**Experiment No.: 7**

**Title:** Preparing Software Design Document (SDD)

**Batch: A3 Roll No.: 16010421119 Experiment No.: 7**

**Aim: To prepare Software Design Document (SDD)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Resources needed:** Internet Explorer, LaTex Editor **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Theory**

The SDD shows how your software will be structured to satisfy the requirements. It describes the software structure, software components, interfaces, and data necessary for the implementation phase. In a complete SDD, each requirement must be traceable to one or more components.

An SDD is a representation or model of the software system to be created. The model should provide the precise design information needed for planning, analysis, and implementation of the software system. It should represent a partitioning of the system into components and describe the important properties and relationships between them.

SDD Template:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1 INTRODUCTION**  **1.1 Design Overview**  Give a description of the design approach, highlighting essential features that allow the design to meet the stated requirements.  **1.2 Requirements Traceability Matrix**  Provide a matrix showing where each feature identified in the SRS is supported by the design components.    **2 SYSTEM ARCHITECTURAL DESIGN**  **2.1 Chosen System Architecture**  Describe the system architectural design, identifying the major component groupings and the interfaces (both internal and external).  Make sure to identify any significant technical risks, and identify contingency plans for each.  **2.2 Discussion of Alternative Designs**  Discuss in a reasonable level of detail other design options explored, and the reasons for not choosing them.  **2.3 System Interface Description**  Describe the system interfaces in detail: O/S interface, files, networking, libraries, graphics libraries etc. (\*Describe the user interface in section 4.)  **3 DETAILED DESCRIPTION OF COMPONENTS**  **3.n Component-n**  For each component, the following items should be described here as appropriate: responsibilities, constraints, composition, interactions, and resources. Use appropriate diagrams or other notation to describe your design.  **4 USER INTERFACE DESIGN**  In this section describe the design of the user interface in detail.  **4.1 Description of the User Interface**  **4.1.1 Screen Images**  Show the design of layout and menus for each screen.  **4.1.2 Objects and Actions**  Identify all the objects on each screen, and define the actions to be taken by each object for each event.  **5. System Architecture**  In this section high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components are provided.  Use cases from the SRS Document  **Use Case Specification using template**   |  |  |  |  | | --- | --- | --- | --- | | **Use Case ID:** | **1** | | | | **Use Case Name:** |  | | | | Created By: |  | Last Updated By: |  | | Date Created: |  | Date Last Updated: |  |  |  |  | | --- | --- | | Primary Actors: |  | | Secondary Actors: |  | | Description: |  | | Trigger: |  | | Preconditions: |  | | Postconditions: |  | | Normal Flow: |  | | Alternative Flows: |  | | Exceptions: |  | | Includes: |  | | Priority: |  | | Frequency of Use: |  | | Business Rules: |  | | Special Requirements: |  | | Open Issues |  | | Assumptions: |  | | Notes and Issues: |  |   **6. Data flow specifications**  DFD is created from the SRS document provided.  **6.1 Level 0 DFD with description**  **6.2 Level 1 DFD with description** |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Procedure:**

**1.** Prepare SDD document for chosen problem definition in LaTeX.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Results:** Software Design Document (SDD) in given format

\documentclass{article}

\usepackage{graphicx}

\title{Software Design Document}

\author{}

\date{}

\begin{document}

\maketitle

\section{Introduction}

\subsection{Design Overview}

Give a description of the design approach, highlighting essential features that allow the design to meet the stated requirements.

\subsection{Requirements Traceability Matrix}

Provide a matrix showing where each feature identified in the SRS is supported by the design components.

\section{System Architectural Design}

\subsection{Chosen System Architecture}

Describe the system architectural design, identifying the major component groupings and the interfaces (both internal and external). Make sure to identify any significant technical risks, and identify contingency plans for each.

\subsection{Discussion of Alternative Designs}

Discuss in a reasonable level of detail other design options explored, and the reasons for not choosing them.

\subsection{System Interface Description}

Describe the system interfaces in detail: O/S interface, files, networking, libraries, graphics libraries etc.

\section{Detailed Description of Components}

\subsection{Component-n}

For each component, describe responsibilities, constraints, composition, interactions, and resources. Use appropriate diagrams or other notation to describe your design.

\section{User Interface Design}

\subsection{Description of the User Interface}

Describe the design of the user interface in detail.

\subsubsection{Screen Images}

Show the design of layout and menus for each screen.

\subsubsection{Objects and Actions}

Identify all the objects on each screen, and define the actions to be taken by each object for each event.

\section{System Architecture}

In this section high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components are provided.

Use cases from the SRS Document.

\section{Data Flow Specifications}

\subsection{Level 0 DFD with Description}

The Level 0 DFD provides an overview of the system's major processes and their interactions.

\subsection{Level 1 DFD with Description}

The Level 1 DFD elaborates on the processes identified in Level 0, breaking them down into more detailed subprocesses.

\end{document}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Questions:**

**1. Explain Architecture design patterns and styles with examples.**

Architecture design patterns and styles are fundamental principles and guidelines that help organize and structure the components of a software system. These patterns provide solutions to common design problems, promote code reusability, maintainability, and scalability. Here are some key architecture design patterns and styles, along with examples:

Layered Architecture:

Description: Divides the software into layers, where each layer has a specific responsibility, and communication happens only between adjacent layers.

Example: In a web application, the presentation layer (UI), business logic layer, and data access layer can be separated into distinct layers.

Model-View-Controller (MVC):

Description: Separates an application into three interconnected components - Model (data and business logic), View (presentation and user interface), and Controller (handles user input and manages the flow).

Example: Web frameworks like Ruby on Rails, Django, and Spring MVC follow the MVC pattern.

Microservices Architecture:

Description: Breaks down a monolithic application into small, independent services that communicate through APIs. Each service is responsible for a specific business capability.

Example: Netflix, Amazon, and Spotify use microservices architecture to build scalable and maintainable systems.

Event-Driven Architecture:

Description: Components communicate through events, and the system responds to events by triggering actions. This promotes loose coupling and scalability.

Example: Message queues like Apache Kafka or event-driven systems like AWS Lambda are used to implement event-driven architectures.

Service-Oriented Architecture (SOA):

Description: Organizes software as a collection of loosely coupled, reusable services. These services can be independently developed, deployed, and scaled.

Example: An e-commerce system might have separate services for user authentication, product catalog, and payment processing.

Hexagonal (Ports and Adapters) Architecture:

Description: Separates the application into internal components (hexagon) and external components (ports and adapters). The hexagon contains business logic, while adapters connect it to external systems.

Example: A hexagonal architecture can be applied to systems where external components like databases, APIs, or UIs are interchangeable.

These design patterns and styles provide a foundation for building robust, scalable, and maintainable software systems by addressing common architectural challenges. The choice of a specific pattern or style depends on the requirements and constraints of the given project.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Outcomes: CO3 Demonstrate requirements, modeling and design of a system**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Conclusion:** We learnt about the Software Design and made a SDD Document

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of faculty in-charge with date**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**References:**

**Books:**

1. Roger S. Pressman, Software Engineering: A practitioners Approach, 7th Edition, McGraw Hill, 2010.
2. Technical report on Guidelines for Documents Produced by Student Projects In Software Engineering based on IEEE standards
3. Timothy C. Lethbridge, Robert Laganiere “ Object-Oriented Software Engineering – A practical software development using UML and Java”, Second Edition, Tata McGraw-Hill, New Delhi,2004