VENDING MACHINE

PROJECT REPORT

18CSC202J/ 18AIC203J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY

(2018 Regulation)

II Year/ III Semester

Academic Year: 2022 -2023

By

RAKESH SANISETTY (RA2111026010222)

SUMANTH TALASILA (RA2111026010239)

Under the guidance of

Dr. M. Uma

Assistant Professor

Department of Computational Intelligence

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur

NOVEMBER 2022

BONAFIDE

This is to certify that **18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY project report** titled “**VENDING MACHINE”** is the bonafide work of **RAKESH SANISETTY (RA2111026010222) & SUMANTH TALASILA (RA2111026010239)** who undertook the task of completing the project within the allotted time.

Signature of the Guide	**Signature of the II Year Academic Advisor**
Dr. M. Uma	------------------------
Assistant Professor	**Professor and Head**
Department of CINTEL,	Department of CINTEL
SRM Institute of Science and Technology	SRM Institute of Science and Technology

About the course:-

18CSC202J/ 8AIC203J - Object Oriented Design and Programming are 4 credit courses with **L T P C as 3-0-2-4** (Tutorial modified as Practical from 2018 Curriculum onwards)

Objectives:

The student should be made to:

- Learn the basics of OOP concepts in C++
- Learn the basics of OOP analysis and design skills.
- Be exposed to the UML design diagrams.
- Be familiar with the various testing techniques

Course Learning Rationale (CLR): The purpose of learning this course is to:

1. Utilize class and build domain model for real-time programs
2. Utilize method overloading and operator overloading for real-time application development programs
3. Utilize inline, friend and virtual functions and create application development programs
4. Utilize exceptional handling and collections for real-time object-oriented programming applications
5. Construct UML component diagram and deployment diagram for design of applications
6. Create programs using object-oriented approach and design methodologies for real-time application development

Course Learning Outcomes (CLO): At the end of this course, learners will be able to:

1. Identify the class and build domain model
2. Construct programs using method overloading and operator overloading
3. Create programs using inline, friend and virtual functions, construct programs using standard templates
4. Construct programs using exceptional handling and collections
5. Create UML component diagram and deployment diagram
6. Create programs using object-oriented approach and design methodologies

Table 1: Rubrics for Laboratory Exercises

(Internal Mark Split-up :- As per Curriculum)

CLAP-1	5=(2(E-lab Completion) + 2(Simple Exercises)(from CodeZinger, and any other coding platform) + 1(HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-2	7.5=(2.0(E-lab Completion)+ 2.0 (Simple Exercises)(from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-3	7.5=(2.0(E-lab Completion(80 Pgms)+ 2.0 (Simple Exercises)(from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge)	**2 Mark -** E-lab Completion **80 Program** Completion from 10 Session (Each session min 8 program) **2 Mark -** Code to UML conversion GCR Exercises **3.5 Mark - Hacker Rank** Coding challenge completion
CLAP-4	5= 3 (Model Practical) + 2(Oral Viva)	• **3 Mark** – Model Test • **2 Mark** – Oral Viva
Total	25	

COURSE ASSESSMENT PLAN FOR OODP LAB

S.No	List of Experiments	Course Learning Outcomes (CLO)	Blooms Level	PI	No of Programs in each session
1.	Implementation of I/O Operations in C++	CLO-1	Understand	2.8.1	10
2.	Implementation of Classes and Objects in C++	CLO-1	Apply	2.6.1	10
3,	To develop a problem statement. 1. From the problem statement, Identify Use Cases and develop the Use Case model. 2. From the problem statement, Identify the conceptual classes and develop a domain model with a UML Class diagram.	CLO-1	Analysis	4.6.1	Mini Project Given
4.	Implementation of Constructor Overloading and Method Overloading in C++	CLO-2	Apply	2.6.1	10
5.	Implementation of Operator Overloading in C++	CLO-2	Apply	2.6.1	10
6.	Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams and Collaboration diagrams	CLO-2	Analysis	4.6.1	Mini Project Given
7.	Implementation of Inheritance concepts in C++	CLO-3	Apply	2.6.1	10
8.	Implementation of Virtual function & interface concepts in C++	CLO-3	Apply	2.6.1	10
9.	Using the identified scenarios in your project, draw relevant state charts and activity diagrams.	CLO-3	Analysis	4.6.1	Mini Project Given
10.	Implementation of Templates in C++	CLO-3	Apply	2.6.1	10
11.	Implementation of Exception of Handling in C++	CLO-4	Apply	2.6.1	10
12.	Identify the User Interface, Domain objects, and Technical Services. Draw the partial layered, logical architecture diagram with UML package diagram notation such as Component Diagram, Deployment Diagram.	CLO-5	Analysis	4.6.1	Mini Project Given
13.	Implementation of STL Containers in C++	CLO-6	Apply	2.6.1	10
14.	Implementation of STL associate containers and algorithms in C++	CLO-6	Apply	2.6.1	10
15.	Implementation of Streams and File Handling in C++	CLO-6	Apply	2.6.1	10

LIST OF EXPERIMNENTS FOR UML DESIGN AND MODELLING:

To develop a mini-project by following the exercises listed below.

1. To develop a problem statement.
2. Identify Use Cases and develop the Use Case model.
3. Identify the conceptual classes and develop a domain model with UML Class diagram.
4. Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams.
5. Draw relevant state charts and activity diagrams.
6. Identify the User Interface, Domain objects, and Technical services. Draw the partial layered, logical architecture diagram with UML package diagram notation.

Suggested Software Tools for UML:

StarUML, Rational Suite, Argo UML (or) equivalent, Eclipse IDE and Junit

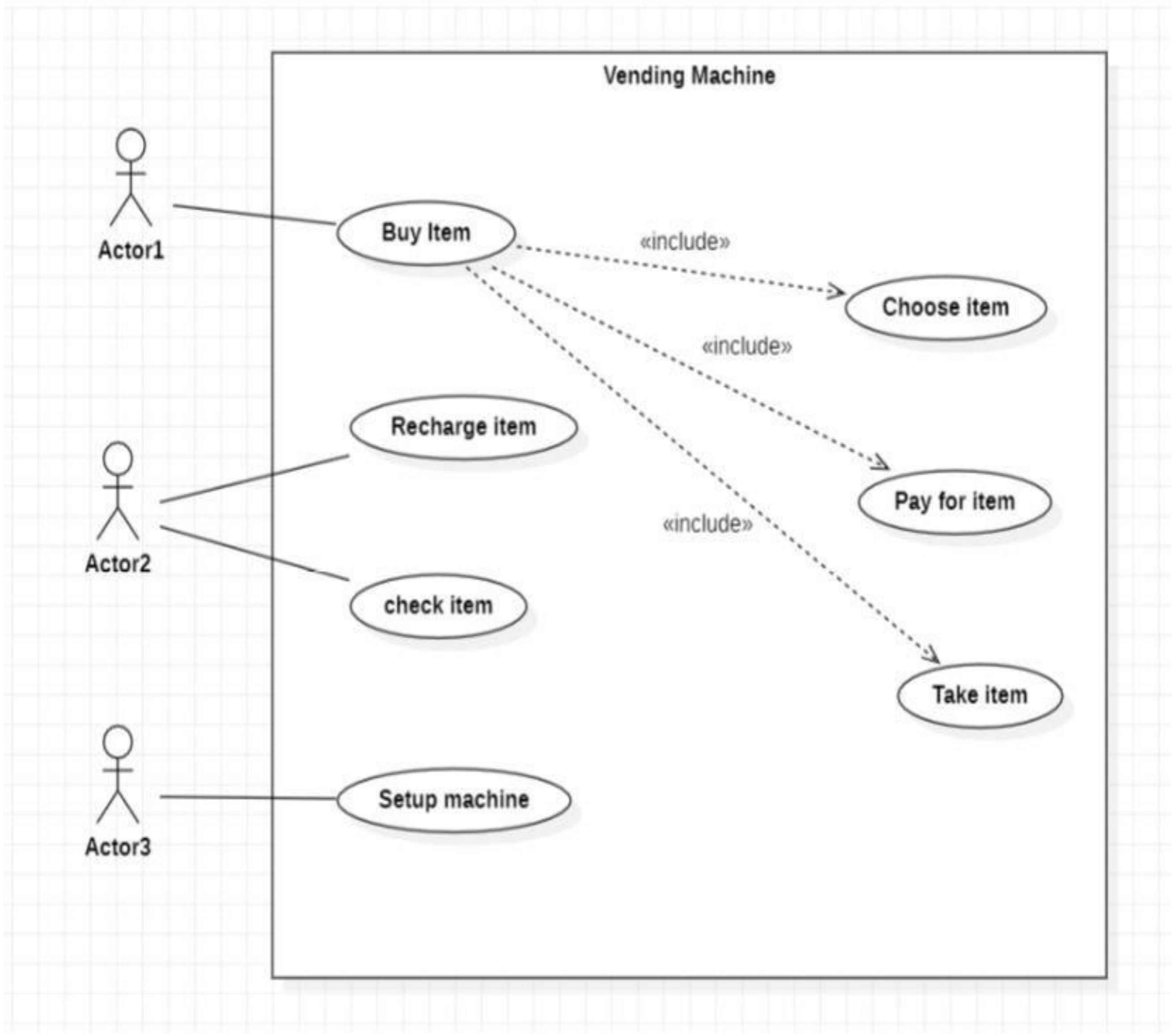
1. PROJECT DESCRIPTION

Have you ever skipped breakfast on a busy morning and went into the office feeling distracted by hunger? If so, you’re not alone. The odds are that many of your coworkers and employees have experienced the same thing. Installing a vending machine in your workplace provides convenient access to energizing snacks to help relieve hunger and boost your team’s productivity.
Relieving workplace hunger is just one of many benefits of adding a vending machine to your business.
Vending machines are a convenient way to feed a large staff with minimal overhead. They require a fraction of the budget to run a concession stand or cafeteria. Modern vending machines are built to conserve energy and only require a small amount of electricity to run 24 hours a day.

So, we now design UML diagrams for the vending machine.

USE CASE DIAGRAM

USE CASE DIAGRAM FOR VENDING MACHINE



2. USE CASE DIAGRAM

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve
- The scope of your system

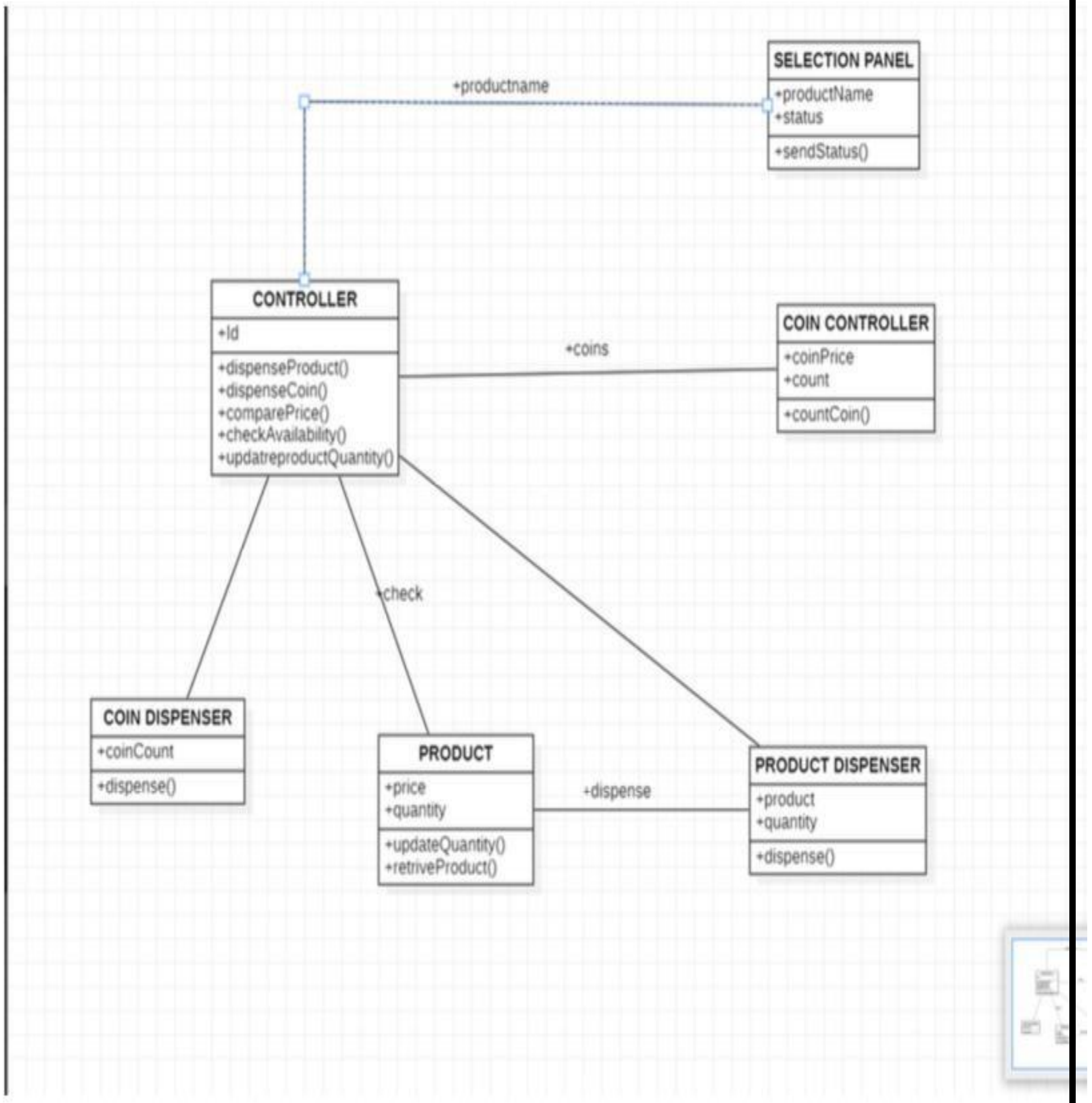
We have three actors :
Actor 1 is the customer who will buy the items (choosing the item, paying for it, and take it from the dispenser).

Actor 2 is the one who oversees the vending machine is empty or not and recharges the vending machine when required.

Actor 3 is the mechanic, the one who sets up the system and repairs if any trouble occurred.

CLASS DIAGRAM

CLASS DIAGRAM FOR VENDING MACHINE



3. CLASS DIAGRAM

Class diagram describes the attributes and operations of a class and the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

We have classes like controller, selection panel, product dispenser etc., where we also have attributes and methods for each class specifying the task that’s going to be done.

The class in the diagram are :

- Controller
- Selection Panel
- Coin Controller
- Product Dispenser
- Product
- Coin Dispenser

The attributes of the Vending machine:

Id, productname, status, coin price, count, product, quantity, coincount

The methods of the Vending machine:

dispenseProduct(),dispenseCoin(),comparePrise(),Checkavali blity() ,dispense()

CODE FOR CONTROLLER CLASS:

```
/**

* Project Untitled

*/


#include "CONTROLLER.h"

/**

* CONTROLLER implementation

*/


void CONTROLLER::dispenseProduct() {


}

void CONTROLLER::dispenseCoin() {

}

void CONTROLLER::comparePrice() {

}

void CONTROLLER::checkAvailability() {

}

void CONTROLLER::updatreproductQuantity() {
}
```

CODE FOR SELECTION PANEL:

```
/**
* Project Untitled
*/

#include "SELECTION PANEL.h"
/**
* SELECTION PANEL implementation
*/

void SELECTION PANEL::sendStatus() {
}
```

CODE FOR PRODUCT:

```
/**
* Project Untitled
*/

#include "PRODUCT.h"
/**
```

```
* PRODUCT implementation
*/

void PRODUCT::updateQuantity() {
}
void PRODUCT::retriveProduct() {
}
```

CODE FOR COIN DISPENSER:

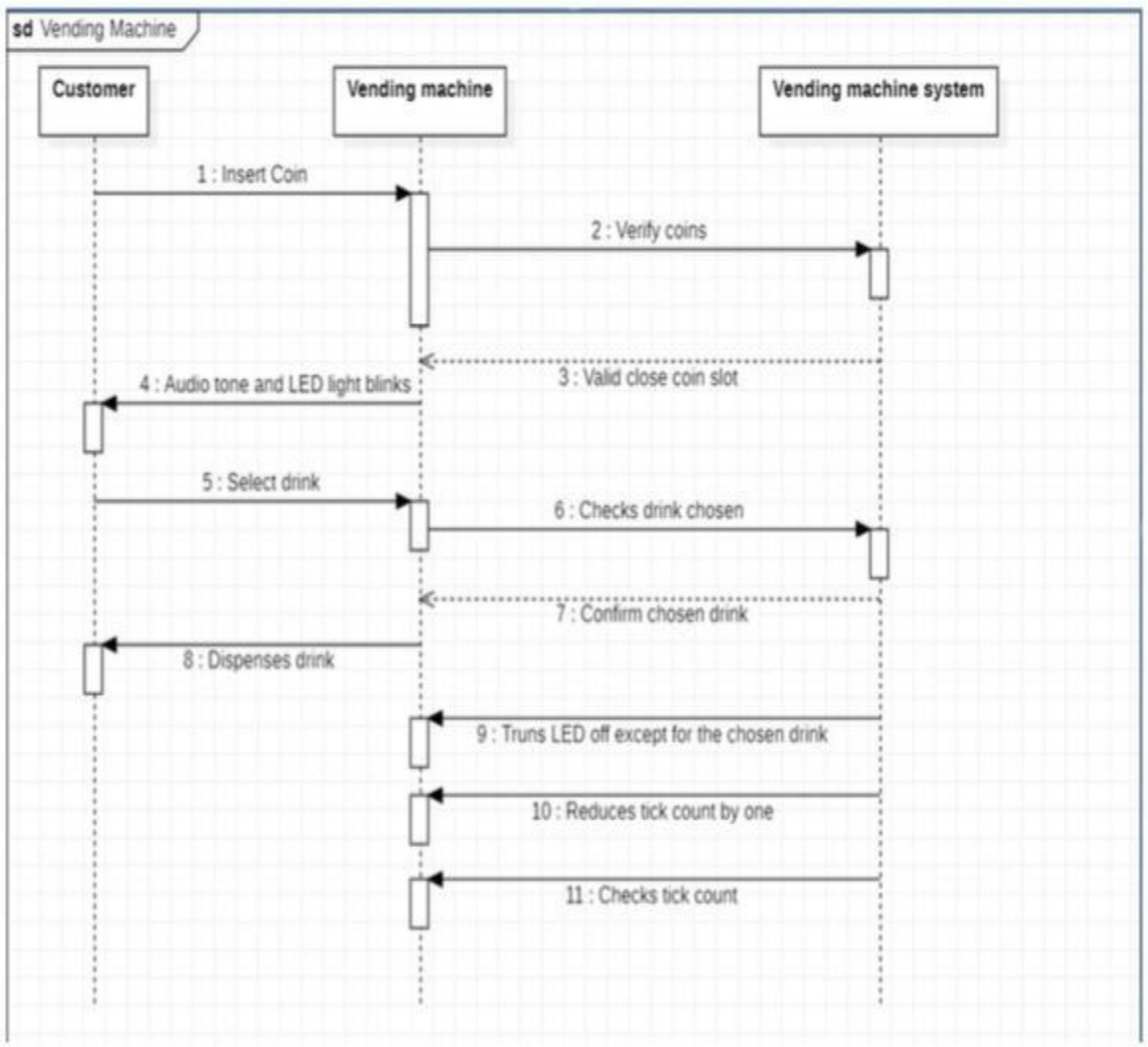
```
/**
* Project Untitled
*/

#include "COIN DISPENSER.h"
/**
* COIN DISPENSER implementation
*/

void COIN DISPENSER::dispense() {
}
```

SEQUENCE DIAGRAM

SEQUENCE DIAGRAM FOR VENDING MACHINE



4. SEQUENCE DIAGRAM

A sequence diagram is the most used interaction diagram. Interaction diagram – An interaction diagram is used to show the interactive behavior of a system.

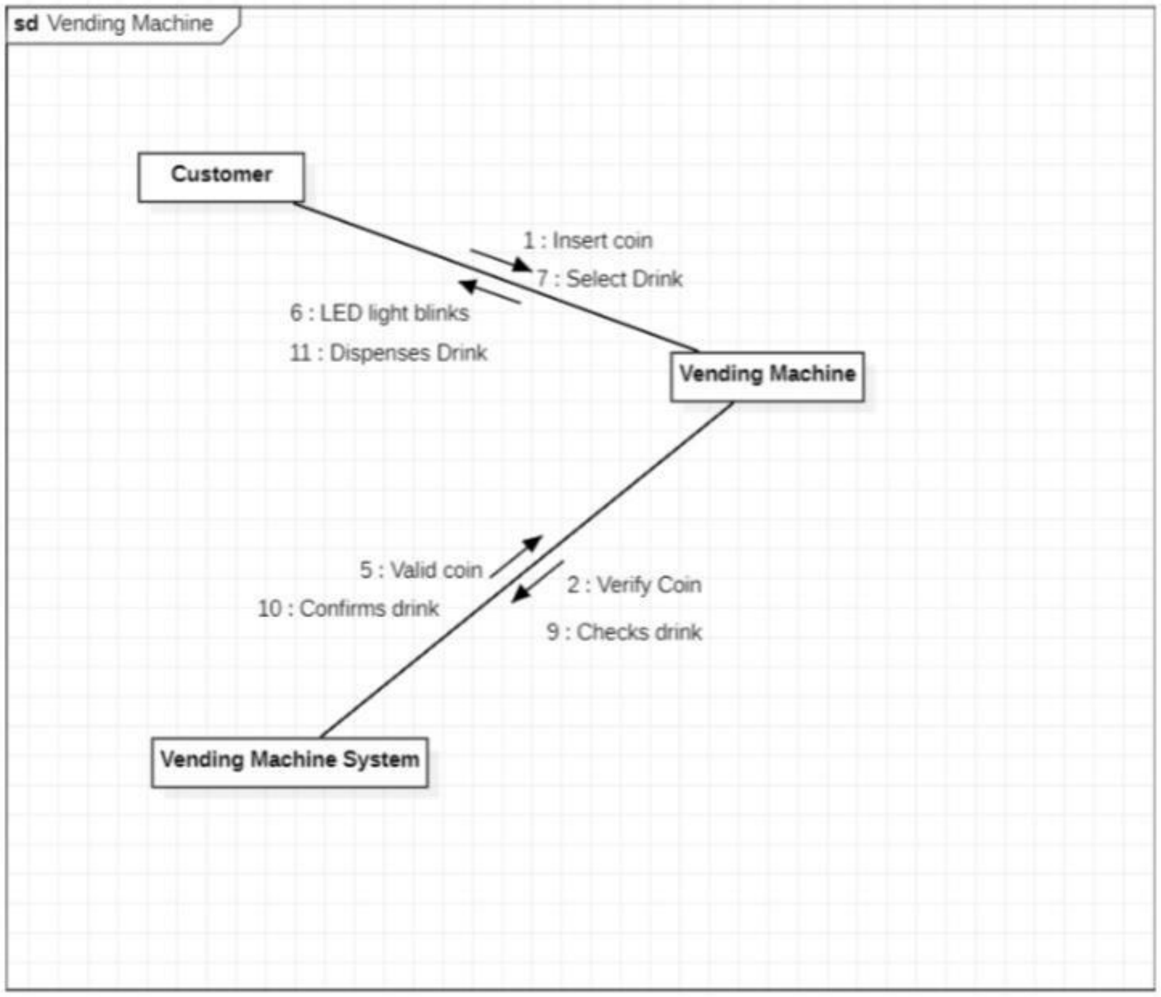
We have actors as a customer who are going to insert the coin in the vending machine and the machine will verify the coin in its system. After that we will be given a choice to select our menu/drink and the machine will put it in the dispenser. If the machine has run out of drinks/snacks it will notify the vending machine system to recharge/restock the stuff.

Sequence Diagrams captures:

- The interaction that takes place in a collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams)
- High-level interactions between user of the system and the system, between the system and other systems, or between subsystems (sometimes known as system sequence diagrams).

COMMUNICATION DIAGRAM

COLLABORATION DIAGRAM FOR VENDING MACHINE



5. COMMUNICATION DIAGRAM.

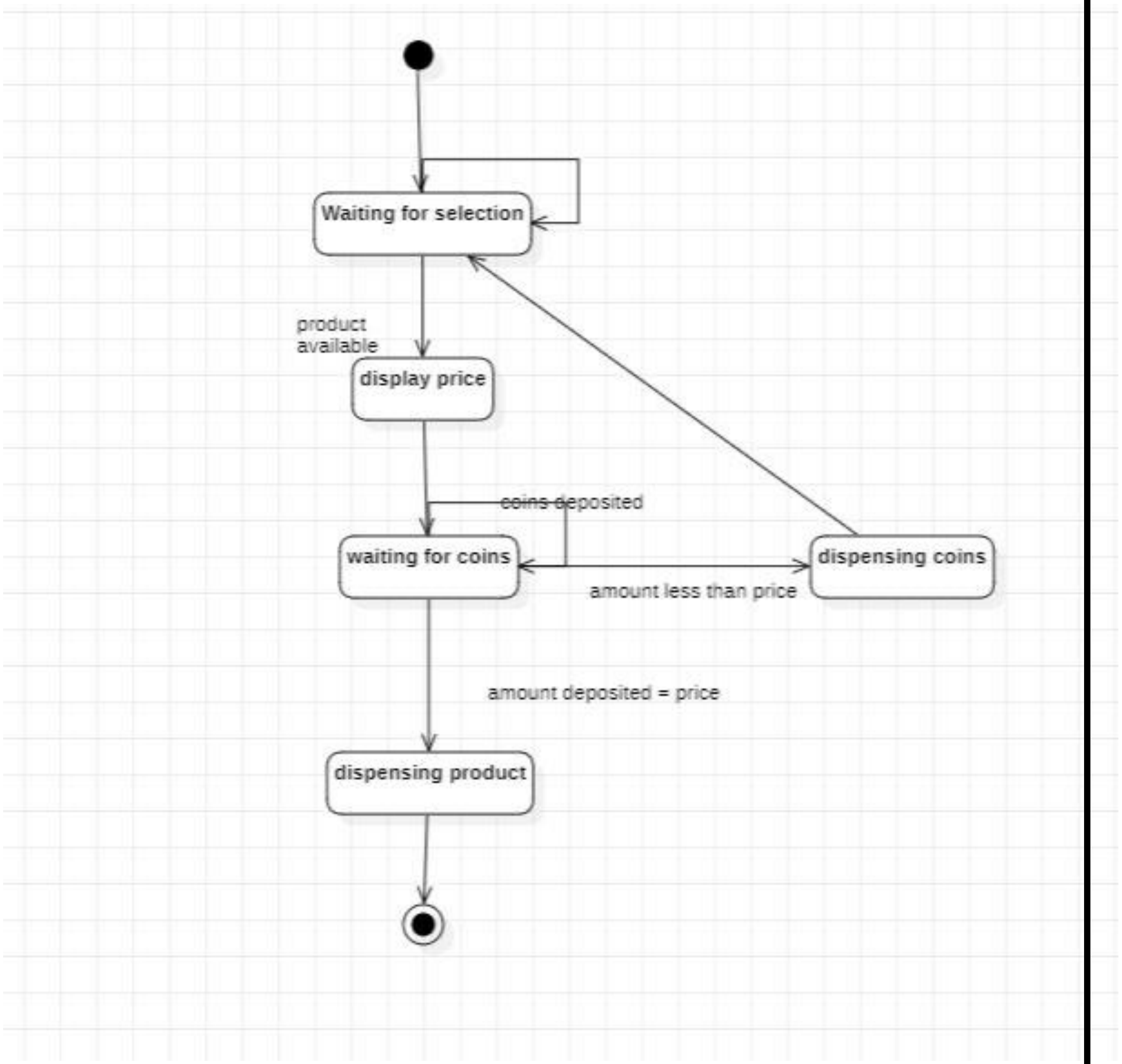
The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently.

Purpose of Communication Diagram:

- Model message passing between objects or roles that deliver the functionalities of use cases and operations
- Model mechanisms within the architectural design of the system
- Capture interactions that show the past messages between objects and roles within the collaboration scenario
- Model alternative scenarios within use cases or operations that involve the collaboration of different objects and interactions
- Support the identification of objects (hence classes), and their attributes (parameters of message) and operations (messages) that participate in use cases.

STATE CHART DIAGRAM

STATE CHART DIAGRAM FOR VENDING MACHINE



6. STATE CHART DIAGRAM

A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as State machines and State-chart Diagrams. We have space for dispenser coins, dispenser for drinks, we check the availability for the products we are going to select, giving us the option to select the menu.

Coponents of state chart diagram:

- Initial state – We use a black filled circle represent the initial state of a System or a class.
- Initial state – We use a black filled circle represent the initial state of a System or a class.
- State – We use a rounded rectangle to represent a state. A state represents the conditions or circumstances of an object of a class at an instant of time.
- Fork – We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent state and outgoing arrows towards the newly created states. We use the fork notation to represent a state splitting into two or more concurrent states.

- Join – We use a rounded solid rectangular bar to represent a Join notation with incoming arrows from the joining states and outgoing arrow towards the common goal state. We use the join notation when two or more states concurrently converge into one on the occurrence of an event or events.
- Final state – We use a filled circle within a circle notation to represent the final state in a state machine diagram.

ACTIVITY DIAGRAM

7. ACTIVITY DIAGRAM

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

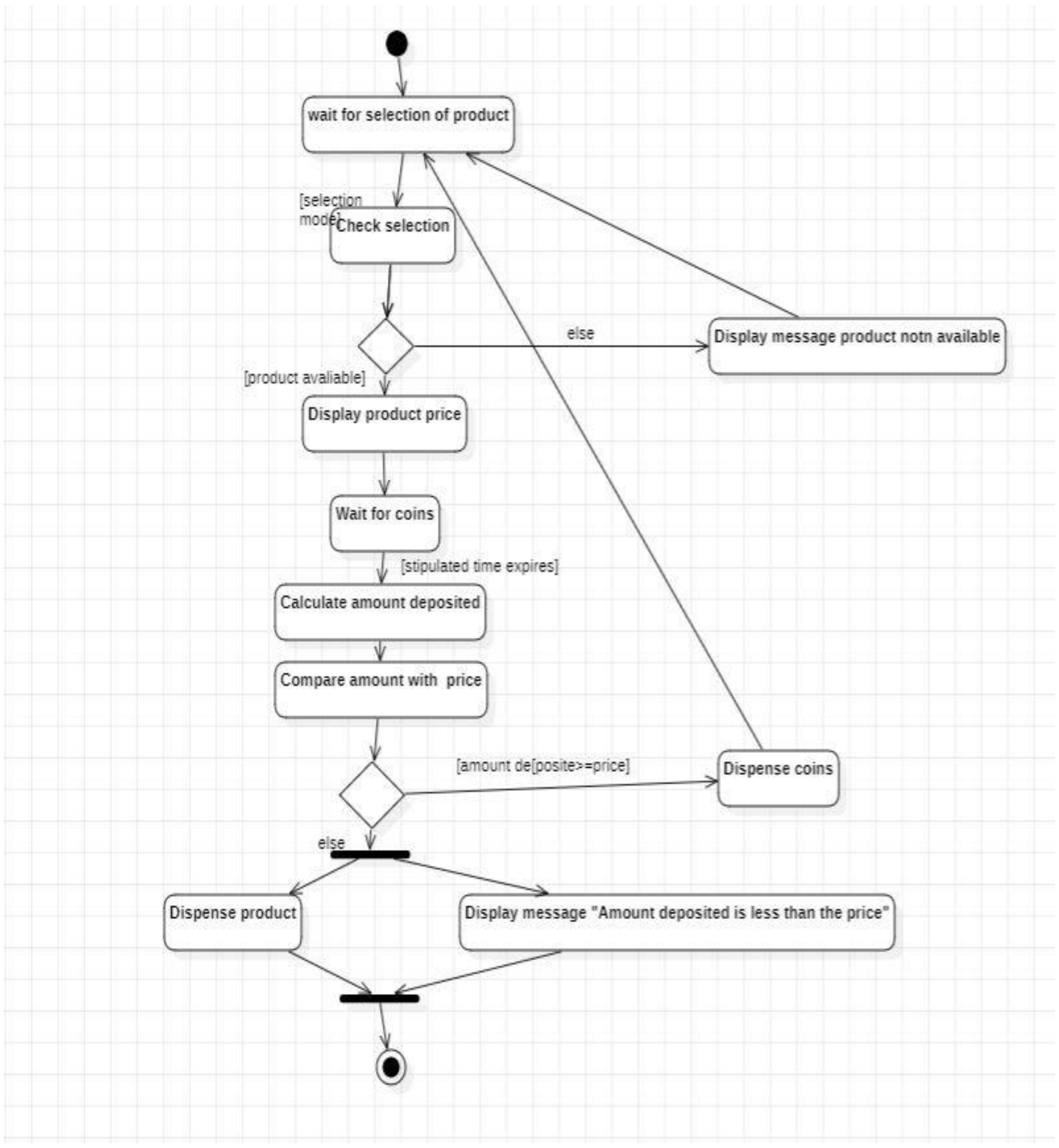
We have an actor input for the selection of the product where we check selection in the system setup. Later we go through alternative or extension flow for if/else case having to display price or the unavailability of the product in the system. We also have basic flow of transitions and alternative flows if required. Then we have parallel activities happening. And finally, we get down to the post condition where we end our UML Diagram.

We use Activity diagram because :

- Identify candidate use cases, through the examination of business workflows
- Identify pre- and post-conditions (the context) for use cases
- Model workflows between/within use cases
- Model complex workflows in operations on objects

- Model in detail complex activities in a high level activity Diagram

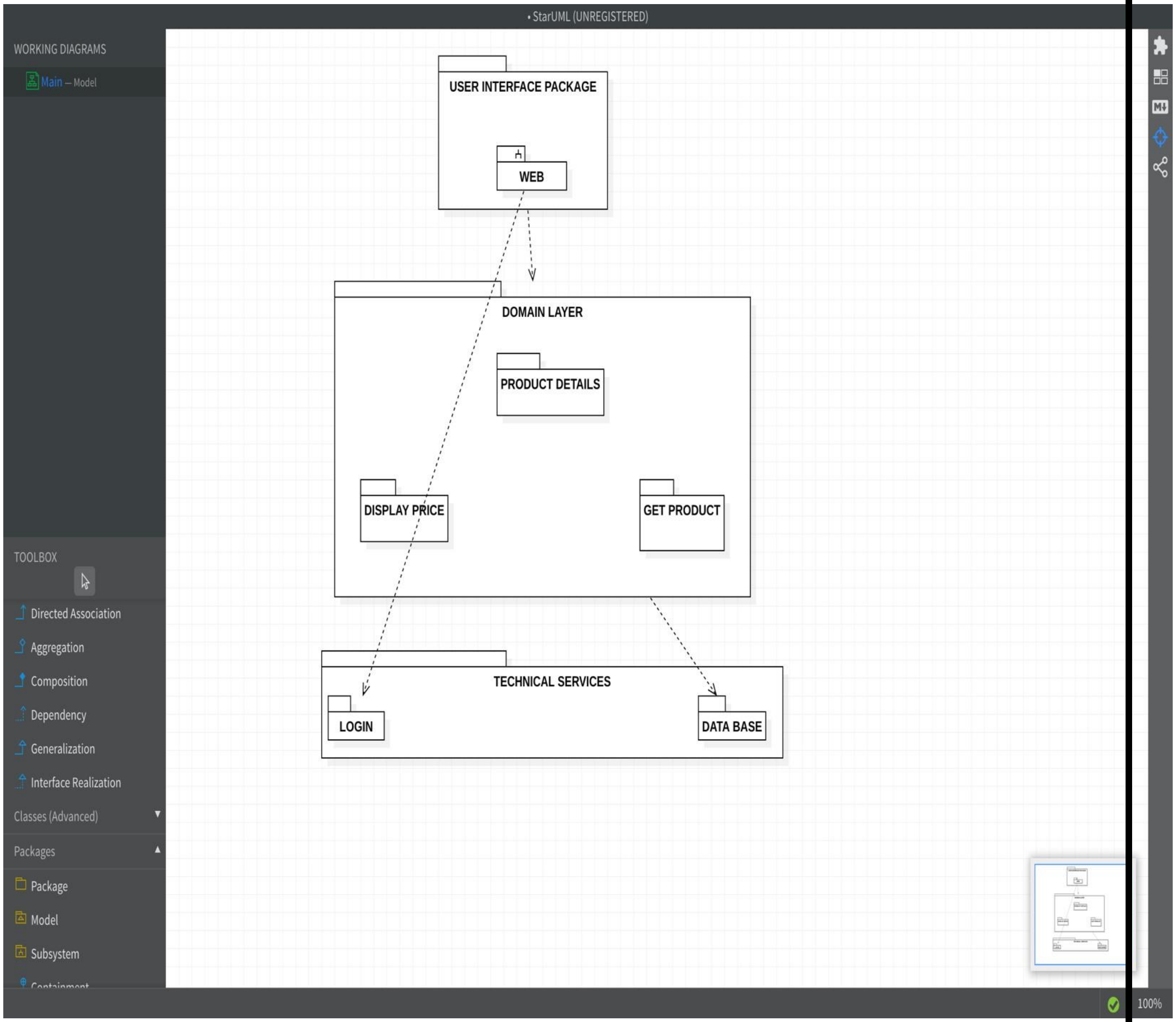
ACTIVITY DIAGRAM FOR VENDING MACHINE



wait for selection of product
[selection mode]
Check selection
else
Display message product notn available
[product avaliable]
Display product price
Wait for coins
[stipulated time expires]
Calculate amount deposited
Compare amount with price
[amount de[posite>=price]
Dispense coins
else
Dispense product
Display message "Amount deposited is less than the price"

PACKAGE DIAGRAM

PACKAGE DIAGRAM FOR VENDING MACHINE



8. PACKAGE DIAGRAM

Package diagrams are structural diagrams used to show the organization and arrangement of various model elements in the form of packages. A package is a grouping of related UML elements, such as diagrams, documents, classes, or even other packages. Each element is nested within the package, which is depicted as a file folder within the diagram, then arranged hierarchically within the diagram. Package diagrams are most used to provide a visual organization of the layered architecture within any UML classifier, such as a software system.

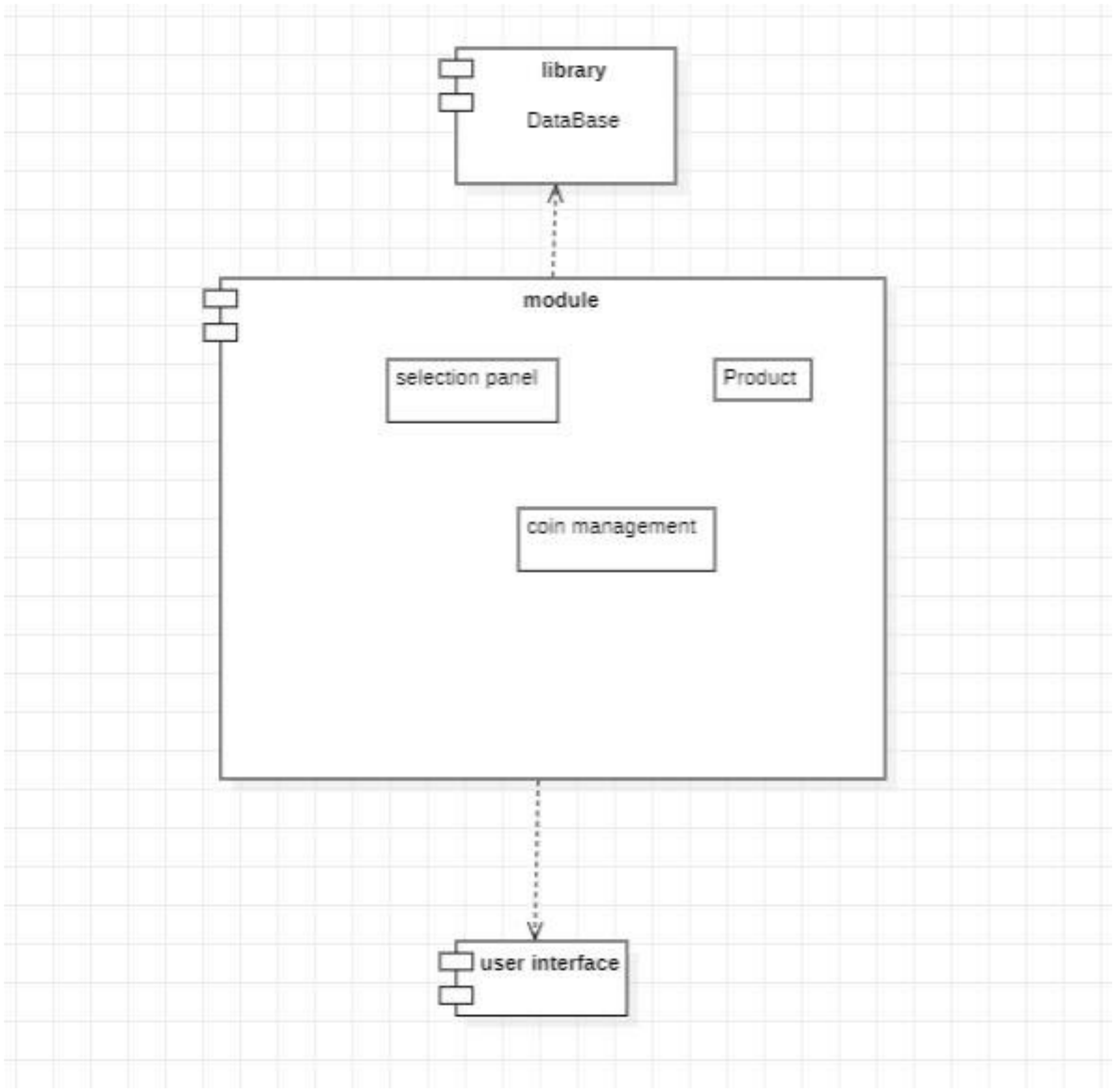
Purpose of Package Diagrams:

Package diagrams are used to structure high level system elements. Packages are used for organizing large system which contains diagrams, documents, and other key deliverables.

- Package Diagram can be used to simplify complex class diagrams, it can group classes into packages.
- A package is a collection of logically related UML elements.
- Packages are depicted as file folders and can be used on any of the UML diagrams.

COMPONENT DIAGRAM

COMPONENT DIAGRAM FOR VENDING MACHINE



9. COMPONENT DIAGRAM

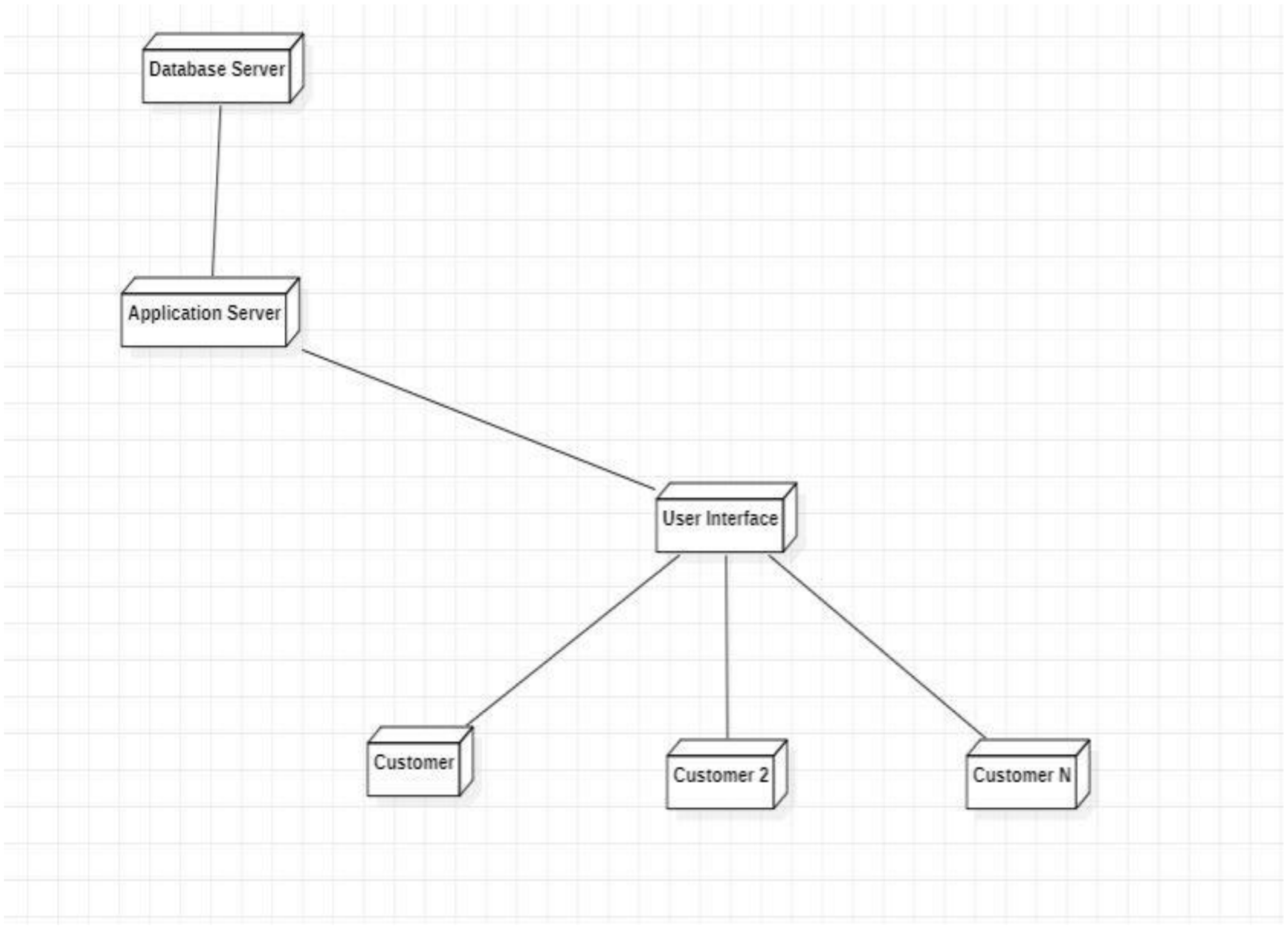
Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system. We have a rectangle box with name and the icon to specify the component.

We have components in it, along with interface where it has two subdivisions namely required interface and provided interface.

We also have sub systems which again consists components and interfaces in it.

DEPOLYMENT DIAGRAM

DEPLOYMENT DIAGRAM FOR VENDING MACHINE



10. DEPLOYMENT DIAGRAM

Deployment diagrams are used to visualize the hardware processors/ nodes/ devices of a system, the links of communication between them and the placement of software files on that hardware. We usually have nodes, artifacts, communication association, devices, deployment specifications.

Purpose of Deployment Diagrams :

- They show the structure of the run-time system
- They capture the hardware that will be used to implement the system and the links between different items of hardware.
- They model physical hardware elements and the communication paths between them
- They can be used to plan the architecture of a system.
- They are also useful for Document the deployment of software components or nodes

11. CONCLUSION

Vending Machine these days has become an essential part of our lives. Many individuals use these machines for easy access to food/drinks reducing the amount of time spent standing in the canteen and to minimize the use of budget in many companys.

12. REFERENCE

http://www.programsformca.com/2012/03/umldiagrams-vending-machine.html

https://docs.staruml.io/working-with-uml-diagrams/classdiagram