SDLC

Topic: Software Development Life Cycle (SDLC)

Introduction:

Definition: SDLC is a structured process for developing software applications.

Purpose: Ensures the systematic and methodical development of software products.

Phases of SDLC:

- Requirements Gathering: Collecting and documenting user requirements.
- Planning: Defining project scope, timeline, and resource allocation.
- Design: Creating architecture, system design, and user interface.
- Implementation: Writing code based on design specifications.
- Testing: Verifying software functionality and quality assurance.
- Deployment: Releasing the software to production environment.
- Maintenance: Providing support, updates, and bug fixes post-deployment.

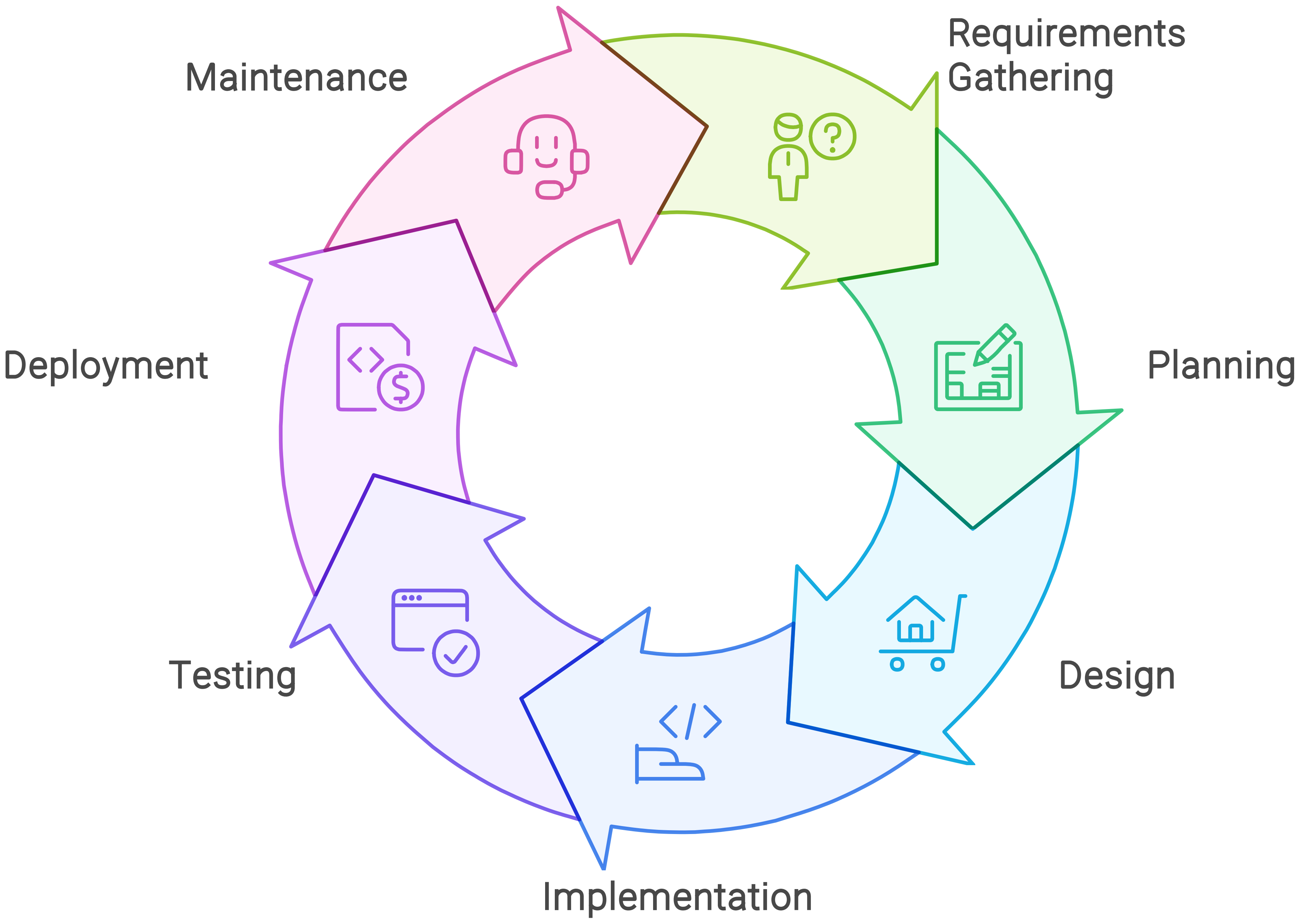
Key Principles:

Sequential process flow.

Emphasizes documentation and planning.

Limited flexibility for changes once development begins.

Software Development Life Cycle



Topic: Waterfall Model

Overview:

Sequential, linear approach to software development.

Phases progress downwards like a waterfall, with no backward movement.

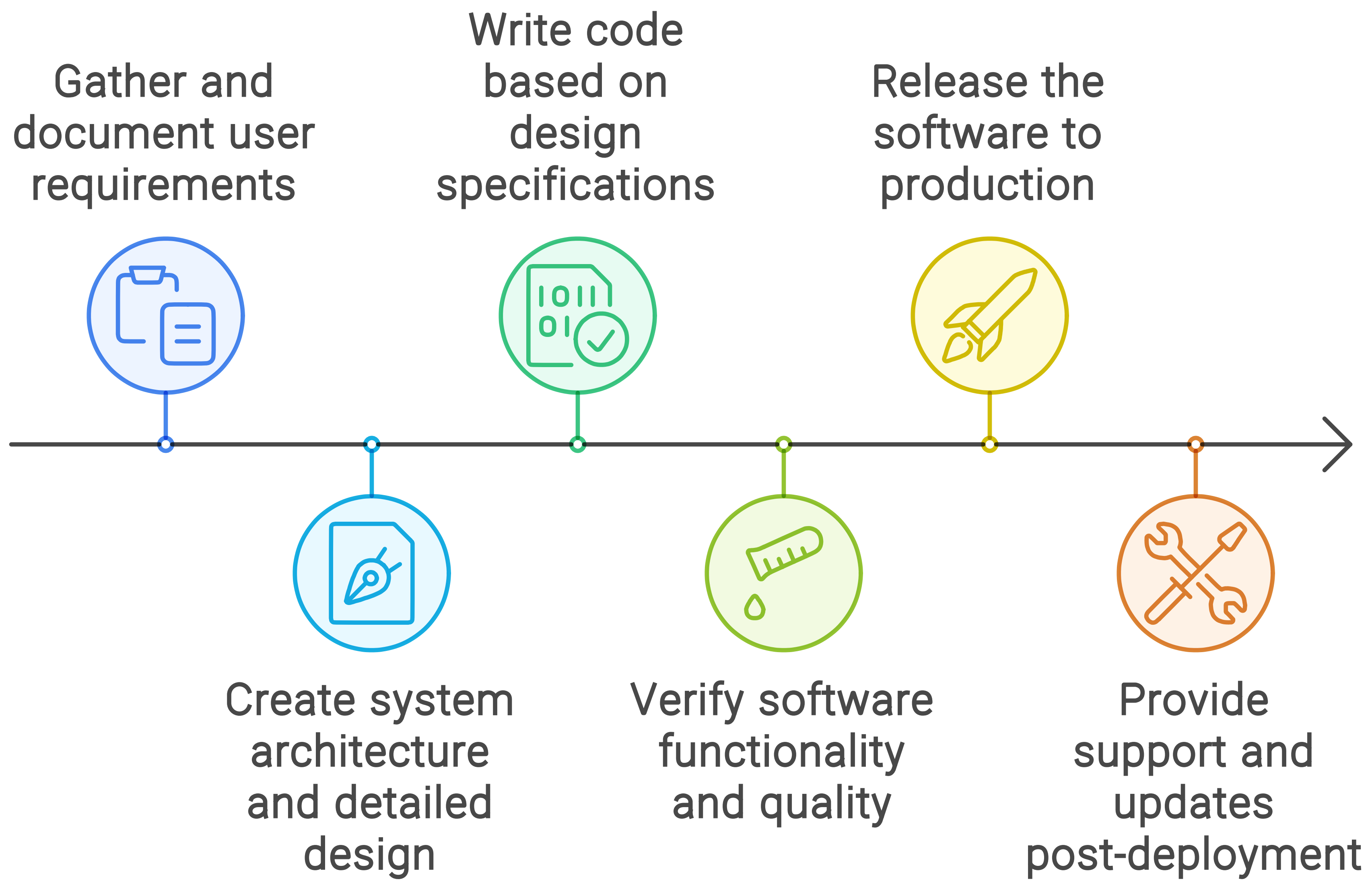
Phases of Waterfall Model:

- Requirements: Gather and document user requirements.
- Design: Create system architecture and detailed design.
- Implementation: Write code based on design specifications.
- Testing: Verify software functionality and quality.
- Deployment: Release the software to production.
- Maintenance: Provide support and updates post-deployment.

Advantages:

Simple and easy to understand.
Well-suited for projects with stable requirements.
Disadvantages:
Limited flexibility for changes.
High risk of project failure if requirements change.

Waterfall Model: A Sequential Approach to Software Development


Gather and document user requirements
Create system architecture and detailed design
Write code based on design specifications
Verify software functionality and quality
Release the software to production
Provide support and updates post-deployment

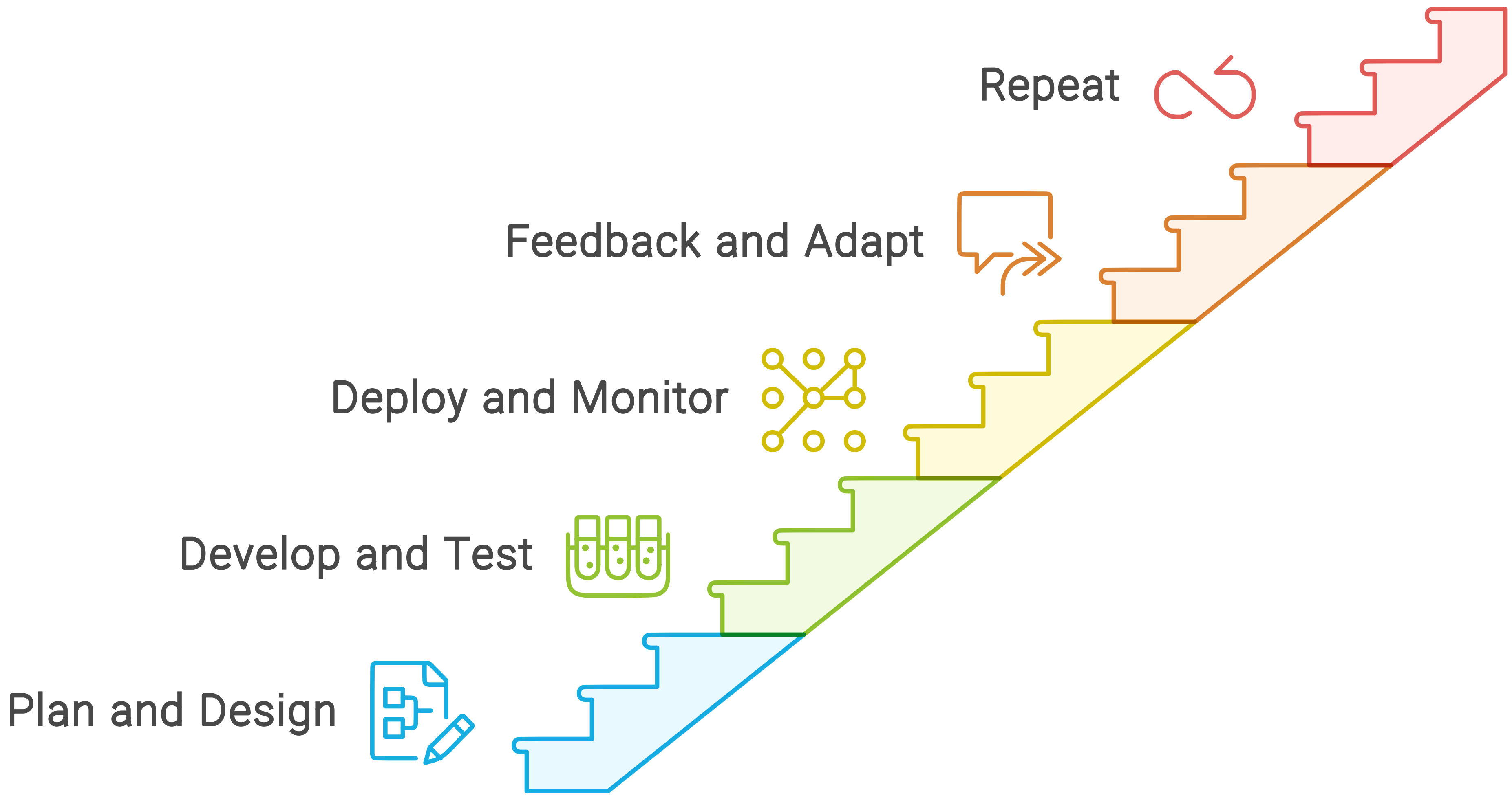
Topic: Agile Methodologies (Scrum, XP)
Overview:
Agile is an iterative and incremental approach to software development.
Values collaboration, adaptability, and delivering working software.
Scrum Methodology:
Framework for managing and organizing Agile projects.
Roles: Product Owner, Scrum Master, Development Team.
Artifacts: Product Backlog, Sprint Backlog, Increment.
Ceremonies: Sprint Planning, Daily Standup, Sprint Review, Sprint Retrospective.

Extreme Programming (XP):
Agile methodology focused on technical excellence and customer satisfaction.
Core practices: Test-driven development (TDD), Pair programming, Continuous integration (CI), Refactoring.
Values: Communication, Simplicity, Feedback, Courage, Respect.
Advantages:
Flexibility to adapt to changing requirements.
Emphasizes customer collaboration and feedback.
Disadvantages:
Requires high level of team discipline and communication.
May not be suitable for projects with fixed deadlines or requirements.

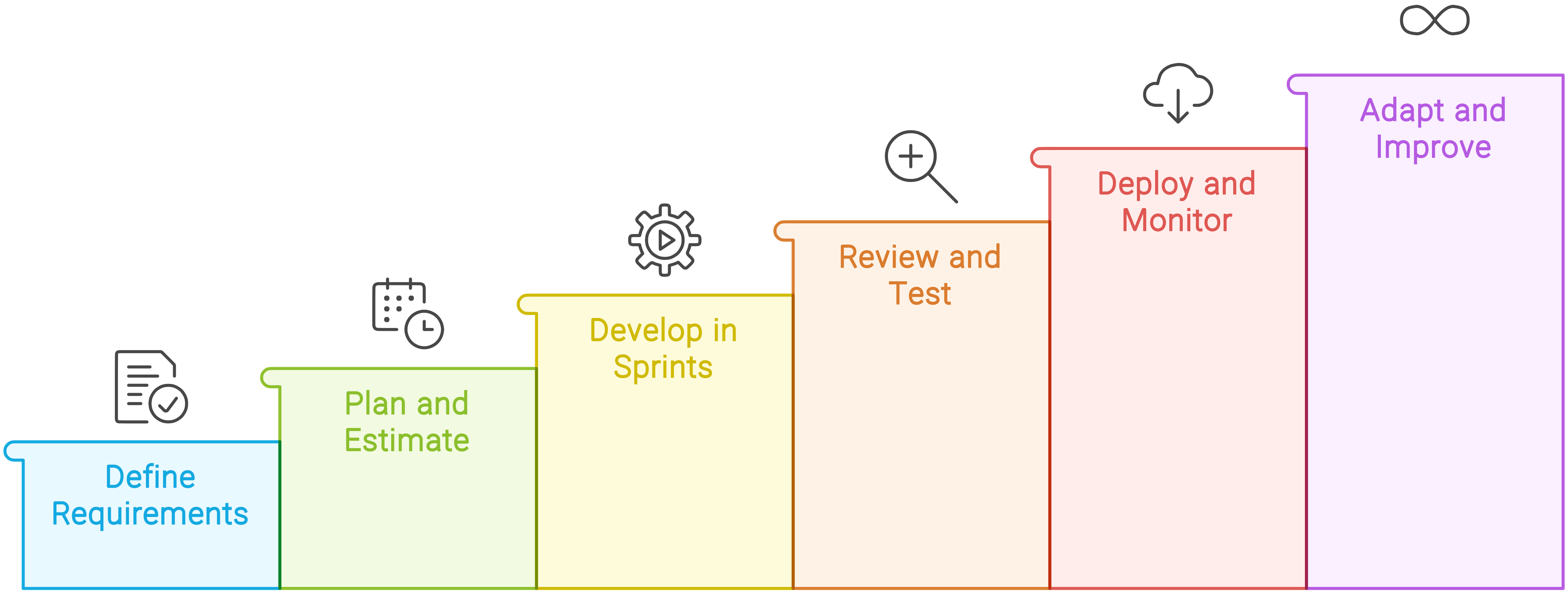
Topic: Iterative and Incremental Development
Overview:
Approach to software development that breaks down the project into smaller iterations or increments.
Each iteration produces a working product increment.
Key Concepts:

Iterative: Repeated cycles of development, testing, and feedback.
Incremental: Delivering functionality in small, usable increments.
Advantages:
Allows for early and continuous delivery of valuable software.
Facilitates adaptive planning and flexibility to change.
Disadvantages:
Requires frequent collaboration and communication.
May result in increased complexity if not managed properly.

Iterative and Incremental Development



Agile Software Development Methodologies


Define Requirements
Plan and Estimate
Develop in Sprints
Review and Test
Deploy and Monitor
Adapt and Improve