DAY 5 SUMMARY

1. Software Development Life Cycle (SDLC) & Agile Requirements
1.1 Defining Requirements

The session began with understanding how clearly defined requirements form the backbone of successful software projects. Requirements describe what the system should do and the constraints under which it must operate. Students learned the difference between functional requirements (features and behaviors) and non-functional requirements (performance, security, usability, etc.).

Well-defined requirements reduce misunderstandings between stakeholders and development teams. They act as a reference point throughout the project lifecycle, guiding design, development, testing, and validation. Emphasis was placed on writing requirements that are clear, testable, and unambiguous.

1.2 Requirements Elicitation

Students explored the process of requirements elicitation — the practice of gathering requirements from stakeholders and other sources. Since stakeholders often struggle to fully express their needs, elicitation involves structured techniques to uncover real expectations.

Common elicitation techniques discussed included interviews, workshops, brainstorming sessions, observation, surveys, and document analysis. The focus was on active listening, asking the right questions, and identifying hidden assumptions. Effective elicitation ensures that the final system solves the right problem, not just the stated problem.

1.3 Salience Model

The Salience Model was introduced as a way to identify and prioritize stakeholders based on their influence in a project. It helps teams understand whose opinions matter most when making decisions.

Stakeholders are evaluated based on three attributes: Power (ability to influence the project), Legitimacy (valid involvement in the project), and Urgency (need for immediate attention). Depending on how many of these attributes a stakeholder possesses, they fall into categories such as latent, expectant, or definitive stakeholders. This model helps teams manage communication and expectations more effectively.

1.4 Requirements Lifecycle

Students learned that requirements are not static; they evolve throughout the project. The Requirements Lifecycle describes how requirements are managed from their initial identification to their eventual retirement.

Key stages include documenting, analyzing, validating, prioritizing, tracing, and managing changes. Traceability was emphasized as a way to connect requirements with design, code, and test cases. Proper lifecycle management ensures that changes are controlled and that the product continues to align with business goals.

2. Agile Planning & Estimation
2.1 Sprint Planning

Sprint Planning was explained as a core Scrum event where the team decides what work will be completed in the upcoming sprint and how it will be done. The Product Owner presents prioritized backlog items, and the team selects work based on capacity and sprint goals.

Students learned that sprint planning results in a Sprint Goal and a Sprint Backlog. Collaboration is key — developers clarify requirements, break stories into tasks, and estimate effort. This session highlighted the importance of realistic commitments and shared ownership of sprint outcomes.

2.2 Story Points

Story Points were introduced as a relative estimation technique used in Agile to measure the effort, complexity, and uncertainty of a user story. Unlike time-based estimates, story points compare tasks to one another rather than assigning exact hours.

Teams often use sequences like Fibonacci numbers (1, 2, 3, 5, 8, 13...) to represent increasing complexity. Students learned that story points improve forecasting, encourage team discussion, and help calculate velocity over time. The focus is not on precision, but on building a shared understanding of effort and risk.