**Lollipop Guild**

Ryan Gutierrez  
Zachary Gutierrez

GameFinder Report

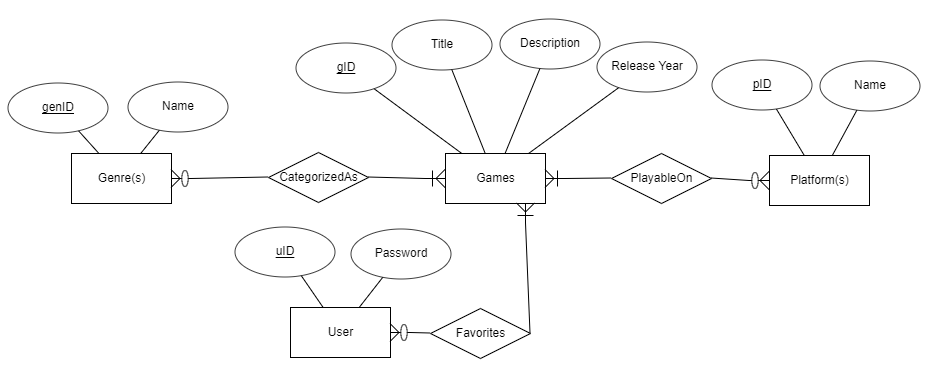
**Introduction**

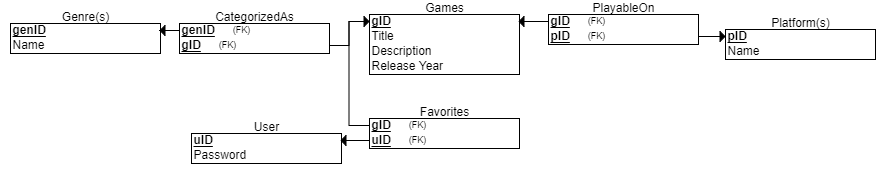
There are thousands of published video games in existence today and finding the one you want to play can be a long and tedious process. With that in mind, we created what we believe to be an adequate solution to that long and tedious process: GameFinder. The motivation in choosing the application for this project was largely our love of video games and a considerable amount of time spent on critically analyzing the possibilities in creating a useful database-oriented application with video games in mind. The result of this critical analysis was utilizing a database to hold a large number of published video games that a user could pore through to their hearts content. At the end of the day, the real purpose of GameFinder is to search for new games that a user might like to play, and so we set a goal in order to satisfy that ideal.

Our goal when designing GameFinder was to create the ability to search for different games using an array of parameters. The parameters that can be used to search within our application are the release year of the game, the genre of the game (i.e. Action Adventure or Role Playing Game), and the platforms that the game is available for (i.e. PlayStation 4 or Nintendo Switch) and using our tool, you will quickly get the results you want. The reason we chose these three parameters are as follows: searching based on the year that a game was released will allow a user to select games from any era they happen to favor (for example an older user searching for a video game they haven’t played before and want to try might want to search for video games that were released when they were a child); searching based on genre will allow the user to search for games of the same genre type of their favorite games—an obviously useful tool; searching based on the platform a game is available for will allow the user to search for video games they can play based on the platform(s) they own. As avid players of video games ourselves, we thought these parameters would be best to include in our application.

When designing GameFinder, we considered the design of applications that were similar to ours, to hopefully inspire us in certain ways. Unfortunately, to our knowledge, the only similar applications that we found were what we would call *BuzzFeed style quizzes* that will give you a recommendations on what genre of music to listen to or even what genre of game to play, but we could not find any fully fledged database-oriented applications modeled like ours; although, to be fair it might be ignorance that drove us to this conclusion.

**Database Details**

When designing our database, we first looked at all the parameters in order to see what could exist as an entity and what could exist as a relationship. Games, an obvious entity and the focal point of our application, would clearly hold certain data including the title of the game and the year it was released as there would be no point in keeping a table with listed years. We also decided to add a description of the game to better aid the user in their decision to play said game. Then we looked at Genre(s) and Platorm(s), both seemingly an attribute of Games; although, upon further inspection, we realized that since games could be categorized under multiple genres and played on multiple platforms, that it would be best to make them entities with a many-to-many relationship with Games. User would also be its own entity as you can see in our ER diagram. The only change that we made to our entity relationship during the build process was to the Favorites entity. In building our application, we realized that Favorites would function better as a many-to-many relationship between User and Games—as opposed to being the child of Games as shown in the diagram of our Stage 2 submission. Please note that the ER diagram shown here is an improvement on our initial concept with the modifications that we mentioned. This would allow us to simply create a Favorites table that held two foreign keys (user\_id and game\_id) to link the User and Games tables together. Please also note that to create our ER diagram, we used a web application called ERDPlus.

Fortunately, ERDPlus offered a function to translate an ER diagram into a relational model—which is the model you see here. Again, this relational model does reflect our final design as the translation was of the ER diagram you see above and not the same one we provided during Stage 3 of our project. We figured that it would be unnecessary to adjust the models to reflect our current design but decided to do so anyways to better illustrate the changes we made to our schema. Furthermore, if we reviewed the tables in our model correctly, then our schema in in 3NF.

**Functionality Details**

As far as basic functions go, we implemented the ability to insert records into the database through what shall henceforth be known as *admin input*: a fancy way of saying records were added through the terminal when our code was run using a special flag, “-adm.“ As mentioned in the Database Details section, when inserting records, the information included for a record (or Game) was Title, Description, Release Year, Genre(s), and Platform(s). Using *admin input* we are also able to update records and delete records. The rest of the basic functions within our application, namely searching the database and listing or printing returned results through various types of queries was implemented through the user GUI. This would be accessed by running the application normally which would create a window allowing a user to create a profile or login. Once logged in, the user could take advantage of the application to search for different games within the database by filling different fields used as search parameters. The user could also use the application to create a list of their favorite games or a Favorites list.

The Favorites list is what we designated as or advanced function. Although it is the only described advanced function that we implemented from our plans among the previous stages of the project, the login system also goes beyond the scope of basic functions. As such we have decided to also designate it as one of our advanced functions. To implement both the login system and the Favorites list, we simply made a user table that held the username—also the primary key—and password for each user, and many-to-many relationship between User and Games: a table designated Favorites which just held two foreign keys.

**Implementation Details**

When planning out our application, we considered many different languages and platforms but ultimately decided to use Python, alongside MySQL. The reason we chose Python was because it is an extremely simple scripting language that we are familiar with; in other words, we could get a lot more done in a shorter amount of time than if we used other non-scripting languages. Fortunately, we were also already familiar with tkinter: the GUI library we used to create the front-end interface of GameFinder. Our application logic was created on the back-end using two different Python scripts—one that handled the created of the database and its tables, and another that handled the application logic and the interaction with MySQL. The reason we used MySQL was because of its apparent popularity when searching for what platform we wanted to use. Its popularity was a good indication that there would be plenty of resources on its use: a tremendous benefit to us, who were inexperienced with using SQL.

**Experiences**

Through this project, we have genuinely gained some invaluable experience in the world of databases. Prior to taking this class we were horribly unaware of how deep-seated databases are in the applications we use every day. Apart from the eye-opening experiences that the class and this project were, we learned how to apply our knowledge of sequel and got a considerable amount of practice using Python to interact with other platforms. Luckily, we didn’t face very many difficult problems during this project and whenever we did, we had the resources to solve our issues (the resources we used are the class lecture slides as well as some of the references). In thinking of how to extend our project to more advanced, mature systems in the future, we thought it would be best to use a remote database as opposed to the local one we used for our application. In order to possibly extend our project in the future we would also like to make GameFinder a web application as opposed to its current state as a desktop application. As far as enhancements to the application itself goes, we might like to add a recommendations feature that would recommend games to users based on the games they have stored in their favorites list. This could be done using the parameters we already have (such as recommending games based on the genres of their favorited games) or maybe some other more complicated formula that takes more parameters into account.

**References**

https://erdplus.com/

https://www.youtube.com/watch?v=x7SwgcpACng&list=PLB5jA40tNf3tRMbTpBA0N7lfDZNLZAa9G

http://www.mysqltutorial.org/

https://github.com/pyrokuna/Databases\_FinalProject.git