketActor**

| **Key** | **Type** | **Meaning** |
| --- | --- | --- |
| **\*id** | Serial (bigint unsigned) | Unique id |
| ma\_name | varchar(50) | The name of the market actor |
| link | varchar(100) | Url to the market actor |
| founded\_date | date | Date founded |
| descr | varchar(250) | Short description |

* **Publisher**

| **Key** | **Type** | **Meaning** |
| --- | --- | --- |
| ma\_id | Bigint unsigned | Foreign key of MarketActor(id) |
| game\_id | Bigint unsigned | Foreign key of VideoGame(id) |

* **Developer**

| **Key** | **Type** | **Meaning** |
| --- | --- | --- |
| ma\_id | Bigint unsigned | Foreign key of MarketActor(id) |
| game\_id | Bigint unsigned | Foreign key of VideoGame(id) |

* **StorePage**

| **Key** | **Type** | **Meaning** |
| --- | --- | --- |
| game\_id | Bigint unsigned | Foreign key of VideoGame(id) |
| price | decimal | Game’s price on this store |
| link | varchar(100) | Link to the store page |

## Entity relations

Platforms support Video Games N:M

Users can review Video Games N:M

A user can own Video Games 1:M

A user can friend other users N:M

A market actor acts on video games in some way.

Publishers publish them, 1:M

Developers develop them, 1:M

Stores sell them at a price; they maintain listings that our site might link to. N:M

The drawing specifies on each connection whether it is a 0, 1, or many to 1 or many.

## Functions

1. **User** adds a **game** to their library (could specify where/how they own it, etc)
2. **User** adds a review for a **game** (given they own it; needs a score and a text review)
3. **User** sets profile picture/bio/other profile details
4. Filter **games** by…
   * Developer
   * Publisher
   * Publish year
   * Medium (i.e. platform it is meant for, like PS4. Xbox, PC)
   * User rating (for example, seeing the highest rated games)
   * Seller type (physical store or online store)
   * Seller (ie filter to Steam alone, for example)
5. See **games** published each year filtered by genre

## Function Specifications

1. user\_add\_game:
   1. **Description**:
      1. This function allows a user to add a game to their library; it accesses the “Video Game” and “User” tables primarily.
   2. **Input**:
      1. VideoGame uid
   3. **Procedure**:
      1. User selects game to add
      2. If already present in library, do nothing
      3. Add the game UID to the user’s list of games.
2. user\_remove\_game:
   1. **Description**:
      1. This function allows a user to remove a game from their library; it accesses the “Video Game” and “User” tables primarily.
   2. **Input**:
      1. VideoGame uid
   3. **Procedure**:
      1. User selects game to remove
      2. If not in library, do nothing.
      3. Remove the game from the user’s list of games.
3. user\_review\_game:
   1. **Description**:
      1. This function allows a user to review a game from their library; it accesses the “Video Game” and “User” tables primarily.
   2. **Input**:
      1. VideoGame uid
      2. Score
      3. Review\_text
   3. **Procedure**:
      1. User writes review
      2. Create a new review entry:
         1. Generate a review UID.
         2. Apply user’s input
      3. Add review to review table.
4. filter:
   1. **Description**:
      1. Filtering can be done by a variety of aspects; filtering is self explanatory
   2. **Input**:
      1. Filter criterion
      2. Games table
   3. **Procedure**:
      1. Select games from the games table where the criterion is met.
5. See games by year:
   1. **Description**:
      1. This is not really an on demand function because it requires extensive preprocessing. The goal is to, for any genre, see how many games were published that year in a graph. This would be like rateyourmusic.com, for example [this page](https://rateyourmusic.com/genre/experimental-hip-hop/)’s graph.
   2. **Input**:
      1. Games table
      2. Genre criteria
   3. **Procedure**:
      1. Select release date from games
      2. For every entry increment a count of that year
      3. When new games are added increment the respective year
      4. Graph and display to user