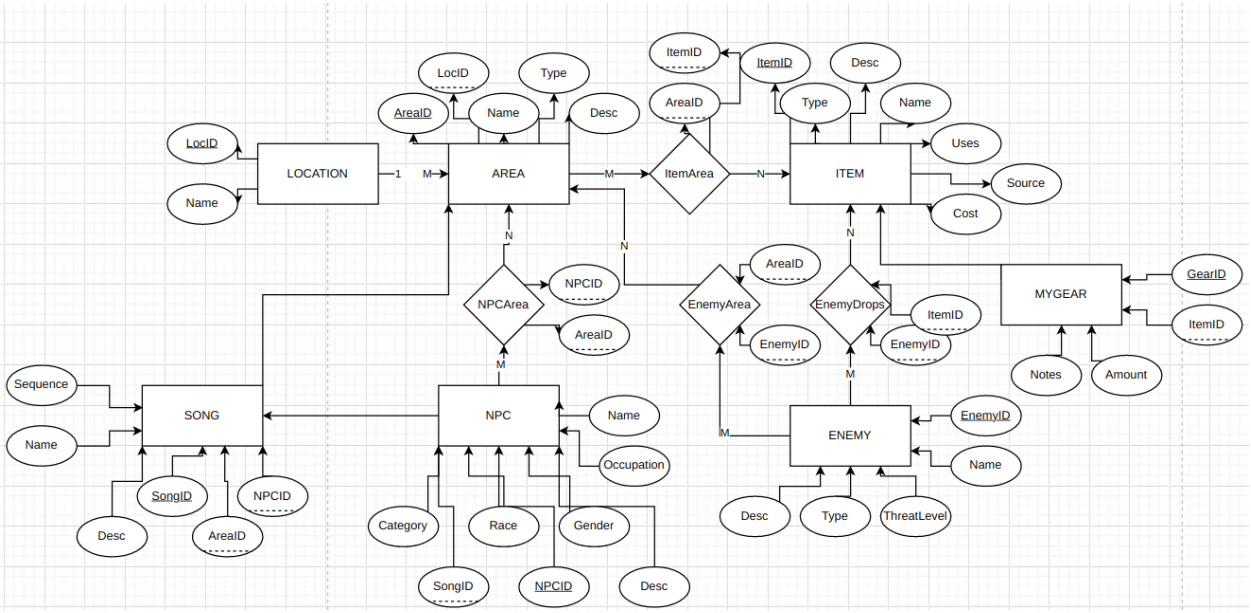
Daniel Weninger
CS 364 Project

The Legend of Zelda: Ocarina of Time Video Game NPC and Item Locator and Guide
Synopsis

This will be an item, NPC, song, enemy, and area guide to the game The Legend of Zelda: Ocarina of Time. Users will be able to see the location of items, enemies, NPCs and songs, as well as give additional information about each such as the cost of an item.
Tables



Location

- This will be a bigger, overarching area on the map including towns, fields, etc.
- One location can have many areas or dungeons.

Area/Dungeon

- Areas are more refined places within a location such as important houses, shops, or dungeons.
- As this will be a guide/locator app, I will allow the user to be able to filter items,

NPC

- Users will be able to search by NPC like they would with items and it will show where they are, and if they give a song or item. The user will then get information about the NPC such as what their occupation is.

Enemy

- I will allow the user to search by enemy and show where the enemy is and what the reward for defeating them is. The user will then get information about the enemy such as how challenging it is, what kind of enemy it is, what items/weapons can defeat it, and what the rewards for defeating it are.

Song

- Songs and items are similar in the sense where users will be able to search by item and see where to find them within the game and if an NPC teaches the song to them. The user will then get information about what the song does.

MyGear

- My gear is a table that users will be able to add, update, and delete from to keep track of the items that they currently have. They will add/update/delete items from the items table by typing in the item's name.

Functionality

- Users are able to search through each of the Location, Area, Enemy, Item, and Song tables by name
- Users are able to search for NPCs by location
- Users are able to search for enemy by area
- Users are able to search for areas that have the area's average threat level
- Users are able to add items from the items table to their MyGear list
- Users are able to update the items on their MyGear list by the item's notes and amount
- Users are able to delete items on their MyGear list

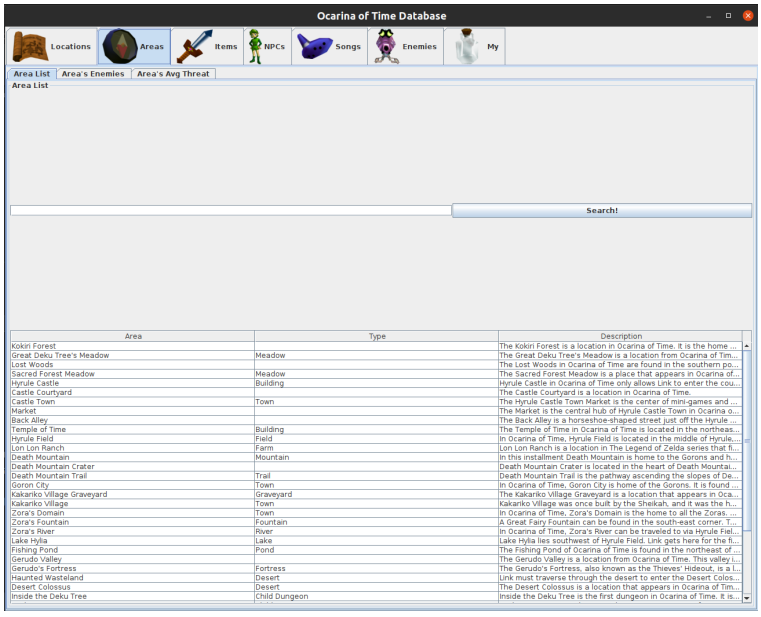
Demographic

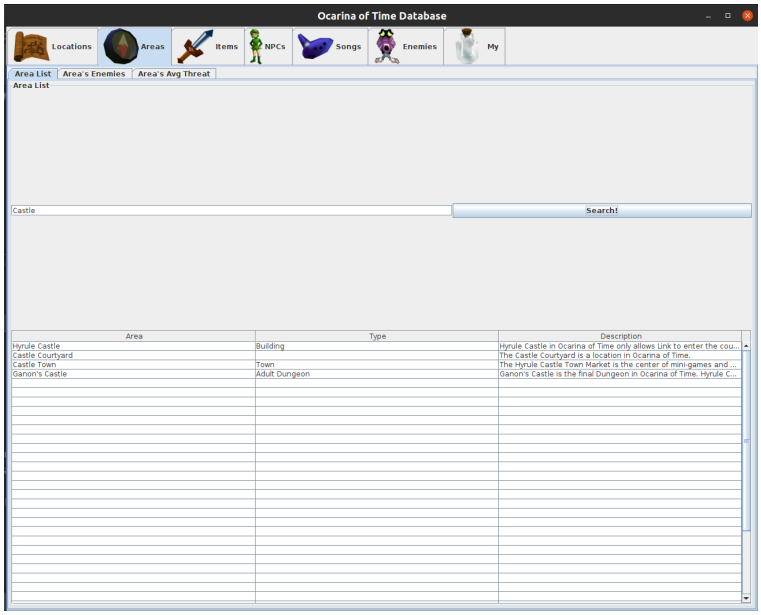
Anyone that is playing or looking to play this game would find this app helpful. Different uses might be for 100%ing the game or even using it as a semi-guide.

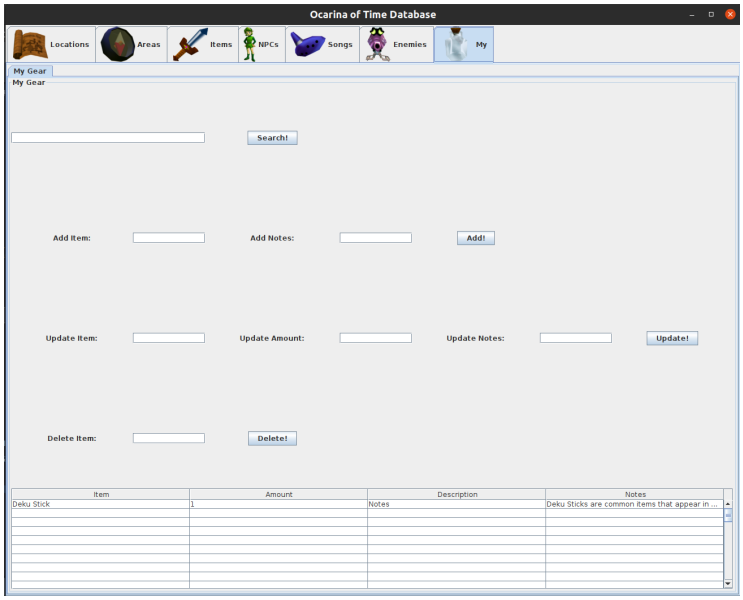
Technical Requirements

I made a desktop application using Java and SQLite. I also used GitHub for version control.

Screenshots







Advanced Queries

Group 1)

```
SELECT Area.Name, count(Enemy.EnemyId) AS EnemyCount
                        FROM Area JOIN EnemyArea JOIN Enemy
                        ON Area.AreaId = EnemyArea.AreaId AND EnemyArea.EnemyId
= Enemy.EnemyId
                        GROUP BY Area.AreaId
                        ORDER BY EnemyCount DESC
```

Finds the amount of enemies in each area and orders them from most to least enemies.

Group 2)

```
SELECT Location.Name AS Loc, NPC.Name AS N
                        FROM Location JOIN Area JOIN NPCArea JOIN NPC
                        ON Location.LocationId = Area.LocationId AND Area.AreaId =
NPCArea.AreaId AND NPCArea.NPCId = NPC.NPCId
                        WHERE NPC.Name LIKE '%<nameVar>%'
                        ORDER BY Location.LocationId
```

Finds the locations that each NPC is in.

Group 3)

```
SELECT ItemCounts.Area AS Area, avg(Item.Cost) As  ItemCost
        FROM (SELECT Area.Name AS Area, count(Item.Name) AS  ItemCount
                FROM Item JOIN ItemArea JOIN Area
                ON Item.ItemId = ItemArea.ItemID AND ItemArea.AreaId = Area.AreaId
                GROUP BY Area
                HAVING ItemCount > 2
                ORDER BY ItemCount DESC) AS ItemCounts
        JOIN Area JOIN ItemArea JOIN Item
        ON ItemCounts.Area = Area.Name AND Area.AreaId = ItemArea.AreaId  AND
        ItemArea.ItemId = Item.ItemId
        GROUP BY ItemCounts.Area
        HAVING ItemCost > 0
```

Of the areas that have more than two items in it (item count > 2) find the average price of the items in each of those areas as long as the average item cost is greater than 0.