2. Software Development Life Cycle

2.1 Process Model

Extreme Programming (XP) is the most widely used software development model. This model is an Agile methodology consisting of effective development practices to achieve client satisfaction. It is a combination of iterative and incremental process models. To apply XP, there are 12 practices to follow at an extreme level. Those are:

- Practice 1: The Planning
- Practice 2: Small Releases
- Practice 3: System Metaphor
- Practice 4: Simple Design
- Practice 5: Continuous Testing
- Practice 6: Refactoring
- Practice 7: Pair Programming
- Practice 8: Collective Code Ownership
- Practice 9: Continuous Integration and Daily Build
- Practice 10: 40-Hour Work Week
- Practice 11: On-Site Customer
- Practice 12: Coding Standards

In an XP project, work happens in short iterations that can last from one to three weeks. Before each iteration, a meeting happens where developers, managers, and the customer decide how much work can be done during that iteration. The customer prioritizes the work that needs to be done, and the team members commit to the amount of work they estimate they can deliver during the iteration.

We are choosing this model because:

1. The client and Developer always communicate with each other to develop the product. So, Product failure risk going to be low.
2. It is also cost efficiency.
3. Teams implement exactly what was asked—and nothing more—striving for a simple design and clean code.
4. Using this model, we can get early and frequent feedback from the automated unit tests, feedback from team members, and feedback from the client itself.
5. Using this model, we can use pair programming, where two developer work side by side.
6. This model highly emphasizes teamwork.
7. Because of its highly emphasizes on teamwork, we can share our ideas.
8. This model is aim to be programmer-friendly. It respects the developer's balance of work and life.
9. Using this model, we can change requirements when it would need.
10. Using this model, we can get a good environment between the developer and the customer.

11. In this model, developers get higher flexibility. At crunch time, up to 1 week of overtime is allowed.
12. Using this model, we can adopt collective ownership of code.
13. Gain the support, trust, and motivation of the people involved.
14. Frequent delivery of working software.
15. Get regular reflections on how to become more effective.
16. It is more versatile and less prone to rigidity, thus allowing programmers to respond to shifting demands as needed.
17. Easy to understand the customer and get proper knowledge about the project.