Software Design Description for

Ahmed Mohamed, Mohamed Ahmed, Nour Ahmed, Lobna Ahmed
Supervised by: Dr. Supervisor Name, Eng. TA Name

December 19, 2023

Table 1: Document version history

Version	Date	Reason for Change
1.0	25-Jan-2022	SDD first version's description are defined.
1.1	2-Feb-2022	Added Sequence Diagram.
1.3	5-Feb-2022	Requirement Matrix updated.

GitHub: https://github.com/MohammadHishamm/SWE-Project

Contents

Abstract

Add your project abstract here. (Word Limit 150)

1 Introduction

1.1 Purpose

In this section identify the purpose of this software design description (SDD) document and its intended audience. For example: 'This software design description (SDD) describes the architecture and system design of XX.. '.
Software design is a process by which the software requirements are translated into a representation of software components, interfaces, and data necessary for the implementation phase. This template is an annotated outline for MIU graduation projects SDD document adapted from the IEEE Standard for Information Technology-Systems Design-Software Design Descriptions. Please refer to IEEE 1016-2009 [1] for the full IEEE Recommended Practice for Software Design Descriptions document.

1.2 Scope

Provide a description of the document scope such as: This software design description (SDD) describes xx..... system design and provides the main design viewpoints of the system to communicate to key design stakeholders. This SDD document is used to record design decision and their rationale in order to avoid disputes over what was previously agreed upon.
The SDD shows how the software system will be structured to satisfy the requirements. SDD is the primary reference for code development and, therefore, it must contain all the information required by a programmer to write code. The SDD should define the overall system architecture in addition to design patterns, data model, data structures, and algorithms.
provide full identification of the system and the software to which this document applies, including, as applicable, title(s), abbreviation(s), and version number(s).

1.3 Intended audience

Identify here the stakeholders that this SDD document is intended for. Describe any security or privacy considerations associated with the use of this SDD document.

1.4 Reference Material

List any documents, including SRS and testing plan, which are used as sources of information for the SDD.

1.5 Definitions and Acronyms

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.

Term	Definition
Software Design Document (SDD)	Used as the primary medium for communicating software design information.
Design Entity	An element of a design that is structurally and functionally distinct from other elements.
Design rationale	Information capturing the reasoning of the designer that led to the system as designed, including design options, trade-offs considered, decisions made, and the justifications of those decisions. .

2 System Overview

This section shall briefly describe the system to which this document applies. It shall give a general description of the functionality, context and design of your project.Provide any background information if necessary. Include a diagram to provide high level overview of the system.

2.1 System Scope

Define The scope of your project. This section should briefly:

- Describe the system main features.
- Define the boundaries of the project.
- Define the expected outcomes of the project (A web application for xx...).
- Do not describe in details how the system works.

2.2 System objectives

In this section; write out a short list of your project main objectives.
Use as much as possible the SMART approach to writing objectives as it helps lay the groundwork to make sure you have got everything you need. SMART is an acronym that stands for:
S = Specific
M = Measurable
A = Achievable (or attainable, actionable, appropriate)
R = Realistic
T = Time-bound (or timely, traceable)

Avoid using generic objectives with no clear measurement, For example:
- To build an easy to use application. (bad example)

Instead you may use:
- To build a usable system that achieve a score equal to or higher than 68 in System Usability Scale (SUS) [2].

2.3 System Timeline

This section provides the project plan from previous phase to next phase (SRS to Technical), including the major tasks to be accomplished, their inter-dependencies, and their tentative start/stop dates. The plan also includes information on hardware, software, and resource requirements. The project plan should be accompanied by one or more PERT or GANTT charts such as the chart shown in Figure 1.
The plan should include the projects task and who is responsible for this task. Table 2 shows an example.

Table 2: Project name time plan

Id	Task	Start Date	Number of Days	Team Member
1	Collect Dataset	12/10/2020	10	X, Y
2	Work on GUI	12/21/2020	15	Z, Y
3	Pre-Processing	12/21/2020	5	X
4	Feature Extraction	12/26/2020	5	X, Z, Y
5	Classification	12/31/2020	10	Z
6	Writing Paper	01/05/2021	30	X, Z, Y
7	Experiments	01/10/2021	20	X, Z, Y

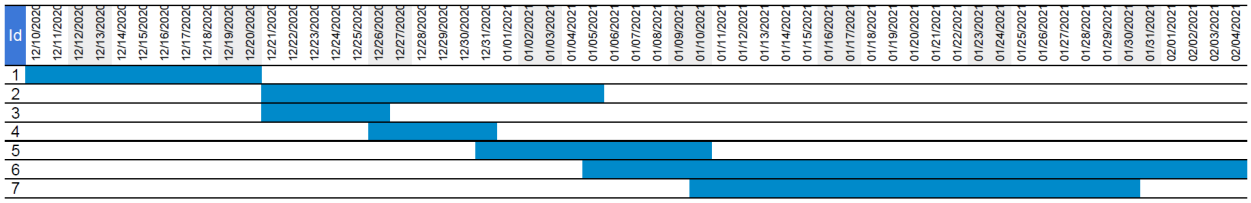

Figure 1: Project name GANTT Chart

3 Design viewpoints

3.1 Context viewpoint

Services provided by Arab Data Hub:

- Enroll in any course
- Apply to be an Instructor
- Chat with his instructor or with any other user

In this section you can:

- Write a description of the offered services and actors.
- Provide UML context diagram as in figure 2.
- Provide UML use case diagram as in figure 3.

Design concerns: Systems services and users.

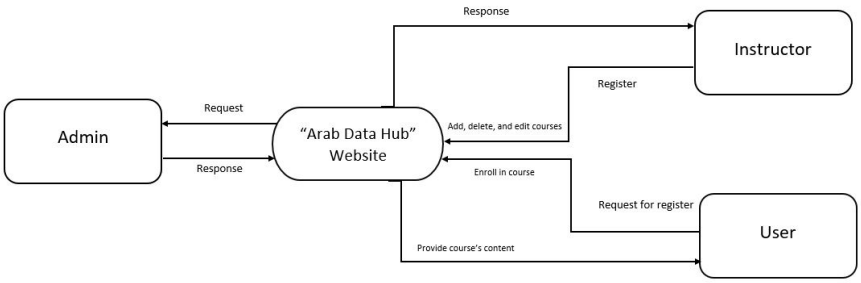

Figure 2: Context Diagram for Arab Data Hub

3.2 Composition viewpoint

The Composition viewpoint describes the way the design subject is structured into constituent parts and establishes the roles of those parts. The main purpose is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together. The Composition viewpoint supports the recording of the part-whole relationships between design entities using realization, dependency, aggregation, composition, and generalization relationships.
In this section you can:

- Provide a description of the architectural design, used architecture patterns such as MVC or layered architecture. Describe *design entities* such as: system decomposition into subsystems, components, modules; also used libraries, frameworks, software repositories.

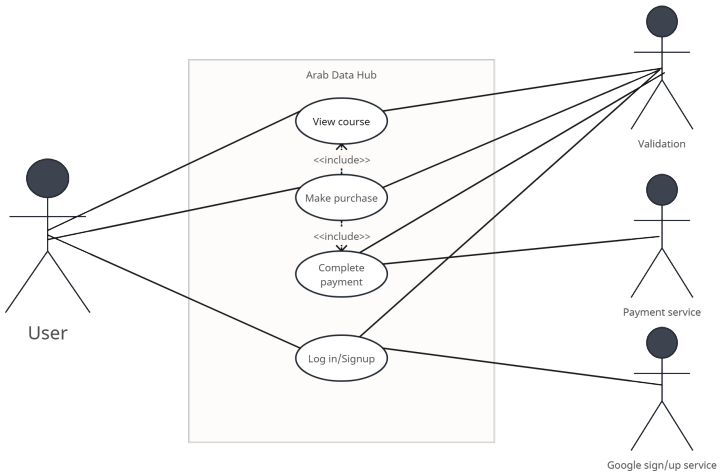

Figure 3: Use Case Diagram

- Write 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 the subsystems collaborate with each other in order to achieve the desired functionality.
- Use diagrams to show the major subsystems, data repositories and their interconnections such as: architectural design as in figure 4 and/or UML package diagram.

Design concerns: Composition and modular assembly of systems in terms of subsystems and (pluggable) components, buy vs. build, reuse of components.

3.2.1 Design Rationale

Discuss the rationale for selecting the architecture described in section 3.2 including critical issues and trade/offs that were considered. You may discuss other architectures that were considered, provided that you explain why you didn't choose them.

3.3 Logical viewpoint

The purpose of the Logical viewpoint is to elaborate existing and designed types and their implementations as classes and interfaces with their structural static relationships.

Design concerns: The Logical viewpoint is used to address the development and reuse of adequate abstractions and their implementations. For any implementation platform, a set of types is readily available for the domain abstractions of interest in a design subject, and a number of new types is to be designed, some of which may be considered for reuse. The main concern is the proper choice of abstractions and their expression in terms of existing types.

In this section you can:

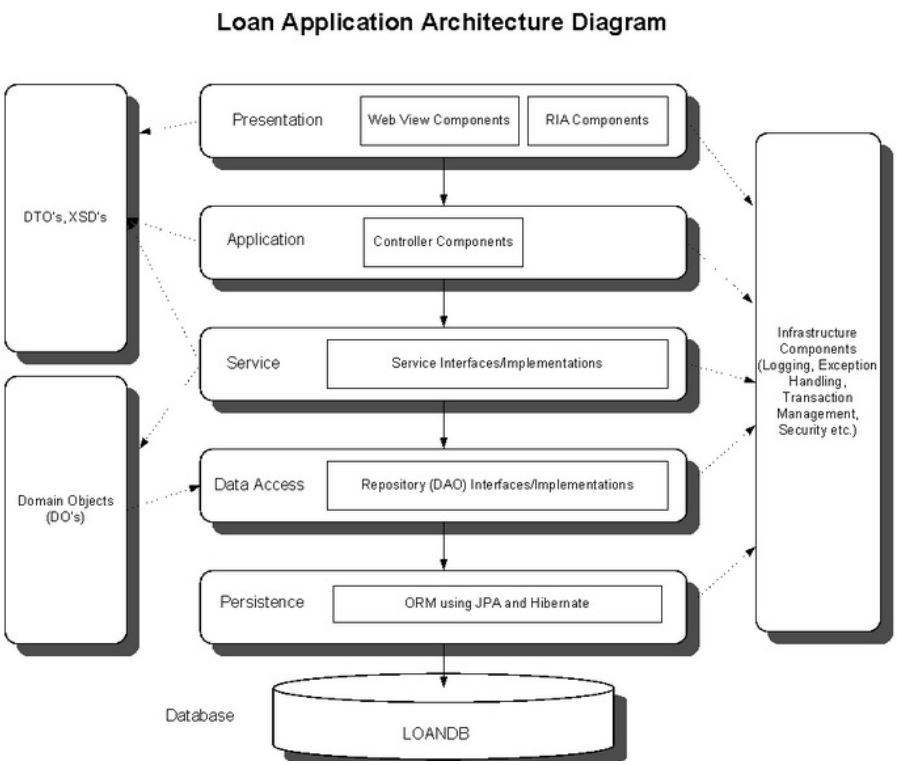

Figure 4: Architectural Design Diagram example

- Provide a UML class diagram to Illustrate static structure (classes, interfaces, and their relationships).
- Provide a table to describe each of your classes.

Table 3: ClassName

Abstract or Concrete:	xxxx
Superclasses	xxxx
Subclasses	xxxx
Purpose	xxxx
Collaborations	xxxx
Attributes	xxxx
Operations	xxxx

3.4 Patterns use viewpoint

This viewpoint addresses design ideas focusing on the used design patterns. UML class diagram and the UML package diagram can be used here to illustrate the used design patterns.

3.4.1 Design Rationale

You need to provide the design rationale for using these design patterns.

3.5 Algorithm viewpoint

In this section, provide closer look at what each component does in a more systematic way. If you gave a functional description, provide a summary of your algorithm for each function listed in section 3.3 in procedural description language (PDL) or pseudo-code. If you gave an OOP description, summarize each object member function for all the objects listed in 3.3.
Decision tables and flowcharts, "pseudo-code," or (actual) system code may also be used.

3.6 Interaction viewpoint

Provides description of Object communication, messaging using UML sequence diagrams

3.7 Interface viewpoint

The Interface viewpoint provides programmers, and testers the means to know how to correctly use the provided services. This viewpoint consists of a set of interface specifications for each entity.
NOTE: User interfaces are addressed separately in section 5.

4 Data Design

4.1 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.
You may consider the answer for the following questions:

- What is the original format of the data (paper based, excel files, old system database)?
- Identify the method used to capture the data into your system(web page forms, import documents etc..)?
- How large is the database expected to be (in numbers of rows and columns for the main entities)?
- What is the expected number of users/customers for the system?
- How is your entity keys(ids) constructed (is there a specific code or format for ID definition)?
- Does your data contain date/time, if yes which format are they presented in?

Provide data models diagrams such as Entity Relationship Diagram (ERD) in this section.

4.2 Database design description

Describe any databases (provide database schema diagram) and/or description of other data storage items.

5 Human Interface Design

5.1 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.

5.2 Screen Images

Display screenshots showing the interface from the user s perspective. These can be hand drawn or you can use a wireframe tool such as Adobe XD or you can include actual screen shots of your GUI if already developed.

5.3 Screen Objects and Actions

A discussion of screen objects and actions associated with those objects.

6 Requirements Matrix

Provide a cross reference that traces components and data structures to the requirements in your SRS document. Use a tabular format to show which system components satisfy each of the functional requirements from the SRS. Refer to the functional requirements by the numbers/codes that you gave them in the SRS as shown in Table 4.

Table 4: Requirements Ratrix

Req. ID	Req Desc	Class	Test Cases ID	Status
FR01	xxxx	class name	TC01, TC02	In Progress
FR02	xxxx	class name	TC03, TC04	Developed

7 APPENDICES

Appendices may be included, either directly or by reference, to provide supporting details that could aid in the understanding of the Software Design Document.

7.1 Github

Add screenshots from Github repository showing your project.

7.2 Other appendices as appropriate

Optional section.

References

[1] "IEEE Standard for Information Technology–Systems Design–Software Design Descriptions". In: *IEEE STD 1016-2009* (2009), pp. 1–35.

[2] John Brooke. "Sus: a "quick and dirty'usability". In: *Usability evaluation in industry* 189 (1996).