***Software Design Description***

**<Add Project Title here>**

**Version <x.x>**

*Prepared by*

**<the group/company name and course code>**

**<e.g.>**

**Group A**

**SECJ4423 – Real-Time Software Engineering**

**Faculty of Computing**

**Universiti Teknologi Malaysia**

**(*Lecturer: xxx*)**

**Revision Page**

1. **Overview**

<Describe the content of the current version>

1. **Target Audience**

<State the targeted audience, e.g. Software Architect, Software Engineer, Developer, and System Administrator>.

1. **Project Team Members**
2. **<Add the name of Project Leader> (Project Leader)**
3. <Team member>
4. **Version Control History**

| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| --- | --- | --- | --- |
| 1.0 |  |  |  |
| 1.1 |  |  |  |

**Note:**

This document is adapted from ***IEEE SA - 830-1998 -*** *IEEE Recommended Practice for Software Requirements Specifications* and ***IEEE SA - 1016-1998 -*** *IEEE Recommended Practice for Software Design Descriptions*.

**Table of Contents**

[1](#_heading=h.gjdgxs) Introduction 1

[1.1](#_heading=h.30j0zll) Purpose 1

[1.2](#_heading=h.1fob9te) Scope 1

[1.3](#_heading=h.3znysh7) Definitions, Acronyms and Abbreviations 1

[1.4](#_heading=h.2et92p0) References 1

[1.5](#_heading=h.tyjcwt) Overview 2

[1.6](#_heading=h.3dy6vkm) Real-time profile and modeling tools 2

[1.7](#_heading=h.1t3h5sf) Innovation solution for IOT and Data analytics 2

[2](#_heading=h.4d34og8) System Software Process Model and Type of Prototyping 3

[2.1](#_heading=h.2s8eyo1) Introduction 3

[3](#_heading=h.17dp8vu) System Software Requirements 3

[3.1](#_heading=h.3rdcrjn) Functional 3

[3.2](#_heading=h.26in1rg) Non-Functional 3

[3.2.1](#_heading=h.lnxbz9) Performance 3

[3.2.2](#_heading=h.35nkun2) Security 3

[3.3](#_heading=h.1ksv4uv) Constraints 3

[3.4](#_heading=h.44sinio) Interface 3

[4](#_heading=h.2jxsxqh) System Software Architectural Design 3

[4.1](#_heading=h.z337ya) System Architecture 3

[4.2](#_heading=h.3j2qqm3) Software Architecture – Component Diagram 3

[4.3](#_heading=h.1y810tw) Class Diagram 4

[4.4](#_heading=h.4i7ojhp) Concurrency Design 4

[5](#_heading=h.2xcytpi) Detailed Description of Components 5

[5.1](#_heading=h.1ci93xb) Complete Package Diagram 5

[5.2](#_heading=h.3whwml4) Detailed Descriptions 5

[5.2.1](#_heading=h.2bn6wsx) Module <Name of Module 1> 5

[5.2.1.1](#_heading=h.qsh70q) P001: Package <Name of Package> 5

[5.2.1.2](#_heading=h.3as4poj) Class Diagram 5

[5.2.1.3](#_heading=h.1pxezwc) Sequence Diagrams 5

[5.2.2](#_heading=h.49x2ik5) Module <Name of Module 2> 6

[5.2.2.1](#_heading=h.2p2csry) P002: Package <Name of Package> 6

[5.2.2.2](#_heading=h.147n2zr) Class Diagram 6

[5.2.2.3](#_heading=h.3o7alnk) Sequence Diagrams 6

[5.2.3](#_heading=h.23ckvvd) Module <Name of the n Module> 6

[5.2.3.1](#_heading=h.ihv636) Package <Name of Package> 6

[5.2.3.2](#_heading=h.32hioqz) Class Diagram 6

[5.2.3.3](#_heading=h.1hmsyys) Sequence Diagrams 6

6User Interface Design 7

[7.1](#_heading=h.1v1yuxt) Overview of User Interface 7

[7.2](#_heading=h.4f1mdlm) User Interface 7

7Requirements Matrix 8

# Introduction

<The following section and subsections of the Software Design Descriptions (SDD) should provide the details of the entire SDD. Remove the notes in read texts including these notes.>

## Purpose

*<Identify the purpose of this SDD and its intended audience. (e.g. “This software design document describes the architecture and detailed design of System XX. ….”).>*

This SDD provides ……

## Scope

*<Provide a description and scope of the software and explain the goals, objectives and benefits of your project. This will*

*provide the basis for the brief description of your product.>*

## Definitions, Acronyms and Abbreviations

*<Provide definitions of all terms, acronyms, and abbreviations that might exist to properly interpret the SDD.*

*These definitions should be items used in the SDD that are most likely not known to the audience. >*

| **Acronyms / Abbreviations** | **Definitions** |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## References

*<List any documents, if any, which were used as sources of information for the SDD.*

*This subsection should:*

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

e.g.

1. IEEE SA - 830-1998 - IEEE Recommended Practice for Software Requirements Specifications
2. IEEE SA - 1016-1998 - IEEE Recommended Practice for Software Design Descriptions
3. The Framework and Implementation of DCP (Institutional Leadership Track)
4. The Framework and Implementation of ALTM

## Overview

*<Give a details description of your proposed systems*

## Real-time profile and modeling tools

## Innovation solution for IOT and Data analytics

# System Software Process Model and Type of Prototyping

## Introduction

# System Software Requirements

## Functional

*<Provide use case diagram In this section.>*

## Non-Functional

### Performance

### Security

## Constraints

## Interface

# System Software Architectural Design

*<This section of the SDD should describe the architectural style and the rationale or justification of your selection. The component and subsystem diagram should also be included.>*

## System Architecture

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

## Software Architecture – Component Diagram

*<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 and subsystem diagram here based on the architecture style you have chosen – see example of the three-layer internet system.]*

## Class Diagram

*<Include the use case diagram that you have developed previously in the SRS to ease reference in this SDD. The use case should show clearly your modules. In this example of CSS for RMO, it is organized by subsystem. You may also label your use case with the code as given in the SRS.>*

## Concurrency Design

# Detailed Description of Components

*<This section of the SDD describes each module or subsystem in your project. In the example of CSS of RMO, each software division is referred as a subsystem because CSS is a huge system. For the scope of this course, it is sufficient to refer each division of a system as a module as used in SRS.>*

## Complete Package Diagram

*<Include the overall package diagram of your system here [Example, RMO subsystem from Satzinger Figure 12-25 page 461]. For a small system, the subsystem can be labelled as package itself. Include all layers (view, domain and data access) for each subsystem/package. In this example, only order-entry has the three layers. Should do for all packages. The controlled classes (handler) should be in the domain layer. 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. That can be shown in the following class diagram section for respective module/package.>*

## Detailed Descriptions

*< For each module there must be ONE package diagram, ONE class diagram and several sequence diagrams based on how many use cases you have in your module. Use alt notation 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.>*

### Module <Name of Module 1>

*<Provide a brief description for the module. Give a name to the module>*

#### P001: Package <Name of Package>

*<Provide code for each package e.g. P001. Include a brief description for the package.>*

#### Class Diagram

*<Include class diagram to represent all classes in the respective module. Provide the algorithm or pseudocode for each method in each domain class (exclude controller/handler). In this example, the first sub-system/module is Order Entry.>*

#### Sequence Diagrams

*<Include sequence diagram for each respective use case in your module. For example in Figure 2.1, Module Order Entry has five use cases. Thus, there will be five or more sequence diagrams created based on use case realizations. Sequence diagrams included in this section only shows a partial of sequence diagrams in Module Order Entry. In this example only sequence diagram Create New Phone Order Scenario and Cancel an Order Scenario are shown in Figure 3.3 – 3.4. Include the final sequence diagram (not first-cut version) that comprises view layer, controller and its problem domain (entity) and data access layer. Provide code for each scenario of sequence diagram to be used in Section 6: Requirements Matrix>*

a) SD001: Sequence diagram for Create New Phone Order

### Module <Name of Module 2>

#### P002: Package <Name of Package>

#### Class Diagram

#### Sequence Diagrams

### Module <Name of the n Module>

#### Package <Name of Package>

#### Class Diagram

#### Sequence Diagrams

# User Interface Design

## 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.>*

## User Interface

*<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 modules.>*

# Requirements Matrix

The use cases and respective packages are as below.

|  | P001 | P002 | P003 | … |
| --- | --- | --- | --- | --- |
| Module 1, UC001 | X |  |  |  |
| Module 1, UC002 | X |  |  |  |
| Module 1, UC003 | X |  |  |  |
| Module 2, UC004 |  | X |  |  |
| Module 2, UC005 |  | X |  |  |
| . |  |  |  |  |
| . |  |  |  |  |
| . |  |  |  |  |
| Module *n* |  |  |  |  |

The packages and respective sequence diagrams for the scenarios are as below.

|  | SD001 | SD002 | SD003 | SD004 | SD005 | SD006 | SD007 | SD008 | … |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P001 | X | X |  |  |  |  |  |  |  |
| P002 |  |  | X |  |  |  |  |  |  |
| P003 |  |  |  | X |  |  |  |  |  |
| P004 |  |  |  |  | X | X | X |  |  |
| P005 |  |  |  |  |  |  |  | X |  |
| . |  |  |  |  |  |  |  |  |  |
| . |  |  |  |  |  |  |  |  |  |
| . |  |  |  |  |  |  |  |  |  |
| *n* |  |  |  |  |  |  |  |  |  |

The sequence diagrams and respective classes are as below.

|  | Customer | Order | Catalog | CatalogProduct | ProductItem | OrderItem | ReturnItem | InventoryItem | … |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SD001 | X | X |  |  |  |  |  |  |  |
| SD002 |  | X |  |  |  |  |  |  |  |
| SD003 |  |  | X | X |  |  |  |  |  |
| SD004 |  |  |  |  | X | X | X | X |  |
| SD005 |  | X |  |  |  |  |  | X |  |
| . |  |  |  |  |  |  |  |  |  |
| . |  |  |  |  |  |  |  |  |  |
| . |  |  |  |  |  |  |  |  |  |
| *n* |  |  |  |  |  |  |  |  |  |

**Appendices (if any)**