

Software Engineering Lab

ETMA 252

Faculty: **Ms. Garima Gupta**

Name: **Syeda Reeha Quasar**

Roll No.: **14114802719**

Semester: **5**



Maharaja Agrasen Institute of Technology, PSP Area,
Sector – 22, Rohini, New Delhi – 110085



MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

VISION

To Produce “Critical thinkers of Innovative Technology”

MISSION

To provide an excellent learning environment across the computer science discipline to inculcate professional behavior, strong ethical values, innovative research capabilities and leadership abilities which enable them to become successful entrepreneurs in this globalized world.

1. To nurture an **excellent learning environment** that helps students to enhance their problem solving skills and to prepare students to be lifelong learners by offering a solid theoretical foundation with applied computing experiences and educating them about their **professional, and ethical responsibilities**.
2. To establish **Industry-Institute Interaction**, making students ready for the industrial environment and be successful in their professional lives.
3. To promote **research activities** in the emerging areas of technology convergence.
4. To build engineers who can look into technical aspects of an engineering solution thereby setting a ground for producing successful **entrepreneur**.

VISION

To nurture young minds in a learning environment of high academic value and imbibe spiritual and ethical values with technological and management competence.

MISSION

The Institute shall endeavor to incorporate the following basic missions in the teaching methodology:

Engineering Hardware - Software Symbiosis

Practical exercises in all Engineering and Management disciplines shall be carried out by Hardware equipment as well as the related software enabling deeper understanding of basic concepts and encouraging inquisitive nature.

Life - Long Learning

The Institute strives to match technological advancements and encourage students to keep updating their knowledge for enhancing their skills and inculcating their habit of continuous learning.

Liberalization and Globalization

The Institute endeavors to enhance technical and management skills of students so that they are intellectually capable and competent professionals with Industrial Aptitude to face the challenges of globalization.

Diversification

The Engineering, Technology and Management disciplines have diverse fields of studies with different attributes. The aim is to create a synergy of the above attributes by encouraging analytical thinking.

Digitization of Learning Processes

The Institute provides seamless opportunities for innovative learning in all Engineering and Management disciplines through digitization of learning processes using analysis, synthesis, simulation, graphics, tutorials and related tools to create a platform for multi-disciplinary approach.



MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

VISION

To Produce “Critical thinkers of Innovative Technology”

MISSION

To provide an excellent learning environment across the computer science discipline to inculcate professional behavior, strong ethical values, innovative research capabilities and leadership abilities which enable them to become successful entrepreneurs in this globalized world.

1. To nurture an **excellent learning environment** that helps students to enhance their problem solving skills and to prepare students to be lifelong learners by offering a solid theoretical foundation with applied computing experiences and educating them about their **professional, and ethical responsibilities**.
2. To establish **Industry-Institute Interaction**, making students ready for the industrial environment and be successful in their professional lives.
3. To promote **research activities** in the emerging areas of technology convergence.
4. To build engineers who can look into technical aspects of an engineering solution thereby setting a ground for producing successful **entrepreneur**.

Lab Objective:

The Software Engineering Lab has been developed to impart state-of-the-art knowledge on Software Engineering practical aspects and UML in an interactive manner. This course also provide scope to students where they can solve small, real life problems and present case studies, can further demonstrate practical applications of different concepts.

Course Outcomes

At the end of the course, a student will be able to:

- 1.** To elicit, analyse and specify software requirements of given problem statement.
- 2.** Analyse and translate a specification in to a function oriented diagram.
- 3.** To perform user's view analysis and structural view analysis of system.
- 4.** To draw behavioural view of system.
- 5.** To build implementation and environmental view of system.
- 6.** Create the test cases in accordance with the testing plan and to verify and validate the implementation of the software specifications.

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Outcome Based Learning Course Outcomes (Revision)

Subject:	Software Engineering Lab	Max Marks External	60
Subject Code:	ETCS 353	Max Marks Internal	40
Total Credit:		Evaluation Scheme	
Contact Hours:		Theory/ Lab	Lab

Course Objectives:

The Software Engineering Lab has been developed to impart state-of-the-art knowledge on Software Engineering practical aspects and UML in an interactive manner. This course also provide scope to students where they can solve small, real life problems and present case studies that can further demonstrate practical applications of different concepts.

S.NO	Course Outcomes	BL	PO	PI Code
ETCS353.1	Able to elicit, analyze and specify software requirements through a productive working relationship with various stakeholders of the project.	1,2,4	PO1,PO2,PO3,PO7,PO8,PO9,PO10,PO11,PO12	1.7.1,2.5. (1,2), 2.6(2,3),2.7.1,3.5. (1,2,3,5,6),7.3.1,8.3. 1,9.4.(1,2),9.6.1,10. 4.(1,2),11.4.(1,2), 11.5.1,12.4.1, 12.6.2
ETCS353.2	Analyze and design function-oriented, user's view, structural view diagram using SRS.	4,6	PO2,PO3,PO4,PO5,PO9,PO10	2.7.2,3.6.(1,2,3), 3.7.23.8(1,2,3),4.5.1 , 4.6.(1,3,4),5.4.2,9.4. 2,10.4.3,10.6.1 - -
ETCS353.3	Build behavioral, implementation and environmental view of system.	6	PO1,PO2,PO3,PO4,PO5,PO7,PO9,PO10	1.7.1,2.7.2,3.6.(1,2, 3),3.7.2,3.8(1,2,3),4 .5.1, 4.6.(1,3,4),5.4.2,7.3. 1,7.4.1,9.4.2,9.5(2,3),10.4.3, 10.6.1
ETCS353.4	Create the test cases in accordance with the testing plan to inspect and justify the implementation of the software specifications.	3,6	PO1,PO4,PO5,PO9,PO11	1.7.1,4.6.(1,2),5.4.2, 5.6.1,9.4.2, 9.5(2,3),11.6.2

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	1	3	3	-	-	-	1	1	2	1	2	2
CO2	-	1	3	2	1	-	-	-	1	2	-	-
CO3	1	1	3	2	1	-	2	-	2	2	-	-
CO4	1	-	-	1	2	-	-	-	2	-	1	-
Average	0.75	1.25	2.25	1.25	1	-	0.75	0.25	1.75	1.25	0.75	0.5

	PSO1	PSO2	PSO3
CO1	2	2	-
CO2	1	3	-
CO3	1	2	-
CO4	1	3	-
Average	1.25	2.5	-

Name of the Coordinator: Ms. Karuna Middha

Name and Signature of the Subject teachers

Signature

1. Mr. Moolchand Sharma

Date of Revision of Course Outcomes

2.

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
C01	1.7.1	2.5.(1,2), 2.6(2,3), 2.7.1	3.5.(1,2,3,5,6)	-	-	-	7.3.1	8.3.1	9.4.(1,2), 9.6.1	10.4.(1,2)	11.4.(1,2), 11.5.1	12.4.1, 12.6.2
C02	-	2.7.2	3.6.(1,2,3), 3.7.2,3.8(1,2,3)	4.5.1, 4.6.(1,3,4)	5.4.2	-	-	-	9.4.2	10.4.3,10.6.1	-	-
C03	1.7.1	2.7.2	3.6.(1,2,3), 3.7.2,3.8(1,2,3)	4.5.1, 4.6.(1,3,4)	5.4.2	-	7.3.1,7.4.1	-	9.4.2, 9.5(2,3)	10.4.3, 10.6.1	-	-
C04	1.7.1	-	-	4.6.(1,2)	5.4.2, 5.6.1	-	-	-	9.4.2, 9.5(2,3)	-	11.6.2	-

	PSO1	PSO2	PSO3
CO1	13.1(2,3),13.2(2,3)	14.1.(2,3)	-
CO2	13.2.2	14.1.3, 14.2.3	-
CO3	13.2.2	14.1.3	-
CO4	13.3.(1,2)	14.1.3, 14.2.(1,2,3)	-

Department of Computer Science and Engineering

Rubrics for Lab Assessment

Rubrics		0	1	2	3
		Missing	Inadequate	Needs Improvement	Adequate
R1	Is able to identify the problem to be solved and define the objectives of the experiment.	No mention is made of the problem to be solved.	An attempt is made to identify the problem to be solved but it is described in a confusing manner, objectives are not relevant, objectives contain technical/ conceptual errors or objectives are not measurable.	The problem to be solved is described but there are minor omissions or vague details. Objectives are conceptually correct and measurable but may be incomplete in scope or have linguistic errors.	The problem to be solved is clearly stated. Objectives are complete, specific, concise, and measurable. They are written using correct technical terminology and are free from linguistic errors.
R2	Is able to design a reliable experiment that solves the problem.	The experiment does not solve the problem.	The experiment attempts to solve the problem but due to the nature of the design the data will not lead to a reliable solution.	The experiment attempts to solve the problem but due to the nature of the design there is a moderate chance the data will not lead to a reliable solution.	The experiment solves the problem and has a high likelihood of producing data that will lead to a reliable solution.
R3	Is able to communicate the details of an experimental procedure clearly and completely.	Diagrams are missing and/or experimental procedure is missing or extremely vague.	Diagrams are present but unclear and/or experimental procedure is present but important details are missing.	Diagrams and/or experimental procedure are present but with minor omissions or vague details.	Diagrams and/or experimental procedure are clear and complete.
R4	Is able to record and represent data in a meaningful way.	Data are either absent or incomprehensible.	Some important data are absent or incomprehensible.	All important data are present, but recorded in a way that requires some effort to comprehend.	All important data are present, organized and recorded clearly.
R5	Is able to make a judgment about the results of the experiment.	No discussion is presented about the results of the experiment .	A judgment is made about the results, but it is not reasonable or coherent.	An acceptable judgment is made about the result, but the reasoning is flawed or incomplete.	An acceptable judgment is made about the result, with clear reasoning. The effects of assumptions and experimental uncertainties are considered.

INDEX

Sr. no.	Program Name	Date	R1	R2	R3	R4	R5	Total Marks	Signature
1.	Write down the problem statement for a suggested system of relevance.	23-09-2021							
2.	Do requirement analysis and develop Software Requirement Specification Sheet (SRS) for suggested system.	21-10-2021							
3.	To perform the function-oriented diagram: Data Flow Diagram (DFD) and Structured chart.	30-092021							
4.	To perform the user's view analysis for the suggested system: Use case diagram.	07-10-2021							
5.	To draw the structural view diagram for the system: Class diagram, object diagram.	15-10-2021							
6.	To draw the behavioural view diagram: State-chart diagram, Activity diagram.	28-10-2021							
7.	To perform the behavioral view diagram for the suggested system: Sequence diagram, Collaboration diagram.	05-11-2021							
8.	To perform the implementation view diagram: Component diagram for the system.	12-11-2021							
9.	To perform the environmental view diagram: Deployment diagram for the system.	18-11-2021							
10.	To perform various testing using the testing tool unit testing, integration testing for a sample code of the suggested system.	25-11-2021							
11.	To Perform Estimation of effort using FP Estimation for chosen system.	02-12-2021							



EXPERIMENT - 1

Software Engineering Lab

Aim

Write down the problem statement for a suggested system of relevance.

Syeda Reeha Quasar

14114802719

4C7

EXPERIMENT – 1

Aim:

Write down the problem statement for a suggested system of relevance.

Theory:

The problem statement is the initial starting point for a project. It is basically a one to three-page statement that everyone on the project agrees with that describes what will be done at a high level. The problem statement is intended for a broad audience and should be written in non-technical terms. It helps the non-technical and technical personnel communicate by providing a description of a problem. It doesn't describe the solution to the problem.

The input to requirement engineering is the problem statement prepared by customer.

It may give an overview of the existing system along with broad expectations from the new system.

The first phase of requirements engineering begins with requirements elicitation i.e., gathering of information about requirements. Here, requirements are identified with the help of customer and existing system processes. So, from here begins the preparation of problem statement.

So, basically a problem statement describes **what** needs to be done without describing **how**.

Performance Instruction:

1. Choose any one project from given list.
2. Collect all requirements
3. Identify functionalities
4. Write a one to three-page statement that everyone on the project agree with that describes what will be done at a high level.

Output:

PROBLEM STATEMENT FOR B2C SYSTEM

This would be a B2C System with 3 interfaces: an interface of an individual seller or small business an interface of a customer, and an efficient delivery system.

Current B2B systems don't support intermediary help for product delivery or transfer so we would provide aid for the same. This would also create revenue and employee opportunities

We aim at automating many operations performed at delivering the item to the customer. In many b2c services that already exist trade is carried out entirely at personal levels, with the risks of leaking of personal information and product not reaching the customer.

This can be solved if the seller uses one of the partner courier's services. The seller can fill up an online request with all the details for their parcel, get the parcel picked up at home get the cost of the delivery instantly when the pickup person arrives and a tracking ID is shared with both the seller and customer. A feedback system can also be implemented on both the sender and receiver end.

Features

1. **E-commerce website (Interface for the customer):** where the seller could decide what they want to buy.
2. **Interface for Seller:** This module allows the seller to put up their product's ad on the market.
3. **Personalized Delivery System:** This module allows the seller and customer to use the generated Tracking ID to track the current status of the product.
4. **Login Process:** This module allows valid customers to access the functionalities.
5. **Product Enquiry:** This module maintains the product details of a particular product.

6. **Update Products:** This module allows the customer to update the product status of their previous products they uploaded for sale.
7. **Product adds:** This module allows the customers to add products they want to sell.
8. **Update profile:** This module allows customers to change their password/ profile/ location/ details and verify contact details.
9. **Deals:** This module allows customers to view the details of their products and browse to view someone else's.

Q1 what is problem statement?

Ans A problem statement is a concise description of an issue to be addressed or a condition to be improved upon. It identifies the ~~top~~ gap b/w the current (problem) state & desired (goal) state of a process or product. The first condition of solving a problem is understanding the problem, which can be done by way of a problem statement.

Q2 what are the benefits of writing problem statement?

Ans A problem statement is basically a statement that illustrated a clear vision of the overall method that will be used to solve the problem at hand. Usually used when doing research, a problem statement discusses any foreseeable tangible or intangible problem that the researcher may face throughout the course of the project.

Q3 writing a problem statement, is really a beneficial for you in proceeding project?

Ans Yes, writing a problem statement should help in careful decision-making for project approval. Often, the problem statement will serve as the basis for the introductory section of a final proposal & directing the readers attention to the issues that your proposed project will address.

Q4 Explain 5W's can be used to spark the problem?

The five 'Ws' are questions whose answers are considered basic in problem solving & information gathering. They are:

- * who
- * what
- * when
- * where
- * why

The 5 'Ws' can be used to spark the discussion about the problem. A problem statement expresses the ~~what~~ that will be used to keep the effort focused, it should represent a solvable problem.

Q5 What are steps that need to follow while writing problem statement?

Ans Steps that we need to follow while writing problem statement are:

- ① Define the problem
- ② Reason for the problem's occurrence
- ③ when the problem began or was first noticed.
- ④ Place of the problem's first occurrence or sighting.
- Ans ⑤ The person or thing that the problem affects.
- ⑥ The sequence of event that resulted in the problem.

Viva Questions

1. What is problem statement?

Ans.

A problem statement is a concise description of an issue to be addressed or a condition to be improved upon. It identifies the gap between the current (problem) state and desired (goal) state of a process or product. The first condition of solving a problem is understanding the problem, which can be done by way of a problem statement.

2. What are the benefits of writing problem statement?

Ans.

A problem statement is basically a statement that illustrates a clear vision and the overall method that will be used to solve the problem at hand. Usually used when doing research, a problem statement discusses any foreseeable tangible or intangible problems that the researcher may face throughout the course of the project.

3. Writing a problem statement, is really a beneficial for you in proceeding project?

Ans.

Yes, writing a problem statement should help in careful decision-making for project approval. Often, the problem statement will serve as the basis for the introductory section of a final proposal, directing the reader's attention to the issues that your proposed project will address.

4. Explain 5W's can be used to spark the problem?

Ans.

The Five 'W's are questions whose answers are considered basic in problem solving and information gathering. They are:

- Who
- What
- When
- Where
- Why

The 5 'W's can be used to spark the discussion about the problem. A problem statement expresses the words that will be used to keep the effort focused and it should represent a solvable problem.

5. What are steps that need to follow while writing problem statement?

Ans.

Steps that we need to follow while writing problem statement are:

1. Define the problem.
2. Reason for the problem's occurrence.
3. When the problem began or was first noticed.
4. Place of the problem's first occurrence or sighting.
5. The person or thing that the problem affects.
6. The sequence of events that resulted in the problem.



EXPERIMENT - 2

Software Engineering Lab

Aim

Do requirement analysis and develop Software Requirement Specification Sheet (SRS) for suggested system.

Syeda Reeha Quasar

14114802719

4C7

EXPERIMENT – 2

Aim:

Do requirement analysis and develop Software Requirement Specification Sheet (SRS) for suggested system.

Theory:

Software Requirement Specification (SRS) is a document that describes the requirements of a computer system from the user's point of view. An SRS document specifies: The required behavior of a system in terms of: input data, required processing, output data, operational scenarios and interfaces. The attributes of a system including: performance, security, maintainability, reliability, availability, safety requirements and design constraints.

A well-designed, well-written SRS accomplishes four major goals:

- It provides feedback to the customer. An SRS is the customer's assurance that the development organization understands the issues or problems to be solved and the software behavior necessary to address those problems. Therefore, the SRS should be written in natural language (versus a formal language, explained later in this article), in an unambiguous manner that may also include charts, tables, data flow diagrams, decision tables, and so on.
- It decomposes the problem into component parts. The simple act of writing down software requirements in a well-designed format organizes information, places borders around the problem, solidifies ideas, and helps break down the problem into its component parts in an orderly fashion.
- It serves as an input to the design specification. As mentioned previously, the SRS serves as the parent document to subsequent documents, such as the software design specification and statement of work. Therefore, the SRS must contain sufficient detail in the functional system requirements so that a design solution can be devised.
- It serves as a product validation check. The SRS also serves as the parent document for testing and validation strategies that will be applied to the requirements for verification.

SRSs are typically developed during the first stages of "Requirements Development," which is the initial product development phase in which information is gathered about what requirements are needed--and not. This information-gathering stage can include onsite visits, questionnaires, surveys, interviews, and perhaps a return-on-investment (ROI) analysis or needs analysis of the customer or client's current business environment. The actual specification, then, is written after the requirements have been gathered and analyzed.

Performance Instruction:

Use IEEE SRS Document Template for drafting SRS for your system.

Content

1. Introduction
 - 1.1. Purpose
 - 1.2. Scope
 - 1.3. Definitions, Acronyms, and Abbreviations
 - 1.4. References
 - 1.5. Overview
2. Overall Description
 - 2.1. Product Perspective
 - 2.1.1. System Interfaces
 - 2.1.2. User Interfaces
 - 2.1.3. Hardware Interfaces
 - 2.1.4. Software Interfaces
 - 2.1.5. Communication Interfaces
 - 2.1.6. Memory Constraints
 - 2.1.7. Operations
 - 2.1.8. Site Adaptation Requirements
 - 2.2. Product Functions
 - 2.3. User Characteristics
 - 2.4. Constraints
 - 2.5. Assumptions and dependencies
 - 2.6. Apportioning of requirements
3. Specific Requirements

3.1. External Interface Requirements

3.1.1. User Interfaces

3.1.2. Hardware Interfaces

3.1.3. Software Interfaces

3.1.4. Communications Interfaces

3.2. Software Product Features

3.2.1. USER ACCOUNTS INFORMATION MAINTENANCE

3.2.1.1. Description

3.2.1.2. Validity Checks

3.2.1.3. Sequencing Information

3.2.1.4. Error Handling/ Response to Abnormal Situations

3.2.2. PRODUCT INFORMATION MAINTENANCE

3.2.2.1. Description

3.2.2.2. Validity Checks

3.2.2.3. Sequencing Information

3.2.2.4. Error Handling/ Response to Abnormal Situations

3.2.3. DELIVERY SYSTEM MAINTENANCE

3.2.3.1. Description

3.2.3.2. Validity Checks

3.2.3.3. Sequencing Information

3.2.3.4. Error Handling/ Response to Abnormal Situations

3.2.4. ORDER GENERATION AND CONFIRMATION

3.2.4.1. Description

3.2.4.2. Validity Checks

3.2.4.3. Sequencing Information

3.2.4.4. Error Handling/ Response to Abnormal Situations

3.3. Performance Requirements

3.4. Design Constraints

3.5. Software System Attributes

3.5.1. Security

3.5.2. Maintainability

3.5.3. Portability

3.6. Logical Database Requirements

3.7. Other Requirements

B2C System SRS

1. INTRODUCTION

This document aims at defining the overall software requirements for 'BUYER TO CONSUMER SYSTEM'. Efforts have been made to define the requirements exhaustively and accurately. The final product will be having only features/functionalities mentioned in this document and assumptions for any additional functionality/feature should not be made by any of the parties involved in developing/testing/implementing/ using this product. In case it is required to have some additional features a formal change request will need to be raised and subsequently a new release of this document and/or product will be produced.

1.1 PURPOSE

This specification document describes the capabilities that will be provided by the software application 'BUYER TO CONSUMER SYSTEM'. It also states the various required constraints by which the system will abide. The intended audiences for this document are the development team, testing team and end users of the product.

1.2 SCOPE

The software product 'BUYER TO CONSUMER SYSTEM ' is an application that will be used for buying and delivering products put up by a local seller or business, by a customer. The application will manage the information about products put up by a seller on our website, which can be bought by an interested customer and delivered by an available delivery man . The seller can fill up an online request with all the details for their parcel, get the parcel picked up and get the cost of the delivery instantly when the pickup person arrives and a tracking ID is shared with both the seller and buyer to check the status of delivery, which is updated by the delivery agent. A communication system is also implemented for communication between each party. This application will greatly simplify and speed up the delivery and management of the delivery and product process of this system.

1.3 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

The following abbreviation has been used throughout this document

B2C : Buyer to Consumer

1.4 REFERENCES

(i) GitHub Link - <https://github.com/syedareehaquasar/B2C-System>

(ii) Deployment Link

1.5 OVERVIEW

The rest of this SRS document describes the various system requirements, interfaces, features and functionalities in detail.

2.OVERALL DESCRIPTION

This would be a B2C System with 3 interfaces: an interface of an individual seller or small business, an interface of a customer, and an efficient delivery system.

Current B2B systems don't support intermediary help for product delivery or transfer so we would provide aid for the same. This would also create revenue and employee opportunities

We aim at automating many operations performed at delivering the item to the customer. In many b2c services that already exist trade is carried out entirely at personal levels, with the risks of leaking of personal information and product not reaching the customer.

This can be solved if the seller uses one of the partner courier services. The seller can fill up an online request with all the details for their parcel, get the parcel picked up at home get the cost of the delivery instantly when the pickup person arrives and a tracking ID is shared with both the seller and customer. A feedback system can also be implemented on both the sender and receiver end.

2.1 PRODUCT PERSPECTIVE

The application will be a browser-based, self-contained and independent software product.

2.1.1 SYSTEM INTERFACES

None

2.1.2 USER INTERFACES

The application will have a user-friendly and web based interface.

Following users will interact with the application:

- Buyer
- Seller
- Deliverer

Following screens will be provided:

- (i) A signup screen for registering as a buyer or seller, a Login screen for entering the username, password and logging in as buyer, seller or a delivery agent) Access to different screens will be based upon the role of the user.
- (ii) For buyers, there will be a screen which will allow them to view the products put up by the seller. This screen consists of features like viewing the product, adding them to cart, tracking an ordered product. There will be a payment system implemented for buyers so that they can pay for their order at the time of checkout.
- (iii) For Sellers, there will be a screen that will allow them to add products, manage them and remove them. There will be an interface that will allow the seller to assign a delivery agent of his choice.
- (iv) For the delivery agent , there will be a screen where the orders which are assigned to him, are available. The information about the order, address and contact information of seller and buyer is also shared with the delivery agent . The delivery agent can generate a tracking ID which is shared with both the buyer and seller and the delivery agent can update the status of the shipping from his side.

- (v) There will be a communication system between each party which will allow us to resolve issues during phases of order and delivery.
- (vi) There will be a payment portal that supports many forms of payment like UPI, credit card etc. Cash on delivery option is also provided.

2.1.3 HARDWARE INTERFACES

1. Access to the internet is required to access the API Interfaces.
2. A computer or mobile device
3. Screen resolution of at least 800x600-required for proper and complete viewing of the screen. Higher resolution would not be a problem.

2.1.4 SOFTWARE INTERFACES

1. Any window -based operating system.
2. Any Browser to access the website based application.
3. Visual Studio Code —for coding /developing the software.
4. Libraries and programming languages such as react, nodejs, bootstrap, JavaScript and few others.

Software mentioned in pts. 3. and 4. above will be required only for development of the application. The final application will be packaged as an independent setup program that will be delivered to the client and can be accessed via any web browser.

2.1.5 COMMUNICATION INTERFACES

The B2C system shall use HTTP protocol for communication over the internet and for the intranet communication will be through TCP/IP protocol suite.

2.1.6 MEMORY CONSTRAINTS

For running the program no memory is needed.

At least 64 MB RAM and 4 GB space on the hard disk will be required for development of the program.

2.1.7 OPERATIONS

This product release will not cover any automated housekeeping aspects of the database. The DB used is mongoDB. The DB at the seller side will be responsible for adding, modifying or deleting product data local to their shop. Database backup and recovery will be handled by the DBA(mongoDB server).

2.1.8 SITE ADAPTATION REQUIREMENTS

The terminals at the client site will have to support the hardware and software interface specified in the above section.

2.2 PRODUCT FUNCTIONS

The system will allow access only to registered users with specific roles. Depending upon the user's role he/she will be able to access only specific modules of the system.

A summary of the major functions that the software will perform:

- (i) **A SIGNUP facility** for registering buyer and seller
- (ii) **A LOGIN facility** for logging in as a buyer, customer or as delivery agent.
- (iii) **Users (with role Seller)** will be able to add/modify/delete products on their respective shops which can be viewed by Buyer. For a given product in a shop, the amount of stock available, price of the product is set by the seller.
- (iv) **Users (with role Seller)** will be able to assign the delivery agent to an ordered product by the buyer.

- (v) **Users (with role Buyer)** will be able to view different shops of sellers and the products that each shop is offering. Buyer can select the amount of any product they want to buy and add them to cart.
- (vi) **Users (with role Buyer)** will be able to pay for their order when checking out their cart via two modes available : Online Payment and Cash on Delivery. Another Interface is available for Online Payment, where payment can be done using various methods.
- (vii) **Users (with role Delivery Agent)** will be able to get assigned to the delivery of a product by the seller. They get access to the details required for delivery.
- (viii) **Users (with Delivery Agent)** will be able to update the status of delivery to either : Delivering or Delivered.
- (ix) **Users (with role Buyer, Seller and Delivery Agent)** will be able to communicate with each other.

2.3 USER CHARACTERISTICS

- (i) Educational Level : At least a graduate should be comfortable with English.
- (ii) Experience : Should be well informed about the working of an e-commerce website and interacting with it.
- (iii) Technical expertise : Should be comfortable with general purpose applications of computers.

2.4 CONSTRAINTS

1. Since the DBMS being used in MongoDB standard/ free version, it will be able to store records upto a certain limit very large data records can't be stored for a longer duration of time.
2. Realtime online payments are not possible yet as integrated razorpay test credentials have to change to a real time account to be able to have live payments.

3. Admin of B2C System will have to implement a security policy to safeguard the product to be delivered and related information from being accessed by unauthorized users (by means of gaining access to the backend database).

2.5 ASSUMPTIONS AND DEPENDENCIES

The deliverer will approach the seller to be able to sign up as a deliverer and the seller has to authenticate the person before giving them access to be a deliverer.

2.6 APPORTIONING OF REQUIREMENTS

OTP verification of deliverers everytime they sign up as no credentials are associated with a deliverer.

3.SPECIFIC REQUIREMENTS

This section contains the software requirements to a level of detail sufficient to enable designers to design the system, and testers to test that system.

3.1 EXTERNAL INTERFACE REQUIREMENTS

3.1.1 USER INTERFACES

The following screens will be provided:

SignUp Screen :

It will allow buyers and sellers to register to the website. Signup Screen consists of details such as Full Name, email , phone number , shop name,

shop address, shop description etc. There is no Signup screen for the delivery agent , as they are registered and assigned by the seller.

Login Screen:

It will allow already registered users to access different screens based upon the user's role. Various fields available on this screen will be:

Role: Buyer or Seller

Phone Number: which acts as a user ID for a given user.

Password: Alphanumeric string which gives user access to their account

Role: Delivery Agent

Name: Registered name of the deliverer.

Phone No.: Registered for no, of the deliverer.

Buyer Screen:

This screen will be accessible only to users with role Buyer. It will allow the user to browse through different shops of different sellers, and view their products. In each shop there is contact information available about the seller with which the buyer can contact the seller.

Product Screen:

They are accessed on viewing a product from a shop. It includes description of product, its price, options like choosing the amount of the particular product buyer wants to order and an add to cart option

Cart Screen:

This screen will be accessible only to users with role Buyer. All the products that the buyer adds to cart while browsing the items in shops are

present here. An item present here can be removed too. This screen leads to two options for payment : Pay online , or Cash on Delivery.

Payment Screen :

The user is given 2 options either online payment or Cash on delivery. On selecting the Online Payment Option. It has a razorpay payment interface, which has several payment modes available, like UPI, debit card, credit card etc. In the UPI mode, the buyer can scan a QR generated via their mobile device and pay for the order.

Post-Payment Screen:

This screen comes on after successfully paying. It shows OrderID, paymentID. An order request is also sent to the seller after the buyer makes an order. There are other options on the screen that can be used to track the orders made by the buyer and check their status.

Seller Screen :

This screen will be accessible only to users with role Seller. It has options that allow sellers to post new products in their shop, to remove an existing product, update the stock of an existing product, check orders made by the buyers and assign a delivery agent to make these orders.

Add Product Screen :

This screen will be accessible only to users with role Seller. It is used by sellers to add a new product to their shop. To add a product, the seller has to enter details about it like, Product Name, Product Description, Price, Amount , Product Picture etc. After successfully adding a product , it can be viewed by the buyer in their interface.

Seller Order Screen :

This screen will be accessible only to users with role Seller. In this screen , sellers can see all the orders that are made for their products by the buyers, and accordingly they register and assign a delivery agent of their preference by adding the delivery agent's phone number and name. In this screen , the status of delivery can also be checked.

Buyer Order Screen :

This screen will be accessible only to users with role Buyer. In this screen , buyers can see all the orders that they have made. Here, they can check the status of delivery of their order, and contact the seller and delivery agent too.

Delivery Agent Screen :

This screen will be accessible only to users with role Delivery Agent. A Delivery Agent is registered by a Seller and an order is assigned to them for delivery by that Seller only. In their Interface they can see the assigned orders that are to be delivered. They can see the details like the details of Buyer, Seller and their contact information for delivery purposes. They can also update the status of the delivery which is reflected in both buyer and seller interface.

3.1.2. HARDWARE INTERFACES

As stated in section 2.1.3.

3.1.3 SOFTWARE INTERFACES

As stated in section 2.1.4.

3.1.4 COMMUNICATIONS INTERFACES

As stated in section 2.1.5.

3.2 SOFTWARE PRODUCT FEATURES

3.2.1 USER ACCOUNTS INFORMATION MAINTENANCE

Description : The system will maintain information about various users who will be able to access the system. The following information would be maintained that is common between all roles :

The following information is maintained for each role :

For Buyer : Name, Email, Phone No., Address, Password

For Seller : Name, Email, Phone No., Shop Name, Shop Address, Password

For Delivery Agent : Name, Phone No.

Validity Checks:

Only the database will be authorized to access the User Accounts Information Maintenance module. Any changes made by the user such as change of password will be reflected in the database

- Phone Number cannot be blank.
- Phone Number should be unique for every user.
- Email cannot be blank.
- Passwords cannot be blank.
- Address cannot be blank.
- Shop Name cannot be blank

Sequencing Information:

A User Account for a particular user has to be created in order for the system to be accessible to that user. In the website, the user has to be registered first using the sign up option , then they can log in anytime after that.

Error Handling/ Response to Abnormal Situations:

If any of the above validations/ sequencing flow does not hold true, appropriate error messages. will be prompted for users to do the needful.

3.2.2 PRODUCT INFORMATION MAINTENANCE**Description:**

The system database will maintain information about the products that various sellers put up in their shop. The following information would be maintained for each product :

Product Name, Product Description, Product Picture, Product' Stock Available, Seller of the Product

The system database will allow creation/modification/deletion of new/existing products.

Validity Checks:

Only system databases will be authorized to access and maintain the Product information Maintenance module. Only Seller's can make changes to their respective products that will be reflected in the database.

- Product Name cannot be blank.
- Product Picture cannot be blank.
- Product Description cannot be blank.
- If Product's Stock becomes 0, it will show Out of Stock.
- Product Price cannot be zero.

Sequencing Information:

Product Info will have to be entered into the system database by the Seller using the interface provided before any they can be reflected on the webpage.

Error Handling/ Response to Abnormal Situations:

If any of the above validations/ sequencing flow does not hold true, appropriate error messages will be prompted for users to do the needful.

3.2.3 DELIVERY SYSTEM MAINTENANCE**Description :**

The system database will maintain information like buyer's address and phone number, seller's address and phone number and the status of the product, whether it is delivered or yet to be delivered, it is updated by the delivery agent, stored in the database and shown to buyer and seller when they check the order status. The following information would be maintained for each delivery :

- Order Details.
- Order of the Status.
- The system will allow updates of status of the delivery by the delivery agent for a particular order.

Validity Checks:

- Only system databases will be authorized to access and maintain the Product information Maintenance module. Only users with the role Delivery Agent assigned the particular order can update the delivery status.
- Order Status can be changed to Delivered or shipped.

Sequencing Information:

Order will have to be made by the Buyer and Delivery agent will have to be assigned by the seller to that order before any delivery agent can access this module.

Error Handling/ Response to Abnormal Situations:

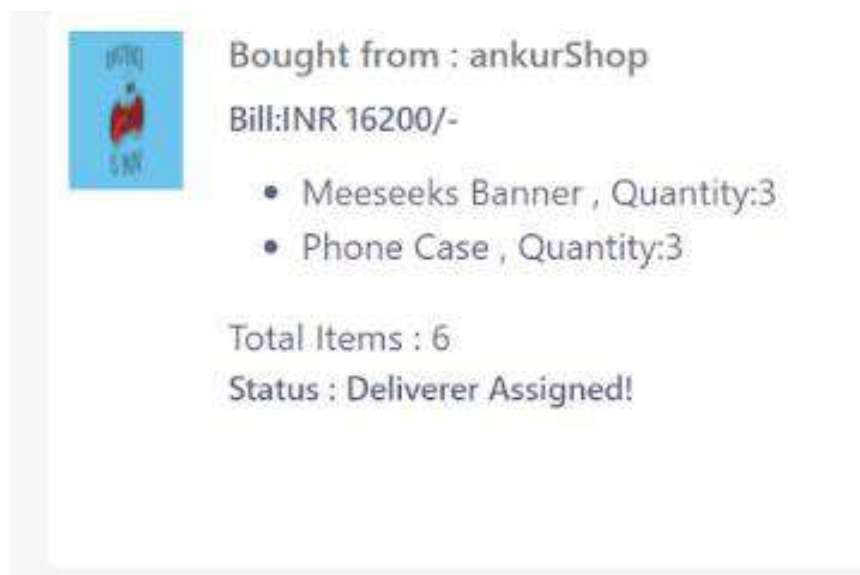
If any of the above validations/ sequencing flow does not hold true, appropriate error messages. and status will be prompted for users to do the needful.

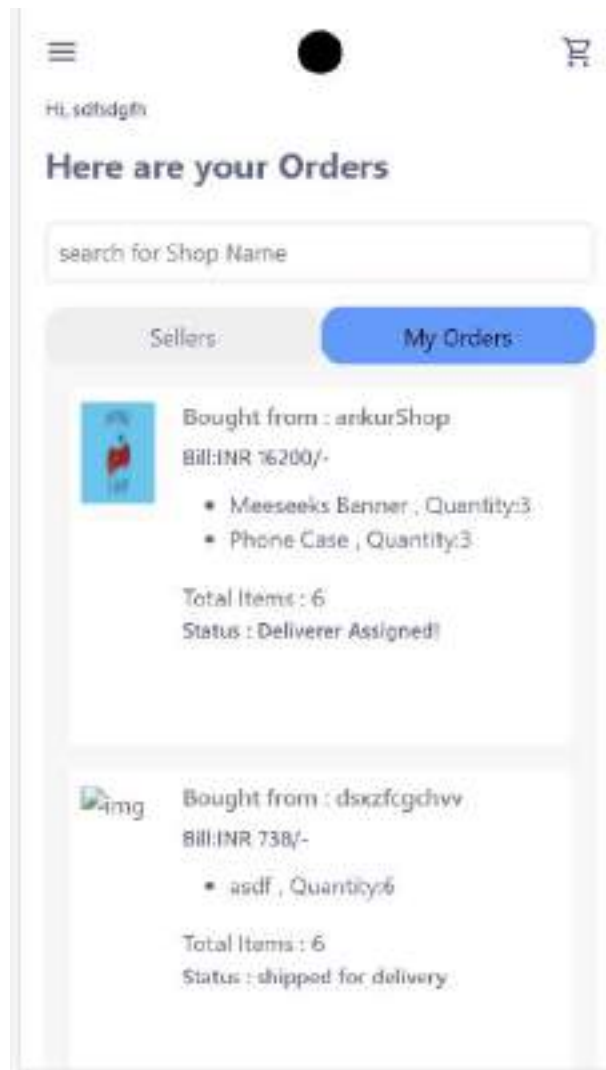
3.2.4 ORDER GENERATION AND CONFIRMATION

Description :

The system will generate order receipt with a unique id along with the payment status and details to verify and confirm a particular order. The system database will be maintained and verified using razorpay APIs and Razorpay Dashboard.

ORDERS WILL HAVE THE FOLLOWING FORMAT :





Validity Checks:

Users who have ordered the product, seller of the product and assigned deliverer will be able to access this module.

Sequencing Information:

Order for a particular Buyer can be generated by the system only after the payment is done by the buyer and verified by the system using Razorpay API. OrderID and a receipt will be generated after that.

Error Handling/ Response to Abnormal Situations:

If any of the above validations/ sequencing flow does not hold true, appropriate error messages will be prompted for users to do the needful.

3.3 PERFORMANCE REQUIREMENTS

None

3.4 DESIGN CONSTRAINTS

For designing Figma was used and the system is device responsive but still has to work on improving the user Interface.

3.5 SOFTWARE SYSTEM ATTRIBUTES

3.5.1 SECURITY

The application will be password protected. Users will have to enter a valid phone number, password and role in order to access the application.

3.5.2 MAINTAINABILITY

The application will be designed in a maintainable manner. It will be easy to incorporate new requirements in the individual modules when the need arises.

3.5.3 PORTABILITY

The application will be easily portable on any windows-based system that has a Web Browser.

3.6 LOGICAL DATABASE REQUIREMENTS

The following information will be placed in the database :

- User Information according to their Role (Buyer, Seller, Deliverer)
- Order Details
- Product details
- Cart Details
- Reviews



3.7 OTHER REQUIREMENTS

Clients should have a working phone no. along with an email.

Conclusion

The SRS was written successfully by following the template described above.

VIVA QUESTIONS

02.12.2

Q1
Ans What are the objective of requirement analysis?
The purpose of the Requirements Analysis phase is to transform the need of high-level requirements specified in earlier phases into unambiguous (measurable & testable), traceable, complete, & stakeholder approved required.

Q2
Ans Define different type of requirement?
According to IEEE standard 729, a requirement is defined as follow:

* A software requirement can be of 3 types

- * Functional requirements
- * Non-functional requirements
- * Domain requirements

Functional Requirements: These are the requirement that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Non-functional requirements: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other.

Domain requirement - Domain requirement are the requirements which are characteristic of a particular category or domain of project. The basic functions that a system of a specific domain must necessarily exhibit come under this category.

Q3 Outline structure of SRS Document?

Ans ① Introduction

- (i) Purpose of this document
- (ii) Scope of this document
- (iii) Overview

② General description

③ Functional Requirement

④ Interface Requirements

⑤ Performance Requirements

⑥ Design Constraints

⑦ Non-functional Attributes

⑧ Preliminary Schedule & Budget

⑨ Appendices.

Q4 what are benefits of writing SRS document?

Ans * An SRS establishes the basis for agreement between the customer & the supplier on what the software product will perform.

* An SRS provides a reference for validation of the final product/software.

* A high-quality SRS is a prerequisite to high-quality product/software.

* A high-quality SRS reduces the development cost.

Q5 Define functional & non functional requirement?

FUNCTIONAL VS NONFUNCTIONAL REQUIREMENT

	FUNCTIONAL REQUIREMENT	NONFUNCTIONAL REQUIREMENT
OBJECTIVE	Describe what the product does	Describe how the product works
END RESULT	Define product feature	Define product properties
FOCUS	Focus on user requirement captured in use case	Focus on user expectation captured as a quality attribute.
DOCUMENTATION	They are mandatory	They are not mandatory, but desirable
ORIGINITY	Usually defined by user	Usually defined by developers or other tech experts
TESTING	Component, API, UI testing etc. Tested before nonfunctional testing	Performance, usability, security testing etc. Tested after functional testing.
Types	External interface, authentication, authorization levels, business rules etc.	Usability, reliability, scalability, performance etc.

Viva Questions

1. What are the objectives of requirement analysis?

Ans.

The purpose of the Requirements Analysis Phase is to transform the needs and high-level requirements specified in earlier phases into unambiguous (measurable and testable), traceable, complete, consistent, and stakeholder-approved requirements.

2. Define different types of requirements?

Ans.

According to IEEE standard 729, a requirement is defined as follows:

- A condition or capability needed by a user to solve a problem or achieve an objective
- A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed documents
- A documented representation of a condition or capability as in 1 and 2.

A software requirement can be of 3 types:

- Functional requirements
- Non-functional requirements
- Domain requirements

Functional Requirements: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

For example, in a hospital management system, a doctor should be able to retrieve the information of his patients. Each high-level functional requirement may involve several interactions or dialogues between the system and the outside world. In order to accurately describe the functional requirements, all scenarios must be enumerated.

There are many ways of expressing functional requirements e.g., natural language, a structured or formatted language with no rigorous syntax and formal specification language with proper syntax.

Non-functional requirements: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.

They basically deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability
- Flexibility

NFR's are classified into following types:

- Interface constraints
- Performance constraints: response time, security, storage space, etc.
- Operating constraints
- Life cycle constraints: maintainability, portability, etc.
- Economic constraints

The process of specifying non-functional requirements requires the knowledge of the functionality of the system, as well as the knowledge of the context within which the system will operate.

Domain requirements: Domain requirements are the requirements which are characteristic of a particular category or domain of projects. The basic functions that a system of a specific domain must necessarily exhibit come under this category. For instance, in an academic software that maintains records of a school or college, the functionality of being able to access the list of faculty and list of students of each grade is a domain requirement. These requirements are therefore identified from that domain model and are not user specific.

3. Outline structure of SRS Document?

Ans.

1. Introduction

- (i) Purpose of this document
- (ii) Scope of this document
- (iii) Overview

2. General description

3. Functional Requirements

4. Interface Requirements

5. Performance Requirements

6. Design Constraints

7. Non-Functional Attributes

8. Preliminary Schedule and Budget

9. Appendices

Outline used for this SRS:

1. Introduction

- 1.1. Purpose
- 1.2. Scope
- 1.3. Definitions, Acronyms, and Abbreviations
- 1.4. References
- 1.5. Overview

2. Overall Description

- 2.1. Product Perspective
 - 2.1.1. System Interfaces

- 2.1.2. User Interfaces
- 2.1.3. Hardware Interfaces
- 2.1.4. Software Interfaces
- 2.1.5. Communication Interfaces
- 2.1.6. Memory Constraints
- 2.1.7. Operations
- 2.1.8. Site Adaptation Requirements
- 2.2. Product Functions
- 2.3. User Characteristics
- 2.4. Constraints
- 2.5. Assumptions and dependencies
- 2.6. Apportioning of requirements
- 3. Specific Requirements
 - 3.1. External Interface Requirements
 - 3.1.1. User Interfaces
 - 3.1.2. Hardware Interfaces
 - 3.1.3. Software Interfaces
 - 3.1.4. Communications Interfaces
 - 3.2. Software Product Features
 - 3.2.1. USER ACCOUNTS INFORMATION MAINTENANCE
 - 3.2.1.1. Description
 - 3.2.1.2. Validity Checks
 - 3.2.1.3. Sequencing Information
 - 3.2.1.4. Error Handling/ Response to Abnormal Situations
 - 3.2.2. PRODUCT INFORMATION MAINTENANCE
 - 3.2.2.1. Description
 - 3.2.2.2. Validity Checks
 - 3.2.2.3. Sequencing Information
 - 3.2.2.4. Error Handling/ Response to Abnormal Situations
 - 3.2.3. DELIVERY SYSTEM MAINTENANCE
 - 3.2.3.1. Description
 - 3.2.3.2. Validity Checks
 - 3.2.3.3. Sequencing Information
 - 3.2.3.4. Error Handling/ Response to Abnormal Situations
 - 3.2.4. ORDER GENERATION AND CONFIRMATION
 - 3.2.4.1. Description
 - 3.2.4.2. Validity Checks
 - 3.2.4.3. Sequencing Information

- 3.2.4.4. Error Handling/ Response to Abnormal Situations
- 3.3. Performance Requirements
- 3.4. Design Constraints
- 3.5. Software System Attributes
 - 3.5.1. Security
 - 3.5.2. Maintainability
 - 3.5.3. Portability
- 3.6. Logical Database Requirements
- 3.7. Other Requirements

4. What are benefits of writing SRS document?

Ans.

- An SRS establishes the basis for agreement between the customer and the supplier on what the software product will perform.
- An SRS provides a reference for validation of the final product/software.
- A high-quality SRS is a prerequisite to high-quality product/software.
- A high-quality SRS reduces the development cost.

5. Define Functional and non-functional requirements?

Ans.

Functional Requirements: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Non-functional requirements: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are

also called non-behavioral requirements.
They basically deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability
- Flexibility

FUNCTIONAL vs NONFUNCTIONAL REQUIREMENTS

	Functional requirements	Nonfunctional requirements
Objective	Describe what the product does	Describe how the product works
End result	Define product features	Define product properties
Focus	Focus on user requirements	Focus on user expectations
Documentation	Captured in use case	Captured as a quality attribute
Essentiality	They are mandatory	They are not mandatory, but desirable
Origin type	Usually defined by user	Usually defined by developers or other tech experts
Testing	Component, API, UI testing, etc. Tested before nonfunctional testing	Performance, usability, security testing, etc. Tested after functional testing
Types	External interface, authentication, authorization levels, business rules, etc.	Usability, reliability, scalability, performance, etc.



EXPERIMENT - 3

Software Engineering Lab

Aim

To perform the function oriented diagram: Data Flow Diagram (DFD), ER Diagram and Structured chart.

Syeda Reeha Quasar

14114802719

4C7

EXPERIMENT – 3

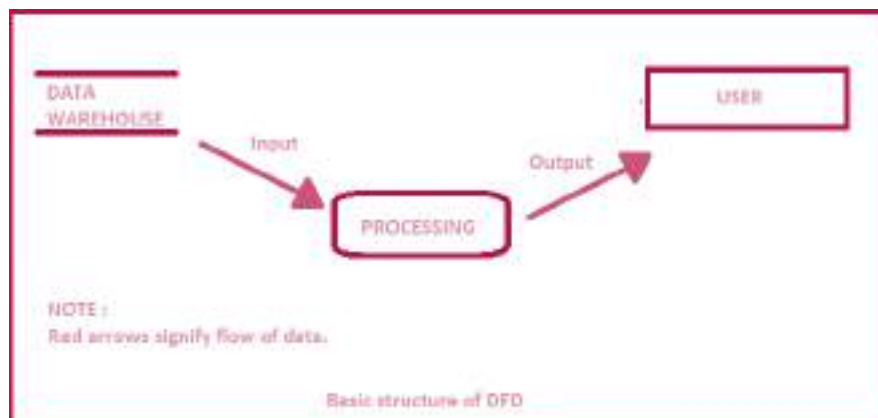
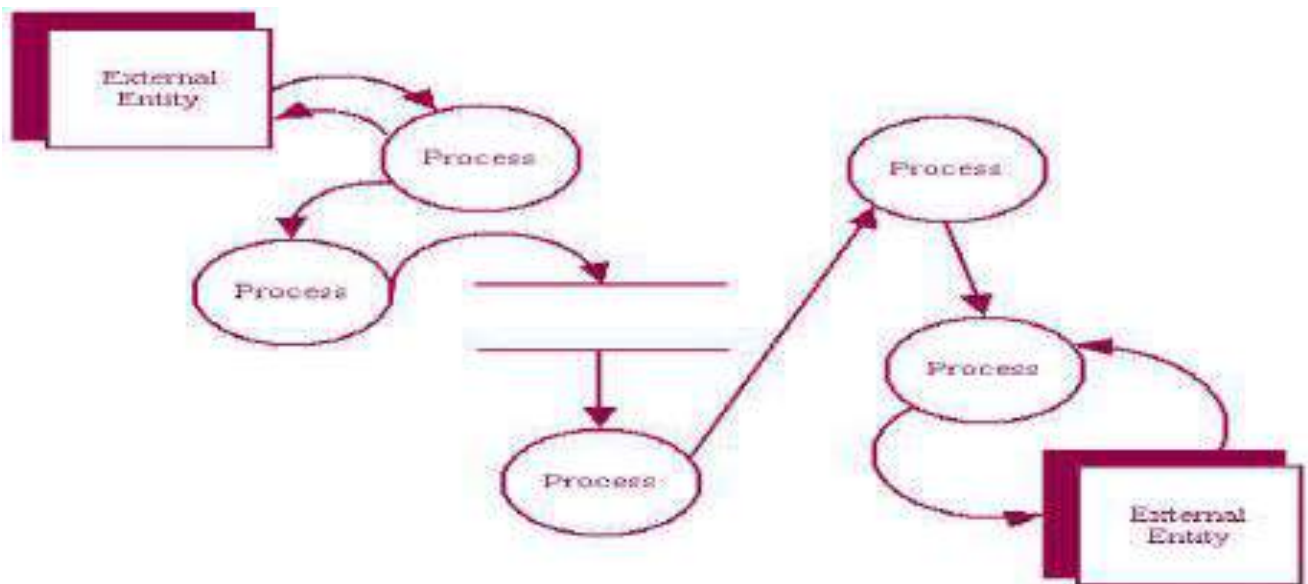
Aim:

To perform the function-oriented diagram: Data Flow Diagram (DFD), ER Diagram and Structured chart.



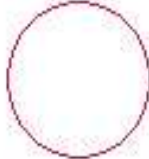





Theory:

Data flow diagrams are versatile diagramming tools. With only four symbols, data flow diagrams can represent both physical and logical information systems. The four symbols used in DFD representation are data flows, data stores, processes, and sources / sinks (or external entities).

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs.

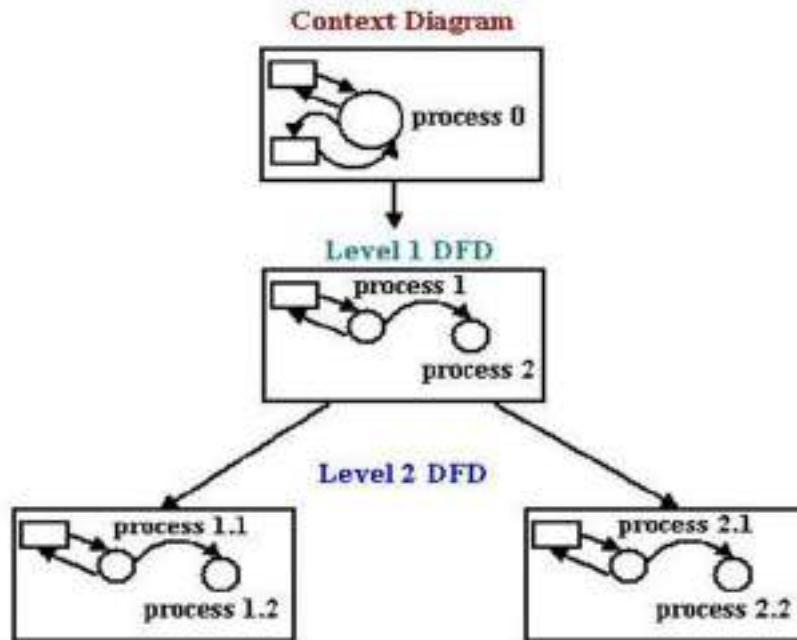


Symbols of DFD:

Name	Symbol	Description	Example
Entity		Used to represent people and organizations outside the system. They either input information to the system, accept output information from the system or both	
Process		These are actions that are carried out with the data that flows around the system. A process accepts input data and produces data that it passes on to another part of the DFD	
Data Flow		These represent the flow of data to or from a process	
Data Store		This is a place where data is stored either temporarily or permanently	

Data Flow Diagram Layers

Draw data flow diagrams in several nested layers. A single process node on a high-level diagram can be expanded to show a more detailed data flow diagram. Draw the context diagram first, followed by various layers of data flow diagrams.



ER Diagram:

An entity relationship model, *also called an entity-relationship (ER) diagram*, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems. An entity is a piece of data-an object or concept about which data is stored.

Relationships Between Entities

A relationship is how the data is shared between entities. There are three types of relationships between entities:

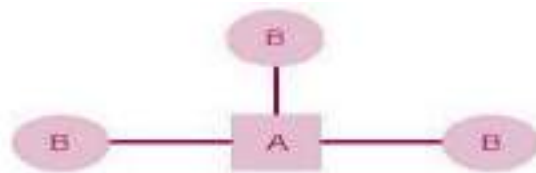
1. One-to-One

One instance of an entity (A) is associated with one other instance of another entity (B). For example, in a database of employees, each employee name (A) is associated with only one social security number (B).



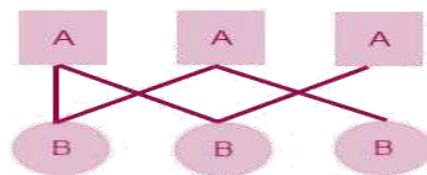
2. One-to-Many

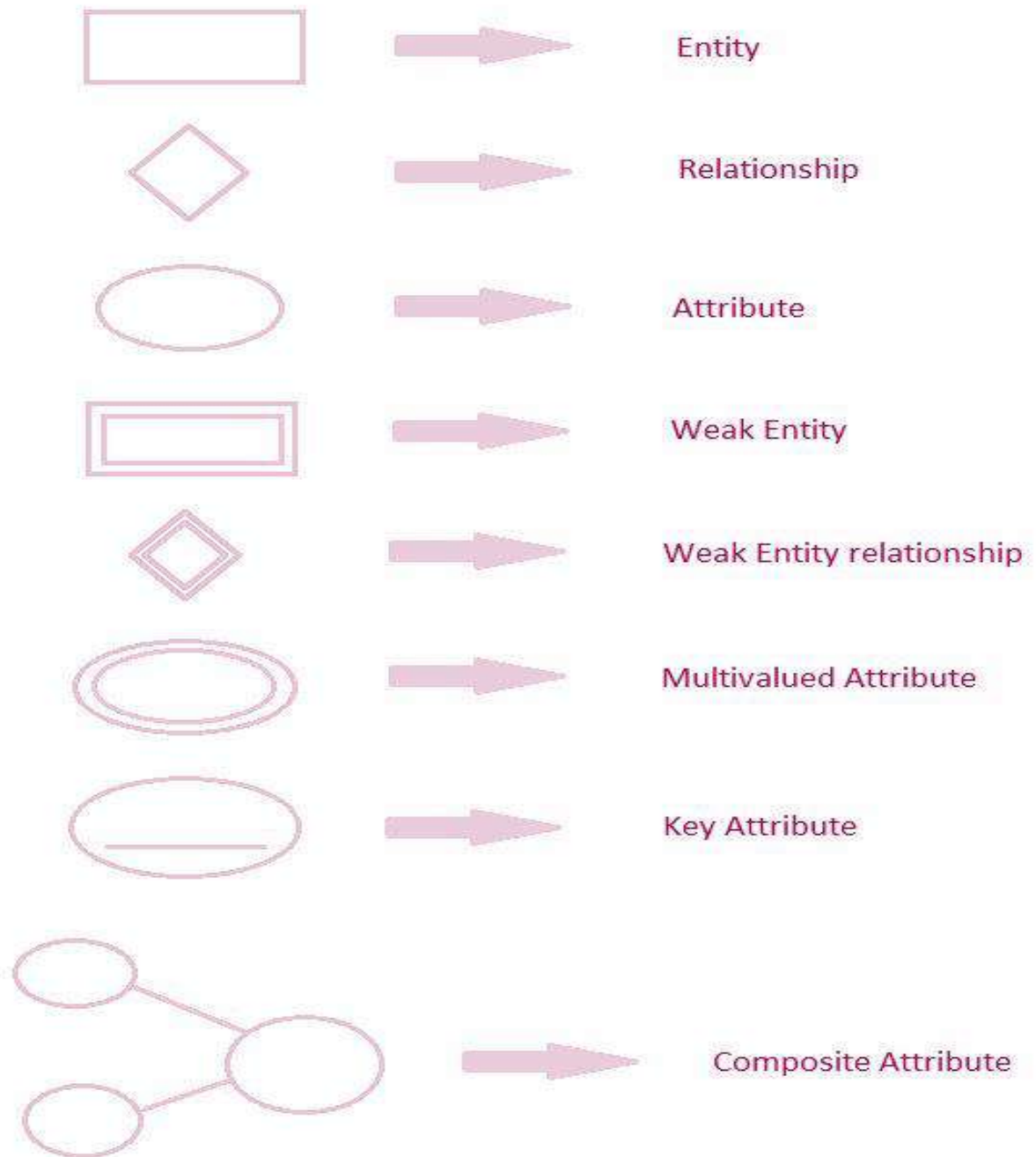
One instance of an entity (A) is associated with zero, one or many instances of another entity (B), but for one instance of entity B there is only one instance of entity A. For example, for a company with all employees working in one building, the building name (A) is associated with many different employees (B), but those employees all share the same singular association with entity A.



3. Many-to-Many

One instance of an entity (A) is associated with one, zero or many instances of another entity (B), and one instance of entity B is associated with one, zero or many instances of entity A. For example, for a company in which all of its employees work on multiple projects, each instance of an employee (A) is associated with many instances of a project (B), and at the same time, each instance of a project (B) has multiple employees (A) associated with it.



Symbols used to draw ER diagram:

A **Structure Chart** (SC) in software engineering is a chart which shows the breakdown of a system to its lowest manageable levels. They are used in structured programming to arrange program modules into a tree. Each module is represented by a box, which contains the module's name. The lines represent the connection and or ownership between activities and sub activities as they are used in organization charts. The tree structure visualizes the relationships between modules.

Performance Instructions:

To draw DFD

1. Identify various processes, data store, input, output etc. of the system and analyse it.
2. Use processes at various levels to draw the DFDs.

To draw ER Diagram

1. Identify all entities and their attributes.
2. Identify weak entities, attributes and their identifying relationship.
3. Design ER diagram according to norms.
4. Imply cardinalities on diagram.

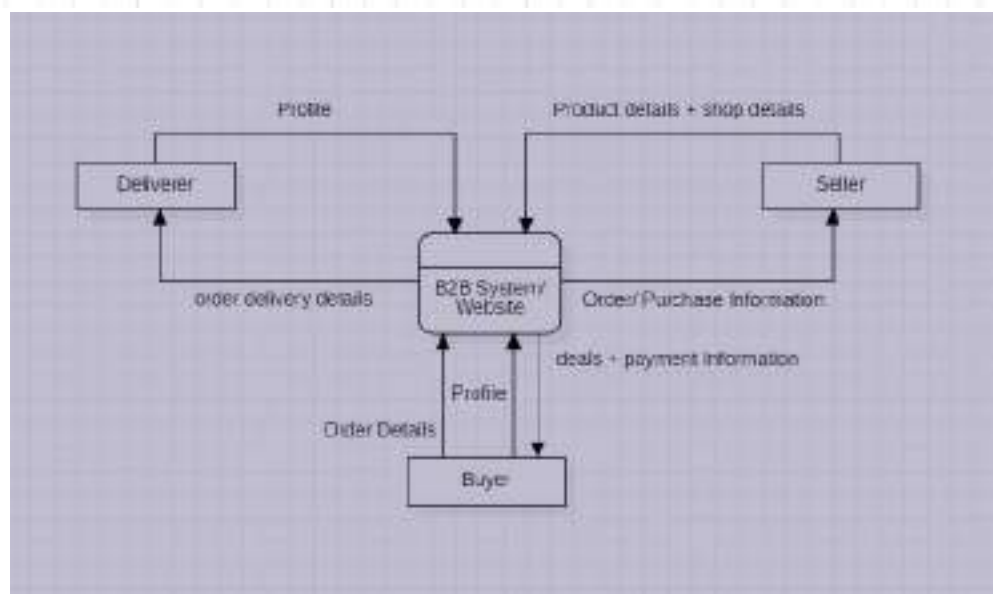
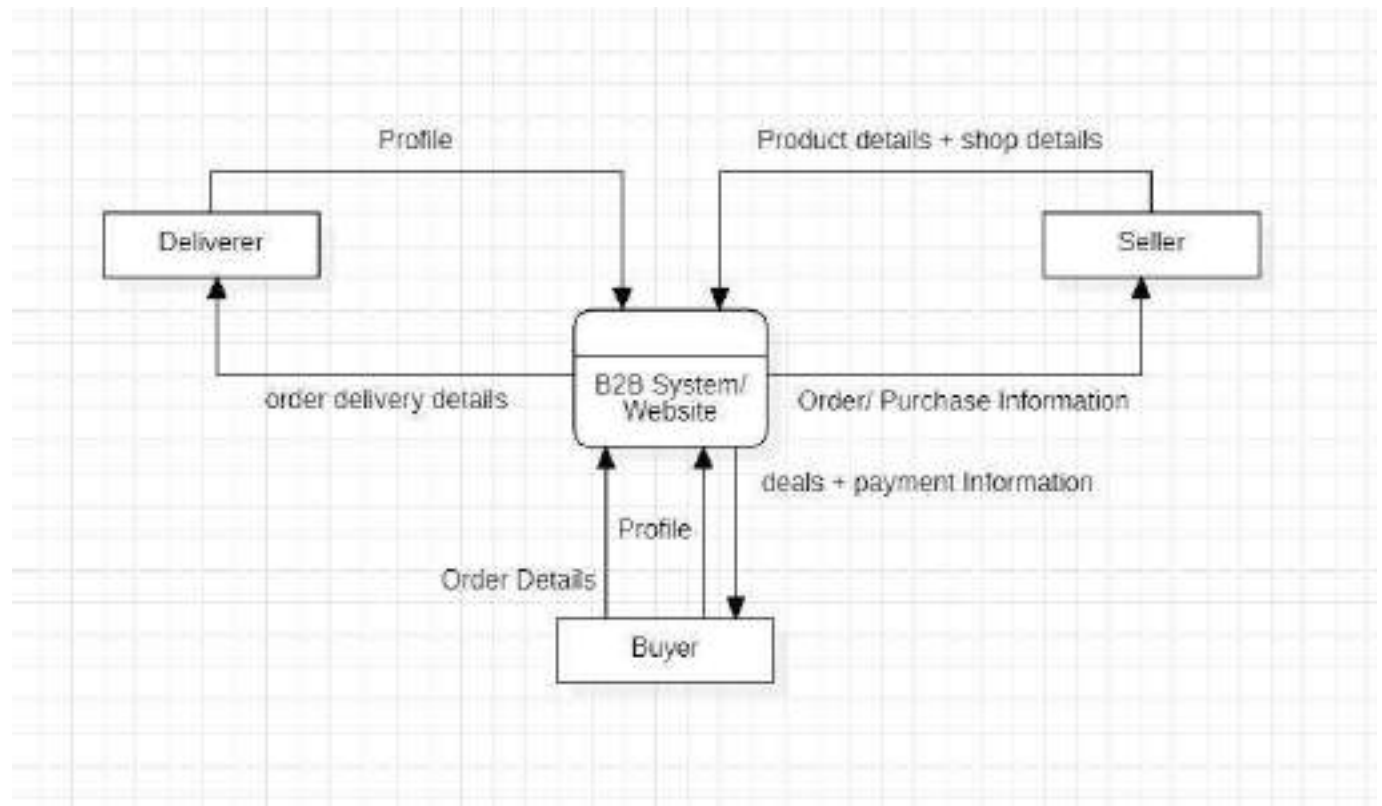
To draw structured Chart Diagram

1. Identify various modules, input, output etc. of the system.
2. Draw structured chart diagram describing it in form of levels.

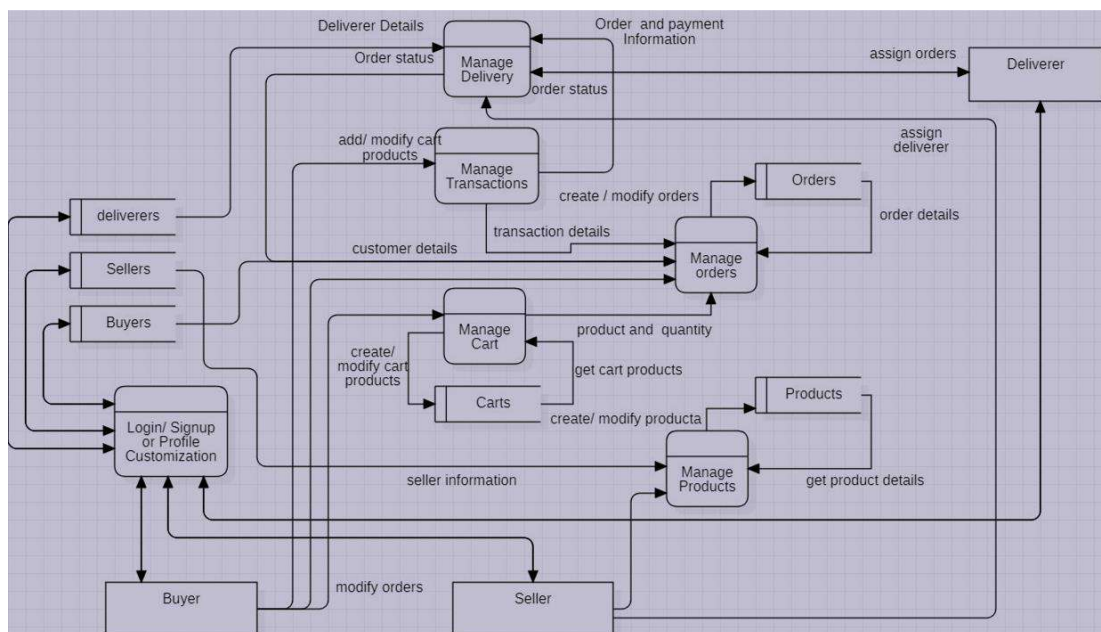
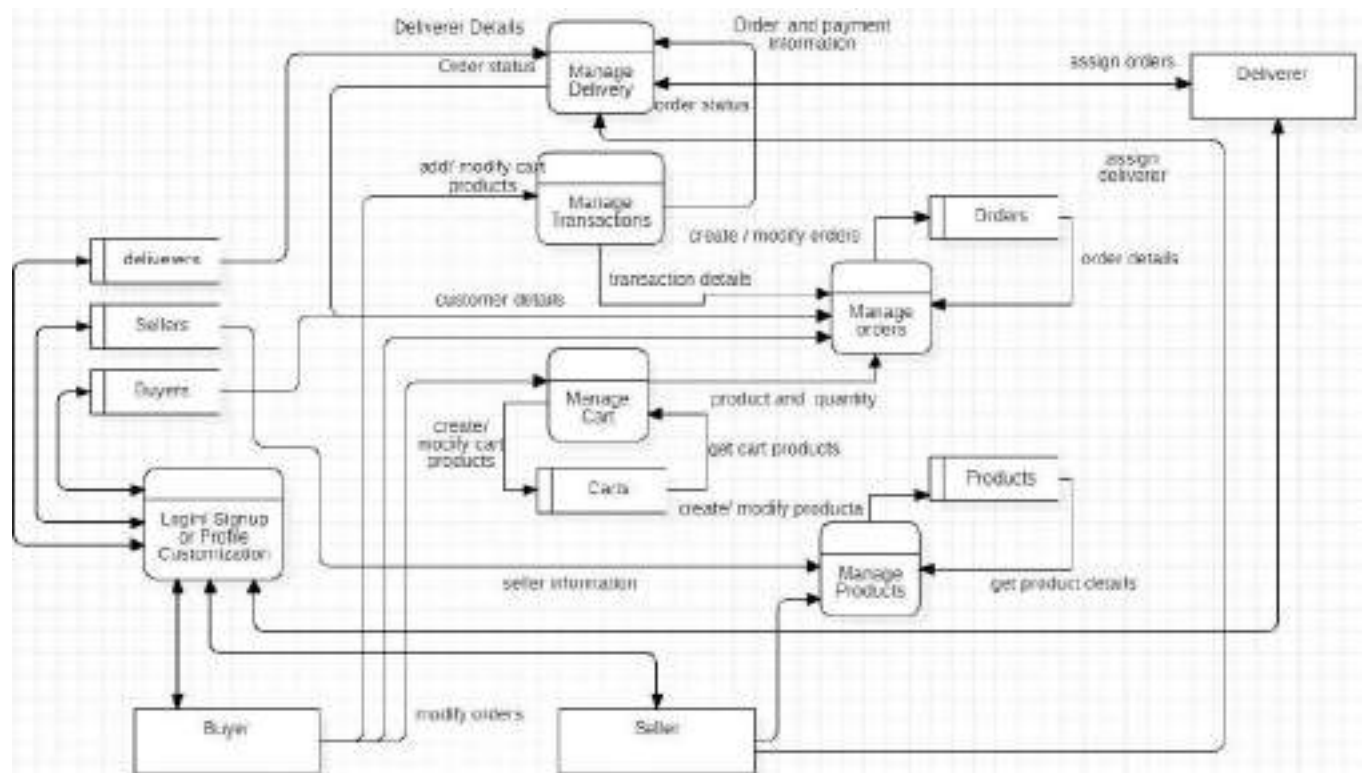
OUTPUT:

DFD (Data Flow Diagram):

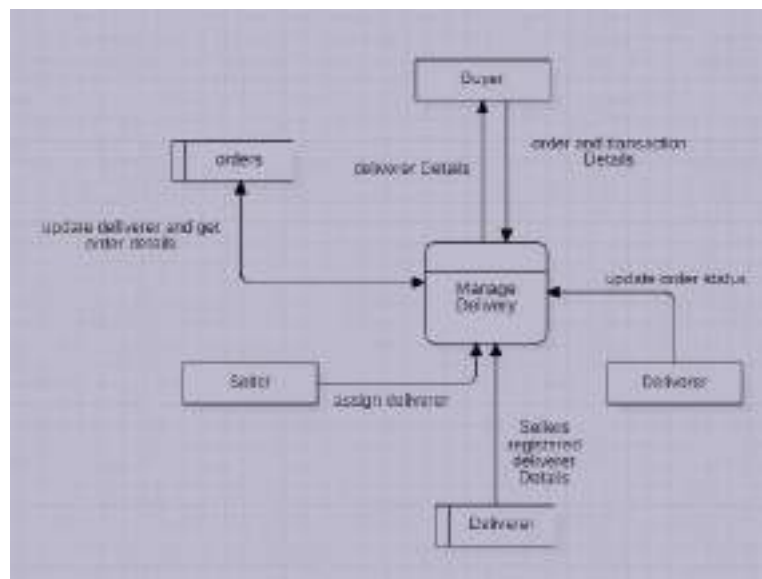
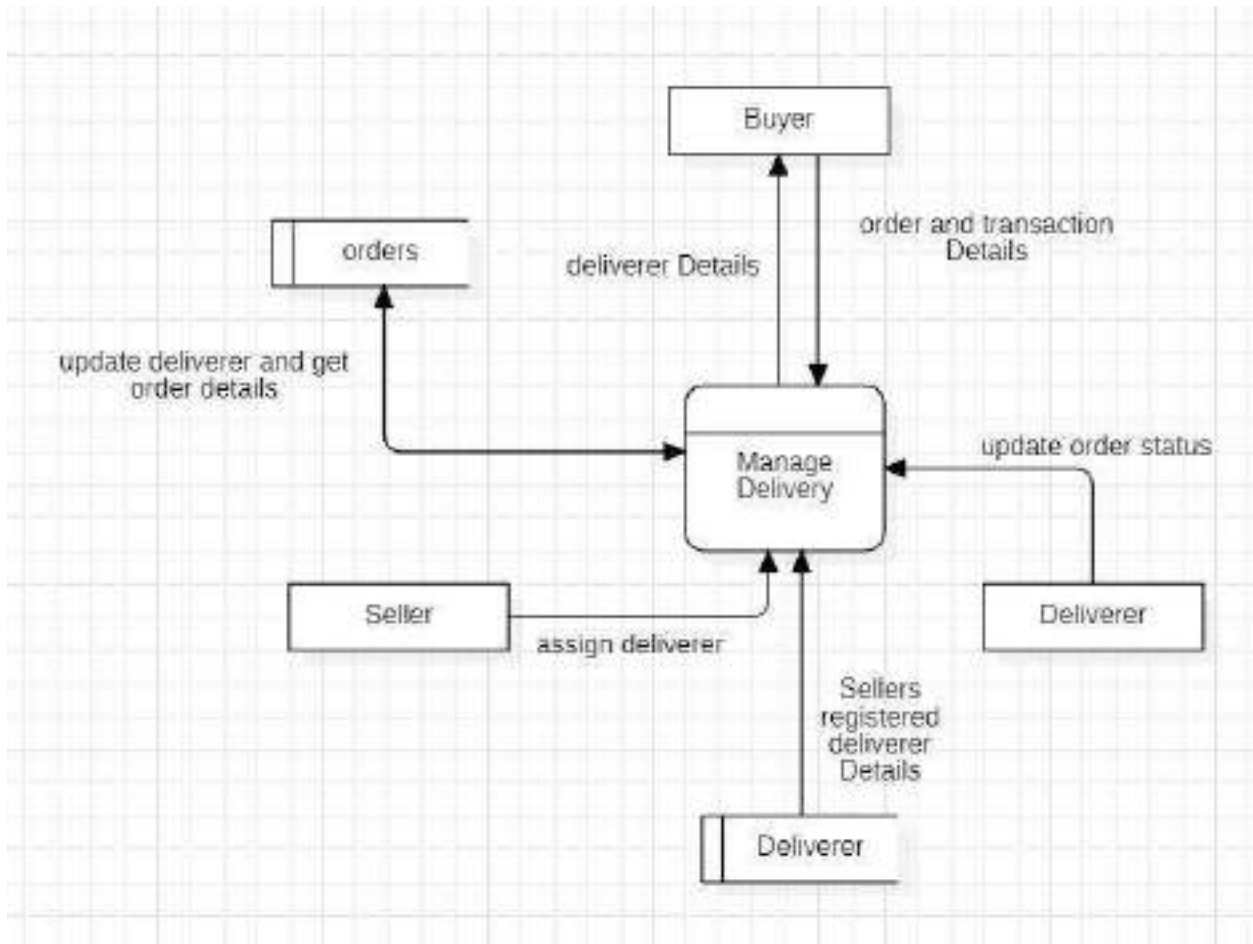
Level 0:



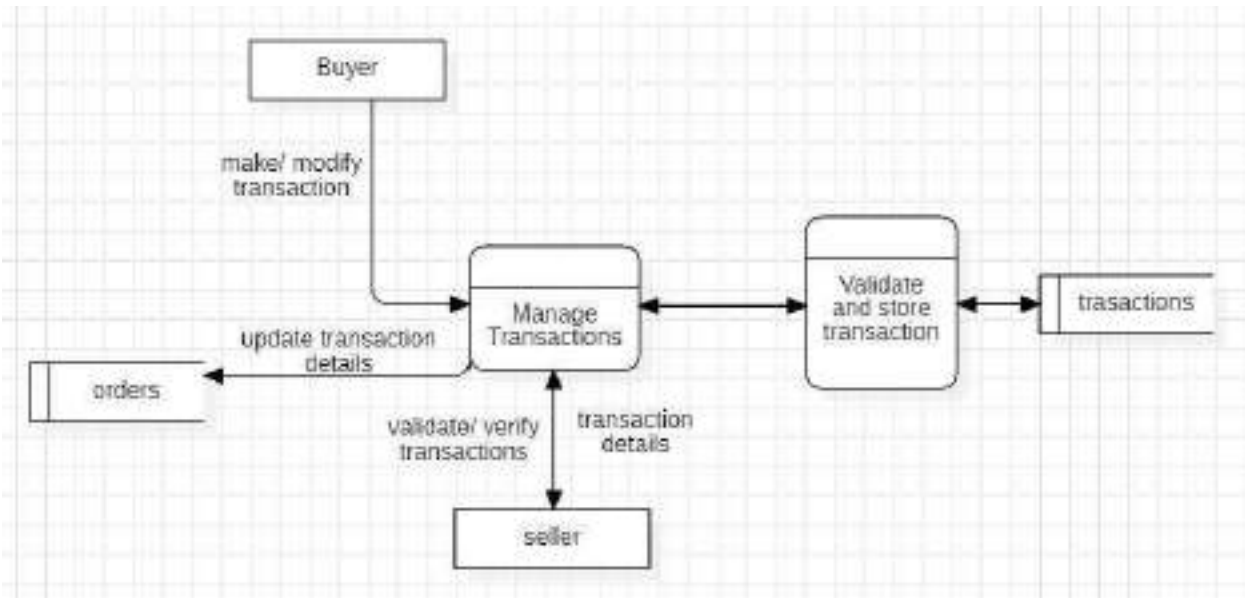
Level 1:



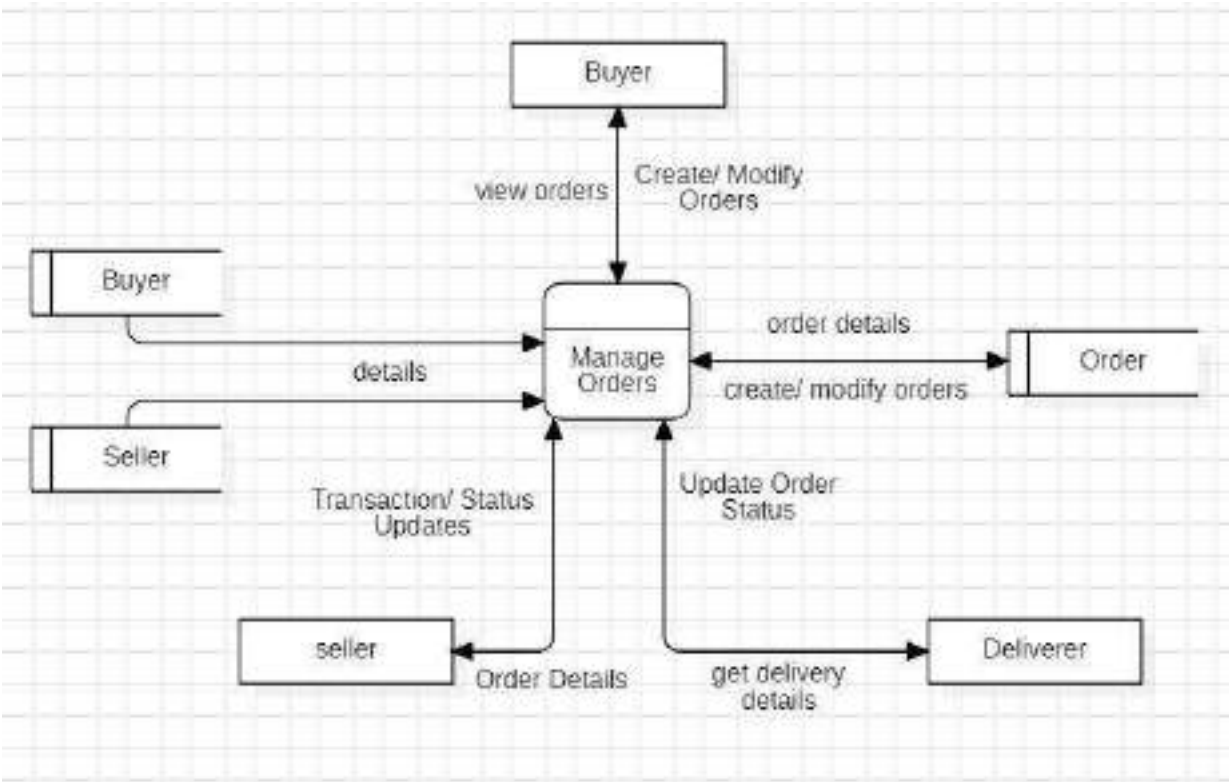
Level 2.1:



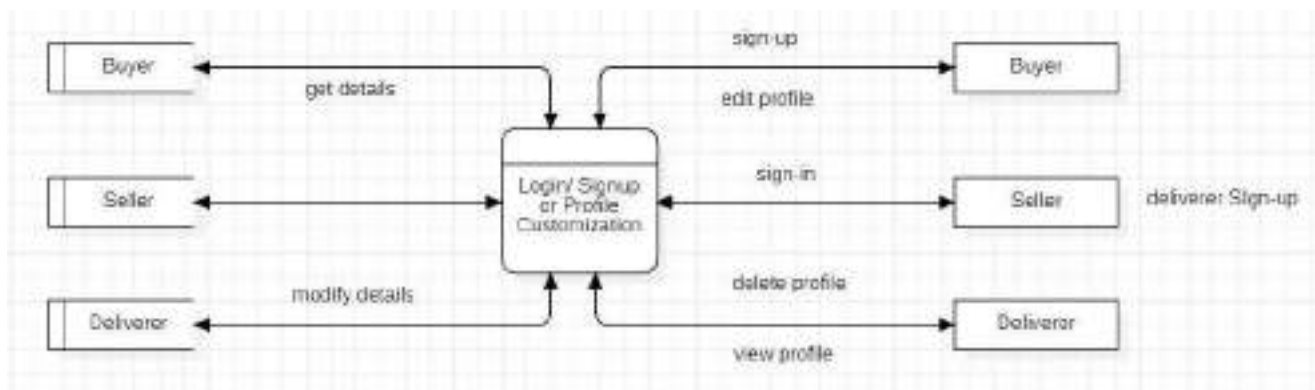
Level 2.2:



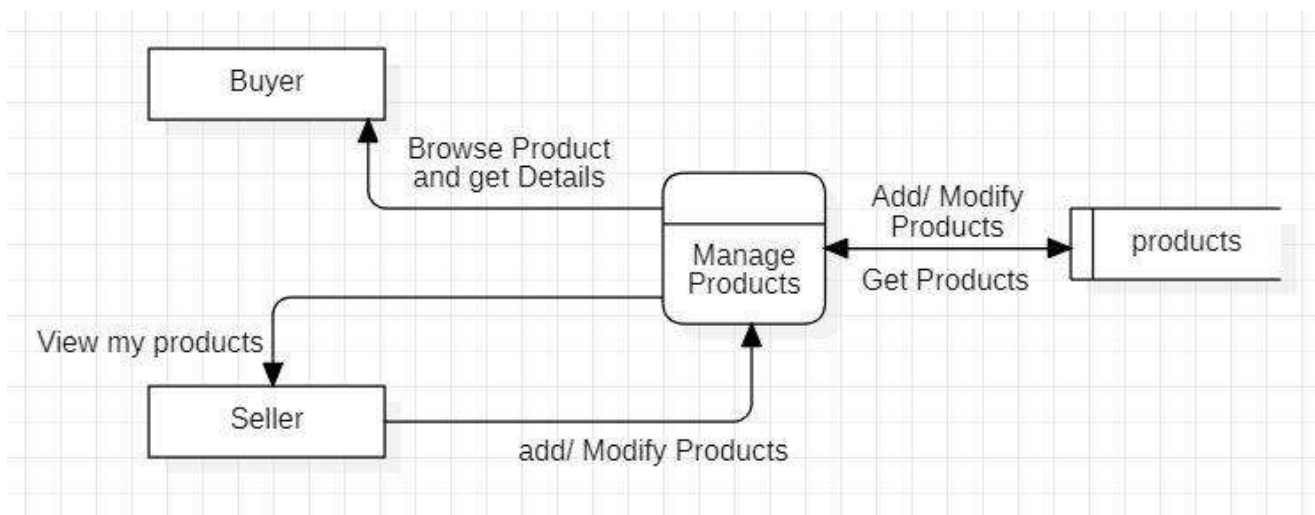
Level 2.3:



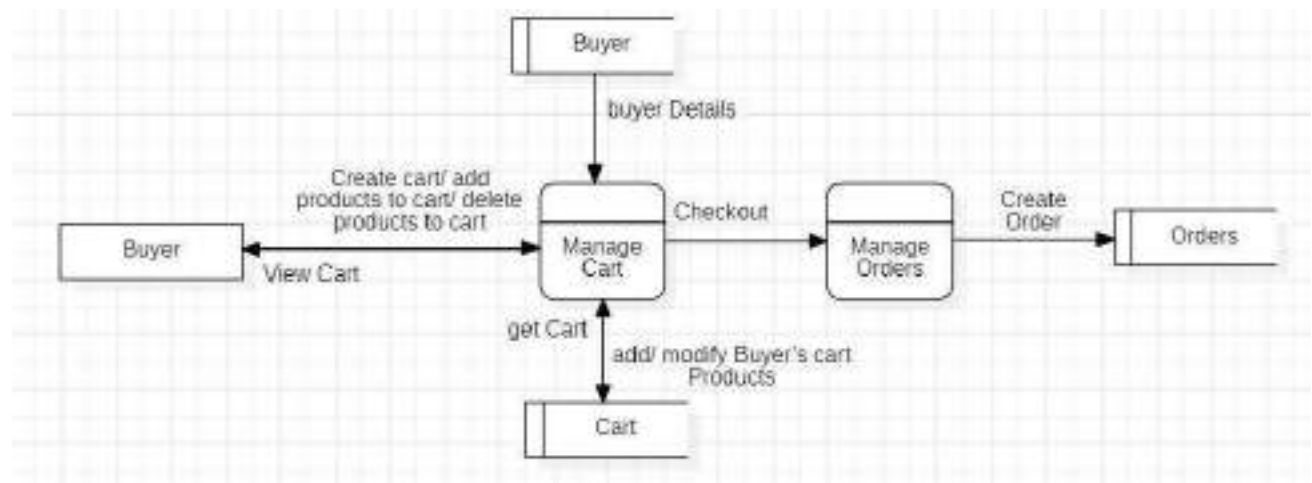
Level 2.4:



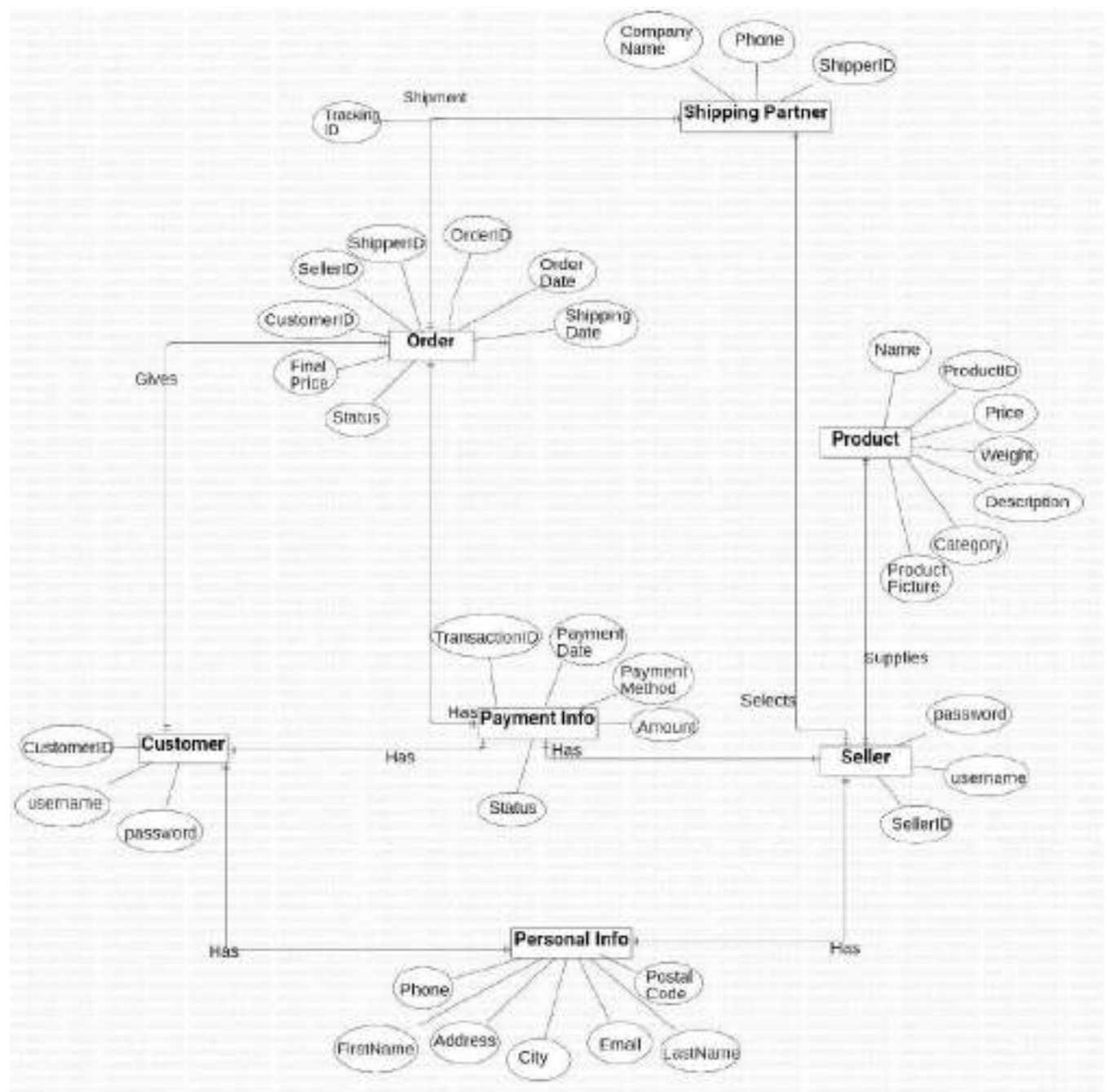
Level 2.5:



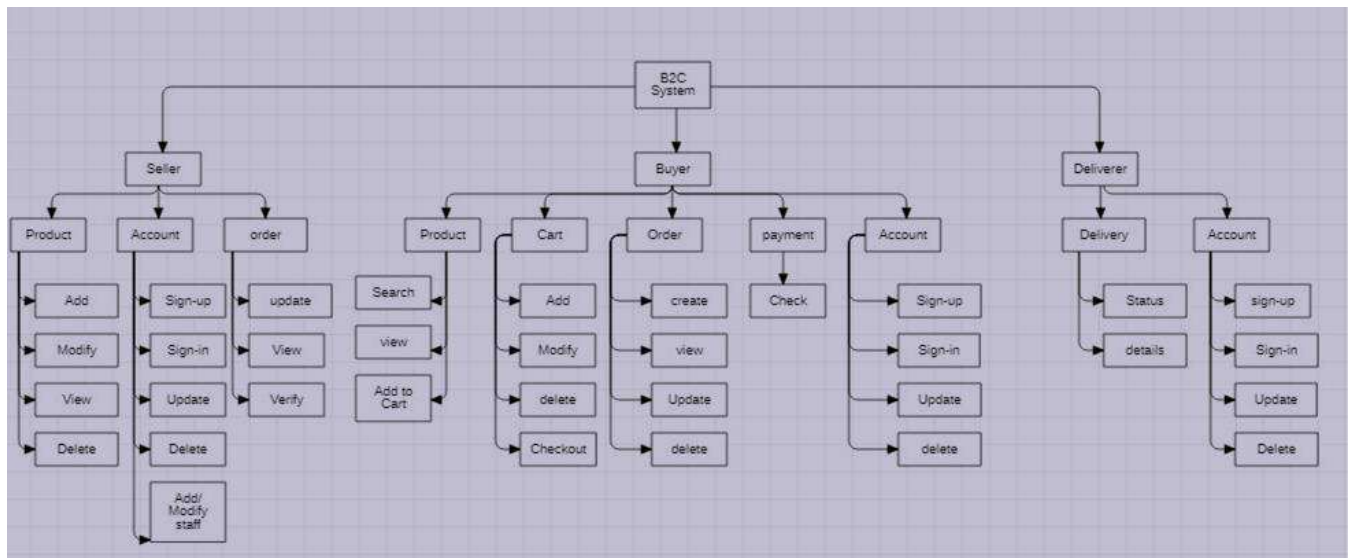
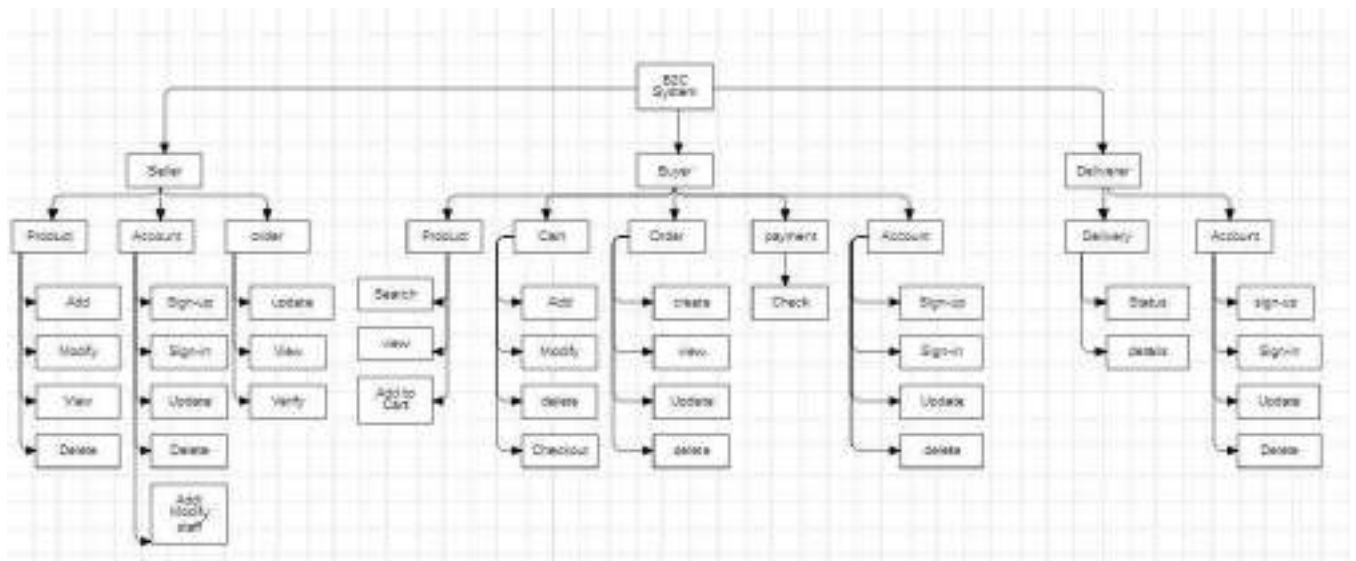
Level 2.6:



ER (Entity Relation) Diagram:



Structured Diagram:



Q1 Define DFD? what are different level of DFD?

Ans

Data flow diagrams are versatile diagramming tools. With only four symbols, data flow diagrams can represent both physical & logical information system. The four symbols used in DFD representation are data flow, data stores, process, & sources/sinks (as external entities).

0-level DFD:

It is also known as a context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represent the entire system as a single bubble.

1-level DFD

In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level we highlight the main functions of the system.

2-level DFD

2-level DFD goes one step deeper into parts of 1-level DFD. It can be used to plan as related the specific/ necessary detail about the system's functioning.

Q2 Describe symbols used for constructing DFDs?

Ans

The four symbols used in DFD representation are data flows, data stores, processes & sources/sinks. (as external entities).

Q3

Distinguish b/w a data flow diagram & a flow chart with example?

Ans

The main difference b/w DFD & flowchart is that DFD is graphical diagram that represent the data flow of a system while flowchart is a graphical diagram that represents the sequence of step to solve a problem.

Q4 Explain structure chart diagram?

Ans A structure chart (SC) in software engineering is a chart which shows the breakdown of a system to its lowest manageable levels.

Q5 Describe symbols used for constructing structured chart diagram?

Ans Module - It represents process or subroutine or task. A control module branches to more than one sub-module. Library module are re-used & invokable from any module.

Condition - It is represented by small diamond to the base of module. It depicts that control module can select any of sub-routine based on some condition.

Jump - An arrow is shown pointing inside the module to depict that the control will jump in the middle of the sub-module.

Loop - A loop curved arrow represent loop in the module. All sub-module covered by loop repeat execution of module.

Data flow - A directed arrow with empty circle at the end represent data flow.

Control flow - A directed arrow with filled circle at the end represent control flow.

Q6 Explain ER diagram?

Ans An entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities & their relationship to each other, typically used in computing in regard to the organization of data within databases or information system.

Viva Questions

1. Define DFD? What are different levels of DFD?

Ans.

Data flow diagrams are versatile diagramming tools. With only four symbols, data flow diagrams can represent both physical and logical information systems. The four symbols used in DFD representation are data flows, data stores, processes, and sources / sinks (or external entities).

0-level DFD:

It is also known as a context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble

1-level DFD:

In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system

2-level DFD:

2-level DFD goes one step deeper into parts of 1-level DFD. It can be used to plan or record the specific/necessary detail about the system's functioning.

2. Describe symbols used for constructing DFDs?

Ans.

The four symbols used in DFD representation are data flows, data stores, processes, and sources / sinks (or external entities).

3. Distinguish between a data flow diagram and a flow chart with example?

Ans.

The main difference between DFD and Flowchart is that DFD is a graphical diagram that represents the data flow of a system while a flowchart is a graphical diagram that represents the sequence of steps to solve a problem.

4. Explain structured chart diagram?

Ans.

A Structure Chart (SC) in software engineering is a chart which shows the breakdown of a system to its lowest manageable levels.

5. Describe symbols used for constructing structured chart diagram?

Ans.

Module - It represents process or subroutine or task. A control module branches to more than one sub-module. Library Modules are re-usable and invokable from any module.

Condition - It is represented by small diamond at the base of module. It depicts that control module can select any of sub-routine based on some condition.

Jump - An arrow is shown pointing inside the module to depict that the control will jump in the middle of the sub-module.

Loop - A curved arrow represents loop in the module. All sub-modules covered by loop repeat execution of module.

Data flow - A directed arrow with empty circle at the end represents data flow.

Control flow - A directed arrow with filled circle at the end represents control flow.

6. Explain ER diagram?

Ans.

An entity relationship model, *also called an entity-relationship (ER) diagram*, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems.

7. What is entity? Explain strong and weak entity?

Ans.

Entity : An entity is an object or component of data. An entity is represented as rectangle in an ER diagram. The entity set which does not have sufficient attributes to form a primary key is called as Weak entity set. An entity set that has a primary key is called as Strong entity set.

8. Explain structured chart diagram?

Ans.

Attributes are the properties which define the entity type. They are characteristics of the entity that help users to better understand the database. Attributes are included to include details of the various entities that are highlighted in a conceptual ER diagram.



EXPERIMENT - 4

Software Engineering Lab

Aim

To perform the user's view analysis for the suggested system: Use case diagram.

Syeda Reeha Quasar

14114802719

4C7

EXPERIMENT – 4

Aim:

To perform the user's view analysis for the suggested system: Use case diagram.

Theory:

The use-case diagram can provide the user's view for designing of the software product. And it can also be tested by matching up the requirements with the use-cases.

When to Use: Use Cases Diagrams

Use cases are used in almost every project. They are helpful in exposing requirements and planning the project. During the initial stage of a project most use cases should be defined, but as the project continues more might become visible.

Actors: Are NOT part of the system – they represent anyone or anything that must interact with the system.

- Only input information to the system.
- Only receive information from the system.
- Both input to and receive information from the system.
- Represented in UML as a stickman.

Use Case

- A sequence of transactions performed by a system that yields a measurable result of values for a particular actor.
- A use case typically represents a major piece of functionality that is complete from beginning to end. A use case must deliver something of value to an actor

Use Case Relationships Between actor and use case.

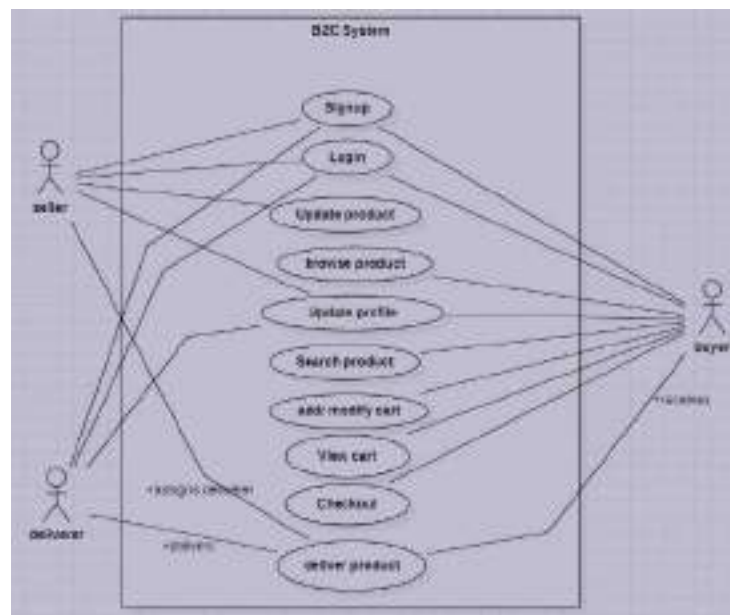
- Association / Communication.
- Arrow can be in either or both directions; arrow indicates who initiates communication.

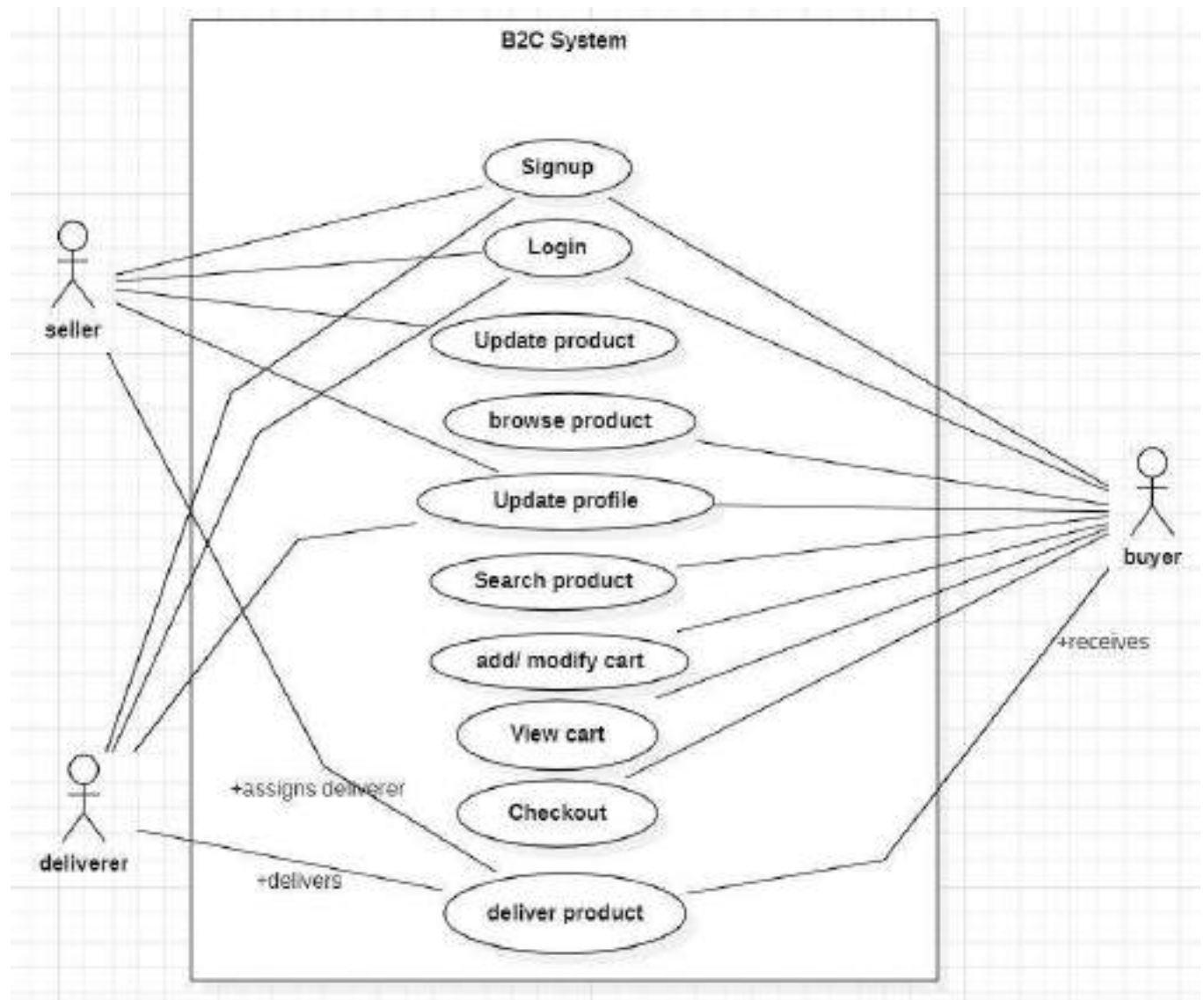
Between use cases (generalization):

Uses: Where multiple use cases share pieces of same functionality.

Performance Instruction:

1. Identify various processes, use-cases, actors etc. of the system and analyse it.
2. Use processes at various levels and draw use case diagram.

Output:



Q1 Explain use case approach of requirement elicitation?

Ans This technique combines text & pictures to provide a better understanding of the requirement. The use cases describe the 'What' of a system & not 'How'. Hence, they only give a functional view of system. The components of the use case design include three major things - Actors, Use cases, use case diagram.

Q2 Explain term: Use case, use-case scenarios, use-case diagrams?

Ans A use case is a written description of how users will perform tasks on your website. It outlines, from a user's point of view, a system's behaviour as it responds to a request.

A user case scenario is a single path through the use case. Unlike a use case which is a step-by-step enumeration of the task carried out during a process (with the associated actors), a scenario is much more free-form.

A use case diagram can summarize the details of your system users (also known as actors) & their interactions with the system.

Q3 What are Actors of use cases?

Ans An Actor is use case modelling specifies a role played by a user or any other system that interacts with the subject. An actor model type of role played by an entity that interacts with the subject. (eg., by exchanging signals & data), by which is external to the subject.

A use case is a written description of how users will perform task on your website. It outlines, from a user's point of view, the system's behaviour as it responds to a request.

Q4 Explain guidelines that should be kept in mind while creating use cases?

Ans Consider the following:

- * Single statement per line
- * Always have a subject - "User" or "system"
- * Be concise - remember, use cases are not required. you should be demonstrating the interaction between the system & user, but not detailed specifications.
- * Use an active voice.

Q5 Name the person who invented use case approach?

Ans Ivar Jacobson first formulated textual & visual modelling techniques for specifying use cases.

1/11

Viva Questions

1. Explain use case approach of requirement elicitation?

Ans.

This technique combines text and pictures to provide a better understanding of the requirements. The use cases describe the 'what', of a system and not 'how'. Hence, they only give a functional view of the system. The components of the use case design include three major things – Actor, Use cases, use case diagram.

2. Explain term: use-case, use-case scenarios, use-case diagrams?

Ans.

A use case is a written description of how users will perform tasks on your website. It outlines, from a user's point of view, a system's behavior as it responds to a request.

A use case scenario is a single path through the use case. Unlike a use case which is a step-by-step enumeration of the tasks carried out during a process (with the associated actors), a scenario is much more free-form.

A use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system.

3. What are actors and use cases?

Ans.

An **actor** in use case modelling specifies a role played by a user or any other system that interacts with the subject. An Actor models a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data), but which is external to the subject.

A **use case** is a written description of how users will perform tasks on your website. It outlines, from a user's point of view, a system's behaviour as it responds to a request.

4. Explain guidelines that should be kept in mind while creating use cases?

Ans.

Consider the following:

- Single statement per line
- Always have a subject – “User” or “System”
- Be concise – remember, use cases are not end requirements – you should be demonstrating the interaction between the system and user, but not detailed specifications
- Use an active voice

5. Name the person who invented use case approach?

Ans.

Ivar Jacobson first formulated textual and visual modelling techniques for specifying use cases.



EXPERIMENT - 5

Software Engineering Lab

Aim

To draw the structural view diagram for the system: Class diagram, object diagram.

Syeda Reeha Quasar

14114802719

4C7

EXPERIMENT – 5

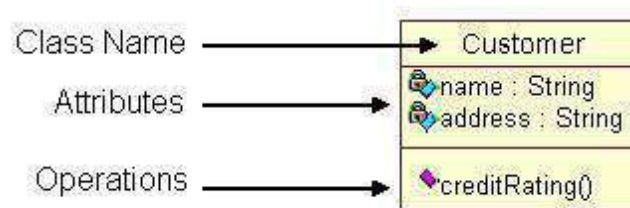
Aim:

To draw the structural view diagram for the system: Class diagram, object diagram.

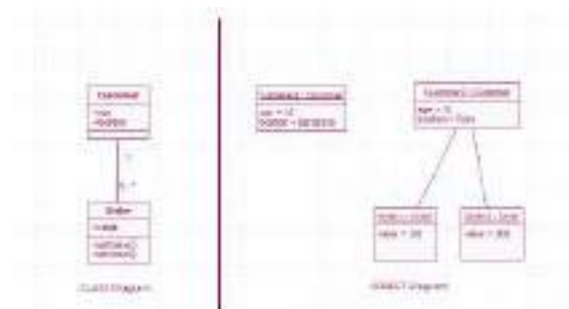
Theory:

Class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation.

Classes are composed of three things: a name, attributes, and operations. Below is an example of a class:



Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment. Object diagrams are used to render a set of objects and their relationships as an instance.



Performance Instruction:

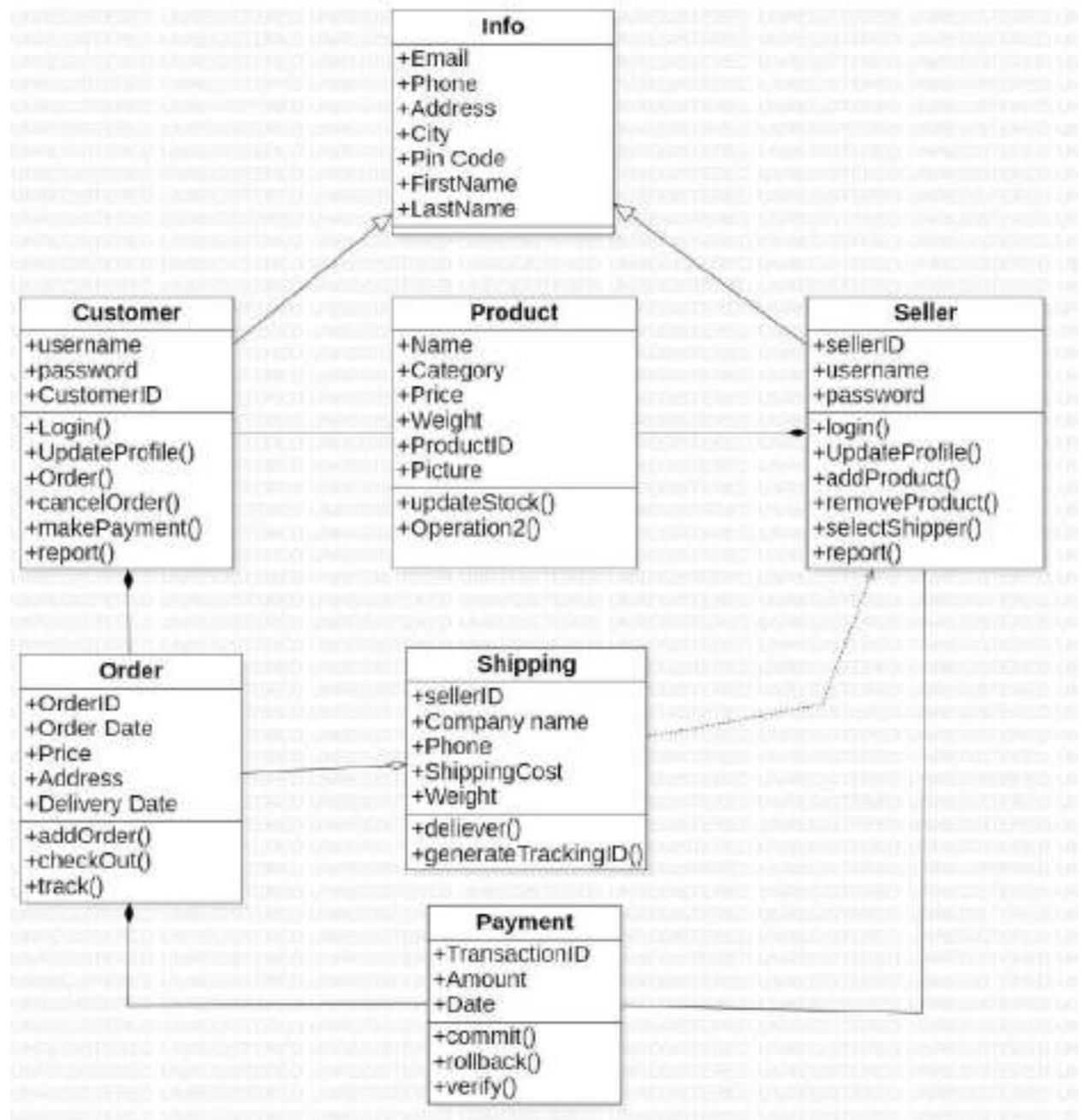
To draw class diagram

1. Identify various elements such as classes, member variables, member functions etc. of the class diagram
2. Draw the class diagram as per the norms

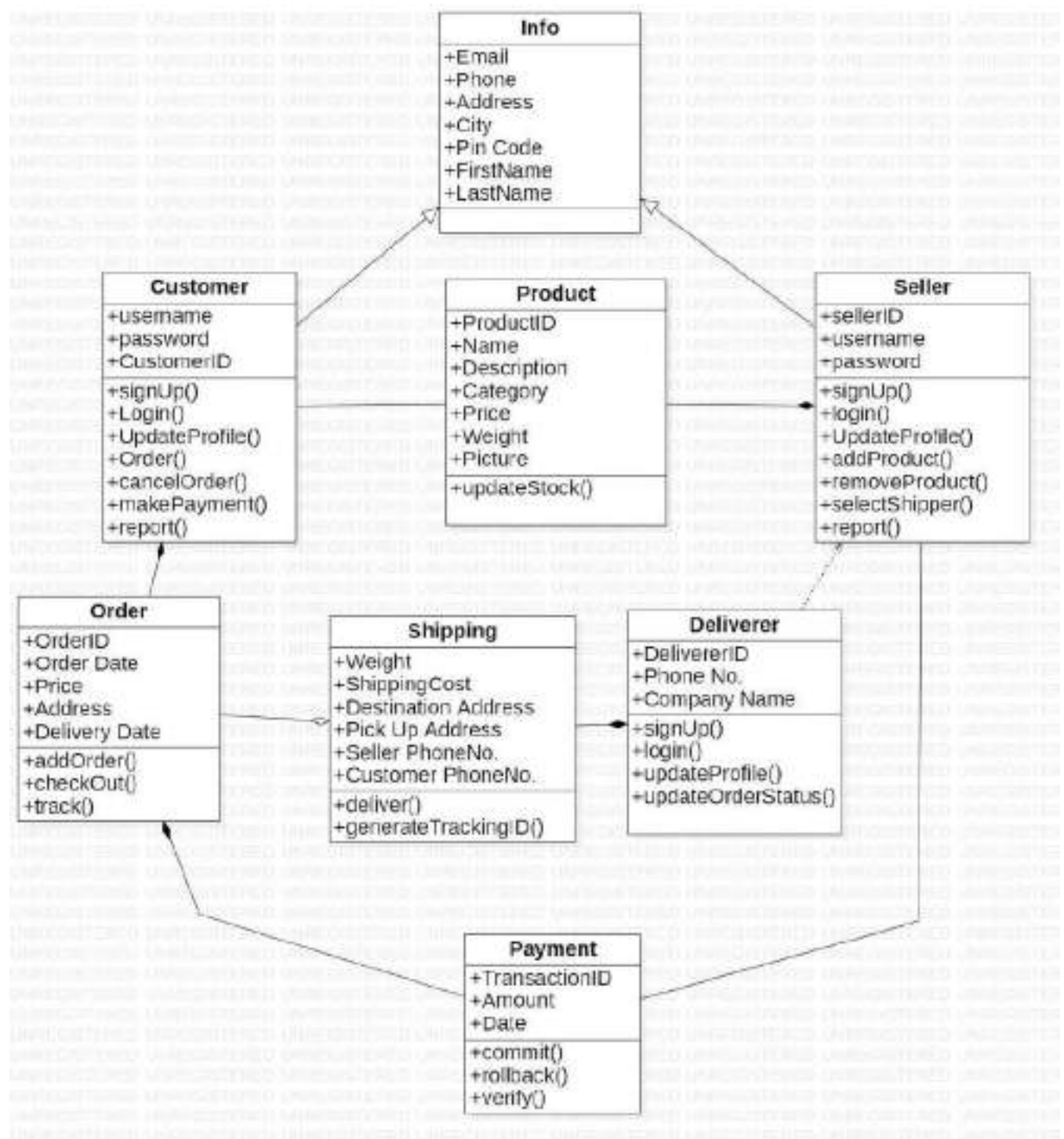
To draw object diagram

1. First, analyse the system and decide which instances have important data and association.
2. Second, consider only those instances, which will cover the functionality.
3. Third, make some optimization as the number of instances are unlimited.

Class Diagram vs Object Diagram		
More Information Online: WWW.DIFFERENCEBETWEEN.COM		
	Class Diagram	Object Diagram
DEFINITION	A type of static structural diagram that describes structure of the system by showing classes, their attributes, methods and relationship among classes.	A type of static structural diagram that shows a complete or partial view of the structure of a modeled system at a specific time.
MAIN REPRESENTATION	Defines classes or blue prints and show how they relate to each other.	Shows the objects and their relationships.
METHOD OF WRITING	In a class diagram, the class name starts with uppercase. e.g. - Student	In an object diagram, the object name is in lowercase, and it is underlined. e.g., <u>sl</u> : Student

Output:**Class Diagram**

Object Diagram



VIVA QUESTIONS

Cap - 5

Q1 Explain class diagram?

Ans A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Q2 Explain four types of relationship used in class diagram?

Ans Inheritance (or generalization)
A generalization is a taxonomic relationship b/w a more general classifier & a more specific classifier. Each instance of the specific classifier is also an indirect instance of the general classifier.

Association - Association are relationship b/w classes in a UML class diagram. They are represented by a solid line b/w classes. Association are typically named using a verb or verb phrase which reflected the real-world problem domain.

Realization - Realization is a relationship b/w the blueprint class & the object containing its respective implementation level details. This object is said to realize the blueprint class.

Dependency - An object of one class might use an object of another class in the code of a method. If the object is not stored in any field, then this is modelled as a dependency relationship.

Q3 Explain tern classes, interfaces, collaboration & dependency?

Ans Classes - A template for creating objects & implementing behaviour in a system. In UML, a class represents an object or a set of objects that share a common structure & behaviour. They are represented by a rectangle that include two parts of the class name, its attribute & its operation.

Interface - Interface are model elements that define a set of operations that other model elements such as classes, or components must implement. A model element realized an interface by overridding each of the operations that the interface declares.

Collaborations - It is used to show the relationship b/w the object in a system. Both the sequence & the collaboration diagram represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features.

Dependency - An object of one class might use an object of another class in the code of a method. If the object is not stored in any field then this modeled as a dependency relationship.

Q4 Explain object diagram?

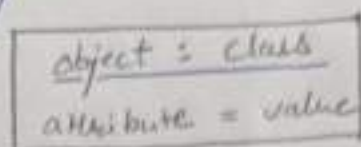
Ans - object diagram represent an instance of a class diagram. The basic concept are similar diagram of object diagrams. object diagram ~~is~~ also represent the static view of a system but this static view is a snapshot of the system at a particular moment.

Q5 Explain symbols used in it?

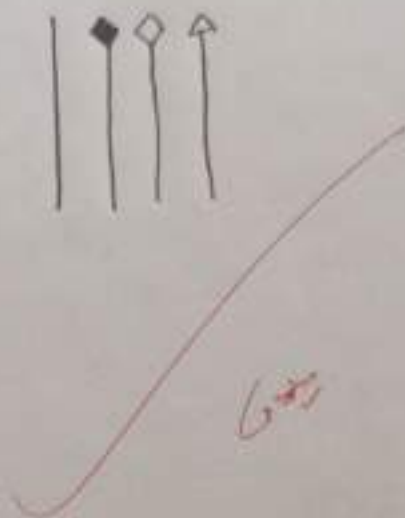
Object Names - Every object is actually symbolized like a rectangle, that offers the name from the object & its class underlined as well as divided with a colon.

object : class

Object Attributes - Similar to classes, you are able to list object attributes inside a separate compartment. However, unlike classes, object attributes should have values assigned for them.



Links - Links tend to be instances associated with associations - you can draw a link while using the lines utilized in class diagrams.



Viva Questions

1. Explain class diagram?

Ans.

A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

2. Explain four types of relationship used in class diagram?

Ans.

Inheritance (or Generalization):

A generalization is a taxonomic relationship between a more general classifier and a more specific classifier. Each instance of the specific classifier is also an indirect instance of the general classifier.

Association

Associations are relationships between classes in a UML Class Diagram. They are represented by a solid line between classes. Associations are typically named using a verb or verb phrase which reflects the real-world problem domain.

Realization

Realization is a relationship between the blueprint class and the object containing its respective implementation level details. This object is said to realize the blueprint class.

Dependency

An object of one class might use an object of another class in the code of a method. If the object is not stored in any field, then this is modelled as a dependency relationship.

3. Explain terms classes, interfaces, collaborations and dependency?

Ans.

Classes

A template for creating objects and implementing behavior in a system. In UML, a class represents an object or a set of objects that share a common structure and behavior. They're represented by a rectangle that includes rows of the class name, its attributes, and its operations

Interfaces

Interfaces are model elements that define sets of operations that other model elements, such as classes, or components must implement. An implementing model element realizes an interface by overriding each of the operations that the interface declares.

Collaborations

It 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. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features.

Dependency

An object of one class might use an object of another class in the code of a method. If the object is not stored in any field, then this is modeled as a dependency relationship.

4. Explain object diagram?

Ans.

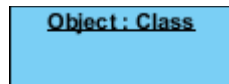
Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.

5. Explain symbols used in it?

Ans.

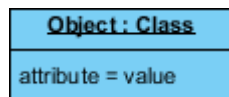
Object Names:

- Every object is actually symbolized like a rectangle, that offers the name from the object and its class underlined as well as divided with a colon.



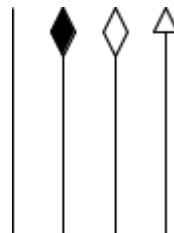
Object Attributes:

- Similar to classes, you are able to list object attributes inside a separate compartment. However, unlike classes, object attributes should have values assigned for them.



Links:

- Links tend to be instances associated with associations. You can draw a link while using the lines utilized in class diagrams.





EXPERIMENT - 6

Software Engineering Lab

Aim

To draw the behavioral view diagram: State-chart diagram, Activity diagram.

Syeda Reeha Quasar

14114802719

4C7

EXPERIMENT – 6

Aim:

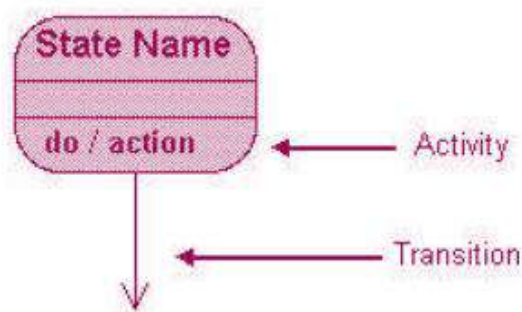
To draw the behavioural view diagram: State-chart diagram, Activity diagram.

Theory:

behavioral diagrams visualize, specify, construct, and document the dynamic aspects of a system. The behavioral diagrams are categorized as follows: use case diagrams, interaction diagrams, state-chart diagrams, and activity diagrams.

State-Chart Diagrams

State chart diagrams represent the behavior of entities capable of dynamic behavior by specifying its response to the receipt of event instances.



A state-chart diagram shows a state machine that depicts the control flow of an object from one state to another. A state machine portrays the sequences of states which an object undergoes due to events and their responses to events.

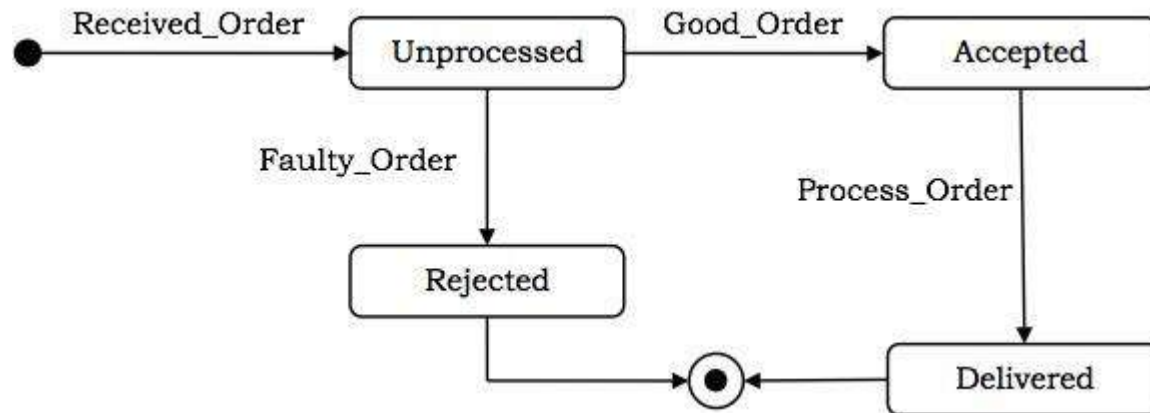
State-Chart Diagrams comprise of –

- States: Simple or Composite
- Transitions between states
- Events causing transitions
- Actions due to the events

State-chart diagrams are used for modeling objects which are reactive in nature.

Example

In the Automated Trading House System, let us model Order as an object and trace its sequence. The following figure shows the corresponding state-chart diagram.



Activity Diagrams

An activity diagram depicts the flow of activities which are ongoing non-atomic operations in a state machine. Activities result in actions which are atomic operations.

Activity diagrams describe the workflow behavior of a system. Activity diagrams are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams can show activities that are conditional or parallel. Activity diagrams show the flow of activities through the system. Diagrams are read from top to bottom and have branches and forks to describe conditions and parallel activities. A fork is used when multiple activities are occurring at the same time. The diagram below shows a fork after activity1. This indicates that both activity2 and activity3 are occurring at the same time. After activity2 there is a branch. The branch describes what activities will take place based on a set of conditions. All branches at some point are followed by a merge to indicate the end of the conditional behavior started by that branch. After the merge all of the parallel activities must be combined by a join before transitioning into the final activity state.

Activity diagrams comprise of –

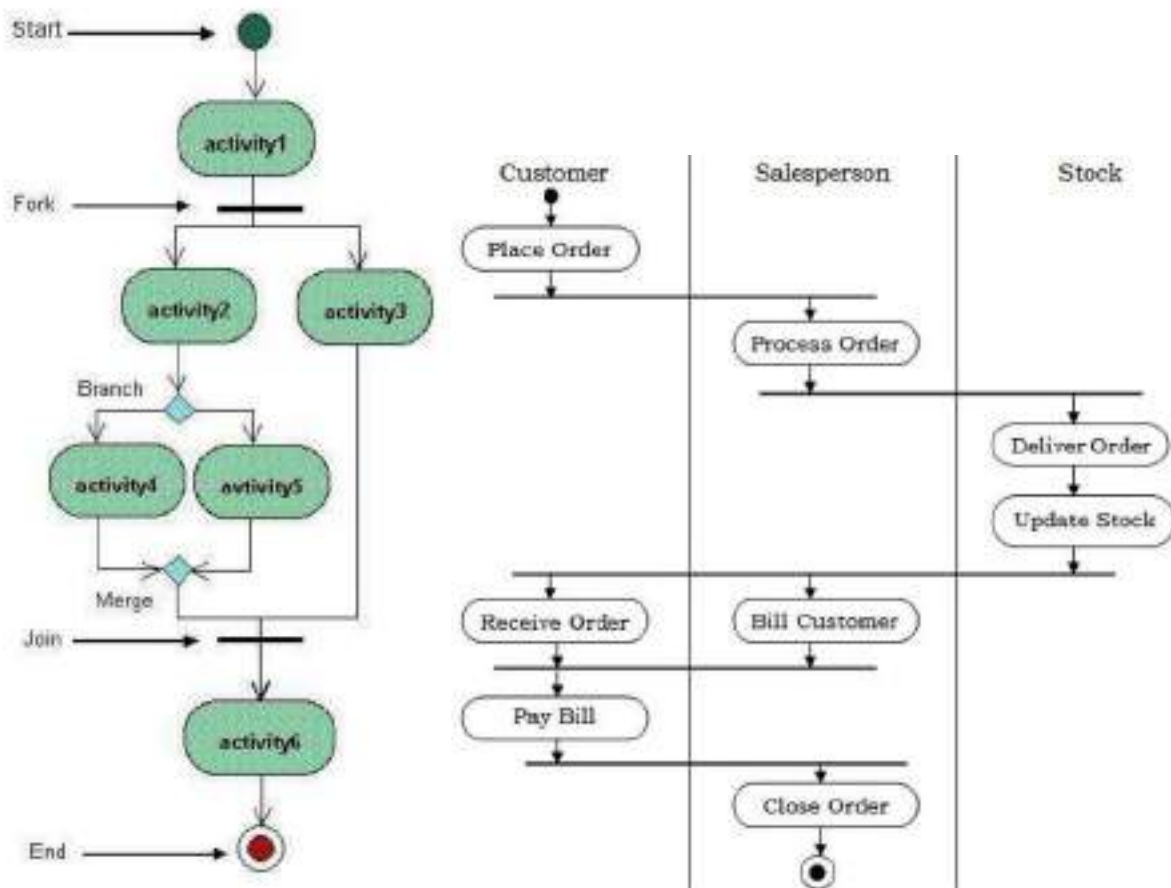
- Activity states and action states
- Transitions
- Objects

Activity diagrams are used for modeling –

- workflows as viewed by actors, interacting with the system.
- details of operations or computations using flowcharts.

Example

The following figure shows an activity diagram of a portion of the Automated Trading House System.



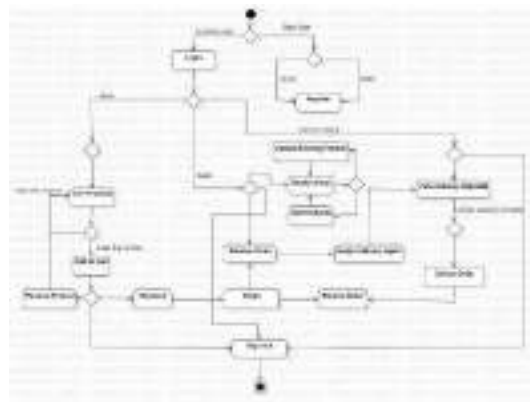
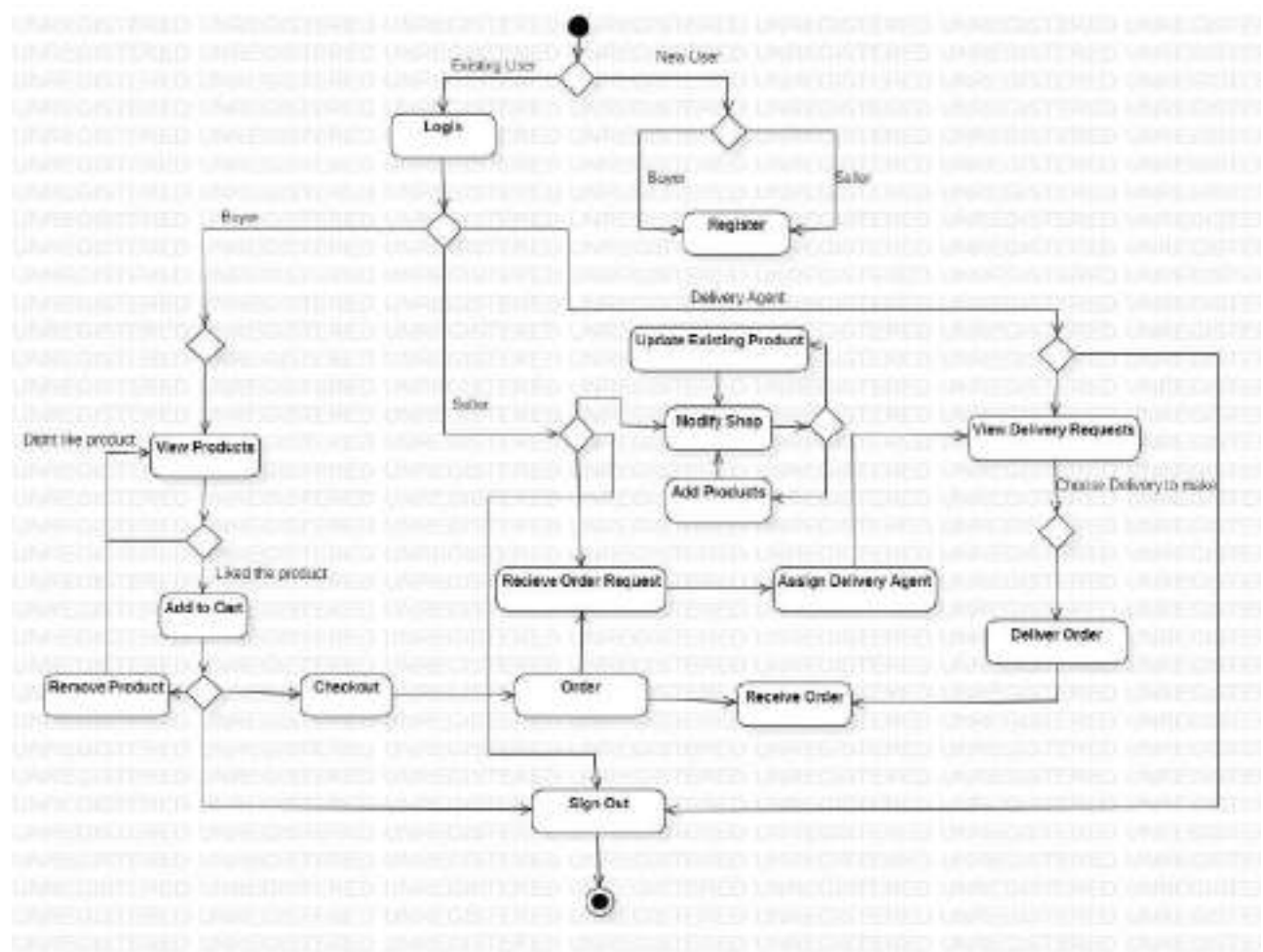
Performance Instruction:

To draw state chart diagram

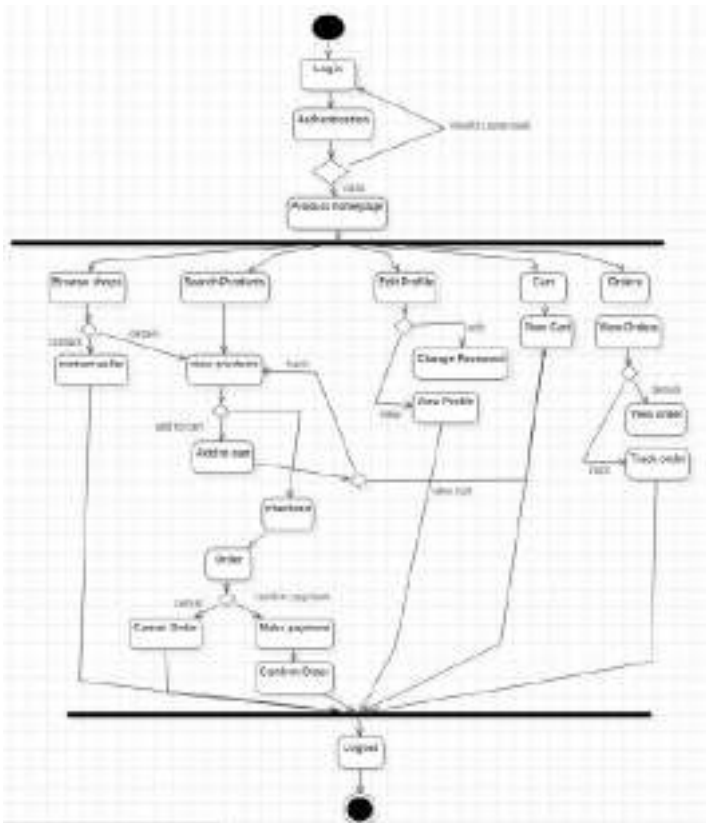
1. Identify various elements states and their different transition of the state-chart diagram.
2. Draw the state-chart diagram as per the norms.

To draw activity diagram

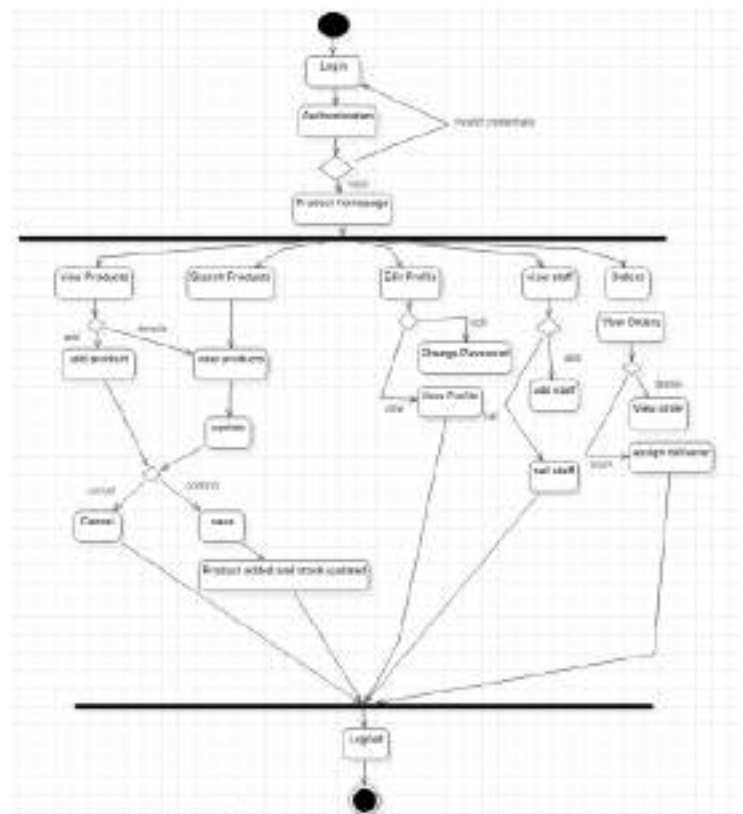
1. Identify various elements such as different activity their boundaries etc. of the activity diagram.
2. Draw the activity diagram as per the norms.

Output:**State Chart Diagram**

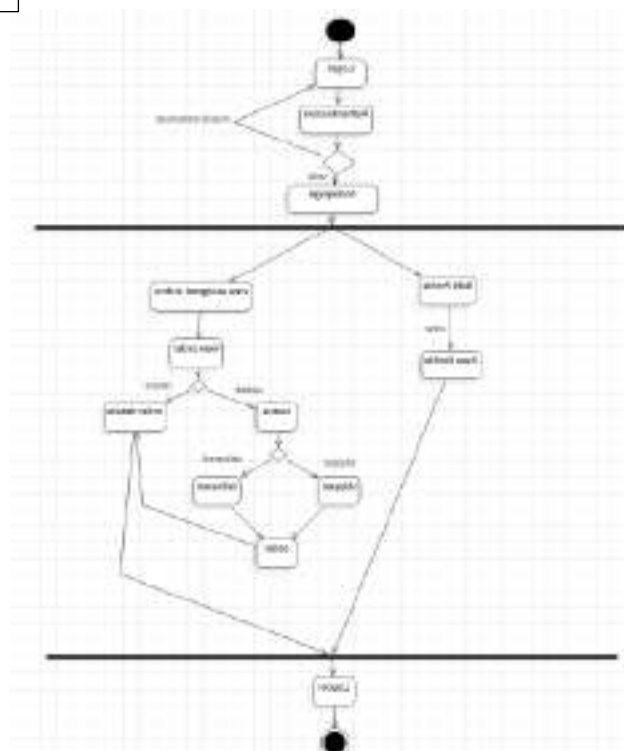
Activity Diagram



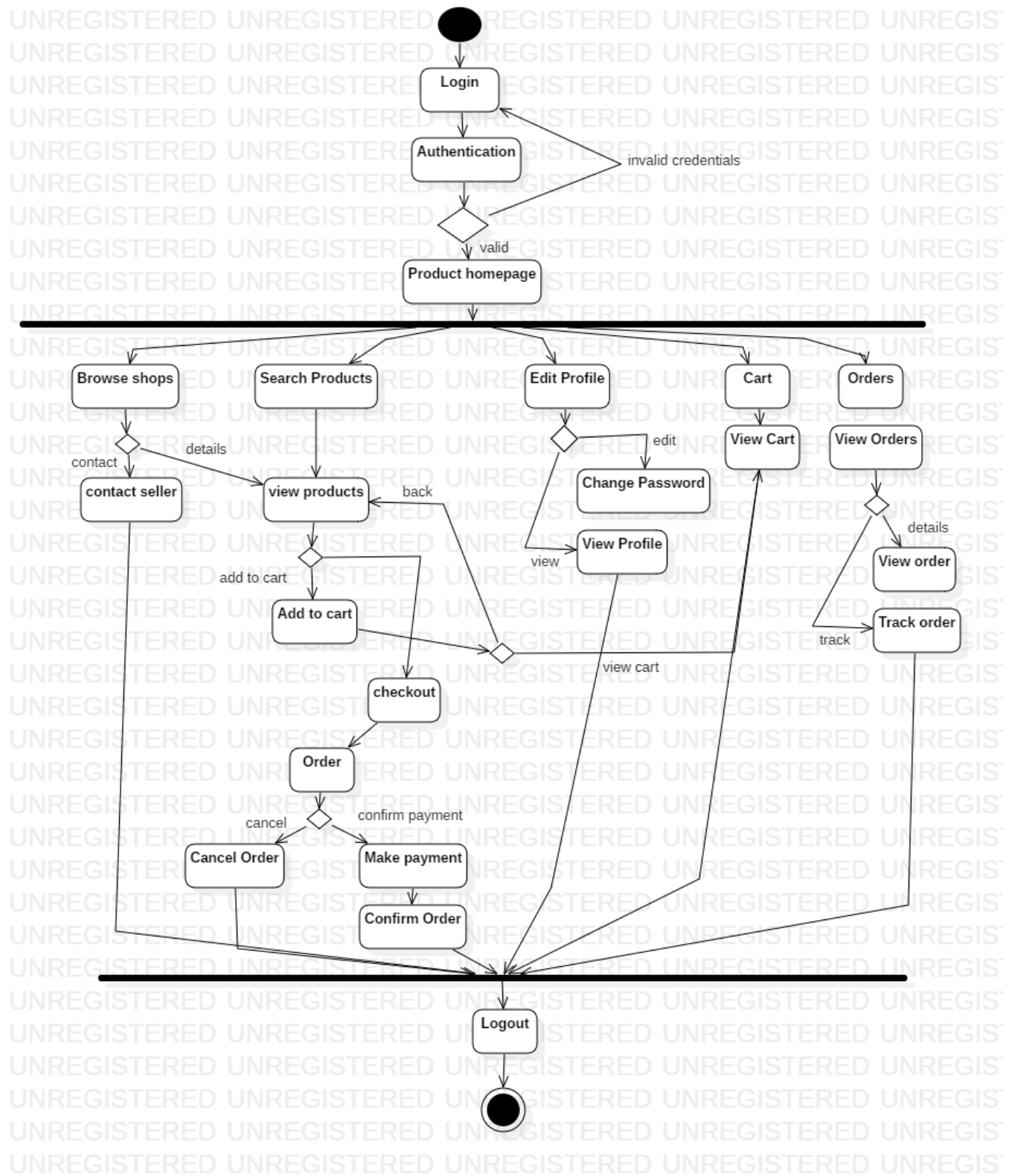
Buyer



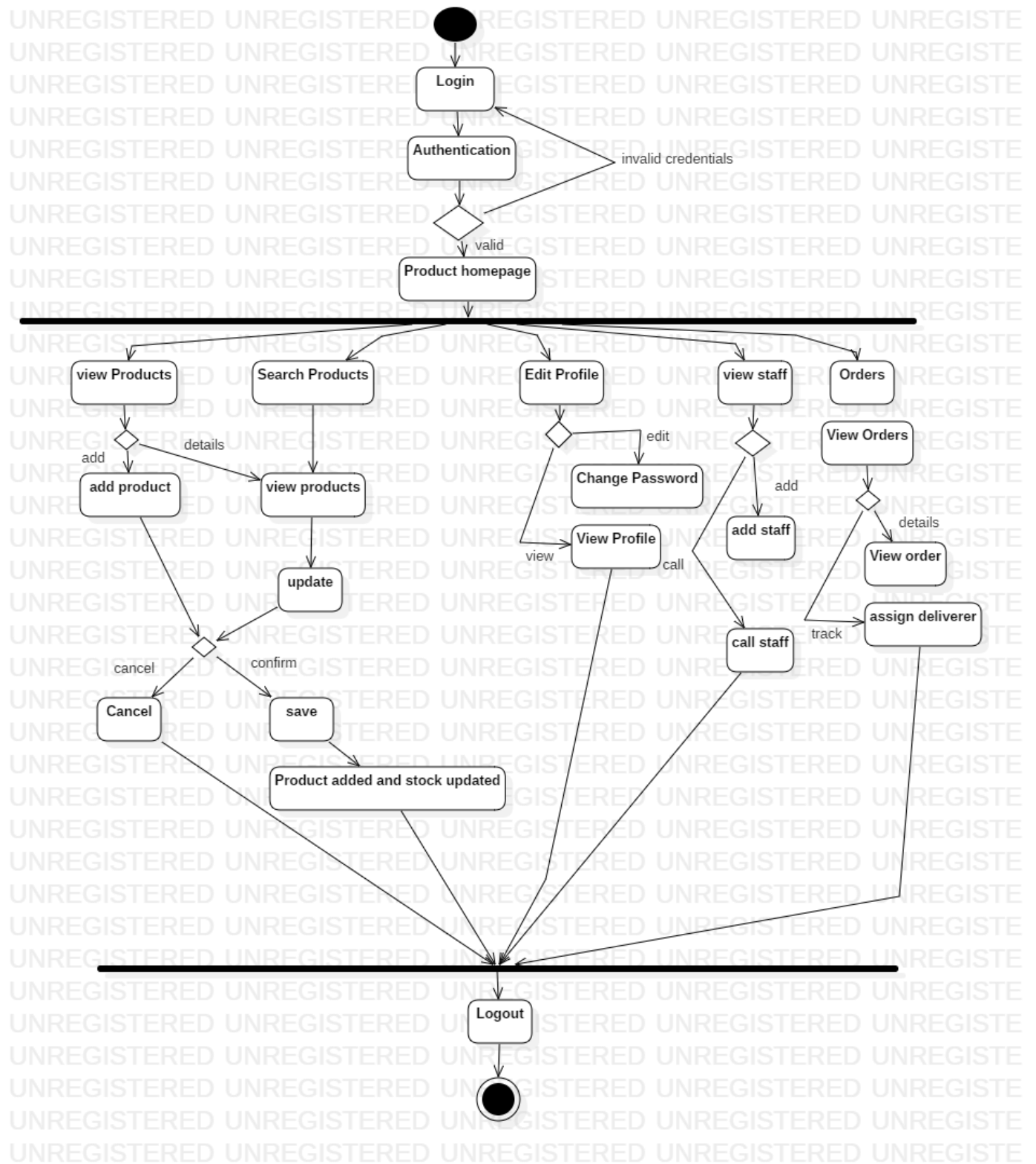
Seller



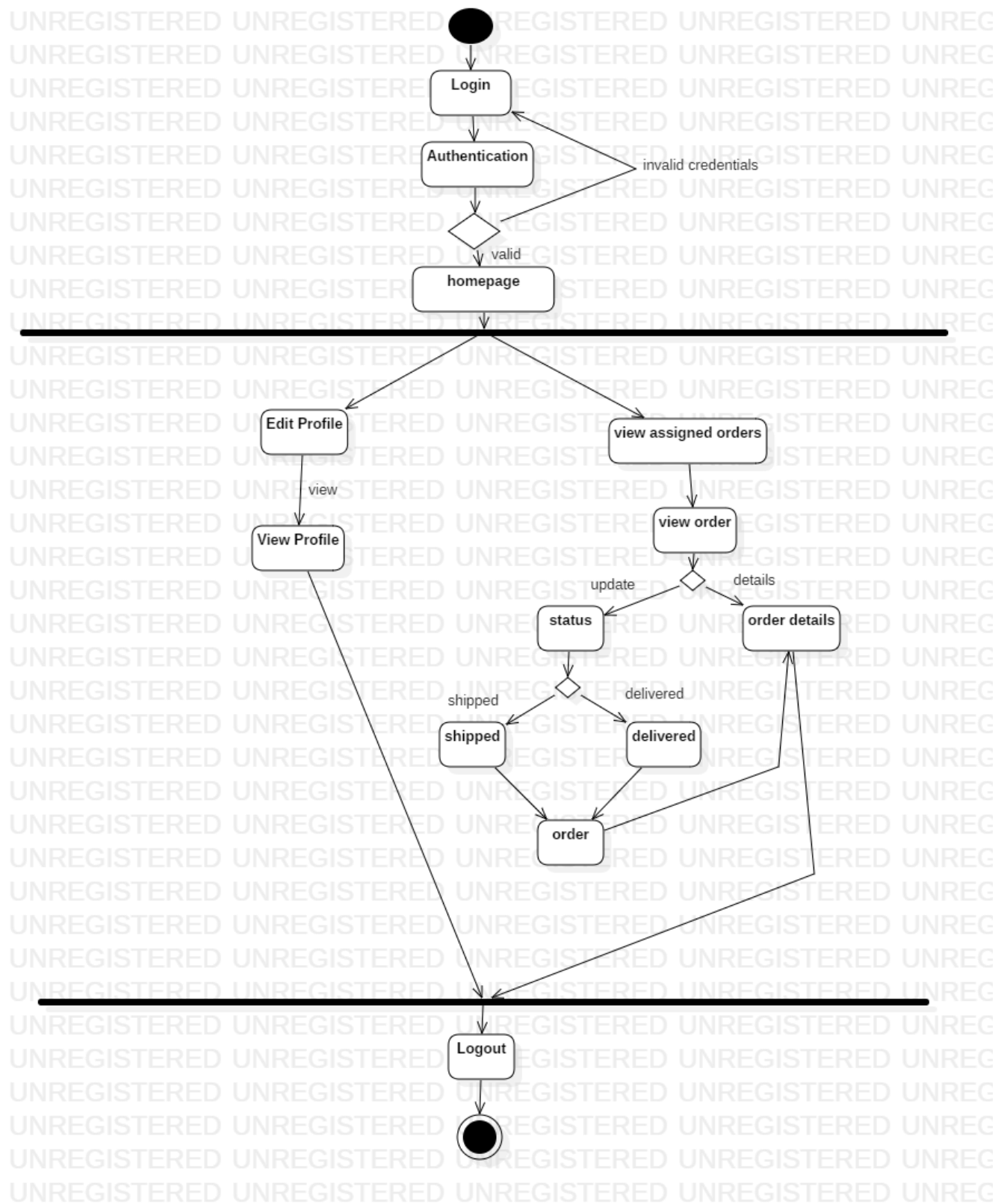
Deliverer



Buyer



Seller



Deliverer

Conclusion:

State Chart and Activity diagram were made successfully by following above steps.

Viva Questions

1. How activity diagram explains behavioral view of system?

Ans.

An activity diagram is a behavioral diagram i.e. it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

2. Explain steps followed to construct activity diagram?

Ans.

Step 1: Figure out the action steps from the use case

Here you need to identify the various activities and actions your business process or system is made up of.

Step 2: Identify the actors who are involved

If you already have figured out who the actors are, then it's easier to discern each action they are responsible for.

Step 3: Find a flow among the activities

Figure out in which order the actions are processed. Mark down the conditions that have to be met in order to carry out certain processes, which actions occur at the same time and whether you need to add any branches in the diagram. And do you have to complete some actions before you can proceed to others?

Step 4: Add swimlanes

You have already figured out who is responsible for each action. Now it's time to assign them a swimlane and group each action they are responsible for under them.

3. How state chart diagram explains behavioral view of system?

Ans.

Statechart diagram defines the states of a component and these state changes are dynamic in nature. Its specific purpose is to define the state changes triggered by events. Events are internal or external factors influencing the system.

4. Explain steps followed to construct state chart diagram?

Ans.

1. Identify the initial state and the final terminating states.
2. Identify the possible states in which the object can exist (boundary values corresponding to different attributes guide us in identifying different states).
3. Label the events which trigger these transitions.

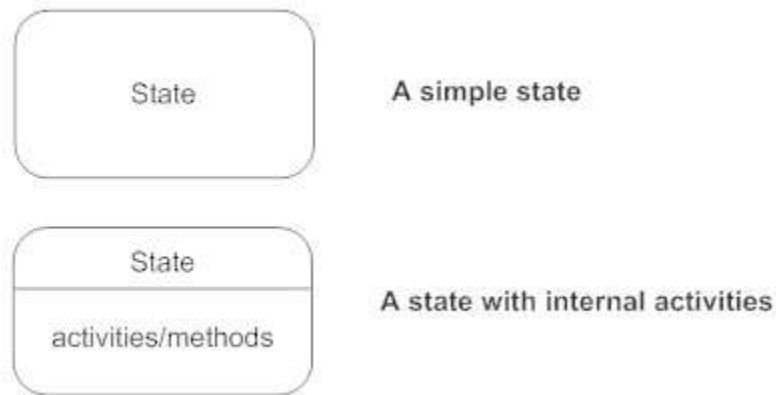
5. Explain symbols used to construct these diagrams?

Ans.

STATE CHART DIAGRAM

States

States represent situations during the life of an object. You can easily illustrate a state in SmartDraw by using a rectangle with rounded corners.



Transition

A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it. A state can have a transition that points back to itself.



Initial State

A filled circle followed by an arrow represents the object's initial state.



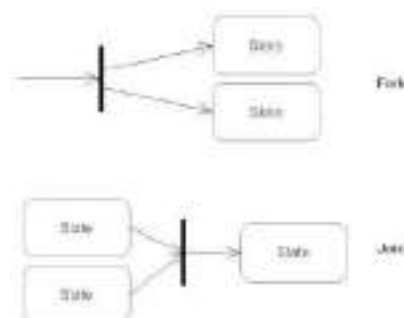
Final State

An arrow pointing to a filled circle nested inside another circle represents the object's final state.



Synchronization and Splitting of Control

A short heavy bar with two transitions entering it represents a synchronization of control. The first bar is often called a fork where a single transition splits into concurrent multiple transitions. The second bar is called a join, where the concurrent transitions reduce back to one.



ACTIVITY DIAGRAM

Initial State or Start Point

A small filled circle followed by an arrow represents the initial action state or the start point for any activity diagram. For activity diagram using swimlanes, make sure the start point is placed in the top left corner of the first column.



Activity or Action State

An action state represents the non-interruptible action of objects. You can draw an action state in SmartDraw using a rectangle with rounded corners.



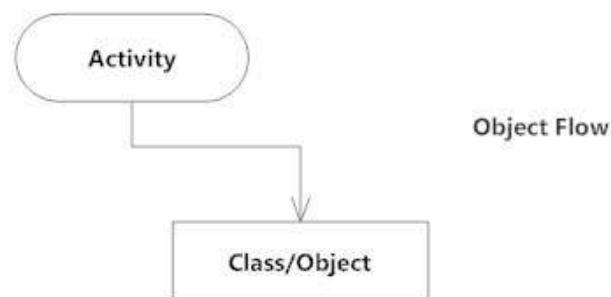
Action Flow

Action flows, also called edges and paths, illustrate the transitions from one action state to another. They are usually drawn with an arrowed line.



Object Flow

Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.



Decisions and Branching

A diamond represents a decision with alternate paths. When an activity requires a decision prior to moving on to the next activity, add a diamond between the two activities. The outgoing alternates should be labeled with a condition or guard expression. You can also label one of the paths "else."



Guards

In UML, guards are a statement written next to a decision diamond that must be true before moving next to the next activity. These are not essential, but are useful when a specific answer, such as "Yes, three labels are printed," is needed before moving forward.

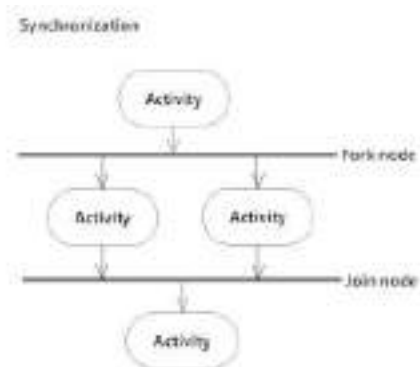


Synchronization

A fork node is used to split a single incoming flow into multiple concurrent flows. It is represented as a straight, slightly thicker line in an activity diagram.

A join node joins multiple concurrent flows back into a single outgoing flow.

A fork and join mode used together are often referred to as synchronization.



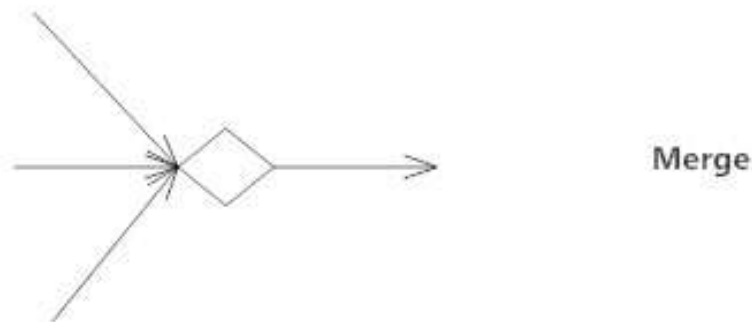
Time Event

This refers to an event that stops the flow for a time; an hourglass depicts it.



Merge Event

A merge event brings together multiple flows that are not concurrent.



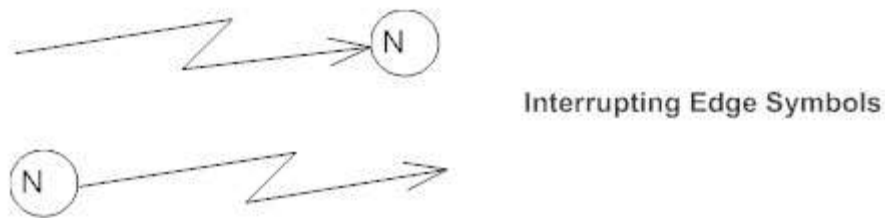
Sent and Received Signals

Signals represent how activities can be modified from outside the system. They usually appear in pairs of sent and received signals, because the state can't change until a response is received, much like synchronous messages in a [sequence diagram](#). For example, an authorization of payment is needed before an order can be completed.



Interrupting Edge

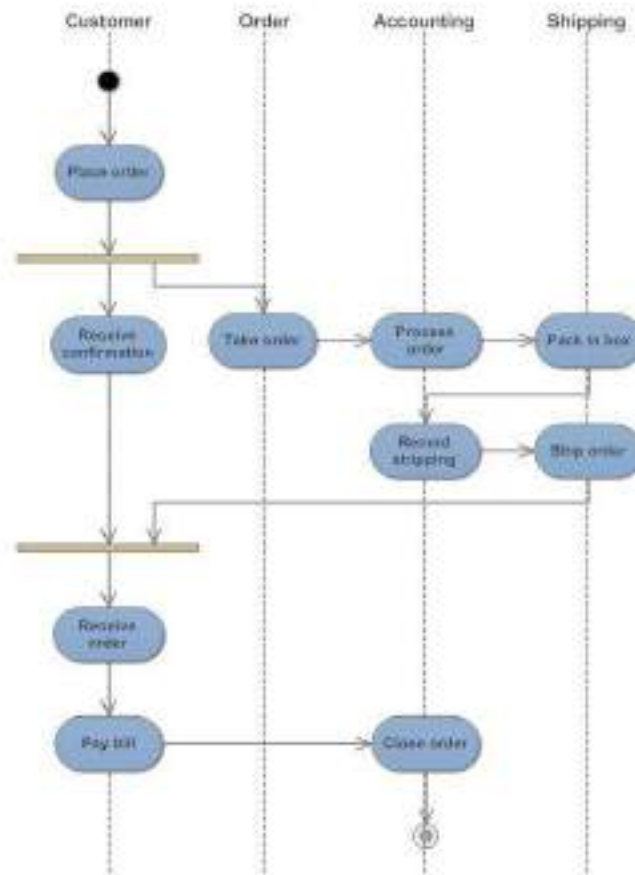
An event, such as a cancellation, that interrupts the flow denoted with a lightning bolt.



Swimlanes

Swimlanes group related activities into one column.

UML Activity Diagram: Order Processing



Final State or End Point

An arrow pointing to a filled circle nested inside another circle represents the final action state.



VIVA QUESTIONS

20-10

Q1 How activity diagram explain behavioural view of system?
Ans An activity diagram is a behavioural diagram. It depicts the behaviour of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

Q2 Explain steps followed to construct activity diagram?

Step 1 - Figure out the action step from the use case
Here you need to identify the various activities of actions your business process or system is made up of.

Step 2 Identify the actors who are involved
If you already have figured out who the actor are, then it's easier to discern each action they are responsible for.

Step 3 Find a flow among the activities
Figure out in which order the actions are processed. Mark down the conditions that have to be met in order to carry out certain processes, which actions occur at the same time & whether you need to add any branches in the diagram. And do you have to complete some actions before you can proceed to others?

Step 4 Add swimlanes - You have already figured out who is responsible for each action. Now it's time to assign them a swimlane & group each action they are responsible for under them.

Q3 How state chart diagram explain behavioural view of system?

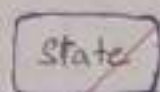
Ans State chart diagram defines that states of a component & these changes are dynamic in nature. Its specific purpose is to define the state changes triggered by events. Event are internal or external factors influencing the system.

- Q4 Explain step followed to construct state chart diagram)
- Identify the final state & the final terminating states.
 - Identify the possible state in which the object can exist (boundary values corresponding to different attributes guides us in identifying different states).
 - Label the event which trigger these transitions.

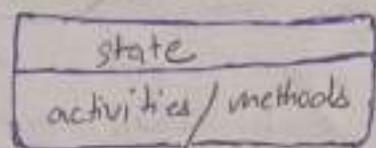
Q5 Explain symbols used to construct these diagram?

STATE CHART DIAGRAM

State



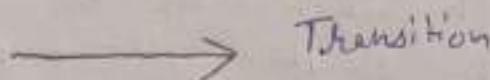
A simple state



A state with internal activities

Transition

Initial state

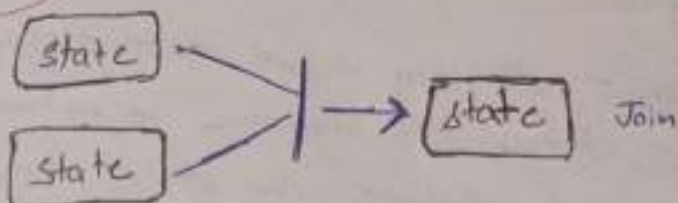
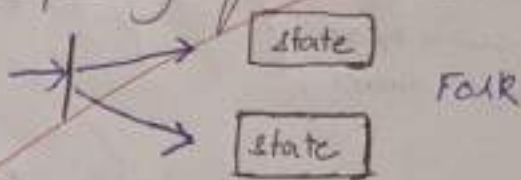


● Initial state

Final state

⦿ Final state

Synchronization & splitting of control



Initial point or start point

● → Start point / Initial point



EXPERIMENT - 7

Software Engineering Lab

Aim

To perform the behavioral view diagram for the suggested system:
Sequence diagram, Collaboration diagram.

Syeda Reeha Quasar

14114802719

4C7

EXPERIMENT – 7

Aim:

To perform the behavioral view diagram for the suggested system: Sequence diagram, Collaboration diagram.

Theory:

Sequence diagrams show potential interactions between objects in the system being defined. Normally these are specified as part of a use case or use case flow and show how the use case will be implemented in the system.

They include:

Objects - oblong boxes or actors at the top - either named or just shown as belonging to a class, from, or to which messages are sent to other objects.

Messages - solid lines for calls and dotted lines for data returns, showing the messages that are sent between objects including the order of the messages which is from the top to the bottom of the diagram.

Object lifelines - dotted vertical lines showing the lifetime of the objects.

Activation - the vertical oblong boxes on the object lifelines showing the thread of control in a synchronous system.

Sequence diagrams show a detailed flow for a specific use case or even just part of a specific use case. They are almost self-explanatory; they show the calls between the different objects in their sequence and can show, at a detailed level, different calls to different objects.

A sequence diagram has two dimensions:

- The vertical dimension shows the sequence of messages/calls in the time order that they occur

- The horizontal dimension shows the object instances to which the messages are sent.

Collaboration Diagram

They are the same as sequence diagrams but without a time axis:

- Their message arrows are numbered to show the sequence of message sending.
- They are less complex and less descriptive than sequence diagrams.

These diagrams are very useful during design because you can figure out how objects communicate with each other.

Performance Instruction:

To draw Sequence Diagram

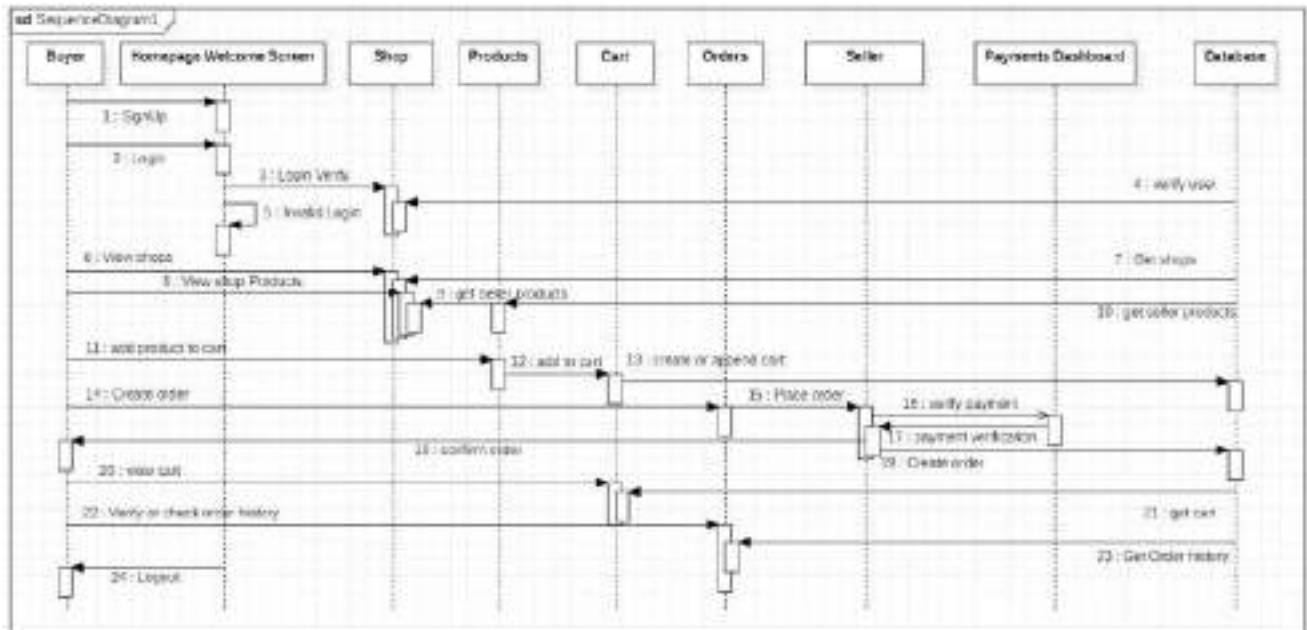
- Identify the class instances (objects) by putting each class instance inside a box.
- If a class instance sends a message to another class instance, draw a line with an open arrowhead pointing to the receiving class instance; place the name of the message/method above the line.
- Optionally, for important messages, you can draw a dotted line with an arrowhead pointing back to the originating class instance; label the return value above the dotted line.

To draw collaboration Diagram

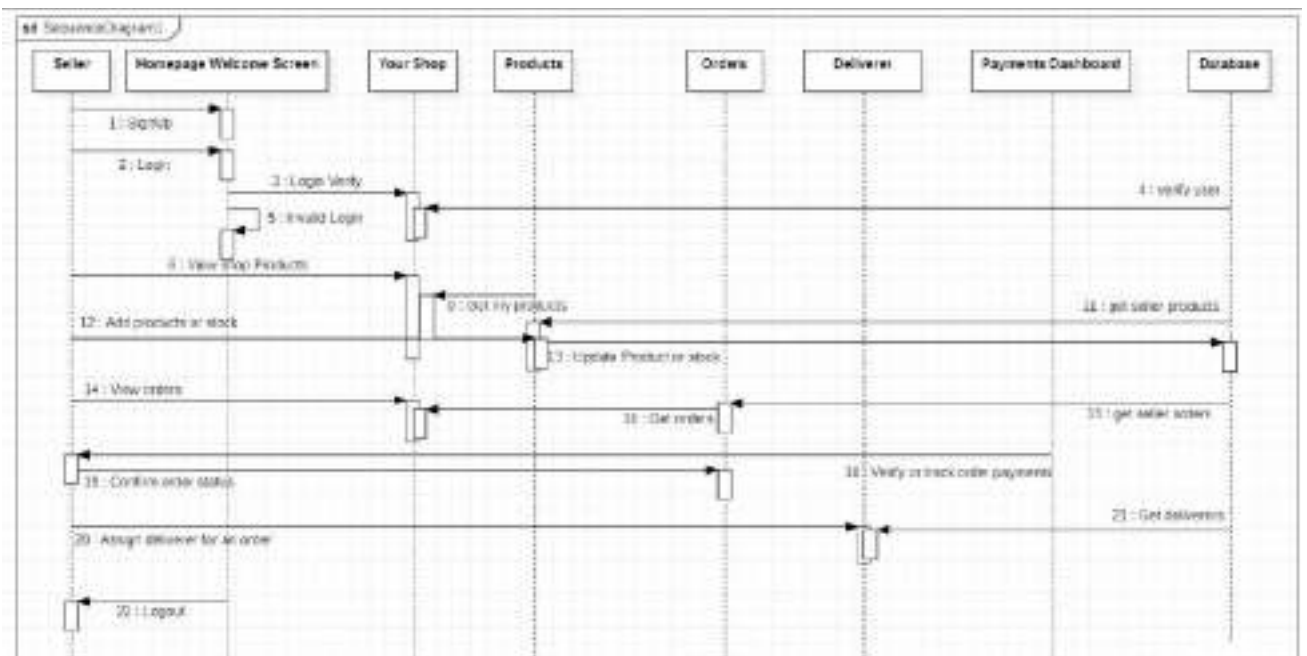
1. By simply pressing combination of keys, we can design collaboration diagram from sequence diagram.

Output:

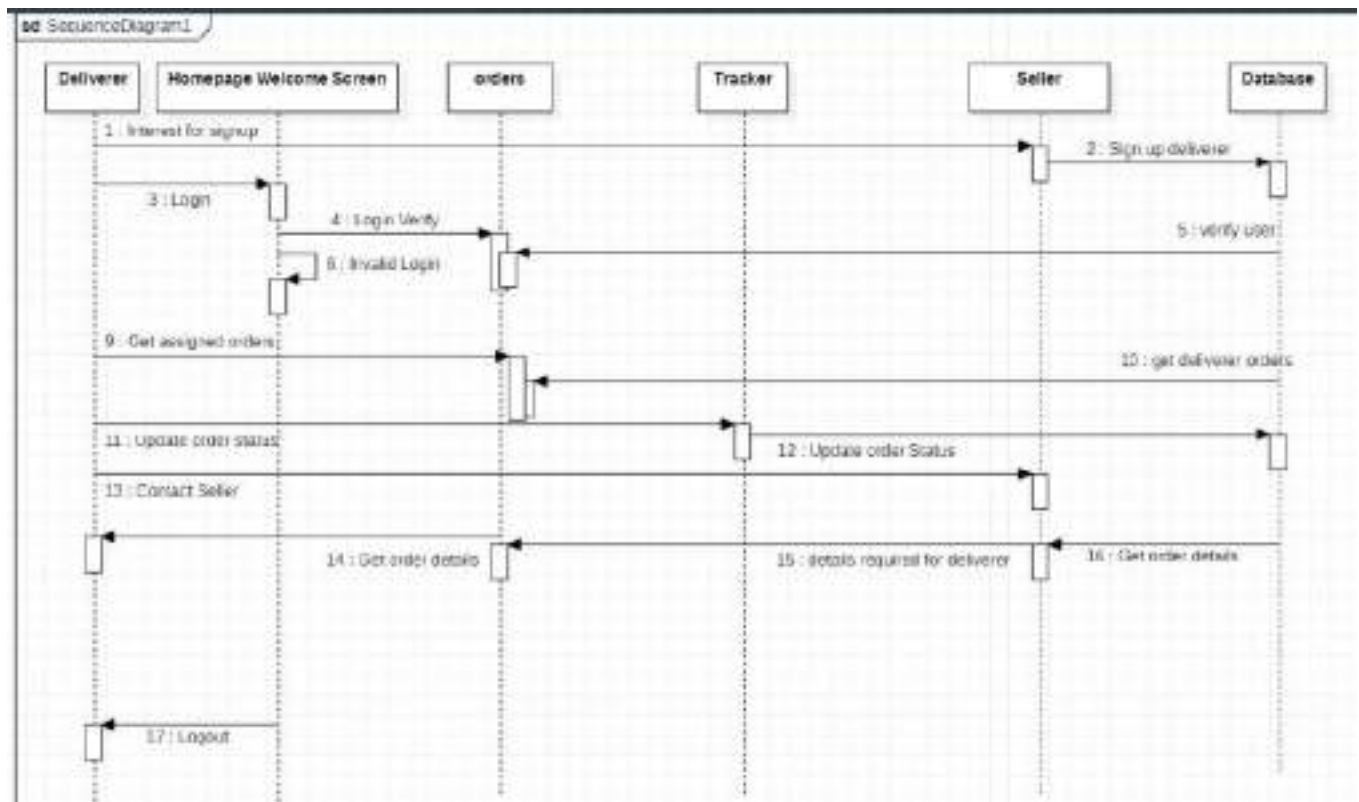
Sequence Diagram for Buyer



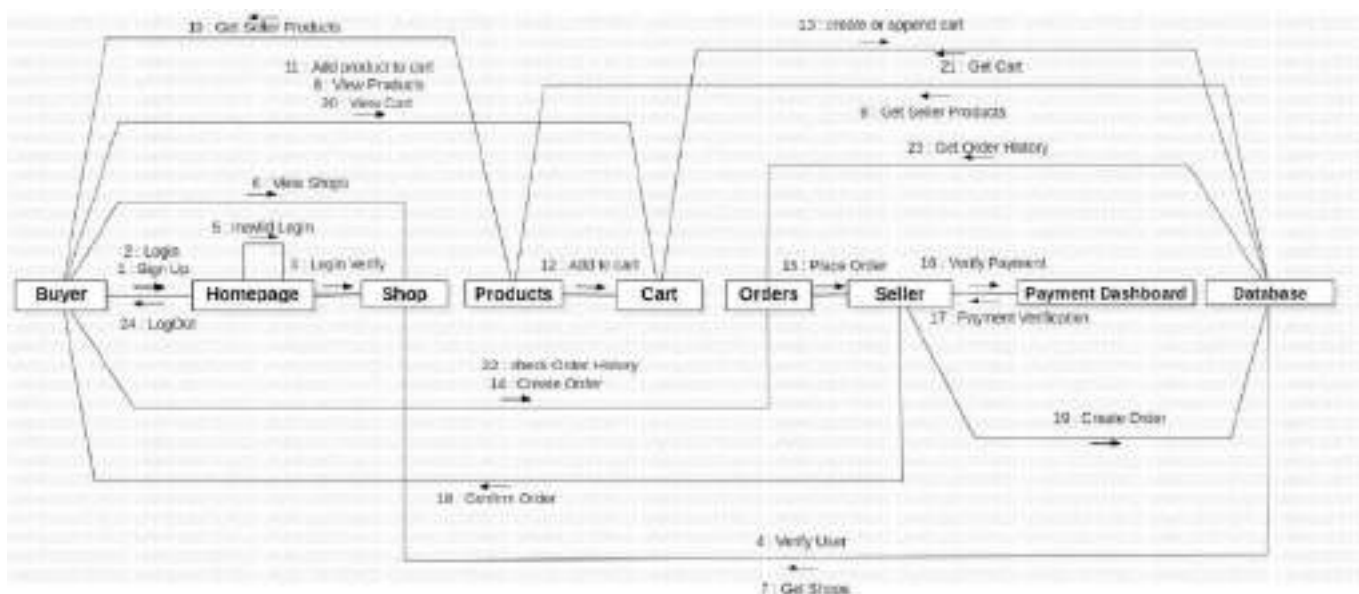
Sequence Diagram for Seller



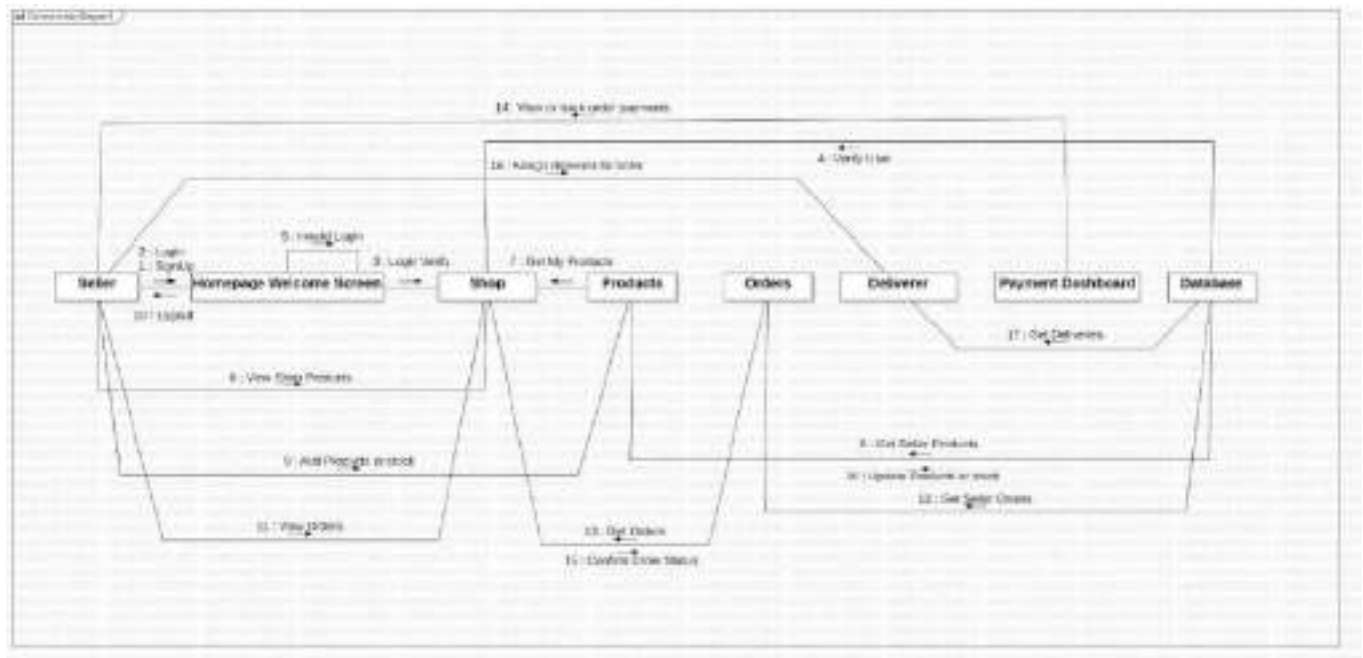
Sequence Diagram for Deliverer



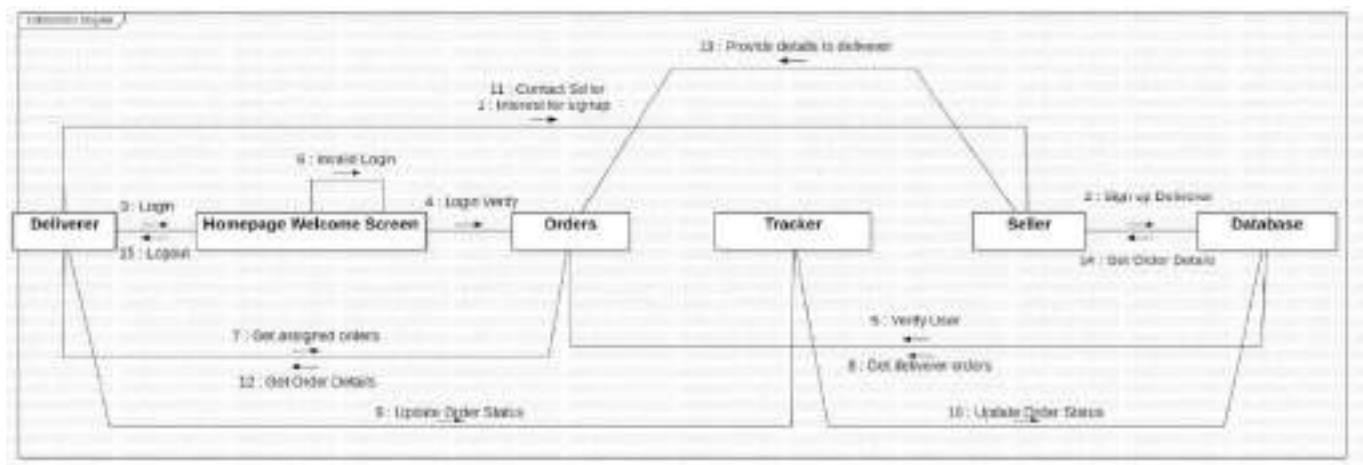
Collaboration Diagram for Buyer



Collaboration Diagram for Seller



Collaboration Diagram for Deliverer



Conclusion:

Sequence diagram and Collaboration Diagram were made successfully by following above steps.

VIVA QUESTIONS

cap - 7

Ques 1. Explain use of sequence diagram?

Ans Following scenarios are ideal for using a sequence diagram:

- Use case scenarios
- Method logic
- Service logic
- Sequence diagram view

Ques 2. Explain steps to draw a sequence diagram.

Ans 1. Identify the class instances (objects) by putting each class instance inside a box.

2. If a class instance sends a message to another class instance draw a line with an open arrowhead pointing to receiving class instance; place the name of message/method above the line.
3. Optionally, for important messages, you can draw a dotted line with an arrowhead pointing back to originating class instance; label the return value above the dotted line.

Ques 3. Explain the need of Collaboration diagram.

- Ans
1. Modelling collaborations, mechanisms, or the structural organization within a system design.
 2. Providing an overview of collaborating objects within a OO object oriented system.
 3. Exhibiting many alternative scenarios for same use case.
 4. Demonstrating forward and reverse engineering.
 5. Capturing the passage of information b/w objects.
 6. Visualizing the complex logic behind an operation.

Ques 4. Explain steps to draw collaboration Diagram.

Ans By simply pressing combination of keys, we can design Collaboration diagram from sequence diagram.

Ques 5. Explain term - entity object, interface object and control object.

Ans Entity Object
Objects that encapsulate the business model, including rules, data, relationships and persistence behaviour.

Interface objects
Objects which provides all details on how 1 piece component exchanges data with another.
eg- communication mode and data structure

Control Objects
These are widget or gadgets and can be used in window dialog box. They cannot exist outside a window or dialog box. You can define controls.

Viva Questions

1. Explain use of sequence diagram?

Ans.

The following scenarios are ideal for using a sequence diagram:

- **Usage scenario:** A usage scenario is a diagram of how your system could potentially be used. It's a great way to make sure that you have worked through the logic of every usage scenario for the system.
- **Method logic:** Just as you might use a UML sequence diagram to explore the logic of a use case, you can use it to explore the logic of any function, procedure, or complex process.
- **Service logic:** If you consider a service to be a high-level method used by different clients, a sequence diagram is an ideal way to map that out.
- **Sequence diagram Visio** - Any sequence diagram that you create with Visio can also be uploaded into Lucidchart. Lucidchart supports .vsd and .vdx file import and is a great Microsoft Visio alternative. Almost all of the images you see in the UML section of this site were generated using Lucidchart.

2. Explain steps to draw sequence diagram?

Ans.

1. Identify the class instances (objects) by putting each class instance inside a box.
2. If a class instance sends a message to another class instance, draw a line with an open arrowhead pointing to the receiving class instance; place the name of the message/method above the line.
3. Optionally, for important messages, you can draw a dotted line with an arrowhead pointing back to the originating class instance; label the return value above the dotted line.

3. Explain need of collaboration diagram?

Ans.

1. Modeling collaborations, mechanisms or the structural organization within a system design.
2. Providing an overview of collaborating objects within an object-oriented system.
3. Exhibiting many alternative scenarios for the same use case.
4. Demonstrating forward and reverse engineering.
5. Capturing the passage of information between objects.
6. Visualizing the complex logic behind an operation.

4. Explain steps to draw collaboration diagram?

Ans.

By simply pressing combination of keys, we can design collaboration diagram from sequence diagram.

5. Explain terms- entity objects, interface objects and control objects?

Ans.

Entity Objects : Entity objects are classes that encapsulate the business model, including rules, data, relationships, and persistence behavior, for items that are used in your business application.

Interface objects: Interface objects provide all details on how one process component exchanges data with another, for example, the mode of communication and the data structures.

Control objects : Control objects are also known as widgets or gadgets and they can be used in windows and dialog boxes. They cannot exist outside a window or dialog box, so you have to define a window or dialog box and select it before you can define controls.



EXPERIMENT - 8

Software Engineering Lab

Aim

To perform the implementation view diagram: Component diagram for the system.

Syeda Reeha Quasar

14114802719

4C7

EXPERIMENT – 8

Aim:

To perform the implementation view diagram: Component diagram for the system.

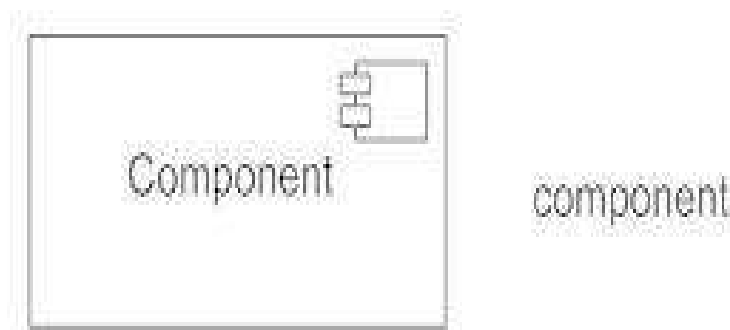
Theory:

A component diagram provides a physical view of the system. Its purpose is to show the dependencies that the software has on the other software components (e.g., software libraries) in the system. The diagram can be shown at a very high level, with just the large-grain components, or it can be shown at the component package level. [Note: The phrase component package level is a programming language-neutral way of referring to class container levels such as .NET's namespaces (e.g., System.Web.UI) or Java's packages (e.g., java.util).

Basic Component Diagram Symbols and Notations

Component

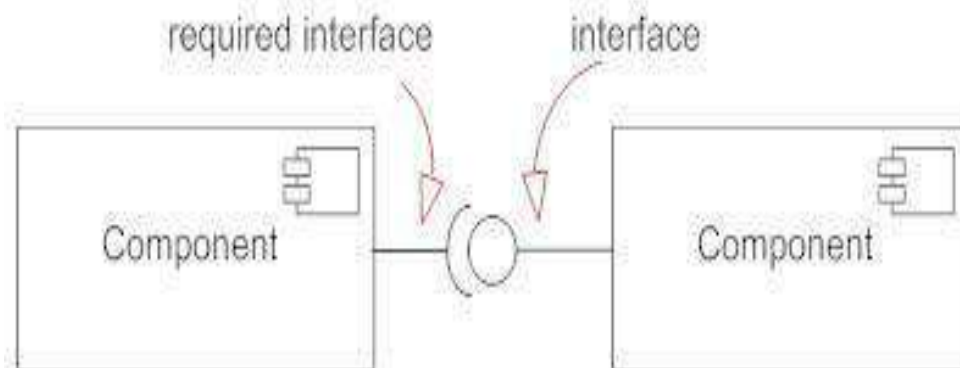
A component is a logical unit block of the system, a slightly higher abstraction than classes. It is represented as a rectangle with a smaller rectangle in the upper right corner with tabs or the word written above the name of the component to help distinguish it from a class.



Interface

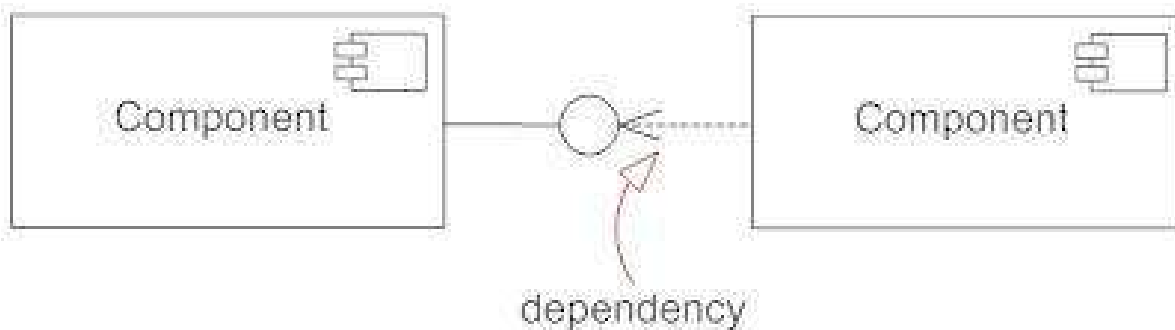
An interface (small circle or semi-circle on a stick) describes a group of operations used (required) or created (provided) by components. A full circle represents an interface

created or provided by the component. A semi-circle represents a required interface, like a person's input.



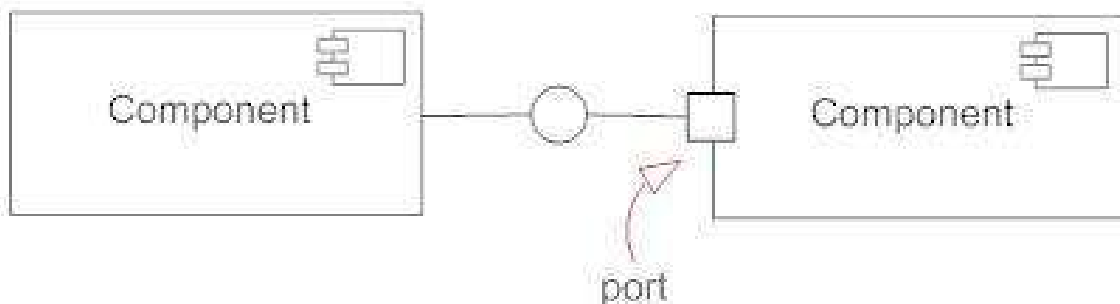
Dependencies

Draw dependencies among components using dashed arrows.



Port

Ports are represented using a square along the edge of the system or a component. A port is often used to help expose required and provided interfaces of a component.

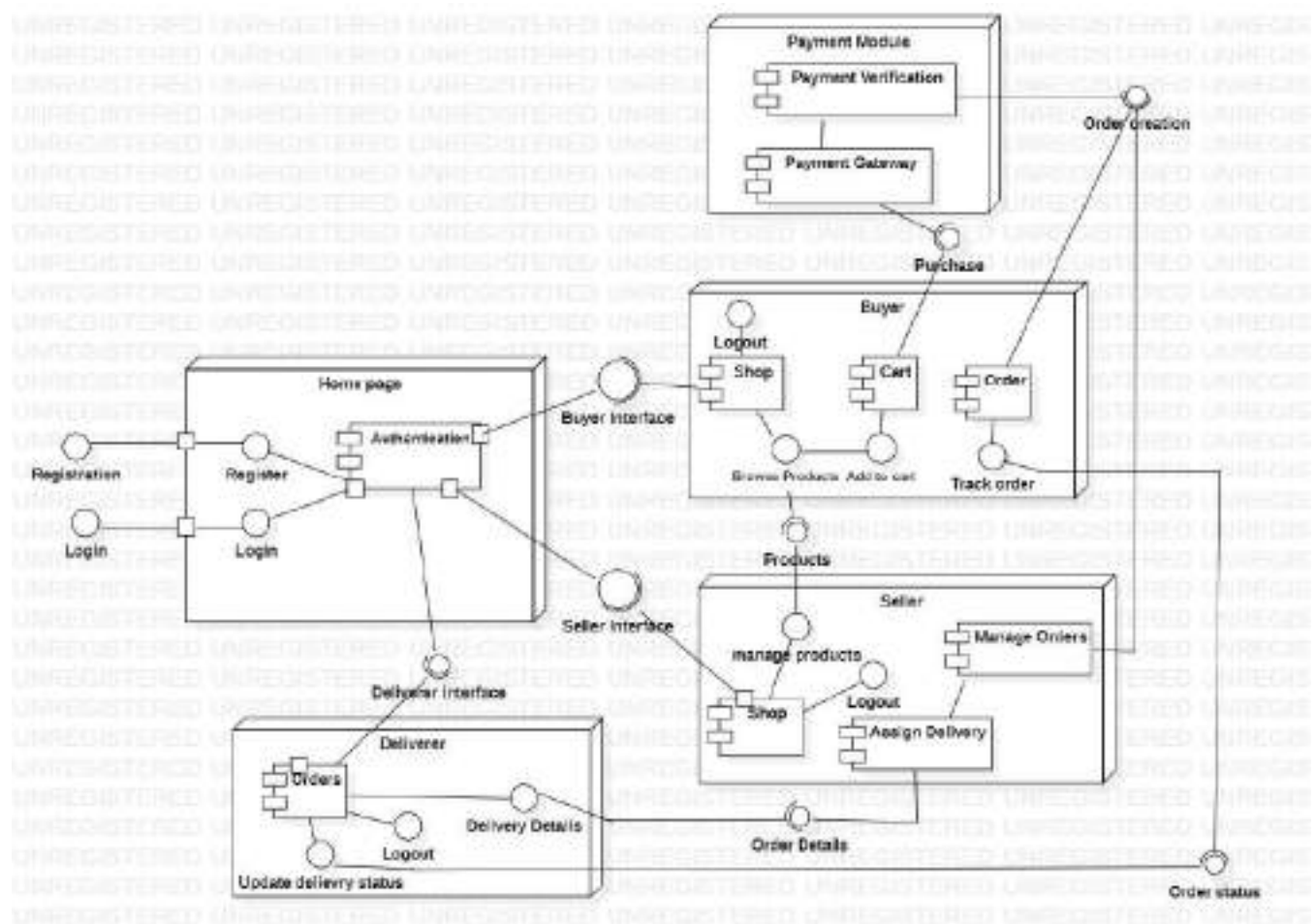


Performance Instruction:

To Draw a Component Diagram

- Take stock of everything needed to implement the planned system. For example, for a simple e-commerce system, you'll need components that describe products, orders, and customer accounts.
- Create a visual for each of the components.
- Describe the organization and relationships between components using interfaces, ports, and dependencies.

Output:



Conclusion:

Component diagram was made successfully by following above steps.

Viva Questions

1. Explain term component diagram?

Ans.

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.

2. Component diagram explains which view of system?

Ans.

The component diagram also describes the static view of a system, which includes the organization of components at a particular instant.

3. Explain steps to draw component diagram?

Ans.

- Take stock of everything needed to implement the planned system. For example, for a simple e-commerce system, you'll need components that describe products, orders, and customer accounts.
- Create a visual for each of the components.
- Describe the organization and relationships between components using interfaces, ports, and dependencies.

4. What is benefit of drawing component diagram?

Ans.

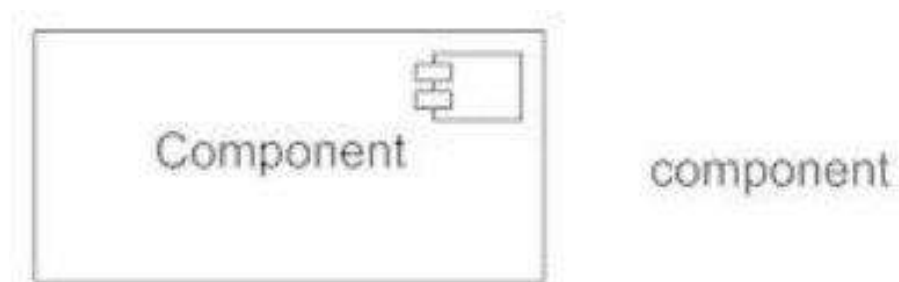
Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

5. Explain symbols used to draw component diagram?

Ans.

Component

A component is a logical unit block of the system, a slightly higher abstraction than classes. It is represented as a rectangle with a smaller rectangle in the upper right corner with tabs or the word written above the name of the component to help distinguish it from a class.

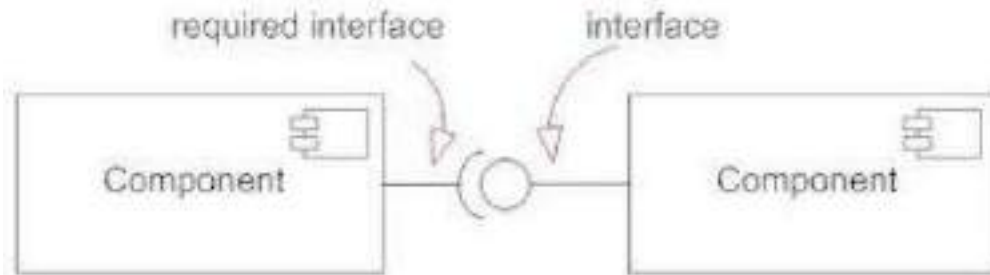


Interface

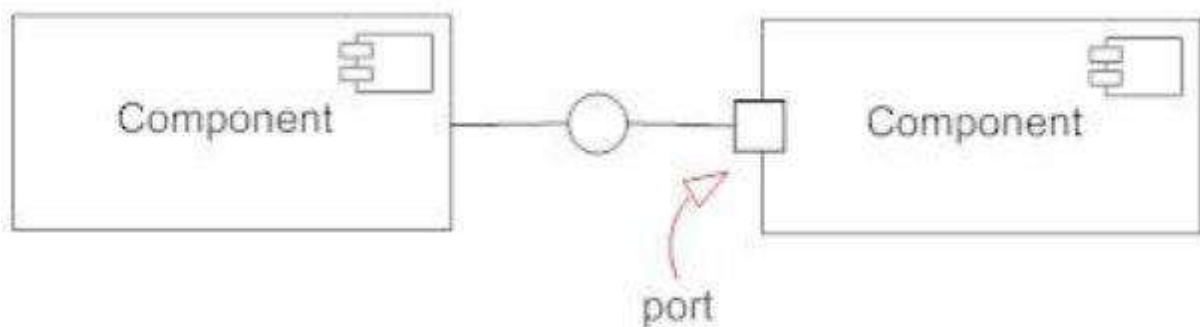
An interface (small circle or semi-circle on a stick) describes a group of operations used (required) or created (provided) by components.

A full circle represents an interface created or provided by the component.

A semi-circle represents a required interface, like a person's input.

**Port**

Ports are represented using a square along the edge of the system or a component. A port is often used to help expose required and provided interfaces of a component.



VIVA QUESTIONS

Ques-2

Ques 1. Explain component diagram.
 Ans 1. Component diagram or UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double check that every aspect of system's required function is covered by planned development.

Ques 2. Component diagram explains which view of system?
 Ans 2. Component diagram describes the static view of the system which includes the organization of components at a particular instant.

Ques 3. Explain steps to draw component diagram.
 Ans 3. 1. Take stock of everything required to implement planned system. For eg. a simple e-commerce system, you'll need components that describe products, orders, and customer account.
 2. Create a visual for each of the components.
 3. Describe the organization and relationships between components using interfaces, ports and dependencies.

Ques 4. What is benefit of drawing component diagram?
 Ans 4. Component diagram is a special kind of diagram in UML. Its purpose is also different from all other diagrams discussed so far. It does not describe the components used to make those functionalities.

Ques 5. Explain symbols used to draw components diagram?
 Component → Logical unit block of the system, a slightly higher abstraction than classes. It is represented as a rectangle with a smaller rectangle in the upper right corner with tabs or the words written above the name of the component to help distinguish it from a class.

Interface → An interface (small circle or semi-circle on a stick) describe a group of operations used (required) or created (provides) by components.

○ → an interface created by component

) → interfaces requiring person's input



EXPERIMENT - 9

Software Engineering Lab

Aim

To perform the environmental view diagram: Deployment diagram for the system.

Syeda Reeha Quasar

14114802719

4C7

EXPERIMENT – 9

Aim:

To perform the environmental view diagram: Deployment diagram for the system.

Theory:

The deployment diagram shows how a system will be physically deployed in the hardware environment. Its purpose is to show where the different components of the system will physically run and how they will communicate with each other. Since the diagram models the physical runtime, a system's production staff will make considerable use of this diagram.

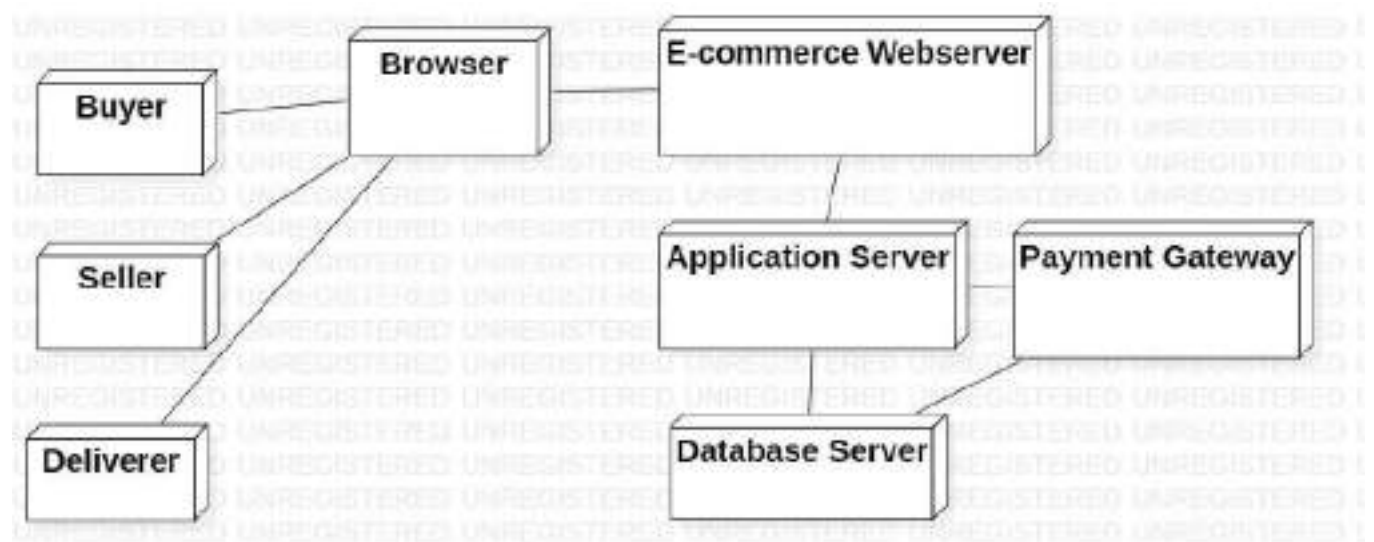
The notation in a deployment diagram includes the notation elements used in a component diagram, with a couple of additions, including the concept of a node. A node represents either a physical machine or a virtual machine node (e.g., a mainframe node). To model a node, simply draw a three-dimensional cube with the name of the node at the top of the cube.

Performance Instruction:

To create a deployment diagram

- Identify component
- Add shapes
- Connect Nodes
- Format arrows

Output:



Conclusion:

Deployment diagram was made successfully by following above steps.

Viva Questions

1. What is deployment diagram?

Ans.

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed. Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

2. Deployment diagram explains which view of system?

Ans.

Deployment diagrams are used to describe the static deployment view of a system.

3. Explain steps to draw deployment diagram?

Ans.

To create a deployment diagram

1. Identify component
2. Add shapes
3. Connect Nodes
4. Format arrows

4. Explain symbols used to draw deployment diagram?

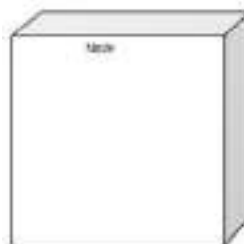
Ans.

Communication path: A straight line that represents communication between two device nodes.

Artifacts: A box with the header ">" and then the name of the file.

Package: A file-shaped box that groups together all the device nodes to encapsulate the entire deployment.

Nodes



There are two types of nodes in a deployment diagram: device nodes and execution environment nodes. Device nodes are computing resources with processing capabilities and the ability to execute programs. Some examples of device nodes include PCs, laptops, and mobile phones.

5. What is benefit of drawing deployment diagram?

Ans.

System engineers mainly consume deployment diagrams. These diagrams help us to describe the physical components like hardware involved, participant nodes, their distribution and how they are inter-connected. Deployment diagrams could be assumed as the hardware components where the software components reside.

VIVA QUESTIONS

Ent - 7

Ques 1. What is deployment diagram?
Ans 1. Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed. Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

Ques 2. Deployment diagram explains which view of system?
Ans 2. Deployment diagrams are used to describe the static deployment view of a system.


Ques 3. Explain steps to draw deployment diagram?

Ans 3. To create a deployment diagram:
1. Identify component
2. Add shapes
3. Connect nodes
4. Format arrows

Ques 4. Explain symbols used to draw deployment diagram?
Ans 4. Communication path: straight line that represents communication between two device nodes.

Artifact: A box with header ">" and then the name of the file

Package: A file-shaped box that groups together all the device nodes to encapsulate the entire deployment

Nodes:  There are 2 types of nodes in a deployment diagram: device nodes and execution environment nodes.

Ques 5. What is the benefit of drawing deployment diagram?

Ans 5. System engineering mainly consumes deployment diagrams. These diagrams help us to describe the physical components like hardware involved, participant nodes, their distribution and how they are inter-connected. Deployment diagrams could be assumed as the hardware components where the software components reside.



EXPERIMENT - 10

Software Engineering Lab

Aim

To perform various testing using the testing tool unit testing, integration testing for a sample code of the suggested system.

Syeda Reeha Quasar

14114802719

4C7

EXPERIMENT – 10

Aim:

To perform various testing using the testing tool unit testing, integration testing for a sample code of the suggested system.

Theory:

“Testing is the process of executing a program with the intent of finding errors.”

The purpose of software testing is:

1. To demonstrate that the product performs each function intended.
2. To demonstrate that the internal operation of the product performs according to specification and all internal components have been adequately exercised.
3. To increase our confidence in the proper functioning of the software.
4. To show the product is free from defect.
5. All of the above.

Unit Testing — Checks each coded module for the presence of bugs. Unit testing's purpose is to ensure that each as-built module behaves according to its specification defined during detailed design.

Integration Testing ---- Interconnects sets of previously tested modules to ensure that the sets behave as well as they did as independently tested modules. Integration testing's purpose is to ensure that each as-built component behaves according to its specification defined during preliminary design.

Performance Instruction:

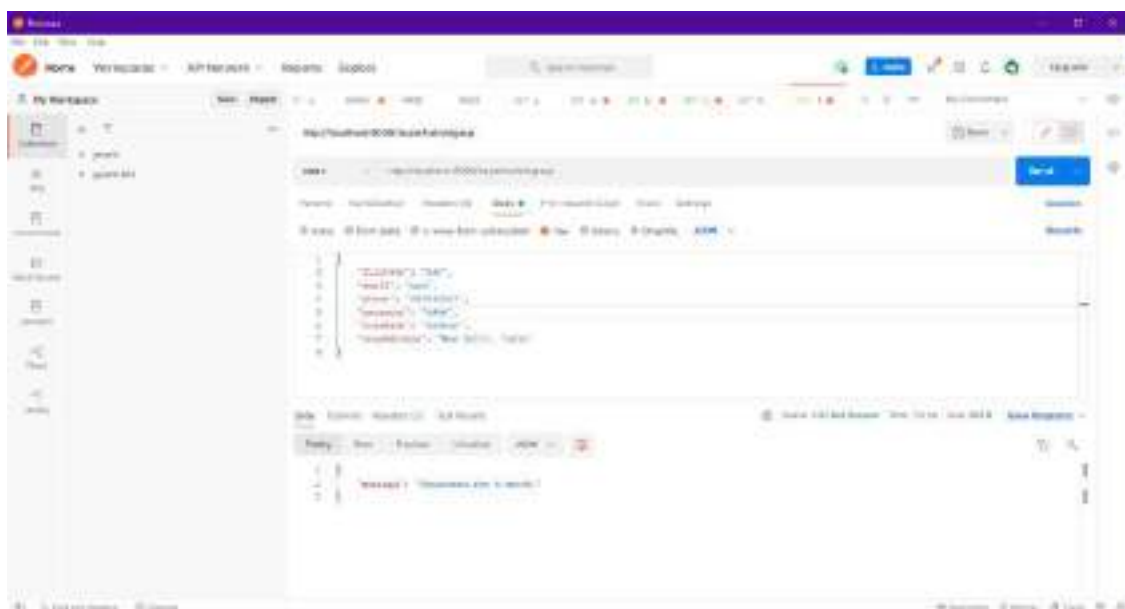
Unit test planning—

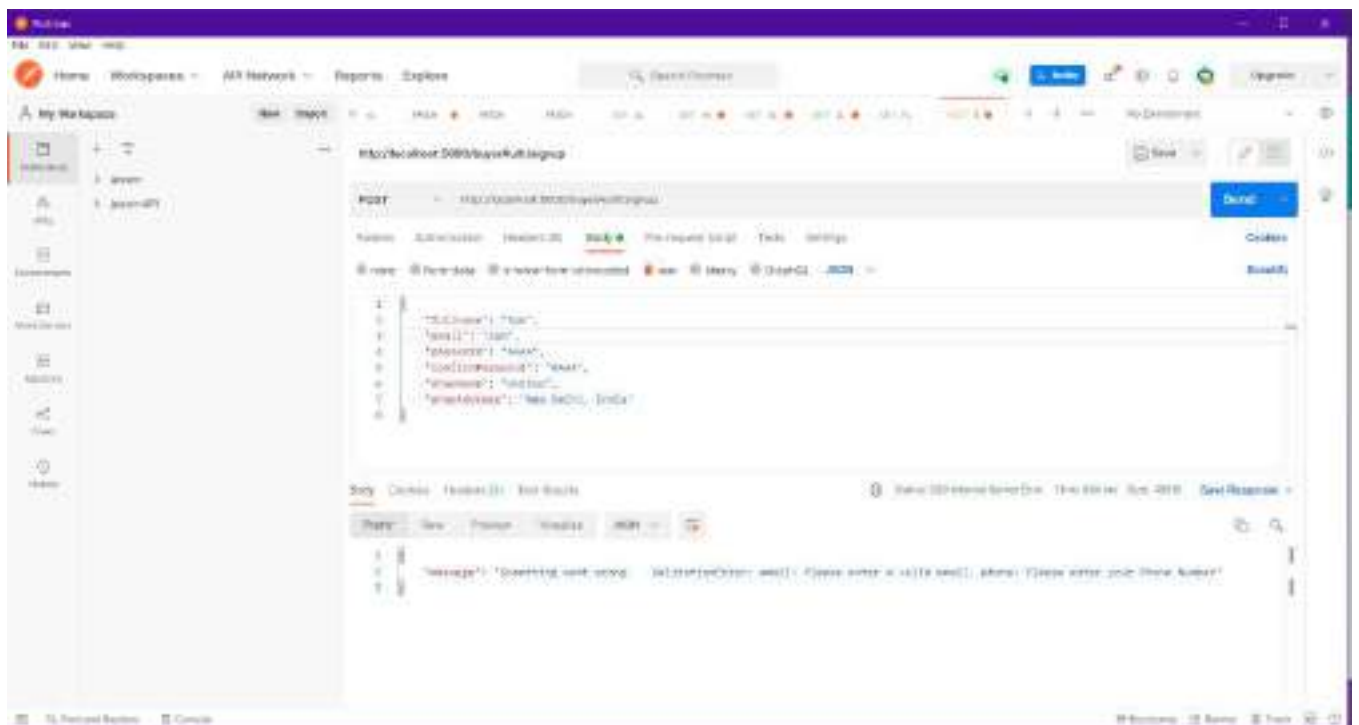
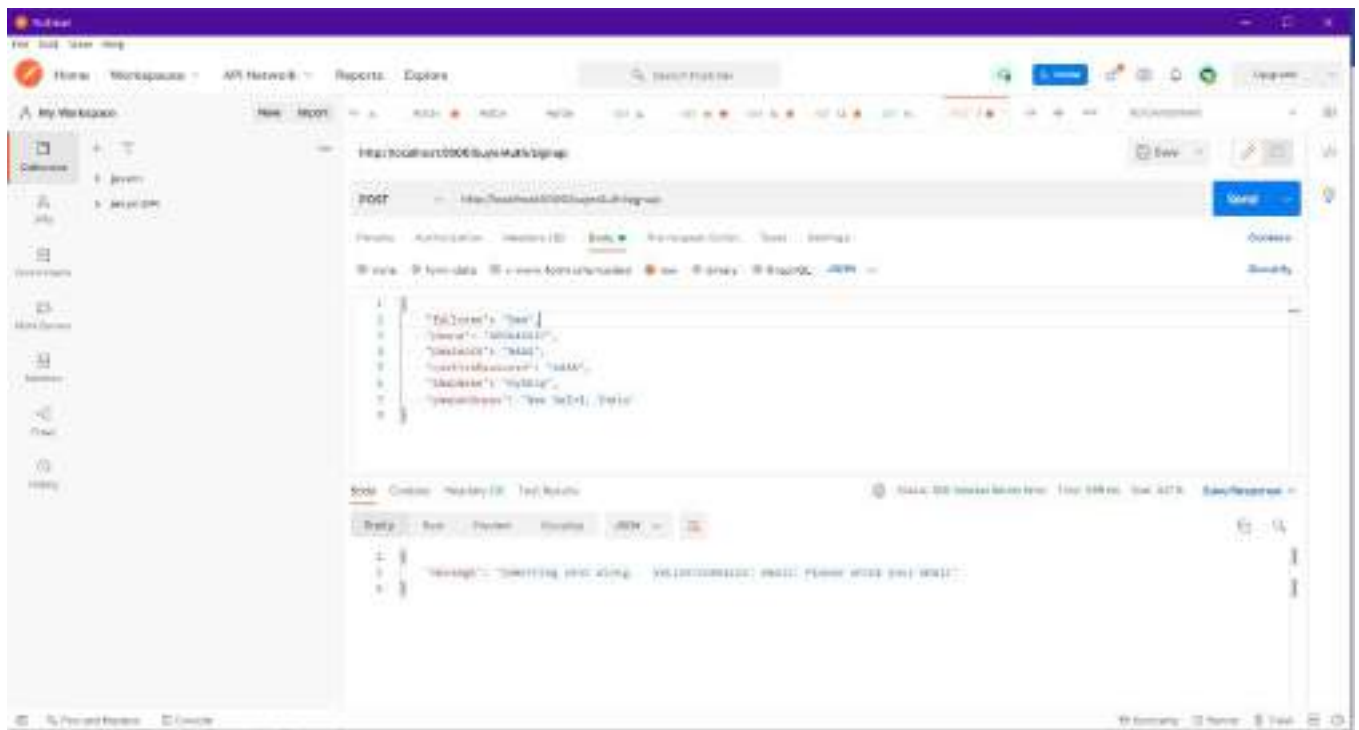
Generate plans and procedures to test each module independently and thoroughly.

Generate plans and procedures to effect orderly system integration.

Backend validations: - Authentication module -> (same for buyer, seller and deliverer)

The screenshot shows the Azure portal interface for configuring an App Service. The 'App' tab is active, showing the application name 'MyClouded330Winged43Group'. Below the name, there are tabs for 'Properties', 'Authentication', 'Monitor', 'Log', 'Deployment Center', 'Tools', and 'Settings'. The 'Log' tab is selected, displaying a list of logs with columns for 'Time', 'Level', 'Message', and 'Source'. The logs show a successful GET request to the application, returning a JSON response with fields like 'name', 'age', 'gender', 'address', and 'email'. The 'Source' tab is also visible, showing the deployment history.





Not a valid email or phone

“Validation errors”

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:5000/buyerAuth/signup`
- Method:** `POST`
- Body (JSON):**

```
{  "full_name": "Sam",  "email": "sam",  "phone": "907845555",  "password": "AAAA",  "confirm_password": "AAAA",  "shopname": "myShop",  "shopAddress": "New Delhi, India"}
```
- Status:** `500 Internal Server Error`
- Response Body (JSON):**

```
{  "message": "Something went wrong. validationError: email: Please enter a valid email"}
```

When everything is valid user get's signed up

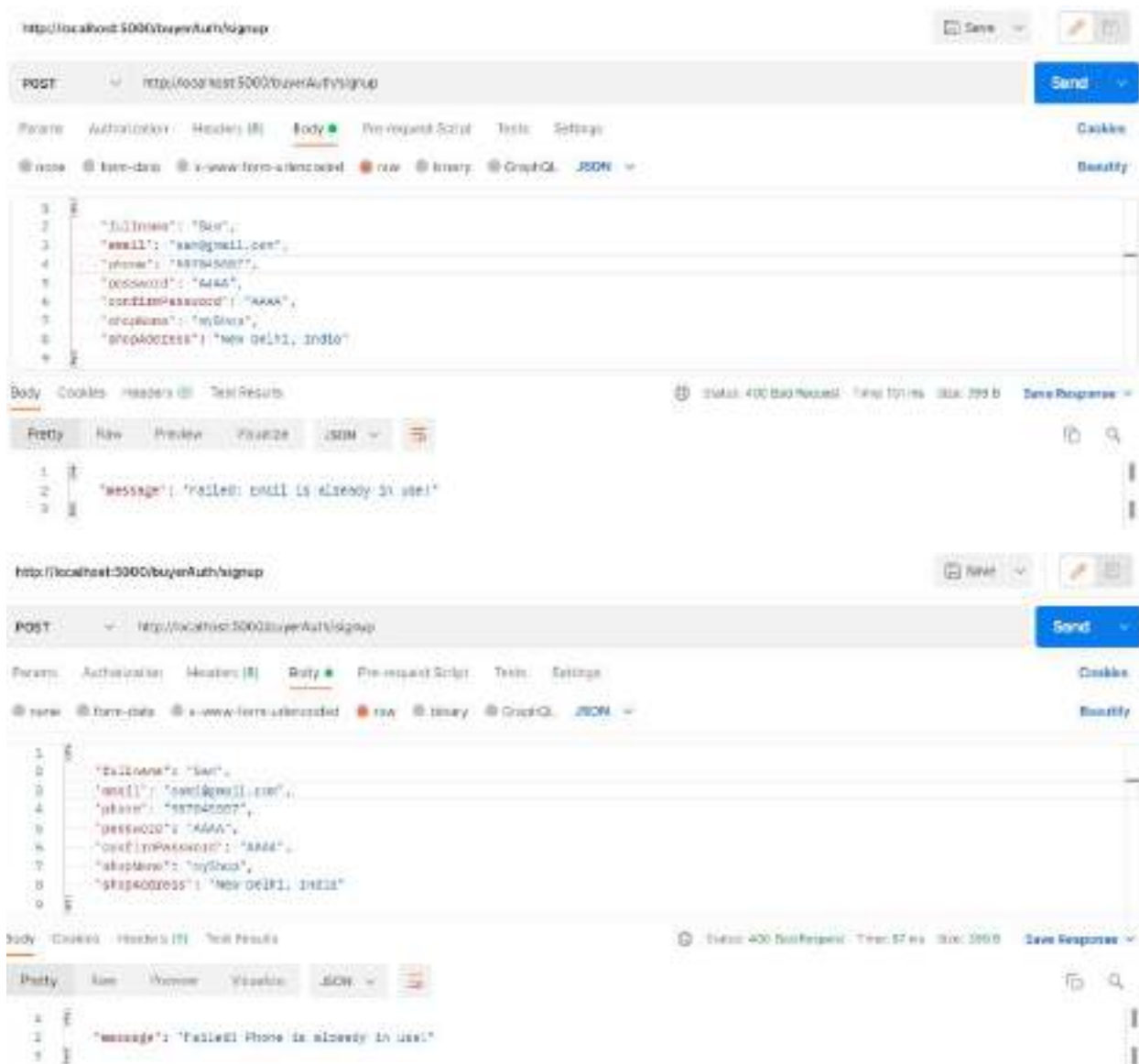
The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:5000/buyerAuth/signup`
- Method:** `POST`
- Body (JSON):**

```
{  "full_name": "Sam",  "email": "sam@gmail.com",  "phone": "907845555",  "password": "AAAA",  "confirm_password": "AAAA",  "shopname": "myShop",  "shopAddress": "New Delhi, India"}
```
- Status:** `200 OK`
- Response Body (JSON):**

```
{  "result": {    "role": "buyer",    "id": "51a563d49f6dc229fc3794d9",    "email": "sam@gmail.com",    "password": "42a8328p4eEP9j1uQwV5comp42ex0sFFVb18p56d3188kth6CyWk8t0C",    "full_name": "Sam",    "phone": "907845555",    "shopname": "myShop",    "shopAddress": "New Delhi, India",    "createdAt": 0  }  }
```

If current email and phone number already in use!



Password doesn't match error



The screenshot shows a REST client interface with the following details:

- URL:** `https://swapi.dev/api/people/1/`
- Method:** GET
- Status:** 200 OK
- Time:** 0.74 s
- Size:** 3.6 KB
- Response:**

```
{
  "id": 1,
  "name": "Luke Skywalker",
  "height": 172,
  "mass": 77,
  "hair_color": "blond",
  "skin_color": "fair",
  "eye_color": "blue",
  "birth_year": "1983-04-19",
  "gender": "male",
  "homeworld": "Tatooine",
  "species": "Human",
  "vehicles": [
    "X-wing",
    "Millennium Falcon"
  ],
  "ships": [
    "X-wing",
    "Millennium Falcon"
  ],
  "friends": [
    "https://swapi.dev/api/people/2/",
    "https://swapi.dev/api/people/3/",
    "https://swapi.dev/api/people/4/",
    "https://swapi.dev/api/people/5/",
    "https://swapi.dev/api/people/6/",
    "https://swapi.dev/api/people/7/",
    "https://swapi.dev/api/people/8/",
    "https://swapi.dev/api/people/9/",
    "https://swapi.dev/api/people/10/"
  ],
  "created": "2014-12-18T21:30:33.600Z",
  "updated": "2014-12-18T21:30:33.600Z"
}
```

Welcome Buyer

ABC

Enter Your Full Name

abc

Enter Your Email

Please include an '@' in the email address. 'abc' is missing an '@'.

12345

Enter Your Phone Number

XYZ Shop

Enter Your Trade Name

xyz road, ABC complex

Enter Your Shop Address

Create Your Password

Confirm Your New Password

Submit

Create Your Password

Confirm Your Full Password

password not matching

SIGN UP

Full Name

Enter Your Full Name

Enter valid name

Email

Enter Your Email

Enter valid email

Phone Number

Enter Your Phone Number

Enter valid Phone number

abc shop

Enter Your Shop Name

xyz road, abc mall

Enter Your Shop Address

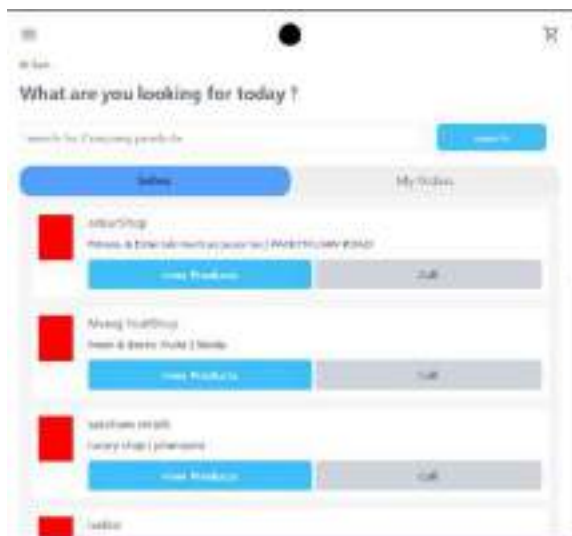
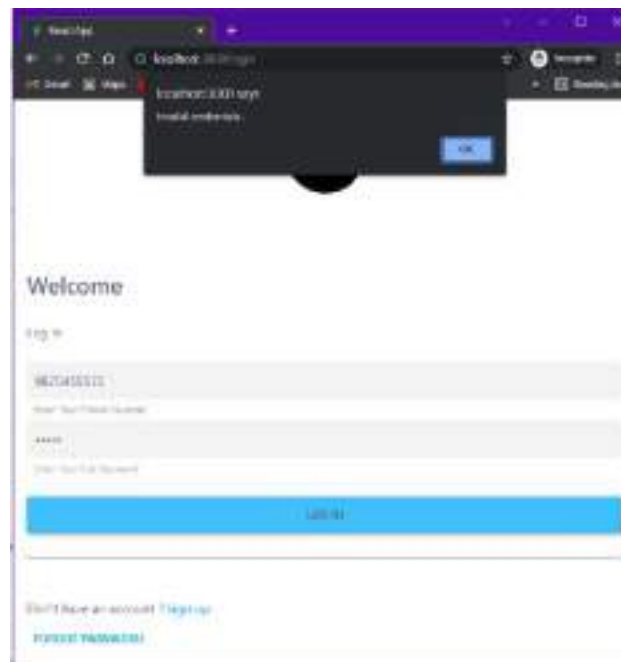
Create Your Password

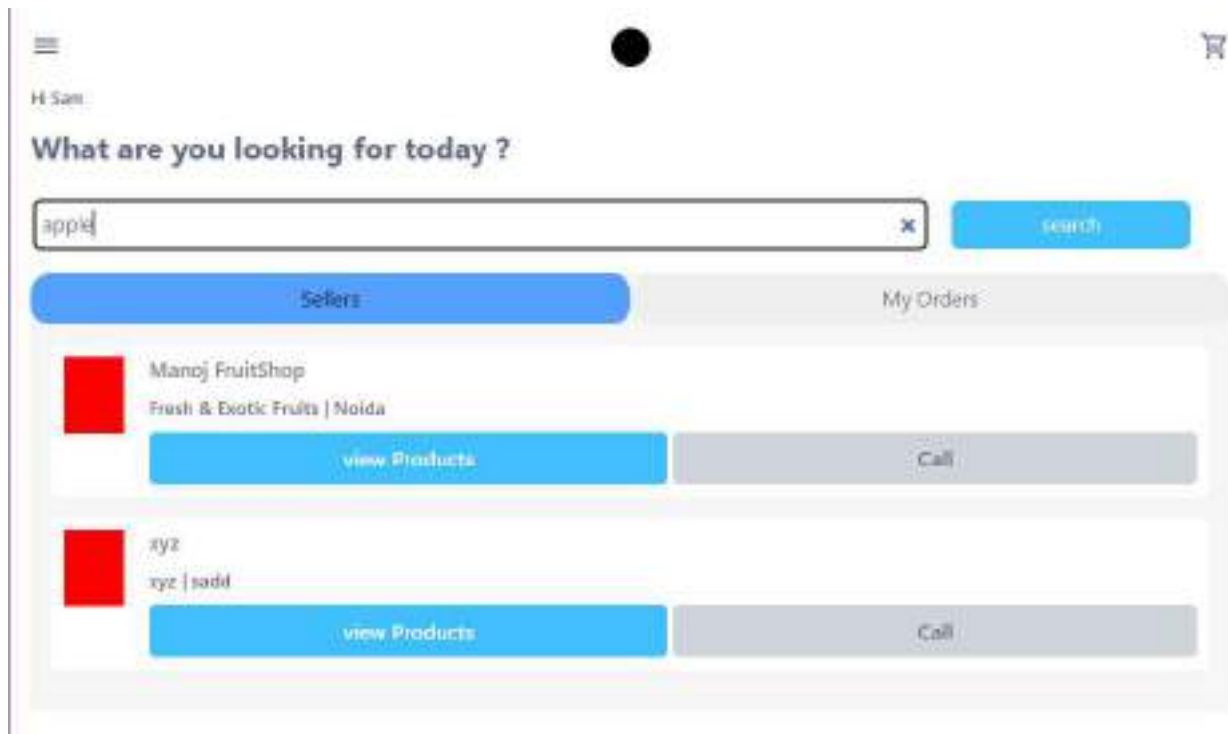
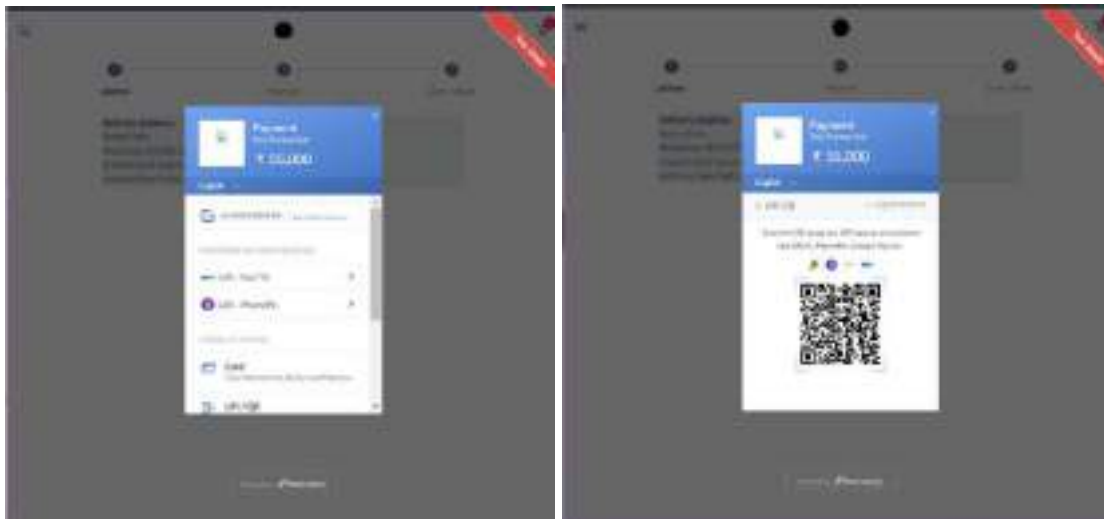
Confirm Your Full Password

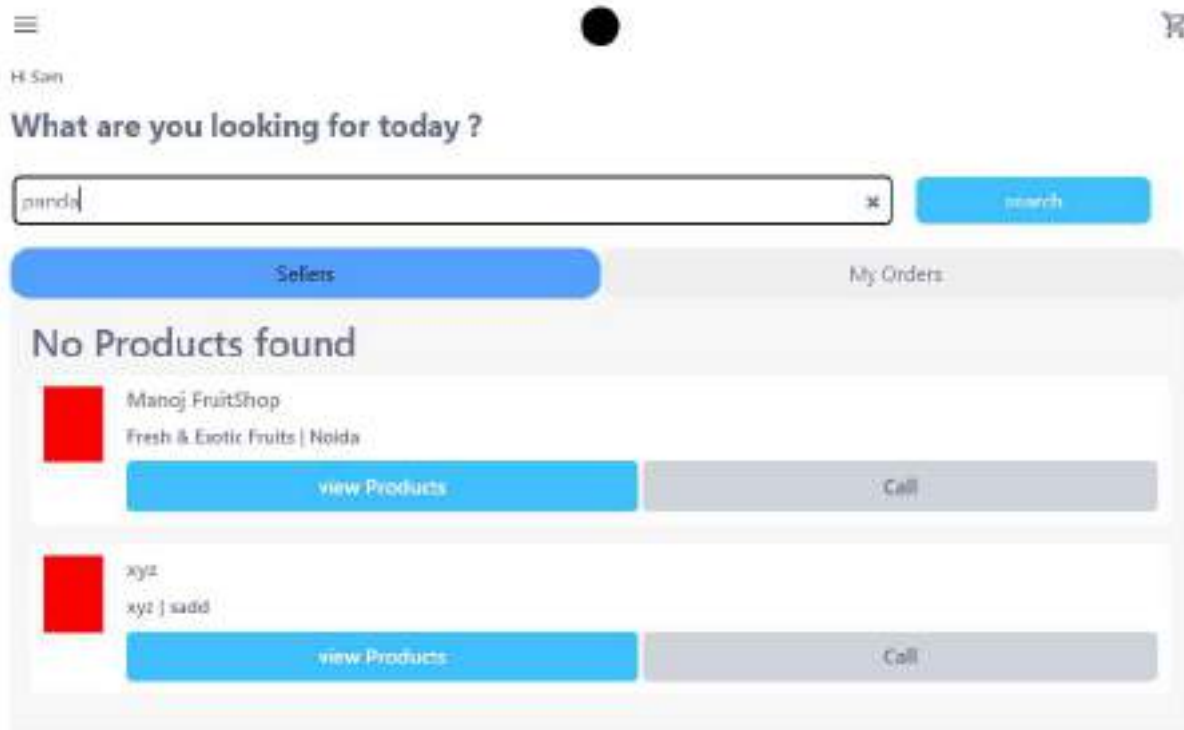
password not matching

SIGN UP

Invalid Credentials







Signup is valid if:

1. Email is valid.
2. Phone is valid phone.
3. Password and confirm Password fields match.
4. Email used is not used by any other customer to signup.
5. Phone is not used by any other customer to signup.
6. Fields are not empty.

Signup is invalid if:

1. Email is invalid.
2. Phone is invalid.
3. Password and confirm password fields do not match.
4. Email already exists in the DB.
5. Phone already exists in the DB.
6. Any of the fields are empty.

Boundary Value Test Cases

T C	Full Name	Email	Phone	Shop Name	Shop Address	Password	Confirm Password	Role	Expected Output	Actual Output	Status
1	Abc	abc@test.com	999999 99999	Abc shop	ABC Mall, Delhi	ABC@a #123	ABC@a #123	Selle r	Seller Signedup	Seller Signed up	True
2	XYZ	xyz@gmail.com	123456 7890	XYZ Shop	XYZ road, JJ colony	AAAA	AAAA	Selle r	Seller Signedup	Seller Signed up	True
3	Ree ha	reeha@google.com	000000 0000	Hom e	Sector – 16, rohini	AAAA@ 1233	AAAAA @1233	Buy er	Buyer Signed up	Buyer Signed up	True
4	Sab a	saba@ceo.in	121212 1212	Hom e	House no – 122, ...	saba@1 23	saba@1 23	Buy er	Buyer Signed up	Buyer Signed up	True
5	Que en	queen@elizabeth.com	111111 1111	Hous e	Villa no. 12, Delhi...	AAAA	AAAA	Deli vere r	Deliverer Signedup	Deliver er Signed up	True
6	ABC shar ma	Abcs@yahoo.com	100000 0000	ABC baker s	Ambie nce Mall, ...	ABC@a #123	ABC@a #123	Selle r	Seller Signedup	Seller Signed up	True
7	KBC	kbc@rediff.com	987045 5572	Hom e	Golde n colony ...	kbcccc cccc@	kbcccc cccc@	Buy er	Buyer Signed up	Buyer Signed up	True
8	Amit abh Bac han	Amitabh@cases.com	995893 7319	Hom e	Appt – 19, delhi...	AAAA	AAAA	Buy er	Buyer Signed up	Buyer Signed up	True
9	My Nam e	khan@outlook.com	123123 1231	Hous e	House – 10, 3 rd floor, Delhi...	1234567 89	1234567 89	Deli vere r	Deliverer Signedup	Deliver er Signed up	True

***Phone no. should be valid as OTP is sent for the verification purpose

***Phone no. should be valid as OTP is sent for the verification purpose

As OTP is sent to verify the same

Robust Test Cases

TC	Full Name	Email	Phone	Shop Name	Shop Address	Password	Confirm Password	Expected Output	Actual Output	Status
1.	xyz	abc	9870455572	Abc_shop	X	AAAA	AAAA	Inavlid	Inavlid	True
2.								Invalid	Invalid	True
3.	xyz	abc@gmail.com	9870455572	a	X	AAAA	AA	Invalid	Invalid	True
4.	xyz	abc@gmail.com	9870455	abc	X	AAAA	AA	Invalid	Invalid	True
5.	Abc	abc@gmail.com	99999999999	Abc shop	ABC Mall, Delhi	ABC@a#123	ABC@a#123	Valid	Valid	True
6.	Xyz	abc@gmail.com	1111111111	Xyz shop	Xyz road	AA@123	AA@123	Invalid	Invalid	True
7.	Abc	Abc1@gmail.com	99999999999	Abc shop	ABC Mall, Delhi	ABC@a#123	ABC@a#123	Invalid	Invalid	True
8.	Abc	Abc1@gmail.com	98704555572	Abc shop	ABC Mall, Delhi	ABC@a#123	ABC@a#123	Valid	Valid	True
9.	XYZ	abc@	00000000000	xyz	xyz	AAAA	AAAA	Invalid	Invalid	True
10.	XYZ	abc@test.com	00000000000	xyz	xyz	AAAA	AAAA	Valid	Valid	True
11.	abc	Abc@test.com	XXXX	123	123	saba	saba	Invalid	Invalid	True
12.	abc	Abc@test.com	9313305723	My shop	Karol bagh, shop no. 12	abc@123@abc\$	abc@123@abc\$	Valid	Valid	True
13.	reeha	reeha@google.com	9899234567	Pacifi c	Rohini, delhi	123abc@omg	123abc@omg	Valid	Valid	True

Conclusion:

Test Cases for given code was made successfully.

VIVA QUESTIONS

exp-10

Ques 1. Explain the role of testing in software development.
Ans Software testing comes into play at different times in different software development methodologies.

→ Waterfall

→ Agile

Testing modular functions and testing help us find bugs quicker, easy to make changes which is part and parcel of agile software development methodology and reduces the risk of errors/faults.

Ques 2. How much testing is sufficient? Is it possible to do exhaustive testing of the software?

Ans It is impossible to exhaustively test software or prove absence of errors, no matter how specific your test strategy is.

Exhaustive testing doesn't ensure no errors or have discovered it all so there is no accurate measure to say how much testing is best but it should cover major and trivial factors.

Ques 3. Why developers shouldn't test the software they wrote?

Ans Developer make poor testers. Here are the reasons why:

→ They try to test code to make sure that it works, rather than testing all the way in which it doesn't work.

→ Since they wrote it themselves, developers tend to be very optimistic about the software and don't have the correct attitude needed for testing: to break software.

Ques 4. What is functional testing?

Ans Functional testing is a form of black-box testing. As the name suggests, it focuses on software's functional requirements rather than its internal implementation. A functional requirement refers to required behaviour in the system, in terms of its I/O.

Ques 5. What is non-functional testing?

Ans It's tests for non-functional requirements, which refers to an attribute or quality of the system explicitly requested by the client. These include - security, scalability and usability.

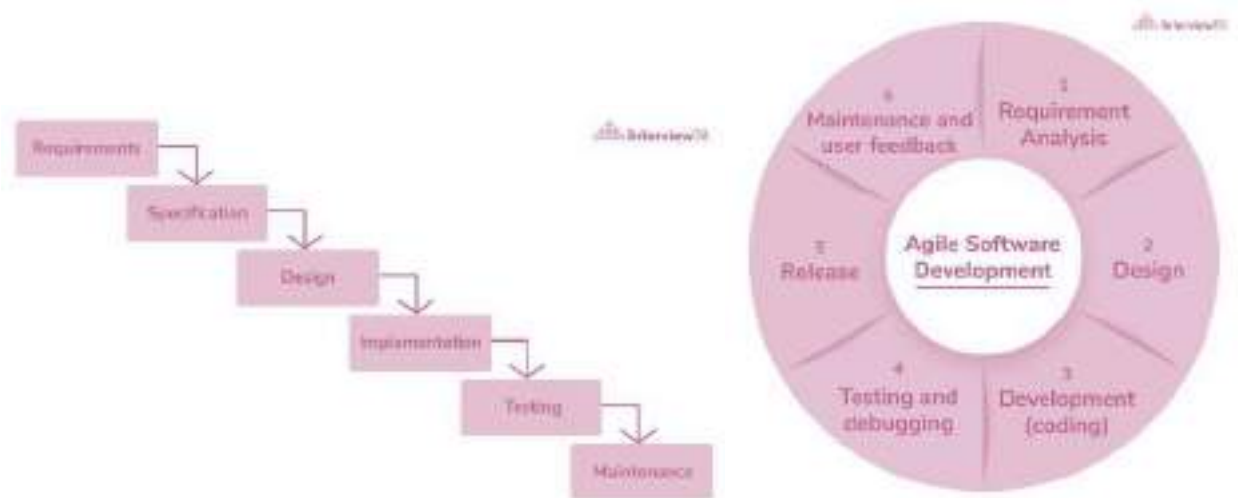
Viva Questions

1. Explain the role of testing in software development?

Ans.

Software testing comes into play at different times in different software development methodologies. There are two main methodologies in software development, namely Waterfall and Agile.

In a traditional waterfall software development model, requirements are gathered first. Then a specification document is created based on the document, which drives the design and development of the software. Finally, the testers conduct the testing at the end of the software development life cycle once the complete software system is built.



Waterfall Software Development Model

An agile software development model works in small iterations. You test the software in parallel as it is getting built. The developers build a small functionality according to the requirements. The testers test it and get customer feedback, which drives future development.

2. How much testing is sufficient? Or, is it possible to do exhaustive testing of the software?

Ans.

It is impossible to exhaustively test software or prove the absence of errors, no matter how specific your test strategy is.

An extensive test that finds hundreds of errors doesn't imply that it has discovered them all. There could be many more errors that the test might have missed. The absence of errors doesn't mean there are no errors, and the software is perfect. It could easily mean ineffective or incomplete tests. To prove that a program works, you'd have to test all possible inputs and their combinations.

Consider a simple program that takes a string as an input that is ten characters long. To test it with each possible input, you'd have to enter 2610 names, which is impossible. Since exhaustive testing is not practical, your best strategy as a tester is to pick the test cases that are most likely to find errors. Testing is sufficient when you have enough confidence to release the software and assume it will work as expected.

3. Why developers shouldn't test the software they wrote?

Ans.

Developers make poor testers. Here are some reasons why:

- They try to test the code to make sure that it works, rather than testing all the ways in which it doesn't work.
- Since they wrote it themselves, developers tend to be very optimistic about the software and don't have the correct attitude needed for testing: to break software.
- Developers skip the more sophisticated tests that an experienced tester would perform to break the software. They follow the happy path to execute the code from start to finish with proper inputs, often not enough to get the confidence to ship software in production.

However, it doesn't mean that developers shouldn't test the software before sending it to the tester. Developer testing helps find many bugs that are caused by programming errors. These are hard to find for a tester because they don't always have access to the source code.

4. What is functional testing?

Ans.

Functional testing is a form of black-box testing. As the name suggests, it focuses on the software's functional requirements rather than its internal implementation. A functional requirement refers to required behavior in the system, in terms of its input and output.

It validates the software against the functional requirements or the specification, ignoring the non-functional attributes such as performance, usability, and reliability.

Functional testing aims to answer the following questions, in particular:

- Does the software fulfill its functional requirements?
- Does it solve its intended users' problems?

5. What is non-functional testing?

Ans.

Non-functional testing tests the system's non-functional requirements, which refer to an attribute or quality of the system explicitly requested by the client. These include performance, security, scalability, and usability.

Non-functional testing comes after functional testing. It tests the general characteristics unrelated to the functional requirements of the software. Non-functional testing ensures that the software is secure, scalable, high-performance, and won't crash under heavy load.



EXPERIMENT - 11

Software Engineering Lab

Aim

To Perform Estimation of effort using FP Estimation for chosen system.

Syeda Reeha Quasar

14114802719

4C7

EXPERIMENT – 11

Aim:

To Perform Estimation of effort using FP Estimation for chosen system.

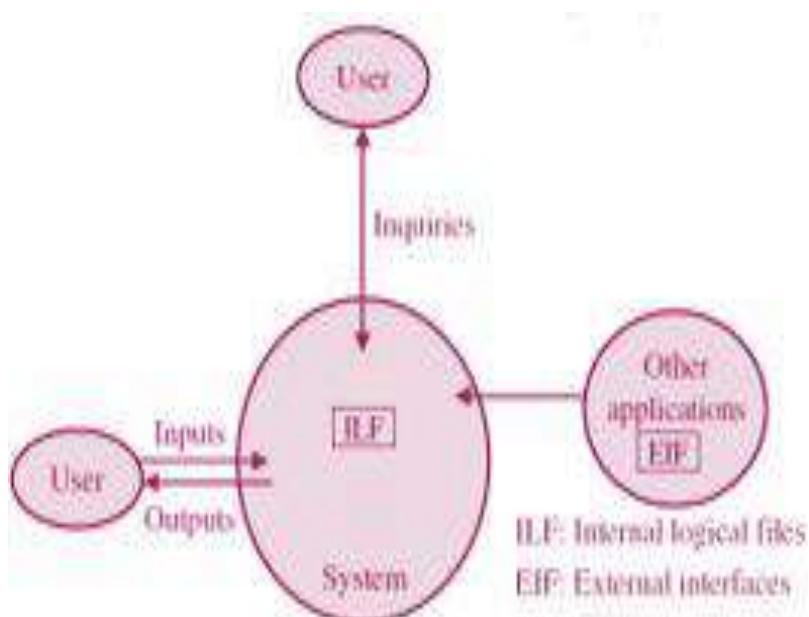
Theory:

A **function point** is a "unit of measurement" to express the amount of business functionality an information system (as a product) provides to a user. Function points are used to compute a functional size measurement (FSM) of software.

The principle of Albrecht's function point analysis (FPA) is that a system is decomposed into functional units.

1. Inputs: information entering the system
2. Outputs: information leaving the system
3. Enquiries: requests for instant access to information
4. Internal logical files: information held within the system
5. External interface files: information held by other system that is used by the system being analyzed.

The FPA functional units are shown in figure given below:



The five functional units are divided in two categories:

(i) Data function types

- Internal Logical Files (ILF): A user identifiable group of logical related data or control information maintained within the system.
- External Interface files (EIF): A user identifiable group of logically related data or control information referenced by the system, but maintained within another system. This means that EIF counted for one system, may be an ILF in another system.

(ii) Transactional function types

- External Input (EI): An EI processes data or control information that comes from outside the system. The EI is an elementary process, which is the smallest unit of activity that is meaningful to the end user in the business.
- External Output (EO): An EO is an elementary process that generate data or control information to be sent outside the system.
- External Inquiry (EQ): An EQ is an elementary process that is made up to an input-output combination that results in data retrieval.

Special features

- Function point approach is independent of the language, tools, or methodologies used for implementation; i.e. they do not take into consideration programming languages, data base management systems, processing hardware or any other data base technology.
- Function points can be estimated from requirement specification or design specification, thus making it possible to estimate development efforts in early phases of development
- Function points are directly linked to the statement of requirements; any change of requirements can easily be followed by a re-estimate.
- Function points are based on the system user's external view of the system, non-technical users of the software system have a better understanding of what function points are measuring.

Functional Units	Weighting factors		
	Low	Average	High
External Inputs (EI)	3	4	6
External Output (EO)	4	5	7
External Inquiries (EQ)	3	4	6
External logical files (ILF)	7	10	15
External Interface files (EIF)	5	7	10

Performance Instruction:

- 1.Observe functional units and their weighting factors.
- 2.Compute them in formula to find value of UFP count.
- 3.Find value of FP by using formula.

Output:

Consider a project with the following parameters.

(i) External Inputs:

- (a) 10 with low complexity
- (b) 15 with average complexity
- (c) 17 with high complexity

(ii) External Outputs:

- (a) 6 with low complexity
- (b) 13 with high complexity

(iii) External Inquiries:

- (a) 3 with low complexity
- (b) 4 with average complexity
- (c) 2 with high complexity

(iv) Internal logical files:

- (a) 2 with average complexity
- (b) 1 with high complexity

(v) External Interface files:

- (a) 9 with low complexity

In addition to above, system requires

- i. Significant data communication
- ii. Performance is very critical
- iii. Designed code may be moderately reusable
- iv. System is not designed for multiple installation in different organizations.

Other complexity adjustment factors are treated as average.
Compute the function points for the project.

The functional complexities are multiplied with the corresponding weights against each function, and the values are added up to determine the UFP (Unadjusted Function Point) of the subsystem.

Files

External Inputs Els	Eternal Outputs EOs	External Inquiries EQs	External Logic files ELFs	External Interface Files EIFs
<ul style="list-style-type: none"> • Seller Signup • Deliverer Signup • Buyer Signup • Seller Login • Deliverer Login • Buyer Login • Seller signs deliverer • Seller assign Deliverer • Deliverer updates order status • Place order • Add products • Add to cart • Add address • Customize profile • Update stocks • Give review • Comment on review • Update product • Remove deliverer • checkout 	<ul style="list-style-type: none"> • Get products • Get orders • Get deliverers • Get reviews • Get cart • Get order details • Get buyer details • Get profile • Get sellers • Search products • Search shops • Get status • Track order • Get my stock • Get my deliverers • Get assigned deliverer • Get history • get contact info • get product details • get product reviews • get my reviews • get shop reviews • get seller details • get deliverer contact • get payment details • get seller orders 	<ul style="list-style-type: none"> • Login • Payment • Payment receipt • Payment verification • Login token • Wrong password • Customer not found • Seller not found • Buyer not found • Deliverer not found • Signed up • Update profile • Place order • Check out • Cancel order • Add stock • Update stock • Assign deliverer • Add deliverer • No products • New review • Order status • Reset credentials • Login expire 	<ul style="list-style-type: none"> • Sellers • Buyers • Deliverers • Products • Reviews • Cart • Orders 	<ul style="list-style-type: none"> • Products • Orders • Reviews • Deliverer

Unadjusted function points may be counted using below table:

Functional Units	Count	Complexity	Complexity Totals	Functional Unit Totals
External Inputs (EIs)	20	Low * 3 Average * 4 High * 6	20 * 4	80
External Outputs (EOs)	26	Low * 4 Average * 5 High * 7	26 * 5	130
External Inquiries (EQs)	24	Low * 3 Average * 4 High * 6	24 * 4	96
External logic Files (ILFs)	7	Low * 7 Average * 10 High * 15	7 * 10	70
External Interface Files (EIFs)	4	Low * 5 Average * 7 High * 10	4 * 7	28
Total Unadjusted Function Point Count = 404				

$$\begin{aligned}\Sigma F_i &= 14 * \text{scale} \\ &= 14 * 3 = 42\end{aligned}$$

$$\begin{aligned}\text{CAF} &= (0.65 + 0.01 * \Sigma F_i) \\ &= (0.65 + 0.01 * 42) \\ &= 1.07\end{aligned}$$

$$\begin{aligned}\text{FP} &= \text{UFP} * \text{CAF} \\ &= 404 * 1.07 \\ &= 432.28000000000003\end{aligned}$$

Hence FP = 432

Conclusion:

FP estimation was done successfully.

QUESTIONS

cap 11

Ques 1. Explain 5 functional units of FPA (functional Point Analysis)?

- No. of External Inputs (EI) → Input screens and tables
- No. of External Outputs (EO) → Output screens and reports
- No. of External Inquiries (EQ) → Prompts and interrupts
- No. of External files (ILF) → Databases and directories
- No. of External Interfaces (EIF) → Shared DB and shared routines

Ques 2. Explain special features of FPA?

1. FPs of an application is found out by counting the number and type of function used in applications.
2. FP characterizes the complexity of software system and hence can used to depict the project time and manpower requirements.
3. The effort used to develop project depends on what software dev.
4. FP is language independent.
5. FP is used for data processing systems like information systems.

Ques 3. Explain weighing factors (3) of functional units.

It's the weight given to data points to assign lighter or heavier importance in a group. It usually used for calculating a weighted mean, to give less (or more) importance to group members.

	Low	Medium	High
EI	3	4	6
EO	4	5	7
EQ	3	4	15
ILF	5	10	10
EIF	7	7	

Ques 4. Explain term unadjusted function point count (UFP)?

For each function there is weight associated. The sum of weights quantifies the size of information processing and is referred to as UFP (Unadjusted Function Points).

Ques 5. What are uses of Function Point (FP)?

- The FP is a "unit of measurement" to express amount of business functionality an information system (as a product).
- FP is used to compute FSM (functional size management).
- The cost (in dollars or hours) of single unit from past projects.

Viva Questions

1. Explain five functional units of functional point analysis(FPA)?

Ans.

Measurements Parameters	Examples
Number of External Inputs(EI)	Input screen and tables
Number of External Output (EO)	Output screens and reports
Number of external inquiries (EQ)	Prompts and interrupts.
Number of internal files (ILF)	Databases and directories
Number of external interfaces (EIF)	Shared databases and shared routines.

2. Explain special features of FPA?

Ans.

1. FPs of an application is found out by counting the number and types of functions used in the applications.
2. FP characterizes the complexity of the software system and hence can be used to depict the project time and the manpower requirement.
3. The effort required to develop the project depends on what the software does.
4. FP is programming language independent.
5. FP method is used for data processing systems, business systems like information systems.
6. The five parameters mentioned above are also known as information domain characteristics.
7. All the parameters mentioned above are assigned some weights that have been experimentally determined.

3. Explain three weighting factors of functional units?

Ans.

A weighting factor is a weight given to a data point to assign it a lighter, or heavier, importance in a group. It is usually used for calculating a weighted mean, to give less (or more) importance to group members.

There are majorly 3 weighing Factors –

Low, High and Average and their value varies from different parameters.

Parameters	Counts	Complexity		
		<i>Low</i>	<i>Medium</i>	<i>High</i>
Number of Inputs		3	4	6
Number of Outputs		4	5	7
Number of Files		3	4	6
Number of External Interfaces		5	10	15
Number of User Inquiries		7	7	10

4. Explain term unadjusted function point count (UFP)?

Ans.

For each function identified above the function is further classified as simple, average or complex and a weight is given to each. The sum of the weights quantifies the size of information processing and is referred to as the Unadjusted Function points.

5. What are uses of function point?

Ans.

The function point is a "unit of measurement" to express the amount of business functionality an information system (as a product) provides to a user. Function

points are used to compute a functional size measurement (FSM) of software. The cost (in dollars or hours) of a single unit is calculated from past projects.

Function points are a unit of measure used to define the value that the end user derives, or the functional business requirements the software is designed to accomplish. Each application has a specific number of function points, which are used to benchmark cost and productivity or development and maintenance activity.