Reflection

Group B

Task 2

Reflection on Software Engineering Course

Helpful Topics from the Lectures

1. **Requirements Engineering**
    - **Specific Topic:** Requirements Engineering Definition and Importance (Lecture 2, Slides 19-20)
    - **Project Challenge:** Our project had issues with unclear and incomplete requirements, causing confusion and unmet expectations with stakeholders.
    - **Impact:** The lecture on requirements engineering taught us how to clearly define and communicate what a software system is supposed to do. This helped us create more detailed and accurate requirement documents, reducing confusion and making sure our development matched what stakeholders wanted. We improved how we talked with stakeholders and documented their needs, which helped us avoid creating unnecessary features and better understand the project's goals【6:0†source】【6:1†source】.
2. **Design Workflow**
    - **Specific Topic:** Transition from Requirements to Analysis and Design (Lecture 5, Slides 64-66)
    - **Project Challenge:** We had trouble turning high-level requirements into detailed design plans. Our team often found it hard to keep designs aligned with what was needed.

- **Impact:** The lecture on the design workflow gave us a clear process for connecting requirements to design. By formalizing requirements and considering how to implement them during the analysis phase, we created more consistent and practical design documents. This process made sure our designs matched the requirements and were doable, improving the quality and maintainability of our software【6:4†source】【6:5†source】【6:6†source】.

Missing Topics in the Lecture

1. **Risk Management**
    - **Why It Should Be Included:** Software projects are often affected by risks such as changing requirements, resource limitations, and technical challenges. Understanding risk management helps teams identify, assess, and mitigate potential risks early in the project.
    - **Impact on Development:** Including risk management in the curriculum would prepare students to anticipate and handle various risks, ensuring smoother project progression. In our project, having knowledge of risk management would have helped us identify potential issues earlier and develop strategies to mitigate them, reducing delays and improving project outcomes.
2. **Software Testing and Continuous Integration/Continuous Deployment (CI/CD)**
    - **Why It Should Be Included:** Detailed testing strategies (unit, integration, system testing) and CI/CD pipelines are crucial for maintaining software quality and speeding up delivery. Learning how to set up automated tests and deployments can greatly improve the development process.
    - **Impact on Development:** Including lectures on software testing and CI/CD would give students the skills to create automated testing and deployment systems. This knowledge is key for keeping code quality high and delivering features quickly and reliably. In our project, better testing and deployment practices could have prevented many bugs from reaching production and made our release process smoother.

Conclusion

The lectures on requirements engineering and design workflow provided valuable insights that helped us improve the clarity and consistency of our project requirements and designs. However, adding risk management and detailed software testing and CI/CD strategies to the curriculum would cover important areas that are currently missing, better preparing students for the challenges of real-world software development.