3.5 Use Case Approach

The use case approach is primarily designed for object-oriented systems. However, the same may also be used for traditional systems to represent and analyse requirements and for modelling them in a systematic way. For many years, requirement writers used to write stories to specify the expected behaviour of the proposed software system and its interactions with the external world. Jacobson et al. (1999) formalized this story-writing approach into a more systematic and disciplined use case approach. They designed a UML for the software development of object-oriented systems. The use cases address only the functional requirements which explain the expectations of users sitting outside the system. Functional requirements express "what do we expect from the system" without bothering about "how it will be implemented". The use cases capture the expectations in terms of achieving goals and interactions of the users with the system. Many persons are not able to differentiate amongst the use case, use case scenario and use case diagram, and use these terms interchangeably. However, all the three terms are different. Use cases are structured outlines or templates for the description of requirements, written in a natural language like English. A use case scenario is an instance of a use case. It represents a path through a use case. Use case diagrams are graphical representations that may be decomposed into further level of abstraction.

3.5.1 Use Cases and Actors

Use cases and actors are used to define the scope of the system we are planning to build. Use cases incorporate anything that is within the system, but and actors include anything that is external to the system. We identify actors and use cases for the system under development. To do this, a proper understanding of the system is essential. A use case is a description of a system in terms of a sequence of actions.

A use case is initiated by an actor with a specific purpose in mind and completes successfully when that purpose is achieved. The use case describes the sequence of interactions between actors and the system to fulfil the desired purpose. It is nothing but a high-level piece of functionality that the system will provide. It includes a main sequence which a system will follow when an actor interacts with the system. It also includes possible alternative sequences which may occur depending on the interaction and input conditions. Fournier (2009) has rightly given his views about use cases as:

> *The real value of a use case is the dynamic relationship between the actor and the system. A well written use case clarifies how a system is used by the actor for a given goal or reason. If there are any questions about what a system does to provide some specific value to someone or something outside the system, including conditional behaviour and handling conditions of when something goes wrong, the use case is the place to find the answer.*

A use case describes who (any user) does what (interaction) with the system, for what goal, without considering the internal details of the system. The use cases are written in a simple language which should be understandable to every stakeholder. There is no standard outline for a use case. However, a few templates are available in the literature. Requirements are captured in a systematic way and discipline the requirement writers to follow a specified format.

An actor is someone or something that interacts with the system and lies outside the system. The actor may be users of the system, customers, persons giving information to the system or any external system providing data. All stakeholders on the customer's site may be treated as actors if they wish to interact with the system. Hence, everything that is outside the system and wishes to interact with the system is known as an actor.

3.5.2 Identification of Actors

An actor interacts with the system with a particular purpose. The actor may be a person or external system. We should not confuse the actors with devices they use. Devices may help the actors to interact with the system and they are not the actors themselves. We use computers with the help of a keyboard, but the keyboard is not a user (actor), we are the user. Bittner and Spence (2003) have explained the concept as:

> *The purpose of devices is to support some required behaviors of the system, but devices do not define the requirements of the system. Often systems must produce a printed report of information that it contains. We may want to show printer as an actor that then forwards the report to the real actor. This is not correct, printer is not an actor, it is just a mechanism for conveying information.*

We consider the LMS and identify the following actors:

(i) Administrator
(ii) Data entry operator
(iii) Librarian
(iv) Library staff
(v) Faculty
(vi) Student
(vii) Employee

There are two types of actors—primary actors and secondary actors. Primary actors are the persons who will use the system. Customers are the primary actors for whom the system is built. Secondary actors are the persons who will maintain or monitor the system. For example, administrator is a secondary actor.

The identification of actors with their specified roles may define the scope of the system for every actor and expected actions. An actor may interact with one or more use cases depending on its defined role in the system.

3.5.3 Identification of Use Cases

Use cases describe the functionality of the system. To achieve that functionality, actors interact with the system with a defined purpose. An actor acts from the outside and provides some input(s) to the system. One or more output may be generated by the system after such interaction of actors. From the IRD, we create use cases for the system. Some guidelines for the creation of use cases are given as:

(i) Every specified functionality should have a use case.
(ii) A unique name is assigned to a use case. The name should be meaningful and should be able to indicate the purpose of a use case.
(iii) One or more actors may interact with a use case.
(iv) The use case should describe the sequence of actions.
(v) The use case is initiated by an actor with a specified purpose.
(vi) Role of actors must be defined clearly.
(vii) The use case should represent a complete and meaningful flow of events.

As explained earlier, we should always remember that use cases describe who (actor) does what (interaction) with the system, for what goal, without considering the internal details of the system.

In the LMS, we may identify the following use cases from the IRD:

- Maintain book details
- Maintain student details
- Maintain faculty/employee details
- Maintain login details
- Issue book
- Return book
- Fine calculation
- Reserve book
- Query book
- Search book
- Report generation

3.5.4 Defining Relationships between Use Cases

The use case diagram may model two kinds of relationships, namely, extend and include.

Extend Relationship

Extend relationship is used to model the occurrence of some optional, alternative or special part of the use case. This relationship is used to extend the functionality of the original use cases. The new use case is inserted into the original use case. The procedure when the extended use case is inserted into the original is as follows:

1. The original use case executes in usual manner to the point where the new use case has to be inserted.
2. At this point, the new use case is inserted and executed.
3. After the inserted use case completes, the original use case resumes again.

For example, in the LMS, the concept of extend can be observed when a student returns the book after the due date. The fine is calculated for late return of the book. We can represent fine calculation use case as a new use case that extends return book use case. The fine calculation use case is called whenever the book is returned after the due date by the student. The example of extend relationship for the LMS is shown in Figure 3.4.

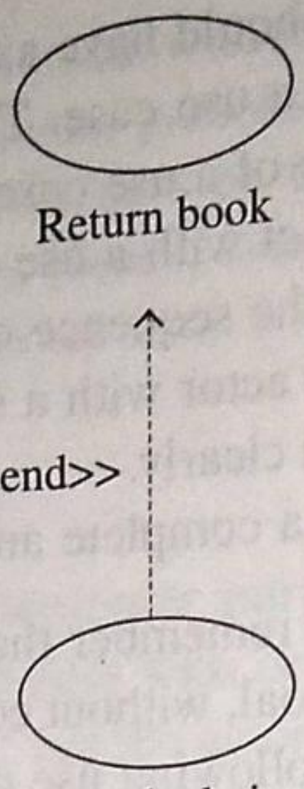


Figure 3.4 Extend relationship.

Include Relationship

The redundant and repeated functionalities amongst use cases can be modelled into include relationship. Thus, the redundancies can be grouped into a single use case. In order to use include relationship, there must be common text in two or more use cases. As in the case of extend relationship, the new use case is inserted into the original use case. The procedure when the included use case is inserted into the original is the same as given in the extend relationship. For example, in employee management system, print use case is required by three use cases—salary generation, appraisal and generate reports (see Figure 3.5).

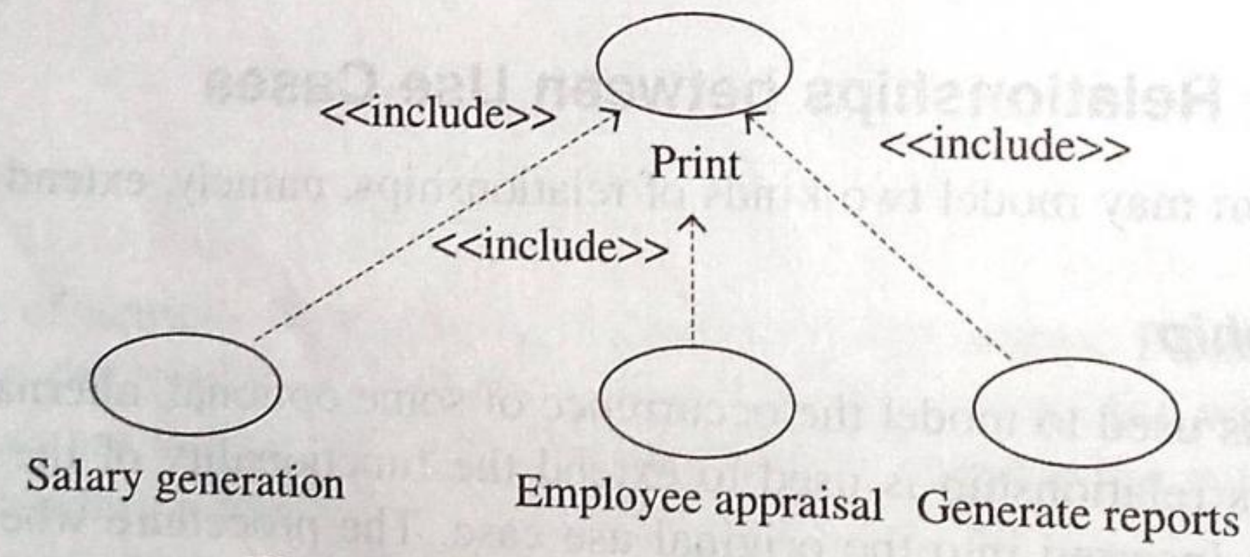


Figure 3.5 Include relationship.

3.5.5 Use Case Diagram

Use case diagram represents the top view of the system. The use cases reside within the system and the actors act from outside the system. The use case diagram is also used to present functionality of the system, but for proper explanation, it should read along with use cases. The use case diagram may also be decomposed into further level of abstraction. It also shows the relationship of use cases and actors. It also explains what happens when an actor interacts with the system. The components of the use case diagram are given in Figure 3.6.

An actor is represented by a stick figure (even non-human one) and a use case by an oval labelled with the name of the use case. The relationship between an actor and a use case is shown by an arrow. For small systems, one use case diagram may be sufficient to represent the

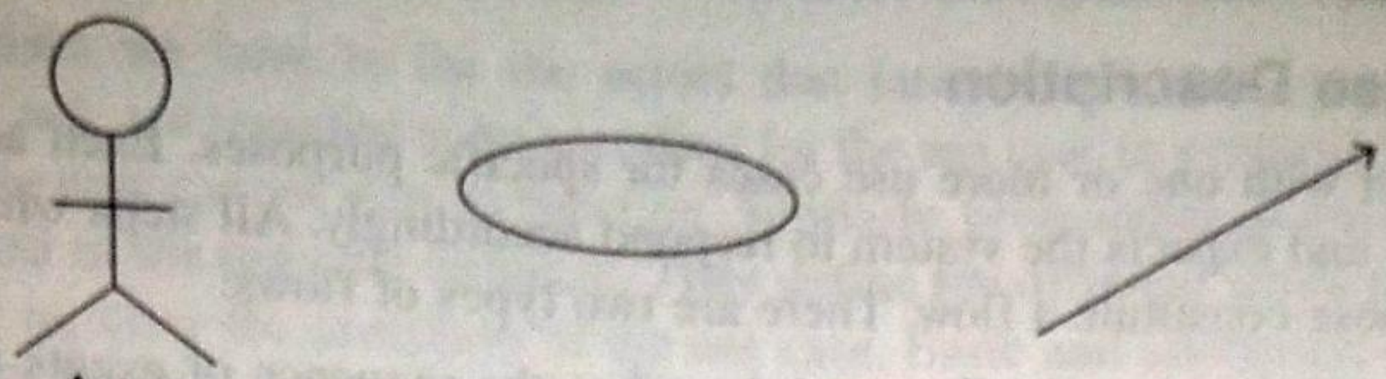


Figure 3.6 Components of use case.

whole system. However, for a large system, we may have to draw many use case diagrams to represent various portions of the system. The use case diagram of the LMS is given in Figure 3.7.

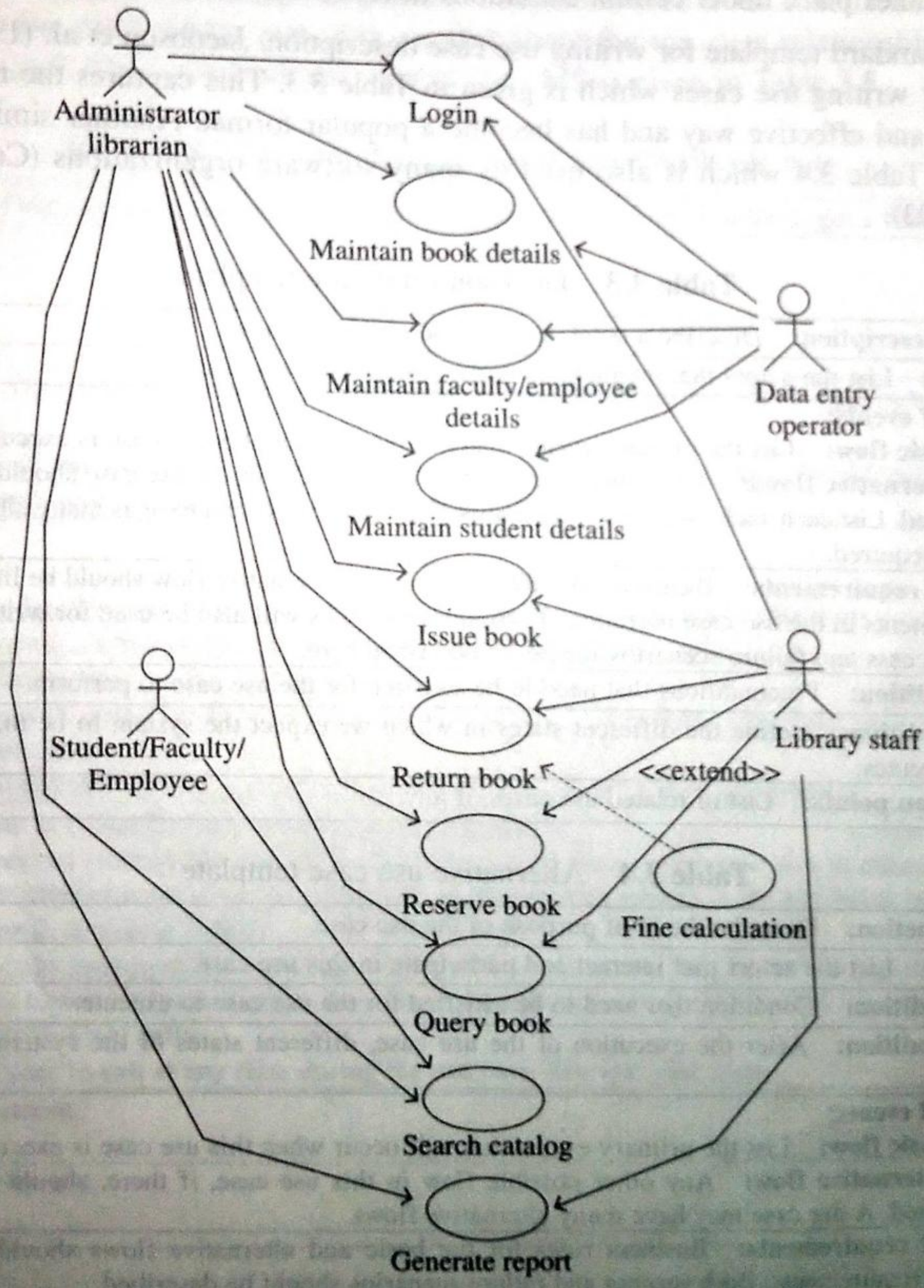


Figure 3.7 Use case diagram for library management system.

3.5.6 Use Case Description

Actors may interact with one or more use cases for specific purposes. Each actor gives some input to the system and expects the system to respond accordingly. All steps which are required to achieve the purpose constitute a flow. There are two types of flows:

1. *Basic flow:* It is the main flow and describes the sequence of events that takes place most of the time between actor and the system to achieve the purpose of the use case.
2. *Alternative flows:* If the basic flow is not successful due to any condition, the system takes an alternative flow. An alternative flow may occur due to failure of an expected service because of occurrence of exceptions/errors. There may be more than one alternative flow of a use case, which may not occur most of the time. Any alternative flow takes place under certain conditions in order to fulfil the purpose of a use case.

There is no standard template for writing use case description. Jacobson et al. (1999) have given a template for writing use cases which is given in Table 3.3. This captures the requirements in a disciplined and effective way and has become a popular format. Another similar template is also given in Table 3.4 which is also used by many software organizations (Cockburn, 2001; Quatrani, 2003).

Table 3.3 Jacobson's use case template

1.	**Brief description:** Describe a quick background of the use case.
2.	**Actors:** List the actors that interact and participate in this use case.
3.	**Flow of events:** 3.1. **Basic flow:** List the primary events that will occur when this use case is executed. 3.2. **Alternative flows:** Any subsidiary events that can occur in the use case should be separately listed. List each such event as an alternative flow. A use case can have as many alternative flows as required.
4.	**Special requirements:** Business rules for the basic and alternative flow should be listed as special requirements in the use case narration. These business rules will also be used for writing test cases. Both success and failure scenarios should be described here.
5.	**Precondition:** Preconditions that need to be satisfied for the use case to perform.
6.	**Postcondition:** Define the different states in which we expect the system to be in, after the use case executes.
7.	**Extension points:** List of related use cases, if any.

Table 3.4 Alternative use case template

1.	**Introduction:** Describe the brief purpose of the use case.
2.	**Actors:** List the actors that interact and participate in this use case.
3.	**Precondition:** Condition that need to be satisfied for the use case to execute.
4.	**Postcondition:** After the execution of the use case, different states of the systems are defined here.
5.	**Flow of events:** 5.1. **Basic flow:** List the primary events that will occur when this use case is executed. 5.2. **Alternative flow:** Any other possible flow in this use case, if there, should be separately listed. A use case may have many alternative flows.
6.	**Special requirements:** Business rules for the basic and alternative flows should be listed as special requirements. Both success and failure scenarios should be described.
7.	**Associated use cases:** List the related use cases, if any.

EXAMPLE 3.2 Consider the case study of the LMS given in Section 3.1. Give use case description of return a book use case.

Solution The use case description of return book is given in Table 3.6.

Table 3.6 Use case description of return book use case

Introduction: This use case documents the steps that must be followed in order to return a book.
Actors Administrator Library staff
Precondition: The administrator/library staff must be logged onto the system before the use case begins.
Postcondition: If the use case is successful, the book is returned back to the library and if needed, the "fine calculation" use case is called, otherwise the system state is unchanged.
Event Flow **Basic Flow** 1. The book information is read from the bar code of the book through the bar code reader. 2. The student/faculty/employee detail on whose name the books were issued is displayed on the system. The date of issue and return is also displayed. 3. The administrator/library staff checks the stamp on the book to check the duration of issue of the book. 4. The database is updated and the book status is updated. 5. The date stamp on the book is cancelled.
Alternative Flows **Alternative Flow 1: Late return of book** If the duration for which the book has been kept by the student is more than 15 days, then a fine calculation use case is called. After the execution of fine calculation use case, the original use case resumes the basic flow. **Alternative Flow 2: User exits** This allows the user to exit at any time during the use case. The use case ends.
Special requirement None
Associated use case Login, fine calculation

3.5.7 Generation of Scenario Diagrams

A use case scenario is an instance of a use case. This is nothing but a path through a use case. A use case may have many paths. A basic flow is a path which is expected to be traversed most of the time and becomes a scenario of the use case. Every alternative path also generates a scenario. We may have various combinations of basic and/or alternative flows and every combination leads to a path in a use case. The basic and alternative flows for a use case are shown in Figure 3.8.

The basic flow is represented by a straight arrow and the alternative flows by the curves. Some alternative flows return to the basic flow, while others end the use case. Preconditions and postconditions are checked at start and end points of a use case, respectively. We consider the issue a book use case, and its basic and alternative flows are shown in Figure 3.9.

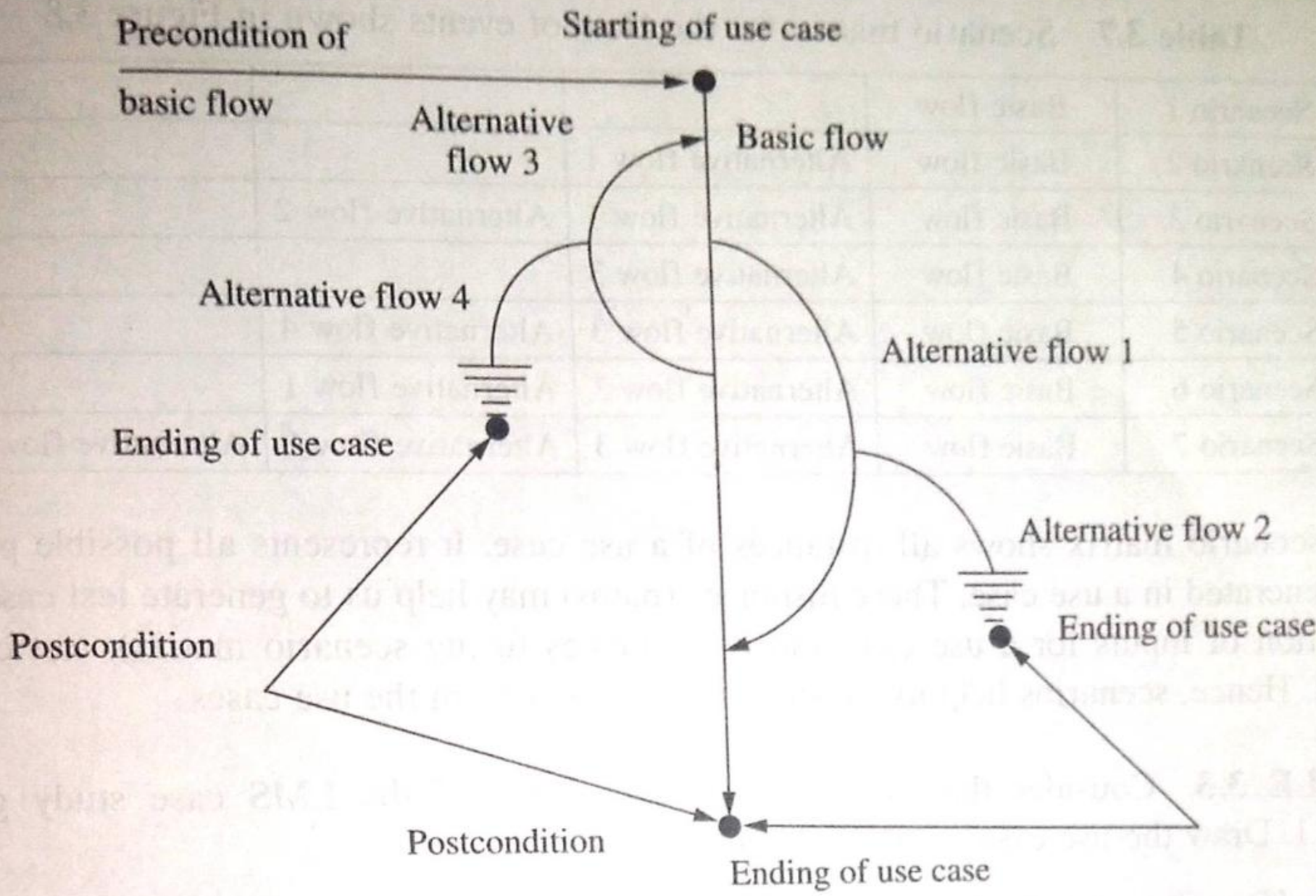
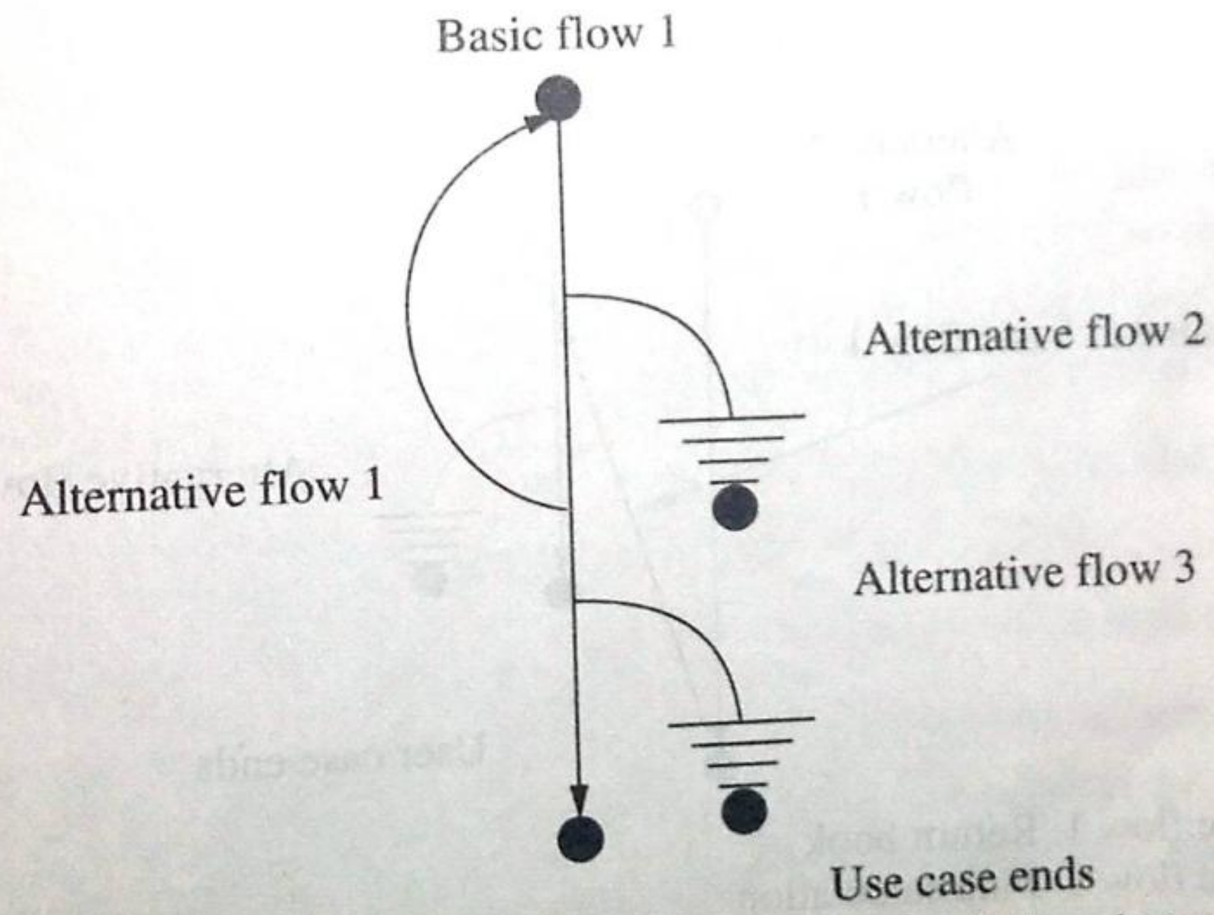


Figure 3.8 Basic and alternative flows with pre- and postconditions.



Alternative flow 1: Unauthorized student/faculty/employee
Alternative flow 2: Account is full
Alternative flow 3: User exits

Figure 3.9 Basic and alternative flows for issue book use case.

3.5.8 Creation of Use Case Scenario Matrix

Many scenarios are generated using the use case scenario diagram due to basic flow(s), alternative flows and various combinations of basic and alternative flows. A scenario matrix represents all scenarios of the use case scenario diagram. The scenario matrix of the use case scenario diagram given in Figure 3.8 is given in Table 3.7.

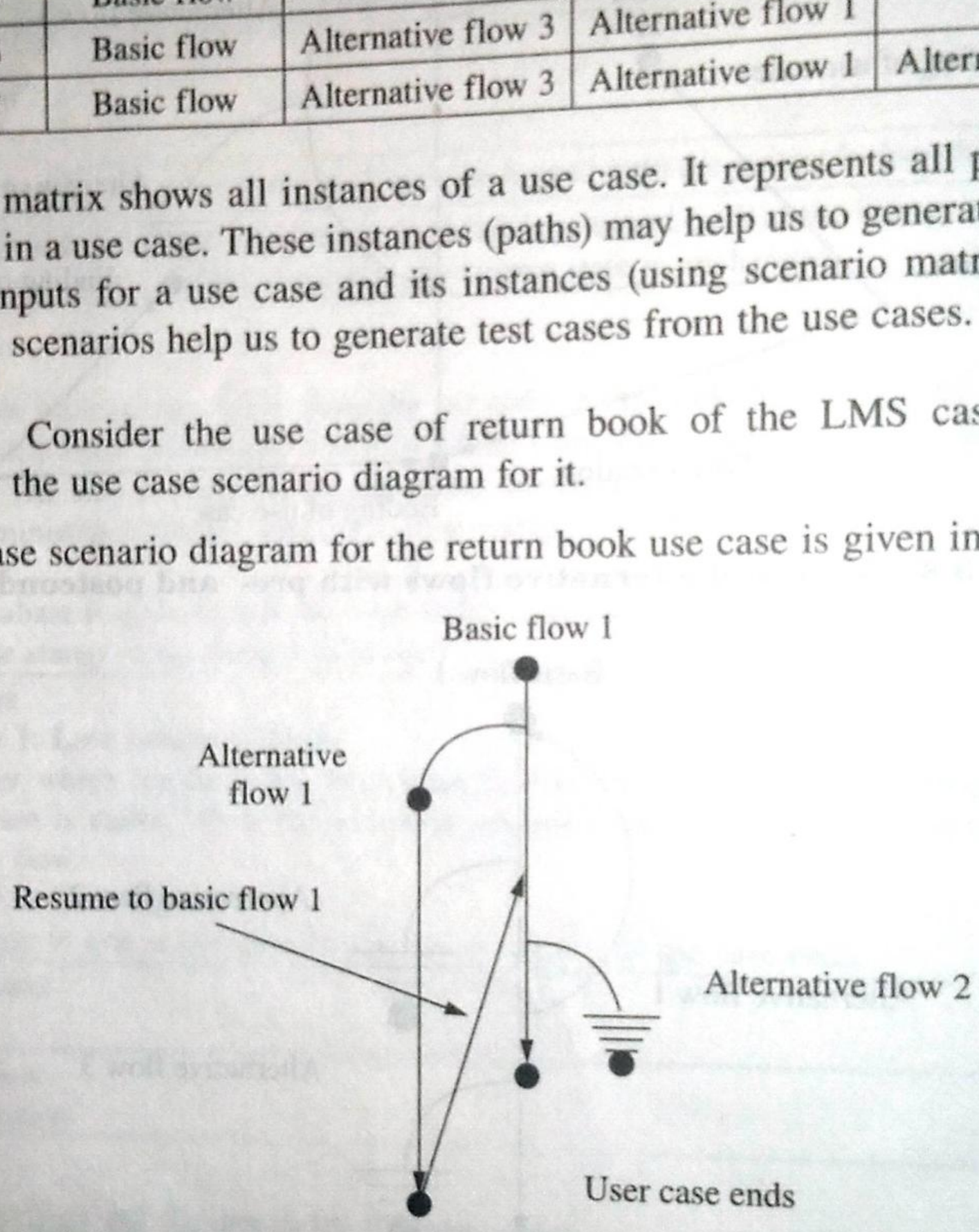
Table 3.7 Scenario matrix for the flow of events shown in Figure 3.8

Scenario 1	Basic flow			
Scenario 2	Basic flow	Alternative flow 1		
Scenario 3	Basic flow	Alternative flow 1	Alternative flow 2	
Scenario 4	Basic flow	Alternative flow 3		
Scenario 5	Basic flow	Alternative flow 3	Alternative flow 4	
Scenario 6	Basic flow	Alternative flow 3	Alternative flow 1	
Scenario 7	Basic flow	Alternative flow 3	Alternative flow 1	Alternative flow 2

The scenario matrix shows all instances of a use case. It represents all possible paths that may be generated in a use case. These instances (paths) may help us to generate test cases. After identification of inputs for a use case and its instances (using scenario matrix), test cases are generated. Hence, scenarios help us to generate test cases from the use cases.

EXAMPLE 3.3 Consider the use case of return book of the LMS case study given in Section 3.1. Draw the use case scenario diagram for it.

Solution Use case scenario diagram for the return book use case is given in Figure 3.10.



Basic flow 1: Return book
Basic flow 2: Fine calculation
Alternative flow 1: Late return of book
Alternative flow 2: User exits

Figure 3.10 Basic and alternative flows for return book use case.