



**GR22 Regulations**

**II B.Tech II Semester**

**Software Engineering Lab**

**(GR22A2103)**

**B. Tech- Computer Science and Business System**

**GOKARAJU RANGARAJU**  
**INSTITUTE OF ENGINEERING AND TECHNOLOGY**  
(Autonomous)

# **GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY**

## **SOFTWARE ENGINEERING LAB**

**Course Code: GR22A2103**

**L/T/P/C: 0/0/2/1**

**II Year II Semester**

### **Course Objectives**

1. To impart state-of-the-art knowledge on Software Engineering and UML.
2. Practice software engineering principles for real time problems.
3. Design various types of diagrams for real time problems.
4. Learn test case generation.
5. Demonstrate software engineering methodologies for various real time problems.

### **Course Outcomes**

1. Analyze and identify requirements for real time problems.
2. Design and implement various software design models.
3. Usage of modern engineering tools for specification, design and implementation.
4. Provide appropriate solutions for the real time problems using software engineering methodology.
5. Design test cases for various real time problems.

**Software's Used: StarUML /Umbrello & JUNIT**

**Develop the following applications using software engineering methodologies.**

1. Unified Library System
2. Online Railway Reservation System

### **TASK1**

Prepare the problem statement for above applications.

### **TASK2**

Develop Software Requirement Specification (SRS) for above applications.

### **TASK3**

Design the data flow diagram for the above applications.

### **TASK4**

Design the class diagrams for above applications.

### **TASK 5**

Design the Use-case diagrams for the above applications.

### **TASK 6**

Design the interaction diagrams for the above applications.

### **TASK 7**

Perform forward engineering for the above application and generate a report of the same.

### **TASK 8**

Perform reverse engineering for the above application and generate a report of the same.

### **TASK 9**

Write a C++ program to demonstrate the working of the following constructs:

- i) while
- ii) if ...else
- iii) Switch
- iv) for Loops in C++ language

## **TASK 10**

Create a test plan document for any application (e.g. Unified Library System)

## **TASK 11**

Implement a Junit Test program and design test cases to find the maximum of an array of numbers.

## **TASK 12**

Implement a Junit Test program and design test cases to count the number of elements in array of numbers.

### **TEXT BOOKS:**

1. The Unified Modeling Language User Guide, Grady Booch, James Rumbaugh, Ivar Jacobson, Pearson Education.
2. UML 2 Toolkit, Hans-Erik Eriksson, Magnus Penker, Brian Lyons, David Fado, WILEYDreamtech India Pvt. Ltd.
3. Software Engineering, Ian Sommerville

### **REFERENCE BOOKS:**

1. Fundamentals of Software Engineering, Carlo Ghezzi, Jazayeri Mehdi, Mandrioli Dino
2. Software Requirements and Specification: A Lexicon of Practice, Principles and Prejudices, Michael Jackson
3. The Unified Development Process, Ivar Jacobson, Grady Booch, James Rumbaugh
4. Design Patterns: Elements of Object-Oriented Reusable Software, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
5. Software Metrics: A Rigorous and Practical Approach, Norman E Fenton, Shari Lawrence Pfleeger
6. Software Engineering: Theory and Practice, Shari Lawrence Pfleeger and Joanne M. Atlee
7. Object-Oriented Software Construction, Bertrand Meyer
8. Object Oriented Software Engineering: A Use Case Driven Approach --Ivar Jacobson
9. Touch of Class: Learning to Program Well with Objects and Contracts --Bertrand Meyer
10. UML Distilled: A Brief Guide to the Standard Object Modeling Language --Martin Fowler

## INDEX

| <b>S.No</b> | <b>Tasks</b>   | <b>Page No.</b> |
|-------------|--|-----------------|
| 1           | Prepare the problem statement for below applications<br>a. Unified Library System<br>b. Online Railway Reservation System                                    | 1-2             |
| 2           | Develop Software Requirement Specification (SRS) for below applications.<br>a. Unified Library System<br>b. Online Railway Reservation System                | 3-10            |
| 3           | Design the data flow diagram for the below applications.<br>a. Unified Library System<br>b. Online Railway Reservation System                                | 11-13           |
| 4           | Design the class diagrams for the below applications<br>a. Unified Library System<br>b. Online Railway Reservation System                                    | 14-15           |
| 5           | Design the Use-case diagrams for the below applications<br>a. Unified Library System<br>b. Online Railway Reservation System                                 | 16-18           |
| 6           | Design the interaction diagrams for the below applications.<br>a. Unified Library System<br><b>b. Online Railway Reservation System</b>                      | 19-21           |
| 7           | Perform Forward Engineering, and generate documentation of the same.   | 22-25           |
| 8           | Perform Reverse Engineering, and generate documentation of the same.   | 26-27           |
| 9           | Write a C++ program to demonstrate the working of the following constructs:<br>i) while    ii) if ...else    iii) Switch    iv) for Loops<br>in C++ language | 28-37           |

|    |   |       |
|----|---|-------|
| 10 | Create a test plan document for any application (e.g. Unified Library System)                             | 38-41 |
| 11 | Implement a Junit Test program and design test cases to find the maximum of an array of numbers.          | 42    |
| 12 | Implement a Junit Test program and design test cases to count the number of elements in array of numbers. | 43    |

# TASK 1

**Task 1:** Prepare the problem statement for below applications

- a. Unified Library System
- b. Online Railway Reservation System

**Aim:** To prepare problem statement for any application.

**Definition:** The problem statement is the initial starting point for a project. It is basically a one to three page statement that everyone on the project agrees with that describes what will be done at a high level. The problem statement is intended for a broad audience and should be written in nontechnical terms. It helps the non-technical and technical personnel communicate by providing a description of a problem. It doesn't describe the solution to the problem. The input to requirement engineering is the problem statement prepared by customer. It may give an overview of the existing system along with broad expectations from the new system. The first phase of requirements engineering begins with requirements elicitation i.e. gathering of information about requirements. Here, requirements are identified with the help of customer and existing system processes. So from here begins the preparation of problem statement. So, basically a problem statement describes what needs to be done without describing how.

**Unified Library System:** is a software system that issues books and magazines to registered students only. The student has to login after getting registered to the system. The borrower of the book can perform various functions such as searching for desired book, get the issued book and return the book. Libraries are used to store books, but require a system to navigate to a specific book or specific content within a book. A library database system is an infrastructure that allows users to search books and book content, add/remove, and download selected books. The problem faced is that library users require an efficient method to find a specific book or keyword(s) within a book given a continuously expanding library. Efficiency requires that the processing time should stay relatively the same even as the library contents increases. The objective of this project was to develop a library database system in which a user can supply books to the server, organize books by location, retrieve books from the server and query the server for books which contain specific keywords.

**Online Railway Reservation System:** Software has to be developed for automating the manual reservation system of railway. The system should be standalone in nature. It should be designed to provide functionalities like booking of tickets in which a user should be able to apply for tickets of any train and of any class. A limitation is imposed when the numbers of tickets for which user apply is greater than available seats or no seats are available. If seats are not available then put user transaction in the waiting queue. If the tickets are available then it is issued to the user and it must be updated in the database concurrently. The system generates the receipt for the same. The software takes the current system date and time as the date of issue and calculates the amount to be paid by the user. It also provide the functionality of cancellation of tickets .If the user wants to cancel the tickets, he/she must enter the details. The system checks the records from the database if it is

matched with the user entered details, and then it cancels the tickets. The system also calculates the amount to be return to the user after deductions. The system must update the database for the same. After that system must check for waiting passenger for that train, if any then these tickets are issued to waiting passenger and update the database. The system displays the details of train of which user enter the name. The information is saved and the corresponding updating take place in the database. In the enquiry, the system should be able to provide information like the availability of tickets of particular train, train schedule. The system should be able to reserve a ticket for a particular user if the tickets are not currently available. The corresponding print outs for each entry (issue/cancel) in the system should be generated.

There should be proper information if the waiting list ticket is confirmed, through mail or via sms. It should tell us as to which all stations it hauls and current status of the train should be informed. Security provisions like the login authenticity should be provided. Each user should have a user id and a password. Record of the users of the system should be kept in the log file. Provision should be made for full backup of the system.

## TASK 2

**Task 2:** Develop Software Requirement Specification (SRS) for below applications.

- a. Unified Library System
- b. Online Railway Reservation System

**Aim:** To develop Software Requirement Specification (SRS) for above applications.

### **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.

## **Non functional 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.

## **SRS Report for Library Management System**

### **Introduction:**

With the increase in the number of readers, better management of libraries system is required. The Library management system focuses on improving the management of libraries in a city or town. “What If you can check whether a book is available in the library through your phone?” or “what if instead of having different library cards for different libraries you can just have one ?” or “you can reserve a book or issue a book from your phone sitting at your home!”. The Integrated Library Management system provides you the ease of issuing, renewing, or reserving a book from an library within your town through your phone. The Integrated Library Management system is developed on the android platform which basically focuses on issuing, renewing and reserving a book.

### **Purpose:**

The purpose of the project is to maintain the details of books and library members of different libraries. The main purpose of this project is to maintain a easy circulation system between clients and the libraries, to issue books using single library card, also to search and reserve any book from different available libraries and to maintain details about the user (fine, address, phone number).Moreover, the user can check all these features from their home.

### **Scope:**

- Manually updating the library system into an android based application so that the user can know the details of the books available and maximum limit on borrowing from their computer and also through their phones.
- The LM System provides information's like details of the books, insertion of new books, deletion of lost books, limitation on issuing books, fine on keeping a book more than one month from the issued date.
- Also user can provide feedback for adding some new books to the library.

### **Definition, Acronyms, Abbreviation:**

- JAVA -> platform independence
- DFD -> Data Flow Diagram
- ER -> Entity Relationship
- IDE -> Integrated Development Environment
- SRS -> Software Requirement Specification

## Overall Description

### Product Perspective:

The proposed Library Management System will take care of the current book detail at any point of time. The book issue, book return will update the current book details automatically so that user will get the update current book details.

### Functional Requirement:

#### R.1: Register

Description : First the user will have to register/sign up. There are two different type of users.

The library manager/head : The manager have to provide details about the name of library ,address, phone number, email id.

Regular person/student : The user have to provide details about his/her name of address, phone number, email id.

##### R.1.1: Sign up

Input: Detail about the user as mentioned in the description.

Output: Confirmation of registration status and a membership number and password will be generated and mailed to the user.

Processing: All details will be checked and if any error are found then an error message is displayed else a membership number and password will be generated.

##### R.1.2 : Login

Input: Enter the membership number and password provided.

Output : User will be able to use the features of software.

#### R.2: Manage books by user.

##### R.2.1: Books issued.

Description : List of books will be displayed along with data of return.

##### R.2.2: Search

Input: Enter the name of author's name of the books to be issued.

Output: List of books related to the keyword.

##### R.2.3 : Issues book

State : Searched the book user wants to issues.

Input : click the book user wants.

Output : conformation for book issue and apology for failure in issue.

Processing : if selected book is available then book will be issued else error will be displayed.

##### R.2.4 : Renew book

State : Book is issued and is about to reach the date of return.

Input : Select the book to be renewed.

Output : conformation message.

Processing : If the issued book is already reserved by another user then error message will be send and if not then conformation message will be displayed.

##### R.2.5 : Return

Input ; Return the book to the library.

Output : The issued list will be updated and the returned book will be listed out.

#### R.2.6 ; Reserve book

Input ; Enter the details of the book.

Output : Book successfully reserved.

Description : If a book is issued by someone then the user can reserve it ,so that later the user can issue it.

#### R.2.6 Fine

Input : check for the fines.

Output : Details about fines on different books issued by the user.

Processing : The fine will be calculated, if it crossed the date of return and the user did not renewed if then fine will be applied by Rs 10 per day.

### R.3 Manage book by librarian

#### R.3.1 Update details of books

##### R.3.1.1 Add books

Input : Enter the details of the books such as names ,author ,edition, quantity.

Output : confirmation of addition.

##### R.3.1.2 Remove books

Input : Enter the name of the book and quantity of books.

Output : Update the list of the books available.

## **Non Functional Requirements:**

### Availability Requirement

The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.

### Efficiency Requirement

Mean Time to Repair (MTTR) - Even if the system fails, the system will be recovered back up within an hour or less.

### Accuracy

The system should accurately provide real time information taking into consideration various concurrency issues. The system shall provide 100% access reliability.

### Performance Requirement

The information is refreshed depending upon whether some updates have occurred or not in the application. The system shall respond to the member in not less than two seconds from the time of the request submittal. The system shall be allowed to take more time when doing large processing jobs. Responses to view information shall take no longer than 5 seconds to appear on the screen.

### Reliability Requirement

The system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect or incomplete data. The system will run 7 days a week, 24 hours a day.

## **User Characteristics:**

We have 3 levels of users :

User module: In the user module, user will check the availability of the books.

Issue book

Reserve book

Return book

Fine details

Library module:

Add new book

Remove books

Update details of book

Administration module:

The following are the sub module in the administration module:

Register user

Entry book details

Book issue

## **Constraints**

Any update regarding the book from the library is to be recorded to have update & correct values, and any fine on a member should be notified as soon as possible and should be correctly calculated.

## TASK 3

**Task 3:** Design the data flow diagram for the below applications.

- Unified Library System
- Online Railway Reservation System

**Aim:** To design data flow diagram for Unified Library System, Online Railway Reservation system

**Data Flow Diagram (DFD) :** DFD show the flow of data through the system.

- All names should be unique
- It is not a flow chart
- Suppress logical decisions
- Defer error conditions & handling until the end of the analysis
- DFD represent a system or software at any level of abstraction.
- A level 0 DFD is called fundamental system model or context model represents entire software element as a single bubble with input and output data indicating by incoming & outgoing arrows.

### Unified Library System:

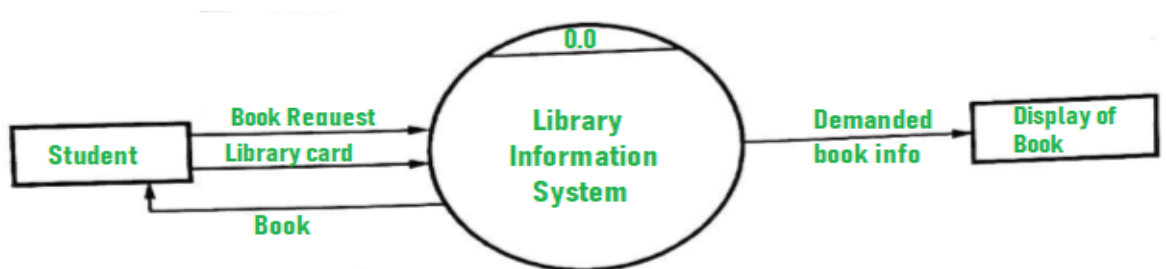
It depicts the flow of information and the transformation applied when a data moves in and out from a system. The overall system is represented and described using input, processing and output in the DFD. The inputs can be:

- **Book request** when a student requests for a book
- **Library card** when the student has to show or submit his/her identity as a proof.

The overall processing unit will contain the following output that a system will produce or generate:

- Book will be the output as the book demanded by the student will be given to them.
- Information of demanded book should be displayed by the library information system that can be used by the student while selecting the book which makes it easier for the student.

### Level 0 DFD :





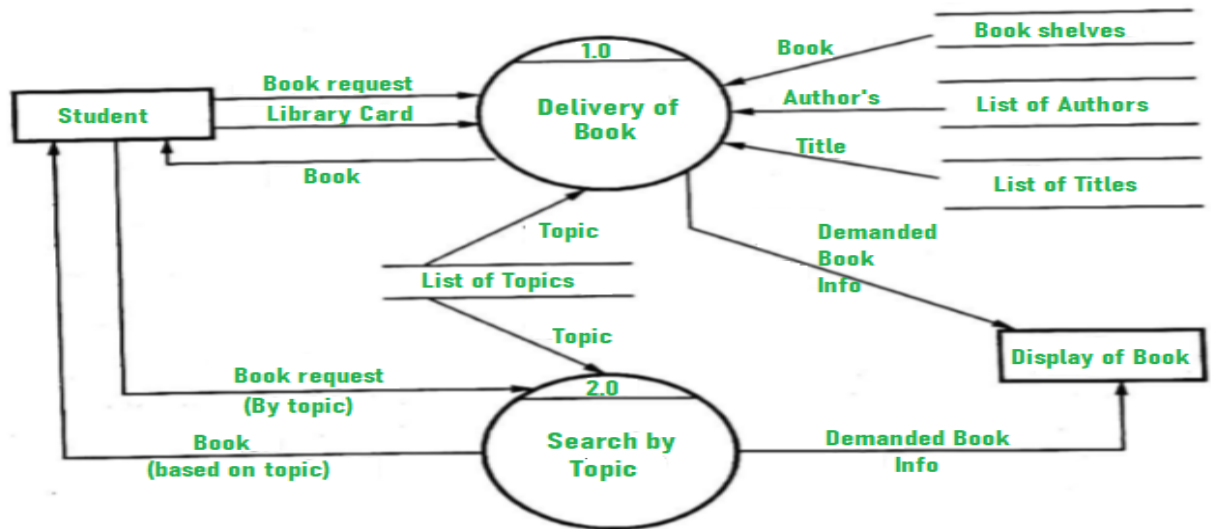
### Level 1 DFD :

At this level, the system has to show or exposed with more details of processing.

The processes that are important to be carried out are:

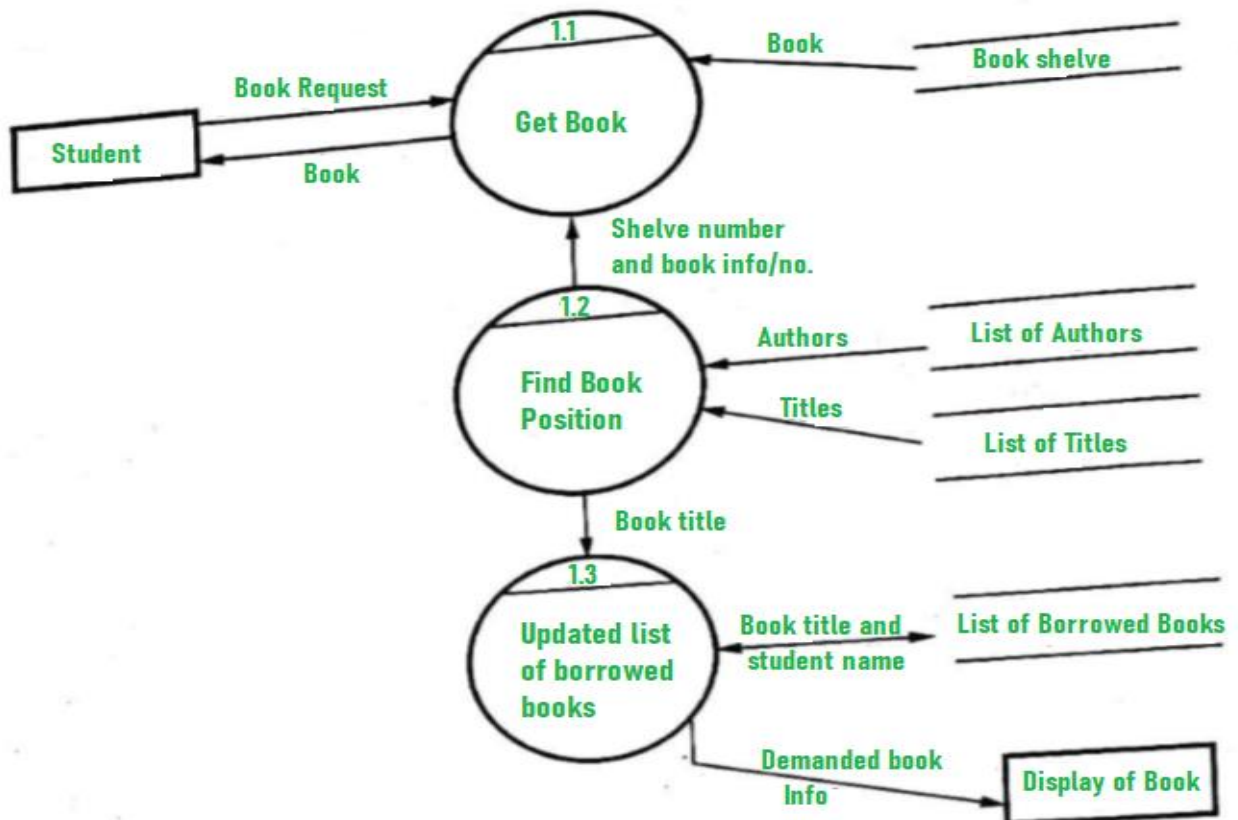
- Book delivery
- Search by topic

List of authors, List of Titles, List of Topics, the bookshelves from which books can be located are some information that is required for these processes. Data store is used to represent this type of information.



**Level 1 DFD**

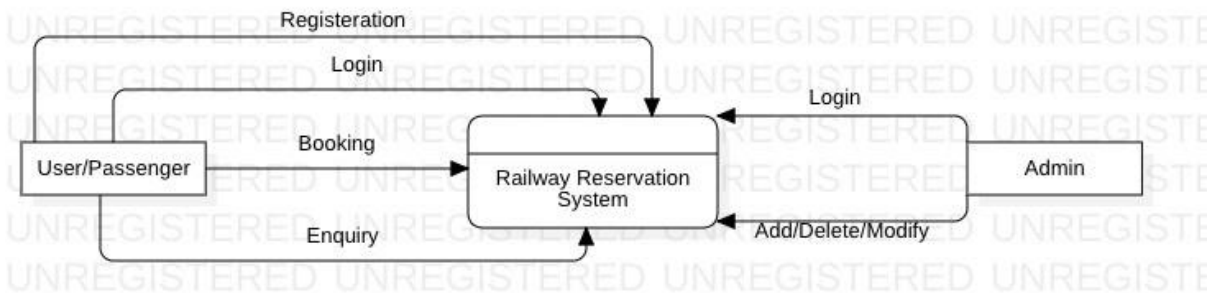
### Level 2 DFD:



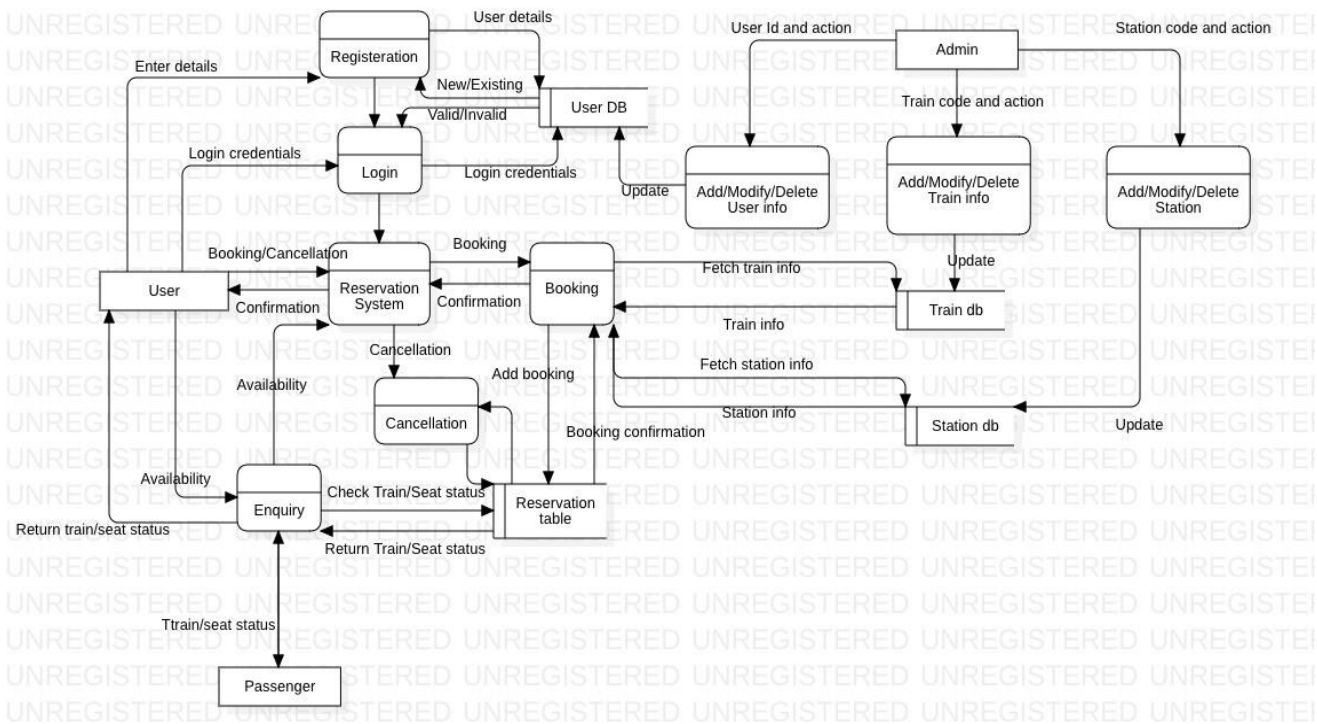
**Level 2 DFD**

# Railway Reservation System:

## Level 0 DFD :



## Level 1 DFD :



## TASK 4

**Task 4:** Design Class diagrams for below applications.

- a. Unified Library System
- b. Online Railway Reservation System

**Aim:** To design Class diagrams for Unified Library System, Online Railway Reservation system

### Overview:

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application. The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram.

### Purpose:

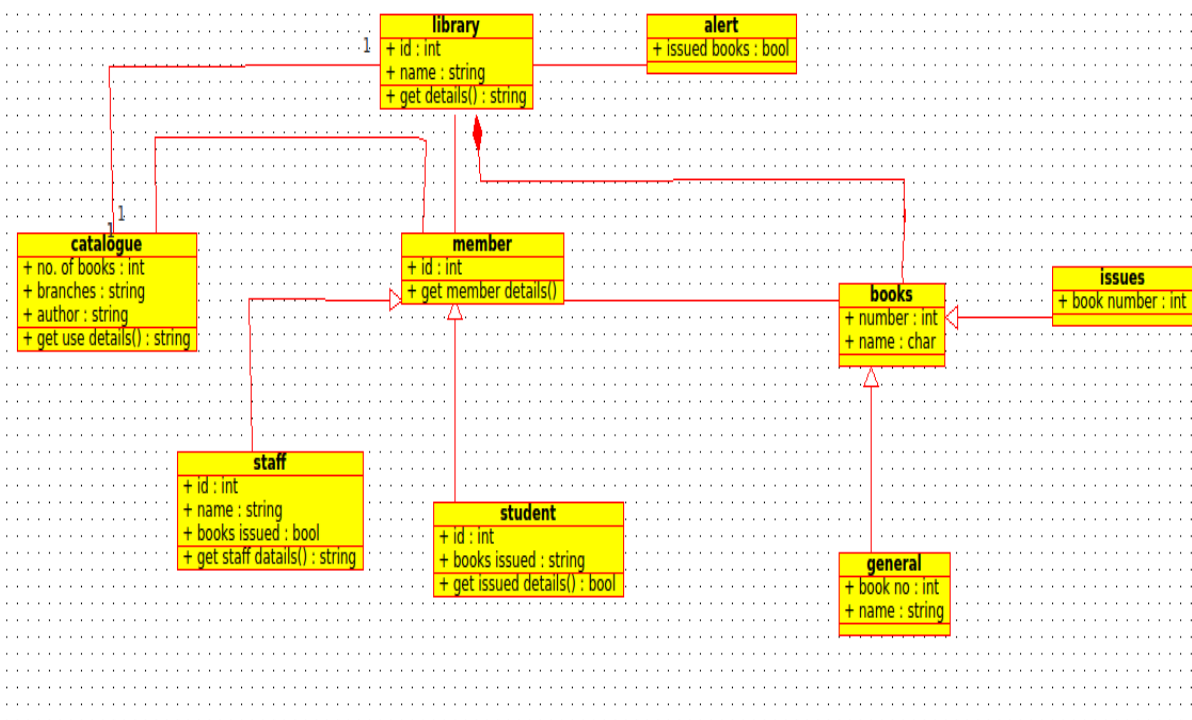
The purpose of the class diagram is to model the static view of an application. The class diagrams are the only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction. The UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application but class diagram is a bit different. So it is the most popular UML diagram in the coder community. So the purpose of the class diagram can be summarized as:

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

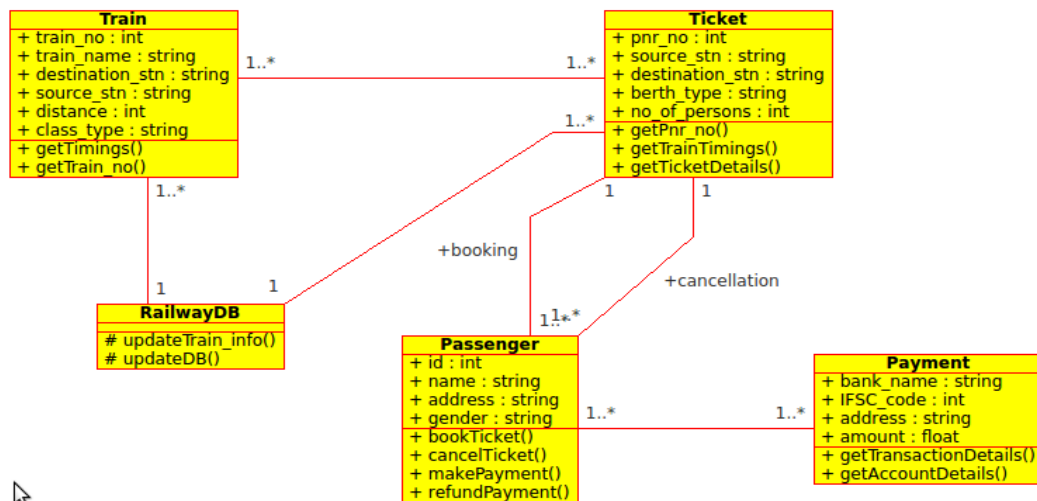
### Contents:

Class diagrams commonly contain the following things

- Classes
- Interfaces
- Collaborations
- Dependency, generalization and association relationships



**Fig:1:** Class diagram for Library Management System



**Fig:2:** Class diagram for Railway Reservation System

## Task 5

**Task 5:** Design Use case diagrams for below applications.

- a. Unified Library System
- b. Online Railway Reservation System

**Aim:** To design Use case diagram for Library, Railway Reservation

### Overview:

To model a system the most important aspect is to capture the dynamic behaviour. Use case diagram is dynamic in nature there should be some internal or external factors for making the interaction. These internal and external agents are known as actors.

Contents: actors, use cases and their relationships.

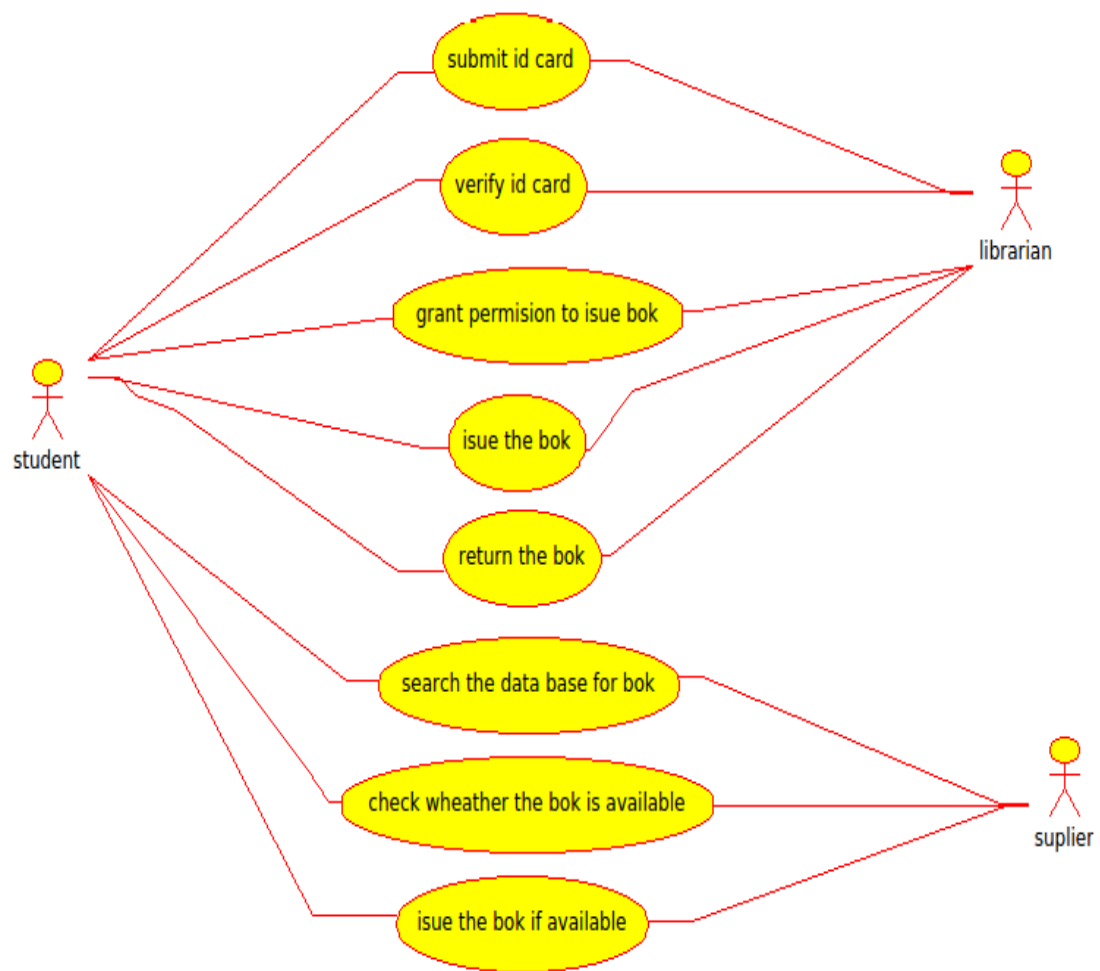
The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

### Purpose:

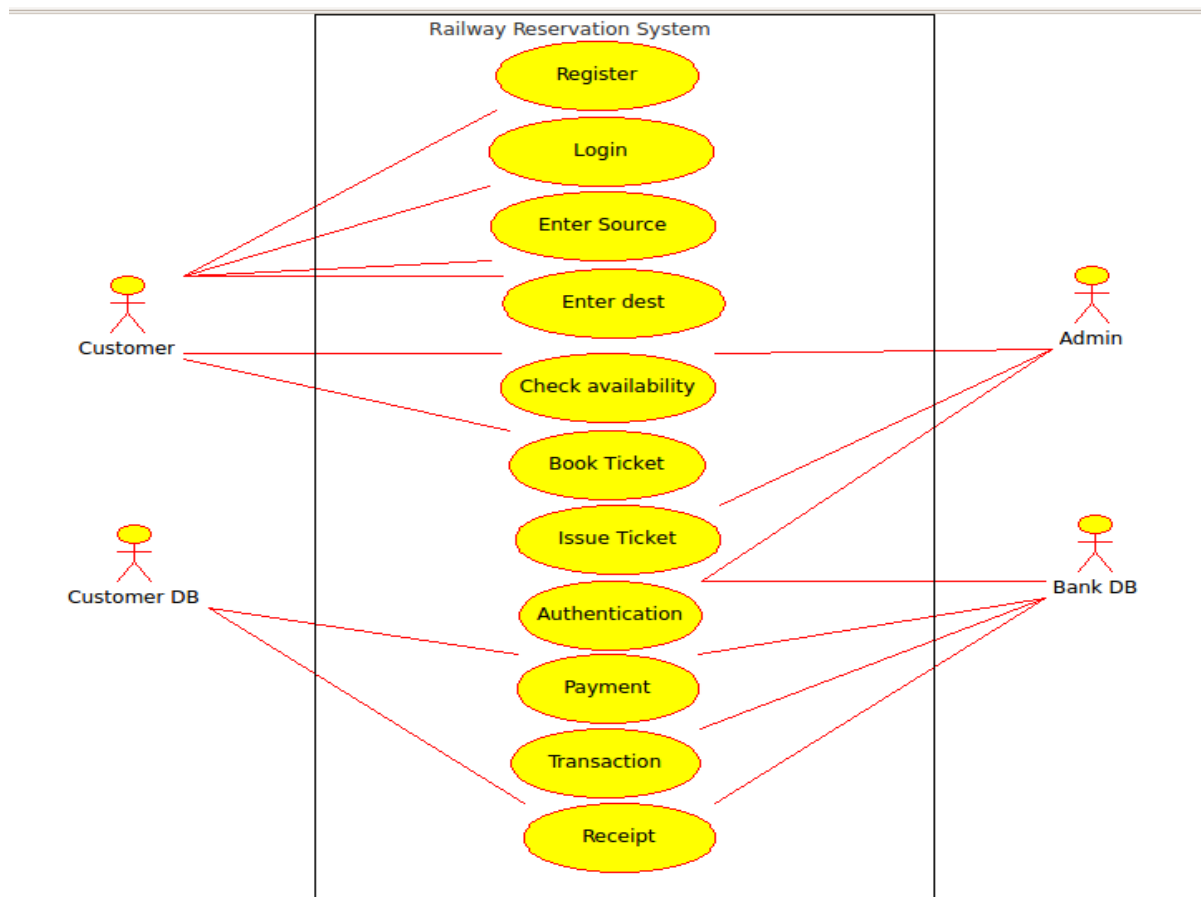
The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose. Because other four diagrams (activity, sequence, collaboration and Statechart) are also having the same purpose. So we will look into some specific purpose which will distinguish it from other four diagrams. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

So in brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.
- Show the interacting among the requirements are actors



**Fig:1:** Usecase diagram for Library Management System



**Fig:2:** Usecase diagram for Railway management system

## TASK 6

**Task 6:** Design Interaction diagrams for below applications.

- a. Unified Library System
- b. Online Railway Reservation System

**Aim:** To design Interaction diagrams for Library, Railway Reservation System.

**Interaction Diagram:** It is clear that the diagram is used to describe some type of interactions among the different elements in the model. So this interaction is a part of dynamic behaviour of the system.

This interactive behaviour is represented in UML by two diagrams known as Sequence diagram and Collaboration diagram. The basic purposes of both the diagrams are similar.

Sequence diagram emphasizes on time sequence of messages and collaboration diagram emphasizes on the structural organization of the objects that send and receive messages.

### **Purpose:**

The purposes of interaction diagrams are to visualize the interactive behaviour of the system. Now visualizing interaction is a difficult task. So the solution is to use different types of models to capture the different aspects of the interaction.

That is why sequence and collaboration diagrams are used to capture dynamic nature but from a different angle.

So the purposes of interaction diagram can be describes as:

- To capture dynamic behaviour of a system.
- To describe the message flow in the system.
- To describe structural organization of the objects.
- To describe interaction among objects.

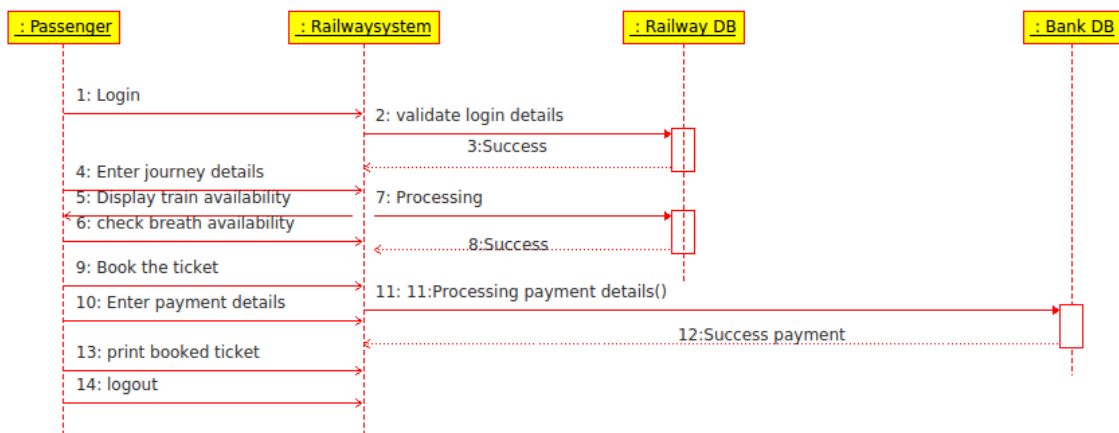
The second interaction diagram is collaboration diagram. It shows the object organization as shown below. Here in collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram.



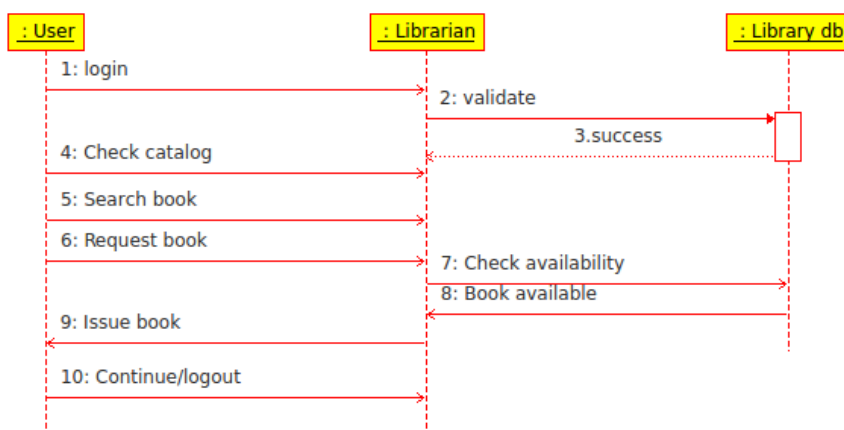
The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization where as the collaboration diagram shows the object organization.

Following are the usages of interaction diagrams:

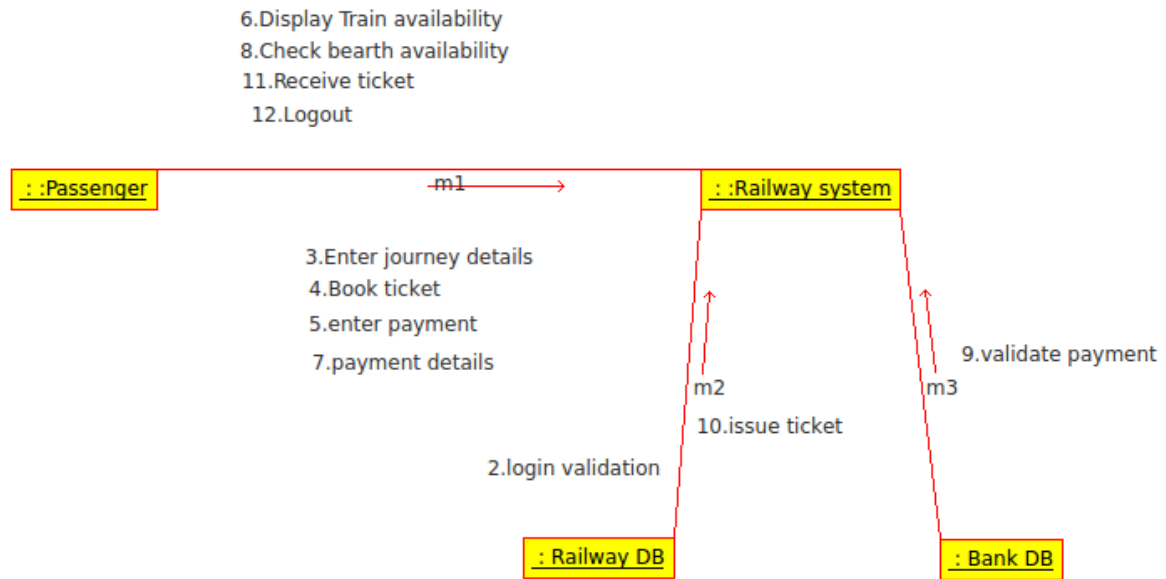
- To model flow of control by time sequence.
- To model flow of control by structural organizations.
- For forward engineering.
- For reverse engineering.



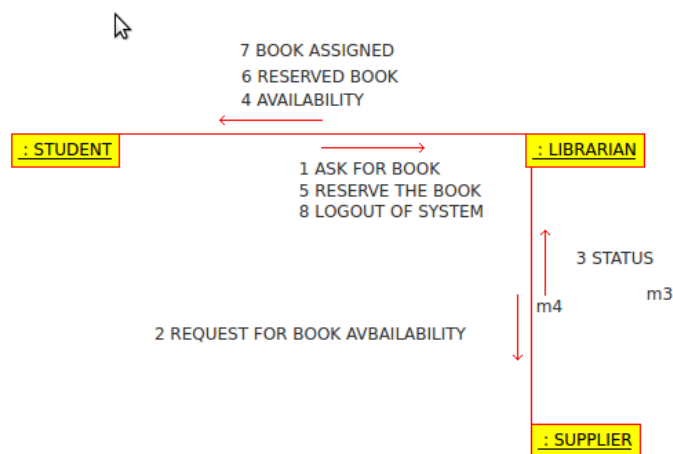
**Fig:1:** Sequence diagram for Railway Reservation System



**Fig:2:** Sequence diagram for Unified Library System



**Fig:3:** Collaboration diagram for Railway Reservation System



**Fig:4:** Collaboration diagram for Unified Library System

## TASK 7

**Task 7:** Perform Forward Engineering, and Generate documentation of the project.

**Aim:** To perform Forward Engineering, and Generate documentation of the project.

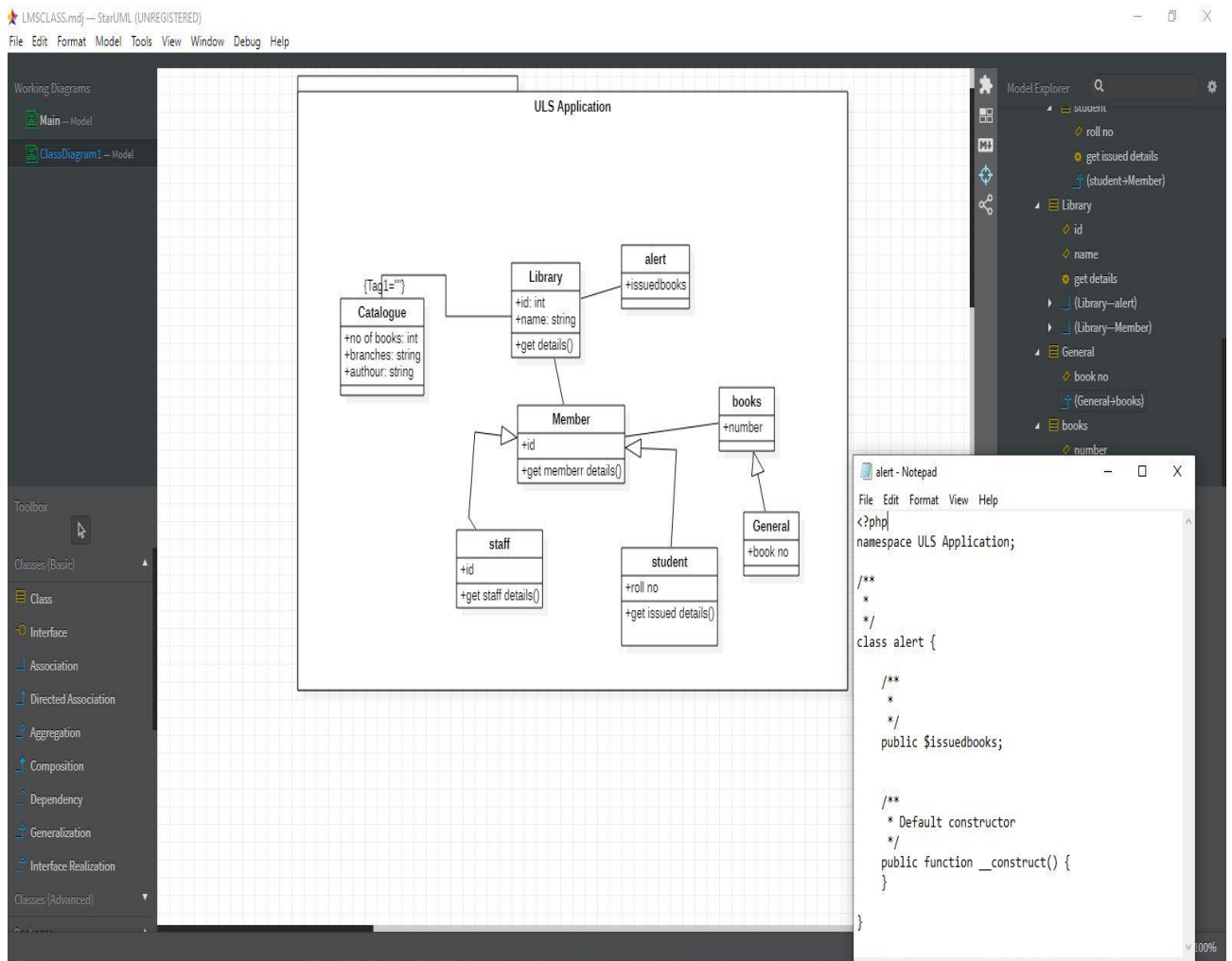
### Forward Engineering:

Forward engineering is the process of transforming a model into code through a mapping to an implementation language. Forward engineering results in a loss of information, because models written in the UML are semantically richer than any current object-oriented programming language. In fact, this is a major reason why you need models in addition to code. Structural features, such as collaborations, and behavioral features, such as interactions, can be visualized clearly in the UML, but not so clearly from raw code.

### To forward engineer a class diagram

- Identify the rules for mapping to your implementation language or languages of choice. This is something you'll want to do for your project or your organization as a whole.
- Depending on the semantics of the languages you choose, you may want to constrain your use of certain UML features. For example, the UML permits you to model multiple inheritance, but Smalltalk permits only single inheritance. You can choose to prohibit developers from modeling with multiple inheritance (which makes your models language-dependent), or you can develop idioms that transform these richer features into the implementation language (which makes the mapping more complex).
- Use tagged values to guide implementation choices in your target language. You can do this at the level of individual classes if you need precise control. You can also do so at a higher level, such as with collaborations or packages.
- Use tools to generate code.

```
public abstract class EventHandler {  
    EventHandler successor;  
    private Integer currentEventID;  
    private String source; EventHandler() {}  
    public void handleRequest() {}  
}
```



**Fig:1:** Forward Engineer a Class Diagram for Unified Library System

Umbrello UML Modeller is a UML modelling tool, and as such its main purpose is to help you in the analysis and design of your systems. However, to make the transition between your design and your implementation, Umbrello UML Modeller allows you to generate source code in different programming languages to get you started. Also, if you want to start using UML in an already started C++ project, Umbrello UML Modeller can help you create a model of your system from the source code by analysing your source code and importing the classes found in it.

## Code Generation

Umbrello UML Modeller can generate source code for various programming languages based on your UML Model to help you get started with the implementation of your project. The

code generated consists of the class declarations, with their methods and attributes so you can “fill in the blanks” by providing the functionality of your classes' operations.

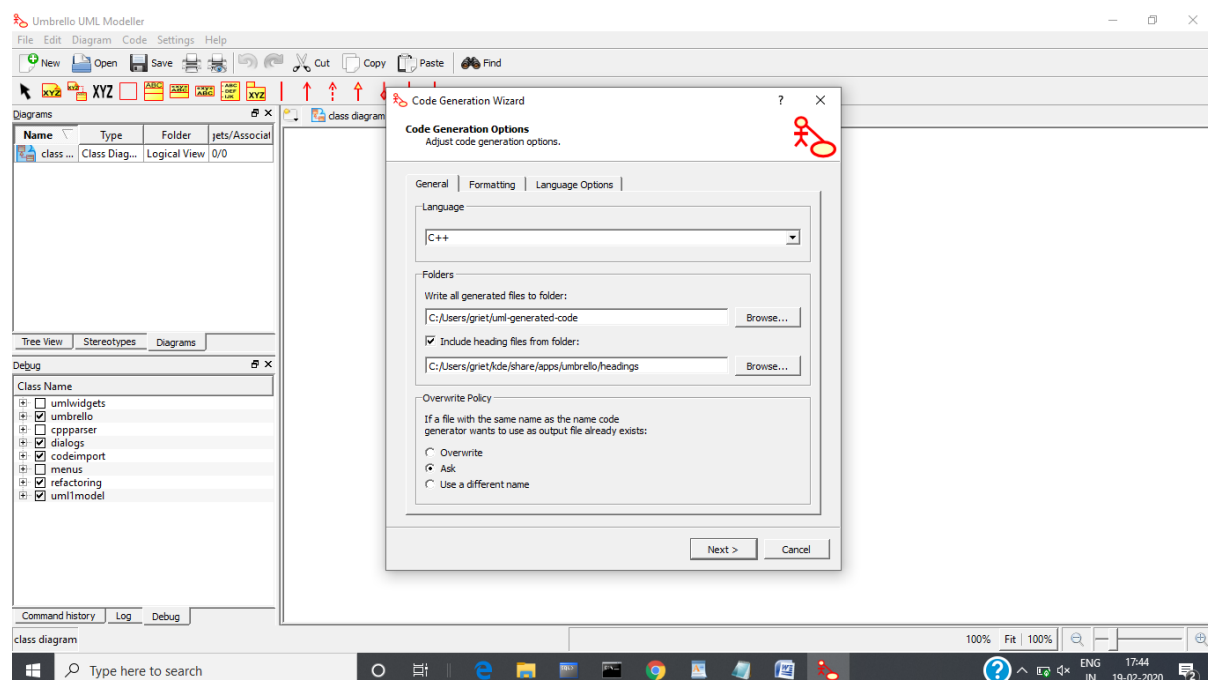
Umbrello UML Modeller 2 comes with code generation support for ActionScript, Ada, C++, C#, D, IDL, Java™, JavaScript, MySQL and Pascal.

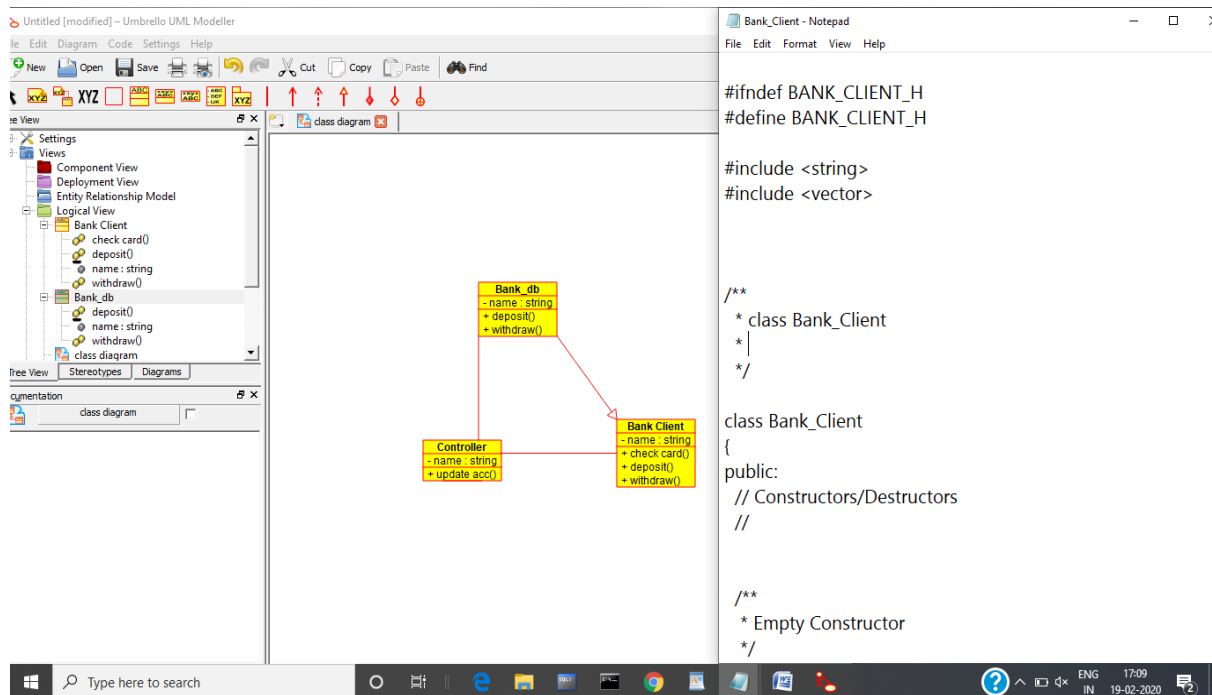
## Generating Code

In order to generate code with Umbrello UML Modeller, you first need to create or load a Model containing at least one class. When you are ready to start writing some code, select the Code Generation Wizard entry from the Code menu to start a wizard which will guide you through the code generation process.

The first step is to select the classes for which you want to generate source code. By default all the classes of your model are selected, and you can remove the ones for which you do not want to generate code by moving them to the left-hand side list.

The next step of the wizard allows you to modify the parameters the Code Generator uses while writing your code. The following options are available:





**Fig:2:** Forward Engineer a Class Diagram for ATM

## TASK 8

**Task 8:** Perform Reverse Engineering, and Generate documentation of the project.

**Aim:** To perform Reverse Engineering, and Generate documentation of the project.

**Reverse engineering:** It is the process of transforming code into a model through a mapping from a specific implementation language. Reverse engineering results in a flood of information, some of which is at a lower level of detail than you'll need to build useful models. At the same time, reverse engineering is incomplete. There is a loss of information when forward engineering models into code, and so you can't completely recreate a model from code unless your tools encode information in the source comments that goes beyond the semantics of the implementation language.

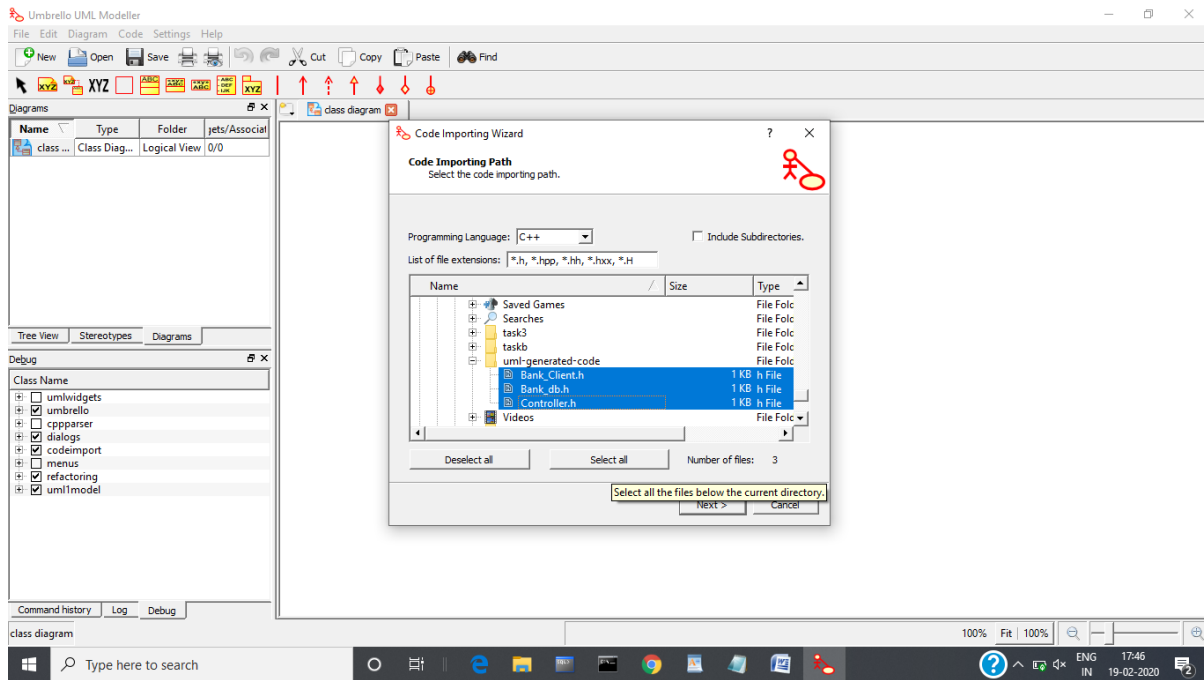
To reverse engineer a class diagram

- Identify the rules for mapping from your implementation language or languages of choice. This is something you'll want to do for your project or your organization as a whole.
- Using a tool, point to the code you'd like to reverse engineer. Use your tool to generate a new model or modify an existing one that was previously forward engineered. It is unreasonable to expect to reverse engineer a single concise model from a large body of code. You need to select portion of the code and build the model from the bottom.
- Using your tool, create a class diagram by querying the model. For example, you might start with one or more classes, then expand the diagram by following specific relationships or other neighboring classes. Expose or hide details of the contents of this class diagram as necessary to communicate your intent.
- Manually add design information to the model to express the intent of the design that is missing or hidden in the code.

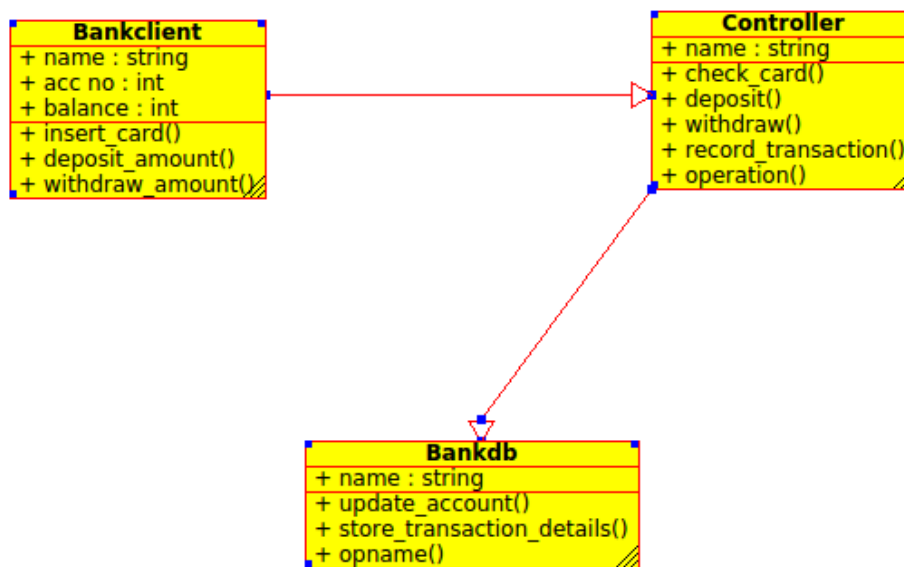
### Code Generation

Umbrello UML Modeller can generate source code for various programming languages based on your UML Model to help you get started with the implementation of your project. The code generated consists of the class declarations, with their methods and attributes so you can “fill in the blanks” by providing the functionality of your classes' operations.

Umbrello UML Modeller can help you create a model of your system from the source code by analysing your source code and importing the classes found in it.



**Fig:3:** Code Importing Path Using Umbrello UML Modeller for Reverse Engineering



**Fig:4:** Class diagram for ATM



## TASK 9

Write a C++ program to demonstrate the working of the following constructs:

i) while      ii) if ...else      iii) Switch      iv) for Loops in C++ language

### i) Using While:

```
#include<iostream>

using namespace std;

int main ()
{
    int i, n=5,j=1;
    cout<<"Enter a no\n";
    cin>>i;
    while ((i>0 && j<n) || (i<0 && j<n))
    {
        if(i%2==0)
        {
            cout<<i;
            cout<<" is a even number";
            i++;
            j++;
        }
        else
        {
            cout<<i;
            cout<<" is a odd number";
            i++;
            j++;
        }
    }
    return 0;
}
```

### Testcase:1

C:\Users\anush\Desktop\TestWhile.exe

```
Enter a no
2
2 is a even number
-----
Process exited after 3.581 seconds with return value 0
Press any key to continue . . .
```

### Testcase:2

C:\Users\anush\Desktop\TestWhile.exe


```
Enter a no
3
3 is a odd number
-----
Process exited after 10.3 seconds with return value 0
Press any key to continue . . .
```

### Testcase:3

C:\Users\anush\Desktop\Desktop\_Dump\Ramesh\_Sir\_C++\_Programs\TestWhile.exe

```
Enter a no
-2
-2 is a even number
-----
Process exited after 3.14 seconds with return value 0
Press any key to continue . . .
```

## Testcase:4

 C:\Users\anush\Desktop\Desktop\_Dump\Ramesh\_Sir\_C++\_Programs\TestWhile.exe

```
Enter a no
-1
-1 is a odd number
-----
Process exited after 8.496 seconds with return value 0
Press any key to continue . . .
```

### ii) Using If..Else:

```
#include<iostream>

using namespace std;

int main ()
{
    int i;
    cout<<"enter a number"<<endl;
    cin>>i;
    if(i%2==0)
    {
        cout<<i;
        cout<<" is a even number"<<endl;
    }
    else
    {
        cout<<i;
        cout<<" is a odd number"<<endl;
    }
    return 0;
}
```

### Testcase:1

C:\Users\anush\Desktop\Ramesh\_Sir\_C++\_Programs\TestIfElse.exe

```
enter a number
2
2 is a even number

-----
Process exited after 158.5 seconds with return value 0
Press any key to continue . . .
```

### Testcase:2

C:\Users\anush\Desktop\Ramesh\_Sir\_C++\_Programs\TestIfElse.exe

```
enter a number
3
3 is a odd number

-----
Process exited after 6.541 seconds with return value 0
Press any key to continue . . .
```

### Testcase:3

C:\Users\anush\Desktop\Ramesh\_Sir\_C++\_Programs\TestIfElse.exe

```
enter a number
-2
-2 is a even number

-----
Process exited after 1.682 seconds with return value 0
Press any key to continue . . .
```

## Testcase:4

C:\Users\anush\Desktop\Ramesh\_Sir\_C++\_Programs\TestIfElse.exe

```
enter a number
-3
-3 is a odd number

-----
Process exited after 3.464 seconds with return value 0
Press any key to continue . . .
```

### iii) Using Switch:

```
#include<iostream>
#include<stdexcept>
using namespace std;
int Division(int a, int b)
{
    if (b == 0) {
        throw runtime_error("Math error: Attempted to divide by Zero\n");
    }
    return (a /b);
}
int main()
{
    int i,a,b,c;

    cout<<"1.Add/n 2.Sub /n 3.Mul /n 4.Div /n Enter Your choice"<<endl;
    cin>>i;
    cout<<"Enter a,b values";
    cin>>a>>b;
    switch(i)
    {
        case 1: c=a+b;
```

```

cout<<"The sum of a & b is:" << c<<endl;
break;
case 2: c=a-b;
cout<<"The Diff of a & b is: " << c<<endl;
break;
case 3: c=a*b;
cout<<"The Mul of a & b is: " << c<<endl;
break;
case 4: try
    {
        int c =Division(a,b);
        cout<<"The Div of a & b is: " << c<<endl;
    }
    catch (runtime_error &e)
    {

        // prints that exception has occurred
        // calls the what function
        // using runtime_error object
        cout << "Exception occurred " << e.what()<<endl;
    }

break;
default: cout<<"Enter your choice"<<endl;
break;
}
return 0;
}

```

### Testcase:1

```
C:\Users\anush\Desktop\Ramesh_Sir_C++_Programs\TestSwitch.exe
1.Add/n 2.Sub /n 3.Mul /n 4.Div /n Enter Your choice
1
Enter a,b values 3 2
The sum of a & b is:5

-----
Process exited after 5.387 seconds with return value 0
Press any key to continue . . .
```

### Testcase:2

```
C:\Users\anush\Desktop\Ramesh_Sir_C++_Programs\TestSwitch.exe
1.Add/n 2.Sub /n 3.Mul /n 4.Div /n Enter Your choice
4
Enter a,b values 3 2
The Div of a & b is: 1

-----
Process exited after 7.921 seconds with return value 0
Press any key to continue . . .
```

### Testcase:3

```
C:\Users\anush\Desktop\Ramesh_Sir_C++_Programs\TestSwitch.exe
1.Add/n 2.Sub /n 3.Mul /n 4.Div /n Enter Your choice
3
Enter a,b values 3 2
The Mul of a & b is: 6

-----
Process exited after 6.218 seconds with return value 0
Press any key to continue . . .
```

### Testcase:4

C:\Users\anush\Desktop\Ramesh\_Sir\_C++\_Programs\TestSwitch.exe

```
1.Add/n 2.Sub /n 3.Mul /n 4.Div /n Enter Your choice
2
Enter a,b values 3 2
The Diff of a & b is: 1

-----
Process exited after 4.863 seconds with return value 0
Press any key to continue . . .
```

### Testcase:5

C:\Users\anush\Desktop\Desktop\_Dump\Ramesh\_Sir\_C++\_Programs\TestSwitch.exe

```
1.Add/n 2.Sub /n 3.Mul /n 4.Div /n Enter Your choice
4
Enter a,b values 12 0
Exception occurred Math error: Attempted to divide by Zero

-----
Process exited after 5.383 seconds with return value 0
Press any key to continue . . .
```

### iv) Using For:

```
#include<iostream>

using namespace std;

int main ()
{
    int i,n;

    cout<<"enter a no"<<endl;
    cin>>n;
    if (n>0)
    {
```



```

for(i=n;i<=5;i++)
{
    if(i%2==0)
    {
        cout<<i ;
        cout<<" is a even no\n";
    }
    else
    {
        cout<<i;
        cout<<" is a odd no\n";
    }
}
else
{
    for(i=n;i<=0;i++)
    {
        if(i%2==0)
        {
            cout<<i ;
            cout<<" is a even no\n";
        }
        else
        {
            cout<<i;
            cout<<" is a odd no\n";
        }
    }
}
return 0;
}

```

## Testcase:1

C:\Users\anush\Desktop\Desktop\_Dump\Ramesh\_Sir\_C++\_Programs\TestFor.exe

```
enter a no
2
2 is a even no
3 is a odd no
4 is a even no
5 is a odd no

-----
Process exited after 9.739 seconds with return value 0
Press any key to continue . . .
```

## Testcase:2

C:\Users\anush\Desktop\Desktop\_Dump\Ramesh\_Sir\_C++\_Programs\TestFor.exe

```
enter a no
-2
-2 is a even no
-1 is a odd no
0 is a even no

-----
Process exited after 5.264 seconds with return value 0
Press any key to continue . . .
```

## TASK 10

**Task 10:** Create a test plan document for any application (e.g. Unified Library System)

**Aim:** To create a test plan document for any application (e.g. Unified Library System)

The Unified Library System is an online application for assisting a librarian managing book library in a University. The system would provide basic set of features to add/update clients, add/update books, search for books, and manage check-in / checkout processes. Our test group tested the system based on the requirement specification. This test report is the result for testing in the Unified Library System.

It mainly focuses on two problems

### 1. What we will test

### 2. How we will test.

### 3. GUI test

**Pass criteria:** Librarians could use this GUI to interface with the backend library database without any difficulties.

### 4. Database test

**Pass criteria:** Results of all basic and advanced operations are normal.

### 5. Basic function test Add a student

Each customer/student should have following attributes: Student ID/SSN (unique), Name, Address and Phone number.

The retrieved customer information by viewing customer detail should contain the four attributes.

### 6. Update/delete student

1. The record would be selected using the student ID.
2. Updates can be made on full. Items only: Name, Address, Phone number The record can be deleted if there are no books issued by user. The updated values would be reflected if the same customer's ID/SSN is called for.

### 7. Check-in book

Librarians can check in a book using its call number

1. The check-in can be initiated from a previous search operation where user has selected a Set of books.
2. The return date would automatically reflect the current system date.
3. Any late fees would be computed as difference between due date and return date at rate of 10 cents a day

## Test Cases for Unified Library System

### Login Form:

| S. No | Test Case   | Expected Result                         | Test Result |
|-------|---|---|-------------|
| 1     | Enter valid name and Password & click on login button | Software should display main window     | Successful  |
| 2     | Enter Invalid   | Software should not display main window | Successful  |

### Book Entry Form:

| S. No | Test Case                     | Expected Result  | Test Result |
|-------|-------------------------------|--|-------------|
| 1     | On the click of ADD Button    | At first user have to fill all fields with proper data, if any error like entering text data instead of number or entering number instead of text is found then it gives proper message otherwise adds record to the database. | Successful  |
| 2     | On the click of DELETE Button | This deletes the details of book by using accession no.  | Successful  |
| 3     | On the click of UPDATE Button | Modified records are updated in database by clicking update button.  | Successful  |
| 4     | On the click of SEARCH Button | Displays the details of Book for entered accession no. Otherwise it gives proper error message.  | Successful  |
| 5     | On the click of CLEAR Button  | Clears all fields  | Successful  |
| 6     | On the click of EXIT Button   | Exit the current book details of form  | Successful  |
| 7     | On the click of NEXT Button   | Display the next form  | Successful  |

### User Account Form:

| S. No | Test Case                     | Expected Result  | Test Result |
|-------|-------------------------------|--|-------------|
| 1     | On the click of ADD Button    | At first user have to fill all fields with proper data, if any error like entering text data instead of number or entering number instead of text is found then it gives proper message otherwise adds record to the database. | Successful  |
| 2     | On the click of DELETE Button | This deletes the details of student by using register no.  | Successful  |
| 3     | On the click of UPDATE Button | Modified records are updated in database by clicking update button.  | Successful  |
| 4     | On the click of SEARCH Button | Displays the details of Book for entered register no. Otherwise it gives proper error message.   | Successful  |

|   |                              |                                       |            |
|---|------------------------------|---------------------------------------|------------|
| 5 | On the click of CLEAR Button | Clears all fields                     | Successful |
| 6 | On the click of EXIT Button  | Exit the current book details of form | Successful |
| 7 | On the click of NEXT Button  | Display the next form                 | Successful |

### Book Issue Form:

| S.No | Test Case                     | Expected Result   | Test Result |
|------|-------------------------------|---|-------------|
| 1.   | On the click of ADD button    | At first user have to fill all fields with proper data, if the accession number book is already issued then it will giving proper message | Successful  |
| 2.   | On the click of DELETE button | This deletes the details of book by using Register no.  | Successful  |
| 3.   | On the click of UPDATE button | Modified records are updated in database by clicking UPDATE button  | Successful  |
| 4.   | On the click of SEARCH button | Displays the details of returned book...Otherwise gives proper Error message  | Successful  |
| 5.   | On the click of CLEAR button  | Clears all the fields   | Successful  |
| 6.   | On the click of EXIT button   | Exit the current book details form  | Successful  |
| 7.   | On the click of NEXT button   | Displays the nest form  | Successful  |

### Book Return Form:

| S. No | Test Case                     | Expected Result   | Test Result |
|-------|-------------------------------|---|-------------|
| 1.    | On the click of ADD button    | At first user have to fill all fields with proper data, if any error like entering text data instead of number or entering number instead of text..is found then it gives proper message otherwise Adds Record to the Database. | Successful  |
| 2.    | On the click of DELETE button | Which deletes the details of book by using Register no.   | Successful  |
| 3.    | On the click of UPDATE button | Modified records are updated in database by clicking UPDATE button  | Successful  |

|    |                               |  |            |
|----|-------------------------------|--|------------|
| 4. | On the click of SEARCH button | Displays the details of returned book...Otherwise gives proper Error message | Successful |
| 5. | On the click of CLEAR button  | Clears all the fields  | Successful |
| 6. | On the click of EXIT button   | Exit the current book details form   | Successful |
| 7. | On the click of NEXT button   | Displays the nest form   | Successful |

## TASK 11

**Task 11:** Implement a Junit Test program and design test cases to find the maximum of an array of numbers.

**Aim:** To implement a Junit Test program and design test cases to find the maximum of an array of numbers.

### TEST LOGIC

```
package com.javatpoint.logic;

public class Calculation {

    public static int findMax(int arr[]) {
        int max = arr[0]; // arr[0] instead of 0
        for (int i = 1; i < arr.length; i++) {
            if (max < arr[i])
                max = arr[i];
        }
        return max;
    }
}
```

### TEST CASE

```
import static org.junit.Assert.*;
import org.junit.Test;

public class TestLogic {

    @Test
    public void testFindMax(int arr[])
    {
        assertEquals(4, Calculation.findMax(new int[] { 1, 3, 4, 2 }));
        assertEquals(-1, Calculation.findMax(new int[] { -12, -1, -3, -4, -2 }));
    }
}
```

## TASK 12

**Task 12:** Implement a Junit Test program and design test cases to count the number of elements in array of numbers.

**Aim:** Implement a Junit Test program and design test cases to count the number of elements in array of numbers.

### TEST LOGIC

```
public class CountArray {  
    public static void count( ) {  
        //Initialize array  
        int [] a = new int [] { 1, 2, 3, 4, 5};  
        //Number of elements present in an array can be found using the length  
        System.out.println("Number of elements present in given array: " +  
            arr.length);  
    }  
}
```

### TEST CASE

```
import static org.junit.Assert.*;  
import org.junit.Test;  
public class TestLogic {  
    @Test  
    public void testcount(){  
        assertEquals(4,Calculation.count(new int[]{ 1,3,4,2}));  
        assertEquals(2,Calculation.count(new int[]{ 2,3}));  
        assertEquals(0,Calculation.count(new int[]{}));  
    }  
}
```