Cairo University, Faculty of Computers and Artificial Intelligence

CS342

**Cairo University**

**Faculty of Computers and Artificial Intelligence**

Software Engineering 2

2nd Semester 2020 Research

Analyze and Design a System

|  |  |  |
| --- | --- | --- |
| ID | Name | Email |
| 20170212 | Mohamed Ahmed Saad | Mohamedsaad@[stud.fci-cu.edu.eg](mailto:sehes333@stud.fci-cu.edu.eg) |
| 20170094 | Hussien Tarek | [sehes333@stud.fci-cu.edu.eg](mailto:sehes333@stud.fci-cu.edu.eg) |
| 20170160 | Abdallah Naguib | [abdonaguib99@gmail.com](mailto:abdonaguib99@gmail.com) |

Table of Contents

[Requirements: 3](#_Toc42791691)

[Analysis level Class Diagram: 4](#_Toc42791692)

[Updated Class Diagram with Design Patterns: 5](#_Toc42791693)

[System Decomposition: 9](#_Toc42791694)

[Component Diagram: 10](#_Toc42791695)

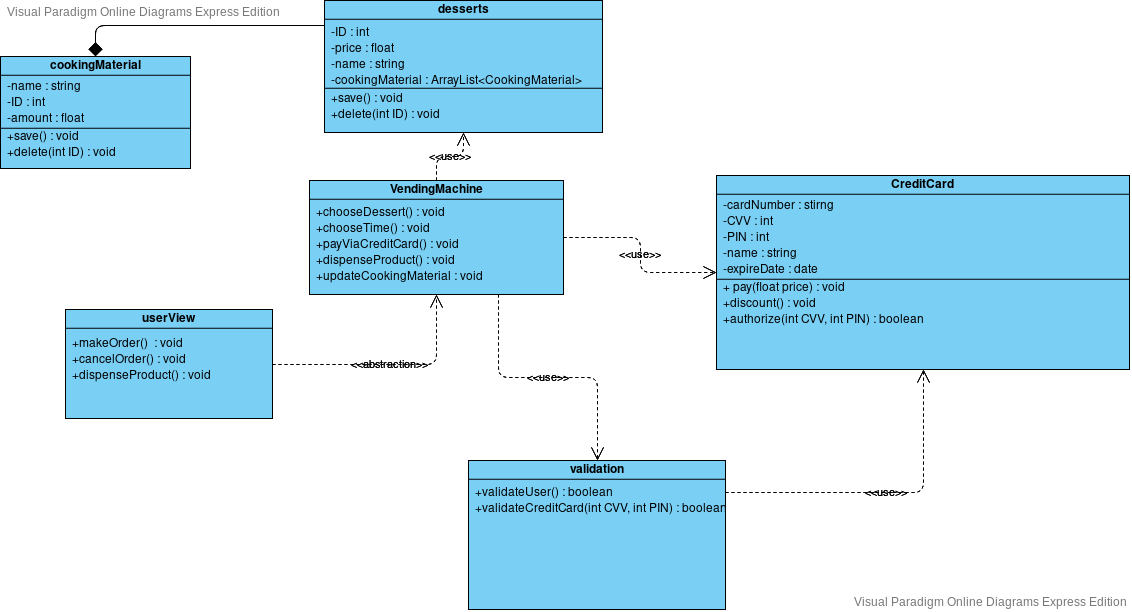
[Deployment Diagram: 11](#_Toc42791696)

[Design Goals: 11](#_Toc42791697)

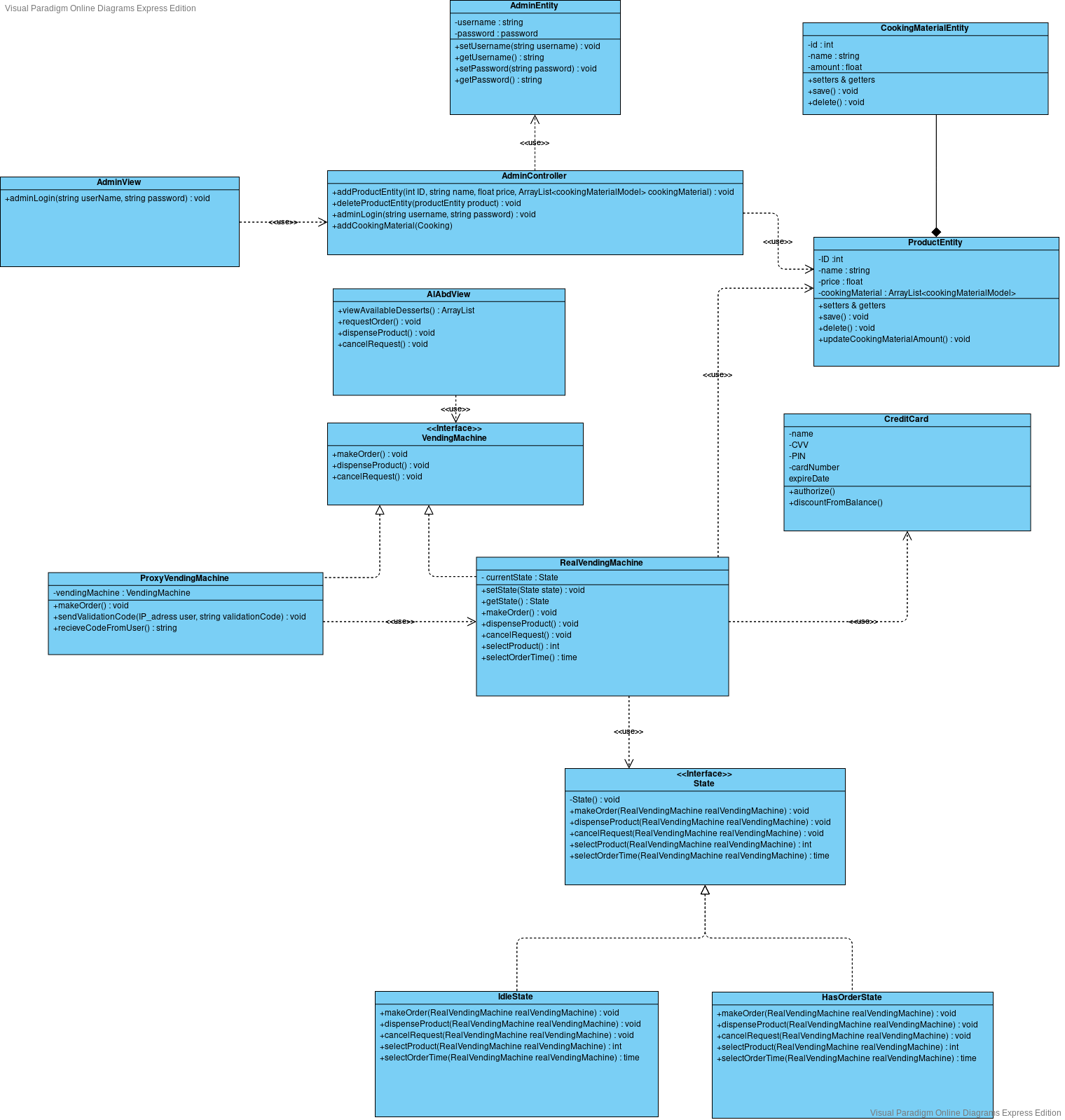
# Requirements:

1. Buyers can make order through internet, he has to download “El-Abd on the go” application to be able to make online orders by his phone.
2. The system should have a list of cooking material and it’s satisfied amount, for example, satisfied amount of the sugar is 20KG, “El-Abd” is responsible for determining the list of material, and it’s needed amount.
3. If any needed cooking material is less than one third (1/3) of it’s amount, then support agency is texted by mail to supply the needed material, The vending machine writes a list of needed material, and sends it to supply agency mail.
4. The user (buyer) has to validate himself to be able to make an order, the system will ask the user to enter his phone number, and it will send a verification code to this phone number. The buyer is validated if he enters the code successfully.
5. If the user is validated, he can make an order by choosing desserts he want, and then it’s price will be calculated, but if there is a shortage in cooking material needed to prepare his desserts, it will refuse the order, and print “This dessert is not available right now, you can choose something else!”, and let him request another order (of course this case will be rare, as the system communicate with the support agency if any cooking material is less than 1/3).
6. The system will ask the user to enter the time he wants to receive the order, in addition to his credit card info which by he pay for the dessert.
7. The system will also validate the credit card info, to make sure he enters correct info, if validation failed, it will said “wrong credit card info”, and ask him to enter his info again.
8. If the balance in the buyer credit card is not enough, the machine cancels the order, and asks the buyer to charge his balance first to be able to make an order.
9. If there is enough money in his credit card, “El-Abd on the go” system informs the credit card company to discount the price from the buyer’s credit card.
10. the system commences producing the ordered desserts to be ready at the time requested by the buyer.
11. If the order is made successfully, subtract cooking material used in preparing the dessert from it’s amount, to make the list of desserts and it’s amount updated after each order.
12. The buyer can get his prepared desserts from the pickup interface of the machine.

# Analysis level Class Diagram:



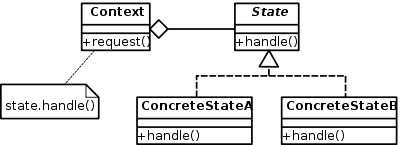
# Updated Class Diagram with Design Patterns:



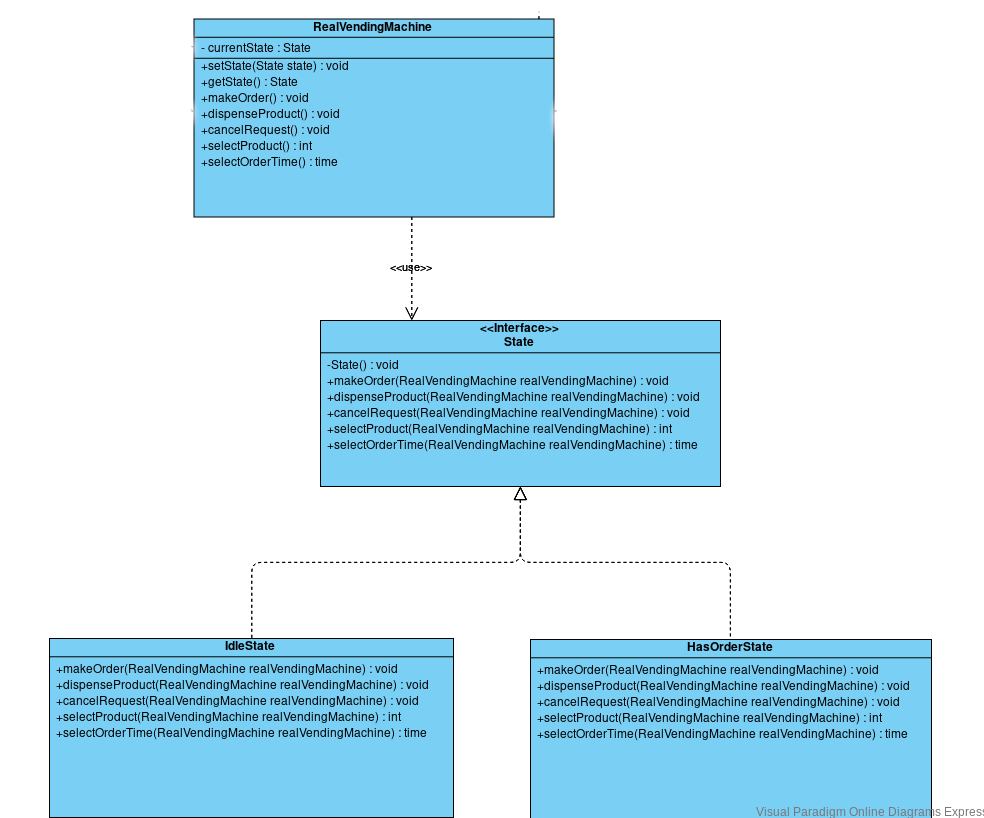
**Used design pattern:**

**State Design Pattern**

**State Pattern Class Diagram**

****

RealVendingMachine (is our context class), State (is our State interface), IdleState (is our concreteStateA), HasOrderState (is our ConcreteStateB)

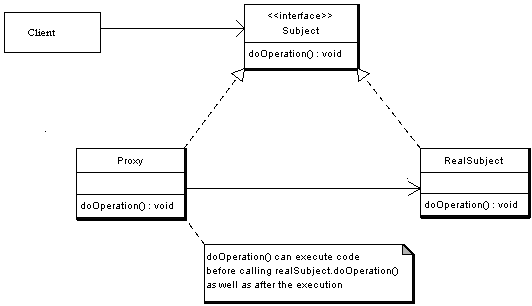
 **State Pattern used in our design**

**Why we are using State Pattern**

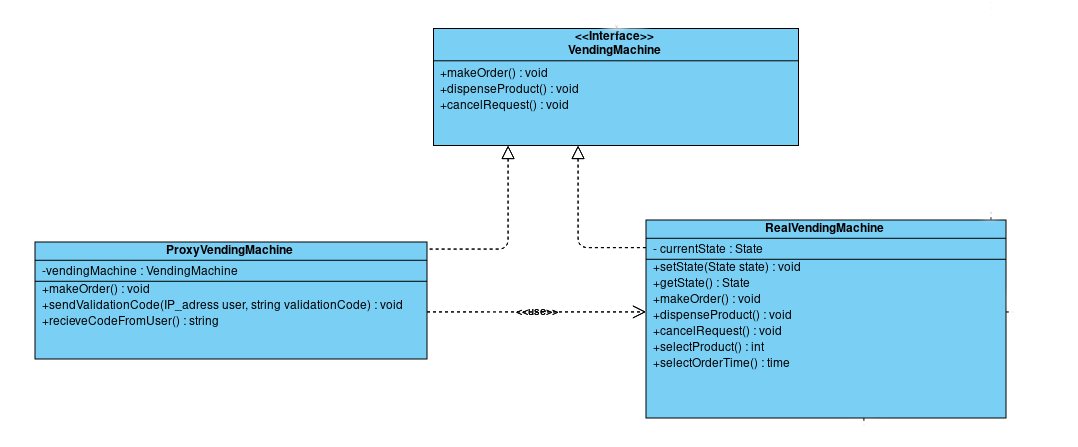
The machine is in the idle state by default, if there is an order requested, the machine will be in hasOrderState, the behavior of the 4 functions (makeOrder, dispenseProduct, cancelRequest) is different in each state, so we need state pattern.

**Proxy Design Pattern**

**Proxy Pattern Class Diagram**

****

VecndingMachine (is our Subject interface), ProxyVendnigMachine (is our Proxy class), RealVendingMachine (is our RealSubject)

****

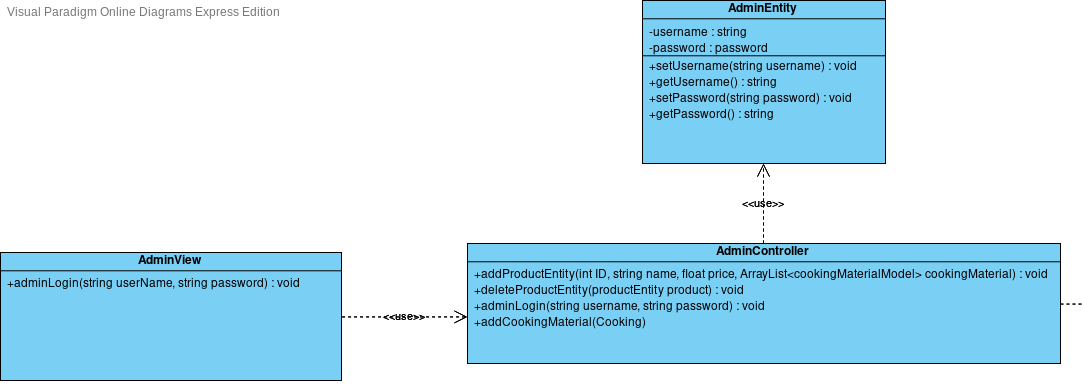
**Why we are using State Pattern**

We need to execute code before calling RealVendingMachine.makeOrder(), the code we want to execute first is the validation of the user, so we want proxy design pattern, protection proxy specifically .

**MVC design pattern**

we use MVC in many areas in our class diagram, like AdminView, AdminEntity, AdminController and   
 AlAbdView(View), RealVendingMachine(Controller), ProductEntity(Model).

**MVC Design Pattern used in our design**



**Why we are using MVC Pattern**

Because we have desserts(product) in database, and we won’t communicate with it from the view, so we need something in between.

# System Decomposition:

1-Admin:

A) Contained Classes

1-AdminEntity

2-AdminController

3-AdminView

B) Provided Interfaces

1-Login:

Log in to Admin's account Service

2-Manage Available Products:

Manage Vending Machine Services For Logged Admin

2-Products Stock:

A) Contained Classes

1-ProductEntity

2-CookingMaterialEntity

B) Provided Interfaces

1-Manage Stock:

Manage Vending Machine Stock Services For Admin

(Save, Delete Products & Update their cooking material)

2-Withdraw Product:

Delete Bought Products From Stock Service For

dispense operation by Customer

3- El-Abd On The Go:

A) Contained Classes

1-AlAbdView

B) Provided Interfaces

1-Order Dessert:

Order Dessert Service For Customer (get Validation Code for the order, Show Current Available

Desserts & Request Available Dessert, Pay For it By Credit Card Services For Buyer)

2-Manage Order Session:

Manage Order Service For Order Owner

(Dispense Product Or Cancel Order)

4-VendingMachine:

A) Contained Classes

1-VendingMachine

2-ProxyVendingMachine

3-RealVendingMachine

4-State

5-IdleState

6-HasOrderState

B) Provided Interfaces

1-Use Vending Machine:

Use Vending Machine Services For Application (Make Order, Dispense Product & Cancel Request)

5- Credit Card:

A) Contained Classes

1-CreditCard

B) Provided Interfaces

1-Discount Card Balance:

Authorize & Discount Balance For Credit Card Account Service For the Vending Machine

a)

Such Decomposition Main Goal is to make Closely Directly-Dependent

Classes are in single subsystem & Loosely Indirectly-Dependent Classes

Are in Different Subsystems & Communicate with each other through their

Subsystems interfaces to maximize Dependencies in Subsystems Which Increase

Cohesion & minimize Dependencies Across Subsystems Which Decrease Coupling

b)

1-Admin: Low Coupling as Depend on single subsystem &

medium Cohesion As Depend on single other Subsystem

2-Products Stock: Medium Coupling As two Subsystems Depend on it

& High Cohesion As Doesn't Depend on any other subsystem

3-El-Abd On The Go: low Coupling As Depend on single Subsystem

& medium Cohesion As Depend on single other Subsystem

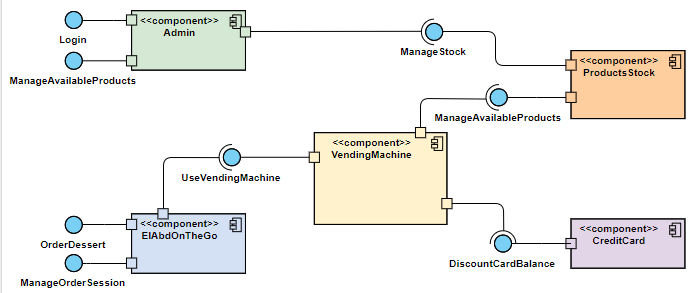
4-VendingMachine: High Coupling one Subsystem Depend on it & it Depends

on two Subsystems & medium Cohesion As Depend on two other subsystems

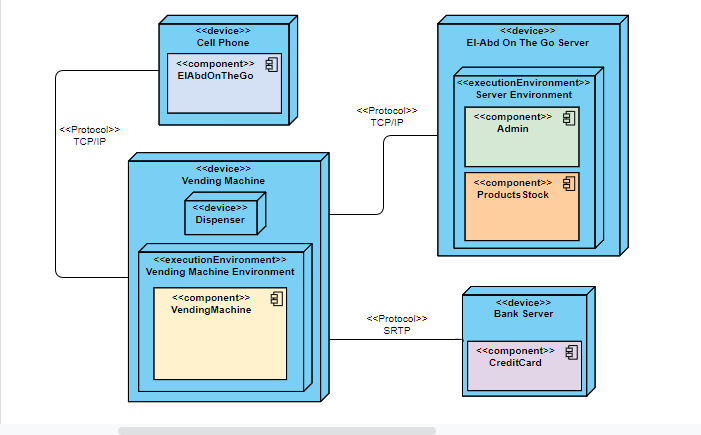
5-Credit Card: Low Coupling as a single subsystem depend on it

& High Cohesion As Doesn't Depend on any other subsystem

# Component Diagram:



# Deployment Diagram:

****

# Design Goals:

**First design goal : The system has to be very usable**, as it will used by all types of people, because it’s a vending machine in the street, that will be used by everyone, those who have technical experience, and people with no technical experience, so the process of ordering from their cell phone has to be easy,

Mapping To the FURPS+ model : **Usability**

**second design goal : Response time shouldn’t exceed 30 second,** to make the dessert ready for the user. because if the order is late for the user, user will cancel the order.

Mapping To the FURPS+ model : **Performance**

**Third design goal : The system should allow the user to select the time of the order**, and it should be accurate for the time user select, because if the system doesn’t provide the dessert to the user in the time, maybe the user stop using the system.

Mapping To the FURPS+ model : **Functionality**

**Fourth design goal : The system has to be secure,** as the user will user his credit card, so it will be a problem if there is a security breach, as it will affect the user very sensitive data.

Mapping To the FURPS+ model : **Functionality**

**Suggestions to achieve these design goals**

First design goal(usability) : To achieve the first only 3 buttons, make order, dispense product, cancel order.

Second design goal(performance) : the system should have every kind of dessert prepared previously, so if the user select this dessert, he will get it immediately, and the application response time will be achieved may be if we used binary search to search for the dessert in the dessert list, and make sure every algorithm is enhanced well.

Fourth design goal(functionality) : we can make the communication with the bank server secure may be if we used SRTP(Secure Real-time Transport Protocol) protocol instead of TCP/IP.