# Unit 2 – Chapter 5

# Requirements

## Objectives

The objective of this part of the unit is to understand the need for the requirement analysis. At the end of this chapter, the student will be able to:

- Understand the importance of requirement analysis
- Understand the classes of requirements
- Understand the issues with gathering requirements
- Understand the process of requirements engineering

## 5.1 Importance of Requirement Analysis

Understanding requirements is key to developing a good software project. Requirements are the things that a software developer should discover before starting to build a software product. Unless the requirements are clear and well-specified, a software product cannot be developed, and the effort spent on the development will be a waste. The functions of a software product must match the requirements. If the requirements are captured incorrectly, the software project would fail. Also, if the software product is modified to incorporate lately understood requirements, then the effort spent up until then is wasted and hence the cost of the project becomes very high.

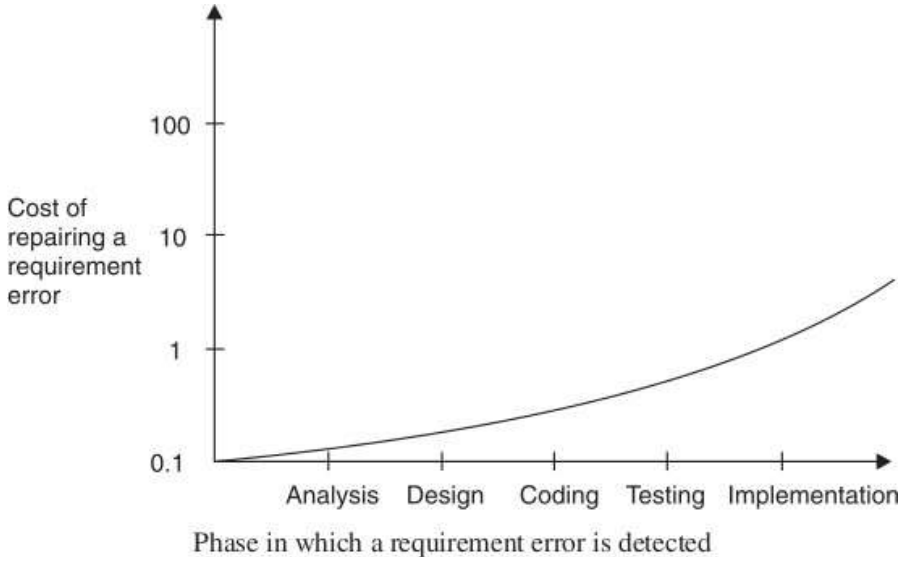



Figure 5.1 Relative cost to repair a requirement error

Figure 5.1 shows the relative cost to repair a requirement error and indicates how it varies when detected at various development phases. The later we find the requirement error in an SDLC phase, the more costly it becomes.

## 5.1 Types of Requirements

The requirements for a system are the descriptions of the services that a system should provide and the constraints on its operation. These requirements reflect the needs of customers for a system that serves a certain purpose. The process of findingout, analyzing, documenting, and checking these services and constraints is called requirements engineering.

User requirements is the term used to specify the high-level abstract requirements and system requirements specify the detailed description of what the system should do.

### 5.1.1 User Requirements

User requirements specify what services the system is expected to provide to system users and the constraints under which it must operate. The user requirements may vary from broad statements of the system features required to detailed, precise descriptions of the system functionality. The user needs are expressed in the languageof the user. A user requirement may be expanded into several system requirements. User requirements are very general.

### 5.1.2 System Requirements

System requirements are more detailed descriptions of the software system's functions, services, and operational constraints. The system requirements document (sometimes called a functional specification) should define exactly what is to beimplemented. It may be part of the contract between the system buyer and thesoftware developers. The system requirements provide more specific information aboutthe services and functions of the system that is to be implemented.

### 5.1.3 Stakeholders

You need to write requirements at different levels of detail because different types of readers use them in different ways. The readers of the user requirements are not usually concerned with how the system will be implemented and maybe managers who are not interested in the detailed facilities of the system. The readers of the system requirements need to know more precisely what the system will do because they are concerned with how it will support the business processes or because they are involved in the system implementation. Different types of readers are the stakeholders of the system. System stakeholders include anyone who is affected by the system in some way and so anyone who has a legitimate interest in it.

### Case Study: Patient Management Software System

The stakeholders for a Patient Management System would be -

1. Patients whose information is recorded in the system and relatives of these patients.
2. Doctors who are responsible for assessing and treating patients.
3. Nurses who coordinate the consultations with doctors and administer some treatments.
4. Medical receptionists who manage patients' appointments.
5. IT staff who are responsible for installing and maintaining the system.
6. Medical records staff who are responsible for ensuring that system information can be maintained and preserved and that record-keeping procedures have been properly implemented.

Examples of User Requirements for this system could be -

1. The system should generate monthly management reports showing the cost of the drugs prescribed during that month.

Examples of System Requirements for this system could be -

1. On the last working day of the month, a summary of the drugs prescribed, theircosts shall be generated.
2. The system shall generate the report for printing after 17:30 on the last workingday.
3. If drugs are available in different dose units (e.g.10mg, 20mg, etc.) separatereports shall be generated for each dose unit.
4. Access to drug cost reports shall be restricted to authorized users as listed on the access control list.

The user and system requirements can be further divided into Functional and Non-Functional requirements.

## 5.2 Functional and Non-Functional Requirements

Requirements are generally split into two types:

Functional and Non-functional requirements.

### 5.2.1 Functional Requirements

These are the requirements that the end-user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. The functional requirements of the system should clearly describe each functionality that the system would support along with the corresponding input and output data set. They are the requirements stated by the user which one can see directly in the final product. They are generally described as use cases in the requirement specification document.

For example,

1. Authentication of the user whenever they log into the system.
2. System shutdown in case of a Cyber-attack.
3. A verification email is sent to the user whenever they register for the first time onsome software system.

### 5.2.2 Non-Functional Requirements

These are the quality constraints that the system must satisfy according to theproject contract. These requirements are not related directly to any function provided by the system. They are also called non-behavioral requirements. Non-functional requirements usually address aspects concerning external interfaces, user interfaces, maintainability, portability, usability, the maximum number of concurrent users, timing, and throughput (transactions per second, etc.). The non-functional requirements can be critical in the sense that any failure to achieve some of these requirements can be considered as a failure and make the software unacceptable by the customer.

For example,

1. Emails should be sent with a latency of no greater than 12 hours.
2. The processing of each request should be done within 10 seconds
3. The site should load in 3 seconds when the number of simultaneous users is > 10000

Non-functional requirements can be classified broadly as Product requirements, Organizational requirements, and External requirements.

**Product requirements** specify the constraints on the behavior of the product. For example, execution speed or reliability of the product.

**Organizational requirements** are a consequence of organizational policies and procedures e.g., process standards used, implementation requirements, etc.

**External requirements** arise from factors that are external to the system and its development process e.g., interoperability requirements, legislative requirements, etc.

Figure 5.2 below shows various types of Non-Functional requirements.

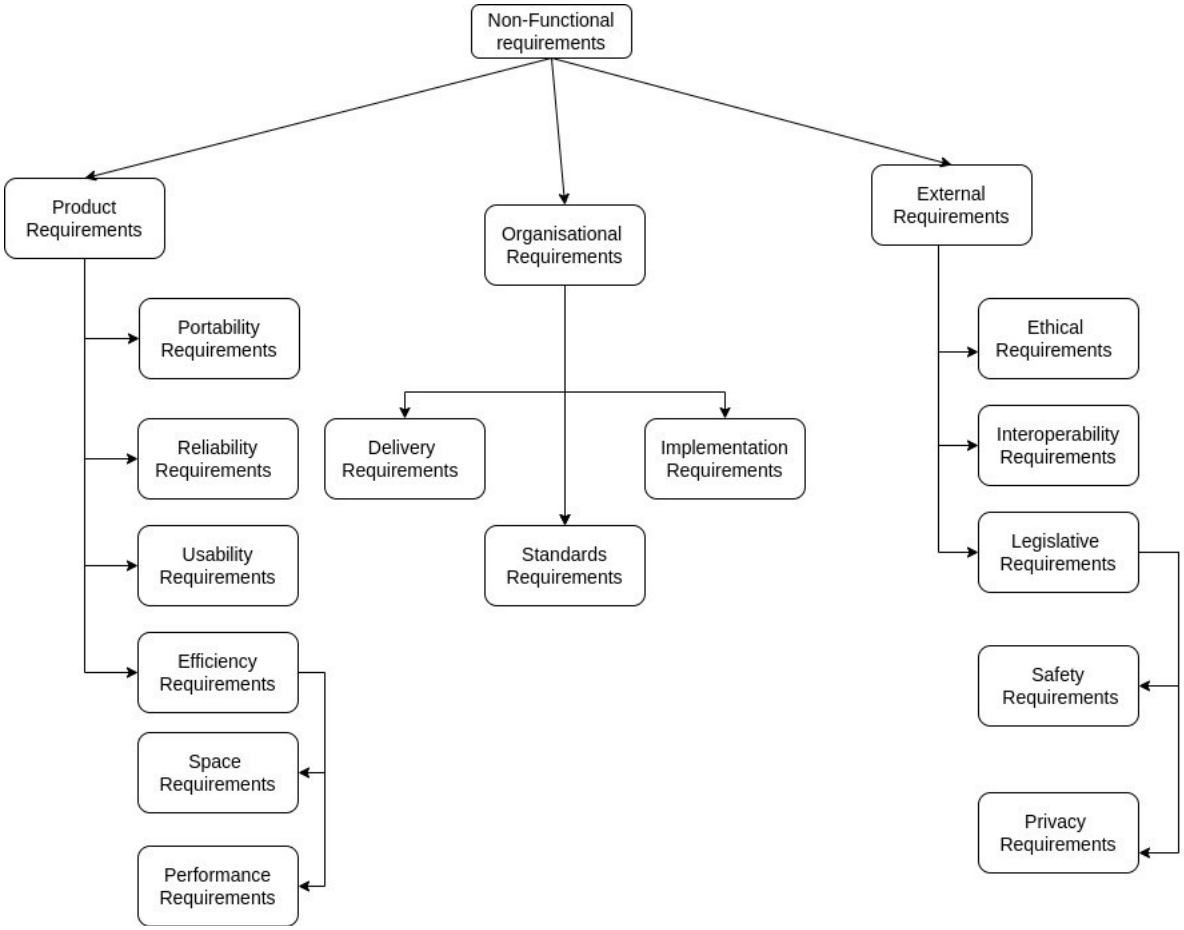



Figure 5.2 Types of Non-Functional Requirements

Non-functional Requirements are difficult to test and verify. Hence wherever possible non-functional requirements should be written quantitatively so that they can be objectively tested. Table 5.1 shows metrics that can be used to specify non-functional system requirements.

| Property | Measure |
|---|---|
| Speed | Processed transactions/second |
| Size | User/event response time Screen refresh time, Mbytes, Number of ROM chips |
| Ease of use | Training time |
| Reliability | Number of help frames, Mean time to failure<br>Probability of unavailability Rate of failure occurrence Availability |
| Robustness | Time to restart after failure |

| | |
|---|---|
| Portability | Percentage of events causing failure Probability of data corruption on failure, Percentage of target-dependent statements Number of target systems |

Table 5.1 Metrics for specifying non-functional requirements

## 5.3 Requirements Engineering Process

Requirements Engineering is the process of defining, documenting, and maintaining therequirements. It is a process of gathering and defining services provided by the system.Requirements Engineering Process consists of the following main activities:

1. **Feasibility Study:** A feasibility study is a compact estimate of whether the system to be developed is satisfying the customer's needs within the existing budget and technology conditions. At the end of this activity, a feasibility report is generated that mainly determines whether the system fulfills the objectives of the organization. The feasibility report must be brief and economical, and it should decide whether the further detailed requirement analysis can be conducted or not.

2. **Requirement elicitation and analysis**: It is related to the various ways used to gather information about the project domain and requirements. The various sources of domain knowledge include customers, business manuals, the existingsoftware of the same type, standards, and other stakeholders of the project. The techniques used for requirements elicitation include interviews, brainstorming, task analysis, prototyping, etc

3. **Requirement specification:** This activity is used to produce formal software, requirement models. All the requirements including the functional as well as the non-functional requirements and the constraints are specified by these models in totality. The models used at this stage include ER diagrams, data flow diagrams (DFDs), data dictionaries, etc.

4. **Requirement verification and validation:** Verification refers to the set of tasks that ensures that the software correctly implements a specific function. Validation refers to a different set of tasks that ensures that the software that has been built is traceable to customer requirements. Reviews, test cases, etc. are some of the methods used for this.

5. **Requirements management:** Requirement management is the process of analyzing, documenting, tracking, prioritizing, and agreeing on the requirement and controlling the communication to relevant stakeholders. This stage takes care of the changing nature of requirements.

Figure 5.3 below shows the requirements engineering process.

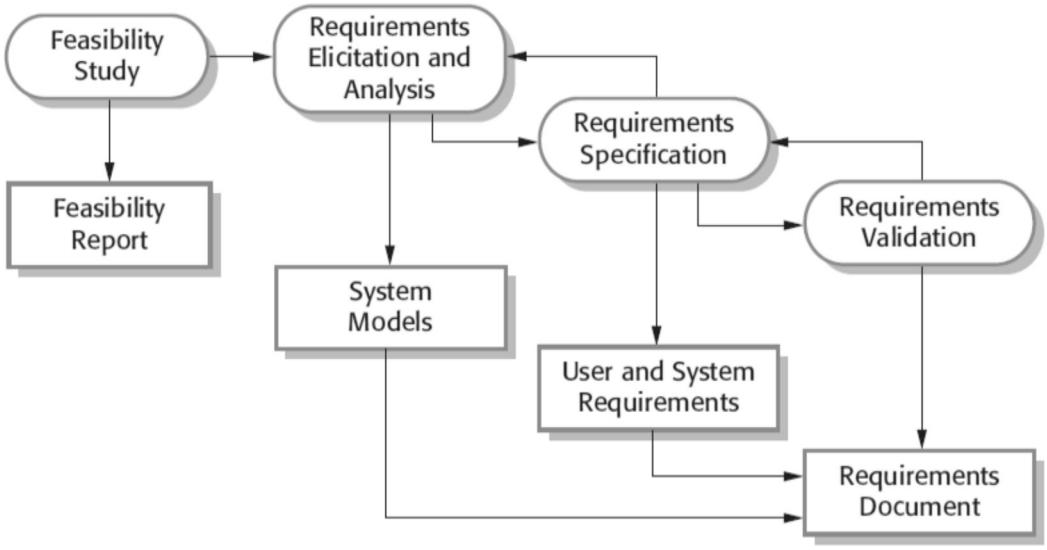



Figure 5.3 The requirement engineering process

## 5.4 Feasibility Study:

As shown in Figure 5.3, the feasibility study takes place in the early stage of the RE process. This study is focused on answering the questions such as whether the objectives of the organization are met by the system or whether the system can be implemented by the latest technology, can it be clubbed with the other parts of the system that are already in use, etc. Thus, a conclusion can be drawn whether the proposed system can be continued or stopped at the end of the feasibility study.
The different types of feasibility study include:

**Technical Feasibility** – In this feasibility study, the technical details related to the system development are analyzed. Whether the product can be developed using existing technology or not, whether it is possible to do the maintenance of the selected

technology is determined. Details of the up-gradation of the technology in the current system are reported in this type of study.

**Operational Feasibility** – In this report whether the product can be operated or used easily or not is determined. Also, whether the operations in the product can be maintained after its deployment is analyzed. Details of the operating skill required to work on the newly developed, limitations of the current system, and how the proposed system will help overcome these pitfalls are checked in this study.

**Legal Feasibility** – In the legal feasibility study, the system to be developed is checked whether it can fulfill the legal and ethical needs such as copyright, license, security laws, etc.

**Schedule Feasibility** – In this feasibility study, importance is given to the deadlines or time management to complete the given system development. For example, how many numbers of teams are assigned for project development and whether their schedule can complete the project in time or not is checked.

**Activities in Feasibility Study:**

1. **Introduction**: In this step, the statement of the problem is defined. Here, all the background details of the projects are described. For example, if a company is in increasing the sales of say supermarket products, then the description of the project will be “This company is providing a site for improving your sales in online /offline form”. The different functions related to the new project are listed in this step.
2. **Summary & Recommendations**: In this step, who are the customers, competitors, and the general market audience is summarized. The technical details like what type of machines are required and for how long time to develop a project are recommended.

3. **Alternatives**: Is it good to stick with the present system or does it needs some modification is the main subject studied in this step. The different possible solutions for the existing project are discussed. This will help to get different ideas for implementing the project in the future.

4. **System Description**: The function of the newly proposed solution, how better it is than the existing one for doing its job are mentioned. The other details about the project such as how many teams of people are necessary, what modifications are required in equipment, software, maintenance of the machines, etc. are described in this activity.

5. **Cost-Benefit Analysis**: In this activity, the cost of the project, initial investment, expected income, and balance sheet showing cash flow is analyzed.

6. **Evaluation of Risks**: In this step, the different risks such as is worth spending time, money, and efforts to build the new project are evaluated. For example, a bookshop needs to expand its sale internationally, then the shop owner must think about the different legal risks, financial risks or can he satisfy millions of customers worldwide, etc. At the end of this step, a cautious decision with calculated risk is taken to start a new project.

7. **Conclusion:** The feasibility study ends with a conclusion. Generally, it is a binary decision of Yes/No for the new project. This is the outcome that either approves or rejects the proposed project.

## 5.6 Requirement Elicitation and Analysis

The aims of the requirements elicitation process are to understand the work that stakeholders do and how they might use a new system to help support that work. During requirements elicitation, software engineers work with stakeholders to findout about the application domain, work activities, the services, and system features that stakeholders want, the required performance of the system, hardware constraints,and so on.

### 5.6.1 Requirements Elicitation Process

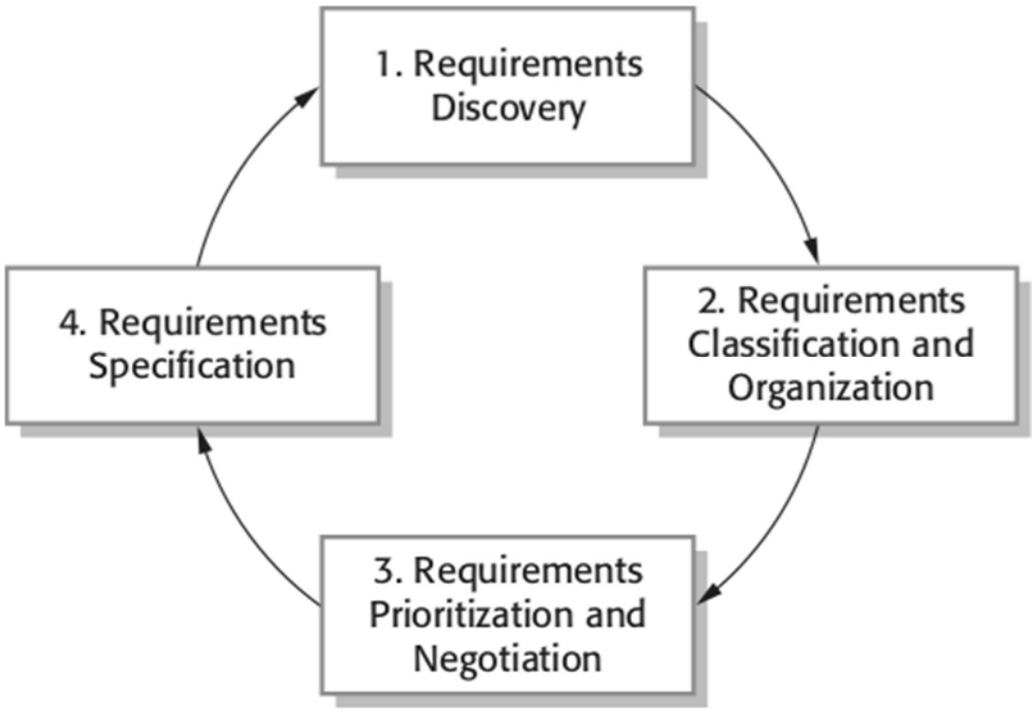



Figure 5.4 Requirements elicitation and analysis process

The stages for requirements elicitation and analysis include -

1. Requirements discovery: The process of gathering information about the required and existing systems and figuring out the user and system requirements from this information. This stage involves interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
2. Requirements classification and organization: This activity takes the unstructured collection of requirements, groups related requirements, and organizes them into coherent clusters.
3. Requirements prioritization and negotiations: Generally when many stakeholders are involved there are conflicts in requirements. This activity is concerned with prioritizing requirements and resolving requirements conflicts.

4. Requirements documentation: Requirements are documented and an early draftof the software requirements document may be produced at this stage. You must use simple language and diagrams to describe the requirements. This makes it possible for stakeholders to understand and comment on these requirements.

### 5.6.2 Requirements Elicitation Techniques

Various techniques can be used to gather requirements. They generally involve meeting with stakeholders. This information can be further supplemented with knowledge of existing systems and their usage and documentation about them. The main approaches to requirements elicitation are –

1. **Interviewing**: Interviews are good for getting an overall understanding of what stakeholders do, how they might interact with the new system and the difficulties that they face with current systems.

2. **Observation or Ethnography**: Ethnography is an observational technique that can be used to understand operational processes and help derive requirements for software to support these processes. An analyst immerses himself or herself in the working environment where the system will be used. The day-to-day work is observed, and notes are made of the actual tasks in which participants are involved. The value of ethnography is that it helps discover implicit system requirements that reflect the actual ways that people work, rather than the formal processes defined by the organization.

3. **Scenarios:** These are real-life examples of how users would interact with the system. Each scenario usually covers one or a small number of possible interactions. Different forms of scenarios are developed, and they provide different types of information at different levels of detail about the system.

4. **Use cases**: Use cases are a way of describing interactions between users and a system using a graphical model and structured text.

5. **Questionnaires:** A questionnaire is a list of questions about the project requirements. Questionnaires are used to find information from multiple sources such as team information, work items, usage of existing software, and its drawbacks (if any). The questionnaire also includes queries about security weakness, maintaining source code, and analyzing bug reports. The use of questionnaires helps in mainly three disciplines: development, testing, and program management. Some of the examples of questions in software development may be as follows:

    a) What is the goal of developing this system? Who are the users and which goal is the most important?
    b) Will this system help your office to be more efficient? How?
    c) This project should be developed as a desktop application/Database/Mobile App/Web service/Client-server type?
    d) What is the time limit to make this project functional? (days/months/years)
    e) How many IT staff members can be involved? (Depending on the budget)
    f) How many lines of code may be developed to complete the project?

.

6. **Survey**: Survey document captures responses about the project from its users. It includes questions that lead to answers such as agree/disagree, strong/weak, Yes/No, etc. Here customer's response is taken as an opinion, and it is further analyzed and distributed in different actions in software development. Survey questions may be closed or open-ended. For example, "do you want this software only on desktop or web?", is a closed question, whereas "what different features do you want in this product?", is an open-ended question.

7. **Existing manuals and documentation**: These documents provide the details about the current system in use. The documentation includes technical, operational, and

financial details. The manual is an important part of a system, and it includes many operational details such as installation procedure, troubleshooting, etc.

8. **Group Discussion:** In a group discussion, people from similar backgrounds or experiences together to understand the problem domain. Different challenges and issues that are encountered in handling the project development are identified. The discussion is about clarity of customer needs, understanding the system boundaries, conflicts in requirements, division of work, budget, the timeline for the work, etc.
9. **Prototyping:** A prototype is a sample/mock model that expresses the ideas of the requirements in a better way. For example, A virtual digital model of a building before construction or a simulation of a front end of software to be launched covers many requirements given by the customer.
10. **Brainstorming:** It is a group activity where new ideas get generated by sharing views about the project requirements by different people. A group leader handles the difference of opinions among the members and sees that the idea is not influenced by only a few members. At the end of brainstorming, a document containing a detailed list of requirements is prepared.

### 5.6.3 Barriers to Eliciting User Requirements

Getting requirements from users is one of the most difficult tasks in the software development life cycle. Some of the major reasons are as follows -

1. Stakeholders often don't know what they want from a computer system. They may find it difficult to articulate what they want the system to do. They may make unrealistic demands because they don't know what is and isn't feasible.
2. The variety and complexity of information requirements. Software normally serves a variety of users; each user focuses on different issues associated with the overall problem addressed by the software. Each has a separate view of the problem.
3. The complex patterns of interaction among users and analysts/developers in

defining requirements. A lack of communication between the users and analysts/developers can cause issues while gathering requirements. Even characteristics like the analysts' interpersonal skills and personality can cause issues.

4. The unwillingness of some users to provide requirements. Users sometimes do not like to disclose information due to personal reasons. For example, they may feeluncomfortable with a new system trying to replace part of their jobs in the organization.

5. Constraints on humans as specifiers of information requirements. Human beings are not very good at information processing, they generally have a bias in the selection and use of data.

## 5.7 Requirements Specification

The requirement specification is the process of writing down the user and system requirements in a requirements document. Both user and system requirements should be documented, completely, and unambiguously. User requirements are generally written in simple natural language and are supplemented with diagrams and tables in a requirements document. The requirements document should not contain any details of the system design, it should be easy to understand. System requirements may also be documented in natural language, but we often use other mathematical models and graphical notations to bring clarity.

### 5.7.1 Ways of Writing System Requirements

System requirements are the expanded versions of user requirements, that the software engineers use for system design. They add detail and explain how the system should provide the user requirements. They may be used as part of the contract for the implementation of the system and should therefore be a complete and detailed specification of the whole system.

The system requirements can be specified using -

1. Natural Language Specification: Natural language is expressive, intuitive, and universal, but it could also be vague and ambiguous. However, it is still used widely. If certain formatting is followed, it could be used to capture the essence of the requirement. For example – The system shall measure the blood sugar and deliver insulin if required, every 10 minutes. (Changes in blood sugar are relatively slow, so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.)

2. Structured Specification: Structured natural language is a way of writing system requirements where requirements are written in a standard way rather than as free-form text. An example of an Insulin Pump control software structure specification is shown in Figure 5.5.

***Insulin Pump/Control Software/SRS/3.3.2***

| | |
|---|---|
| **Function** | Compute insulin dose: Safe sugar level. |
| **Description** | Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units. |
| **Inputs** | Current sugar reading (r2), the previous two readings (r0 and r1). |
| **Source** | Current sugar reading from sensor. Other readings from memory. |
| **Outputs** | CompDose—the dose in insulin to be delivered. |
| **Destination** | Main control loop. |
| **Action:** | CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered. (see Figure 4.14) |
| **Requires** | Two previous readings so that the rate of change of sugar level can be computed. |
| **Precondition** | The insulin reservoir contains at least the maximum allowed single dose of insulin. |
| **Postcondition** | r0 is replaced by r1 then r1 is replaced by r2. |
| **Side effects** | None. |

Figure 5.5 Sample of a structured specification

3. Graphical Notations: Graphical models, supplemented by text annotations, are used to define the functional requirements for the system. UML (unified modeling language) use case and sequence diagrams are commonly used. Usecases are a way of describing interactions between users and a system using a graphical model and structured text. Use cases are documented using

a high-level use case diagram. The set of use cases represents all the possible interactions that will be described in the system requirements. Actors in the process, who may be human or other systems, are represented as stick figures. Each class of interaction is represented as a named ellipse. Lines link the actors with the interaction. Optionally, arrowheads may be added to lines to show how the interaction is initiated. For example, a use-case diagram for medical applications is shown in Figure: 5.6.

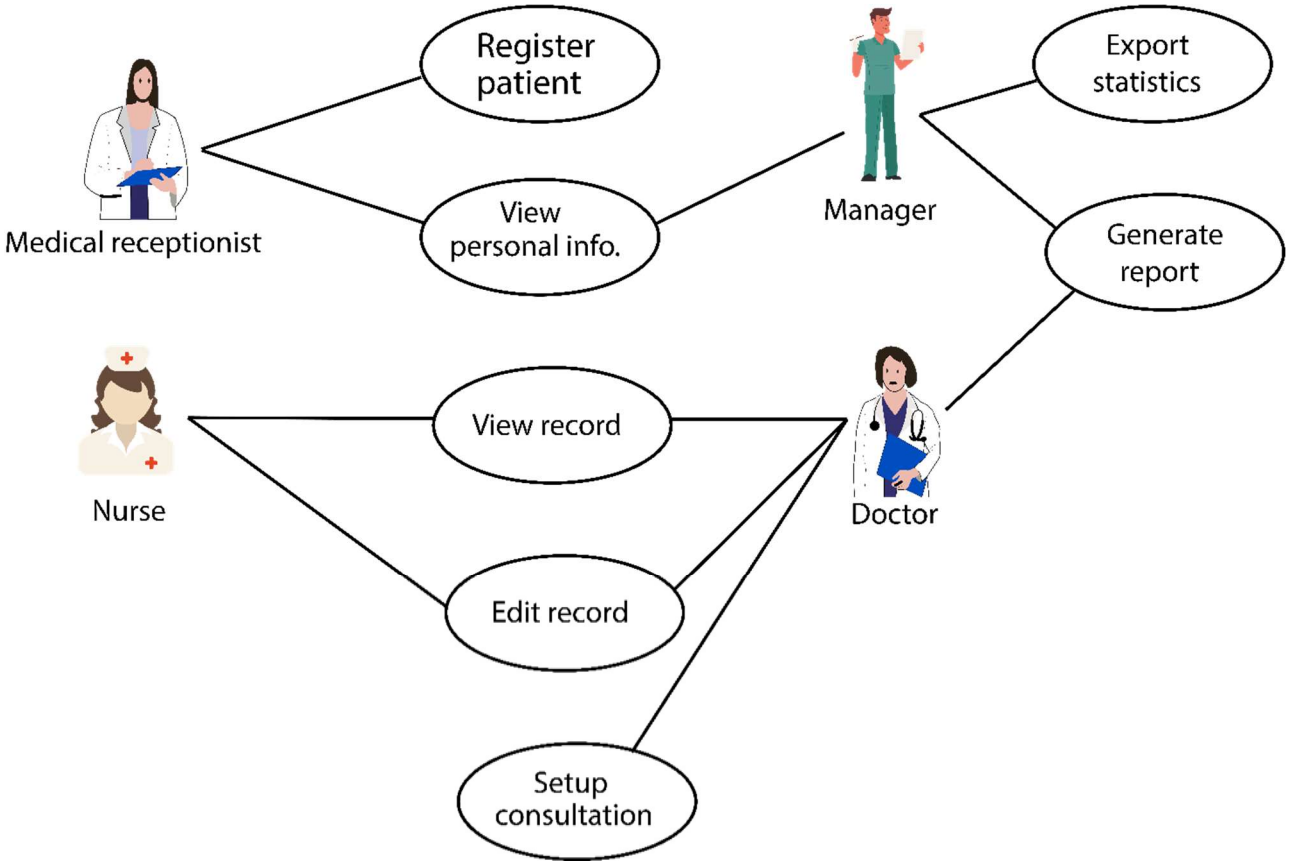



Figure 5.6 Sample Use-Case Diagram

4. Mathematical Specifications: These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that

it represents what they want, and they are reluctant to accept it as a system contract.

## 5.8 The Software Requirements Document and SRS Standards

After the analyst has gathered all the required information regarding the software to be developed and has removed all incompleteness and inconsistencies from the specification, they start to systematically organize the requirements in the form of a Software Requirements Specification (SRS) document.

The SRS is the official statement of what is required of the software system. It includes a definition of user requirements and a specification of the system requirements.

Among all the documents produced during a software development life cycle, the SRS document is probably the most important and is the toughest to write.

SRS contains functional requirements, non-functional requirements, and goals of implementation.

There are many users of the SRS document.

1. Customers / Users / Clients specify the requirements and read them to check that they meet their needs. Customers can also specify changes in requirements.
2. Managers use the requirements document to plan a bid for the system and toplan the system development process.
3. System engineers use the requirements to understand the system that must be developed.
4. System test engineers use the requirements document to develop validation tests for the system.
5. System maintenance engineers use the requirements document to understandthe system and the relationships between its parts.

### 5.8.1 Characteristics of a good SRS

Following are the characteristics of a good SRS:

1. Correctness: User review is used to ensure the correctness of requirements stated in the SRS. SRS is said to be correct if it covers all the requirements that are expected from the system.
2. Completeness: Completeness of SRS indicates every sense of completion including the numbering of all the pages, resolving the to be determined parts toas much extent as possible as well as covering all the functional and non-functional requirements properly.
3. Consistency: Requirements in SRS are said to be consistent if there are no conflicts between any set of requirements. Examples of conflict include differences in terminologies used at separate places, logical conflicts like the time of report generation, etc.
4. Concise: A good SRS is a short and concise document.
5. Unambiguous: SRS is said to be unambiguous if all the requirements stated have only one interpretation.
6. Structured and easy to understand by the customers.
7. Black-box view: SRS should not include any design or implementation details. It should have a black-box view of the system as a whole and just describe the requirements.
8. Verifiable: SRS is verifiable if there exists a specific technique to quantifiable measure the extent to which every requirement is met by the system.
9. Modifiable/Maintainable: SRS should be made as modifiable as possible and should be capable of easily accepting changes to the system to some extent.
10. Traceable: One should be able to trace a requirement to the design component andthen to code segment in the program. Similarly, one should be able to trace a requirement to the corresponding test cases.
11. Testable: SRS should be written in such a way that it is easy to generate test cases and test plans from the document.

### 5.8.2 Characteristics of a bad SRS

1. Over-specification: If the SRS document is too detailed, it would be difficult to understand and use.
2. Forward references: Requirements should not contain any forward references. First, all terms and prerequisites should be made clear, and only then should requirements be specified.
3. Wishful thinking: Requirements in the SRS should be implementable.
4. Noise: Unnecessary details should not be included in the SRS.

### 5.8.3 Standard Software Requirements Specification

The standard IEEE SRS format represents a generic requirements document. This document works as a standard for defining the requirements of a specific system. It is a very good starting point for defining the standard for detailed specific requirements. The general structure for a standard IEEE SRS format is as given below.

1. Introduction
   1.1 Purpose
   1.2 Project Scope
   1.3 Definitions, acronyms, and abbreviations
   1.4 References
   1.5 Overview
2. Overall Description
   2.1 Product perspective
   2.2 Product features
   2.3 User classes
   2.4 Operating environment
   2.5 Design and implementation constraints:

2.6 User documentation

2.7 Assumptions and Dependencies

3. Specific Requirements

3.1 Functional requirements 1

3.2 Functional requirements 2. …

4. External Interface Requirements

4.1 User interfaces

4.2 Hardware interfaces

4.3 Software interfaces

4.4 Communication interfaces

5. Other non-functional Requirements

5.1 Performance Requirements

5.2 Safety Requirement

5.3 Security Requirements

6. Other Requirements

7. Appendix A: Glossary

8. Appendix B: Analysis Models

9. Appendix C: To Be Determined List

1. **Introduction:** This section should describe why the software is needed. It should briefly describe the software's function and explain how it will work with other systems. It should also describe how the software system fits into the overall business or objectives of the organization for whom the software system is being made.

**1.1 Purpose**: This section should describe the purpose of the SRS. It should specify the intended audience for the SRS.

**1.2 Project scope**: This section should briefly describe the overall context within which the software is being developed. It should explain what the software woulddo and describe its applications.

**1.3 Definitions, acronyms, and abbreviations**: This section should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS.

**1.4 References**: This section should provide a complete list of all documents referenced in the SRS. It should specify all sources from where the referencesare obtained.

**1.5 Overview**: This section should describe what the rest of the SRS contains and explain how it is organized.

**2 Overall description:** This section of the SRS should describe the general factors thataffect the product and its requirements. This section does not state-specific requirements. Instead, it provides a background for those requirements and makes them easier to understand.

**2.1 Product perspective**: This section needs to briefly state whether the software is intended to be a replacement for a certain existing system, or it is new software. If the software being developed would be used as a component ofa larger system, a simple schematic diagram can be given to show the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.

**2.2 Product features**: This section should summarize the major ways in which the software would be used. Only a summary should be presented here.

**2.3 User classes**: Various user classes that are expected to use this software are identified and described here. The different classes of users are identified by the types of functionalities that they are expected to invoke or their levels of expertise in using computers.

**2.4 Operating environment:** This section should discuss in some detail the hardware platform on which the software would run, the operating system, and other application software with which the developed software would interact.

**2.5 Design and implementation constraints**: In this section, the different constraints on the design and implementation are discussed. These might include corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; specific programming language to be used; specific communication protocols to be used; security considerations; design conventions or programming standards.

**2.6 User documentation**: This section should list out the types of user documentation, such as user manuals, online help, and trouble-shooting manuals that will be delivered to the customer along with the software.

**2.7 Assumptions and Dependencies**: This section should list any assumed factors that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints.

## 3 Specific Requirements

**3.1 Functional requirements:** This section can classify the functionalities either based on the specific functionalities invoked by different users, or the functionalities that are available in different modes. The functions describe the system behavior based on the input and output. All the requirements for the feature must be described completely. Functional requirements can be captured with use cases.

## 4 External Interface Requirements

**4.1 User interfaces**: This section should describe a high-level description of various interfaces and principles to be followed. The user interface description may include sample screen images, any GUI standards or style guides that are to be followed, screen layout constraints, standard push buttons (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, etc. The details of the user interface design should be documented in a separate user

interface specification document.

**4.2 Hardware interfaces**: This section should describe the interface between the software and the hardware components of the system. This section may include the description of the supported device types, the nature of the data and control interactions between the software and the hardware, and the communication protocols to be used.

**4.3 Software interfaces**: This section should describe the connections between this software and other specific software components, including databases, operating systems, tools, libraries, integrated commercial components, etc. Identify the data items that would be input to the software and the data that would be output should be identified and the purpose of each should be described.

**4.4 Communications interfaces**: This section should describe the requirements associated with any type of communications required by the software, such as e-mail, web access, network server communications protocols, etc. This section should define any pertinent message formatting to be used. It should also identify any communication standards that will be used, such as TCP sockets, FTP, HTTP, or HTTPS. Specify any communication security or encryption issues that may be relevant, and the data transfer rates, and synchronization mechanisms.

## 5 Other Non-Functional Requirements

This section should describe the non-functional requirements other than the design and implementation constraints. It does not discuss the external interface requirements that have been described in earlier sections.

**5.1 Performance requirements**: Aspects such as the number of transactions to be completed per second should be specified here. Some performance requirements may be specific to individual functional requirements or features. These should also be specified here.

**5.2 Safety requirements**: Those requirements that are concerned with possible lossor damage that could result from the use of the software are specified here. For example, recovery after a power failure, handling software, and hardware

failures, etc. may be documented here.

**5.3 Security requirements**: This section should specify any requirements regarding security or privacy requirements on data used or created by the software. Any user identity authentication requirements should be described here. It should also refer to any external policies or regulations concerning security issues. Any security or privacy certifications that must be satisfied should be mentioned in this section.

6 **Other Requirements:** Define any other requirements which are not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on.

7 **Appendix A:** Glossary. This defines the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.

8 **Appendix B:** This section describes analysis models optionally, it may include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.

9 **Appendix C:** This list is known as the To Be Determined List (TBD). The TBD list includes the references that are yet to be determined or will be determined in the future. This list is given by the client. The details of this document conclude what is the need for software to be developed and what business goals are fulfilled.

## 5.9 Case Study of the SRS for a Real-Time System

Let us consider SRS for a flight management project. The relevant SRS document in IEEE structure format can be given as follows:

### Table of Contents

2. **Overall Description**

    2.1 Product Perspective
    2.2 Product Features
    2.3 User Classes and Characteristics
    2.4 Operating Environment
    2.5 Design and Implementation Constraints
    2.6 User Documentation
    2.7 Assumptions and Dependencies

3. **System Features**

    3.1 Functional Requirement

4. **External Interface requirements**

    4.1 User Interfaces
    4.2 Hardware Interfaces
    4.3 Software Interfaces
    4.4 Communication Interfaces

5. **Nonfunctional Requirements**

    5.1 Performance requirements
    5.2 Safety Requirement
    5.3 Security Requirements
    5.4 Software Quality Attribute

## 1. Introduction

### 1.1 Purpose

The purpose of this SRS document is to describe the proposed online flight management system. This document will help the passengers to manage their travel. It also helps the airline system to coordinate and control passenger and flights travel.

### 1.2 Project Scope

The purpose of the online flight management system is to ease flight management and to create a convenient and easy-to-use application for passengers, trying to buy airline

tickets. The system is based on a relational database with its flight management and reservation functions. We will have a database server supporting hundreds of major cities around the world as well as thousands of flights by various airline companies. Above all, we hope to provide a comfortable user experience along with the best pricing available.

### 1.3 Definitions, Acronyms, and Abbreviations

This document uses the following conventions.

| | |
|---|---|
| DB | Database |
| DDB | Distributed Database |
| ER | Entity Relationship |

### 1.4 References

- https://krazytech.com/projects
- Fundamentals of database systems by Ramez Elmarsi and Shamkant Navathe

### 1.5 Overview

We provide the overall description of the project in Section 2. The specific requirements are listed in Section 3. Sections 4 and 5 give the external interface requirements and non-functional requirements respectively.

## 2. Overall Description

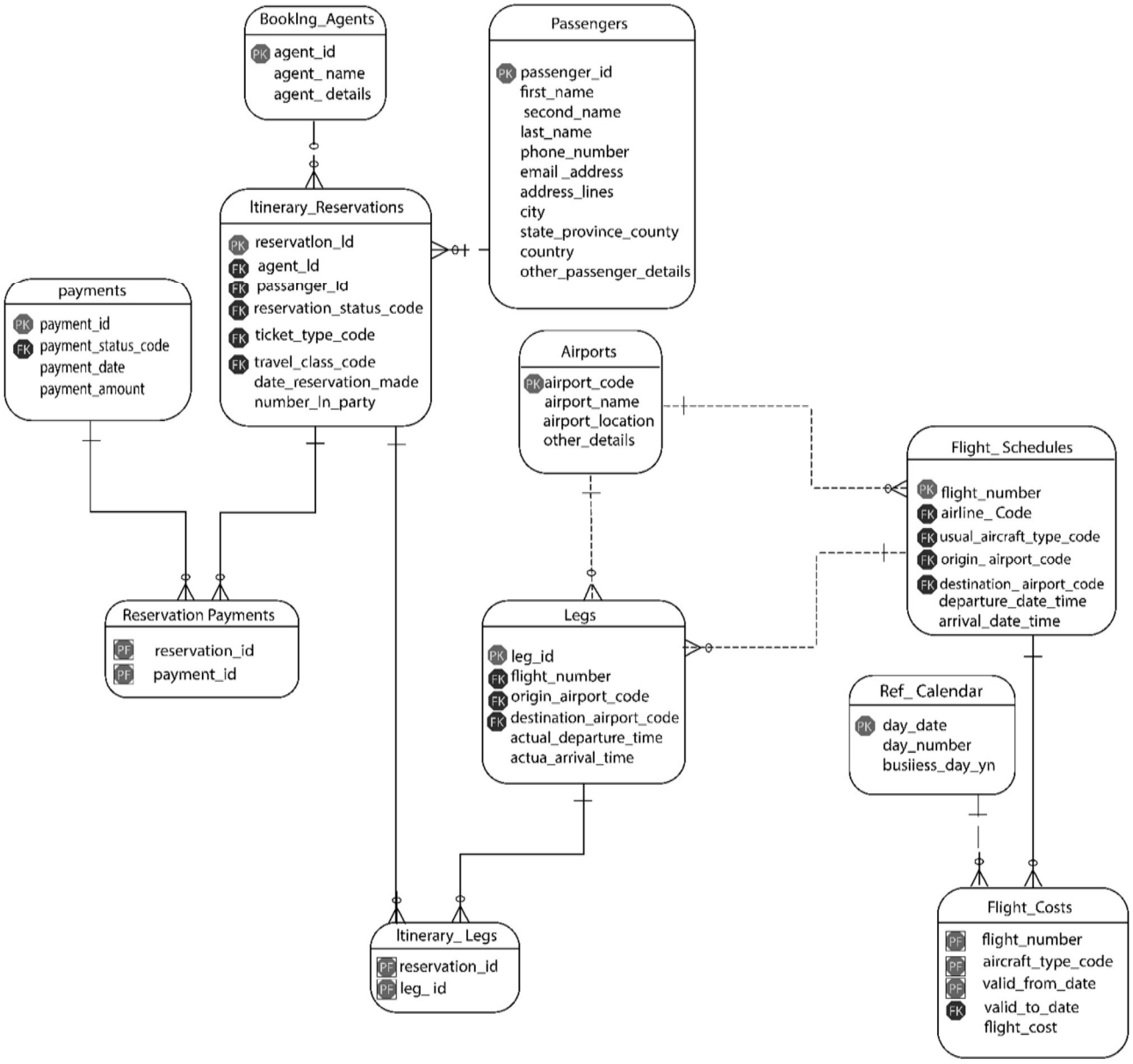
### 2.1 Product Perspective

The software application is being designed to replace the legacy application. The legacy software was not suitable to serve the customers and internal users in the latest technological developments facilitated by Internet accessibility.

The earlier system was restricted to internal staff and travel agents. The new software is meant to provide access to customers to manage their travel requirements by themselves. A distributed airline DB system stores information such as flight details, customer, their reservation description, etc.

## 2.2 Product Features

The major features of the flight management system are as shown in the diagram below.




*The diagram shows the layout of the airline database system*

## 2.3 User Classes

Users of the system should be able to retrieve flight information between two given cities with the given date/time of travel from the database. A route from city A to city B is a sequence of connecting flights from A to B such that: a) there are at most two connecting stops, excluding the starting city and destination city of the trip. b) the connecting time is between one to two hours. The system will support two types of user privileges, Customer, and Employee. Customers will have access to customer functions, and the employees will have access to both customer and flight management functions.

The customer should be able to do the following functions:

- Make a new reservation
- One-way
- Round-Trip
- Multi-city
- Flexible Date/time
- Confirmation
- Cancel an existing reservation
- View their itinerary

The Employee functions should have the following management functionalities:

- Get all customers who have seats reserved on a given flight.
- Get all flights for a given airport.
- View flight schedule.
- Get all flights whose arrival and departure times are on time/delayed.
- Calculate total sales for a given flight.

The Employee – Administrative functions are as given below.

- Add/Delete a flight
- Add a new airport
- Update fare for flights.

- Add a new flight leg instance.
- Update departure/arrival times for flight leg instances.

Each flight has a limited number of available seats. Several flights depart from or arrive at different cities on different dates and times.

### 2.4 Operating Environment

The operating environment for the airline management system is as listed below.

- Distributed Database
- Client/Server System
- Operating System: Windows.
- Database: sql+ database
- Platform: vb.net/Java/PHP

The client-side components of the software system must operate within common web-browser environments using Secure Sockets Layer (SSL) / Transport Layer Security (TLS) cryptographic protocols at a suitable encryption level.

### 2.5 Design and Implementation Constraints

The different constraints in the design are identified. For example, flight dates and hours should be displayed according to the city of departure and destinations' time zones. The daylight-saving time settings for each country should be considered. There should be provision for the support of different languages other than English in the future for user communication. In the implementation, the details about the database such as schema of tables, SQL commands, different global queries, etc. should be considered.

### 2.6 User Documentation

It is important to describe the use of the software system for customers and staff. This must be done using a range of short document types. The user documentation can include, guidelines, tutorials, frequently asked questions in the form of Hypertext Markup Language (HTML) and/or Portable Document Format (PDF) format.

## 2.7 Assumptions and Dependencies

In this section, the assumptions such as users have internet access with standard browsers for booking and other reservation-related queries are given. The details of the dependencies such as users can do online payments using Visa, MasterCard, and Net Banking only, the system supports Android-based mobile devices only or it shall be using third-party components for payment processing (for example ticket booking agents) are explained in this section. Some more examples of the dependencies are the calculation of the most frequent fliers and awarding them with appropriate reward points. A request for booking/cancellation of a flight from any source to any destination. Allotting connected flights in case no direct flight between the specified source-destination pair exists.

## 3. Specific Requirements

This section highlights the description and priority of specific functional requirements related to the system. The airline reservation system maintains information on flights, classes of seats, personal preferences, prices, and bookings. Of course, this project has a high priority because it is very difficult to travel across countries without prior reservations. Some of the specific requirements are given as follows:

**Stimulus/Response Sequences**

- Search for Airline Flights for two Travel cities
- Displays a detailed list of available flights and make a “Reservation” or Book a ticket on a particular flight.
- Cancel an existing Reservation.

Some of the system related features include the following:

**Distributed Database**:

Distributed database implies that a single application should be able to operate transparently on data that is spread across a variety of different databases and connected by a communication network.

**Client/Server System:**

The term client/server refers primarily to architecture or logical division of responsibilities, the client is the application (also known as the front-end), andthe server is the DBMS (also known as the back end).

A client/server system is a distributed system in which,

- Some sites are client sites and others are server sites.
- All the data resides at the server sites.
- All applications execute at the client sites.

## 4. External Interface Requirements

This section gives the details of interface requirements related to the user, hardware, software, and communication interfaces. The user interface defines to what browser the software shall be compatible with. Browsers such as Internet Explorer, Mozilla, Google Chrome, etc. may be mentioned for accessing the system. The user interface shall be implemented using any tool or software package Java Applet, MS FrontPage, EJB. The backend software can be used as SQL+.

### 4.1 User Interfaces

The user interface defines to what browser the software shall be compatible with. Browsers such as Internet Explorer, Mozilla, Google Chrome, etc. may be mentioned for accessing the system. The user interface shall be implemented using any tool or software package Java Applet, MS FrontPage, EJB. The backend software may be SQL+

### 4.2 Hardware Interfaces

Since the application must run over the internet, all the hardware shall require connecting internet will be hardware interface for the system. E.g., Modem, WAN – LAN, Ethernet Cross-cable are the details of hardware interfaces required.

### 4.3 Software Interfaces

Users connect to the software using an interface program. Application programs connect to a database management system where there is a need to permit access. Once the access is given, the client program can send the query and get the response. Some of the software descriptions used in online flight management systems are given below.

| Software used | Description |
|---|---|
| Operating system | Windows O.S. for a user-friendly and interactive approach |
| Database | SQL+ for saving passenger related records |
| API Framework | VB.net to provide access to clients from the web as well as mobile devices |

### 4.4 Communication Interfaces

The system shall use the HTTP protocol for communication over the internet and for the intranet communication will be through TCP/IP protocol suite.

## 5. Non-Functional Requirements

### 5.1 Performance Requirement

The flight management system should respond within 10ms for any request.

### 5.2 Safety Requirements

The databased used in the software may get destroyed due to disk crash/failure. In such circumstances, it is essential to have a backup or a recovery method. The recovery method can restore the last saved copy of the database to archived storage. This is used to rebuild the current state by reapplying or redoing the operations. The ongoing transactions during failures can be completed using the backed-up log.

### 5.3 Security Requirements

Security systems are used for data storage and transfer. For data transfer, the system shall use secure sockets in all transactions that include any confidential customer information. If the customer is inactive for a period, then the system shall automatically log out. The system will prevent leaving cookies on the customer's computer with a password and other confidential information.

The customer's web browser shall never display a customer's password. It shall always

be echoed with special characters representing typed characters. The customer's web browser shall never display a customer's credit card number after retrieving it from the database. It shall always be shown with just the last 4 digits of the credit card number. The system's back-end servers shall only be accessible to authenticated administrators. The system's back-end databases shall be encrypted.

### 5.4 Software Quality Attributes

Software quality attributes are defining the performance of online airline management applications. The software quality attributes related to this system are given as follows:

- Availability: The application should be available all the time. This application is critical to business communication, we will have a goal of four nines (99.99%) availability. In case of hardware or database failure, the alternative replacement page should be shown. The data must be retrieved from the backup server and service should be available.
- Maintainability: There should not be downtime for the system. The application should use continuous integration so that features and bug fixes can be deployed quickly. The modular approach will be helpful to do efficient maintainability.
- Usability: The developed system must have an easy interface. The users should be able to learn the system and reach their goals without errors. The acceptance from the users must be good, the time and cost of training to users must be less.

*****End of Case Study for SRS *****

## 5.10 Requirements Verification and Validation

Requirements validation is the process of checking that requirements define the system that the customer wants. Requirements validation is very important because errors in a requirements document can lead to extensive rework costs when these problems are discovered during development or after the system is in service.

### 5.10.1 Requirements Validation Checks

During the requirements validation process, different types of checks should be carried out on the requirements in the requirements document. These checks are:

1. Validity checks: These checks are to verify if the requirements meet the real needs of the system users.
2. Consistency checks: Requirements in the document should not conflict.
3. Completeness checks: These checks are to verify that the requirements document contains all the requirements that define all functions and constraints given by the system users.
4. Realism checks: These checks are to verify if the requirements can be implemented withinthe available budget and technology constraints.
5. Verifiability: This means that you should be able to write a set of tests that can demonstrate that the delivered system meets each specified requirement.

### 5.10.2 Requirements Validation Techniques

Several validation techniques are used.

1. Requirements Reviews: The requirements are analyzed systematically by a team of reviewers who check for errors and inconsistencies.
2. Prototyping: This involves developing an executable model of a system and using this with end-users and customers to see if it meets their needs and expectations. Stakeholders can use this prototype and give their suggestions back to the development team.
3. Test-case generation: Tests are written for requirements as a part of the validation process. These tests often reveal requirements problems. Developing such tests even before writing any code is an important part of test-driven development.

## 5.11 Requirements Management

Requirements keep on changing. During the software development process, the stakeholder's understanding of the problem changes. Requirements should be changed to reflect this. Also, once a system has been installed and is being used, new requirements crop up. Requirements management is the process of managing changing requirements during the requirements engineering process and system development.

Most changes to system requirements arise because of changes to the business

environment of the system:

1. The business and technical environment of the system always changes after installation. New hardware may be introduced, business priorities may change, new legislation may cause compliance changes.
2. System customers who have paid for the system impose requirements as per organizational and budgetary constraints. Whereas end users of the system can have different requirements, hence new features may be added after delivery.
3. The diverse set of stakeholders has differing requirements. The final system requirements are a compromise and often need to be re-prioritized at later stages.

### 5.11.1 Requirements Management Planning

Changing requirements must be managed. During the planning stage, several issues have to be decided:

1. Requirements identification: Each requirement must be uniquely identified so that it can be cross-referenced with other requirements and used in traceability assessments.
2. A change management process: This is the set of activities that assess the impact and cost of changes. This is further discussed in section 1.12.2.
3. Traceability policies: These policies define the relationships between each requirement and between the requirements and the system design that should be recorded. The traceability policy should also define how these records should be maintained.
4. Tool support: Requirements management involves the processing of large amounts of information about the requirements. Tool support is required for:
    a. Requirement storage: All the system requirements have been to be recorded and stored using some tools.
    b. Change management: Tools can keep track of suggested changes and responses to these suggestions.
    c. Traceability management: Tool support for traceability allows related requirements to be discovered.

### 5.11.2 Requirements Change Management

Change management is essential because you need to decide if the benefits of implementing new requirements are justified by the costs of implementation. There are three stages to a change management process:

1. Problem analysis and change specification: The process starts with a change proposal that is analyzed to check if it is valid. The change requester cangive feedback with more specific changes or drop the change request.
2. Change analysis and costing: The cost of making this change is estimated based on the changes required to the requirements document and system design and implementation. Once this analysis is complete, then a decision is made with regards to proceeding with this requirement change.
3. Change implementation: If the requirements change is accepted, then changes are made to the requirements document, design, and implementation. A requirements document should be easy to modify without extensive re-writing.

***(Reference Book:** Sommerville, I. (2010), Software Engineering, Addison-Wesley, Harlow, England)*

## 5.12 Multiple Choice Questions (MCQ's)

**1. The user system requirements are part of which documentation?**

A. SDLC

B. SRS

C. DFD

D. UML

**2. Which of the following is a functional requirement?**

A. Maintainability

B. Portability

C. Robustness

D. None of the above

**3. "Consider a system where a heat sensor detects an intrusion and alerts the security company." What kind of a requirement is the system providing?**

A. Functional

B. Non-functional

C. Known requirement

D. None of the above

**4. Which of the following requirements fits in the developer's module?**

A. Availability

B. Test-ability

C. Usability

D. Flexibility

**5. What is the first step of requirement elicitation?**

A. Identifying Stakeholders

B. Listing out requirements

C. Requirements gathering

D. All of the above

**6. Why is requirement elicitation a difficult task?**

A. Problem of scope

B. Problem of understanding

C. Problem of volatility

D. All of the above

**7. Which of the following is not a diagram studied in requirement analysis?**

A. Use cases

B. Entity-relationship diagram

C. State transition diagram

D. Activity diagram

**8. Which of the following property does not correspond to a good Software Requirement Specification (SRS)?**

A. Verifiable

B. Ambiguous

C. Complete

D. Traceable

**9. Which of the following is included in SRS?**

A. Cost

B. Design Constraints

C. Staffing

D. Delivery Schedule

**10. Which of the following is not included in SRS?**

A. Performance

B. Functionality

C. Design Solutions

D. External Interfaces

**11. The two requirements with priority during Product Requirement Management:**

A. User and Developer

B. Functional and Non-functional

C. Enduring and Volatile

D. All of the above

## 5.13 Review Question

1. What is Requirements Engineering? What are the steps in the requirements engineering process?
2. Why do requirements change so much?
3. Why is requirement engineering considered to be the most critical phase of SE?
4. Discuss IEEE format of SRS document. Elaborate with an example.
5. What are the characteristics of SRS?
6. Describe the basic issues in preparing the SRS.
7. What is the role of modeling in developing SRS?
8. Differentiate between functional and non-functional requirements.
9. List functional and non-functional requirements for :
   a) Hostel Management System
   b) Result Management System at a University
   c) Library Management System
10. Clearly distinguish "User Requirements" from "System Requirements". State user and system requirement for generating Monthly reports for each hospital for Mental Health

Care - Patient Management System (MHC-PMS).

11. Consider a Library Management System.
    a) Who are the stakeholders for this system?
    b) Identify user and system requirements for generating a monthly report for the stock of books.
12. Discuss all possible techniques used for requirements elicitation.

## 5.14 MCQ Solution Key

| | | | | | |
|---|---|---|---|---|---|
| 1 B | 2 D | 3 A | 4 B | 5 A | 6 D |
| 7 D | 8 B | 9 B | 10 C | 11 C | |

## 5.15 References


- The SRS sample has been adapted from *https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database*.
- Sommerville, Ian. Software Engineering. Wokingham, Eng: Addison-Wesley, 1985. Print.


*****