Turiba University

(*Kartik Jyoti*)

(Friendly Coding)

QUALIFICATION PAPER/ACADEMIC BACHELOR THESIS/PROFESSIONAL BACHELOR THESIS/MASTER THESIS/

Study programme (Bachelor's in Computer System)

Author: ***(Kartik Jyoti*)**

***Thesis* Advisor:** **(Jānis Pēkša, Dean of IT dept.)**

Riga, 2025

1. Introduction: The Human Side of Coding

Friendly coding means writing code that is easy for people to read and work with. Even though code must run correctly for computers, friendly coding focuses on making sure humans—like other developers, future team members, or even your future self—can easily understand it.

This way of coding values clear, simple, and well-organized code. It recognizes that building software is a team effort, and a project succeeds in the long run when everyone can understand, update, and fix the code without difficulty.
This report will look at the main ideas behind friendly coding and how it improves software quality, teamwork, and the long-term health of a project.

Hu
<SOC>
Human Side of Code

2. Core Principles of Friendly Coding

Friendly coding is not just one rule—it's a collection of good habits. These habits fall into four main groups: readability, maintainability, collaboration, and learning.

2.1. Readability and Clarity
The most important part of friendly coding is making your code easy to read.
Clear Names: Use names that describe what something does. For example, calculateTotal() is clearer than calc(), and customerName is better than c.

Consistent Style: Follow a style guide so your code always looks clean and organized. This includes using the same indentation, spacing, and formatting rules.

Helpful Comments: Good code explains itself, but comments help explain why something is done, especially if the logic is complicated. Docstrings and README files also help others understand how the project works.

Keep It Simple: Avoid making code more complicated than it needs to be. Use small functions and classes that each do one clear job.

2.2. Maintainability and Future-Proofing
Friendly code is written with the future in mind. It should be easy for someone else to change or fix later.

Modular Design: Split the code into smaller pieces or modules, so developers only need to understand the part they are working on.

Good Error Messages: Use clear error messages that help developers understand what went wrong.

Testable Code: Write code that can be tested easily. This makes your code more reliable and less likely to break when changes are made.

2.3. Collaboration and Culture
Friendly coding isn't just about the code—it's also about how people work together.
Code Reviews: Letting teammates review code leads to better solutions and helps everyone learn.

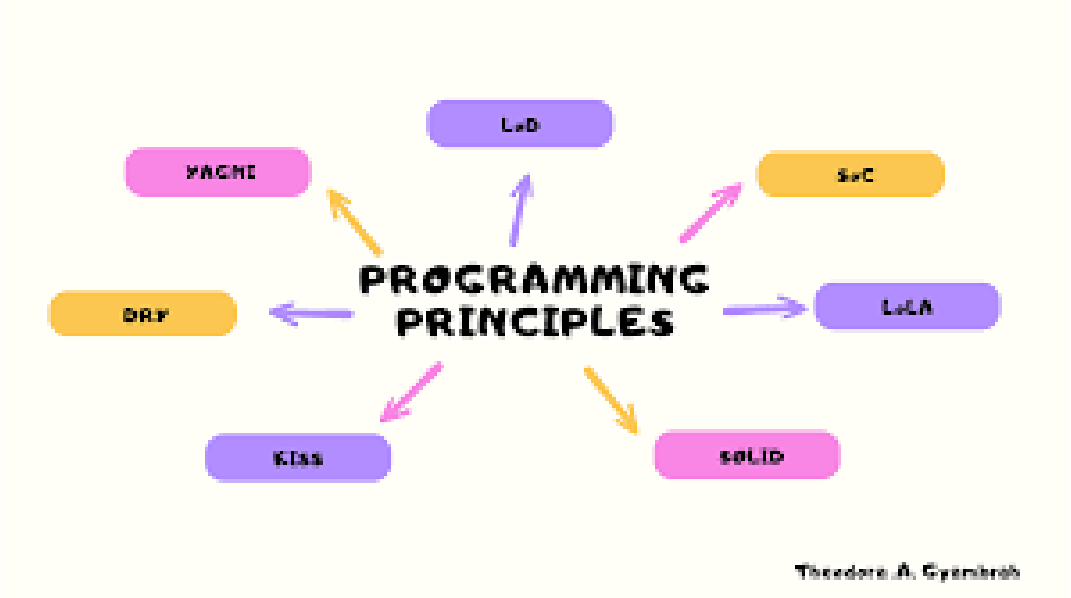
Pair Programming: Two people working together can write clearer code and catch mistakes early.

Open Communication: A friendly environment makes it easier for developers to ask questions and share ideas without fear of being judged.

2.4. Learning and Accessibility
Friendly coding also helps new developers learn more easily.

Beginner-Friendly Languages: Some languages, like Python, are easier for beginners.
Learning Resources: Websites like FreeCodeCamp and CodeNewbie offer beginner-friendly lessons and communities.

AI Tools: Modern tools with AI help make coding more approachable and easier to understand.



3. Benefits of Friendly Coding

3.1. Better Productivity
Code that is easy to read and maintain saves time. Developers spend less time trying to understand old code and more time building new features.

3.2. Stronger Teamwork
When code is easy to understand, working together becomes easier. Code reviews become more positive, and team members feel less stressed or confused.

3.3. Long-Term Sustainability
Friendly code lasts longer. It's easier to fix, update, and improve. This is especially important for big or long-term projects.

3.4. Higher Quality Code
Practices like clear names, modular design, and good error handling naturally lead to better-quality software with fewer bugs.

3.5. Better Developer Skills
Friendly coding encourages learning. Through reviews and teamwork, developers improve their skills and learn from each other.

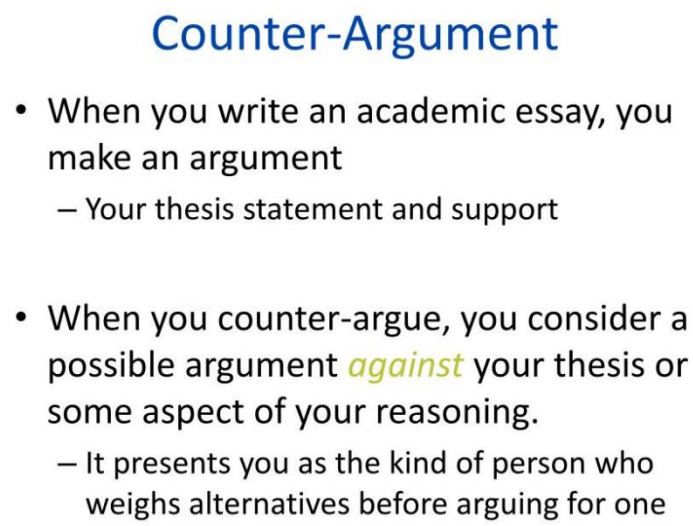
4. Challenges and Counterarguments

4.1. “Working Code” vs. “Good Code”
Sometimes teams rush to finish features quickly. This can lead to messy code that causes problems later. Balancing speed and quality is always a challenge.

4.2. It Seems Time-Consuming
Some people think writing clean code or documentation takes too much time. But in reality, it saves time in the long run because the code is easier to maintain.

4.3. “Friendly” Is Subjective
Different developers have different opinions about what friendly code looks like. A style guide can help keep things consistent.

4.4. Focus on Tools Over People
Sometimes teams get excited about new tools or frameworks and forget about long-term maintainability. Friendly coding reminds us to choose tools that are easy for the whole team to use.

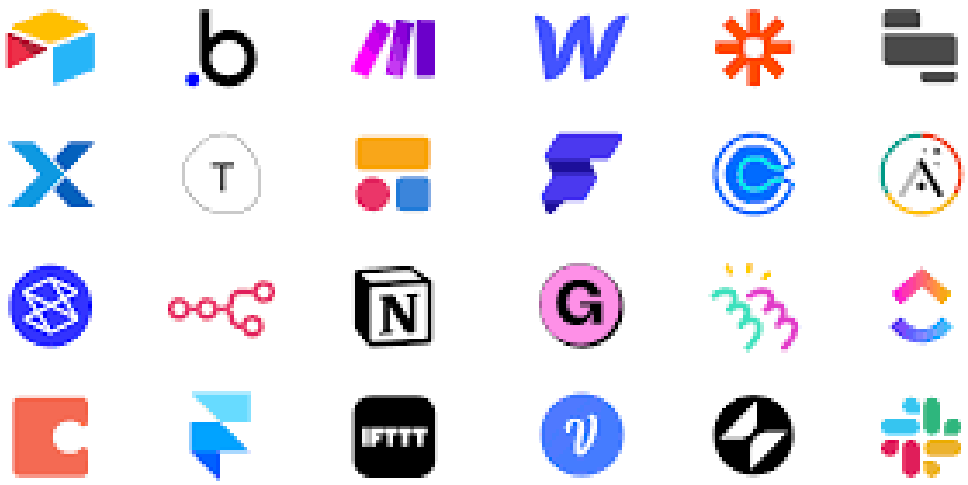


5. Tools and Communities That Support Friendly Coding

Tools and Communication for Friendly Coding
Friendly coding means making software together in a way that feels easy, clear, and respectful. The tools we use and how we talk to each other can make teamwork smooth—or confusing. When we use the right tools and communicate well, coding becomes a much nicer experience.

Version control tools like GitHub and GitLab help people work on the same code without problems. They let teammates share updates, track changes, and review each other's work calmly. A pull request can turn into a great discussion when feedback is kind and helpful.
Communication tools—such as Slack, Teams, or group chats—are also very useful. They make it simple to ask questions, share quick updates, or help someone out. When messages stay friendly and patient, people feel comfortable speaking up, which improves teamwork.
Good documentation helps too. Clear and simple notes or guides make it easier for new developers to understand the project. This saves time and helps everyone feel welcome.
Project management tools like Trello or Jira keep tasks organized. When everyone can see what needs doing and who's doing it, teamwork runs smoothly and misunderstandings are fewer.

In the end, friendly coding isn't just about the tools—it's about being kind. When people listen, show respect, and use tools that keep things clear, coding becomes something everyone can enjoy and learn from together.



6. Conclusion: The Future of Friendly Coding

Friendly coding focuses on writing code for humans, not just machines. It values readability, maintainability, and teamwork. Even though it can take extra effort, the long-term benefits—like better productivity, happier teams, and healthier projects—make it worth it.
As software becomes more complex and teams become more spread out, friendly coding will become even more important. It isn't just a best practice—it's an investment in better software and a better experience for all developers.

If you want, I can summarize it even more, turn it into bullet points, or make it kid-friendly.