

F25-314-D-CoWriteIA

Project Team

Aisha Siddiqa	22I-1281
Ayesha Ejaz	22I-0899
Junaid Asrar	22I-0770

Session 2022-2026

Supervised by

Dr. Ali Zeeshan



Department of Computer Science

**National University of Computer and Emerging Sciences
Islamabad, Pakistan**

June, 2026

Contents

1	Introduction	1
1.1	Existing Solutions	1
1.2	Problem Statement	2
1.3	Scope	2
1.4	Modules	3
1.4.1	Module 1: Project Indexing and Semantic Memory	3
1.4.2	Module 2: Context-Aware Writing Assistant	3
1.4.3	Module 3: Character and Scene Management	3
1.4.4	Module 4: Dialogue Generation	3
1.4.5	Module 5: Research Integration	4
1.4.6	Module 6: Project Query Interface	4
1.4.7	Module 7: Gap Analysis Module	4
1.5	Work Division	5
2	Project Requirements [AS PER FYP1 MID REPORT]	7
2.1	Use-case/Event Response Table/Storyboarding	7
2.2	Functional Requirements	8
2.2.1	Module 1	8
2.2.2	Module 2	8
2.3	Non-Functional Requirements	8
2.3.1	Usability	8
2.3.2	Performance	9
3	System Overview [AS PER FYP1 MID REPORT]	11
3.1	Architectural Design	11
3.2	Data Design	12
3.3	Domain Model	12
3.4	Design Models [UPTO THE CURRENT ITERATION ONLY]	13
4	Implementation and Testing [UPTO THE CURRENT ITERATION ONLY]	15
4.1	Algorithm Design	15
4.2	External APIs/SDKs	16
4.3	Testing Details	16

4.3.1	Unit Testing	16
References		17
A Appendices		19
A.1	Appendix A	19
A.1.1	Use Case Diagram example (Online Shopping System)	19
A.1.2	Detail Use Case Example	20
A.1.3	Event-Response Table for a Highway Intersection	21
A.1.4	Story Board Example For Android App	22
A.2	Appendix B	23
A.2.1	Domain Model Example For Online Shopping	23
A.3	Appendix C	24
A.3.1	Box And Line Example For Online Shopping	24
A.3.2	Architecture Pattern Example For Online Shopping	25
A.4	Appendix D	26
A.4.1	Activity Diagram	26
A.4.2	Class Diagram	27
A.4.3	Sequence Diagram	28
A.4.4	State Transition Diagram	28
A.4.5	Data Flow Diagram	29

List of Figures

4.1	Example Of Algorithm Design	15
4.2	Example for Unit Testing	16
A.1	Use Case Diagram for the Online Shopping System	19
A.2	Detail Use Case Example	20
A.3	Example of Event Response Table	21
A.4	Example of Story Board	22
A.5	Domain Model Example For Online Shopping Application	23
A.6	Box and Line Diagram For Online Shopping Application	24
A.7	Architecture Pattern For Online Shopping Application	25
A.8	Activity Diagram For Online Shopping Application	26
A.9	Class Diagram For Online Shopping Application	27
A.10	Sequence Diagram For Online Shopping Application	28
A.11	State Transition Diagram For Online Shopping Application	28

List of Tables

1.1	Comparison of Existing Solutions	2
1.2	Work Division for Iteration I	5
1.3	Work Division for Iteration II	5

Chapter 1

Introduction

CoWriteIA is an AI-powered writing assistant designed to support creative writers, novelists, researchers, and content creators. Modern writing workflows require managing notes, drafts, characters, scenes, and references across several disconnected tools, which often breaks focus and reduces productivity. Writers struggle to maintain narrative consistency, track earlier ideas, and keep a coherent writing style when working on long projects. Existing tools offer only isolated features such as grammar correction or basic generation and do not provide project-level understanding or semantic memory [1].

CoWriteIA addresses these issues by creating a unified, intelligent workspace where all project files are indexed, searchable, and semantically connected. By integrating project-level memory, context-aware writing, dialogue generation, character management, and research support, the system helps writers maintain consistency and improve their creative process. The platform is designed to reduce cognitive load, avoid fragmented workflows, and provide meaningful AI assistance throughout the writing journey. This chapter presents the background, existing solutions, problem statement, scope, project modules, and work division that form the foundation of this system.

1.1 Existing Solutions

Several writing and AI-assisted tools exist, but each focuses on limited aspects of the writing process. Grammarly offers grammar correction and style suggestions but lacks deep contextual awareness across long projects. Notion AI and Jasper AI provide generative assistance but do not maintain story-level continuity or user-specific writing style. Tools like Scrivener help with organization but have no semantic understanding or AI memory. As a result, writers must repeatedly switch between applications to manage notes, drafts, characters, and research, leading to inefficiency and inconsistent writing flow.

Table 1.1: Comparison of Existing Solutions

System Name	System Overview	System Limitations
Grammarly	Provides grammar correction, clarity improvements, and tone suggestions.	No project awareness, no memory of earlier chapters, no semantic search.
Notion AI	Offers AI-assisted content generation and note organization.	Cannot maintain narrative consistency; lacks dialogue and character support.
Scrivener	Strong organizational tool for large writing projects with chapter/scene structure.	No AI support, no semantic retrieval, no automated gap or style analysis.

1.2 Problem Statement

Writers working on long-form projects often lose track of earlier ideas, character traits, plot points, and stylistic decisions. This leads to inconsistencies, repeated ideas, and a break in narrative flow. Existing tools either provide isolated writing support or document organization but do not combine both with meaningful context. AI tools can generate text but fail to maintain project-level continuity, making the generated text feel disconnected from the writer’s established style or storyline. Writers spend significant time searching through older drafts to recall information [1].

CoWriteIA aims to solve these problems by offering an intelligent system that continuously indexes all project content, retrieves relevant context, and assists writers in generating text that aligns with their established narrative and writing style. By maintaining a unified knowledge base and supporting character consistency, scene management, dialogue generation, and semantic search, the system reduces cognitive overhead and improves creative flow.

1.3 Scope

The scope of CoWriteIA includes building an AI-driven writing environment that supports project management, semantic search, context-aware writing, dialogue generation, research integration, and style adaptation. The system will allow users to upload documents, create new content, store character information, generate dialogues, and retrieve context from a vector-based semantic memory [1]. It will support long-form writing projects such as novels, research documents, and story-driven content.

The system will not include plagiarism detection, multimedia editing, or full publishing

workflows. It focuses strictly on improving the writing process, maintaining consistency, and providing intelligent assistance throughout the creative workflow.

1.4 Modules

The project consists of several modules, each responsible for a unique part of the writing workflow.

1.4.1 Module 1: Project Indexing and Semantic Memory

This module extracts, embeds, and organizes all project content into a semantic database for context-aware retrieval [1].

1. Automatic project indexing and embedding generation.
2. Semantic search based on meaning instead of keywords.

1.4.2 Module 2: Context-Aware Writing Assistant

This module provides intelligent writing suggestions that match the user's tone and project context [1].

1. Generates coherent drafts aligned with past content.
2. Retrieves relevant information to maintain consistency.

1.4.3 Module 3: Character and Scene Management

This module stores and manages characters, scenes, and narrative details.

1. Character profiles with traits and relationships.
2. Scene storage and tracking.

1.4.4 Module 4: Dialogue Generation

Generates natural, character-consistent dialogue suggestions.

1. Dialogue generation based on personality and context.
2. Supports narrative flow within scenes.

1.4.5 Module 5: Research Integration

Fetches factual data from external sources for realistic writing.

1. Web-based research retrieval.
2. Insertable factual references.

1.4.6 Module 6: Project Query Interface

Allows users to ask natural-language questions about their own project.

1. Semantic question-answering.
2. Source-linked responses.

1.4.7 Module 7: Gap Analysis Module

Analyzes draft content to find missing or weak areas.

1. Identifies incomplete sections.
2. Suggests improvements for clarity and consistency.

1.5 Work Division

The work completed during FYP-1 was divided into two major iterations. A summary of responsibilities is presented in the tables below.

Iteration I Tasks

Table 1.2: Work Division for Iteration I

Task	Ayesha	Junaid	Aisha
SRS Document	✓	✓	✓
UML Diagrams	✓	✓	✓
UI Design		✓	✓
Database Connection		✓	✓
Frontend (Main Pages)	✓		
Project Indexing Agent	✓	✓	
Knowledge Retrieval Agent		✓	✓

Iteration II Tasks

Table 1.3: Work Division for Iteration II

Task	Ayesha	Junaid	Aisha
Inline Copilot Support	✓		
Context-Aware Writing Module		✓	
Gap Analysis Module			✓
Documentation (All Sections)	✓	✓	✓

Chapter 2

Project Requirements [AS PER FYP1 MID REPORT]

This section outlines the necessary requirements for the successful completion of the project. Project requirements can be divided into two main categories: functional requirements, which describe the system's core operations, and non-functional requirements, which specify performance, security, and usability standards.

2.1 Use-case/Event Response Table/Storyboarding

This section describes how users will interact with the system through various scenarios, often referred to as use cases or event responses.

A use case represents a specific interaction between the user and the system, detailing the steps involved and the expected system responses. Each use case typically includes key components such as a unique identifier, a description of the interaction, the user or system actor involved, preconditions, the main flow of events, and postconditions.

Additionally, storyboarding can be used as a visual aid to depict the sequence of actions, illustrating how the user progresses through the system step by step. Storyboards are helpful for capturing the user experience and ensuring that the design aligns with the intended functionality.

Depending on the project, use cases, storyboarding or Event-response table can be chosen as appropriate methodologies to illustrate user interactions. Examples of all the approaches are provided in Appendix A.

2.2 Functional Requirements

This section describes the functional requirements of the system expressed in the natural language style. This section is typically organized by feature as a system feature name and specific functional requirements associated with this feature.

2.2.1 Module 1

Following are the requirements for module 1:

1. FR1
2. FR2

2.2.2 Module 2

Following are the requirements for module 2:

1. FR1
2. FR2

2.3 Non-Functional Requirements

This section specifies nonfunctional requirements. These quality requirements should be specific, quantitative, and verifiable. The following are some examples of documenting guidelines.

2.3.1 Usability

Usability requirements deal with ease of learning, ease of use, error avoidance and recovery, the efficiency of interactions, and accessibility. The usability requirements specified here will help the user interface designer create the optimum user experience. Example:

USE-1: The COS shall allow a user to retrieve the previous meal ordered with a single interaction.

2.3.2 Performance

State specific performance requirements for various system operations. If different functional requirements or features have different performance requirements, it's appropriate to specify those performance goals right with the corresponding functional requirements, rather than collecting them in this section. Example:

PER-1: 95% of webpages generated by the COS shall download completely within 4 seconds from the time the user requests the page over a 20 Mbps or faster Internet connection.

Chapter 3

System Overview [AS PER FYP1 MID REPORT]

Give a general description of the functionality, context, and design of your project. Provide any background information if necessary.

3.1 Architectural Design

Develop a modular program structure and explain the relationships between the modules to achieve the complete functionality of the system. This is a high-level overview of how the system's modules collaborate with each other to achieve the desired functionality.

Don't go into too much detail about the individual subsystems. The main purpose is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together.

Provide a diagram showing the major subsystems and their connections.

- In the initial design stage, create a Box and Line Diagram for a simpler representation of the systems.
- After finalizing the architecture style/pattern diagram (MVC, Client-Server, Layered, Multi-tiered), create a detailed mapping of modules/components to each part of the architecture.

To view examples of box and line diagrams and architecture styles, see Appendix C.

3.2 Data Design

Explain how the information domain of your system is transformed into data structures. Describe how the major data or system entities are stored, processed, and organized.

List any databases or data storage items.

3.3 Domain Model

The domain model is a conceptual representation of the various entities, their attributes, and the relationships between them within the system. It serves as a bridge between the requirements and the design of the software, providing a clear understanding of the business domain and how it interacts with the system being developed.

A domain model typically includes key components such as:

- **Entities:** These are the primary objects within the domain that hold data and represent real-world concepts. Each entity is defined by its attributes and behaviors.
- **Attributes:** Characteristics or properties of the entities that define their state. For example, a "Customer" entity might have attributes such as `customerID`, `name`, `email`, and `address`.
- **Relationships:** These illustrate how different entities interact with each other. Relationships can be one-to-one, one-to-many, or many-to-many, depending on how entities are associated.
- **Associations:** Connections between entities that help clarify their interdependencies. For instance, a "Product" may be associated with a "Category," indicating the type of product it belongs to.

By constructing a domain model, developers can better understand the system's requirements, facilitate communication among stakeholders, and guide the subsequent design and implementation phases. It also aids in identifying potential issues early in the development process, ensuring a more robust architecture.

An example of a domain model for an online shopping system is provided in Appendix B.

3.4 Design Models [UPTO THE CURRENT ITERATION ONLY]

Create design models as are applicable to your system. Provide detailed descriptions with each of the models that you add. Also ensure visibility of all diagrams.

Design Models for Object Oriented Development Approach

The applicable models for the project using object oriented development approach may include:

- Activity Diagram
- Class Diagram
- Class-level Sequence Diagram
- [OPTIONAL] State Transition Diagram (for the projects which include event handling and backend processes)

Design Models for Procedural Approach

The applicable models for the project using procedural approach may include:

- Activity Diagram
- Data Flow Diagram (data flow diagram should be extended to 2-3 levels. It should clearly list all processes, their sources/sinks and data stores.)
- System-level Sequence Diagram
- [OPTIONAL] State Transition Diagram (for the projects which include event handling and backend processes)

Examples of above diagrams are given in Appendix D.

Chapter 4

Implementation and Testing [UPTO THE CURRENT ITERATION ONLY]

Give a general description of the functionality, context, and design of your project. Provide any background information if necessary.

4.1 Algorithm Design

Mention the algorithm(s) used in your project to get the work done with regards to major modules. Provide a pseudocode explanation regarding the functioning of the core features. Following are few examples of algorithms/pseudocode.

Example:

Algorithm 1 MHCF co-authorsBasedClustering
Input: n groups G_n where each group has set of papers (p_i)
Output: Set of system generated clusters/groups G_n
1: merge \leftarrow true 2: Flag \leftarrow false 3: While (merge=='true') do : 4: merge \leftarrow false 5: for i in range (0: len(G)-1): 6: for j in range (i+1: len(G)): 7: if (similarCoauthors(G_i L _{co-authors} , G_j L _{co-authors}) == true) then 8: Flag \leftarrow true 9: Else (checkNameFragments(G_i L _{co-authors} , G_j L _{co-authors}) == true) then 10: Flag \leftarrow true 11: if (Flag == true) then 12: $G_i \leftarrow G_i \cup G_j$ 13: $G \leftarrow G.pop(j)$ 14: merge \leftarrow true 15: end if 16: end if 17: end for 18: end while

Figure 4.1: Example Of Algorithm Design

4.2 External APIs/SDKs

Describe the third-party APIs/SDKs used in the project implementation in the following table. Few examples of APIs are provided in the table.

API and version	Description	Purpose of usage	API endpoint/function/class used
Stripe (version 2020-08-27)	Credit Card payment integration	Sandbox used orders payment	stripe.paymentMethods.create
Cloudinary	Image and Video management	Uploading Product Images	https://api.cloudinary.com/v1

4.3 Testing Details

Once the system has been successfully developed, testing has to be performed to ensure that the system working as intended.

4.3.1 Unit Testing

Each unit test is designed to test a specific function or method independently from other components, helping to identify issues directly related to the functionality being tested.

Following is the example of Unit testing:

Test case ID	Test Objective	Precondition	Steps	Test data	Expected result	Post-condition	Actual Result	Pass/fail
TC001	Verify admin login with username and password	Admin should be registered with valid email and password before login.	Click on Login button Enter valid username and password	Email-id: abc@xyz.com Password: Xyz123	System displays Admin homepage	Admin should be kept logged in until logout.	As Expected,	Pass

Figure 4.2: Example for Unit Testing

Bibliography

- [1] A Kolyshkin and S Nazarovs. Stability of slowly diverging flows in shallow water. *Mathematical Modeling and Analysis*, 2007.

Appendix A

Appendices

A.1 Appendix A

A.1.1 Use Case Diagram example (Online Shopping System)

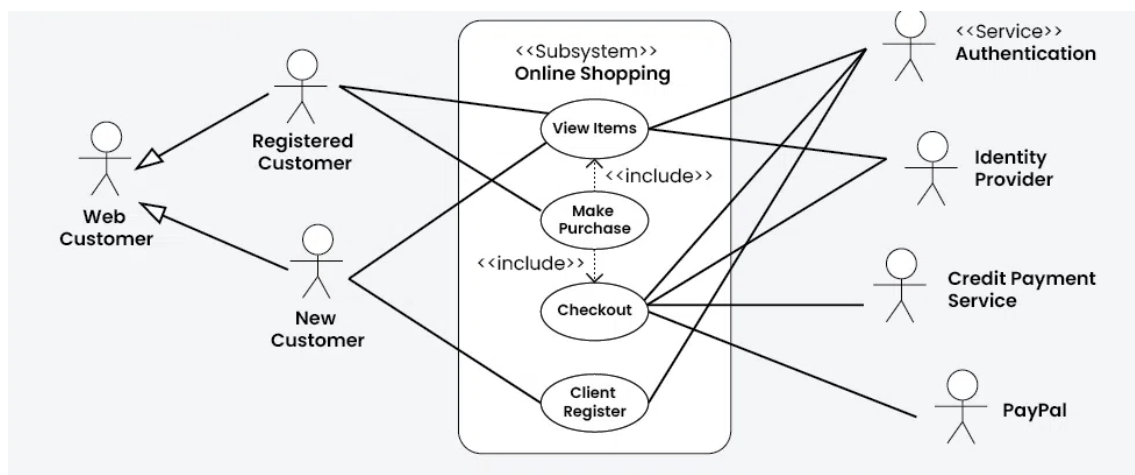


Figure A.1: Use Case Diagram for the Online Shopping System

A.1.2 Detail Use Case Example

ID	#1
Name	Overview elements
Short Description	All elements are shown in a list, where the user can set different filters.
Goal	Displaying elements in a changeable view.
Preconditions	The user got access to the system and is logged in. The user got the right to see schedules.
Success End Condition	The correct elements (filter and sorting) are displayed.
Fall End Condition	Elements not matching the criteria are displayed.
Stakeholder	Customer manager
Trigger	Login in as customer manager (because overview is on the starting screen for this role).
Normal Flow	<ol style="list-style-type: none">1. The list of all elements matching default criteria are shown.2. The user changes the filter criteria.3. The user presses the Filter button.4. The list shows all matching elements. Optional: <ol style="list-style-type: none">5. The user presses the Clear button.6. The list of all elements matching default criteria are shown.
Alternative Flows	With click on the column headers, the list sorting can be changed. Different sorting for the different columns is described in the table below.
Includes	Login
Frequency of Use	About 50 times per day
Constraints and Special Requirements	None
Assumptions	None
Notes and Issues	None

Figure A.2: Detail Use Case Example

A.1.3 Event-Response Table for a Highway Intersection

Event	System State	Response
Road sensor detects vehicle entering left-turn lane.	Left-turn signal is red. Cross-traffic signal is green.	Start green-to-amber countdown timer for cross-traffic signal.
Green-to-amber countdown timer reaches zero.	Cross-traffic signal is green.	<ol style="list-style-type: none"> 1. Turn cross-traffic signal amber. 2. Start amber-to-red countdown timer.
Amber-to-red countdown timer reaches zero.	Cross-traffic signal is amber.	<ol style="list-style-type: none"> 1. Turn cross-traffic signal red. 2. Wait 1 second. 3. Turn left-turn signal green. 4. Start left-turn-signal countdown timer.
Pedestrian presses a specific walk-request button.	Pedestrian sign is solid Don't Walk. Walk-request countdown timer is not activated.	Start walk-request countdown timer.
Pedestrian presses walk-request button.	Pedestrian sign is solid Don't Walk. Walk-request countdown timer is activated.	Do nothing.
Walk-request countdown timer reaches zero plus the amber display time.	Pedestrian sign is solid Don't Walk.	Change all green traffic signals to amber.
Walk-request countdown timer reaches zero.	Pedestrian sign is solid Don't Walk.	<ol style="list-style-type: none"> 1. Change all amber traffic signals to red. 2. Wait 1 second. 3. Set pedestrian sign to Walk. 4. Start don't-walk countdown timer.

Figure A.3: Example of Event Response Table

A.1.4 Story Board Example For Android App

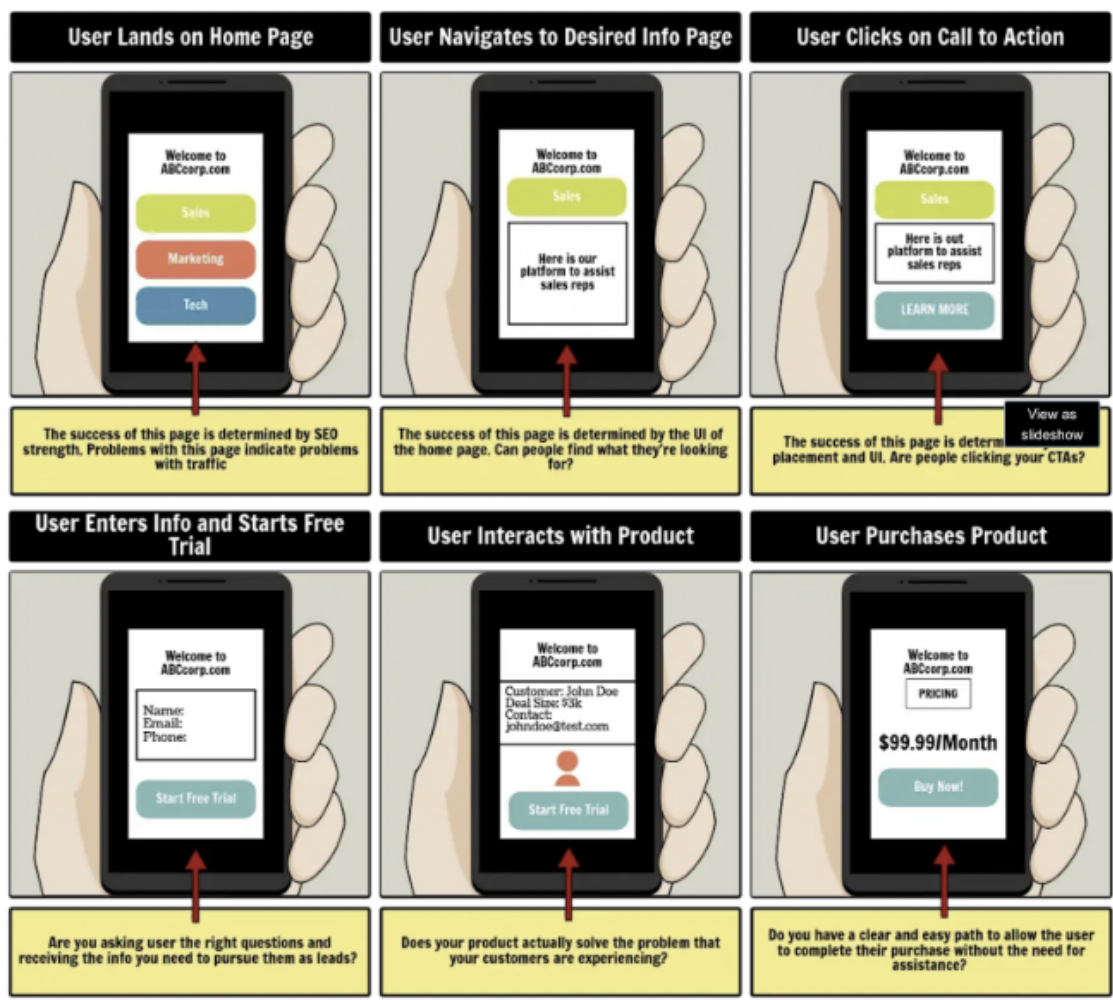


Figure A.4: Example of Story Board

A.2 Appendix B

A.2.1 Domain Model Example For Online Shopping

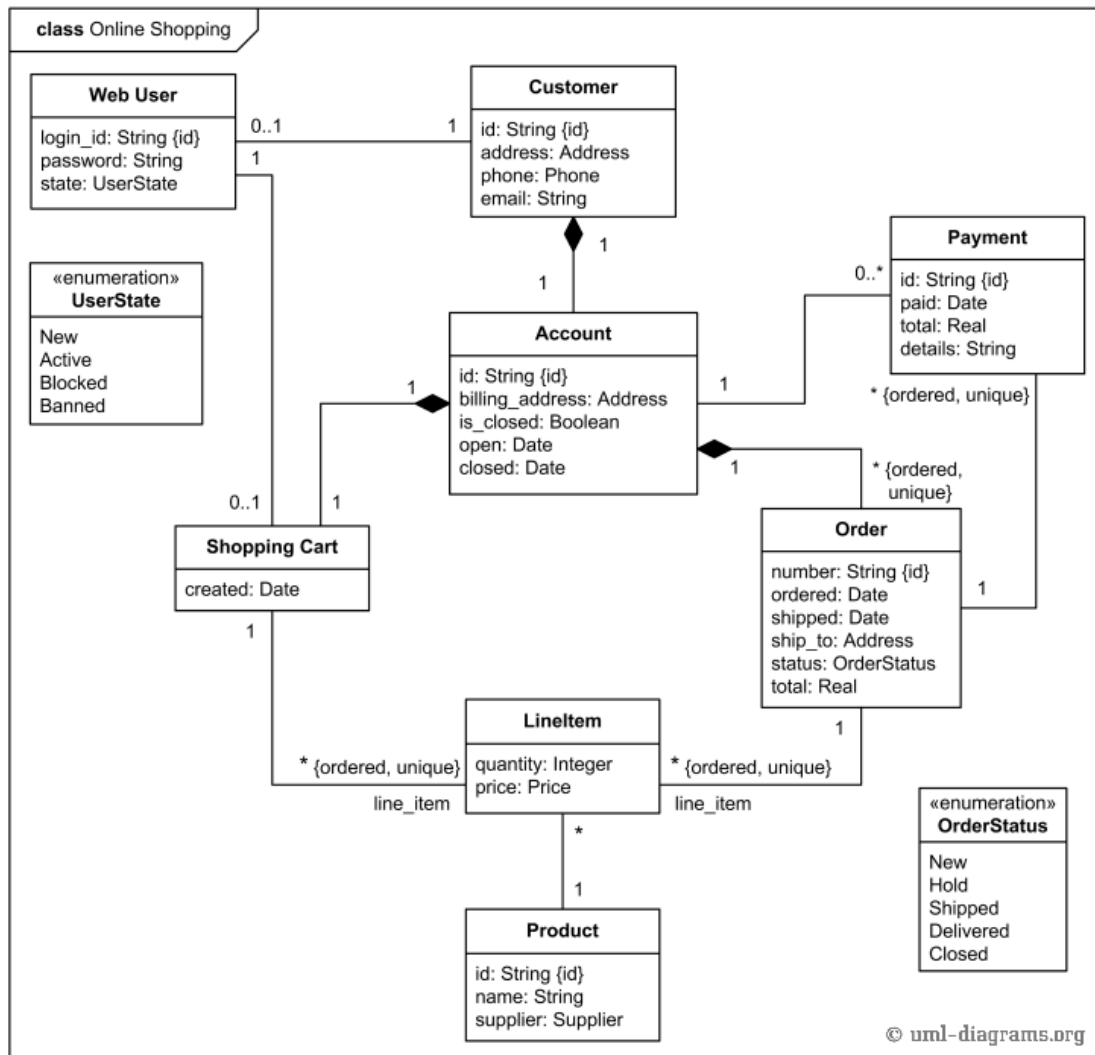


Figure A.5: Domain Model Example For Online Shopping Application

A.3 Appendix C

A.3.1 Box And Line Example For Online Shopping

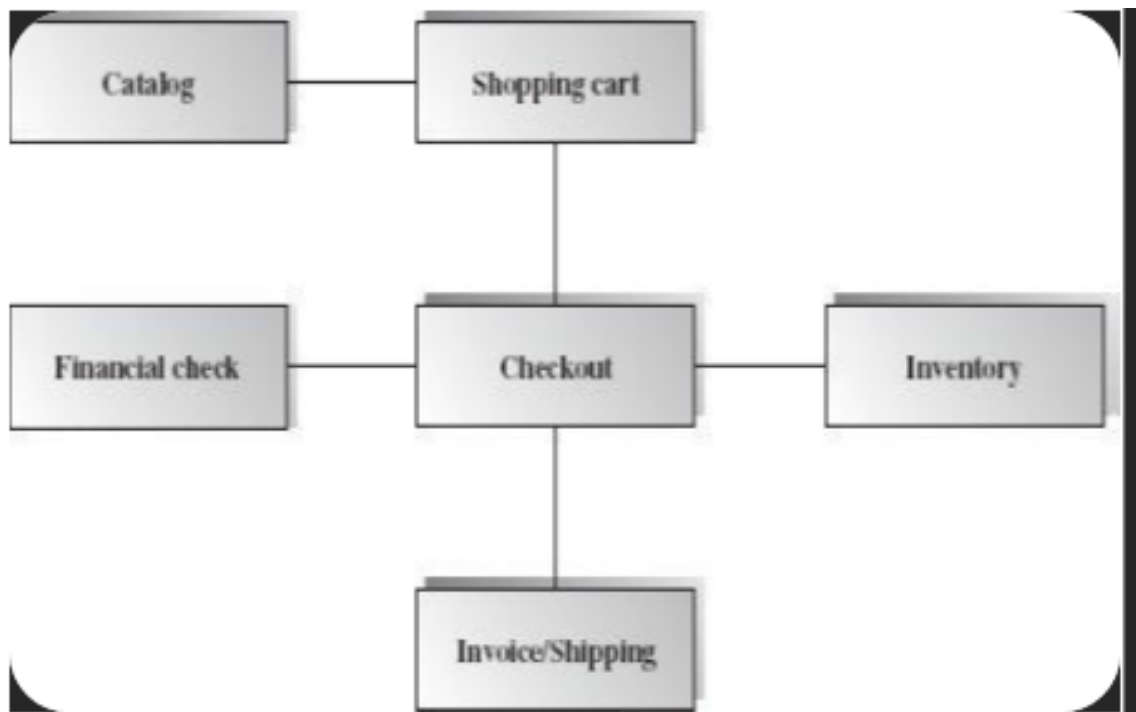


Figure A.6: Box and Line Diagram For Online Shopping Application

A.3.2 Architecture Pattern Example For Online Shopping

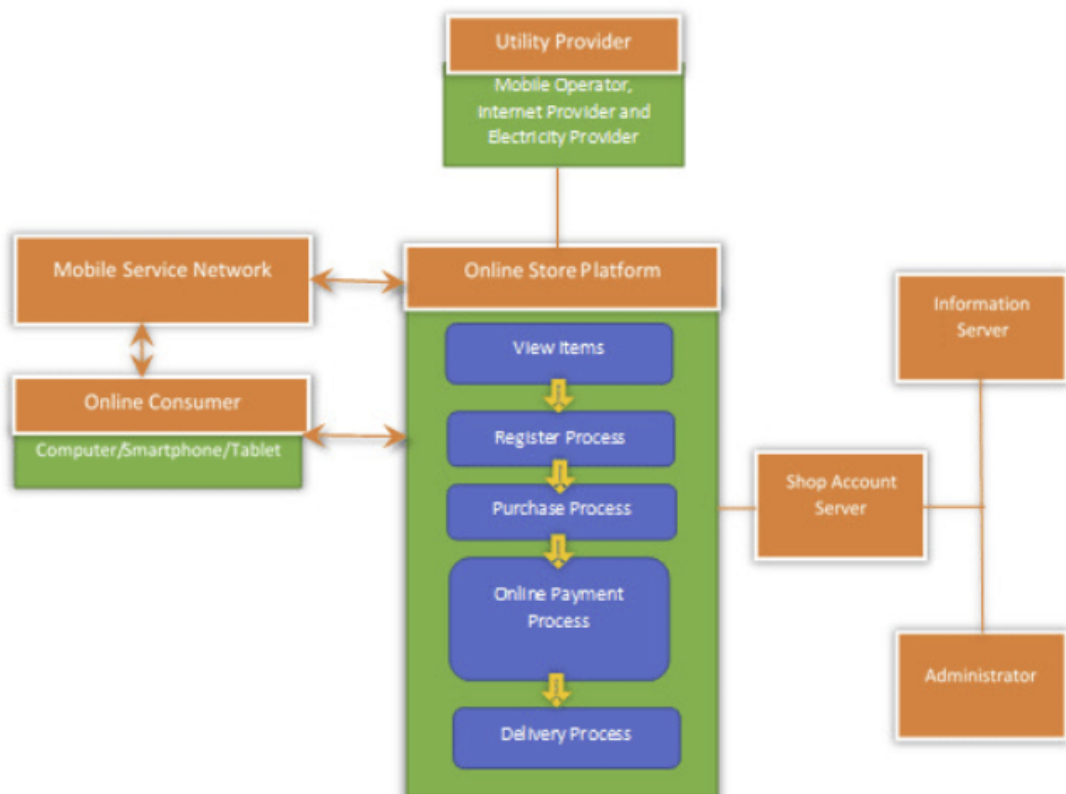


Figure A.7: Architecture Pattern For Online Shopping Application

A.4 Appendix D

A.4.1 Activity Diagram

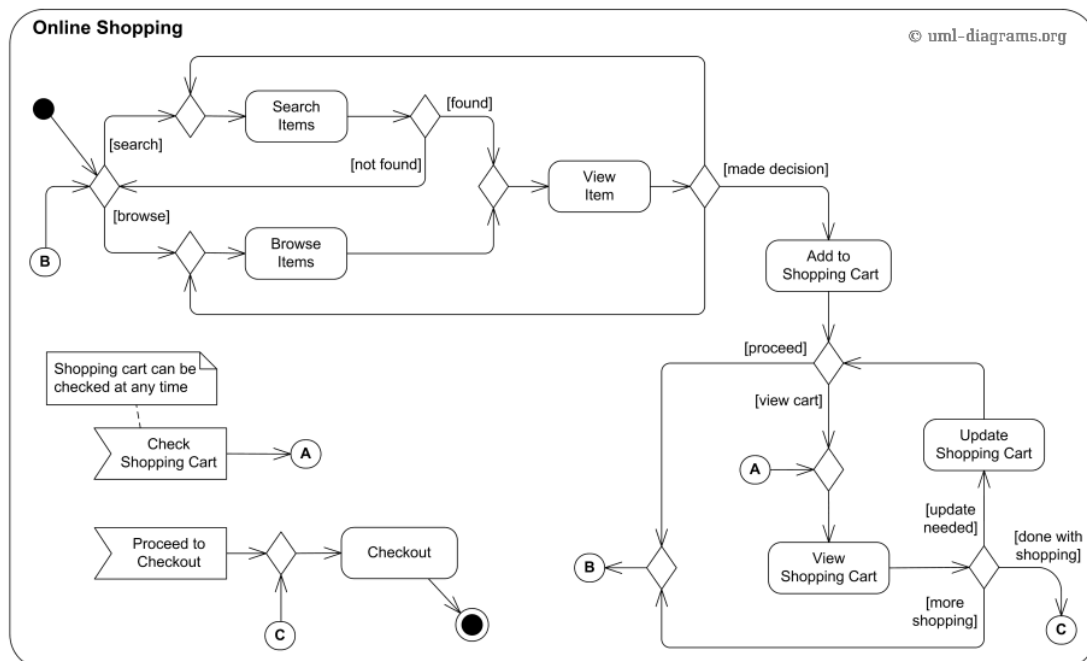


Figure A.8: Activity Diagram For Online Shopping Application

A.4.2 Class Diagram

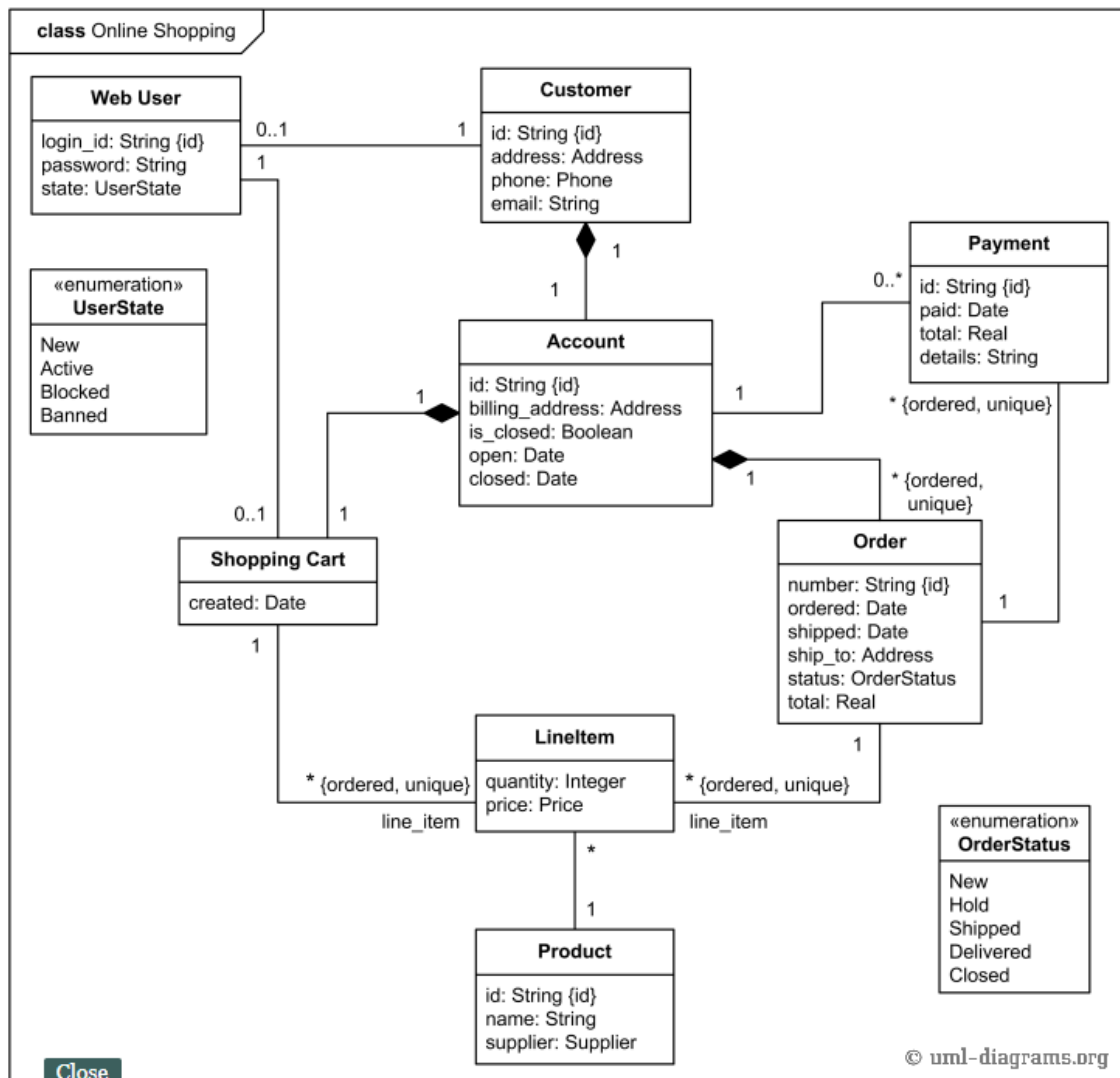


Figure A.9: Class Diagram For Online Shopping Application

A.4.3 Sequence Diagram

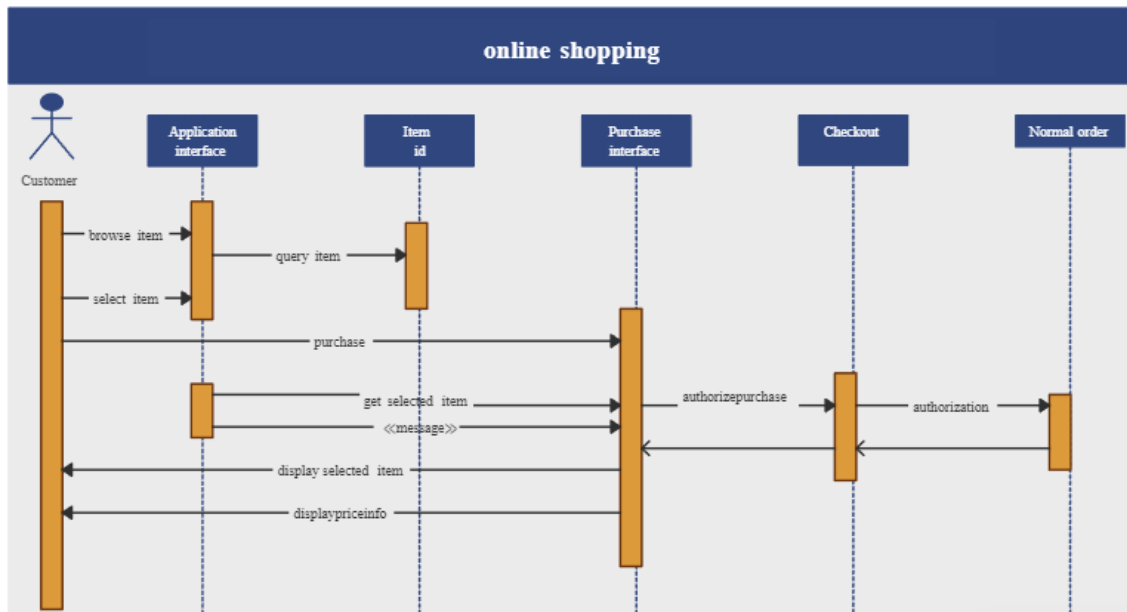


Figure A.10: Sequence Diagram For Online Shopping Application

A.4.4 State Transition Diagram

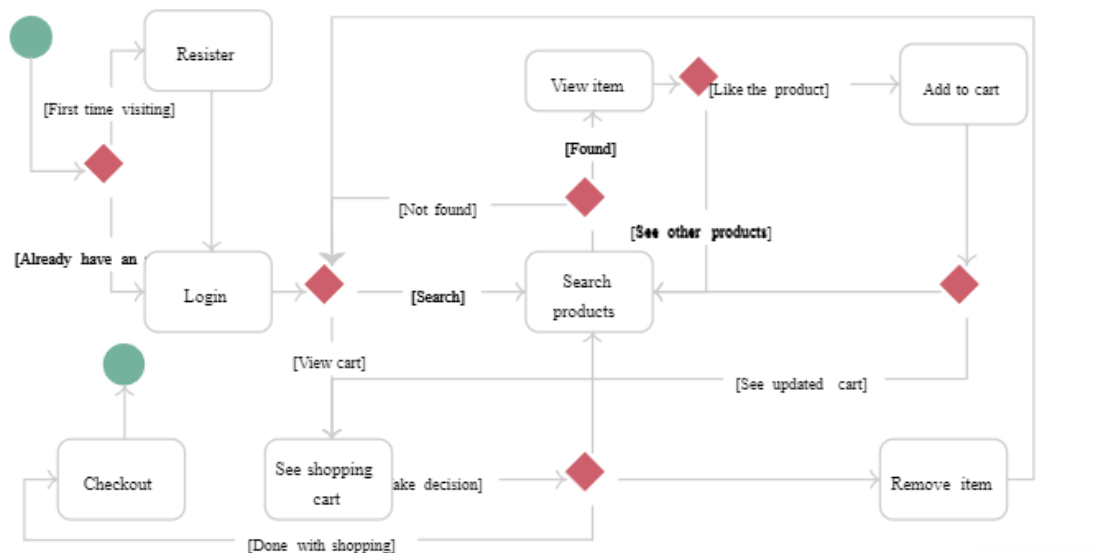


Figure A.11: State Transition Diagram For Online Shopping Application

A.4.5 Data Flow Diagram

Data Flow Diagram

Data flow diagram symbols, symbol names, and examples

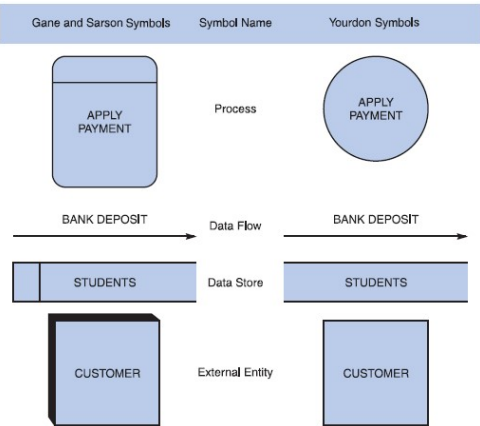


Figure B-5 Data flow diagram symbols, symbol names, and examples of the Gane and Sarson and Yourdon symbol sets.

Guidelines for Drawing DFDs

Step 1: Draw a Context Diagram: The first step in constructing a set of DFDs is to draw a context diagram. A **context diagram** is a top-level view of an information system that shows the system's boundaries and scope. Data stores are not shown in the context diagram because they are contained within the system and remain hidden until more detailed diagrams are created.

Example

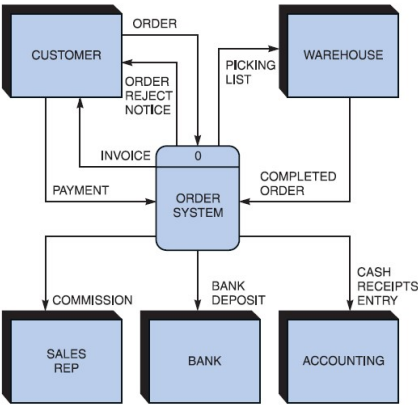


Figure B-6 Context diagram DFD for an order system.

Step 2: Draw a Diagram 0 DFD: To show the detail inside the black box, you create DFD diagram 0. **Diagram 0** zooms in on the system and shows major internal processes, data flows, and data stores. Diagram 0 also repeats the entities and data flows that appear in the context diagram. When you expand the context diagram into DFD diagram 0, you must retain all the connections that flow into and out of process 0.

Example

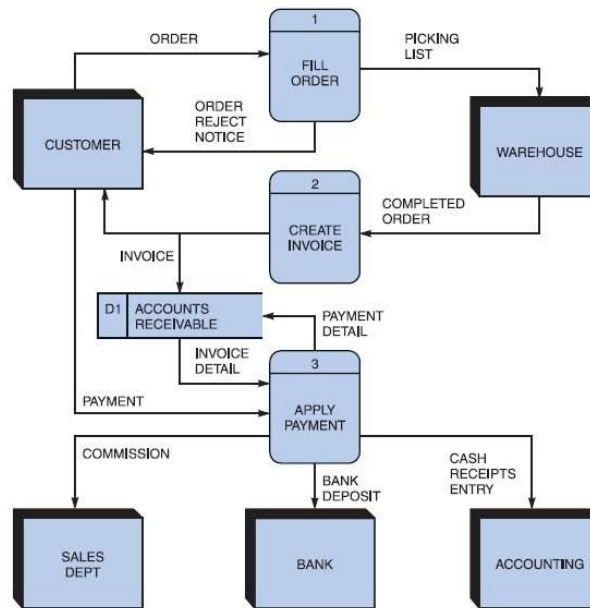


Figure B-7 Diagram 0 DFD for the order system.

Step 3: Draw the Lower-Level Diagrams:

To create lower-level diagrams, you must use leveling and balancing techniques. **Leveling** is the process of drawing a series of increasingly detailed diagrams, until all functional primitives are identified.

Leveling Example

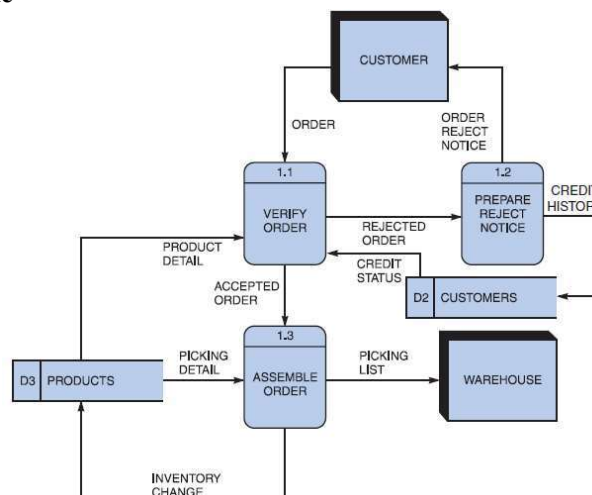


Figure B-8 Diagram 1 DFD shows details of the FILL ORDER process in the order system.