Niklas Dahl

Moon Manager Web Application

Metropolia University of Applied Sciences

Engineer (UAS)

Media Technology Degree Programme

Thesis

14 April 2015

Metropolia

Tekijä Otsikko Sivumäärä Aika	Niklas Dahl Moon Manager -verkkosovellus 47 sivua 14.4.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaaja	Lehtori Antti Laiho

Moon Manager, tai Kuuhallinnoija, on kolmannen osapuolen verkkosovellus käytettäväksi islantilaisen tietokonepeliyhtiön julkaiseman avaruusaiheisen tietokonepelin yhteydessä. Peliyhtiö tarjoaa sovellusrajapinnan peliään varten, mikä mahdollistaa ulkoisen sovelluskehityksen. Peli käsittää lentämistä avaruusaluksilla muun muassa taistelun, tutkimusmatkailun, tieteen, teollisuuden ja kaupankäynnin parissa. Insinöörityö käsitti sekä sovelluksen suunnittelun että toteutuksen. Sovellus on tarkoitettu käytettäväksi asiakkaan Kuutiimin ylläpitäjille. He voivat sovelluksen avulla hallinnoida ja saada yleiskuvan erinäisille yksilöille ja korporaatioiksi kutsutuille pelaajayhteisöille jaetuista kuiden käyttöoikeuksista. Sovellus antaa myös tavallisten käyttäjien tarkastella heille määriteltyjä kuiden käyttöoikeuksia. Heille esitellään myös näkymät resursseista ja polttoaineesta ”torneissa”, jotka he ovat sijoittaneet kuiden kiertoradoille käyttöoikeuksiensa mukaisesti.

Sovellus tehtiin korvaamaan asiakkaan vanha järjestelmä, ja se toi mukanaan joukon uusia toiminnallisuuksia, kuten uusia hakutoimintoja tuhansien kuiden ja kuumineraalien joukosta. Tämä lähes mahdollistaa kuiden yhteensovittamisen automatisoinnin kemiallisia reaktioita varten vaadituiksi joukkioiksi. Reaktioiden avulla valmistetaan kehittyneempiä rakennusmateriaaleja. Skaalautuvuuden ansiosta sovelluksella on mahdollista ylläpitää jopa satojen tuhansien kuiden ja usean tuhannen käyttäjän tehokasta hallinnointia.

Sovellus valmistui onnistuneesti kehitysprosessin aikaisista aikatauluvaikeuksista huolimatta. Nämä aikatauluongelmat liittyivät geopoliittiseen muutokseen, jonka peliin tulossa olevat tiettyihin pelimekaniikkoihin kohdistuvat päivitykset ovat aikaansaamassa. Ketterien sovelluskehitysmenetelmien käyttö ohjelmointityössä vahvisti projektinhallintaa insinöörityössä. Sovellukseen lisättiin useita ominaisuuksia alkuperäisten suunnitelmien lisäksi, mutta moni toiminto ja lisäominaisuus jäi pois sovelluksen ensimmäisestä versiosta aikarajoitteiden vuoksi. Sovelluksen jatkokehitys alkaa tämän insinöörityön valmistumisen jälkeen.

Avainsanat	Verkkosovellus, sovellusrajapinta, kolmannen osapuolen sovellus, hakualgoritmit, sisällönhallinta

Författare Rubrik Antalet sidor Datum	Niklas Dahl Moon Manager webbapplikation 47 sidor 14.4.2015
Tutkinto	Ingenjör (Yrkeshögskola)
Studieprogram	Medieteknik
Inriktning	Digital Media
Handledare	Antti Laiho, Lektor

Moon Manager, eller ”Mån Förvaltaren”, är ett tredjepartsprogram som kan användas i samband med CCP Games datorspel EVE Online. Spelbolaget CCP Games erbjuder ett programmeringsgränssnitt som kan utnyttjas av spelare för att skapa deras egna program med hjälp av data från spelet. Spelet handlar om att flyga omkring i rymdskepp och man kan underhålla sig bland annat genom att delta i strider, rymdutforskning, vetenskaplig undersökning, industri, eller handel. Slutarbetet inkluderar både programmets planering och genomförning. Programmet kommer att användas av kundens “Mån team” administratörer. Med hjälp av programmet kan de administrera och få en överblick över månar som har blivit tilldelade åt individer eller spelargemenskap som också kallas företag. Programmet låter individer granska detaljer för månar som de har rätt att använda. De presenteras även med information om resurser och bränsle i “kontrolltorn”, som de har placerat på omloppsbanor kring månar enligt sina användningsrättigheter.

Programmet kommer att ersätta kundens gamla program och för med sig ett antal nya funktionaliteter, som till exempel nya sökfunktionaliteter för tusentals månar och månmineraler. Detta gör det nästan möjligt att automatisera kombinering av månar till helheter som behövs för att konstruera kemikaliska reaktioner. Genom reaktionerna produceras byggmaterial av högre kvalitet utav månmineralerna. Tack vare programmets skalbarhet kan hundratusentals månar och flera tusen användare kontrolleras med samma system.

Programmet blev färdigt oberoende av svårigheter med tidtabellerna i utvecklingprocessen. Dessa problem med tidtabellerna är relaterade till den geopolitiska förändringen som kommer att ske i EVE Online -spelet, då vissa spelmekanismer blir uppdaterade inför sommaren. Utnyttjandet av agila systemutvecklingsmetoder i programmeringsarbetet förstärkte projektledningen för ingenjörsarbetet. Ett antal funktionaliteter har tillsats utöver den ursprungliga planen, men flera funktionaliteter blev utanför den första versionen av programmet på grund av tidsbegränsningar. Vidare utveckling av programmet kommer att fortsätta efter detta ingenjörsarbete är färdigt.

Nyckelord	Webbapplikation, nätverks api, tredjepartsprogram, sökalgoritm, innehållshantering

Author Title Number of Pages Date	Niklas Dahl Moon Manager Web Application 47 pages 14 April 2015
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructor	Antti Laiho, Senior Lecturer

The Moon Manager web application is third party software designed for use with a computer game published by an Icelandic software company. The company provides an API for the game for use in exterior software development. The game involves interstellar activities with spaceships including combat, exploration, science, industry and commerce. This thesis portrays an application designed as a supplemental toolset for community management and meta gameplay in a limited section within industry and logistics in the game. Presented in the thesis are both the plans and the implemented product. The application is intended for use by the client's Moon Team administrators to manage and gain an overview of the allocation of moons to various entities such as individuals and player communities called corporations in the game. The application also provides the ability for regular users to view information about moon setups that have been assigned to them as well as data projections of resources and propellants in "control towers" they have deployed on the assigned moons.

As an upgrade to the client's old software, the Moon Manager application provides a number of new features to the client, such as new search functionalities within the scope of thousands of moons and moon mineral deposits, allowing administrators to nearly automate the matching of moons into reaction setups (or assignments) required for advanced material production. Scalability will allow the software to potentially support data entries for hundreds of thousands of moons and provide efficient management overview for several thousands of users.

The application was created successfully despite undergoing rough schedule changes during the development process. These schedule changes relate to the geopolitical implications of changes that have been announced to certain gameplay elements. Working with agile software development methods improved the project management aspect of the thesis. Multiple features were added to the application beyond the original plans, but several features and extra functionalities also had to be left out of the first version of the application due to time constraints. Further development of the Moon Manager web application will continue beyond the scope of this thesis.

Keywords	Web application, API, third party application, search algorithms, content management

Contents

Abbreviations

CCP	CCP Games is an Icelandic game development company for computer and console games. CCP Games have published games such as “EVE Online” and “Dust514” and are developing “EVE Online: Valkyrie”.
EVE	EVE Online is a spaceship game with large amounts of interstellar traveling along with possibilities to do combat, exploration, science, industry and commerce.
Sandbox	A sandbox game refers to a game world/universe where the game company takes as little part as possible in the development of the game’s lore and content creation. This means players themselves are a large factor in creating actual gameplay content for themselves through for example large scale warfare with other player entities.
NPC	Non-Player Character. This refers to an entity within a game that is not controlled by a human player.
PvP, PvE	The terms ”Player versus Environment” and “Player versus Player” are well-established terms used in connection to computer games. They portray in what kind of activities players are involving themselves, whether they are playing against the game world NPCs or competing against other players.
API	Application Programming Interface. A server may provide a connection point where no actual web page resides, and instead respond with unformatted data as for example JSON or XML.
CDN	Content Delivery Networks are servers where providers in a uniform manner deliver their digital contents to users.

1 Introduction

The video game company CCP Games published a game called EVE Online (commonly known just as “EVE”) in 2003. The game has amassed hundreds of thousands of players over the course of its 12 year history. Players are all connected to one another and can both communicate and interact in other more direct ways. This is thanks to what is called a single shard universe architecture with multiple physical and virtual server machines hosting a singular persistent environment. [1.]

In EVE Online players first create a character to represent them and have the option to complete tutorial missions to familiarize themselves with the game’s mechanics regarding space travel and basic combat with their first spaceship. After the tutorials are completed the player is free to pursue their own path within the game. The multitude of options available to the player and the open nature to which they are presented is often referred to as a “sandbox” style of game play. Beyond the starting tutorials players can also opt to follow chains of missions that delve deeper into the various ‘careers’ within the game, for instance advanced combat, commerce, industry or exploration. In the early stages of the game players often remain in the inner ‘empire controlled’ areas of the game world. These areas are protected by game controlled police forces and offer the new players a measure of security. As players gain more experience with how the game functions, they inevitably encounter the choice of remaining in secure space, or venturing into more lucrative and dangerous areas.

The game world in EVE is comprised of roughly 7,800 star systems, most of which are connected to each other by stargates, allowing for nearly instantaneous interstellar travel between two nearby stars. The known parts of the galaxy which EVE is based on are divided into galactic regions, some controlled by the non-player factions, and some declared as outlaw space. Each star system in EVE has a security value determining how safe the star system is. This value comes into play in certain game mechanics relating to unauthorized combat, but it also has another implication.

A star system’s security value can range from 1.0 to -1.0. A system with a security value between 1.0 and 0.5 is determined as a Hi-Sec (High Security Space) system and is controlled by one of the game’s four main NPC factions. A Hi-Sec system is also monitored by “CONCORD”, the computer controlled police faction in the game. A system

with its security value between 0.4 and 0.1 is rated as a Low-Sec (Low Security Space) system and is likewise also controlled by one of the main NPC factions, but does not have CONCORD policing it. This makes Low-Sec areas preferable locations to host PvP combat (Player versus Player). The lore of the game details an interstellar war being waged between the four main factions and this is manifested in the game with capturable Low-Sec areas along the borders of each of the empires. Lastly there are the vast areas of space where star systems have a security value ranging between 0.0 and -1.0. These areas are called Nulsec (Null Security Space, or Outlaw Space), and the systems are free for player controlled corporations and alliances to claim as their own.

Corporations are player created communities with a large number of game mechanics involved in their management. The number of players in a single corporation may range from a single member to several thousand players. Any number of corporations may band together to form an alliance. The largest and most influential alliances nearly hold control of up to a hundred star systems [2].

These large alliances wage war against each other for control over better space and valuable resources. When a spaceship in EVE is destroyed in combat, it is lost forever, unlike in most other games where a player would simply spawn anew in a similar ship. Thus the economy flourishes as players harvest resources, build new ships and sell them on the in-game markets. The influx of new ships to the market is more or less a daily event even in the most remote areas of the galaxy.

This final year project is based on a small part of the resources available in the game. Essentially it is a third-party toolset which also utilizes the API provided by CCP Games for software developers to enrich their external tools and applications. When alliances in EVE Online claim space for themselves, the corporations therein gain access to a group of special commodities that are obtainable only from harvesting the moons in a star system. These ‘moon minerals' are the second-most valuable crafting materials in the game. A browser based application was created for this final year project. The application is a toolset for individuals appointed by an alliance to manage assignments of moons to corporations and individual players as well as providing the possibility to perform complex searches for combinations of different deposits of moon minerals. Players will also be able to view their own assignments and those owned by their corporation. There are over 100,000 moons in the game, and 20 different moon minerals ap-

pearing in different quantities and rarities. The application has been tailored for an alliance called “Fatal Ascension” (essentially the client of this work) that inhabits a region called “Fountain” in the western part of the game map. For this purpose only the 4,849 moons in the Fountain region are included in the first installment of the application. [3.]

A database including the moon mineral deposits already exists. The data in it has been gathered through the combined effort of players in Fatal Ascension flying to each of the moons in Fountain one by one and scanning it with slow moving probes. This database has been data mined and the contents combined together with external data from the EVE API to create the basis for the database used in the Moon Manager toolset – the application made for this final year project. At the time of writing this, alpha testing of the application is already underway and once the application is deemed finished, it will enter a beta testing stage where the client (the Fatal Ascension moon team administrators) will use both the old and the new application simultaneously before the application is officially handed over.

2 Background

2.1 Algorithms

An algorithm is an ordered list of steps to take in order to reach a certain kind of result. In computer science, these steps are lines of code and specific commands or rules that have been ordered to one line after another programmatically strive towards the desired outcome. When a large amount of computational processing is required, the way that the computations are organized and ordered makes a great difference. An algorithm can be repeated multiple times in order to achieve absolute accuracy and time efficiency. It may be that a certain block of code achieves a certain goal, but takes a great deal of time in doing so. At this point a developer often has to find a way to optimize the algorithm in order to find a more suitable arrangement that takes less time to execute. Occasionally mistakes can even be made where the result is correct, but the same processes are repeated multiple times by accident without noticing it. [4; 5; 6.]

The way algorithms are constructed very often affects their processing time. This becomes extremely apparent when dealing with a large number of mathematical computations. Two differently constructed algorithms leading to exactly the same result may have very large differences in their execution times. The main reason for this is the train of thought put into the functional sequence the algorithm goes through. Very often similar results can be achieved through very different roads and unnecessary steps are included which ultimately could be transformed into fewer lines of code doing the same thing. [4; 5; 6.]

2.2 Search Algorithms and Recursion

Search algorithms are simply algorithms constructed to find targets by comparing a set of values to those found on a gathering of entities. The search algorithms exist in many different forms and for various purposes. Some of the most common search algorithm types focus on matching a string of text against longer text blocks, or mapping connections and routes in a network.

Text searches are very straightforward and are most often programmed through the use of a pattern that is processed through a body of text. This is commonly known as

the “needle in a haystack” problem. A simple way to perform such a search is to compose a set of steps in which the two strings, the searched text and the pattern to find, are looped through in a multidimensional “for loop” – a loop cycling through a set of commands for a predetermined number of times. The first loop completes a circle for each of the characters in the text to be searched through. Within the first loop, a second loop is initiated which compares the first character of the pattern to the character active in the first loop. If it is a match, the inner loop continues to circulate comparing the following pattern characters to the next characters in the searched text until either a mismatch breaks the loop, or the entire pattern gets matched returning a desired outcome. In PHP there are several built-in functions to handle basic string searching, for example strpos() to find the first position of a pattern in a string, strstr() to select the rest of the string after the first match of a pattern, and strpbrk() used to select the rest of the string after an occurrence of any character in a pattern. [7.]

Another common way to perform a pattern match is through the use of regular expressions. Regular expressions are text strings by themselves, but they follow a certain syntax in order to express a pattern. A regular expression can either depict a specific character or a collection of characters that need to be matched, and multiple such collections can be combined and ordered after one another to form search patterns that do not necessarily point to a specific word or phrase. [8.]

```
\b[A-Za-z0-9]{3-5}\b
```

Listing 1. A regular expression pattern.

In the above code example (see Listing 1) a string pattern is shown which will match a string containing any capital or small letters or even numbers, but is specifically 3-5 characters long. The \b markings (or \^ and \$ for comparing a string’s entire structure) at the beginning and end of the pattern point out the borders of the pattern. Everything inside them is a part of the pattern. The [and] characters indicate the beginning and end of the criteria for a specified length of characters to be matched at a specific point in a string. Paired by a dash in-between, the capital A and capital Z indicate that any capital letter from A to Z may be matched. Any number of such pairings may be coupled together to form a collection of possible matches. The length of the string portion to be matched can be indicated in three different ways by adding the indication directly after the [ ]: the { and } characters containing a single number or the minimum and

maximum number of characters to specify an exact length. A * can be used to indicate that a character can appear zero or more times in that specific portion of the pattern, or a + sign can be used to show that at least one or more characters must be matched towards this portion of the pattern.

The second code example below (see Listing 2) shows a complete pattern, which can be used to find out if a string collected to be analyzed is an email address. At any point where a backslash appears alone, it means that the following character will ignore its regular use and be a part of the matched characters instead.

```
\^[A-Za-z0-9._-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}\$
```

Listing 2. A complete regular expression pattern for email validation.

An example with accidental bad algorithm design was discussed in section 2.1. A faulty construction such as the one in the example may happen for instance when initiating an object, and then populating it multiple times in several different places, once right after the object is created, and once within the object, and perhaps even a third time in an internal function called when a supplemental function is called. When this is done in a family tree, the result can be a lot of wasted computing time, and processing times may turn from half a second to half a minute. The increased processing time would be the result of each of a person's children getting initiated in the same manner several times, with each looping recursion cumulatively adding up more and more processes on top of each other. Recursive code must be precise and rarely forgives developers for making errors such as the one exemplified. In extreme cases a section of code can be put into a state of endless recursion that never completes and results in an application going unresponsive.

Recursion in programming may be presented in a number of ways, but generally it implies that a function calls itself for a deeper level of computing [9]. A function may also process a number of functionalities which end up calling the same function from within another object. In the above example of the family tree, there are two places where recursion occurs. At first when a user requests to see a person's family tree, a function is run that creates a Person object, and that object's function "initFamily()" is called. Within the second function, a database query is done to find out if the person has any children, and for each child a new Person object is created, and that object's "initFami-

ly()" function is called, until a person does not have any registered children. At this point the child objects are simply stored nested within their parent's object. Then in the view-layer of the family-tree application this nested family-tree is then circulated through a view-helper object's "parseWithFamily()" function, which compiles a data projection on a person's information. During compilation of the data projection, the function calls a new instance of itself to create a data projection for each child, nested within the parent's view element.

Searching through a map of entities connected by links and plotting routes or finding connections is a network search. These searches are commonly made directly into databases, but may also be done through data on the scripting layer. In the case of direct database queries, initial requirements for datalinks are made first. After that a database table containing information about connections or links between two nodes in a network, is called multiple times – recursively – and the query statement connects data from the table to the other end of a copy of the same table. This connection of one end of the table to the other end of a new copy is done over and over again until a desired amount of recursions are reached.

It is not always necessary for network search algorithms to indicate the best route from point A to point B, if for example only finding out whether A and B are connected is enough. However, in most cases directly finding the best suited option is a desired outcome. Occasionally this cannot be achieved by going through the database with only one query, and a supplementary scoring system needs to be implemented. The scoring system can score routes based for example on arbitrary distance by calculating the amount of links, by physical total distance of a route, total time required to traverse the route, or a number of other factors related to supplemental data. Usually a mixture of these values are used, and results are rated by the best overall score. In the case of for example traffic navigators however, the quickest route is usually selected by default. The situation may even be that no scoring is given for a route's length or travel time, but instead the route is scored for the direct physical distance between the beginning and end of the network route, as long as a connection exists. Such would be the case for example when finding out which two members of the same family-tree live physically closest to one another.

2.3 Technologies behind the Application

HTML5 was introduced at the end of October in 2014. It is the 5th installation of the Hypertext Markup Language and provides interface designers with new opportunities for faster and easier workflow. This becomes evident through the large amount of modified functionality as well as several new elements having been included. One new element in particular is worth mentioning: the <canvas /> element. The canvas element has caused a lot of excitement among web developers and is a revolutionary step in terms of what is and has been possible in web development. It allows a developer to draw graphs onto the screen, within the boundaries of the canvas, directly in front of the end user. This requires scripting, but the concept of being able to draw graphics in a browser window is something developers have waited a long time for. Partially this has already been accomplished with the new functionalities brought forth by CSS3, but even CSS3 has not had full support from all web browsers until very recently. [10; 11.]

The biggest source for enabling interaction and real time events in a user's client (in this case on a web browser) is JavaScript. JavaScript is a scripting language that runs locally within the client's browser. This differs from what traditional web programming does, in that there is nothing happening on the host's server when a JavaScript command is executed, unless it involves an AJAX call. AJAX (Asynchronous JavaScript and XML) is not as much a coding language itself, but rather stands for a method of using multiple pre-existing technologies in a very fluid and unobtrusive way. What this means is that a user's client does not get a new page request every time they click something, and the browser does not freeze up as the main thread (computer process sequence [12]) that the page is running on is not stalled for the duration of the AJAX call. Instead for example a loading bar can be displayed where the new content is about to show up, while the user can still continue to use/read the rest of the webpage they have active in an unobstructed manner.

JavaScript can be utilized for much more than simply altering things in a browser view. JavaScript also enables communication with the server without having to bother the user. An end user will generally not see all the JavaScript that is loaded alongside a single page-view. The amount of code exceeds multiple full screens' worth of text even in a "minimal" compact form. The JavaScript code can communicate with the servers asking them for updates and checking for new content. Such processes can be run on top of large layers of timers created in the JavaScript, building an overhead (computa-

tional load) within the browser, but minimizing the amount of traffic the servers actually have to endure. All essential parts of an application are loaded at one single time, and most processes and content that do not require transferring the user to a third party application page, for instance moving to a settings page, can be loaded through real time asynchronous XMLHttpRequests (with AJAX). This improves the user experience as the user constantly sees the page he or she is on, and barely ever has to stare at a blank new page with content slowly loading into the browser. [13.]

'jQuery' and 'Less' are a library and framework respectively for JavaScript and CSS. They provide premade functions and data structures that ease the use of both JavaScript and CSS in web application development. jQuery contains a number of functions such as pre-programmed animations and shorthand links that hasten a developer's work. The main idea behind it is to hide away the heaviest lower-level details of the JavaScript, and make it more user friendly. For example the syntax to select an individual element is extremely simple compared to multiple selectors in regular JavaScript code. 'Less' in turn is a kind of framework essentially providing more in-depth ways to work with stylesheets on the server-side of an application, essentially enabling structured and heavier code packages to be delivered to a user's browser and allowing performing a large part of code refactoring by simply altering variables. The idea behind Less is to directly provide the server means to read and understand new and simplified ways to construct CSS markup, through which a developer can spend less time coding, and more time getting quick results. In other words, jQuery is a library providing an easier way to write JavaScript code, while Less aims at partially removing the need to write new CSS every time alterations need to be made. [14; 15.]

The main programmatic structure of the Moon Manager application is based on the Zend Framework core utilizing PHP and MySQL. PHP is a server side programming language for building dynamic web pages, and MySQL is a relational database system for storing and connecting large amounts of information efficiently.

When a bundle of premade code, classes of various kinds, is organized into neat folder hierarchies, and those classes are generic or abstract in nature in order to provide supporting tools for application building, the bundle can be called a framework. A framework is just that, a collection of pre-made abstractions of useful objects, which a developer can extend to their own needs. The Zend Framework also provides the fun-

damentals of a fully working application, but also allows for a use-at-will style of programming, akin to a regular class library. [16.]

Most server-side frameworks provide (or even expect) a pre-organized application structure that a developer can tap into with ease. Particularly in web development the phrase Model-View-Controller (MVC) is often brought up in frameworks and other object oriented coding. Model-View-Controller stands for application structure patterns for creating and delivering projected data to user clients. What it involves is creating your entire application divided into three main areas. Controller refers to the application logic, what happens, what models are called for processing information, and what variables are passed onwards to the view layer (data contents are unimportant at this level). Model refers to the code designed from a company's or specific website's point of view, including the business logic for the specialized needs of the software in question. View refers to the view layer that includes all of the application's data projection and other visual content, as well as determines exactly what data is presented and how it is projected to the user client. [17.]

The entire MVC-process assumes that all application logic can be found on the controller layer. Incoming data is processed and output data is requested from business models. This data is then passed to the view layer, which displays the data as a response to the client browser. At no point in time does the controller actually know what kind of data is going through. It simply receives the original page request, passes it onto models of choice, and transfers the response from those models over to the view layer. Technically multiple applications could be created, doing diverging but still very similar tasks without as much as touching the controller layer, and simply applying changes in the model layer with minor changes to the view layer. [17.]

3 Planning the Moon Manager Application

3.1 Reasons for Replacing the Old Application

The EVE alliance Fatal Ascension (FA) houses over two thousand players. In addition it controls space in the game covering nearly the entirety of the Fountain region (73 Star Systems) as well as several other key locations. It would be a monumental if not impossible task for a handful of administrators to keep track on paper over thousands of players and their allocated assignments, not to mention nearly five thousand individual moons. The Moon Manager application functions as a daily toolset used by the appointed administrators to oversee all aspects of owner relations for moons and monitor activity. An example of possible straightforward use cases has been compiled (see Figure 1).

The application must contain all the necessary functionalities for searching through moons and mineral deposits. There are 20 different moon minerals including mundane gases and basic metals. The rarest alloys only appear a handful of times in thousands of moons in a single region. Within the game these minerals are distinguished by rarity: R0, R8, R16, R32, R64; the higher the number, the rarer and more valuable the mineral usually is. Administrators of the application must be able to distinguish their searches by basic rarity as Fatal Ascension has a guideline stating only a certain number of the rarest minerals are distributed per corporation, and only the most common metals will be available to individual players to make use of according to the rules of the FA Moon Franchise program (an internal program allowing players to utilize moon minerals in alliance owned space for purely personal benefit, by paying a small rental fee per type of moon mineral on the moons used).

In addition to simply searching through moons and the mineral deposits, the application must also be capable of coupling minerals together for pre-determined reaction sets. The way moon minerals are essentially utilized is by erecting a starbase control tower close to the moon (only one can be in place for each moon) and installing a moon harvesting array on the starbase. The harvester will gather the resources from the moon on an hourly cycle, and transfer them into a silo that the player has also installed onto the control tower. Two silos with specific minerals in them are then connected to a “simple reaction array”. EVE employs a construction system utilizing blueprints. One

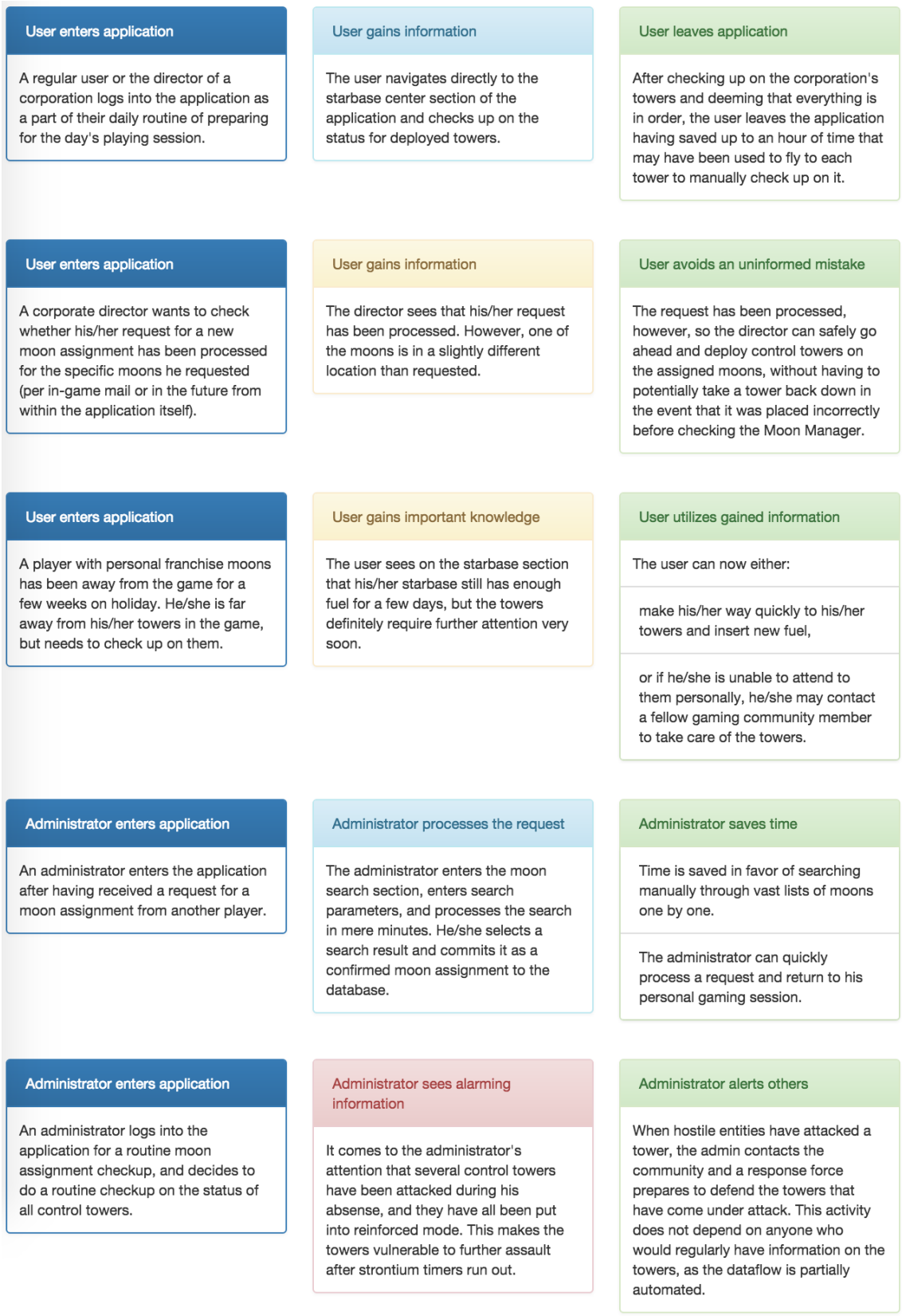

Figure 1. Portrayed from left to right: Use case scenarios for the application.

desired simple reaction blueprint is then placed inside the reaction array, which can then be coupled to an output silo. The blueprint determines which materials are required for the reaction and what the product is. A simple reaction array outputs intermediate materials which are then coupled into a “complex reaction array” with a complex reaction blueprint (see Figure 2). This process results in the end product, advanced moon materials, which are used in the building process of higher quality spaceships.

Figure 2. Silos and reactors are linked to each other in the EVE Online in-game moon interface.

The coupling of moons for reaction sets becomes an important factor in the application because it is rare to find a moon containing multiple types of minerals, and even more rare for those minerals to be useable in the same reaction. The most simple of the advanced reactions require a total of four different moon minerals, while the most complex ones need a total of seven different moon minerals (the most complex reactions require eight materials, but in each case two are of the same type). Also very commonly it is not possible to get all of the materials from within a single region, so some of the

moon minerals will have to be purchased on the open market from players residing in other distant regions and imported. The application thus needs to be able to determine what a feasible amount of moons would be to use for a single reaction setup. There are some moons that may hold both components for the simplest gas based reactions on the same moon. The moon assignments thus have to be gathered through a mix of settings determining for example how many minerals must be included, how far apart the moons may be (in different star systems) and whether moons have to contain more than one mineral.

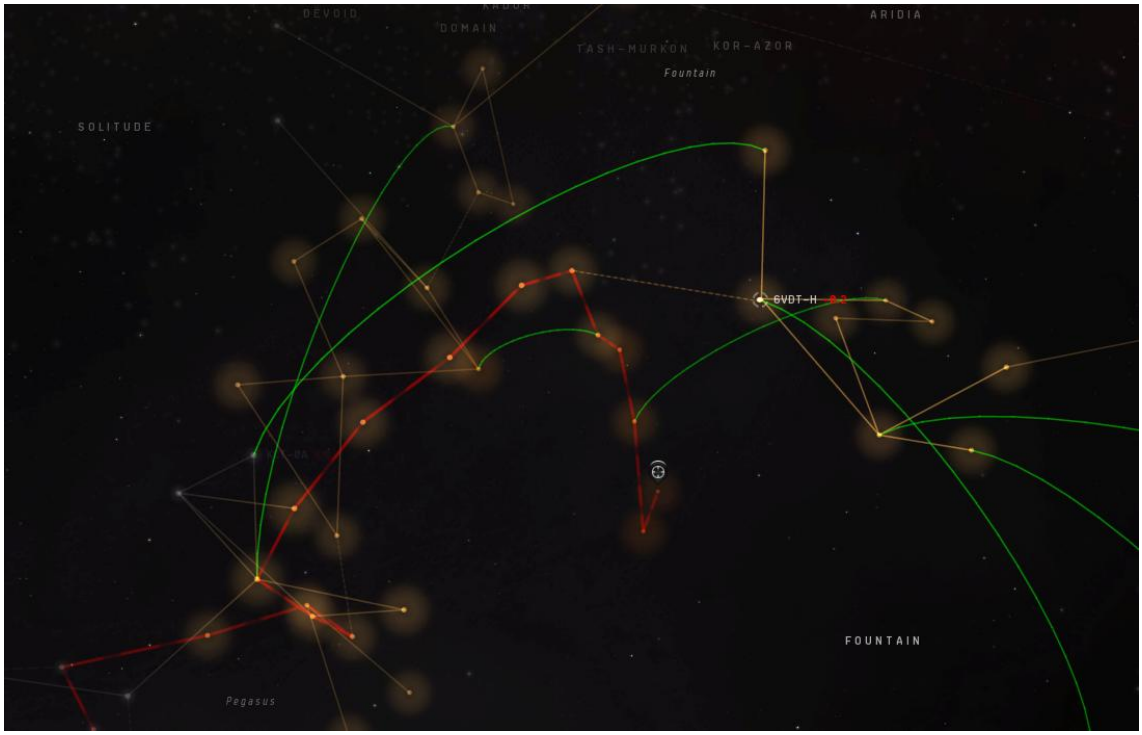

Figure 3. The EVE Online in-game map shows stargate and jump bridge links between stars.

The search functions will also rely heavily on route mapping algorithms to determine distance and viability of specific moons for a reaction setup. A recursive database search is necessary to navigate the starmap (see Figure 3.). The distance between stars (amount of jumps via stargates) will be used for a ranking system to maintain preference for possible setups that have all their moons very close by, even all in the same star system if possible. One possible feature suggested by the end users has also been to map the distance between stars by their physical distance derived from their positions in the xyz-planes (these values are provided via the EVE API). This would enable a search functionality that limits searches to star systems that are within the maximum jump distance of a given ship's jump drive engine (e.g. ten light years for

a jump capable freighter class vessel and five light years for all other jump capable ship classes).

3.2 Starbase Section

Beyond merely being a search and assign toolset for moons, the Moon Manager application will also include a section for monitoring activity and fuel levels on starbase control towers deployed onto select moons. Regular users who have starbase roles in their respective corporations will be able to view details about the control towers their corporation has deployed, anywhere in the game world. To achieve this, a leader or director of a corporation will need to input valid corporation-type API credentials into the application. The EVE API will then be queried periodically for information regarding all the control towers that have been deployed on behalf of the corporation. Something worth mentioning is that individual players are unable to deploy control towers for themselves. In terms of game mechanics a control tower will always be owned by a corporation, but with Fatal Ascension's Moon Franchise program, trusted individuals are permitted to deploy towers with the consent of their corporation and utilize the moons for personal benefit.

The information provided by the EVE API is very detailed. A character's skills and amount of experience (skill points) as well as all other details about a character can be queried and retrieved as an XML feed. Even financial details including full logs and in-game mails can be retrieved as well as a character's active industrial installations. Players are able to create their own tailored API keys to EVE Online via the game's account management webpage. These keys can focus on particular aspects of the full EVE API and thus provide the player with a measure of data privacy. The Moon Manager application requires an API key that has been tailored to expose information at the 'corporation' level [18]. By enabling a set of API credentials to allow director level access, data on the corporation that the player is a director or leader in can be queried directly from the API. Part of this data includes the details of all control towers the corporation has deployed, whether they are active or offline. Listed in the application will be every tower's fuel levels, estimated online time, material amounts in the silos, as well as internal data to whom the tower has been assigned, be it a corporation or an individual, or whether the tower has been earmarked as alliance property (profits would then be going to alliance funds such as the ship replacement program). Individual us-

ers will be able to monitor their personal towers, and corporation leadership will have full view over both their own and all other corporate towers.

3.3 Communication with the Client

As the initial assignment from the Fatal Ascension Moon Team (the client) had simply been to make a plan for a new management tool for moons and assignments, the first meeting where the plan was presented gave good insights as to further specifications the client desired. The base plan was accepted as is, but some additions were made to the search options for moons, most specifically certain flags and notes for assignments. Fatal Ascension taxes moon owners in their sovereign space under a certain guideline, but some moons need to have the possibility of being flagged as not taxed. There are various undisclosed reasons for this.

Another high priority addition that had not come up in initial planning is the ability to add notes and certain secondary information to assignments and connect data to moons regardless of existing assignments. This included the existence of possible control towers owned by hostile entities, and generally the capability to see whether assigned moons were not occupied. A history listing for moons was also desired, to be able to view state changes for a specific moon, and when an assignment had been changed, and by whom. Some search functionalities emerged late in the conversation and included a hint towards the taxation system by proposing a search filter allowing for all taxable moons that are not towered – not occupied by a control tower nor part of an assignment – to be listed together with other filters such as mineral rarity.

There were lengthy discussions surrounding the client’s taxation system and how it would be implemented into the application, along with a number of other ideas that the client would like to see. However, the taxation related features were not deemed a high priority for the first beta release of the application, and will instead be deployed in later releases.

By far the highest priority requirement from the client was that of security. The old system seemingly does not properly differentiate between alliance owned moons' fuelers and the actual moon team. It is important that only members of the administration team can view and alter any assignments. Meanwhile alliance appointed fuelers handling

towers owned by the alliance for fundraising purposes must have access to not only their personal and their own corporation's control towers, but also viewing rights (no editing rights) to fuel and silo content levels in control towers held by the Fatal Holdings corporation.

Another requirement from the client was to facilitate alternative views. It is common for corporations to create secondary ‘subsidiary’ corporations to host the ‘personal franchise moons’. Allowing a corporation’s director to view control tower lists for their main corporation and those deployed under the subsidiaries was deemed highly desirable. This can be accomplished directly through EVE API queries and checking account details from the host website also containing the alliance's forums. On the alliance forums users may connect multiple EVE game accounts to their forum account.

3.4 EVE Online API

The Application Programming Interface or API provided by CCP Games for EVE Online is based on simple HTTP requests, which return an XML response. The API allows third party application developers to query both public and private data from the EVE database. There are very few gaming companies providing similar amounts of data for external use. The only restriction for use of the data is that commercial selling of products utilizing the data is prohibited. Beyond this there is a limit to the frequency of requests sent to the API server, and a certain amount of errors (incorrectly formed requests) per time limit leads the requesting IP address to be banned from the API for a limited amount of time. Another thing to note is that the EVE API does not provide x-origin support. This prohibits use of XMLHttpRequests to the API webpage, essentially preventing AJAX calls. When an AJAX update would feel like the best choice for a simple request to the API during runtime of the third-party web application, the request must instead be made via the application’s host server, which functions as a middleman and passes the requested data back to the user’s browser. While the method may seem cumbersome, it may be a necessary step to provide fluid functionality.

A developer or player gains access to the EVE API through the game’s account management webpage. On the webpage a player can create new API keys and manage already existing keys. A player can have multiple API keys at any given time. Keys are distinguished from one another by a key ID (keyID) and have a long verification code

(vCode). The keyID and vCode form the credentials used to access the API data. Every key is also assigned an accessmask, compiled by numeric values determined for each accessible data set (see Figure 4). In this manner even though the data available from the API is extremely broad, a player can share certain relevant data with a third party application – for example a community web page to confirm the player is affiliated with the forum's owners – while still retaining his/her privacy in terms of in-game mail and owned assets.

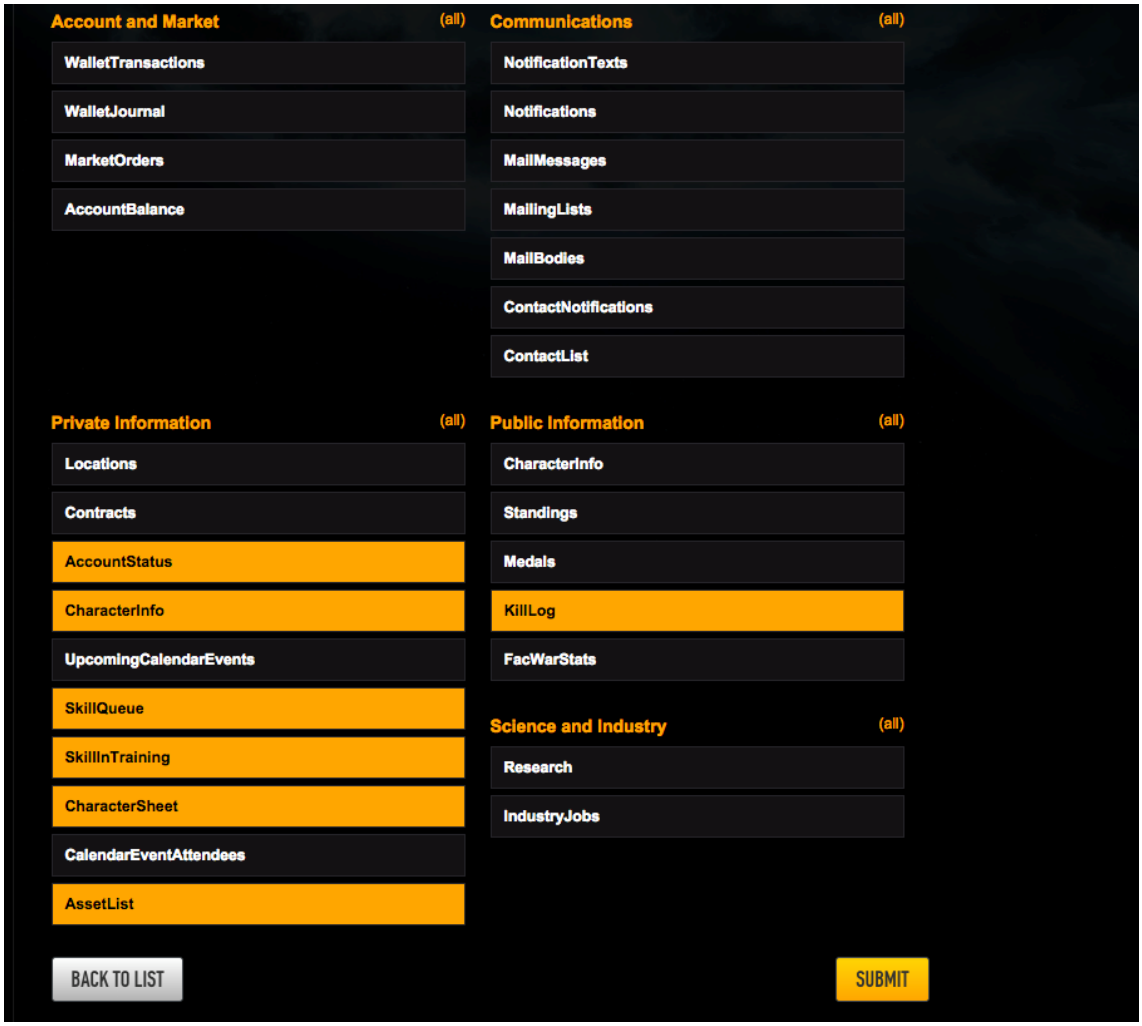

Figure 4. The EVE Online account management webpage allows for API key customization.

A corporation marked API key requires that a character on the player's account has director or leader level roles in a corporation. This will enable the API key to be used for queries regarding a corporation's financial logs, installed industrial jobs along with many other details that regular members would not have access to. The interesting data set in regards to the Moon Manager application is the starbase control tower data.

As the returned data is provided in the form of an XML document, it is extremely easy to navigate with a parser and desired details can be accessed without trouble. [19; 20.]

3.5 User Friendliness and Visual Design

When creating dynamic websites the first thing a developer often thinks of is how certain features can best be deployed and delivered to a user. Occasionally the developer forgets to give consideration to what the user actually wants. The Moon Manager toolset emphasizes user friendliness and ease of access. No feature available to a regular user or administrator is hidden behind multiple links and page loads. The majority of the functionality is deployed directly after navigating to the desired main section, and sub-functionalities are embedded as asynchronous AJAX calls within the main interface. Other details which have been taken into consideration include employing a minor level of interaction to the application. For example the starbase center will show active timers ticking downwards, instead of merely the clock times for fuel depletion on towers. Countdown timers are also present for displaying the time left on moon mineral silos filling up.

A level of consistency has also been upheld in order to make the application look like a unified entity, not just a collection of different features. Future users of the application have also been interviewed briefly on matters of visuals in addition to functional features. Visual elements as well as the styles and appearance of buttons and links have been selected to represent the purpose of the application. These graphical elements will however not be fully implemented for the beta phase of the application.

4 Implementation

4.1 Base Installations

In order to clarify the construction of the entire application and to settle the types of data available and expected, the MySQL database was elected to be the first part of the application to be built. At first the database included 19 tables set up to contain various information to be both data mined from the “EVE static DB” – a database containing all the static data for EVE Online such as star system information, map connections between systems, what planets and moons are in those respective systems, along with a multitude of other information. The application’s database also includes a lot of manually inserted data as well as input through actual use of the application. The core of the database is the registry-table, which holds records of all entities (even abstract) entered into the database with a universal ID assigned for internal use within the application. This includes for example users, administrators, moons, planets, regions, moon minerals, moon setup assignments, and update/history events for moons and assignments. As the database uses a universal ID for all appearing entities and multiple database tables cross reference each other or provide recursive stepping stones, no foreign key mappings are used, and instead all linking of tables and data entries is done at runtime.

The section of the database of highest interest contains the actual moons and directly linked relations to the mineral deposits, but a large number of supplementary tables are in place to reduce the load on the tables with the highest frequency in usage, thus enhancing performance speed. This was particularly important to note and implement already for the first version of the application due to the future plans for broadening the scope of the application to include all 150-200,000 moons in the game, as well as potentially hosting over ten thousand monthly users. The extra tables include data such as constellations and regions that star systems are located in, free text notes for specific moons and assignments separated from the actual minimal information anchor tables for said entities. The amount of database tables increased from the original 19 to over 30 with the addition of notes and select other features requested by the client, along with the necessity to locally cache parts of the data retrieved from the EVE API. The table count is expected to rise over 40 for later versions of the application. With the

possible implementation of a caching tool called ‘Memcached’ and some code refactoring, this may be brought back down again later. [21.]

The database scheme (see Figure 5) shows the structure of the database and reveals the relation tables (for example moonrels and systemrels) containing the actual data linking entities to one another. For example the systemrels table is the most vital part of the recursive mapping search algorithms detailed later in this thesis. The tables containing temporarily locally cached data are prefixed by “cache_”. The main reasons for opting to cache certain data locally are both the desire to optimize bandwidth usage and the fact that CCP Games actually provides a “cachedUntil” value with every single API pull, and wishes that third party developers respect these timers and cache related data locally at least until the timers have expired (this varies from 5 minutes to 1 hour, depending on the data type).

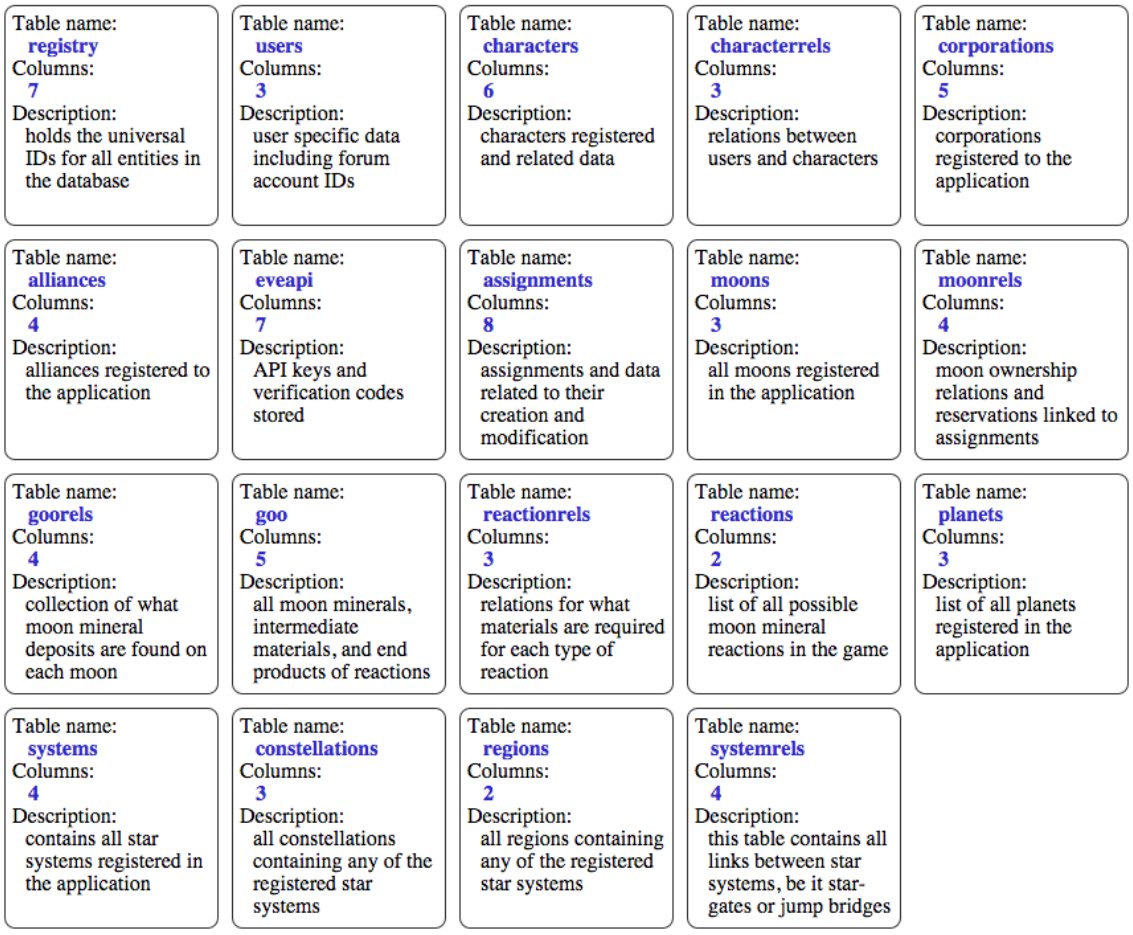

Figure 5. The original database tables.

The second portion of the initialization process for the database was the data mining and importing of the information stored in the old application. This meant performing a clean system transfer and rebuilding the information into a new format. This was accomplished through a simple re-organizer function under the administrator section of the new application. The data transfer and an updating algorithm that ended up never being necessary, were the first functionalities built for the application along with the importing functions for the EVE static DB information. This enabled proper testing with existing real data for the rest of the application's development process.

For the data transfer functionalities to be properly implemented into locations where they could be accessed again at a later time – with as little trouble as possible – the core of the backend had to be in place first. The installation of the Zend Framework core began with setting up a folder hierarchy for the application, separating the application logic from the content viewable for a user (see Figure 6). The first release of the application uses the latest version of Zend Framework (version 1.12.11), but will later utilize the migration options available, in order to transfer over to a 2.x.x version of the framework at a later date.

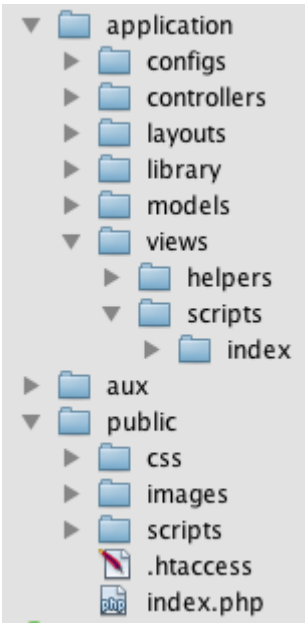

Figure 6. The folder structure in a Zend Framework application follows a clear hierarchy.

The bootstrap file index.php is the heart of the application and contains all setup information and handles the forwarding of every single request sent to the application. A .htaccess file in the server root activates the rewrite engine and handles pointing all

non-resource requests back to the index.php file. This technically prevents 404 errors from the server and allows handling of even possibly non-existing request addresses. In order to avoid possible mishaps due to changing operating systems and database migrations in the future, a command “SET NAMES UTF-8;” is also forced into the database at the start of every request loop. The bootstrap file also instantiates the working environment for the application. At development stages, the application.ini file in the configs folder contains two namespaces containing various details such as visibility of PHP errors and database login credentials: production and development.

The actual page distribution is handled by dividing functions (or more specifically actions) across a set of controller layer objects. Essentially a controller represents a section of the application visible to a user. This provides clarity for both development as well as end user experience. In addition to the contents of the main sections there are also a number of access points within each controller for handling AJAX calls. The main sections are:

- IndexController: The landing section of the application serves as the main index page, welcoming the user and providing a login form as well as links to view known hostile towers.
- AdminController: The administrator section contains all management functionalities available for the application. This includes listing all moons, searching for specific moons and viewing the moon’s details, listing assignments with filters for all / by franchise / by corporation / by alliance, managing personnel with various special access levels to the application (administrators, senior fleet commanders, alliance fuel technicians, individuals with viewing rights to other corporations’ towers for undisclosed other reasons), viewing the status for corporate API keys to see the validity of their verification codes, managing various requests for new assignments and alterations, test actions and update/maintenance functionalities with multiple warning prompts confirming intended use.
- AuthController: The AuthController handles all functionalities related to user sessions and functions as a contact layer for the future SSO (Single Sign On system) to be integrated with the forthcoming new alliance website (not directly related to this project).
- AssignmentController: This contains all functionalities for regular users to view their own and their corporation’s moon assignments, as well as the forms to file in requests for new setups, and requests to alter existing setups. This may be for example changing a single moon in an assignment for another one, but maintaining the rest of the assignment intact.
- ErrorController: This section handles all errors rising out of the use, misuse, or inappropriate use of the software; for example database errors, non-existing URIs.

- StarbaseController: Users can view the active control towers they have viewing rights for in the starbase section of the application. Select individuals (mainly directors and alliance fuel technicians) will have the ability to select between viewing their own towers or those of another corporation.

- UserController: Users must maintain valid API keys with active verification codes for being able to access the starbase section. Also each corporation must have valid corporate level API keys entered into the system. The user section of the application provides functionality for entering said API keys and displays a compact summary of contents to which the active users have viewing rights.

4.2 Optimizing Performance and Bandwidth Usage

As the Fatal Ascension web servers are operated and paid for by the community through internal donations, it is imperative that the forums and all supplementary applications hosted on the web servers cause as little CPU load as possible and limit to some extent the usage of bandwidth. This has also played a factor in the design of the moon manager application. It proved difficult to achieve ways to reduce direct computational load on the servers, as many of the functionalities may require fairly complex and bulky algorithm processing, but there are many ways to optimize bandwidth usage.

Many front end libraries and frameworks provide the code required through a CDN (Content Delivery Network). This is the case with for example jQuery. By simply linking a resource in the HTML layout's header to the jQuery CDN, it is possible to bypass the necessity to always download the latest version, and inclusion of the jQuery library does not cause any bandwidth use for the website's host servers. Also by avoiding unnecessary updates to local CSS and JavaScript files, bandwidth can be saved. This can be achieved through setting cache expiry timers for certain types of resources in the ".htaccess" file in the server's web root. These values have been set to two months for CSS and JavaScript files as well as various images downloaded from the image server hosted together with the EVE API. In order to avoid situations where a recently updated file would end up not being refreshed in a user's browser, a special syntax for the filename is used including the output of the filemtime() PHP-function, which delivers the timestamp when the file was last modified. Use of the filemtime() function may be cumbersome for an extremely high usage website, but use frequency where the layout of the Moon Manager application is loaded should not exceed more than a handful of page reloads per minute at most. [22.]

When creating a website with a lot of AJAX calls, bandwidth usage can be further optimized by only providing the data a user requires at any given time. In the Moon Manager application this is achieved by minimizing the amount of data sent with each request, and delivering AJAX responses only in the form of JSON. This way only the necessary information is delivered for each request, while the JavaScript that has already been set up on the user's browser handles the data projection onto the user's view. An extra feature being worked on for a later release of the application is also created with the help of the jqPlot -plugin for jQuery, making delivery of visually appealing graphs on a <canvas /> element extremely efficient as all the data required comes in a simple JSON array format.

```
<div class="table-name">Table name:<br />   <span style="font-weight:bold;color:#3333CC;"><?php ech
<div class="column-count">Columns:<br />   <span style="font-weight:bold;color:#3333CC;"><?php echo
<div class="free-desc">Description:<br /><div style="margin-left:10px;width:140px;"><?php echo $table['t']; ?></di
```

Figure 7. The difference of tidy classes in HTML markup, versus inline CSS.

Local caching of certain data retrieved through the EVE API – only retrieve the data again when necessary and when the API's cache timers have expired – will also reduce bandwidth usage significantly. One final optimization step which technically supersedes necessity is to entirely avoid using inline style definitions in an HTML document, and always provide specific style definitions through HTML-element ID or class references in a .css file (see Figure 7). The amount of times a set of style definitions causes bandwidth usage is minimized, and only the class keyword for the given set is used multiple times.

4.3 User Management and Access Control

The Moon Manager application deals with user authentication and comparing against access control lists (ACL) already in the bootstrap. At the end of the bootstrap file (index.php), the Zend Framework FrontController is called and a number of plugins are registered. First the "Moonmanager_Controller_Plugin_Auth" is registered. It has been designed to ascertain the validity of the user's credentials and his/her assigned user group. The Zend_Auth class is called within the plugin to fetch the information of the current user from the session namespace, and a selfmade adapter compares the user's credentials to the application's database. The Zend_Auth various success and failure result constants are then used to determine the validity of the authentication. If a

user is found and credentials match, the Zend_Auth session namespace is populated with user data and the session ID is regenerated. Otherwise the user is classified as a guest user with access to nothing but the login screen.

The authentication process includes a query to the web server's main database holding the registered API keys for all users. In the future this process will be dealt with by integrating a Single Sign-On (SSO) system that CCP Games has published for use with third party EVE Online applications [23]. This will initially be integrated on the alliance web server separately from this final year project. For now the database must be checked for existing forum accounts and whether an account has valid API credentials confirming they have characters in a corporation in Fatal Ascension and what viewing roles they have to starbases. The EVE in-game settings for "starbase technician" and "equipment manager" roles can be queried from the API, and this is partially used to determine user groups in the Moon Manager application. A user that does not have up to date and functioning verification codes associated with their API keys, or who lack starbase roles in their corporation, are automatically stripped of access roles in the application. Administrators are exempt from this functionality as they are alliance appointed officials, and their access rights may only be edited by select chief administrators (also called superusers). Administrators also have a separate interface in the application where a user's access rights may be modified to represent for example alliance fuel technicians, and the API keys for corporations may be monitored.

After the auth plugin has completed its checkup, another plugin called Moonmanager_Controller_Plugin_Acl is registered. The ACL plugin then compares the data instantiated by the auth plugin and the intended destination of the user to the access control list composed in the local extension of the Zend_Acl class: Moonmanager_Acl. The user's destination is derived out of the 'controller' and 'action' variables drawn from the HTTP request object. These together form the main resource and activity measured within the ACL. All access is defined by the user's user group. For example a guest user has access to nothing but the IndexController resource and its indexAction. A director of a corporation has access to everything but the AdminController resource, while administrators have access to nearly everything with a few exceptions. There will be a few select chief administrators who will have access to everything, including the production test page and a special resource "appointAdmins", only to be queried when an administrator enters the personnel management section.

The ACL essentially provides the capability to hardcode any singular allowed or denied resource and action to any given user group. It is important to remember to add any new subpages and controller resources to the ACL after they have been implemented. Otherwise the system will automatically determine that nobody has access to the resource. It will in fact produce an error stating that such a queried resource is not registered in the first place even if administrators have correctly been granted access to all resources through an empty “$admin->allow()” statement. The Zend_Acl class which the Moonmanager_Acl is extended from also allows inheritance from one user group to another, meaning absolutely all assigned access rights do not have to be written out explicitly.

4.4 Moon Search Functionalities

All managemental functionalities in the Moon Manager application are located within the administrator section. This section is entirely handled by the AdminController class. The controller is held to a relatively small size and maintains readability even though it contains a plethora of functions. This is thanks to all computational processing being passed over to business model classes, thus minimizing the amount of raw code in the controller. The amount of viewable pages in the section is not extremely large, but the number of different functionalities on each page is numerous. One of these subsections is the user management portion, which has already been discussed in the previous section of this thesis.

The part of the Moon Manager application that makes it distinctively stand out from the old application used by the Fatal Ascension moon team is the plethora of search algorithms included. The old application employs only basic database searches and value modifications, projecting data only as static lists. Thus the administrators have had to go through all moons in a region by hand in order to locate matches for reaction setups. This may often lead to lengthy search operations where time and effort are essentially wasted on managemental background work where the same result can be achieved much faster through the use of highly diversified search algorithms.

The first and foremost subsection in the administrator area is the page for listing and sorting moons. This page provides search filters for finding a specific desirable moon. The main criteria for finding a moon concern the moon’s location. This can be sorted by

filters for the region, constellation and even specific system the moon is in. These location lists are grayed out and only become available one by one after a specific surrounding area has first been selected, so a constellation cannot be selected until a region has been specified, and a star system cannot be selected before a constellation. This however can be overridden through the use of the search box located above the location specific selectors. The search box populates a dropdown menu through an AJAX call after user input into the search box in real time. When a search result is selected, the webpage recognizes whether it is a region, constellation or a star system, and automatically fills in and enables all appropriate location selectors. This is also true for the secondary set of search filters containing information about the moon mineral deposits that can be found on a specific moon. After the search filters have all been set to desirable values, the administrator is presented with the option to display the search results in any one of six additional search lists. Once at least two of these lists have been populated, the administrator can either manually look through the lists for potential reaction setups, or they can process an additional search based on these secondary search lists and a final set of filters.

The additional search on the moon searching page produces an ordered list of suggestions for reaction setups containing moons with all the desired mineral deposits. The list will be ordered based upon the final set of filters. The options on these filters are for example to order the resulting list by a weight factor prioritizing either absolute proximity of all the mineral deposits in a network, absolute physical proximity measured from the star systems' X-Y-Z coordinates in the EVE universe, or even by overall proximity in relation to a certain origin point (a star system) that can be determined separately. During initial testing of the algorithm behind the matchup search, the execution time of the query lasted for tens of seconds, and extreme cases could use up to two minutes to finish a request to the server. This was unacceptable and improvements had to be found. After several evenings of refining and optimizing the formation of the algorithm – partially through trial and error – the execution time was greatly diminished and overall search time was cut down to merely a few seconds. This was acceptable and work could continue on the rest of the application.

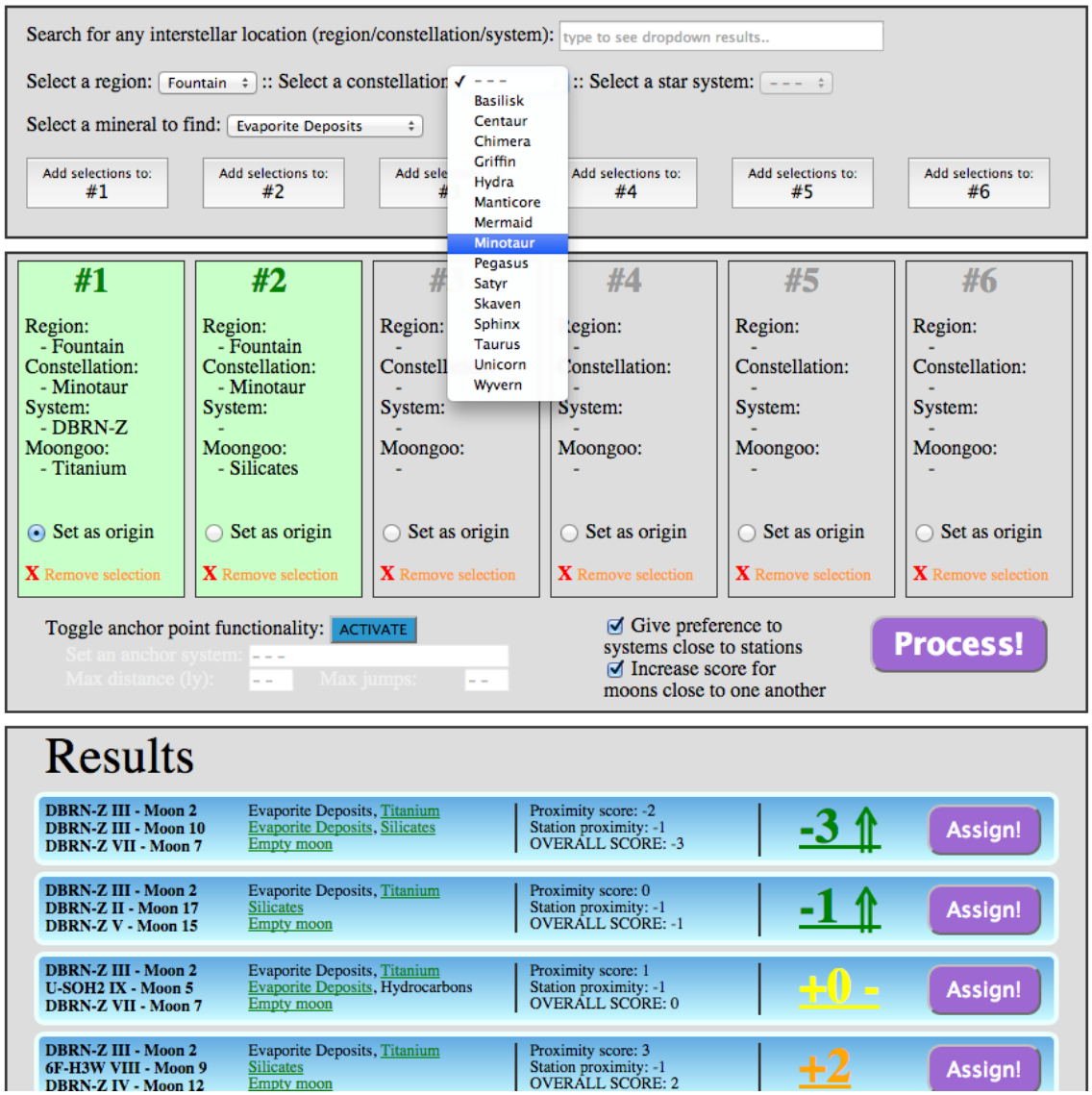

Figure 8. One of the search sections in the Moon Manager application.

The search algorithm mapping the matchups first declares requirements for all the moon minerals that have been selected and initiates a number of additional functions looping through first all, and then a series of fewer moons. By also performing the search for a lesser amount of moons than desired, search results can also be provided for possible outcomes where deposits are not available for all desired moon minerals. Thus corporations and individuals may still be presented with moons for the majority of the moon minerals required, and they simply have to import some of the materials required to complete reaction setups. Within the secondary functions a number of recursive loops for buildup of database queries is processed. The recursion involves a function dynamically building several temporal instances of the database table “systemRels” containing all information about the connections between star systems, be it

through regular stargates or by jump bridges that can connect two distant star systems to one another. Jump bridges require no pre-existing connection between two systems, but to activate the bridge a module must exist at both ends of the link in order to provide the considerably much higher energy levels required compared to a regular stargate. The query is gradually executed once and results are scored with pre-determined multipliers in accordance to given criteria in the search filters. The results are then sorted for data projection and the active function calls itself with new altered parameters for the next iteration. The recursive calls continue until 50 or more total search results have been collected or until no more results can be found for metal moons in the search set. A maximum of 50 search results are displayed to the user (see Figure 8).

The heaviest test runs of the algorithm have produced queries longer than what can fit on two full computer screens when printed out. Several such queries are made to the database for each reaction matchup search. Currently the algorithm also ignores all moons which are already occupied by a control tower. In a future revision however also occupied moons may be included in the search, and administrators will be able to distinguish the search filter even further by whether to include towers already occupied by the corporation the search is being made for, or if all moons regardless of being occupied should be included. A similar search is also employed on another administration page where the administrator can directly perform a search by inputting a number of moon minerals to search for and criteria for limiting the search area. A final search criteria worth mentioning is the ability to have the search only include systems in which Fatal Ascension holds sovereignty (this is related to reduced fuel consumption for control towers in sovereign space). The sovereignty clause only affects the destination systems in each recursion and does not exclude systems from simply being a part of the connection chain, making it particularly useful as an entire constellation in the Fountain region is permanently controlled by a non-player pirate faction.

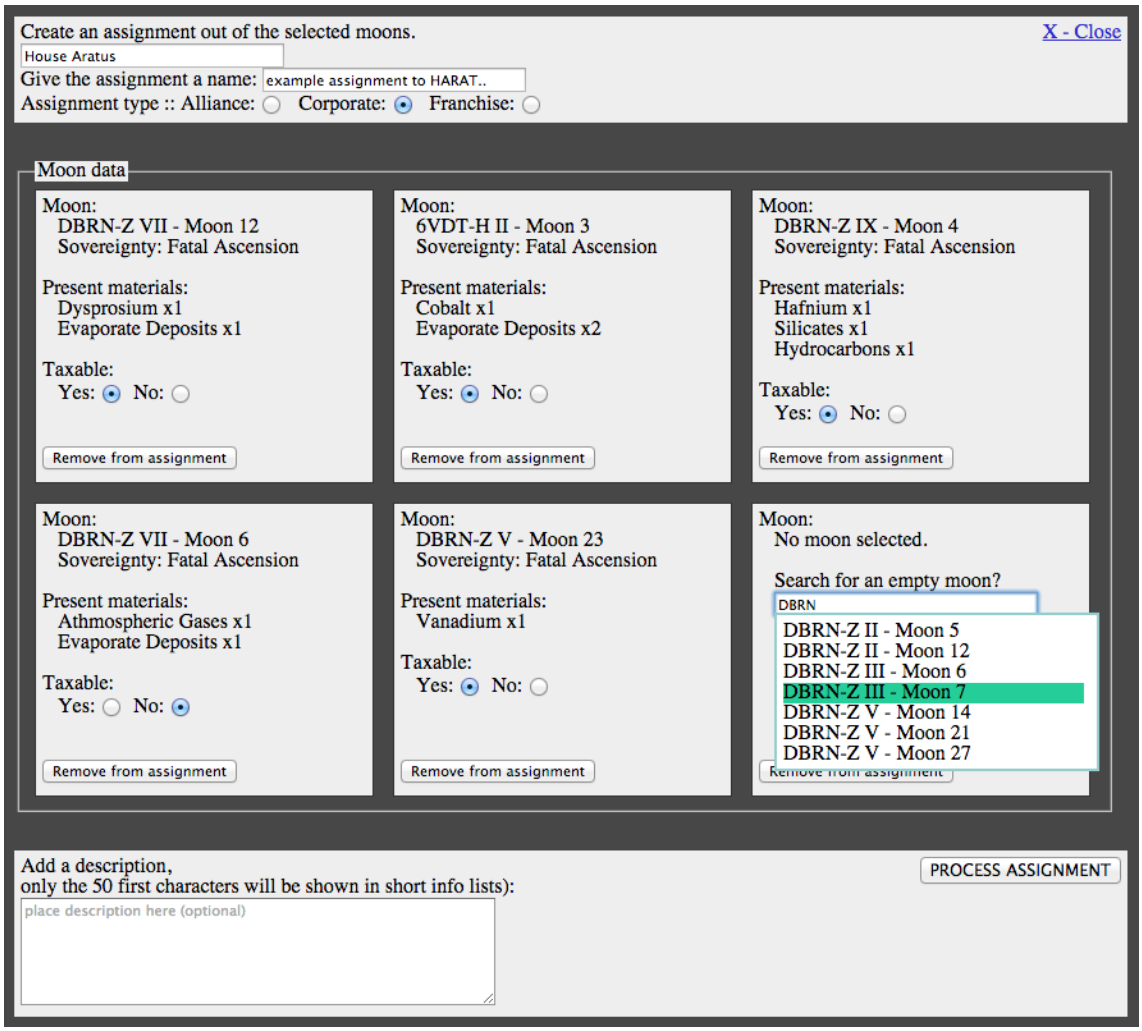

Figure 9. An assignment form presented in the moon search section.

Supplementary functionality found on the same pages with the matchup searches include a small overlay with a form enabling the administrator to directly assign the designated moons to an entity of choice (see Figure 9). This overlay always requires that an assignment set is connected to a corporation, additionally connecting the assignment to an individual (in case of a moon franchise assignment). If the assignment is designed to be used for alliance income, it will be allocated to a special administrative corporation in the alliance: in this case ‘Fatal Holdings’. In special cases where moons are alliance property and for other undisclosed reasons, a moon may be marked as not being subject to taxation. This is done with a simple checkbox provided in the assignment overlay and can also later be adjusted in the assignment listings.

4.5 Other Restricted Subsections

As previously described, administrators are provided with a personnel management area where roles can be verified and special permissions can be assigned. When entering this area an administrator will first be presented with two lists in side by side columns, portraying the most recent visitors to the web application and a list containing the most recent additions and changes to roles. More extensive views for the two lists are also available. At the top of the page a tab navigation allows an administrator to see for example complete lists of all personnel and base access right codes as well as roles internally assigned in the application. A second tab contains a search box for finding a user. Selecting a user from the dropdown list will enable a view providing role editing options. Clicking a user in any other tab automatically changes tab and opens the same view. Possible roles that can be assigned to a user include starbase technician (default value), director, alliance fueler, admin, and “banned”. The select few chief administrators will also appear as regular administrators within the system, they simply have the extra ability to select to promote or demote a regular administrator. For a regular administrator other administrators will have their role selector grayed out and functionality disabled (extra checks also prevent changes on the server even if a brute force change was attempted).

The ability to easily update the application’s database is important in the event of future updates to the EVE Online star systems. These could be changes such as the further addition of new star systems or added/changed connections between stars. For this purpose a number of functionalities have been pre-built to allow for scanning through a new rendition of the EVE static DB and applying any found new occurrences to the Moon Manager database. As jump bridges are player controlled entities – which may be taken down and re-deployed elsewhere at any given time – a tool has also been constructed for keeping track of the existing connections and altering them at will. The site updating subsection also includes the tools to monitor error messages generated that any user may have encountered during regular use of the application. These error messages will always precisely describe where they have been logged in the system, for easy updating and error management corrections.

4.6 Moon Assignment Listings and the Starbase Center

All registered users have access to the moon assignments section in the application. Here users will be able to view both their personal and corporate moon assignments. This also applies to certain users that do not have starbase technician roles in their corporation. If these users have the said roles in an alternative corporation, they are allowed to see all towers in that corporation instead. Administrators will not have any other listings available in the regular assignment listings section beyond their own corporation and possible alternative corporations. Administrators also need to be specifically assigned the alliance fuel technician role in the system to be able to view details about Fatal Holdings control towers. This can also only be appointed by chief administrators, who themselves have the possibility to view the details on any corporation's control tower details at will - such data will only be loaded per request through AJAX in case of corporations other than a chief administrator's own corporation, alternative corporation, or Fatal Holdings.

At the top of the page users with access rights to moons in multiple corporations may use a selector to choose which corporation's assignments they wish to view. By default a corporation is selected according to the user's main character selection on their forum account. The initially visible list of assignments will display only the assignment ID, the amount of moons and which systems are involved, along with a short name/note the processing administrator has given the assignment. Users can also set a very short name/note for the assignment to be visible directly in the list. When the mouse pointer is moved over a list item it is highlighted to indicate further details may be viewed. When the list item is clicked, an information box is animated in-between list items showing more detailed data for the assignment (see Figure 10). This extra data includes details such as specific moon locations, taxation information, and details about which administrator and when last processed the assignment or made changes to it. A colored indicator is also included for whether each assigned moon has a control tower deployed on it.

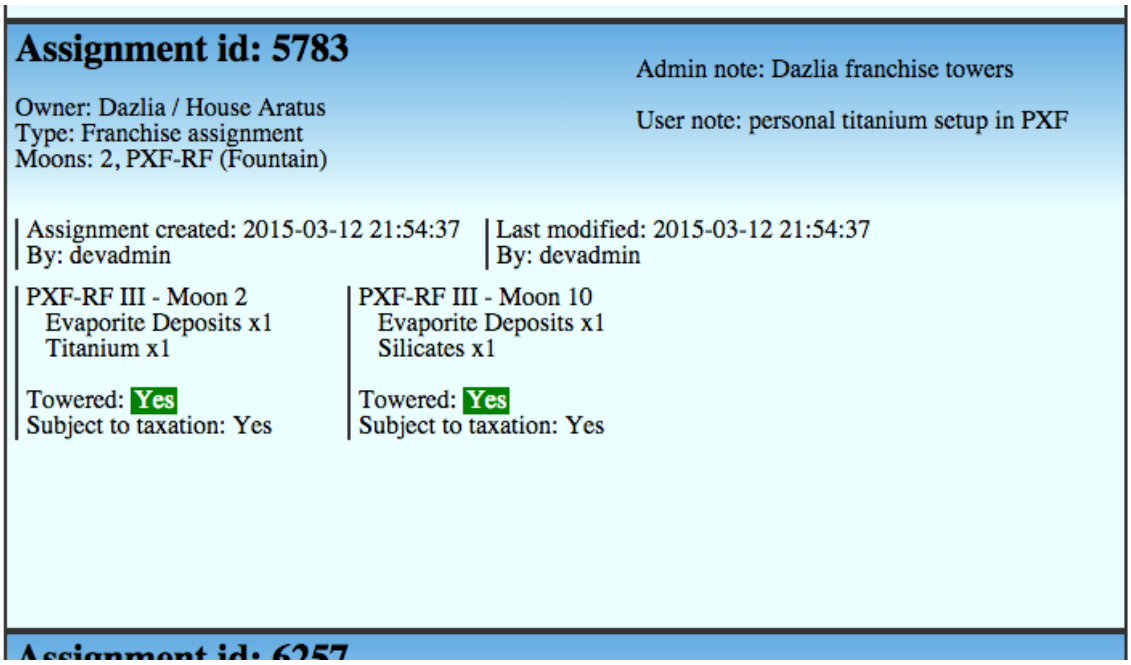

Figure 10. The Moon Manager assignment page with details displayed for an assignment.

The assignment listings page also has a button for requesting a new assignment. When the button is clicked, an overlay appears over the assignment list. This overlay includes a request form users may fill in with moon mineral types (including empty moons with no mineral deposits), location restrictions, and a free text field for additional information. Once the request is submitted, it will appear to administrators in the “manage requests” section. When an administrator views a request, it will have a button to reserve it as being processed by the administrator in question. This way two administrators will not unknowingly work on the same assignment request simultaneously. The user who made the request will also see it as being processed. When a request has been claimed by an administrator, he/she can hit a new button that appears in order to transfer the request and all pre-selected data over to the moon search view for administrators. There the request can be processed as a regular assignment. At this time the administrator will then have to go back to the request management view and mark the request as either fulfilled or rejected. A note can be added to provide further details and a response to the requesting user.

The starbase center is the section expected to be most used by regular users. In this section of the application users are presented with control towers their own corporation (or a corporation they have viewing rights for) has deployed. Towers that have been at least “anchored” in orbit of a moon will be displayed, regardless of a tower’s status be-

ing online or offline. An offline tower will display a red tinged background, as being offline leaves the tower entirely unprotected by its force field. A tower can only be turned online if it has sufficient fuel. For this purpose the starbase center displays real time information about control towers, especially regarding their fuel levels. A control tower may at most hold roughly 38 days worth of fuel in its hold.

Each control tower in the list is named accordingly what the tower has been manually named within the game, and towers are grouped according to assignments that the relevant moons belong to (see Figure 11). Other specific data displayed for each tower includes the amount of materials gathered in each silo, timers indicating how much time is left before fuel runs out or silos become full, as well as the amount of “Strontium Clathrates” that are stored in a secondary fuel compartment in the tower. Strontium is used to reinforce a control tower’s force field into an invulnerable state for a limited amount of time in the event of coming under attack.

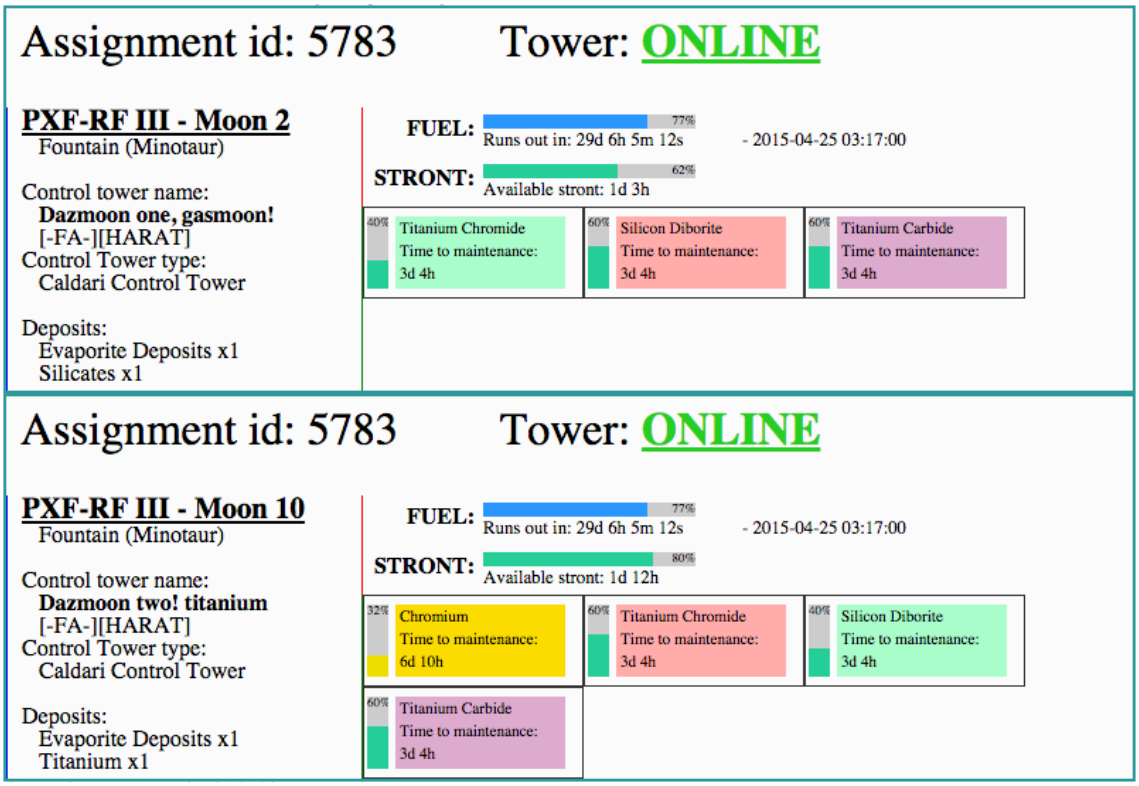

Figure 11. The Moon Manager starbase center with tower details.

When a moon assignment has been confirmed, it does not automatically show up in the starbase center section, unless there are already towers deployed on any of the moons. Each deployed control tower owned by a corporation will be displayed in the

starbase center, regardless of being deployed on a moon that is part of an assignment or not. This allows corporations to also monitor control towers that may be in low security space or elsewhere outside alliance territory. Also alliance owned towers that may be located outside the Fountain region will be shown for the Fatal Holdings corporation and anyone with the alliance fuel technician role.

4.7 Additional Requests from the Client

Many extra features requested by the client have already been described, but multiple requests came in relatively close to the end of the development process. It was possible to still add some of these requests to the beta release of the application, but a major portion of the requests had to be postponed for future releases of the application. Many of the postponed features are described later in this thesis.

A suggestion that was possible to easily merge into the application before the initial version was delivered to the client was the implementation of a section displaying any hostile towers that had been detected by players. Users can enter the application and navigate to the hostile towers section and enter details for a discovered hostile tower, which fleet commanders can then set out to vanquish. The listing of the hostile towers entered into the application is the only additional feature publicly visible beyond the login screen, and any player registered on the Fatal Ascension forums may file a report for a hostile tower. For this purpose the “fleet commander” user role was added for internal use in the application.

Users with any of the fleet commander, director or administrator roles may select a hostile tower in the list and report it as having been put into reinforced mode or destroyed. A tower enters reinforced mode when attacked with a sufficiently large fleet to cause the automated reinforcement after a certain amount of damage to a tower has been dealt. If a hostile control tower is reported as having been put into reinforced mode, a countdown timer will be associated with the said tower. This timer will be displayed at the top of the public section of the application so that anyone may see when they can expect a fleet operation to take place. A similar timer may also be initiated for friendly towers if a control tower is attacked by a hostile entity.

Another requested feature that was included in the first version of the application was the ability to add records for star system ownership. A star system in EVE Online is essentially denoted as being owned by the entity with an online Territorial Claim Unit (TCU) in the system. These TCUs are controlled by corporations, but the direct ownership of the star system is by definition related to the alliance that the corporation belongs to. A lone corporation that is not a part of an alliance may not deploy a TCU and turn it online, thus making alliance creation a necessity for claiming sovereignty in outlaw space. The fact that a star system is controlled by a specific alliance reduces fuel costs by 25% for any control tower owned by an entity within that alliance. In the Moon Manager application this is also related to whether or not a moon is actually subject to taxation. Even though the actual taxation fee calculations will be automated in a later revision of the application, it is imperative for administrators to be able to determine whether control towers are in Fatal Ascension's sovereign space, as no moons located outside Fatal Ascension's domain are subject to any form of taxation.

5 Results

5.1 Finishing the Application

All planned features for the application have been finished and seem to be in a functional state. Additional features that came up during the development have been added either partially or fully depending on the importance of their implementation. A large number of supplementary functionalities and entirely new features that have arisen during the development process have been transferred to the future task list, and will be included in later revisions of the application.

At the beginning of this project there was a fair amount of uncertainty whether the final application would end up being used at all. This was due to political strife within the EVE Online game and the implications these conflicts had on the future of the Fatal Ascension alliance. There are also other reasons for these concerns, which due to their delicate nature must be kept secret. The application ended up still getting permission from all parties to be created and certain third parties within the EVE community exclaimed that a use would be found for the application regardless of the future of the alliance.

The application has been delivered to the client for beta testing and improvements to initial features continue to be developed. While beta testing the client's representatives will use both the old and the new application to store moon assignments. This is to ensure that data is properly stored in the old system, while confirming working functionalities and completing performance test on the new Moon Manager application. At the time of writing this all systems appear to be functioning properly excluding an error caused by user session storage relating to the late addition of the hostile tower feature. This is being fixed as soon as possible.

5.2 Analysis

The aim of the project was to develop a working management toolset for the Fatal Ascension moon team administrators to use as a toolset in their daily work. The application was to replace the old application that had been deemed to provide insufficient functionalities to the aid of the moon team.

The project proceeded step by step in a fairly smooth fashion. The original timetable set for the development process underwent heavy changes due to waiting times and unplanned ideas arising for new features. The application itself was however intended to be done before the end of May, but this had to be moved up due to previously unknown deadlines coming to the awareness of the developer. These changes to the timetable were not directly caused by the client. There are impending changes to certain game mechanics relating to how sovereign space is controlled and maintained. The resulting impact on the geopolitical landscape in EVE is that many widespread coalitions of several alliances are forced to move in more tightly together. For the client this means that due to the impending relocation they are faced with, they cannot properly commence testing of the application yet. [24.]

At the same time beginning work on writing the thesis was delayed largely due to uncertainty related to communication with select external parties. The initial designs for the application also changed a lot during the first days of the development process, but the effort put into refining the plans early on made the rest of the project easier to grasp as a whole.

A number of functionalities and sub-features that were included in the first plans for the application were not implemented at all and have also been removed from future plans. Upon reviewing the plans during the development process some features were deemed entirely unnecessary and others already exist within other third party applications for EVE Online, thus making their inclusion obsolete. These discarded functionalities also included a number of search algorithms that were eventually deemed to be merely overlapping other functionalities, nearly duplicating their results. Choices relating to whether or not to include a functionality often took a lot of time away from the actual development process. From another perspective this may be a good thing, as unnecessary features were left out, not wasting time on developing something in vain. One such feature would have been to include search algorithm patterns for alchemy. Alchemy is a way to turn certain moon minerals into other types with increased cost and extra materials. This was seen as unnecessary due to the fact that barely any residents of outlaw space (such as Fatal Ascension) have to rely on alchemy thanks to the abundance of moons available. Alchemy is mostly utilized by players dwelling in low security space.

The most time consuming task in the entire development process was the creation and refining of the network search algorithms. A lot of time was invested into getting very precise search results and cutting out overlapping results and execution time factors. Due to their in-depth complexity the search algorithms had to go through a lot of trial and error development, but the end results are satisfactory.

The development process was undertaken utilizing agile software development paradigms. This helped in maintaining good development speed for the application as well as speeding up the writing process of the thesis. Developing features one at a time into functional form and only having to modify them at later stages if necessary is much more efficient than potentially leaving something halfway done in order to first finish another feature. If something is left unfinished, it may potentially end up hampering testing of other features. The only scenario where such a secondary prioritization should be undertaken is if the secondary task is a dependency for the initial task. The project also benefited from personal investments of the developer. A magnetic flipchart along with an Open Atrium installation for task management were purchased and utilized for project management. The flipchart proved essential for easy visual access to various structure designs and core plan listings.

The choice to begin by designing and constructing the MySQL database proved to be an excellent foundation for the project. The database only had to be directly altered twice, excluding minor tweaks and enumeration value additions. This aided in keeping the project directives clear in terms of the types of data that would be used in the development process. The structure of the rest of the application was easy to build up on top of the structure of the database. The structure changed a lot during the first few days but was then fairly consistent throughout the rest of the development process.

Towards the end of the development schedule cuts had to be made on the amount of extra requested features that could be implemented in the first version. This well represents an agile development schedule, where features have to be either implemented properly, or they will be pushed to the next stable release version of an application. The amount of last minute work caused some stress to the developer, but properly formulated plans and workflows kept the development process of the Moon Manager application feasibly controllable until the end. One major cut that had to be made due to lack of time was the implementation of the Backbone JavaScript framework. As jQuery was already a familiar tool for the developer, it was deemed unwise for the scope of this

project to force the use of Backbone with a learning curve that may have pushed back development progress by several days.

5.3 Future

The first version of the application meets all demands for the initial agreement. The future plans for the Moon Manager application involve a great deal of further development. The current list of features and additional functionalities are not yet ordered into a specific priority order. Plans are in place however to change from using the Open Atrium platform into the use of Jira Agile for project management [25]. Jira is not a free software, but as perpetual use license costs when personally hosted are only $20, queries will be made as to whether Fatal Ascension would be willing to host the service. This would allow the Fatal Ascension IT team to collaborate and keep track of any and all development processes that volunteer members were doing for services within the alliance, while the alliance would provide them with a project management toolset.

The Moon Manager application will only incorporate the Fountain region in its first release. Once further development commences, however, the entire EVE universe will eventually be incorporated into the system, including all of the 7,800+ star systems and between 150,000 - 200,000 moons. The purpose for enlarging the area of space covered by the application is to possibly enable its use for entities other than Fatal Ascension (either as a single installation for allied entities or as separate instances). If other affiliated alliances would also gain access to the application, this would also require alterations to the access control lists and tweaking of the data actually visible to each admin. Administrators might for example be able to view assignments and moons allocated to anyone, but only have editing rights and allocation privileges to star systems their own alliance holds sovereignty in.

A feature is planned that will enable administrators to send multiple search results as suggestions to a user. The user will then view the suggestions on a page included in the moon assignments and requests section. After accepting a specific set of moons, the setup will be confirmed as an assignment and a confirmation notice will be sent to the admin. This will include every single moon in any of the included suggestions to be tagged as reserved until either a 24 hour period passes (subject to change) or the user

accepts a setup suggestion. Reserved moons will then not show up in any reaction matchup searches.

When a user views an assignment, a button will be added to show an overlay giving a suggested setup for moon harvester arrays, silos and reaction arrays to be used in the assignment, based on user input for what racial control tower the user would prefer using on each moon. There is a difference between control towers for each of the four main factions in EVE, with certain tower types allowing for more industrial modules, and others emphasizing the ability to fit more armaments onto the control tower. The setup suggestion will take a variable input in the form of what specific reaction a user wants to deploy for the assignment, thus also including input silos for any material not present in the assignment, thus having to be imported.

A regular user of the old application has made a request for a profit estimation to be included for each assignment. This has been approved to be included in a later release. The way such a profit estimation will be calculated is through another third-party web application called EVE Central. EVE Central is a community collaboration effort to provide constantly up to date information about market prices all over the EVE universe. Any user can open the in-game browser in EVE and navigate to the EVE central website. Here they can find a script to run while having the EVE Central website open in their in-game browser. The script data mines through hundreds of market orders and stores them onto the EVE Central servers. The web application then provides up to date market information about sell and buy orders to anyone who seeks such knowledge. The way this helps the Moon Manager application and the intended moon assignment profit estimations is through the XML API also provided by EVE Central. The basic functionality of this API is very similar to the actual EVE API provided by CCP Games, using very similar queries and results. However, data provided is limited only to market information. Appropriate average values will be retrieved for prices of control tower fuel and market values for the end products for various moon mineral reactions. These values fluctuate constantly, so a rough moving average has to be calculated from data collected over longer periods of time. This data will then be used to provide estimates for expenses and profits associated with each moon assignment setup for a user’s convenience. [26.]

An information message system will be implemented in a later version of the application. This will be implemented in the form of short messages appearing in a corner of

the application layout. The purpose of these messages is to inform users and administrators about events relating to matters they are involved in. For regular users this may for example involve a resolved moon assignment request, an alert for fuel about to run out in a control tower, or a suspected siphoning alert. Siphoning units are modules a hostile player may deploy in close proximity to a control tower. The siphoning units will gradually steal moon mineral resources from the control tower's silos until they are removed through collecting them into a ship or destroying them. The alert will be produced through calculations made from the hourly cycles of material produced into a silo. If a silo has received less than the intended amount of resources, a suspected siphoning unit will be reported. A proposal has been made to also create a mobile phone application to support the Moon Manager application. This application would include push notifications from the server in order to alert users of for example control tower fuel running out or suspected siphoning units.

Late into the development process the client asked whether checking for the existence of jump bridges could be automated into the system through API pulls. At this time there is an uncertainty as to whether this data is available through the API or the recently added CREST API. The matter will be investigated though and if possible, the manual entering of jump bridge connections will be removed and automated functionalities implemented. There is, however, a concern relating to whether this feature will require a corporate level API key to always be valid. For this reason the investigation will include the possible outcome of having both automated functionality as well as the possibility to override the data pulled from the API. In addition regardless of the outcome, a history mapping of jump bridge locations will be added.

6 Conclusions

This thesis has portrayed the planning, development and future plans of the ‘Moon Manager’ third party application for the game ‘EVE Online’ published by ‘CCP Games’. The client of the project was the ‘Moon Team’ of the player community ‘Fatal Ascension’. The application is both a content management system as well as a search tool to be used as a primary toolset in the daily tasks of the Moon Team administrators. The Moon Manager application was created in order to replace the old application used by the Moon Team. The application’s starbase section will also serve as a data projection and tracker toolset for regular users from the Fatal Ascension community as well as appointed alliance fuel technicians.

Several unexpected issues arose with the timetables originally set for the project. The project has been finished on time, but the client is presently unable to begin proper beta testing due to factors relating to the impending changes to the EVE Online game. The geopolitical landscape in the game will be vastly affected by the coming changes and thus the application is unlikely to be put into use until the game has been updated at the beginning of June 2015. Before entering proper testing, new moon data will also have to be entered into the system for the client’s new local area. There should also be time to enhance the visual looks of the application.

The client’s initial reactions to the application have been positive, but several suggested improvements and new features have been presented. These additions have been added to a backlog for future development, which will continue during the application’s beta testing. The application will provide many new features and tracking possibilities for the client and other users. Many of the tasks that can be completed with the application have previously been done entirely manually. The client has actively taken part in open discussion during the development process and thus greatly aided in forming a better picture of the final application, beyond what could be interpreted from the initial plans.

References

1 EVE Online [online]. CCP Games.
URL: http://www.eveonline.com. Accessed 20 March 2015.

2 EVE Online Influence Map [online]. Eve-podcasts.com.
URL: http://evepodcasts.com/eve-influence-map/. Accessed 20 March 2015.

3 Moons [online]. Dotlan.net.
URL: http://evemaps.dotlan.net/region/moons. Accessed 21 March 2015.

4 Michael Littman. Intro to Algorithms [online]. Udacity.
URL: https://www.udacity.com/course/cs215. Accessed 21 March 2015.

5 Algorithms [online]. Khan Academy.
URL: https://www.khanacademy.org/computing/computer-science/algorithms.
Accessed 21 March 2015.

6 Lbackstrom. The Importance of Algorithms [online]. Topcoder.com.
URL: https://www.topcoder.com/community/data-science/
data-science-tutorials/the-importance-of-algorithms/. Accessed 22 March 2015.

7 PHP String Functions [online]. The PHP Group.
URL: http://php.net/manual/en/ref.strings.php. Accessed 22 March 2015.

8 Regular Expressions Quick Start [online]. Regular-Expressions.info.
URL: http://www.regular-expressions.info/quickstart.html. Accessed 22 March
2015.

9 Matt Doyle. PHP Recursive Functions: How to Write Them, and Why They're
Useful [online].
URL: http://www.elated.com/articles/php-recursive-functions/. Accessed 22
March 2015.

10 Peter Bright. HTML5 specification finalized, squabbling over specs continues
[online]. Ars Technica.
URL: http://arstechnica.com/information-technology/2014/10/
html5-specification-finalized-squabbling-over-who-writes-the-specs-continues/.
Accessed 23 March 2015.

11 HTML5 Introduction [online]. w3schools.com.
URL: http://www.w3schools.com/html/html5_intro.asp. Accessed 23 March 2015.

12 Thread (computing) [online]. Wikipedia, The Free Encyclopedia.
URL: http://en.wikipedia.org/wiki/Thread_%28computing%29. Accessed 23 March 2015.

13 Ajax (programming) [online]. Wikipedia, The Free Encyclopedia.
URL: http://en.wikipedia.org/wiki/Ajax_(programming). Accessed 23 March 2015.

14 How jQuery Works [online]. The jQuery Foundation.
URL: http://learn.jquery.com/about-jquery/how-jquery-works/. Accessed 23 March 2015.

15 Features of the Less Language [online]. lesscss.org.
URL: http://www.lesscss.org/features/. Accessed 24 March 2015.

16 Zend Framework [online]. Zend Technologies Ltd.
URL: http://framework.zend.com/about/. Accessed 24 March 2015.

17 Model-view-controller [online]. Wikipedia. The Free Encyclopedia.
URL: http://en.wikipedia.org/wiki/Model-view-controller. Accessed 14 April 2015.

18 API [online]. EVE - Development Network.
URL: http://wiki.eve-id.net/API. Accessed 25 March 2015.

19 XML API [online]. CCP Games.
URL: https://developers.eveonline.com/resource/xml-api. Accessed 25 March 2015.

20 APIv2 Page Index [online]. EVE - Development Network.
URL: http://wiki.eve-id.net/APIv2_Page_Index. Accessed 25 March 2015.

21 Memcached [online].
URL: http://memcached.org/. Accessed 25 March 2015.

22 Apache Module mod_expires [online]. The Apache Software Foundation.
URL: http://httpd.apache.org/docs/2.2/mod/mod_expires.html. Accessed 25 March 2015.

23 Single Sign-On (SSO) [online]. CCP Games.
URL: https://developers.eveonline.com/resource/single-sign-on. Accessed 29 March 2015.

24 Politics by Other Means: Sovereignty Phase Two [developer blog online]. CCP Games.
URL: http://community.eveonline.com/news/dev-blogs/politics-by-other-means/. Accessed 3 April 2015.

25 JIRA Agile [online]. Atlassian.
URL: http://www.atlassian.com/software/jira/agile. Accessed 9 April 2015.

26 EVE-Central: The EVE-Online Market Aggregator and Toolkit [online]. EVE-central.com.
URL: http://www.eve-central.com/. Accessed 10 April 2015.