Evan Wang and Stephan Manley
CMPT 307
IGUN Database Final Report
12.10.15

[Note: The database doesn't require a specific password to log in, only access to the database, which I don't think I can change.]

Project Summary:

IGUN DB is a database for the Intergalactic United Nations (IGUN). The Intergalactic United Nations was founded in the year 20XX by the Terran Race after they ascended from their homeworld Earth and spread into the stars. It is a peace-keeping organization that spans galaxies and territories. It exists solely to prevent war and the perpetuation of atrocious crimes against sentience throughout the known universe. However, being a more-or-less legislative organization, they suffer from the typical ailments of ruling bodies. Namely: they are *completely and utterly* ***incompetent***. Especially in regards to managing data pertaining to the known races of the universe. In other words, they can't do anything that they want to do because they can't keep track of that they need to keep track of in order to anything.

The purpose IGUN DB is to solve the Intergalactic United Nations problems, by helping them keep track of all of the known races in the universe. IGUN DB records general information about the known races of the galaxy. This general data includes language, homeworld, leader, population and species name. This data, however, is not enough to be of use to the IGUN, since they need to know more than general information. For this reason, IGUN DB also records for separate categories for each known race: Territorial information, military information, diplomatic information and faction information. Each of these four categories contain much more information relative to that respective category. This provides the IGUN with a wealth of information about each race in an easily browsable format. Since the IGUN is largely concerned with individual races, IGUN DB has races as the central point of its design. Most of the other data (except faction data) is recorded in relation race (For example, races have planets, but the IGUN does not worry about unclaimed planets). The reason that faction is not "race-centric" is because factions are the only data recorded that 'have' races. Furthermore, one faction can have multiple races as members, so making faction race-centric doesn't make as much sense as the otherway around. When the user selects one of the four categories, there are more specific subcategories on the next screen displayed. As the user continues to choose more specific categories, they go down one branch of the database (which can be seen clearly in the ER diagram shown in a few sections.) until they reach the most specific data (such as vehicles, under the military branch). This way, the user can browse through broad categories and find out more about any specific information that they want.

Designing The Database:

In order to design this database, we first started by deciding on the data that was important to the IGUN. The IGUN, of course, knows nothing about databases. They only know what they wanted to be able to do with the database.

Er Diagram:

(Diagram is also attached if it is hard to read)

Table Descriptions:

Below is a list of tables that appear in IGUN DB, along with a short description.

Tables_in_IGUN
Admiral
Captain
FacConditions
Faction
FlShips
Flcraft
Flground
MilCom
MilFleet
Military
Moon
Planet
Race
SancConditions
Sanction
SanctionEnforcer
SanctionTarget
Sov
SpaceStation
StarSys
Trade
TradePartners
TradeRoutes
Vehicle

Admiral: In charge of a fleet. Each fleet has one.
Captain: In charge of a ship. Each ship has one.
FacConditions: Conditions for membership in a faction. A faction can have many.
Faction: A group of races.
FlShips: Ships in a fleet. A fleet can have many.
Flcraft: Flight craft in a fleet. A fleet can have many.
Flground: Ground vehicles in a fleet. A fleet can have many.
MilCom: Commander of a military. Each military has one.
MilFleet: A fleet in a military. A military can have many.
Military: A race's military. A race has one.
Moon: A planet's moon. A planet can have many.
Planet: A planet in a star system. A star system can have many.
Race: A known intelligent race.
SancConditions: Conditions for the imposition of a sanction. A sanction can have many
Sanction: A restriction meant to restrain a dangerous race.
SanctionEnforcer: The races that enforce a certain sanction.
SanctionTarget: The races that are the targets of a certain sanction.
Sov: A race's sovereignty. A race can have one.
SpaceStation: A race's space stations. A race can have many.
StarSys: A sovereignty's star system. A sovereignty can have many.
Trade: A race's trade. A race can have many.
TradePartners: A trade's partners. A trade can have many.
TradeRoutes: Routes that a particular trade takes. A trade can have many.
Vehicle: A vehicle in a fleet. This is a generic way of representing FlShips, Flground and Flcraft.

Schemas:

Below is a list of schemas used for the tables in our database. Foreign keys are indicated as FK and primary keys are underlined.

Race{RaceId, Name, Description, Species, GalPop, Homeworld, Language, leader, FactionsId(FK:Factions.FactionId)

Factions{FactionId,FactionName, UnMember}

FacConditions{ConditionsId,FactionsId(FK:Faction.FactionId),Rule}

Sov{SovId,Name,RaceId(FK:Race.RaceId)}

StarSys{StarSysId, PlanetNum, Resources, SovId(FK:Sov.SovId), Population, Diameter}

Planet{PlanetId,Size,Density,Mass,Name,Type,Atmosphere,Gravity,Colonized,LocLanguage,Population,StarSysId(FK:StarSys.StarSysId}

Moon{MoonId,Name,Size,Density,Mass,Type,AtmoMakeUp,Gravity,Inhabit,Population,PlanetId(FK:Planet.PlanetId)}

SpaceStation(SpaceStationId,Use,Population,ShipCap,FlatCraftCap,IndustrialCap,RaceId(FK:Race.RaceId)}

Trade{TradeId,TradeProfit,Imports,Exports,RaceId(FK:Race.RaceId)}

TradeRoutes{TradeRoutesId,Route,TradeId(FK:Trade.TradeId)}

TradePartner{TradeId(FK:Trade.TradeId),RaceId(FK:Race.RaceId)}

Sanction{SanctionId,SanctionName,EstabDate}

SancConditions{SancConditionsId,Rule,SanctionId(FK:Sanctions.SanctionId)}

SanctionEnforcer{RaceId(FK:Race.RaceId),Sanctions(FK:Sanction.SanctionId)}

SanctionTarget{SanctionsId(FK:Sanction.SanctionId),RaceId(FK:Race.RaceId)}

Military{MilitaryId,FleetSize,GroundForceSize,MilitaryCol,RaceId(FK:Race.RaceId)}

MilCom{MilComId,ComFirstName,ComLastName,MilitaryId(FK:Military.MilitaryId)}

MilFleet{MilFleetId,FleetName,NumShips,NumPersonnel,MilitaryId(FK:Military.MilitaryId),AdmiralId(FK:Admiral.AdmiralId)}

Admiral{AdmiralId,AdFirstName,AdLastName}

Vehicle{VehicleId,MilFleetId(FK:MilFleet.MilFleetId),Speed,CrewCap}

FlShips{FlShipId,ShipName,Classification,Tier,Size,Armament,CaptainId(FK:Captain.CaptainId)

FlCraft(FlCraftId,Type,Size,Armament, VehicleId(FK:Vehicle.VehicleId)}

FlGround{FlGroundId,Structure,MainWeapon, VehicleId(FK:Vehicle.VehicleId)}

Captain{CaptionId,CapFirstName,CapLastName, VehicleId(FK:Vehicle.VehicleId)}

Query Examples:

Some important queries and their results are shown below:
The condition variables that begin with $'s are variables stored on the current page.

Query in sanctionEnforcer.php:

This query displays all of the races that enforce a particular sanction (identified by $sanc)

SELECT Name FROM Race INNER JOIN SanctionEnforcer USING(RaceId) WHERE SanctionId = $sanc;

Query in sanctionTarget.php:

This query displays all of the races targeted by a participate sanction (identified by $sanc)

SELECT Name FROM SanctionEnforcer INNER JOIN Race USING(RaceId) WHERE SanctionId = $sanc;

Query in raceMain.php:

This query displays all information relevant to a given race (identified by $Id)

SELECT * FROM Race INNER JOIN Faction USING(FactionId) WHERE RaceId = $Id;

Query in ship.php:

This query displays all information about a given ship in a given military fleet (identified by $ShipId)

SELECT * FROM FlShips INNER JOIN Captain USING(FlShipsId) INNER JOIN Vehicle USING(VehicleId) WHERE FlShipsId = $shipId;

The data from the Flground and Flcraft (Fleet ground vehicles and fleet flight crafts) is fetched with a very similar query.

Of course, all of the queries are important, as the client wants to be able to access all of the data they have stored in IGUN DB. These ones listed here, however, give a good sense of the how the data is typically retrieved. Most of the variables denoted with $'s are received from the previous page, where the user would select which object they want to view next and then click submit. One of the queries would then fetch the data pertaining to the object they choose, matching it based on a variable.

Designing The Interface:

The interface that we designed is structured to show information pertaining to the selected race on each screen with options to view more specific data about whichever branch you are in. You can also return to the main screen or to the previous page. This allows for browsing down and then jumping back up again, similar to a file folder structure in a computer os file browser.

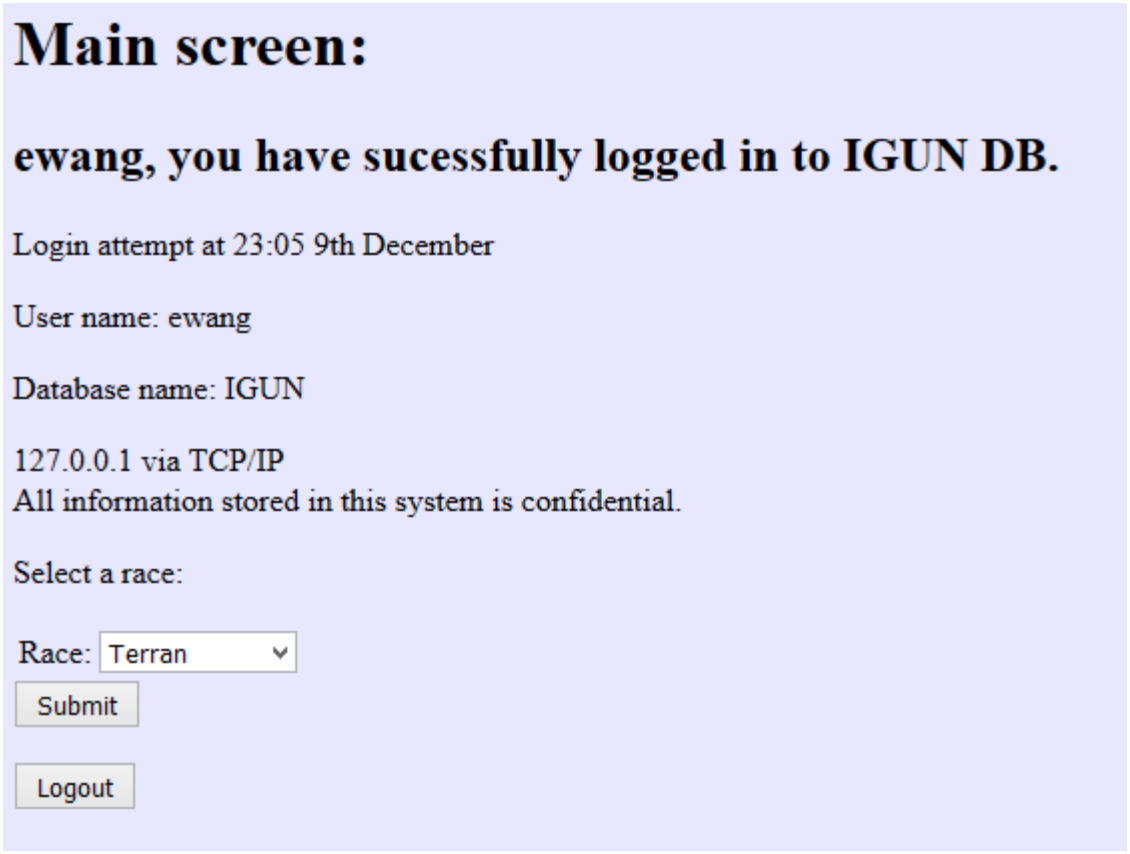
Each page displays information relevant only to that level of specificity. The data displayed is typically made of queries executed on anywhere from one (the fewest amount) to three (the largest amount) tables. All of the pages display their data in a table, with each row showing a different data item pertaining to whatever the screen is supposed to be displaying. This makes it easy and straightforward. The identifier (whether it be name or numeric ID) of whatever you are viewing is displayed at the top of the screen. For example, if you are viewing the Terran race's star system identified as "2", the top of the star system screen will read as "Star System 2". This was done to make it easy to go to a star system (or something else, such as a planet) and see all of the important and relevant data associated with it. The user doesn't have to know anything about that star system and try to search for it, since the purpose of the IGUN DB is to keep track of things. They can simply navigate to the Terran's "territory" branch and see which star system they hold sovereignty over.

All user input and navigation, except on the login screen, is done through drop down lists and buttons. This allows the user to have no knowledge of the inner workings of the system as well as no knowledge of the data stored with in. They can browse it easily and quickly.

Screenshots And Explanations:

For the users convenience, shown below are screenshots of each page in IGUN DB and a short explanation of each. Each image below is a separate page navigated to through buttons from the previous page.

Login Screen:



Display the information of the race selected on the previous page. Notice there are four buttons on the bottom labeled territory, diplomacy, military and faction. These buttons each lead to one of the branches described above.

Race Description:

Race:	Terran
Species:	Homosapien
Leader:	President Michael Jackson
Language:	Terran
Faction Name:	Triumvirate
Galactic Population:	4550000*10^3
Homeworld:	Earth

One of the three most powerful races in the galaxy, the Terran Race are masters of technology, and perseverance. They have spread into the stars for the sake of exploring the unknown.

Territory | Diplomacy | Military | Faction

Back

The territory branch:

Territory Information:

Terran

Terran holds sovereignty over Space America

Space America Contains 700 planets

Space America has a total population of 1289000*10^3

Select a star system for more information: 2 Submit

Space America has 2 space stations:

Space station 2 is a Mil station. Population: 40000. Ship capacity: 2000. Flight craft capacity: 9300. Industrial capacity?: Y

Space station 3 is a Mil station. Population: 120. Ship capacity: 10. Flight craft capacity: 2900. Industrial capacity?: N

Back | Return to Main

Explore a star system. Notice the drop down on the previous page shows 2, but we chose 9 instead:

Explore Star System 9

Diameter: 10 (au)

Number of Planets: 2

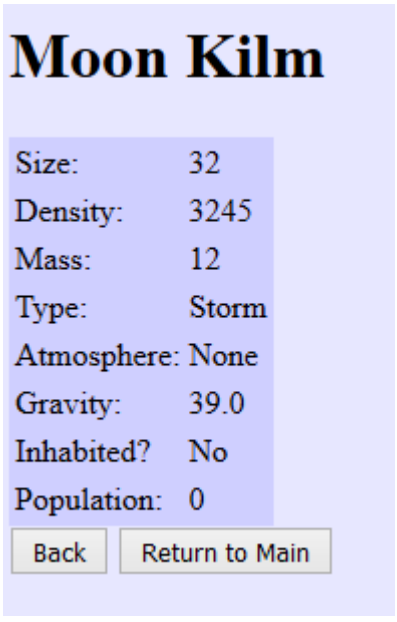
Population: 854000

Planets: Beltrag explore planet

Back | Return to Main

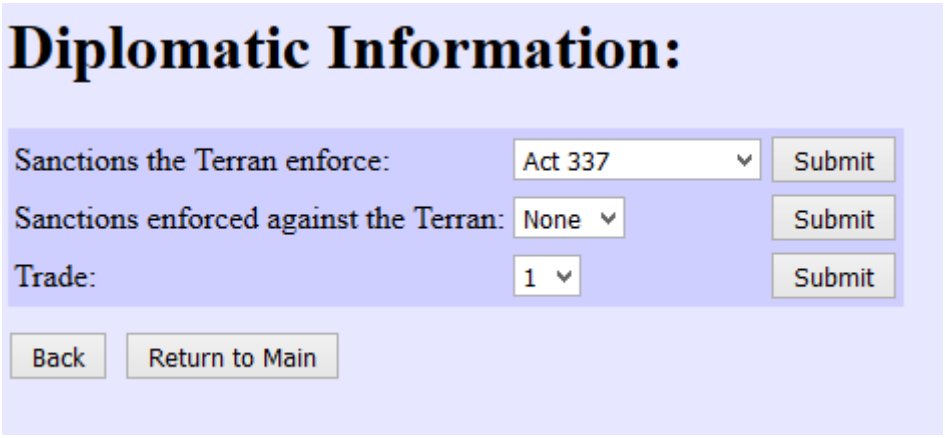
Explore the planet from the previous page:

Explore selected moon:

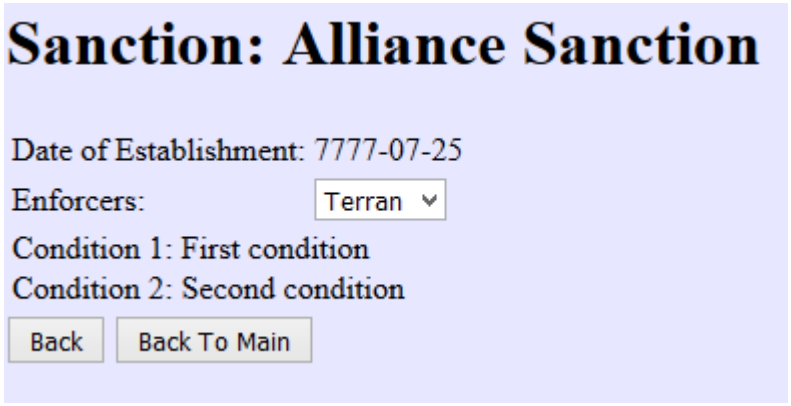


The diplomacy branch:

The diplomatic branch is not deep, extending only one page down with three sub-branches shown below.



This page shows the details of a sanction enforced by the Terran. All other partner enforcer are shown in the drop down:



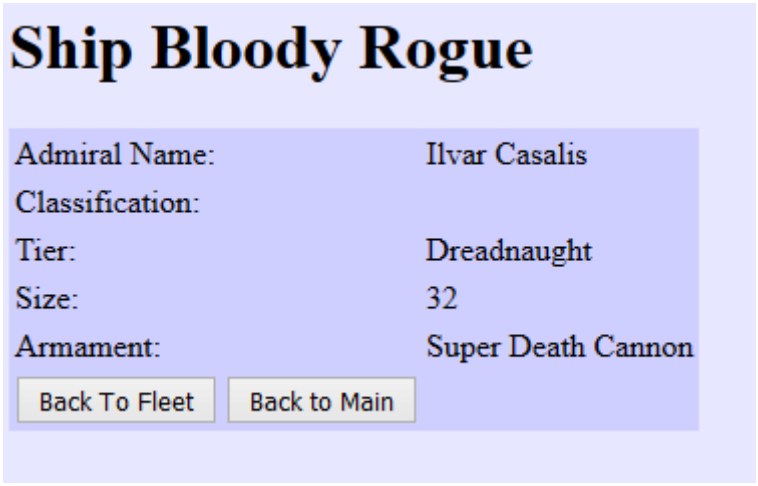
This page shows the details of a sanction enforced against a given race. Since the Terran have no sanctions enforced against them, this shows a sanction enforced against the Sikeerian. Again, all races that enforce this sanction are shown in the drop down:

This screen shows one particular trade for a particular race. Both the trade routes taken by this trade and trade partners are shown:

The military branch:

Show information for the selected fleet:
This screen displays all of the ground craft and flight craft in this fleet and has an option to select a ship to explore, since ships are more important with more individual data.

Explore a ship:
Shows information such as Admiral that is not applicable to another type of vehicle. For this reason, ships have their own pages.



Finally, the faction branch. This is the shortest branch by far, with only one page. The page has a drop down list that shows all other members of the faction:

Challenges:

Below is a list of some of the problems that we faced while designing the IGUN DB

- Our biggest problems were learning the PHP language, and learning how to use it with HTML, and learning HTML. Additionally, we had a huge amount of tables and that gave us cause to make more data than perhaps might have been recommended.
- After getting familiar with php, another difficult problem was figuring out how to keep data as we navigated between pages. Because of how our interface is set up with each separate screen being another html/php file, we had to figure out some way that important data could be passed down through all of the pages and kept, while new data gathered on a specific page could be passed to any subsequent pages that required it. After some research, we settled on using sessions to pass down data necessary for all of the pages (such as username, password, and database name). In addition, on just about every page, we used forms to post new data gathered for the next page to access. This works well as you navigate down the pages (i.e. military->military fleet->ship), however, it did not work trying to navigate *back* (ship->military fleet->military). The problem occurred because the lower page did not have the post data that the previous back needed, it only had the post data that it needed. In order to go up, then, the lower page had to give the previous page the same data so it could use it again without getting errors. We tried several methods in order to get this functionality without success. The first and simplest was to simply navigate backwards using the browser back arrow. This works in keeping the data, but the browser forces you to refresh the previous page when you go to it because it has removed it from the cache. In order to fix this, we created a back button in our html that would go to a specified location, the previous page. This would take care of the cache issue, but then the data the previous page needed would disappear. The first thing we tried in order to fix this was using Jquery.post, which posted the data you gave it (in the form key:value) to the previous page. This worked to a point, since it successfully post the data. However, it posted it asynchronously, so if you navigated back, the data would be gone again. Another solution we tried was to have the form submit the data to session instead of post, but this proved to be difficult and would have also required overhauling all of the UI that we had up to that point (more than half). Instead, what we settled on was a simple method. As soon as we arrived at any given page, we would simply check to see if the post data that we needed was available. If not, we got it from session. After getting the data, we immediately saved it to session regardless of where we got it from. This meant that the data would be stored for navigated backwards and it would not effect the post data coming down. It's not the prettiest solution, but it works well.

- Another challenge was designing the queries for the sanctions branch of the diplomacy page. Because of the way that its set up, which works exactly like it was supposed to, the queries were somewhat difficult to come up with. First, you had to reference, the sanction with the RaceId of the race that you're viewing. Then you have to join sanction with race again, this time using the RaceId's from SanctionTarget that the sanction is aimed at and display those.
- Finally, another challenge was simply designing all of the pages, since their were so many tables that to all had to be coded into the pages. The volume of pages and coded was large.

Other Things:

IGUN DB has been implemented and delivered, along with this description, to the client. We are told that they are pleased with our work. However, given more time (and more payment), we would have added some feature and functionality that we did not quite have time to get to. A list of feature that we would like to add in future versions appears below.

- Search functionality
  Although IGUN DB is designed so the user does not have to know about the inner workings of the database or even anything about the data, we would still like to have an option for them to search the database if they have something in particular they would like to see.
- Insertion functionality:
  Currently, IGUN DB only supports browsing of the vast amount of data stored within. This is acceptable for the time being, as the various races are in a relative state of peace. Territorial borders are not changing, nor are planets being invaded. A new intelligent race has not made contact with the IGUN in well over three millennia. Because of this, the data is safe not changing. Still, we would like to add this functionality add a later date. It would be very important to make it user friendly, so they do not need to know more about how the database operates inside. This would require an extensive setup. For example, we allow option for the user to add a whole new race, in which case, IGUN DB should walk them through filling out all of the data for that race. This includes general race information (name, language, leader, etc.) as well as data for all four branches. Obviously, this would necessitate a great deal of extra UI. Especially, since we need to also implement options to add an instance of any of the data in the branches. For example, if the Chozo began to colonize a planet, then IGUN DB should have an option for the user to add a planet in a designated star system. To do this, they would be walked through adding the data for the planet (name, size, type, etc.) and the data for the ages under planet (which is just the moon page). In other words, we would need to implement

an option to add an instance of every kind of object recorded in IGUN DB and each option would also have to walk the user through adding data for all subsequent pages. This would require much more UI work and the time required to do this is the reason that we did not implement this.

- One more functionality that we would have implemented given more time and funds would be a treaty functionality under the diplomacy branch. Currently, races in the same faction are implied to have treaties with one another because of the nature of a faction. However, we would like to add functionality that would make explicit if the race had a treaty with any other non-faction races.