Final Report - Deliverable #5
Team Name: Five Star
Team Members: Kayla Bachler, Alex Golovin, Linda (Kyla) NeSmith, & Chris Otey

Proposal

Business Objective We're Trying To Solve

Video game players need help finding video games to play. Our application will provide players the ability to view video games, compare available prices, read reviews, filter video games by genre, and even add video games they are interested into a wishlist.

What Kind Of Data Will The Data Hold

Each video game will have a title; release date; age suggestion (using PEGI); a brief summary of the game (optional); a unique identifier ASINfor the video game; and system requirements such as storage space, memory, OS, graphics, and processor. We will also have a genre entity that has a unique genre type. For pricing, we will have outside sites which each have their own game price, site name, and URL (unique). There will also be a developer with a composite developer name that comes from the studio and unique company name. A user that consists of: a unique username, unique email address, password, name (first and last), DOB, derived age, and profile creation date. There will also be the option of reviews written by users for video games that will have a rating (from 1 to 5) and optional comments. Finally, we will have a wishlist that the user can create that holds multiple different games they would like to purchase.

What Kinds Of Queries We Plan To Answer

- Which video games have the highest rating? Worst rating?
- How many video games are there of each genre?
- Who is the developer of each video game?
- How many video games are created by a given developer?
- When was a video game released?
- What video games were released in a given year?
- How many video games did a single developer make?
- Which video games work on a Mac OS X platform?
- Which video games take up the most storage space?
- How many video games are rated PEGI 18?

Anticipated Schedule

Item	**Delivery Date**
Proposal	Tuesday, January 16th
ER and Rational Design	Tuesday, January 30th
Tool Assessment	Saturday, February 10th
Database Queries & Commands	Thursday, February 22th
Project Submission	Tuesday, March 13th

Project Presentation	Thursday, March 15th

Main and Sub Deliverables

1. **Design Document ER and Schema**
   - Create an ER diagram for our database
   - Complete the Relation data model
   - Verify the relations, attributes, primary keys, foreign keys and entities as a team
   - Write introductions, descriptions, and explanations for the diagrams
   - Discuss design decisions, concerns, assumptions and any limitations
2. **Tooling Document and Presentation**
   - Determine the tools that we plan to use for our application
   - Present our project to the class
3. **Database Queries & Commands**
   - Draft a report showing the SQL commands we will use to populate our test database
   - Populate the database and report the tables
   - Finalize our test plan
4. **Testing Complete - statement documentation**
   - Create a document for our projects SQL statements and a description of the purpose of each statement
   - Identify our testing strategy
   - Identify which test cases we used
   - Determine how we plan on doing regression testing in the future
5. **Final Presentation**
   - Create poster for final project
   - Create a final document and present our project to the class

Individual Roles

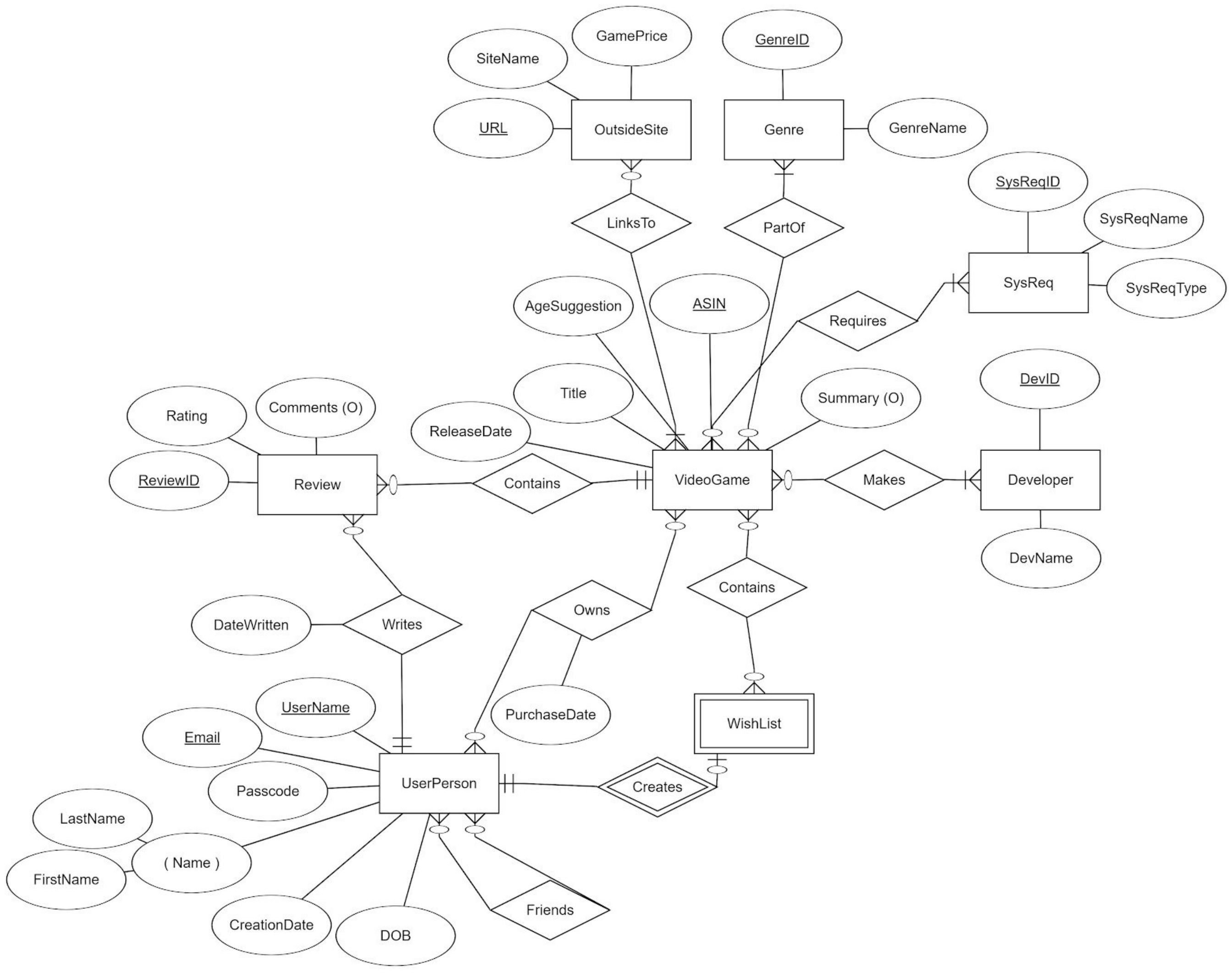
1. Kayla Bachler - **Verifier** confirms the completeness of all deliverables before submission and delegates tasks within the team
2. Alex Golovin - **Head Jedi Master.** “Learn, I do not. Keep a balance in the force, I must.” (OK, maybe in a different dimension. Here on Earth, I motivate & keep us focused.)
3. Linda (Kyla) NeSmith - **Council Spectre**. I gather up people for a meeting while the galaxy falls apart (or you could just call me a coordinator)
4. Chris Otey - **Monitor Evaluator** will evaluate and analyse the proposals, ideas, and contributions of others in the team

Design Documents

Introduction

Below are the Entity-Relationship (ER) Diagram and Relational Data Model for our project. The ER Diagram displays the relationships between different entities we will be using such as VideoGame and UserPerson having a relationship of a UserPerson owning a VideoGame. The Relational Data Model shows the data that will be stored for each entity as well as join tables. It also displays where foreign keys are used. Afterwards, we give descriptions, explanations, decisions, assumptions, and limitations relating to the diagrams.

Entity-Relation Diagram



Relational Data Model

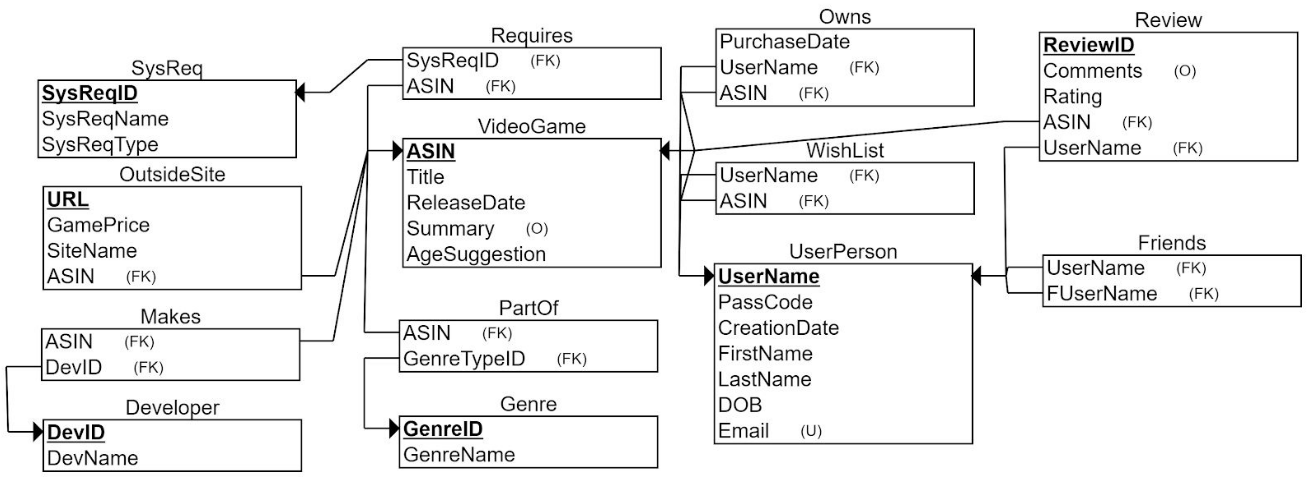


Diagram Description & Explanations

- Developer
    - Description: an entity containing information relation to VideoGame developers
    - Attribute: developer ID (primary key), developer name (unique)
    - Relationships: Each Developer makes one to many VideoGames
    - Reasoning: UserPersons can search for VideoGames made by the same Developer
- Genre
    - Description: an entity to hold a genre type
    - Attributes: genre Id (primary key), genre name (unique)
    - Relationships: Each Genre has optional none to many VideoGames
    - Reasoning: UserPerson can find VideoGames based on a Genre and hopefully prevents UserPerson from not finding a VideoGame due to a typo of which Genre the VideoGame belongs to
- OutsideSite
    - Description: an entity containing information from websites where we pull VideoGame prices from
    - Attributes: URL (primary key), site name, game price, game's ID
    - Relationships: A VideoGame can be found on multiple OutsideSites
    - Reasoning: Having multiple different prices will help the UserPerson get the best price for a VideoGame
- Review
    - Description: an entity that contains a UserPerson's review of a VideoGame
    - Attributes: review ID (primary key), rating, optional comments, game's ID, reviewer's username
    - Relationships: Each Review is written by only one UserPerson (has attribute of date written), contains one VideoGame

    - Reasoning: UserPersons have the option to write Reviews for VideoGames, UserPersons can read feedback about a VideoGame before choosing to purchase it, thus helping them select a VideoGame that interests them
- SysReq
    - Description: an entity that contains a system requirements of different types
    - Attributes: system requirement ID (primary key), system requirement name (unique), system requirement type
    - Relationships: VideoGames have one to many system requirements
    - Reasoning: Multiple games may need the same system requirements, UserPerson may want to find a game by a system requirement type
- UserPerson
    - Description: an entity that contains information about a user
    - Attributes: username (primary key), email address (unique), password, name (composite consisting of first and last name), creation date for the UserPerson's account, DOB
    - Relationships: Each UserPerson friends optional none to many other UserPersons (has attribute of date added friend), owns optional none to many VideoGames (has attribute of purchase date), creates optional one WishList, writes optional none to many Reviews (has attribute of date written)
    - Reasoning: Allows a person to have records such as which VideoGames they have purchased or put in a WishList, allows person to write Reviews of VideoGames
- VideoGame
    - Description: an entity containing information on a video game
    - Attributes: ASIN (primary key), title, release date, age suggestion, optional summary
    - Relationships: Each VideoGame entity is linked to optional none to many OutsideSites, has at least one Genre, made by at least one Developer, contains optional none to many Reviews, contained on optional none to many WishLists, owned by optional none to many UserPersons (has attribute of purchased date)
    - Reasoning: Having these features about the VideoGame will help the UserPerson determine if they are interested in purchasing the VideoGame
- WishList
    - Description: a weak entity that holds a list of VideoGames created by a UserPerson
    - Attributes: none
    - Relationships: Each WishList is created by only one UserPerson (identifying), contains none to many VideoGames
    - Reasoning: Allows a UserPerson to create a list of VideoGames which they may not wish to purchase immediately but may wish to return to later

Design Decisions, Assumptions, & Limitations

- We have decided to gather our VideoGame pricing from OutsideSites to compare prices. This allows the UserPerson to view the lowest available price for the VideoGame.
- Overall Review ratings can be derived from averaging all of a VideoGame's individual Reviews' ratings.
- We assume there can be one to many Developers for a VideoGame.
- A UserPerson's username and email address is unique to them.
- A UserPerson can search for VideoGames made by a specific Developer.
- UserPersons can have friends which are other UserPersons.
- UserPersons do not always want to purchase a VideoGame immediately. For that reason, we allow UserPersons to make a WishList.
- Company names are unique and cannot have duplicates.
- Reviews have a review ID as a unique attribute.
- We are assuming that our UserPersons have a variety of gaming platforms, therefore, we included a system requirements attribute in the VideoGame entity.
- We are assuming that a VideoGame has at least one Genre, but a specific Genre may not have a VideoGame that is stored in our database.
- When a UserPerson chooses to rate a VideoGame, we assume that they will give the VideoGame at least a rating but not always will they write a Review.
- In the relationship called owns between UserPerson and VideoGame, we assume that the UserPerson will provide us with a date of purchase or it will be automatically uploaded when they purchase a VideoGame through us.
- If a VideoGame has no release date, it has not yet been released and/or the release date is unknown.
- For an OutsideSite, if the price of a VideoGame is negative, it has not been released and if it is null or 0, the VideoGame is free.

Tooling Assessment

Development Tools

We intend to use MySql for our database development because MySql is a popular database tool with a lot of libraries to integrate with Java. As a team we have novice experience using MySql, so we feel that working with MySql will be a challenge to each of us.

User Interface Tools

The software we will use for developing our UI in web will be Java and JDBC. We chose Java and JDBC because each member of the team is familiar with programming in Java. We also plan on building a GUI so Java would be a good choice for that over C++. Our team will be using JDBC since Professor Chenault advised us in class to not use anything else but JDBC when working with Java.

Database Hosting

We are going to host our database and allow users access through Microsoft's Azure cloud computing services. We chose Azure because it has a large number of helpful guides and a diverse range of services provided as well as it is provided free to us through the ImagineX project at UW.

Database Queries & Commands

Queries and Their Business Need

Below is a list of the top 10 queries and commands that we anticipate needing in order to achieve the business goal of helping users of our application find video games to play. Each statement below has its own business need listed that contributes to the overall business goal.

1. **Query:** What are the different Genre types available to the UserPersons to search by?
   Business Need: Help the user see the available genre types for the video games within our database.
2. **Query:** What are the comments for VideoGames with 5 star ratings?
   Business Need: Show the comments on reviews with 5 stars so they can potentially purchase a video game that other players enjoy.
3. **Query:** Which VideoGames are compatible with the UserPerson's system?
   Business Need: Help the user view the video games that are compatible with their PC so they don't have to weed through games they may not be able to play.
4. **Query:** Who are the Friends of the UserPerson?
   Business Need: Show the user their friends/friend-list.
5. **Query:** What VideoGames the company X made?
   Business Need: Give user a list of video games that a company they like has made so they can play other video games by them.
6. **Query:** Which VideoGames are on the UserPerson's Wishlist?
   Business Need: Show the user the video games that they have on their wishlist so they can potentially purchase a video game from their wishlist.
7. **Query:** Which VideoGames would be considered age appropriate for the UserPerson?
   Business Need: Show the user only video games that are considered age approriate for them based on their date of birth and the video game age suggestion.
8. **Query:** Which Outside Site has the lowest price for the VideoGame X?
   Business Need: Show the user the lowest price available for the video game they want in order to give them the best deal.
9. **Query:** What are the top 3 rated VideoGames?
   Business Need: Show the user the top 3/most popular video games right now to potentially purchase for playing.
10. **Query:** How many VideoGames are in each Genre?

Business Need: Give user the ability to see how many video games are within each available genre so they can see which genres are the most popular, and how many video games they can view of a specific genre.

SQL Commands

Below are the 10 queries and commands written in SQL. Some queries have been written with specific use cases for testing purposes.

```
-- What are the different Genres available to the User to search by?
SELECT GenreName
FROM Genre
ORDER BY GenreName ASC;

-- What are the comments for VideoGames with 5 star ratings?
SELECT v.Title, r.Rating, IFNULL(r.Comments, "No comment")
FROM VideoGame v, Review r
WHERE v.ASIN = r.ASIN and
        r.Rating > 4;

-- Which VideoGames are compatible with Mac OSX?
SELECT v.Title, s.SysReqName
FROM VideoGame v, SysReq s, Requires r
WHERE r.ASIN = v.ASIN and
        r.SysReqID = s.SysReqID and
        s.SysReqName LIKE "Mac OSX%";

-- Who are the Friends of the UserPerson CmndrShepard?
SELECT f.FUserName AS FriendList
FROM Friends f, UserPerson u
WHERE u.Username = f.Username and
        u.Username LIKE "CmdrShepard"
ORDER BY f.FUserName ASC;

-- What VideoGames has Developer Bioware made?
SELECT v.Title, d.DevName
FROM VideoGame v, Developer d, Makes m
WHERE m.ASIN = v.ASIN and
        m.DevID = d.DevID and
        d.DevName LIKE "Bioware"
ORDER BY v.Title ASC;

-- Which VideoGames are on the CmdrShepard's Wishlist?
SELECT v.Title
```

```
FROM UserPerson u, WishList w, VideoGame v
WHERE w.Username = u.Username and
        u.Username LIKE "cmdrshepard" and
        v.ASIN = w.ASIN
ORDER BY v.Title;
```

-- Which VideoGames would be considered age appropriate for UserPerson CmndrShepard?

```
SELECT DISTINCT(v.Title)
FROM VideoGame v, UserPerson u
WHERE u.Username LIKE "CmdrShepard" and
        v.AgeSuggestion IS NOT NULL and
        v.AgeSuggestion <= TIMESTAMPDIFF(YEAR, u.DOB, CURDATE())
ORDER BY v.Title ASC;
```

-- Which Outside Site has the lowest price for the VideoGame Stardew Valley?

```
SELECT o.SiteName, MIN(o.GamePrice) AS Price
FROM OutsideSite o, VideoGame v
WHERE o.ASIN = v.ASIN and
        o.GamePrice IS NOT NULL and
        v.Title LIKE "Tiny Bubbles"
GROUP BY o.GamePrice;
```

-- What are the top 3 rated VideoGames?

```
SELECT v.Title, AVG(r.Rating) AS Rating
FROM VideoGame v, Review r
WHERE v.ASIN = r.ASIN
GROUP BY v.Title
ORDER BY r.Rating DESC LIMIT 3;
```

-- How many VideoGames are in each Genre?

```
SELECT g.GenreName, COUNT(v.ASIN) AS NumOfGenre
FROM VideoGame v, Genre g, PartOf p
WHERE g.GenreTypeID = p.GenreID and
        v.ASIN = p.ASIN
GROUP BY g.GenreID;
```

Final Project

SQL Statements for Project

Below are the exact SQL statements that we used for our project in order to get result sets from our database to display on the GUI. Since the GUI is programmed to handle a lot of the comparisons for the below SQL statements, we did not need to write as many select statements as we originally anticipated. Instead, we just used '?' to store the user-selected variables for searching within our tables.

SQL Statement	**Purpose**

SELECT v.ASIN
FROM dbo.VideoGame v, dbo.PartOf p, dbo.Genre g
WHERE v.ASIN = p.ASIN and
p.GenreTypeID = g.GenreTypeID and
g.GenreName LIKE ?;
	Grabs the ASIN (video game unique identifier) from the Genre table so that the GUI can connect the video game to the selected Genre. Allows the user to search by genre.

SELECT v.ASIN
FROM dbo.VideoGame v, dbo.Developer d, dbo.Makes m
WHERE d.DevID = m.DevID and
m.ASIN = v.ASIN and
d.DevName LIKE ?;
	Used to in the GUI to connect the Developer with the VideoGame that the user selects (which is the '?'). This will allow us to display the developer name of the video game on the results page.

SELECT v.ASIN
FROM dbo.VideoGame v, dbo.outsidesite os
WHERE v.ASIN = os.ASIN and
os.SiteName LIKE ?;
	Used to in the GUI to connect the OutSide site name with the VideoGame that the user selects (which is the '?'). This will allow us to display the outside sites that sell the video game on the results page.

SELECT r.Rating, r.Comments
FROM dbo.Review r
WHERE r.asin LIKE ?;
	Gets the reviews for the video game that the user selects (which is the '?'). The reviews are displayed on the video game's description page.

SELECT v.ASIN, v.Title, v.Agesuggestion, v.ReleaseDate,v.Summary
FROM dbo.VideoGame v
WHERE v.Title LIKE ?;
	Used to retrieve the information of a selected video game for the video game's description page. This statement is used when the user chooses a video game title (which is the '?') from a list.

SELECT os.url, os.SiteName, os.GamePrice
FROM dbo.OutsideSite os
WHERE os.ASIN LIKE ?;
	Gets the outside site for the video game that the user selects (which is the '?'). The outside site's are displayed on the video game's description page to allow the user to purchase the game from an outside site and view the price.

Additional SQL Statements

Below are additional queries that would work with our GUI if we had more time to build a user window for the user profile to display the user's friends list, and expand our current application by adding a wishlist linked to each userprofile. We also would also allow the user to see which games are compatible with their operating system in order to use all the tables within our database.

SELECT u.Username
FROM dbo.UserPerson u
WHERE u.Username LIKE ?;
	Allows for the user to search for another user of our application based on their username.

SELECT v.Title, s.SysReqName
FROM dbo.VideoGame v, dbo.SysReq s, dbo.Requires r
WHERE r.ASIN = v.ASIN and
r.SysReqID = s.SysReqID and
s.SysReqName LIKE ?;
	Shows the user a list of video games that are compatible with the system that they play on. The user enters the operating system that they use, and we pass that into the query in place of the ?.

SELECT v.Title
FROM dbo.UserPerson u, dbo.WishList w, dbo.VideoGame v
WHERE w.Username = u.Username and
u.Username LIKE ? and
v.ASIN = w.ASIN;
	Displays the video games that the user currently has on their wishlist. This query would require us to make the GUI able to authenticate the user and build a seperate page for the user profile which we ran out of time for.

SELECT f.FUserName
FROM dbo.Friends f, dbo.UserPerson u
WHERE u.Username = f.Username and
u.Username LIKE ?;
	Used for displaying the users friends list. We did not get to implement this, but the user would authenticate by logging in and the friends list would be displayed.

Technologies Used

For our project we chose to build a GUI using JDBC-Java and SSMS (Microsoft SQL Server Management Studio) with our video game database populated on Azure. The GUI is an application written in Java that utilizes JDBC. We chose to go with have because it is amongst the most commonly used programming languages for applications. Java Database Connectivity (JDBC) API is used as the industry standard for connecting a Java application to a database. The databases that the JDBC API is capable to connecting is not limited to SQL databases, but can be used with a range of other database.

We used the SSMS to connect to our Azure hosted database in part because both are made by Microsoft. SSMS provided an interface that made it easier to:

- Test our queries
- View all the tables in the DB
- View the constraints that each table had
- View the attributes and attribute types associated with each table
- View any triggers, indexes, keys associated with each table

The primary use of the SSMS was for testing our queries as it took longer to load the Java GUI and debug it.

The Azure system handles all of the data itself and hosts the database in the cloud. This means that the database can be accessed from whatever ports are allowed by the servers firewall settings. Azure allows us to create a virtual server located anywhere in the world and create databases on those servers. This would allow further expansion and ease of use if we wanted to implement a vertical or horizontal distribution for our information. We would consider, if this project got larger and involved more countries, applying a vertical distribution for all the base client handling and locate multiple servers in different countries with information distributed to all of them. Azure also allows us to do app hosting if we desire which means we can implement the GUI into the cloud for a web application down the line.

Project Recap

The project brought new challenges to some members of the team with a novice background in databases. Overall our team had little experience with database technologies so a lot of effort was put into selecting technologies that would work well together, and designing the GUI to work with the database. After spending a lot of time testing our database for the videogame finder, we created an original GUI that works with our database using Java. The database was successfully populated with both pretend data and real data from online, and tested using various queries. We also tested our application manually with correct and incorrect user input to see how it would respond. The application will perform as expected with correct user input, and will display an empty list for incorrect user input. Due to time constraints we were unable to build special test data or an error message display, but we would definitely do so if given another week.

Although our GUI design does not offer as many options to the user as we would like, it does provide the user with videogames to play and the ability to search for video games. We were happy that we built a functional GUI using Java, and that we were able to make queries on our database. However, since we had to learn how to use Azure and JDBC for this project we ran into a problem. The team struggled with making our design work with all the queries we originally intended to use, and due to time constraints we chose to prioritize the queries to satisfy the business problem. We ended up prioritizing the GUI to be able to display all video game titles, search by genre, search by developer, view video game ratings and available prices for the video games. If we had another week, we would implement the queries above that we needed to build additional windows for, and expand our GUI both in design and functionality to offer more video game options for the user.

Database and Code submitted separately