Eve Online requirements

Within eve online there is the ability to manufacture capital ships, and we need to use different technologies to allow us to manage this

Considerations:

Ensure all chosen operating systems for within docker and docker itself contain as much detail as possible in each step, noting that the people developing it are new to the software,

All operating systems should be built with AI fully and scripted to complete where possible using POWERSHELL and docker commands, and instructions on how to execute the scripts within windows 11

If files need to be constructed, please supply me a script that will make the files and the contents, including all variables, please output these scripts separately, with specific instructions on how to use them

The location for all files and scripts to be downloaded and used are E:\DockerProject folder,

IDE being used will be VS Code, consider this with instructions to perform the duties in your answers:

Docker Requirements

* Mariadb to be used as the database
* There should be a generated web front end with hooks to all data built in, and the ability to call sql commands and store the results are important
* Include logging and monitoring for all of these and a document which explains the use
* Do not assume any knowledge of docker is present, therefore all work that can be created by you as an AI, should be

OAuth2:

* has to be set up for SSO for the web front end,
* this should be done in the most complete manner,
* using tokens that automatically refresh, store, request the scopes from the game eve online as required, opposed to 1 at a time,
* Oauth should be constructed using a script with all the values in it, please supply this Error handling should be present for all eventualities

Eve Swagger Interface:

* The ability to make calls as required, should be added to the web front end, and store the variables as used in a map, array or table that can reused

Worker agents:

* Any large data set pulls should be done by worker agents, and should be scalable, both authomatically and manually from within the web interface
* This would be most ideally made using ephemeral workers, that will output data to the databases
* Set these to scale up and down automatically depending on the size of the pulls
* Workers should be able to multithread
* Workers should power off when their role is finished
* Workers are required to keep logging output to a central file within the database
* Error handling should be present for all eventualities

Scheduled repeated tasks:

* ESI has allowances for APIs, which we cannot surpass, however, we should be utilising the shortest possible update lengths for any repeated pulls, e.g. corporation manufacturing jobs may allow 2 minutely API pulls, and Character Assets may allow for 15 minute API pulls, these should be set to these lowest values every time

I need all code to be checked and validated using multiple sources before being presented to me to ensure it works, no unvalidated, or unverified code, if there is something you are unable to check, or requires confirmation from me, this should be explained with guidance of how to do it

Application stack should be built using Docker Desktop,

The front end should be a fully integrated web interface, perhaps PHP,

Database is MariaDB, there should be a separate page/interface linked to the main site for administrators that are able to see the data stored easily, and query and perform calculations against it to gather information such as, total build job materials required, skills required to build the build job, time taken to build the item, assets in systems for corpation required to build it

Corporation Blueprints:

All corporation blueprints need to be stored and looked up with their own database, this information needs to be updated as often as possible with locationIDs we enter in to the web front end,

This also needs to be accessible for humans, therefore the array constructed will need a search option, it will need to be filterable, it will need a good user interface,

Following the skeleton being created, the purpose of this local web app is to:

Monitor capital ship production

Show each step of the capital ship production and the order in which each step and build job is required

Reference corporation assets in specific citadels, if assets are not present, then a look up for the blueprint to construct the missing component needs to be shown, the simplest way to do this, would be something like:

Capital ships build page, showing each capital ship we want to order/build or corp in a given solar system (make this a variable we can add to, change, etc, as the locations may me several places), allowing for locationIds to be added adhoc

Capital Components build page, showing each capital component we need to build, that is in build (and deduct it from the required materials list, in order of which capital ship was ordered first), in a given solar system (make this a variable we can add to, change, etc, as the locations may me several places), allowing for locationIds to be added adhoc

Reactions build page, this will show each missing reaction which is required for each of the generated build list from the previous steps, using the blueprints in the corp own in a given solar system (make this a variable we can add to, change, etc, as the locations may me several places), allowing for locationIds to be added adhoc

All other components/items that may be missing from corp assets, or not in the active building list, should be laid out as above, to ensure we build everything that has a blueprint, and use the blueprints the corp uses

Current locationIds for corporation assets:

|  |
| --- |
| 1046366846654 |
| 1046366846654 |
| 1046366846654 |
| 1046366846654 |
| 1046366846654 |
| 1046366846654 |
| 1046366846654 |
| 1046366846654 |
| 1046366846654 |
| 1046366846654 |
| 1046366846654 |
| 1046366846654 |
| 1046366847541 |
| 1046366847541 |
| 1046366847541 |
| 1046366847541 |
| 1046366847541 |
| 1046366847541 |
| 1046366847541 |
| 1046366847541 |
| 1046366847541 |
| 1046366847051 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043416658531 |
| 1043416658531 |
| 1043416658531 |
| 1043416658531 |
| 1043416658531 |
| 1043416658531 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1043438107602 |
| 1038822509801 |
| 1043416705594 |
| 1043416705594 |
| 1038822509801 |
| 183005955 |
| 1022031082428 |
| 1022031065784 |
| 1022031057513 |
| 183005955 |
| 183005955 |
| 1023442177185 |
| 1023442186916 |
| 1023442189186 |
| 1023442178657 |
| 1023442182391 |
| 1023442187888 |
| 1023442186916 |
| 1023442177185 |
| 1023442184945 |
| 1023442176142 |
| 1038822509801 |
| 1040002700259 |
| 1022031082428 |
| 1022031069253 |
| 1022031074134 |
| 1022030993593 |
| 1039029841461 |
| 1043416658531 |
| 1038551791239 |
| 1021575654966 |
| 1038602901539 |
| 1038551791239 |
| 1043416705594 |
| 1043293464865 |
| 1022031061195 |
| 183005955 |
| 183005955 |
| 1043293464865 |
| 1043416705594 |
| 1043416705594 |
| 1040002689476 |
| 1040002688245 |
| 1040002686492 |
| 1040002685116 |
| 1022031059123 |
| 1022031061195 |
| 1022031061195 |
| 1022031078446 |
| 1022031075451 |
| 1022031084444 |
| 1022031086080 |
| 1022031082428 |
| 1022031065784 |
| 1022031064676 |
| 1043416705594 |
| 1043293464865 |
| 1040002700887 |
| 1038603171141 |
| 1022031055762 |
| 1038822509801 |
| 183005955 |
| 183005955 |
| 1022031186249 |
| 1024787997376 |
| 1038822509801 |
| 1038822509801 |
| 1043416705594 |
| 1043416705594 |
| 1043416705594 |
| 1043416705594 |
| 183005955 |
| 1038822509801 |
| 1038400465103 |
| 1022031068276 |
| 1022031072017 |
| 1022031082428 |
| 1022031059123 |
| 1022031054195 |
| 1038400445179 |
| 1039347182172 |
| 1038400466192 |
| 183005955 |
| 1023442190165 |
| 1023442181256 |
| 1023442179970 |
| 1023442183774 |
| 1043293464865 |
| 1043293464865 |

Other important information for the generating of the application:

Store static data:

InventoryIDs are used often, to pull names of items, this should be pulled and stored in the application, updated once a month with only values which have changed

BlueprintIDs are used in most of these calculation, these need to be stored with the different types, and as we need this information to be portrayed to the users, they should be in an interface which allows the sorting, filtering, looking up for individual values in the arrays, allowing the application to the heavy lifting, - should be searchable too

CharacterIDs are used in production Jobs and should be stored when looked up, with the string name fo the users, create a similar table to BluePrintIDs

LocationIDs this should be a table and for those locations that do not have a string name defined, we need the ability to directly add a name to this that can be referenced anywhere

Here are the details needed for Oath2, use these with your generated steps:

**Client ID:**

523dcd00cdfe40e8b11a455adf02fc65

**Secret Key:**

ZbQiT940vfmxTun4riqZjGNw2imohWgiehn5cEaB

**Callback URL:**

http://localhost/callback

CharacterID if required: 1862958351

CorporationID 146531499