

|  |
| --- |
| **Software Engineering and Project Management** |
| Workbook |
| **Department of Computer Science and Engineering**  **The NorthCap University, Gurugram** |

**Software Engineering and Project Management**

**CSL229**

**Dr. Neeti Kashyap**



Department of Computer Science and Engineering

NorthCap University, Gurugram- 122001, India

Session 2022-23

*Published by:*

**School of Engineering and Technology**

**Department of Computer Science & Engineering**

**The NorthCap University Gurugram**

**• Laboratory Manual is for Internal Circulation only**

**© Copyright Reserved**

*No part of this Practical Record Book may be*

*reproduced, used, stored without prior permission of The NorthCap University*

Copying or facilitating copying of lab work comes under cheating and is considered as use of unfair means. Students indulging in copying or facilitating copying shall be awarded zero marks for that particular experiment. Frequent cases of copying may lead to disciplinary action. Attendance in lab classes is mandatory.

Labs are open up to 7 PM upon request. Students are encouraged to make full use of labs beyond normal lab hours.

**PREFACE**

Software Engineering and Project Management Lab Manual is designed to meet the course and program requirements of NCU curriculum for B.Tech III year students of CSE branch. The concept of the lab work is to give brief practical experience for basic lab skills to students. It provides the space and scope for self-study so that students can come up with new and creative ideas.

The Lab manual is written on the basis of “teach yourself pattern” and expects that students who come with proper preparation should be able to perform the experiments without any difficulty. Brief introduction to each experiment with information about self-study material is provided. The laboratory exercises include understanding Unified Modeling Language and its implementation using open source tool. Building basic workflows for all the system’s views like Requirements’ view, structural and behavioral view and finally deployment view are discussed as part of the curriculum. Finally, the students would require doing guided and unguided projects. Students are expected to come thoroughly prepared for the lab. General disciplines, safety guidelines and report writing are also discussed.

The lab manual is a part of curriculum for TheNorthCap University, Gurugram. Teacher’s copy of the experimental results and answers for the questions are available as sample guidelines.

We hope that the lab manual would be useful to students of CSE, ECE branches and the author requests the readers to kindly forward their suggestions / constructive criticism for further improvement of the work book.

Author expresses deep gratitude to Members, Governing Body-NCU for encouragement and motivation.

**Authors**

**The NorthCap University**

**Gurugram, India**

|  |  |
| --- | --- |
| **CONTENTS** | |
| **Sr No** | **Details** |
| **1** | **Syllabus** |
| **2** | **Introduction** |
| **3** | **Lab Requirement** |
| **4** | **General Instructions** |
| **5** | **List of Experiments** |
| **6** | **List of Projects** |
| **7** | **Rubrics** |
| **8** | **Annexure 1 (Format of Lab Report)** |
| **9** | **Annexure 2 (Format of Project Report)** |

1. **INTRODUCTION**

That ‘learning is a continuous process’ cannot be over emphasized. The theoretical knowledge gained during lecture sessions need to be strengthened through practical experimentation. Thus practical makes an integral part of a learning process.

The purpose of conducting experiments can be stated as follows:

1.To familiarize the students with the basic concepts of Unified Modelling Language. The lab sessions will be based on exploring the concepts discussed in class.

2. Learn the object-oriented analysis phase by understanding the methods of class elicitation and finding the classes in an object- oriented Systems

3. Design the Interaction diagrams, sequence and collaboration diagrams with the help of software engineering tool.

4. Learn to design the test cases.

5. Hands-on experience

1. **LAB REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Requirements** | **Details** |
| **1** | **Software Requirements** | **Open Source Tools**  <https://app.diagrams.net/> |
| **2** | **Operating System** | **Modern Operating System** |
| **3** | **Hardware Requirements** | ●**Modern Operating System:**  ●**x86 64-bit CPU (Intel / AMD architecture)**  ●**4 GB RAM.**  ●**5 GB free disk space** |
| **4** | **Required Bandwidth** | NA |

1. **GENERAL INSTRUCTIONS**

**a.General discipline in the lab**

●Students must turn up in time and contact concerned faculty for the experiment they are supposed to perform.

●Students will not be allowed to enter late in the lab.

●Students will not leave the class till the period is over.

●Students should come prepared for their experiment.

●Experimental results should be entered in the lab report format and certified/signed by concerned faculty/ lab Instructor.

●Students must get the connection of the hardware setup verified before switching on the power supply.

●Students should maintain silence while performing the experiments. If any necessity arises for discussion amongst them, they should discuss with a very low pitch without disturbing the adjacent groups.

●Violating the above code of conduct may attract disciplinary action.

●Damaging lab equipment or removing any component from the lab may invite penalties and strict disciplinary action.

**b.Attendance**

●Attendance in the lab class is compulsory.

●Students should not attend a different lab group/section other than the one assigned at the beginning of the session.

●On account of illness or some family problems, if a student misses his/her lab classes, he/she may be assigned a different group to make up the losses in consultation with the concerned faculty / lab instructor. Or he/she may work in the lab during spare/extra hours to complete the experiment. No attendance will be granted for such case**.**

**c.Preparation and Performance**

●Students should come to the lab thoroughly prepared on the experiments they are assigned to perform on that day. Brief introduction to each experiment with information about self study reference is provided on LMS.

●Students must bring the lab report during each practical class with written records of the last experiments performed complete in all respect.

●Each student is required to write a complete report of the experiment he has performed and bring to lab class for evaluation in the next working lab. Sufficient space in work book is provided for independent writing of theory, observation, calculation and conclusion.

●Students should follow the Zero tolerance policy for copying / plagiarism. Zero marks will be awarded if found copied. If caught further, it will lead to disciplinary action.

●Refer **Annexure 1** for Lab Report Format

1. **LIST OF EXPERIMENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No.** | **Title of the Experiment** | **Software/Hardware based** | **COs** | **Time Required** |
| 1. | Design the software requirements specification (SRS). | Software | 2 | 4 hours |
| 2. | Develop understanding of system modeling by using Data Flow Diagram (DFD) using a CASE tool. | Software | 3 | 2 hours |
| 3. | Design of activity diagram by using a CASE tool. | Software | 3 | 2 hours |
| 4. | Learn the object-oriented analysis phase by understanding the methods of class elicitation and finding the classes in an object- oriented system. | Software | 3 | 2 hours |
| 5. | Design the Interaction diagrams, sequence and collaboration diagrams with the help of software engineering tool. | Software | 3 | 2 hours |
| 6. | Design Test Cases and Test Scenario using Excel. | Software | 4 | 2 hours |
| 7. | To create project plan document for the assigned project | Software | 6 | 2 hours |
| 8. | To design the Work Break Down Structure for assigned Project | Software | 6 | 2 hours |
| 9. | To create resources and assign resources to tasks | Software | 6 | 2 hours |
| 10. | To design the Gantt Chart | Software | 6 | 2 hours |
| 11. | To find and view the critical path | Software | 6 | 2 hours |
| 12. | To calculate the total costs of the project by tasks and resources | Software | 6 | 2 hours |
| **Value Added Experiments** | | | | |
| 1. | Execute automation testing using an appropriate tool. | Automation tool (Web/Mobile etc.) | 4 | 2 hours |
| 2. | Design User story/epic using tool | Taiga.io | 1 | 2 hours |

**5.LIST OF PROJECTS**

|  |  |  |
| --- | --- | --- |
| **Sr No.** | **Project Title** | **Mapped CO** |
|  | Consider the development of E commerce website and design all the software artifacts for the same. | CO1, CO2, CO3, CO4, CO5 |
|  | Identify all the requirements of ingenious hospital management system and design the complete SRS for the same. | CO1, CO2 |
|  | Consider the scenario of how to increase the number of active users and revenue for Google pay and draw use case diagram and class diagram for the same. | CO1, CO2, CO3 |
|  | Creating and Managing the various projects on open source project management tool. | CO5 |

**6.RUBRICS**

|  |  |
| --- | --- |
| **Marks Distribution** | |
| **Continuous Evaluation(50 Marks)** | **Project Evaluations/End term Viva (20 Marks)** |
| **Each experiment shall be evaluated for 10 marks and at the end of the semester proportional marks shall be awarded out of total 50.** | **Both the projects shall be evaluated and at the end of the semester viva will be conducted related to the projects as well as concepts learned in labs and this component carries 20 marks.** |
| **Following is the breakup of 10 marks for each**  **4 Marks: Observation & conduct of experiment. Teachers may ask questions about experiments.**  **3 Marks: For report writing**  **3 Marks: For the 15 minutes quiz/viva to be conducted in every lab.** |

**Annexure 1**

****

**Software Engineering and Project Management**

**(CSL229)**

Lab Practical Report

Faculty name: **Dr. Neeti Kyashyap**

Student name: **Avtar Singh**

Roll No.: **20csu241**

Semester: **VI**

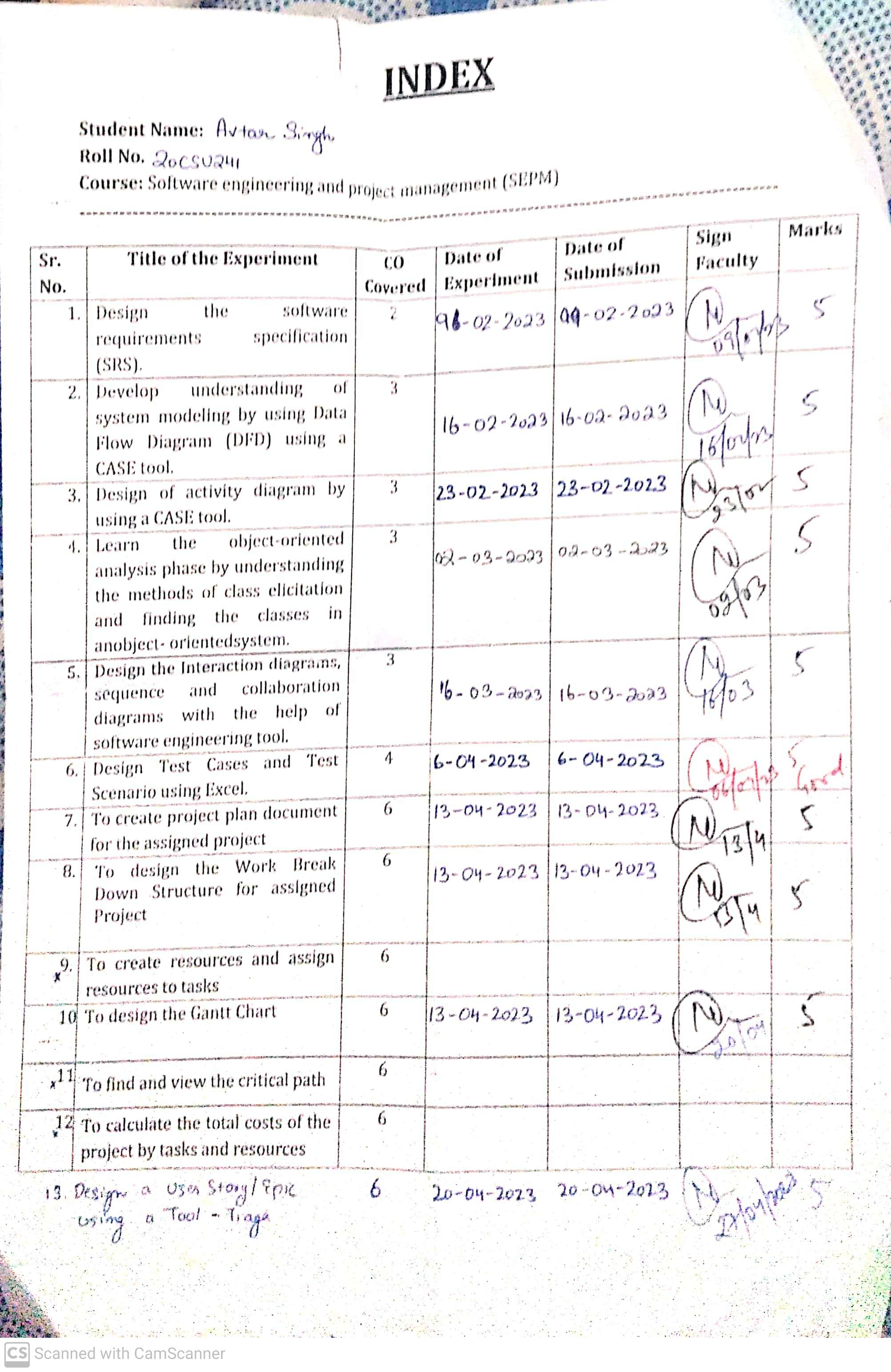
Group: **FS-B**

Department of Computer Science and Engineering

The NorthCap University, Gurugram- 122001, India

Session 2021-22

**INDEX**

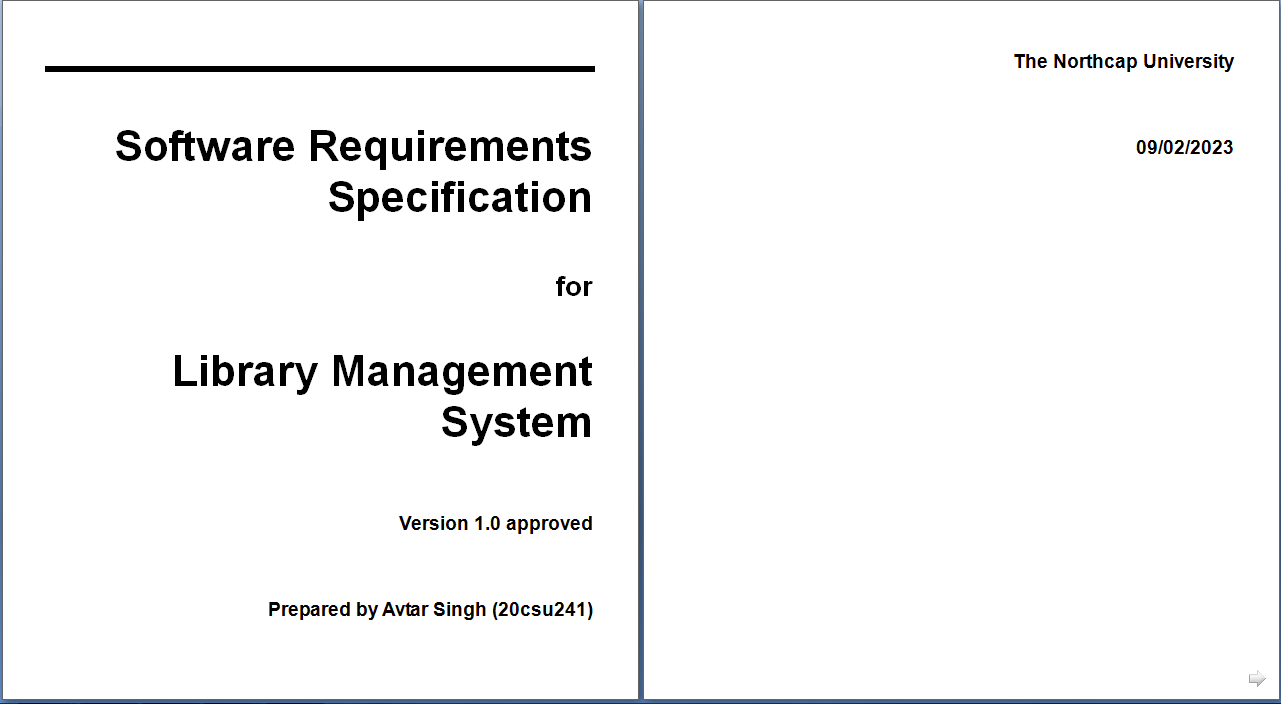
****

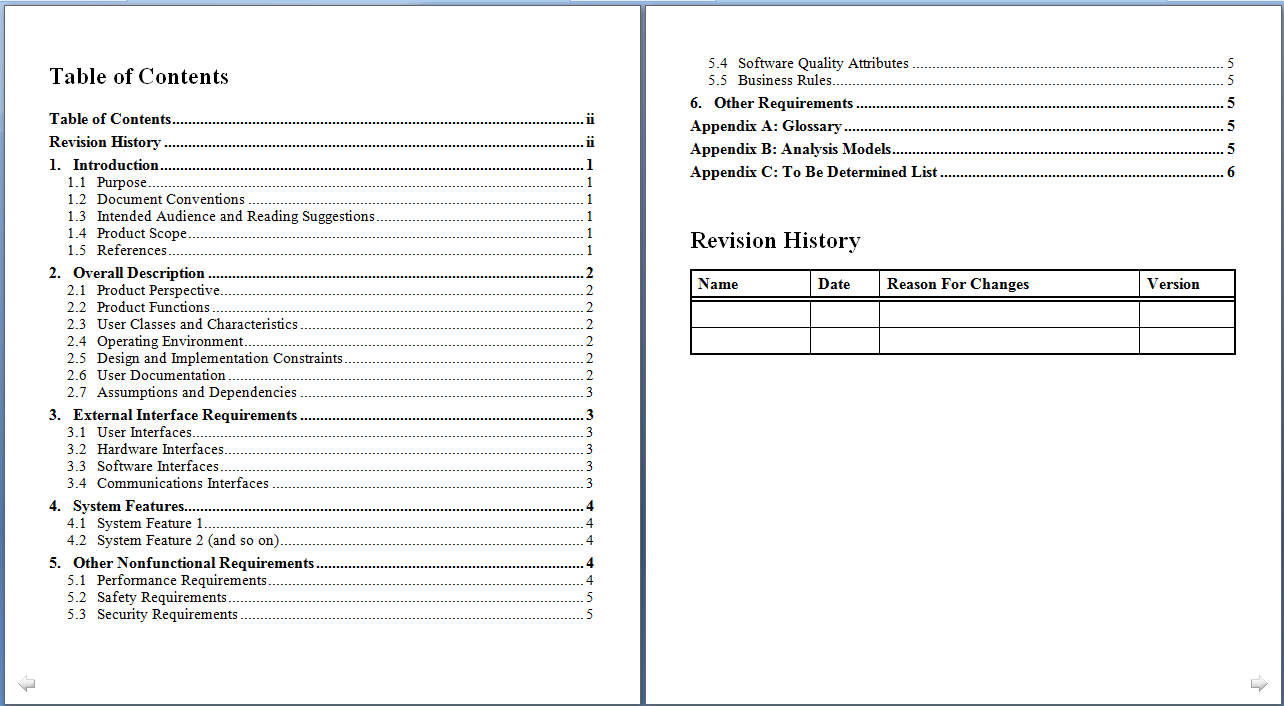
**EXPERIMENT NO. 1**

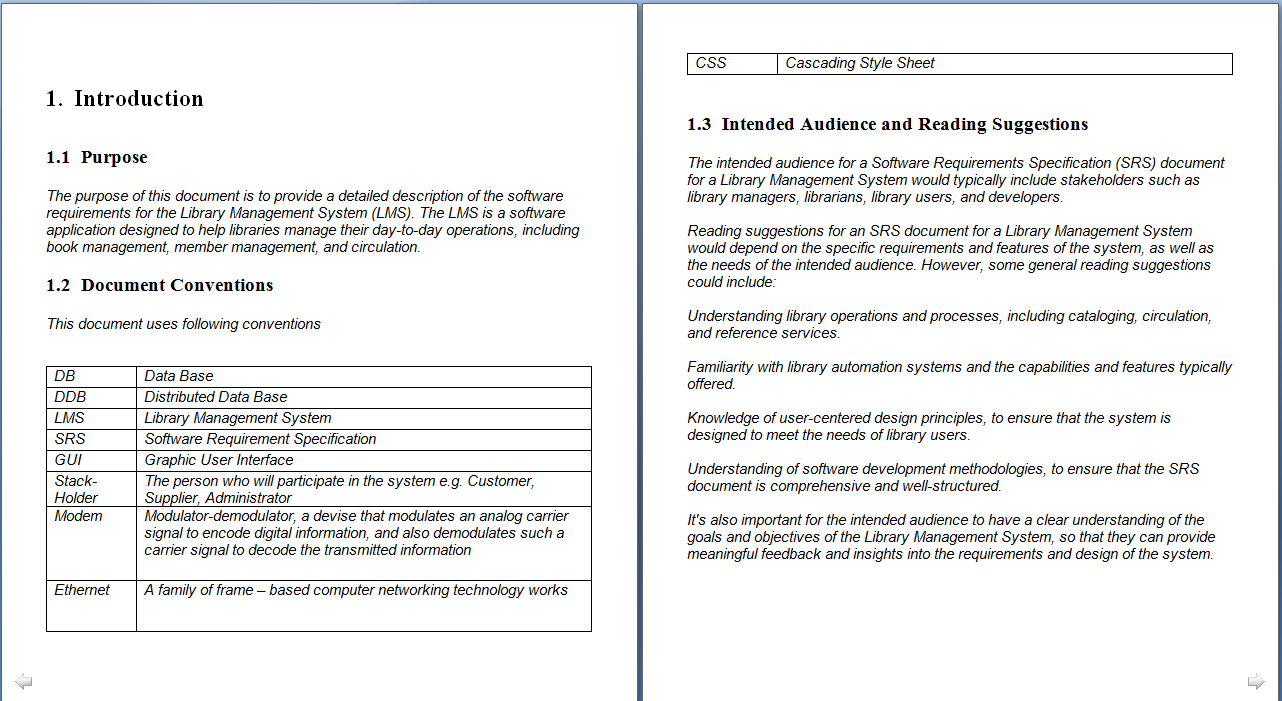
|  |
| --- |
| **Student Name and Roll Number: Avtar Singh - 20csu241** |
| **Semester /Section: FS-B VI sem** |
| **Link to Code:** |
| **Date: 09.02.23** |
| **Marks:** |
| **Objective:**  ·Introduce the lab environment and tools used in the software engineering lab.  ·Discuss the Project & learn how to write project definition. |
| **Outcome:**  Students will understand the importance of SEPM and about various software artifacts  Students will learn to perform feasibility analysis by designing Feasibility Study Document |
| **Problem Statement:**   * Introduction to the lab plan and objectives. * Software Engineering and Project definition and, artifacts discussion. |
| **Background Study:**  The software engineer is a key person analyzing the business, identifying opportunities for improvement, and designing information systems to implement these ideas. It is important to understand and develop through practice the skills needed to successfully design and implement new software systems.  1.In this lab we will practice the software development life cycle (project management, requirements engineering, systems modeling, software design, prototyping, and testing) using CASE tools within a teamwork environment.  2.UML notation is covered in this lab as the modeling language for analysis and design. |

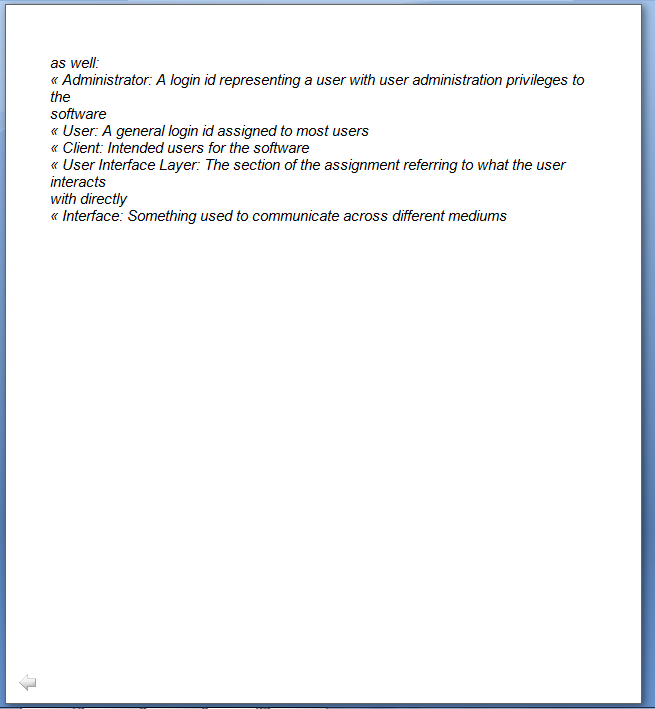
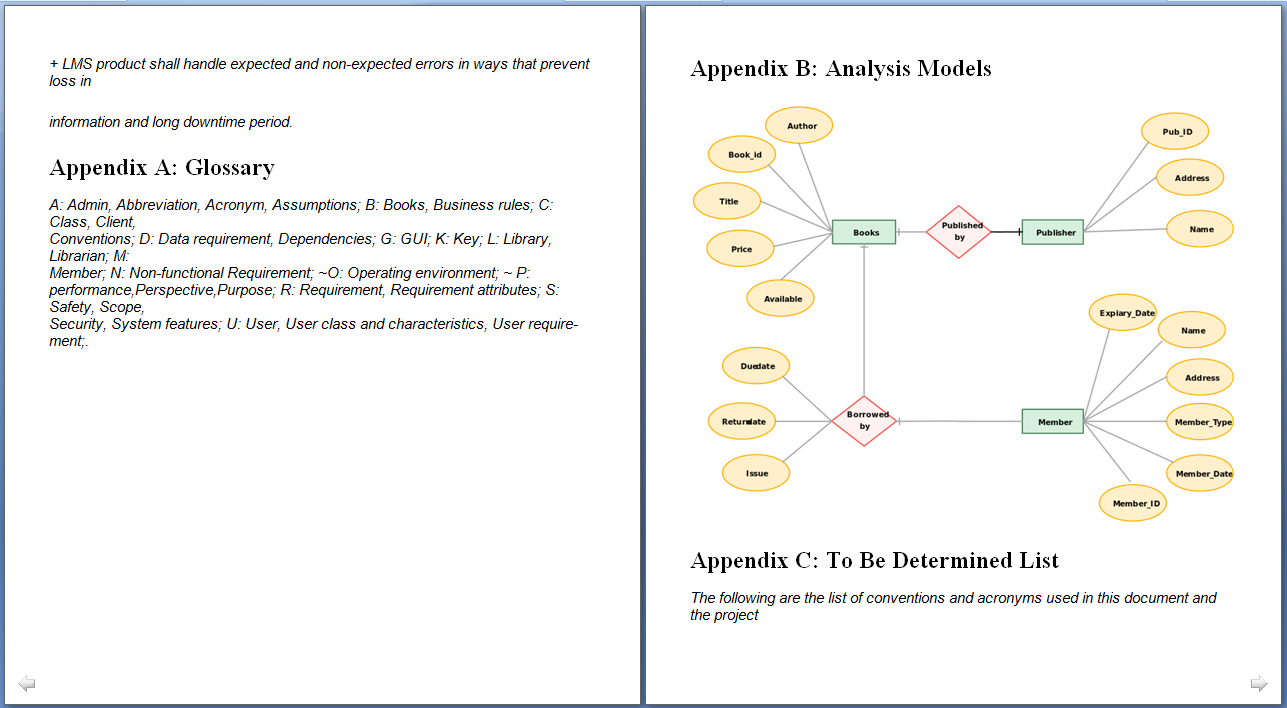
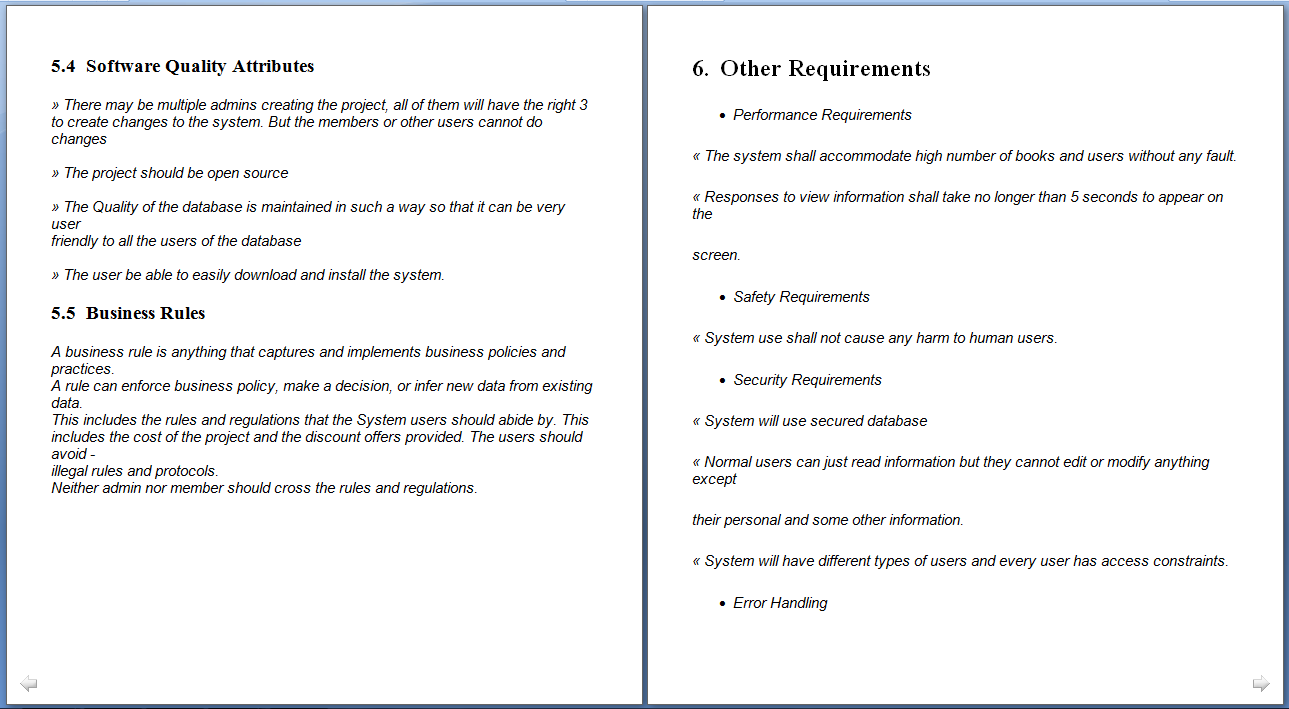
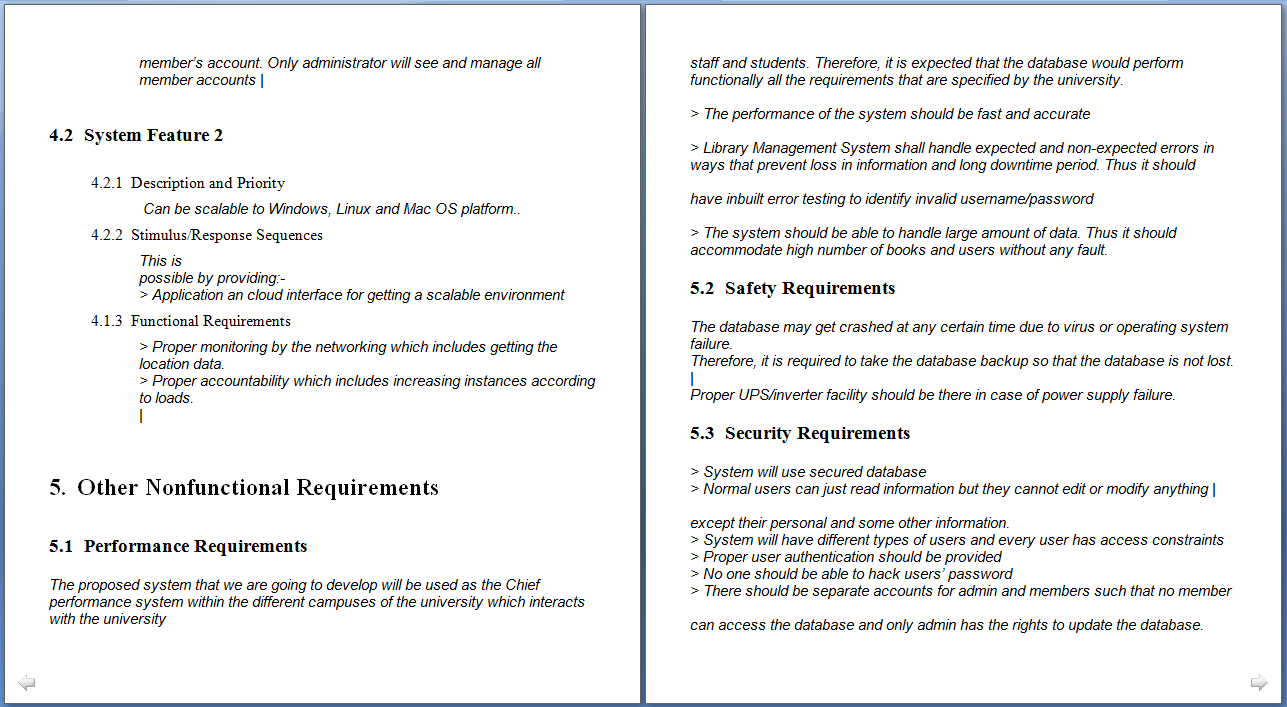
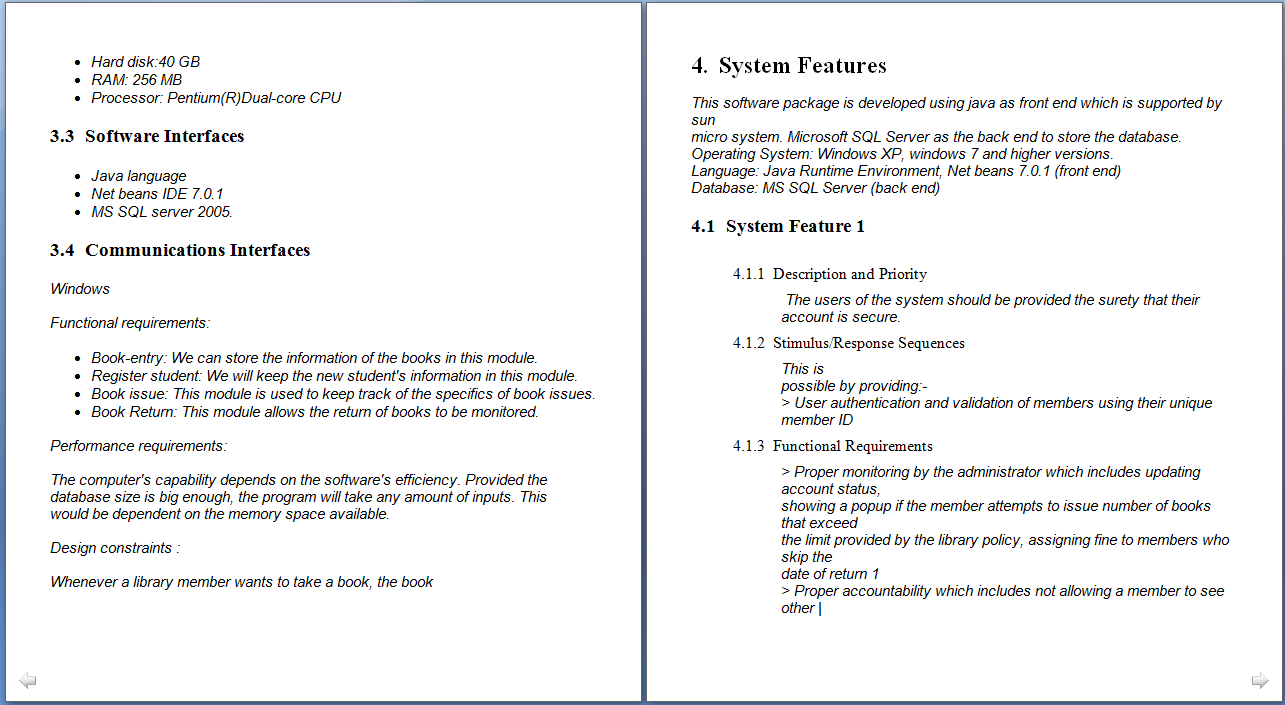
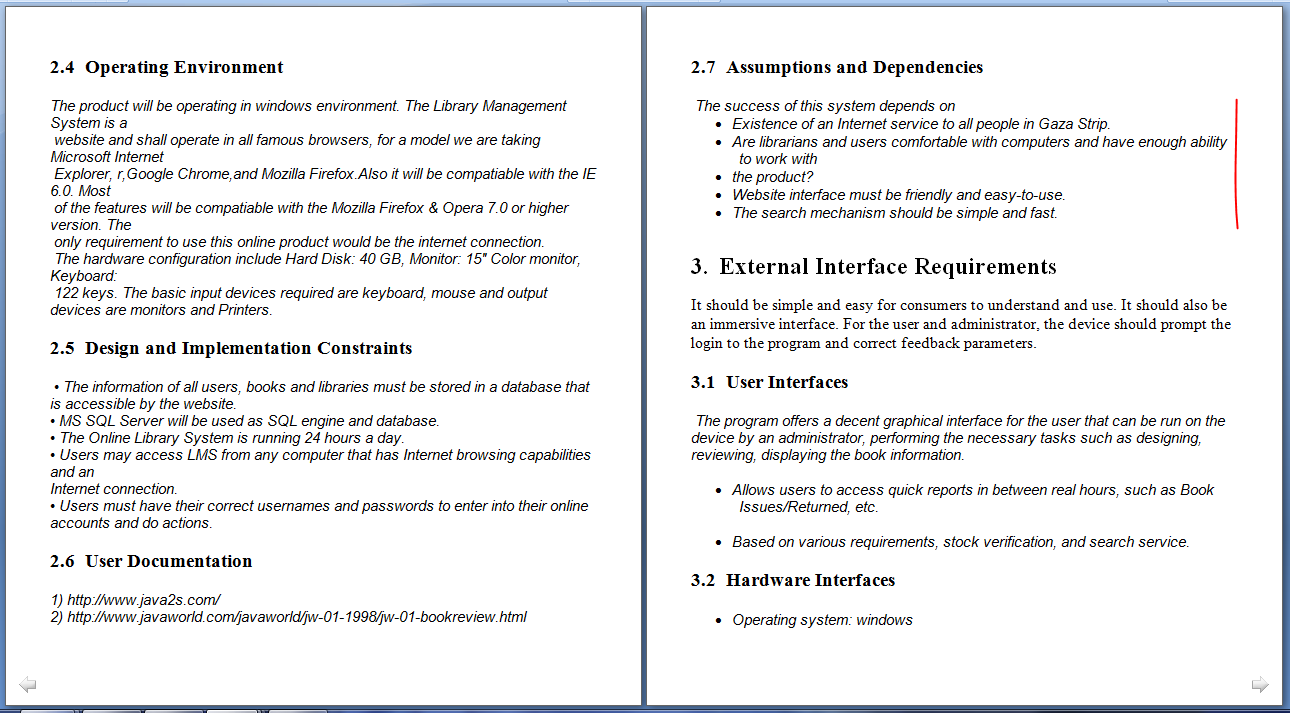
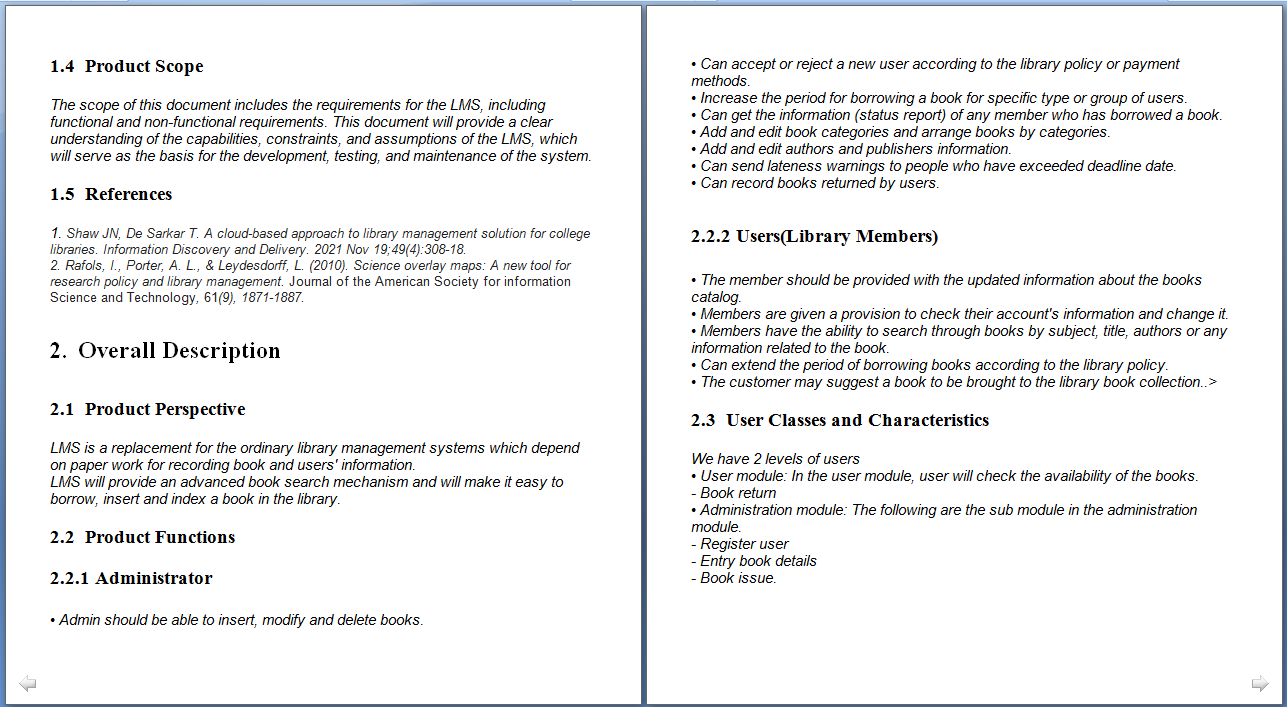
**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

****

****

****

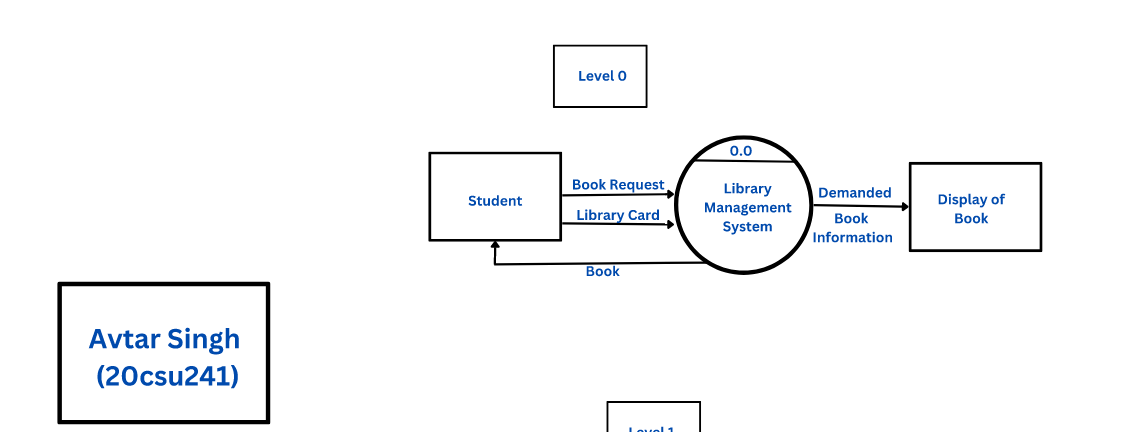
****

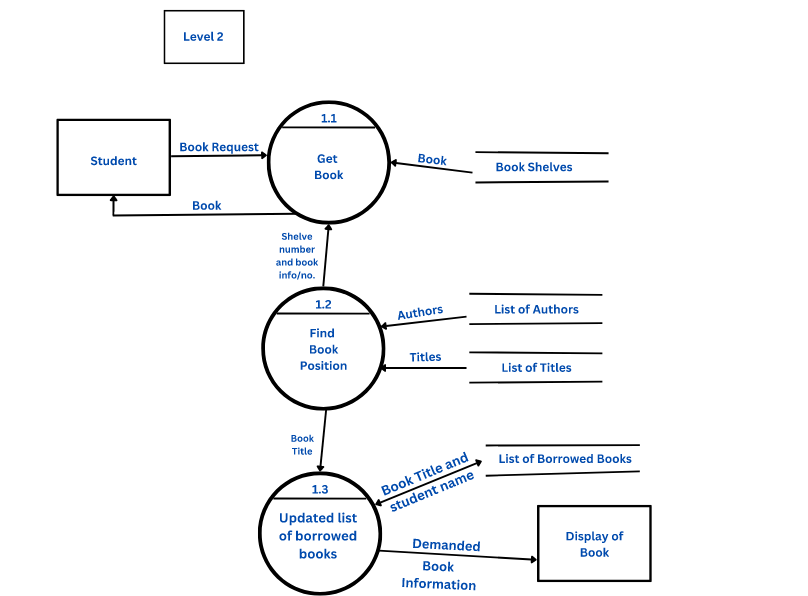
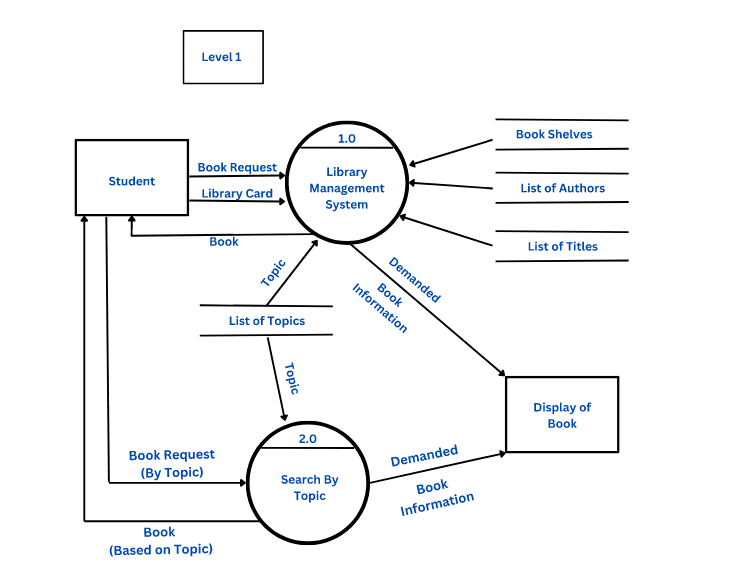
**EXPERIMENT NO. 2**

|  |
| --- |
| **Student Name and Roll Number:Avtar Singh - 20csu241** |
| **Semester /Section: VI FS-B** |
| **Date : 16.02.23** |
| **Faculty Signature:** |
| **Marks:** |
| **Objective:**  To identify best suited process model for the given statement. |
| **Outcome:**  Student will understand the importance of process models as per business use case and shall define the suitable process model for given project statement. |
| **Problem Statement:**  Develop understanding of system modeling by using Data Flow Diagram (DFD) using a CASE tool. |
| **Background Study:**  A Software Process Model  oForms a common understanding of activities among the software developers.  oHelps in identifying inconsistencies, redundancies, and omissions in the development process.  oHelps in tailoring a process model for specific projects.  oThe development team must identify a suitable life cycle model and then adhere to it.  oHelps monitor the progress of the project otherwise the project manager would have to depend on the guesses of the team members. This usually leads to a problem known as the 99% complete syndrome. |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

****

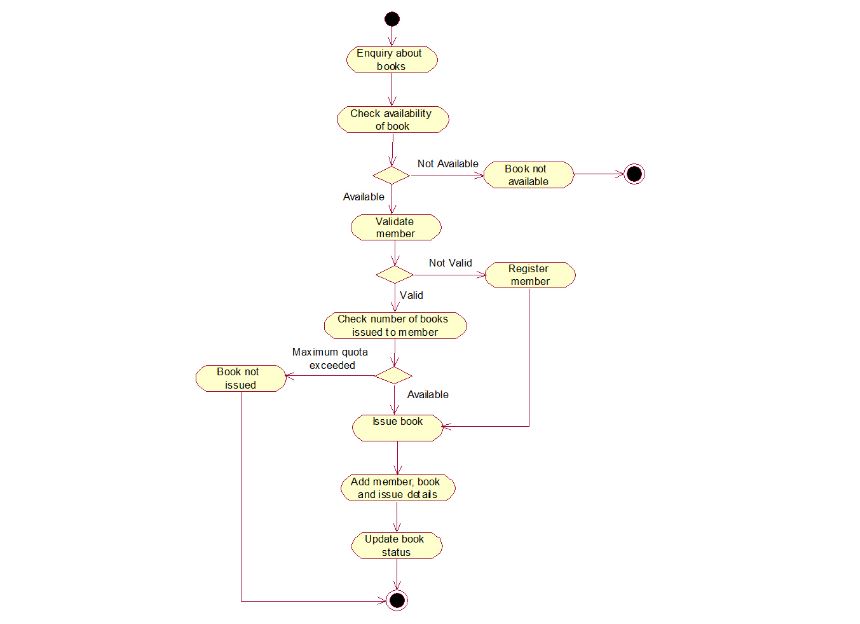
****

**EXPERIMENT NO. 3**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh - 20csu241** |
| **Semester /Section: Vi FS-B** |
| **Link to Code:** |
| **Date: 23.02.23** |
| **Faculty Signature:** |
| **Marks:** |
| **Objective:**  ·Gain a deeper understanding of the Software Requirement Specification phase and the Software Requirement Specification (SRS).  ·Learn to write requirements and specifications. |
| **Outcome:**  Students will understand the importance of SRS and design the SRS as per their system’s requirement |
| **Problem Statement:**  Design of activity diagram by using a CASE tool. |
| **Background Study:**  **Procedure:**  Step 1:  Introduction: Purpose Identify the product whose software requirements are specified in this document. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.Describe the different types of user that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type. Project Scope Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here. An SRS that specifies the next release of an evolving product should contain its own scope statement as a subset of the long-term strategic product vision.  Step 2: **Overall Description**Product Perspective Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful. Product Features Summarize the major features the product contains or the significant functions that it performs or lets the user perform. Only a high level summary is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or a class diagram, is often effective. User Classes and Characteristics Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the favored user classes from those who are less important to satisfy. Operating Environment Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist. Design and Implementation Constraints Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software). **Step 3:****System Features** This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product. System Feature 1 Don’t really say “System Feature 1.” State the feature name in just a few words.  1 Description and Priority  Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).  2 Stimulus/Response Sequences  List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.  3 Functional Requirements  Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary.  <Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>  REQ-1:  REQ-2: Step 4:**External Interface Requirements**User Interfaces Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification. Hardware Interfaces Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used. Software Interfaces Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint. Communications Interfaces Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms. **Nonfunctional Requirements**Performance Requirements If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features. Safety Requirements Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied. Security Requirements Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied. Software Quality Attributes Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning. **Other Requirements** Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project. |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

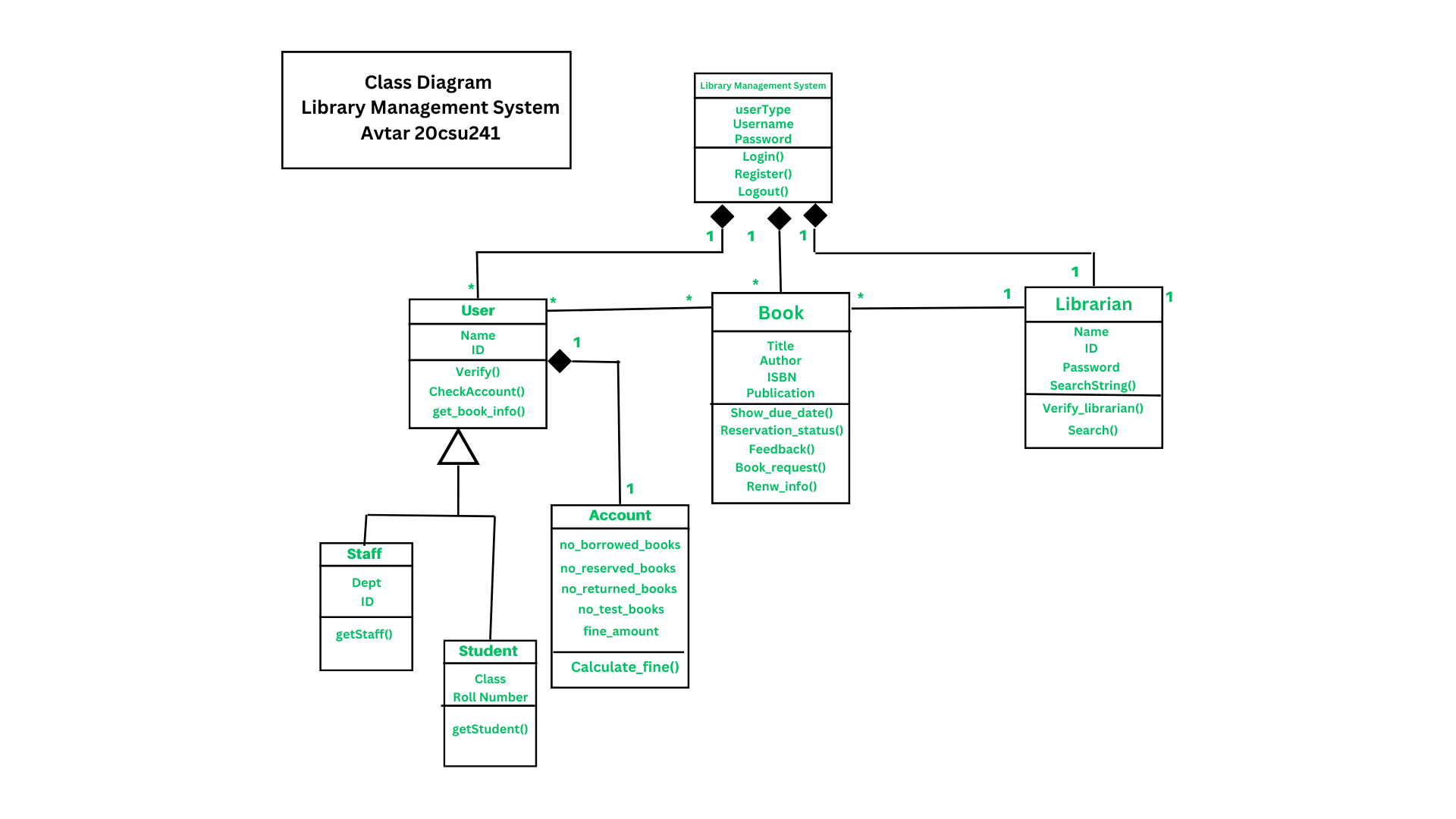
****

**EXPERIMENT NO. 4**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh - 20csu241** |
| **Semester /Section: VI Fs-B** |
| **Link to Code:** |
| **Date: 02.03.23** |
| **Faculty Signature:** |
| **Marks:** |
| **Objective:**  •Study the benefits of visual modeling.  •Learn use case diagrams: discovering actors and discovering use cases.  •Practice use cases diagrams using CASE tool. |
| **Outcome:**  Students will understand the importance of SEPM and about various software artifacts  Students will learn to perform feasibility analysis by designing Feasibility Study Document |
| **Problem Statement:**  Learn the object-oriented analysis phase by understanding the methods of class elicitation and finding the classes in an object- oriented system. |

**Student Work Area**

**Algorithm/Flowchart/Code**

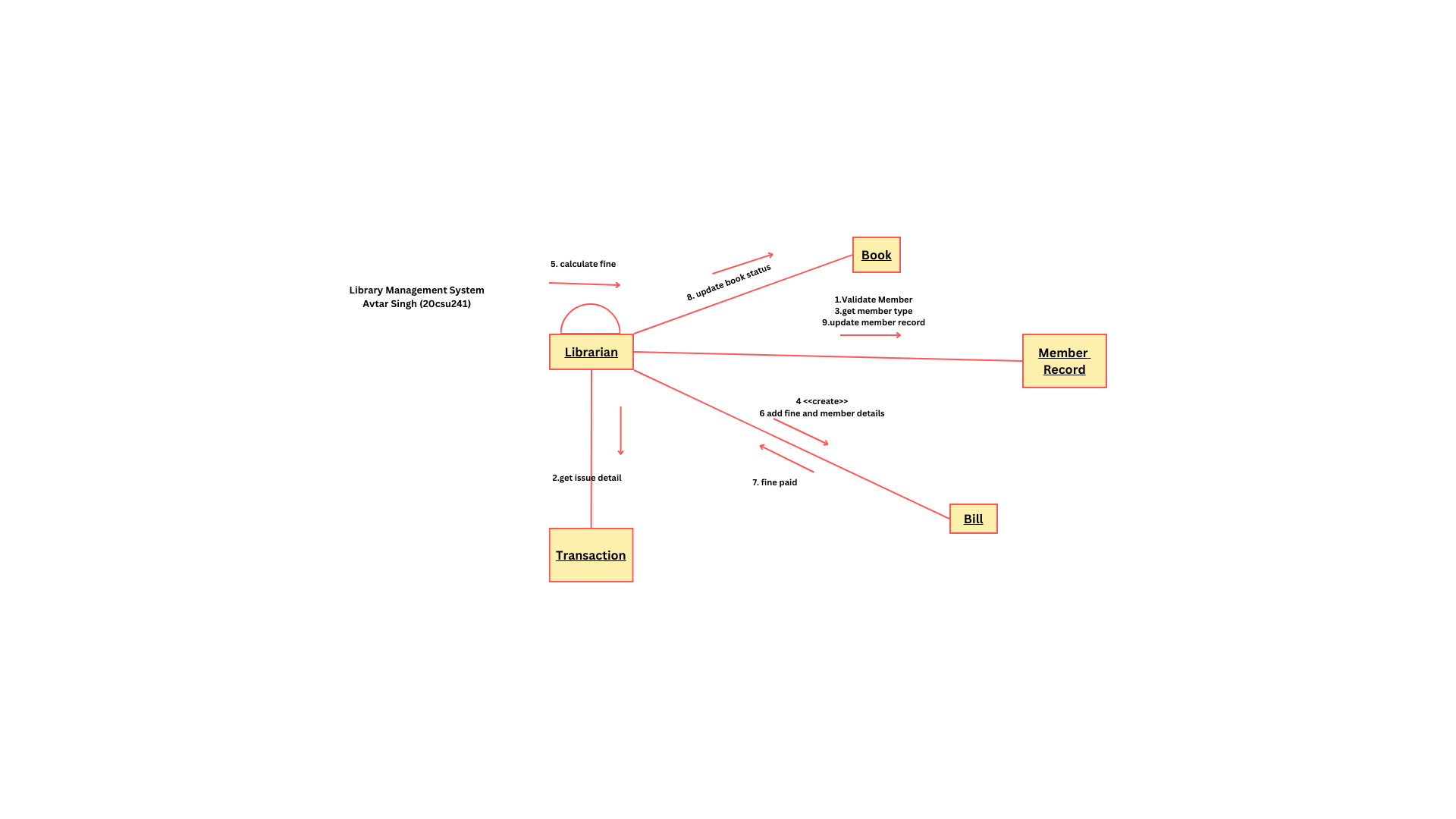


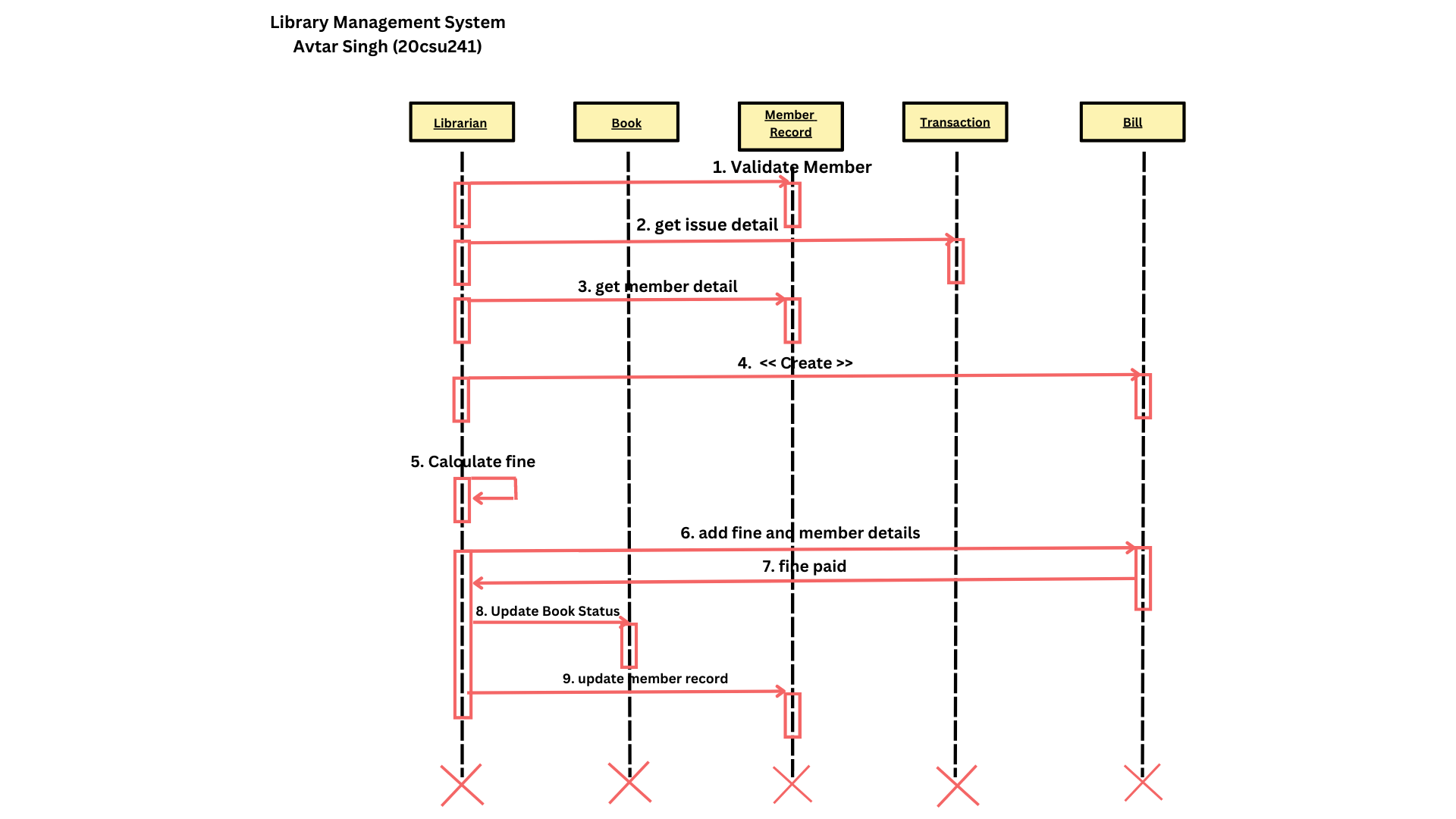
**EXPERIMENT NO. 5**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh - 20csu241** |
| **Semester /Section:VIFs-B** |
| **Link to Code:** |
| **Date: 16.03.23** |
| **Faculty Signature:** |
| **Marks:** |
| **Objective:**  Learn the object-oriented analysis phase by understanding the methods of class elicitation and finding the classes in an object-oriented system. |
| **Outcome:**  Students will learn to discover the classes and mapping from use case to class diagram. |
| **Problem Statement:**  Design the Interaction diagrams, sequence and collaboration diagrams with the help of software engineering tool. |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

****

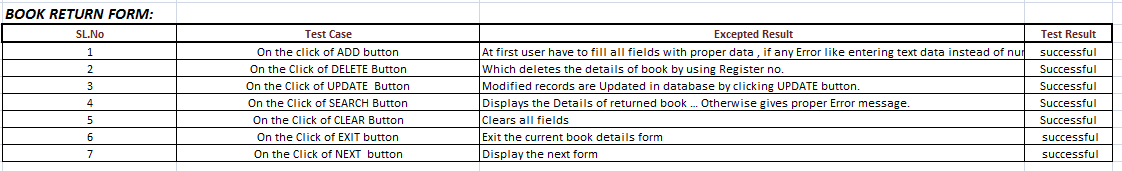
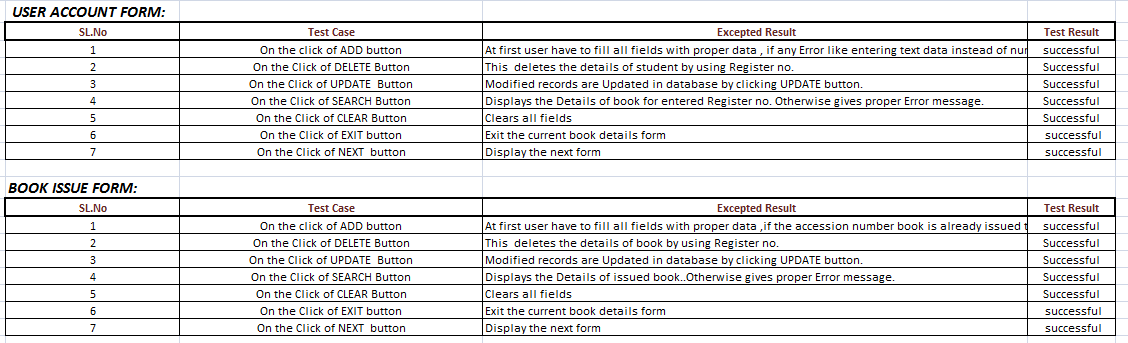
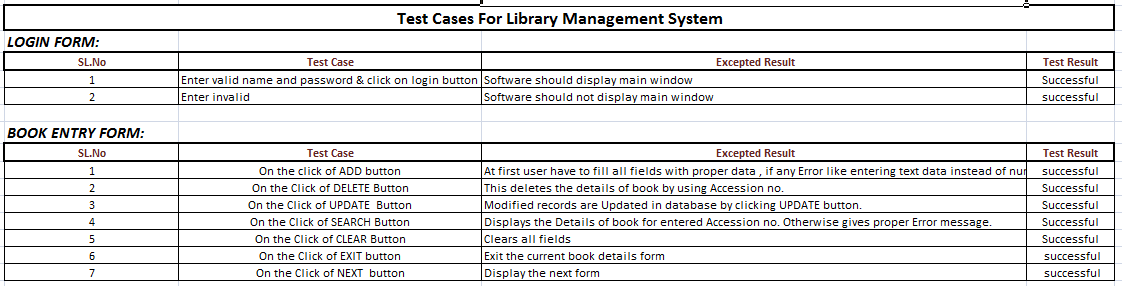
****

**EXPERIMENT NO. 6**

|  |
| --- |
| **Student Name and Roll Number:Avtar Singh - 20csu241** |
| **Semester /Section:VI FS-B** |
| **Link to Code:** |
| **Date:06.04.23** |
| **Faculty Signature:** |
| **Marks:** |
| **Objective:**  ·Deeper understanding of UML state transition diagrams (STD).  ·Practicing using CASE tool. |
| **Outcome:**  Students will learn to design the state and activity diagrams. |
| **Problem Statement:**  Design Test Cases and Test Scenario using Excel. |
| **Background Study:**  While coding, it is necessary to understand the details of the modes an Object of a Class can go through and its transitions at time intervals with the occurrence of any event or action.  State diagrams (also called State Chart diagrams) are used to help the developer better understand any complex/unusual functionalities or business flows of specialized areas of the system. In short, State diagrams depict the dynamic behavior of the entire system, or a sub-system, or even a single object in a system. This is done with the help of *Behavioral elements*.  It is important to note that having a State diagram for your system is not a compulsion, but must be defined only on a need basis  State diagrams have very few elements. State diagrams have very few elements. The basic elements are rounded boxes representing the state of the object and arrows indicting the transition to the next state. The activity section of the state symbol depicts what activities the object will be doing while it is in that state.  All state diagrams being with an initial state of the object. This is the state of the object when it is created. After the initial state the object begins changing states. Conditions based on the activities can determine what the next state the object transitions to.  Below is an example of a state diagram might look like for an Order object. When the object enters the Checking state it performs the activity "check items." After the activity is completed the object transitions to the next state based on the conditions [all items available] or [an item is not available]. If an item is not available the order is canceled. If all items are available then the order is dispatched. When the object transitions to the Dispatching state the activity "initiate delivery" is performed. After this activity is complete the object transitions again to the delivered state.  Figure- State diagram for an order object  State diagrams can also show a super-state for the object. A super-state is used when many transitions lead to the a certain state. Instead of showing all of the transitions from each state to the redundant state a super-state can be used to show that all of the states inside of the super-state can transition to the redundant state. This helps make the state diagram easier to read.  The diagram below shows a super-state. Both the Checking and Dispatching states can transition into the Canceled state, so a transition is shown from a super-state named Active to the state Cancel. By contrast, the state Dispatching can only transition to the Delivered state, so we show an arrow only from the Dispatching state to the Delivered state.  Figure: state diagram showing super state Activity Diagram: Activity diagram are fancy flowcharts. Activity diagrams represent the business and operational workflows of a system. An Activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the particular state. Creating an Activity Diagram Let us consider the example of attending a course lecture, at 8 am.  Figure —an example Activity diagram  As you can see in Figure 6 the first activity is to get dressed to leave for the lecture. A decision then has to be made, depending on the time available for the lecture to start, and the timings of the public trains (metra). If there is sufficient time to catch the train, then take the train; else, flag down a cab to the University. The final activity is to attend the lecture, after which the Activity diagram terminates.  Swim Lanes:  Activity diagrams provide another ability, to clarify which actor performs which activity. If you wish to distinguish in an Activity diagram the activities carried out by individual actors, vertical columns are first made, separated by thick vertical black lines, termed "swim lanes," and name each of these columns with the name of the actor involved. You place each of the activities below the actor performing these activities and then show how these activities are connected.  *A state chart diagram shows the possible states of the objects and the transition that cause a change in the state.*  *An activity diagram are related to state chart diagram except that state chart diagram focuses attention on an object undergoing a process(or on a process as an object), an activity diagram focuses on the flow of activities involved in a single process. The activity diagram shows the how those activities depend on one another.* |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

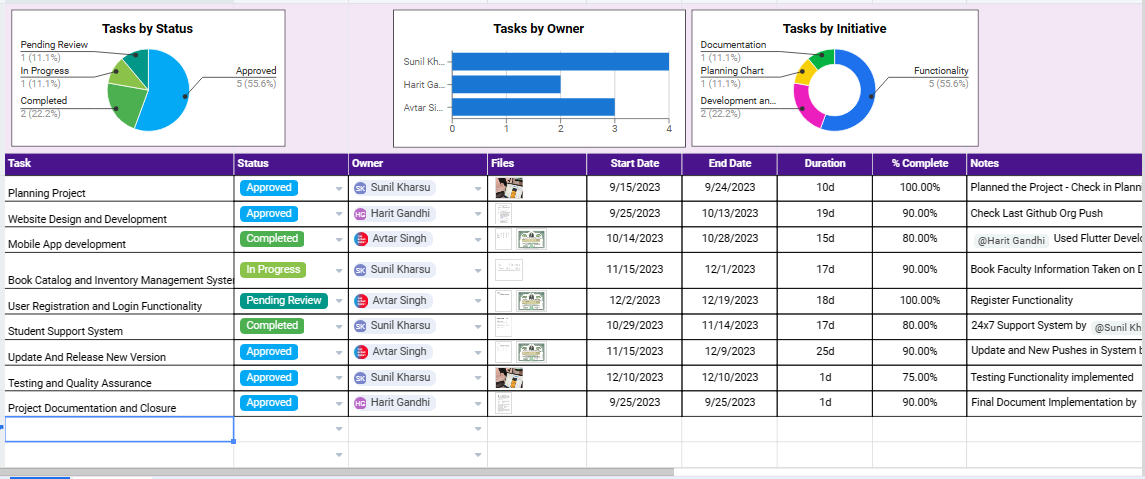
****

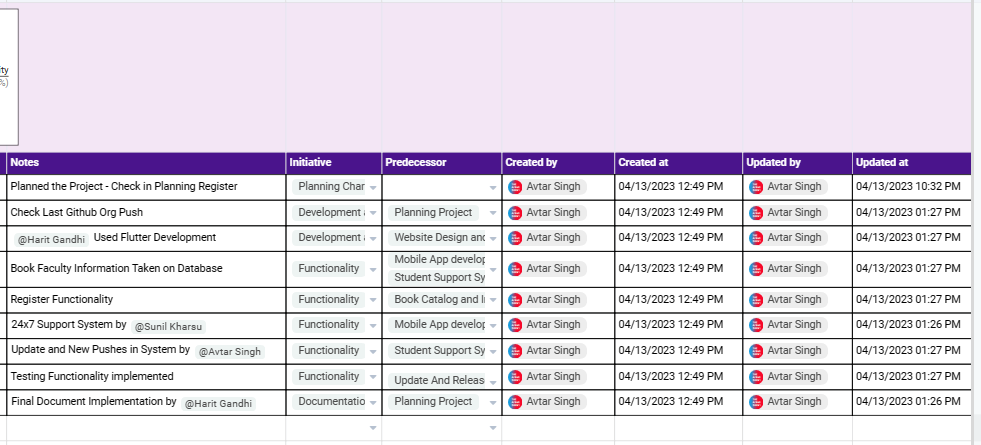
**EXPERIMENT NO. 7**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh - 20csu241** |
| **Semester /Section:VI FS-B** |
| **Link to Code:** |
| **Date:13.04.23** |
| **Faculty Signature:** |
| **Marks:** |
| **Objective:**  Better understanding of the interaction diagrams.  Get familiar with sequence & collaboration diagrams.  Practice drawing the interaction diagrams using CASE tool. |
| **Outcome:**  Students will learn to design sequence and collaboration diagrams. |
| **Problem Statement:**  To create project plan document for the assigned project |
| **Background Study:**  *A use case diagram presents an* ***outside*** *view of the system.*  *Then, how about the* ***inside*** *view of the system?*  Interaction diagrams describe how use cases are ***realize***d in terms of interacting objects.  Two types of interaction diagrams   1. Sequence diagrams 2. Collaboration *(Communication)* diagrams   Once the use cases are specified, and some of the core objects in the system are prototyped on class diagrams, we can start designing the dynamic behavior of the system.  Let's start with the simple example above: a user logging onto the system. The Logon use case can be specified by the following step:  1.Logon dialog is shown  2.User enters user name and password  3.User clicks on OK or presses the enter key  4.The user name and password are checked and approved  5.The user is allowed into the system  Alternative: Logon Failed - if at step 4 the user name and password are not approved, allow the user to try again  **Figure- An example sequence diagram for our logon collaboration** **Collaboration Diagrams** Collaborations are more complex to follow than sequence diagrams, but they do provide the added benefit of more flexibility in terms of spatial layout.  A Sequence diagram is dynamic, and, more importantly, is time ordered. A Collaboration diagram is very similar to a Sequence diagram in the purpose it achieves; in other words, it shows the dynamic interaction of the objects in a system. A distinguishing feature of a Collaboration diagram is that it shows the objects and their association with other objects in the system apart from how they interact with each other. The association between objects is not represented in a Sequence diagram.  A Collaboration diagram is easily represented by modeling objects in a system and representing the associations between the objects as links. The interaction between the objects is denoted by arrows. To identify the sequence of invocation of these objects, a number is placed next to each of these arrows.  **Defining a Collaboration diagram**  A sophisticated modeling tool can easily convert a collaboration diagram into a sequence diagram and the vice versa. Hence, the elements of a Collaboration diagram are essentially the same as that of a Sequence diagram.  Let us see in detail what the elements of Collaboration diagram are.  **Figure-:A Collaboration diagram for Logon** i**nteraction .**  Notice that each message is numbered in sequence, because it is not obvious from the diagram, the order of the messages.  *Sequence diagrams emphasize the order in which things happen, while collaboration diagrams give more flexibility in their layout. You can use whichever you prefer when drawing interactions, as both show the same information.* |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**



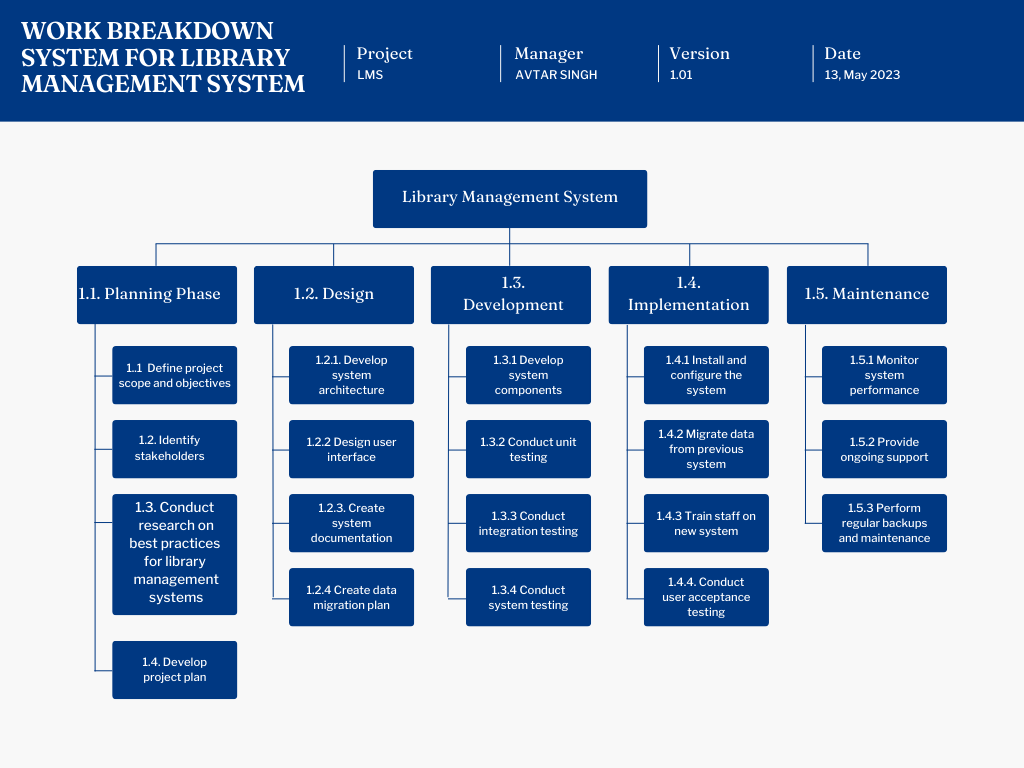


**EXPERIMENT NO. 8**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh - 20csu241** |
| **Semester /Section: VI FS-B** |
| **Link to Code:** |
| **Date:13.04.23** |
| **Faculty Signature:** |
| **Marks:** |
| **Objective:**  ·Gain a deeper understanding of software testing and the software testing document.  ·Become familiar with a Test Plan Document |
| **Outcome:**  Test Plan document using excel. |
| **Problem Statement:**  To design the Work Break Down Structure for assigned Project |
| **Background Study:**  Testing is the process of executing a program with the intent of finding errors. A good test case is one with a high probability of finding an as-yet undiscovered error. A successful test is one that discovers an as-yet-undiscovered error.  The causes of the software defects are: specification may be wrong; specification may be a physical impossibility; faulty program design; or the program may be incorrect.  **Basic Definitions**   * **A failure** is an unacceptable behavior exhibited by a system. * **A defect** is a flaw in any aspect of the system that contributes, or may potentially contribute, to the occurrence of one or more failures. It might take several defects to cause a particular failure. * **An error** is a slip-up or inappropriate decision by a software developer that leads to the introduction of a defect.   **Good Test Attributes**  A good test has a high probability of finding an error, not redundant, and should not be too simple or too complex.  The characteristics of a Banking application are as follows:   * Multi tier functionality to support thousands of concurrent user sessions * Large scale Integration , typically a banking application integrates with numerous other applications such as Bill Pay utility and Trading accounts * Complex Business workflows * Real Time and Batch processing * High rate of Transactions per seconds * Secure Transactions * Robust Reporting section to keep track of day to day transactions * Strong Auditing to troubleshoot customer issues * Massive storage system * Disaster Management.   **Test Cases**  File Open - Test case  Steps to reproduce:  1. Launch Application  2. Select "File" menu  File menu pulls down  3. Choose "Open"  "Open" dialog box appears  4. Select a file to open  5. Click OK  Result: File should open  Test case  Test case ID: B 001  Test Description: verify B - bold formatting to the text  Function to be tested: B - bold formatting to the text  Environment: Win 98  Actual Result: pass |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

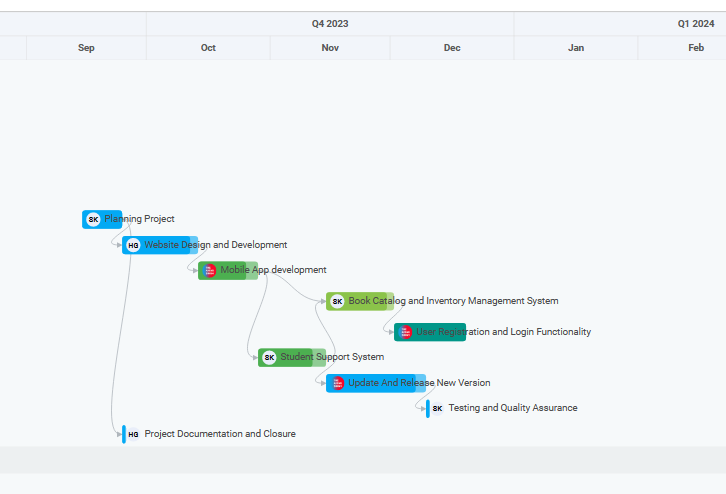
****

**EXPERIMENT NO. 09**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh - 20csu241** |
| **Semester /Section: VI FS-B** |
| **Link to Code:** |
| **Date: 20.04.23** |
| **Faculty Signature:** |
| **Marks:** |
| **Objective:**  To design the Gantt Chart using open source tool.  To find and view the critical path  To calculate the total costs of the project by tasks and resources |
| **Outcome:**  Student will be familiarized with the study of the benefits of Gantt Chart that illustrates a project schedule based on start/finish dates, task durations, and dependencies |
| **Problem Statement:**  Create the Gantt Chart for your assigned project. |
| **Background Study:**  [A gantt chart](https://www.teamgantt.com/what-is-a-gantt-chart) is a horizontal bar chart used in project management to visually represent a project plan over time. Modern gantt charts typically show you the timeline and status—as well as who’s responsible—for each task in the project.  In project management, gantt charts are used for planning and scheduling projects. A gantt chart is incredibly useful because it allows you to simplify complex projects into an easy-to-follow plan and track the status of tasks as work progresses.  Gantt charts also help you keep track of project deadlines, milestones, and hours worked so you can spot and address delays or overages before they wreak havoc on your project. |
| **Question Bank:**  ·Write the basic elements of gantt chart  ·Illustrate all the steps to create gantt chart of your own project. |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

****

**EXPERIMENT NO. 10**

|  |
| --- |
| **Student Name and Roll Number: Avtar Singh - 20csu241** |
| **Semester /Section: VI FS-B** |
| **Link to Code:** |
| **Date: 20.04.23** |
| **Faculty Signature:** |
| **Marks:** |
| **Objective:**  To design the User Story and Epic for Library Management System |
| **Outcome:**  Student will be familiarized with the study of the benefits of Gantt Chart that illustrates a project schedule based on start/finish dates, task durations, and dependencies |
| **Problem Statement:**  To design the User Story and Epic for Library Management System |
| **Background Study:**  [A gantt chart](https://www.teamgantt.com/what-is-a-gantt-chart) is a horizontal bar chart used in project management to visually represent a project plan over time. Modern gantt charts typically show you the timeline and status—as well as who’s responsible—for each task in the project.  In project management, gantt charts are used for planning and scheduling projects. A gantt chart is incredibly useful because it allows you to simplify complex projects into an easy-to-follow plan and track the status of tasks as work progresses.  Gantt charts also help you keep track of project deadlines, milestones, and hours worked so you can spot and address delays or overages before they wreak havoc on your project. |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

***Epic*: Implement a library management system.**

***User Stories:***

*As a librarian, I want to be able to add new books to the library's collection so that Readers can check them out.*

*As a librarian, I want to be able to update the status of a book (checked in or checked out) so that I can keep track of what books are available for borrowing.*

*As a Reader, I want to be able to search for books by title, author, or subject so that I can find books that interest me.*

*As a Reader, I want to be able to see the availability of a book so that I know whether it's currently checked out or not.*

*As a librarian, I want to be able to generate reports on library usage, such as which books are being checked out the most, so that I can make informed decisions about purchasing new books.*

*As a Reader, I want to be able to renew a book online so that I don't have to come into the library to do so.*

*As a librarian, I want to be able to see a Reader's borrowing history so that I can keep track of their reading habits and make recommendations for future reading.*

*As a Reader, I want to be able to rate and review books so that I can share my opinions with others and discover new books to read.*