|  |
| --- |
| **MARSE SEP2 S18 – Software development** |

****

**Mihai Draghiciu, 238971  
  
Eric Volmer, 273448  
  
Andrei Mungiu, 273473  
  
Ronalds Andris Kalnins, 260506**

**Supervisor: Troels Mortensen, Bo Brunsgaard Christensen,   
Ib Havn**



**[Number of characters]**

**Software Engineering**

**2nd semester, Spring 2018**

**DATE HERE**

Contents

[1. Introduction. 3](#_Toc532812164)

[2. Requirements 3](#_Toc532812165)

[2.1 Functional Requirements 4](#_Toc532812166)

[2.2 Non-Functional Requirements 5](#_Toc532812167)

[3. Analysis 5](#_Toc532812168)

[3.1 Use Case Description Diagram 5](#_Toc532812169)

[3.2 Activity Diagram 6](#_Toc532812170)

[3.3 Model diagram class 7](#_Toc532812171)

[4. Design 7](#_Toc532812172)

[4.1 MARSE design 7](#_Toc532812173)

[4.2 UML Class Diagram Design 8](#_Toc532812174)

1. **Introduction**.

The customer is a small group of around 17 people that coordinate their effort to mine cryptocurrency. The group is named Javelin Mining Project.

As the number of members is relatively small the information management is relatively easy to handle but recently more people have shown interest in the group and it looks like the number of members will grow therefore management will become cumbersome, time consuming and more open to errors.

The customer is the one that records all the information about mined crypto-currency, inventory in terms of GPU-s, motherboards and the information involving the date purchase, arrival date of the product, time of activity and this information is becoming difficult to maintain in its current state through Excel.

Another issue that needs to be addressed is communication between members since the nationalities differ meeting up is rather difficult so all the communication takes place online therefore the customer requires an implementation of a safe communication app in order to make the monthly meetings go smoother.

Delimitations regarding the application are:

1. Use other tools to record LIVE hardware (server) performance and keep this information separate.
2. It is not possible to display relevant market updates as that is a subjective matter.
3. Do not display graphical representations of data in an attempt to make the data more readable.
4. There is not enough time to work on automatic data inputs from external sources.
5. **Requirements**

The requirements of the application itself are as follows:

* Access and security (user separate access required, data editing access for administrators only)
* Users should have the possibility to access the available financial and inventory information from the project.
* Data visualization should be intuitive and easy to refer to in words.

The admin will be able to manage all the data in the database, which consists of editing, creating and viewing.

The user will be able to view the available information from the database.

* 1. **Functional Requirements**

The users and admin will have access to the application at any given time.

The functional requirements of the application will be the following:

**The admin:**

* The admin should be able to view the financial and inventory data.
* The admin should be able to chat with people who are online in the application.
* The admin should have ability to view the system logs regarding user login.
* The admin should be able to edit the financial and inventory data.
* The editing will happen in the same windows where the data is being viewed.

**The user:**

* The user should be able to view the financial and inventory data.
* The user should be able to chat with the people who are online in the application.
* The user should have the ability to view the system logs regarding user login.  
    
  1. **Non-Functional Requirements**

Non-functional requirements will be:  
  
Performance:

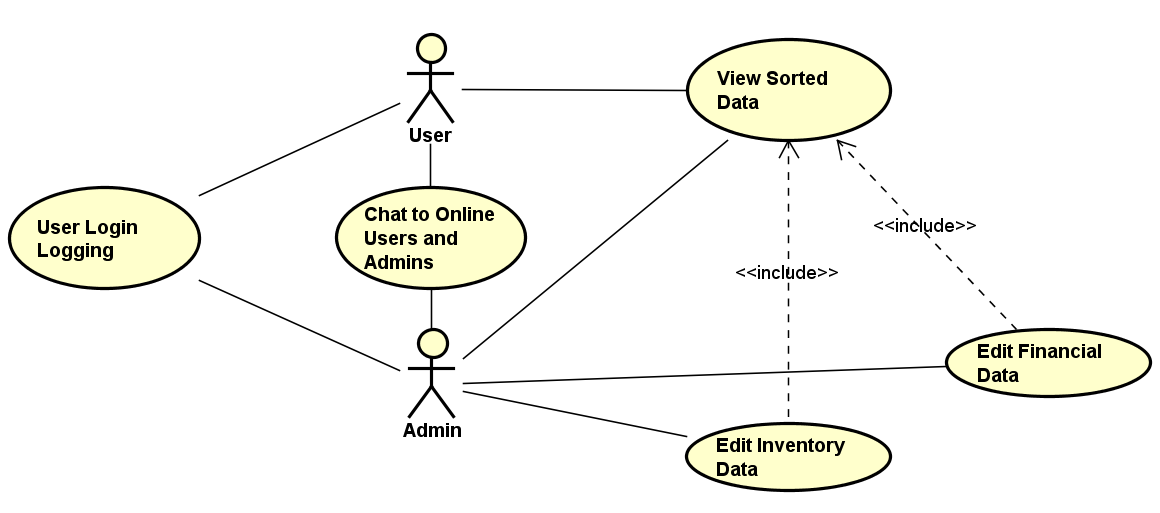
* At any given time, the system response time should be:

                   - Viewing data under 10 seconds.

-Editing data under 1 minute.  
  
     - Each method should have a full description of its use case.

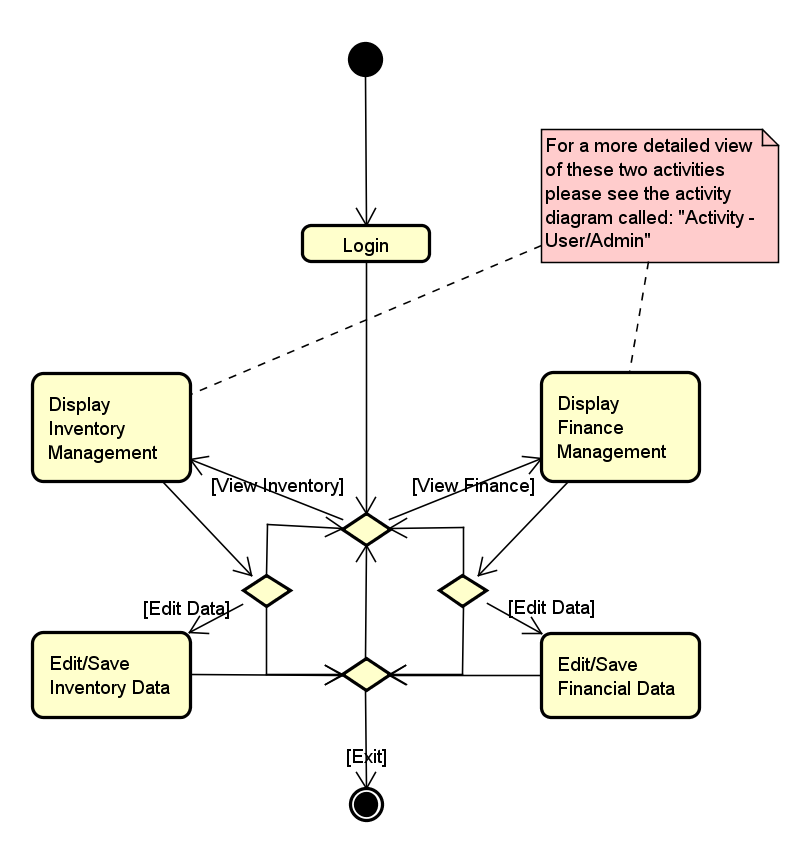
-    The mean time between failures is not more than 1 day.

1. **Analysis**
   1. **Use Case Description Diagram**



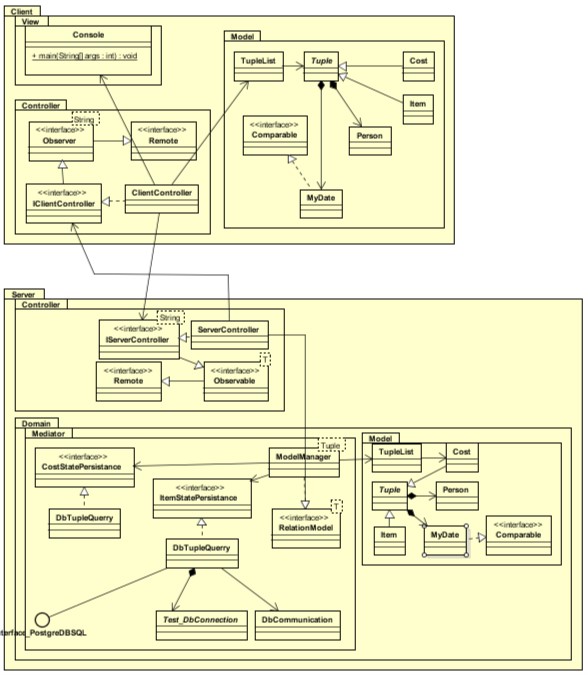
**User Login Logging:** Users log in will be logged.   
**Chat to Online Users and Admins:** Users and Admins can have a communication between each other.  
**View Sorted Data:** The user and admin can view the sorted data taken from the database which includes the inventory data and financial data.  
**Edit Inventory Data:** The inventory (gpu, cpu etc) data what is stored database, can be edited.  
**Edit Financial Data:** The financial data what is stored in the database, can be edited.

* 1. **Activity Diagram**



The activity diagram shows how the admin can interact with all the data what is stored in the database. The main possibilities for the admin is to edit and save financial data and inventory data. Also, to display the finance management and inventory management.

* 1. **Model diagram class**

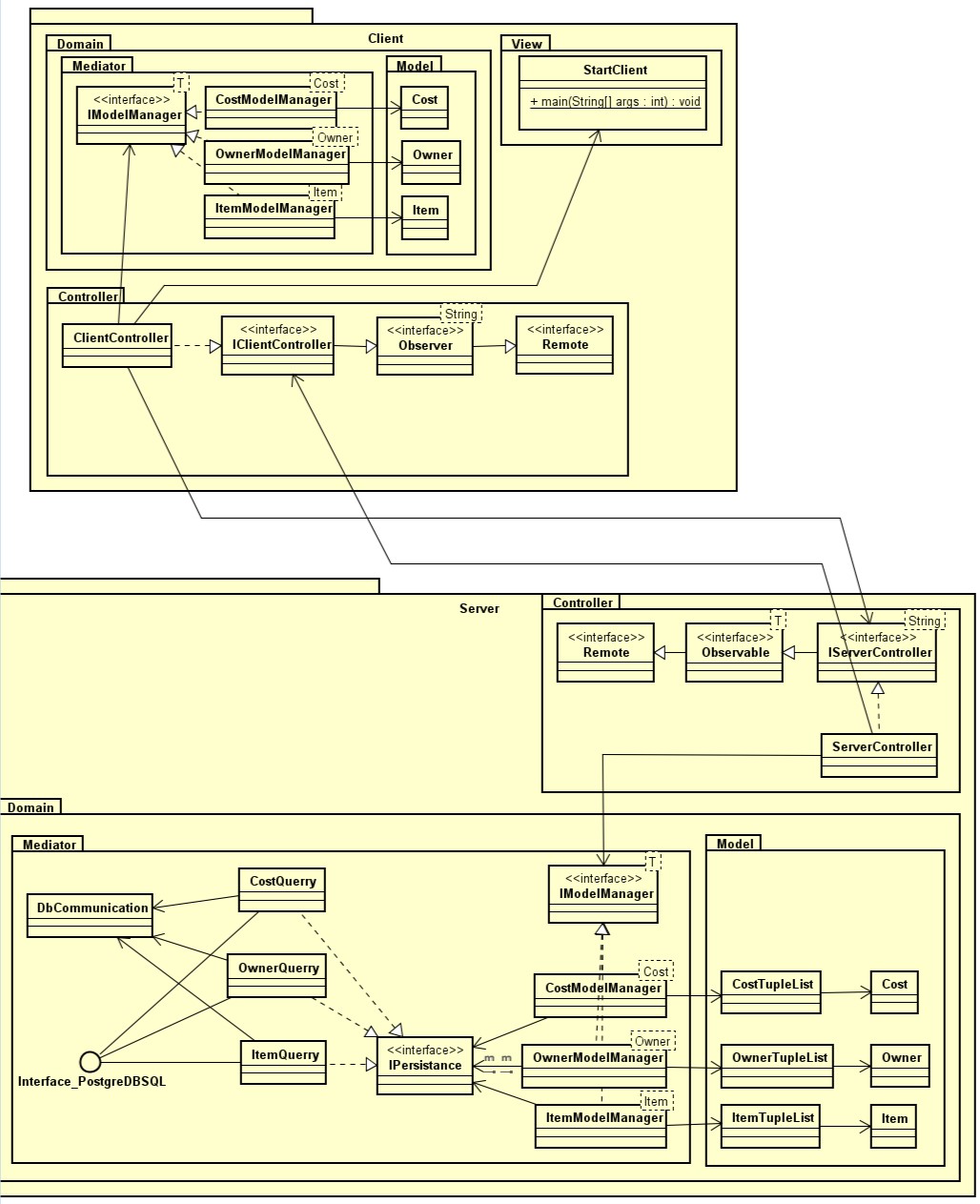
****

The diagram shows the connection between the database and the user required by the program to run and display information for the users. The classes have been separated and placed logically in order to have more convenient approach and for maintenance it will make it simpler.

1. **Design** 
   1. **MARSE design**

The project consisted of importing data from database to the application. The application has an ability for the users to communicate between each other while viewing or editing the data. The viewing and editing has to be fast and easy.  
 The program is based on two parts: **Java** (programming language) and **Database**(Postgres). Additional software was used for planning(Astah), also in order to merge and put all the code together from all the group members, GitHub was used for that.   
 **Java Classes.** In Java, the application was built with classes to access the database in order to import and export the data from database and display it for the users. In order to make changes to the database, there are methods to create, edit and display information.  
 **Database.**  
 The data inputted from Java will be sent to the database to store all the information inserted by the users. This will sort the input accordingly to corresponding tables. This will keep clean overview of the information in the database.

* 1. **UML Class Diagram Design**

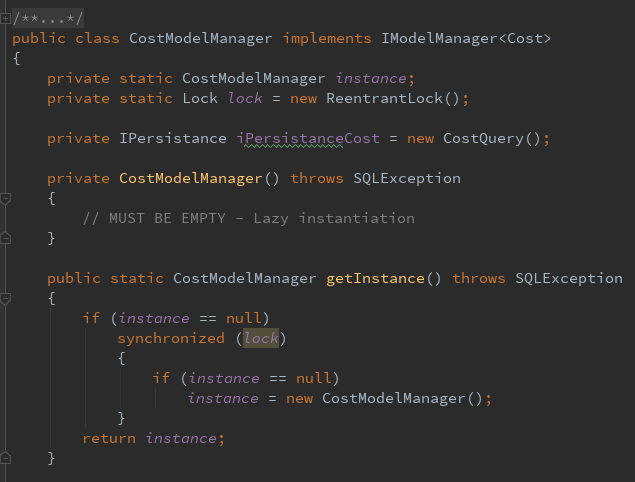
****

1. **Implementation**



(image 1)  
 Data and Model transfer and unification.

Creating an ArrayList<Cost> and inserting all the data from the model and the tuples and inserts them into the ArrayList<Cost> through a “while” loop in which a temporary array is created and returning it.



(image 2)

Class Method Cost Model Manager.

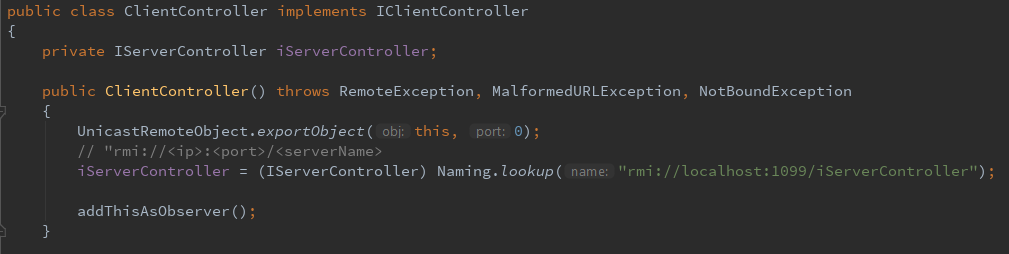
This class acts as an adaptor between the database and the server controller in the single point of merging data into the model. The class was implemented poorly at first and did not fit the functionality it required therefore after adapting a Singletons Pattern for it all the classes that implement the model interfaces have been adapted to Singletons.



(Image 3)

Class StartCLient.

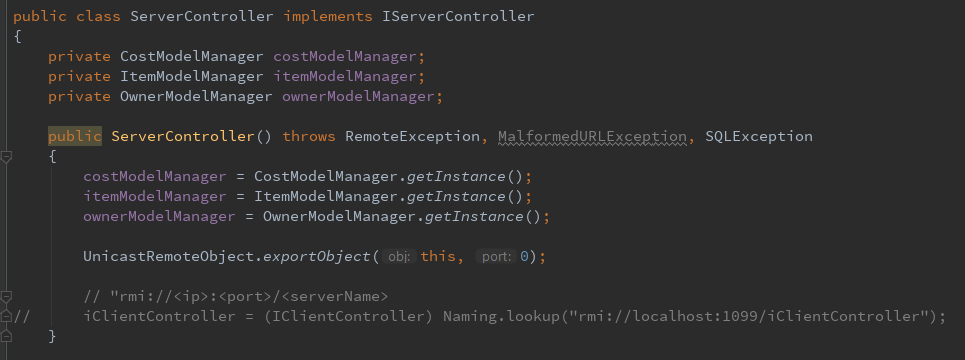
The client uses this class to request a cost relation from the Database. This class also contains the “help” request in order to show the user the list of available commands and what they do.



(Image 4)

Class ClientController.

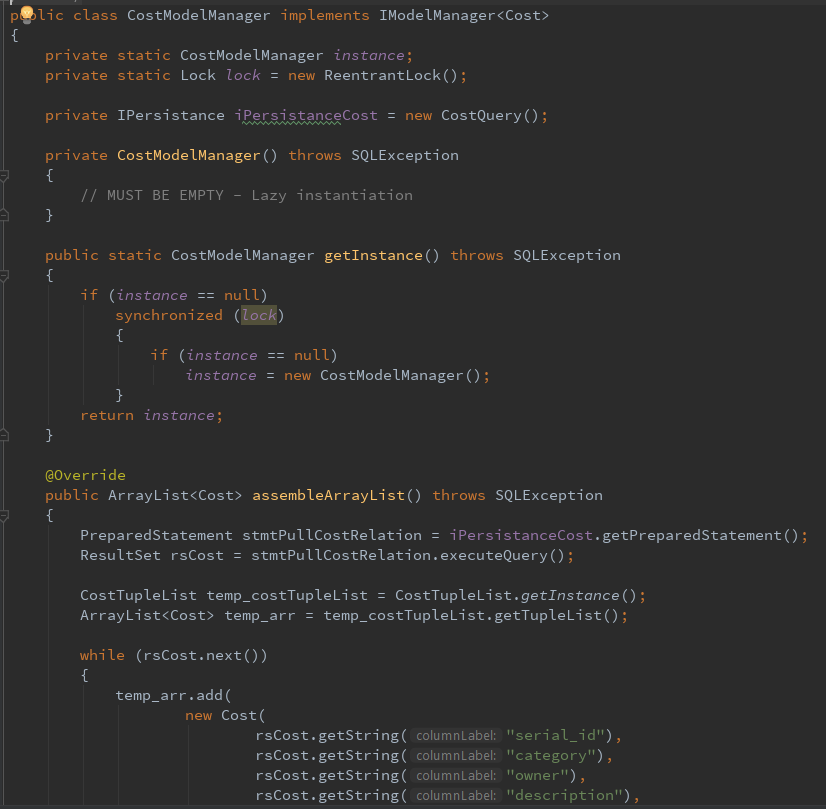
The client controller receives the request and forwards it to the server controller through remote method invocation. (RMI)



(Image 5)

Class ServerController.

The server controller receives the remote method call and executes the command.



(Image 6)

Class CostModelManager

The cost model manager receives the request and starts assembling objects by:

- querying the database for information.

- instantiating the model.

- looping through the ResultsSet received from the database and inserting it into the instantiated model.

- returning the instantiated Object with all the data stored in it.

RMI forwards the data to the client since it is serializable. The client receives it in its original shape since it also has a copy of the model.

**Test  
 Test Specifications**

**Results and Discussion  
  
 Conclusions  
  
 Project future**

**Source of information  
  
 Appendices**   
  
 The summary.  
  
The summary is the first-glance, which means that it has to include everything from the introduction to the conclusion and **Can NOT** include things that are not mentioned anywhere else in the report.   
  
 The main matter  
  
The main matter has to include everything from setting the stage in the introduction to describe the perspectives of the conclusions. It has to be understandable without reading the appendices.

The main matter must be divided into two or three levels (chapters, sections and sub-sections if needed). The levels must be numbered using Arabic numbers.   
 This has to give a good overview of the project and make it easy to find specific information. Report **Structure is super important.** The Head/Introduction  
  
**Introduction** includes background information, the problem statement, a justification of the project, a delimitation and so on.