Practical-1

Aim: An introduction to software engineering & Studying Various phases of Water-Fall Model. For each SDLC phase, identify the objectives and summaries outcomes.

What is Software Engineering?

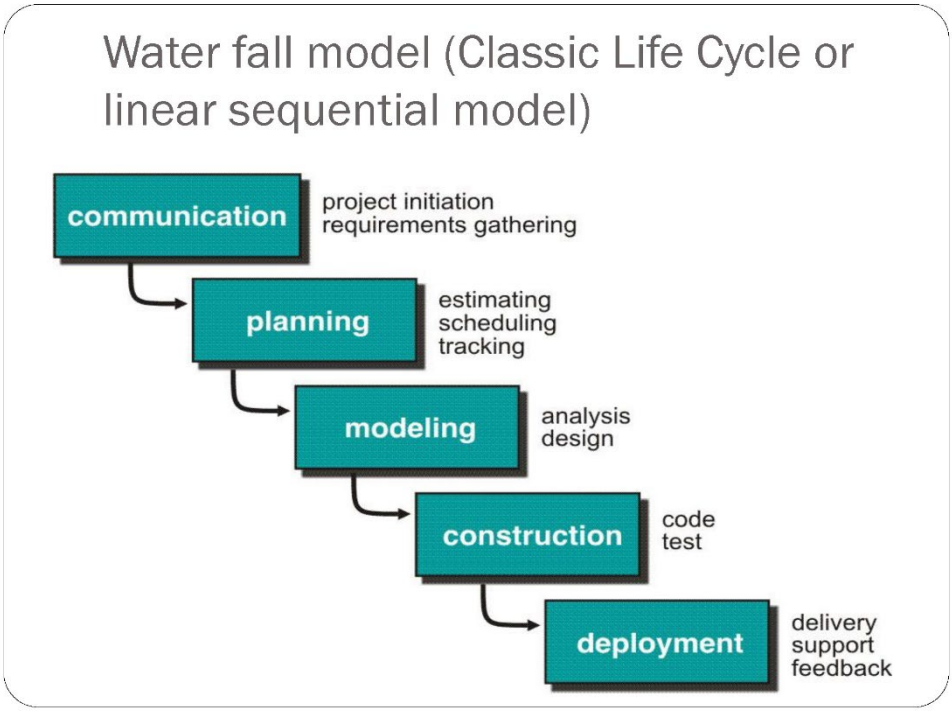
Software engineering is defined as a process of analyzing user requirements and then designing, building, and testing software applications which will satisfy those requirements.

Software Engineering is the science and art of building a software systems that are:

1) on time
2) on budget
3) with acceptable performance
4) with correct operation

Waterfall Model

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. The Waterfall model is the earliest SDLC approach that was used for software development. The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.



The sequential phases in Waterfall model are –

Requirement Gathering and analysis

All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

System Design

The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

Implementation

With inputs from the system design, the system is first developed in small programs called

units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

Testing

All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

Deployment of system

Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

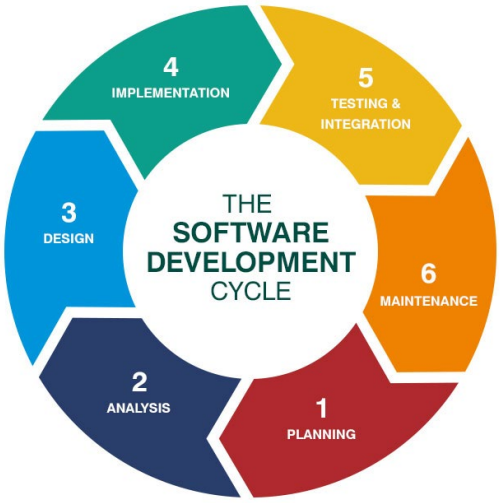
Maintenance

There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the a defined set of goals are achieved for the previous phase and it is signed off. In this model, phases do not overlap.

Software Development Life Cycle (SDLC):

Software Development Life Cycle (SDLC) is a well-defined sequence of stages in software engineering to develop the intended software product.
SDLC provides a series of steps to be followed to design and develop a software product efficiently. SDLC framework includes the following steps:



1. Planning

This step onwards the software development team works to carry on the project. The team holds discussions with various stakeholders from the problem domain and tries to bring out as much information as possible on their requirements. The requirements are contemplated and segregated into user requirements, system requirements and functional requirements. The requirements are collected using a number of practices such as studying the existing or obsolete system and software, conducting interviews of users and developers, referring to the database or collecting answers from the questionnaires.

After requirement gathering, the team comes up with a rough plan of the software process. At this step the team analyzes if a software can be made to fulfill all requirements of the user and if there is any possibility of software being no more useful. It is found out, if the project is financially, practically, and technologically feasible for the organization to take up.

2. Analysis

At this step the developers decide a roadmap of their plan and try to bring up the best software model suitable for the project. System analysis includes Understanding of software product limitations, learning system related problems or changes to be done in existing systems

beforehand, identifying and addressing the impact of a project on organization and personnel etc. The project team analyzes the scope of the project and plans the schedule and resources accordingly.

3. Design

Next step is to bring down the whole knowledge of requirements and analysis on the desk and design the software product. The inputs from users and information gathered in the requirement gathering phase are the inputs of this step. The output of this step comes in the form of two designs; logical design and physical design. Engineers produce meta-data and data dictionaries, logical diagrams, data-flow diagrams and in some cases pseudo codes.

4. Implementation

This step is also known as the programming phase. The implementation of software design starts in terms of writing program code in the suitable programming language and developing error- free executable programs efficiently.

5. Integration and Testing

An estimate says that 50% of the whole software development process should be tested. Errors may ruin the software from critical level to its own removal. Software testing is done while coding by the developers and thorough testing is conducted by testing experts at various levels of code such as module testing, program testing, product testing, in-house testing and testing the product at user's end. Early discovery of errors and their remedy is the key to reliable software.

Software may need to be integrated with the libraries, databases and other program(s). This stage of SDLC is involved in the integration of software with outer world entities.

6. Maintenance

This phase confirms the software operation in terms of more efficiency and less errors. If required, the users are trained on, or aided with the documentation on how to operate the software and how to keep the software operational. The software is maintained timely by updating the code according to the changes taking place in the user end environment or technology. This phase may face challenges from hidden bugs and real-world unidentified problems.

Library Information System

Objective:

The main objective of the Library Information System is to manage the details of Address, Member, Issues, Books, Student. It manages all the Information about Address, Student, Address. The project is totally built at administrative end and thus only the administrator is guaranteed the access. The purpose of the project is build an application program to reduce the manual work for managing the Address, Member, Librarian, Issues. It tracks all the details about the issues, Books, Student.

Outcomes:

1. Helps streamline day-to-day library operations
2. Simple and easy to use
3. Slashes down the operating costs
4. Boosts learner engagement
5. Creates a smart library
6. Highly secure, scalable, and reliable library companion
7. 24/7 Mobile App access
8. Students become better readers

Practical-2

Aim: Define requirement Gathering and technical requirement specification for selected project.

What are requirements gathering?

Requirements gathering is the process of understanding what you are trying to build and why you are building it. Requirements gathering is often regarded as a part of developing software applications or of cyber-physical systems like aircraft, spacecraft, and automobiles (where specifications cover both software and hardware). It can, however, be applied to any product or project, from designing a new sailboat to building a patio deck to remodeling a bathroom.

A 6-Step Requirements Gathering Process:

The requirements gathering process consists of six steps. Three of those (steps three through five—the bulk of the process) we've already mentioned as the key subprocesses of requirements gathering. The full six steps are:

- Identify the relevant stakeholders
- Establish project goals and objectives
- Elicit requirements from stakeholders
- Document the requirements
- Confirm the requirements
- Prioritize the requirements

System Requirements:

The key requirements that need to be offered by the library information system can be classified into functional and non-functional requirements.

➢ Functional Requirements

- Allow the librarian to add and remove new members.
- Allow the user to search for books based on title, publication date, author, etc., and find their location in the library.
- Users can request, reserve, or renew a book.
- Librarian can add and manage the books.
- The system should notify the user and librarian about the overdue books.
- The system calculates the fine for overdue books on their return.

A more detailed list of key features that need to be supported by the system is given in the use case diagram.

There are 3 actors in the use case diagram. The User, The Librarian, and the System.

- **User** - The user can log in, view the catalog, search for books, checkout, reserve, renew and return a book.
- **Librarian** - The librarian registers new users, adds and maintains the books, collects fines for overdue books, and issues books to users who need them.
- **System** - The system is the library information system itself. It keeps track of the borrowed books and sends notifications to the user and librarian about the overdue books.

➢ Non - Functional Requirements

- **Usability** - Usability is the main non-functional requirement for a library information system. The UI should be simple enough for everyone to understand and get the relevant information without any special training. Different languages can be provided based on the requirements.

- **Accuracy** - Accuracy is another important non-functional requirement for the library information system. The data stored about the books and the fines calculated should be correct, consistent, and reliable.
- **Availability** - The System should be available for the duration when the library operates and must be recovered within an hour or less if it fails. The system should respond to the requests within two seconds or less.
- **Maintainability** - The software should be easily maintainable and adding new features and making changes to the software must be as simple as possible. In addition to this, the software must also be portable.

Software Requirements:

1. A server running Windows Server/Linux OS
2. A multi-threading capable backend language like Java
3. Front-end frameworks like Angular/React/Vue for the client
4. Relational DBMS like MySQL, PostgreSQL, etc.
5. Containers and orchestration services like Kubernetes (for a large setting like a national library).

Hardware Requirements:

Hardware is the primary requirement for library automation; different types of hardware are available in the market. A hardware specification depends on:

- Available budget.
- Size of the data to store.
- Usage load. • Required speed.
- Features to upgrade when it required.
- Availability of servicing (maintenance).
- Compatible with operating system, what we are going to use.

- Warranty period.

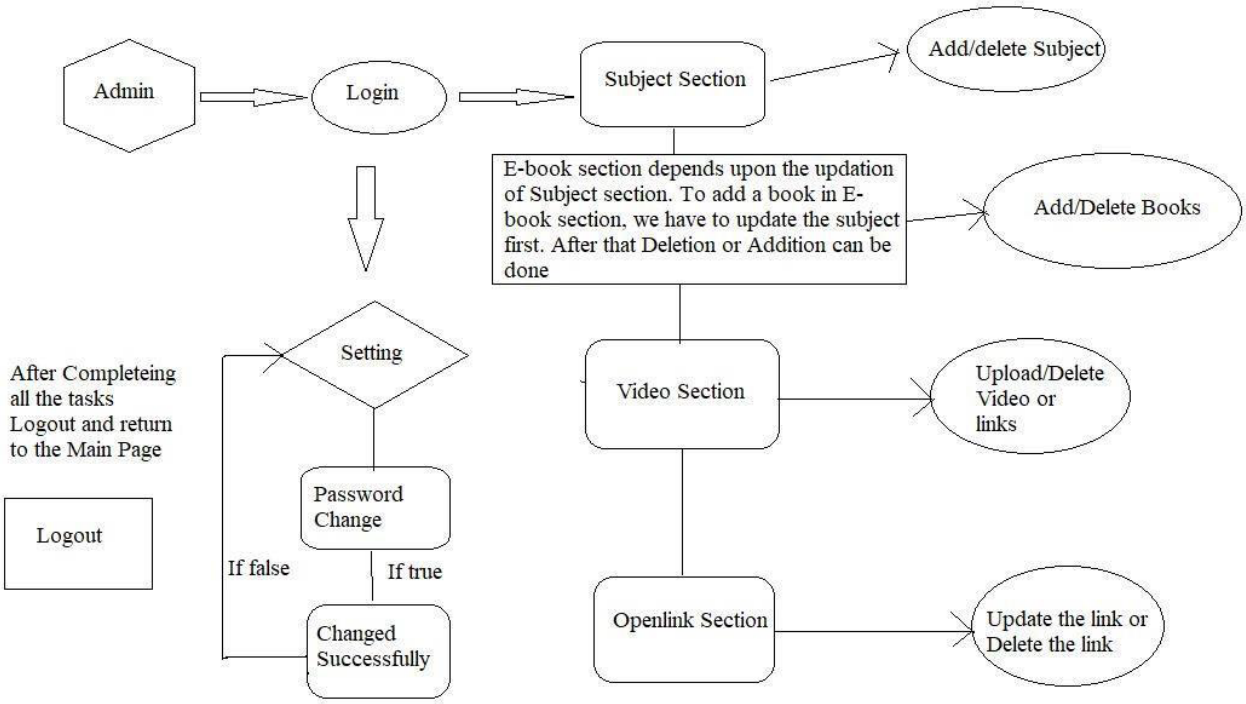
➢ Hardware Interfaces:

- Operating System: Windows
- Hard Disk: 40 GB
- RAM: 256 MB
- Processor: MINIMUM SPECS / 30 FPS. CPU //Intel Core 2 Duo E8400 (Intel), Athlon 200GE (AMD) GPU //Intel HD 4000, Radeon R5 200

Practical-3

Aim: Development of DFD and E-R diagram for the software domain problem. E-R diagram

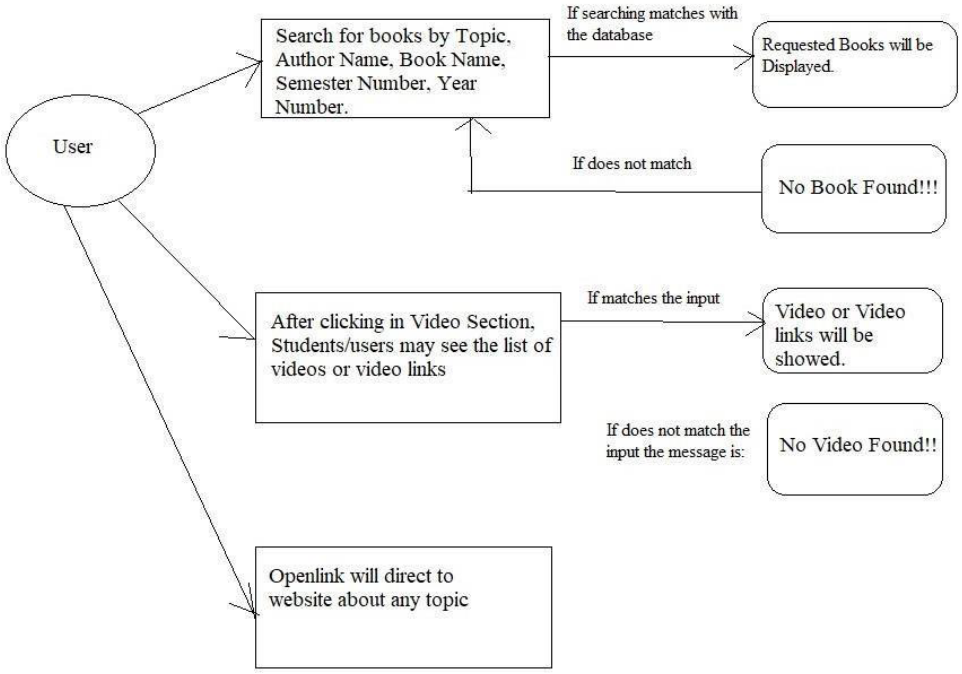
➢ Data Flow Diagram



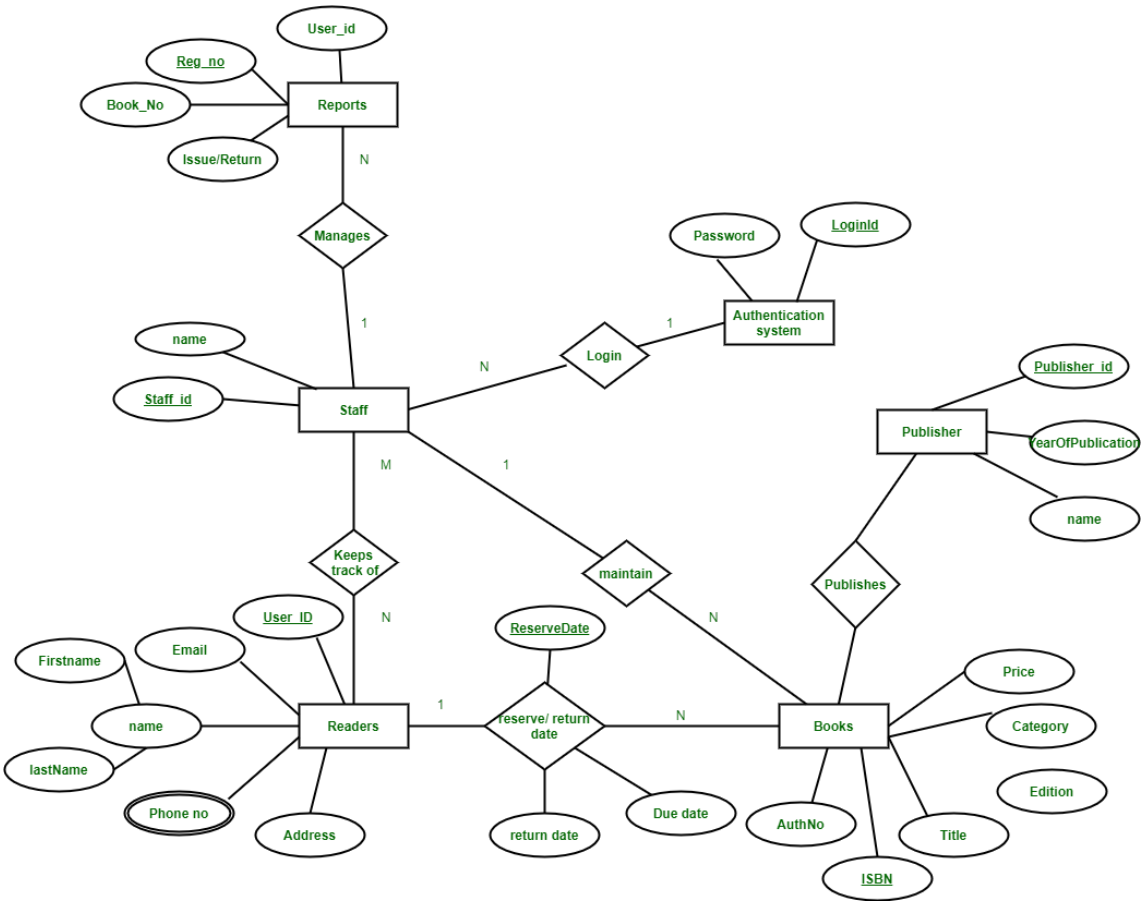
DATA FLOW DIAGRAM FOR ADMIN LOGIN

After entering to the home page of the website, Admin can choose the Admin Login option where they are asked to enter username & password, and if he/she is a valid user then a teacher login page will be displayed.

➢ Data Flow Diagram For User



➢ E-R Diagram



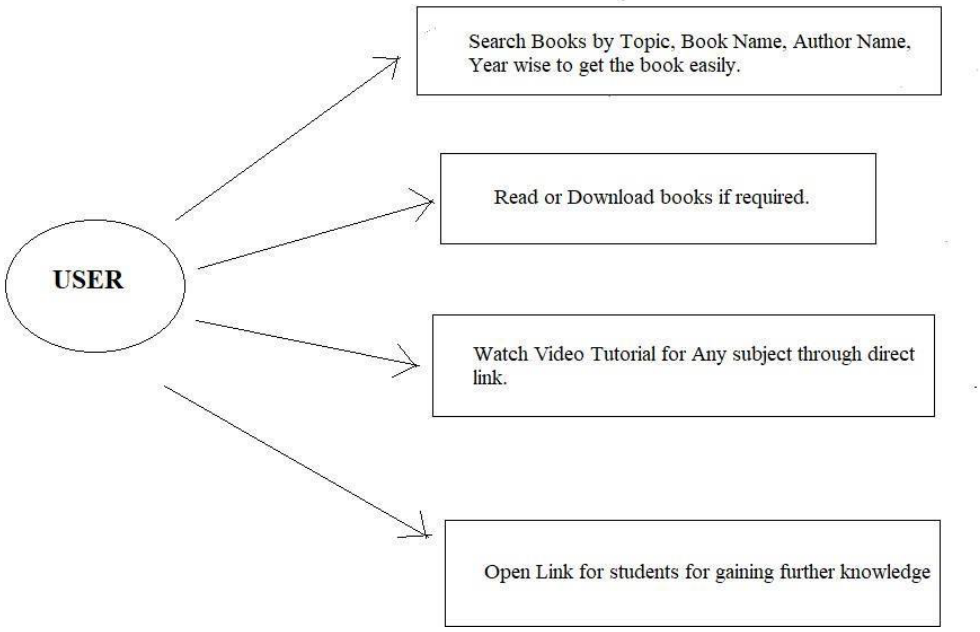
- **Entities and their Attributes –**
  1. **Book Entity:** It has authno, isbn number, title, edition, category, price. ISBN is the Primary Key for Book Entity.
  2. **Reader Entity:** It has UserId, Email, address, phone no, name. Name is composite attribute of firstname and lastname. Phone no is multi valued attribute. UserId is the Primary Key for Readers entity.

3. **Publisher Entity:** It has PublisherId, Year of publication, name. PublisherID is the Primary Key.
4. **Authentication System Entity:** It has LoginId and password with LoginID as Primary Key.
5. **Reports Entity:** It has UserId, Reg_no, Book_no, Issue/Return date. Reg_no is the Primary Key of reports entity.
6. **Staff Entity:** It has name and staff_id with staff_id as Primary Key.
7. **Reserve/Return Relationship Set:** It has three attributes: Reserve date, Due date, Return date.

Practical-4

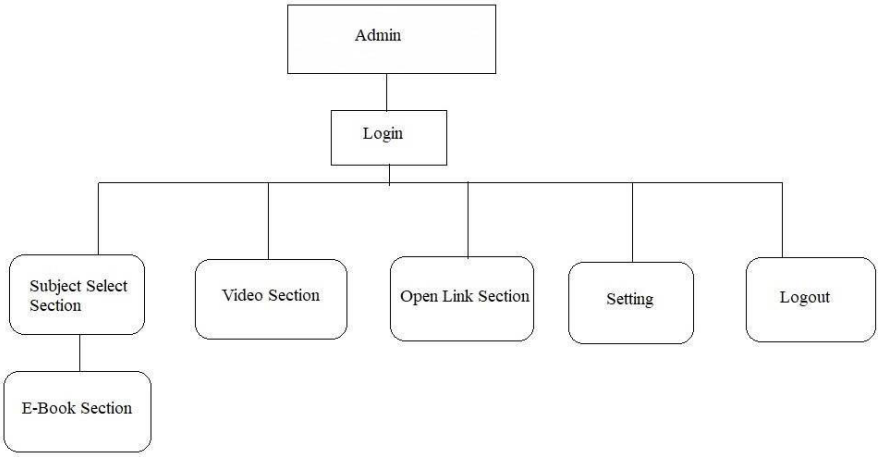
Aim: Draw Use case and Activity Diagrams for the project definition.

➢ User Case Diagram For User

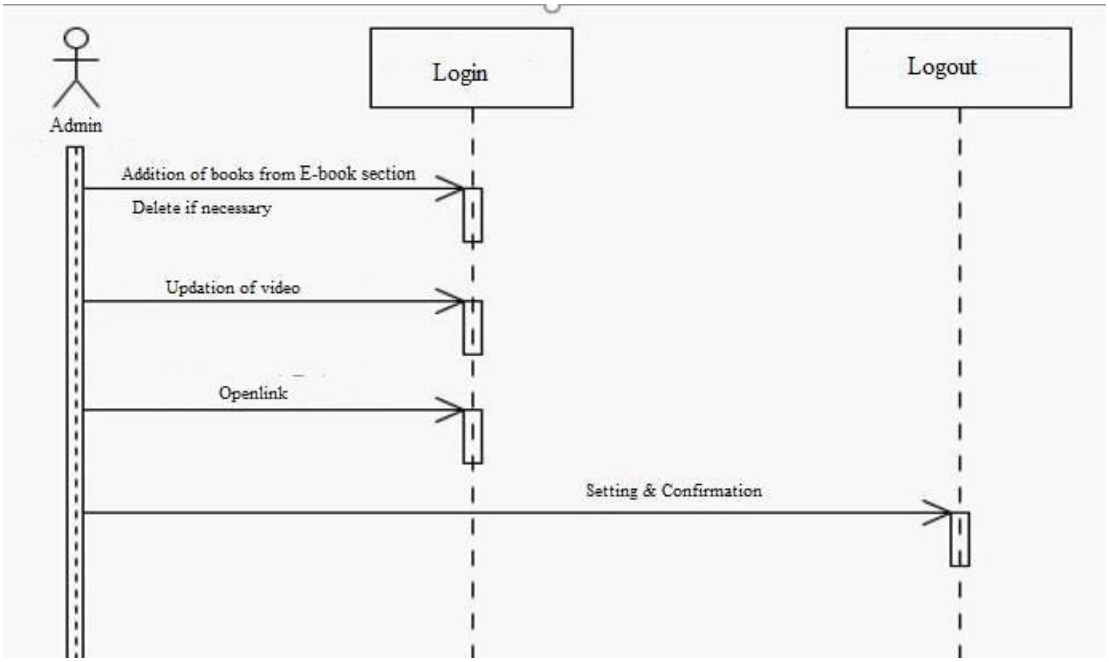


After entering to the home page of the website, student can choose the USER LOGIN option where they are asked to enter username & password, and if he/she is a valid user then a student login page will be displayed.

➢ User Case Diagram For Admin

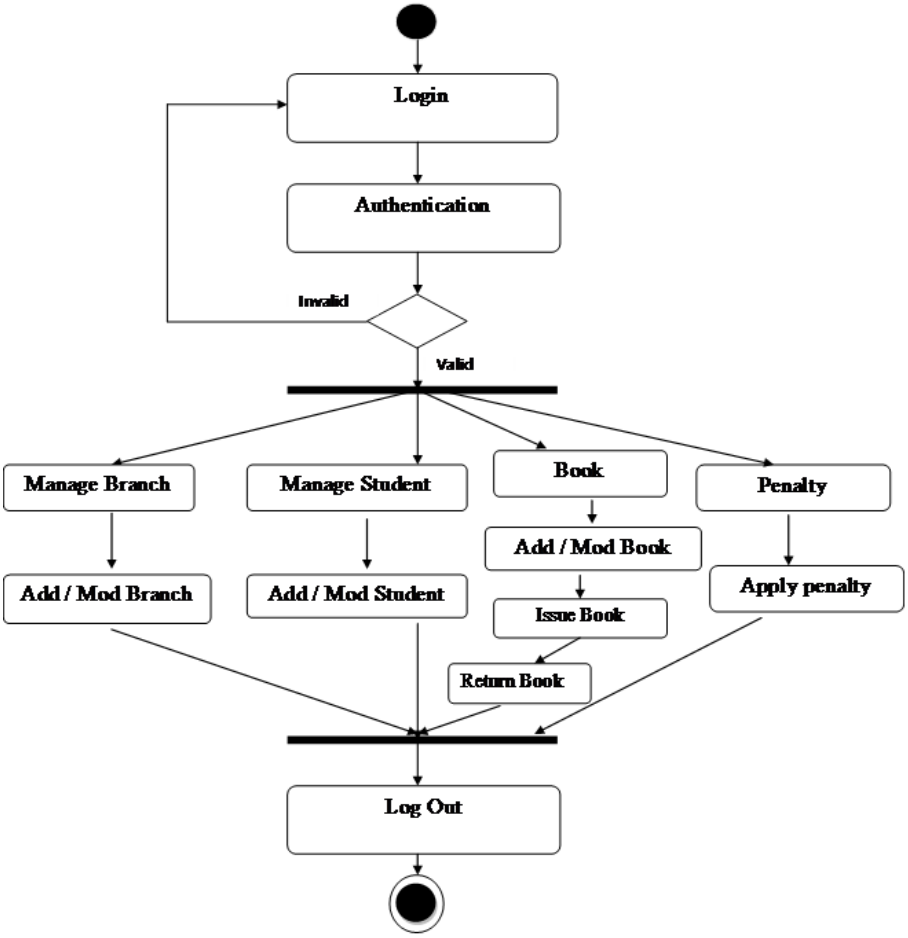


➢ Sequence Diagram

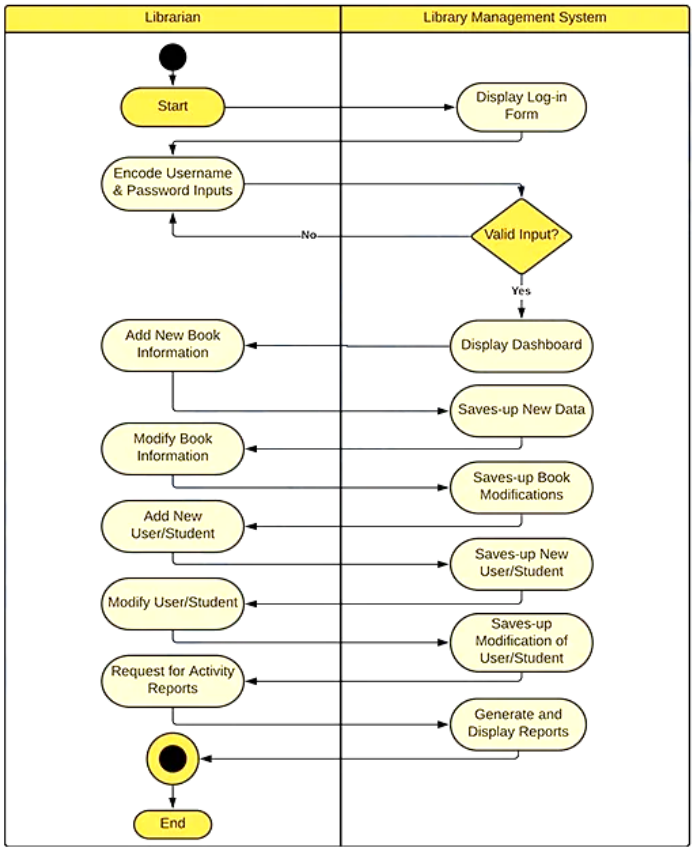


➢ Activity Diagram

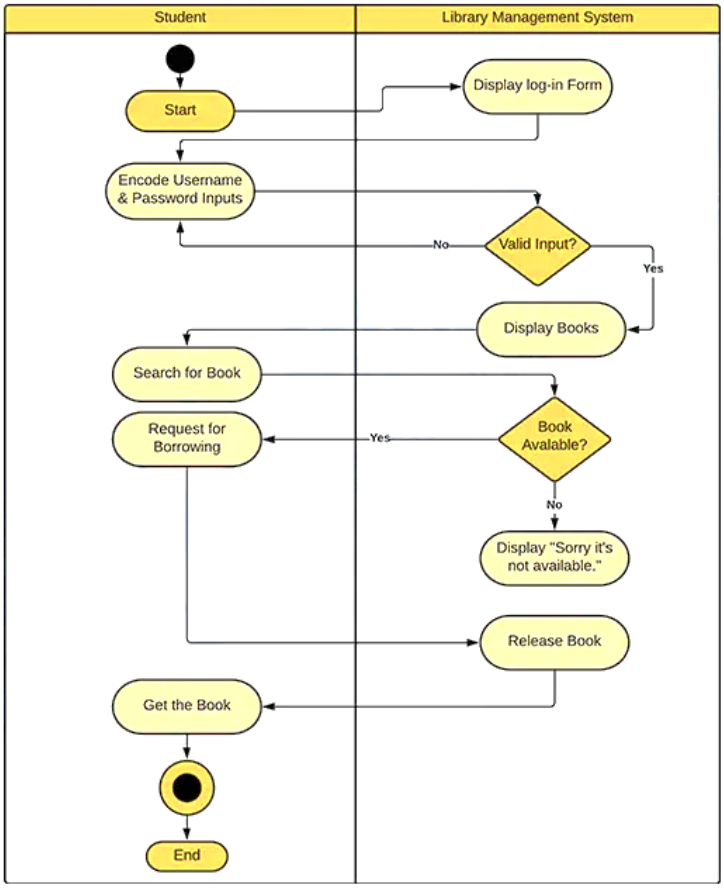
The activity diagram used to describe flow of activity through a series of actions. Activity diagram is an important diagram to describe the system. The activity described as an action or operation of the system.



Activity Diagram (Librarian) – This illustration shows the activities and scenarios done when the librarian accesses the system. All the actions and decisions included are all emphasized here.



Activity Diagram (for Students) – This diagram now shows the series of scenarios while using the library system. It illustrates the activities or events that are done when the system is in use. The system design can show you the functions on the student side.



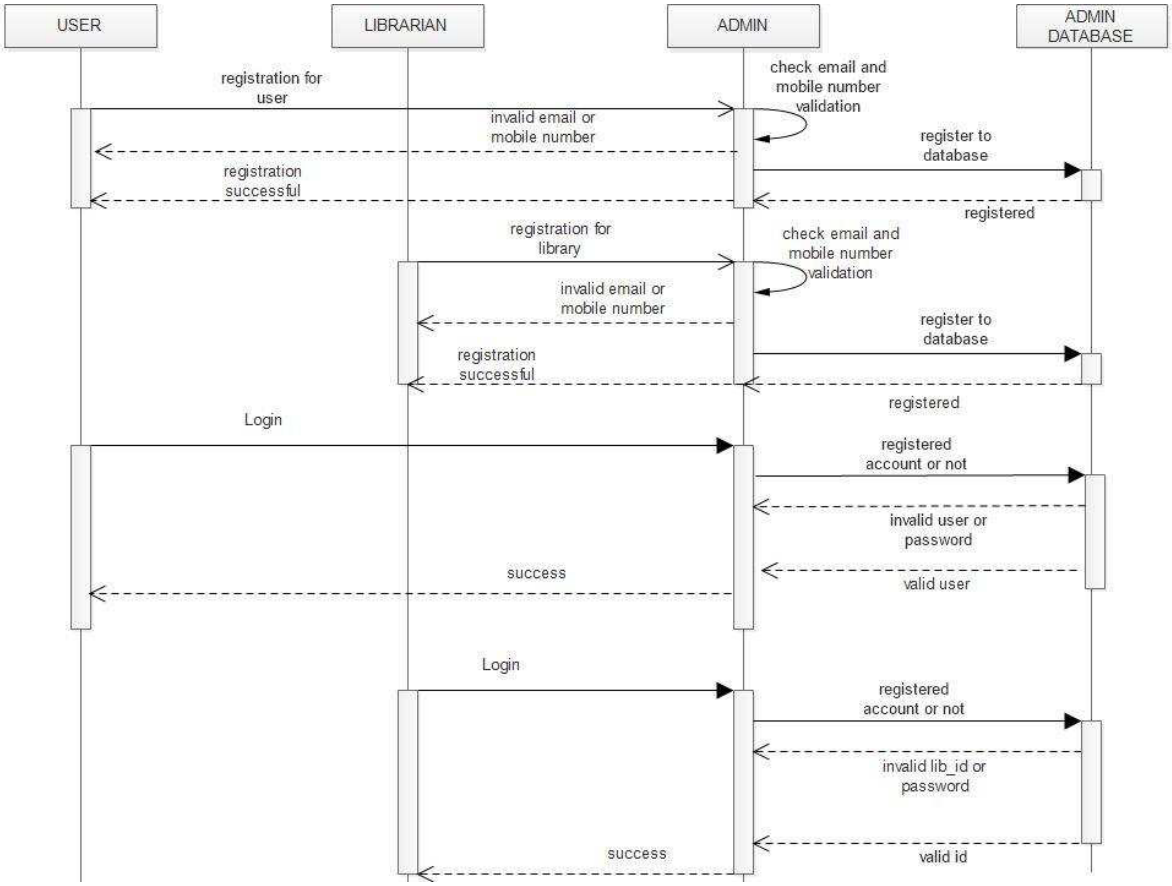
Practical-5

Aim: Draw the Sequence Diagram for the project definition.

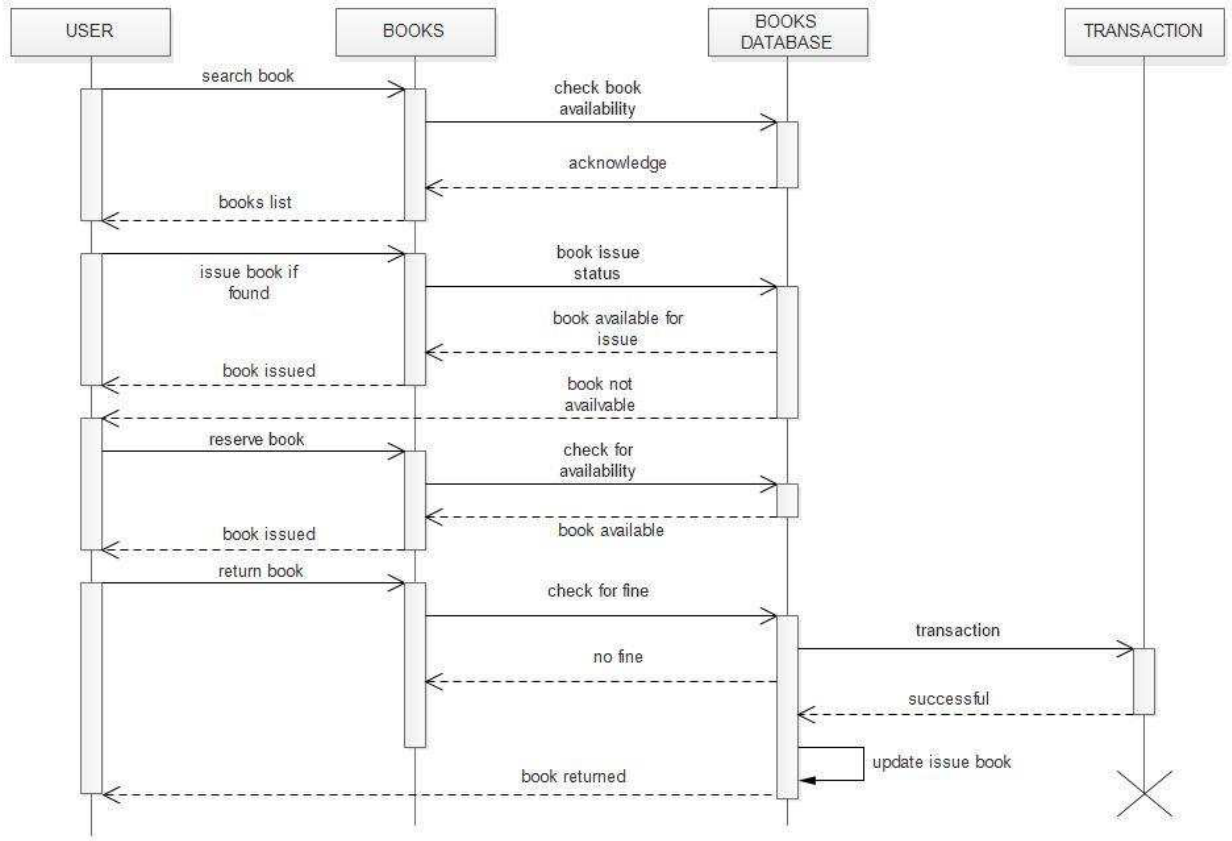
➢ Sequence Diagram

- A sequence diagram simply depicts interaction between objects in a sequential order i.e., the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

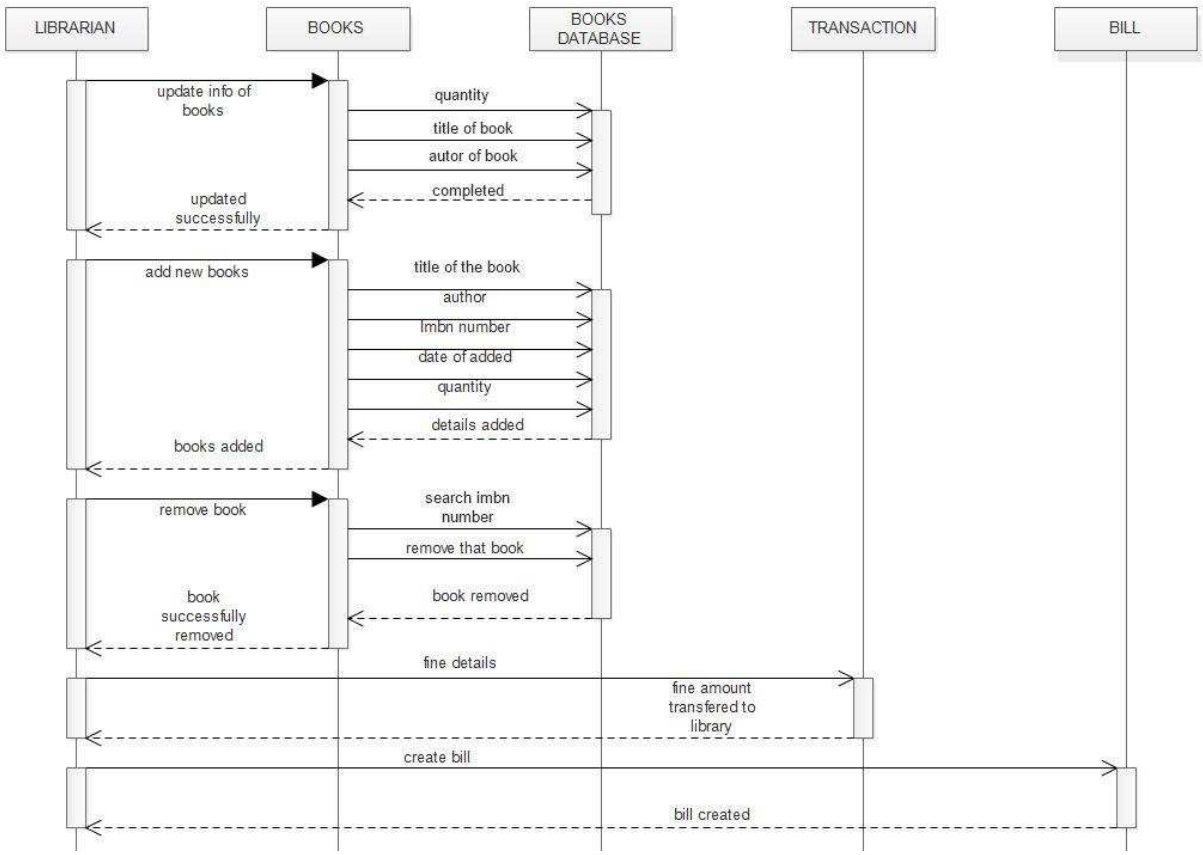
- **Register And Login Sequence Diagram**



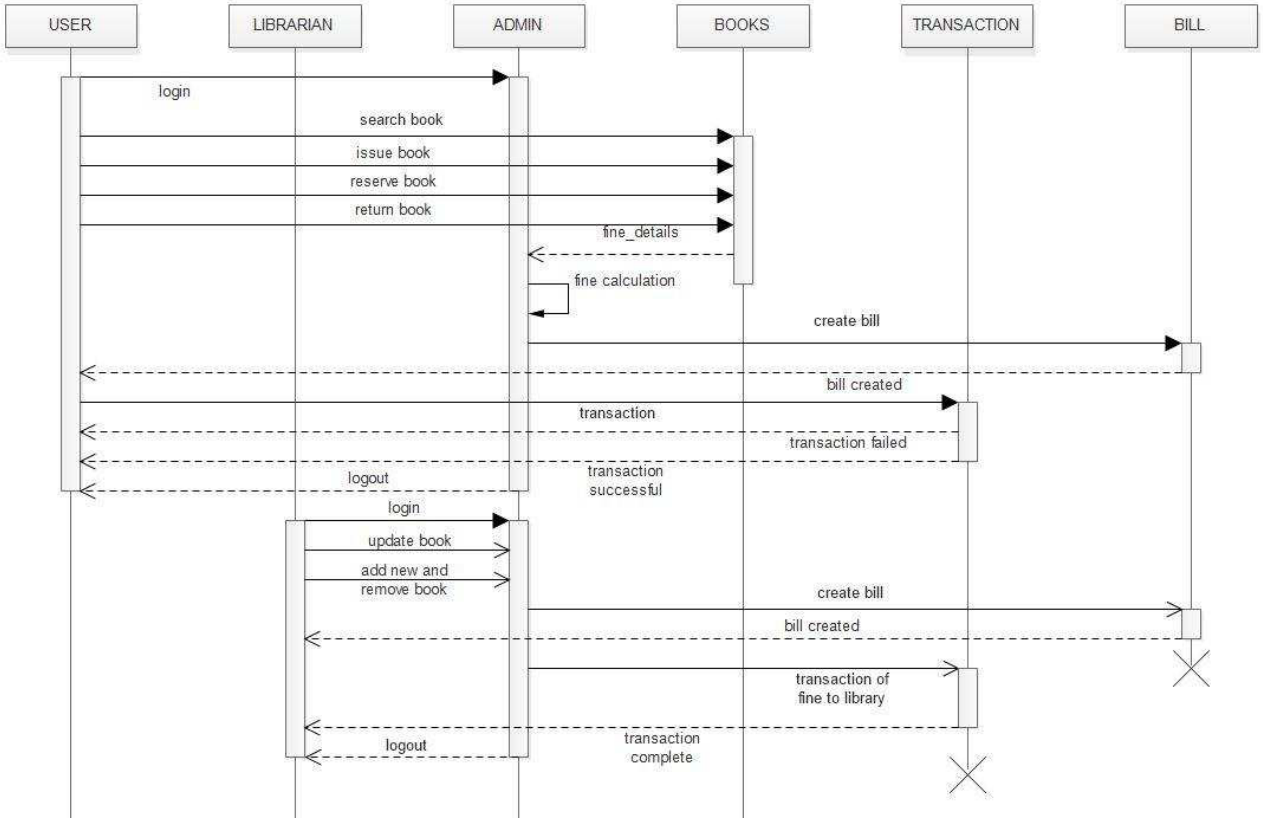
- **Services USER Could Use**



- **Services Of a Library**



- **Full Working of The System**



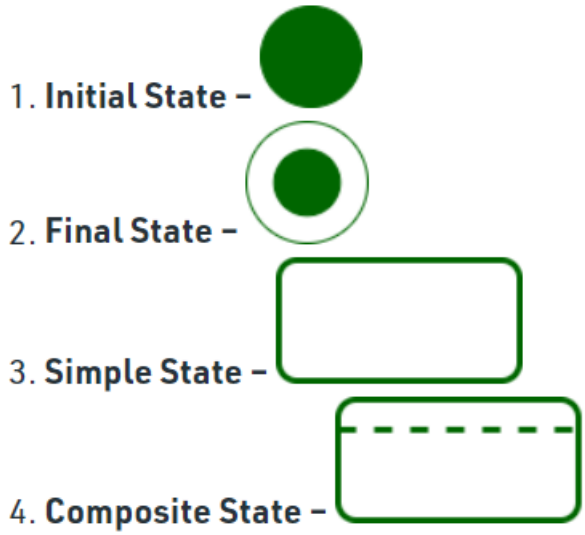
Practical-6

Aim: Development of State Transition Diagram for Software project definition.

➢ What is State Transition Diagram?

State Transition Diagram are also known as Dynamic models. As the name suggests, it is a type of diagram that is used to represent different transition (changing) states of a System. It is generally used to graphically represent all possible transition states a system can have and model such systems. It is very essential and important and right for object-oriented modeling from the beginning. The System consists of various states that are being represented using various symbols in the state transition diagram.

You can see the symbols and their description given below:



➢ Purpose of State chart Diagrams

Following are the main purposes of using State chart diagrams −

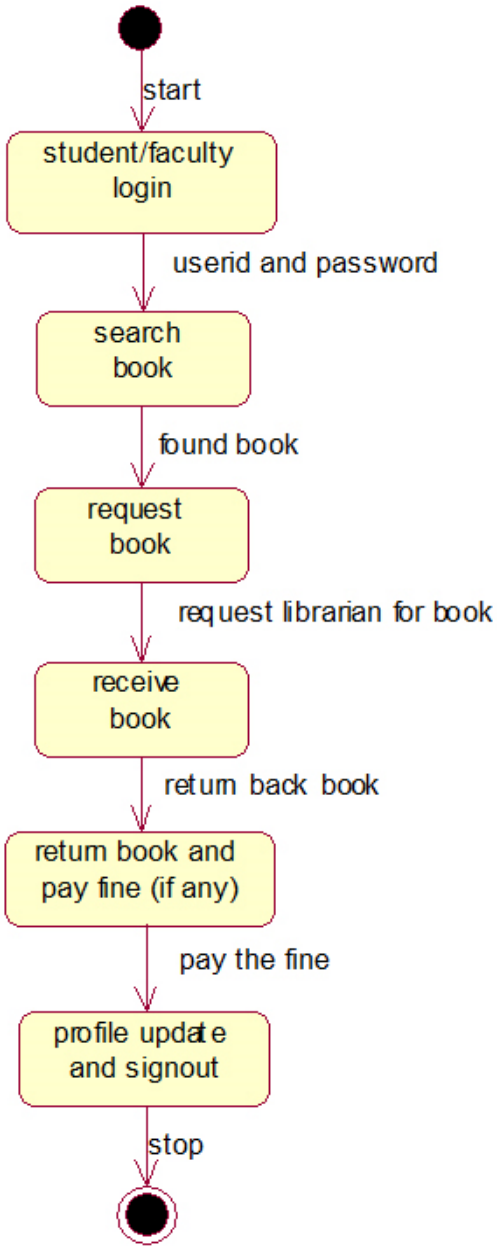
- To model the dynamic aspect of a system.
- To model the lifetime of a reactive system.
- To describe different states of an object during its lifetime.
- Define a state machine to model the states of an object.

➢ Where to Use State chart Diagrams?

The main usage can be described as −

- To model the object states of a system.
- To model the reactive system. Reactive system consists of reactive objects.
- To identify the events responsible for state changes.
- Forward and reverse engineering.

- **Development of State Transition Diagram for Library Information System**


start
student/faculty login
userid and password
search book
found book
request book
request librarian for book
receive book
return back book
return book and pay fine (if any)
pay the fine
profile update and signout
stop

Practical-7

Aim: Design Class & Object Diagrams for software domain problem.

➢ Class Diagram

- Class diagrams are generally used for conceptual modeling of static view of a software application, and for modeling translating models into programming code in a detailed manner. At time of developing or construction software systems, a class diagram is widely used. They are also used for data modeling. It is used to show classes, relationships among them, interface, association, etc. Class in a class diagram simply is a blueprint of an object. It simply describes and explains different type of objects in system, and different types of relationships that exist between them.
- Class Diagram for Library Information System simply describes structure of Library Information System class, attributes, methods or operations, relationship among objects.
- **Classes of Library Information System:**
    i. Library Information System class – It manages all operations of Library Management System. It is central part of organization for which software is being designed.
    ii. User Class – It manages all operations of user.
    iii. Librarian Class – It manages all operations of Librarian.
    iv. Book Class – It manages all operations of books. It is basic building block of system.
    v. Account Class – It manages all operations of account.
    vi. Library database Class – It manages all operations of library database.
    vii. Staff Class – It manages all operations of staff.
    viii. Student Class – It manages all operations of student.

- **Attributes of Library Information System :**

i. Library Information System Attributes – UserType, Username, Password
ii. User Attributes – Name, Id
iii. Librarian Attributes – Name, Id, Password, SearchString
iv. Book Attributes – Title, Author, ISBN, Publication
v. Account Attributes – no_borrowed_books,no_reserved_books,no_returned_books, no_lost_books fine_amount
vi. Library database Attributes – List_of_books
vii. Staff Class Attributes – Dept
viii. Student Class Attributes – Class

- **Methods of Library Information System :**

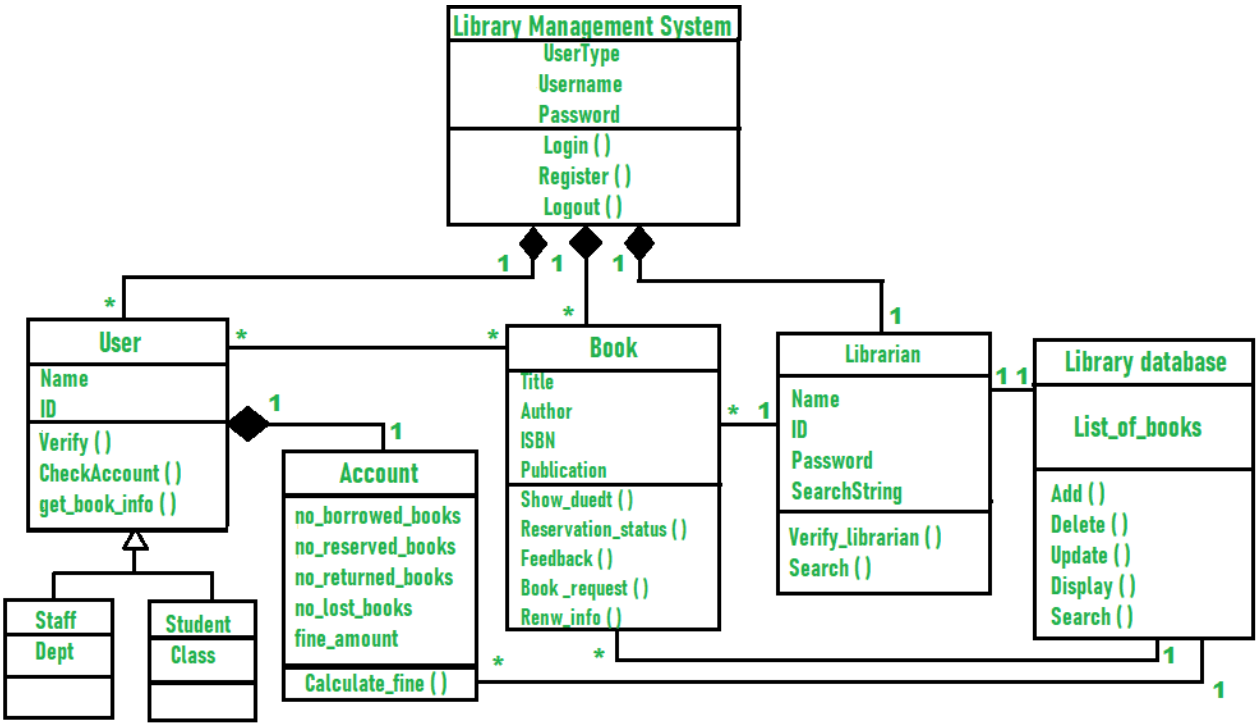
i. Library Information System Methods – Login(), Register(), Logout()
ii. User Methods – Verify(), CheckAccount(), get_book_info()
iii. Librarian Methods – Verify_librarian(), Search()
iv. Book Methods – Show_duedt(),Reservation_status(),Feedback(),Book_request(), Renew_info()
v. Account Methods – Calculate_fine()
vi. Library database Methods – Add(), Delete(), Update(), Display(), Search()

➢ Object Diagram

- Object diagrams are derived from class diagrams, so object diagrams are dependent upon class diagrams.
- Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system, but this static view is a snapshot of the system at a particular moment.
- Object diagrams are used to render a set of objects and their relationships as an instance.

➢ Purpose of Object Diagram

- The purpose of a diagram should be understood clearly to implement it practically. The purposes of object diagrams are similar to class diagrams.
- The difference is that a class diagram represents an abstract model consisting of classes and their relationships. However, an object diagram represents an instance at a particular moment, which is concrete in nature.
- It means the object diagram is closer to the actual system behavior. The purpose is to capture the static view of a system at a particular moment.
- The purpose of the object diagram can be summarized as −
    i. Forward and reverse engineering.
    ii. Object relationships of a system
    iii. Static view of an interaction.
    iv. Understand object behavior and their relationship from practical perspective.


Library Management System
UserType
Username
Password
Login ( )
Register ( )
Logout ( )
User
Name
ID
Verify ( )
CheckAccount ( )
get_book_info ( )
Account
no_borrowed_books
no_reserved_books
no_returned_books
no_lost_books
fine_amount
Calculate_fine ( )
Book
Title
Author
ISBN
Publication
Show_duedt ( )
Reservation_status ( )
Feedback ( )
Book _request ( )
Renw_info ( )
Librarian
Name
ID
Password
SearchString
Verify_librarian ( )
Search ( )
Library database
List_of_books
Add ( )
Delete ( )
Update ( )
Display ( )
Search ( )
Staff
Dept
Student
Class

Practical-8

Aim: Prepare a data dictionary for Software project definition.

➢ Data Dictionary

- A data dictionary contains metadata of the database.

- **What is the metadata of the database?**

  The data about the database is called metadata.

- **Advantages of data dictionary Data Dictionary**

  The data dictionary contains different kinds of information about the database. Some of the information is mentioned below.

  i. Names of database tables
  ii. owners of the table
  iii. User permissions of the table
  iv. security constraints
  v. Schemas of database tables
  vi. when database tables were created etc.
  vii. Information about keys
  viii. such as primary key attributes and foreign key attributes etc.
  ix. Information about the database views having visibility.
  x. Physical information about the storage of the tables

Table Name: ActivityLog				
Description : store all the activity done by user in the system				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
LogId	int	Not null	-	-
LogEmp	nchar(10)	Not null	-	-
LogTime	datetime	Not null	-	-
LogActivity	varchar(MAX)	Not null	-	-

Field	**Example Data**
LogId(PK)	113
LogEmp	M0001
LogTime	2011-02-16 16:35:07.000
LogActivity	Information of PublisherID P0003 has been updated

Table 3.4.1 Data Dictionary for ActivityLog table

Table Name: Admin				
Description : store the information of user who used the library system				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
Admin_ID	nvarchar(50)	Not null	-	-
Admin_Name	nvarchar(50)	Not null	-	-
Admin_Level	nvarchar(50)	Not null	-	Format : 1,0
Password	nvarchar(50)	Not null	-	-
Admin_ic	nvarchar(50)	Not null	-	-
admin_contact	nvarchar(50)	Not null	-	-
admin_email	nvarchar(50)	Not null	-	-
admin_address	nvarchar(MAX)	Not null	-	-

Field	**Example Data**
Admin_ID	M0001
Admin_Name	Tan Chaur Chuan
Admin_Level	1
Password	12341234
Admin_ic	880704-35-5263
admin_contact	04-3984851
admin_email	sdfsdf@hotmail.com
admin_address	30, lintang perai 2, taman chai leng, 34567 pahang.

Table 3.4.2 Data Dictionary for Admin table

Table Name: Book				
Description : store the information of the books				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
ISBN	nvarchar(50)	Not null	-	-
BookTitle	nvarchar(50)	Not null	-	-
Author	nvarchar(50)	Not null	-	-
PublisherID	nvarchar(50)	Not null	-	-
Language	nvarchar(50)	Not null	-	-
Category	nvarchar(50)	Not null	-	-
Description	nvarchar(MAX)	Not null	-	-
BookCover	nvarchar(MAX)	Not null	-	-

Field	**Example Data**
ISBN(PK)	9781587132049
BookTitle	Handphone King
Author	C.Y
PublisherID	P0001
Language	Chinese
Category	Technology
Description	A book which show the latest information of all brand handphone
BookCover	Handphone.JPG

Table 3.4.3 Data Dictionary for Book table

Table Name: BookComment				
Description : to store the comment for particular book				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
CID	Int	Not null	-	-
ISBN	Nvarchar(50)	Not null	-	-
UserID	Nvarchar(max)	Not null	-	-
Comment	Nvarchar(max)	Not null	-	-

Field	**Example Data**
CID	13
ISBN	1234567891234
UserID	M0001
Comment	This book is very interesting…thanks

Table 3.4.4 Data Dictionary for BookComment table

Table Name: BookCopy				
Description : to store the quantities of books and the detail of each book				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
BarcodeID	nvarchar(50)	Not null	-	-
ISBN	nvarchar(50)	Not null	-	-
Status	nvarchar(50)	Not null	-	Format : L, A, N
PurchasePrice	Money	Not null	-	-
purchaseDate	Datetime	Not null	-	-

Field	**Example Data**
barcodeID	978158713204901
ISBN	9781587132049
Status	L
PurchasePrice	200.0000
PurchaseDate	2011-02-16 00:00:00.000

Table 3.4.5 Data Dictionary for BookCopy table

Table Name: News				
Description : to post the latest news at web site's main page				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
ID	Int	Not null	-	-
Date	Date	Not null	-	-
[content]	Nvarchar(50)	Not null	-	-

Field	**Example Data**
ID	9781587132049
Date	2011-01-13
[content]	The popular book 'The Lord of the Ring' is now available !!!

Table 3.4.6 Data Dictionary for News table

Table Name: LibraryDetail				
Description : to store the information of the library				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
libno	Char(10)	Not null	-	-
Libname	varchar(50)	Not null	-	-
Libadd1	varchar(50)	Not null	-	-
Libadd2	varchar(50)	Not null	-	-
Libposcode	Char(5)	Not null	-	-
Libstate	varchar(50)	Not null	-	-
Libtown	varchar(50)	Not null	-	-
Libphone	varchar(50)	Not null	-	-
Libfax	varchar(50)	Not null	-	-
Libemail	varchar(50)	Not null	-	Format : abc@abc.abc
Libweb	varchar(50)	Not null	-	-

Field	**Example Data**
libno	001
Libname	Chen Library
Libadd1	30, lintang perai 5,
Libadd2	Taman putang,
Libposcode	13506
Libstate	Pulau Pinang
Libtown	Butterworth
Libphone	04-3859451
Libfax	04-3225645
Libemail	librarycs@hotmail.com
Libweb	www.google.com

Table 3.4.7 Data Dictionary for LibraryDetail table

Table Name: Reservation				
Description : to record the book reservation for the member				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
UserID	Nvarchar(50)	Not null	-	-
BarCodeId	Nvarchar(50)	Not null	-	-
DateReserve	date	Not null	-	-

Field	**Example Data**
UserID	M0001
BarCodeId	584961352652401

Table 3.4.8 Data Dictionary for Reservation table

Table Name: LostBook				
Description : to keep the information of lost book				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
ID	int	Not null	-	-
Userid	varchar(50)	Not null	-	-
Barcodeid	varchar(50)	Not null	-	-
ISBN	varchar(50)	Not null	-	-
LostDate	Date	Not null	-	-

Field	**Example Data**
ID	3
Userid	M0002
Barcodeid	978158713204901
ISBN	9781587132049
LostDate	2011-02-16

Table 3.4.7 Data Dictionary for LostBook table

Table Name: RetalInfo				
Description : to keep the book transaction detail				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
userID	nvarchar(50)	Not null	-	-
BarCodeID	nvarchar(50)	Not null	-	-
DateReturned	Datetime	-	-	-
DateRented	Datetime	Not null	-	-
DateDue	Datetime	Not null	-	-
Total Fine	money	-	-	-

Field	**Example Data**
userID	M0003
BarCodeID	123456789123401
DateReturned	2011-03-02 00:00:00.000
DateRented	2010-09-09 00:00:00.000
DateDue	2010-10-10 00:00:00.000
Total Fine	200.0000

Table 3.4.7 Data Dictionary for Rental Info table

Table Name: Publisher				
Description : to store the publisher so it is available when register book				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
PublisherId	Nvarchar(50)	Not null	-	-
PublisherName	nvarchar(50)	Not null	-	-

Field	**Example Data**
PublisherId	P0001
PublisherName	Tan Khen Khen

Table 3.4.7 Data Dictionary for Publisher table

Table Name: User				
Description : to store the information of the member				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
UserID	Char(10)	Not null	-	-
Username	varchar(50)	Not null	-	-
Useraddress	varchar(50)	Not null	-	-
UserPhone	varchar(50)	Not null	-	-
UserIC	Char(5)	Not null	-	-
UserRegDate	varchar(50)	Not null	-	-
AvailableBook	varchar(50)	Not null	-	-
Userpass	varchar(50)	Not null	-	-
Userphoto	varchar(50)	Not null	-	-
Useremail	varchar(50)	Not null	-	Format : abc@abc.abc
UserExpiredDate	varchar(50)	Not null	-	-

Field	**Example Data**
UserID	M001
Username	Ooi Yee Neng
Useraddress	30 lintang talang 2, taman perai. 13600 Perai, Penang.
UserPhone	04-3568956
UserIC	880407-35-5266
UserRegDate	2011-02-16 00:00:00.000
AvailableBook	4
Userpass	12345678
Userphoto	Tan_chen_khen.JPG
Useremail	mrtan@hotmail.com
UserExpiredDate	2012-03-05 00:00:00.000

Table 3.4.7 Data Dictionary for User table

Practical-9

Aim: Draw the Deployment Diagram for the project definition.

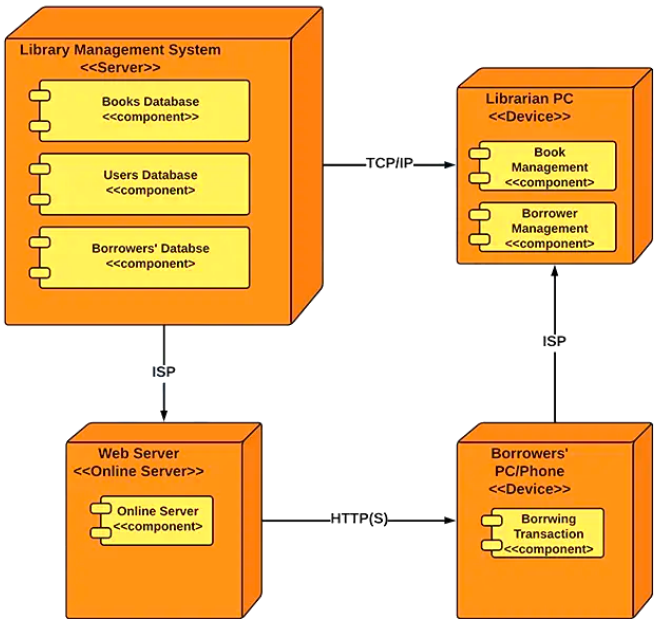
➢ Deployment Diagram

- The Deployment Diagram for Library Information System is the specification of software and hardware used to deploy the library system. It contains the details and design which will guide users on how the system works. It consists of nodes such as the software itself, the users' devices, and their connections.
- A deployment diagram is a form of UML model used to describe the execution architecture of the Library System. It contains elements such as hardware, software, and the middleware that connects them. UML Deployment Diagram presents the system's physical hardware and software.

➢ Benefits of UML Deployment Diagram

They visualize a system's hardware processors/nodes/devices, communication linkages between them, and software file layout on that hardware.

- It aids in the visualization of the various aspects involved.
- Aids in a more accurate description of all the hardware elements used by software components.
- It clarifies the description of the runtime involved in processing nodes.
- Provides hardware specified details for a distributed application.
- Helps in modeling the system's hardware topology.
- It aids in the modeling of inserted or included software.
- Provides more information on the hardware system.
- Reverse engineering



DEPLOYMENT DIAGRAM

➢ Elements used in Creating your Deployment Diagram

Here are some elements used in creating your deployment diagram by Lucid chart. Deployment diagrams come in a number of shapes. Most of these things are shown in the graphic below, and this list gives you a general idea of what you might see.

i. Artifact: A rectangle with the name and term “artifact” enclosed by two arrows represents a software-created product.
ii. Association: A message or other sort of communication between nodes is indicated by a line.
iii. Component: A rectangle with two tabs that indicate a software part is called a component.
iv. Dependency: A dashed line that ends in an arrow denotes the dependency of one node or component on another.

v. Interface: A contract relationship is indicated by a circle. Those items that realize the interface are required to fulfill some sort of task.

vi. Node: A three-dimensional box represents a hardware or software object.

vii. Node as Container: This is a node that has another node within it, such as the nodes that contain components.

viii. Stereotype: A device housed within the node, displayed at the top of the node and flanked by two arrows.

Practical-10

Aim: Design the Test Cases for software domain problem.

➢ Test Cases

This page has Test Cases For Library Management System. Library is the place with the huge collection of books. It is place from where the students and the faculties issue the books for their reference purposes. But the maintenance of keeping the records of issuing and borrowing is difficult if you use a normal book as a registry. To make this task easier, the library management system will be very useful. The test cases for library management system is an application that explains the test cases for library management system. Software testing is a critical part that is involved in the overall development of the application. This will be one of the interesting projects that one can work on and implement in real time world.

- **Adding Login page**

Serial No.	Condition To be Tested	Test Data	Expected Output	Remarks
1.	If we are not enter the valid roll no or LOGIN ID.	LOGIN ID or Roll no	Please fill out this field.	SUCCESSFUL
2.	If we only put the valid roll no or LOGIN ID and after pressing the login button.	Your password	Please fill out this field.	SUCCESSFUL
3.	If we are putting the correct user name and wrong password and after selecting the valid login credential.	Select the user type	Kindly select the status	SUCCESSFUL

- **Adding Registration form:**

Serial No.	Condition To be Tested	Test Data	Expected Output	Remarks
1	If we are not selected any field.	Select Course	Kindly select the fields.	SUCCESSFUL
2	If we are not entered the name.	Name	Please enter the name.	SUCCESSFUL
3	If we are not entered the valid roll no.	Roll no	Please enter the valid roll no.	SUCCESSFUL
4	If we are not entered the valid password.	Enter Password	Please enter the valid password.	SUCCESSFUL
5	If the password is not same as previous password.	Confirm password	Please enter the valid password once again.	SUCCESSFUL
6	If we are not entered the valid emailid.	Email id	Kindly insert the emailid.	SUCCESSFUL
7	If we are not suppose to enter contact no.	Contact no.	Please enter the contact no.	SUCCESSFUL
8	If we are not suppose to add any image files on the choose File.	Choose file	Kindly upload Image.	SUCCESSFUL

- **Adding admin/branch**

Serial No.	Condition To be Tested	Test Data	Expected Output	Remarks
1.	If we are not enter the branch name.	Branch name	Please fill out this field.	SUCCESSFUL

2.	If the branch description not to entered.	Branch description	Please fill out this field.	SUCCESSFUL
3.	If we not selected any status	Branch status	Kindly select the status	SUCCESSFUL

- **Admin/book:**

Serial No.	Condition To be Tested	Test Data	Expected Output	Remarks
1.	If you are not selected category.	Book category	Please fill out this field.	SUCCESSFUL
2.	If we are not enter the book name.	Book name	Please enter book name.	SUCCESSFUL
3	If we are not enter the book cost fields.	Book cost	Please enter book cost.	SUCCESSFUL
4	If we are not enter book author fields.	Book author	Please enter book author.	SUCCESSFUL
5	If we not selected any status.	Book status	Kindly select status.	SUCCESSFUL
6	If we are not suppose to add any image files on the choose File.	Book Image	Kindly upload Image.	SUCCESSFUL
7	If we are not enter book description.	Book description	Book description should not be empty.	SUCCESSFUL

- **Admin/Book category:**

Serial No.	Condition To be Tested	Test Data	Expected Output	Remarks
1.	If we are not enter the valid book category.	Book category	Please fill out this fields.	SUCCESSFUL
2.	If we are not suppose the book description.	Book description	Book description should not be empty..	SUCCESSFUL
3.	If we are not selected status.	Book status	Kindly select status	SUCCESSFUL

- **Admin/Course:**

Serial No.	Condition To be Tested	Test Data	Expected Output	Remarks
1.	If you are not select the branch	Select branch.	Please enter the valid fields.	SUCCESSFUL
2.	If the course name not entered.	Course	Please enter the course.	SUCCESSFUL
3.	If the course note did not entered.	Course note	Please You should fill out course note.	SUCCESSFUL
4.	If we are not selected status.	Course status	Kindly select the status.	SUCCESSFUL

- **Admin/Librarian:**

Serial No.	Condition To be Tested	Test Data	Expected Output	Remarks
1	If we are not entered the name.	Name	Please enter the name.	SUCCESSFUL
2	If we are not selected the type.	Type	Please select the type.	SUCCESSFUL
3	If we are not entered the valid login id	Login id	Enter the valid Login id.	SUCCESSFUL
4	If we are not entered the password field.	Password	Enter the valid password.	SUCCESSFUL
5	If we are not entered the same password.	Confirm password	Enter the valid password once again.	SUCCESSFUL
6	If we are not selected any status.	Status	Kindly select the status	SUCCESSFUL

- **Admin/Student:**

Serial No.	Condition To be Tested	Test Data	Expected Output	Remarks
1	If we are not selected any course.	Course	Please fill out the fields.	SUCCESSFUL
2	If we are not entered the student name.	Student name	Please enter the student name.	SUCCESSFUL
3	If we are not entered the valid roll no.	Roll no	Please enter the valid roll no.	SUCCESSFUL
4	If we are not entered the valid password.	Password	Please enter the valid password.	SUCCESSFUL

5	If the password is not same as previous password.	Confirm password	Please enter the valid password once again.	SUCCESSFUL
6	If we are not entered the valid emailid.	Email id	Kindly insert the emailid.	SUCCESSFUL
7	If we are not suppose to enter contact no.	Contact no.	Please enter the contact no.	SUCCESSFUL
8	If we are not selected status	Status	kindly select status.	SUCCESSFUL
9	If we are not suppose to choose the image file.	Student image	Kindly upload the image.	SUCCESSFUL

Practical-11

Aim: Generate SRS Document as per given format.

1. 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.

1.1 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.

1.2 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 ILM 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.

1.3 Definition, Acronyms, Abbreviation:

- JAVA -> platform independence
- SQL -> Structured query Language
- DFD -> Data Flow Diagram
- CFD -> Context Flow Diagram
- ER -> Entity Relationship
- IDE -> Integrated Development Environment
- SRS -> Software Requirement Specification

2. OVERALL DESCRIPTION

2.1 PRODUCT PRESPECTIVE

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.

2.2 SOFTWARE REQUIREMENT

- **Front end:**
- Android developer tool
- Advance java
- **Back end:**
- MySQL

2.3 HARDWARE REQUIREMENT

- Android version 2.3 ginger bread(minimum, android user's)
- 2GB ram
- 1.2 GHz processor
- Intel i5
- Windows 7/8/8.1/10

2.4.1 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 displaced 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.

2.4.1 Non-Functional Requirements

- **Usability Requirement**

  The system shall allow the users to access the system from the phone using android application. The system uses a android application as an interface. Since all users are familiar with the general usage of mobile app, no special training is required. The system is user friendly which makes the system easy.

- **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.

2.5 USER CHARACTERSTICS

We have 3 levels of users :

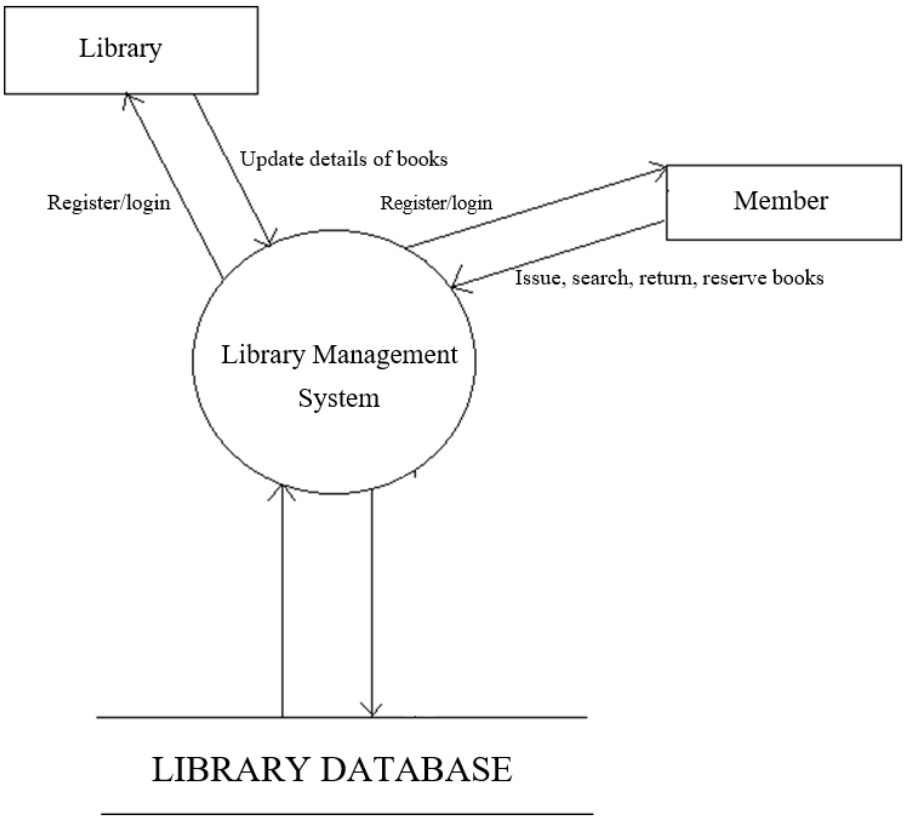
- User module: In the user module, user will check the availability of the books.
  - Issue book
  - Reserve book
  - Return book
  - Fine details
- Library module:
  - Add new book
  - Remove books
  - Update details of book
- Administration module:

  The following are the sub module in the administration module :
  - Register user
  - Entry book details
  - Book issue

2.6 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.

2.7 FLOW DIAGRAM

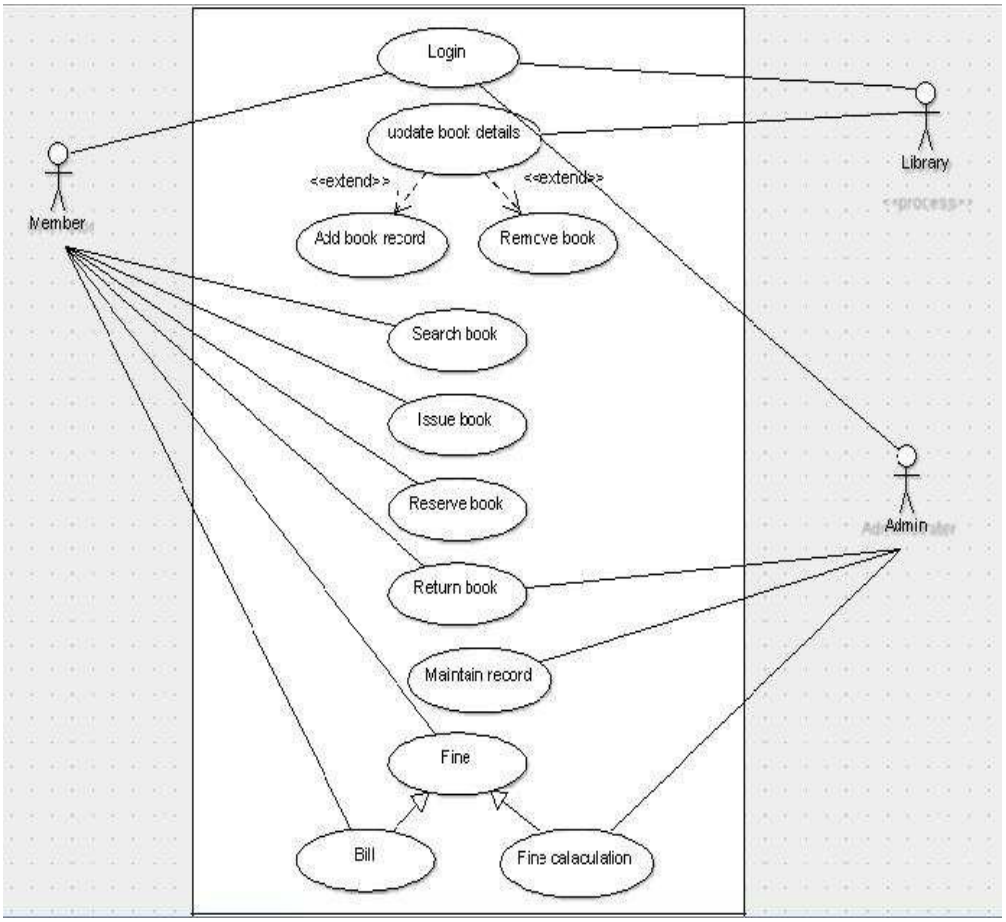


2.8 USE CASE MODEL DESCRIPTION:

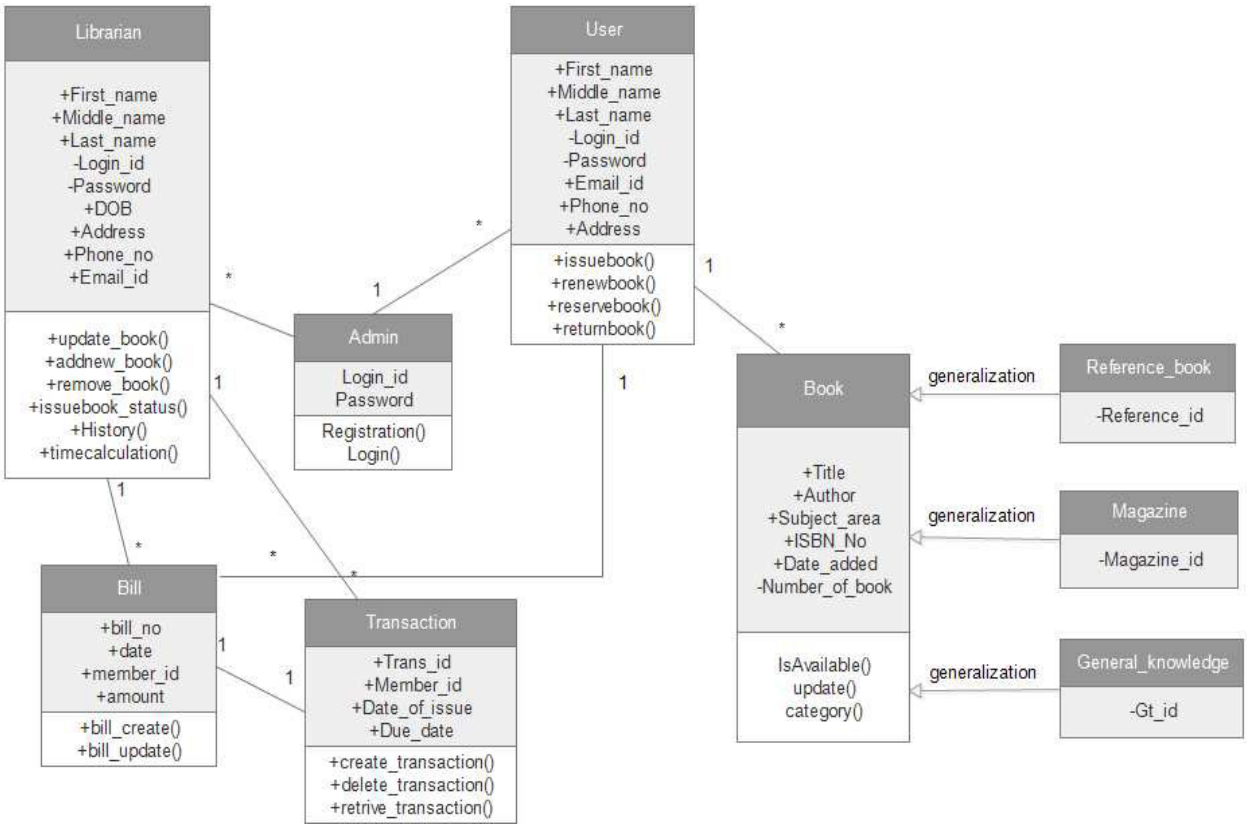
Use Case Selection	Description
Use Case name	Add student and library record
Level	Sub-Functional level
Primary actor	Student, Library
Stakeholders and interest	Student: wants to register into the system. Library: wants to register into the system and update book details. Administrator: responsible for the management of the transaction of fine and also login and register details.
Pre-condition	Students and Library have submitted their registration form.
Post-condition	Record for a student/library has been added.
Main success scenario	1 Student/Library opens the application to access the services of the LMS 2 Student/Library sign-up to get registered online. 3 He/She provides correct information and secret password. 4 He/She got registered.
Alternative flow	1 Student/Library opens the application in their phone 2 He/She tries to sign-up 3 He/She fails and receives an error 4 He/She will report an error and the error will be rectified as soon as possible.
Specific requirement	• The response time for registration is 1 minute. • The response time for login is 1 minute

Table 1: table for use case description.

2.9.1 USE CASE DIAGRAM



2.9.2 CLASS DIAGRAM



2.9.3 SEQUENTIAL DIAGRAM

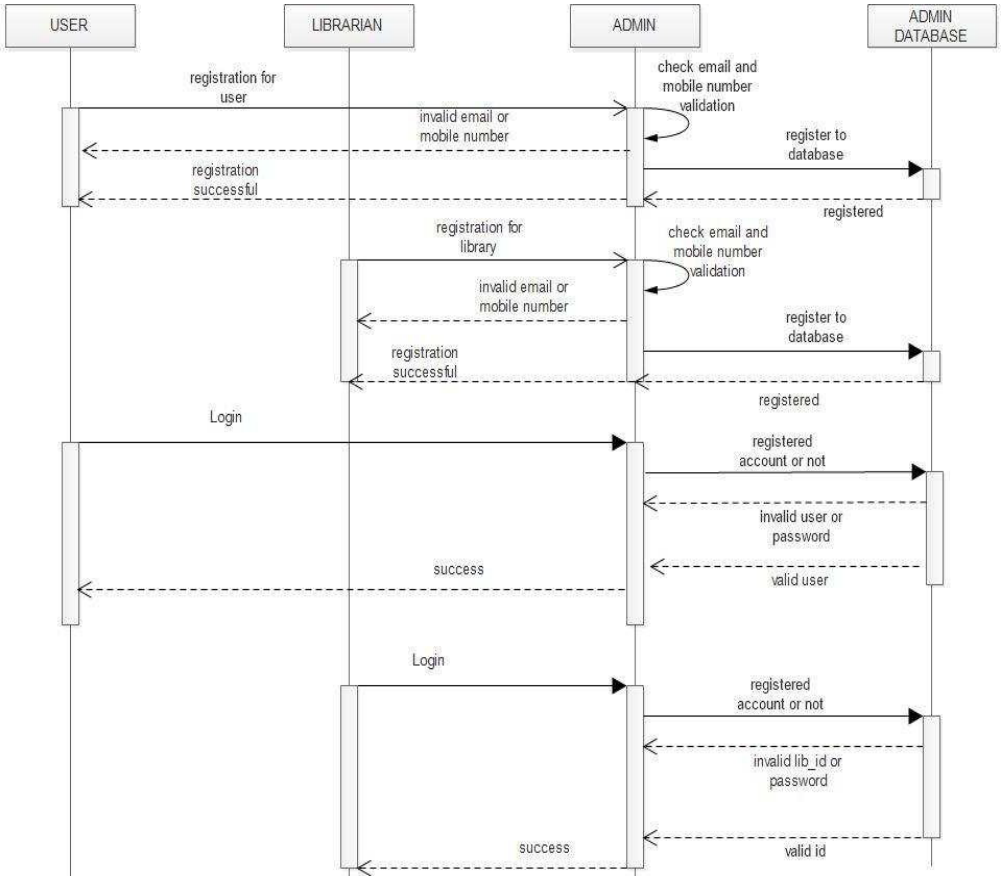
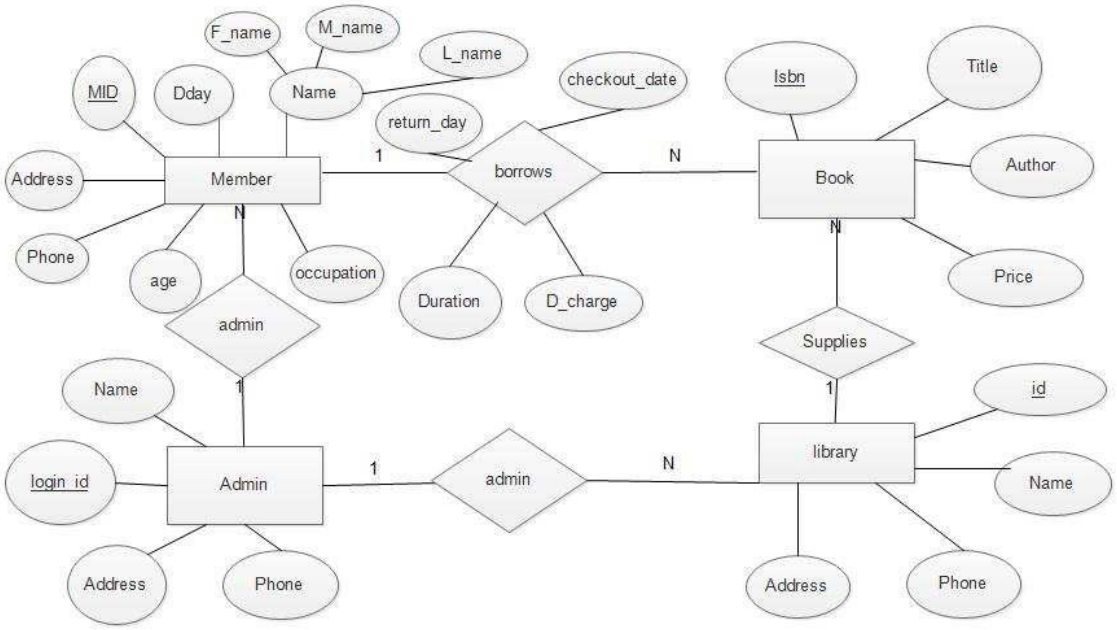


Fig 4(a). Register and login Sequence Diagram

2.10 ER diagram



2.11 Assumptions and Dependencies

The product needs following third party applications for the development of the project:

- Android Studio (for development of android based applications)
- Netbeans
- Photoshop (for editing layouts, icons, buttons, etc)

Practical-12

Aim: Why Agile process models with DevOps are recommended in big companies like Infosys. Justify with one of the case studies (You can explore it with any tool like JIRA). Write a report of 3-5 pages on case study.

Abstract:

The combined adoption of Agile and DevOps enables organizations to cope with the increasing complexity of managing customer requirements and requests. It fosters the emergence of a more collaborative and Agile framework to replace the waterfall models applied to software development flow and the separation of development teams from operations. This study aims to explore the benefits of the combined adoption of both models. A qualitative methodology is adopted by including twelve case studies from international software engineering companies. Thematic analysis is employed in identifying the benefits of the combined adoption of both paradigms. The findings reveal the existence of twelve benefits, highlighting the automation of processes, improved communication between teams, and reduction in time to market through process integration and shorter software delivery cycles. Although they address different goals and challenges, the Agile and DevOps paradigms when properly combined and aligned can offer relevant benefits to organizations. The novelty of this study lies in the systematization of the benefits of the combined adoption of Agile and DevOps considering multiple perspectives of the software engineering business environment.

1. Introduction

The software development process can be viewed as a set of tasks required to produce high-quality software. The literature shows that the quality of the software produced reflects the way the process was carried out [1–3]. Although several software development processes exist, generic activities common to all of them stand out, as Sommerville [4] high- lights, such as software specification (e.g., requirements definition, software constraints), software development (e.g., software design and implementation), software validation (e.g., software must be validated to ensure that the implemented functionality conforms to what was specified), and software evolution (e.g., software evolves as per customer need). The software development process provides an interaction between users and software engineers, between users and technology, and between system engineers and technology. In this sense, software development is an interactive learning process, and the result is an embodiment of knowledge gathered, transformed, and organized as the process is conducted.

A software development methodology includes a set of activities that assist in the production of software. These activities result in a product that demonstrates how the development process was conducted. Agile methodologies arise from the need to over- come the difficulties and disadvantages of applying traditional methodologies in project management and implementation. The Agile methodology assumes short periods of time between each delivery to ensure early and continuous delivery of software susceptible to evaluation [5]. In [6] it is also recognized that software implementation according to this paradigm is interactive and incremental, enabling early confirmation of whether or not the delivered artifact meets the needs and making the respective corrections with low risk and cost.

2. Literature Review

2.1. DevOps Concepts and Model

In 2009, Paul Hammond and John Allspaw presented the talk "10+ Deploys Per Day: Dev and Ops Cooperation at Flickr" [20]. They explained how the developers' teams (Dev) and operations teams (Ops) could contribute to more agile and scalable software development. Tight integration between Dev and Ops to safely achieve several software deployments (more than 10) in a single day was a disruptive idea regarding software development and its evolution. Later, Patrick Debois coined the term DevOps (Development and Operations) and created the DevOps Day event [21]. Although the DevOps movement has been discussed for more than a decade [13–15,22] it still lacks a unique formal definition. For Wiedemann et al. [23], the lack of a unique definition could be intentional to allow each team to choose the definition that better suits its needs. Nevertheless, several authors proposed definitions such as the one by Leite et al. [13]— "DevOps is a collaborative and multidisciplinary effort within an organization to automate continuous delivery of new software versions while guaranteeing their correctness and reliability"—and others view it as a combination of values, principles, methods, practices, and tools [24]. Some other common definitions can be found in [23].

One of the key points in the execution of a project is the approach used to manage it. The traditional approach based on the waterfall model looks at the project in a linear way with several events, while in the iterative approach the development of software is undertaken through successive progress [25]. Therefore, it is common that the system is presented still incomplete or with some deficient parts. The objective is that the refine- ment of the product happens in stages until the desired result is achieved. The software development process does not end with the release of the code, but only when it closes the feedback loop between those who write the code and those who use it. DevOps aims to remove the barriers between traditionally independent teams: development and operations. Under the DevOps approach, these teams should work together across the entire software life cycle, from development and testing through deployment to operations. More than only a technical subject, DevOps deals with the organizational and

human issues that arise in the software life cycle. It promotes a culture of collaboration, integration, and communication between teams to reduce the disconnect between them while assuring the delivery of software in an agile, safe, and stable way. According to Rajapakse et al. [14], the DevOps movement is currently a widely adopted software development approach having as a major benefit the ability to deploy releases more frequently and at a higher rate.

Some related concepts used in conjunction with DevOps are Continuous Integration, Continuous Delivery, and Continuous Deployment. As noted in [14], these concepts are considered key practices within DevOps, but are not always clearly used, as stressed by Stahl et al. [24]. Continuous Integration is a development practice where developers frequently integrate the code they produce, that has successfully passed testing, to the project under development [24]. Those integrations occur typically once a day. Continuous Delivery is a development practice where the software is kept in a reliable deployable state at any time [14]. Potentially, after every change, the software can be released. This leads to several release candidates that are evaluated. The deployment to production is made manually by a team member, with the appropriate authority, who decides when and which candidate should be released [14]. Apart from Continuous Delivery, in Continuous Deploy- ment, release candidates resulting from software changes are automatically deployed to production without the intervention of any team member [14,24].

2.2. Similarities and Differences between DevOps and Agile

In the context of Software Engineering, as discussed in this paper, DevOps can be understood as a behavioral evolution of Agile development [27], which was gradually conceived through practical experiences of implementation in software development. However, it is important to point out that the Agile method has its focus directed specifically to software development [28], while DevOps aims to involve the software development area in the implementation and operation of the software developed or still under development, which shows us that DevOps processes are being implemented within the Agile processes. Currently, in the professional community of

Information and Communication Technologies (ICT) there is a growing consensus, in practice, that DevOps can be understood as "DevOps = Agile + Lean + IT service management (ITSM)" [29]. In its method and processes, DevOps adopts characteristics of frameworks related to the technical area of Agile software development together with ICT management processes. Complementarily, other relevant methodologies (e.g., Extreme Programming, Dynamic Systems Development Method, Kanban, SCRUM) have approaches intrinsically related to the Agile philoso- phy [30–33]. SCRUM is very well-known and intensely used in the Agile method, and is generally enhanced by the Kanban tool, for managing the workflow of software development, but which also fits very well into the DevOps development process itself [34]. Table 1 presents the problems or gaps in the Agile method that are solved by adopting the DevOps method.

Table 1. Problems with Agile development and DevOps solution (adapted from [35]).

Problem with Agile Development	DevOps Solution
Delivery of new features to the customer is often delayed.	DevOps tools are used to test and release new features as they are completed.
Completed software components are not compatible with each other.	Open interfaces and test automation make it possible to divide development into independent yet compatible parts.
Quality of the product is not ensured properly prior to release.	DevOps tools and practices help automating quality assurance and reduce the need for repetitive manual work.
New features break old functions.	The quality of existing functions is ensured quickly and automatically after each change.
Budget goals and deadlines are missed.	The tools and procedures of DevOps increase the transparency and predictability of the development work.
Developer teams and IT operations crews are not cooperating.	Developer teams and IT operations crews agree upon responsibilities together. Their goals are unified.

3. Methodology

This study adopts a qualitative methodology to explore the benefits of the combined adoption of DevOps and Agile. This type of methodology is used in the context of social sciences and engineering and, according to Merriam and Tisdell [42], is concerned with a level of reality that cannot be quantified, exploring meanings, aspirations, beliefs, values, and attitudes that correspond to a deeper space of relationships, processes, and phenomena, and cannot be reduced for operationalization of variables. Dyba et al. [43] recognize that the software industry presents lines of research that are not only limited to exploring technical software engineering issues, but also need to look at the challenges of the intersection between technical and non-technical aspects. In this sense, adopting a purely quantitative approach is insufficient. Moreover, phenomena addressed in the field of project develop- ment in a DevOps and Agile paradigm are complex and unique. Therefore, the qualitative approach adopted in this study allows for exploring and understanding the relationships and activities performed by organizations in their software development activity.

In the scope of this study, twelve case studies were incorporated that report the simultaneous inclusion of DevOps and Agile methodologies in their software development teams. These cases correspond to reports and press releases from commercial vendors of reference in this area. The data come from secondary sources and the authors have not made any changes to the press releases that represent the view of each manufacturer. No summarization of the press releases has been made. It cannot be guaranteed that there is no risk of bias since the identified benefits come from press releases from commercial companies in the area and that have commercial goals in the market. However, to minimize this risk, external and internal validity mechanisms were used. To ensure external validity, multiple case studies were used of companies in different geographic areas, and to ensure internal validity, the same identifier was used to associate similar benefits between the

case studies.

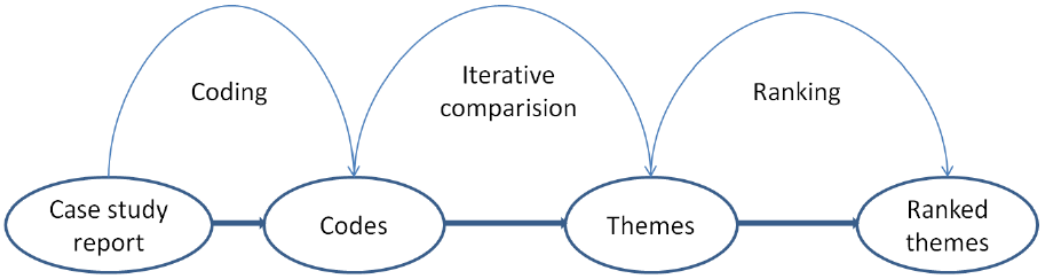


Figure 1. Thematic analysis process (authors own illustration).

4. Results

The twelve case study reports associated with each company presented in Table 2 were thoroughly read and each identified benefit was assigned a unique identifier. Each theme is identified by the acronym "BF" and a number is associated to differentiate each benefit. Common themes have the same acronym. A brief description of how each benefit is understood in the case studies is also included. Table 3 presents the identification process of the themes associated with each case study. Table 3 shows the themes for all the case studies mentioned in Table 2. We highlight the existence of themes that are common to several case studies. Although the themes are common, the vision of each case study in relation to them has some relevant oscillations, which indicate a complementary vision of the institutions present in the case studies. For example, in CS1 time to market emerges due to increased collaboration between teams, whereas in CS8 process integration is highlighted. Something similar emerges in relation to cost. Cost reduction is understood in CS6 as achieved from the existence of multi-skilled human resources, whereas in CS8 cost reduction is motivated by the effects of increased team performance.

Table 4. Comparative analysis of benefits and ranking.

Benefit	CS1	CS2	CS3	CS4	CS5	CS6	CS7	CS8	CS9	CS10	CS11	CS12	Ranking
BF1	X		X			X		X					#3
BF2	X	X			X		X		X	X	X	X	#1
BF3		X			X		X		X	X			#2
BF4		X		X	X								#4
BF5			X		X								#5
BF6				X				X					#5
BF7				X									#10
BF8				X									#10
BF9						X		X					#5
BF10							X				X		#5
BF11							X						#10
BF12									X			X	#5

5. Discussion

Although Agile and DevOps are widespread and different concepts, they can be combined and offer relevant benefits to organizations. As reported in [45], companies have problems in the process of implementing and releasing new software versions because most of the time this is a process performed manually. In addition, this approach leads to a high quantity and frequency of errors [46]. To reduce the incidence of problems and increase flexibility and automation, non-operational resources can be used and in environments that are not in production. The combined adoption of Agile and DevOps allows the developer to gain greater control over the environment, infrastructure, and applications.

The seamless integration between Agile and DevOps generates a more collaborative and Agile framework. This approach leads to a simplification and automation of model processes to make them more rational and efficient. A classic example of this benefit is given by Fabro [47] when highlighting the reduction in delivery cycles, endowing small development packages with a previously unrecognized value. Hemon-Hildgen et al. [48] also highlight the role of orchestration, which consists of automating tasks to optimize the process and reduce repetitive steps that add little to the development cycle. Finally, automated testing along the Agile and DevOps chain allows the reuse of tests between environments and makes them more sustainable [49].

6. Conclusions

This study demonstrates that the Agile and DevOps paradigms are not incompatible but can bring benefits to organizations when properly aligned. While Agile brought about a fast delivery model aligned with customer expectations, DevOps optimized this system. In this sense, an alternative that usually gives great results is the adoption of both methodologies. They not only complement each other but also help companies to face changes in a team.

Changing the strategy and methodology of a team can be a delicate process full of obstacles. Therefore, organizations must address this challenge in a cross-cutting way within the organization to avoid isolated silos that do not contribute to collaborative work. Agile creates a space for more agile work with partial deliveries, while DevOps creates an environment conducive to managing these processes, with effective communication.

This study offers both theoretical and practical relevant contributions. In the theoretical dimension, this study has enabled the identification of a set of benefits of the combined adoption of both paradigms through the adoption of multiple case studies of software companies in the international market. The study identifies a total of 12 benefits and allows us to explore the relative relevance of each of them. From a practical perspective, the benefits identified are relevant to companies that, having adopted Agile and DevOps alone, have not yet taken steps towards the combined adoption of both models. The findings of the study made it evident that the two models are not incompatible, but when combined they can amplify their impacts on organizations.

Limitations and Future Research Directions

This study presents some limitations. Firstly, the case studies included in this study come from secondary sources, which does not allow us to deepen the knowledge on the themes with the use of interviews that may evidence the application of both paradigms. Furthermore, the case studies come from companies with commercial purposes, which may not give a totally unbiased view of the benefits to the organizations or represent very specific groups of the population. Nevertheless, this study adopted external and internal validation mechanisms to reduce this risk of bias. As future work it is recommended that the business view be complemented with a scientific view of the benefits of combining DevOps and Agile and, to this end, a systematic review of the literature in the field can be conducted. Moreover, the qualitative approach used does not allow us to systematically identify all the advantages and make a rigorous quantification of these benefits. As future work, a quantitative study based on a large dataset is suggested to identify the advantages of the combined adoption of both paradigms considering also different sectors of activity of the organizations. It would also be relevant to consider the degree of maturity in the implementation of Agile and DevOps in these organizations and, thus, explore its relevance in the benefits found, since it is expected that some of the benefits may be more easily achieved by organizations with lower levels of maturity in these processes.