Define Software Engineering:

- ".. An engineering discipline that is concerned **with all aspects of software production** from the **early stages** of system specification to **maintaining the system after it has gone into use.**" *Sommerville, pg.7*

- Software engineering is the application of principles used in the field of engineering, which usually deals with physical systems, to the design, development, testing, deployment and management of software systems.

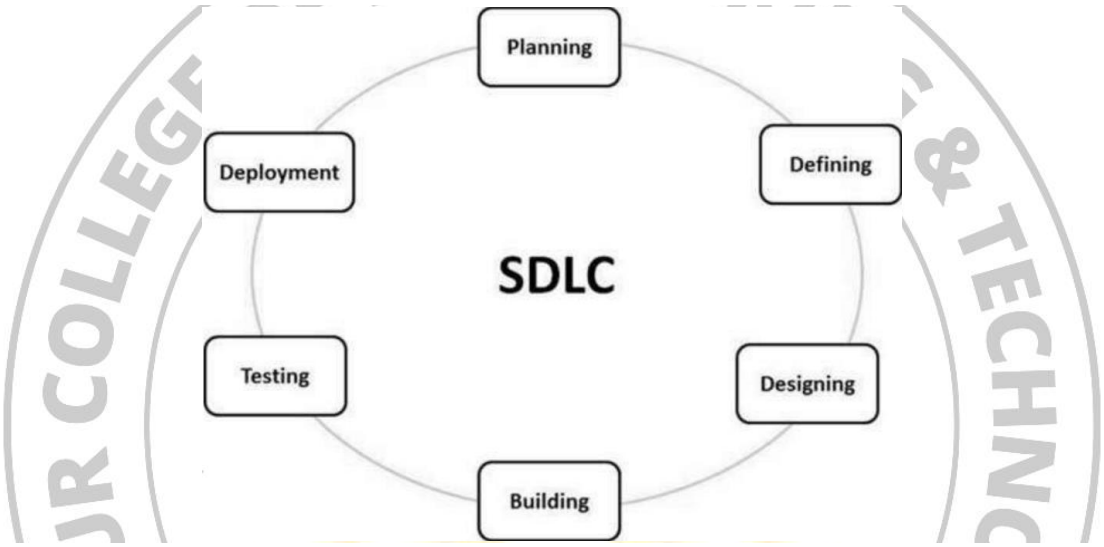
Importance of Software Engineering:

- **1. Reduces Complexity**
- Dealing with big Software is very complicated and challenging. Thus to reduce the complications of projects, software engineering has great solutions. It simplifies complex problems and solves those issues one by one.
- **2. Handling Big Projects**
- Big projects need lots of patience, planning, and management, which you never get from any company. The company will invest its resources; therefore, it should be completed within the deadline. It is only possible if the company uses software engineering to deal with big projects without problems.
- **3. To Minimize Software Costs**
- Software engineers are paid highly as Software needs a lot of hard work and workforce development. These are developed with the help of a large number of codes. But programmers in software engineering project all things and reduce the things which are not needed. As a result of the production of Software, costs become less and more affordable for Software that does not use this method.
- **4. To Decrease Time**
- If things are not made according to the procedures, it becomes a huge loss of time. Accordingly, complex Software must run much code to get definitive running code. So it takes lots of time if not handled properly. And if you follow the prescribed software engineering methods, it will save your precious time by decreasing it.
- **5. Effectiveness**
- Making standards decides the effectiveness of things. Therefore a company always targets the software standard to make it more effective. And Software becomes more effective only with the help of software engineering.
- **6. Reliable Software**
- The Software will be reliable if software engineering, testing, and maintenance are given. As a software developer, you must ensure that the Software is secure and will work for the period or subscription you have agreed upon.

SDLC:

What is SDLC?

- SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software.
- The life cycle defines a methodology for improving the quality of software and the overall development process.
- The following figure is a graphical representation of the various stages of a typical SDLC.



- Stage 1: Planning and Requirement Analysis
- Requirement analysis is the most important and fundamental stage in SDLC.
- It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry.
- This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.
- Stage 2: Defining Requirements
- Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts.
- This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

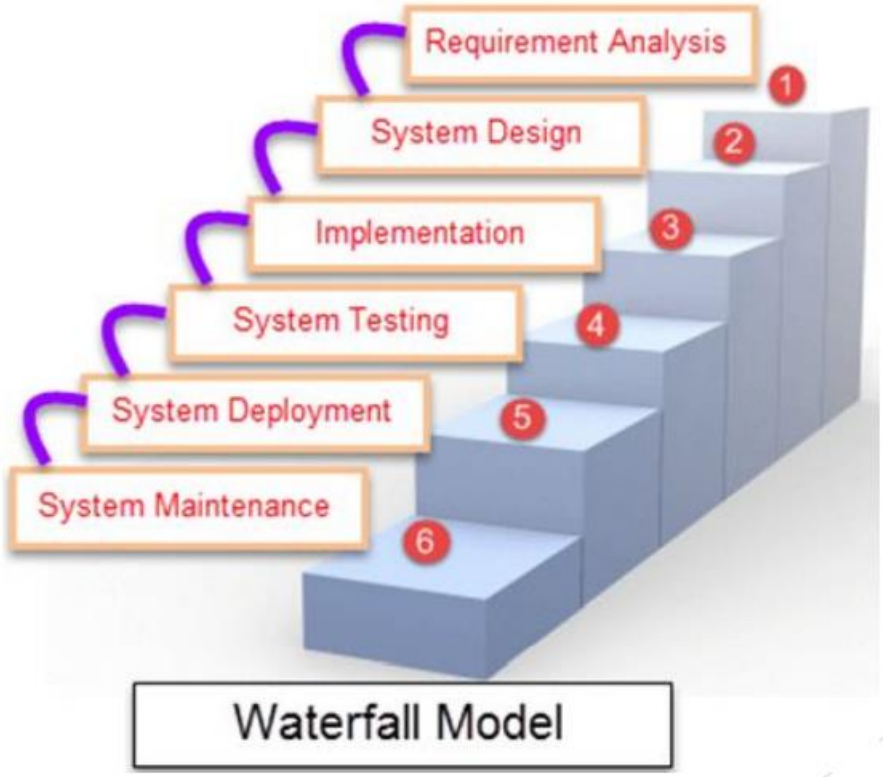
- Stage 3: Designing the Product Architecture
- SRS is the reference for product architects to come out with the best architecture for the product to be developed.
- Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.
- This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

- Stage 4: Building or Developing the Product
- In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage.
- The programming language is chosen with respect to the type of software being developed.

- Stage 5: Testing the Product
- This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC.
- However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

- Stage 6: Deployment in the Market and Maintenance
- Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment.
- After the product is released in the market, its maintenance is done for the existing customer base.

Waterfall



- It was introduced in 1970 by Winston Royce.
- The classical waterfall model is the basic software development life cycle model.
- It is very simple but idealistic.
- Earlier this model was very popular but nowadays it is not used.
- But it is very important because all the other software development life cycle models are based on the classical waterfall model.

- **Waterfall Model** is a sequential model that divides software development into pre-defined phases.
- Each phase must be completed before the next phase can begin with no overlap between the phases.
- Each phase is designed for performing specific activity during the SDLC phase.

Different phases	Activities performed in each stage
Requirement Gathering stage	•During this phase, detailed requirements of the software system to be developed are gathered from client
Design Stage	•Plan the programming language, for Example Java, PHP, .net •or database like Oracle, MySQL, etc. •Or other high-level technical details of the project
Built Stage	•After design stage, it is built stage, that is nothing but coding the software
Test Stage	•In this phase, you test the software to verify that it is built as per the specifications given by the client.
Deployment stage	•Deploy the application in the respective environment
Maintenance stage	•Once your system is ready to use, you may later require change the code as per customer request

Advantages:

Easy to Understand: The waterfall model is easy to understand and simple to implement. Its linear and sequential nature makes it easier for developers to follow and understand.

Clear and Well-Defined Phases: The waterfall model is broken down into clear and well-defined phases, each with its own set of objectives and deliverables. This makes it easier for developers to plan and track their progress throughout the project.

Requirements are Clearly Defined: The waterfall model requires a complete and detailed understanding of the requirements before any development work can begin. This ensures that developers have a clear understanding of what is expected from them, reducing the risk of misunderstandings or errors.

Emphasis on Documentation: The waterfall model places a strong emphasis on documentation, ensuring that all aspects of the project are documented thoroughly. This can be beneficial for future maintenance or updates, as it provides a clear understanding of the project's history.

Good For Short Projects

Disadvantages:

Rigid and Inflexible: The waterfall model is rigid and inflexible, making it difficult to make changes once the development process has begun. This can be problematic in situations where the requirements or scope of the project change mid-way through development.

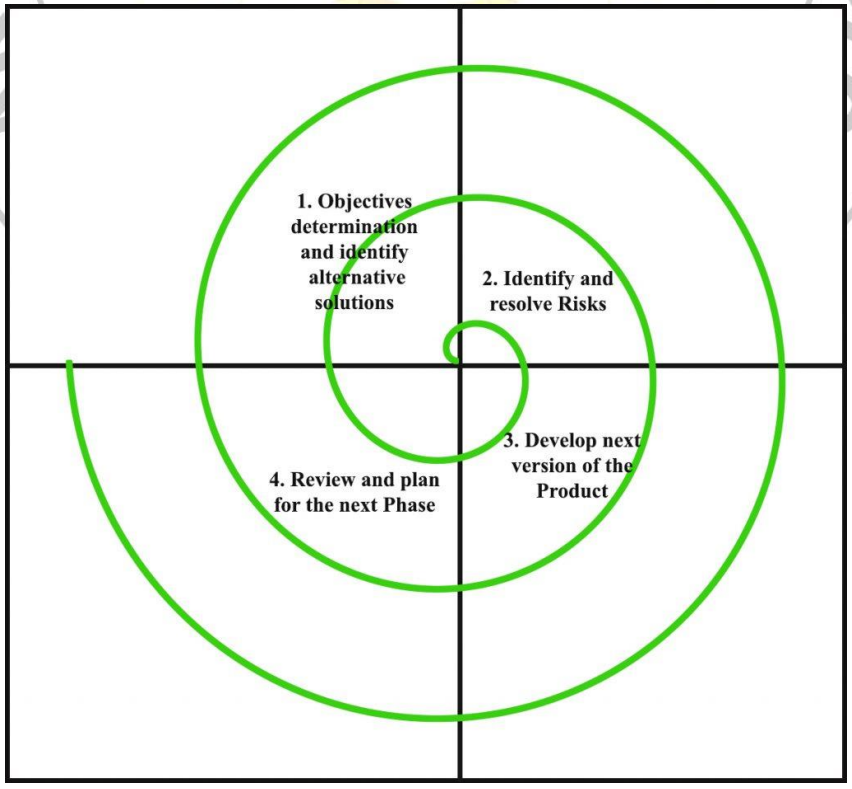
No Room for Error: The waterfall model assumes that all requirements are known at the beginning of the project and that there will be no changes throughout the development process. This leaves no room for error or changes in requirements, which can lead to delays or even project failure.

Lengthy Development Cycle: The waterfall model has a lengthy development cycle, with each phase being completed before the next can begin. This can result in longer development times and delayed delivery of the final product.

Testing is Performed at the End: In the waterfall model, testing is performed at the end of the development cycle. This means that any issues or defects that are identified during testing may require significant rework or even a complete restart of the development process.

Spiral:

The Spiral Model is a software development life cycle (SDLC) model that provides a systematic and iterative approach to software development. It is based on the idea of a spiral, with each iteration of the spiral representing a complete software development cycle, from requirements gathering and analysis to design, implementation, testing, and maintenance. The Spiral Model is a risk-driven model, meaning that the focus is on managing risk through multiple iterations of the software development process. It is also known as the **Meta** Model, It consists of the following phases:



Each iteration of the 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 analysed 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 are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the 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.

Risk Handling in Spiral Model: A risk is any adverse situation that might affect the successful completion of a software project. The most important feature of the spiral model is handling these unknown risks after the project has started. Such risk resolutions are easier done by developing a prototype.

Advantages of Spiral Model:

Below are some advantages of the Spiral Model.

1. **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.
2. **Good for large projects:** It is recommended to use the Spiral Model in large and complex projects.
3. **Flexibility in Requirements:** Change requests in the Requirements at later phase can be incorporated accurately by using this model.
4. **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.
5. **Improved Quality:** The Spiral Model allows for multiple iterations of the software development process, which can result in improved software quality and reliability

Disadvantages of Spiral Model:

Below are some main disadvantages of the spiral model.

1. **Complex:** The Spiral Model is much more complex than other SDLC models.
2. **Expensive:** Spiral Model is not suitable for small projects as it is expensive.

3. **Too much dependability on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced experts, it is going to be a failure to develop a project using this model.
4. **Difficulty in time management:** As the number of phases is unknown at the start of the project, so time estimation is very difficult.

Agile Model

Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like −

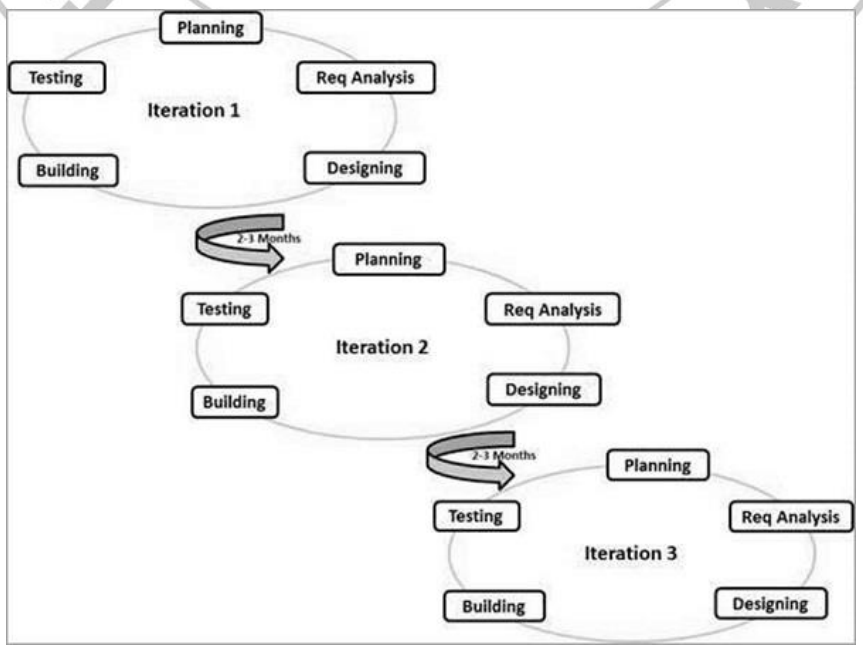
- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.

At the end of the iteration, a working product is displayed to the customer and important stakeholders.

In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

Here is a graphical illustration of the Agile Model −



The Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.

Following are the Agile Manifesto principles −

- **Individuals and interactions** − In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
- **Working software** − Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.
- **Customer collaboration** − As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
- **Responding to change** − Agile Development is focused on quick responses to change and continuous development.

Agile Model - Pros and Cons

Agile methods are being widely accepted in the software world recently. However, this method may not always be suitable for all products. Here are some pros and cons of the Agile model.

The advantages of the Agile Model are as follows −

- Is a very realistic approach to software development.
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements.
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required.
- Easy to manage.
- Gives flexibility to developers.

The disadvantages of the Agile Model are as follows −

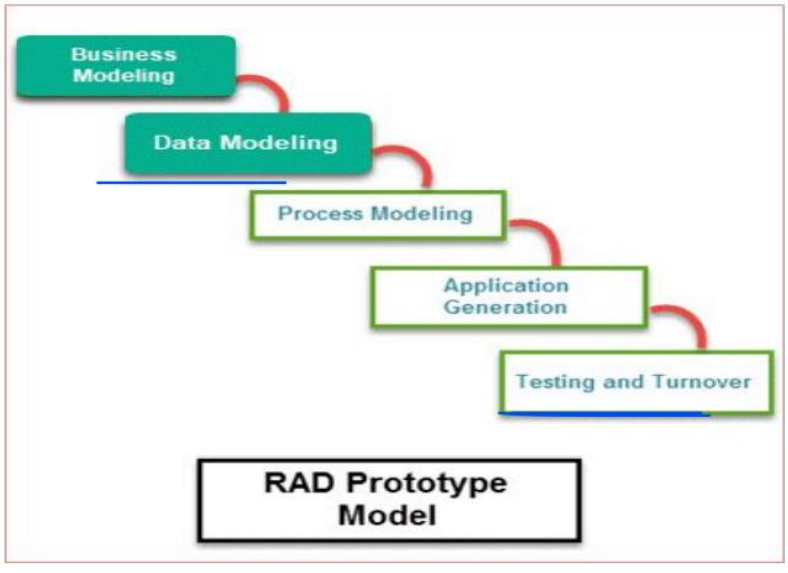
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

RAD MODEL:

- RAD Model or Rapid Application Development model is a software development process.
- It is based on prototyping without any specific planning.
- In RAD model, there is less attention paid to the planning and more priority is given to the development tasks.
- It targets at developing software in a short span of time.

- It focuses on input-output source and destination of the information.
- It emphasizes on delivering projects in small pieces; the larger projects are divided into a series of smaller projects.
- The main features of RAD modeling are that it focuses on the reuse of templates, tools, processes, and code.

RAD Model Diagram:



RAD Model Phases	Activities performed in RAD Modeling
Business Modeling	•On basis of the flow of information and distribution between various business channels, the product is designed
Data Modeling	•The information collected from business modeling is refined into a set of data objects that are significant for the business
Process Modeling	•The data object that is declared in the data modeling phase is transformed to achieve the information flow necessary to implement a business function
Application Generation	•Automated tools are used for the construction of the software, to convert process and data models into prototypes
Testing and Turnover	•As prototypes are individually tested during every iteration, the overall testing time is reduced in RAD.

Advantage of RAD Model

- This model is flexible for change.
- In this model, changes are adoptable.
- Each phase in RAD brings highest priority functionality to the customer.
- It reduced development time.
- It increases the reusability of features.

Disadvantage of RAD Model

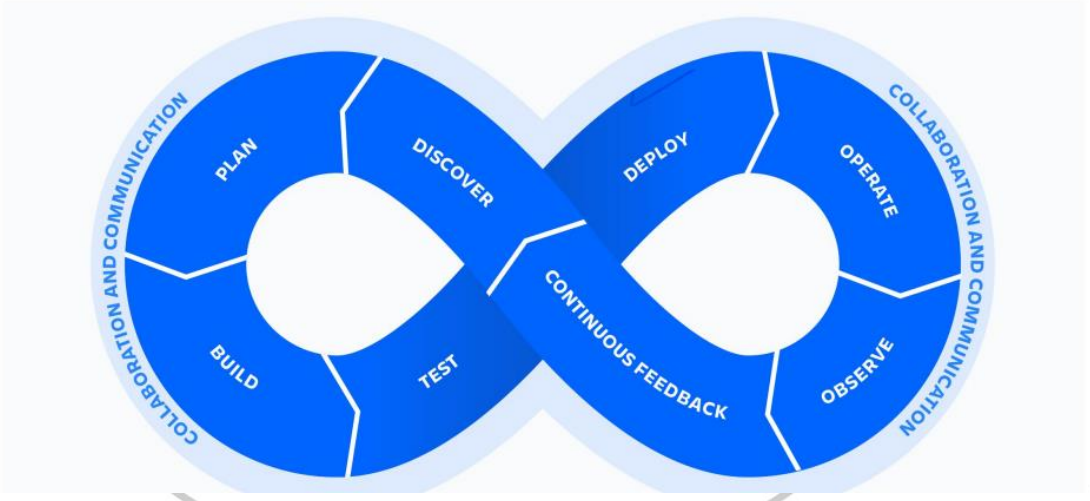
- It required highly skilled designers.
- All application is not compatible with RAD.
- For smaller projects, we cannot use the RAD model.
- On the high technical risk, it's not suitable.
- Required user involvement.

DEVOPS:

- DevOps is a set of practices, tools, and a cultural philosophy.
- It automate and integrate the processes between software development and IT teams.
- It emphasizes team empowerment, cross-team communication and collaboration, and technology automation.

- The DevOps movement began around 2007 when the software development and IT operations communities raised concerns about the traditional software development model.
- where developers who wrote code worked apart from operations who deployed and supported the code.
- The term DevOps, a combination of the words development and operations, reflects the process of integrating these disciplines into one, continuous process.

The DevOps lifecycle



The Eight Phases of a DevOps Pipeline | by Jakob Pennington | Taptu | Medium

What are the benefits of DevOps?

Speed

Teams that practice DevOps release deliverables more frequently, with higher quality and stability. In fact, the DORA 2019 State of DevOps report found that elite teams deploy 208 times more frequently and 106 times faster than low-performing teams. Continuous delivery allows teams to build, test, and deliver software with automated tools.

Improved collaboration

The foundation of DevOps is a culture of collaboration between developers and operations teams, who share responsibilities and combine work. This makes teams more efficient and saves time related to work handoffs and creating code that is designed for the environment where it runs.

Rapid deployment

By increasing the frequency and velocity of releases, DevOps teams improve products rapidly. A competitive advantage can be gained by quickly releasing new features and repairing bugs.

Quality and reliability

Practices like continuous integration and continuous delivery ensure changes are functional and safe, which improves the quality of a software product. Monitoring helps teams keep informed of performance in real-time.

Security

By integrating security into a continuous integration, continuous delivery, and continuous deployment pipeline, DevSecOps is an active, integrated part of the development process. Security is built into the product by integrating active security audits and security testing into agile development and DevOps workflows.

Disadvantages of DevOps

Increased Complexity

The adoption of DevOps can often lead to an increase in complexity within an organization's IT infrastructure. By integrating multiple tools and technologies into the software development process, businesses can create a more complex production environment that is difficult to manage and troubleshoot. In addition,

DevOps can require organizations to invest in additional hardware and software resources, which can further increase complexity and costs.

Lack of Standardization

There is currently no industry-wide standardization for DevOps. As a result, businesses that adopt DevOps may need to create their own customized process and toolsets, which can be time-consuming and costly. Furthermore, the lack of standardization can lead to confusion among employees about how to best use DevOps techniques.

Increased Costs

The adoption of DevOps can often lead to increased costs for businesses. By requiring the purchase of additional hardware and software resources and the hiring of experienced DevOps professionals, businesses can see a significant increase in their IT expenses. In addition, the complexity of a DevOps environment can often lead to problems with the reliability and performance of business-critical applications.

INCREMENTAL MODEL:

Incremental Model

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation, and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.

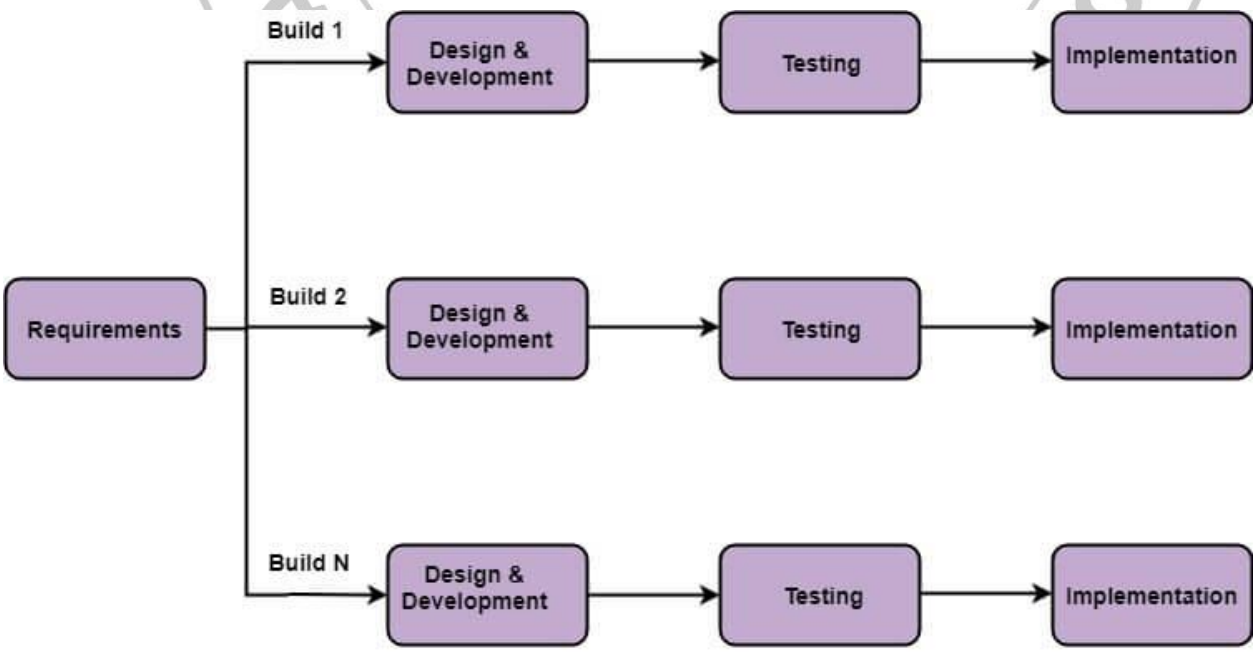


Fig: Incremental Model

The various phases of incremental model are as follows:

1. Requirement analysis: In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

2. Design & Development: In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

3. Testing: In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behaviour of each task.

4. Implementation: Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product.

Advantage of Incremental Model

- Errors are easy to be recognized.
- Easier to test and debug.
- More flexible.
- Simple to manage risk because it handled during its iteration.
- The Client gets important functionality early.

Disadvantage of Incremental Model

- Need for good planning.
- Total Cost is high.
- Well defined module interfaces are needed.

MODULE 2

Types of requirements:

Functional Requirements:

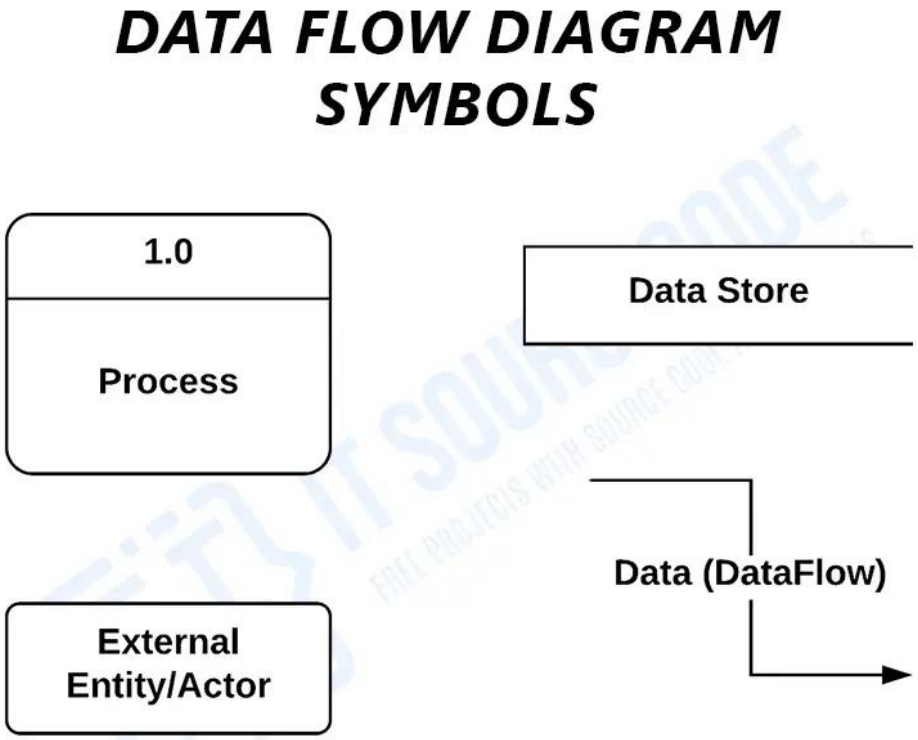
- These are the requirements that the end user specifically demands as basic facilities that the system should offer.
- It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.
- All these functionalities need to be necessarily incorporated into the system as a part of the contract.
- 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.
- Non-functional requirements, not related to the system functionality, rather define how the system should perform.
- The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioural requirements.
- They basically deal with issues like:
    - **Portability**
    - **Security**
    - **Maintainability**
    - **Reliability**
    - **Scalability**
    - **Performance**
    - **Reusability**
    - **Flexibility**

Domain requirements: Domain requirements are the requirements which are characteristic of a particular category or domain of projects. Domain requirements can be functional or non-functional.

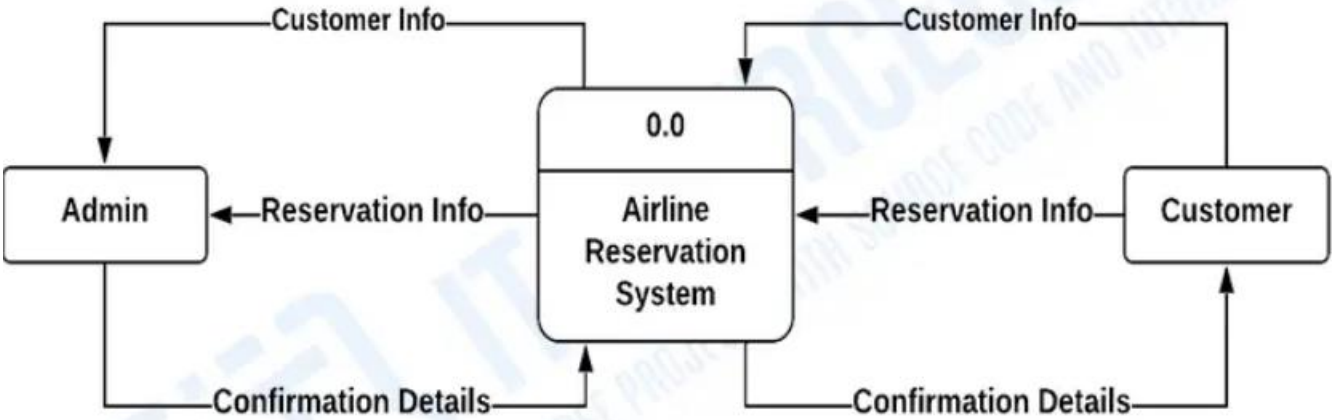
DFD:



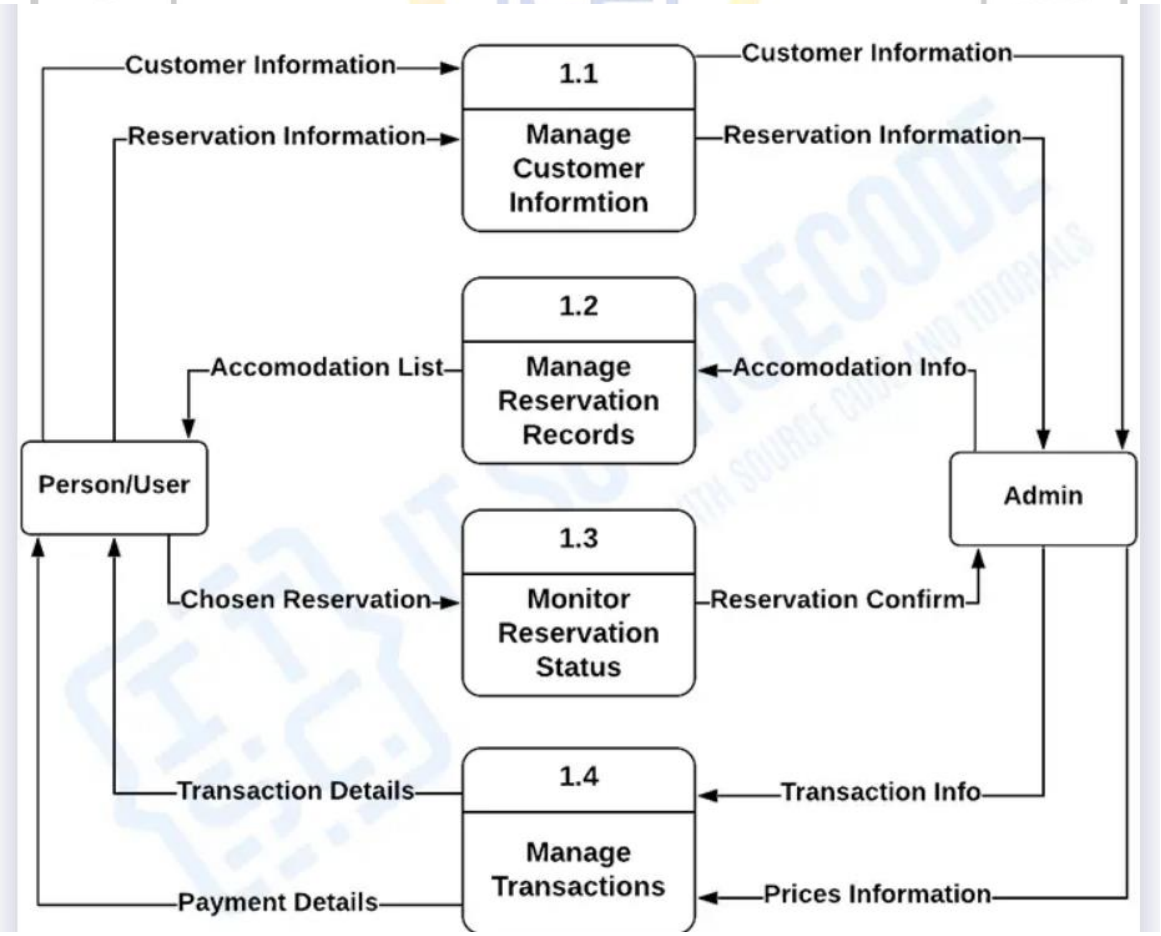
Airline Reservation System:

LEVEL 0 :

The Airline Reservation System DFD level 0 is also known as context diagram. It's supposed to be an abstract view, with the mechanism represented as a single process with external parties. This DFD for the Airline Reservation System depicts the overall structure as a single bubble. It comes with incoming/outgoing indicators showing input and output data.

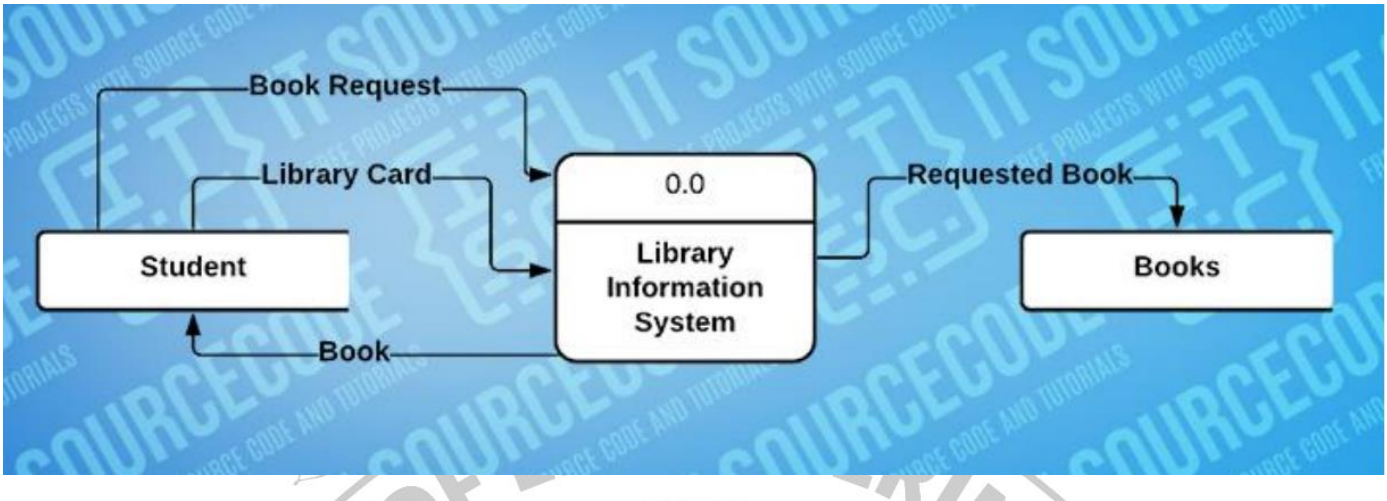


Level 1:

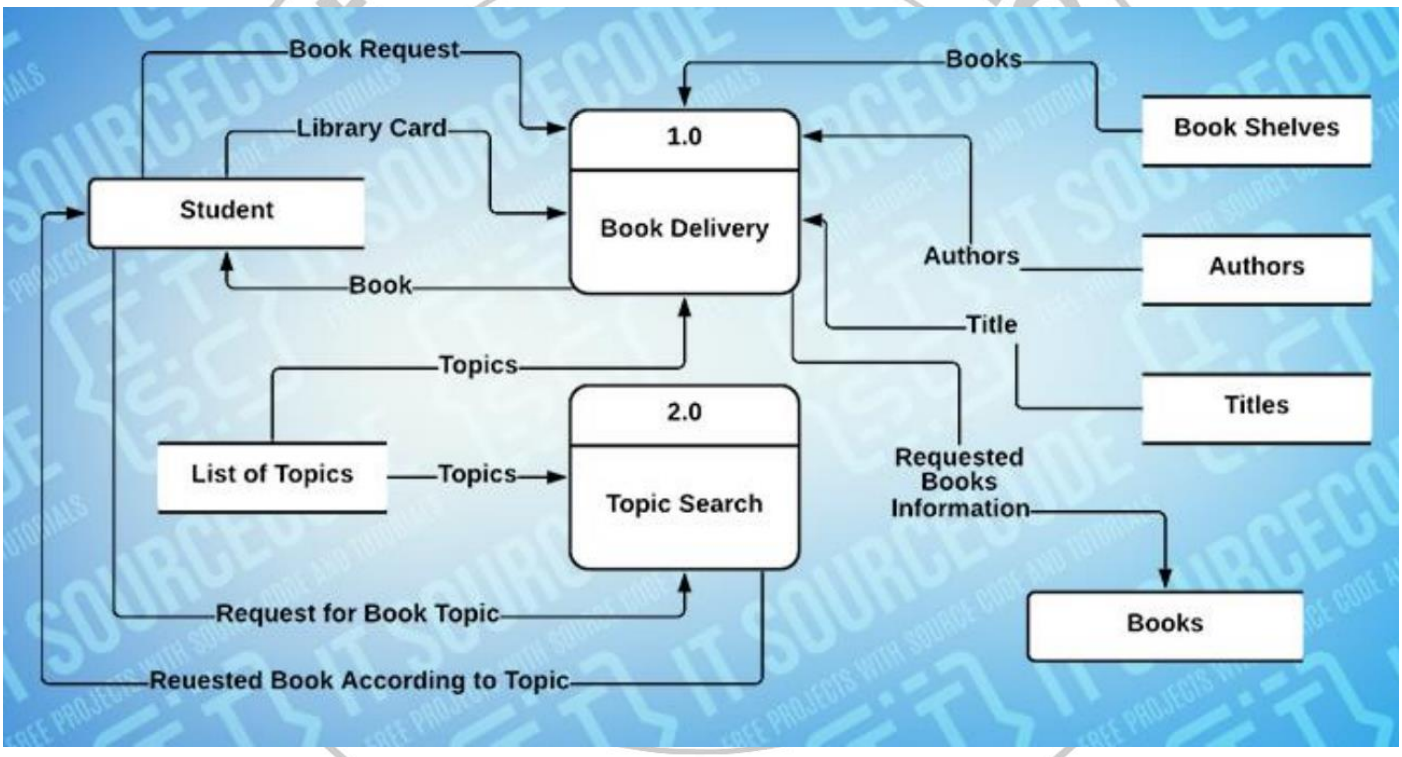


Library Management System:

Level 0:



Level 1:



Requirement Analysis of Travel Management System Project

Functional Requirements of Travel Management System Project

Admin Requirements:

The admin can add new packages.

The admin can update any package.

The admin can delete any package.

The admin can search any package.

The admin can also change his name or password from the dashboard settings.

End User Requirements:

The end users book tour plan according to their choice.

Users share their feedback with comments.

The user can sign up after filling all the fields mentioned in the sign-up form.

The user can login after validation of his/her details from the database.

The end users can reserve their seats for tour plan which they choose or like.

System Requirements:

Travel management system offer logout functionality to end users.

Travel management system will only accept a valid login detail to enrol on a travel Management System in PHP.

Travel management system will provide password recovery facility.

Travel management system will redirect the user to WhatsApp whenever the WhatsApp icon is pressed for online payment purposes.

Non-Functional Requirements of Travel Management System Project

Database Security:

An unauthorized person cannot access the panel and database, do not read, and write the information. It should maintain the security of the client's payment method.

Reservations Requirement:

Travel management system should reserve a travel package in maximum 30 to weekly evaluation by the project guide.

Reliability Requirement:

Travel management system should provide a reliable environment to both customers and owner.

Admin should be able to upload delete update new packages without any error.

Usability Requirement:

The travel Management System in PHP is designed for user friendly environment and ease of use.

Availability:

The travel Management System in PHP should be available for 24 hours because it offers international tourists reserved packages from different countries so it should be available for 24 hours.

Efficiency Requirement:

When an online package of travel implemented customer can have reserved packages in an efficient manner.

Functional and Non Functional Requirements for E-Commerce System:

Functional Requirements for E-commerce system-

1. **Login/ Registration**

Customers can register using a valid email id. Only one account can be created with an associated email id.

Using the UID and password, customers can login. If someone forgets password, OTP can be generated and sent on registered email ID and mobile number.

2. **Adding Items to cart**

Customer can add items from various sections to the cart. They can add maximum 25 items to cart at one time.

3. **Adding items to wish list**

Customers can add atleast 1000 items to wish list. They can also create sub groups within their wish list. Wish list can also be shared thru whatsapp/ email to other friends.

4. **Payment of Bill**

Customers can pay via various payment modes like the e-wallets – Paytm, GooglePay, Debit cards, Credit cards. For particular items they can also pay via cash on delivery. If the amount of bill is lesser than 500 rupees, delivery charges would be added to the bill. For orders above 500, the delivery charges would be NIL.

5. **Adding feedback**

Customers can give feedback to the items bought using images/videos and textual description.

Non-Functional Requirements for E-commerce system-

1. **Security:**

For Data encryption, the application would be using a Caesar's cipher model.

2. **Availability:**

Application will be available thru multiple platforms like the Desktops, Mobile phones and Laptops. It shall also be available 24*7.

3. **Compatibility:**

The application shall be compatible with all older versions of windows, Mac operating systems. It shall also be compatible with older versions of Chrome browsers.

4. **Recovery:**

The application shall have its entire data backup in 2 servers. Backup shall be taken every day mid night. Hence users can be relieved about the data being lost or corrupted.

5. **Scalability:**

The application shall be able to scale itself up to 1 lakh customers at the same time.