Guildwars 2 gold webapp

Purpose:

This project allows the user to decide how to best use their time to gain the most gold while playing Guildwars 2. The goal is to allow players to spend the most amount of time playing the game the way they want to while being able to have the gold they need to buy whatever items/skins/gems/etc. they need or want.

Django

djangorestframework

djoser

djangorestframework\_simplejwt

mysqlclient

requests

How to use the project:

-------------------------------

After starting the app welcome message is displayed

User is given options to select from:

1. User setup
2. Manually update salvage data
3. Calculate actual profit of salvage process
4. Calculate potential profit
5. Item crafting profit

User setup:

* Needs user API for inventory, materials, bank, etc. to calculate salvage rate, calculate optimal items to be crafted, etc.

1. User inputs
   * Username
   * Password
   * Email
   * API key
2. Use Djoser base endpoints to handle user account creation and authorization

Manually update salvage data:

* Salvage data can be updated manually or automatically when user calculates actual profit

1. User inputs rarity of unid gear, quantity of unid gear, and raw materials salvaged
2. App updates user\_salvage\_rates and returns new rates

Calculate potential profit:

* Calculates user potential profit for salvaging any number of unid gear.

1. User selects “calculate potential profit”
2. Check user API, if user does not have salvage rate data then app can’t calculate potential profit
3. User selects rarity of unid gear, inputs quantity they wish to process, and inputs buy price (default is current buy price)
4. App returns estimation of raw materials gained and profit vs loss of buying Unid gear and salvaging into raw materials vs. buying the equivalent raw materials
   1. Estimates are calculated by salvage rates associated with the user

Calculate Actual profit:

* Calculates user profit from buying and salvaging unid gear

1. User selects “calculate actual profit”
2. Check user API
3. User initiates calculation of actual profit
   1. User initiates data collection call
4. User opens and salvages unid gear with following salvage kits
   1. Blue gear: Copper-Fed Salvage-o-Matic
   2. Green gear: Runecrafter's Salvage-o-Matic
   3. Yellow gear: Silver-Fed Salvage-o-Matic
   4. Orange gear: sell on trading post
   5. Future function: allow user to select or determine which kit they are salvaging the unid gear with
5. User completes salvaging unid gear
   1. User clicks “end” which calls the data collection function
6. App calculates difference between initial data collection call and final data collection call
   1. Returns raw materials gained and orange gear
      1. Stored in appropriate data model
   2. Compares buying unid gear and salvaging vs. buying the equivalent raw materials
      1. Returns
         1. Profit if materials are sold = Difference between cost of unid and salvaging vs **selling** equivalent raw materials
         2. Profit if materials are used for crafting = Difference between cost of unid and salvaging vs **buying** equivalent raw materials
            1. Is this unrealized gains/profit?
         3. Discount = (cost of buying the equivalent raw materials / Cost of buying unid gear and salvaging)
      2. Cost of buying unid gear and salvaging = (quantity of unid gear \* buy price) + (cost of salvaging) - (revenue from selling orange gear - taxes/fees)
      3. If cost of buying unid gear and salvaging is less than buying the equivalent raw materials then that can be considered a profit
   3. Call update salvage rate function
      1. Uses previous calculated data as input

----------------------------------------------

[GW2 API wiki](https://wiki.guildwars2.com/wiki/API:Main)

----------------------------------------------

Datasets & Data<https://wiki.guildwars2.com/wiki/API:Main> models:

1. CustomUser(AbstractUser)
   1. Populated by user input
   2. API key can be blank

Saving salvage records in 2 data models vs 1

* Decrease repeat storage of salvaged\_item\_count
* Allows deletion of data connected to record number

1. User\_Salvage\_Records
   1. Populated by data processing call
2. User\_Outcome\_Records
   1. Populated by data processing call
3. User\_Salvage\_Rates
   1. Calculated by data processing call
   2. Stores salvage rate for a given item for each salvage input

Notes:

* Eliminated the User\_salvage\_results data model because it was redundant and would cause extra calls to be performed to calculate the salvage rate

\*\*\*See Diagram\*\*\*

Views/Viewsets:

1. Class CustomUser\_List\_ViewSet(generics.ListAPIView)
   1. GET (Requires admin permissions)
      1. Allows admin to view list of all users
2. Def GET\_User\_Raw\_Data\_View(request)
   1. This function pulls data from User\_Bank\_Data, User\_Inventory\_Data, User\_Shared\_Inventory\_Data, User\_Materials\_Data, User\_Wallet\_Data GW2 APIs, parses item id and quantity, saves the data in a dictionary and stores it as a JSON object in a local directory. This function will need to be updated to save data to a noSQL database during production and only saves data to the local directory for development/testing currently.
   2. Users data will be partitioned into files under their username and record number (one for one with record number in User\_salvage\_records)
      1. /test\_data/<username>/<record\_number>/{Initial\_recording|Final\_recording}.json
3. Def POST\_User\_Salvage\_Outcome\_Data\_View(request):
   1. This view retrieves JSON object from /test\_data/<username>/<record\_number>/{Initial\_recording|Final\_recording}.json and creates a new user\_salvage\_record object and user\_outcome\_data object. The record number is automatically determined from the User\_salvage\_record model. The salvaged\_item\_id is determined by which unid had the largest difference in count.
4. Def Manual\_User\_Salvage\_Outcome\_Data\_View(request):
   1. This function allows a user to manual submit data to create a new user\_salvage\_record object and user\_outcome\_data object. The record number is automatically determined from the User\_salvage\_record model. The salvaged\_item\_id is determined by which unid had the largest difference in count.
5. Class User\_Salvage\_Record\_View(viewsets.Modelview)
   1. This function allows the user to view, partial update, and delete objects.
      1. Queryset should be limited to request.user only
   2. GET
      1. This function returns salvaged\_date, salvaged\_item\_id, and salvaged\_item\_count for specified record number.
   3. PATCH
      1. This function is used to update existing objects
   4. Delete
      1. This function deletes specified user\_salvage\_record object and associated user\_outcome\_data objects
6. Class User\_Outcome\_Data\_View(viewsets.Modelview)
   1. This function allows user to view and partial update objects.
      1. Queryset should be limited to current user.
   2. GET
      1. This function returns gained\_item\_id, and gained\_item\_count for specified record number
   3. PATCH
      1. This function is used to update existing objects
7. Def POST\_User\_Salvage\_Rate\_View(request)
   1. This function updates user\_salvage\_rates for all unid gear for the user. This function will be used after a record is created, deleted, or updated.
   2. Create a dictionary to hold record numbers by unid gear. While creating the dictionary calculate the total number of each unid as well and store in variables
      1. {85016: [], 84731:[], 83008:[]}
   3. Create a dictionary to hold outcome data for each gained\_item\_id. The dictionary will hold the volume of the item gained from a specific unid gear.
      1. {gained\_item\_id:{85016: count, 84731:count, 83008:count}
   4. Use update\_or\_create() to update user\_salvage\_rates
   5. Compare gained\_item\_id in user\_salvage\_rates and user\_outcome\_data to find any objects in user\_salvage\_rates that should be deleted due to either item deletion from the game, correction for manual input error, etc.
8. Calculate actual salvage profit
   1. This function allows the user to calculate the profit from buying unid gear, opening, and salvaging.
   2. This function will return, Initial investment, price of materials if bought from the TP, revenue earned if all materials sold on the TP minus fees
   3. Calculate initial investment
      1. Count of unid x TP buy price
         1. Count of unid is pulled from user\_salvage\_records
         2. Unid TP buy price is an input from the user, defaults to current buy price which is pulled from GW2 TP api
   4. Calculate cost of gained raw materials
      1. Count of each item x TP buy price
         1. Count of each item is pulled from user\_outcome\_data
         2. TP buy price is pulled from GW2 TP api
   5. Calculate gross revenue of gained raw materials
      1. Count of each item x TP Sell price x 0.85 (for tax/fees)
         1. Count of each item is pulled from user\_outcome\_data
         2. TP sell price is pulled from trading\_post\_data
      2. If gross revenue > cost of unid then there was a net profit
9. Calculate potential salvage profit

* This function allows the user to calculate the profit from buying unid gear, opening, and salvaging. Requires input of quantity and type of unid
* This function will return 1.) the total cost processing, 2.) price of materials if bought from the TP, 3.) revenue earned if all materials sold on the TP minus fees
  1. Calculate initial investment
     1. Count of unid x TP buy price
        1. Count of unid is pulled from the request data
        2. Unid TP buy price is an input from the user, defaults to current buy price
  2. Calculate cost of gained raw materials
     1. Count of each item x TP buy price
        1. Count of each item calculated by multiplying the corresponding user salvage rate with the count of unid input by the user
        2. TP buy price is pulled from GW2\_Trading\_Post\_data
     2. If cost of gained raw materials > cost of unid then there was a discount on raw materials
  3. Calculate gross revenue of gained raw materials
     1. Count of each item x TP Sell price x 0.85 (for tax/fees)
        1. Count of each item is calculated above
        2. TP sell price is pulled from trading\_post\_data
     2. If gross revenue > cost of unid then there was a net profit

Appendix:

--------------

Unid gear - Unidentified gear