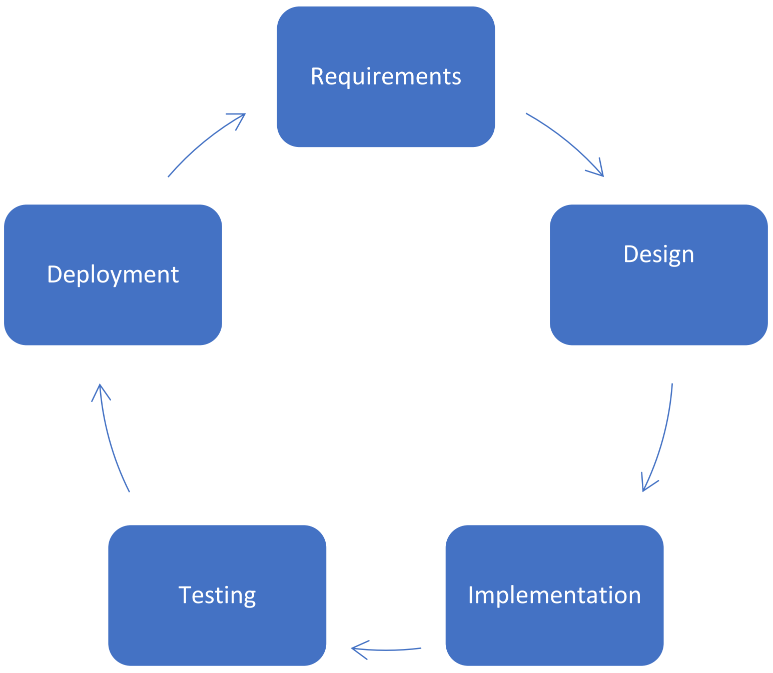
Assignment1: SDLC Overview - Create a one-page infographic that outlines the SDLC phases (Requirements, Design, Implementation, Testing, Deployment), highlighting the importance of each phase and how they interconnect.



- **Requirements Phase:**
    - **Importance:** This phase is crucial because it sets the foundation for the entire project by clearly defining what the software should do and how it should work.
    - **Connection:** The requirements gathered in this phase drive the design and development process. They ensure that the final product meets the needs and expectations of the people involved.
- **Design Phase:**
    - **Importance:** Designing the architecture and user interface of the software is important to ensure that it is scalable, maintainable, and user-friendly.
    - **Connection:** The designs are based on the requirements gathered in the previous phase. They provide a blueprint for the implementation phase and guide the developers in building the software according to the specifications.
- **Implementation Phase:**
    - **Importance:** This is where the actual coding of the software takes place. It transforms the design into working code.
    - **Connection:** The implementation is directly influenced by the design phase. Developers refer to the design documents to write code that aligns with the intended architecture and functionality.

- **Testing Phase:**
  - **Importance:** Testing ensures that the software meets quality standards and functions as intended. It helps identify and fix any defects or issues before deployment.
  - **Connection:** The test cases are derived from the requirements documented in the first phase. Testing validates whether the implemented software meets these requirements and ensures that it aligns with the design specifications.

- **Deployment Phase:**
  - **Importance:** Deployment is the process of releasing the software for use by end-users. It involves installing, configuring, and making the software operational in the production environment.
  - **Connection:** Successful deployment relies on the completion of all previous phases. A well-designed, thoroughly tested software product is essential for a smooth deployment process.

Assignment2: 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.

Let's consider a case study of developing a hotel booking app, analyzing the implementation of SDLC (Software Development Life Cycle) phases and their contribution to the project outcomes.

1. **Requirements Gathering Phase:** In this phase, the development team would gather requirements from various stakeholders, such as hotel owners, travelers, and other potential users. This could involve conducting interviews, surveys, or workshops to understand the desired features, functionalities, and user experience.

   Example: Through discussions with hotel owners and travelers, the team identified key requirements like the ability to search for hotels based on location, dates, and pricing, view room details and amenities, make reservations, and manage bookings.

2. **Design Phase:** Based on the gathered requirements, the team would design the architecture, user interface, and data models for the hotel booking app. This phase would involve creating wireframes, prototypes, and detailed design documents. Example: The design phase could involve creating a user-friendly interface with intuitive search filters, detailed hotel and room information pages, a secure payment gateway, and a user account section for managing bookings.

3. **Implementation Phase:** In this phase, developers would write the actual code for the hotel booking app, following the designs and specifications from the previous phase. They would implement features, integrate third-party services (e.g., payment gateways), and ensure the app functions as intended.

   Example: Developers would build the front-end user interface using technologies like React or Angular, implement server-side logic using languages like Node.js or Java, and integrate with databases to store hotel and user data.

4. **Testing Phase:** Throughout the implementation phase, the team would conduct various types of testing, such as unit testing, integration testing, and user acceptance testing (UAT). This phase ensures that the app meets the defined requirements, functions correctly, and is free from critical bugs or defects.

   Example: The testing phase could involve automated tests for crucial functionalities like search, booking, and payment processes, as well as manual testing by a dedicated team or selected users to identify and resolve any usability issues or edge cases.

5. **Deployment Phase:** After thorough testing and ensuring the app meets quality standards, the team would deploy the hotel booking app to a production environment, such as a cloud platform or a web server. This phase includes activities like setting up the hosting infrastructure, configuring the app, and performing final checks.

   Example: The deployment phase could involve provisioning servers on a cloud platform like AWS or Azure, configuring the app to work with the production environment, and conducting final smoke tests to ensure a smooth launch.

6. **Maintenance Phase:** Once the app is deployed and in use, the maintenance phase begins. This involves monitoring the app's performance, addressing any reported issues or bugs, implementing security updates, and potentially adding new features based on user feedback or changing market demands.

   Example: The maintenance phase could involve tracking user feedback, fixing any reported bugs or performance issues, implementing security patches, and planning for future enhancements like adding support for multiple languages or integrating with more payment gateways.

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: The Waterfall model follows a linear, sequential paradigm, where each phase of the software development life cycle is completed before moving on to the next phase. The phases are distinct and do not overlap, with the output of one phase serving as the input for the next phase.

- Advantages: Simple and easy to understand, well-defined phases, suitable for projects with well-defined requirements.

- Disadvantages: Inflexible, difficult to adapt to changing requirements, no working software until the end of the cycle.

- Applicability: Best suited for small-scale projects with stable requirements, such as firmware development or legacy system maintenance.

z

Agile Model: The Agile model is based on an iterative and incremental paradigm. It emphasizes flexibility, collaboration, and adaptability to changing requirements. Instead of following a rigid, sequential approach, Agile encourages frequent iterations, continuous feedback, and the delivery of working software in short cycles.

- Advantages: Flexible and adaptive, encourages customer collaboration, delivers working software quickly, suitable for changing requirements.

- Disadvantages: Requires experienced team members, less documentation, may not be suitable for projects with strict regulatory requirements.

- Applicability: Ideal for projects with rapidly changing requirements, such as web applications, mobile apps, or software products with frequent updates.

Spiral Model: The Spiral model adopts a risk-driven paradigm. It combines elements of both sequential and iterative approaches, with a focus on risk analysis and mitigation. The model follows a spiral pattern, where each cycle or iteration involves planning, risk assessment, prototyping, and evaluation before proceeding to the next cycle.

- Advantages: Incorporates risk management, allows for progressive development, supports changing requirements.

- Disadvantages: Complex and costly, requires highly skilled personnel, may not be suitable for small projects.

- Applicability: Well-suited for large-scale, complex projects with high risk, such as critical systems development or software projects with stringent quality requirements.

V-Model: The V-Model follows a verification and validation paradigm. It emphasizes the importance of testing and validation at each stage of the software development life cycle. The model resembles the shape of a "V," with one side representing the development phases (requirements, design, implementation) and the other side representing the corresponding testing and validation phases.

- Advantages: Focuses on testing and validation, suitable for projects with well-defined requirements, provides a structured approach.

- Disadvantages: Inflexible to changing requirements, no working software until the end of the cycle, heavy documentation.

- Applicability: Appropriate for projects with strict compliance or regulatory requirements, such as safety-critical systems, medical device software, or projects with strict quality standards.