SECJ2203: Software Engineering

System Documentation (SD)

Project Title

Version X

Date

Faculty of Computing

Prepared by: <Team name>

Revision Page

a. Overview

Describe the content of the current version of SD (see below – the note in red).

b. Target Audience

State the targeted audience for the SD.

c. Project Team Members

List the team members in a table by stating their roles and the status for each assigned task e.g. by sections for this SD version (complete, partially complete, incomplete). If the assigned tasks are not done and have been assigned to other team members, state accordingly.

Member Name	Role	Task	Status

d. Version Control History

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Team leader 1 (Full Name)	Completed Chapter X, Section…	dd/mm/yyyy
…			

Note: The title page should show the latest version. For your first submission (PR2-Chap.1 and 2), it should be 1.0, then 2.0 for PR3 and 3.0 for PR4. Primary author(s) could be the team leaders who should compile all materials for the purpose of SD submission. [remove this note in red texts and the examples in the table above]

Note:
This System Documentation (SD) template is adapted from IEEE Recommended Practice for Software Requirements Specification (SRS) (IEEE Std. 830-1998), Software Design Descriptions (SDD) (IEEE Std. 1016-1998 1), and Software Test Documentation (IEEE Std. 829-2008) that are simplified and customized to meet the need of SECJ2203 course at School of Computing, UTM. Examples of models are from Arlow and Neustadt (2002) and other sources stated accordingly.

Table of Contents

<table>
<tr><td></td><td></td><td>8.1.2</td><td>TC001_02: Test <Scenario of Sequence Diagram 2 (SD002)></td><td></td></tr>
<tr><td></td><td>8.2</td><td colspan="2">TC002: Test <Name of Package 2> Subsystem: <Name of Use Case (UC002)></td><td></td></tr>
<tr><td></td><td>8.3</td><td colspan="2">TC003: Test <Name of Package 3> Subsystem: <Name of Use Case (UC003)></td><td></td></tr>
<tr><td colspan="5"></td></tr>
<tr><td></td><td colspan="3">Appendices</td><td></td></tr>
<tr><td></td><td colspan="3">Appendix A: Traceability Matrix</td><td></td></tr>
</table>

1.

Introduction

The following sections of the System Documentation (SD) should provide the details of the entire document comprises SRS, SDD, STD. Remove the notes in red texts when submitting including these notes.

1.1 **Purpose**

This subsection should:

a) *Delineate the purpose of the SD;*
b) *Specify the intended audience for the SD.*

This SD describes…

1.2 **Scope**

This subsection should:

a) *Identify the software product(s) to be produced by name (e.g., Host DBMS, Report Generator, etc.);*
b) *Explain what the software product(s) will, and, if necessary, will not do;*
c) *Describe the application of the software being specified, including relevant benefits, objectives, and goals;*
d) *Be consistent with similar statements in higher-level specifications (e.g., the system requirements specification), if they exist.*

The software product is…

1.3 **Definitions, Acronyms and Abbreviation**

This subsection should provide the definitions of all terms, acronyms, and abbreviations used in the SD.

Definitions of all terms, acronyms and abbreviations used are to be defined here.

1.4 **References**

This subsection should:

a) *Provide a complete list of all documents referenced elsewhere in the SD;*
b) *Identify each document by title, report number (if applicable), date, and publishing organization;*
c) *Specify the sources from which the references can be obtained.*

Specify a complete list of references using a standardized reference format.

1.5 **Overview**

This subsection should:

a) *Describe what the rest of the SD contains;*
b) *Explain how the SD is organized.*

Give an overview of the content of this SD document.

2.

Specific Requirements

This section of SD serves as the SRS that should contain all the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should ***include at a minimum a description of every input (stimulus) into the system, every output (response) from the system, and all functions performed by the system*** *in response to an input or in support of an output.*

2.1 **Persona**

Describes the intended users of the system and their characteristics, such as their technical expertise, knowledge, and physical abilities. This section is important because it provides the development team with a clear understanding of whom the system is designed for, and helps to ensure that the system is developed to meet the needs of its intended users.

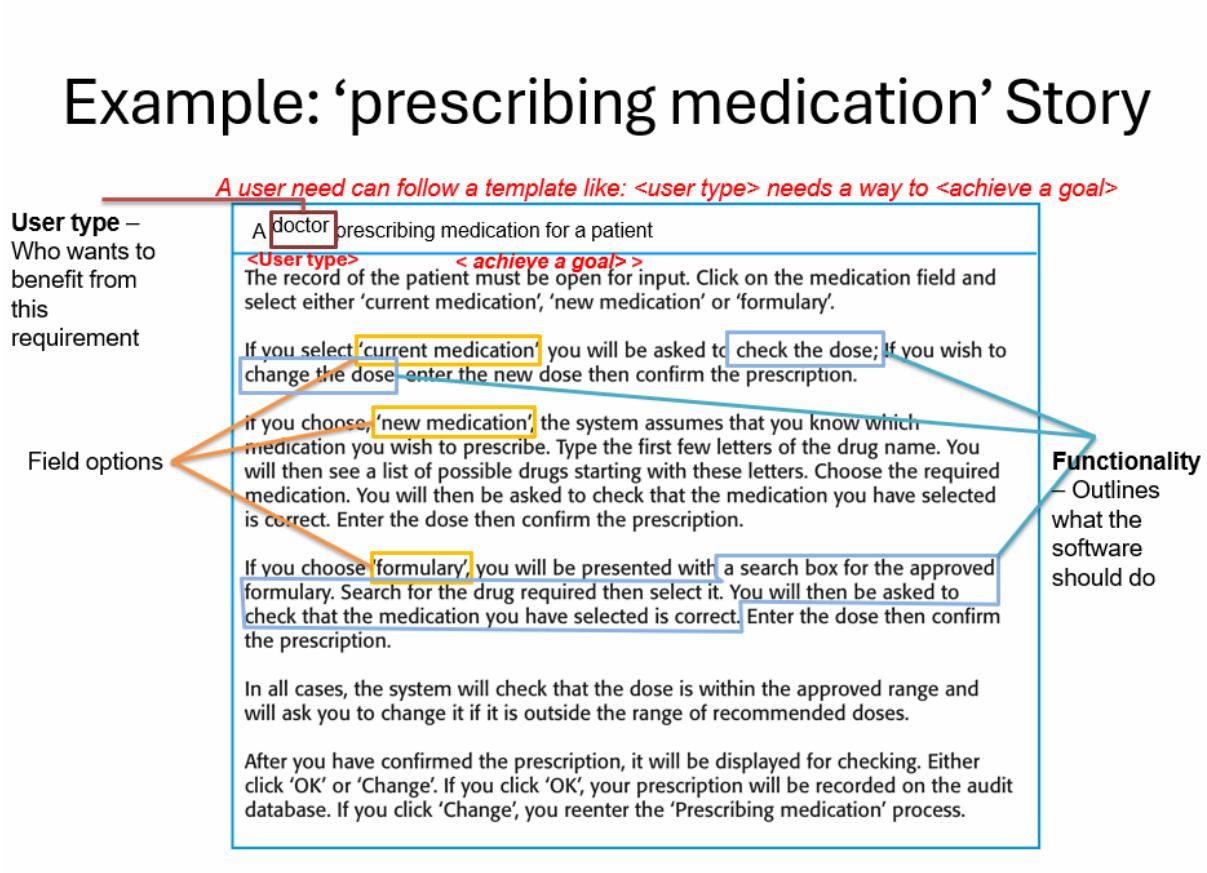
2.1.1 Persona 1 (User type 1/Actor 1)

User Need

- *A user need can follow a template like: <user type> needs a way to <achieve a goal>*

User Stories

- *User stories are short descriptions of a feature told from the perspective of the user who desires the new capability.*
- *A user story generally follows a template like: As a <user type>, I want <feature/interface/functionality> so that <reason>*
- *Sometimes, a user story may cover a large amount of functionality. In these cases, these stories may be referred to as 'epics' and may need to be broken out into individual, smaller user stories.*



2.1.2 Persona 2 (User type 2/Actor 2)

User Need

User Stories

2.1.3 Persona n (User type n/Actor n)

User Need

User Stories

2.2 System Features

Specify the product perspective.
Example.
The Emotion Chatbot Analyzer is a software system that operates on mobile devices, including both Android and iOS operating systems. The system provides a means for UTM students to better manage their emotional health and at the same time simplifies the process of seeking emotional support from UTM counselors The system also implemented a real-time cloud-based database for better user experiences.

Specify the product module and function..
Include use case diagram here – see an example of Emotion Chatbot Analyzer below

Then, provide the description of each module and function using a table.

Example.
The system features is illustrated in Figure X.X below. The detail description of each module and functions is tabulated in Table X.X.

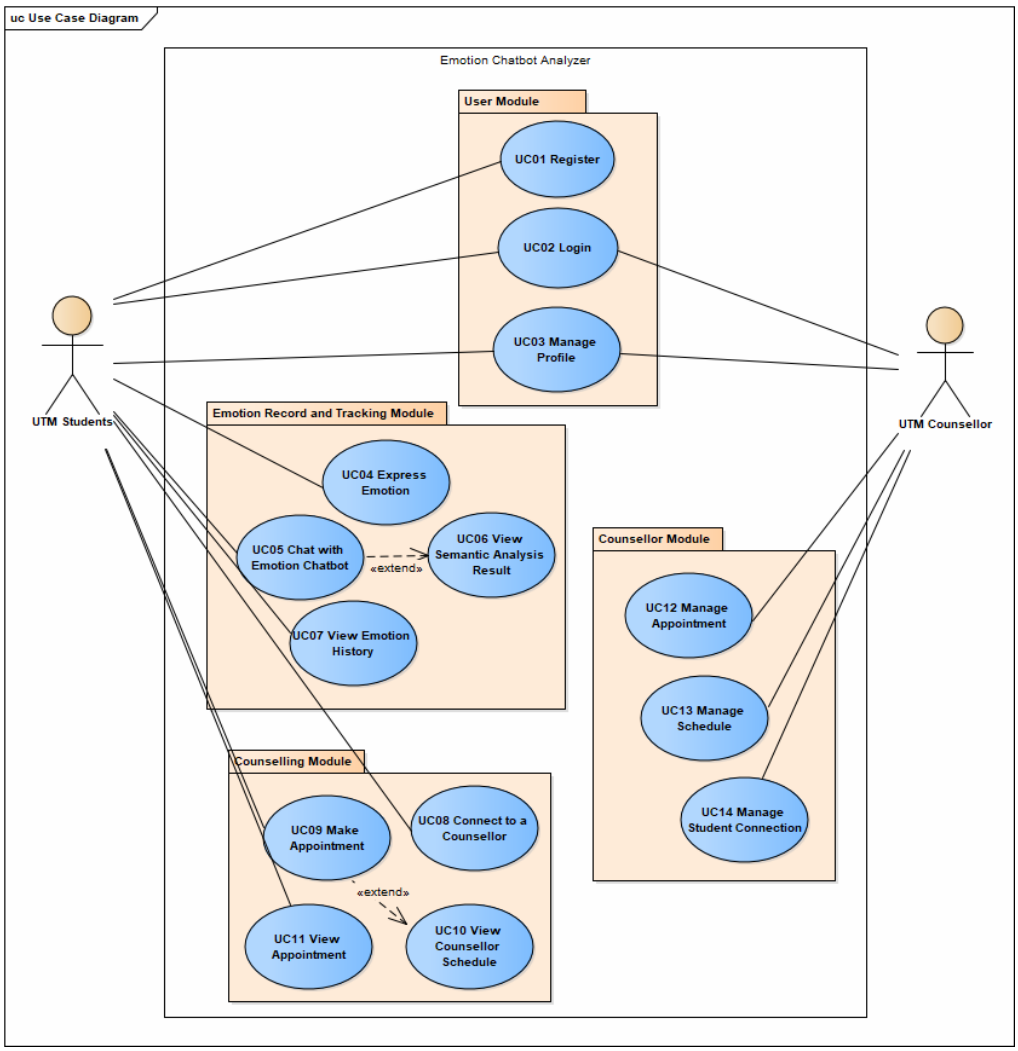


Figure X.X: Use Case Diagram for <Name of the System>

Table X.X: Description of Module and Functions for <Name of the System>

Use case	***Function***	***Description***
UC001	*Register*	*This use case allows students to sign up as a user for the mobile application.*

*[Include domain model i.e. **class diagram** without the operation part – only attributes without details on visibility and type, explain each class and its attributes including the relationships among the classes, see the example.]*

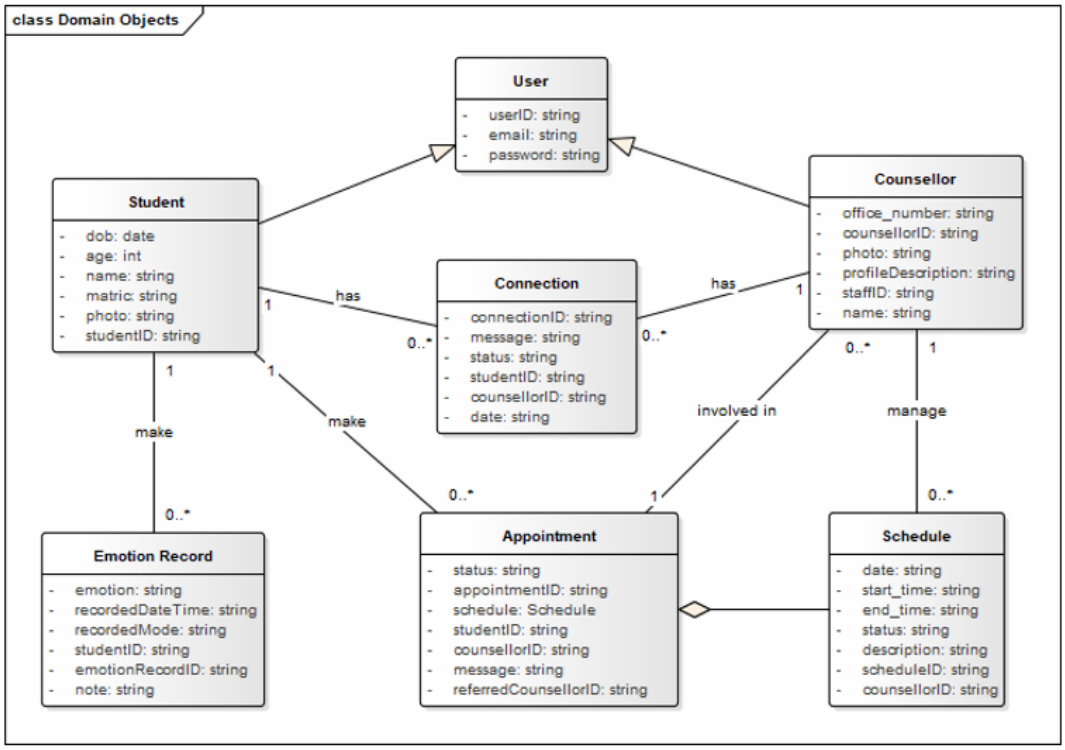


Figure X.X: Domain Model for <Name of the System>

*[Include domain model i.e. **class diagram** without the operation part – only attributes without details on visibility and type, explain each class and its attributes including the relationships among the classes, see the example. For the **class with the states only**, consider including its state machine diagram, see the example for state machine diagram of Account class]*

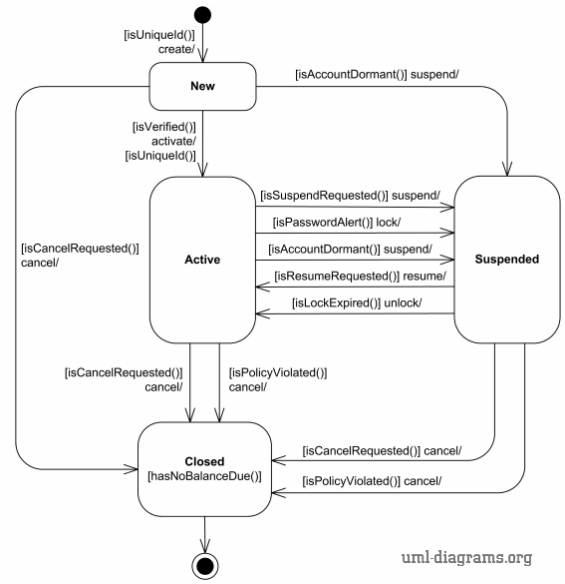


Figure X.X: State Machine Diagram for <Name of the System>

As this is often the largest and most important part of the SRS, the following principles apply:

a) *Specific requirements should be stated in conformance with all the characteristics described in 4.3. (IEEE Std 830-1998)*
b) *Specific requirements should be cross-referenced to earlier documents that relate.*
c) *All requirements should be **uniquely identifiable**. [Provide ID to each functional requirement]*
d) *Careful attention should be given to organizing the requirements to maximize readability.*

*[For each functional requirement, indicate its details using **use case description**. Include **sequence diagram** for each use case i.e. functional requirement. Combine alternate flows in the same sequence diagram. Split only if they are too cluttered to be combined.]*

2.3 **Launch Phase**

Product Backlog [describe more on product backlog];

Sprint	**Team members assigned**
Sprint #1 *Which user story assigned to this Sprint* [you can also specify based on which module you want to develop for each sprint – based on your use case diagram]	
Sprint #2 *Which user story assigned to this Sprint*	
Sprint #n *Which user story assigned to this Sprint*	

2.4 **User Story Details**

[Provide ID for each user story such as US001 and so on…

Include User Story Description for each use case as in the example below. If there are any alternative flows, add the rows accordingly. Otherwise, just remove the rows. Ensure each User Story has its own unique ID and the name of the heading corresponds to the name of use case in the use case diagram (refer to Figure 2.1). Different scenarios of a user story require separate descriptions.]

2.4.1 US001: User Story <User Story 1>

Table 2.1: User Story Description for <Name of Use Case>

User story: <Name of Use Case>
ID: USxxx
User Story Description **As a** customer of the bank **I want** to be able to log into the system **So that** I can use bank products
Flow of events: 1. 2. 3. …
Alternative flow *n*:
Acceptance Criteria Postcondition precondition other conditions
Exception flow: (if any in the event of error)

[Include sequence diagram and activity diagram for each respective user story. See example below. May consider including different scenarios in different diagrams, if necessary, to avoid clutter.]

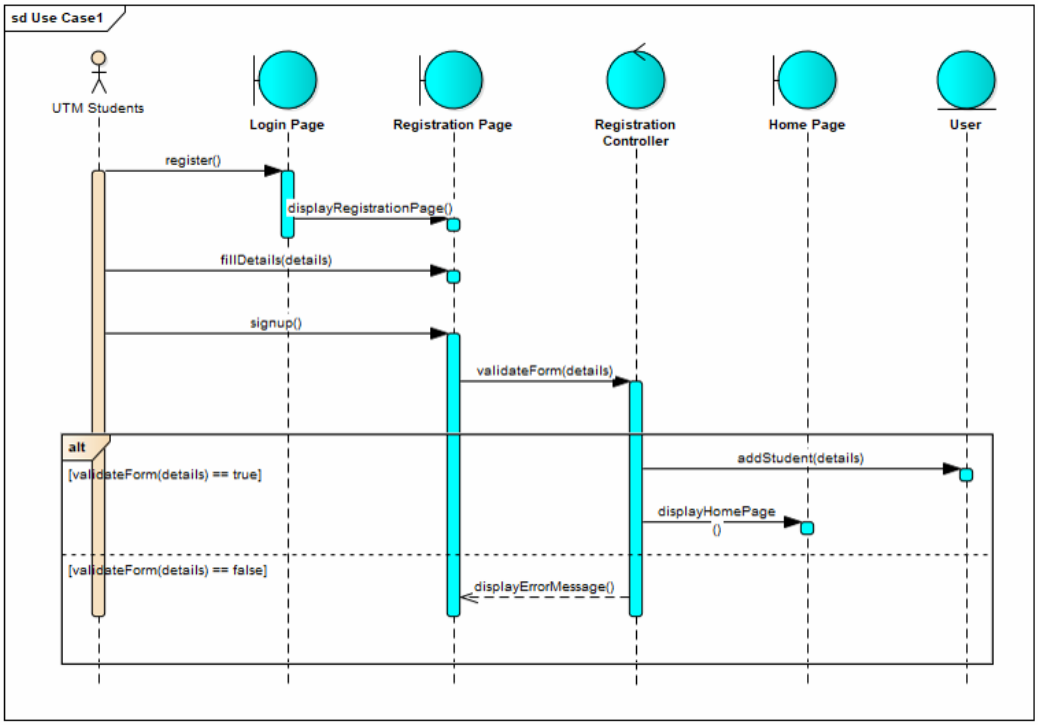


Figure 2.5: Sequence Diagram for <Name of User Story/Scenario if more than one scenario>

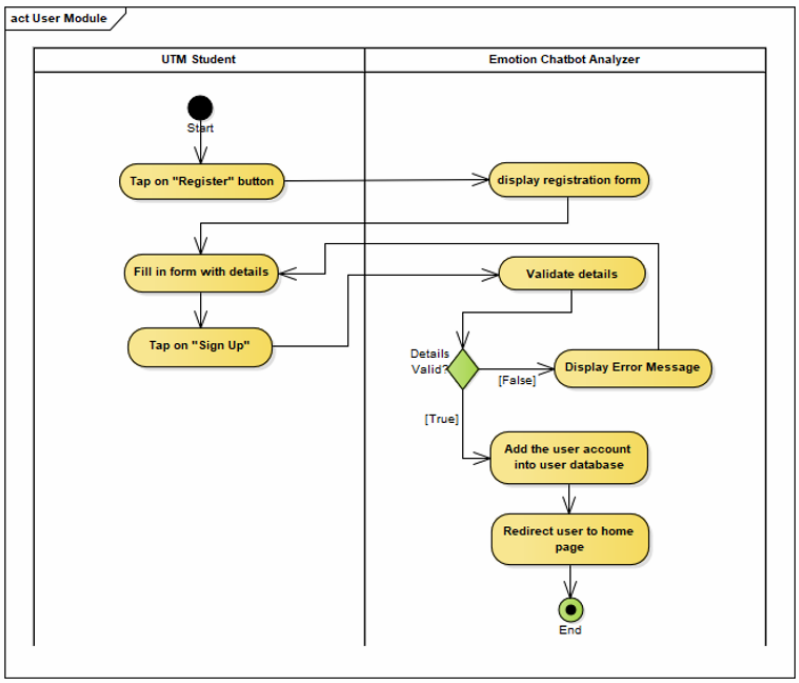


Figure 2.2: Activity Diagram for <<Name of User Story>>

2.4.2 US002: User Story <User Story 2>

2.4.3 US003: User Story <User Story 3>

2.4.4 USxx*n* User Story <User Story *n*>

…

2.5 Performance and Other Requirements

Non-functional requirements are the requirements that describe how the system should behave or operate, rather than what it should do. They address the characteristics or qualities of the system, such as its usability, reliability, performance, security, maintainability, and compatibility. Non-functional requirements are important to write in the SD because they help ensure that the system meets the stakeholders' needs and expectations.The ISO/IEC/IEEE 29148 standard suggests that non-functional requirements should be specified under three main categories: Software System Attributes, Performance and Other Requirements.

Example of Performance is as follows:
Performance requirements define the system's capability to respond to user requests and handle data in a timely manner. These requirements include the following:

- *Response Time: The time it takes for the system to respond to a user request.*
- *Throughput: The number of requests the system can handle in a given period of time.*
- *Capacity: The maximum number of users or amount of data that the system can handle.*
- *Availability: The percentage of time the system is operational and accessible to users.*

Example of Software System Atrributes are as follows:
Software system attributes define the overall qualities or characteristics of the software. These attributes are the foundation on which the software is built, and they include the following:

- *Usability: The ease with which users can interact with the system.*
- *Reliability: The ability of the system to perform its functions correctly and consistently.*
- *Maintainability: The ease with which the system can be modified, repaired, and enhanced.*
- *Portability: The ability of the system to run on different hardware and software platforms.*
- *Compatibility: The ability of the system to work with other systems and components.*
- *Security: The protection of the system and its data from unauthorized access and malicious attacks.*
- *Safety: The ability of the system to operate safely without causing harm to users or the environment.*
- *Legal and Regulatory: Compliance with laws, regulations, and standards.*
- *Environmental: The impact of the system on the environment.*
-

2.6 Design Constraints

Explain any constraints imposed by the organization where the software product will be used such as the system must adhere to certain organizational standard and other external non-functional requirements.

Example is as follows:
Environmental constraints: These constraints relate to the physical environment in which the system will operate, such as temperature, humidity, and lighting conditions. An example of an environmental constraint is: "The system must be designed to operate in temperatures ranging from 0°C to 40°C."

Hardware constraints: These constraints relate to the specific hardware components that will be used in the system, such as processors, memory, and storage. An example of a hardware constraint is: "The system must be designed to run on a minimum of 8GB of RAM."

Security constraints: These constraints relate to the security requirements for the system, such as data encryption, authentication, and access control. An example of a security constraint is: "The system must be designed to encrypt all sensitive user data using AES-256 encryption."

Compatibility constraints: These constraints relate to the compatibility requirements for the system, such as compatibility with specific software or hardware components. An example of a compatibility constraint is: "The system must be designed to be compatible with Windows 10 operating system."

Performance constraints: These constraints relate to the performance requirements for the system, such as response time, throughput, and capacity. An example of a performance constraint is: "The system must be designed to support a maximum of 1000 concurrent users with a response time of less than 5 seconds."

3

System Architectural Design

This section of the SD serves as part of SDD that should describe the architectural style and the rationale or justification of your selection. The component and subsystem diagram should also be included.

3.4 Architecture Style and Rationale

State your chosen architectural style and the rationale of choosing that particular style.

State here

3.5 Component Model

Develop a component model and explain the relationships between the components to achieve the complete functionality of the system. This is a high level overview of how responsibilities of the system were partitioned and then assigned to subsystems. Identify each high level subsystem and the roles or responsibilities assigned to it. Describe how these subsystems collaborate with each other in order to achieve the desired functionality. Do not go into too much detail about the individual subsystem. 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 data repositories and their interconnections. Describe the diagram clearly.

[Include component diagram here]

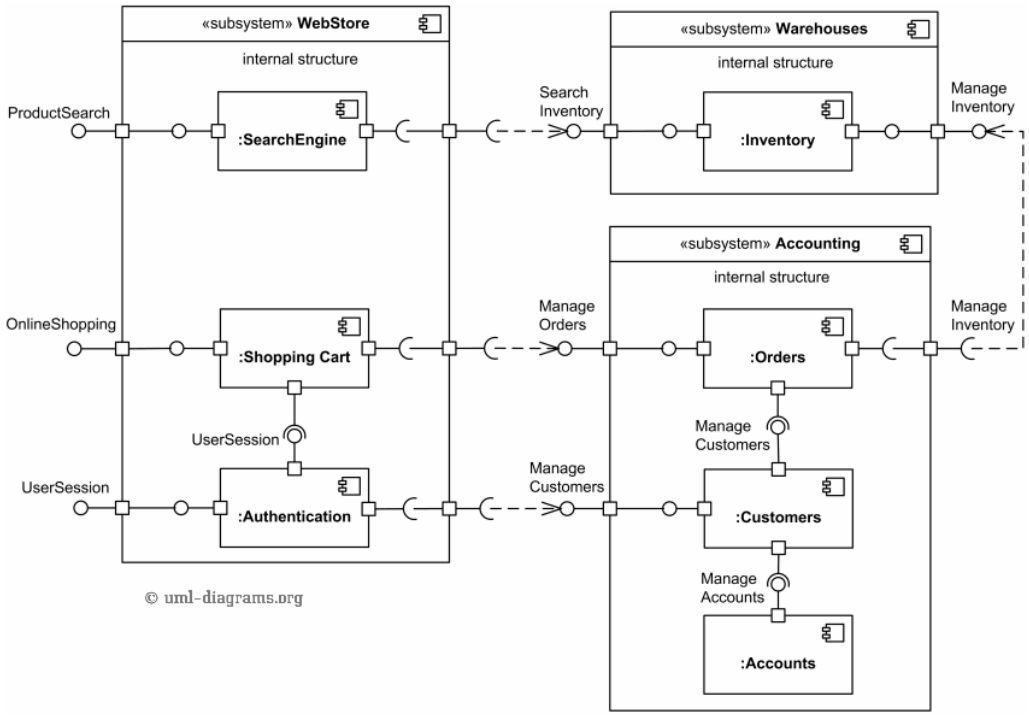


Figure 3.1: Component Diagram of <Name of the System>

4

Detailed Description of Components

This section of SD serves as part of SDD that describes each module or subsystem in the project.

4.4 Complete Package Diagram

Include the overall package diagram of your system here. [Example package diagram for a Sale System adapted from Satzinger (2011)] Indicate the navigation visibility based on the dependency among classes in the design class diagram. If the diagram is too cluttered, simplify the classes by showing the class name only without showing the attributes and methods. The details can be shown in the following class diagram sub-section for respective subsystem/package.

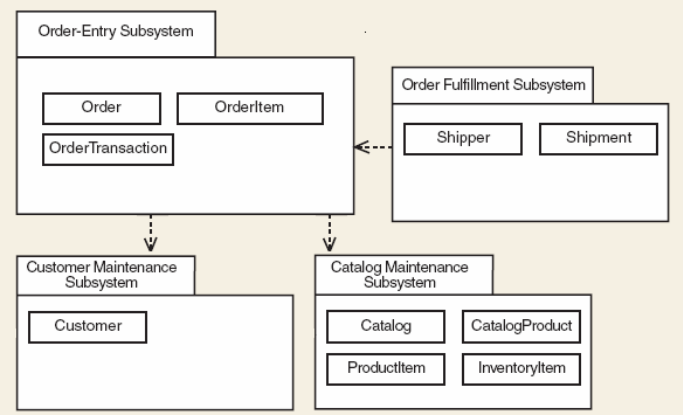


Figure 4.1: Package Diagram for <Name of the System>

4.5 Detailed Description

For each subsystem/package there must be ONE class diagram and several sequence diagrams based on how many use cases you have in the subsystem/package. Use branching in sequence diagram to combine alternate flow in the same sequence diagram. If the diagram is cluttered, consider a new sequence diagram for respective scenario/alternate flow. The given example includes view, domain and data access layer in respective packages. Organise subsystem/package according to the chosen architectural style. Note that if you choose model-view-controller (MVC) or other architectural styles, then the packages should follow the selected styles. However, for the scope of this course, you may follow the example.

4.5.2 P001: <Name of Package 1> Subsystem

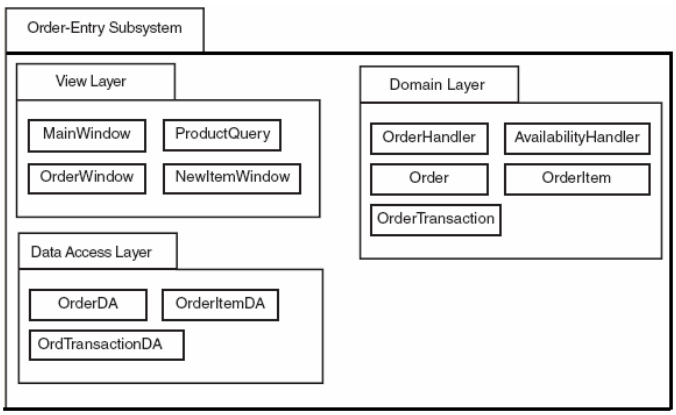


Figure 4.2: Package Diagram for <Name of Package 1> Subsystem

4.5.2.1 Class Diagram

Include class diagram to represent all classes in the respective subsystem/package. Include the controller classes.

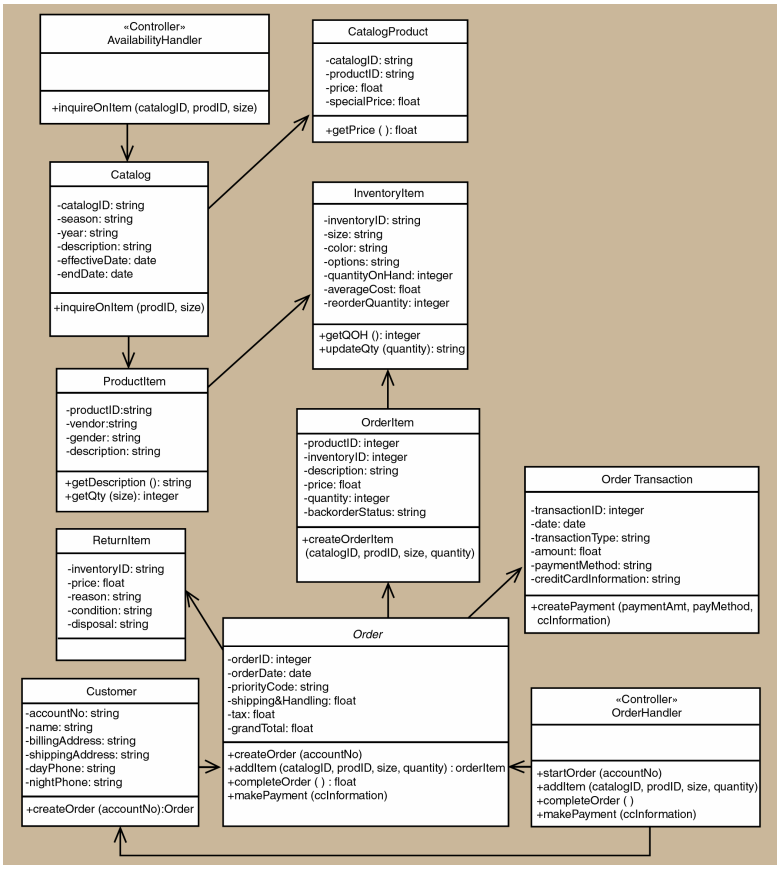


Figure 4.3: Class Diagram for <Name of Package 1> Subsystem

List all methods in a table for respective entity (add the table accordingly), then write its algorithm. Example of algorithm as shown below.

Step1: Start

Step2: Read/input A and B

Step3: If A greater than B then C=A

Step4: If B greater than A then C=B

Step5: Print C

Step6: End

Entity Name	e.g. Order
Method Name	e.g. createOrder
Input	
Output	
Algorithm	1. Start 2. … 3. End

4.5.2.2 Sequence Diagram

Include sequence diagram for each respective use case in your package. In these examples only sequence diagram Create New Phone Order Scenario and Cancel an Order Scenario are shown. Include the final sequence diagram that comprises view layer, controller, and its problem domain (entity) and data access layer. Provide a unique code for each scenario of sequence diagram to be used in Section 7: Requirements Matrix. If you have only one scenario, then you only need one sequence diagram.

a) SD001: Sequence diagram for Create New Phone Order

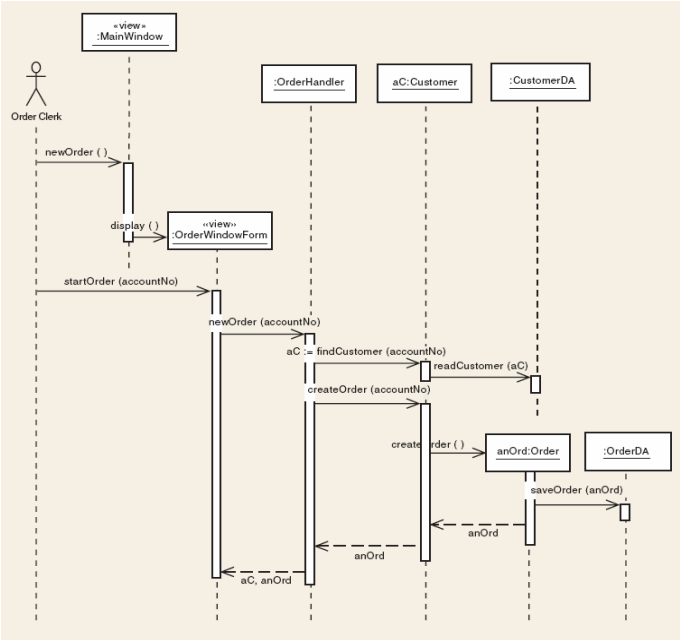


Figure 4.4: Sequence Diagram for <Create New Phone Order Scenario>

b) SD002: Sequence diagram for Create Cancel an Order Scenario

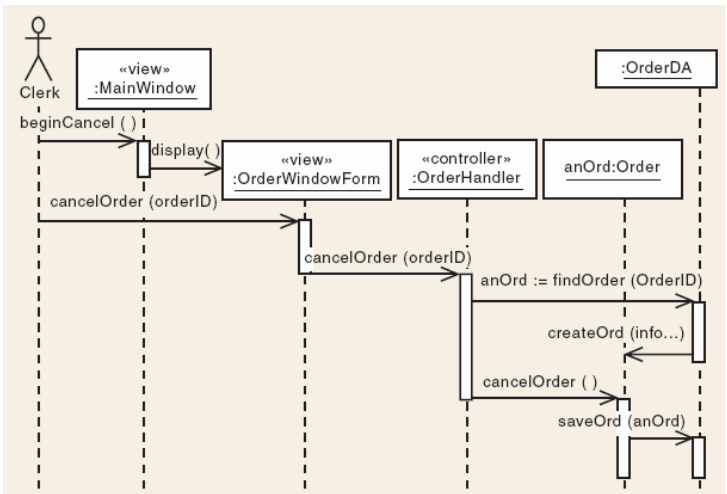


Figure 4.5: Sequence Diagram for <Cancel an Order scenario>

4.5.3 **P002: <Name of Package 2> Subsystem**

4.5.3.1 **Class Diagram**

4.5.3.2 **Sequence Diagram**

4.5.4 **P003: <Name of Package *n*> Subsystem**

4.5.4.1 **Class Diagram**

4.5.4.2 **Sequence Diagram**

5

Data Design

5.4 Data Description

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 the database(s) or data storage items. For a small system, there is normally only one database. It consists of all the tables in which for object-oriented they are classes/objects as also stated in the domain model. Use table for the listing.

The major data or systems entities are stored into a relational database named as…, processed and organized into *n* entities as listed in Table 5.1.

Table 5.1: Description of Entities in the Database

No.	Entity Name	Description

5.5 Data Dictionary

Alphabetically list the system entities or major data along with their types and descriptions (class/object, attributes). Focus on classes in domain layer; omit the controller/handler class. Use tables for easy listing as shown below.

5.5.2 Entity: <First Entity>

Attribute Name	Type	Description

5.5.3 Entity: <Second Entity>

Attribute Name	Type	Description

6

User Interface Design

6.4 Overview of User Interface

Describe the functionality of the system from the user's perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback information that will be displayed for the user.

The interface includes…

6.5 Screen Images

Display screenshots showing the interface from the user's perspective. These can be drawn using an automated drawing tool. Just make them as accurate as possible. Organise the images by subsystem/package.

Insert the screenshot here

Figure 6.1: Interface for <State the Feature

7

Requirements Matrix

Provide a cross-reference that traces components and data structures to the requirements in the SD (Chapter 2).

Use a tabular format to show which system components (sequence diagram vs. class) satisfy each of the functional requirements represented as use cases. Refer to the functional requirements by the numbers/codes given to each use case in Chapter 2. Examples of traceability to ease tracing of related components (sequence diagram vs. class) for each use case in respective subsystem/package are as below.

The sequence diagrams for each use case vs. corresponding classes (entities) are listed as in Table 7.1.

Table 7.1: Description of Entities in the Database

	Customer	Order	Catalog	CatalogProduct	ProductItem	OrderItem	ReturnItem	InventoryItem	…
P001, UC001, SD001	X	X							
P001, UC001, SD002		X							
P002, UC002, SD003			X	X					
P003, UC003, SD004					X	X	X	X	
P004, UC004, SD005		X						X	
…									

8

Test Cases

This is a numbered list of tests. Use tables to group similar tests. For each test, specify:

- *Test ID and name*
- *Additional description if test name is not descriptive enough*
- *The input data*
- *The expected output data*
- *The actual output data (not in the scope of this course – leave blanks)*
- *Result: pass or fail (not in the scope of this course – leave blanks)*

8.4 TC001: Test <Name of Package 1> Subsystem: <Name of Use Case (UC001)>

List all test cases before providing the details for each under each package/subsystem and use case.

This test contains the following test cases:

(a) TC001_01: Test <Scenario of sequence diagram1 (SD001)>

(b) TC001_02: Test <Scenario of sequence diagram2 (SD002)>

(c) …

8.4.2 TC001_01: Test <state scenario of sequence diagram1 (SD001)>

*Provide the details for each test case in the test case template (Excel). For the scope of this course, leave blanks for the columns on actual results and pass/fail status. See the example below for better understanding. If there are **alternate and exception scenarios**, include respective test cases under this sub-section also.*

This test contains the following alternate and exception scenarios (if any):

(a) TC001_01_01: Test <alternate scenario1 of sequence diagram1 (SD001)>

(b) TC001_01_02: Test <exception scenario1 of sequence diagram1 (SD001)>

(c) …

Test Case ID	BU_001	Test Case Description	Test the Login Functionality in Banking		
Created By	Mark	Reviewed By	Bill	Version	2.1
QA Tester's Log	Review comments from Bill incorprate in version 2.1				
Tester's Name	Mark	Date Tested	1-Jan-2017	Test Case (Pass/Fail/Not	Pass

S #	Prerequisites:	S #	Test Data
1	Access to Chrome Browser	1	Userid = mg12345
2		2	Pass = df12@434c
3		3	
4		4	

Test Scenario	Verify on entering valid userid and password, the customer can login

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://demo.guru99.com	Site should open	As Expected	Pass
2	Enter Userid & Password	Credential can be entered	As Expected	Pass
3	Click Submit	Cutomer is logged in	As Expected	Pass

8.4.3 TC001_02: Test <Scenario of sequence diagram2 (SD002)>

Provide the details for this test case.

8.4.4 TC001_*n*: Test <Scenario of sequence diagram *n* (…)>

Provide the details for this test case.

8.5 TC002: Test <Name of Package 2> Subsystem: <Name of Use Case (UC002)>

List all test cases before providing the details for the second use case in module1. Include the sub-sections accordingly.

This test contains the following test cases:

(a) TC002_01: Test <Scenario of sequence diagram4 (SD004)>

(b) TC002_02: Test <Scenario of sequence diagram5 (SD005)>

(c) …

8.6 TC003: Test <Name of Package 3> Subsystem: <Name of Use Case (UC003)>

List all test cases before providing the details for the first use case in module2. Include the sub-sections accordingly.

This test contains the following test cases:

(a) TC003_01: Test <Scenario of sequence diagram6 (SD006)>

(b) TC003_02: Test <Scenario of sequence diagram7 (SD007)>

(c) …

Appendix A: Traceability Matrix

Test Case ID	**Use Case ID/ Sequence Diagram ID**	**Package ID**
TC001 for <Name of Package 1> Subsystem • TC001_01 • TC001_02	UC001 • SD001 • SD002	P001
TC002 for <Name of Package 2> Subsystem • TC002_01 • TC002_02	UC002 • SD004 • SD005	P001
TC003 for <Name of Package 3> Subsystem • TC003_01 • TC003_02	UC003 • SD006 • SD007	P002
…		