Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Continuous Assessment for Laboratory / Assignment sessions

Academic Year 2022-23

Name: Ayush Jain SAP ID: 60004200132

Course: Software Engineering Course Code: **DJ19CEC601**

Year: **T.Y. B.Tech.** Sem: **VI** Batch: B2

Department: Computer Engineering

Performance Indicators (Any no. of Indicators) (Maximum 5 marks per indicator)	1	2	3	4	5	6	7	8	9	10	11	Σ	Avg	A1	A2	Σ	Avg
Course Outcome	1	2	2	2	3	2	5	4	4	6							
1. Knowledge (Factual/Conceptual/Procedural/ Metacognitive)	5	5	4	4	4	4	4	5	4	5							
2. Describe (Factual/Conceptual/Procedural/ Metacognitive)	4	-	4	4	4	4	4	4	4	4							
3. Demonstration (Factual/Conceptual/Procedural/ Metacognitive)	-	-	4	-	-	-	-	-	-	4							
4. Strategy (Analyse & / or Evaluate) (Factual/Conceptual/ Procedural/Metacognitive)	4	5	-	4	4	4	4	4	-	-	-						
5. Interpret/ Develop (Factual/Conceptual/ Procedural/Metacognitive)	-	4	4	3	4	4	4	4	4	-							
6. Attitude towards learning (receiving, attending, responding, valuing, organizing, characterization by value)	4	4	4	4	4	4	-	-	4	5							
7. Non-verbal communication skills/ Behaviour or Behavioural skills (motor skills, hand-eye coordination, gross body movements, finely coordinated body movements speech behaviours)	5	5	-	-	-	-	5	5	4	5							
Total	22	23	20	19	20	20	21	22	20	23							
Signature of the faculty member																	

Outstanding (5), Excellent (4), Good (3), Fair (2), Needs Improvement (1)

Laboratory marks Σ Avg. =	Assignment marks Σ Avg. =	Total Term-work (25) =
Laboratory Scaled to (15) =	Assignment Scaled to (10) =	Sign of the Student:

Signature of the Faculty member:
Name of the Faculty member:

Signature of Head of the Department
Date:

Scanned with CamScanner

Experiment - 1

Date of Performance: 27/02/2023 **Date of Submission:** 27/02/2023

DIV: B **Batch**: B3

Team Members:

Name:	**SAP ID:**
Dhruv Bheda	60004200102
Sahej Jain	60004200111
Ayush Jain	60004200132
Varun Vekaria	60004200167

Aim: To identify a suitable life cycle model for your case study and justify your choice

ABSTRACT:

What is the product:

Amazon is a leading global e-commerce platform that provides a wide range of products and services to customers worldwide.

Need: Amazon arose from the growing demand for a convenient and reliable online marketplace where customers could purchase products from the comfort of their homes. Amazon's primary users are online shoppers who are looking for a user-friendly and secure way to purchase products. Amazon's platform provides a personalized shopping experience, with recommendations based on previous purchases and browsing history.

How it begun:

Amazon was founded by Jeff Bezos in 1994 as an online bookstore. Bezos had previously worked on Wall Street and was looking for a business opportunity that could take advantage of the growing popularity of the internet. He recognized that the internet could be used to sell books, which were easy to ship and had a broad customer base.

Lean Model: The Lean model is another option for small ecommerce businesses, as it emphasizes a streamlined development process and continuous improvement. With Lean, the focus is on delivering value to the customer as quickly and efficiently as possible, while minimizing waste and maximizing resources. This model is ideal for small businesses with limited budgets and time constraints, as it encourages experimentation and learning through feedback from customers and data analysis

Features:
Customers can easily search and browse products, read reviews, and make purchases, all in one place. Amazon also serves as a platform for third-party sellers to sell their products, providing a global reach to a wide audience of potential customers.Amazon's software engineering process has been critical to the success of its platform. The company has developed a robust software system that handles a large volume of traffic and transactions, providing a high-quality user experience to customers and sellers alike.The software engineering process involves various stages, including requirement gathering, designing, coding, testing, deployment, and maintenance, which has allowed Amazon to continuously improve its platform and expand its offerings. Amazon's software engineering process has been instrumental in the success of its platform, allowing the company to continuously improve and expand its offerings.

Process Model: Agile model, Incremental model, V model
Our Choice: Agile Model

JUSTIFICATION:
Why the Incremental model?
The Incremental model is an iterative software development model in which the software product is developed and delivered incrementally, in multiple stages or iterations. Each iteration typically builds on the previous one, adding new features or functionality, until the final product is delivered. The Incremental model can be divided into several phases or modules, with each module representing a complete functionality of the software. The modules are developed and tested

separately, and once all modules have been integrated, the complete software product is tested and delivered. This approach allows for greater flexibility and adaptability to changing requirements, as well as improved feedback from users throughout the development process.

Why the V model?
The V-Model is a variation of the Waterfall model that emphasizes testing and quality assurance at every stage of the development process. Given the importance of security and reliability in ecommerce, this model could be a good fit for Amazon's software development process. By rigorously testing each component of the platform, from individual modules to the overall system, Amazon can ensure that its e-commerce offerings are secure and reliable for its millions of users.

Why the Agile model?
The Agile model is well-suited for projects with rapidly evolving requirements, which is likely the case for an ecommerce giant like Amazon. Agile is iterative and incremental, which means that new features can be added and tested in short cycles, allowing Amazon to stay ahead of the curve and quickly adapt to changes in the market. This model also emphasizes customer collaboration, which is important for an ecommerce platform that relies on customer feedback to improve its offerings.The Agile life cycle model is a flexible and iterative approach to software development that emphasizes collaboration between cross-functional teams and the ability to quickly respond to change. Amazon's e-commerce platform is built using an Agile approach to ensure continuous delivery of high-quality software.

The Agile life cycle model for Amazon e-commerce typically involves the following phases:

- **Planning**: In this phase, the product owner, stakeholders, and development team collaborate to create a product backlog, which is a prioritized list of features to be developed.
- **Development**: This phase involves iterative development sprints, typically lasting two to four weeks, in which the team works on a subset of the product backlog. The team creates a potentially shippable increment of the software at the end of each sprint.

- **Testing**: The team performs testing throughout the development phase to ensure that the software meets the acceptance criteria for each feature. The testing includes unit testing, integration testing, and acceptance testing.
- **Deployment**: Once a sprint is complete, the team deploys the potentially shippable increment to a staging environment for further testing and validation.
- **Release**: When the product owner approves the potentially shippable increment, the team releases it to production.
- **Monitoring and Feedback:** The team monitors the production environment to detect and fix any issues that may arise. The team also gathers feedback from users to identify areas for improvement.

CONCLUSION:

We learnt about the different types of process models and justified the most suitable life cycle model for project execution in our case study.

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

Academic Year: 2023

Experiment No. 2

Aim: To develop Software Requirement Specification (SRS) document in IEEE format for the project.

Software Requirements Specification

for

ShopIt

Version 1.0

Prepared by

Sahej Jain	**60004200111**
Varun Vekaria	**60004200167**
Ayush Jain	**60004200132**
Dhruv Bheda	**60004200102**

Date: 02/03/2023

Contents

Revisions			
Version	**Primary Author(s)**	**Description of Version**	**Date Completed**
1.0	Sahej Jain, Varun Vekaria, Dhruv Bheda, Ayush Jain	A one stop shop ecommerce website for all your needs. Products ranging from essentials to luxury items, it is truly a product worthy of its name ‘ShopIt’.	09/03/23

1 Introduction

1.1 Document Purpose

This is a SRS Document for the project ShopIt. The goal of this document on software requirements specifications is to describe in detail the technical components of the product we plan to develop. It describes how the application's initial iteration will function. Along with the users, equipment, programmes, and other connecting applications, it outlines the specifications, technical details, and project constraints that will assist us make sure we make the greatest use of all the available resources, meet all needs for the product, and offer our users the finest possible product.

1.2 Product Scope

- The project scope is not limited to a specific city or a country but it has an extensive scope at a global level ShopIt will provide customers with a user-friendly platform to browse and purchase products online, while also providing businesses with a powerful tool to manage and fulfil those orders.
- The website will include a product catalogue with detailed information and images, a shopping cart and checkout process that is intuitive and secure, and a customer account management system that allows customers to track orders and manage their personal information.
- The website will also include features such as search functionality, product recommendations, and personalized promotions to enhance the customer experience and increase sales.
- The website will be accessible from desktop and mobile devices and will integrate with third-party systems such as payment gateways and shipping providers.

1.3 Intended Audience and Document Overview

The document includes a brief abstract of the product ShopIt. It is set up in a continuous progression. It first describes the features and present the issue statement followed by the summary of the project's users, surroundings, and constraints follows. The document continues by describing the various interfaces and project requirements. References to relevant research articles and other sources are also included in the paper. This is followed by an Overall description of the product and specific requirements and use cases. It concludes with functional, non-functional and all the other types of requirements.

The paper is meant for all users as well as the technical developers of the project. This may include the UI/UX designers, development team, Business analysts, Quality Assurance team, the stakeholders and the different types of end users. It also involves

the database admin along with other supporting staff members and the future possible clients.

1.4 Definitions, Acronyms and Abbreviations

1. **GUI** - Graphical User Interface.
2. **RAM** - Random Access Memory
3. **API** - Application Programming Interface
4. **GPS** - Global Positioning System

1.5 Document Conventions

The following conventions were followed while creating the document:

● We have used the IEEE standards for document formatting.

● The font used is Arial, font size for title is 14 and font size for text is 12.

● Italics have been used for comments.

● 1” margin has been maintained throughout the document.

● The text is single spaced.

1.6 References and Acknowledgments

[1] Prof. Betty H.C. Cheng, “Homework 2”, CSE 435, East Lansing, MI, September 2007.
[2] Prof. Betty H.C. Cheng, “Requirements Assignment”, CSE 435, East Lansing, MI, September 2007.
[3] Mr. Borzoo Bonakdarpour, Elicitation Meeting, September 25th 2007.
[4] IEEE-SA Standards Board, “IEEE Recommended Practice for Software Requirements Specifications”, Software Engineering Standards Committee of the IEEE Computer Society, June 25th 1998

2 Overall Description

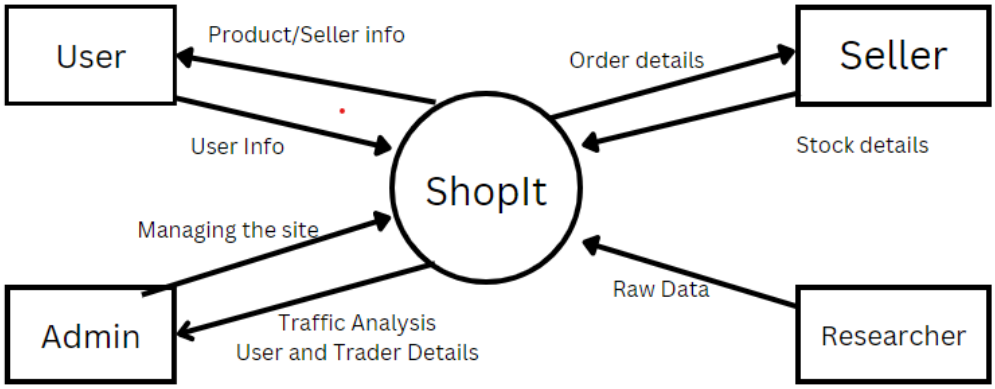
2.1 Product Perspective

A platform that connects buyers and sellers, providing a robust product menu, search and navigation, detailed product pages, a seamless checkout process, and strong customer support. It enables transactions and facilitates the exchange of goods and services, allowing users to easily find and purchase products while providing sellers with tools for managing and fulfilling orders.

2.2 Product Functionality

1. Ecommerce website with user friendly UI/UX.
2. Complete details about product and the seller.
3. Assurance of legit sellers verified by company.
4. Regular updating category wise product.

DFD:



2.3 Users and Characteristics:

1. **Casual shoppers:** These users may use ShopIt occasionally to purchase products for personal use or as gifts. They may not be highly experienced with online shopping or have in-depth technical knowledge.
2. **Power shoppers:** These users may use ShopIt frequently to purchase a variety of products, ranging from groceries to electronics. They may have a high level of technical expertise and are comfortable using advanced features such as Alexa voice commands, subscriptions, and one-click ordering.
3. **ShopIt Prime members**: These users pay an annual fee to access exclusive benefits such as free two-day shipping, streaming of movies, TV shows and music, and early access to deals. They may be frequent shoppers who value convenience and speed.
4. **ShopIt Business users:** These are businesses that use ShopIt to purchase products and supplies for their operations. They may have specific requirements such as bulk ordering, tax-exempt purchases, and customized invoicing.
5. **Third-party sellers:** These users are independent sellers who use ShopIt's platform to sell their products to ShopIt customers. They may have different levels of experience and technical expertise, ranging from small-scale hobbyists to large-scale professional sellers.
6. **Educators and students:** These users may use ShopIt to purchase textbooks, educational resources, and other materials. They may have specific requirements such as discounted pricing, access to digital content, and support for academic research.

2.4 Operating Environment

- **Recommended browsers:** Chrome, Firefox, Safari, Edge and Brave.

- **Recommended Operating systems:** Windows, MacOS, IpadOS, iOS, wear OS, watchOS, Android and Linux.

2.5 Design and Implementation Constraints

The system is limited by its operating server in terms of the maximum number of users and queries it can support at a given time.

- Isn't compatible with devices without GPS.
- Network Connectivity issues
- Requires Large RAM.

2.6 User Documentation

The user manual will contain all the guidelines for handling software as well as FAQ section for reference. Academic Year: 2023

- Contact us & support centres.
- Cultural differences.

2.7 Assumptions and Dependencies

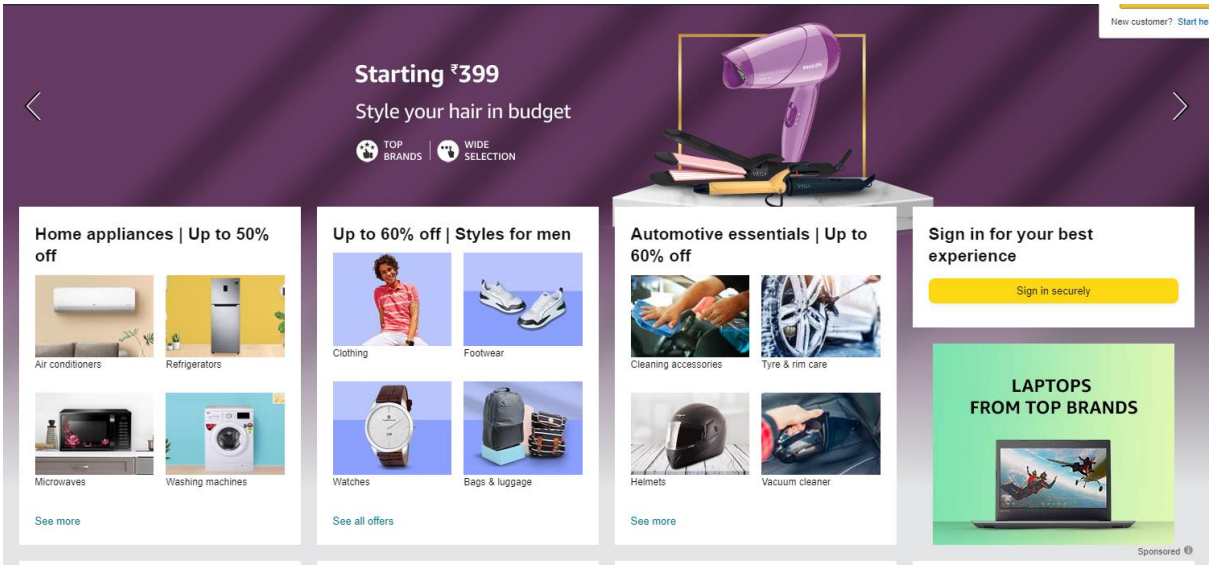
- For ShopIt, there are several assumed factors that could affect the requirements stated in the SRS. These include third-party or commercial components, development or operating environment issues, constraints, changes in technology, and dependencies on external factors such as software components reused from another project.
- Assumptions about the reliability, compatibility, and availability of these factors could impact the project's success, and changes or discrepancies in assumptions may lead to unexpected delays, additional costs, or changes in project direction. Therefore, it is important to identify and communicate assumptions clearly to ensure that they are validated and any necessary adjustments are made in a timely manner.
- If ShopIt assumes that they can reuse a certain piece of software from another project, but it turns out that the software is not compatible with the current project, this could cause delays or issues with the project.

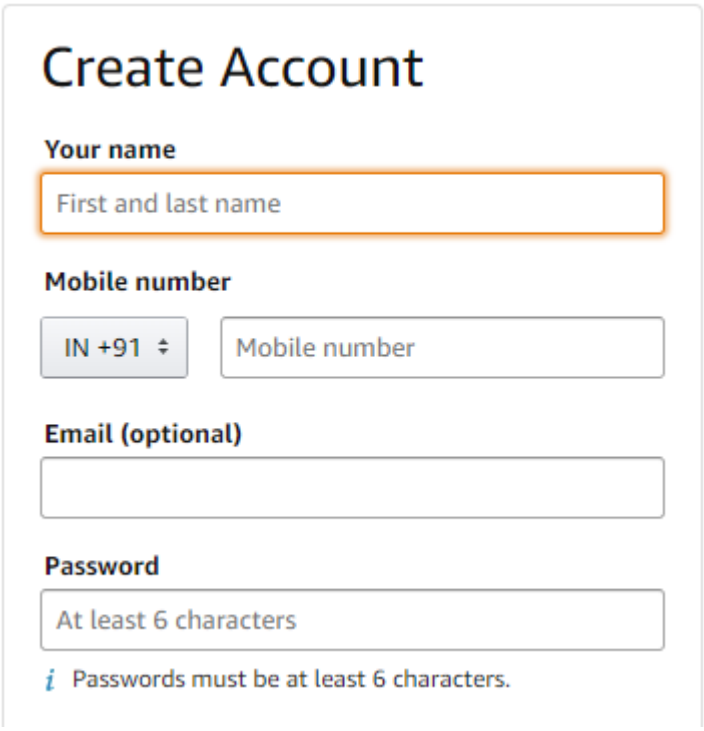
3 Specific Requirements

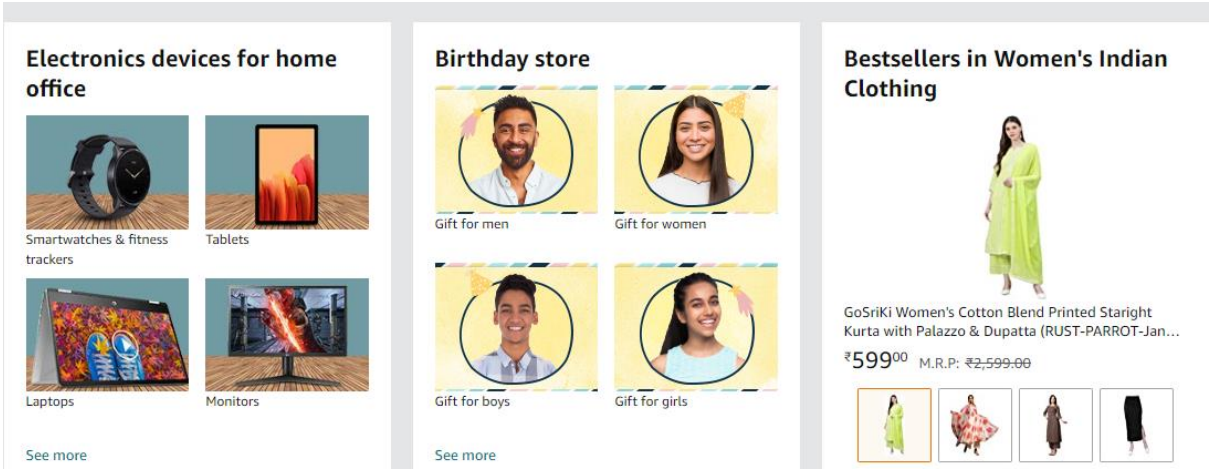
3.1 External Interface Requirements

3.1.1 User Interfaces

Landing Page: This will be the first page the users will see upon opening the webapp. It will give an introduction about our organization and will show the current trends.







3.1.2 Hardware Interfaces

Server:

RAM: 8 GB

Storage**:** 2TB SSD

Processor: Intel Pentium 4 processor or later that's SSE2 capable

GPU: NVidia GTX 1050

User Device:

RAM: 500 MB

Storage: 4 GB Storage

GPS Sensor

3.1.3 Software Interfaces

Browsers: Chrome, Firefox, Safari, Edge and Brave.

Operating systems: Windows, MacOS, IpadOS, iOS, wear OS, watchOS, Android and Linux. Tools: Google Colab, Jupyter Notebook

3.1.4 Communications Interfaces

1. Minimum 40 Kbps Internet Speed to ensure lossless connectivity.
2. HTTP protocols for servicing the requests and for transmission of data in JSON format.
3. AES protocol will be used to encrypt the sensitive data being transmitted.

3.2 Functional Requirements

1. **Product search:** The system should allow customers to search for products based on keywords, categories, or other criteria. The search results should be relevant and provide customers with the ability to filter and sort the results as needed.

2. **Product recommendations:** The system should be able to recommend products to customers based on their browsing and purchase history, as well as other factors such as popularity and ratings.

3. **Shopping cart:** The system should allow customers to add items to their shopping cart, modify the quantities, and remove items as needed. The system should also calculate the total price of the items in the shopping cart and apply any applicable discounts or promotions.

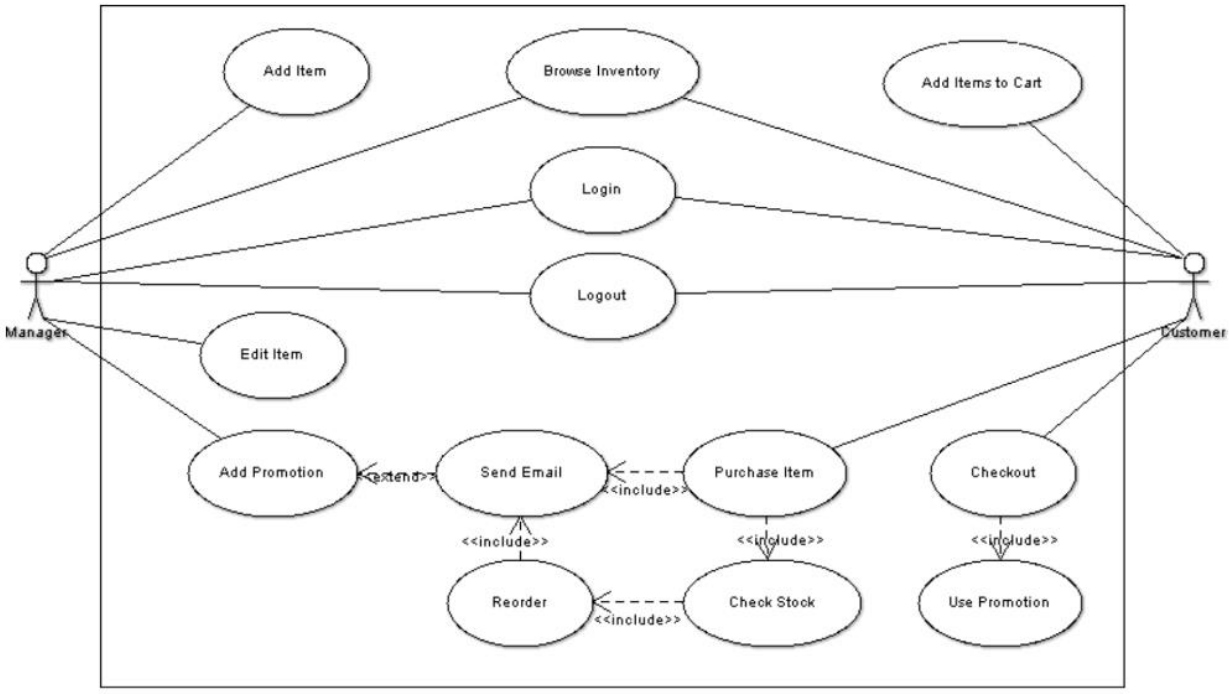
4. **Checkout process:** The system should guide customers through the checkout process, collecting their shipping and billing information, and providing a summary of the order before it is placed. The system should also provide customers with various payment options and confirm the order once it has been placed.

5. **Order tracking:** The system should allow customers to track their orders, view the status, and receive updates on any changes or delays.

6. **Customer service:** The system should provide customers with the ability to contact customer service for assistance with their orders, returns, or other issues. The system should also provide a knowledge base or FAQ section to help customers find answers to their questions.

3.3 Behaviour Requirements

3.3.1 Use Case View



4 Other Non-functional Requirements

4.1 Performance Requirements

1.If the transaction fails due to any reason, Amount deducted should be reversed to the Source account.

2.Inventory will be updated real-time, to avoid ordering of out-of-stock items.

3.The system should support concurrent users.

4.Customer support for solving any grievances of the customer.

4.2 Safety and Security Requirements

- System will use secured database. The safety, security, and privacy requirements for an ShopIt product are crucial to ensure user safety, protect against data breaches or unauthorized access, and comply with applicable regulations.
- Safety requirements may include compliance with safety regulations and designing the product to minimize risks of injury or harm.
- Security requirements may include appropriate user authentication measures and protection against cyber-attacks or data breaches.
- Privacy requirements may include compliance with privacy regulations and protecting the privacy of users' personal information. User identity authentication requirements may include unique usernames and passwords or limiting the number of login attempts.
- Safety certifications may also be necessary, and the product should be designed and tested to meet these requirements, with appropriate documentation provided to demonstrate compliance

4.3 Software Quality Attributes

1. Reliability: The ShopIt product should be designed to perform consistently and without errors or failures. Reliability can be measured by calculating the mean time between failures (MTBF) or the mean time to repair (MTTR), with a goal of achieving high MTBF and low MTTR values.

2.Portability: The ShopIt product should be designed to run on multiple platforms and devices, with minimal modifications required for each platform. Portability can be

measured by testing the product on different platforms and devices and documenting any modifications required.

3. Maintainability: The ShopIt product should be designed to be easy to maintain and update, with clear documentation and modular code. Maintainability can be measured by tracking the time and effort required to make updates or fixes.

4. Scalability: The ShopIt product should be designed to handle increasing amounts of data and users without significant performance degradation. Scalability can be measured by testing the product with increasing loads of data and users and documenting any performance issues.

5.Security: The ShopIt product should be designed to protect against unauthorized access, use, or disclosure of data. Security can be measured by conducting security audits or penetration testing to identify vulnerabilities and ensuring that appropriate safeguards are in place.

5 Other Requirements

- Maintenance of the application.
- Updating new functionalities.
- Security of database.
- Frequent updation of data by admin.

Appendix A – Data Dictionary

Field Name	Data Type	Description	Example
User_name	Text	Name of each user	Ayush Jain
User_address	Text	Address of each user	Borivali, Mumbai
Product_details	List	Product_name, product_type, cost	Iphone 12, smartphones, 85000/-
Seller_details	List	Seller_name, seller_address	Yogesh, Panvel

Appendix B - Group Log

Date	Actors	Work Done
28/02/2023	Sahej, Dhruv, Ayush, Varun	Anaylze requirement
06/03/2023	Sahej, Dhruv, Ayush, Varun	Prepared SRS

Experiment No. 3

Dhruv Bheda - 60004200102

Sahej Jain - 60004200111

Ayush Jain - 60004200132

Varun Vekaria - 60004200167

Aim: Identify scenarios & develop UML Use case and Class Diagram for the project.

Theory:

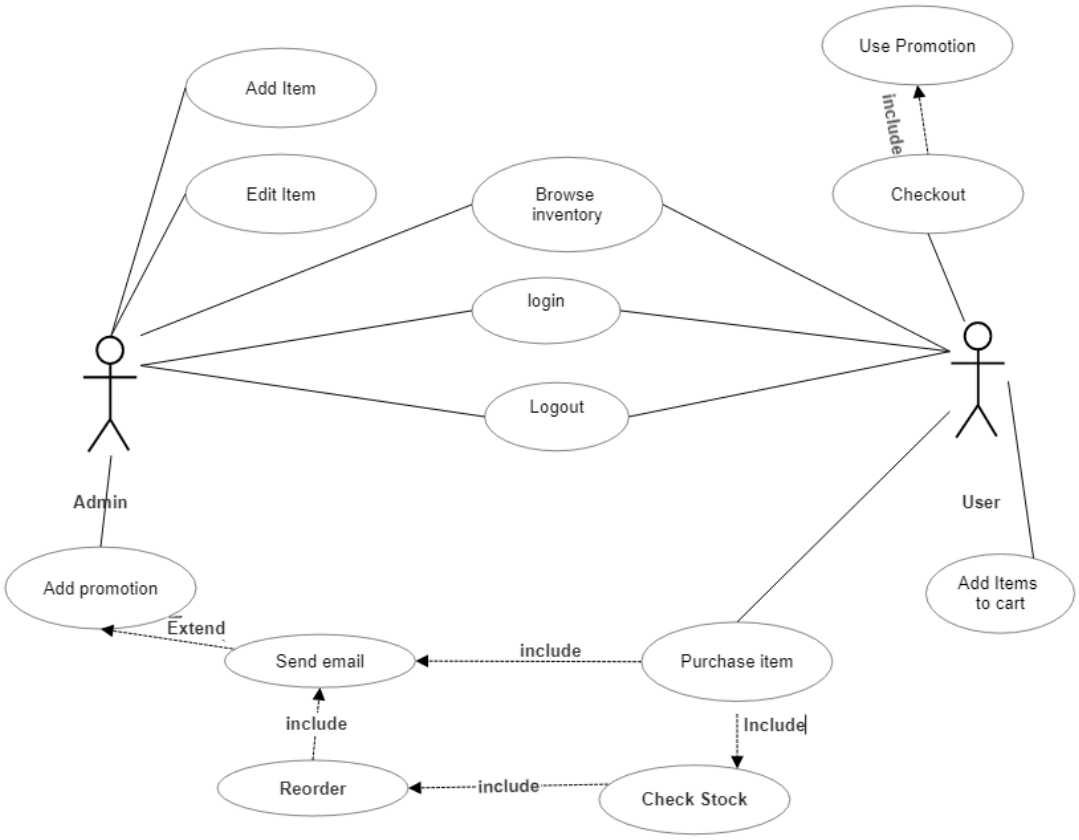
Use Case Diagrams:

In UML, use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

The following topics describe model elements in use-case diagrams:

- **Use-cases**
  A use case describes a function that a system performs to achieve the user's goal.
- **Actors**
  An actor represents a role of a user that interacts with the system that you are modeling.
- **Subsystems**
  In UML models, subsystems are a type of stereotyped component that represent independent, behavioral units in a system.
- **Relationships in use-case diagrams**
  In UML, a relationship is a connection between model elements. A UML

relationship is a type of model element that adds semantics to a model by defining the structure and behavior between the model elements.



Users:

The users will have specific login credentials. After login users will have access to the entire inventory and can search for their desired product, shipping details and other product details. Users can then wish to add a product to cart for instant or future buy. However, before this, there is a check if the stock is available for the desired product. If the user purchases the product a verification email is sent to the user. This email may include promotions.

Admin:

The admin has a specific login ID to perform specific tasks. The admin has responsibilities as to browse the inventory to check if a product is out of stock. The admin can add, edit and delete products. On purchase the admin can send a promotion to the user related to the purchase.

Class Diagram:

In UML, class diagrams are one of six types of structural diagrams. Class diagrams are fundamental to the object modeling process and model the static structure of a system. Depending on the complexity of a system, you can use a single class diagram to model an entire system, or you can use several class diagrams to model the components of a system. Class diagrams are the blueprints of your system or subsystem. You can use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services that they provide.

1. Capture and define the structure of classes and other classifiers
2. Define relationships between classes and classifiers
3. Illustrate the structure of a model by using attributes, operations, and signals
4. Show the common classifier roles and responsibilities that define the behavior of the system
5. Show the implementation classes in a package
6. Show the structure and behavior of one or more classes
7. Show an inheritance hierarchy among classes and classifiers
8. Show the workers and entities as business object models

During the implementation phase of a software development cycle, you can use class diagrams to convert your models into code and to convert your code into models. The following topics describe model elements in class diagrams:

- **Classes**
  In UML, a class represents an object or a set of objects that share a common structure and behavior. Classes, or instances of classes, are common model elements in UML diagrams.
- **Objects**
  In UML models, objects are model elements that represent instances of a class or of classes.
- **Packages**
  Packages group related model elements of all types, including other packages.
- **Signals**
  In UML models, signals are model elements that are independent of the

classifiers that handle them. Signals specify one-way, asynchronous communications between active objects.

- **Enumerations**

  In UML models, enumerations are model elements in class diagrams that represent user-defined data types.

- **Data types**

  In UML diagrams, data types are model elements that define data values.

- **Artifacts**

  In UML models, artifacts are model elements that represent the physical entities in a software system.

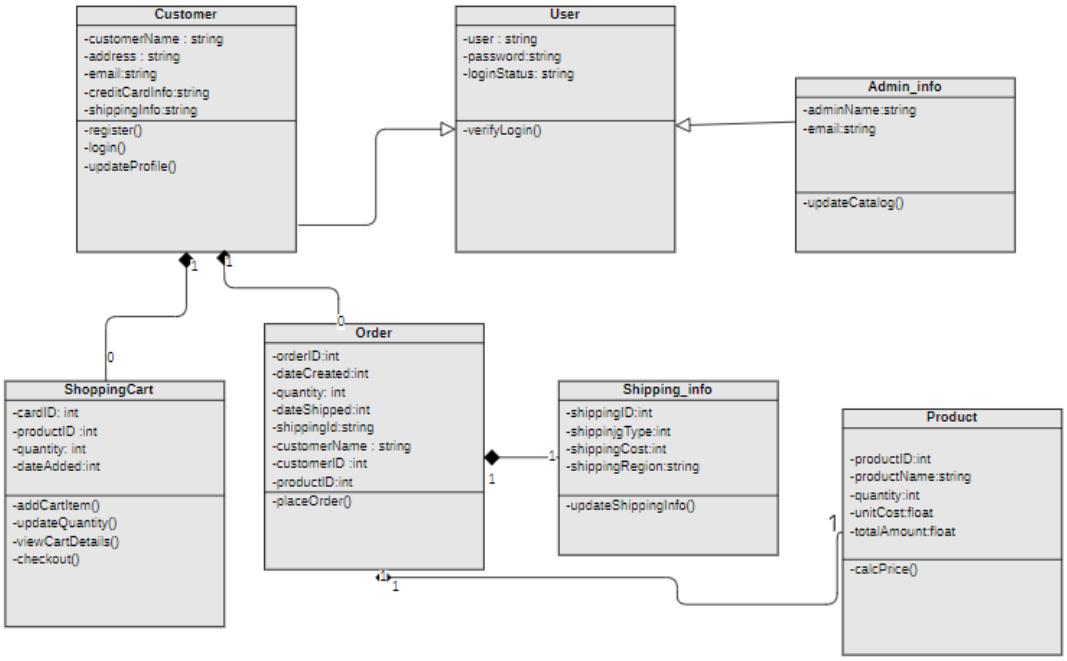
- **Relationships in class diagrams**

  In UML, a relationship is a connection between model elements. A UML relationship is a type of model element that adds semantics to a model by defining the structure and behavior between model elements.

- **Qualifiers on association ends**

  In UML, qualifiers are properties of binary associations and are an optional part of association ends.

CLASS DIAGRAM:



CLASS DIAGRAM DESCRIPTION:

- **Class:** Customer

  Attribute:

  - customerName
  - address
  - email
  - creditCardInfo
  - shippingInfo

  Operations:

  - register()
  - login()
  - updateProfile()

- **Class:** User

  Attribute:

  - userId

- password
- loginStatus

Operations:

- verifyLogin()

- **Class:** Admin Info

  Attribute:

  - adminName
  - email

  Operations:

  - updateCatalog()

- **Class:** Shipping Info

  Attribute:

  - shippingId
  - shippingType
  - shippingCost
  - shippingRegionId

  Operations:

  - updateShippingInfo()

- **Class:** Product

  Attribute:

  - productId
  - productName
  - quantity
  - unitCost
  - subTotal

  Operations:

  - calcPrice()

- **Class:** Orders

  Attribute:

  - orderId
  - productId

- dateCreated
- dateShipped
- customerName
- customerId
- status
- shippingId

Operations:

- placeOrder()

- **Class:** Shopping Cart

  Attribute:

  - cartId
  - productId
  - quantity
  - dateAdded

  Operations:

  - addCartItem()
  - updateQuantity()
  - viewCartDetails()
  - checkOut()

Conclusion: We learnt about UML Use-Case Diagrams and Class Diagrams and have created them for our e-commerce project.

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

SE - Experiment No. 4

Div: B Batch:B3

Team Members:

Dhruv Bheda: 60004200102
Sahej Jain: 60004200111
Ayush Jain: 60004200132
Varun Vekaria: 60004200167

Aim: Develop Activity diagram and DFD (up to 2 levels) for the project.

Theory:

Activity Diagram:

A UML activity diagram depicts the dynamic behavior of a system or part of a system through the flow of control between actions that the system performs. It is similar to a flowchart except that an activity diagram can show concurrent flows. The main component of an activity diagram is an action node, represented by a rounded rectangle, which corresponds to a task performed by the software system. Arrows from one action node to another indicate the flow of control. That is, an arrow between two action nodes means that after the first action is complete the second action begins. A solid black dot forms the initial node that indicates the starting point of the activity. A black dot surrounded by a black circle is the final node indicating the end of the activity. A fork represents the separation of activities into two or more concurrent activities. It is drawn as a horizontal black bar with one arrow pointing to it and two or more arrows pointing out from it. Each outgoing arrow represents a flow of control that can be executed concurrently with the flows corresponding to the other outgoing arrows. These concurrent activities can be performed on a computer using different threads or even using different computers.

To indicate how the actions are divided among the participants, one can decorate the activity diagram with swim lanes. Swim lanes are formed by dividing the diagram into strips or “lanes,” each of which corresponds to one of the participants. All actions in one lane are done by the corresponding participant.

Data Flow Diagrams:

The data flow diagram enables you to develop models of the information domain and functional domain. As the DFD is refined into greater levels of detail, you perform an implicit functional decomposition of the system. At the same time, the DFD refinement results in a corresponding refinement of data as it moves through the processes that embody the application.

Guidelines for drawing a data flow diagram:

(1) the level 0 data flow diagram should depict the software/system as a single bubble;

(2) primary input and output should be carefully noted;

(3) refinement should begin by isolating candidate processes, data objects, and data stores to be represented at the next level;

(4) all arrows and bubbles should be labeled with meaningful names;

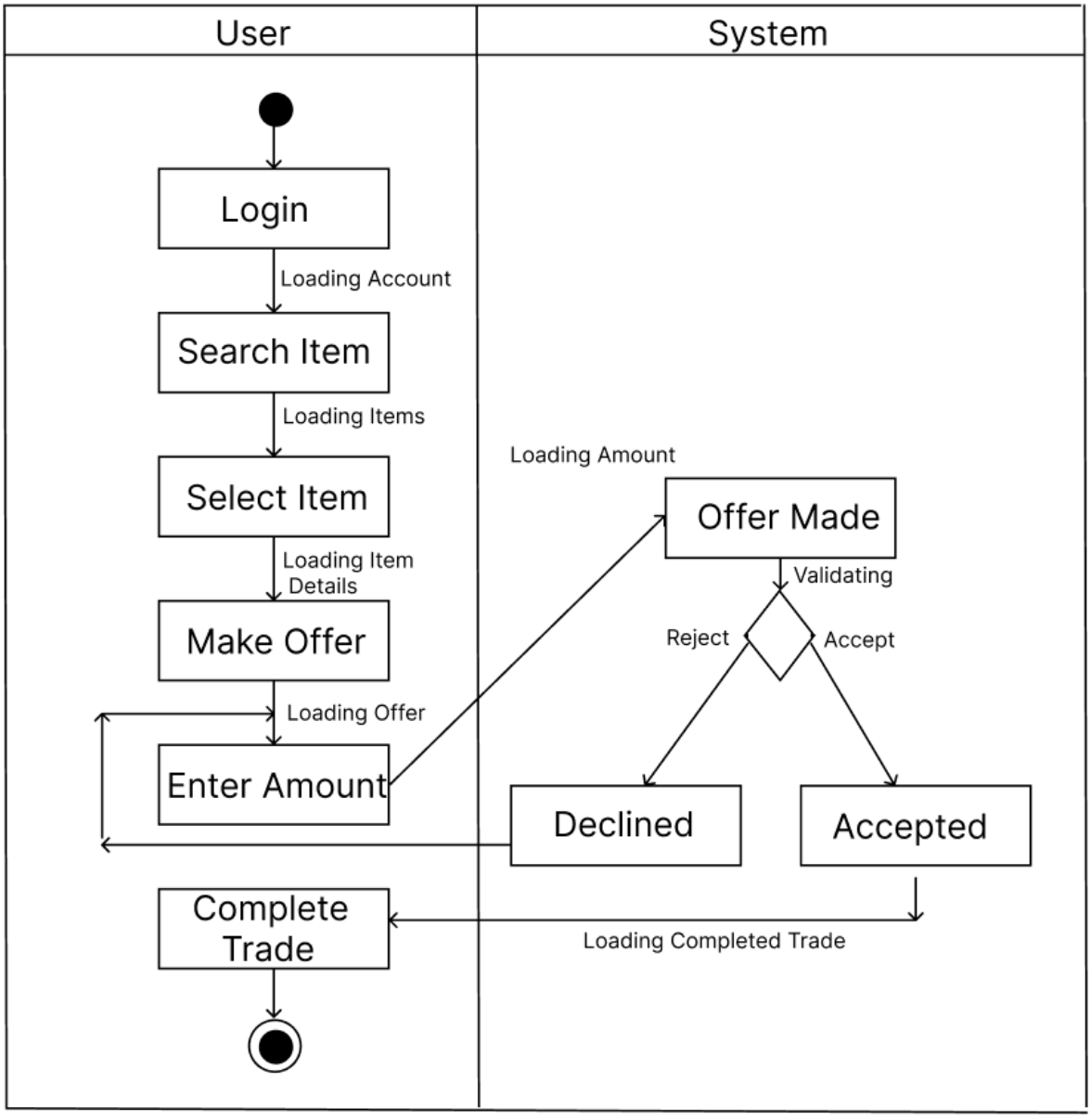
(5) information flow continuity must be maintained from level to level,2 and

(6) one bubble at a time should be refined.

Practical:

For Activity diagram

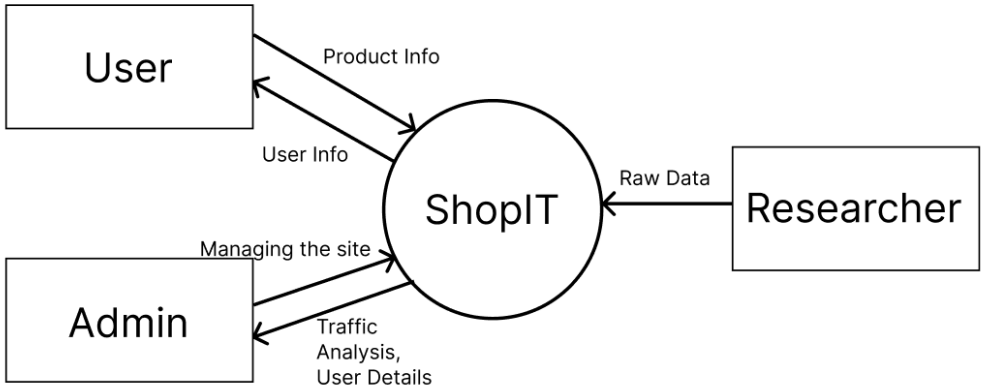
1. Identify any Use Case of your case study from Expt 3. Use case: Buy Food item on an Online Food Delivery System
2. Draw Activity and Swimlane Activity diagram for that Use-case


User
System
Login
Loading Account
Search Item
Loading Items
Select Item
Loading Item Details
Make Offer
Loading Offer
Enter Amount
Loading Amount
Offer Made
Validating
Reject
Accept
Declined
Accepted
Complete Trade
Loading Completed Trade

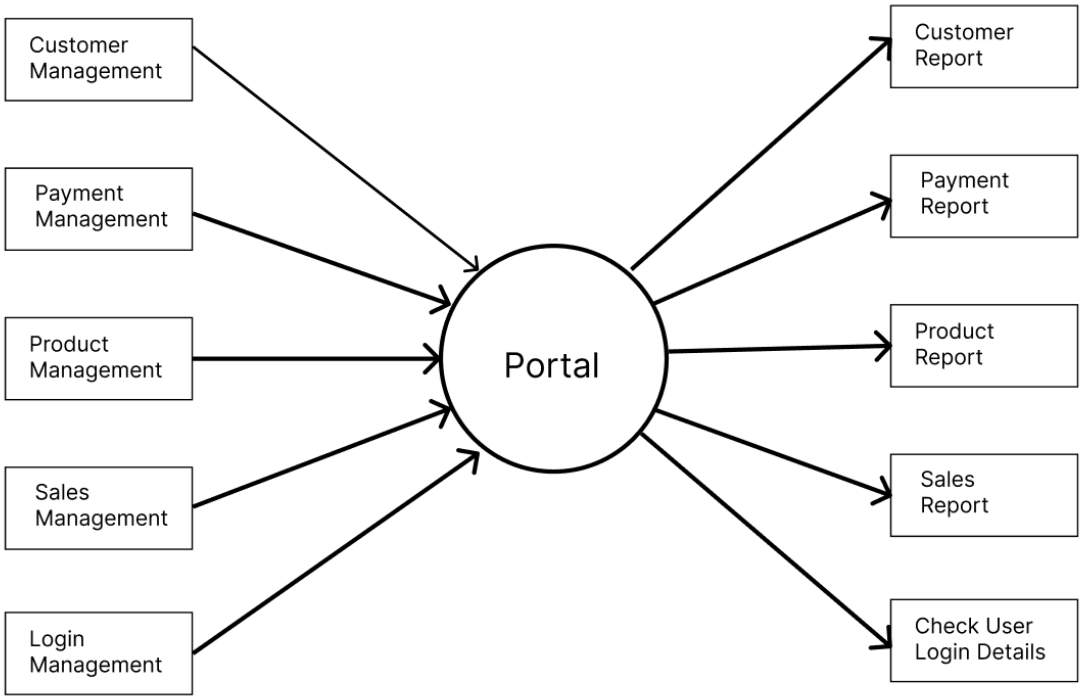
Academic Year: 2021_22

For DFD

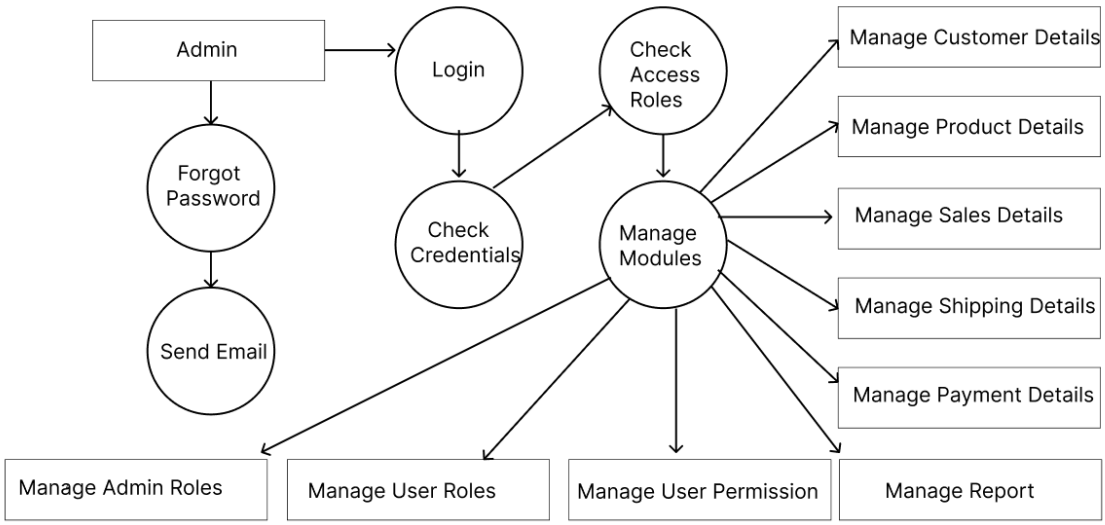
1. Draw a Context Level / Level 0 DFD to depict the data flow of your entire system.



2. Identify various processes in your case study and depict in the Level 1 DFD.



3. Refine the Level 1 DFD further to create Level 2. Refinement continues till each bubble performs only one function at a time.



Conclusion:

Thus, we are able to draw an Activity and Swim lane diagram for our case study. We are also able to depict the flow of data through various processes through different level DFDs.

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

Academic Year: 2021_22

SE - Experiment No. 5

Div: B **Batch:B3**

Team Members:

Dhruv Bheda: 60004200102
Sahej Jain: 60004200111
Ayush Jain: 60004200132
Varun Vekaria: 60004200167

Aim: Estimate effort and cost required using FP/COCOMO for the project. Create WBS and Gantt Chart for the same. Use PM Tool to depict a project plan.

Theory:

Work Breakdown Structure:

Work Breakdown Statement

A work breakdown statement (WBS) is a categorized list of tasks with an estimate of resources required to complete the task. An example WBS appears below.

WBS #	Task Description	Est Person-Hrs	Who	Resources	M&S
5	Profile motor power				
5.1	Design test stand	20	SE, JM	Pro/E	
5.2	Build test stand	15	SE, JM	Frame & brake parts	$35
5.3	Test 3 motors	3	SE, JM	Stroboscope	$75
5.4	Plot torque vs. speed	2	JM	Excel	

(M&S = Materials & Supplies)

Gantt Chart Basics

Gantt charts are a project planning tool that can be used to represent the timing of tasks required to complete a project. Because Gantt charts are simple to understand and easy to construct, they are used by most project managers for all but the most complex projects.

In a Gantt chart, each task takes up one row. Dates run along the top in increments of days, weeks or months, depending on the total length of the project. The expected time for each task is represented by a horizontal bar whose left end marks the expected beginning of the task and whose right end marks the expected completion date. Tasks may run sequentially, in parallel or overlapping.

As the project progresses, the chart is updated by filling in the bars to a length proportional to the fraction of work that has been accomplished on the task. This way, one can get a quick reading of project progress by drawing a vertical line through the chart at the current date. Completed tasks lie to the left of the line and are completely filled in. Current tasks cross the line and are behind schedule if their filled-in section is to the left of the line and ahead of schedule if the filled-in section stops to the right of the line. Future tasks lie completely to the right of the line.

In constructing a Gantt chart, keep the tasks to a manageable number (no more than 15 or 20) so that the chart fits on a single page. More complex projects may require subordinate charts which detail the timing of all the subtasks which make up one of the main tasks. For team projects, it often helps to have an additional column containing numbers or initials which identify who on the team is responsible for the task.

Often the project has important events which you would like to appear on the project timeline, but which are not tasks. For example, you may wish to highlight when a prototype is complete or the date of a design review. You enter these on a Gantt chart as "milestone" events and mark them with a special symbol, often an upside-down triangle.

For Estimation

1. Use FP / COCOMO model to estimate Effort and subsequently Cost required to develop the project.
2. Show all the tables and steps of the estimation model.

FP Estimation:

External Inputs: User registration/login, Product Search, Add Product to cart, Checkout Process, Payment processing

External Inquiry: Product details, Order Tracking

Internal Logical Files: User Profile data, Product catalog data, Order history data

External Outputs: Order Confirmation, Payment receipt, Shipping Confirmation, Order cancellation.

External interface files: None

Information Domain Value	**Count**	**Simple**	**Average**	**Complex**	**Total**
External Inputs	5	3	**4**	6	5*4 = 20
External Inquiry	2	4	**5**	7	2*5 = 10
Internal logical Files	3	3	**4**	6	3*4 = 12
External Outputs	4	7	**10**	15	4*10 = 40
External Interface Files	0	5	**7**	10	0*7 = 0
Total					**82**

Total Count: 82

Value Adjustment Factors:

The Fi (i = 1 to 14) are value adjustment factors (VAF) based on responses to the following questions:

1. **Does the system require reliable backup and recovery?**
   - 4 - Reliable backup and recovery is critical for an e-commerce platform, as it deals with sensitive information such as customer profiles, payment information, and order history.
2. **Are specialized data communications required to transfer information to or from the application**?
   - 3 - An e-commerce platform may require specialized data communications to integrate with external systems such as payment gateways, shipping providers, and other third-party services.
3. **Are there distributed processing functions**
   - 4 - An e-commerce platform may have distributed processing functions to handle high traffic and distribute workload across multiple servers or nodes.
4. **Is performance critical?**
   - 4 - Performance is a critical factor for an e-commerce platform, as it directly impacts user experience and customer satisfaction.
5. **Will the system run in an existing, heavily utilized operational environment?**
   - 4 - An e-commerce platform is likely to run in an existing, heavily utilized operational environment to ensure availability and reliability
6. **Does the system require online data entry?**
   - 5 - An e-commerce platform requires online data entry, as it deals with real-time transactions and updates to customer accounts, orders, and inventory.
7. **Does the online data entry require the input transaction to be built over multiple screens or operations?**
   - 2 - Online data entry for an e-commerce platform may require input transactions to be built over multiple screens or operations, such as selecting products, entering delivery and payment details, and confirming orders.
8. **Are the ILFs updated online?**
   - 5 - Updating ILFs online can ensure that the system operates with up-to-date information and avoids errors or inconsistencies

9. **Are the inputs, outputs, files, or inquiries complex?**
   - 3 - Inputs, outputs, files, and inquiries in an e-commerce platform can be complex due to the variety of data and operations involved, such as customer details, product listings, order processing, and payment processing
10. **Is the internal processing complex?**
    - 4 - Internal processing in an e-commerce platform like Amazon can be complex due to the business logic and rules involved, such as order processing, payment processing, inventory management, and recommendation systems.
11. **. Is the code designed to be reusable?**
    - 3- Code reusability is an important aspect of software engineering that can save time and effort in developing and maintaining software systems.
12. **Are conversion and installation included in the design?**
    - 3 - Conversion and installation are important aspects of the software development process, particularly for complex systems like an e-commerce platform.
13. **Is the system designed for multiple installations in different organizations?**
    - 4 - If the system is designed for multiple installations in different organizations, it will require additional effort to make it adaptable to different environments and configurations.
14. **Is the application designed to facilitate change and ease of use by the user**
    - 4 - User experience and ease of use are critical factors in the success of any software application.

Σ (Fi) = 53

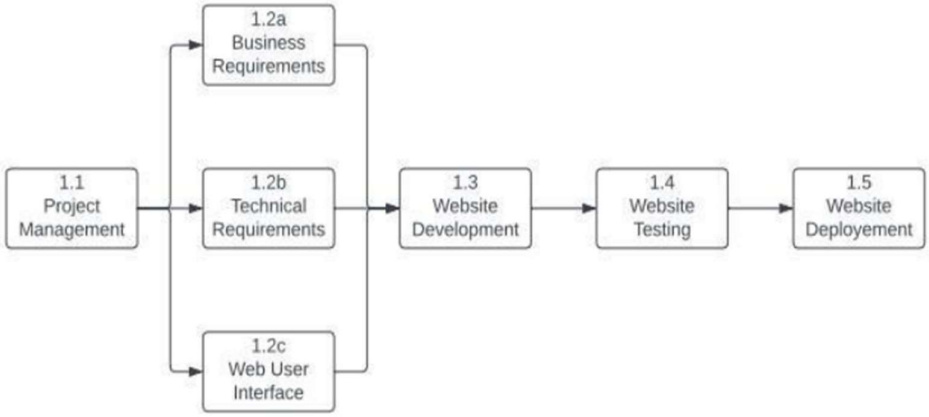
The estimated number of FP is derived:

FP estimated = count-total x [0.65 + 0.01 x Σ (Fi)] =

= 82* [0.65 + 0.01 x 53] = 96.76

Therefore, FP estimated is 96.76 pm

WBS:



1.1. Project Management

1.2. Requirements

1.2.1. Business

1.2.1.1. Domain Name

1.2.1.2. Marketing

1.2.1.3. Process

1.2.1.3.1. Customer Ordering

1.2.1.3.2.Order delivering

1.2.1.3.3. Order Fulfillment

1.2.1.3.4. Existing Systems Integration

1.2.1.3.4.1. Shipping

1.2.1.3.4.2. Financial

1.2.1.3.4.3.Item details and critical data

1.2.1.3.4.4. Payment

- 1.2.1.4. Operations Management/Site Maintenance
- 1.2.2. Technical
  - 1.2.2.1. Security
  - 1.2.2.2. Network
    - 1.2.2.2.1. Hosting
    - 1.2.2.2.2. Platform
- 1.2.3. Web User Interface
  - 1.2.3.1. Unrestricted Content
  - 1.2.3.2. Restricted Content
  - 1.2.3.3. eCommerce Capabilities

1.3. Website Development

- 1.3.1. Create Code Design Document
- 1.3.2. Code Control System
- 1.3.3. Development Environment
- 1.3.4. User Interface Design
- 1.3.5. Database Model
- 1.3.6. Code Generation
  - 1.3.6.1. Release Testing
  - 1.3.6.2. Bug Fixes
- 1.3.7. Content
  - 1.3.7.1. Product Catalog
    - 1.3.7.1.1. Descriptions
    - 1.3.7.1.2. Images

1.4. Website Testing

1.4.1. Test Environment Setup

1.4.2. Test Plan

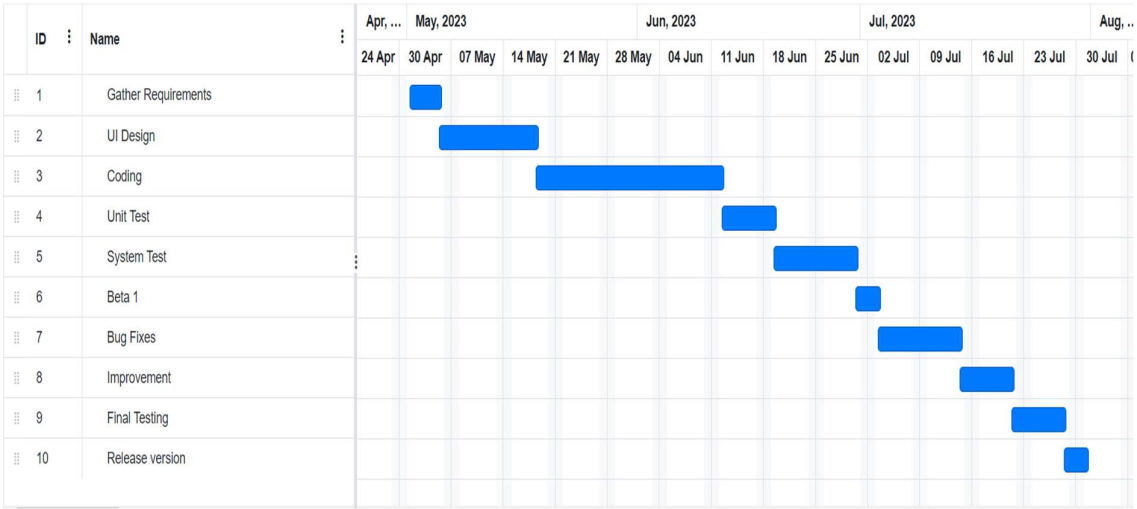
1.4.3. Results Reporting

1.5. Website Deployment

1.5.1. Application Integration

1.5.2. Release Documentation

Gantt Chart:



Conclusion:

Thus, we are able to estimate effort required for our project and also create Gantt Chart.

Experiment No. 6

Dhruv Bheda-60004200102

Sahej Jain– 60004200111

Ayush Jain– 60004200132

Varun Vekaria– 60004200167

Aim: Develop Sequence and Collaboration diagram for the project.

Sequence Diagram:

A sequence diagram is used to show the dynamic communications between objects during execution of a task. It shows the temporal order in which messages are sent between the objects to accomplish that task. One might use a sequence diagram to show the interactions in one use case or in one scenario of a software system.

A sequence diagram shows method calls using horizontal arrows from the caller to the callee, labelled with the method name and optionally including its parameters, their types, and the return type.

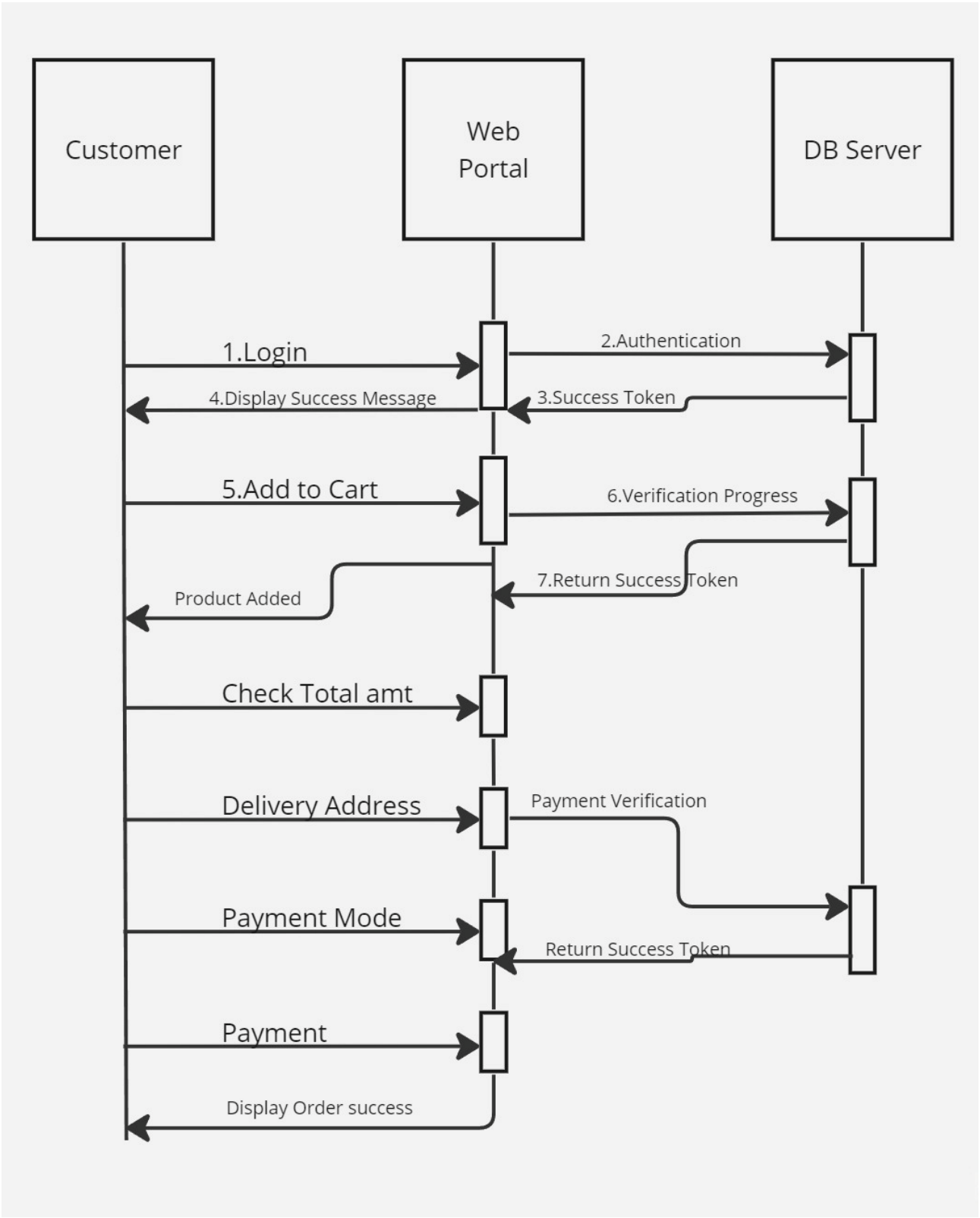
The **Sequence Diagram for E-Commerce Website** is a form of interaction diagram that shows how a group of entities interact and in a specific order. It is used to understand and describe a series of processes on the website.

In addition, this **E-commerce Website Sequence Diagra**m is created based on Unified Modeling Language (UML) that depicts the flow of messages between objects in a scenario. It's composed of entities connected by lifelines, and the communications they exchange over time.

Here there are 3 instances of objects:

1. Customer
2. Web Portal of Company
3. Database Server of the company

The diagram has the message and replies messages with sequence numbering, and the lifeline of each object.


Customer
Web Portal
DB Server
1.Login
2.Authentication
4.Display Success Message
3.Success Token
5.Add to Cart
6.Verification Progress
7.Return Success Token
Product Added
Check Total amt
Delivery Address
Payment Verification
Payment Mode
Return Success Token
Payment
Display Order success

Collaboration (communication) Diagrams:

In a collaboration diagram the interacting objects are represented by rectangles. Associations between objects are represented by lines connecting the rectangles. There is typically an incoming arrow to one object in the diagram that starts the sequence of message passing. That arrow is labelled with a number and a message name.

UML Collaboration Diagram

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. 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. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.

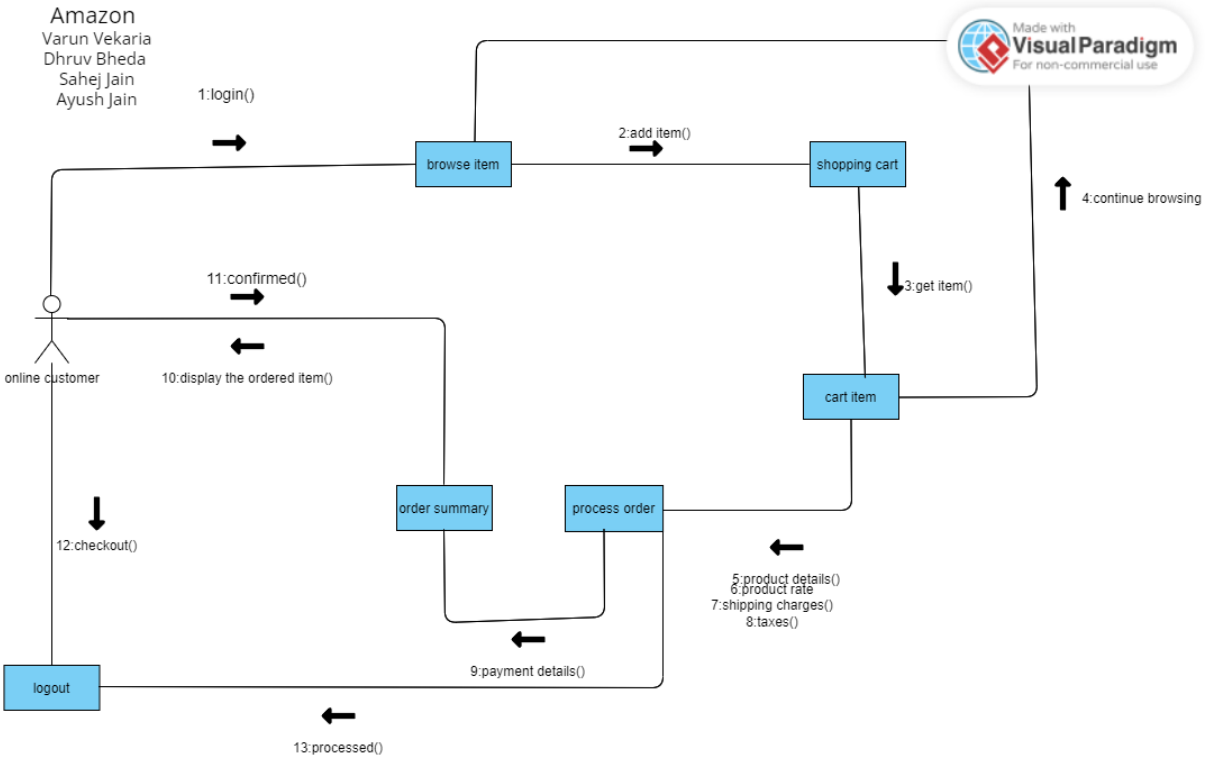
Benefits of a Collaboration Diagram

1. The collaboration diagram is also known as Communication Diagram.
2. It mainly puts emphasis on the structural aspect of an interaction diagram, i.e., how lifelines are connected.
3. The syntax of a collaboration diagram is similar to the sequence diagram; just the difference is that the lifeline does not consist of tails.
4. The messages transmitted over sequencing is represented by numbering each individual message.
5. The collaboration diagram is semantically weak in comparison to the sequence diagram.
6. The special case of a collaboration diagram is the object diagram.
7. It focuses on the elements and not the message flow, like sequence diagrams.
8. Since the collaboration diagrams are not that expensive, the sequence diagram can be directly converted to the collaboration diagram.
9. There may be a chance of losing some amount of information while implementing a collaboration diagram with respect to the sequence diagram.

The drawback of a Collaboration Diagram

1. Multiple objects residing in the system can make a complex collaboration diagram, as it becomes quite hard to explore the objects.

2. It is a time-consuming diagram.
3. After the program terminates, the object is destroyed.



Conclusion:

Thus, we are able to draw a Sequence and Collaboration diagram for a functionality of our case study.

Experiment No. 7

Dhruv Bheda - 60004200102

Sahej Jain - 60004200111

Ayush Jain - 60004200132

Varun Vekaria - 60004200167

AIM: Design test scenarios and test cases for your SRS.

PERFORMANCE:

Test Scenarios:

Test Scenario ID	Requirement	Test Scenario Description	Importance	No. of Test Cases
TS_SHOPIT_LOGIN_01	Login Functionality	Verify user can login with correct credentials	High	3
TS_SHOPIT_LOGIN_02	Login Functionality	Verify user cannot login with incorrect password	Medium	1
TS_SHOPIT_LOGIN_03	Login Functionality	Verify user cannot login with incorrect username	Medium	1
TS_SHOPIT_SEARCH_01	Search Functionality	Verify user can search for products by name	High	2
TS_SHOPIT_SEARCH_02	Search Functionality	Verify user can filter search results by category	Medium	1
TS_SHOPIT_CART_01	Add to Cart Functionality	Verify user can add a product to their cart	High	2
TS_SHOPIT_CART_02	Add to Cart Functionality	Verify user can remove a product from their cart	Medium	1
TS_SHOPIT_CHECKOUT_01	Checkout Functionality	Verify user can successfully checkout and place an order	High	3
TS_SHOPIT_CHECKOUT_02	Checkout Functionality	Verify user cannot checkout with an empty cart	Medium	1

Test Cases:

1. LOGIN:

Test Case ID	Test Objective	Precondition	Test Steps	Test Data	Expected Result	Post Condition
TC_LOGIN_01	Verify successful login with valid credentials	The user is on the login page	1. Enter valid username and password 2. Click on "Login" button	Username: johndoe Password: password123	The user is redirected to the home page and can access the system features	The user is logged in and can perform operations as expected
TC_LOGIN_02	Verify login failure with invalid username	The user is on the login page	1. Enter invalid username and valid password 2. Click on "Login" button	Username: invaliduser Password: password123	An error message is displayed stating that the username or password is incorrect	The user is still on the login page and cannot access the system features
TC_LOGIN_03	Verify login failure with invalid password	The user is on the login page	1. Enter valid username and invalid password 2. Click on "Login" button	Username: johndoe Password: invalidpassword	An error message is displayed stating that the username or password is incorrect	The user is still on the login page and cannot access the system features
TC_LOGIN_04	Verify login failure with empty username and password fields	The user is on the login page	1. Leave both username and password fields empty 2. Click on "Login" button	N/A	An error message is displayed stating that both username and password are required fields	The user is still on the login page and cannot access the system features.

2. SEARCH RESULTS:

Test Case ID	Test Objective	Precondition	Test Steps	Test Data	Expected Result	Post-condition
TC_SR_01	Verify that user is able to search for a product using the search bar	User is on the ShopIt homepage	1. Enter the product name in the search bar 2. Press Enter key or click on the search icon	Product name: "laptop"	Product listing page is displayed with all relevant products matching the search keyword	User is able to view and interact with the search results
TC_SR_02	Verify that user is able to search for a product using filters	User is on the product listing page	1. Apply relevant filters for the product search 2. Click on "Apply" button	Product category: "Electronics" Brand: "Dell"	Product listing page is displayed with all relevant products matching the applied filters	User is able to view and interact with the filtered search results
TC_SR_03	Verify that user is not able to search for an invalid product	User is on the ShopIt homepage	1. Enter an invalid product name in the search bar 2. Press Enter key or click on the search icon	Product name: "qwerty"	Error message is displayed on the page stating that no product was found for the given search keyword	User is still on the same page and can perform a new search
TC_SR_04	Verify that user is able to view all products on clearing the search bar	User has performed a search operation using the search bar	1. Click on the clear search icon in the search bar	Search bar is cleared and all the products are displayed on the page	User is able to view and interact with all the products on the page	

3. ADD TO CART:

Test Case ID	Test Objective	Precondition	Test Steps	Test Data	Expected Result	Post-condition
TC_AC_01	Verify that a user can add a product to their cart by clicking on the "Add to Cart" button	The user is logged in and on the product page	1. Navigate to the product page 2. Click on the "Add to Cart" button	Product SKU, quantity	The product is added to the cart and the cart icon updates to reflect the number of items in the cart	The product is added to the user's cart
TC_AC_02	Verify that the user is prompted to login if they try to add a product to their cart without logging in	The user is not logged in and on the product page	1. Navigate to the product page 2. Click on the "Add to Cart" button	Product SKU, quantity	The user is redirected to the login page with a message asking them to log in to add the product to their cart	The user is redirected to the login page
TC_AC_03	Verify that a user can add multiple products to their cart	The user is logged in and on the product page	1. Navigate to the product page 2. Click on the "Add to Cart" button for multiple products	Product SKUs, quantities	The products are added to the cart and the cart icon updates to reflect the total number of items in the cart	The products are added to the user's cart
TC_AC_04	Verify that a user cannot add a product to their cart if it is out of stock	The user is logged in and on the product page for an out of stock product	1. Navigate to the product page 2. Verify that the "Add to Cart" button is disabled	N/A	The "Add to Cart" button is disabled and a message is displayed indicating that the product is out of stock	The user cannot add the out of stock product to their cart
TC_AC_05	Verify that a user cannot add more items to their cart than are in stock	The user is logged in and on the product page	1. Navigate to the product page 2. Enter a quantity that is greater	Product SKU, quantity	A message is displayed indicating that there is not enough	The user cannot add more items to their cart

Test Case ID	Test Objective	Precondition	Test Steps	Test Data	Expected Result	Post-condition
			than the quantity in stock 3. Click on the "Add to Cart" button		stock to fulfill the order	than are in stock
TC_AC_06	Verify that the user can view their cart after adding items to it	The user has added at least one item to their cart	1. Click on the cart icon in the navigation bar	N/A	The user is taken to their cart where they can view the items they have added and their total cost	The user can view their cart and its contents

4. CHECKOUT:

Test Case ID	Test Objective	Precondition	Test Steps	Test Data	Expected Result	Post-condition
TC_CHECK_1	Verify checkout process with valid product and payment details	User has added product(s) to cart and is on the checkout page	1. Enter valid shipping information 2. Select a payment method 3. Submit the order	Valid shipping information and payment details	Order is successfully processed and a confirmation message is displayed	User is redirected to the order confirmation page
TC_CHECK_2	Verify error message is displayed when shipping information is missing	User has added product(s) to cart and is on the checkout page	1. Leave shipping information fields blank 2. Select a payment method 3. Submit the order	Missing shipping information	An error message is displayed indicating that shipping information is required	User is prompted to enter the required shipping information
TC_CHECK_3	Verify error message is displayed when payment information is missing	User has added product(s) to cart and is on the checkout page	1. Enter valid shipping information 2. Leave payment information fields blank 3. Submit the order	Missing payment information	An error message is displayed indicating that payment information is required	User is prompted to enter the required payment information

Test Case ID	Test Objective	Precondition	Test Steps	Test Data	Expected Result	Post-condition
TC_CHECK_4	Verify order summary is displayed before submitting the order	User has added product(s) to cart and is on the checkout page	1. Verify that order summary is displayed on the checkout page	Valid product(s) in the cart	Order summary is displayed with details of the product(s) and the total amount	User can verify the details of the order before submitting it
TC_CHECK_5	Verify that order is not processed when payment fails	User has added product(s) to cart and is on the checkout page	1. Enter valid shipping information 2. Select a payment method that is known to fail 3. Submit the order	Valid shipping information and payment details that are known to fail	An error message is displayed indicating that payment failed	User is prompted to select a different payment method or update the payment details

CONCLUSION: Thus, we are able to create test scenarios and test cases for our case study.

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

A.Y. 2021 – 22

Experiment No. 8

Dhruv Bheda - 60004200102

Sahej Jain - 60004200111

Ayush Jain - 60004200132

Varun Vekaria – 60004200167

Aim: Study of Azure Devops

Theory:

Azure DevOps provides developer services for allowing teams to plan work, collaborate on code development, and build and deploy applications. Azure DevOps supports a collaborative culture and set of processes that bring together developers, project managers, and contributors to develop software. It allows organizations to create and improve products at a faster pace than they can with traditional software development approaches.

Azure DevOps provides integrated features that you can access through your web browser or IDE client.

Azure Repos:

Azure Repos is a set of version control tools that you can use to manage your code. Version control systems are software that help you track changes you make in your code over time. As you edit your code, you tell the version control system to take a snapshot of your files. The version control system saves that snapshot permanently so you can recall it later if you need it. Use version control to save your work and coordinate code changes across your team.

Azure Repos provides two types of version control:

1. Git repositories: Git is the most commonly used version control system today and is quickly becoming the standard for version control. Git is a distributed version control system, meaning that your local copy of code is a complete version control repository. These fully functional local repositories make it is easy to work offline or remotely.

You commit your work locally, and then sync your copy of the repository with the copy on the server.

2. Team Foundation Version Control (TFVC): Azure Repos also supports Team Foundation Version Control (TFVC). TFVC is a centralized version control system. Typically, team members have only one version of each file on their dev machines. Historical data is maintained only on the server. Branches are path-based and created on the server.

Azure Pipelines:

Azure Pipelines automatically builds and tests code projects to make them available to others. It works with just about any language or project type. Azure Pipelines combines continuous integration (CI) and continuous delivery (CD) to test and build your code and ship it to any target.

Continuous Integration (CI) is the practice used by development teams of automating merging and testing code. Implementing CI helps to catch bugs early in the development cycle, which makes them less expensive to fix. Automated tests execute as part of the CI process to ensure quality. Artifacts are produced from CI systems and fed to release processes to drive frequent deployments. The Build service in Azure DevOps Server helps you set up and manage CI for your applications.

Continuous Delivery (CD) is a process by which code is built, tested, and deployed to one or more test and production environments. Deploying and testing in multiple environments increases quality. CI systems produce deployable artifacts, including infrastructure and apps. Automated release processes consume these artifacts to release new versions and fixes to existing systems. Monitoring and alerting systems run continually to drive visibility into the entire CD process.

Azure Boards:

Delivers a suite of Agile tools to support planning and tracking work, code defects, and issues using Kanban and Scrum methods. Azure Boards provides software development teams with the interactive and customizable tools they need to manage their software projects. It provides

a rich set of capabilities including native support for Agile, Scrum, and Kanban processes, calendar views, configurable dashboards, and integrated reporting. These tools scale as your business grows.

Quickly and easily track work, issues, and code defects associated with your project. The Kanban board, shown in the following image, is just one of several tools that allows you to add, update, and filter user stories, bugs, features, and epics.

Azure Test Plans:

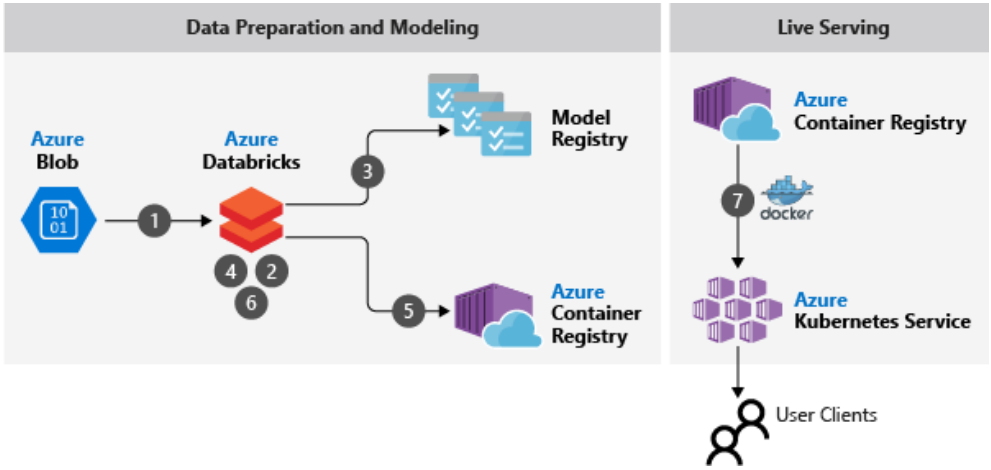
Azure Test Plans provides rich and powerful tools everyone in the team can use to drive quality and collaboration throughout the development process. The easy-to-use, browser-based test management solution provides all the capabilities required for planned manual testing, user acceptance testing, exploratory testing, and gathering feedback from stakeholders.

Azure Artifacts:

Azure Artifacts enable developers to consume and publish different types of packages to Artifacts feeds and public registries such as NuGet.org and npmjs.com. You can use Azure Artifacts in conjunction with Azure Pipelines to deploy packages, publish build artifacts, or integrate files between your pipeline stages to build, test, or deploy your application.

Azure Architecture Solutions:

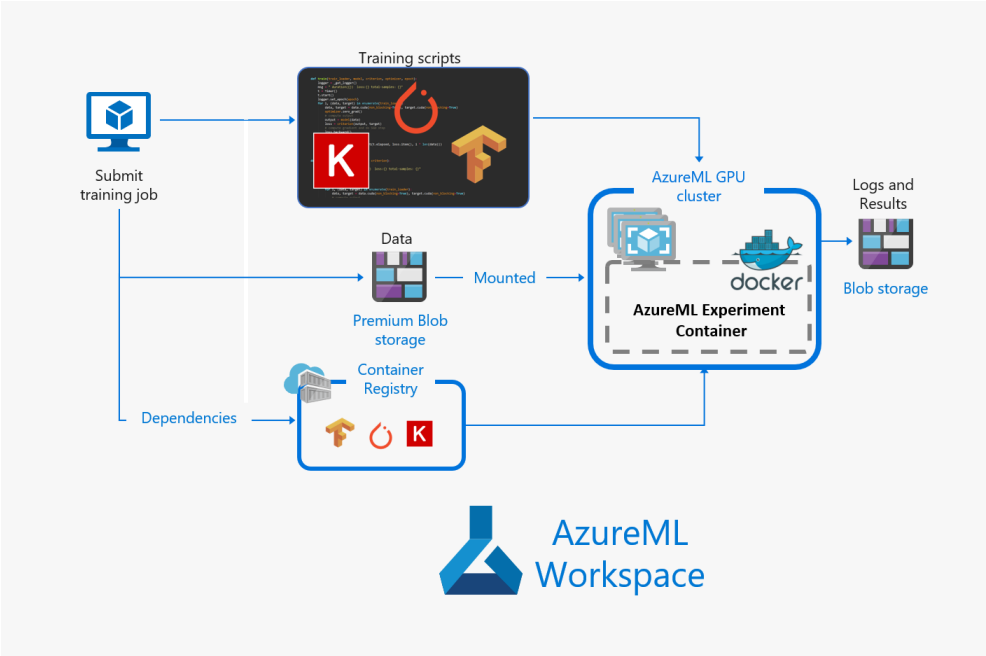
1. Azure Architecture for Content based Recommendation System:



This example scenario covers the training, evaluation, and deployment of a machine learning model for content-based personalization on Apache Spark using Azure Databricks. In this case, a model is trained with a supervised classification algorithm on a dataset containing user and item features. The label for each example is a binary value indicating that the user engaged with (for example, clicked) an item. This scenario covers a subset of the steps required for a full end-to-end recommendation system workload. The broader context of this scenario is based on a generic e-commerce website with a front end that serves rapidly changing content to its users. This website uses cookies and user profiles to personalize the content for that user. Along with user profiles, the website may have information about every item it serves to each user.

2. Azure Architecture for Distributed Training Deep learning models:

This reference architecture shows how to conduct distributed training of deep learning models across clusters of GPU-enabled VMs. The scenario is image classification, but the solution can be generalized to other deep learning scenarios such as segmentation or object detection.



Workflow

This architecture consists of the following services:

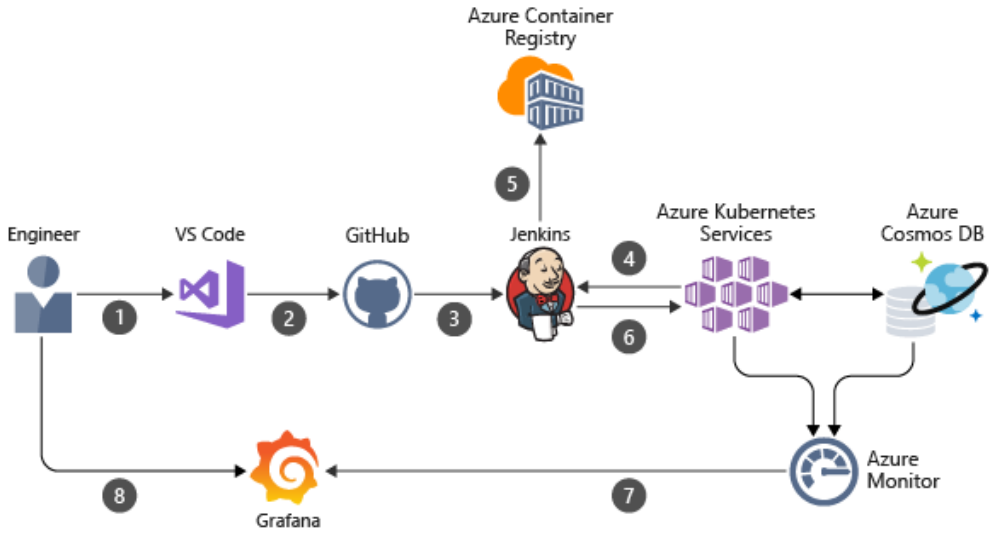
Azure Machine Learning Compute plays the central role in this architecture by scaling resources up and down according to need. Azure ML Compute is a service that helps provision

and manage clusters of VMs, schedule jobs, gather results, scale resources, and handle failures. It supports GPU-enabled VMs for deep learning workloads.

Standard Blob storage is used to store the logs and results. Premium Blob storage is used to store the training data and is mounted in the nodes of the training cluster using blobfuse. The Premium tier of Blob storage offers better performance than the Standard tier and is recommended for distributed training scenarios. When mounted using blobfuse, during first the epoch, the training data is downloaded to the local disks of the training cluster and cached. For every subsequent epoch, the data is read from the local disks, which is the most performant option.

Container Registry is used to store the Docker image that Azure Machine Learning Compute uses to run the training.

3. Azure Architecture for CI/CD:



Azure Web Apps is a fast and simple way to create web apps using ASP.NET, Java, Node.js, or PHP. Deliver value faster to your customers with a continuous integration and continuous deployment (CI/CD) pipeline that pushes each of your changes automatically to Web Apps.

Experiment No. 9

Dhruv Bheda - 60004200102

Sahej Jain - 60004200111

Ayush Jain - 60004200132

Varun Vekaria – 60004200167

Aim: To create a RMMM plan: Create risk assessment template for a case study.

Theory:

The RMMM plan:

Risk Mitigation, Monitoring and Management Plan (RMMM) — documents all work performed as part of risk analysis and is used by the project manager as part of the overall project plan. RIS is maintained using a database system, so that creation and information entry, priority ordering, searches, and other analysis may be accomplished easily. Risk monitoring is a project tracking activity.

Three primary objectives:

- Assess whether predicted risks do, in fact, occur.
- Ensure that risk aversion steps defined for the risk are being properly applied.
- Collect information that can be used for future risk analysis.

Performance:

1. **Identify Risks:**

   Product Size Risks:
   - **Estimated size in lines of code (LOC)**
     Property Management will have an estimated 19,555 lines of code.

- **Degree of confidence in estimated size**
  We are highly confident in our estimated size.

- **Estimated size in number of programs, files, and transactions**
  1. We estimate 2 programs.
  2. We estimate 10 large files for the engine, 5 large files for the user-interface.
  3. We estimate 40 or more transactions for the engine, and 20 transactions for the user-interface.

- **Percentage deviation in size from average for previous products**
  We allow for a 20% deviation from average.

- **Size of database created or used**
  The size of the database that we will use will be an estimated 7 tables. The number of fields will vary per table and will have an overall average of 8 fields per table. The number of records in each table will vary with the number of sprites that the user adds to the project, and the number of instances of each sprite that the user creates.

- **Number of users**
  The number of users will be fairly low. There will be one user per instance of the software running, as the software is not client/server or intended for multi-user use.

- **Number of projected changes to the requirements**
  We estimate 3 possible projected changes to the requirements. These will be as a result of our realization of what is required and not required as we get further into implementation, as well as a result of interaction with the customer and verification of the customer's requirements.

- **Amount of reuse of software**
  Reuse will be very important to get the project started. DirectX is very simple to reuse (for the most part) and previous programs used to code for with DirectX will be reviewed and much DirectX code will be recopied.

Business Impact Risks

- **Effect on company revenue**
  None, Property Management will be distributed as freeware. It will be developedusing existing tools. The staff will be compensated with their term graderather than financially. The revenue of PA Software will not be affectedpositively or negatively by its development and release.

- **Visibility of product to senior management**
  N/A, PA software does not have an established senior management; therefore Property Management cannot be visible to them.

- **Reasonableness of delivery deadline**
  Fairly reasonable. The project deadline was established before the project was undertaken. The initial planning for Property Management was executed with the deadline in mind. The scope of the project was limitedto keep the project "doable" within the allowed period of time.

- **Number of customers and the consistency of their needs**
  The number of customers will be fairly low. Property Management is intended as a tool to aid people in real estate. Their needs are considered consistent, as all target users will be people who are builders who want to sell there property and people who are willing to buy it.

- **Number of other systems/products that product must be interoperable with**
  1. Microsoft Access. More specifically the Microsoft JET Database Engine, which is included with Visual Basic and Visual C++.
  2. Microsoft Visual Basic.
  3. Microsoft Visual C++.
  4. Microsoft DirectX.

- **Sophistication of end users**
  Low. The target users are novice game programmers. Property Management is designed to be easy to use, and is supplied with Wizards to guide the users through all necessary steps in buying and listing the property.

- **Amount and quality of documentation that must be produced and**

delivered to customer
The customer will be supplied with a complete online help file and user's manual for Property Management. Coincidentally, the customer will have access to all development documents for Property Management, as the customer will also be grading the project.

- **Governmental constraints in the construction of the product**
  Rera ID required for property to list and OC(occupation certificate) is required which is approved by the government.

- **Costs associated with late delivery**
  Late delivery will prevent the customer from issuing a letter of acceptance for the product, which will result in an incomplete grade for the course for all members of the organization

- **Costs associated with a defective product**
  Unknown at this time.

Customer Related Risks

- **Have you worked with the customer in the past?**
  Yes, All team members have completed at least one project for the customer, though none of them have been to the magnitude of the current project.

- **Does the customer have a solid idea of what is required?**
  Yes, the customer has access to both the System Requirements Specification, and the Software Requirements Specification for the Property Management project.

- **Will the customer agree to spend time in formal requirements gathering meetings to identify project scope?**
  Unknown. While the customer will likely participate if asked, the inquiry has not yet been made.

- **Is the customer willing to establish rapid communication links with the developer?**
  Yes, the customer is available through email, as well as in person, to all project developers.

- **Is the customer willing to participate in reviews?**
  Unknown. While the customer will likely participate if asked, the inquiry has not yet been made.

- **Is the customer technically sophisticated in the product area?**
  Yes. The customer trained some members of the design team in game development. He is the instructor for the course CIS 587, Computer Game Design and Implementation at the University of Michigan-Dearborn

- **Is the customer willing to let your people do their job?**
  Yes. As the Property Management project is a senior design project, the customer is available if needed, but does not interfere with development operations otherwise.

- **Does the customer understand the software process?**
  Yes. The customer was the instructor of CIS 375, the Software Engineering course attended by all members of the design team.

Process Risks

- **Does senior management support a written policy statement that emphasizes the importance of a standard process for software development?**
  N/A. PA Software does not have a senior management. It should be noted that the structured method has been adopted for the Property Management project. At the completion of the project, it will be determined if the software method is acceptable as a standard process, or if changes needto be implemented.

- **Has your organization developed a written description of the software process to be used on this project?**
  Yes. Property Management is under development using the structured method as described in part three of Roger S. Pressman's Software Engineering, APractitioner's Approach.

- **Are staff members willing to use the software process?**
  Yes. The software process was agreed upon before development work began.

- **Is the software process used for other products?**
  N/A. PA Software has no other projects currently.

- **Has your organization developed or acquired a series of software engineering training courses for managers and technical staff?**
  Yes. All members of the design team have attended CIS 375, Introduction to Software Engineering at the University of Michigan – Dearborn.
- **Have documented outlines and examples been developed for all deliverables defined as part of the software process?**
  Yes. The course instructor has supplied outlines for all deliverables.

- **Are formal technical reviews of the requirements specification design and code conducted regularly?**
  No. Although informal reviews are conducted.

- **Are formal technical reviews of test procedures and test cases conducted regularly?**
  No. Although informal reviews are conducted.

- **Are the results of each formal technical review documented, including errors found and resources used?**
  N/A. As formal technical reviews have not been conducted, they cannot be documented.

- **Is there some mechanism for ensuring that work conducted on a project conforms with software engineering standards?**
  No. There has been no planned method to ensure software-engineering standards will be met.

- **Is configuration management used to maintain consistency among system/software requirements, design, code and test cases?**
  Yes. The accompanying Software Configuration Management document outlines the plan for maintaining consistency among all technical documents in the Property Management project.

- **Is a mechanism used for controlling changes to customer requirements that impact software?**
  No. The customer requirements for the Property Management project are fairly flexible. The customer has allowed a great deal of freedom to the project developers. This could become a problem if the customer does change the requirements, by the likelihood of that is extremely low.

- **Is there a documented statement of work, a software requirements specification, and a software development plan for each subcontract?**

N/A. All work is done by a single development team. No subcontracting will take place on the Property Management project.

- **Is there a procedure followed for tracking and reviewing the performance of subcontractors?**
  N/A. All work is done by a single development team. No subcontracting will take place on the Property Management project.

Technical Issues

- **Are facilitated application specification techniques used to aid in communication between the customer and the developer?**
  The development team will hold frequent meetings directly with the customer. No formal meetings are held (all informal). During these meetings the software is discussed and notes are taken for future review.

- **Are specific methods used for software analysis?**
  Special methods will be used to analyze the software's progress and quality. These are a series of tests and reviews to ensure the software is up to speed. For more information, see the Software Quality Assurance and Software Configuration Management documents.

- **Do you use a specific method for data and architectural design?**
  Data and architectural design will be mostly object oriented. This allows for a higher degree data encapsulation and modularity of code.

- **Is more than 90 percent of your code written in a high-order language?**
  Yes. Code will be written in a combination of Visual Basic, C++, DirectX, with a bit of SQL.

- **Are specific conventions for code documentation defined and used?**
  No. Specific conventions have not been established, but all design members have agreed to comment code as completely as possible.

- **Do you use specific methods for test case design?**
  Yes. Test cases will be attempts to model existing basic model such as magicbricks, 99acres and possibly Nobroker.

- **Are software tools used to support planning and tracking activities?**
  No. No software tools are to be used. Due to the existing deadline, the development team felt it would be more productive to begin implementing the project than trying to learn new software tools. After the completion of the project software tools may be implemented for future projects.

- **Are configuration management software tools used to control and track change activity throughout the software process?**
  No. No software tools are to be used. Due to the existing deadline, the development team felt it would be more productive to begin implementing the project than trying to learn new software tools. After the completion of the project software tools may be implemented for future projects.

- **Are software tools used to support the software analysis and design process?**
  No. No software tools are to be used. Due to the existing deadline, the development team felt it would be more productive to begin implementing the project than trying to learn new software tools. After the completion of the project software tools may be implemented for future projects.

- **Are tools used to create software prototypes?**
  Yes. Prototypes are created using pencil and paper, as well as interface mock-ups using Microsoft Visual Basic.

- **Are software tools used to support the testing process?**
  No. No software tools are to be used. Due to the existing deadline, the development team felt it would be more productive to begin implementing the project than trying to learn new software tools. After the completion of the project software tools may be implemented for future projects.

- **Are software tools used to support the production and managing of documentation?**
  Yes. Microsoft Word will be used to support the production and management of all technical documentation.

- **Are quality metrics collected for all software projects?**
  No. No plans have been made to collect quality metrics at this time.

- **Are productivity metrics collected for all software projects?**
  No. No plans have been made to collect productivity metrics at this time.

Technology Risks

- **Is the technology to be built new to your organization?**
  Property Management is a software tool to aid in Property Management. Development team members are familiar with this, as well as the necessary database implementation.

- **Do the customer's requirements demand the creation of new algorithms or input or output technology?**
  No. Property Management will be implemented using existing algorithms. Inputand output are handled in a traditional manner.

- **Does the software interface with new or unproven hardware?**
  Since Property Management is taking advantage of DirectX, the software willallow for a multitude of new or future hardware products. This is doneautomatically as an advantage of DX.

- **Does the software to be built interface with vendor supplied software products that are unproven?**
  No. Property Management interfaces with Microsoft Access, Microsoft Visual Basic, Microsoft Visual C++, and DirectX; all are proven software products.

- **Does the software to be built interface with a database system whose function and performance have not been proven in this application area?**
  No. Property Management utilizes the Microsoft JET database engine. This engine is used in Microsoft Access, and is used to develop database applications in many of the Microsoft Visual languages, includingVisual Basic and Visual C++. The database system is stable and used widely.

- **Is a specialized user interface demanded by the product requirements?**
  Yes. The interface is completely specialized. It is not based on anything other than every other Microsoft Windows application out. The GUI is completely our design and no other application out (to our knowledge) contains exactly what is expected of our software.

- **Do requirements for the product demand the creation of program components that are unlike any previously developed by your organization?**
  Yes. The entire GUI is composed of subsystems that our VB software engineer has never had experience with**.**

- **Do requirements demand the use of new analysis, design, or testing methods?**
  No. The development team will implement existing analysis, design, and testing methods for the project.

- **Do requirements demand the use of unconventional software development methods?**
  No. Property Management uses C++ code in header files, which is not unconventional. It also integrates with Visual Basic, which is not unconventional.

- **Do requirements put excessive performance constraints on the product?**
  Yes since 3D viewing feature push the limit of computer systems, the DirectX engine has to be efficient enough to handle large numbers of logical calculations and fast memory accessing techniques to handle the 30fps limit.

- **Is the customer uncertain that the functionality required is "doable"?**
  No. The customer has full confidence in the project as described in the System Specification Document and the Software Specification Document.

Development Environment Risks

- **Is a software project management tool available?**
  No. No software tools are to be used. Due to the existing deadline, the development team felt it would be more productive to begin implementing the project than trying to learn new software tools. After the completion of the project software tools may be implemented for future projects.

- **Is a software process management tool available?**
  No. No software tools are to be used. Due to the existing deadline, the development team felt it would be more productive to begin implementing the project than trying to learn new software tools. After the completion of the project software tools may be implemented for future projects.

- **Are tools for analysis and design available?**
  No. No software tools are to be used. Due to the existing deadline, the development team felt it would be more productive to begin implementing the project than trying to learn new software tools. After the completion of the project software tools may be implemented for future projects.

- **Do analysis and design tools deliver methods that are appropriate for the product to be built?**
  N/A. No analysis or design tools are to be used.

- **Are compilers or code generators available and appropriate for the product to be built?**
  Yes. Microsoft Visual C++ and Microsoft Visual Basic will be used to build Property Management.

- **Are testing tools available and appropriate for the product to be built?**
  No. No software tools are to be used. Due to the existing deadline, the development team felt it would be more productive to begin implementing the project than trying to learn new software tools. After the completion of the project software tools may be implemented for future projects.

- **Are software configuration management tools available?**
  No. No software tools are to be used. Due to the existing deadline, the development team felt it would be more productive to begin implementing the project than trying to learn new software tools. After the completion of the project software tools may be implemented for future projects.

- **Does the environment make use of database or a repository?**
  Yes. DirectX/C++ code is stored in header files. All other necessary files are stored in a Microsoft Access database.

- **Are software tools integrated with one another?**
  N/A. No software tools are to be used therefore they cannot be Integrated.

- **Have members of the project team received training in each of the tools?**
  N/A. No software tools are to be used. Due to the existing deadline, the development team felt it would be more productive to begin implementing the project than trying to learn new software tools. After the completion of the project software tools may be implemented for future projects.

- **Are local experts available to answer questions about the tools?**
  N/A. No software tools are to be used. Local experts will not be necessary.

- **Is on-line help and documentation for the tools adequate?**
  N/A. No software tools are to be used. Online help will not be necessary.

Staff Size and Experience Risks

- **Are the best people available?**
  Yes. PA Software has assembled a team of the most qualified software engineers to implement the Property Management project.

- **Do the people have the right combination of skills?**
  Yes. The team members have experience in C++, Visual Basic, DirectX, Microsoft Access, and software development skills.

- **Are enough people available?**
  Yes. Though the team is small, a larger team could take away from the productivity due to increased lines of communication.

- **Are staff committed for entire duration of the project?**
  Yes. Any staff members that do not complete the project will not receive a grade for the course.

- **Will some project staff be working only part time on this project?**
  No. All staff members will be working on the project for the duration.

- **Does staff have the right expectations about the job at hand?**

Yes. All team members understand what is required to complete the project, and are committed to accomplishing them.

- **Has staff received necessary training?**
  Yes. Team members are familiar with game programming and software engineering techniques.

- **Will turnover among staff be low enough to allow continuity?**
  Yes. Due to the nature of the staff, there will be no staff turnover.

Risk Refinement: At various points in the checklist, lack of software tools is identified as a potential risk. Due to time constraints, the members of the design team felt that searching forand learning to use additional software tools could be detrimental to the project, as it would take time away from project development. For this reason, we have decided to forgo the use of software tools. It will not be explored as a potential riskbecause all planning will be done without considering their use.

Risk Mitigation, Monitoring and Management

Risk: Computer Crash

- **Mitigation**
  The cost associated with a computer crash resulting in a loss of data is crucial. A computer crash itself is not crucial, but rather the loss of data. A loss of data will result in not being able to deliver the product to the customer. This will result in a not receiving a letter of acceptance from the customer. Without the letter of acceptance, the group will receive a failing grade for the course. As a result the organization is taking steps to make multiple backup copies of the software in development and all documentation associated with it, in multiple locations.

- **Monitoring**
  When working on the product or documentation, the staff member should always be aware of the stability of the computing environment they're working in. Any changes in the stability of the environment should be recognized and taken seriously.

- **Management**
  The lack of a stable-computing environment is extremely hazardous to a software development team. In the event that the computing environment is found unstable, the development team should cease work on that system until the environment is made stable again, or should move to a system that is stable and continue working there

Risk: Late Delivery

- **Mitigation**
  The cost associated with a late delivery is critical. A late delivery will result in a late delivery of a letter of acceptance from the customer. Without the letter of acceptance, the group will receive a failing grade for the course. Steps have been taken to ensure a timely delivery by gauging the scope of project based on the delivery deadline.

- **Monitoring**
  A schedule has been established to monitor project status. Falling behind schedule would indicate a potential for late delivery. The schedule will be followed closely during all development stages.

- **Management**
  Late delivery would be a catastrophic failure in the project development. If the project cannot be delivered on time the development team will not pass the course. If it becomes apparent that the project will not be completed on time, the only course of action available would be to request an extension to the deadline form the customer.

Risk: Technology Does Not Meet Specifications

- **Mitigation**
  In order to prevent this from happening, meetings (formal and informal) will be held with the customer on a routine business. This insures that the product we are producing, and the specifications of the customer are equivalent.

- **Monitoring**
  The meetings with the customer should ensure that the customer and our organization understand each other and the requirements for the product.

- **Management**
  Should the development team come to the realization that their idea of the product specifications differs from those of the customer, the customer should be immediately notified and whatever steps necessary to rectify this problem should be done. Preferably a meeting should be held between the development team and the customer to discuss at length this issue.

Risk: End Users Resist System

- **Mitigation**
  In order to prevent this from happening, the software will be developed with the end user in mind. The user-interface will be designed in a way to make use of the program convenient and pleasurable.

- **Monitoring**
  The software will be developed with the end user in mind. The development team will ask the opinion of various outside sources throughout the development phases. Specifically the user-interface developer will be sure to get a thorough opinion from others.

- **Management**
  Should the program be resisted by the end user, the program will be thoroughly examined to find the reasons that this is so. Specifically the user interface will be investigated and if necessary, revamped into a solution.

Risk: Changes in Requirements

- **Mitigation**
  In order to prevent this from happening, meetings (formal and informal) will be held with the customer on a routine business. This insures that the product we are producing, and the requirements of the customer are equivalent.

- **Monitoring**
  The meetings with the customer should ensure that the customer and our organization understand each other and the requirements for the product.

- **Management**
  Should the development team come to the realization that their idea of the product requirements differs from those of the customer, the customer should be immediately notified and whatever steps necessary to rectify this problem should be taken. Preferably a meeting should be held between the development team and the customer to discuss at length this issue.

Risk: Lack of Development Experience

- **Mitigation**
  In order to prevent this from happening, the development team will be required to learn the languages and techniques necessary to develop this software. The member of the team that is the most experienced in a particular facet of the development tools will need to instruct those who are not as well versed.

- **Monitoring**
  Each member of the team should watch and see areas where another team member may be weak. Also if one of the members is weak in a particular area it should be brought to the attention by that member, to the other members.

- **Management**
  The members who have the most experience in a particular area will be required to help those who don't out should it come to the attention of the team that a particular member needs help.

Risk: Database is not Stable

- **Mitigation**
  In order to prevent this from happening, developers who are in contact with the database, and/or use functions that interact with the database, should keep in mind the possible errors that could be caused due to poor programming/error checking. These issues should be brought to the attention of each of the other members that are also in contact with the database.

- **Monitoring**
  Each user should be sure that the database is left in the condition it was before it was touched, to identify possible problems. The first notice of database errors should be brought to the attention of the other team members.

- **Management**
  Should this occur, the organization would call a meeting and discuss the causes of the database instability, along with possible solutions.

Risk: Poor Quality Documentation

- **Mitigation**
  In order to prevent this from happening, members who are in charge of developing the documentation will keep in contact with each developer on the team. Meetings will be held routinely to offer documentation suggestions and topics. Any topic deemed missing by a particular developer will be discussed and

it will be decided whether or not to add that particular topic to the documentation. In addition, beta testers will be questioned about their opinion of the documentation.

- **Monitoring**
  Throughout development or normal in and out of house testing, the development team and or beta testers will need to keep their eyes open for any possible documentation topics that have not been included.

- **Management**
  Should this occur, the organization would call a meeting and discuss the addition of new topics, or removal of unnecessary topics into the documentation.

Risk: Deviation from Software Engineering Standards

- **Mitigation**
  While it is possible to deviate from software engineering standards, it is unlikely to occur. All team members have a full understanding of the software process, and how we plan to implement them in the process.

- **Monitoring**
  Technical reviews involving comparison between documentation and the actual project will help to determine if deviation will occur. All relevant documents must be as complete and accurate as possible to ensure that work will conform to expressed software engineering standards.

- **Management**
  Should deviation occur, steps must be taken to guide the project back within the standards expressed in accompanying documents. Technical reviews help to determine what must be done to keep the project in line with established software engineering standards.

Risk: Poor Comments in Code

- **Mitigation**
  Poor code commenting can be minimized if commenting standards are better expressed. While standards have been discussed informally, no formal standard yet exists. A formal written standard must be established to ensure quality of comments in all code.

- **Monitoring**
  Reviews of code, with special attention given to comments will determine if they are up to standard. This must be done frequently enough to control comment quality. If they are not done comment quality could drop, resulting in code that is difficult to maintain and update.

- **Management**
  Should code comment quality begin to drop, time must be made available to bring comments up to standard. Careful monitoring will minimize the impact of poor commenting. Any problems are resolved by adding and refining comments as necessary.

Risk Table

Risks	Category	Probability	Impact
Computer Crash	TI	30%	1
Less reuse then planned	DE	60%	2
Changes in requirements	PS	55%	2
Late Delivery	BU	30%	1
Technology will not Meet Expectations	TE	25%	1
End Users Resist System	BU	20%	1
Lack of Development Experience	TI	20%	2
Lack of Database Stability	TI	40%	2
Poor Quality Documentation	BU	35%	2
Deviation from Software Engineering Standards	PI	10%	3
Poor Comments in Code	TI	20%	4

RISK INFORMATION SHEET (RIS):

Risk Information Sheet			
Risk ID: P06-758	Date:03/05/2023	Prob: 65%	Impact: Marginal
Description: The software used to maintain was developed for the initial systems and is still proprietary. Due to this, many methods might be deprecated hence exposing the system to newer software vulnerabilities. The major part of requirements will be same only 30% might change so we have kept impact as marginal.			
Refinement/context: Sub condition 1: Certain components were developed by a third party with no knowledge of newer software vulnerabilities. Sub condition 2: Certain reusable components have not been coded in a language which does not directly support the newer custom features. Sub condition 3: Requirements can be realized later in development phase.			
Mitigation/monitoring: 1. Contact third party to determine conformance with the proposed design standards. 2. Consider component structure when deciding on interface protocol. 3. If the resources required for the requirement is available before the final version release we can implement it.			
Management/contingency plan/trigger: Allocate the computed amount within project contingency cost. After further calculation it has been estimated that 10 other requirements need to be created out of which 3 can be implemented in the current version and others will be implanted in future with the available resourses. Trigger: Mitigation steps unproductive as 15/02/2022.			
Current status: 03/05/2023 Mitigation steps initiated.			
Originator: Kiran Bhowmick		Assigned: Ayush Jain	

Conclusion:

Hence, we have successfully prepared a RMMM plan for our project –ShopIT and learnt that RMMM is an activity used for project tracking. It has the following primary objectives as follows. To check if predicted risks occur or not. To ensure proper application of risk aversion steps defined for risk but it incurs additional project costs. It takes additional time. For larger projects, implementing an RMMM may itself turn out to be another tedious project. RMMM does not guarantee a risk-free project, infact, risks may also come up after the project is delivered.

Experiment No. 10

Dhruv Bheda - 60004200102

Sahej Jain - 60004200111

Ayush Jain - 60004200132

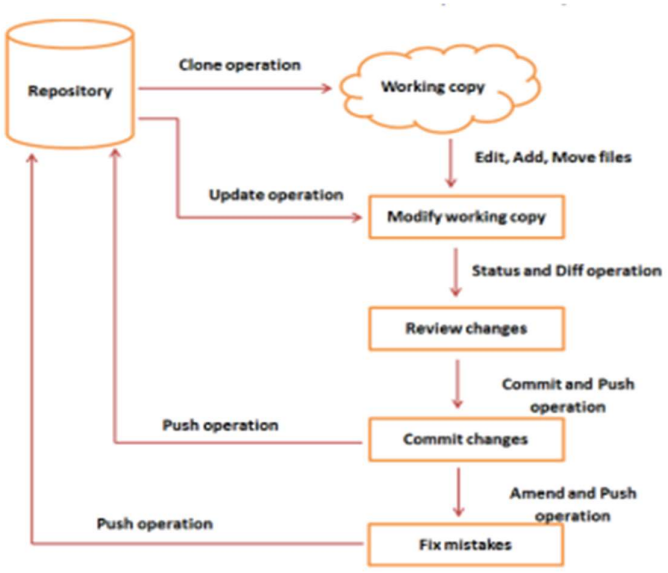
Varun Vekaria – 60004200167

Aim: Study of Configuration Management using GitHub

Theory: Git is a distributed revision control and source code management system with an emphasis on speed. Git was initially designed and developed by Linus Torvalds for Linux kernel development. Git is a free software distributed under the terms of the GNU General Public License version 2.

Git Life Cycle General workflow is as follows –

1. Clone the Git repository as a working copy.
2. Modify the working copy by adding/editing files.
3. If necessary, update the working copy by taking other developer's changes.
4. Review the changes before committing.
5. Commit changes. If everything is fine, then push the changes to the repository.
6. After committing, if something is wrong, then correct the last commit and push the changes to the repository.



Git Life Cycle

1. Git command to create an empty repository and add a text file into it.

```
C:\Users\Admin>cd ../..

C:\>cd .vscode

C:\.vscode>cd college

C:\.vscode\college>cd experiments

C:\.vscode\college\experiments>git clone https://github.com/Greninja28/SE-10.git
Cloning into 'SE-10'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

C:\.vscode\college\experiments>cd SE-10

C:\.vscode\college\experiments\SE-10>code .

C:\.vscode\college\experiments\SE-10>
```

2. Add and commit the new file into GitHub and check git status and git log

```
PS C:\.vscode\college\experiments\SE-10> git add .
PS C:\.vscode\college\experiments\SE-10> git status -s
M  start.py
PS C:\.vscode\college\experiments\SE-10> git commit -m 'file changed'
[main 8992e87] file changed
 1 file changed, 1 insertion(+)
PS C:\.vscode\college\experiments\SE-10> git status -s
PS C:\.vscode\college\experiments\SE-10> git log
commit 8992e871e8f52ea791ea992f0dcd717ec04516fb (HEAD -> main)
Author: Sahej Jain <89766122+Greninja28@users.noreply.github.com>
Date:   Wed May 3 09:20:21 2023 +0530

    file changed

commit 2accd86c64e87864ffb442b5e68bb33432cb82bc (origin/main, origin/HEAD)
Author: Sahej Jain <89766122+Greninja28@users.noreply.github.com>
Date:   Wed May 3 09:16:49 2023 +0530

    Initail commit
```

3. Git push and change the content and pull from git

```
PS C:\.vscode\college\experiments\SE-10> git pull origin main
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused
Unpacking objects: 100% (3/3), 665 bytes | 41.00 KiB/s, done.
From https://github.com/Greninja28/SE-10
 * branch            main       -> FETCH_HEAD
   8992e87..7ff546c  main       -> origin/main
Updating 8992e87..7ff546c
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
PS C:\.vscode\college\experiments\SE-10> git add .
PS C:\.vscode\college\experiments\SE-10> git commit -m 'Added a sum program'
[main f20de92] Added a sum program
 1 file changed, 3 insertions(+), 2 deletions(-)
PS C:\.vscode\college\experiments\SE-10> git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 343 bytes | 171.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Greninja28/SE-10.git
   7ff546c..f20de92  main -> main
PS C:\.vscode\college\experiments\SE-10>
```

Conclusion: Thus, we have created a repository on GitHub and added files onto it.

SE Assignment 1

Page No.
Date : / /

Name : Ayush Jain
Sap Id : 60004200132
Div : B

1 Elaborate the task set for creating Component Level Design in OO Projects

Ans :

- Component level design is elaborative in nature
- It transforms information from requirements and architectural models into a design representation that provides sufficient detail to guide the construction (coding and testing) activity
- The following steps represent a typical task set for component level design when applied to object oriented Projects:-

STEP 1: Identify all design classes that correspond to the Problem domain

-> using the requirements and architectural model each analysis class and architectural component is

STEP 2: Identify all design classes that correspond to the infrastructure domain

->These classes are not described in the requirements model and are often missing from the architecture model but they must be described at this point.

->classes and components in this category include :

GUI components

Operating system components

object and data management components.

Scanned with CamScanner

Page No.
Date : / /

STEP 3: Elaborate all design classes that aren't acquired as reusable components
-> Elaboration requires that all interfaces attributes and operations necessary to implement the class be described in detail.
Design heuristics [eg. component cohesion and coupling]

STEP 3(a): Specific message details when classes or components collaborate.
->The requirements model makes use of a collaboration diagram to show how analysis classes collaborate with one another.
STEP 3(b): Identify appropriate interfaces for each
->within the context of component-level design a UML Interface group of Public [externally visible] operations.
->the interface contains no internal structure has no attributes or associations

STEP 3(c): Elaborate attributes and define data types and data structures required to implement them.
-> In general data structures and types used to define attributes are defined within the content Programming language that is used for the implementation.
->UML defines an attributes data type using the following syntax:
name: type-expression initial-value {property string}
->Here name is the attribute name type expression is the data type initial value is the value that the attribute takes when an object is created and

Scanned with CamScanner

Page No.
Date : / /

Property string defines a property or characteristic of the attribute

STEP 3 (D): Describe Processing flow within each operation in detail
->This may be accomplished using a Programming language-based Pseudocode or with a UML activity
-> Each software component is elaborated through number of iterations that apply the refinement
STEP 4: Describe Persistent data sources [databases and files] and identify the classes required to them
->Databases and files normally transcend the design description of manage them. an individual component In most cases these Persistent data stores are initially specified as Part of architectural design

STEP 5: Develop and elaborate behavorial representations for a class or component
-> UML State diagrams were used as a Part of requirements model to represent the externally observable behaviours of the system
STEP 6: Elaborate deployment diagrams to Provide additional implementational detail Deployment diagrams are used as Part of architecture
STEP 7: Refractor every component-level design representation and always consider alternatives

2. Explain the golden rules of user Interface Design
Ans: User Interface Design creates an effective communication medium between a human and a

Scanned with CamScanner

Page No.
Date : / /

computer following a set of interface design Principles design identifies interface objects and actions and then creates a screen layout that forms the basics for a user interface Prototype

The Golden Rules of User - Interface Design are :-
(i) Place the user in control
a) Design interaction modes in a way that does not force a user into unnecessary or undesired actions
->An interaction mode is the current state of the interface For example if spell check is selected in a word Processor menu the software moves to a spell-checking mode The user should be able to enter and exit the mode with little or more effort

b)Provides flexible interaction
->Because different users have different interaction Preferences choices should be Provided
->For example software might allow a user to interact via keyboard commands mouse movements digitizer Pen a multi-touch screen recognition commands However not every action is amenable to every interaction mechanism
For example There will be difficulty in using keyboard Commands or voice inputs to draw a complex shape
c) allow user interaction to be interruptible and undoableEven when involved in a sequence of actions the user should be able to interrupt the sequence to do something else.
->The user should also be able to "undo" any action

Scanned with CamScanner

Page No.
Date : / /

d) Streamline interaction as skill levels advance and allow the interaction to be customized
-> users often find that they perform the same sequence of interactions repeatedly.
-> It is worth while to design a "macro" mechanism that enables an advanced user to customise the Interface to facilitate interaction.

(ii) Reduce the user's memory load.
-> The more a user has to remember the more error Prone the interaction with the system will be. It is for the reason that a well-designed user interface doesn't tax the user memory.
a) Reduce demand on short-term memory.
-> when users are involved in complex tasks memory can be significant. The interface should be designed to reduce the requirement to remember Past actions inputs and results
-> This can be accomplished by Providing visual cues that enable a user to recognise Past actions rather than recalling them
b) Establish meaningful defaults
-> The initial set of defaults should make sense to the average user but he should be able to specify his individual Preferences as well However a "reset" option should be available enabling the redefination of original default values

c) Define shortcuts that are intuitive.
-> when mnemonics are used to accomplish a system function (eg alt + P to invoke the Print function)

Scanned with CamScanner

the mnemonic should be tied to the action in a way that is easy to remember.

d) The visual layout of the interface should be based on a real-world metaphor.

->For example a bill payment system should use a checkout and check register metaphor guide the user through the bill paying process.

->This enables the user to rely on well understood visual cues rather than memorizing an arcane interaction sequence.

d) Disclose information in a progressive fashion.

->The interface should be organized hierarchically.

->For example in word processing applications underlining function is available in the style menu when the user picks it all underlying options such as single underline double underline dashed underline are presented.

(iii) Make the interface consistent.

->The interface should present and acquire information in a consistent fashion.

This implies that:

(1) All visual information is organized according to design screen rules that are maintained throughout all screen displays

(2) Input mechanisms are constrained to a limited set that is used consistently throughout the

(3) Mechanisms for navigating from task to task are consistently defined and implemented.

Design Principles :-

Scanned with CamScanner

Page No.
Date : / /

a-> Allow the user to put the current task into a meaning meaningful context.
->Many interfaces implement complex layers of interactions with dozens of screen images.
->It is important to Provide indicators [eg. window titles graphical icons the user Consistent color coding] that enables to know the content of the work at hand.

b) Maintain consistency across a family of applications.
->A set of applications (or Products) should all implement the same design rules so that consistency is maintained for all interactions.

c) If Past interactive models have created user expectations do not make changes unless there is a compelling reason to do so.
->For example once a Particular interactive sequence has become a defacto standard [eg: use of ctrl + V for Pasting] the user expects this in every application he encounters A change [eg. using alt +V to invoke Pasting] will cause confusion.

Scanned with CamScanner

Name : Ayush Jain
SAP ID : 60004200182
Div : B
Computer Science

SE - Assignment 2

Q. 1) Explain equivalence partitioning and boundary value analysis?

→ **Equivalence Partitioning:**

1) It is a software technique that involves dividing the input data of a software applications into groups, or partitions, that are expected to exhibit similar behaviour. The idea behind this technique is to reduce the no. of test cases required to adequately test a system while still ensuring that all possible scenarios are covered.

2) In equivalence partitioning, each partition is tested using a representative test case from that partition. The goal is to ensure that the software is tested thoroughly, without unnecessary duplication of test cases.

3) For eg: if a software application accepts a user's age and input, we can partition the i/p into three groups: ages below 18, ages b/w 18 and 65, and ages above 65. We would then test each partition with an i/p value that is representative of that partition.

Boundary value Analysis:

1) Boundary value analysis is a software testing tech that involves testing the boundaries b/w different i/p data partitions. The goal is to ensure that the software handles boundary conditions correctly, which are often where errors are most likely to occur.

2) For eg: if a software app accepts a users age as i/p, we can test the boundary value for each partition. For ages below 18, we would test with values such as 17 and 18

Sundaram
FOR EDUCATIONAL USE

Scanned with CamScanner

for ages b/w 18 and 65, we would test with values such as 18, 19, 64, 65. For ages above 65, we would test with values such as 65 and 66.

3) By testing the boundaries, we can ensure that the software handles edge cases correctly and that is robust to unexpected i/p values.

Q. 2> With a suitable example, explain OAT.

→ 1) Operational Acceptance Testing (OAT) is a type of software testing that is carried out to ensure that a software system or application is ready to be used in a production environment. OAT is typically conducted by end-users or business stakeholders who will be using the software in real-world scenarios.

2) An eg of OAT is testing a banking application that has been developed to handle online transactions. Before the application is launched, the bank would conduct OAT to ensure that the app can handle the expected level of traffic, user concurrency, and transaction volumes in real word scenarios.

3) During the OAT process, the bank would simulate various use cases, such as logging in to the system, checking account balances, making transactions, and logging out of the systems. The bank would also test the application's response time, security features, and error handling capabilities.

Sundaram
FOR EDUCATIONAL USE

Scanned with CamScanner

4) Once the OAT process is complete, the bank can be confident that the application is ready to be used by customers, and that it will provide a reliable and secure platform for online banking transactions.

Q. 3) Explain version control in SCM.

→ 1) Version control is an essential part of software configuration management (SCM). It is the process of tracking changes to source code, documentation, and other files over time.
2) The primary goal of version control is to allow multiple developers to work on the same project simultaneously without interfering with each other's work.
3) VCS provides a way to manage changes to files over time. The VCS maintains a history of all changes made to file, allowing developers to track the evolution of the codebase and roll back to earlier versions if needed.
4) There are two main types of version control system: centralized and distributed. Centralized control system, such as Subversion, store all files and their history on a control server. Distributed version control system, such as GIT allow developers to maintain their own local copies of the entire codebases, including its history.
5) Version control system offer several benifits to software development teams including: (1) collaboration (2) Versioning. (3) Branching and merging. (4) Traceability (5) Continuous integration
6) Overall, version control is a critical component of software development, providing developers with the tools they need to collaborate effectively, manage changes and maintain integrity.

FOR EDUCATIONAL USE

Scanned with CamScanner