Week-1 Learning Summary

SDLC (Software Development Life Cycle)

I learned what SDLC is and why it's important for building high-quality software in a structured way. I explored all the phases of SDLC, including:

- **Requirement Gathering** – Understanding what the customer needs (functional and non-functional requirements).
- **Design** – Creating system designs using **UML diagrams** (structural and behavioral).
- **Development** – Writing the actual code.
- **Testing** – Verifying the software works as expected.
- **Deployment** – Releasing the software to users.
- **Maintenance** – Updating and fixing issues post-release.

SDLC Models I Studied:

Model	Description
Waterfall	Linear and sequential.
V-Model	Testing is planned parallel to development stages.
Spiral	Risk-driven and iterative.
Iterative	Builds the software in parts and improves over time.
Agile	Fast, flexible, and customer-focused.

Each model has its **advantages**, **limitations**, and **use cases**.

Software Documentation:

- **SRS** – Software Requirements Specification
- **Design Documents** – Architecture and flow
- **Test Plans** – Testing strategy
- **User Manuals** – End-user guidance

Software Testing Methodology

I understood the **principles and objective** of software testing and how it supports SDLC. I studied the **Software Testing Life Cycle (STLC)**:

- Planning
- Design
- Execution
- Defect Reporting
- Closure

Testing Process:

- **Test Scenario** – High-level idea of what to test
- **Test Suite** – Group of related test cases
- **Test Plan** – Strategy and scope of testing
- **Test Cases** – Step-by-step test instructions

Levels of Testing:

- **Unit Testing** – Individual code units
- **Integration Testing** – Interactions between modules
- **System Testing** – Whole system verification
- **Acceptance Testing** – Meets business requirements

Testing Methodologies:

- **Functional Testing** – What the software does
- **Non-Functional Testing** – How the software performs

Testing Techniques:

- **Black-box Testing** – No knowledge of internal code
- **White-box Testing** – Full knowledge of code
- **Grey-box Testing** – Mix of both

Automation Tools:

- **Selenium** – Web automation
- **JUnit** – Unit testing for Java
- **Postman** – API testing
- **LoadRunner** – Performance testing

Agile Methodology

Agile is an **iterative and flexible** software development method. It focuses on:

- **Individuals and interactions**
- **Working software**
- **Customer collaboration**
- **Responding to change**

Agile vs Traditional:

- Agile is more **adaptive**, with **faster feedback**.
- Agile promotes **iterative delivery** in **sprints**.

Agile Frameworks I Studied:

Framework	**Focus**
Scrum	Roles, events, and artifacts
Kanban	Visual workflow, WIP limits
XP	Technical excellence
SAFe	Scales Agile for large organizations

Scrum Components:

- **Roles**:
    - Product Owner
    - Scrum Master
    - Development Team
- **Events**:
    - Sprint Planning
    - Daily Stand-ups
    - Sprint Review
    - Sprint Retrospective
- **Artifacts**:

- Product Backlog
- Sprint Backlog
- Burndown Chart

CI/CD (Continuous Integration / Continuous Deployment)

CI/CD automates the software delivery pipeline. It makes the development process **faster**, **safer**, and **more reliable**.

CI/CD Concepts:

- **CI (Continuous Integration)** – Merging and testing code automatically
- **CD (Continuous Delivery)** – Automated preparation for release
- **CD (Deployment)** – Fully automated deployment to production

---

CI/CD Pipeline Stages:

1. Code Commit
2. Build
3. Test
4. Deploy
5. Monitor

Code Quality Practices

Unit Testing:

Writing test cases for small components of code.

Code Coverage:

Measuring how much code is tested.

Static Code Analysis:

Using tools to automatically find bugs or bad practices.

- **SonarQube** – Detects bugs and code smells
- **Linting Tools** – ESLint, Pylint for formatting and quality