**CASE STUDY 1**

**1. How to gather customer requirements?**

--->We use a four step process to collect and analyse customer data. These four steps are:

Identify

Collect

Analyse

Translate

**THE COLLECT PHASE**

There are basically 2 ways of obtaining data about customer needs, these are Leading and Lagging indicators. Lagging indicators are useful to give information about past customer behaviour, for example warranty or complaint data, the data usually already exists. Leading indicators give information about future customer behaviour or needs, for example customer surveys. Often this data does not already exist and needs to be collected proactively.

The cost of collecting lagging data is typically low as it already exists, but of course the business will most likely have already incurred costs due to having to correct warranty problems or put right complaints. The cost of collecting leading data is higher, but as it is forward looking there will most likely not be any cost already incurred.

It is important when considering data collection methods to bear in mind the total cost, which is a combination of the Cost of Quality (cost already incurred) and the data collection cost.

The chart below illustrates the total cost of data collection. The blue line shows the data collection costs, and the red line shows failure costs. Lagging indicators have low collection costs, but incur high failure costs, whilst leading indicators have low failure costs but incur high collection costs. The total cost is the sum of the two elements.

Examples of lagging indicators include the following:

-Existing customer surveys

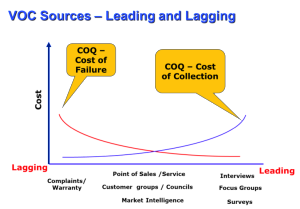
-Warranty information

-Industry publications and articles

-Research reports

-Market forecasts

-Strategic planning documents



-Specifications and Service Level Agreements (SLA’s)

Specifications and Service Level Agreements are very useful sources of data about customer requirements, and of course should not be overlooked. In many cases capturing the customer requirements is no more complex than obtaining the specification! However, just because there is a specification does not mean it fully captures the requirements, or that it is accurate. It is always worth asking questions of the customer about the validity and accuracy of the specification.

Leading indicators include the following:

-Interviews

-Focus Groups

-Surveys

-Market Research

An interview is a formal methodology for collecting customer needs using phone or face to face methods. An interview would usually be used to learn about a particular customer segment, and is an opportunity to find out a lot of qualitative data about customer needs. It is a useful method to get in-depth information from customers where there is no real understanding of their needs.

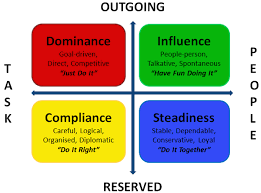
A focus group brings together a small group of customers to uncover general needs. It is used to capture a collective point of view from several customers at the same time, exploring the needs and concerns of specific customer segments.Focus groups are reflection oriented, and designed to understand factors influencing needs, wants and delighters. They are often used as a follow up to interviews, or as an introductory step before a big survey effort.Surveys are a research method to understand customer wants and perceptions using a large sample to gather valid, reliable and useful information that can help to provide quantitative data on customer wants, needs & opinions. They are most often carried out on a scale large enough to draw statistically valid information. They can also be used to benchmark a position, and the survey can be repeated at agreed intervals to track progress over time.

Market research is a general heading for any organized effort to gather information about target markets or customers. Market research provides important information to identify and analyse the market need, market size and competition. Any of the above techniques may be used in market research, but it is also possible to commission market research from an independent agency, who will then systematically gather information for the organisation to gain insight into customers and markets.

**2. How to identify which type of customer dealing with?**

Those of us that interact with customers have noticed that each one has a little bit different personality that comes out during our interactions with them. Thankfully, we’re not all the same. These personalities are individual and depending on your interaction with them may determine your success or failure of striking an understanding, agreement, or sale. Thrown into this mix is your personality. You have unique personality traits that make you you.

While everyone has a unique personality, there are 4 dominant traits that is weaved in all of us, and it’s a blend of these 4 dominant traits that make us unique. Here’s where the science of personality and your ability to identify those dominant traits and which are more and which are less dominant can affect your success or failure in dealing with another individual. While an in depth study of this subject may take weeks or months to fully understand, here are a few tips that you can practice on the job and hone your skills to make this a positive for you and your customer.



Lets discuss these four types of customers in details.

**1. Dominant Customer**

his person is a bottom line type person. They want to cut to the chase and make a decision with a few facts that they tend to piece together to make a decision. With this type of customer, you will only hurt yourself continuing on with a detailed explanation. He’s identified the problem in his mind, made a decision to move forward and fix it and only wants to know the bottom line price and time line.If your personality type is the next one we will discuss and you continue on with what you feel comfortable explaining you will most likely lose his attention and he’ll move on to finding someone who can give him the bottom line information he wants.

**2. Steady Customer**

The next type of customer is a **steady** type personality. They like details and the more the better. They are the folks who read all of the information you give them and usually want more. If you are the same type, you will both revel in the detail and most likely come to agreement. They will expect to see all of the specifications, ASTM’s if applicable, data, studies, and other information. They also will read every review your company has ever received by online rating sites. He wants to be sold this way and the more documentation you feed him the better.

**3.Influencer Customer**

An**influencer** is our next type. They’re relationship orientated and would rather gather with you over golf or cocktails, and form a friendship bond with you. They could care less about how you fix the issue as long as you’re friends and vow to fix the problem as a friend would fix it. Giving this person a bottom line proposal will most likely fail as well as feeding them all of the specifications will fail. Friends first, then they’ll do business with you. Short of making that friendship bond your proposal will end in the dead file.

**4.Compliant Customer**

The final type of customer is referred to as **compliant**. Being friends with this type doesn’t make much difference. They do like most of the details of what you are proposing, but not in the quantify of information the steady type wants. They enjoying friendly relationships with those they do business with, but not to the extent found by the influencer. Finally they will get to the bottom line, but need to know all of the information they believe they need to be comfortable and will want the advice of others to make a decision. Referrals from previous customers is very helpful in getting these folks to do business with you.

The other types of sales customer are follows-

**1. Potential customer -  The Potential Paul(example)**

### The Potential Paul is a type of customer that is on the very beginning of your sales funnel. Technically, Paul is not your customer yet.

How to deal with Potential Paul:

* **Show him value**: You can capitalize on his interest by clearly showing him what he can get from your product. You can do this yourself or point him to a resource like a landing page or a case study that will do it for you.
* **Reveal yourself**: Make sure to let the potential customer know that they can ask for help or advice at any time. Even if the customer won’t need it immediately, they will appreciate the offer.

**2. New customer – New Neil(example)**

New Neil is the fresh customer that just bought something from you. He is still learning the ropes of using your product. You need to do everything in your power to make that adoption period smooth.

How to deal with New Neil:

* **Guide them to success**: You can earn a long-lasting customer by investing a bit of your time into explaining how your product works and making sure the new customer knows how to use it. You can do that with a proper onboarding process.
* **Leave a contact option open**: Even if you offer an automated onboarding to customers, have a live customer service option available. It will go a long way in situations when a customer has a question that’s not covered in the onboarding.

**3.Implulsive Customer - Impulsive Iggy(example)**

This is the type of customer that can make a buying decision in an instant, provided that the conditions are right.

How to deal with Impulsive Iggy:

* **Clear the way to checkout**: Make sure nobody needs a manual to make a purchase on your website. The less clicks and information needed to make a purchase, the better.
* **Quick and concise help**: If you get a question from Iggy, make the answer short. Stick to the brass tacks. You also need to provide the response quickly. If you take too long, the buying impulse will fade and Iggy will leave your website.

**4.Loyal Customer - Loyal Larry(example)**

This type of customers **keeps coming back for more**. Apart from having a significant impact on your revenue, Larry will be also your **brand’s ambassador**.

How to deal with Loyal Larry -

* **Give him a platform**: You can help Larry spread the love and feature him in a [case study](https://www.livechatinc.com/blog/case-study/). A bit of social proof like that will make your landing pages much more appealing to potential customers.
* **Learn from his experience**: See what turned Larry into a loyal follower and make sure that happens more often with other customers.

**3. How to prepare questionnaire based on your observation?**

## Definition of questionnaire

1. a set of question for obtaining statistically useful or personal information from individuals
2. a written or printed questionnaire often with spaces for answers
3. a survey made by the use of a questionnaire

## The following are characteristics of good questionnaire:

* Questionnaire should deal with important or significant topic to create interest among respondents.
* It should seek only that data which can not be obtained from other sources.
* It should be as short as possible but should be comprehensive.
* It should be attractive.
* Directions should be clear and complete.
* It should be represented in good Psychological order proceeding from general to more specific responses.
* Double negatives in questions should be avoided.
* It should avoid annoying or embarrassing questions.
* It should consist of a written list of questions.

## Types of Questionnaires:

Based on the type of questions used, questionnaires are as follows:

### 1. Structured questionnaire:

Comes under quantitative research. It includes the low number of researchers and the high number of respondents. They are also called as closed questionnaires. They usually include answers such as very bad, bad, good, very good and so on.

* They have a definite and concrete questions
* They have to be prepared well in advance so as to ask as much questions and receive info from the respondent.
* A formal inquiry is initiated.
* Supplements and checks the previously accumulated data.
* Commonly used in for social and economic problems, to study about the changes caused due to change in policies, laws etc.

These question come sunder structured questionnaire.

#### Contingency questions:

This comes under structured questionnaire. Here a question is asked only if the respondent is able to give a answer to the previous question.

#### Matrix questions:

Similar kind of options are provided to multiple questions. The questions are provided one under the other, forming a matrix with response categories on top and questions down the side

### 2. Unstructured questionnaire:

A version of qualitative survey. They are usually based around more open questions. Open questions also means recording more data as the respondents can point out what is important for them, in their own words and methods. But it is more difficult from the researcher’s side, since it does not give the correct idea of the topic and moreover proper understanding of the data is needed.

* Usually used at the time of an interview.
* Doesn’t require much planning and time.
* More flexible for applying in many areas.
* Usually used to collect data about people and their personal info such as family, debates, beliefs etc.

### 3. Scaled questionnaires:

The respondents are asked to scale the answers based on a given rating prescribed by the question

## Advantages of Questionnaires:

1. Questionnaires are really inexpensive when they are handled properly. They can be cheaper than taking surveys which requires a lot of time and money.

2. Questionnaires can be of different types, written, postal, telephone and many other methods.

3. A single question or a topic can be asked to many at the same time without any kind of delay. Unlike surveys they don’t have to go to each and everyone to get an opinion.

4. It is an effective method to get an opinion from a large number of people.

5. Large number of respondents can be possible varying in age, sex, occupation etc.

6. Question responses can be highly defined and specific, depending upon the type of questions asked in the questionnaire.

7. These results can also be included as statistical survey, the deciding factor is the nature of the questionnaire and on what topic was the questionnaire based on.

8. Unlike face to face surveys where the respondent has to answer within that moment itself, questionnaires gives time to the respondents to think carefully, before giving the answers.

9. Questionnaires are easily replicable and can be repeated, and if well-constructed and properly piloted, they can be used as comparative materials for future studies and projects.

10. Standardized questionnaires can already be validated and can be used to compare between works and studies.

## Disadvantages of Questionnaires:

1. The results for questionnaires are based only on the type of question being asked. If the questions are poorly worded or is biased in nature, then the result analysed will also be of the same nature.

2. Questionnaires can pose difficulties to the analyst if he/she is not familiar with the system based on which the questions are being asked. That is, the analyst may not be able to produce the required questions, and hence the required results cannot be achieved.

3. Questionnaires tend to give an alien feeling to many respondents and hence they are very impersonal irrespective of the situation. Thus, many people do prefer face to face conversations than answering questionnaires.

4. The response rate maybe poor in questionnaires, if people do not have time or they don’t feel any importance in answering them. This is one of the main disadvantages of questionnaires.

5. Questionnaires do make it impossible for people to answer questions according to their own opinion. This makes them very constricted in terms of answering such questions. This feels true particularly when the questionnaires have closed end questions. They limit the opinions of the respondent by a huge factor. Hence, less honest and detailed answers can be received.

6. Some participants may forget about the whole issue and tend to forget why such questionnaire was present in the first place.

7. Open ended questions may take a long time and will produce a large amount of data that will take time to analyze.

8. Respondents may answer the questionnaire superficially, if it takes time to answer such questions. This might lead to inadequate and maybe unwanted data to analyze the final result.

9. Do not try to ask too many question since it might bore the respondent and ultimately it will lead to incorrect answers.

10. Try to make the questionnaire as anonymous as possible as it will be more beneficial for the respondent to explain their opinions in detail.

* **Six steps to creating an effective questionnaire:**

1. **Work as a Partner.** Align your research theme with your client’s overall business objectives so outcomes will complement the communications strategy. Together you can begin to craft the concept, timing and desired findings.
2. **Keep it Simple.** Write short, simple, specific questions using as few words as possible. To capture the respondent’s actual beliefs, it’s best to write a clear statement that can be responded to without too much deliberation. The more instinctual reaction you receive, the better.
3. **Choose the Best Delivery Method.** Today's surveys can be delivered over the computer, in person, on the phone or by mail. Postal surveys can be cheap but responses can be slow. Face-to-face can be expensive but will generate the fullest responses. Web surveys can be cost-effective but inconsistent with response rates. Telephone can be expensive, but will often generate high response rates and will allow for follow-up questions to enhance findings. So, a choice must be made.
4. **Ask the Same Question Twice but in Different Ways**. To ensure you are understanding a person’s true opinion on a given topic, it’s smart to ask the same question a couple of times. It will help you avoid the respondent bias that inevitably presents itself with each survey, and gives you a better chance at finding the person's true opinion on a given topic.
5. **Be Selective From the Start.** Although you may feel a person is the right one to take the survey, it’s best to ask a series of screening questions to make sure. Position those at the beginning so you're not wasting anyone's time. Examples would be demographic benchmarks, such as salary, education and geography, etc.
6. **Pilot the Questionnaire**. By testing the survey with a small population, you’ll determine if it’s set to do what you need it to do. This soft launch enables you to determine whether some questions may need to paraphrased, reordered or removed.

* **Order of the questions :**

1. Begin with simple and genral question.
2. No sensitive an embarassing question at beginning.
3. Move from genral to specific logical manner.
4. sensitive question at the end.

* **Conclusion:**
  1. Questionnaire are the main way collecting data.
  2. But the questionnaire must be highly reliable and valid.
  3. Using standardized questionnaire will give us the appropriate data and will yield a valid study.
  4. One man follow all the basic guidelines and method of constructing a questionnaire and test itbefore using it.

EXPERIMENT-2

**AIM:** To know about use-case and sequence diagrams.

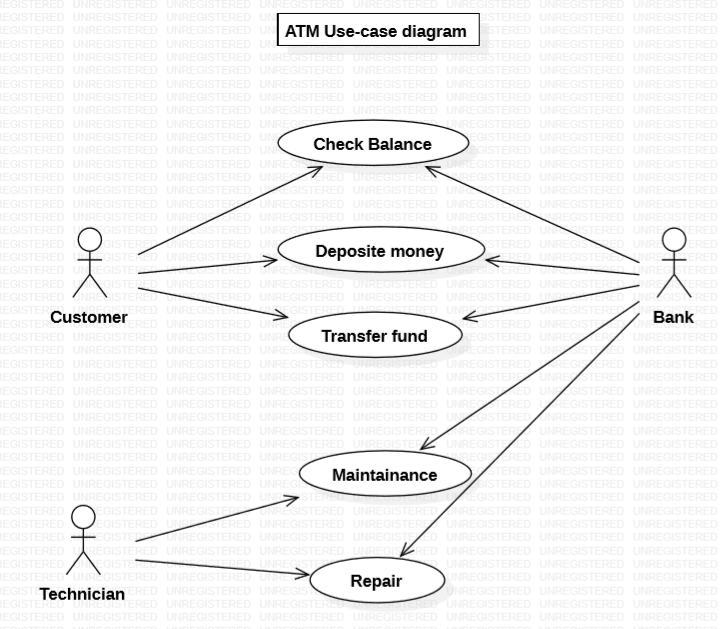
1. **USE-CASE DIAGRAM**

The purpose of use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other four diagrams (activity, sequence, collaboration, and State chart) also have the same purpose. 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. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.

Actors can be a human user, some internal applications, or may be some external applications. When we are planning to draw a use case diagram, we should have the following items identified.

* Functionalities to be represented as use case
* Actors
* Relationships among the use cases and actors.

**Use-case Diagram of ATM:**



1. **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.

**Sequence Diagram Notations –**

1. **Actors –** An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram.
2. **Lifelines –** A lifeline is a named element which depicts an individual participant in a sequence diagram.
3. **Messages –** Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline.

Messages can be broadly classified into the following **categories**:

* **Synchronous messages –** A synchronous message waits for a reply before the interaction can move forward.
* **Asynchronous Messages –** An asynchronous message does not wait for a reply from the receiver.
* **Create message –** We use a Create message to instantiate a new object in the sequence diagram.
* **Delete Message –** We use a Delete Message to delete an object.
* **Self Message –** Certain scenarios might arise where the object needs to send a
* message to itself.
* **Reply Message –** Reply messages are used to show the message being sent from the receiver to the sender.
* **Found Message –** A Found message is used to represent a scenario where an unknown source sends the message.
* **Lost Message –** A Lost message is used to represent a scenario where the recipient is not known to the system.

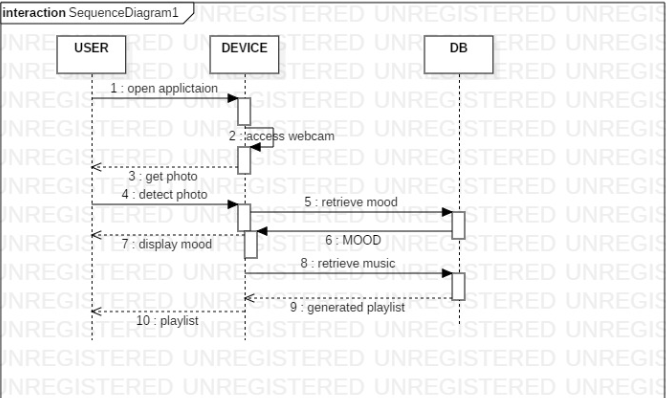
1. **Guards –** To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

**Uses of sequence diagrams –**

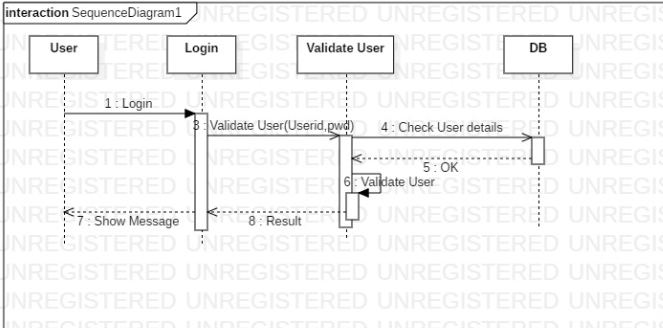
* Used to model and visualise the logic behind a sophisticated function, operation or procedure.
* They are also used to show details of UML use case diagrams.
* Used to understand the detailed functionality of current or future systems.

.

**Sequence Diagram of user device and database**



**Sequence Diagram of validate user and database**



EXPERIMENT-3

**AIM:** ER Modelling From the Problem Statements.

**Entity Relationship Model**

Entity-Relationship model is used to represent a logical design of a database to be created. In ER model, real world objects (or concepts) are abstracted as entities, and different possible associations among them are modeled as relationships.For example, student and school -- they are two entities. Students study in school. So, these two entities are associated with a relationship "Studies in".As another example, consider a system where some job runs every night, which updates the database. Here, job and database could be two entities. They are associated with the relationship "Updates".

**Entity Set and Relationship Set**

An entity set is a collection of all similar entities. For example, "Student" is an entity set that abstracts all students. Ram, John are specific entities belonging to this set. Similarly, a "Relationship" set is a set of similar relationships.

**Attributes of Entity**

Attributes are the characteristics describing any entity belonging to an entity set. Any entity in a set can be described by zero or more attributes.For example, any student has got a name, age, an address. At any given time a student can study only at one school. In the school he would have a roll number, and of course a grade in which he studies. These data are the attributes of the entity set Student.

**Keys**

One or more attribute(s) of an entity set can be used to define the following keys:

• **Super key:** One or more attributes, which when taken together, helps to uniquely identify an entity in an entity set. For example, a school can have any number of students. However, if we know grade and roll number, then we can uniquely identify a student in that school.

• **Candidate key:** It is a minimal subset of a super key. In other words, a super key might contain extraneous attributes, which do not help in identifying an object uniquely. When such attributes are removed, the key formed so is called a candidate key.

• **Primary key:** A database might have more than one candidate key. Any candidate key chosen for a particular implementation of the database is called a primary key.

• **Prime attribute:** Any attribute taking part in a super key

**Weak Entity**

An entity set is said to be weak if it is dependent upon another entity set. A weak entity can't be uniquely identified only by it's attributes. In other words, it doesn't have a super key.For example, consider a company that allows employees to have travel allowance for their immediate family. So, here we have two entity sets: employee and family, related by "Can claim for". However, family doesn't have a super key. Existence of a family is entirely dependent on the concerned employee. So, it is meaningful only with reference to employee.

**Entity Generalization and Specialization**

Once we have identified the entity sets, we might find some similarities among them. For example, multiple person interacts with a banking system. Most of them are customers, and rest employees or other service providers. Here, customers, employees are persons, but with certain specializations. Or in other way, person is the generalized form of customer and employee entity sets.ER model uses the "ISA" hierarchy to depict specialization (and thus, generalization).

**Mapping Cardinalities**

One of the main tasks of ER modeling is to associate different entity sets. Let's consider two entity sets E1 and E2 associated by a relationship set R. Based on the number of entities in E1 and E2 are associated with, we can have the following four type of mappings:

• **One to one:** An entity in E1 is related to at most a single entity in E2, and vice versa

• **One to many:** An entity in E1 could be related to zero or more entities in E2. Any entity in E2 could be related to at most a single entity in E1.

• **Many to one:** Zero or more number of entities in E1 could be associated to a single entity in E2. However, an entity in E2 could be related to at most one entity in E1.

• **Many to many:** Any number of entities could be related to any number of entities in E2, including zero, and vice versa.

**ER Diagram**

From a given problem statement we identify the possible entity sets, their attributes, and relationships among different entity sets. Once we have these information, we represent them pictorially, called an entity-relationship (ER) diagram.

**Importance of ER modeling**

Figure - 01 shows the different steps involved in implementation of a (relational) database.

Database design steps

Database design steps Figure - 01: Steps to implement a RDBMS

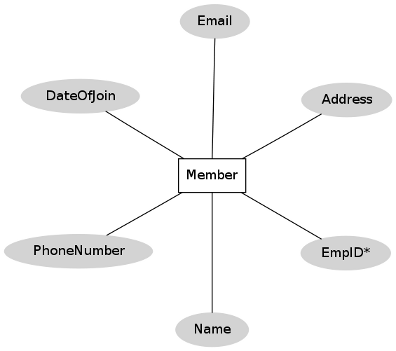
Given a problem statement, the first step is to identify the entities, attributes and relationships. We represent them using an ER diagram. Using this ER diagram, table structures are created, along with required constraints. Finally, these tables are normalized in order to remove redundancy and maintain data integrity. Thus, to have data stored efficiently, the ER diagram is to be drawn as much detailed and accurate as possible.

## Graphical Notations for ER Diagram

| **Term** | **Notation** | **Remarks** |
| --- | --- | --- |
| Entity set | Entity | Name of the set is written inside the rectangle |
| Attribute | Attribute | Name of the attribute is written inside the ellipse |
| Entity with attributes | Entity with attributes | Roll is the primary key; denoted with an underline |
| Weak entity set | Weak entity |  |
| Relationship set | Relationship | Name of the relationship is written inside the diamond |
| Related enity sets | Entity relationship |  |
| Relationship cardinality | Relationship cardinality | A person can own zero or more cars but no two persons can own the same car |
| Relationship with weak entity set | Weak entity relationship |  |

**CASE STUDY**

The first step towards ER modelling is to identify the set of relevant entities from the given problem statement. The two primary, and obvious, entity sets in this context are "Member" and "Book". The entity set "Member" represents all students, professors, or employees who have registered themselves with the LIS. While registering with the LIS one has to furnish a lot of personal and professional information. This typically includes name (well, that is trivial), employee ID (roll # for students), email address, phone #, age, date of joining in this institute. The system may store some not-so-important information as well like, blood group, marital status, and so on. All these pieces of information that an user has to provide are sufficient to describe a particular member. These characteristics are the attributes of the entities belonging to the entity set "Member”. It is essential for an entity to have one or more attributes that help us to distinguish it from another entity. 'Name' can't help that -- two persons could have exactly the same name. However, ('Name', 'Phone #') combination seems to be okay. No two persons can have the same phone number. 'Employee ID', 'Email address' are other potential candidates. Here, 'Employee ID', 'Email address' and ('Name', 'Phone #') are super keys. We choose 'Employee ID' to uniquely identify an user in our implementation. So, 'Employee ID' becomes our primary key (PK) for the "Member" entity set.

  
Figure 1: "Member" entity set

A graphical representation of the "Book" entity set is shown in figure 2

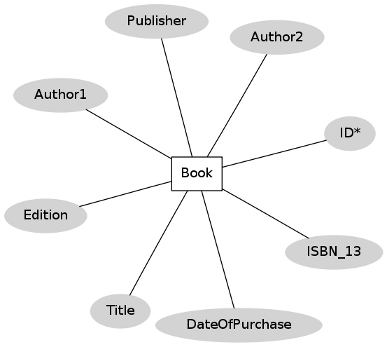


Figure 2: "Book" entity set

EXPERIMENT-4

**AIM:** State chart and Activity Modelling.

**Statechart diagrams**

In case of Object Oriented Analysis and Design, a system is often abstracted by one or more classes with some well defined behaviour and states. A statechart diagram is a pictorial representation of such a system, with all it's states, and different events that lead transition from one state to another.

Statechart diagrams are normally drawn to model the behaviour of a complex system. For simple systems this is optional.

Building Blocks of a Statechart Diagram:

**State:** A state is any "distinct" stage that an object (system) passes through in it's lifetime. An object remains in a given state for finite time until "something" happens, which makes it to move to another state

**Transition**: It is movement from one state to another state in response to an external stimulus (or any internal event). A transition is represented by a solid arrow from the current state to the next state.

**Action:**As mentioned in [ii], actions represents behaviour of the system. While the system is performing any action for the current event, it doesn't accept or process any new event.

Guidelines for drawing Statechart Diagrams

Following steps could be followed, as suggested in todraw a statechart diagram:

* For the system to developed, identify the distinct states that it passes through
* Identify the events (and any precondition) that cause the state transitions. Often these would be the methods of a class as identified in a class diagram.
* Identify what activities are performed while the system remains in a given state

**Activity Diagrams**

Activity diagrams fall under the category of behavioural diagrams in Unified Modeling Language. It is a high level diagram used to visually represent the flow of control in a system. It has similarities with traditional flow charts. However, it is more powerful than a simple flow chart since it can represent various other concepts like concurrent activities, their joining, and so on . Activity diagrams, however, cannot depict the message passing among related objects. As such, it can't be directly translated into code. These kind of diagrams are suitable for confirming the logic to be implemented with the business users. These diagrams are typically used when the business logic is complex. In simple scenarios it can be avoided entirely

The following table shows commonly used components with a typical activity diagram:

| **Component** | **Graphical Notation** |
| --- | --- |
| Activity | Activity |
| Flow | Flow |
| Decision | Decision |
| Merge | Merge |
| Fork | Fork |
| Join | Join |
| Note | Note |

The SE VLabs Institute has been recently setup to provide state-of-the-art research facilities in the field of Software Engineering. Apart from research scholars (students) and professors, it also includes quite a large number of employees who work on different projects undertaken by the institution.

As the size and capacity of the institute is increasing with the time, it has been proposed to develop a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.

From the given problem we can identify at least four different functionality offered by the system:

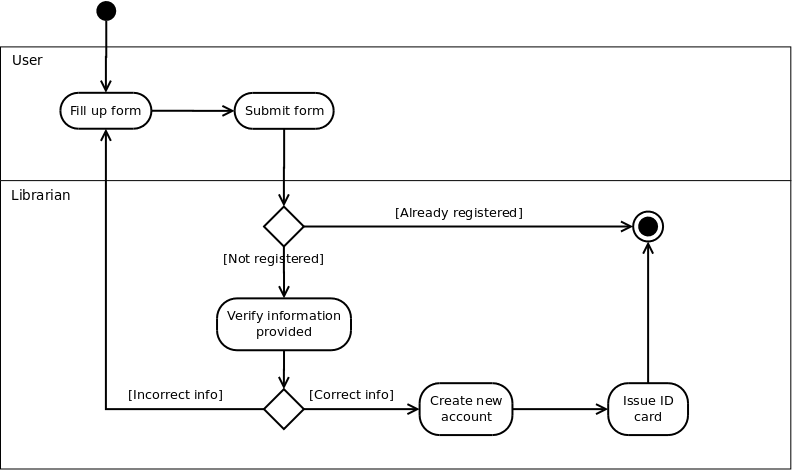
Register a new member

Issue book

Reissue book

Update inventory

To begin with, let's consider the activity diagram for user registration, as shown in figure - 01.



User Registration Figure-01: Activity diagram for new user registration

A new user fills up the registration form for library membership (either online or in paper), and submits to the librarian. Of course, an already registered user can't create another account for himself (or, herself). For users' who don't have an account already and have submitted their registration forms, the librarian verifies the information provided, possibly against the central database used by the institution. If all information have been provided correctly, librarian goes on with creating a new account for the user. Otherwise, the user is asked to provide all and correct information in his (her) registration form. Once a new account has been created for the user, he (she) is being issued an ID card, which is to be provided for any future transaction in the library.

Note that in the above diagram two swim lanes haven been shown indicated by the labels User and Librarian. The activities have been placed in swim lanes that correspond to the relevant role.

One of the major events that occur in any library is issue of books to it's members. Figure-02 tries to depict the workflow involved while issuing books.

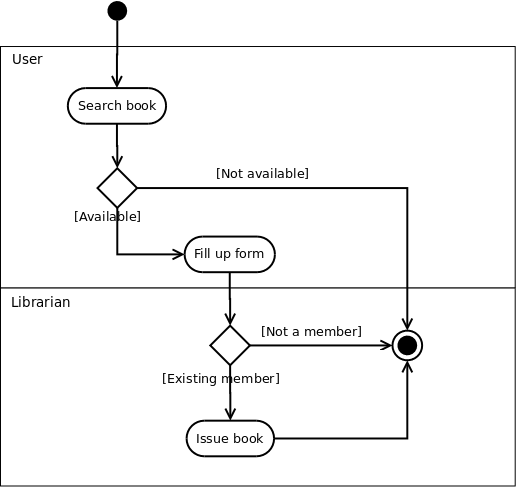


Figure-02:Activity diagram for issuing books

EXPERIMENT-5

**AIM:** Modelling Data Flow Diagrams.

**Data Flow Diagram**

DFD provides the functional overview of a system. The graphical representation easily overcomes any gap between ’user and system analyst’ and ‘analyst and system designer’ in understanding a system. Starting from an overview of the system it explores detailed design of a system through a hierarchy. DFD shows the external entities from which data flows into the process and also the other flows of data within a system. It also includes the transformations of data flow by the process and the data stores to read or write a data.

**Graphical notations for Data Flow Diagram**

| **Term** | **Notation** | **Remarks** |
| --- | --- | --- |
| External entity | External entity | Name of the external entity is written inside the rectangle |
| Process | Process | Name of the process is written inside the circle |
| Data store | Data store | A left-right open rectangle is denoted as data store; name of the data store is written inside the shape |
| Data flow | Data flow | Data flow is represented by a directed arc with its data name |

**Explanation of Symbols used in DFD**

**Process:** Processes are represented by circle. The name of the process is written into the circle. The name of the process is usually given in such a way that represents the functionality of the process. More detailed functionalities can be shown in the next Level if it is required. Usually it is better to keep the number of processes less than 7 If we see that the number of processes becomes more than 7 then we should combine some the processes to a single one to reduce the number of processes and further decompose it to the next level .

**External entity:** External entities are only appear in context diagram.External entities are represented by a rectangle and the name of the external entity is written into the shape. These send data to be processed and again receive the processed data.

**Data store:** Data stares are represented by a left-right open rectangle. Name of the data store is written in between two horizontal lines of the open rectangle. Data stores are used as repositories from which data can be flown in or flown out to or from a process.

**Data flow**: Data flows are shown as a directed edge between two components of a Data Flow Diagram. Data can flow from external entity to process, data store to process, in between two processes and vice-versa.

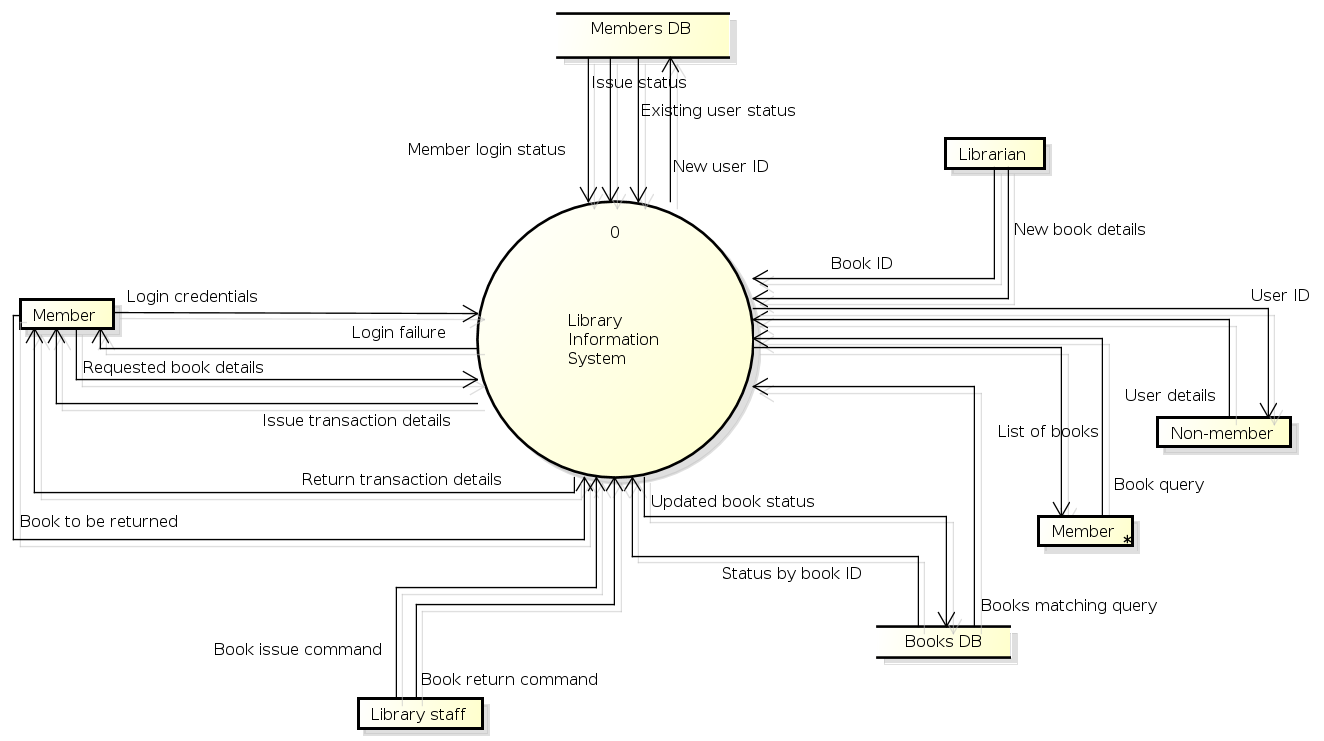
**Context diagram and leveling DFD**

We start with a broad overview of a system represented in level 0 diagram. It is known as context diagram of the system. The entire system is shown as single process and also the interactions of external entities with the system are represented in context diagram.Further we split the process in next levels into several numbers of processes to represent the detailed functionalities performed by the system. Data stores may appear in higher level DFDs.Numbering of processes : If process ‘p’ in context diagram is split into 3 processes ‘p1’, ‘p2’and ‘p3’ in next level then these are labeled as 0.1, 0.2 and 0.3 in level 1 respectively. Let the process ‘p3’ is again split into three processes ‘p31’, ‘p32’ and ‘p33’ in level 2, so, these are labeled as 0.3.1, 0.3.2 and 0.3.3 respectively and so on.Balancing DFD: The data that flow into the process and the data that flow out to the process need to be match when the process is split into in the next level[2]. This is known as balancing a DFD.

**CASE STUDY**

The SE VLabs Institute has been recently setup to provide state-of-the-art research facilities in the field of Software Engineering. Apart from research scholars (students) and professors, it also includes quite a large number of employees who work on different projects undertaken by the institution.As the size and capacity of the institute is increasing with the time, it has been proposed to develop a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.The final deliverable would a web application, which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (eg., passwords) is stored in plain text.

Figure 1 shows the context-level DFD for LIS. The entire system is represented with a single circle (process). The external entities interacting with this system are members of LIS, library staff, librarian, and non-members of LIS. Two database are used to keep track of member information and details of books in the library.Let us focus on the external entity, Member. In order to issue or return books a member has to login to the system. The data flow labeled with "Login credentials" indicate the step when a member authenticates himself by providing required information (user ID, password). The system in turn verifies the user credentials using information stored in the members database. If all information are not provided correctly, the user is shown a login failure message. Otherwise, the user can continue with his operation. Note that a DFD does not show conditional flows. It can only summarize the information flowing in and out of the system.The data flow with the label "Requested book details" identify the information that the user has to provide in order to issue a book. LIS checks with the books database whether the given book is available. After a book has been issued, the transaction details is provided to the member

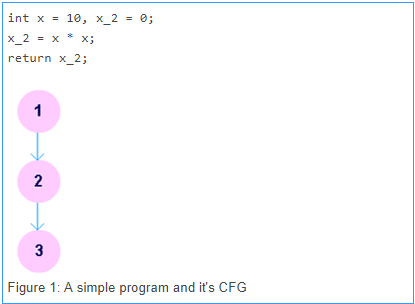


EXPERIMENT-6

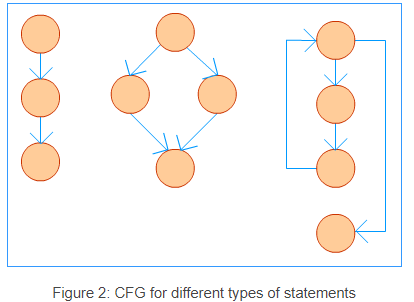
**AIM:** Estimation of Test Coverage Metrics and Structural Complexity.

**Control Flow Graph**

A control flow graph (CFG) is a directed graph where the nodes represent different instructions of a program, and the edges define the sequence of execution of such instructions. Figure 1 shows a small snippet of code (compute the square of an integer) along with it's CFG. For simplicity, each node in the CFG has been labeled with the line numbers of the program containing the instructions. A directed edge from node #1 to node #2 in figure 1 implies that after execution of the first statement, the control of execution is transferred to the second instruction.



A program, however, doesn't always consist of only sequential statements. There could be branching and looping involved in it as well. Figure 2 shows how a CFG would look like if there are sequential, selection and iteration kind of statements in order.



**Terminologies**

**Path**

A path in a CFG is a sequence of nodes and edges that starts from the initial node (or entry block) and ends at the terminal node. The CFG of a program could have more than one terminal nodes.

**Linearly Independent Path**

A linearly independent path is any path in the CFG of a program such that it includes at least one new edge not present in any other linearly independent path. A set of linearly independent paths give a clear picture of all possible paths that a program can take during it's execution. Therefore, path-coverage testing of a program would suffice by considering only the linearly independent paths.

In figure 3 we can find four linearly independent paths:

1 - 3 - 6 - (7, 8) - 10

1 - 3 - 6 - (7, 8) - 9 - 10

1 - 3 - (4, 5) - 6 - (7, 8) - 10

1 - 3 - (4, 5) - 6 - (7, 8) - 9 - 10

Note that 1 - 3 - (4, 5) - 3 - (4, 5) - 6 - (7, 8) - 10, for instance, won't qualify as a linearly independent path because there is no new edge not already present in any of the above four linearly independent paths.

**McCabe's Cyclomatic Complexity**

McCabe had applied graph-theoretic analysis to determine the complexity of a program module [vi]. Cyclomatic complexity metric, as proposed by McCabe, provides an upper bound for the number of linearly independent paths that could exist through a given program module. Complexity of a module increases as the number of such paths in the module increase. Thus, if Cyclomatic complexity of any program module is 7, there could be up to seven linearly independent paths in the module. For a complete testing, each of those possible paths should be tested.

**Optimum Value of Cyclomatic Complexity**

A set of threshold values for Cyclomatic complexity has been presented in , which we reproduce below.

| **V(G)** | **Module Category** | **Risk** |
| --- | --- | --- |
| 1-10 | Simple | Low |
| 11-20 | More complex | Moderate |
| 21-50 | Complex | High |
| > 50 | Unstable | Very high |

**CASE STUDY**

The SE VLabs Institute has been recently setup to provide state-of-the-art research facilities in the field of Software Engineering. Apart from research scholars (students) and professors, it also includes quite a large number of employees who work on different projects undertaken by the institution.As the size and capacity of the institute is increasing with the time, it has been proposed to develop a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.The final deliverable would a web application (using the recent HTML 5), which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (eg., passwords) is stored in plain text.

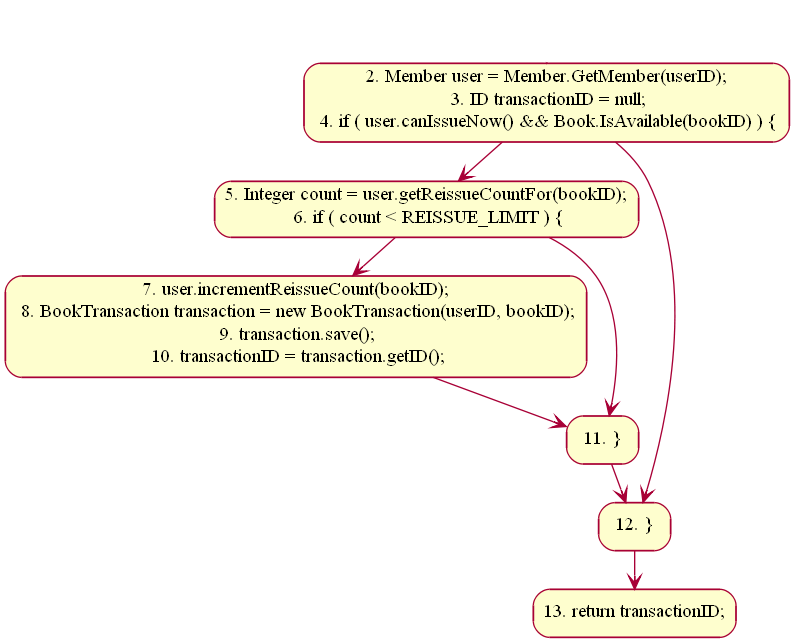


Figure 1. CFG for "ReissueBook" method

EXPERIMENT-7

**AIM:** Designing Test Suites.

**Software Testing**

Testing software is an important part of the development life cycle of a software. It is an expensive activity. Hence, appropriate testing methods are necessary for ensuring the reliability of a program. According to the ANSI/IEEE 1059 standard, the definition of testing is the process of analyzing a software item, to detect the differences between existing and required conditions i.e. defects/errors/bugs and to evaluate the features of the software item.The purpose of testing is to verify and validate a software and to find the defects present in a software. The purpose of finding those problems is to get them fixed.

* Verification is the checking or we can say the testing of software for consistency and conformance by evaluating the results against pre-specified requirements.
* Validation looks at the systems correctness, i.e. the process of checking that what has been specified is what the user actually wanted.
* Defect is a variance between the expected and actual result. The defect’s ultimate source may be traced to a fault introduced in the specification, design, or development (coding) phases.

**Types of Software Testing**

Testing is done in every stage of software development life cycle, but the testing done at each level of software development is different in nature and has different objectives. There are different types of testing, such as stress testing, volume testing, configuration testing, compatibility testing, recovery testing, maintenance testing, documentation testing, and usability testing. Software testing are mainlyof following types [1]

**Unit Testing**

Unit testing is done at the lowest level. It tests the basic unit of software, that is the smallest testable piece of software. The individual component or unit of a program are tested in unit testing. Unit testing are of two types.

**Black box testing**: This is also known as functional testing , where the test cases are designed based on input output values only. There are many types of Black Box Testing but following are the prominent ones.

**- Equivalence class partitioning**: In this approach, the domain of input values to a program is divided into a set of equivalence classes. e.g. Consider a software program that computes whether an integer number is even or not that is in the range of 0 to 10. Determine the equivalence class test suite. There are three equivalence classes for this program. - The set of negative integer - The integers in the range 0 to 10 - The integer larger than 10

**- Boundary value analysis :** In this approach, while designing the test cases, the values at boundaries of different equivalence classes are taken into consideration. e.g. In the above given example as in equivalence class partitioning, a boundary values based test suite is { 0, -1, 10, 11 }

**White box testing:** It is also known as structural testing. In this testing, test cases are designed on the basis of examination of the code.This testing is performed based on the knowledge of how the system is implemented. It includes analyzing data flow, control flow, information flow, coding practices, exception and error handling within the system, to test the intended and unintended software behavior. White box testing can be performed to validate whether code implementation.

**System Testing**

System testing tends to affirm the end-to-end quality of the entire system. System testing is often based on the functional / requirement specification of the system. Non-functional quality attributes, such as reliability, security, and maintainability are also checked. There are three types of system testing.

**CASE STUDY**

The SE VLabs Institute has been recently setup to provide state-of-the-art research facilities in the field of Software Engineering. Apart from research scholars (students) and professors, it also includes quite a large number of employees who work on different projects undertaken by the institution.

As the size and capacity of the institute is increasing with the time, it has been proposed to develop a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.

The final deliverable would a web application (using the recent HTML 5), which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (eg., passwords) is stored in plain text.

As already discussed under the theory section, test case preparation could begin right after requirements identification stage. It is desirable (and advisable) to create a Requirements Traceability Matrix (RTM) showing a mapping from individual requirement to test case(s). A simplified form of the RTM is shown in table 1 (the numbers shown in this table are arbitrary; not specific to LIS).

| Table 1: A simplified mapping from requirements to test cases | |
| --- | --- |
| **Requirement #** | **Test Case #** |
| R1 | TC1 |
| R2 | TC2, TC3, TC4 |
| R3 | TC5 |
| R4 | TC6 |