

**Batch: A4 Roll No.: 1914078 Experiment No.: 6**

**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

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

**Questions:**

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

1. Layered pattern
   1. This pattern can be used to structure programs that can be decomposed into groups of subtasks, each of which is at a particular level of abstraction. Each layer provides services to the next higher layer.
   2. General desktop applications.
   3. E commerce web applications.
2. Client-server pattern
   1. This pattern consists of two parties; a **server** and multiple **clients**. The server component will provide services to multiple client components. Clients request services from the server and the server provides relevant services to those clients. Furthermore, the server continues to listen to client requests.
3. Master-slave pattern
   1. This pattern consists of two parties; **master** and **slaves**. The master component distributes the work among identical slave components, and computes a final result from the results which the slaves return.
4. Peer-to-peer pattern
   1. In this pattern, individual components are known as **peers**. Peers may function both as a **client**, requesting services from other peers, and as a **server**, providing services to other peers. A peer may act as a client or as a server or as both, and it can change its role dynamically with time.
   2. File sharing networks
   3. Cryptocurrency based protocols
5. Model-view-controller pattern

This pattern, also known as MVC pattern, divides an interactive application in to 3 parts as,

1. **model** — contains the core functionality and data
2. **view** — displays the information to the user (more than one view may be defined)
3. **controller** — handles the input from the user

This is done to separate internal representations of information from the ways information is presented to, and accepted from, the user. It decouples components and allows efficient code reuse.

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

**Outcomes:**

**Demonstrate requirements, modeling and design of a system.**

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

**Conclusion:**

Prepared a Software Design Document for the Student Networking Website.

**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