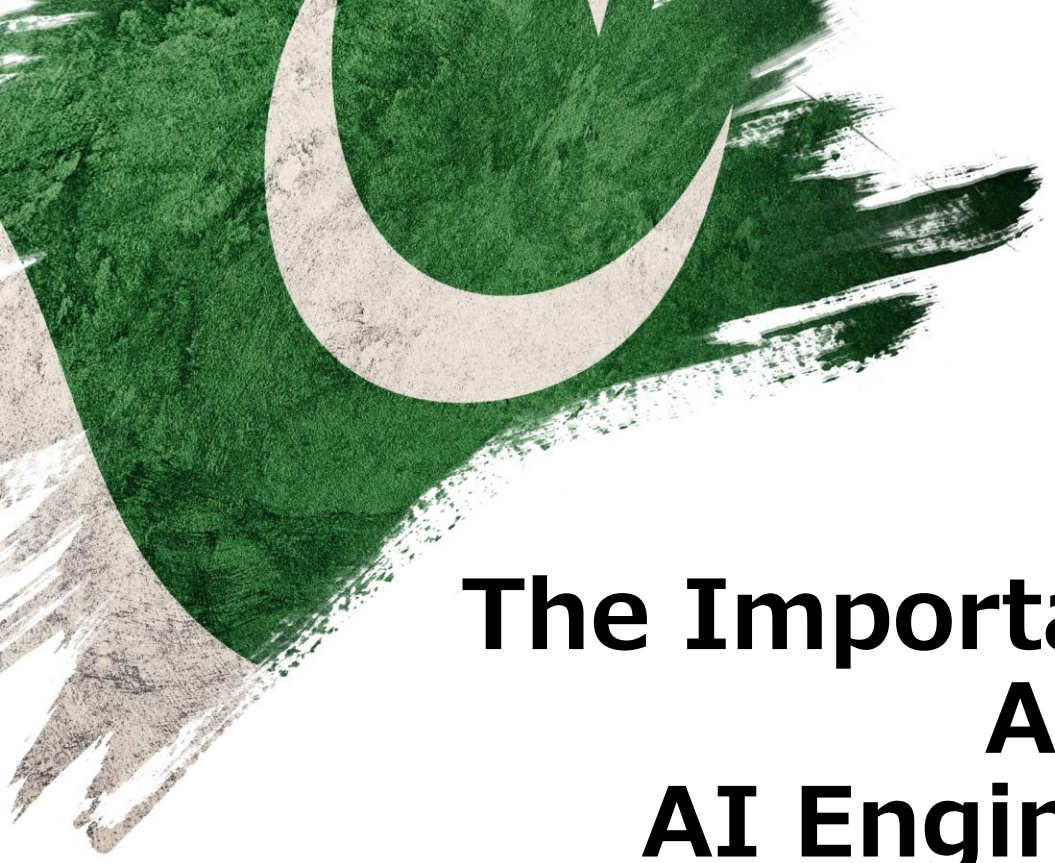


The Importance of Requirement Analysis for AI Engineer & Developers

Class 9

22/2/2025

Acknowledgement

The series of the IT & Japanese language course is Supported by AOTS and OEC.

Ministry of Economy, Trade and Industry

Overseas Employment Corporation

What you have Learnt Last Week

We were focused on following points.

- Usage of control and loop flow statement
- Performing Linear Algebra in Numpy
- Inspecting and Understanding Data
- Basics of creating, loading, and exploring DataFrames
- Array indexing and slicing
- Linear & Multi Linear Regression
- Support Vector Machine
- Cost Function

What you will Learn Today

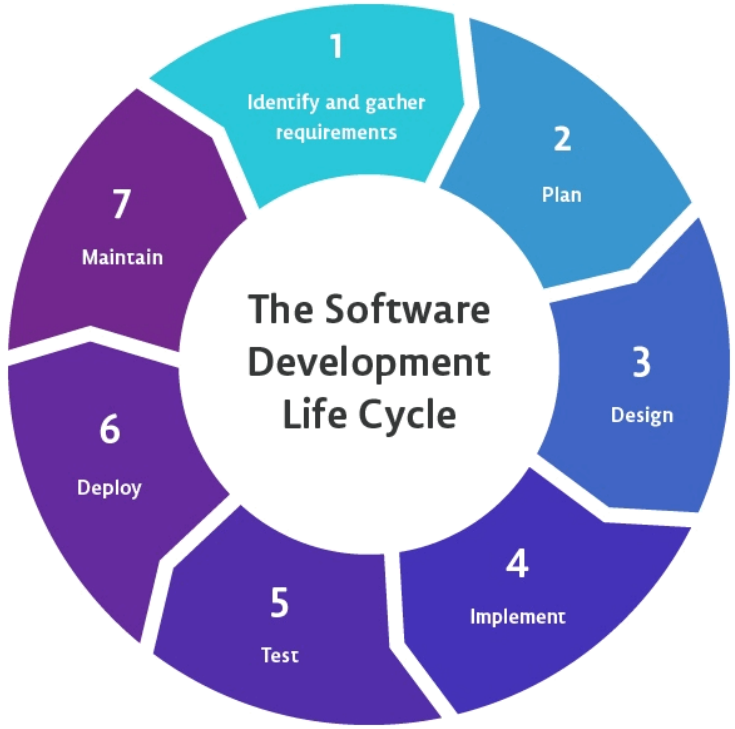
We will focus on following points.

- Why Requirement Analysis is so important in the process?
- Review case studies that demonstrate successful requirement analysis practices
- Quiz
- Q&A Session

Software Development Life Cycle

The Software Development Life Cycle (SDLC) is a structured process used to design, develop, test, and deploy software applications

[Phases of SDLC]



1. Requirement Analysis

1. Gather business and technical requirements.
2. Identify stakeholders and project scope.

2. Planning

1. Define project timeline, cost estimation, and resource allocation.
2. Identify risks and mitigation strategies.

3. Design

1. Create architectural and UI/UX design.
2. Define database structures and system components.

Software Development Life Cycle

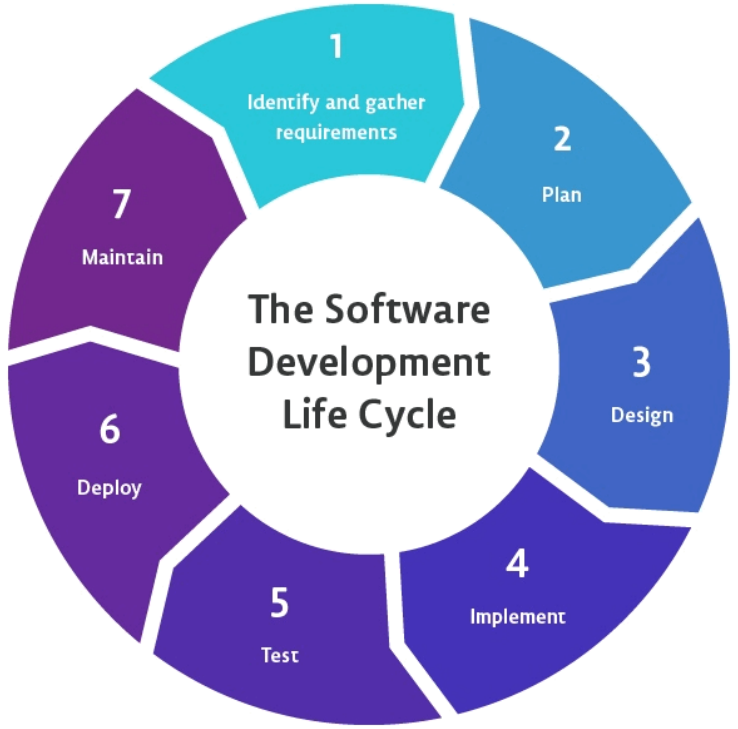
The Software Development Life Cycle (SDLC) is a structured process used to design, develop, test, and deploy software applications

4. Development (Implementation)
1. Code the software based on the design specifications.
2. Follow best practices, use version control, and implement coding standards.

5. Testing
1. Perform unit testing, integration testing, and system testing.
2. Identify and fix bugs to ensure software reliability.

6. Deployment
1. Release the software to production.
2. Implement CI/CD pipelines for automated deployment.

7. Maintenance & Support
1. Monitor performance, fix bugs, and update features as needed.



Requirement Analysis is one of the most important part, Not only Coding/Programming

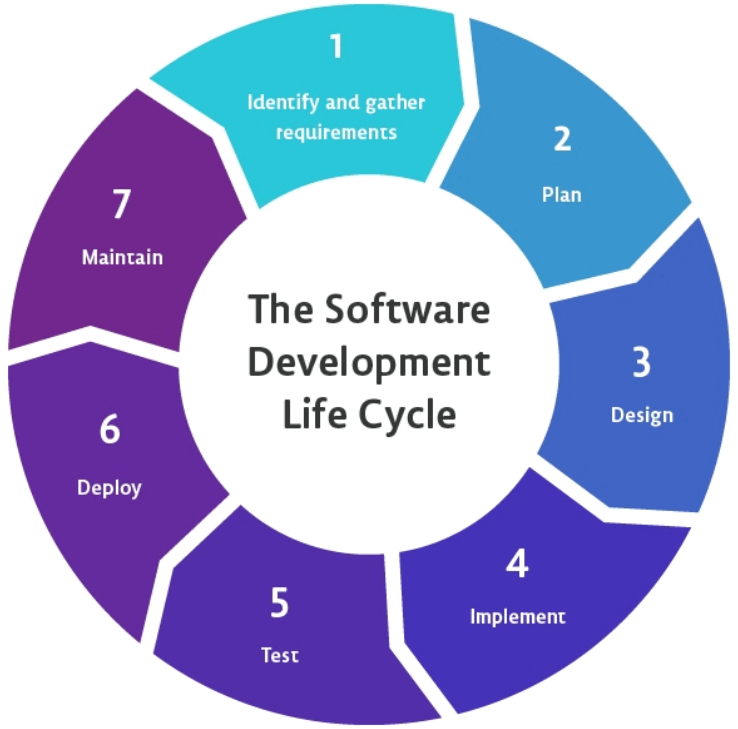
Requirement Analysis

It is the first and most crucial phase of the Software Development Life Cycle (SDLC)

[Gather Business and Technical Requirements]

- **[Business Requirements]**
  - Identify the **purpose** of the software.
  - Understand **business goals and objectives**.
  - Define expected **features and functionalities**.

- **[Technical Requirements]**
  - Determine **technology stack** (e.g., programming languages, databases, frameworks).
  - Define **performance, scalability, and security** requirements.
  - Identify **system integration needs** with existing software.



It involves gathering, analyzing, and defining the needs and expectations of stakeholders to ensure that the final software product meets business objectives

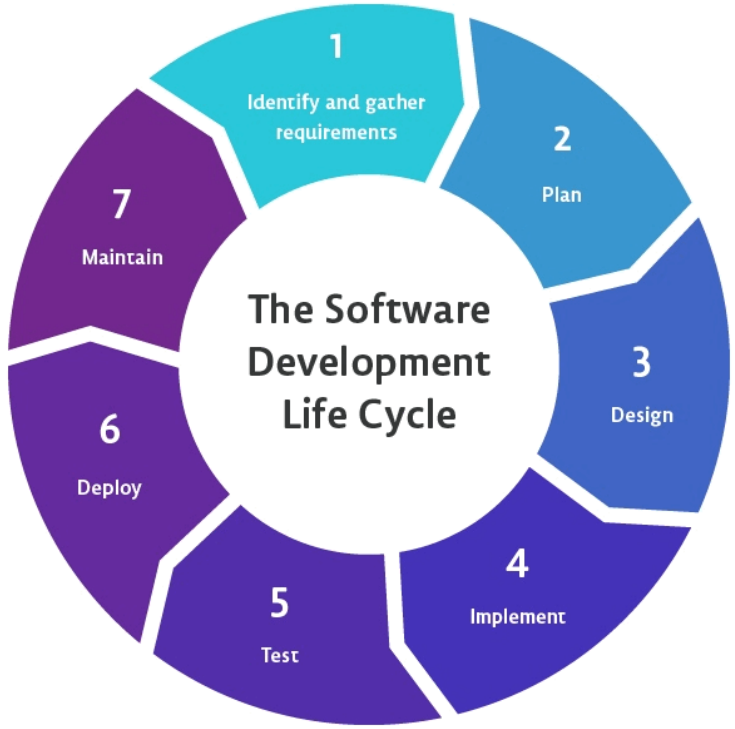
Requirement Analysis

It is the first and most crucial phase of the Software Development Life Cycle (SDLC)

[Identify Stakeholders and Project Scope]

- **Who are the stakeholders?**
  - **Clients/Customers** (who will use the software).
  - **Business Analysts** (who translate business needs into software requirements).
  - **Project Managers** (who oversee the project timeline and budget).
  - **Developers & Architects** (who build the system).
  - **Testers** (who ensure the system works correctly).

- **Define Project Scope:**
  - What **features** will be included and excluded?
  - What **platforms** will the software support (web, mobile, desktop)?
  - Define **constraints** (budget, timeline, regulatory compliance).



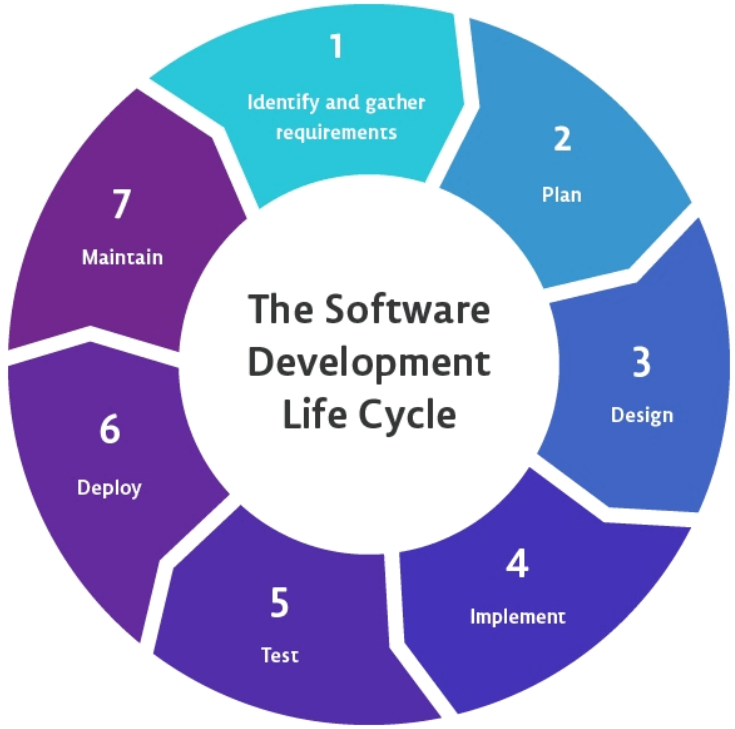
It involves gathering, analyzing, and defining the needs and expectations of stakeholders to ensure that the final software product meets business objectives

Requirement Analysis

It is the first and most crucial phase of the Software Development Life Cycle (SDLC)

[Outcome of Requirement Analysis]

- **Software Requirement Specification (SRS) Document**
  - A detailed document outlining functional and non-functional requirements.
- **Use Cases & User Stories**
  - Descriptions of how users will interact with the system.
- **Wireframes & Prototypes**
  - Basic visual representations of the system's user interface.

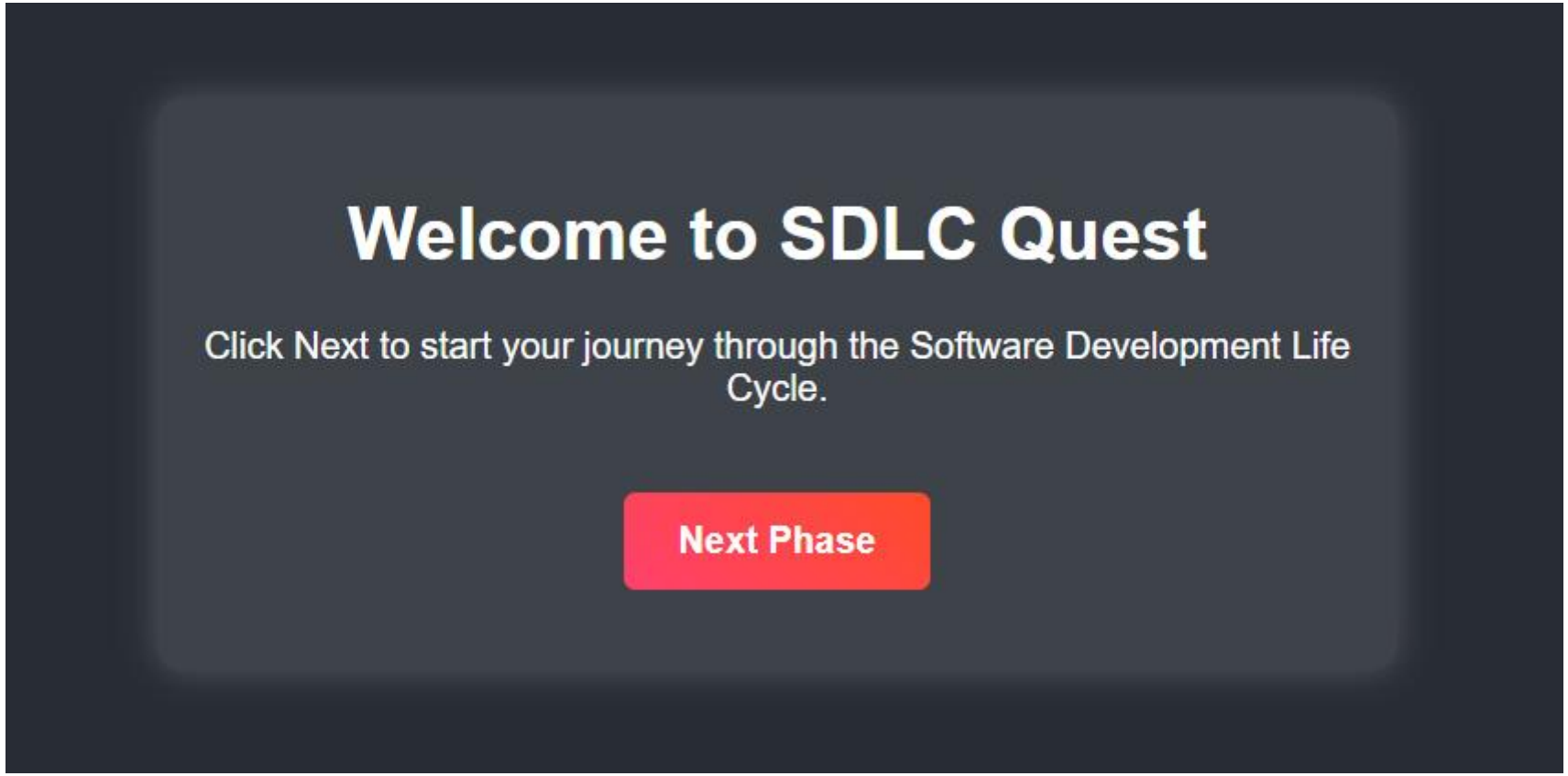


It involves gathering, analyzing, and defining the needs and expectations of stakeholders to ensure that the final software product meets business objectives

Task-1

SDLC Game

Let's learn the software development life cycle by playing the game



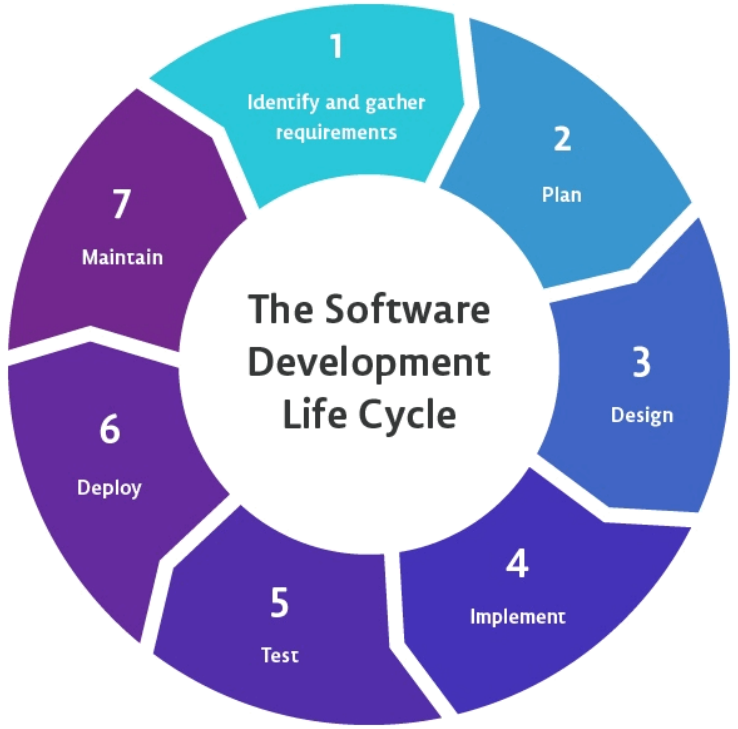
https://codepen.io/Mise-Academy/full/NPWYrNJ

Why Requirement Analysis is so important in the process?

When project Fail or Sprint Fail, most of Root Cause might come from Requirement Analysis

[If Requirement Analysis not done well]

- Project Schedule will be Delayed.
- Sprint Schedule will be Delayed.
- Developer need to work hard with Overtime



It involves gathering, analyzing, and defining the needs and expectations of stakeholders to ensure that the final software product meets business objectives

Case Study:1 Notification Requirement Gap

For example, the project started in September 2024, with delivery planned for October 2025.

[Project Timeline & Issue Discovery]

- The project started in **September**, with delivery planned for **October**.
- During the **testing phase in October**, the client identified a defect related to notifications.

[Development Team's Understanding]

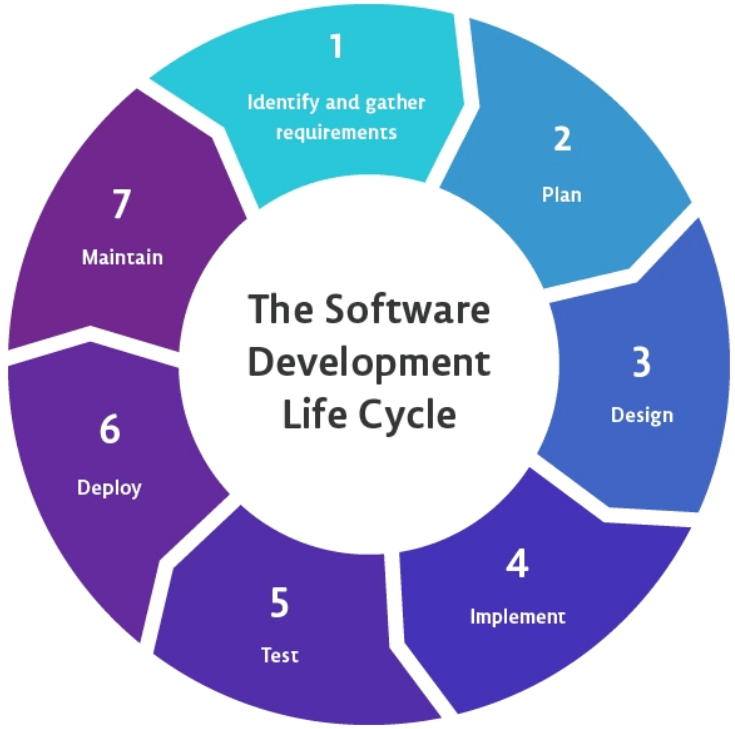
- The team believed the requirement was to **display notifications on the website** when users opened the web application.
- This functionality was already **working correctly**.

[Product Owner's Expectation]

- The **Product Owner (PO)** expected notifications to also be **sent to other systems** (e.g., via a webhook).
- This difference in understanding led to a **requirement gap**.

[Root Cause of the Issue]

- The development team **did not analyze** the product backlog in depth.
- This caused a misunderstanding between the **development team** and the **Product Owner**.



Case Study:1 Notification Requirement Gap

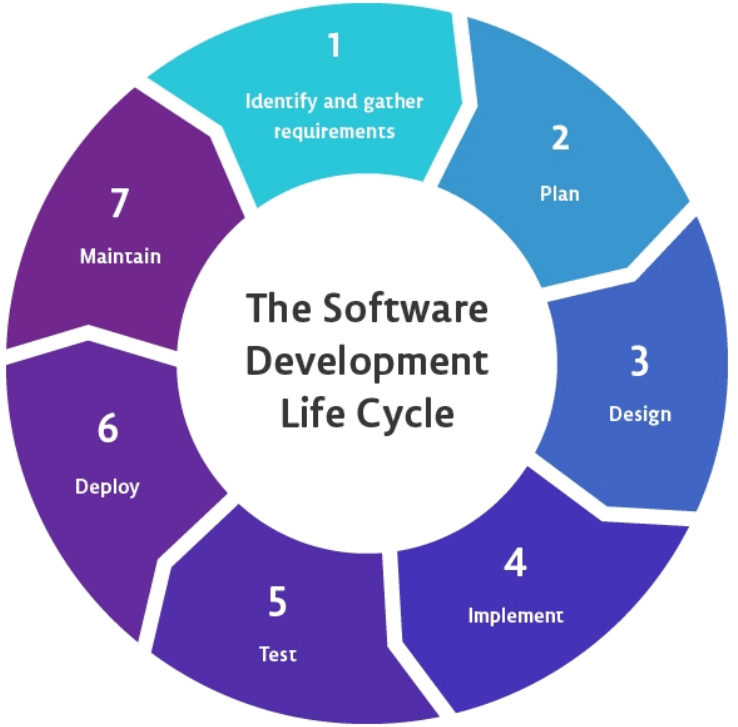
Key Learnings from this case study

✅ Always **clarify requirements** with stakeholders before development.

✅ Conduct a **detailed analysis** of the product backlog.

✅ Maintain **clear communication** between the development team and the Product Owner.

✅ Identify **potential integration points** early in the project.



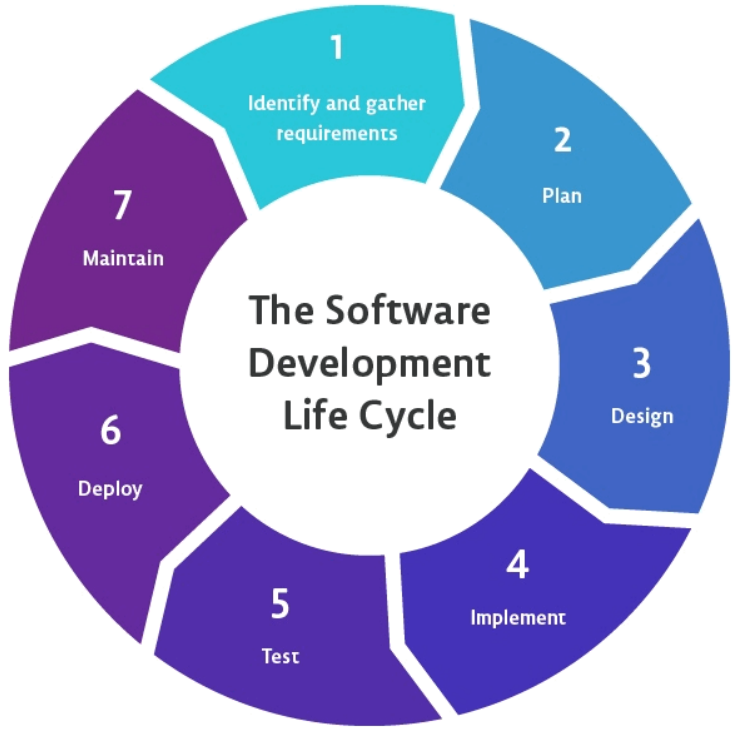
Case Study:2 Missing Backend System in Product Backlog

For example, the project started in September, with delivery planned for October.

[Initial Approach]
- The **Product Owner (PO)** created the product backlog focusing only on the **user-facing system**.
- The **development team** followed the backlog and began development.

[Requirement Gap]
- To fully implement the backlog requirements, a **backend web system (like a CMS)** was needed.
- However, the **PO lacked technical knowledge** and did not include the backend system in the backlog.

[Impact on Development]
- The development team **realized the missing backend system** late in the process.
- To adjust the plan, the team had to **design and develop the backend system**.
- This required working **overtime** to complete the project.



Case Study:2 Missing Backend System in Product Backlog

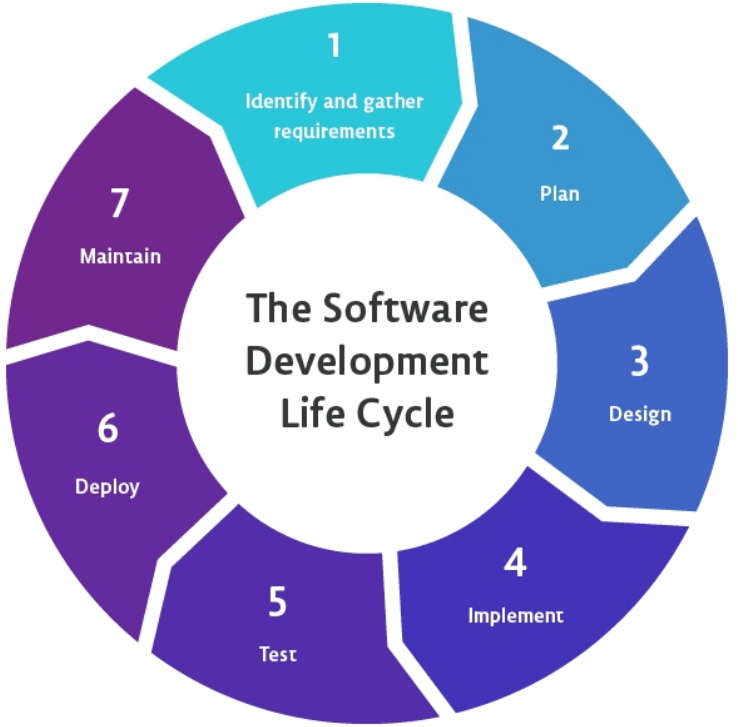
Key Learnings from this case study

✅ **Collaborate closely** with the PO to ensure all technical aspects are considered.

✅ Conduct **requirement validation** before starting development.

✅ Identify and document **both frontend and backend requirements** in the backlog.

✅ Avoid last-minute changes by ensuring **clear and complete project planning**.



Case Study:3 Missing Error Case Handling in Development

For example, the project started in September, with delivery planned for October.

[Initial Approach]

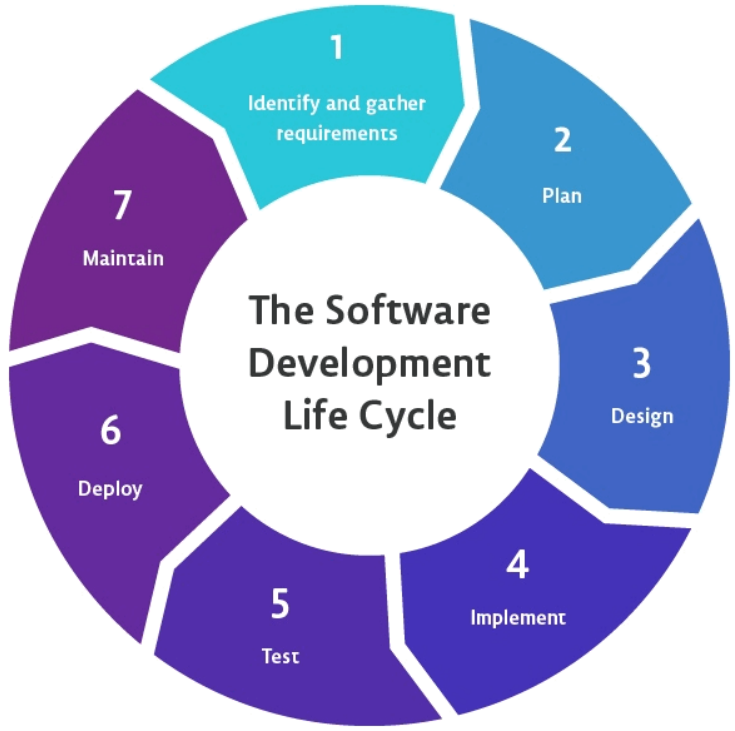
- The **Product Owner (PO)** created the product backlog.
- The **development team** reviewed the backlog but focused only on **normal use cases**.

[Requirement Gap]

- The team **did not consider error cases** during development.
- As a result, only the **normal case** was developed in the sprint.
- **Error handling** was left incomplete.

[Impact on the Project]

- The missing error handling **affected the next sprint's planning**.
- It also caused **delays in the overall project schedule**.



Case Study:3 Missing Error Case Handling in Development

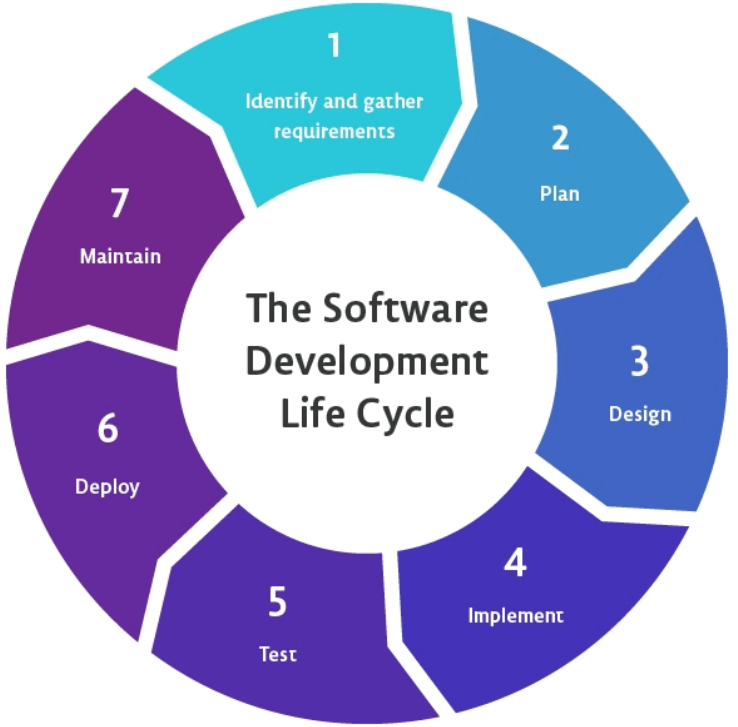
Key Learnings from this case study

✅ Always **analyze both normal and error cases** during development.

✅ **Test different scenarios**, including edge cases, before finalizing a sprint.

✅ Allocate time for **error handling** within the sprint planning.

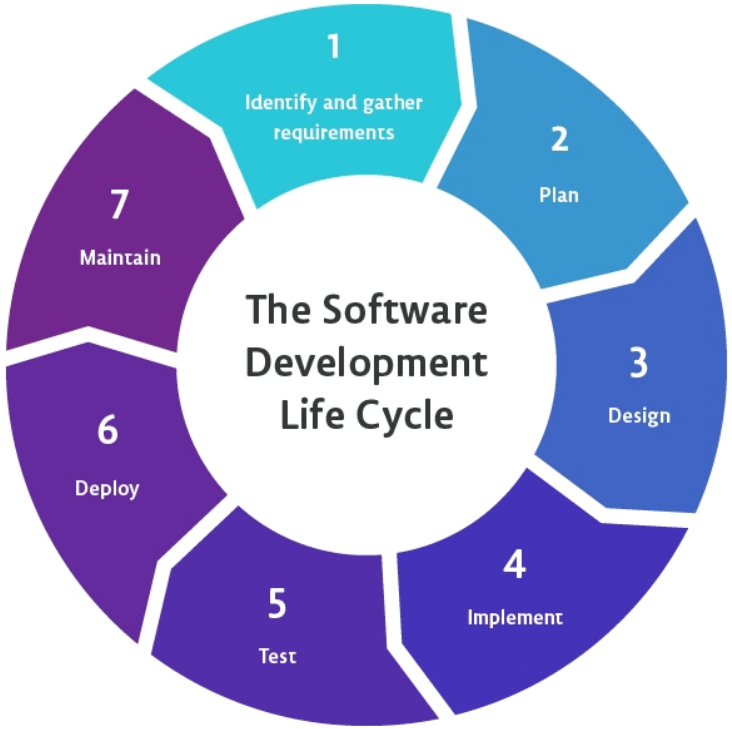
✅ Ensure **thorough backlog review** before development begins.



Summarize the Requirement Analysis

If you have Unclear, please do NOT hesitate to ask questions to PO/Business Side and have mtg with them

- Developer need to understand the **Business** of this project/this product.
- Developer need to understand the **Domain Knowledge** of this business
- Developer need to understand the **Project Overview** (Goal, Purpose, Schedule)
- Developer need to understand about **User**
- Developer need to understand about **Product Backlog Detail**

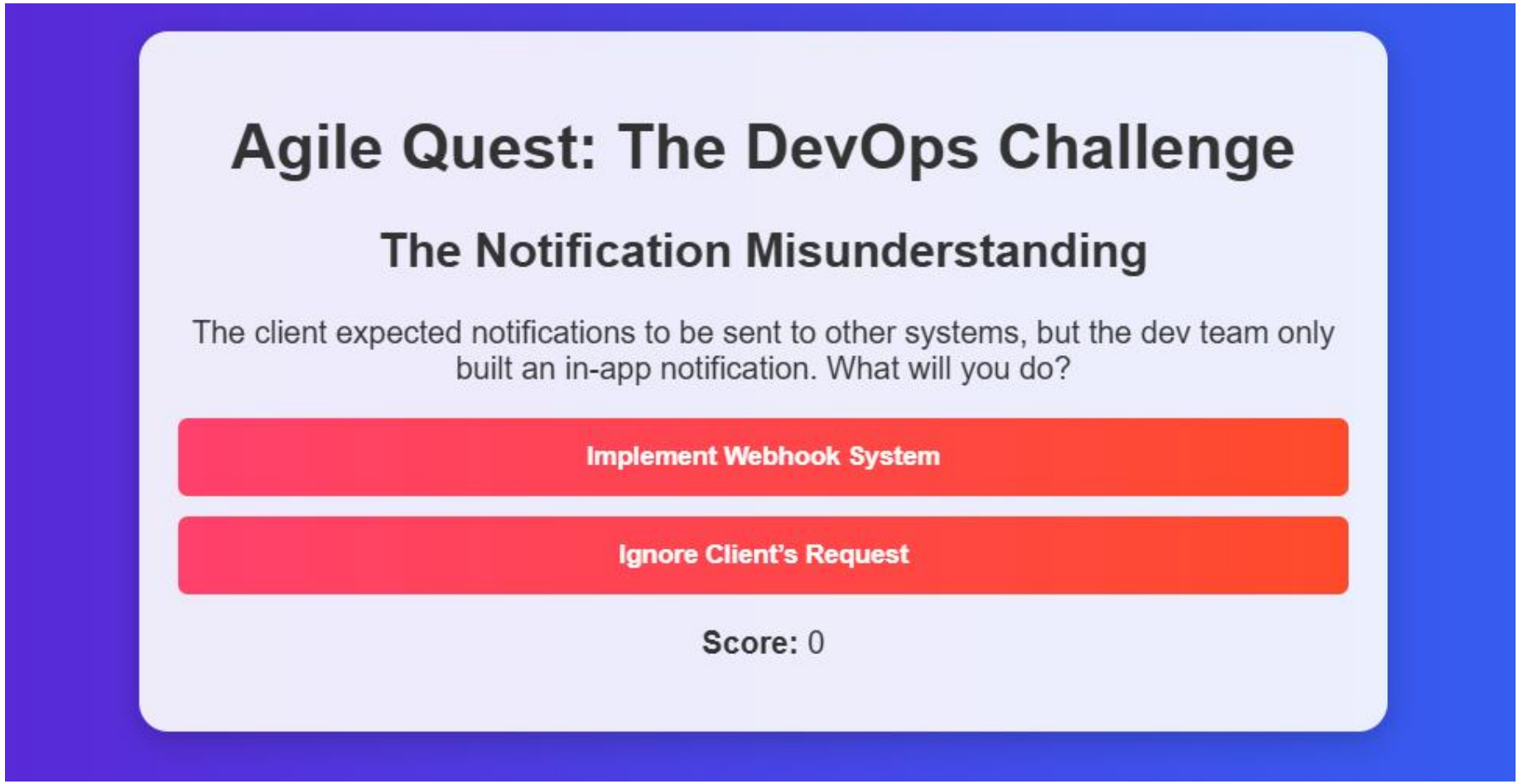


[Note]
Please do NOT believe PO well. Their document does not cover all cases. E.g. lacking of error case , lacking of non-function requirement(ex, batch)

Task-2

Requirement Analysis Game

Let's learn the Requirement Analysis with case study by playing the game



https://codepen.io/Mise-Academy/full/GgRdmJo

Planning Phase in SDLC

It focuses on creating a roadmap for the project by defining the timeline, budget, resources, and risk management strategies

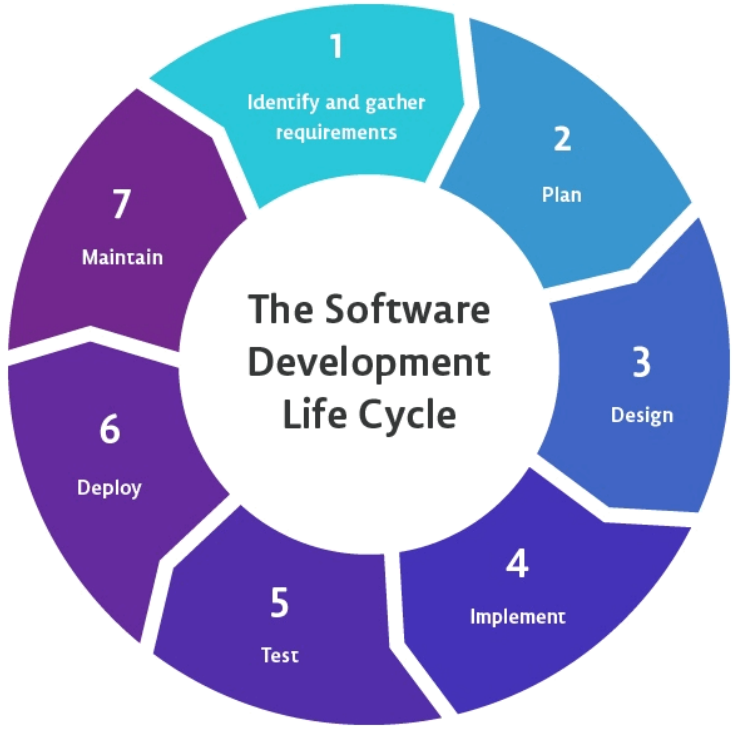
Key Activities in the Planning Phase

1. Define Project Timeline

- Break down the project into **milestones** and **tasks**.
- Set **deadlines** for each phase (Design, Development, Testing, Deployment).
- Use **project management tools** like Jira, Trello, or Asana to track progress.

2. Cost Estimation

- Estimate the **total budget** required, including:
  - Development costs (salaries, outsourcing).
  - Infrastructure costs (servers, cloud services, databases).
  - Software licenses, tools, and third-party integrations.



Planning Phase in SDLC

It focuses on creating a roadmap for the project by defining the timeline, budget, resources, and risk management strategies

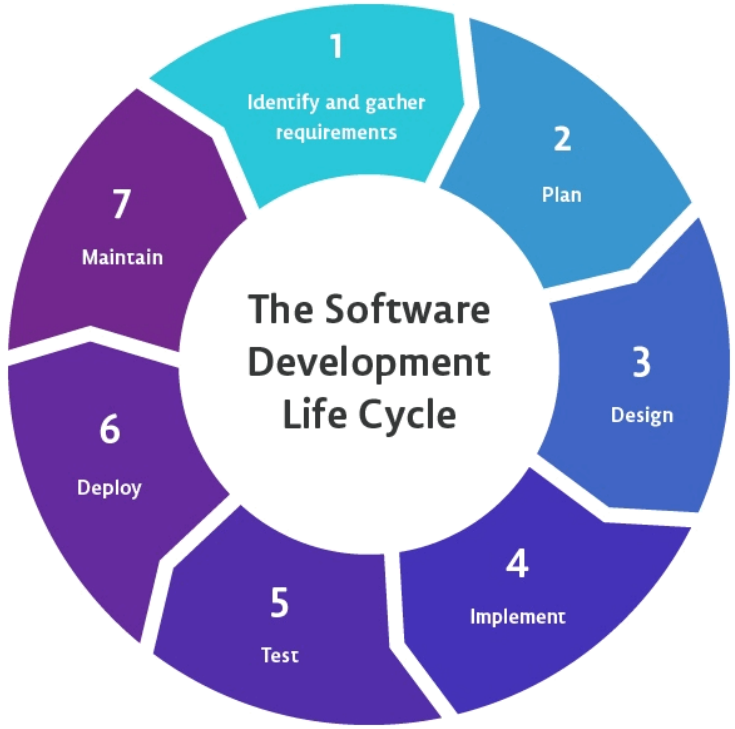
3. Use **cost estimation techniques** like:
    - **Analogous Estimation** (based on past projects).
    - **Parametric Estimation** (calculating cost per unit of work).
    - **Bottom-up Estimation** (breaking down tasks and estimating cost individually).

4. Resource Allocation

•Assign **team roles and responsibilities**, such as:
    - Developers, Testers, Designers, Project Managers.

•Determine **hardware and software resources** needed.

•Ensure **adequate manpower and expertise** for each phase.



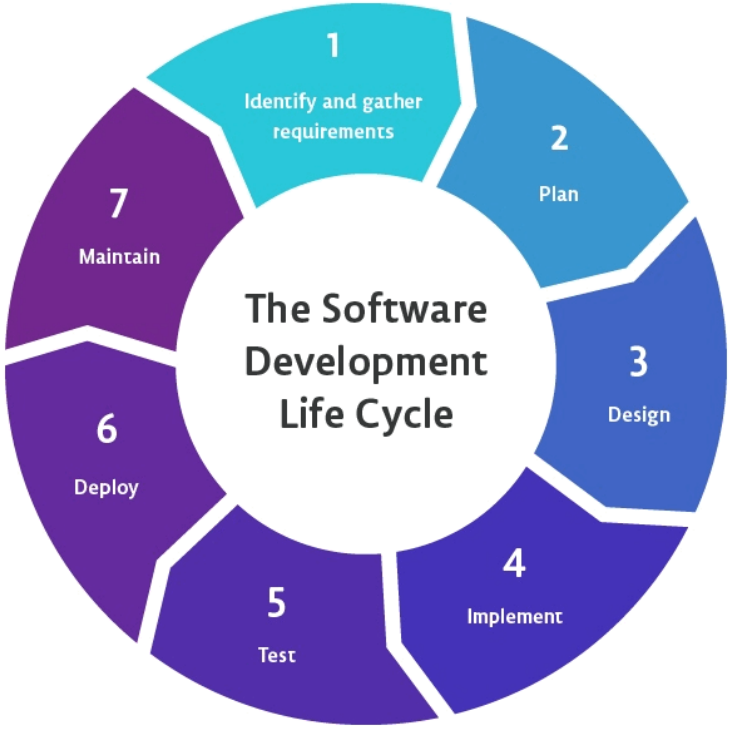
Planning Phase in SDLC

It focuses on creating a roadmap for the project by defining the timeline, budget, resources, and risk management strategies

5. Identify Risks and Mitigation Strategies

Common risks in software projects

- **Scope Creep:** Requirements keep changing, leading to delays.
  - 🔹 **Mitigation:** Define scope clearly and use change management processes.
- **Budget Overruns:** Costs exceed initial estimates.
  - 🔹 **Mitigation:** Regular cost monitoring and buffer allocation.
- **Technical Challenges:** Unexpected software bugs or system incompatibility.
  - 🔹 **Mitigation:** Conduct feasibility studies and prototyping.
- **Resource Shortage:** Lack of skilled personnel.
  - 🔹 **Mitigation:** Hire backup resources or train the existing team.
- **Security Risks:** Data breaches or vulnerabilities.
  - 🔹 **Mitigation:** Implement security best practices (encryption, authentication, audits).



Design Phase in SDLC

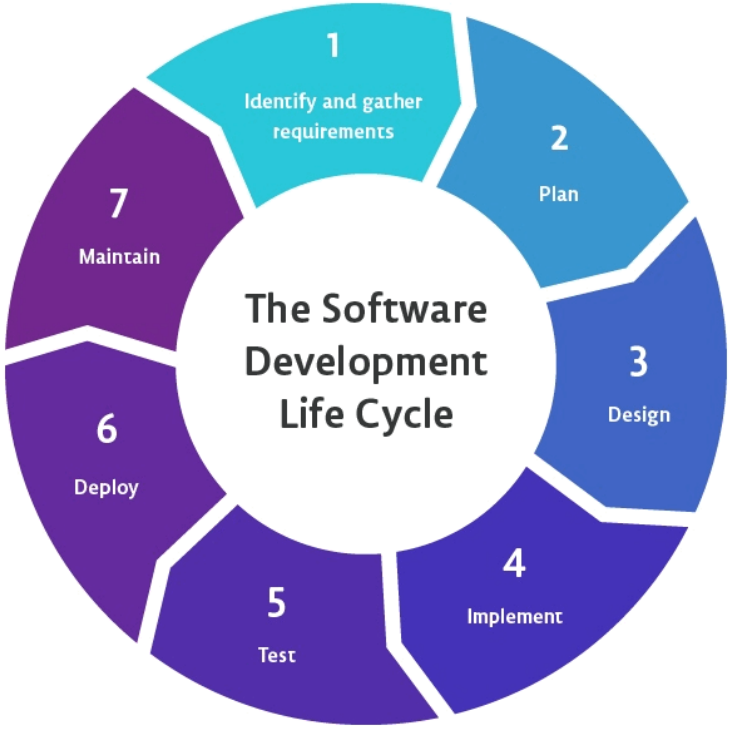
This phase transforms the requirements and planning into technical specifications and visual representations

Key Activities in the Design Phase

1. Create Architectural and UI/UX Design

Software Architecture Design

- Defines the **overall structure** of the software.
- Ensures **scalability, security, and performance**.
- Uses architectural patterns like:
  - **Monolithic Architecture** (all-in-one system).
  - **Microservices Architecture** (divided into independent services).
  - **Layered Architecture** (separates UI, business logic, and database layers).



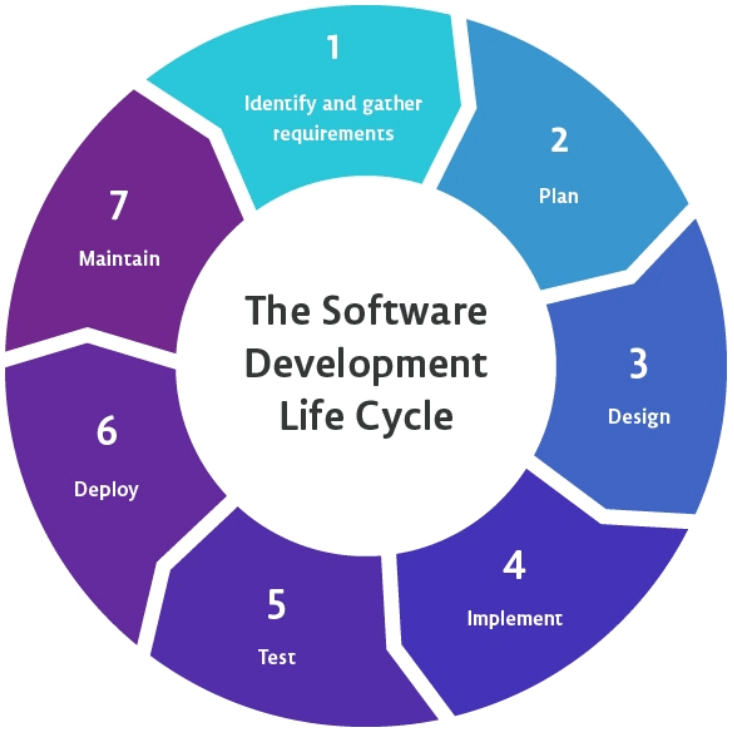
Example:

- A web application might follow the **MVC (Model-View-Controller)** pattern, separating data handling, business logic, and user interface.

Design Phase in SDLC

This phase transforms the requirements and planning into technical specifications and visual representations

Define Database Structures and System Components



Database Design

- Defines **how data is stored, accessed, and managed**.
- Includes:
  - **Entity-Relationship Diagram (ERD)** – Defines tables, relationships, and constraints.
  - **Database Schema** – Structure of tables, columns, and data types.

Types of Databases:

- **SQL databases** (MySQL, PostgreSQL) – Structured data with relationships.
- **NoSQL databases** (MongoDB, Firebase) – Flexible, scalable data storage.

Example:

- A social media app database may have tables for **Users, Posts, Comments, and Likes**, each with defined relationships.

Design Phase in SDLC

This is where developers start coding the software based on the design specifications

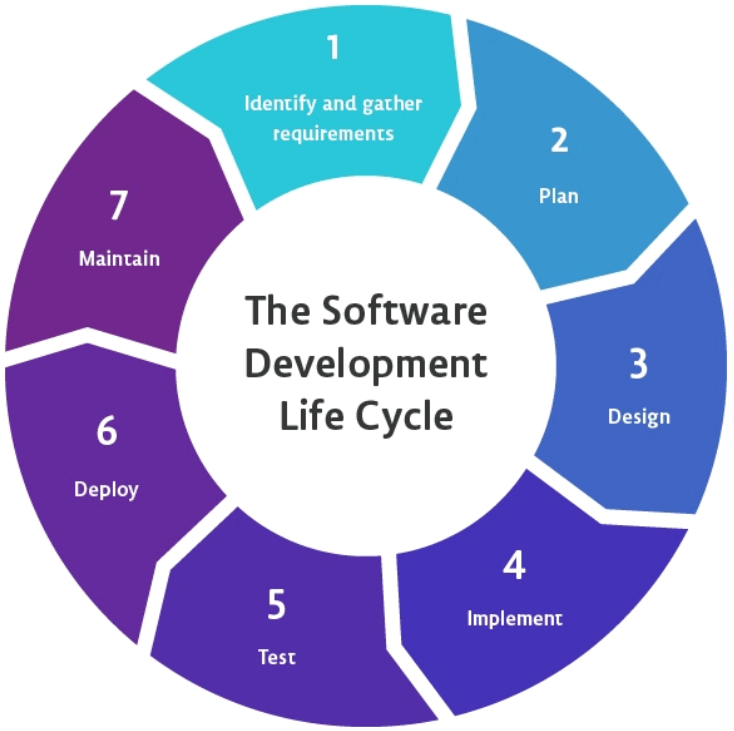
Key Activities in the Development Phase

Code the Software Based on Design Specifications

- Developers write code based on **system architecture, UI/UX designs, and database structures** defined in the design phase.
- Uses **programming languages** relevant to the project (e.g., JavaScript, Python, Java, C#).
- Frontend and backend teams work in parallel if needed.

Example:

- A **React frontend** fetching data from a **Node.js backend** with a **MySQL database**.



It is the most time-consuming phase and requires adherence to best practices, version control, and coding standards to ensure maintainability and scalability.

Design Phase in SDLC

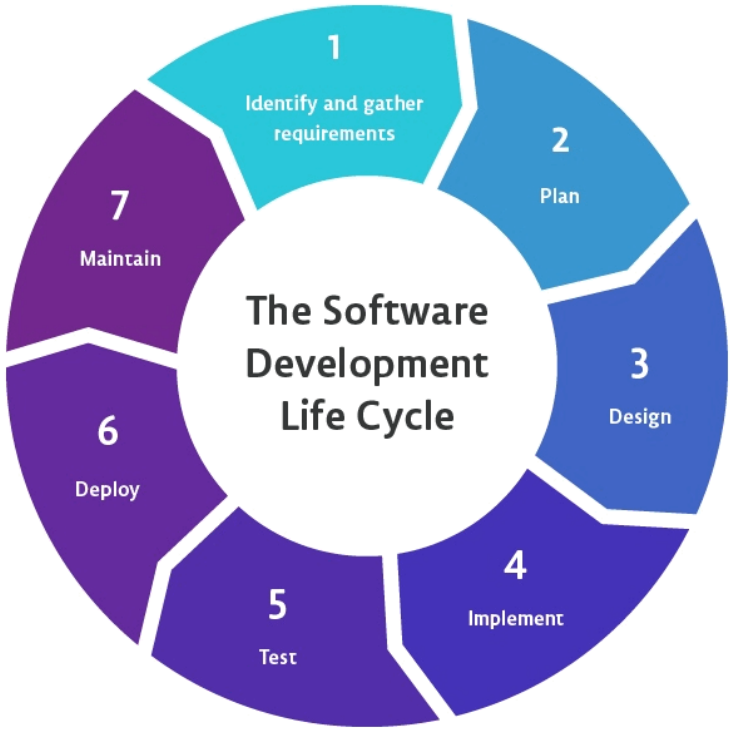
This is where developers start coding the software based on the design specifications

Use Version Control (Git, GitHub, GitLab, Bitbucket)

- Developers use **Git** to track code changes and collaborate efficiently.
- Follow **Git workflows** like:
  - **Feature Branch Workflow** – Each feature has its own branch.
  - **Gitflow Workflow** – Uses develop, main, feature, and hotfix branches.
  - **Trunk-Based Development** – All developers commit to main with small, frequent updates.

Example:

- Developers push code to GitHub and create a **Pull Request (PR)** for review before merging



It is the most time-consuming phase and requires adherence to best practices, version control, and coding standards to ensure maintainability and scalability.

Testing Phase in SDLC

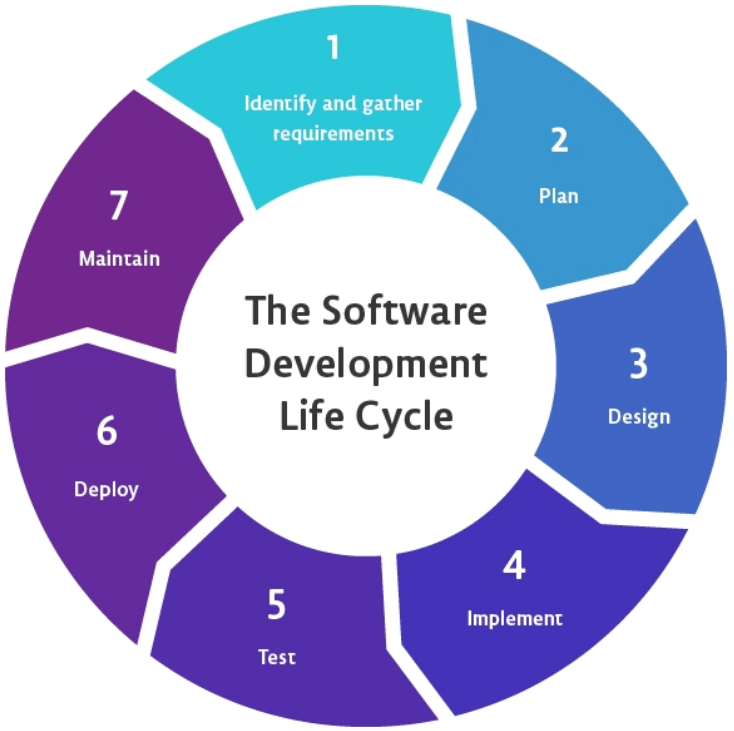
This phase ensures the software works as intended, is free of major bugs, and meets business requirements

[Key Activities in the Testing Phase]

1. Perform Different Types of Testing

Unit Testing (Testing Individual Components)

- Tests **smallest units** (functions, methods, or classes).
- Ensures each part of the software **works in isolation**.
- Automated using tools like **Jest (JavaScript), JUnit (Java), PyTest (Python)**.



It involves different levels of testing, such as **unit testing, integration testing, and system testing**, to verify that all components function correctly.

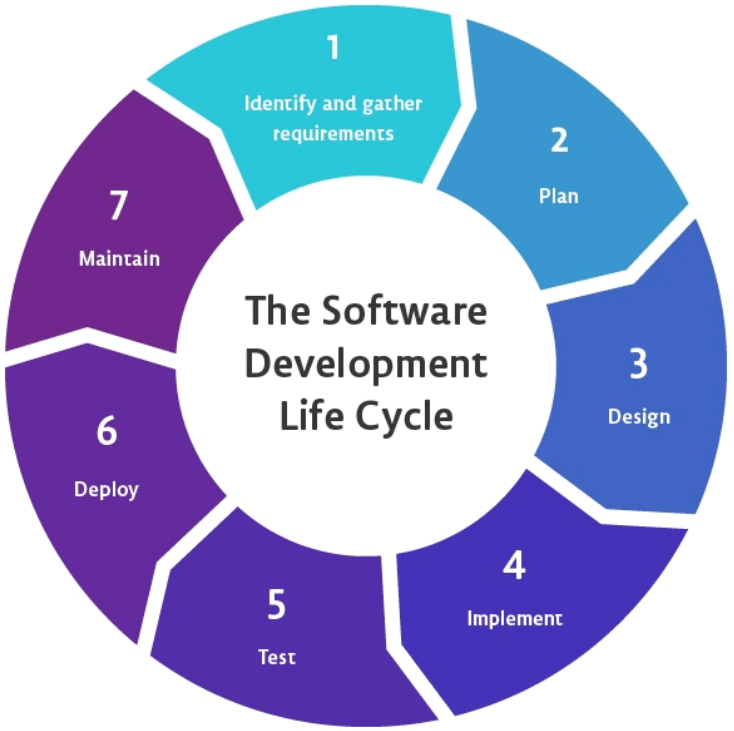
Deployment Phase in SDLC

This is where the software is released to production so users can access it and ensure a smooth, error-free deployment with minimal downtime.

Key Activities in the Deployment Phase

1. Release the Software to Production

- Software is **deployed to live servers** for real users.
- Can be done **manually** (one-time deployment) or **automated** (CI/CD).
- Ensure **backup of the previous version** in case rollback is needed.



Types of Deployment Approaches:

- **Blue-Green Deployment**: Keep two environments (one live, one standby).
- **Canary Deployment**: Release to a small percentage of users before full rollout.
- **Rolling Deployment**: Gradually update instances without downtime.

Example:

- A SaaS product updates the app version overnight to avoid user disruption.

Deployment Phase in SDLC

This is where the software is released to production so users can access it and ensure a smooth, error-free deployment with minimal downtime.

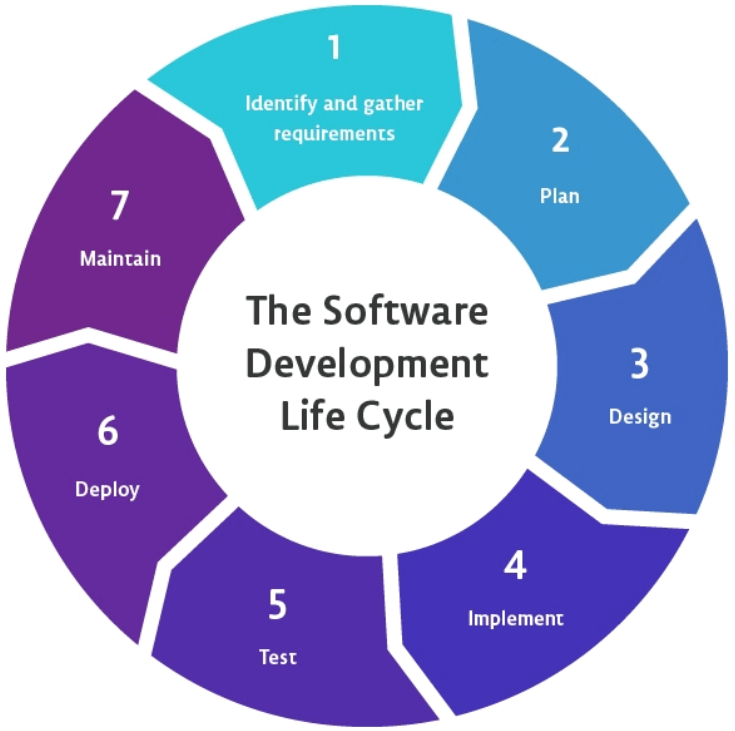
Implement CI/CD Pipelines for Automated Deployment

Continuous Integration (CI)

- Developers push code to GitHub/GitLab.
- Automated **build and test** process runs before merging new code.
- Tools: **GitHub Actions, Jenkins, CircleCI, TravisCI**.

Continuous Deployment (CD)

- Once CI tests pass, code is **automatically deployed** to staging or production.
- Uses **Docker, Kubernetes, AWS, Azure, Google Cloud** for cloud deployment.
- Ensures **frequent, fast, and reliable releases**.



Maintenance & Support Phase in SDLC

This phase ensures that the software remains functional, secure, and up-to-date after deployment

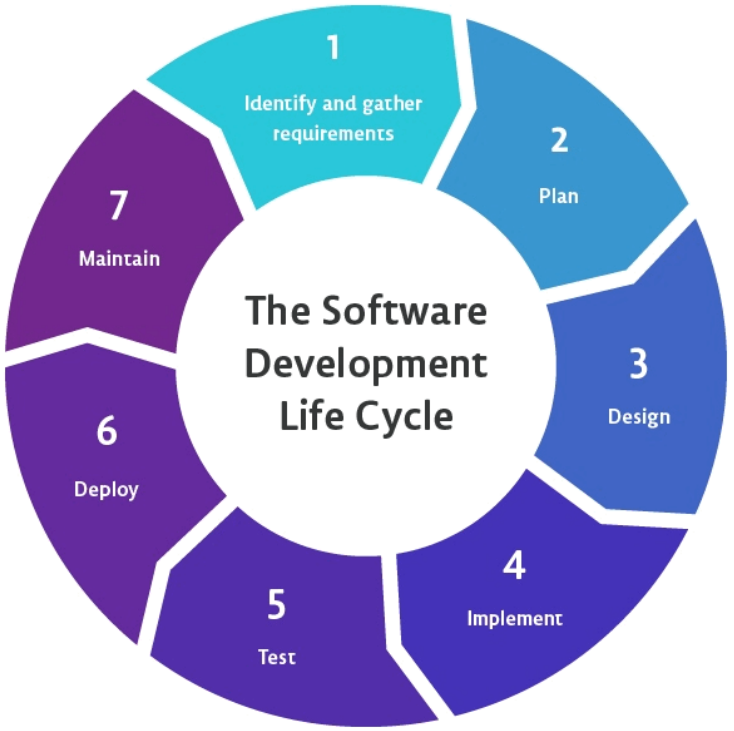
Key Activities in the Maintenance & Support Phase

1. Monitor Performance

- Continuously **track system performance** using monitoring tools.
- **Detect slow load times, high server usage, or crashes**.

Use tools like:

- **New Relic, Datadog, AWS CloudWatch** (for application performance).
- **Google Analytics, Hotjar** (for user behavior tracking).



Example:
An e-commerce website notices **slow checkout times** and optimizes the database queries.

It involves **monitoring performance, fixing bugs, and updating features** based on user feedback.

Maintenance & Support Phase in SDLC

This phase ensures that the software remains functional, secure, and up-to-date after deployment

Fix Bugs and Security Issues

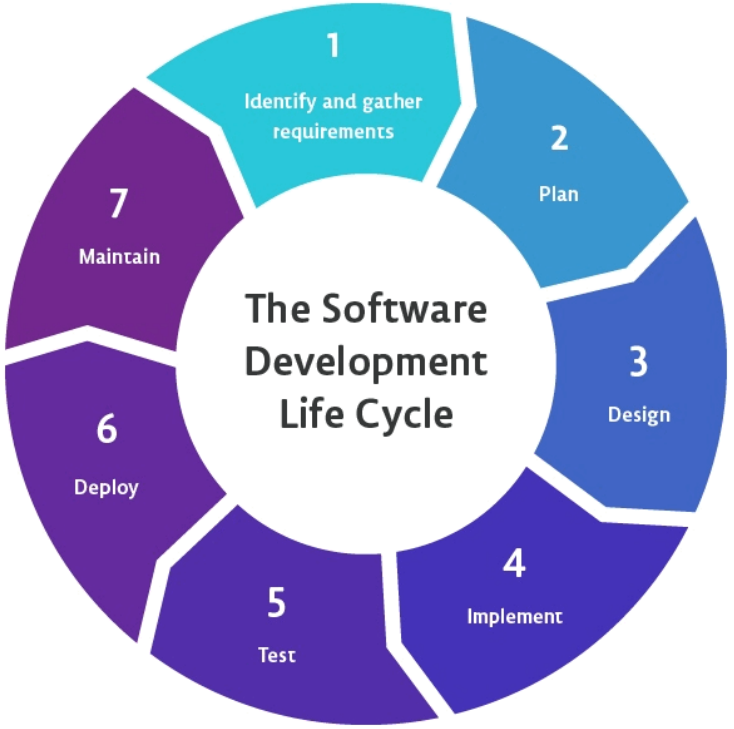
- Identify **unexpected issues** that arise after deployment.
- Regularly update **security patches** to prevent vulnerabilities.

Use **error tracking tools** like:

- **Sentry, LogRocket** (track real-time errors).
- **OWASP ZAP, Nessus** (for security scanning).

Example:

- A SaaS platform detects a **login issue** and deploys a hotfix via CI/CD.



It involves **monitoring performance, fixing bugs, and updating features** based on user feedback.

Maintenance & Support Phase in SDLC

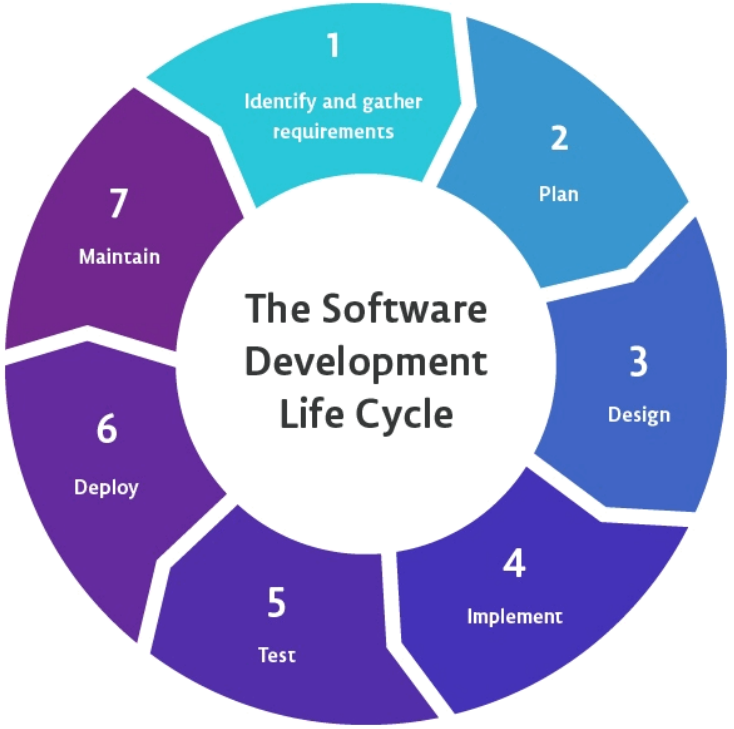
This phase ensures that the software remains functional, secure, and up-to-date after deployment

Update Features and Improve Functionality

- Gather **user feedback** to improve the software.
- Release **feature updates** to stay competitive.
- Use **Agile methodology** for continuous improvement.

Example:

- A mobile banking app adds **biometric authentication** based on user demand.



It involves **monitoring performance, fixing bugs, and updating features** based on user feedback.

Quiz Section

Quiz

Everyone student should click on submit button before time ends otherwise MCQs will not be submitted

[Guidelines of MCQs]

1. There are 30 MCQs
2. Time duration will be 15 minutes
3. This link will be share on 12:20pm (Pakistan time)
4. MCQs will start from 12:25pm (Pakistan time)
5. This is exact time and this will not change
6. Everyone student should click on submit button otherwise MCQs will not be submitted after time will finish
7. Every student should submit Github profile and LinkedIn post link for every class. It include in your performance

Assignment

Assignment should be submit before the next class

[Assignments Requirements]

1. Create a post of today's lecture and post on LinkedIn.
2. Make sure to tag @Plus W @Pak-Japan Centre and instructors LinkedIn profile
3. Upload your code of assignment and lecture on GitHub and share your GitHub profile in respective your region group WhatsApp group
4. If you have any query regarding assignment, please share on your region WhatsApp group.
5. Students who already done assignment, please support other students

Q&A Session

ありがとうございます。
Thank you.
شکریا

For the World with Diverse Individualities