Cairo University

Faculty of Computers and Artificial Intelligence

CS342

Software Engineering (2)

2nd Semester 2020 Research

Analyze and System Design (Topic One)

ID	Name	Email
20170331	Yasser Ashraf Salah El-Din	yasserashraf1812@gmail.com
20170283	Mariam Makaram William	mariammakram1@gmail.com
20170204	Leena Essam Mohamed	leenahessam18@gmail.com
20170203	Lauren Safwat Shokry	lauren.safwat18@gmail.com
20170335	Youssef Ahmed Abdel-Samad	yousef5121998@gmail.com

Table of Contents

1. System Overview

➔ This system will be online vending machine system for El Abd Company. This system will help customers to order desserts and set their pick up time and date. This system includes modules like Management module, Desserts, Vending machine, Materials, Orders and Items.

- **Management Module** ➔ Management Module includes the management of users, desserts and orders, for user management, you should register. The user can register as a customer to explore or buy products. If you have account so you should login, else so you have to register to the system provided data needed to user it, for dessert management, set dessert data and its cooking materials, for order management, carry order data, order items and extras over one item if exists.

- **Desserts Module** ➔ Each Dessert consists of materials that should be determined to prepare dessert when customer order it.

- **Vending Machine** ➔ Vending Machine has location, id to be identified for support agency to manage its system and only one admin to control it, vending machine contains materials to prepare order(dessert(s)).

- **Orders** ➔ User can order one or more type of dessert. Each type can have extras.

- **Items** ➔ Each type of dessert is considered as an item.

- **Materials** ➔ Each Dessert consists of components with specified amount for preparing.

2. Requirement Backlog

1. "El Abd on the go" is oriental dessert vending Machine linked by the system, system could be a website or mobile application, machine has unique ID, location and only one administrator to control the system.

2. User will Sign up to the system, he must enter (email, username, Password, Address, Mobile number, Country, City, Credit Card Number to be able to communicate with credit card company directly when user orders).

3. System will validate the above data, user must enter unique email and username, password must be more than 8 characters, mobile number must be 11 digits and credit card must be activated.

4. User will be able to Login if he already has an account, he will enter username/email and password to be able to access his account.

5. The process of signing in contains two-way authentication, first the user types his username/email and password, if data was correct, system will send him a code that he will insert (SMS Message to phone number or Email Account).

6. There is a type of user, the admin (only one admin), he will be able to add, remove and edit desserts in the system.

7. To be that admin, this person can communicate with the company, take a code(key), signs up as an admin and this code will be requested and checked if it's true he will fill all the details and simply make his account.

8. The other type of user is the Normal User, the user who explores the system in order to find some dessert.

9. The normal user will be able to edit his personal data like Address, Phone number, Credit Card Number and email and system will validate them again as explained in point number 4.

10. On successful login, normal user will be able to see all available desserts and all recent offers(discounts), user can view dessert's image, name, description and price, user can choose one (or more) of these desserts.

11. Normal user will be able to set extra components on the ordered dessert (for example extra chocolate).

12. Now normal user enters the amount of each dessert type in the order (how many boxes) and the time needed (how long it takes him to reach the vending machine to take his order).

13. First, vending machine will check its materials and compare them to the amount ordered by user in the system, if they aren't enough, system will call the support agency to alert them to refill the machine.

14. Second thing to be checked by system is the preparation time and compare it to the time user entered, if it is more than the pickup time, system will tell the user the allowed pickup time (preparation time) and he can accept or cancel order.

15. System will then view all order details to the user if he wants to edit any, some modifications will require returning to some validations from those mentioned above.

16. On Confirming, System will send final recipient order Details (Components - Costs- Pick Up Time - Order ID - Vending Machine Location - Amount of each dessert Type) to the Email Account.

17. Validate the Card Number, Account Data and Check for Transaction Process Acceptance, if Money is Enough for Transaction Method, if not, error message will appear, order will be canceled and the system terminates.

18. System will inform the Credit Card Company to charge The Buyer's Credit Card (after Validating) with the Order Cost, and the system will withdraw that money (cost).

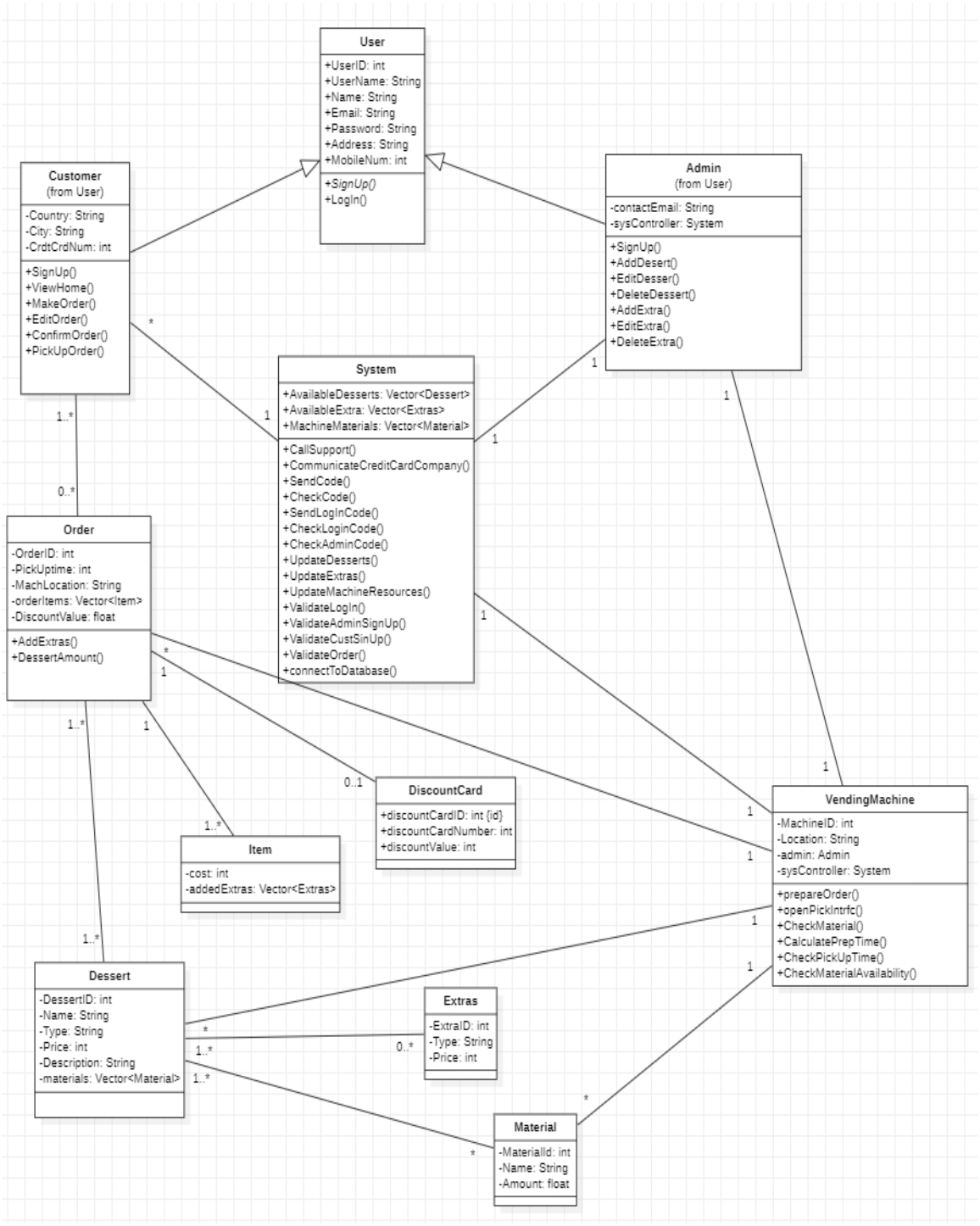
19. At the pickup time, user can ask the system for the order, so system will send an SMS to his phone number carrying a code, system asks for that code to make sure of the receiver identity(authentication).

20. If the entered code is correct, user can then take his order (pickup interface will open) and send confirmation message to email account.

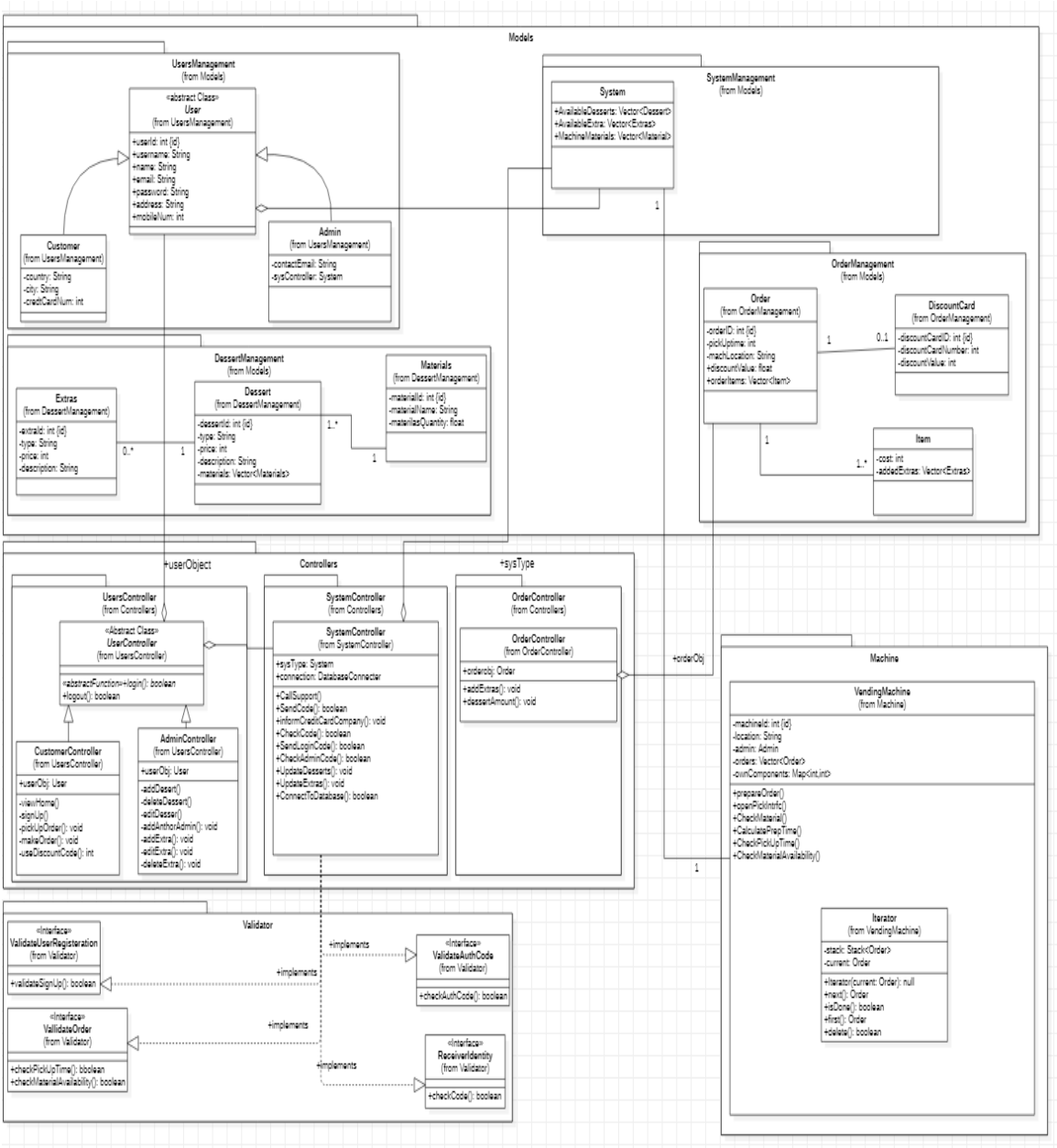
21. Vending Machine System should interact(alert) and Communicate with Support agency on lacking in cooking Materials in order to a good performance.

3. Class Diagram

1. Analysis Level



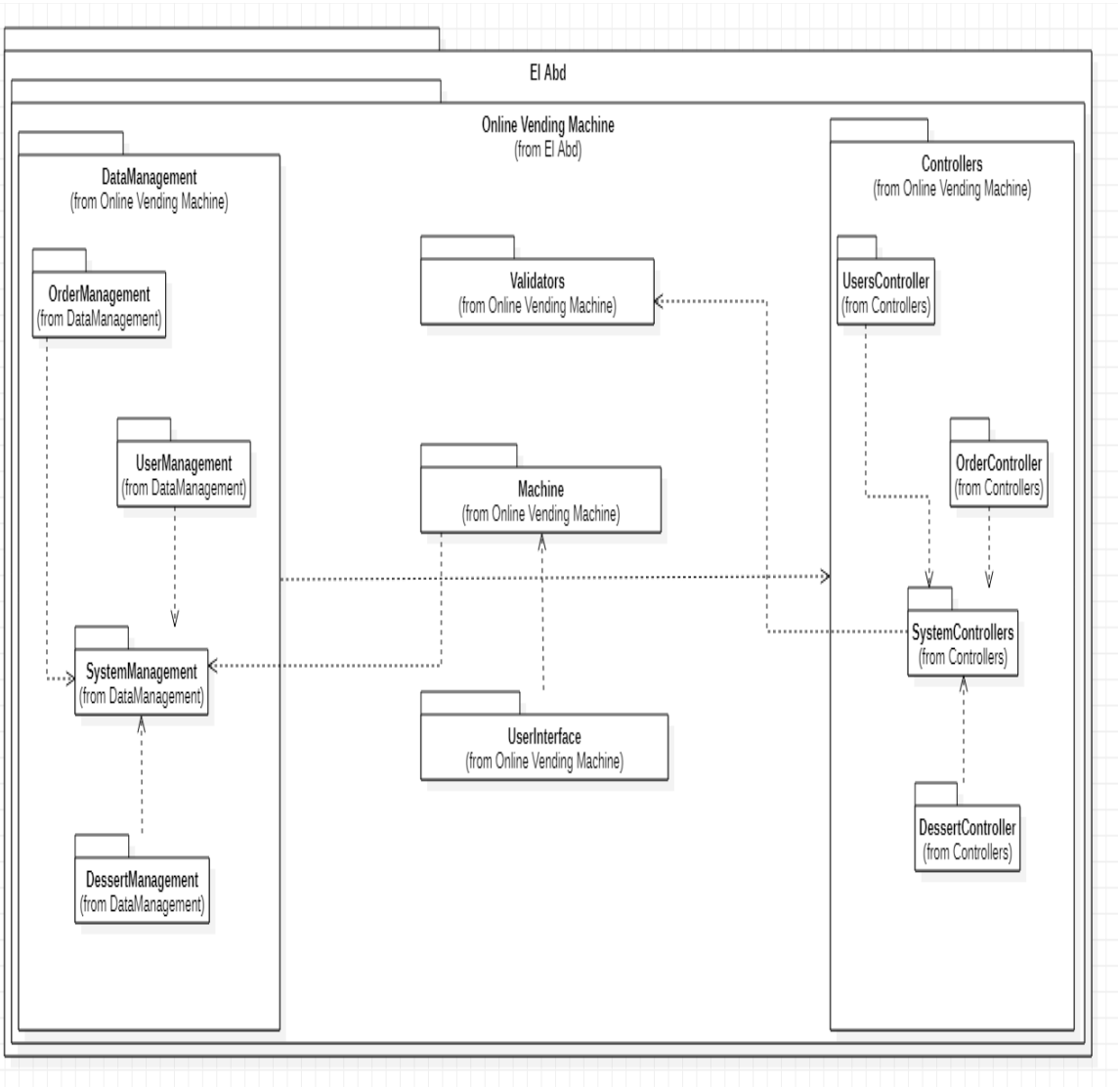
2. Design Pattern Class Diagram



- **We use 5 different types of design pattern:**
  - Abstract Factory Design Pattern (Creational Design Pattern).
  - MVC Design Pattern.
  - Iterator Design Pattern (Behavioral Design Pattern).
  - Proxy Design Pattern (Structural Design Pattern).
  - Private Data Class Design Pattern (Structural Design Pattern).

- **Abstract Factory Design Pattern:**
  - ✓ It is one of the most used patterns. This pattern has a best and easy way to create an object. We use it in user controller for abstracted function **Login** and Override inside two classes admin and customer controllers.
- **MVC Design Pattern**
  - ✓ It stands for Model-View-Controller pattern, our system design architecture based on mvc architecture, where models have data, controllers have the functionality and the views that is machine in our case act as interaction.
- **Iterator Design Pattern**
  - ✓ We use it to iterate collection of data, in our system, the vending machine needs iterator for orders and materials, for the orders, order needs iterator for items and extras it includes.
- **Proxy Design Pattern**
  - ✓ We use this design pattern for security, here we create object from each validator type interface to interface its functionality to the outer world. We use it in system controller link implementing the four interfaces for validating.
- **Private Data Class Design Pattern**
  - ✓ Here we create class with private data, where another class will extend them in order to access its attribute using getters for **encapsulation**. We use it in order and item as example, dessert and Materials.

4. Sub-System Decomposition



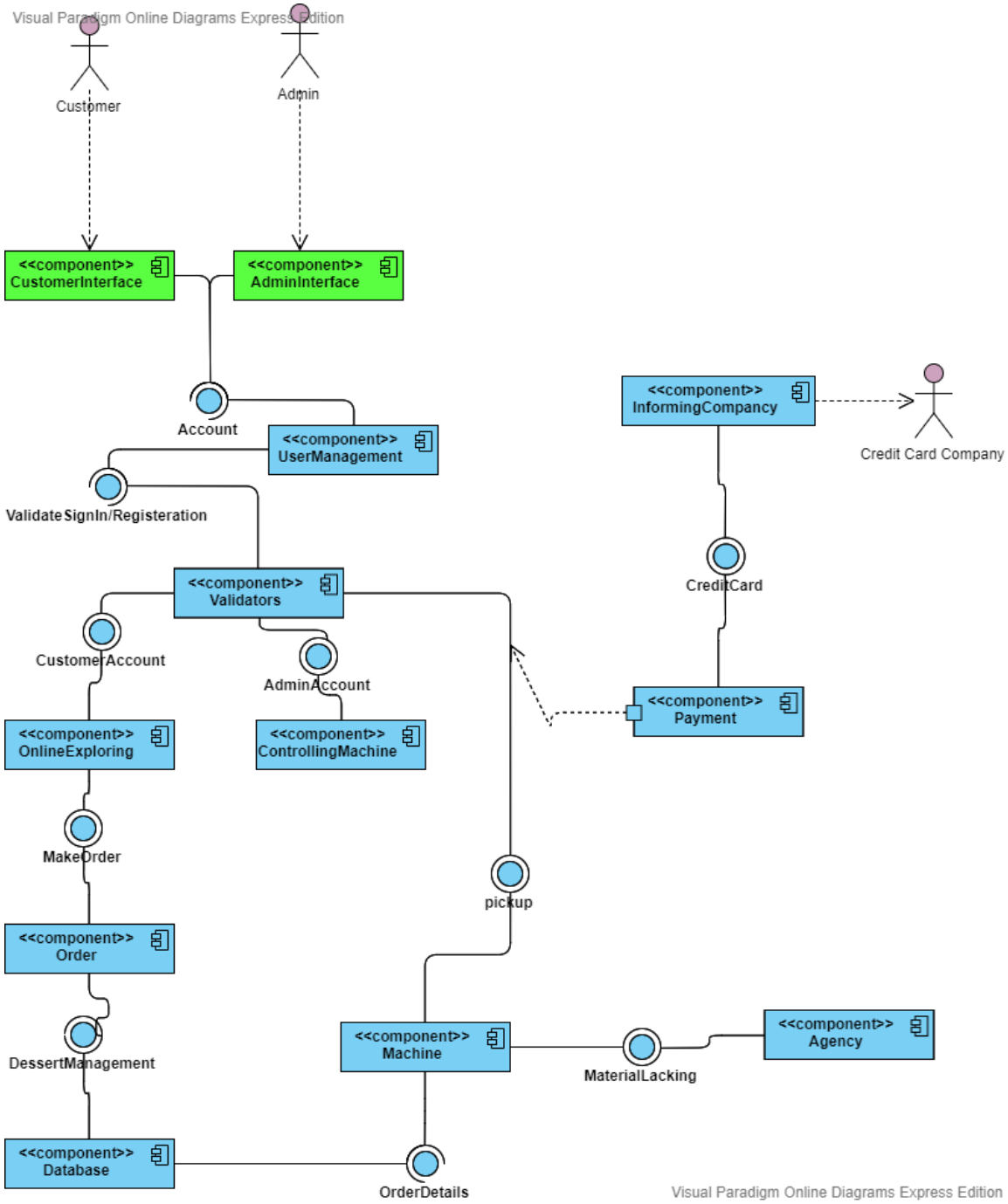
a- Performed Decomposition:

- As we use MVC Architecture, so we divide the packages into 3 main types models(DataManagement), Views(Machine) and Controllers. Each One Contains the specified packages that have close behavior. The remaining packages (User Interface - Validators) are packages that encapsulate classes which have the same aim.

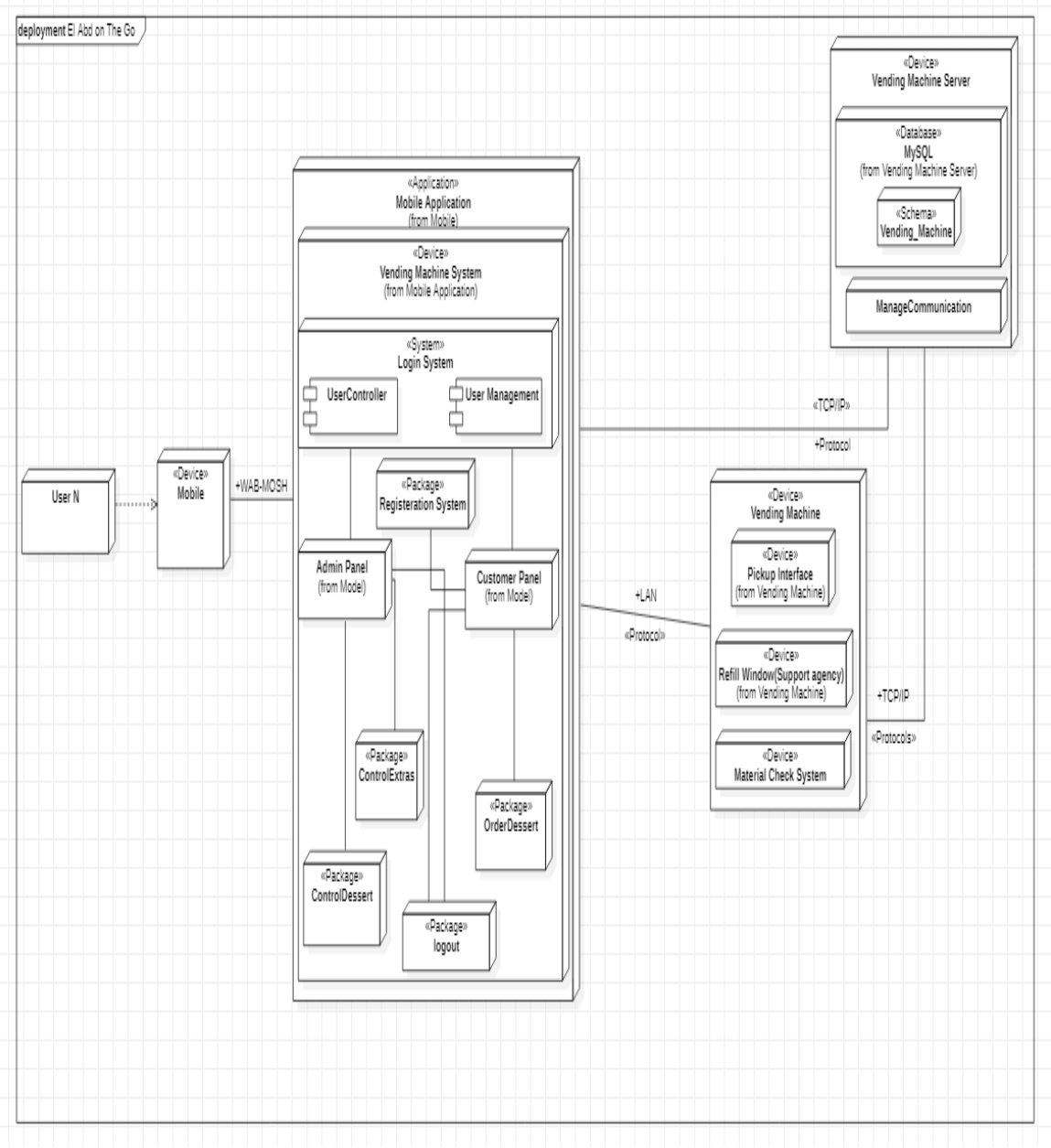
b- Cohesion and Coupling

- UserManagement and UserController: loosely coupled
- DessertManagement and OrderController: loosely coupled
- UserManagement and Validator: coupling
- UserController and Validator: coupling
- SystemController and Validator: coupling
- SystemManagement and UserController: coupling
- Machine and Controllers: very low coherence
- UserManagement and UserController: high coherence
- OrderController and Machine: high Coherence and loosely coupled

5. Component Diagram



6. Deployment Diagram



7. Design Goals

1- One of the design goals is **accuracy,** user who ordered order X in time T, must pick up order X not any other order at that given time.
This goal corresponds to the **performance** in the FURPS+ model, as the requested desserts must be made in the desired time.
Also corresponds **availability** in performance and **safety**.

- This can be achieved by making an instance from **order** class whenever an order is made, so each order will carry its details, also each order has its ID so system can call each order on time easily.
- Time of dessert preparation is calculated in advance and user can't select pickup time before it, after that certain time he is able to pick it up but there is authentication first to make sure that user will take the dessert(s) he ordered.

2- System should have **easy and user-friendly interface** in order to make it easier for the user to interact with the system.
This goal corresponds to **Usability** in the FURPS+ model.

- This can be achieved by putting images of desserts which is easier for the user to explore.
- Also, system will ask user for his address, mobile number, credit card number and name …etc. only once (when user signs up) and use this information whenever that user makes an order, so user won't have to enter this information every time he makes an order which is a kind of being user-friendly system.

- System can ask user to allow it to access the messages and write the incoming codes for him, so user won't have

to open the SMS's and copy the code himself when he signs in or picks up his order.

3- System must deal with the unwanted or invalid data in the best and fastest way, there is an action for all possible exceptions, which is easy for the user to deal with.
This goal corresponds to **reliability and robustness,** as system can still work correctly in the presence of exceptions and invalid as much as possible, like entering invalid pickup time, wrong authentication codes or invalid credit card number.

- This can be achieved by collecting large number of possibilities and discussing how to deal with these inputs, giving user a clear error message and allows him to re-enter only the wrong entered information, and not the whole thing from the beginning.
- Giving user suggestions, like suggested unique usernames when he signs up and minimum accepted pickup time, those could be very useful error messages.
- Some other errors require an action from the user where the system can't help him in, like if the system found money in user's credit card isn't enough so system has to cancel order till the user go and deposit money in his bank account.