AI Implementation Plan for the Project

Objective: Enhance the productivity of the software development team by leveraging AI as a support tool to improve code quality and accelerate task completion.

Core Idea: AI should be integrated into workflows as an advanced tool that assists developers but does not replace them. For successful implementation, the focus should be on improving current processes, utilizing the best tools available on the market, and creating a systematic approach to training and refactoring.

AI Implementation Plan for the Project

Phase 0: Team and Infrastructure Preparation

1. **Assessment of the Current State:**
    - **Developer Skills Audit:**
      Evaluate the team's knowledge of key frameworks, best development practices, and AI-related skills. Identify strengths and areas for improvement.
    - **Codebase Analysis:**
      Assess the codebase's compliance with modern standards, identify areas with technical debt, and outline tasks for refactoring.
    - **Identifying Gaps:**
      Create a skills map for the team to determine which knowledge and skills need to be developed for effective AI utilization.
2. **Training in Prompt Engineering:**
    - Identify the need for and organize additional training on the basics of working with AI, including creating effective prompts.
    - Conduct individual practical sessions where developers learn to task AI, validate results, and refine queries.
3. **Codebase Updates:**
    - Bring the codebase up to current standards to enable AI to work with it more effectively.
    - Eliminate technical debt and implement unified coding standards.

Phase 1: AI Integration into Workflows

1. **Using AI for Refactoring and Addressing Technical Debt:**
    - Leverage AI to automate refactoring, including optimizing algorithms, improving code readability, and aligning it with modern standards.
    - Break refactoring tasks into stages to minimize risks and ensure gradual improvement.
2. **Integrating AI into Development Tools:**
    - Ensure all developers are using GitHub Copilot.

- Configure AI to work with current frameworks and libraries to maximize its efficiency.

3. **Focusing on Improving Current Processes:**

   - Instead of creating new AI solutions, focus on applying proven tools on the market that have already demonstrated their effectiveness.
   - Gradually integrate AI into all stages of the software development lifecycle, including updating the overall development process.

Phase 2: Building a Knowledge-Sharing Culture

1. **Creating a Knowledge Base:**

   - Establish an internal platform for sharing prompts, success stories, and challenging scenarios.
   - Include a section with examples where AI failed to provide relevant answers and suggest alternative approaches.

2. **Regular Meetings and Discussions:**

   - Hold regular meetups and workshops to exchange experiences in using AI.
   - Review new AI capabilities and best practices.

3. **Mentorship and Support:**

   - Assign individuals responsible for training and supporting colleagues in working with AI.
   - Implement a mentorship system for newcomers.

Phase 3: Continuous Improvement

1. **Monitoring Effectiveness:**

   - Regularly collect feedback from the team on AI performance.
   - Evaluate AI's impact on productivity, code quality, and task completion speed.

2. **Skills Updates:**

   - Conduct periodic training on new AI capabilities and current development practices.
   - Keep the knowledge base up to date.

3. **Experimentation and Research:**

   - Continuously test new approaches to using AI.
   - Implement improvements based on experimental results.

Key Focus Areas

1. **Improving Current Processes:**

   - Streamlining the codebase, addressing technical debt, and refactoring with AI assistance.
   - Automating routine tasks and enhancing development quality.

2. **Leveraging the Best Tools:**
    - Focus on applying proven AI solutions, such as GitHub Copilot, rather than developing proprietary tools with a high risk of low adoption.
3. **Team Training and Development:**
    - Enhance the team's knowledge of key frameworks, best development practices, and AI-related skills.
    - Develop prompt engineering skills among developers.
    - Foster a culture of knowledge sharing and continuous learning.

Expected Outcomes

1. **Short-Term:**
    - Increased developer knowledge of key frameworks, best development practices, and AI-related skills.
    - Improved code quality through refactoring and addressing technical debt.
    - Accelerated task completion through effective AI utilization.
2. **Mid-Term:**
    - Reduced errors in the code and improved test coverage.
    - Faster task execution thanks to efficient AI usage.
3. **Long-Term:**
    - Establishing a culture of knowledge sharing and continuous learning.
    - Increased overall team productivity and project competitiveness.
    - Enhanced development quality through AI utilization.

Conclusion

AI has already been integrated into the team's workflows through tools like GitHub Copilot, and general training sessions have been conducted. The next step is to systematically improve current processes, including refactoring, addressing technical debt, and developing developers' skills. The primary focus is on leveraging the best tools available on the market to minimize risks and maximize efficiency. This will lay the foundation for sustainable growth and improved development quality in the long term.

The adoption of AI not only provides an opportunity to improve development quality but also enables the team to become more client-focused — allowing us to go the "extra mile" for the client more easily and quickly at any stage of the work. It is also important to identify employees who are skeptical about AI adoption and organize individual and group sessions with them to reduce resistance and ensure successful integration of new tools.