2 Marks Questions

Define SRS.

A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for [software](https://searchapparchitecture.techtarget.com/definition/software) under development. The SRS fully describes what the software will do and how it will be expected to perform.

Define System.

???

Describe system procurement.

System procurement is the process of acquiring a system (or systems) to meet some identified organizational need. Before procurement, decisions are made on: **scope** of the system, system **budgets and timescales**, high-level system **requirements**. Based on this information, decisions are made on whether to procure a system, the type of system and the potential system suppliers.

Explain the advantages of prototype model.

**Advantages of using Prototype Model :**

1. This model is flexible in design.
2. It is easy to detect errors.
3. We can find missing functionality easily.
4. There is scope of refinement, it means new requirements can be easily accommodated.
5. It can be reused by the developer for more complicated projects in the future.

Explain user interface prototyping.

User interface is the front-end application view to which user interacts in order to use the software. The software becomes more popular if its user interface is:

* Attractive
* Simple to use
* Responsive in short time
* Clear to understand
* Consistent on all interface screens

There are two types of User Interface:

1. **Command Line Interface:**
2. **Graphical User Interface:**

Mention any three attributes of software.

### **Availability:**This attribute is indicative as to whether an application will execute the tasks it is assigned to perform. Availability also includes certain concepts that relate to software security, [performance](https://codoid.com/best-reasons-to-invest-in-performance-testing/), integrity, reliability, dependability, and confidentiality.

### **Security:**This attribute measures the ability of a system to arrest and block malicious or unauthorized actions that could potentially destroy the system. The attribute assumes importance because security denotes the ability of the system to protect data and defend information from unauthorized access.

### **Performance:**This attribute pertains to the ability of a software-driven system to conform to timing requirements. From a testing point of view, it implies that Software Testing engineers must check whether the system responds to various events within defined time limits.

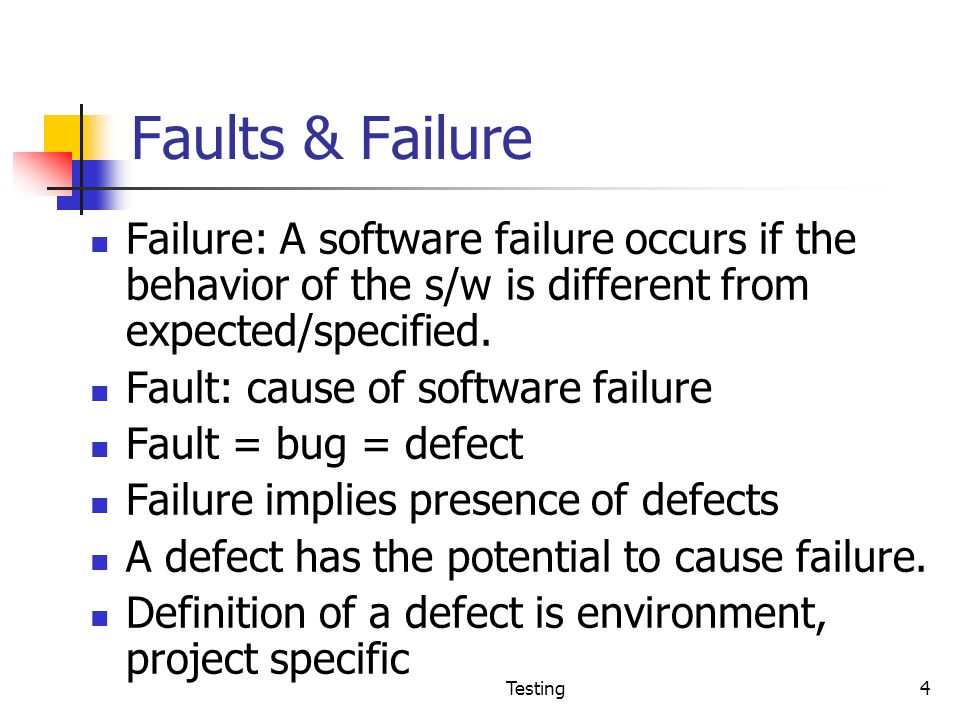
Define SDLC.

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

Explain feasibility study

The feasibility study is basically the test of the proposed system in the light of its workability, meeting user’s requirements, effective use of resources and of course, the cost effectiveness. These are categorized as technical, operational, economic, schedule and social feasibility. The main goal of feasibility study is not to solve the problem but to achieve the scope.

Identify the differences between fault and failure



Define software process

A software process (also knows as software methodology) is a set of related activities that leads to the production of the software. These activities may involve the development of the software from the scratch, or, modifying an existing system.

Explain system engineering

Systems engineering, essentially an application of systems analysis to the design and procurement of hardware systems to accomplish specific ends, can be an effective tool of management when well defined and consistently implemented.

Explain non-functional requirements.

**Non-Functional Requirements** are the constraints or the requirements imposed on the system. They specify the quality attribute of the software. Non-Functional Requirements deal with issues like scalability, maintainability, performance, portability, security, reliability, and many more.

Explain Software Requirement Definition

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client’s point of view.

Explain customized software product.

Custom Software is designing a software application for a specific group of customers within the organization. Such custom software is designed to address their needs precisely as opposed to the more traditional and widespread off-the-shelf software. Such software is typically created just for that specific entity by a third-party by contract or in-house team of developers and is not packaged for reselling.

Define requirement specification and software specification.

 Requirement Specification

A Requirement Specification is a collection of the set of all requirements that are to be imposed on the design and verification of the product. The specification also contains other related information necessary for the design, verification, and maintenance of the product.

software requirements

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client’s point of view.

Define coupling.

**coupling** is defined as a thing that joins together two objects. If we talk about software development, then the term coupling is related to the connection between two modules, i.e. how tight interaction do the two modules hold with each other is defined by coupling. Hence, the term **coupling** is defined as follows: **"“The measure of the degree of the interdependency of two modules on each other is known as coupling."**

Explain a note on factors effecting feasibility study.

• Technical aspects, which address the knowledge required to execute the project and the tools to aid during the code review.

• Economic aspects, which assess the project budget and benefits.

• Legal aspects, which analyse the legal issues that could arise.

• Operational aspects, which consider several aspects of the project, namely:

o The project team.

o The project input (code and its documentation).

o The methodology to carry out the project.

o The project scope.

• Scheduling aspects, which evaluate the time required to execute the project

Define GUI.

A **GUI** (**graphical user interface**) is a system of interactive visual components for computer software. A **GUI** displays objects that convey information, and represent actions that can be taken by the user. The objects change color, size, or visibility when the user interacts with them

10 Marks Questions

Discuss the challenges of Software Engineering.

## Major Challenges in Software Development

### Rapid technology advancement

Every technology advancement is a blessing for the IT industry. But at the same time, technology evolving at a phenomenal rate leads to an added pressure for software development professionals to leverage these upcoming technology trends in software product development to gain a cutting edge over competitors and stand out in the market.

### Increasing customer demands

Software projects are generally conceptual and are aimed at designing and developing software products that meet varied customer demands. To develop even the simplest application or product, developers must clearly understand the underlying business concept and bring in the required features to satisfy the growing customer demands.

### Time limitations

Software development is a time-game. Developers work under pressured environments and strive to complete project requirements within strict and scanty timelines. This is especially a challenge when working with international clients on multiple time-zones. Time constraints often bring down efficiencies of development teams and lead to mediocre quality software products in the end.

### Limited infrastructure/resources

Another challenge faced by majority of software development companies is a lack of resources or IT infrastructure to execute projects effectively. This could mean a lack of high performance software development tools, powerful computing platforms, inefficient data storage architectures or improper networks and connectivity. Such hindrances bring down productivity and performance of software development teams and impact the overall result.

### Conflicts with software testing teams

In a classic software development project, interpersonal conflicts occur inevitably between software development and testing teams. Several factors contribute to such conflicts like working under high performance pressure, different mindsets, difference in job roles and the very opposite nature of development and testing. If not controlled and managed effectively, these conflicts could hamper the overall project adversely.

Apply the user interface testing in the software.

 UI testing is the process of testing the visual elements of an application to validate whether they accurately meet the expected performance and functionality. By testing the GUI, testers can validate that UI functions are free from defects.It involves testing all visual indicators and graphical icons, including menus, radio buttons, text boxes, checkboxes, toolbars, colors, fonts, and more.

The main aspects checked in UI testing include:

* Visual Design
* Functionality
* Usability
* Performance
* Compliance

UI testing is centered around two main things. First, checking how the application handles user actions carried out using the keyboard, mouse, and other input devices. Second, checking whether visual elements are displayed and working correctly.

## **Approaches to UI Testing**

There are three main GUI testing approaches, namely:

### **1. Manual Testing**

In manual testing, a human tester performs a set of operations to check whether the application is functioning correctly and that the graphical elements conform to the documented requirements. Manual-based testing has notable downsides in that it can be time-consuming, and the test coverage is extremely low. Additionally, the quality of testing in this approach depends on the knowledge and capabilities of the testing team.

### **2. Record-and-Playback Testing**

Also known as record-and-replay testing, it is executed using automation tools. The automated UI testing tool records all tasks, actions, and interactions with the application. The recorded steps are then reproduced, executed, and compared with the expected behavior. For further testing, the replay phase can be repeated with various data sets.

### **3. Model-Based Testing**

In this testing approach, we focus on building graphical models that describe the behavior of a system. This provides a deeper understanding of the system, which allows the tester to generate highly efficient test cases. In the models, we determine the inputs and outputs of the system, which are in turn, used to run the tests. Model-based testing works as follows:

* Create a model for the system
* Determine system inputs
* Verify the expected output
* Execute tests
* Check and validate system output vs. the expected output

The model-based approach is great because it allows a higher level of automation. It also covers a higher number of states in the system, thereby improving the test coverage.

Explain the phases of requirement elicitation and analysis process of requirement engineering with diagram.

**Requirement Analysis**

Requirement analysis is taking account of all the possibly conflicting requirements of the various stakeholders, analyzing, validating, and documenting the requirements. It involves requirement elicitation, projecting out all the possible alternate and error flows in the (proposed) solution, and documenting all the dependencies. Typically, development of user scenarios, identifying use cases, and/or mock-ups/prototypes can be considered for the analysis.

### **Requirement Elicitation**

Requirement elicitation is the process of bringing out all the details required for the successful implementation of a solution – catering to the demand of the end-users and other stakeholders – to the complete extent, without any major deviations, either in the planned efforts or in the schedule. Requirement analysis that includes requirement elicitation is the cornerstone of the project success and any let-off in the same would invariably result in effort and schedule overruns.

Requirement elicitation can be broadly classified as functional and non-functional.

### **Functional Requirements Elicitation**

Functional requirements mainly comprise of the use-cases that need to be delivered by the solution. While eliciting the functional requirements, one needs to go much beyond the requirements provided, in the form of a simple definition of a problem or even in a detailed functional requirement document.

### **Non-Functional Requirements Elicitation**

Non-functional requirements are as important as the functional requirements, if not more. Following are the list of important aspects to be noted. Of course,

**Requirements engineering (RE)** refers to the process of defining, documenting, and maintaining requirements in the engineering design process. Requirement engineering provides the appropriate mechanism to understand what the customer desires, analyzing the need, and assessing feasibility, negotiating a reasonable solution, specifying the solution clearly, validating the specifications and managing the requirements as they are transformed into a working system. Thus, requirement engineering is the disciplined application of proven principles, methods, tools, and notation to describe a proposed system's intended behavior and its associated constraints.

## **Requirement Engineering Process**

It is a four-step process, which includes -

1. Feasibility Study
2. Requirement Elicitation and Analysis
3. Software Requirement Specification
4. Software Requirement Validation
5. Software Requirement Management



Explain the professional and ethical responsibilities of a Software Engineering?

### **1 – Confidentiality**

You must have to follow the confidentiality of the organisation, their current projects and products, information of their employees and other things which are included under the agreement of job by the employer.

### **2 – Competence**

This is the major point which we have to follow being a professional. You should never misrepresent your skills and the level of competency. Some persons try to to show off their knowledge with the manipulation of contextual words. It will benefit you initially but in the long run you will get caught. So never accept any work which is out of your competency.

**3 -Intellectual property rights**

You must have the knowledge of the local governing law regarding the software import and export. Intellectual property is a very serious crime and sensitive issue. So never try to reuse property of any other organisation to extend your own product without giving them credit or until they will allow you to use their copyright product.

### **4 – Computer misuse**

In the field of computer science, engineers try to misuse their systems which are provided by the employer to harm the integrity of organisation. It can be a cause of lawsuit against you to illegally use property of the organisation without their permission. For an example if organisation is not allowing you to play video games upon the personal system and you have installed the game that it will become a cause of computer misuse.

Explain System Procurement in detail

**System procurement**

System procurement is the process of acquiring a system (or systems) to meet some identified organizational need. Before procurement, decisions are made on: **scope** of the system, system **budgets and timescales**, high-level system **requirements**. Based on this information, decisions are made on whether to procure a system, the type of system and the potential system suppliers. These decisions are driven by:

* The state of other organizational systems and whether or not they need to be replaced
* The need to comply with external regulations
* External competition
* Business re-organization
* Available budget

It is usually necessary to develop a conceptual design document and high-level requirements before procurement. You need a specification to let a contract for system development. The specification may allow you to buy a commercial off-the-shelf (COTS) system. Almost always cheaper than developing a system from scratch. Large complex systems usually consist of a mix of off the shelf and specially designed components. The procurement processes for these different types of component are usually different.

Three types of systems or system components may have to be procured:

* Off-the-shelf applications that may be used without change and which need only minimal configuration for use.
* Configurable application or ERP systems that have to be modified or adapted for use either by modifying the code or by using inbuilt configuration features, such as process definitions and rules.
* Custom systems that have to be designed and implemented specially for use.

Explain software product. Describe two types of software product with example.

**Software Products** are nothing but software systems delivered to the customer with the documentation that that describe how to install and use the system. In certain cases, software products may be part of system products where hardware, as well as software, is delivered to a customer. Software products are produced with the help of the software process. The software process is a way in which we produce software.

**Types of software products:**  
Software products fall into two broad categories:

**Generic products:**  
Generic products are the stand-alone systems that are developed by a production unit and sold on the open market to any customer who is able to buy them.

**Examples of Generic Software**

* **Word processor** -Ms word
* Spreadsheet – Ms excel

**Customised Products:**  
Customised products are the systems that are commissioned by a particular customer. Some contractor develops the software for that customer.

**Examples of customised Software**

* Content management systems (CMS)
* Customer relationship management (CRM

**Essential characteristics of Well-Engineered Software Product:**  
A well-engineered software product should possess the following essential characteristics:

* **Efficiency:**  
  The software should not make wasteful use of system resources such as memory and processor cycles.
* **Maintainability:**  
  It should be possible to evolve the software to meet the changing requirements of customers.
* **Dependability:**  
  It is the flexibility of the software that ought to not cause any physical or economic injury within the event of system failure. It includes a range of characteristics such as reliability, security and safety.
* **In time:**  
  Software should be developed well in time.
* **Within Budget:**  
  The software development costs should not overrun and it should be within the budgetary limit.
* **Functionality:**  
  The software syatem should exhibit the proper functionality, i.e, it should perform all the functions it is supposed to perform.
* **Adaptability:**  
  The software system should have the ability to get adapted to a reasonable extent with the changing requirements.

Explain the different types of Prototyping.

## Types of Prototyping Models

Four types of Prototyping models are:

1. Rapid Throwaway prototypes
2. Evolutionary prototype
3. Incremental prototype
4. Extreme prototype

### Rapid Throwaway Prototype

Rapid throwaway is based on the preliminary requirement. It is quickly developed to show how the requirement will look visually. The customer's feedback helps drives changes to the requirement, and the prototype is again created until the requirement is baselined.

In this method, a developed prototype will be discarded and will not be a part of the ultimately accepted prototype. This technique is useful for exploring ideas and getting instant feedback for customer requirements.

### Evolutionary Prototyping

Here, the prototype developed is incrementally refined based on customer's feedback until it is finally accepted. It helps you to save time as well as effort. That's because developing a prototype from scratch for every interaction of the process can sometimes be very frustrating.

This model is helpful for a project which uses a new technology that is not well understood. It is also used for a complex project where every functionality must be checked once. It is helpful when the requirement is not stable or not understood clearly at the initial stage.

### Incremental Prototyping

In incremental Prototyping, the final product is decimated into different small prototypes and developed individually. Eventually, the different prototypes are merged into a single product. This method is helpful to reduce the feedback time between the user and the application development team.

### Extreme Prototyping:

Extreme prototyping method is mostly used for web development. It is consists of three sequential phases.

1. Basic prototype with all the existing page is present in the HTML format.
2. You can simulate data process using a prototype services layer.
3. The services are implemented and integrated into the final prototype

Explain the role of SRS

# Software Requirement Specifications

The production of the requirements stage of the software development process is **Software Requirements Specifications (SRS)** (also called a **requirements document**). This report lays a foundation for software engineering activities and is constructing when entire requirements are elicited and analyzed. **SRS** is a formal report, which acts as a representation of software that enables the customers to review whether it (SRS) is according to their requirements. Also, it comprises user requirements for a system as well as detailed specifications of the system requirements.

.

## **Characteristics of good SRS**

**Following are the features of a good SRS document:**

**1. Correctness:** User review is used to provide the accuracy of requirements stated in the SRS. SRS is said to be perfect if it covers all the needs that are truly expected from the system.

**2. Completeness:** The SRS is complete if, and only if, it includes the following elements:

**(1).** All essential requirements, whether relating to functionality, performance, design, constraints, attributes, or external interfaces.

**(2).** Definition of their responses of the software to all realizable classes of input data in all available categories of situations.

**(3).** Full labels and references to all figures, tables, and diagrams in the SRS and definitions of all terms and units of measure.

**3. Consistency:** The SRS is consistent if, and only if, no subset of individual requirements described in its conflict. There are three types of possible conflict in the SRS:

**(1).** The specified characteristics of real-world objects may conflicts. For example,

(a) The format of an output report may be described in one requirement as tabular but in another as textual.

(b) One condition may state that all lights shall be green while another states that all lights shall be blue.

**(2).** There may be a reasonable or temporal conflict between the two specified actions. For example,

(a) One requirement may determine that the program will add two inputs, and another may determine that the program will multiply them.

(b) One condition may state that "A" must always follow "B," while other requires that "A and B" co-occurs.

**(3).** Two or more requirements may define the same real-world object but use different terms for that object. For example, a program's request for user input may be called a "prompt" in one requirement's and a "cue" in another. The use of standard terminology and descriptions promotes consistency.

**4. Unambiguousness:** SRS is unambiguous when every fixed requirement has only one interpretation. This suggests that each element is uniquely interpreted. In case there is a method used with multiple definitions, the requirements report should determine the implications in the SRS so that it is clear and simple to understand.

**5. Ranking for importance and stability:** The SRS is ranked for importance and stability if each requirement in it has an identifier to indicate either the significance or stability of that particular requirement.

Typically, all requirements are not equally important. Some prerequisites may be essential, especially for life-critical applications, while others may be desirable. Each element should be identified to make these differences clear and explicit. Another way to rank requirements is to distinguish classes of items as essential, conditional, and optional.

**6. Modifiability:** SRS should be made as modifiable as likely and should be capable of quickly obtain changes to the system to some extent. Modifications should be perfectly indexed and cross-referenced.

**7. Verifiability:** SRS is correct when the specified requirements can be verified with a cost-effective system to check whether the final software meets those requirements. The requirements are verified with the help of reviews.

**8. Traceability:** The SRS is traceable if the origin of each of the requirements is clear and if it facilitates the referencing of each condition in future development or enhancement documentation.

**There are two types of Traceability:**

**1. Backward Traceability:**  **2. Forward Traceability**

The forward traceability of the SRS is especially crucial when the software product enters the operation and maintenance phase. As code and design document is modified, it is necessary to be able to ascertain the complete set of requirements that may be concerned by those modifications.

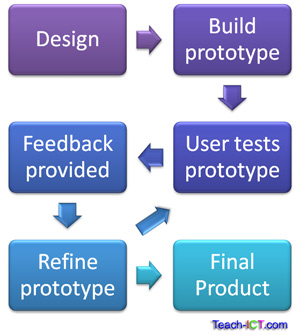
**9. Design Independence:** There should be an option to select from multiple design alternatives for the final system. More specifically, the SRS should not contain any implementation details.

**10. Testability:** An SRS should be written in such a method that it is simple to generate test cases and test plans from the report.

Explain evolutionary prototyping with a neat diagram.

Evolutionary prototyping is a software development method where the developer or development team first constructs a prototype. After receiving initial feedback from the customer, subsequent prototypes are produced, each with additional functionality or improvements, until the final product emerges.

This prototyping scheme differs from the rapid or throwaway prototyping, in that the developer begins with the best understood requirements; whereas in rapid prototyping, the developer implements the least understood requirements. Furthermore, the first prototype need not be built quickly. Note that evolutionary prototyping is similar to incremental development in that parts of the system may be inspected or delivered to the customer throughout the software life cycle model



#### Advantages of Evolutionary Prototyping

* It’s useful for exploratory programming, such as Artificial Intelligence applications, where it is difficult to frame specifications.
* If big problems are anticipated, developers can stop development after several iterations.

#### Disadvantages of Evolutionary protyping

#### As the project is open-ended, no time frame is able to be set.

* It is difficult to monitor the project.
* The visibility is low compared to the waterfall model.

Explain Functional and Non Functional Requirements.

Requirements analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: *Functional* and *Non-functional requirements*.

**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. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

**Non-functional requirements:** These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.  
They basically deal with issues like:

* Portability
* Security
* Maintainability
* Reliability
* Scalability
* Performance
* Reusability
* Flexibility

Compare and contrast between Waterfall Model and Spiral Model.

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **WATERFALL MODEL** | **SPIRAL MODEL** |
| 1. | Waterfall model works in sequential method. | While spiral model works in evolutionary method. |
| 2. | In waterfall model errors or risks are identified and rectified after the completion of stages. | In spiral model errors or risks are identified and rectified earlier. |
| 3. | Waterfall model is adopted by customers. | While spiral model is adopted by developers. |
| 4. | Waterfall model is applicable for small project. | While Spiral model is used for large project. |
| 5. | In waterfall model requirements and early stage planning is necessary. | While in spiral model requirements and early stage planning is necessary if required. |
| 6. | Flexibility to change in waterfall model is Difficult. | Flexibility to change in spiral model is not Difficult. |
| 7. | There is high amount risk in waterfall model. | There is low amount risk in spiral model. |
| 8. | Waterfall model is comparatively inexpensive. | While cost of spiral model is very expensive. |

Describe the various software design principles

**Principles Of Software Design :**

1. **Should not suffer from “Tunnel Vision” –**  
   While designing the process, it should not suffer from “tunnel vision” which means that is should not only focus on completing or achieving the aim but on other effects also.
2. **Traceable to analysis model –**  
   The design process should be traceable to the analysis model which means it should satisfy all the requirements that software requires to develop a high-quality product.
3. **Should not “Reinvent The Wheel” –**  
   The design process should not reinvent the wheel that means it should not waste time or effort in creating things that already exist. Due to this, the overall development will get increased.
4. **Minimize Intellectual distance –**  
   The design process should reduce the gap between real-world problems and software solutions for that problem meaning it should simply minimize intellectual distance.
5. **Exhibit uniformity and integration –**  
   The design should display uniformity which means it should be uniform throughout the process without any change. Integration means it should mix or combine all parts of software i.e. subsystems into one system.
6. **Accommodate change –**  
   The software should be designed in such a way that it accommodates the change implying that the software should adjust to the change that is required to be done as per the user’s need.
7. **Degrade gently –**  
   The software should be designed in such a way that it degrades gracefully which means it should work properly even if an error occurs during the execution.
8. **Assessed or quality –**  
   The design should be assessed or evaluated for the quality meaning that during the evaluation, the quality of the design needs to be checked and focused on.
9. **Review to discover errors –**  
   The design should be reviewed which means that the overall evaluation should be done to check if there is any error present or if it can be minimized.
10. **Design is not coding and coding is not design –**  
    Design means describing the logic of the program to solve any problem and coding is a type of language that is used for the implementation of a design.

Explain the quality characteristics of design.

For good quality software to be produced, the software design must also be of good quality. Now, the matter of concern is how the quality of good software design is measured? This is done by observing certain factors in software design. These factors are:

1. Correctness
2. Understandability
3. Efficiency
4. Maintainability

Now, let us define each of them in detail,

### **1) Correctness**

First of all, the design of any software is evaluated for its correctness. The evaluators check the software for every kind of input and action and observe the results that the software will produce according to the proposed design. If the results are correct for every input, the design is accepted and is considered that the software produced according to this design will function correctly.

### **2) Understandability**

The software design should be understandable so that the developers do not find any difficulty to understand it. Good software design should be self- explanatory. This is because there are hundreds and thousands of developers that develop different modules of the software, and it would be very time consuming to explain each design to each developer. So, if the design is easy and self- explanatory, it would be easy for the developers to implement it and build the same software that is represented in the design.

### **3) Efficiency**

The software design must be efficient. The efficiency of the software can be estimated from the design phase itself, because if the design is describing software that is not efficient and useful, then the developed software would also stand on the same level of efficiency. Hence, for efficient and good quality software to be developed, care must be taken in the designing phase itself.

### **4) Maintainability**

The software design must be in such a way that modifications can be easily made in it. This is because every software needs time to time modifications and maintenance. So, the design of the software must also be able to bear such changes. It should not be the case that after making some modifications the other features of the software start misbehaving. Any change made in the software design must not affect the other available features, and if the features are getting affected, then they must be handled properly.

Explain Evolutionary Development Model.

AS SAME AS Evolutionary Prototyping

Explain iterative enhancement model of software process..

## **Iterative Model - Design**

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

The following illustration is a representation of the Iterative and Incremental model −



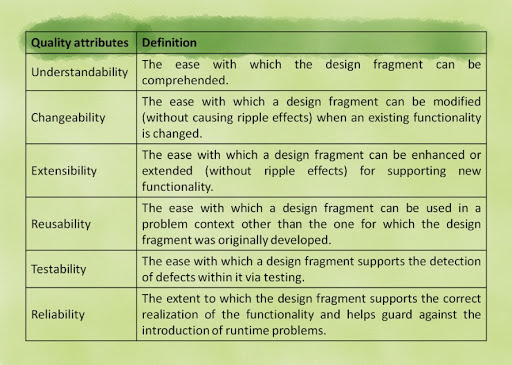
The advantages of the Iterative and Incremental SDLC Model are as follows −

* Some working functionality can be developed quickly and early in the life cycle.
* Results are obtained early and periodically.
* Parallel development can be planned.
* Progress can be measured.
* Less costly to change the scope/requirements.

The disadvantages of the Iterative and Incremental SDLC Model are as follows −

* More resources may be required.
* Although cost of change is lesser, but it is not very suitable for changing requirements.
* More management attention is required.
* System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
* Defining increments may require definition of the complete system.
* Not suitable for smaller projects.

Explain the various Design Quality.



Explain various types of cohesion with example.

### Types of Cohesion in Software Engineering

#### 1. Functional Cohesion

* The execution of the task related to the problem is the only concern from all the elements inside the module.
* The purpose of functional cohesion is single minded, strong and focused.
* All the elements in the module perform only the necessary activities.
* Some of the examples of functional cohesion are read transaction record, cosine angle computation, seat assignment to an airline passenger etc.

#### 2. Sequential Cohesion

* The data which is the output of one activity is used an input data for the next activity, the involvement of elements is as such.
* Sequential cohesion is easy maintenance and provides a good coupling.
* Sequential cohesion cannot be reused readily because of the activities that are not useful even when used together.
* Some of the examples of sequential cohesion are cross validate record and formatting of module, raw records usage, formatting of raw records, cross validation of fields in raw records, returning of formatted cross validated records.

#### 3. Communicational Cohesion

* The activities using the same input data or output data are contributed by the elements inside the module.
* Communicational cohesion is not flexible like we can only focus on some of the activities and not others at once.
* Communicational cohesion consists of the links because of which the activities affect each other.
* The cohesive ones that are functional are split by communicational cohesion.
* Some of the examples of communicational cohesion are customer details determining modules, usage of customer account number, finding the name of the customer, finding the loan balance of the customer, returning loan balance and the name of the customer etc.

#### 4. Procedural Cohesion

* The activities are related if the elements in the module are related by sequence, otherwise they are not related.
* Procedural cohesion and sequential cohesion are like each other except that the elements in the module of procedural cohesion are not related.
* Procedural cohesion is found at the top of the hierarchy like the module of main program.
* Some of the examples of procedural cohesion are read, write, edit of the module, record use out, writing out the record, reading the record, zero padding to the numeric fields, returning records etc.

#### 5. Temporal Cohesion

* The activities related in time consists of elements from the module.
* Temporal cohesion is found in the modules of initialization and termination.
* The module cannot be reused because the elements in the module are not related to each other.
* The best practice in temporal cohesion is to terminate as early as possible and initialize as late as possible.

Explain System Engineering Process.with diagram

# [Systems Engineering Process](http://acqnotes.com/acqnote/careerfields/systems-engineering-process-overview)

The Systems Engineering Process is a comprehensive, iterative and recursive problem solving process, applied sequentially top-down by integrated teams. It transforms needs and requirements into a set of system product and process descriptions, generate information for decision makers, and provides input for the next level of development. The process is applied sequentially, one level at a time, adding additional detail and definition with each level of development. [1]

The four (4) steps that comprise the SE Process are:

* **Step 1:** [**Requirements Analysis**](http://acqnotes.com/acqNote/requirements-analysis)
* **Step 2:** [**System Analysis Control**](http://acqnotes.com/acqNote/system-analysis-and-control)
* **Step 3:** [**Functional Analysis/Allocation**](http://acqnotes.com/acqNote/functional-analysis-and-allocation)
* **Step 4:** [**Design Synthesis**](http://acqnotes.com/acqNote/design-synthesis)

# [Requirements Analysis](http://acqnotes.com/acqnote/careerfields/requirements-analysis)

Requirements Analysis (Step 1) is one of the first activities of the [System Engineering Process](http://acqnotes.com/acqNote/systems-engineering-process-overview) and functions somewhat as an interface between the internal activities and the external sources providing inputs to the process. It examines, evaluates, and translates the external inputs into a set of functional and performance requirements that are the basis for the [Functional Analysis and Allocation](http://acqnotes.com/acqNote/functional-analysis-and-allocation).

# [System Analysis & Control](http://acqnotes.com/acqnote/careerfields/system-analysis-and-control)

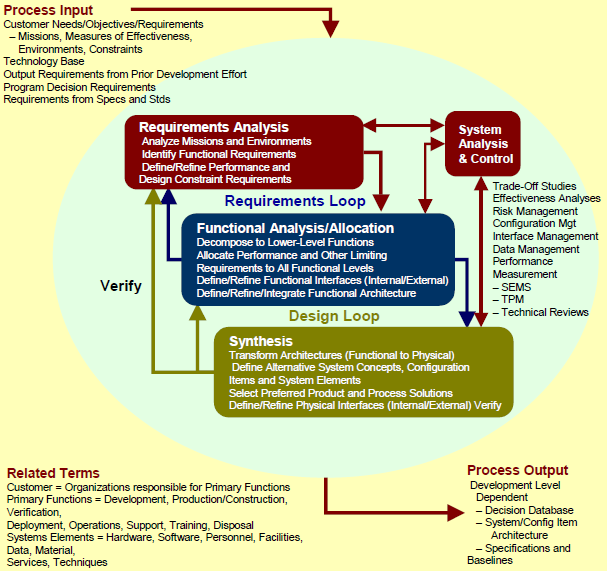
System Analysis and Control manages and controls the overall [Systems Engineering Process](http://acqnotes.com/acqNote/systems-engineering-process-overview). This activity identifies the work to be performed and develops the schedules and costs estimates for the effort. It coordinates all activities and assures that all are operating from the same set of requirements, agreements and design iteration. It’s the center for configuration management throughout the systems engineering process. [1]

# [Functional Analysis and Allocation](http://acqnotes.com/acqnote/careerfields/functional-analysis-and-allocation)

Functional Analysis and Allocation is a top-down process of translating system level requirements into detailed functional and performance design criteria. The result of the process is a defined [Functional Architecture](http://acqnotes.com/acqNote/functional-architecture) with allocated system requirements that are traceable to each system function.

# [Design Synthesis](http://acqnotes.com/acqnote/careerfields/design-synthesis)

Design Synthesis is the process of taken the functional architecture developed in the [Functional Analysis and Allocation](http://acqnotes.com/acqNote/functional-analysis-and-allocation) step and decomposing those functions into a [Physical Architecture](http://acqnotes.com/acqNote/physical-architecture) (a set of product, system, and/or software elements) that satisfy system required functions.



Explain Spiral Model with a neat diagram. Discuss its advantages and disadvantages.

**Spiral model** is one of the most important Software Development Life Cycle models, which provides support for **Risk Handling**. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. **Each loop of the spiral is called a Phase of the software development process.**

 diagram shows the different phases of the Spiral Model:  


Each phase of Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

1. **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
2. **Identify and resolve Risks:** During the second quadrant all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution is identified and the risks are resolved using the best possible strategy. At the end of this quadrant, Prototype is built for the best possible solution.
3. **Develop next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.
4. **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

**Advantages of Spiral Model**: Below are some of the advantages of the Spiral Model.

* **Risk Handling:** The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.
* **Good for large projects:** It is recommended to use the Spiral Model in large and complex projects.
* **Flexibility in Requirements:** Change requests in the Requirements at later phase can be incorporated accurately by using this model.
* **Customer Satisfaction:** Customer can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.

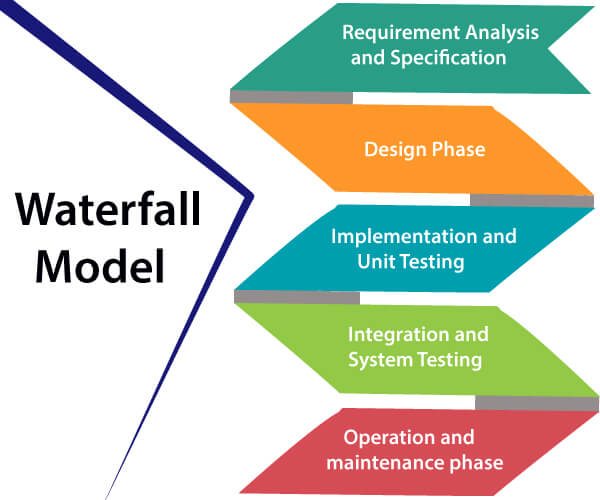
**Disdvantages of Spiral Model**: Below are some of the main disadvantages of the spiral model.

* **Complex:** The Spiral Model is much more complex than other SDLC models.
* **Expensive:** Spiral Model is not suitable for small projects as it is expensive.
* **Too much dependable on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced expertise, it is going to be a failure to develop a project using this model.
* **Difficulty in time management:** As the number of phases is unknown at the start of the project, so time estimation is very difficult.

Explain waterfall model with a neat diagram

Waterfall model

Winston Royce introduced the Waterfall Model in 1970.This model has five phases: Requirements analysis and specification, design, implementation, and unit testing, integration and system testing, and operation and maintenance. The steps always follow in this order and do not overlap. The developer must complete every phase before the next phase begins. This model is named "**Waterfall Model**", because its diagrammatic representation resembles a cascade of waterfalls.

**1. Requirements analysis and specification phase:** The aim of this phase is to understand the exact requirements of the customer and to document them properly. Both the customer and the software developer work together so as to document all the functions, performance, and interfacing requirement of the software. It describes the "what" of the system to be produced and not "how."In this phase, a large document called **Software Requirement Specification (SRS)** document is created which contained a detailed description of what the system will do in the common language.

**2. Design Phase:** This phase aims to transform the requirements gathered in the SRS into a suitable form which permits further coding in a programming language. It defines the overall software architecture together with high level and detailed design. All this work is documented as a Software Design Document (SDD).

**3. Implementation and unit testing:** During this phase, design is implemented. If the SDD is complete, the implementation or coding phase proceeds smoothly, because all the information needed by software developers is contained in the SDD.

During testing, the code is thoroughly examined and modified. Small modules are tested in isolation initially. After that these modules are tested by writing some overhead code to check the interaction between these modules and the flow of intermediate output.

**4. Integration and System Testing:** This phase is highly crucial as the quality of the end product is determined by the effectiveness of the testing carried out. The better output will lead to satisfied customers, lower maintenance costs, and accurate results. Unit testing determines the efficiency of individual modules. However, in this phase, the modules are tested for their interactions with each other and with the system.

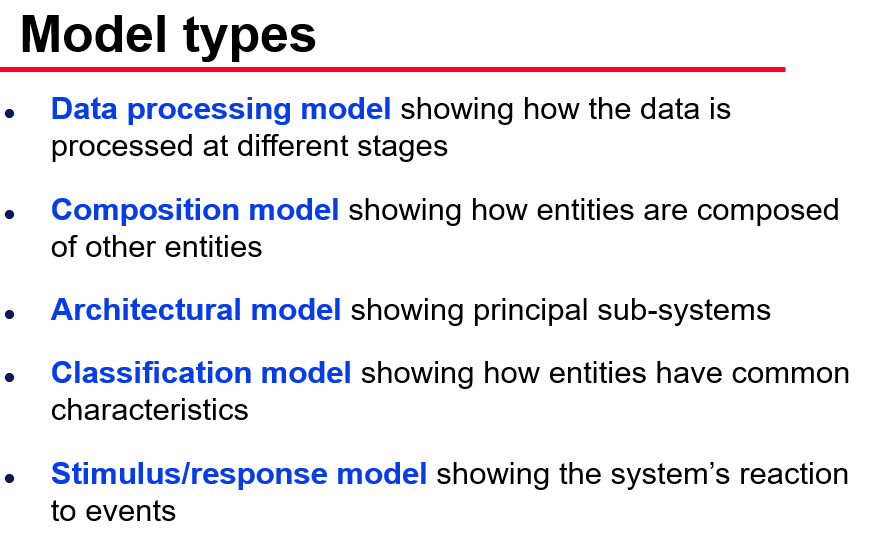
**5. Operation and maintenance phase:** Maintenance is the task performed by every user once the software has been delivered to the customer, installed, and operational.

## **Advantages of Waterfall model**

* This model is simple to implement also the number of resources that are required for it is minimal.
* The requirements are simple and explicitly declared; they remain unchanged during the entire project development.
* The start and end points for each phase is fixed, which makes it easy to cover progress.

## **Disadvantages of Waterfall model**

* In this model, the risk factor is higher, so this model is not suitable for more significant and complex projects.
* This model cannot accept the changes in requirements during development.
* It becomes tough to go back to the phase. For example, if the application has now shifted to the coding phase, and there is a change in requirement, It becomes tough to go back and change it.

Write a short note on various system models.

Explain IEEE structure of SRS.

**IEEE**defines software requirements specification as, ‘a document that clearly and precisely describes each of the essential requirements (functions, performance, design constraints and quality attributes) of the software and the external interfaces. Each requirement is defined in such a way that its achievement can be objectively verified by a prescribed method, for example, inspection, demonstration, analysis or test.’Note that requirements specification can be in the form of a writtendocument, a mathematical model, a collection of graphical models, a prototype,and so on.

1. **Feedback:**Provides a feedback, which ensures to the user that the organization (which develops the software) understands the issues or problems to be solved and the software behavior necessary to address those problems.
2. **Decompose problem into components:**Organizes the [information](https://ecomputernotes.com/fundamental/information-technology/what-do-you-mean-by-data-and-information) and divides the problem into its component parts in an orderly manner.
3. **Validation:**Uses validation strategies applied to the requirements to acknowledge that requirements are stated properly.
4. **Input to design:**Contains sufficient detail in the functional system requirements to devise a design solution.
5. **Basis for agreement between the**user and **the organization:**Provides a complete description of the functions to be performed by the system. In addition, it helps the users to determine whether the specified requirements are accomplished.
6. **Reduce the development effort:**Enables developers to consider user requirements before the designing of the system commences. As a result, ‘rework’ and inconsistencies in the later stages can be reduced.
7. **Estimating costs and schedules:**Determines the requirements of the system and thus enables the developer to have a ‘rough’ estimate of the total cost and schedule of the project.

Explain various types of coupling with example.

**Types of Coupling:**

* **Data Coupling:** If the dependency between the modules is based on the fact that they communicate by passing only data, then the modules are said to be data coupled. In data coupling, the components are independent to each other and communicating through data. Module communications don’t contain tramp data. Example-customer billing system.
* **Stamp Coupling** In stamp coupling, the complete data structure is passed from one module to another module. Therefore, it involves tramp data. It may be necessary due to efficiency factors- this choice made by the insightful designer, not a lazy programmer.
* **Control Coupling:** If the modules communicate by passing control information, then they are said to be control coupled. It can be bad if parameters indicate completely different behavior and good if parameters allow factoring and reuse of functionality. Example- sort function that takes comparison function as an argument.
* **External Coupling:** In external coupling, the modules depend on other modules, external to the software being developed or to a particular type of hardware. Ex- protocol, external file, device format, etc.
* **Common Coupling:** The modules have shared data such as global data structures.The changes in global data mean tracing back to all modules which access that data to evaluate the effect of the change. So it has got disadvantages like difficulty in reusing modules, reduced ability to control data accesses and reduced maintainability.
* **Content Coupling:** In a content coupling, one module can modify the data of another module or control flow is passed from one module to the other module. This is the worst form of coupling and should be avoided.

Explain a SRS for Banking System.

**. Introduction**

This document gives detailed functional and nonfunctional requirements for the bank management system. This product will support online banking transaction. The purpose of this document is that the requirements mentioned in it should be utilized by software developer to implement the system.

**1.1 Purpose**

Online banking system provides is specifically developed for internet banking for Balance Enquiry, Funds Transfer to another account in the same bank, Request for cheque book/change of address/stop payment of cheques, Mini statements (Viewing Monthly and annual statements).

The Traditional way of maintaining details of a user in a bank was to enter the details and record them. Every time the user need to perform some transactions he has to go to bank and perform the necessary actions, which may not be so feasible all the time. It may be a hard-hitting task for the users and the bankers too. The project gives real life understanding of Internet banking and activities performed by various roles in the supply chain. Here, we provide an automation for banking system through Internet. Internet banking system project captures

activities performed by different roles in real life banking which provides enhanced techniques for maintaining the required in- formation up-to-date, which results in efficiency. The project gives real life understanding of Internet banking and activities performed by various roles in the supply chain.

**1.2 Scope**

This Product will automate of banking transaction process.This  Project  investigates  the  entry  threshold  for  providing  a  new  transaction service  channel  via  the  real  options  approach,  where  the  entry  threshold  is established  by using an Internet banking  system designed  for the use of normal users(individuals), Industrialists, Entrepreneurs, Educational Institutions(Financial sections), Organizations and Academicians under transaction rate uncertainty.

**1.3 Overview**

The system provides easy solution to banks.

 Overview: The SRS will include two sections, namely:

Overall Description: This section will describe  major components  of the system, interconnections,  and external interfaces.

Specific Requirements:  This section  will describe  the  functions  of actors, their roles in the system and the constraints faced by sys- tem.

**2. General description**

2.1    Product Perspective:

The client will have client interface in which he can interact with the banking sys- tem. It is a web based interface which will be the web page of the banking application.  Starting a page  is displayed  asking  the  type  of customer  he  is whether  ordinary  or  a corporate customer. Then the page is redirected to login page where the user can enter the login details. If the login particulars are valid then the user is taken to a home page where he has the entire transaction list that he can perform with the bank. All the above activities come under the client interface.

The   administrator   will  have  an  administrative   in- terface which is a GUI so that he can view the entire system. He will also have a login page where  he  can  enter  the  login  particulars  so  that  he  can  perform  all  his  actions.  This administrative  interface provides different environment such that he can maintain data- base & provide  backups for the information  in the database.  He can register the users by providing them with username,  password  & by creating  account  in the  database.  He can view  the cheque book request & perform action to issue the cheque books to the clients.

2.2    Software Interface:

Front End Client:

The  system is  a  web  based  application  clients  are  requiring using modern web browser such as Mozilla Firefox 1.5, PHP.

\* Web Server:

The web application will be hosted on one of the apache server.

\* Back End:

We use backend as MY SQL.

**3. Functional Specifications**

This section provides the functional overview of the product. The project will require the PHP as a front end and at the back end the database MYSQL will be running. Various functional modules that can be implemented by the product will be

1. Login

2. Validation

3. Get balance information

4. Withdrawal of money

5. Transfer Money

6. Customer info.

**3.1 Login:**

Customer logins by entering customer name & a login pin.

**3.2 Validation:**

When a customer enters the ATM card, its validity must be ensured. Then customer is allowed to enter the valid PIN. The validation can be for following conditions

Validation for lost or stolen card

When card is already reported as lost or stolen

then the message “Lost/Stolen card!!!”.

Validation for card’s expiry date

If the card inserted by the customer has crossed the expiry date then the system will prompt

“Expired Card”.

Validation for PIN

After validating the card, the validity of PIN must be ensured. If he/she fails to enter valid code for three times then the card will not be returned to him. That means the account can be locked. The counter for number of logins must be maintained

Get balance information:

This system must be networked to the bank’s computer. The updated

database of every customer is maintained with bank. Hence the balance information of every account is available in the database and can be displayed to the customer.

**3.3 Payment of Money:**

A customer is allowed to enter the amount which he/she wishes to withdraw. If the entered amount is less than the available balance and if after withdraw if the minimum required balance is maintained then allow the transaction.

**3.4 Transfer of Money:**

The customer can deposit or transfer the desired amount of money.

**3.5 Transaction Report:**

The bank statement showing credit and debit information of corresponding account must be printed by the machine.

**3.6 Technical Issues**

This product will work on client-server architecture. It will require an internet server and which will be able to run PHP applications. The product should support some commonly used browsers such as Internet Explorer, Mozilla Firefox.

**4. Interface Requirements**

**4.1 GUI**

This is interface must be highly intuitive or interactive because there will not be an assistance for the user who is operating the System. At most of the places help desk should be provided for users convenience. The screens appearing should be designed in such a manner that it can draw User attaraction towards the new plans for the customers.

Also the pin and password confidentiality should be maintained,

This can be done by using asterisks at the password panel.

Proper security messages should be displayed at most of the  places.

**4.2 Hardware Interface**

Various interfaces for the product could be

1. Touch screen/Monitor

2. Keypad

3. Continuous battery backup

4. Printer which can produce the hard copy.

5. Interface that connects the device to bank’s computer.

6. An interface that can count currency notes.

**4.3 Software Interface**

1. Any windows operating system.

2. The PHP must be installed. For the database handling MYSQL must be installed. These products are open source products.

3. The final application must be packaged in a set up program, so that the products can be easily installed on machines. This application must be networked to corresponding banks.

**5. Performance Requirements**

The system should be compatible enough to hold the general traffic .

It should not get hang or show some other problems arising out due to large no of concurrent users . The system should be fast enough to meet the customer The high and low temperature should not affect the performance of the device. An uninterrupted transaction must be performed.

**6.Constraints**

\* The information of all the users must be stored in a database that is accessible by the On- line

Banking System.

\* The Online Banking System is connected to the computer and is running all 24hours a day.

\* The  users  access  the  Online  Banking  System from any computer  that has Internet  browsing capabilities and an Internet connection.

\*The users must have their correct usernames and passwords to enter into the Online Banking System.

Design Constraints:

\* Software Language Used

The languages that shall be used for coding Online Banking System are c , c++ , java , and  HTML.  For  working  on the  coding  phase  of  the  Online  job portal  System Web Sphere  Application  Server/WebSphere  Application  Server  CE  Server  needs  to  be  installed.

\*Database design

|  |
| --- |
|  |

In our database design, we give names to data flows, processes and data stores. Although the names are descriptive of data, they do not give details .So following DFD, our interest is to build some details of the contents of data flows, processes and data store. A data dictionary is a structured repository of data about data .It is a set of rigorous definitions  of all DFD  data elements  and data  structures  .

**7. Performance**

**7.1 Security**

The banking system must be fully accessible  to only authentic user.

It should require pin for entry to a new environment.

**7.2 Reliability**

The application should be highly reliable and it should generate all the updated information in correct order.

**7.3 Availability**

Any information about the account should be quickly available from any computer to the authorized user. The previously visited customer’s data must not be cleared.

**7.4 Maintainability**

The application should be maintainable in such a manner that if any new requirement occurs then it should be easily incorporated in an individual module.

**7.5 Portability**

The application should be portable on any windows based system. It should not be machine specific.