|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Submission Coversheet | | | |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Student ID Number** | ANI18457381 | | | | | | |
| **Programme Title** | Computing Technologies Degree | | | | | | |
| **Module Title** | Object Oriented Programming | | | | | | |
| **Module Code** (listed on Moodle and in LTAFP) | QAC020C152A | | | | | | |
| **Module Convenor** | Tendai Mhlanga | | | | | | |
| **Coursework Title** | Coursework2: Vending Machine Reflection | | | | | | |
| **Academic Declaration:**  Students are reminded that the electronic copy of their essay may be checked, at any point during their degree, with Turnitin or other plagiarism detection software for plagiarised material. | | | | | | | |
| **Word Count** | 1095 | | | **Date Submitted** | | **15/12/2018** | |
|  |  | | |  | |  | |

Table of Contents

[Delivered Features 3](#_Toc532650749)

[Research 10](#_Toc532650750)

[Future improvements 13](#_Toc532650751)

# Delivered Features

As a bonus I’ve implemented an easy to use User Interface in Java Swing Framework with a Model View Controller architecture. Beside the snack images all the other picture resources were created from scratch in Microsoft Paint.

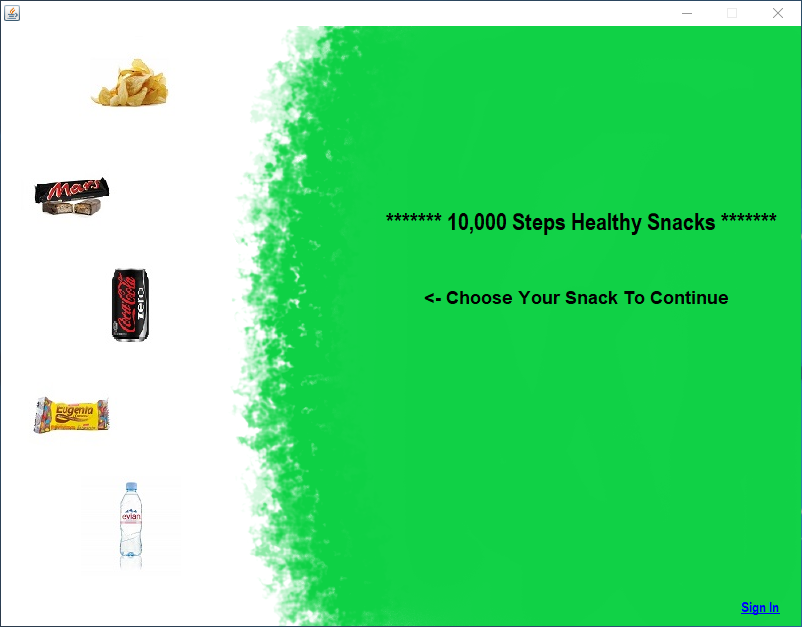
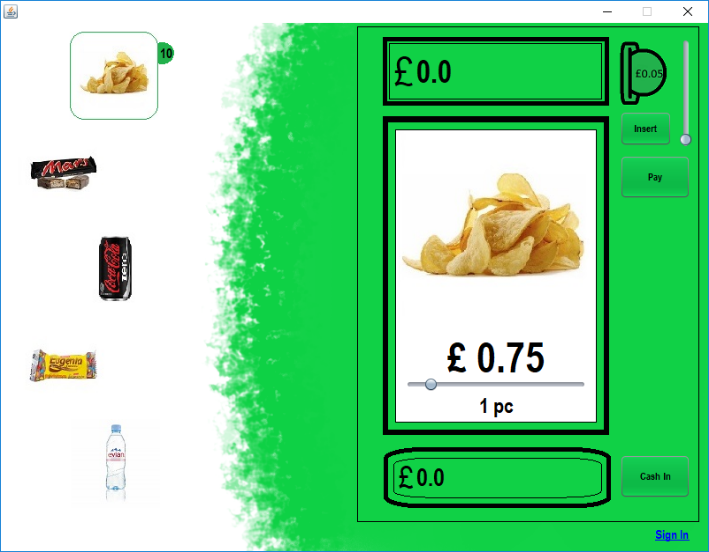
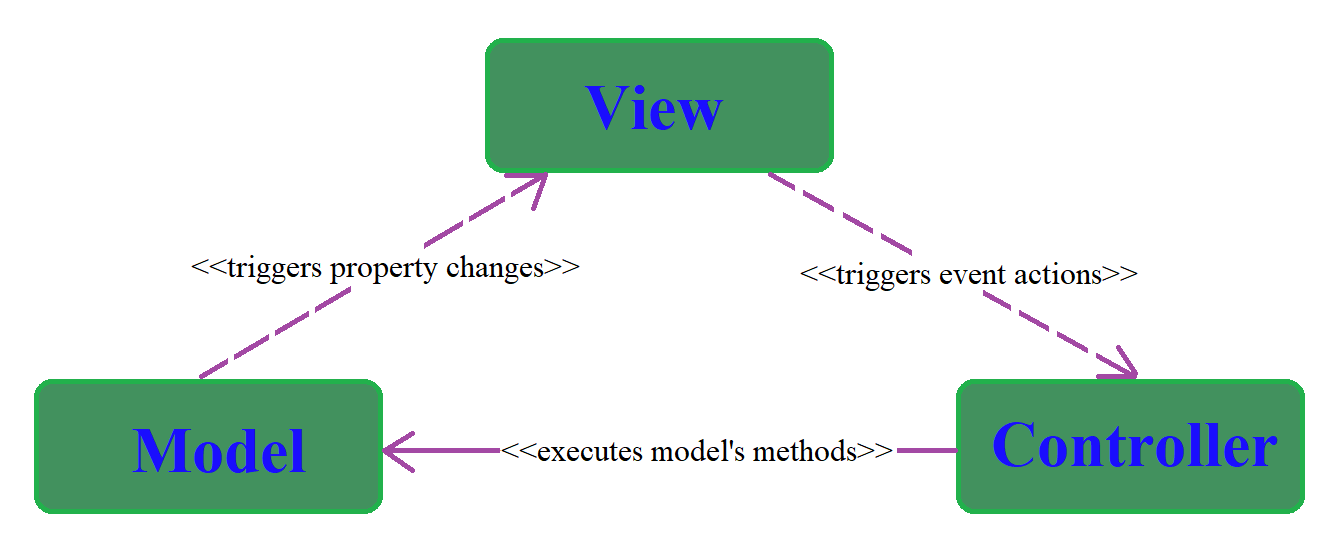


Figure 1



Another interesting thing to mention is that I’ve created from scratch and implemented a mathematical selection function, created specifically for selecting the best option from a pool of possible changes. See Figure 2.

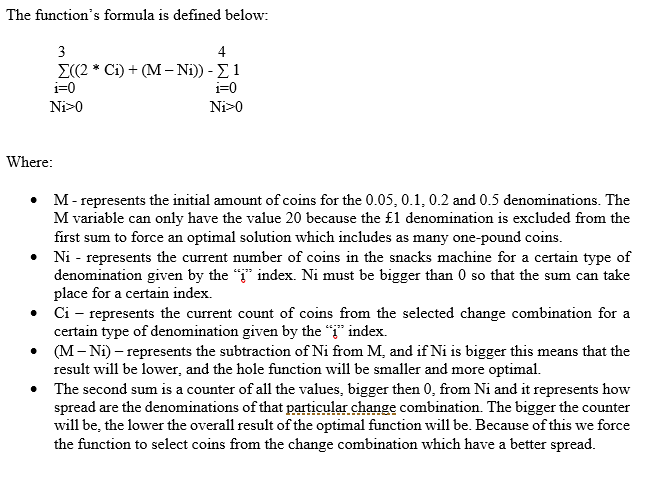
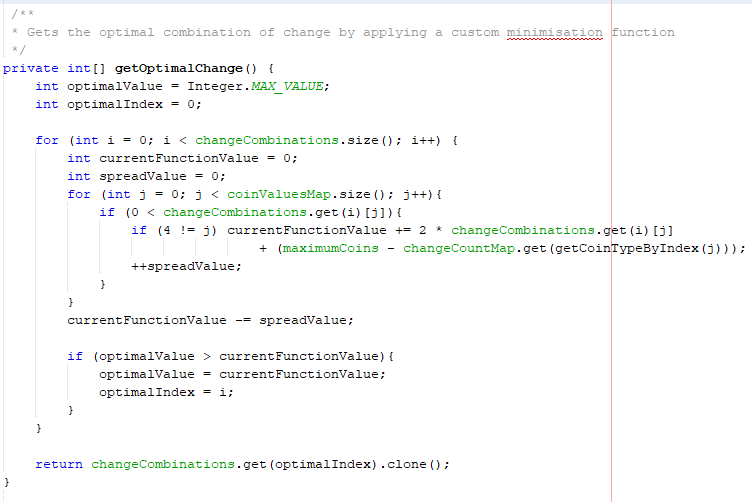


Figure 2



All the requested features have been fully delivered. List of implemented features:

* The selection of a Snack is done by clicking on its representative image-button and the current quantity is shown in a badge widget appended to the highlighted selection. The quantity updates in real time according to the model as represented in Figure 1 in the architecture image.



Figure 3

* Quantity is selected in the main vending machine representation with the help of a slider. The slider has its maximum value updated in real time according to the current quantity in the model. See Figure 4.

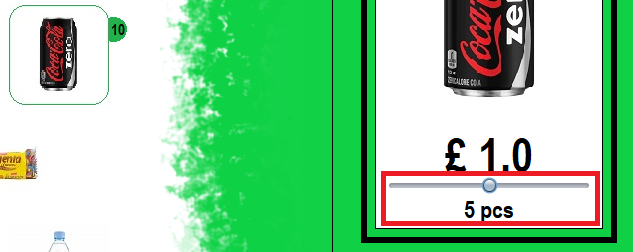


Figure 4

* Selection of the coins is made with the help of a slider and on the left side it has a representation of an inserted coin with its current value. See Figure 5.

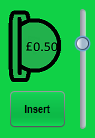


Figure 5

* Insertion and visualization of the inserted coins is done with a label and the insert button. See Figure 6.

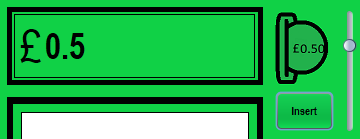


Figure 6

* The paying action is done with the help of a thread so that the main User Interface doesn’t freeze when the change calculation is done. Behind the curtains I implemented the change calculations as a recursive function which gets all the change combinations and the selection is done with the mathematical method portrayed in Figure 2. Nevertheless, getting all the combinations of change from a particular transaction it is not the best option I could have implemented, because the complexity of the current method is O () and the dynamic programming solution has a complexity of O (m\*n) according to Wikipedia. 2001. Change-making problem - Wikipedia. [ONLINE] Available at: <https://en.wikipedia.org/wiki/Change-making_problem>. [Accessed 1 November 2018]. In the figure below, I’m showing the visual side of the payment action and the recursion code.

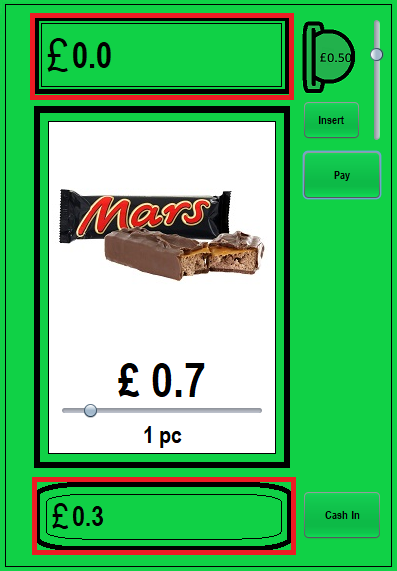


Figure 7

* Implemented the admin login and the report page. See Figure 8 and 9.

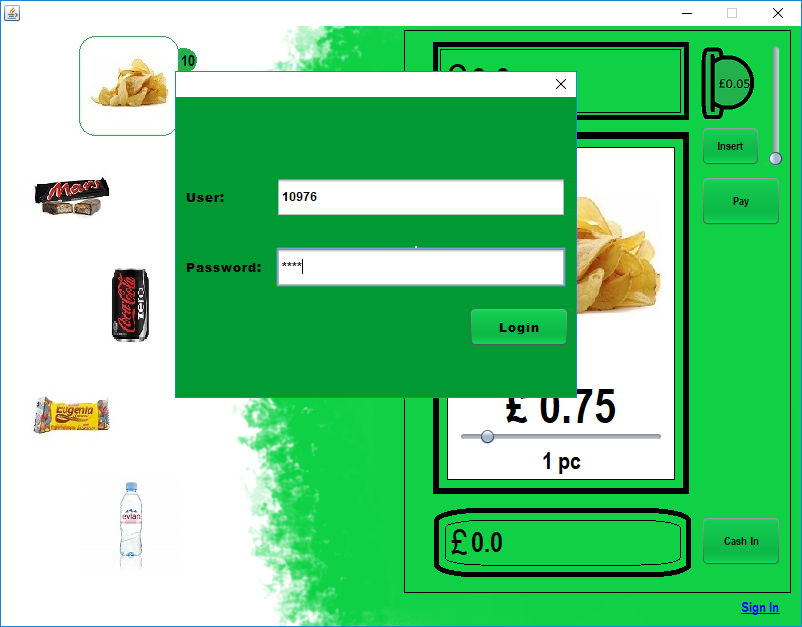


Figure 8



Figure 9

* Added error messages for all possible user interaction mistakes. See Figure 10.

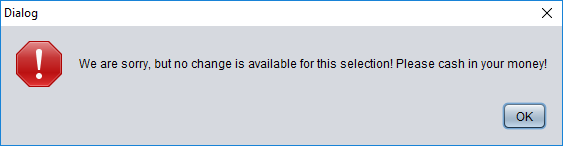
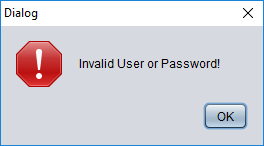
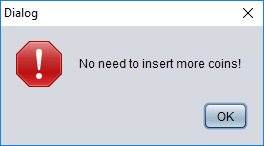
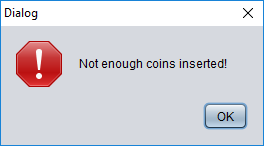
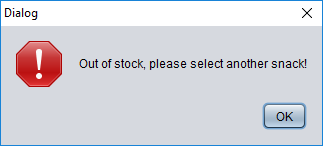
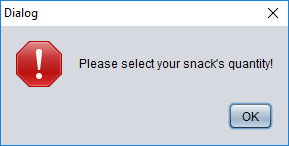


Figure 10

* Added a cash in button so that the user can withdraw his change. See Figure 11.



Figure 11

# Research

A class is a template for the instantiated object according to The Java™ Tutorials. 1995. What Is a Class? (The Java™ Tutorials > Learning the Java Language > Object-Oriented Programming Concepts). [ONLINE] Available at: <https://docs.oracle.com/javase/tutorial/java/concepts/class.html>. [Accessed 1 November 2018] and was the main technique used to group data and do data manipulation.

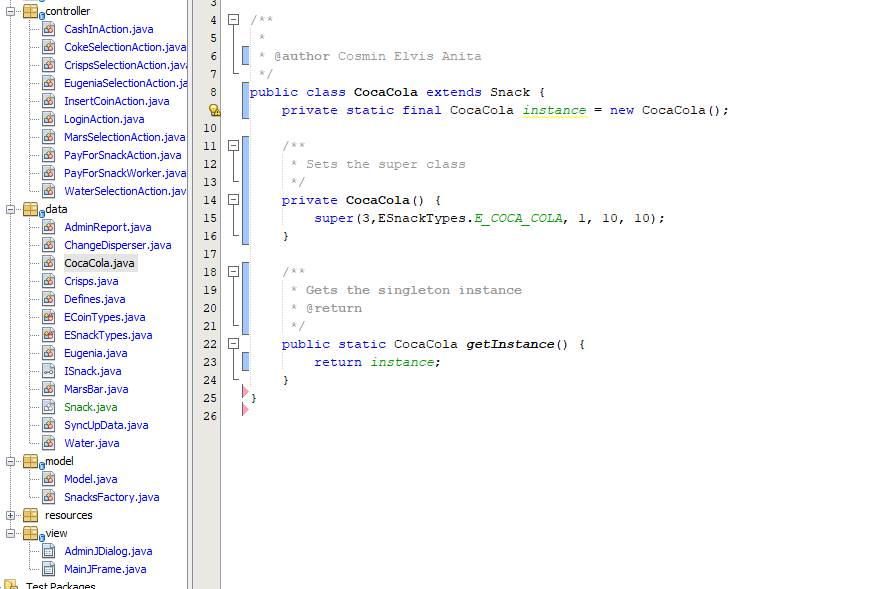


Figure 12

An abstract class is a special type of class which may or may not contain an abstract method and cannot be allocated but can be derived from, according to The Java™ Tutorials. 1995. Abstract Methods and Classes (The Java™ Tutorials > Learning the Java Language > Interfaces and Inheritance). [ONLINE] Available at: <https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html>. [Accessed 1 November 2018]. Sub-classing an abstract class constrains the class to define it as abstract or to implement the abstract methods, because these methods are only declared and don’t have implementation. I’ve used this methodology to share the code to all the snack classes. See Figure 13.

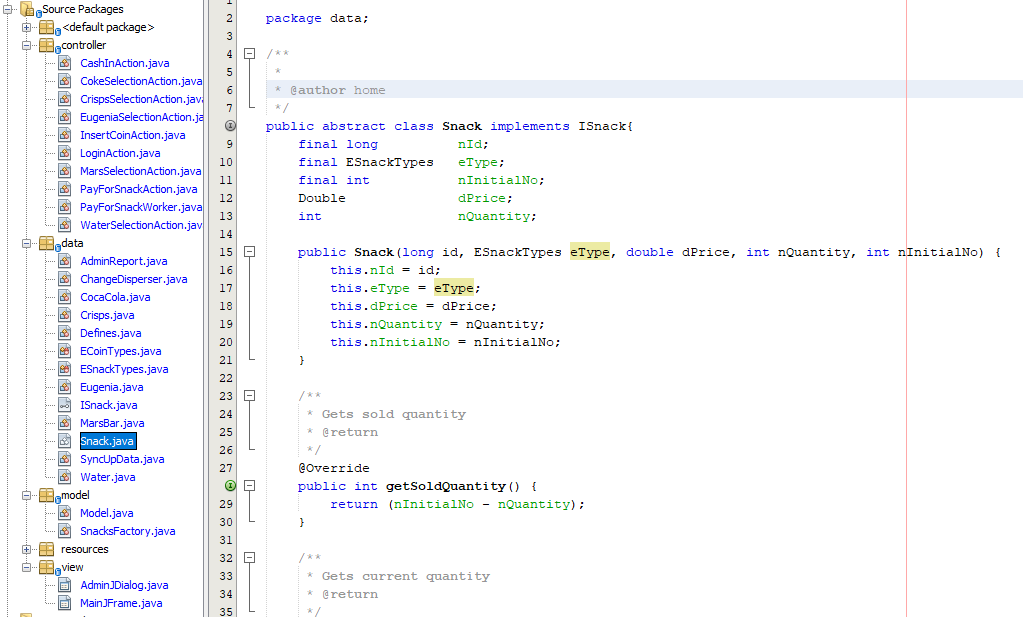


Figure 13

An interface is actually a contract for the class which implements it, with empty bodied methods. Because of this the, class, abstract or not, is obligated to implement the interfaces methods, according to The Java™ Tutorials. 1995. What Is an Interface? (The Java™ Tutorials > Learning the Java Language > Object-Oriented Programming Concepts). [ONLINE] Available at: <https://docs.oracle.com/javase/tutorial/java/concepts/interface.html>. [Accessed 1 November 2018]. I’ve used the interface as a common template implementation for future abstract classes which will have a different meaning from the snack group but will have the interfaces methods in common. For example, we want to divide the snacks into several groups like types of bottled water, chocolates etc. and they will all have a common blueprint. See Figure 14.

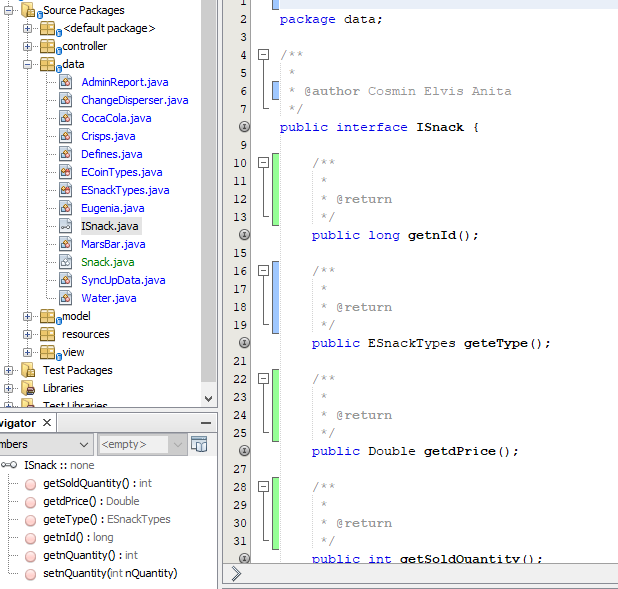


Figure 14

An enum type in Java is a special class which contains one or more constants and/or methods according to The Java™ Tutorials. 1995. Enum Types (The Java™ Tutorials > Learning the Java Language > Classes and Objects). [ONLINE] Available at: <https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html>. [Accessed 1 December 2018]. The enum definition has been used in the project for itemizing the snacks and coins types. See Figure 15.

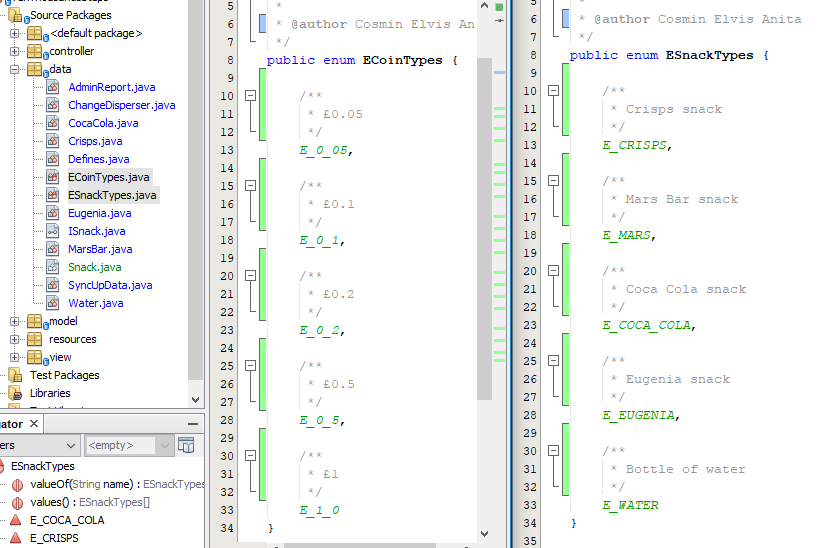


Figure 15

Encapsulation is the action of declaring private the members of a class and add setter and getter methods to modify or use the masked data according to w3schools. 1998. Java Encapsulation and Getters and Setters. [ONLINE] Available at: <https://www.w3schools.com/java/java_encapsulation.asp>. [Accessed 1 November 2018]. This methodology is used extensively all other the project, there are only two three classes who directly expose their members.

Polymorphism in Java’s Object-Oriented Programming is the ability of an interface/abstract class to impersonate the deriving class and to enforce it to implement its own logic according to tutorialspoint. 2006. Java Polymorphism. [ONLINE] Available at: <https://www.tutorialspoint.com/java/java_polymorphism.htm>. [Accessed 1 November 2018]. This was applied in the project by using the interface ISnack in a hash map, see Figure 16.

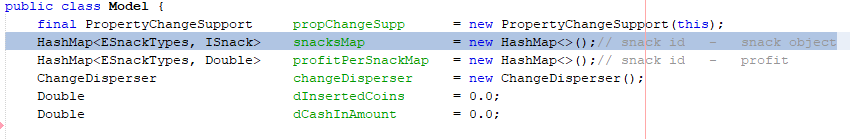


Figure 16

# Future improvements

* I would reimplement and replace the recursive method for finding change combinations to a dynamic programming implementation.
* The application needs local caching for storing the current snack stock.
* The storing of the user, password and report page information should be moved on a server. The credentials should be checked by sending them to the server’s sign-in service, and in turn receive a temporary token and the report page data.
* The metrics data of the snack machine, like the stock of the snacks or amount of coins, should be sent daily or on request on a web-socket connection to the server.
* Implement a non-mandatory survey, for getting the opinion of the users on different issues, like UI improvements for example, and send them to the server to be analyzed.