CIS560 Project Proposal
Sarah Diener and Nicholas Sixbury

Project Summary

-We will be creating a web application that allows for random generation of objects appropriate for loot in a tabletop roleplaying game. The intended users would be people playing tabletop roleplaying games, particularly GURPS, the Generic Universal RolePlaying System. The user will be able to click a button in order to generate loot, and they'll also be able to use a variety of filters to control what objects can be generated. Users will be able to save their loot by copying and pasting information out of the page or by saving a file. Each item will have more or less concrete statistics that describe it and can also be affected by modifiers such as quality, enchantment, embellishments, or decorations.

Users will also be able to create their own custom tables and add, update, or delete items in them. They can set their custom-created tables as private or public and save them as physical files. Any user who is logged in can see and access any public table, but they can only edit the tables that they are credited as creating. Users who don't log in will only be able to generate from default tables, but users who log in can access all the other features of the site.

There will be several pages on the website, listed below:

- Page to view and use only the default tables, available without logging in
- Page to view all public user-created tables, and import them for use
- Page to view imported and public tables, and see what all the items in them are
- Page to edit tables that you have access to edit
- Page to view statistics on various things
- Login page
- Signup page

Technical Details

-We plan to use Javascript (React) for the frontend, webapp part and c# for the backend. We'll use Visual Studio for c# and database integration, SQL Management Studio for database design, and VS Code for javascript coding.

Data Interactions

-each of the tables below lists operations which ***are*** supported for that table

Item: SELECT | INSERT | UPDATE

ItemSubcategory: SELECT | INSERT | UPDATE

ItemTypeOption: SELECT | INSERT | UPDATE

ItemCategory: SELECT | INSERT | UPDATE

User: SELECT | INSERT

EmbellishmentCategory: SELECT | INSERT | UPDATE
Embellishment: SELECT | INSERT | UPDATE
EnchantmentCategory: SELECT | INSERT | UPDATE
Enchantment: SELECT | INSERT | UPDATE
InventoryItem: SELECT | INSERT | UPDATE | DELETE
EmbellishmentRef: SELECT | INSERT | UPDATE | DELETE
CreatedEmbellishment: SELECT | INSERT | UPDATE | DELETE
EnchantmentRef: SELECT | INSERT | UPDATE | DELETE
CreatedEnchantment: SELECT | INSERT | UPDATE | DELETE

Report Queries

1*:

Description: For each user, show number of tables they've created, number of items they've created, number of categories generated from, number of items generated, and the day they joined

Input: User

ResultColumns: UserID, Name, TablesCreated, ItemsCreated, TablesUsed, ItemsGenerated, JoinedOn

2*:

Description: For all items in a particular table, show each combination of variations of single embellishments and enchantments from selected embellishment and enchantment categories, including a final cost and weight for each item, also including category of each item, embellishment, and enchantment in the row they belong in.

Input: ItemCategory, EnchantmentCategory, EmbellishmentCategory

ResultColumns: ItemID, Name, CategoryID, CategoryName, EnchantmentID, EnchantmentName, EmbellishmentID, EmbellishmentName, Cost, Weight

3*:

Description: Statistics information for a particular category, including the creating user, number of items in the table, average cost and weight of items in the table, total cost and weight of all items in the table, createdOn date, sorted by least recent creation

Input: ItemCategory

ResultColumns: CategoryID, CreatorName, AvgCost, AvgWeight, TotalCost, TotalWeight, CreatedOn

4*:

Description: List all items that have been generated, including information on most and least recent generation date, cost and weight of each item, the category each item is from, the name of the item, and number of users which have generated them. Items are ordered descending by the number of users who have that item in their inventory.

Input: No input needed

ResultColumns: ItemName, EarliestGeneration, LatestGeneration, ItemCategoryName, Cost, Weight, NumberGenerated

Initial Database Design

-link to access version of database diagram that is current as of writing is right here. Note: Several tables, such as Item, InventoryItem, or CreatedEnchantment very intentionally do not have unique constraints on things like the name. The purpose of this for the Item table is that the Item table does not have a path back to ItemCategory or ItemSubcategory, so if the Name was constrained, then you could have a situation where two users building their own list of items would be unable to name an item the same thing. This would be problematic, and as such, it makes more sense for Items to be able to have the same name. In the case of InventoryItem or CreatedEnchantment, they need to not have the unique constraint on Name for some of the same reasons, but also because the InventoryItem table is specifically designed to allow entries with the same name but different IDs, such that Enchantment of Embellishment references can link to one instance of an item in this table without linking to all instances. So for instance, if you have 7 wooden tables, and you want to say that 1 of them is embellished with decorations carved into it, you could have 1 entry of a wooden table which is linked by an embellishment ref, while the other six are not.