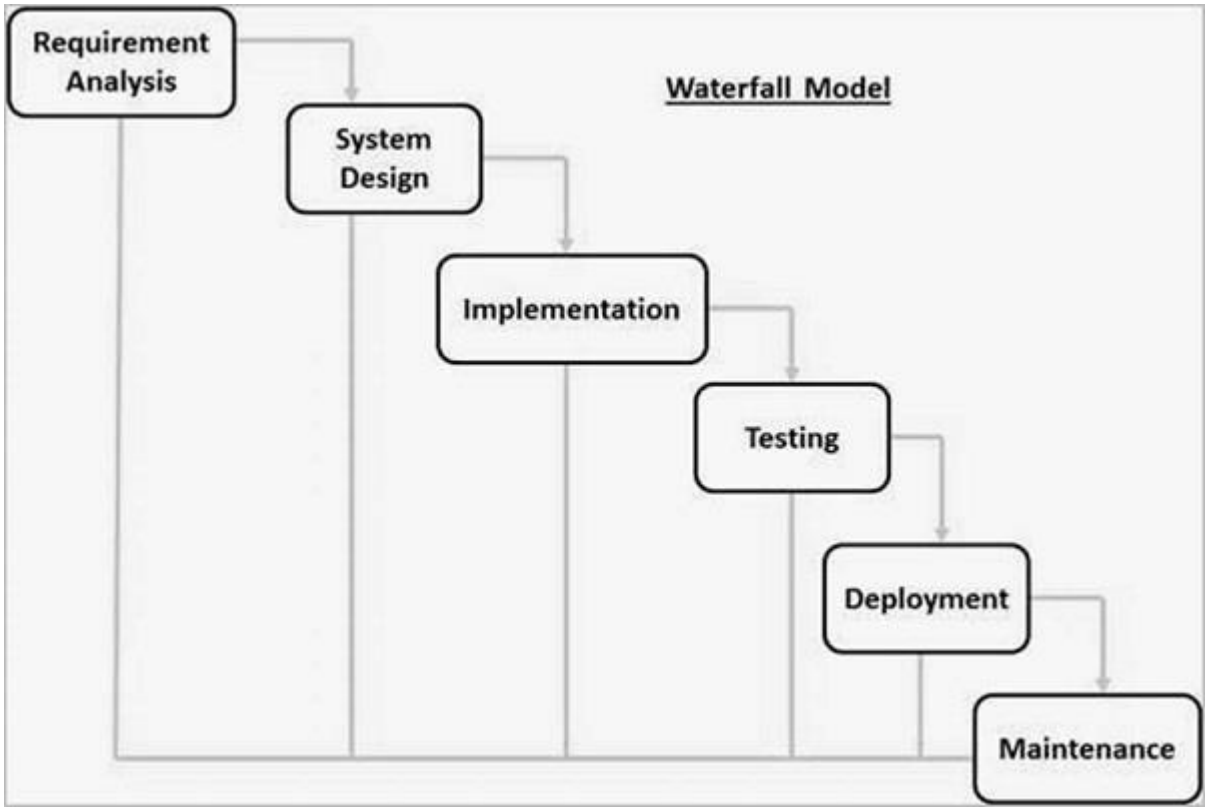
SDLC

Assignment-1

Create a one-page infographic that outlines the SDLC phases (Requirements, Design, Implementation, Testing, Deployment), highlighting the importance of each phase and how they interconnect.



Stage1: Requirement Analysis

Requirement Analysis is the most important and necessary stage in SDLC.

The senior members of the team perform it with inputs from all the stakeholders and domain experts or SMEs in the industry.

Planning for the quality assurance requirements and identifications of the risks associated with the projects is also done at this stage.

Business analyst and Project organizer set up a meeting with the client to gather all the data like what the customer wants to build, who will be the end user, what is the objective of the product. Before creating a product, a core understanding, or knowledge of the product is very necessary.

Once the requirement is understood, the SRS (Software Requirement Specification) document is created. The developers should thoroughly follow this document and should be reviewed by the customer for future reference.

Stage2: System Design

SRS is a reference for software designers to come up with the best architecture for the software. Hence, with the requirements defined in SRS, multiple designs for the product architecture are present in the Design Document Specification (DDS).
This DDS is assessed by market analysts and stakeholders. After evaluating all the possible factors, the most practical and logical design is chosen for development.

Stage3: Implementation

At this stage, the fundamental development of the product starts. For this, developers use a specific programming code as per the design in the DDS. Hence, it is important for the coders to follow the protocols set by the association. Conventional programming tools like compilers, interpreters, debuggers, etc. are also put into use at this stage. Some popular languages like C/C++, Python, Java, etc. are put into use as per the software regulations.

Stage4: Testing

After the development of the product, testing of the software is necessary to ensure its smooth execution. Although, minimal testing is conducted at every stage of SDLC. Therefore, at this stage, all the probable flaws are tracked, fixed, and retested. This ensures that the product confronts the quality requirements of SRS.

Stage-5: Deployment and Maintenance of Products
After detailed testing, the conclusive product is released in phases as per the organization's strategy. Then it is tested in a real industrial environment. It is important to ensure its smooth performance. If it performs well, the organization sends out the product. After retrieving beneficial feedback, the company releases it as it is or with auxiliary improvements to make it further helpful for the customers.

Assignment-2

Develop a case study analysing the implementation of SDLC phases in a real-world engineering project. Evaluate how Requirement Gathering, Design, Implementation, Testing, Deployment, and Maintenance contribute to project outcomes.

Case Study: Food Delivery App

1.Requirement Gathering:

The team starts by gathering requirements from stakeholders such as the app owner, restaurant partners, delivery personnel, and end users. Key features may include user registration/login,

restaurant listings with menus, order placement and tracking, payment integration, ratings and reviews, delivery tracking, and customer support.

2.Design:

Based on the gathered requirements, the team designs the app's architecture, user interface (UI), and user experience (UX). This includes wireframes/mockups for different screens, database schema for storing user and order data, API integrations for payment gateways and maps for delivery tracking, and algorithms for order matching and route optimization.

3.Implementation:

Developers start coding the app using appropriate technologies such as Swift or Kotlin for native iOS/Android development, React Native for cross-platform development, or a combination of frontend (UI) and backend (API) technologies. Agile methodologies are often used, with sprints, daily standups, and continuous integration/deployment (CI/CD) pipelines for faster and iterative development.

4.Testing:

The testing team performs various tests such as unit testing for individual components, integration testing for API interactions, system testing for end-to-end flows, and user acceptance testing (UAT) with real users to validate the app's functionalities, performance, security, and usability across different devices and platforms.

5.Deployment:

Once testing is successful, the app is deployed to app stores (e.g., Apple App Store, Google Play Store) for public access. The deployment process includes app store submissions, configuration of production servers, setting up monitoring tools for performance tracking, and ensuring scalability to handle user traffic.

6.Maintenance:

After deployment, the maintenance phase begins. The team monitors app performance, handles bug fixes reported by users or detected through monitoring tools, implements feature enhancements based on user feedback and market trends, and ensures data security and compliance with regulations (e.g., GDPR, PCI DSS) through regular updates and patches.

Assignment-3

Research and compare SDLC models suitable for engineering projects. Present findings on Waterfall, Agile, Spiral, and V-Model approaches, emphasizing their advantages, disadvantages, and applicability in different engineering contexts.

Waterfall Model:

Advantages:

Sequential Structure: It follows a linear and sequential approach, making it easy to understand and manage.

Documented Requirements: Each phase produces detailed documentation, facilitating clear communication and traceability.

Stability: Once a phase is completed, it's less likely to require revisions, ensuring stability in project planning.

Disadvantages:

Limited Flexibility: Lack of flexibility makes it challenging to accommodate changes or new requirements after the initial planning phase.

High Risk: Risks are identified late in the process, potentially leading to costly rework if issues are discovered during later stages.

Long Development Cycle: The sequential nature can result in longer development cycles, delaying time-to-market.

Applicability:

Waterfall is suitable for projects with well-defined requirements and stable technologies where changes are unlikely.

It's often used in projects with regulatory compliance requirements or where a detailed project plan is essential.

Agile Model:

Advantages:

Flexibility: Agile allows for iterative development and welcomes changes throughout the project, enhancing adaptability.

Customer Collaboration: Continuous customer involvement and feedback ensure the final product meets user expectations.

Quick Delivery: Incremental releases allow for faster delivery of valuable features, promoting early ROI.

Disadvantages:

Resource Intensive: Requires active collaboration among team members, stakeholders, and customers, which can be resource intensive.

Scope Creep: Without proper management, continuous changes may lead to scope creep and project drift.

Documentation Challenges: Agile prioritizes working software over comprehensive documentation, which can be challenging for regulatory compliance or knowledge transfer.

Applicability:

Agile is ideal for projects with evolving requirements, dynamic environments, and where customer feedback is crucial.

It's commonly used in software development but can also be applied in engineering projects requiring flexibility and frequent iterations.

Spiral Model:

Advantages:

Risk Management: Emphasizes risk assessment and mitigation throughout the project life cycle, reducing potential surprises.

Iterative Development: Iterative cycles allow for incremental development and refinement based on feedback and lessons learned.

Flexibility: Supports changes and adjustments during development, accommodating evolving requirements.

Disadvantages:

Complexity: The spiral model can be complex to manage, requiring experienced project managers and team members.

Resource Intensive: Iterative cycles involve multiple phases, which can be resource-intensive and time-consuming.

Documentation Challenges: Similar to Agile, emphasis on working prototypes may lead to less comprehensive documentation.

Applicability:

The Spiral model is suitable for large-scale projects with high complexity and significant risks.

It's often used in projects where risk management, iterative development, and flexibility are critical, such as software with evolving technologies or safety-critical systems.

V-Model:

Advantages:

Early Testing: Emphasizes testing activities early in the development life cycle, leading to early issue identification and resolution.

Traceability: Ensures traceability between requirements and test cases, facilitating comprehensive test coverage.

Structured Approach: Offers a structured and systematic approach, combining elements of waterfall (sequential phases) with testing emphasis.

Disadvantages:

Rigidity: Similar to Waterfall, V-Model can be rigid and less adaptable to changes or iterations.

Documentation Overload: Emphasizes documentation, which can become overwhelming and time-consuming if not managed efficiently.

Limited Customer Involvement: Less emphasis on continuous customer collaboration compared to Agile, potentially leading to misalignment with user needs.

Applicability:

The V-Model is suitable for projects with well-defined requirements and where thorough testing and quality assurance are critical.

It's often used in engineering projects with regulatory compliance requirements, safety-critical systems, or where traceability and documentation are paramount.