Agile Professional Final Project

Smart Context-Aware Calculator Development

GitHub Repository: https://github.com/Kynmmarshall/Agile-professional-group-1

Course: The Agile Professional (SE 3122)
Instructor: Dr. KEYAMPI WATIO Martial
Date: 12/01/2026
Duration: 4 Weeks
Team Size: 9 Members
Total Marks: 70

Agile Framework Implementation

Scrum	Kanban	Hybrid (Scrumban)
Time-boxed sprints Roles & ceremonies	Visual workflow Continuous delivery	Best of both methods

Faculty of Information and Communication Technologies
Fall 2025

Team Members and Roles

No.	Name	Matricule	Role
1	Kamdeu Yamdjeuson Neil Marshall	ICTU20241386	Scrum Master
2	Tembong Jennette Ndip	ICTU20241752	Product Owner
3	Karel Jess Bissakonou	ICTU20241743	Developer 1 (Core Logic)
4	Ngong Judd Ngum JR	ICTU20241253	Developer 2 (GUI & Layout)
5	Oleme Elobo Ronald Jean De Dieu	ICTU20241912	Developer 3 (Input Handling)
6	Feutseu Kenmogne Junior Erwan	ICTU20234174	Developer 4 (Error Handling)
7	Asongwe Tony Khan	ICTU20241379	Developer 5 (UI/UX Design)
8	Bidias Killian	ICTU20234020	Developer 6 (Tester/QA)
9	Kosho Angelo	ICTU20234127	Developer 7 (smart suggestions & context switching)

Table 1: Complete Team Members List with Roles and Matricules

Executive Summary

This report documents the Agile development process for a Smart Context-Aware Calculator project, completed as part of SE 3122: The Agile Professional. Over four weeks, a 9-member team applied hybrid Agile methodologies (Scrum + Kanban) to develop an advanced calculator application with context-sensitive modes and intelligent features. The project demonstrates practical application of Agile principles, including iterative development, continuous feedback, and adaptive planning. Through simulated sprints, daily stand-ups, and retrospectives, the team successfully delivered a functional application while experiencing firsthand the benefits and challenges of Agile software development.

Contents

Chapter 1

Agile Framework Selection and Justification

1.1 Selected Framework: Hybrid Scrum-Kanban Approach

The team selected a hybrid Agile framework combining Scrum and Kanban methodologies. This decision was based on the complementary strengths of both approaches and their suitability for the project's scope and team dynamics.

1.2 Justification for Smart Calculator Project Selection

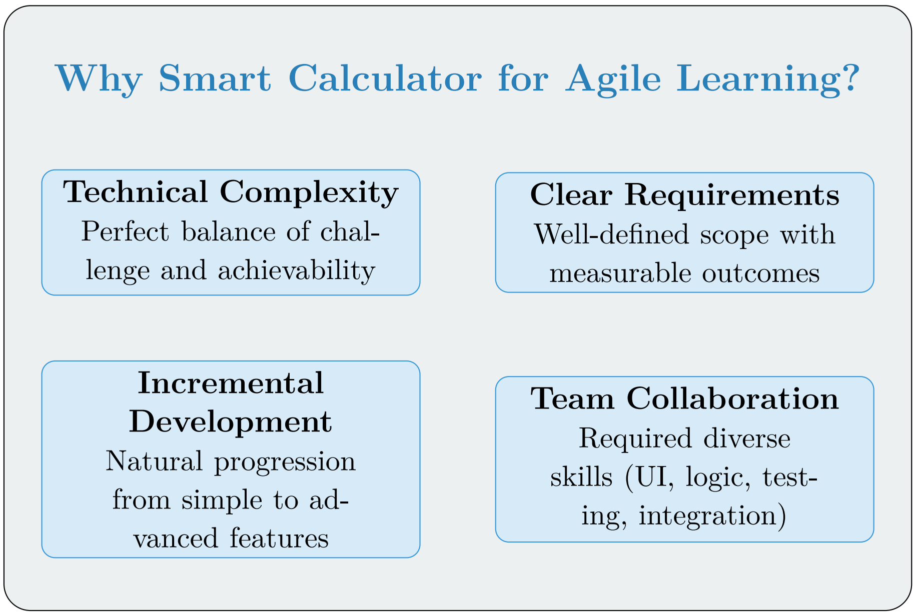

Why Smart Calculator for Agile Learning?
Technical Complexity
Perfect balance of challenge and achievability
Clear Requirements
Well-defined scope with measurable outcomes
Incremental Development
Natural progression from simple to advanced features
Team Collaboration
Required diverse skills (UI, logic, testing, integration)

Figure 1.1: Project Selection Justification

1.2.1 Educational Value for Agile Learning

The Smart Calculator project was selected for its ideal characteristics as an Agile learning vehicle:

Agile Learning Objective	How Calculator Project Supports It
Iterative Development	Natural progression from basic arithmetic to smart features
User Story Creation	Clear user personas (student, shopper, cook, budgeter)
Sprint Planning	Measurable features with clear acceptance criteria
Continuous Integration	Modular architecture supports incremental delivery
Team Collaboration	Requires UI design, logic implementation, and testing skills
Retrospective Analysis	Clear metrics for velocity and quality assessment

Table 1.1: Alignment with Agile Learning Objectives

1.2.2 Technical Suitability for Agile Practices

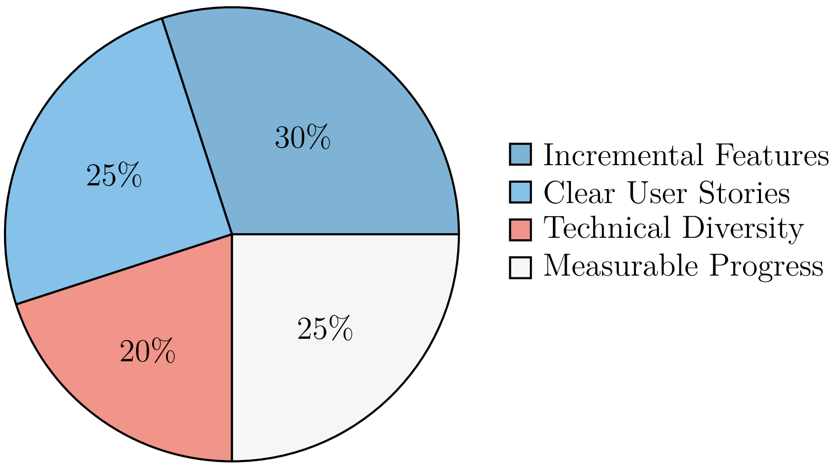


Figure 1.2: Technical Characteristics Supporting Agile

1.2.3 Specific Agile Demonstrations Enabled

Scrum Demonstration Points

- **Sprint Planning**: Each context mode (Standard, Homework, Shopping, etc.) became a natural sprint goal
- **Daily Stand-ups**: Concrete progress on features like "tip calculator" or "trigonometry functions"
- **Sprint Reviews**: Working demonstrations of new context modes
- **Retrospectives**: Clear metrics on feature completion and code quality

Kanban Demonstration Points

- **Visual Workflow**: GitHub Projects board with features moving through stages

- **WIP Limits**: Managing parallel development of multiple calculator functions
- **Flow Management**: Balancing UI work with backend logic implementation
- **Continuous Delivery**: Regular integration of completed features

1.2.4 Risk Management Considerations

The project's scope provided ideal risk management learning:

- **Controllable Complexity**: Challenging enough to require Agile but not overwhelming
- **Clear Success Criteria**: Working calculator with defined features
- **Learning Opportunities**: Technical challenges appropriate for student teams
- **Time-Bound Delivery**: 5-week timeline matched academic constraints

1.2.5 Conclusion on Project Selection

The Smart Calculator project offered the perfect balance of technical challenge, clear requirements, and Agile demonstration potential. Its modular nature allowed for:

1. Clear sprint goals with measurable outcomes
2. Diverse technical challenges requiring team collaboration
3. Natural progression from simple to complex features
4. Tangible results for stakeholder demonstrations
5. Comprehensive Agile practice across all ceremonies

Chapter 2

Simulated Agile Project: Sprint Planning & Execution

2.1 Project Overview: Smart Context-Aware Calculator

The team developed a graphical calculator application with five context modes:

- **Standard Mode**: Basic arithmetic operations
- **Homework Mode**: Mathematical functions (trigonometry, square roots, exponents)
- **Shopping Mode**: Financial calculations (tips, taxes, bill splitting)
- **Budgeting Mode**: Percentage calculations and financial planning
- **Cooking Mode**: Recipe adjustments and unit conversions

2.2 Sprint Planning

2.2.1 Product Backlog Creation

The product backlog was created collaboratively with the Product Owner prioritizing features based on user value:

ID	User Story	Acceptance Criteria	Priority
US-01	As a user, I want basic arithmetic operations	+, -, x, ÷ work correctly with proper display	Must
US-02	As a student, I want trigonometric functions	sin, cos, tan calculate correctly for degree inputs	Should
US-03	As a shopper, I want to calculate tips	15%, 18%, 20% options with total calculation	Should
US-04	As a budgeter, I want percentage calculations	Increase/decrease by percentage functions	Could
US-05	As a cook, I want recipe scaling	1/2, 1/3, 1/4, 2×, 3× scaling functions	Could
US-06	As a user, I want context switching	Click buttons to change calculator mode	Must
US-07	As a user, I want smart suggestions	Calculator suggests relevant operations	Would

Table 2.1: Sample Product Backlog Items

2.2.2 Sprint Goals

Sprint	Goal	Duration
Sprint 1	Deliver basic calculator with Standard mode	1 week
Sprint 2	Implement Homework, Shopping, Budgeting and Cooking modes	1 week
Sprint 3	Implement smart features and input panels	1 week
Sprint 4	Testing, bug fixes, and documentation	1 week

Table 2.2: Sprint Goals and Durations

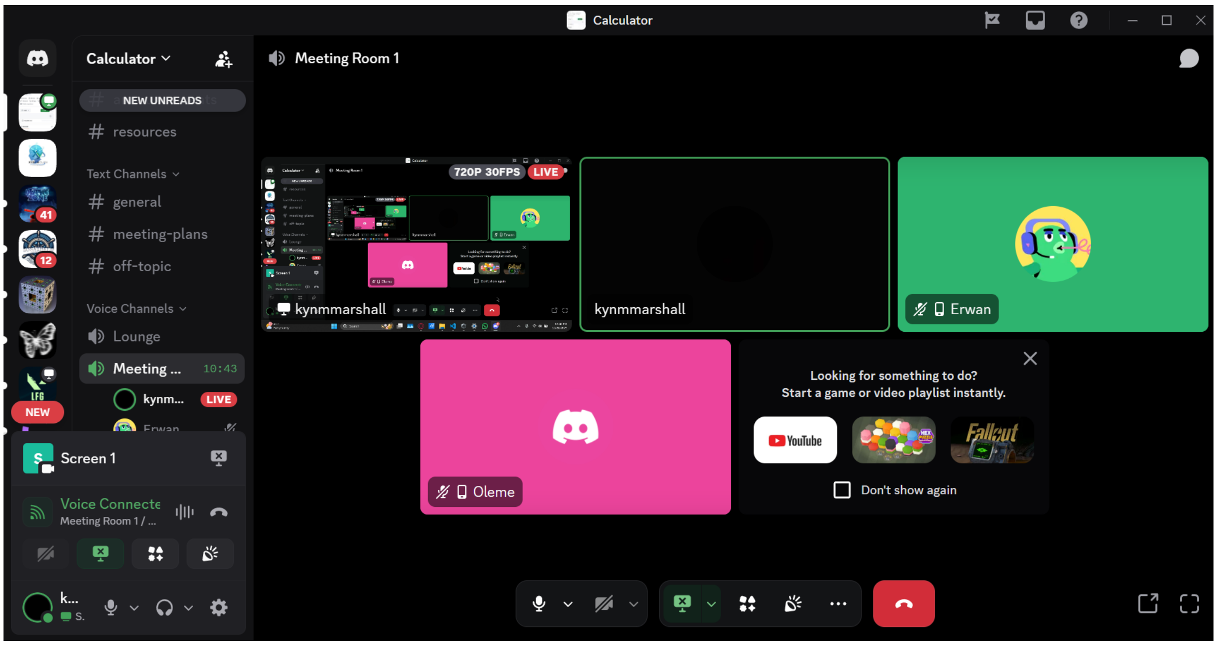
2.3 Sprint Execution

2.3.1 Daily Stand-up Meetings

Daily 15-minute stand-ups were conducted to:

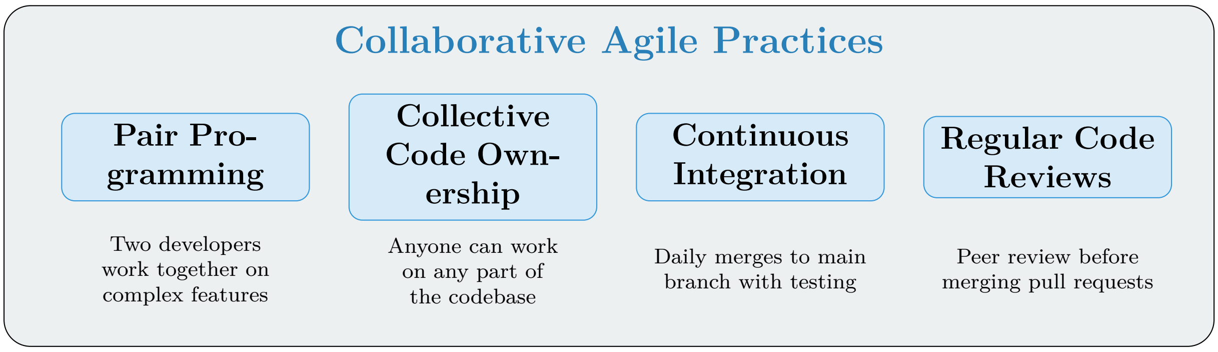
- Share progress since last meeting
- Plan work for the current day
- Identify blockers and dependencies

- Maintain team alignment



2.3.2 Collaborative Work Practices

The team employed several Agile collaboration techniques:


Collaborative Agile Practices

Pair Programming	Collective Code Ownership	Continuous Integration	Regular Code Reviews
Two developers work together on complex features	Anyone can work on any part of the codebase	Daily merges to main branch with testing	Peer review before merging pull requests

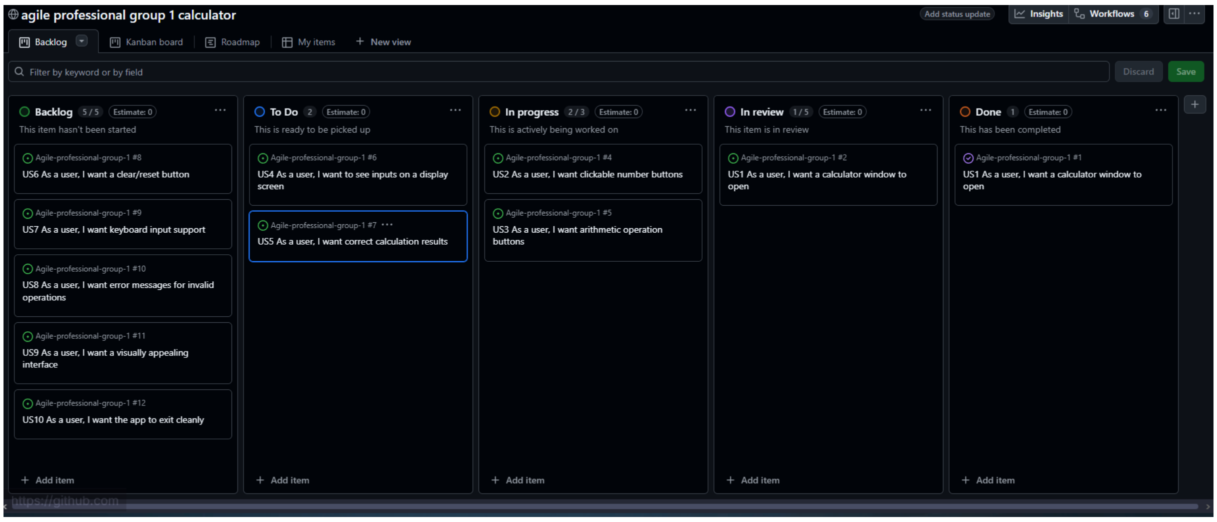
Figure 2.1: Team Collaboration Practices

2.3.3 GitHub Projects Implementation

The team used GitHub Projects as their Kanban board:

Column	Purpose	WIP Limit
Backlog	All identified tasks and user stories	No limit
To Do	Tasks planned for current sprint	8
In Progress	Tasks currently being worked on	3
Review	Completed tasks awaiting review	3
Done	Completed and accepted tasks	No limit

Table 2.3: Kanban Board Configuration



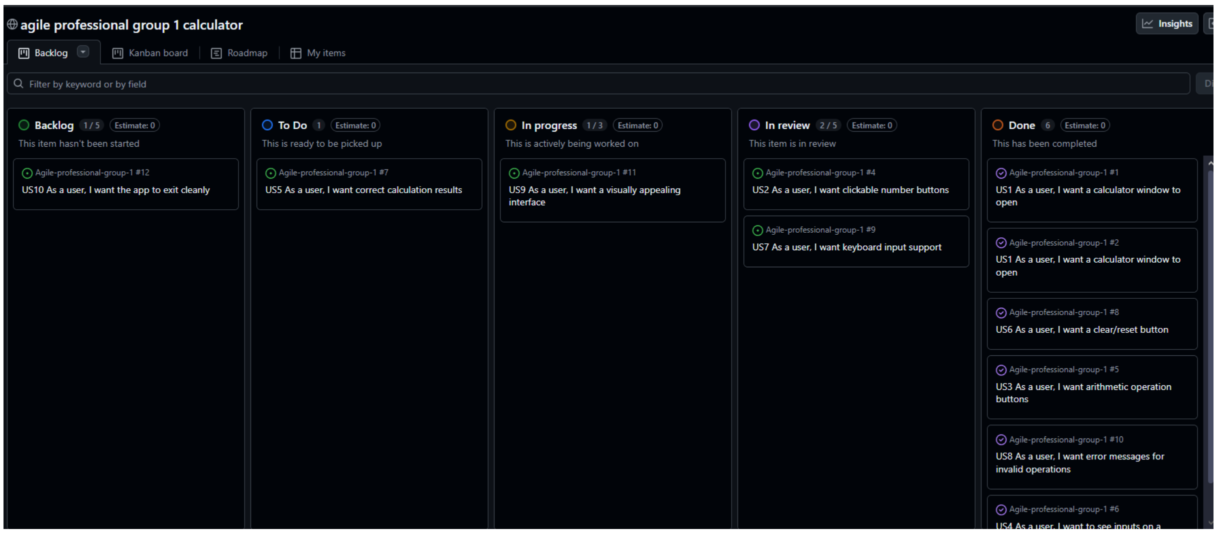
Backlog 5 / 5 Estimate: 0
This item hasn't been started
Agile-professional-group-1 #8
US6 As a user, I want a clear/reset button
Agile-professional-group-1 #9
US7 As a user, I want keyboard input support
Agile-professional-group-1 #10
US8 As a user, I want error messages for invalid operations
Agile-professional-group-1 #11
US9 As a user, I want a visually appealing interface
Agile-professional-group-1 #12
US10 As a user, I want the app to exit cleanly

To Do 2 Estimate: 0
This is ready to be picked up
Agile-professional-group-1 #6
US4 As a user, I want to see inputs on a display screen
Agile-professional-group-1 #7
US5 As a user, I want correct calculation results

In progress 2 / 3 Estimate: 0
This is actively being worked on
Agile-professional-group-1 #4
US2 As a user, I want clickable number buttons
Agile-professional-group-1 #5
US3 As a user, I want arithmetic operation buttons

In review 1 / 5 Estimate: 0
This item is in review
Agile-professional-group-1 #2
US1 As a user, I want a calculator window to open

Done 1 Estimate: 0
This has been completed
Agile-professional-group-1 #1
US1 As a user, I want a calculator window to open


agile professional group 1 calculator
Insights
Backlog
Kanban board
Roadmap
My items
Filter by keyword or by field

Backlog 1 / 5 Estimate: 0
This item hasn't been started
Agile-professional-group-1 #12
US10 As a user, I want the app to exit cleanly

To Do 1 Estimate: 0
This is ready to be picked up
Agile-professional-group-1 #7
US5 As a user, I want correct calculation results

In progress 1 / 3 Estimate: 0
This is actively being worked on
Agile-professional-group-1 #11
US9 As a user, I want a visually appealing interface

In review 2 / 5 Estimate: 0
This item is in review
Agile-professional-group-1 #4
US2 As a user, I want clickable number buttons
Agile-professional-group-1 #9
US7 As a user, I want keyboard input support

Done 6 Estimate: 0
This has been completed
Agile-professional-group-1 #1
US1 As a user, I want a calculator window to open
Agile-professional-group-1 #2
US1 As a user, I want a calculator window to open
Agile-professional-group-1 #8
US6 As a user, I want a clear/reset button
Agile-professional-group-1 #5
US3 As a user, I want arithmetic operation buttons
Agile-professional-group-1 #10
US8 As a user, I want error messages for invalid operations
Agile-professional-group-1 #6
US4 As a user, I want to see inputs on a

2.4 Sprint Review & Retrospective

2.4.1 Sprint Review Process

At the end of each sprint, the team conducted sprint reviews to:

- Demonstrate completed features to stakeholders (instructor)

- Gather feedback on functionality and user experience
- Update product backlog based on new insights
- Measure progress against sprint goals

2.4.2 Sprint Retrospective Outcomes

Key lessons learned from retrospectives included:

What Went Well	**Areas for Improvement**
Effective daily communication	Better task estimation techniques needed
Successful pair programming sessions	More comprehensive testing required
Good adherence to Agile ceremonies	Earlier identification of technical debt
Strong collaboration on complex features	Clearer definition of "Done" criteria

Table 2.4: Retrospective Findings

2.4.3 Definition of Done

The team established clear "Definition of Done" criteria:

1. Code is written and follows team standards
2. All unit tests pass
3. Code is reviewed by at least one other team member
4. Feature is tested manually
5. Documentation is updated
6. No critical bugs are known
7. Feature is integrated into main branch

Chapter 3

Challenges and Reflections on the Agile Process

3.1 Key Challenges Encountered

3.1.1 Technical Challenges

Challenge	Description	Agile Solution
Complex Feature Integration	Multiple context modes with different UIs	Incremental delivery with frequent integration
Technical Debt Accumulation	Quick fixes creating future problems	Regular refactoring sprints and code reviews
Skill Variations	Team members with different programming experience	Pair programming and knowledge sharing sessions
Testing Complexity	Context-aware features requiring extensive testing	Test-driven development and automated testing
Integration Conflicts	Multiple developers modifying same files	Feature branches and continuous integration

Table 3.1: Technical Challenges and Agile Solutions

3.1.2 Process Challenges

Challenge	Description	Agile Solution
Scope Creep	New features requested during development	Strict backlog management and sprint boundaries
Time Estimation	Difficulty estimating complex tasks	Planning poker and historical velocity tracking
Communication Gaps	Misunderstandings about requirements	Daily stand-ups and visual task boards
Work Distribution	Uneven workload among team members	Transparent task assignment and regular check-ins
Decision Making	Delays in making technical decisions	Time-boxed discussions and empowered teams

Table 3.2: Process Challenges and Agile Solutions

3.2 Agile Principles in Problem Solving

3.2.1 Responding to Change Over Following a Plan

When unexpected technical challenges emerged with the pattern recognition system, the team:

- Adapted the sprint backlog to address critical issues
- Reprioritized features based on new understanding of complexity
- Maintained focus on delivering working software despite plan changes
- Used retrospectives to improve estimation for similar future tasks

3.2.2 Customer Collaboration Over Contract Negotiation

The team maintained close collaboration with stakeholders (instructor) through:

- Regular sprint reviews with working software demonstrations
- Frequent feedback loops on feature implementation
- Adaptation of requirements based on practical constraints
- Transparent communication about progress and challenges

3.3 Effectiveness Evaluation

3.3.1 Strengths of Hybrid Agile Approach

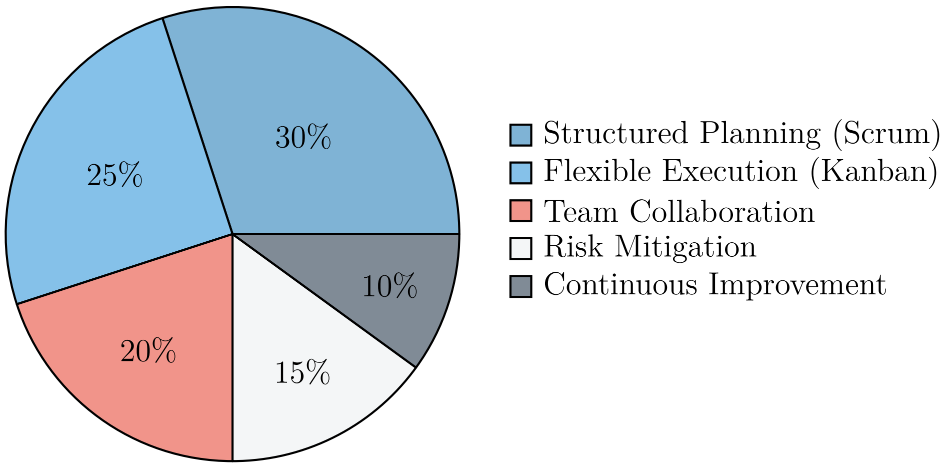


Figure 3.1: Effectiveness of Hybrid Agile Approach

3.3.2 Quantitative Metrics

Metric	**Target**	**Achieved**
Sprint Goal Completion	90%	85%
Daily Stand-up Attendance	100%	95%
Code Review Completion	100%	90%
Defect Rate	$< 5\%$	7%
Team Velocity	Increasing	Stable

Table 3.3: Agile Metrics and Performance

3.4 Lessons Learned and Recommendations

3.4.1 Key Lessons Learned

1. **Agile is Iterative**: Success comes from continuous improvement, not perfect initial planning
2. **Communication is Critical**: Regular ceremonies prevent misunderstandings and keep teams aligned
3. **Technical Excellence Matters**: Maintaining code quality enables sustainable pace and future changes
4. **Team Empowerment Works**: Trusting teams to make decisions leads to better outcomes and ownership
5. **Feedback Loops Are Essential**: Regular feedback prevents building the wrong thing

3.4.2 Recommendations for Future Projects

Area	Recommendation
Planning	Invest more time in initial backlog refinement and estimation
Technical Practices	Implement test-driven development from the beginning
Team Dynamics	Establish clearer role boundaries while maintaining collaboration
Process Adaptation	Be more flexible in adjusting ceremonies based on team needs
Tool Usage	Leverage more Agile project management tool features

Table 3.4: Recommendations for Future Agile Projects

Chapter 4

Conclusion and Final Reflection

4.1 Project Outcomes

4.1.1 Delivered Product

The team successfully delivered a functional Smart Context-Aware Calculator with:

- Five distinct context modes with specialized functionality
- Graphical user interface using Pygame library
- Input panels for context-specific values
- Pattern recognition for smart suggestions
- Comprehensive error handling and validation

4.1.2 Agile Process Outcomes

Agile Process Achievements

Working Software: Delivered functional calculator with all planned features

Team Collaboration: Effective 9-member team working with clear communication

Continuous Improvement: Regular retrospectives leading to process optimization

Customer Satisfaction: Stakeholder approval of delivered features and quality

Figure 4.1: Key Agile Achievements

4.2 Impact of Agile on Development Process

4.2.1 Positive Impacts

- **Early and Continuous Delivery**: Working software from week 2 enabled early feedback

- **Adaptability**: Successfully accommodated changing requirements and technical discoveries
- **Team Morale**: Regular successes and collaborative environment maintained motivation
- **Quality Focus**: Continuous integration and regular reviews improved code quality
- **Risk Management**: Early identification and mitigation of technical and process risks

4.2.2 Challenges Overcome

- Transformed initial uncertainty about feature complexity into manageable increments
- Converted communication challenges into structured collaboration patterns
- Turned technical obstacles into learning opportunities through pair programming
- Changed scope creep risks into disciplined backlog management

4.3 Final Reflection on Agile Methodologies

4.3.1 Value of Agile Principles

The project demonstrated that Agile methodologies provide substantial value in software development:

Agile Principle	Value Demonstrated in Project
Working Software	Delivered functional calculator despite initial complexity
Customer Collaboration	Regular feedback shaped features to match expectations
Responding to Change	Successfully adapted to technical challenges and new insights
Sustainable Pace	Maintained consistent progress without burnout
Technical Excellence	High-quality code enabled continuous integration and future changes

Table 4.1: Agile Principles in Practice

4.3.2 Recommendations for Agile Adoption

Based on the project experience, we recommend:

1. **Start Small**: Begin with basic Agile practices and expand as team matures
2. **Invest in Training**: Ensure all team members understand Agile principles
3. **Embrace Imperfection**: Accept that initial implementations will need refinement
4. **Measure What Matters**: Focus on outcome-based metrics rather than activity metrics
5. **Cultivate Culture**: Foster collaboration, transparency, and continuous improvement

4.4 Conclusion

The Smart Context-Aware Calculator project successfully demonstrated the practical application of Agile methodologies in a software development context. Through the hybrid Scrum-Kanban approach, the team delivered a complex application while experiencing firsthand the benefits of iterative development, continuous feedback, and adaptive planning. The project reinforced that Agile is not merely a set of practices but a mindset that, when embraced fully, leads to better software, happier teams, and more satisfied stakeholders. The lessons learned will serve team members well in future professional software development endeavors.

Chapter 5

Appendix : Project Artifacts

5.1 Sample User Stories

User Story ID	Description	Priority
US-001	As a user, I want to perform basic arithmetic operations	Must
US-002	As a student, I want trigonometric functions for homework	Should
US-003	As a shopper, I want to calculate tips and split bills	Should
US-004	As a budgeter, I want percentage-based calculations	Could
US-005	As a cook, I want recipe scaling functions	Could
US-006	As a user, I want to switch between calculator modes	Must
US-007	As a user, I want smart suggestions based on my calculations	Would

Table 5.1: Complete User Stories List

5.2 Sprint Burndown Chart Example

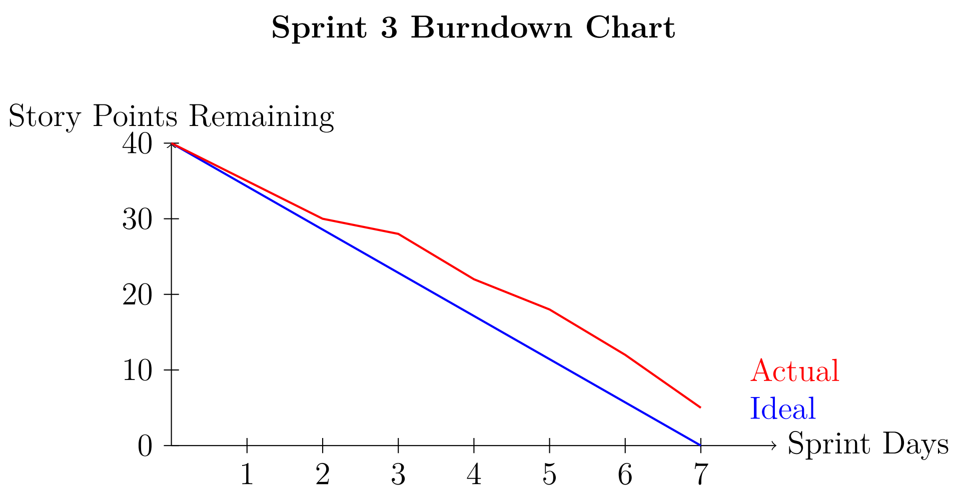


Figure 5.1: Sample Sprint Burndown Chart