ASSIGNMENT 1
(RECYCLING MACHINE)

CIS007-3 Comparative Integrated Systems

Laszlo Szoboszlai
1515931

Contents

Task description

A Recycling Machine is a machine where the customer can return empty bottles, cans and crates. The machine will check each item and return a receipt that lists the items inserted, together with their value. The company *Recycling Inc.* operates 25 recycling machines located all over the country. From time to time an Engineer comes to empty and reset the recycling machine. The Headquarter wants to collect usage data from the individual machines that they can analyse to add more machines (where needed) or remove machines where they are not being used. They also wish to remotely configure and monitor the machine via a Java GUI.

Planning

The project logically is separated into three parts in the following logical way:

- server part which controls each individual recycling machine in the background (backend)
- the client part which is a GUI connecting to the server
- and the administrator part which is another type of GUI to be able to connect to any of the 25 machines. See Figure 1- Architecture of the project.

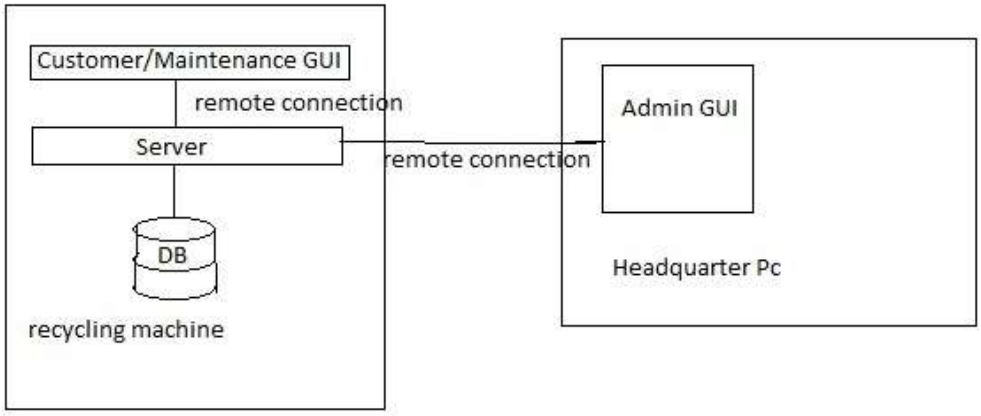

Figure 1- Architecture of the project

One part of the Client GUI - the maintenance part - and the whole Admin GUI can only be accessed after logging into the system by providing a username and password. The password is hashed before sent to the server and it is stored hashed as well. This is a basic MD5 algorithm, just demonstrate the extra security step taken to protect passwords. Users get a session token after logging in and authenticated by this token for every subsequent operation.

The two user interfaces (client and admin part) can be connected using either XML-RPC or RMI remote connection. I have used GIT version control for my development.

Design

During the OO design of the project I tried to follow the SOLID principles as much as possible. According to the above-mentioned principles the major parts of the system depend on abstractions and not on implementation where it is reasonable. To simplify development, I have divided the project's classes into packages according to their responsibilities. See Figure 2 - package structure.

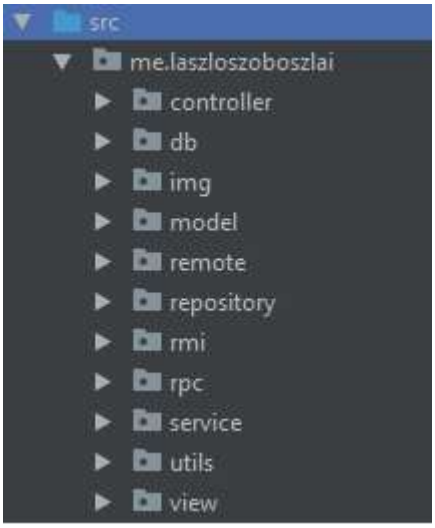

Figure 2 - package structure

At the beginning there was no database planned to be connected to the system on the server side, but for the functionality to collect usage data I decided to add a MongoDB document database after all. Rest of the data is persisted in "json" and "properties" files for simplicity. The data stored in the database and in files are accessed by their repository classes.

There are two interfaces: ItemServiceInterface which abstracts out the functionality for the maintenance and admin functionalities using ItemRepository and UsageRepository classes, DepositItemReceiverInterface does the same for the user facing functionalities and uses the ReceiptBasis class. The ItemService implementation uses Singleton pattern to make sure there is only one instance exists for both classes using it. The three panels: CustomerPanel, MaintenancePanel, LoginPanel expose these services to the remote connections.

```
public interface ItemServiceInterface {
    Map<String, Long> getStatus();
    String emptySlot(int slot) throws IOException;
    String changeItemValue(String name, int value);

    void recordUsage(Map<String, Item> items) throws IOException;

    void recordDeposit(Map<String, Item> items) throws IOException;

    boolean isFull(String itemName);

    int getItemValue(String name);

    Map<String, Long> getCapacity();

    void setCapacity(String name, long value) throws IOException;

    ArrayList<Document> getUsage(long from, long to) throws IOException;
}
```

Realising the remote connection, both XML-RPC and RMI part was designed to implement a common interface (RecycleRemoteConnection), therefore the implementations are interchangeable and can be injected at runtime to the GUIs. On the client side (both ClientGUI and AdminGUI) remote connection type can be set using the ConnectionType enum's "RMI" or "XML-RPC" value. Button events are handled as a call to the methods on the server side to achieve the desired functionality.

```
/**
 * Interface to be implemented by the different implementations of
 * (XML-RPC and RMI) connection classes to handle remote calls for the Recycle
Machine.
 * The different implementations can be injected on demand as they implement this
common interface.
 *
 * @author Laszlo Szoboszlai
 */
public interface RecycleRemoteConnection extends Remote {
    /**
     * Returns the status of the machine (How full it is)
     *
     * @param token the user-token to authenticate the user beforehand
     * @return Map of the status of the machine (ItemName ->  number of items
deposited)
     * @throws IOException
     */
    Map getStatus(String token) throws IOException;

    /**
     * Empties a given slot
     *
     * @param token the user-token to authenticate the user beforehand
     * @param slot  the slot number to be emptied
     * @return true if the operation was successful
     * @throws IOException
     */
    boolean emptySlot(String token, int slot) throws IOException;

    /**
     * Changes the value of a given item
     *
     * @param token the user-token to authenticate the user beforehand
     * @param name  name of th item, the value of which to be changed
     * @param value the new value of the item
     * @return true if the operation was successful
     * @throws RemoteException
     */
    boolean changeItemValue(String token, String name, int value) throws
RemoteException;

    /**
     * Logs in a user with a given username and password
     *
     * @param userName the name of the user
     * @param password the password of the user
     * @return "wrong" if the login process failed, user-token otherwise
     * @throws RemoteException
     * @throws NoSuchAlgorithmException
     */
    String login(String userName, String password) throws RemoteException,
NoSuchAlgorithmException;
```

```
    /**
     * Returns the value of an item.
     *
     * @param token the user-token to authenticate the user beforehand
     * @param name  name of the item, the value of which to be returned
     * @return true if the operation was successful
     * @throws RemoteException
     */
    int getItemValue(String token, String name) throws RemoteException;

    /**
     * Returns the capacity of the items' slots as a map
     *
     * @param token the user-token to authenticate the user beforehand
     * @return Map of the items' capacity  (item -> capacity map)
     * @throws IOException
     */
    Map getCapacity(String token) throws IOException;

    /**
     * Sets the capacity of a given item
     *
     * @param token    the user-token to authenticate the user beforehand
     * @param name     the name of the item
     * @param capacity the new capcity value
     * @return true if the operation was successful
     * @throws IOException
     */
    boolean setCapacity(String token, String name, long capacity) throws
IOException;

    /**
     * Classifies an item inserted to the machine's slot
     *
     * @param itemNumber the number of the item's slot
     * @return true if the operation was successful
     * @throws RemoteException
     */
    boolean classifyItem(int itemNumber) throws RemoteException;

    /**
     * Prints the receipt of the deposited items since the last receipt printed.
     *
     * @return true if the operation was successful
     * @throws IOException
     */
    boolean printReceipt() throws IOException;

    /**
     * Gets the usage information of the machine for a given date range
     *
     * @param token the user-token to authenticate the user beforehand
     * @param from  the date the usage data is required from
     * @param to    the date the usage data is required to
     * @return Vector of Strings in a json format of usage data
     * @throws IOException
     */
    Vector<String> getUsage(String token, String from, String to) throws
IOException;
```

```
    /**
     * Logs a given user out
     *
     * @param username the name of the user to be logged out.
     * @return true if the operation was successful
     * @throws RemoteException
     */
    boolean logout(String username) throws RemoteException;
}
```

Exceptions are handled at the GUI level with appropriate MessageDialog on the screen. On the other hand, the server just returns a specific value like: “wrong”, “-1” or a false to the clients if exception thrown. These specific values are handled with error messages on the GUI level again.

The abstract class DepositItem in the example code could not be instantiated so a new parent class was inserted before the individual items (Item) in the hierarchy.

In the rmi and rpc packages there are starter classes to start:

- the server with the display integrated to it
- the customer/ maintenance GUI with the desired remote connection
- the admin GUI with the desired remote connection

Because of the common interface implemented by the classes using the different protocols, this separation would not even be necessary, as the connection method could be selected at runtime.

Implementation

When I started the implementation, I had created some basic XML-RPC and RMI prototypes with few of the basic functions first, but later I have decided to add the remote capability later on and just complete the functionalities first. I have started it with the core functionality required to be included, and figured out the extra features on the go, so therefore I needed few re-designs.

Once the core functionalities had been ready I have added RMI remote connection first. When the RMI core functionality had worked correctly, I extracted the methods in an interface, and tried to create the XML-RPC version to implement this new interface. During this I had discovered the limitations of this older protocol and I needed to create helper methods in order to convert some of the data types which were easily handled by RMI protocol. Using RMI to implement the remote functionality was much simpler, as I just needed to call the right methods from the relevant panels, but as a compromise the “setter” methods needed to return something other than void to be compatible with XML-RPC. In XML-RPC a static helper method was used to simplify the repeating call to the server.

```
/**
 * Helper method to call methods remotely using XML-RPC
 *
 * @param methodName the name of the method to call
 * @param params     the parameters for the method
 * @return the result of the method call
 */
private static Object executeRemotely(String methodName, Vector params) {
    try {
        Object result = server.execute(methodName, params);
        return result;

    } catch (Exception exception) {
        exception.printStackTrace();
    }
    return null;
}
```

Finally, I was able to make the remote calls with both implementations, and they are interchangeable thanks to the common interface they are implementing. The client side (AdminLoginGUI, RecyclingGUI) both can be set by an enum parameter to create the desired remote connection implementation.

Later the token handling part was added to the project, to handle user sessions, and authorize access to maintenance and admin features. Every subsequent GUI got the main remote connection with the session token “injected” where applicable.

Testing

The project is created in a distributed fashion, so unit testing of the functionalities was getting harder when the remote capability was added, so instead the full system was tested. See test result in the APPENDIX.

Evaluation

The whole project was designed around the implicit functional and non-functional requirements explained in the case study and in the description of the assignment. There are also few extra requirements added, to personalise the final product.

I think it has met the requirements because:

Functional requirements:

- It has a user interface where customers can insert recyclable items into the slot of the machine (it is simulated by a button press with the relevant item)
- It can print out the receipt once the “Receipt button is pressed”
- The counter of the items can be reset
- The admin GUI can access any of the running machines and can collect usage data (can also set date interval)

Non-functional requirements:

- Uses XML-RPC and RMI for remote connection
- Uses basic authentication (session tokens)
- The code is commented (Javadoc comments)

Extra functionality:

- Common interface for the remote connections so they are interchangeable
- Price of items can be changed
- Capacity of the individual slots can be changed
- Admin GUI displays chart of the usage of the machine (date interval can be set)
- Basic hashing of passwords (MD5)
- Uses JSON files and MongoDB to persist data

Resources

https://pixabay.com/en/external-link-right-next-proceed-297789/. (2018). [image] Available at: https://pixabay.com/en/external-link-right-next-proceed-297789/ [Accessed 4 Jan. 2018].

https://pixabay.com/en/green-grass-prato-echo-ecological-1968596/. (2018). [image] Available at: https://pixabay.com/en/green-grass-prato-echo-ecological-1968596/ [Accessed 4 Jan. 2018].

https://pixabay.com/en/milk-carton-drink-packaging-40900/. (2018). [image] Available at: https://pixabay.com/en/milk-carton-drink-packaging-40900/ [Accessed 4 Jan. 2018].

https://pixabay.com/en/cogwheel-gear-gearwheel-cog-145804/. (2018). [image] Available at: https://pixabay.com/en/cogwheel-gear-gearwheel-cog-145804/ [Accessed 4 Jan. 2018].

https://pixabay.com/en/man-gray-silhouette-male-person-303792/. (2018). [image] Available at: https://pixabay.com/en/man-gray-silhouette-male-person-303792/ [Accessed 4 Jan. 2018].

https://pixabay.com/en/crate-pen-box-wooden-slats-311771/. (2018). [image] Available at: https://pixabay.com/en/crate-pen-box-wooden-slats-311771/ [Accessed 4 Jan. 2018].

https://pixabay.com/en/hammer-wrench-repair-work-industry-28636/. (2018). [image] Available at: https://pixabay.com/en/hammer-wrench-repair-work-industry-28636/ [Accessed 4 Jan. 2018].

https://pixabay.com/en/register-cash-register-modern-23666/. (2018). [image] Available at: https://pixabay.com/en/register-cash-register-modern-23666/ [Accessed 4 Jan. 2018].

Codejava.net. (2018). *How to use JDatePicker to display calendar component*. [online] Available at: http://www.codejava.net/java-se/swing/how-to-use-jdatepicker-to-display-calendar-component [Accessed 4 Jan. 2018].

APPENDIX

Project Name: Recycling machine

Test Case ID: 1

Test Designed by: Laszlo Szoboszlai

Module Name: Customer GUI (RMI connection)

Test Executed by: Laszlo Szoboszlai

Test Title: item can be deposited by customer

Test Execution date: 03/01/2018

Description: all item buttons function

Pre-conditions: -

Dependencies: RMI server running

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Press “Can”	-	Can received displayed, usage and deposit data recorded.	Can inserted displayed, usage and deposit data recorded.	Pass	
2	Press “Bottle”	-	Bottle received displayed, usage and deposit data recorded.	Bottle received displayed, usage and deposit data recorded.	Pass	
3	Press “Crate”	-	Crate received displayed, usage and deposit data recorded.	Crate received displayed, usage and deposit data recorded.	Pass	

4	Press “Carton”	-	Carton received displayed, usage and deposit data recorded.	Carton received displayed, usage and deposit data recorded.	Pass	

Post-conditions:

Items are deposited and deposit, usage data recorded

Project Name: Recycling machine

Test Case ID: 2

Test Designed by: Laszlo Szoboszlai

Module Name: Customer GUI (RMI connection)

Test Executed by: Laszlo Szoboszlai

Test Title: Receipt can be printed by customer

Test Execution date: 03/01/2018

Description: after depositing items, receipt can be printed

Pre-conditions: item deposited

Dependencies: RMI server running

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Deposit an item	Can deposited	Can received displayed, usage and deposit data recorded.	Can received displayed, usage and deposit data recorded.	Pass	
2	Press “Receipt”	-	1* Can = price of can and Total: price of can displayed Can removed from the deposited receipt basis	1 * Can = 10 p Total: 10 p	Pass	

Test Case ID: 3

Test Designed by: Laszlo Szoboszlai

Module Name: Customer GUI RMI connection

Test Executed by: Laszlo Szoboszlai

Test Title: status option requires login

Test Execution date: 03/01/2018

Description: status of the machine can only be accessed by admin or maintenance person

Pre-conditions: -

Dependencies: RMI server running

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Press “Status”	-	Login screen opens	Login screen opens	Pass	
2	Login with wrong password	Name:admin password: pass	Login fails	Pop-up message displays, and login fails	Pass	
3	Login with valid credentials	Name: admin, password: admin	Login success, maintenance panel opens	Login success, maintenance panel opens	Pass	

Post-conditions:

User logged into maintenance mode

Project Name: Recycling machine

Test Case ID: 4

Test Designed by: Laszlo Szoboszlai

Module Name: Customer GUI RMI connection

Test Executed by: Laszlo Szoboszlai

Test Title: Individual item properties screen open up

Test Execution date: 03/01/2018

Description: by clicking on each item, their property screen will come up with their actual status

Pre-conditions: RMI server running, user logged in

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Click on “Can”		Can property opens up	Can property opens up	Pass	
2	Click on “Carton”		Carton property opens up	Carton property opens up	Pass	
3	Click on “Crate”		Crate property opens up	Crate property opens up	Pass	
4	Click on “Bottle”		Bottle property opens up	Bottle property opens up	Pass	

Project Name: Recycling machine

Test Case ID: 5

Test Designed by: Laszlo Szoboszlai

Module Name: Customer GUI RMI connection

Test Executed by: Laszlo Szoboszlai

Test Title: Item's properties can be changed

Test Execution date: 03/01/2018

Description: admins and maintenance can change the value, max capacity, and can empty a slot.

Pre-conditions: RMI server running, user logged in and properties panel open for an item

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Click on "empty"	-	Counter set to 0	Counter set to 0 and popup appears	Pass	
2	Click up arrow		Value increases	Value increases	Pass	
3	Click "Set"		New value saved	New value saved, popup	pass	

				appears		
4	Enter new capacity and press “Modify”		New capacity set	New capacity set, popup appears	Pass	

Post-conditions:

Properties of the item changed

Project Name: Recycling machine

Test Case ID: 6

Test Designed by: Laszlo Szoboszlai

Module Name: Admin GUI RMI connection

Test Executed by: Laszlo Szoboszlai

Test Title: admin can log in to a given remote machine

Test Execution date: 03/01/2018

Description: admins can log in to any of the 25 recycling machines

Pre-conditions: RMI server running on the remote machine

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Ente username and password and select the remote machine and click on login	Name:admin, password: admin, correct remote machine	Login successful	Login successful, maintenance panel and admin dashboard opens	Pass	

Project Name: Recycling machine

Test Case ID: 7

Test Designed by: Laszlo Szoboszlai

Module Name: Admin GUI RMI connection

Test Executed by: Laszlo Szoboszlai

Test Title: Usage chart can be displayed

Test Execution date: 03/01/2018

Description: admins can get usage information, visualize it

Pre-conditions: admin logged in, remote machine running and connected

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Select from date on dashboard	12/12/2017	Date selected	Date selected	Pass	
2	Select to date on dashboard	10/01/2018	Date selected	Date selected	Pass	
3	Click “draw chart”		Chart displayed	Chart displayed	Pass	

Post-conditions:

Chart displayed

Project Name: Recycling machine

Test Case ID: 8

Test Designed by: Laszlo Szoboszlai

Module Name: Customer GUI (XML-RPC connection)

Test Executed by: Laszlo Szoboszlai

Test Title: item can be deposited by customer

Test Execution date: 03/01/2018

Description: all item buttons function

Pre-conditions: -

Dependencies: XML-RPC **server running**

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Press “Can”	-	Can received displayed, usage and deposit data recorded.	Can inserted displayed, usage and deposit data recorded.	Pass	
2	Press “Bottle”	-	Bottle received displayed, usage and deposit data recorded.	Bottle received displayed, usage and deposit data recorded.	Pass	
3	Press “Crate”	-	Crate received displayed, usage and deposit data recorded.	Crate received displayed, usage and deposit data recorded.	Pass	
4	Press “Carton”	-	Carton received displayed, usage	Carton received displayed, usage and deposit data	Pass	

			and deposit data recorded.	recorded.		

Post-conditions:

Items are deposited and deposit, usage data recorded

Project Name: Recycling machine

Test Case ID: 9

Module Name: Customer GUI (XML-RPC connection)

Test Title: Receipt can be printed by customer

Description: after depositing items, receipt can be printed

Test Designed by: Laszlo Szoboszlai

Test Executed by: Laszlo Szoboszlai

Test Execution date: 03/01/2018

Pre-conditions: item deposited

Dependencies: XML-RPC server running

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Deposit an item	Can deposited	Can received displayed, usage and deposit data recorded.	Can received displayed, usage and deposit data recorded.	Pass	
2	Press “Receipt”	-	1* Can = price of can and Total: price of can displayed Can removed from the deposited receipt basis	1 * Can = 10 p Total: 10 p	Pass	

Post-conditions:

the deposited items list emptied

Project Name: Recycling machine

Test Case ID: 10

Test Designed by: Laszlo Szoboszlai

Module Name: Customer GUI, XML-RPC connection

Test Executed by: Laszlo Szoboszlai

Test Title: status option requires login

Test Execution date: 03/01/2018

Description: status of the machine can only be accessed by admin or maintenance person

Pre-conditions: -

Dependencies: XML-RPC server running

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Press “Status”	-	Login screen opens	Login screen opens	Pass	
2	Login with wrong password	Name:admin password: pass	Login fails	Pop-up message displays, and login fails	Pass	
3	Login with valid credentials	Name: admin, password: admin	Login success, maintenance panel opens	Login success, maintenance panel opens	Pass	

Post-conditions:

User logged into maintenance mode

Project Name: Recycling machine

Test Case ID: 11

Test Designed by: Laszlo Szoboszlai

Module Name: Customer GUI, XML-RPC connection

Test Executed by: Laszlo Szoboszlai

Test Title: Individual item properties screen open up

Test Execution date: 03/01/2018

Description: by clicking on each item, their property screen will come up with their actual status

Pre-conditions: XML-RPC server running, user logged in

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Click on “Can”		Can property opens up	Can property opens up	Pass	
2	Click on “Carton”		Carton property opens up	Carton property opens up	Pass	
3	Click on “Crate”		Crate property opens up	Crate property opens up	Pass	
4	Click on “Bottle”		Bottle property opens up	Bottle property opens up	Pass	

Project Name: Recycling machine

Test Case ID: 12

Module Name: Customer GUI XML-RPC connection

Test Title: Item's properties can be changed

Description: admins and maintenance can change the value, max capacity, and can empty a slot.

Test Designed by: Laszlo Szoboszlai

Test Executed by: Laszlo Szoboszlai

Test Execution date: 03/01/2018

Pre-conditions: XML-RPC server running, user logged in and properties panel open for an item

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Click on "empty"	-	Counter set to 0	Counter set to 0 and popup appears	Pass	
2	Click up arrow		Value increases	Value increases	Pass	
3	Click "Set"		New value saved	New value saved, popup appears	pass	
4	Enter new capacity and press		New capacity set	New capacity set, popup	Pass	

	"Modify"			appears		

Post-conditions:

Properties of the item changed

Project Name: Recycling machine

Test Case ID: 13

Test Designed by: Laszlo Szoboszlai

Module Name: Admin GUI XML-RPC connection

Test Executed by: Laszlo Szoboszlai

Test Title: admin can log in to a given remote machine

Test Execution date: 03/01/2018

Description: admins can log in to any of the 25 recycling machines

Pre-conditions: XML-RPC server running on the remote machine

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Ente username and password and select the remote machine and click on login	Name:admin, password: admin, correct remote machine	Login successful	Login successful, maintenance panel and admin dashboard opens	Pass	

Project Name: Recycling machine

Test Case ID: 14

Test Designed by: Laszlo Szoboszlai

Module Name: Admin GUI XML-RPC connection

Test Executed by: Laszlo Szoboszlai

Test Title: Usage chart can be displayed

Test Execution date: 03/01/2018

Description: admins can get usage information, visualize it

Pre-conditions: admin logged in, remote machine running and connected

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Select from date on dashboard	12/12/2017	Date selected	Date selected	Pass	
2	Select to date on dashboard	10/01/2018	Date selected	Date selected	Pass	
3	Click “draw chart”		Chart displayed	Chart displayed	Pass	

Post-conditions:

Chart displayed