Current OnTrack Task:

In this week, I completed my two task 2.1p and 3.1P.

Plan for Next Week:

Next week, my primary focus will be on completing 3.5D Midterm Presentation.

Concerns or Questions:

I've begun drafting my thesis and came across a format online that I'm considering using. I'm wondering if we have any specific formatting guidelines we need to follow, such as font size, spacing, and headings. During my bachelor's, we followed a university-provided format for our final project report, so I want to ensure I'm aligned with any current requirements. Could you please review the template I've found to see if it's okay to use?

Also I have concern about research gap. I've identified three research gaps and would appreciate your advice on which one might be most suitable. Could you please review them and provide your feedback?

Research Gap

1. Limited Focus on Non-Functional Attributes in Reusability Metrics

Gap description

Current research often emphasizes functional aspects of reusability, such as code functionality and integration. These metrics, while important, largely ignore non-functional attributes and quality attributes that are critical for a comprehensive evaluation of reusability. Non-functional attributes include performance, security, usability, and user experience, while quality attributes cover adaptability, reliability, and scalability. The current focus on functional metrics alone provides an incomplete assessment of a component's effectiveness and practical applicability in real-world scenarios.

Implications:

Neglecting non-functional and quality attributes in reusability metrics can lead to several issues:

- Performance Issues: Reusable components may not meet required performance standards if non-functional aspects like efficiency are not considered.
- Security Risks: Components that fail to address security concerns can introduce vulnerabilities into the system.
- Usability Challenges: Without assessing usability, components might not align with user needs or expectations, affecting user satisfaction.
- Adaptability and Reliability Concerns: Components may lack the flexibility to adapt to different environments or changes, and their reliability might be compromised.
- Scalability Problems: The ability to handle increasing loads or demands may be overlooked, leading to scalability issues.

These shortcomings can result in increased maintenance efforts, higher costs, and diminished user satisfaction, impacting the overall success and quality of software projects.

Solution:

Addressing the limitations of current reusability metrics that emphasize functional aspects requires incorporating non-functional attributes such as performance, security, usability, adaptability, reliability, and scalability into the evaluation process. By developing comprehensive metrics and utilizing advanced tools like AI and machine learning, along with techniques like static and dynamic analysis, organizations can achieve a more holistic understanding of component effectiveness. This approach enhances the overall quality of reusable components by ensuring they meet performance standards, adhere to security practices, and align with user needs. By adopting frameworks such as multi-criteria decision-making and conducting thorough testing, organizations can address potential shortcomings, ultimately reducing maintenance efforts and costs while delivering more effective and applicable software solutions in real-world scenarios.

2. **Enhancing Code Reusability Metrics with AI for Automated Reuse Detection**

Gap Description:

While current reusability metrics focus on evaluating the quality of code components, there is a lack of advanced tools that automatically detect and suggest reusable code segments using AI. Existing methods primarily rely on manual identification of reusable code, which can be time-consuming and prone to error.

Solution:

Leverage AI and machine learning techniques to develop automated systems that can analyze codebases and identify opportunities for code reuse. By integrating AI-driven analysis with reusability metrics, we can create tools that not only assess the quality of code components but also recommend how they can be reused effectively. This approach could include features like pattern recognition for code duplication, similarity analysis for component reuse, and predictive models to suggest reusable components based on historical data.

3. **Customizing AI Tools for Specific Reusability Metrics**

Scenario:

Imagine a large software development company that uses an AI tool to analyze their codebase for quality metrics. The AI tool currently provides general insights about code complexity and performance but does not specifically evaluate how well the code adheres to important reusability principles such as:

- **Coupling:** How dependent different modules or classes are on each other.
- **Cohesion:** How closely related and focused the responsibilities of a single module or class are.
- **Code Duplication:** Repeated code segments across the codebase.

Gap Description:

In this scenario, the AI tool's general analysis misses critical aspects of code reusability. For example, if the tool cannot measure coupling, it might not identify areas where tightly coupled modules are causing difficulties in making changes or updates. Similarly, if it doesn't assess code duplication, the tool may overlook redundant code that could be refactored for better maintainability.

Solution:

To solve this problem, we can focus on customizing the AI tool to address these specific reusability metrics. For instance:

1. **Customizing for Coupling:** Extend the AI tool to analyze and report on the level of dependency between different modules. This customization could help identify tightly coupled components that may be challenging to modify or extend.
2. **Enhancing for Cohesion:** Adapt the tool to evaluate the cohesion of individual classes or modules, ensuring that each component has a single, well-defined responsibility. This can improve the maintainability and clarity of the code.
3. **Detecting Code Duplication:** Develop features within the AI tool to detect and report on duplicate code segments, making it easier to refactor and reduce redundancy.

Advantage:

By focusing on specific metrics like coupling and cohesion, the code can be better organized into reusable components. This enhances the ability to reuse code across different projects or parts of the same project.