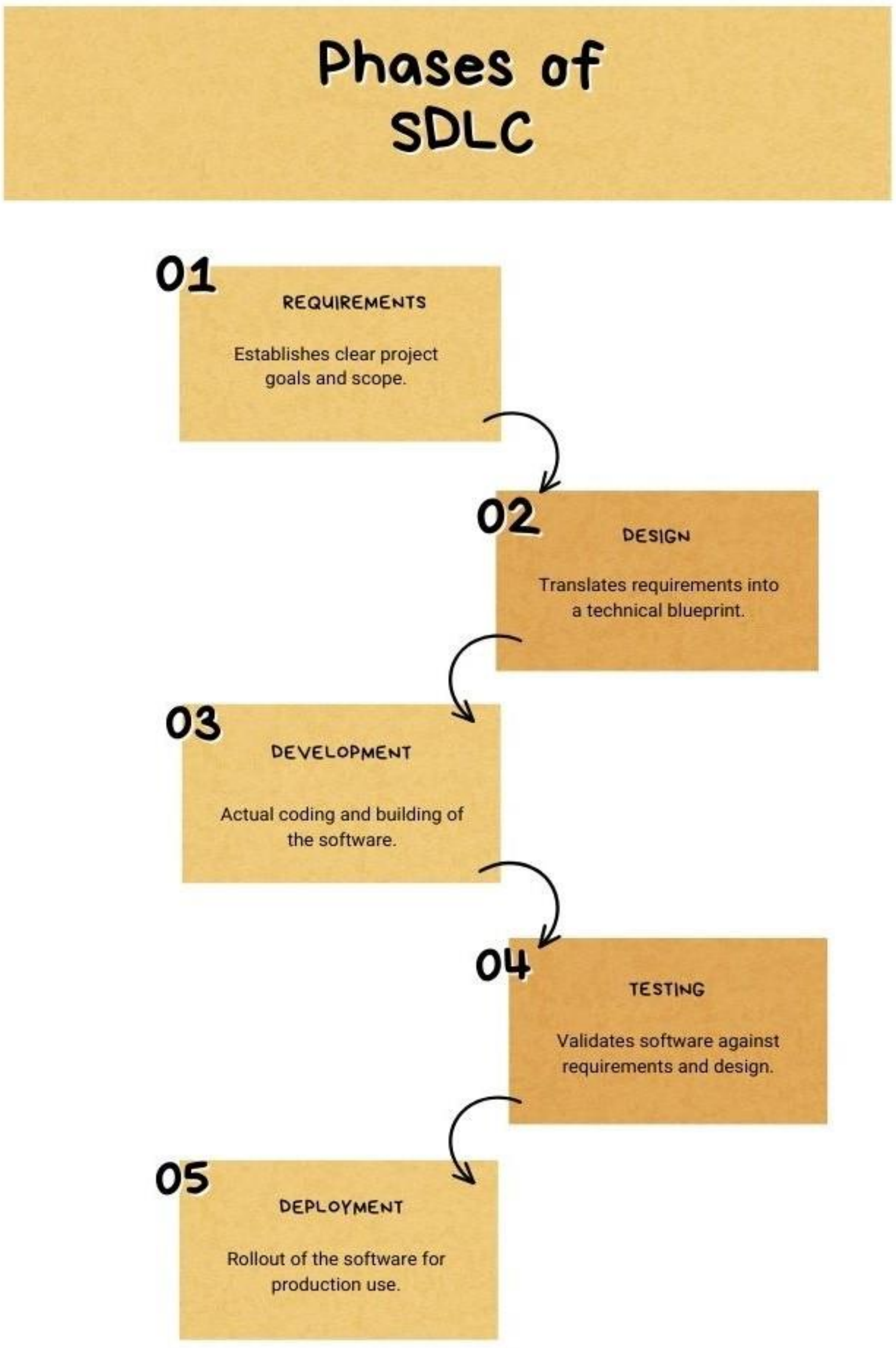
**SDLC Overview - Create a one-page infographic that outlines the SDLC phases (Requirements, Design, Implementation, Testing, Deployment), highlighting the importance of each phase and how they interconnect.**


Phases of SDLC
01
REQUIREMENTS
Establishes clear project goals and scope.
02
DESIGN
Translates requirements into a technical blueprint.
03
DEVELOPMENT
Actual coding and building of the software.
04
TESTING
Validates software against requirements and design.
05
DEPLOYMENT
Rollout of the software for production use.

## Assignment 2: Develop a case study analyzing 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:** Implementing SDLC Phases in a Software Engineering Project

Project Overview:

Company X is a software development firm specializing in building customer relationship management (CRM) solutions for medium to large enterprises. The company undertook a project to develop a new CRM platform aimed at enhancing user experience, scalability, and customization options.

SDLC Phases and Their Implementation:

### 1. Requirement Gathering:

At the onset of the project, the software development team collaborated closely with the client stakeholders, including business analysts and end-users, to gather comprehensive requirements. This phase involved conducting interviews, workshops, and surveys to understand user needs, pain points with existing systems, and desired features for the new CRM platform.

### 2. Design:

Following requirement gathering, the design phase commenced. The architecture and design team translated the collected requirements into technical specifications. This involved creating system diagrams, wireframes, and detailed design documents outlining the software's structure, interfaces, and database schema.

### 3. Implementation:

The development team then proceeded with coding based on the approved design. Agile methodologies were adopted to facilitate iterative development and responsiveness to changing requirements. The team adhered to coding standards, version control practices, and continuous integration processes throughout implementation.

**4. Testing:**

Testing was integrated into each phase, but in this phase, it was intensive. The QA team conducted various types of testing including unit testing, integration testing, system testing, and user acceptance testing (UAT). Test cases were developed based on requirements and executed systematically to identify defects.

**5. Deployment:**

Upon successful testing and client approval, the CRM platform was deployed to production environments. The deployment phase involved activities such as release planning, configuration management, and data migration from legacy systems.

**6. Maintenance:**

Following deployment, the CRM platform entered the maintenance phase. The development team continued to provide support, address bug fixes, and implement minor enhancements based on user feedback and evolving business needs.

**Project Outcomes and Contributions of SDLC Phases:**

**Quality Product Delivery:** By following a structured SDLC, Company X delivered a CRM platform that met or exceeded client expectations in terms of functionality, performance, and usability.

**Cost and Time Efficiency:** Early identification and resolution of requirements and design issues minimized rework, reducing project costs and time to market.

**Stakeholder Satisfaction:** Involving stakeholders throughout the SDLC ensured alignment with business objectives, resulting in high stakeholder satisfaction and adoption.

**Adaptability:** Agile practices allowed for flexibility in responding to changing requirements, ensuring the final product remained relevant in a dynamic business environment.

## 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.

### 1. Waterfall Model:

**Characteristics:** Linear, sequential approach with distinct phases (Requirements, Design, Implementation, Verification, Maintenance).

**Advantages:**

- Simple and easy to understand.
- Well-structured with defined milestones.
- Suitable for projects with stable requirements.

**Disadvantages:**

- Inflexible to changes once a phase is completed.
- Limited user involvement until late stages.
- High risk if requirements are not well understood initially.

**Applicability:** Best suited for projects with clear, fixed requirements and where changes are unlikely.

**2. Agile Model:**

**Characteristics:** Iterative and incremental, with continuous feedback loops and flexibility to adapt to changing requirements.

**Advantages:**

- Emphasizes customer collaboration and feedback.
- Flexible to changing requirements.
- Promotes early and frequent delivery of working software.

**Disadvantages:**

- Requires active customer involvement and frequent communication.
- May lead to scope creep if not managed properly.
- Can be challenging for large, distributed teams.

**Applicability:** Ideal for dynamic projects where requirements are likely to evolve and customer feedback is critical.

**3. Spiral Model:**

**Characteristics:** Iterative model combining elements of both waterfall and prototyping methodologies, focusing on risk assessment and mitigation.

**Advantages:**

- Risk-driven approach with early risk identification.
- Supports gradual building of prototypes and refined iterations.
- Allows for mid-course corrections.

**Disadvantages:**

- Complex management and evaluation of risks.
- Can be costly and time-consuming due to iterative nature.

- May require expertise in risk assessment.

**Applicability:** Suitable for large-scale, complex projects with high-risk elements that require frequent validations and adjustments.

**4. V-Model** (Verification and Validation Model):

Characteristics: Extension of the waterfall model, emphasizing verification and validation at each stage of development.

**Advantages:**

- Ensures early test planning and high test visibility.
- Enhances traceability between requirements and test cases.
- Rigorous approach to quality assurance.

**Disadvantages:**

- Can be rigid and difficult to accommodate changes.
- Testing activities are often seen as a bottleneck.
- Relies heavily on upfront planning and documentation.

**Applicability:** Best suited for projects where quality and verification are paramount, such as safety-critical systems.