PROJETO DE GRADUAÇÃO

AI-DRIVEN AGILE: ESTRATÉGIAS ÁGEIS E INTELIGÊNCIA ARTIFICIAL PARA GESTÃO DE REQUISITOS

Iara Gomes Andrade
170012484

Orientadora
Prof.ª Dr.ª Simone Borges Simão Monteiro

Brasília, 7 de julho de 2025.

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Departamento de Engenharia de Produção

PROJETO DE GRADUAÇÃO

AI-DRIVEN AGILE: ESTRATÉGIAS ÁGEIS E INTELIGÊNCIA ARTIFICIAL PARA GESTÃO DE REQUISITOS

Iara Gomes Andrade
170012484

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de Produção.

Banca Examinadora

Prof.ª Dr.ª Simone Borges Simão Monteiro, UnB/EPR (Orientadora)
Prof.ª Dr.ª Edna Dias Canedo, UnB/Ciência da Computação
Prof. Dr. Daniel Russo, Aalborg University/Ciência da Computação

Brasília, 7 de julho de 2025.

FINAL PAPER

AI-DRIVEN AGILE: AGILE STRATEGIES AND ARTIFICIAL INTELLIGENCE FOR REQUIREMENTS MANAGEMENT

Iara Gomes Andrade
170012484

Advisor
Prof. Simone Borges Simão Monteiro, PhD

Brasília, July 7, 2025.

UNIVERSITY OF BRASÍLIA

FACULTY OF TECHNOLOGY

PRODUCTION ENGINEERING DEPARTMENT

UNIVERSITY OF BRASÍLIA
Faculty of Technology
Production Engineering Department

FINAL PAPER

AI-DRIVEN AGILE: AGILE STRATEGIES AND ARTIFICIAL INTELLIGENCE FOR REQUIREMENTS MANAGEMENT

Iara Gomes Andrade
170012484

Final paper submitted as a partial requirement
to obtain a Production Engineer degree.

Examination Board

Prof. Simone Borges Simão Monteiro, PhD - UnB/EPR (Advisor)
Prof.ª Dr.ª Edna Dias Canedo, UnB/Computer Science
Prof. Dr. Daniel Russo, Aalborg University/Computer Science

Brasília, July 7, 2025.

'Don't let anyone rob you of your imagination, your creativity, or your curiosity. It's your place in the world; it's your life. Go on and do all you can with it, and make it the life you want to live.'

Mae Jemison.

ACKNOWLEDGMENTS

Having role models allows us to dream, opens our eyes to what is possible, and shows us how to go beyond what is expected. Fortunately, I had several who guided me throughout my academic journey. Starting with my mother, Andréia Gomes, an exemplary teacher who was not only a reference for me, but for many of the students who had the opportunity to attend her classes. In addition to being a professional who never stopped striving for excellence, it was from her that I learned a passion for reading. She greatly raised the bar for who I want to be as a woman and reminds me that we can indeed work with what we love and find fulfillment in it. My father, Reuber Andrade, also a teacher, is my role model in terms of how curiosity and enthusiasm for something can make life more fun and interesting. My parents gave me every opportunity to become whoever I wanted to be. I saw the sacrifices you made, and I am grateful for them.

My dear friend, Ana Júlia Oliveira, thank you for your support and for being such a significant part of my life over the last decade, I cherish our friendship. From boss to friend, Letícia Barros, our long professional conversations and reflections on life make me happy and inspired.

My sincere thanks to Professor Daniel Russo for welcoming me to his classes in Copenhagen. It was a laborious and enjoyable process. My work would not have been the same without the guidance of Professor Simone, who was present not only during this final phase of the course but throughout the development of the Educado project, which I have been part of for the last three years.

Having someone to celebrate your achievements with and cry with during difficult times makes the entire academic journey easier. That someone for me is João Vítor Garcia, whom I can call my partner, friend, and companion. Having you by my side (with many home-cooked dinners, laughter, and curiosities) has certainly made all the difference. Thank you for believing in me many times more than I believe in myself. I love sharing memories with you, and I look forward to the next chapters of our story.

RESUMO

A elicitação de requisitos é um aspecto crítico do desenvolvimento de software, pois as equipes ágeis frequentemente enfrentam dificuldades para equilibrar a flexibilidade e a qualidade da documentação. Essa contradição pode resultar em requisitos pouco claros e, consequentemente, em falhas e ineficiências no projeto. Para abordar esse desafio, esta pesquisa propõe melhorias no processo de descrição de requisitos de software, usando metodologias ágeis, especificamente Scrum e Shape-up, e o modelo de inteligência artificial GPT-4o para promover a autonomia do desenvolvedor e garantir entregas de acordo com as expectativas das partes interessadas. Um estudo de caso real foi realizado no projeto Educado, uma iniciativa de software desenvolvida por uma equipe internacional como parte do portfólio Egalitarian ERAMUS+ que visa levar educação aos catadores de lixo do sistema de gestão de resíduos de Brasília. Por meio de uma abordagem de pesquisa-ação, o estudo analisa os aprimoramentos iterativos da documentação de requisitos usando análise de documentos, entrevistas semiestruturadas e observações em tempo real ao longo do ciclo de desenvolvimento. Os resultados revelaram um aumento relativo de 88,89% na taxa de entrega e uma redução de 80% do retrabalho comparado à primeira fase do ciclo. Além disso, o tempo médio do esforço operacional de escrita assistida por inteligência artificial reduziu em aproximadamente 83%. O *roadmap* proposto demonstra como refinar um item do *backlog* de forma a acelerar a documentação sem comprometer a precisão, preenchendo as lacunas do gerenciamento ágil moderno de produtos de software.

Palavras-chave: Inteligência Artificial, Gerenciamento de Backlog, Elicitação de Requisitos, Desenvolvimento de Software, Metodologia Ágil, Gerenciamento de Produtos de Software.

ABSTRACT

Requirements elicitation is a critical aspect of software development since agile teams often struggle to balance flexibility and documentation quality. This contradiction can result in unclear requirements and, consequently, project failures and inefficiencies. To address this challenge, this research proposes improvements to the software requirement description process, using agile methodologies, specifically Scrum and Shape-up, and GPT-4o artificial intelligence model to promote developer autonomy and guarantee deliveries in line with stakeholders' expectations. A real-world case study was conducted in the Educado project, a software initiative developed by an international team as part of the Egalitarian ERAMUS+ portfolio, which aims to bring education to waste pickers from the Brasília waste management system. Through an action research approach, the study analyzes iterative enhancements to requirement documentation using document analysis, semi-structured interviews, and real-time observations throughout the development cycle. The results revealed a relative increase of 88.89% in the delivery rate and a reduction of 80% in rework compared to the first phase of the cycle. Additionally, the average operational effort for AI-assisted writing was reduced by approximately 83%. The proposed roadmap demonstrates how to refine a backlog item to accelerate documentation without compromising accuracy, bridging gaps in modern agile software product management.

Keywords: Artificial Intelligence, Backlog Management, Requirements Elicitation, Software Development, Agile Methodology, Software Product Management.

SUMMARY

1 INTRODUCTION 9

1.1 JUSTIFICATION 10
1.2 OBJECTIVE 10
1.2.1. Overall Objective 10
1.2.2. Specific Objectives 10
1.3 WORK STRUCTURING 11

2 THEORETICAL BACKGROUND 12

2.1 SOFTWARE PRODUCT MANAGEMENT 12
2.2 SOFTWARE REQUIREMENTS THROUGH AGILE METHODOLOGIES 13
2.3 ARTIFICIAL INTELLIGENCE APPLIED TO THE SOFTWARE REQUIREMENTS ELICITATION . 17

3 METHODOLOGY 19

3.1 RESEARCH METHOD 19
3.2 RESEARCH STRUCTURE 19

4 REQUIREMENTS METHODS DEFINITION 22

4.1 APPROACH MAPPING FOR REQUIREMENT DOCUMENTATION 22
4.2 SELECTION OF METHODS FOR EDUCADO'S CONTEXT 24

5 DEVELOPMENT CYCLE PREPARATION 25

5.1 EDUCADO FOUNDATIONAL OVERVIEW 25
5.2 DEVELOPMENT METHODOLOGICAL APPROACH 28
5.3 BACKLOG SETUP 29

6 DEVELOPMENT CYCLE EXECUTION 33

6.1 INITIAL REQUIREMENT DESCRIPTION APPROACH WITH USER STORIES 33
6.2 ADAPTING USER STORIES REQUIREMENTS WITH SHAPE-UP PRINCIPLES 35
6.3 AI-ENHANCED REQUIREMENT DESCRIPTIONS 38

7 RESULTS ANALYSIS 42

7.1 ITERATION RESULTS EVALUATION COMPARISON 42
7.2 SOFTWARE REQUIREMENT DOCUMENTATION ROADMAP 45

8 CONCLUSION 49

REFERENCES 51

APPENDIX I – INSTRUCTION PROMPT FOR EDUCADO REQUIREMENTS AI AGENT 56

APPENDIX II – INPUT AND OUTPUT OF THE EDUCADO REQUIREMENTS AI AGENT 57

APPENDIX III – SEMI-STRUCTURED INTERVIEW WITH DEVELOPERS 59

LIST OF FIGURES

Figure 1. Software Development Life Cycle Phases. 12
Figure 2. Software Requirement Phases. 14
Figure 3. First Educado Mobile App MVP. 26
Figure 4. Second Development Cycle Management through the tool Miro 26
Figure 5. Second Educado Mobile App and Website MVP 27
Figure 6. Scrum Lifecycle 29
Figure 7. Educado Backlog Structure 30
Figure 8. Backlog Setup within GitHub Projects 31
Figure 9. PBI Example within GitHub Projects 31
Figure 10. PBI Documentation of Sprint 1 34
Figure 11. Sprint 1 Outcomes. 35
Figure 12. PBI Documentation of Sprint 2 37
Figure 13. Sprint 2 Outcomes 38
Figure 14. PBI Documentation of Sprint 3 40
Figure 15. Sprint 3 Outcomes 41
Figure 16. Performance Comparison Through the Phases. 43
Figure 17. Relative Variation Compared to the First Phase 44
Figure 18. Requirement Documentation Workflow 47

LIST OF TABLES

Table 1. Research Phases and Techniques. 20
Table 2. Requirements Elicitation Methods Comparison. 22
Table 3. LLMs Comparison. 23
Table 4. Historical Timeline of the Educado Project. 27
Table 5. Documentation Phases Comparison. 45
Table 6. Documentation Roadmap for Agile Backlog Items. 46

1 INTRODUCTION

Software development, characterized by constant technological innovation, has experienced accelerated growth in various industries, representing a business transformation in a dynamic environment (Anitha Gracy et al., 2024). However, 63% of the software projects fail (Pattyn, 2023) due to inadequate project management, low coverage tests, insufficient stakeholder engagement, and poor communication. All these factors can be associated with unclear requirements, where the team misunderstands the user's needs and develops a product that doesn't meet expectations, leading to frustration and project abandonment (Elentukh, 2023; Shokrizadeh & Shokrizadeh, 2024).

In order to produce quality software, the requirements operate as the reference point for multidisciplinary team management (Behutiye et al., 2020). Nevertheless, its documentation faces challenges in agile teams as it is neglected while prioritizing code with limited discovery. Although an alternative is the documentation found in the traditional method, its rigid and overly extensive structures don't fit in with an agile environment (Kalfat, Oussalah, et al., 2023). This contradiction emphasizes the need for a requirements description restructuring. Such a process should aim to reduce operational time while enhancing team autonomy to propose solutions aligned with the product value delivered to users.

In line with these challenges and opportunities, this research is dedicated to the Educado software project, which is part of the Egalitarian's portfolio, founded by the European Union ERASMUS+ Program due to an international partnership between Aalborg University, Saxion University, the University of Minho, and the University of Brasilia (Egalitarian, 2024). This initiative follows Problem-Based Learning (PBL) concepts so that students can apply theoretical knowledge to real-world problems (Smith et al., 2022), in this case, waste management in Brasilia.

The closure of the second largest open dump in Latin America in 2018 solved the work conditions issues, but it has generated others for those who were used to that reality. For example, before workers were paid daily, with the cooperatives, they started to be paid fortnightly or monthly, which led to difficulty in managing the finances (Agência Brasília, 2018). Also, there was an increase in the difficulty of adding value to the material, raising the need for technical training on sorting, upcycling, and quality control. In response to this situation, initiatives are fundamental to empower these workers with essential life and professional skills, promoting well-being, even in conditions of socio-economic vulnerability.

After a few iterations, the project lacks standardization in the description of requirements, which results in misalignment between expectations and delivery. This has led to usability issues and recurring gaps between what is planned and what is developed. Therefore, how can a strategic

requirement management improve software delivery speed and quality, while ensuring that multidisciplinary teams have a shared understanding of what is expected from each one?

1.1 Justification

The Political-Pedagogical Project of the Production Engineering Program at the University of Brasília (University of Brasília, 2016) aims to graduate professionals with a systemic vision and critical thinking to operate in complex contexts while promoting innovative and sustainable solutions. Beyond the economic and ecological aspects, sustainability has a social responsibility, highlighting the importance of digitalization to actively address the Sustainable Development Goals (SDGs) challenges (Brenner & Hartl, 2021).

With that perspective, Educado was idealized in 2018 in a multicultural and resource-constrained context, integrating a website platform for content creation and a mobile application for students to promote education and better living opportunities for waste pickers in Brasília (Melo, 2024). The project is directly related to quality education (SDG 4), decent work and economic growth (SDG 8), and reduced inequalities (SDG 10) (United Nations, n.d.). That context creates an opportunity to explore how agile methodologies and innovative tools can enhance the value delivered to end users, giving a response in each iteration of its software development cycle.

According to the program's pedagogical guidelines, graduates from the Production Engineering course at the University of Brasília are expected to apply technological tools to integrate knowledge in management, viable engineering solutions, and methods for better decision-making. These competencies are reflected in this study through a collaborative environment involving international partnership, enabling more autonomous, organized, and effective deliveries, even under constraints.

1.2 Objective

1.2.1. Overall Objective

Propose improvements to the software requirement description process, using agile methodologies, specifically Scrum and Shape-up, and GPT-4o Artificial Intelligence (AI) to promote developer autonomy and guarantee high-quality deliveries in line with stakeholders' expectations, primarily end-users.

1.2.2. Specific Objectives

I. Identify methods of describing software requirements using agile methodology and artificial intelligence;
II. Select methods that adhere to Educado software, the case study of this work;
III. Analyze the contextual background of previous project phases to support the decisions adopted in this development cycle;
IV. Integrate the scope definition practices from the Shape Up and Scrum methodology with artificial intelligence tools to improve agility in project management;
V. Analyze the results of each sprint and the developer feedback to enhance the development process;
VI. Define a roadmap for software requirements definition, utilizing artificial intelligence techniques to enhance process efficiency and accuracy.

1.3 Work Structuring

This work is structured as follows: Chapter 1 – Introduction, which presents the research problem and its relevance within the context of the Educado software; Chapter 2 – Theoretical Background, provides a foundational literature review to support the study; Chapter 3 – Methodology, outline the research methods employed to achieve the proposed goals; Chapter 4 – Educado Foundational Overview, detail the background context of the Educado project; Chapter 5 – Third Development Cycle Preparation, focus on the preparatory phase for the research implementation; Chapter 6 – Third Development Cycle Execution, describe the application of the research within the development iterations; and Chapter 7 – Result Analysis, present an analysis of the findings and their implications.

2 THEORETICAL BACKGROUND

2.1 Software Product Management

Successful software products are related to how the development delivery aligns with business goals and market needs. To make this happen, Software Product Management (SPM), a strategic discipline, performs to ensure strong communication and collaboration between stakeholders and team members (Peters, 2024). This subject differs from software process management in that it focuses on defining methodologies and practices that govern the software development lifecycle (Sidhu, 2022).

The Product Manager (PM) is commonly recognized as being responsible for overseeing software product management and ensuring coordination across teams while managing risks and resources. That role develops detailed project plans, adjusts scope and schedules as needed, and maintains communication with stakeholders to ensure timely delivery within budget. On the other hand, in some companies, the Product Owner (PO) role is assigned to ensure the product delivers maximum value by working closely with user groups, prioritizing the backlog, and making key decisions on requirements and releases (Patel, 2023). In many cases, these positions are performed by just one professional due to overlapping responsibilities.

The Software Development Life Cycle (SDLC) guides teams through the creation of high-quality software applications, ensuring efficiency and alignment with user requirements (Shetty et al., 2023). As defined in Figure 1, it comprises six key phases: (i) Planning, defining the product scope, resource constraints, and goals; (ii) Requirements Gathering, where project goals and scope are defined with stakeholders; (iii) Design, which outlines the architecture, structure, and components of the software; (iv) Development, where coding transforms designs into a functional product; (v) Testing, ensuring the software meets requirements and functions correctly; and (vi) Deployment and Maintenance, marking the transition to operational use and involving updates and issue resolution to enhance functionality over time (Ghumatkar & Date, 2023).

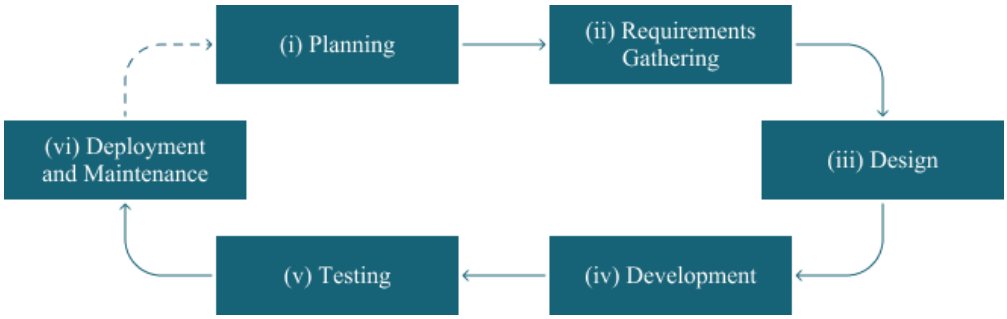


Figure 1. Software Development Life Cycle Phases. Adapted from Ghumatkar & Date (2023).

Intricately to that cycle, improved software management follows the roadmap proposed by Figure 1 with the intention of making well-founded decisions based on the tracked development

progress. Additionally, it incorporates change management to adapt to evolving business needs and agile practices for flexibility and momentum (Ibeh et al., 2024). Therefore, it becomes essential to efficiently oversee the software development lifecycle through systematic management, ensuring that each phase is addressed thoroughly. This strategy aims to easily set clear milestones, reduce costs, allocate resources, and build high-quality software products (Patel, 2023).

These agile practices, as seen in Scrum, Kanban, Lean, and Shape-up methodologies, emerged as a response to and in contrast with traditional methods that represented a more sequential and rigid structure and don't require changes due to a complex and well-defined product scope (Suvvari, 2024). Even though these traditional approaches, such as Waterfall, V-Model, Spiral Model, and Iterative Development, have been validated across multiple organizations, they still exhibit limitations when operating in fast-paced development environments, hindering innovation. These include inflexibility to change, delayed feedback, reliance on clearly defined requirements, limited stakeholder involvement, resource intensity, and challenges in managing cross-functional roles effectively. In that context, there was a need for a more flexible framework that can quickly adapt in response to a market in continuous changes (Parikh, 2024).

The evolution of agile methodologies represents a transformative shift in the technology industry, which began with the creation of the Agile Manifesto in 2001, prioritizing individuals, interactions, and incremental development over rigid processes and plans. Despite obstacles such as resistance to change and scaling issues, agile methodologies are expected to evolve further through the influence of emerging technologies, ensuring ongoing relevance in the dynamic technology landscape (Suvvari, 2024).

2.2 Software Requirements Through Agile Methodologies

Software development starts with understanding stakeholders' needs, which will be translated into the technical team's language, bridging the gap between the user and the developer. This communication should continue throughout the product development since the requirements gathering phase represents 60% of the project cost (Ashraf et al., 2021). This stems from poorly defined, non-user-centric requirements, which lead to rework, disrupting workflows, and reducing productivity. As inefficiencies grow, the development timeline extends, delaying product delivery and driving up costs.

Software requirements are the core of successful software development as they define the problem the product intends to solve and ensure that it is aligned with the user’s needs and technical constraints. This artifact implies preventing project failures caused by incomplete or

misunderstood goals and integrating risk management to address issues as scope changes (Wiegers & Beatty, 2013). Therefore, requirements guide the development cycle from the initial planning phase through design, implementation, testing, and maintenance, demanding a high understanding of all the possible factors affecting scope discovery.

Depending on context and product goals, distinct strategies for overseeing requirements may be necessary, each with specific strengths and challenges that influence their suitability (Popoola et al., 2024). Traditional environments are committed to comprehensive upfront documentation, focusing on thoroughly capturing both functional and non-functional requirements. However, this rigidity often results in challenges when requirements evolve, as changes require formal processes, and communication barriers cause misaligned expectations (Mesquita et al., 2023). Conversely, agile methodologies prioritize fostering continuous stakeholder collaboration and real-time adjustments. While this approach allows for greater adaptability and customer engagement, it may overlook quality requirements such as performance and security unless specifically addressed (Zaveri et al., 2024; Khan & Ahmed, 2023).

According to ISO/IEC 12207:2008, the requirements engineering process involves the stages illustrated in Figure 2. The initial phase is related to the stakeholder need identification, documented and prioritized, followed by the phase where it's transformed into a technical system requirement. These are formalized through the Specification into detailed documents that guide the development and enable traceability. The Validation step ensures that it meets expectations while eliminating conflicts and ambiguities. Finally, Management contains change control throughout the system's life cycle.



Figure 2. Software Requirement Phases. Adapted from ISO/IEC (2008).

The international standard can be used to adapt both agile and traditional contexts, as their processes do not need to be executed sequentially and can occur iteratively or even simultaneously. Following the phases from Figure 2, methods were created to structure the requirements elicitation process in agile environments, ensuring a systematic approach to capture user needs, among them:

- **Design Thinking:** Focused on innovative and user-centered solutions based on the phases: immersion with the context deeply explored in interviews and observations, ideation of the solution using creative techniques as brainstorming, and prototyping for validation and user feedback. This approach is used to identify and validate requirements of the solutions (Martins et al., 2019);
- **Dual-track Agile:** Integrates continuous requirements elicitation into the product

development, involving regular customer collaboration for decision-making. It follows two parallel tracks: discovery to focus on understanding, validating, and testing solutions before implementation; and delivery to track the execution of validated ideas. The requirements elicitation process is treated as an ongoing practice to continuously adapt given the progress (Torres, 2021);

- **USARP (Usability Requirements with Personas and User Stories):** Designed to specify usability requirements by integrating functional usability guidelines with user profiles. It provides structured guidance to facilitate discussions among the team and get the user's needs translated into user stories without heavy formalization (Fonseca et al., 2024);
- **ACUX (Acceptance Criteria of User Experience):** Incorporates user experience requirements into the precise definition of acceptance criteria with the format: "Given [scenario], when [action], then [benefit]". The structure is organized around interaction, information organization, and visual elements, providing examples and templates as instructions. This method promotes the detailed specification of user interactions and behaviors but requires more cognitive effort from the team. (Fonseca et al., 2024);
- **Scrum:** Includes continuous requirements elicitation practices via the product backlog and evolves with each iteration, although it operates as a development framework. Through backlog refinement, the requirements are analyzed and prioritized, allowing adaptation based on user expectations and challenges and continuous management (Ramadan & Megahed, 2016);
- **Shape-up:** Focused on defining problems and feasible solutions, emphasizing the creation of requirements outlines with a balance of autonomous development. The process involves setting boundaries, defining the problem, exploring potential solutions, and addressing risks upfront (Singer, 2019).

Despite the methodology adopted, requirements elicitation is an iterative process that uses data collection techniques specific to the project context. These can include: Interviews, direct and structured conversations with stakeholders to capture detailed functional requirements; Workshops, collaborative alignment but limited by scheduling conflicts or dominant voices; Prototyping, allow to visualize and validate potential solutions; Surveys, reach large or geographically dispersed user groups but with reduced depth of insight; and Observation, facilitating the identification of implicit or behaviors requirements by monitoring user interactions (Zhang, 2007).

Given that it is a cyclical process, as new information is revealed, this can lead to more questions and additional sessions to define the requirements more precisely, requiring several analyses and refinement rounds (Wiegers & Beatty, 2013). A single requirements elicitation

technique is often insufficient to address the varied stakeholders' expectations in complex and multifaceted business environments. Instead, combining multiple and customizing them to the specific context is crucial for effective and comprehensive requirements gathering (Barros et al., 2024).

While these techniques aim to improve clarity and communication, the challenges inherent in how this information reaches the development team, such as ambiguity and incompleteness, highlight the need for artifacts to document and refine requirements. (Kulkarni & Yadav, 2021). The following practices guide on how to better describe the Software Requirement Specifications (SRS):

- **Natural Language:** Expressed in everyday language without clear format so the requirements are accessible to a wide range of stakeholders, including the technical and non-technical teams (Norheim & Rebentisch, 2024);
- **User Stories:** Concise description of each software feature, written from the end user's perspective to capture what they do or need to complete their task. It follows the structure "As a [type of user], I want [action] so that [benefit]" (Kustiawan & Lim, 2023);
- **User Flow:** Visual representation of the path users take while interacting with a system, encompassing their behaviors and decision-making throughout the process, including success and error states (Akiki, 2019);
- **Acceptance Criteria:** Define the conditions under which a software requirement must meet to be considered complete, serving as a checklist for the development team delivery (Martinelli et al., 2022);
- **Shaping:** Strategic practice aimed at refining the scope by creating a solution with clear boundaries and concentrating on the value that needs to be delivered. The approach includes problem identification, high-level solution definition by the technical team, potential risks anticipation, and concept visualization using sketches or diagrams (Singer, 2019).

Unlike traditional requirements, which emphasize what technology the system should use, these approaches prioritize understanding and addressing the value implied in it. By documenting requirements with a user-centered focus, the development team gains decision-making autonomy to determine the most suitable solutions given a set timeframe (Signoretti, 2020). Therefore, it should be underscored that the real priority for users lies in the value delivered to meet their needs and not in the technical details of how this is done.

Effective requirements management concerns the structure for organizing, prioritizing, and continuously refining the requirements. To achieve that, the Product Backlog serves as a mechanism for operationalizing incremental deliveries by acting as a dynamic repository. This

process enables continuous prioritization and centralized communication from the product development overview (Sedano et al., 2019). Product Backlog Items (PBIs) represent all the functionalities, enhancements, and fixes that will incorporate future releases, improving the backlog organization aspect. Each item contains the requirement documentation and is prioritized and refined once a new interaction begins (Pang, 2020).

2.3 Artificial Intelligence Applied to The Software Requirements Elicitation

Software development has continually advanced with problem-solving technologies, requiring the area of software product management to evolve to keep up with these advances. One of the most outstanding and extensively tested innovations in recent times is Artificial Intelligence (AI), an automation tool to reduce repetitive or complex tasks traditionally requiring human intervention (Ozkaya, 2023). The evolution of this technology in software engineering can be traced back to when basic algorithms were developed to automate simple tasks, limited to offering suggestions, formatting code, and identifying errors in code. Recently, AI has transformed from rule-based systems to models that can learn from data and improve over time, generating code, tracing dependencies, and even interpreting requirements (Benitez & Serrano, 2024).

The requirements elicitation process has the potential to become more efficient with AI-powered capabilities integration as it can enhance accuracy and reduce ambiguities, leading to high-quality software product delivery that meets user and business expectations (Popoola et al., 2024). This comprehensive technology is leveraged by the advanced techniques:

- **Natural Language Processing (NLP):** Computer program that has the ability to understand, interpret, and generate human language in a valuable way. It helps to analyze textual requirements documents automatically, extract key information, and identify implicit requirements that may not be stated (Alkabour & Alrwais, 2023);
- **Machine Learning (ML):** Algorithm capable of predicting changes based on historical data. Throughout the product context, it identifies patterns to improve the elicitation process efficiency while automating prioritization, resolving conflicts, and managing dependencies between requirements (Ali Fadlallah & Yahia Eldow, 2024);
- **Human-Computer Interaction (HCI):** Principles integrated into interactive interfaces and visualization tools to facilitate real-time feedback (Popoola et al., 2024).

Leveraging AI techniques, Generative AI, particularly Large Language Models (LLMs), are designed and trained on vast amounts of data within a specific context to generate human-like text based on the input they receive (Sami et al., 2024). These models have practical application in requirements specification by automatically structuring according to description practices and

templates, as presented in Section 2.2. However, the instruction (prompt) sent to the LLMs demands a high level of detail with sufficient context to produce relevant results (Arora et al., 2023). In the software development context, the following models are in great demand, given their applications within the requirement engineering:

- **ChatGPT:** Designed to understand and produce human-like text, optimized for nuanced, context-rich interactions. It can facilitate the collection, analysis, and documentation of requirements, reducing manual effort while speeding up processes that are traditionally time-consuming and prone to human error (Marques et al., 2024);
- **Gemini:** Designed for specialized tasks with precise context, being less flexible in general conversational contexts. It is effective for requirements extraction, classification, and named entity recognition, but its performance varies depending on the level of expert knowledge provided in the prompts (Saleem et al., 2025);
- **Copilot:** Code generation tool designed to assist developers in writing code more efficiently. It can generate function bodies, complete code snippets, and suggest implementations based on natural language. However, it can lead to the reproduction of issues instead of resolution, depending on the prompt quality and structure (OBrien et al., 2024);
- **Claude:** Optimized for interpretability, alignment with human values, and safe decision-making. It provides contextual understanding, infrastructure and architectural recommendations, error reduction and safety, and scalability strategies. Works for projects requiring precise responses in complex software environments and technical documentation (Singh et al., 2025).

AI still has progress to be made since human intervention is necessary to monitor and correct outputs. According to Ronanki et al. (2023), models such as ChatGPT-4o have the potential to enhance the software requirement aspect by saving time and effort in repetitive and technical tasks, allowing the development team to focus on creative and strategic activities.

3 METHODOLOGY

3.1 Research Method

This research is classified as applied and adopts a qualitative approach, considering that it aims to address specific challenges within software development, particularly in the context of defining and managing software requirements. Applied research focuses on solving real-world problems by utilizing theoretical insights and adapting them to practical applications (Kothari, 2008).

The qualitative approach concentrates on exploring and understanding complex phenomena within their natural context, investigating the influence of this scenario on behavior or outcomes. This approach is essential in this work to explore the challenges and opportunities to improve software requirements documentation. This method was implemented using action research, emphasizing active collaboration and iterative cycles (planning, executing, observing, and reflecting) for problem-solving, refining practices closely with developers and stakeholders in real time (Tripp, 2005).

The Educado project, a multicultural and interdisciplinary initiative developed by Danish students at Aalborg University (AAU), in partnership with University of Brasília (UnB), serves as the case study by performing the Product Owner role during three months of the course 'Software Engineering 5' of the Computer Science department. The discipline is designed to offer students hands-on experience in development dynamics with agile methodologies, particularly the Scrum framework. The analyzed semester involved 36 students, organized into six teams.

Although this is the third development cycle, marking the third academic year of collaboration between the universities, prior semesters included user analysis and product feasibility validation, supporting the current implementation. This ongoing project provides a setting to iteratively test the solutions proposed that directly address specific needs.

Furthermore, data were collected using (i) Document Analysis, examination of previous development cycle records; (ii) Observations, monitorization of the task's productivity; (iii) Semi-Structured Interviews, conducted with the development team to capture perceptions and feedback on the implemented methodologies. The analysis to identify patterns, themes, and insights was done through iterative evaluation, where the results from each interaction are used to refine tools and processes in the subsequent ones.

3.2 Research Structure

The research structure is detailed in Table 1, which breaks down the phases developed into

specific stages that guide the progression and focus of the work.

Table 1. Research Phases and Techniques.

Objectives	Research Phase	Research Stages	Techniques
Identify methods	1 Requirements Methods Definition	1.1 Approach Mapping for Requirement Documentation	Literature Review in Indexed Databases (WoS, Scopus)
Select methods		1.2 Selection of Methods for Educado's Context	Literature Review in Indexed Databases (WoS, Scopus)
Analyze the contextual background	2 Develpment Cycle Preparation	2.1 Educado Foundational Overview	Document Analysis
		2.2 Development Methodological Approach	Scrum Methodology
		2.3 Backlog Setup	GitHub Project
Integrate agile methodology with artificial intelligence tools	3 Development Cycle Implementation	3.1 Initial Requirement Description Approach	Scrum User Stories
		3.2 Adapting Requirements with Shape-Up Principles	Shape-up Integration
		3.3 AI-Enhanced Requirement Descriptions	ChatGPT-4o
Analyze the results	4 Results Analysis	4.1 Iteration Results Evaluation Comparison	Semi-Structured Interviews, Observations, Document Analysis, Sprint Performance Metrics
Define a roadmap		4.2 Software Requirement Documentation Roadmap	Findings Consolidation

Created by the author.

The first phase establishes suitable approaches to improve software requirement documentation, which is supported by a literature review (1.1). Based on this mapping, the methods are selected to align with the Educado program's context according to their potential to address development challenges identified (1.2).

The second phase focuses on preparing for the development cycle selected as the focus of this study by covering the contextual background and the groundwork for a development methodology. Between 2018 and 2024, over 40 interdisciplinary teams from the University of Brasília and Aalborg University produced user studies and MVPs (Minimum Viable Product). These previous development cycles are reviewed using document analysis, highlighting key milestones and identifying challenges and results achieved that inform the current research (2.1).

Also, the methodology adopted to meet the project's demands is Scrum, which is outlined in this stage, including sprint ceremonies and collaboration practices (2.2). As a Product Backlog Item (PBI) format, the requirements are organized in the backlog using GitHub Projects, ensuring efficient prioritization and alignment with the product goals (2.3).

The third phase implements the development cycle through testing and refining different requirement description approaches. The initial iteration delivered the requirements as unaltered user stories in accordance with Scrum practices with a low level of detail (3.1). The following stage incorporated the shaping principles from Shape-up methodologies, focusing on a more

detailed scope and refining deliverables before execution (3.2). Finally, the last stage leverages the GPT-4o to speed up the operational work of describing requirements and generating even more elaborate and contextualized PBIs (3.3).

Lastly, the final phase evaluates and compares the outcomes of each iteration so it's possible to propose improvements in the requirement description structure. The impact of each approach was made based on the developer's feedback, sprint metrics, and retrospective discussions (4.1). The results from these findings were consolidated into a software requirement documentation roadmap, offering a structured set of best practices to guide future development cycles (4.2).

4 REQUIREMENTS METHODS DEFINITION

4.1 Approach Mapping for Requirement Documentation

Since agile principles promote a focus on fast delivery, a common misconception is that documenting requirements is incompatible with agility, when in fact, minimal but sufficient documentation is encouraged. When requirements are misunderstood, agile teams tend to develop misaligned features, and documentation improvement comes as a risk mitigation strategy (Araújo et al., 2019). The high frequency of changes in agile development indicates a lack of requirement stability, which results in inconsistent and outdated documentation. This scenario impacts the development process with rework, communication breakdowns, knowledge loss, poor software quality, and delayed schedules (Canedo et al., 2024).

To address these risks, Table 2 provides a comparative analysis between the requirements elicitation methods evaluated, which have already been presented in Section 2.3, across key performance criteria, focusing on the documentation approach in agile development.

Table 2. Requirements Elicitation Methods Comparison.

Parameters	Design Thinking	Dua-Track Agile	Scrum	Shape-up	USARP	ACUX
Context	Problem framing, innovation, user-centered	Complex, dynamic, startups	Fast iteration, general products	Focused and lean teams needing clear scope	User-facing	Products needing precise user validation
Documentation	Empathy Maps, Problem Statements, User Journey, Prototypes	Opportunity Solution Tree, Acceptance Criteria, Validated Experiments	Product Backlog, User Stories, Definition of Done, Acceptance Criteria	Problem, Solution, Boundaries, Risks, Wireframes	Usability Mechanisms, Prototypes, User Stories	Acceptance Criteria, User Experience Checklist
Timing	Discovery (before development)	Continuous (parallel to development)	Incremental (refinement and planning per sprint)	Incremental (before development iteration)	Continuous	Incremental (refinement and planning)
Detail level	Low	Medium	Often Low	High	Medium	High
Benefit	Promotes deep understanding of user needs and drives innovation	Balance discovery and delivery, supports continuous learning, documentation evolves with the product	Easy to maintain, close to developers, supports continuous change	Provides clear and upfront documentation, prevents backlog to overload, supports focus and autonomy	Capture usability requirements with detailed documentation	Formalize user expectations, acceptance criteria are testable, improve usability quality
Limitation	Does not produce structured requirements	High effort, Lacks formal documentation and long-term goals	Poor for long-term traceability, weak for non-functional requirements	Inflexible during development cycle	Limited to usability scope, requires upfront persona definition	High effort, dense documentation, does not address initial requirements discovery

Created by the author.

The predominant use of User Stories proves insufficient to capture complex requirements (Canedo et al., 2024). Its structure fails to empathize with users due to the vague, repetitive, and superficial writing, which prevents the capture of user behavior and context, resulting in requirements based on assumptions rather than evidence (Hudson, 2013).

The shaping practice derives from the Shape-up, a framework developed by the Basecamp company that seeks to solve common problems faced by software teams, such as delays and undefined scope. Different from Scrum, the team following this methodology has the autonomy to deliver previously modeled features in a time frame of six weeks (Singer, 2019). The Educado team consists of people based in different countries, which means that most of them are not aware of the product's core problem. As a result, it is necessary to implement the Shape-up modeling practice to bring the users' daily lives closer to those who are developing the high-quality solutions.

While incorporating more structured and detailed methods could improve requirement quality, it also increases the operational workload. As a response to this trade-off, artificial intelligence offers assistance to automate the documentation process, enhancing productivity and capturing more precise requirements. However, AI tools have limitations in their effectiveness depending on the model selected for the expected task and how it is used (Habiba et al., 2024). Table 3 provides a comparative analysis between the LLMs evaluated across key performance criteria.

Table 3. LLMs Comparison.

Parameters	ChatGPT-4o	Gemini	Copilot	Claude
Context	Text, code, and image generation	Basic tasks	Code generation and technical tasks	Code generation, technical tasks, and complex system re-engineering
Prompt Sensitivity	Responds well to detailed prompts and produces consistent results	Requires well-crafted prompts to avoid failures	Reliable even with moderate prompts	Handles structured data without requiring highly specific prompts
Efficiency	Achieves satisfactory results with fewer iterations	Requires more attempts to reach optimal solutions	Efficient with few iterations	Efficient code and architectural optimizations with fewer iterations
Text Accuracy	High (89%), produces coherent and contextualized texts	Moderate (85%), exhibits inconsistencies in context and accuracy	Good (87%), reliable for textual tasks	Good (86%), with coherence performance
Error Detection	Detects errors and provides effective solutions	Fails to identify some critical errors	Performs well in detection and correction	Consistently identifies constraints and suggests corrections
Limitation	May produce excessively detailed answers and lack illustrative	Frequent inconsistencies in technical tasks	Suboptimal performance in tasks requiring complex reasoning	May require manual adjustments for complex legacy

Parameters	ChatGPT-4o	Gemini	Copilot	Claude
	examples			systems

Adapted from Noveski et al. (2024).

The comparative analysis offers specific strengths and limitations of different methods and AI models depending on context and documentation needs. It is necessary to address the appropriate selection based on the Educado project, which is outlined in the following section.

4.2 Selection of Methods for Educado's Context

In Educado, the risks presented are amplified by the high team turnover every semester, with six development groups working simultaneously on the same codebase. This dynamic increases the demand for knowledge transfer and alignment through requirements change management and complete documentation. Therefore, a hybrid approach can address different needs for this project by combining methods and tools to leverage their complementary strengths.

Focused on enhancing the documentation on agile methodologies, this work maintains Scrum for managing the backlog and integrates Shape-up for defining the scope deliverables. The adaptation of Scrum maintains the flexibility and incremental delivery, while the shaping practice brings the problem definition and solution sketching before backlog items enter development, resulting in deeper and more precise requirements documentation. Additionally, the documentation process includes user flows, interface design prototypes, user stories, and acceptance criteria. This aims to improve team alignment and mitigate user stories' inherent ambiguity.

However, incorporating shaping practices into the requirements workflow introduces a significant operational workload. The more detailed the documentation becomes, the more time-consuming the preparation is before each sprint. For Educado, ChatGPT-4o by OpenAI was selected due to its ability to generate more consistent and precise outputs through clear, context-aware conversations. This model includes an agent functionality that allows users to define task-specific workflows by training to respond according to predefined documentation standards and domain-specific knowledge (OpenAI, 2024).

5 DEVELOPMENT CYCLE PREPARATION

5.1 Educado Foundational Overview

The Educated concept began in 2018, triggered by the closure of the Estrutural landfill in Brasília, and concentrated on the waste pickers' perspective, aiming to support them through educational initiatives. Since then, the project has reached more than 200 students responsible for the product's evolution in more than 50 interdisciplinary teams from the University of Brasilia and Aalborg University. To make this project robust and complete, the students involved were from different academic levels and diverse fields of study, including Production Engineering, Cybersecurity, Design, and Software.

In the first three years (2018-2020), the problem was identified, and a potential solution was proposed within the EPIC SDG Challenge, which is educational partnerships for innovation in communities. During this phase, a software architecture was developed focused on a personal finance management platform that combines smart caching to optimize content delivery. The gamified system was introduced to be accessible for users, but the only usability test, conducted online, revealed that it was not yet functional. Although it offered basic functionality, it lacked formal user research and was never officially deployed.

After phases without structured user input, 2021 marked the connection with the waste pickers through interviews that consolidated the Educado brand as a response to the identified gaps. The demographic analysis conducted in this phase showed that 60% of participants had not completed high school, 100% had internet access at home or work, and 90% expressed interest in learning about personal finance. These insights informed the creation of a two-part system: a web platform for content creators and a mobile app for learners, prioritizing educational user needs and context.

The following year represented the first development cycle with a research-informed MVP. 21 students from the "Software 5" discipline in the Computer Science department at Aalborg University worked on the software, supported remotely by a Product Owner from the Production Engineering program at the University of Brasília, consolidating the partnership between the universities. Meanwhile, user research advanced focused on the technology use behavior analysis, which revealed that 100% of the waste pickers own a smartphone, 90% prefer video-based content, and their main negative experiences with apps relate to payment requirements, slow performance, and accessibility difficulties. It also included a competitive analysis of existing financial education solutions and the mapping of potential content partners. This version was designed to improve accessibility and was adapted for users with low literacy

levels, incorporating visual and interactive elements, illustrated by Figure 3.

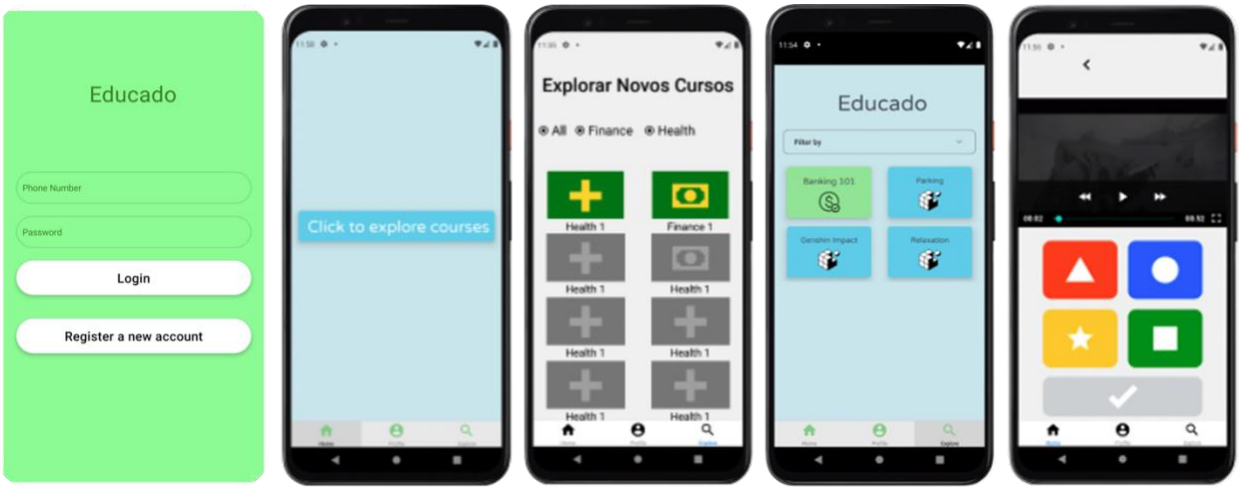


Figure 3. First Educado Mobile App MVP. Created by the author.

Only in the second development cycle in 2023 was the MVP officially handed over for real-world use. A user flow and interface were designed in Figma based on user needs, structured into 13 modules for the app and 6 for the web. This structure was the foundation for requirements and the product backlog, managed using the software Miro as represented in Figure 4 and focused on usability, accessibility, and user motivation.

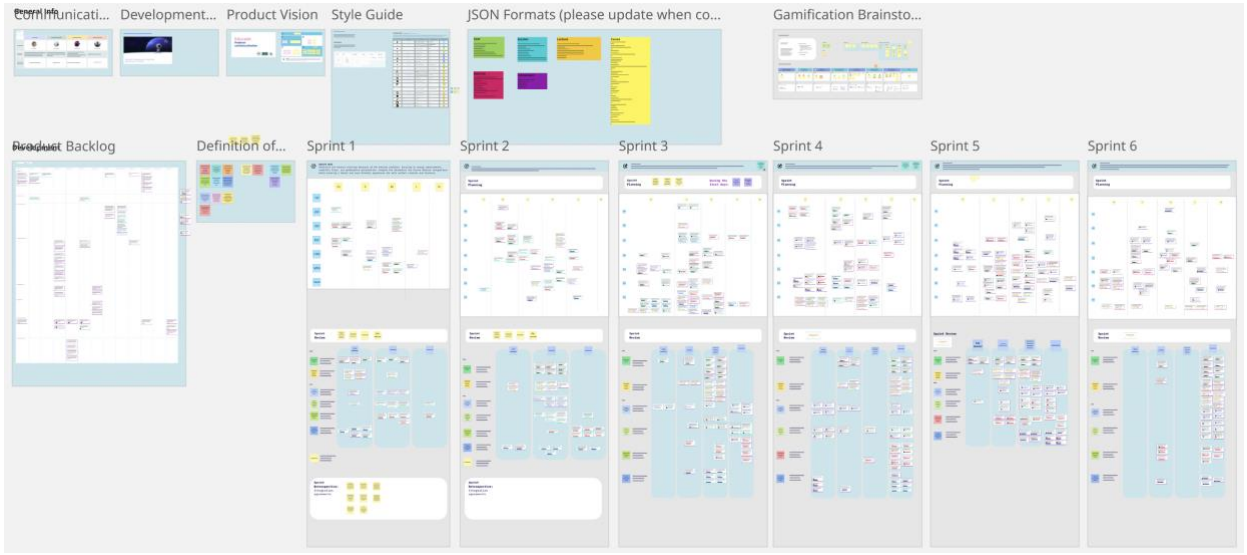


Figure 4. Second Development Cycle Management through the tool Miro. Created by the author.

Conducted within the same software class environment as the first cycle, this phase was led in person by Luiza Cardoso as Product Owner and documented in her graduation project. It also marked a refinement of the development process by a business strategy definition. Although the first sprint faced challenges such as difficulty understanding the existing codebase and fragmented team organization, which reduced productivity and integration issues, this cycle enabled the first real user interaction with both platforms, illustrated by Figure 5.

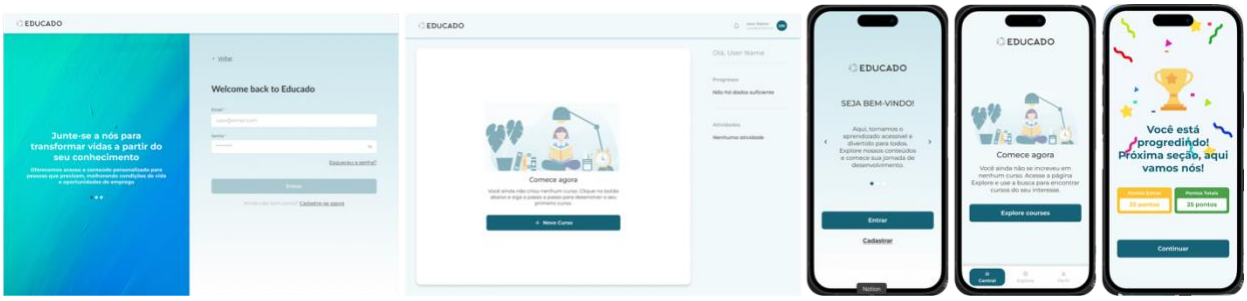


Figure 5. Second Educado Mobile App and Website MVP. Created by the author.

Finally, the third development cycle, case study of this work, is led by the author in the role of Product Owner. Prior to this phase, students from the Production Engineering at the University of Brasília contributed by analyzing the previous development cycle, refining user needs through usability testing and interviews, and defining the requirements for the new iteration.

The study involving waste pickers and content creators applied the think-aloud protocol, post-test surveys, and direct observation, revealing 61 combined errors across the mobile app and website systems. The critical challenges discovered in the platform's functionalities provided support for decision-making throughout the cycle (Sousas et al., 2025).

Additionally, 144 requirements were identified using competitive analysis and performance review, categorized into four groups*:* done*,* improvement*,* bug*,* and new. Then, the prioritization was driven by the Mudge Matrix, ranking them based on impact and urgency to ensure a user-centered development process (Valença et al, 2025).

The main deliverables over the seven years within this historical context are described in the timeline of Table 4.

Table 4. Historical Timeline of the Educado Project.

Timeline	Historical Milestones	Limitations	Teams
2018	- Idealization of the Mobile Education for waste pickers with limited connectivity and lack understanding of finance	- Lacked functional maturity and real-world applicability	1 AAU
2019	- Software architecture for a finance management platform with smart caching	- No user validation conducted	1 AAU
2020	- Introduced gamification software for financial education - Online usability test	- Not functional for the target users	2 AAU 4 UnB
2021	- Demographic analysis based on user interviews - Definition of user personas - Educado brand establishment - Information architecture that supports a duas-platform (web and mobile systems)	- No deployment - Complications on usability, integration, and content availability	9 UnB 2 AAU
2022	- Technological profile analysis based on user interviews - Competitive analysis - Content partner mapping - First development cycle	- Lacked user validation - Developed by distributed teams with limited integrated testing - No deployment	7 UnB 5 AAU
2023	- Business strategy definition - User flow mapping - User interface design - Product backlog creation and management in Miro - Second development cycle - MVP delivery for end users	- Low productivity in the first sprints - Lack of communication between teams - Challenges in tracking tasks	4 UnB 6 AAU

Timeline	Historical Milestones	Limitations	Teams
2024	- Analysis of the second development cycle performance - Usability test and feedback with users - Competitive analysis - Requirement gathering - Requirement prioritization - 3 educational content development - Third development cycle: case study of this work		4 UnB 6 AAU

Created by the author.

Educado's achievements and limitations over the years were the basis for decision-making during the third development cycle. The goal for this phase is to fix the software errors and make the platforms more interesting and easier to use. As in the last cycle, the improvements will be based on a Scrum-based and research-driven approach.

5.2 Development Methodological Approach

The Agile Software Engineering module of the fifth semester of the Software Engineering program at the Copenhagen Campus of Aalborg University prepares professionals to select appropriate theories, methods, and techniques for agile large-scale projects. It includes processes such as code quality, continuous improvement, project management, modeling, and requirements gathering. To achieve this, the module utilizes the Scrum framework to apply these concepts in practice within Educado software development.

The Scrum guides teams in delivering incremental value, which emphasizes quick adaptation that allows for continuous learning depending on the experience. It promotes collaboration and lean thinking, eliminating activities that do not add value to the final product. The roles of this framework are Product Owner, who is responsible for the product decision-making and managing the backlog; Developers, who build the software incrementally; and Scrum Master, who ensures the framework execution (Schwaber & Sutherland, 2020).

At the beginning of the academic semester, the class agreed upon a set of criteria each increment needs to meet, defined as the Definition of Done. These increments are established by the requirements that compose the Product Backlog, a comprehensive list of all the tasks, or Product Backlog Items (PBI), needed to build the product, followed by the Product Goal. Each release is completed in short intervals called sprints.

The sprint starts with Sprint Planning, where a subset of prioritized tasks is selected from the backlog to be delivered on the iteration based on the Sprint Goal. The items selected from the Sprint Backlog are sized, prioritized, and reviewed in the Refinement meeting once a week. During the sprint, the team syncs on progress and identifies blockers through the Daily Scrum meetings. At the end, the Sprint Review provides an opportunity to showcase the increment and gather feedback, adapting the backlog accordingly. Also, Sprint Retrospective allows the team to

recognize their performance and learn how to improve for the next sprint (Sutherland, 2014).

These ceremonies and artifacts are detailed in Figure 6.



Figure 6. Scrum Lifecycle. Reprinted from Sergeev (2020).

This structure provides a flexible project management that adapts according to observation and experience, enabling the preparation of action plans to address identified gaps.

5.3 Backlog Setup

A software product's progress is guided through backlog management, a requirements repository that captures and organizes the product vision. The Product Owner is responsible for creating, maintaining, and prioritizing these requirements, which can include new features, bug fixes, and technical improvements. (Alhazmi & Huang, 2020). In the case of Educado, these types and their prioritization are outlined as part of the 2024 deliverables, detailed in Section 5.2.

Typically, the backlog items are refined hierarchically, starting with strategic and broad concepts and breaking them down into smaller, more specific concepts related to development progress (Ferreira et al., 2022). Illustrated in Figure 7, Educado backlog management follows the same top-down logic to progressively refine the understanding of the product and make the work more manageable for each sprint. The top level represents Systems, which separate the Website and Mobile App systems. Each system is divided into Features, capturing the platform's primary resources. These features are broken down into Capabilities, actionable functions that define what each feature should deliver to support specific user needs. The capabilities are the PBIs of the Educado backlog, and each one is gradually detailed with Documentation, level discussed in this

work, and with Acceptance Criteria, tasks defined that should be completed for release.

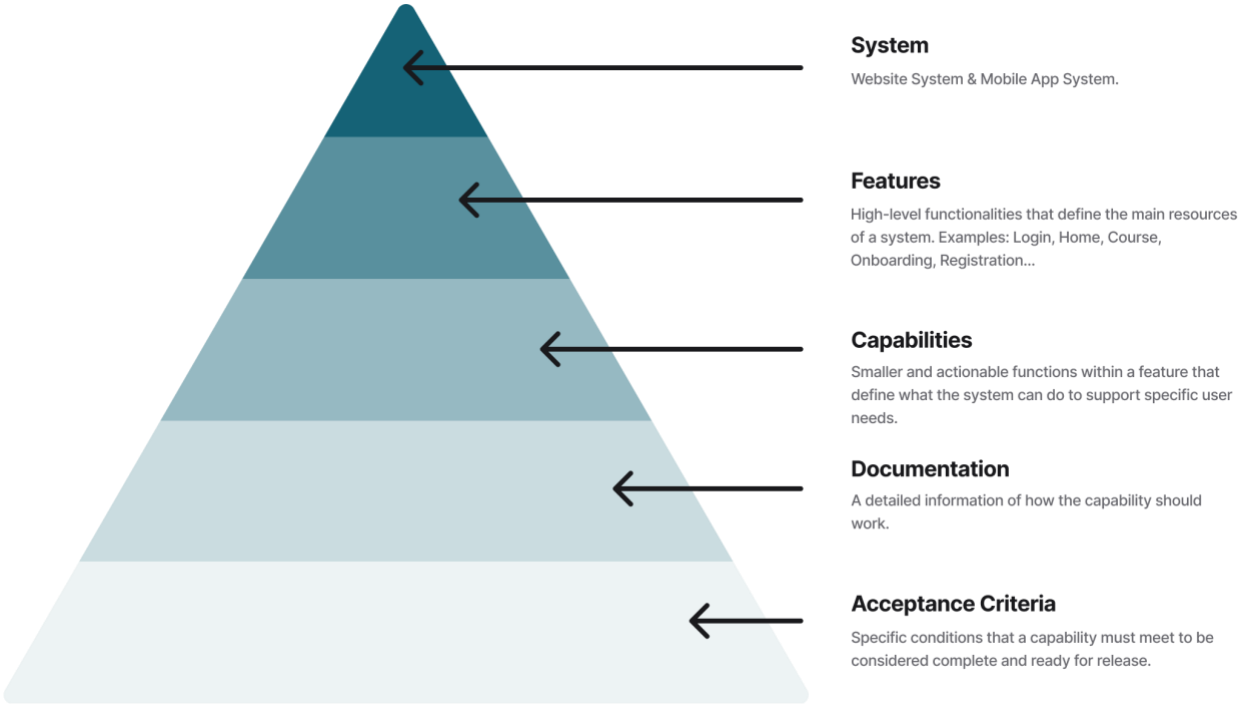


Figure 7. Educado Backlog Structure. Created by the author.

For the current development cycle, the backlog was migrated from Miro, illustrated in Figure 4 in Section 5.2, to GitHub Projects. This decision was made with the intention of increasing developer adherence, which leads to knowledge management on a long-term basis. GitHub is already widely used by developers as a code repository, reducing the need to switch between platforms that could create resistance in maintaining the tool. Unlike Miro, which is more generic and designed for brainstorming, GitHub Projects is specifically built for tracking development progress with items directly linked to repositories, making it easier to verify commits and code changes (GitHub, 2023).

All the requirements mapped in 2024 were transferred to GitHub Projects, created specifically to reflect the Educado sprint process, set to a 2-week iteration. The backlog organization was structured with filters that enabled targeted views by group, system, and current sprint, allowing each group to focus on relevant tasks while maintaining visibility over the full project scope. As illustrated by Figure 8, the board has the following columns to trace progress:

- **Product Backlog:** an ordered list of all the requirements needed for the product, including new features, improvements, and bug fixes;
- **Selected:** most refined and prioritized items selected for the upcoming sprint during Sprint Planning, and established as the Sprint Backlog;
- **In Progress:** items currently being worked on by the development team;
- **In Review:** items under code review
- **Validation:** items that are in the final verification stage, usually by the PO and

stakeholders, ensuring they meet the acceptance criteria

- **Done:** completed items that have been delivered and are ready for use

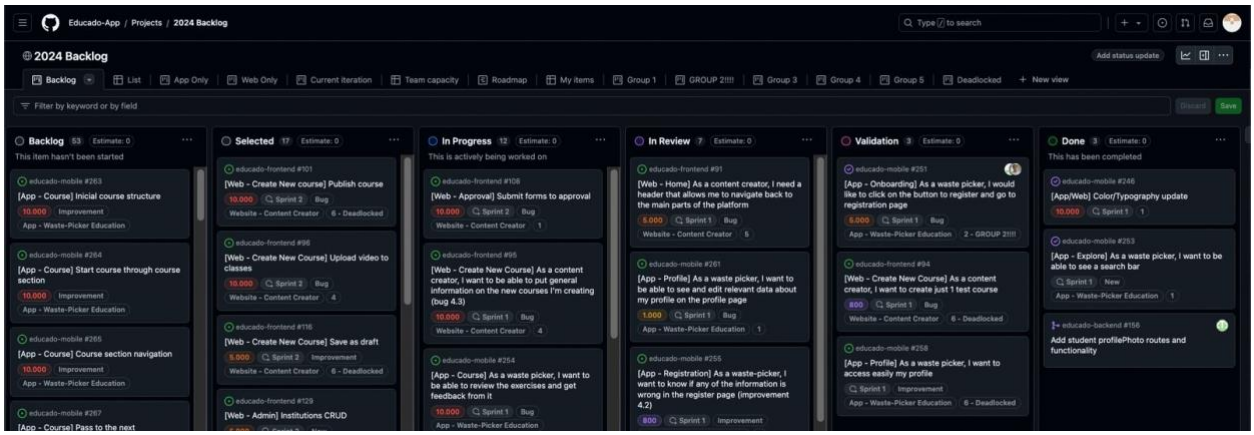


Figure 8. Backlog Setup within GitHub Projects. Created by the author.
https://github.com/orgs/ErasmusEgalitarian/projects/14/views/1

Each item is identified consistently following the structure represented in Figure 9: "*[System – Feature] Capability"*, for example, "[Web – Create New Course] Upload Video to Classes". Inside the item, the team can find the documentation and acceptance criteria to understand what needs to be implemented. The item card also includes tags for estimate, category (new, improvement, or bug), priority, group assigned, and sprint indicator, as illustrated in the example by Figure 9.

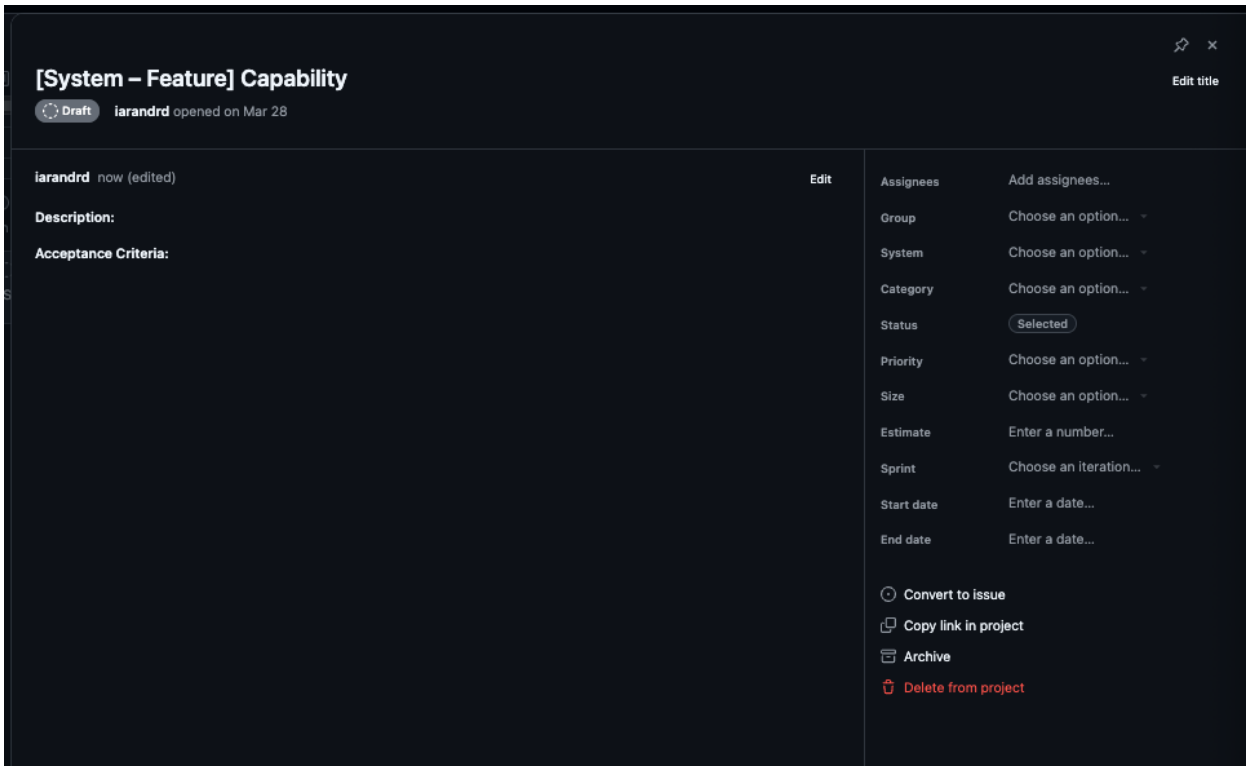


Figure 9. PBI Example within GitHub Projects. Created by the author.
https://github.com/orgs/ErasmusEgalitarian/projects/14/views/1?pane=issue&itemId=104076979

The Product Backlog needs to be constantly reviewed and refined to ensure items are ready for development and that the backlog aligns with the long-term goals (Ferreira et al., 2022). Based

on that, requirements were added, deleted, and updated over the sprints to provide focus and direction while ensuring they are clear and complete, with the Product Owner overseeing it. However, without proper requirement documentation, the project risks delay progress and reduce a reference base for future cycles. The following section will discuss this topic further, as the evolution of requirement documentation is a primary focus of this study.

6 DEVELOPMENT CYCLE EXECUTION

For this development cycle, the Educado project team consists of 36 students as developers, divided into 6 multidisciplinary groups. It removed the thematic separation to promote collaboration and a complete view of the project, with all teams responsible for working on the same feature, as lessons learned from the previous cycle. Additionally, early sprints often lacked progress due to the effort to understand the codebase without a clear goal. To address this, the current cycle started with the correction of low-complexity bugs and improvement items, providing a hands-on opportunity to explore the code from the beginning.

The items for each sprint were selected by the PO based on previous prioritization, explained in Section 5.1, and were sized by the team during the refinement meeting using T-shirt Sizes (XL, L, M, XS, S). At the planning meeting, the distribution of items to development groups followed an affinity-based process, where the PO presented all selected items, and the teams chose them based on interest, expertise, or alignment with their previous work. To ensure balanced workload, allocation rules were established: groups assigned to an XL item would work exclusively on it; those with an L could pair it with an XS; and other combinations, such as M + S + XS, were also allowed.

This chapter will highlight the progress of the development cycle through the phases: (i) PBIs delivered with low description using user stories only; (ii) Integration of shaping practices for more detailed requirements; and (iii) Use of AI to generate even more elaborate and contextualized PBIs with less time-consuming. For this analysis, three of the six sprints conducted were selected to exemplify the progress based on their relevance in capturing the transitions between the phases. The remaining sprints were discarded, as one focused on deployment to the staging environment, another contained duplicated information from previous sprints, and the last lacked PO supervision, compromising the reliability of the collected data.

6.1 Initial Requirement Description Approach with User Stories

In the first sprint, 19 items were selected by the PO for higher importance and lower relative complexity. The requirement documentation method adopted at the beginning of the development cycle was the standard User Stories from Scrum, with a short, complementary description limited to the focused action and low contextual information, as exemplified in Figure 10. For items related to the user interface, a Figma link was attached, which serves as the primary visual source for understanding the features. No formal specification model was applied to ensure consistency across items. Instead, the structure followed a basic template consisting of: “As a [user], I want [goal] so that [reason] + Figma link + Acceptance Criteria”

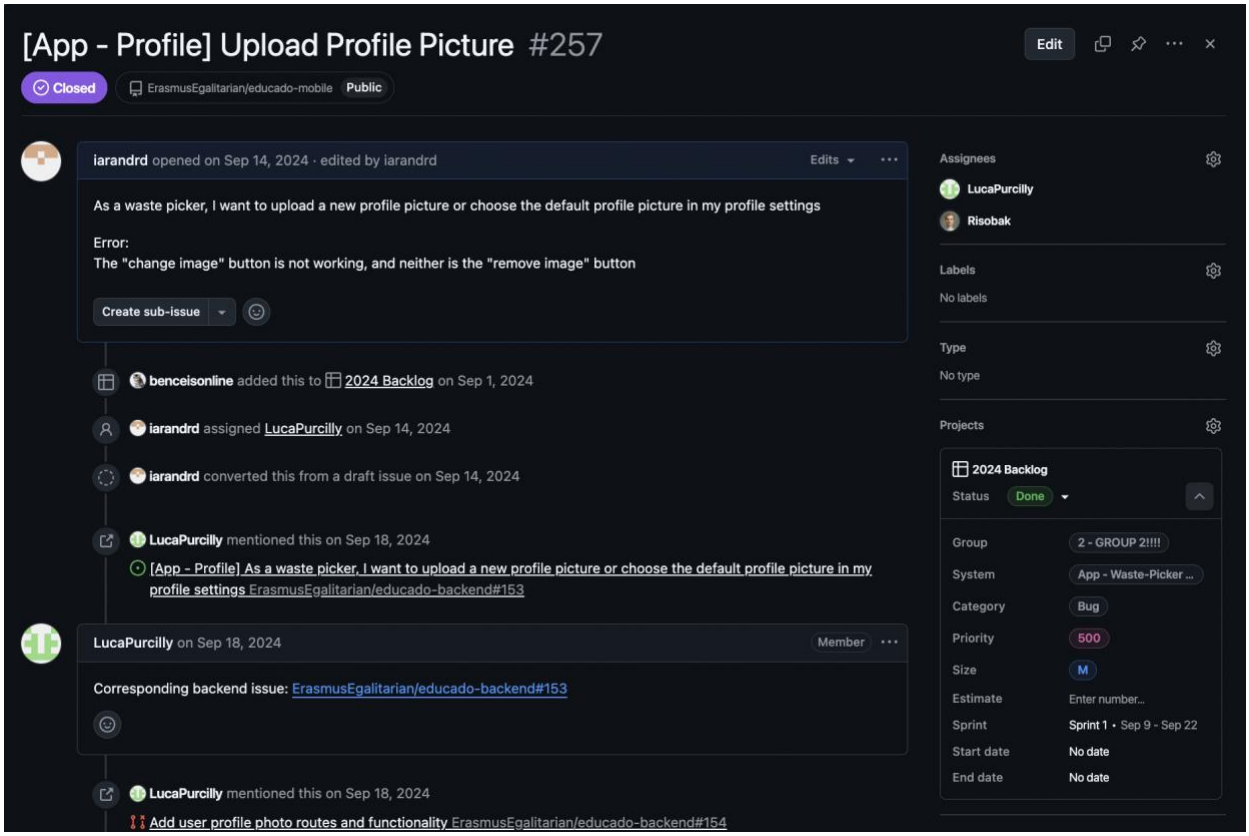


Figure 10. PBI Documentation of Sprint 1. Created by the author.
https://github.com/orgs/ErasmusEgalitarian/projects/14/views/1?pane=issue&itemId=78204160&issue=ErasmusEgalitarian%7Ceducado-mobile%7C257

During this sprint, four of the six groups contacted the PO with numerous questions to clarify what should be delivered, which revealed a significant lack of project and user context. The information provided to the development teams proved insufficient for a proper understanding of the requirements. At the review, these limitations became even more evident, where the following issues were observed:

- One group presented a functionality that was entirely different from what the requirement intended;
- There were no visual indicators for destructive actions, such as confirmation dialogs or error handling mechanisms;
- Some groups prioritized lower-priority items.

The graphs displayed in Figure 11 represents the Sprint 1 results, detailing the items according to their final development status as follows: *Not Started*, items that were prioritized for the sprint, but no development work was initiated; *In Progress*, items that were partially developed but not completed within the sprint timeframe; *To Rework*, items that were delivered but contained issues or did not meet expectations, requiring rework; and *Done*, items that were fully implemented and validated, meeting the expected acceptance criteria.

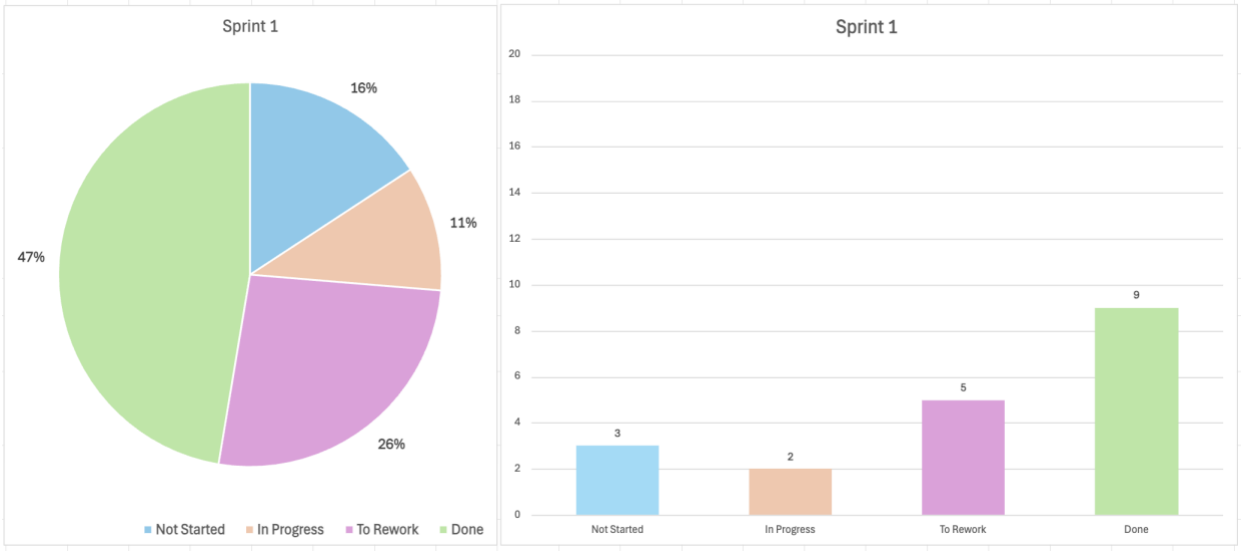


Figure 11. Sprint 1 Outcomes. Created by the author.

The results present that a significant portion of the selected items (53%) were not completed, either due to the need for rework, lack of initiation, or remaining in progress. In contrast, only 47% of the items were successfully delivered by the end of the sprint. Considering the situations, the main operational issues identified that contributed to these results include ambiguity in the written requirements, high dependence on synchronous communication with the PO, and misinterpretations caused by insufficient contextual information.

Based on these findings, the hypothesis is that adopting a more robust and standardized structure for requirement documentation could improve delivery quality. Additionally, enhancing communication channels between the PO and development teams is essential to ensure early validation and clarification, which minimizes rework and increases understanding of the developers' needs.

6.2 Adapting User Stories Requirements with Shape-Up Principles

The observed results highlight the need for strategic responses, given the hypotheses formulated to mitigate the problems identified. In line with this, the second sprint included 26 selected items, 10 of which were carried over from the first sprint with incomplete status, and 16 new items were introduced. To support these adjustments and based on the methodological criteria established in Chapter 4, the Shape-Up method was partially incorporated into the workflow.

While Scrum continued to guide the execution of tasks during the sprint, Shape-Up principles were introduced during the preparation phase, with a focus on defining the work before building. These shaping practices address the common shortcomings in agile methods regarding insufficiently detailed requirements of user stories, which can lead to misinterpretation, as noted in Section 6.1, with 53% of the selected items not delivered. The concept behind these practices

is to explain the underlying problem, helping the team to fully understand the rationale behind each item. Each of the following components was incorporated into a new documentation template for backlog items:

- **Problem Statement:** Brief yet precise explanation of the issue affecting the user or system that the feature is intended to solve, establishing context by outlining the pain point, limitation, or inefficiency;
- **Goal:** Articulation of the expected outcome from both the user's and the product's perspective, and defines what success looks like in terms of value delivered;
- **Scope:** Breakdown of the boundaries, indicating what is included and excluded in the delivery of the feature, which can involve functionalities, constraints, interface elements, and relevant metrics;
- **Risks and Unknowns:** List of potential uncertainties that could affect the implementation of the feature, such as technical limitations, integration challenges, usability concerns, or gaps in available information;
- **Success Criteria:** Set of conditions that must be fulfilled for the feature to be considered successfully delivered and should be verifiable through testing or user validation;
- **Solution Idea:** High-level conceptual proposal for how the identified problem could be addressed, including UI Designs, user flow diagrams, or a textual overview that outlines the logic envisioned.

Given the outputs of the first sprint, the documentation will always include error controls and confirmation notifications as part of the success criteria when dealing with destructive actions mapped in the risks. To adapt these fields to the process, the PO began preparing backlog items prior to each sprint using the structure “User Story + Shaping Method + Figma link + Acceptance Criteria”, as illustrated in Figure 12.

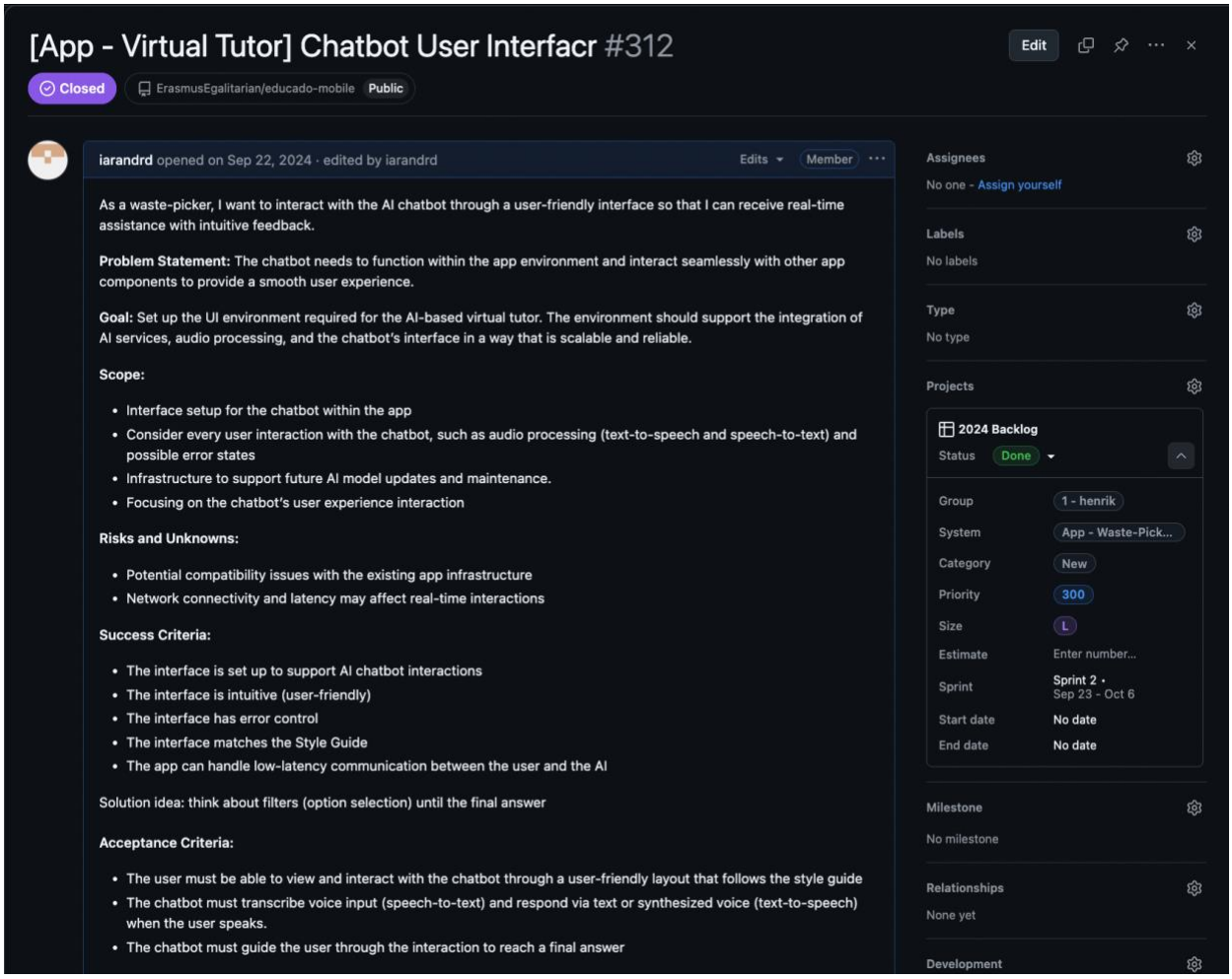


Figure 12. PBI Documentation of Sprint 2. Created by the author.
https://github.com/orgs/ErasmusEgalitarian/projects/14/views/1?pane=issue&itemId=80650947&issue=ErasmusEgalitarian%7Ceducado-mobile%7C312

This structure was introduced to the developers during the refinement session, aiming to promote continuous communication and increase their confidence in understanding the work to be done. As a result, discussions during sprint planning were more focused, and the team expressed fewer uncertainties about the requirements. At the review of the sprint, it was possible to note the impact of the documentation revision, as highlighted by the graph in Figure 13.

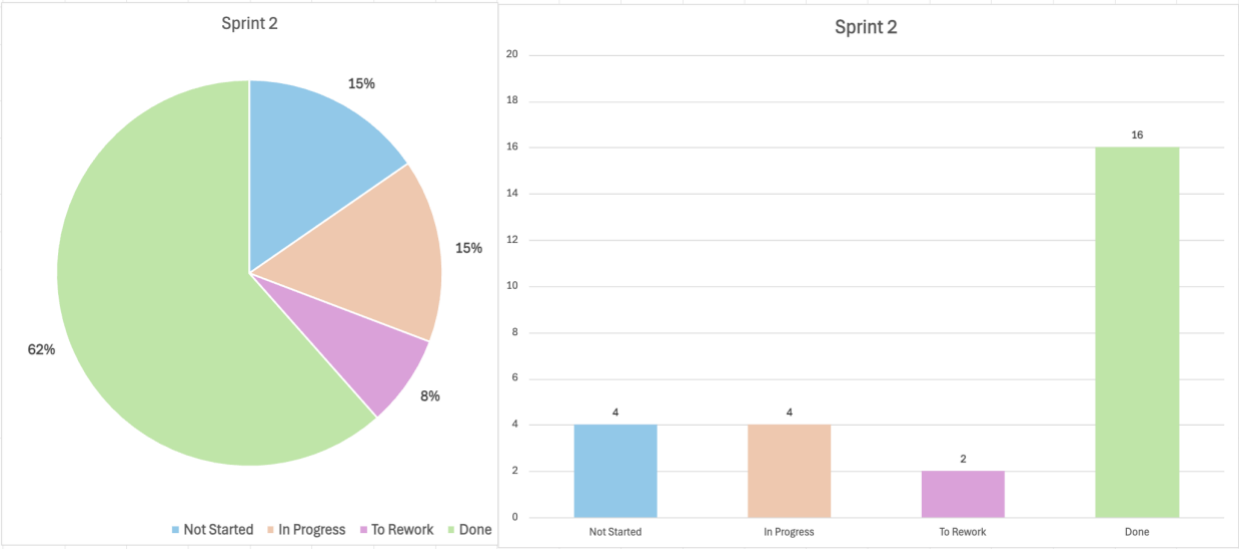


Figure 13. Sprint 2 Outcomes. Created by the author.

Most of the items were completed successfully (62%), while only 8% required rework. Additionally, 30% was not completed by the end of the sprint, remaining unstarted or still in progress. The increase in completed tasks and reduction in rework suggest an enhanced understanding during implementation, even with the growth in the number of work items for the sprint from 19 to 26 (36,84%). However, the operational work on each item in detail remains a time-consuming task for the PO. This limitation revealed the need for a scalable and consistent support mechanism.

6.3 AI-Enhanced Requirement Descriptions

Given the limitations observed when applying a more structured requirement documentation in Section 6.2, there is an opportunity to utilize artificial intelligence as a potential solution to automate operational workflows in the development process. To test this hypothesis, a total of 21 items were selected for this sprint, 10 from the previous sprint and 11 newly prioritized from the product backlog. This initiative aimed to improve the assertiveness and consistency of requirement descriptions by supporting a conversational AI tool.

The model ChatGPT-4o is adopted due to its adaptability to custom prompts that address specific project contexts, suitable for requirement documentation generation, as demonstrated in Chapter 4. Building on this capability, a dedicated AI agent was created specific to the Educado context to interpret inputs with basic user stories and, when available, interface design images to transform them into complete backlog item descriptions aligned with the structure defined in Section 6.2. The prompt used to configure this custom agent, along with its guiding principles, is presented in Appendix I.

This instruction was tested by providing example inputs and adjusting the behavior until

the model responded according to the expected pattern. Additionally, reference materials were uploaded during its configuration, including the current product backlog, interface designs, and complementary documents that describe the Educado project's objectives, users, and implementation strategy. Besides that, the Shape-Up book by Basecamp was provided to align the outputs with the logic and principles outlined in Section 6.2.

The interaction between the PO and the agent was conducted through iterative prompt engineering, where the received output content is reviewed, edited, and validated to align with the project context. This loop validation maintains consistency and quality before publishing the PBI in the GitHub backlog. Notably, developers did not interact directly with the agent and worked exclusively with the curated versions of the requirements. A practical example of the input provided in the chat and the output generated can be found in Appendix II, while Figure 14 presents the final version of the corresponding backlog item as it appeared in the backlog.

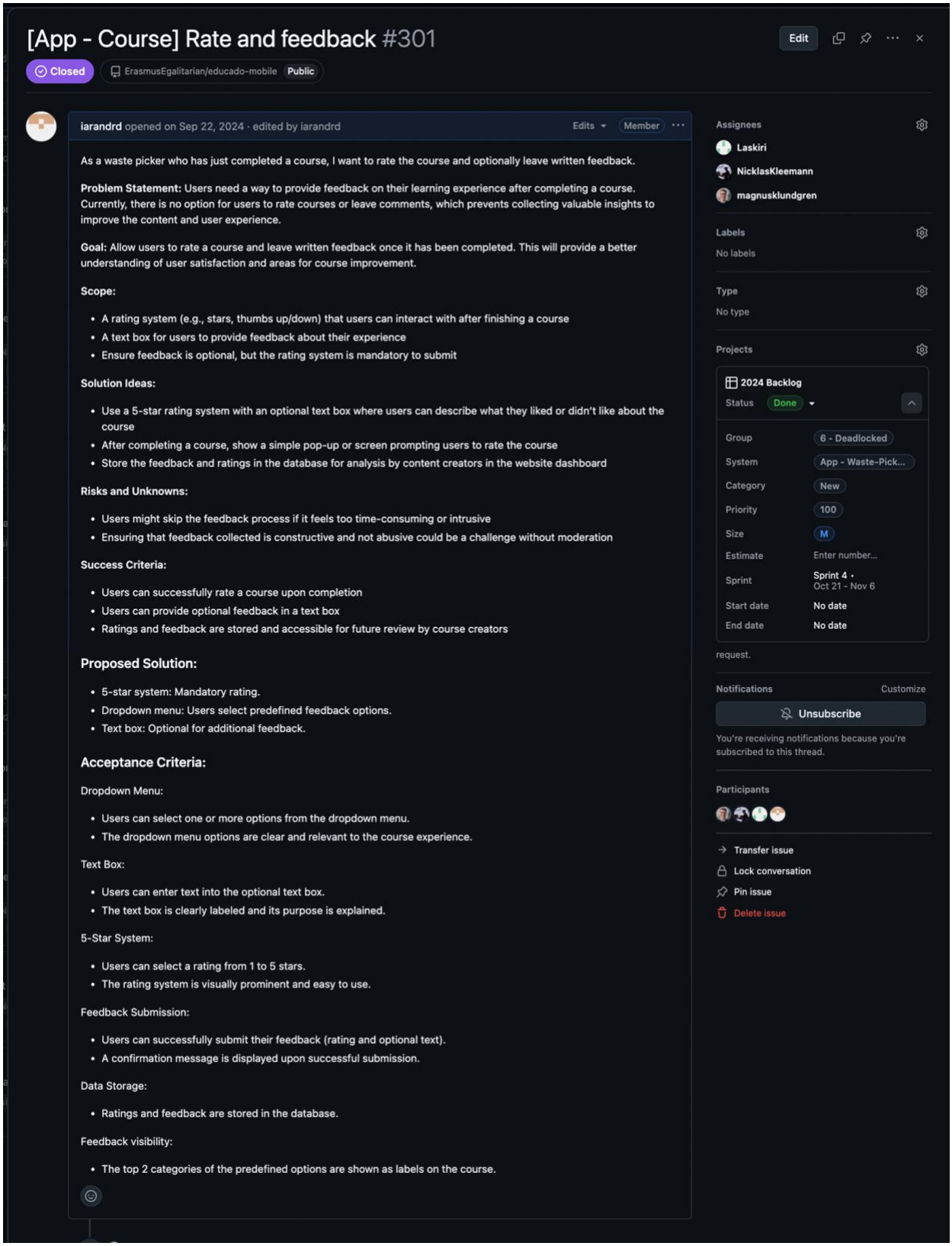


Figure 14. PBI Documentation of Sprint 3. Created by the author. https://github.com/orgs/ErasmusEgalitarian/projects/14/views/1?pane=issue&itemId=80649878&issue=ErasmusEgalitarian%7Ceducado-mobile%7C301

The average time estimated to complete each backlog item dropped from approximately 1 hour and 30 minutes to just 15 minutes, an effort reduction of about 83% in requirement documentation. Beyond the efficiency gains, the chat not only processes what each item requires

with precision, incorporating error handling, but also why it matters from a user perspective, reinforcing the project's importance. The distribution of sprint results reflecting this improvement is presented in Figure 15.

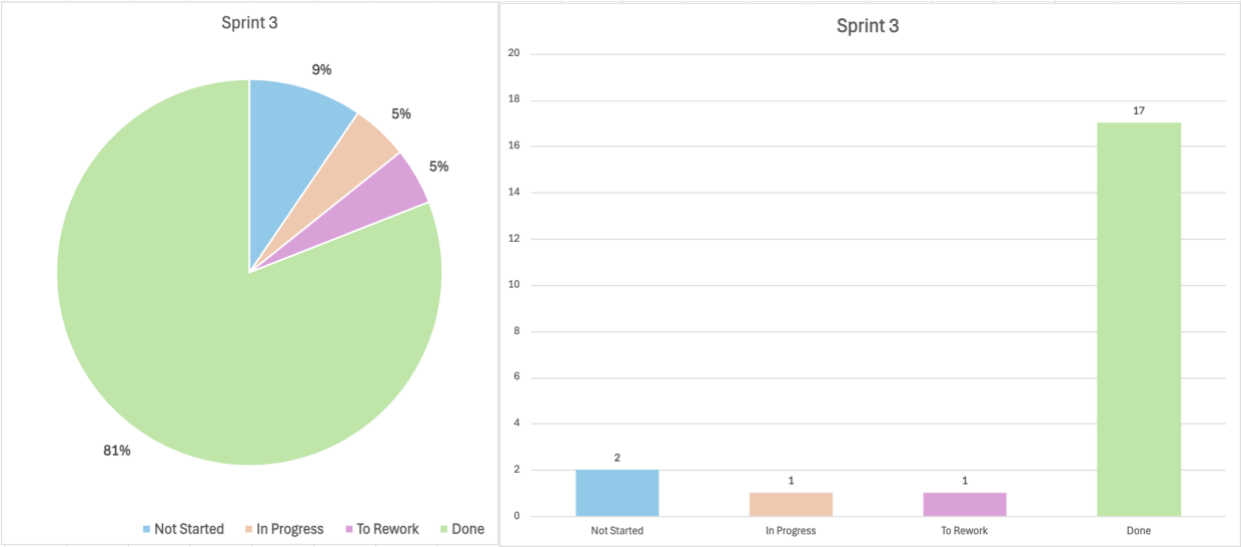


Figure 15. Sprint 3 Outcomes. Created by the author.

The results indicate a continued improvement in delivery performance following the integration of refined documentation practices and AI support. Out of 21 backlog items selected for this sprint, 81% (17 items) were completed successfully. Only 1 item (5%) required rework, while 3 items (14%) remained in progress or not started. Compared to previous sprints, Sprint 3 shows a notable reduction in incomplete or problematic items.

It is important to note that artificial intelligence serves as a support tool in the refinement process, while it still needs human oversight to ensure accuracy and contextual relevance. It optimizes time to prepare the selected items for the sprint, increases standardization, reduces ambiguity, and identifies critical aspects that might otherwise go unnoticed. These advantages are accompanied by a necessity of critical review, clear guidelines for ethical and responsible application, and workflow maintenance.

7 RESULTS ANALYSIS

After the implementation of the three strategies for requirement documentation through the Educado software development, it was possible to notice an incremental output that impacted the team's performance, although improvements are naturally expected as the team gains more knowledge about the project and its context through a learning curve. The first phase began with basic user stories, commonly adopted by Scrum teams, followed by the adoption of Shape-up documentation principles in the second phase, and later enhanced with an AI agent in the third phase. Therefore, in this chapter, the results will be evaluated according to delivery, quality, and team perception. Based on the conclusions, a roadmap will be presented that can be easily replicated by other projects following a similar context.

7.1 Iteration Results Evaluation Comparison

To understand the impact of each documentation phase on the development team's outcomes, a comparative analysis was conducted across the three sprints, which consisted of 19 items in Sprint 1, 26 items in Sprint 2, and 21 items in Sprint 3, as detailed in Chapter 6. Figure 16 presents a bar chart comparing the performance of the development team across the three documentation phases in terms of backlog item outcomes at the end of each sprint.

The vertical axis shows the absolute number of items, while the horizontal axis indicates the status assigned to them as a result. In the Not Started category, Sprint 1, 2, and 3, represented by light blue, medium blue, and dark blue bars, respectively, had 3, 4, and 2 items. For In Progress, the number of items went up from 2 to 4 between Sprint 1 and 2, then declined to 1 in Sprint 3. In the To Rework category, a consistent decrease was observed, with 5, 2, and 1, respectively. Finally, the Done category shows a steady increase in completed items with 9, 16, and 17 across the three sprints.

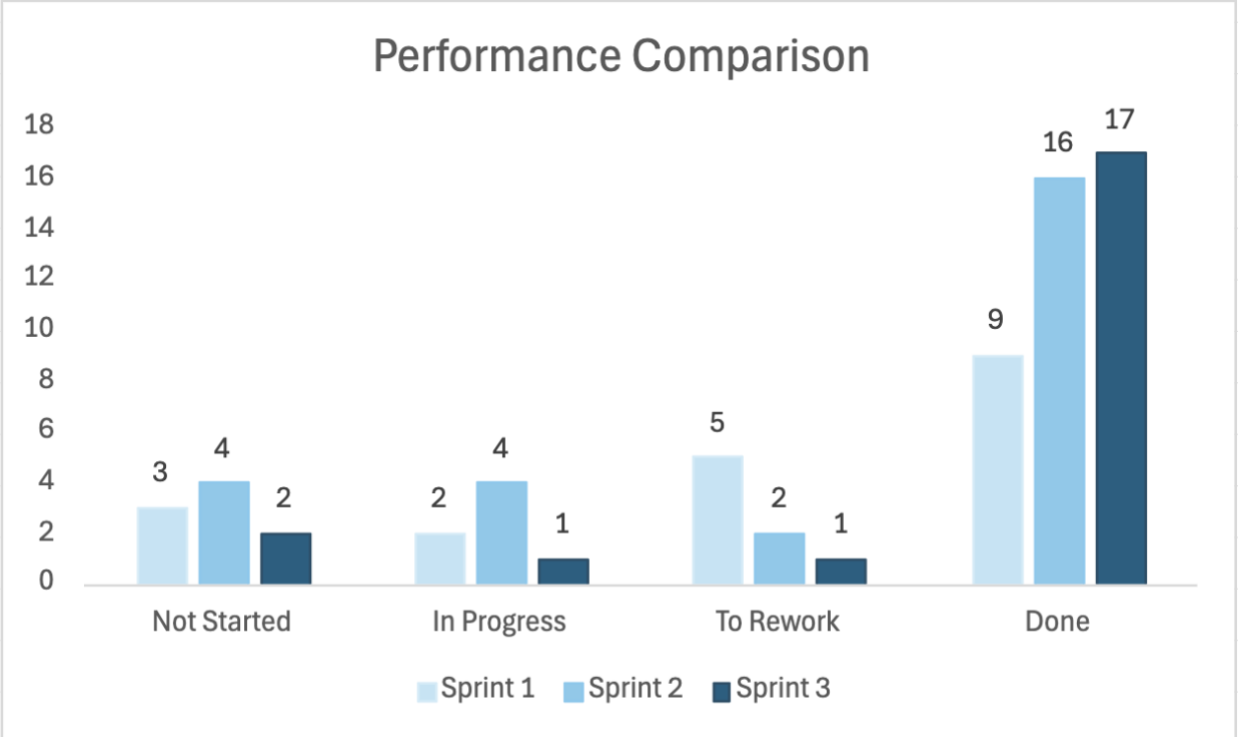


Figure 16. Performance Comparison Through the Phases. Created by the author.

The indicators Not Started and In Progress reflect more planning or prioritization issues rather than the effectiveness of the documentation. An item may not be started or may be in progress due to reasons related to a lack of time, external dependencies, technical complexity, or strategic planning.

On the other hand, the number of completed items indicates the level of productivity achieved, that is, how much the team was able to deliver complete, validated, and ready-to-use features. A consistent increase in the number of items delivered, as observed, reflects predictability and autonomy, which are among the main objectives of well-structured documentation. Following that, the rework rate is associated with the quality of documentation, since the more ambiguous or vague the description of requirements is, the greater the chance of divergent deliveries that need to be redone. The visible drop in rework indicates improvement in the clarity of PBIs and alignment between the PO and the team.

Both indicators offer a more direct view of the relationship between documentation quality and team productivity, making them the main metrics for assessing the impact of writing practices applied throughout the project phases. Therefore, Figure 17 illustrates the relative variation in the number of items marked as Done and To Rework is evaluated, taking Sprint 1 documentation as the baseline (0%), since there are no new implemented methods to bias the analysis. The chart presents the green line for completed items and the pink line to track items to be reworked across the three sprints observed. In Sprint 2, the number of completed items increased by 77.78% compared to Sprint 1 and continued to rise by 88.89% in Sprint 3 compared to the reference sprint once again. Conversely, the number of To Rework items showed a significant decline of -60% in

Sprint 2 and -80% in Sprint 3, relative to the initial values in Sprint 1. It is important to note that the methods are complementary.

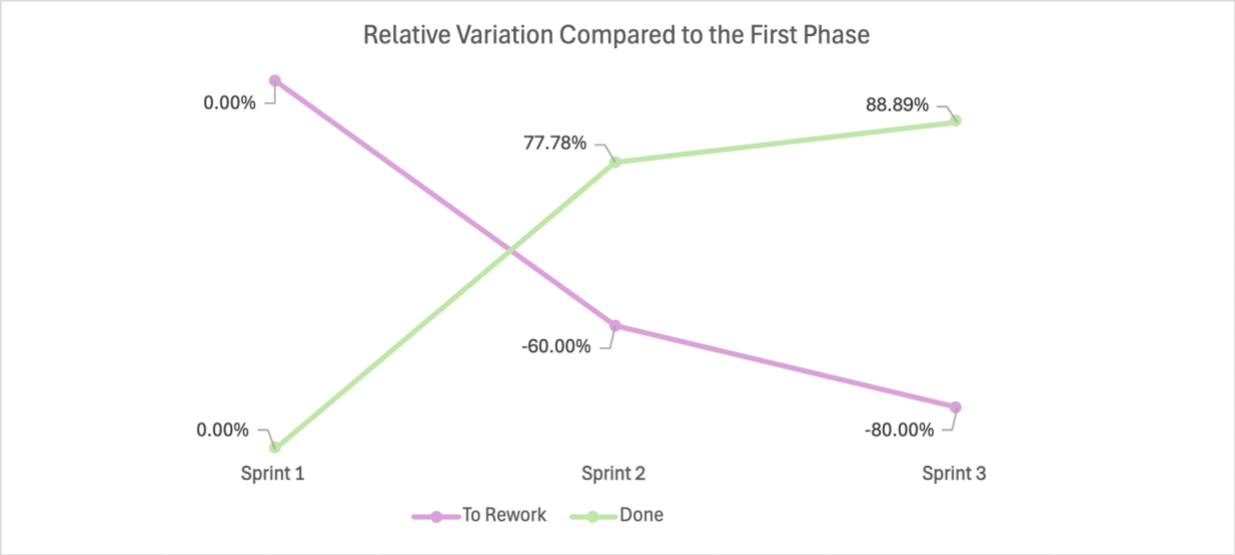


Figure 17. Relative Variation Compared to the First Phase. Created by the author.

It is possible to note the high impact on both indicators between sprints 1 and 2 when shaping practices were included in the requirements. However, the performance gain between sprints 2 and 3 was comparatively smaller, since both sprints followed the same model proposed in Section 6.2. This suggests that the use of AI without context and strategy has less value for team productivity than the establishment of documentation principles.

However, the PO's work can be enhanced when supported by an innovative tool, as indicated in Section 6.3, where PBI writing productivity increased by 83.33%, a reduction from 1 hour and 30 minutes to 15 minutes. In other words, the AI has a more evident influence on the operational efficiency of requirements management work.

To complement the findings presented through sprint performance metrics, the developers' perspective was collected through open-ended questions based on a semi-structured script, available in Appendix III. At the beginning of the development cycle, some developers pointed out that vague or incomplete requirement descriptions made it difficult to get started. However, progress throughout the semester was not driven by documentation alone, since several comments highlighted that regular refinement sessions and open dialogue with the PO played a central role in making the tasks more approachable. One developer noted: "*It was confusing at the start of the semester, but became gradually better as we had refinement meetings to discuss and shape the items to be easy and logical to handle*.".

Although all developers reported understanding the value behind each item, this did not always translate into motivation. This variation suggests that its user impact needs to be meaningful and relatable, not just formally documented, which can be achieved through more user testing. The improvements observed in later stages were the result of more than just better written

requirements, reflecting a learning curve driven by the team's engagement throughout the semester.

To summarize, Table 5 presents a cross-review of developer perception and sprint outcomes throughout the documentation phases, where the rework and performance evaluation are relative to the total number of items in each sprint.

Table 5. Documentation Phases Comparison.

Parameters	Basic Documentation	Shape-up Integration	AI-Enhanced
Structure	User Story + Acceptance Criteria	User Story + Shaping Practices + Acceptance Criteria	User Story + Shaping Practices + Acceptance Criteria
PBI Volume	19 selected items	26 selected items	21 selected items
Detail Level	Low, vague descriptions, simple user stories, limited context	High, structured items following a pattern	High, more complete and contextualized
Perceived Clarity	High dependence on the PO and refinement meetings	Fewer doubts, autonomy, and better understanding	Fewer doubts, autonomy, and better understanding
Rework	High (26%)	Low (8%)	Low (5%)
Performance	Low (47%)	Medium (62%)	High (81%)
Writing Effort	-	1h30 per item	15min per item (-83%)
Standardization	Low, description varies by author	Medium, improved based on proposed model	High, consistent tone and structure
Error Handling	Often missing	Occasionally included	Explicitly included

Created by the author.

These findings confirm the progressive improvement in requirement documentation quality, which has a positive impact on delivery efficiency. The results also revealed that good communication is crucial as the software development process itself. When developers operate in isolated workflows, there is a higher risk of delivering generic or disconnected solutions that fail to meet user needs. Therefore, team members should be encouraged to engage with the project's mission and user impact to ensure that the solutions developed are purposeful and contextualized.

This interpretation is consistent with broader research on agile team effectiveness, which emphasizes that shared understanding, communication, and alignment with team purpose are factors for meaningful delivery (Verwijs & Russo, 2022). In this sense, a structured roadmap is necessary to guide and consolidate the improvements observed in requirement documentation practices.

7.2 Software Requirement Documentation Roadmap

When agile projects face team turnover and cultural diversity, the complexity of coordination increases, and a replicable method for preparing backlog items is necessary to maintain understanding, alignment, and consistent delivery, as shown in the Educado project. To address this, a standardized roadmap was created to support an incremental and iterative requirement engineering process within a Scrum framework, aligning with the results analyzed in

Section 7.1.

Table 6 outlines the complete step-by-step process, from selecting a product backlog item to publishing into the project management board, integrating the Shape-up principles into the documentation, and the use of AI to generate detailed requirement descriptions.

Table 6. Documentation Roadmap for Agile Backlog Items.

Step	Purpose	Action	Responsible
1. Backlog Item Selection	Define the items to be shaped for the upcoming sprint	Select the backlog item based on prioritization criteria and sprint goal to enter the sprint backlog	PO
2. Gather Context and Inputs	Collect all required information and context for shaping	Gather user story, interface design, constraints, expected impact, and relevant technical or process context	PO + Stakeholders + Developers
3. AI Support	Generate the first draft of structured documentation	Use AI agent to generate the requirement documentation based on a template	PO
4. Review	Edit, format, and finalize the content	Review the output, revise inconsistencies, ensure completeness, and align with project standards	PO
5. Refine collaboratively	Eliminate ambiguity and promote shared understanding	Refinement sessions with developers before the sprint	PO + Developers
6. Publish to backlog	Mark the item as ready to be developed	Upload to the product backlog with effort and priority	PO

Created by the author.

Complementary, Figure 18 illustrates the flowchart for software requirement documentation, together with the AI agent prompt and the outcome template. The process begins with the selection of the backlog item for the next sprint and gathering all necessary contextual information, including user stories, interface designs, and user needs. Once these inputs are collected, the AI agent, configured with a custom prompt as detailed in Section 6.3, generates the first draft of the documentation based on predefined fields inspired by the Shape-up methodology. The output undergoes manual review and collaborative refinement with the tech team before being published to the sprint backlog.

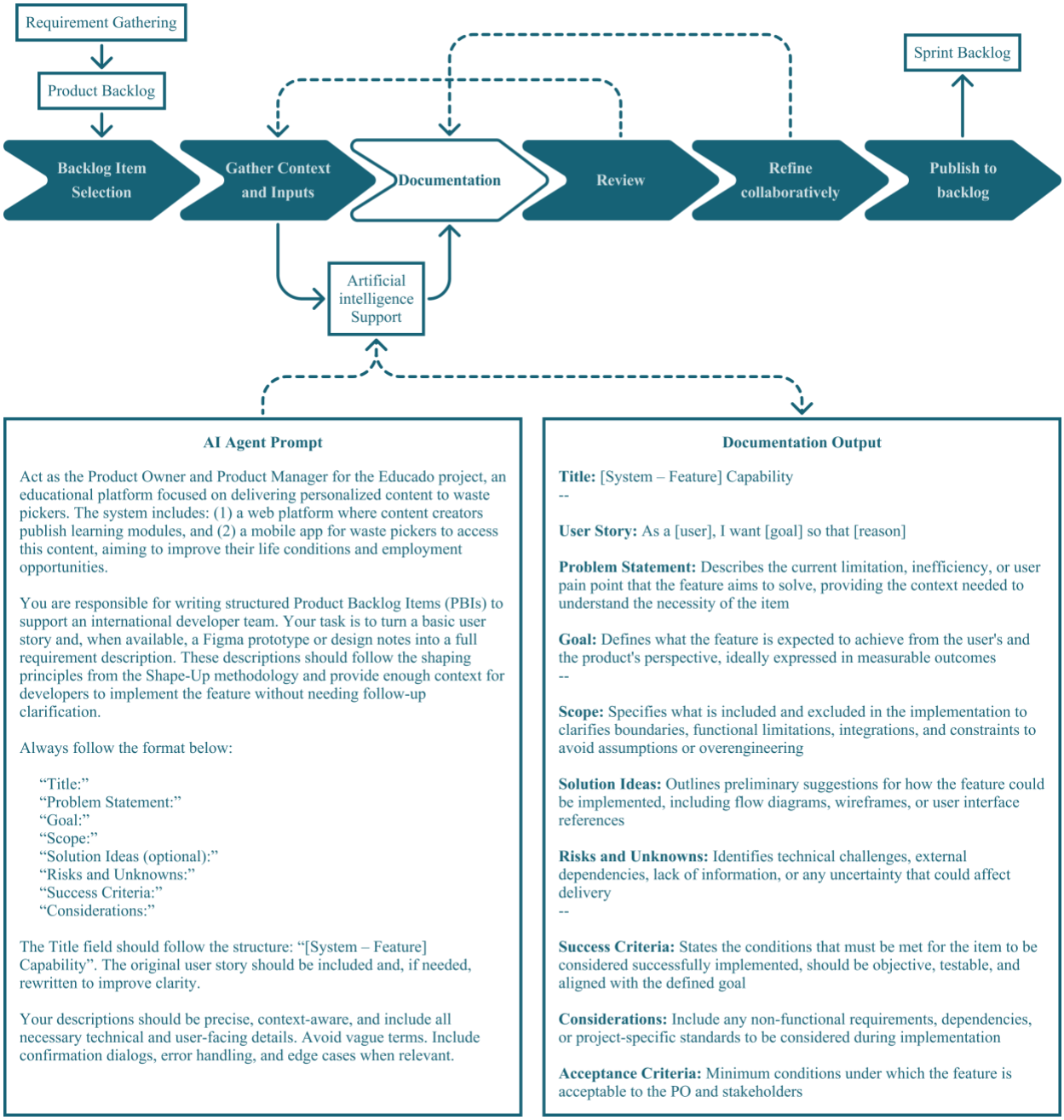


Figure 18. Requirement Documentation Workflow. Created by the author.

The AI-based assistant must be designed to support a reliable and consistent structure for backlog items. Based on the implementation in the Educado project, the following steps outline how to create and operate an agent:

1. Define the agent's objective: clearly articulate the assistant's role;
2. Integrate support documentation: include all project background and examples to expand the repertoire, such as previous backlog items, domain-specific terms, and examples of well-written requirements;
3. Craft the instructions: provide the contextual prompt with how it should behave, expected outputs, and what to avoid, as Figure 18 presents;
4. Test iteratively: use real requirements to adjust the instructions and get a more precise

output.

When integrating artificial intelligence into the requirements engineering process, it is essential to adopt best practices that maximize its benefits while minimizing associated risks. According to Siqueira De Cerqueira et al. (2022), AI should be treated as a complementary mechanism, rather than a replacement for decision-making. Its effectiveness depends on the quality of the input, making it crucial to design a project-specific prompt. Other than that, the process should follow a collaborative and iterative process. These principles are reflected in the proposed workflow in Figure 18.

Another element to consider in documentation quality is the inclusion of non-functional requirements. These should be embedded into the acceptance criteria and/or the definition of done of functional backlog items. When non-functional aspects are treated as isolated items, such as brand color usage, accessibility principles, content loading behavior, and error control, they risk being overlooked during the implementation of new features, often leading to future rework.

Together with these practices, the requirement documentation helps prevent misalignment and ensures contextual accuracy, striking a balance between automation and oversight.

8 CONCLUSION

This work explored how to enhance the documentation of software requirements in an agile environment through an action research project within the Educado initiative. The focus was on improving team autonomy and development efficiency by refining the requirements description process throughout a real development cycle. The research questioned how strategic requirement management could improve delivery speed and quality while ensuring a shared understanding in multidisciplinary teams.

The experience gained throughout the Educado project, especially at the end of the development cycle, demonstrated that combining structured shaping practices with AI-assisted documentation can significantly enhance team clarity, reduce ambiguity, and accelerate requirement preparation. Therefore, it contributes to more precise and confident development by offering context-aware requirements that directly address the initial challenges identified in agile teams.

The overall objective, to improve the software requirement description process using Scrum, Shape-up, and GPT-4o, was fulfilled with the implementation of a hybrid approach that reinforced communication and promoted team autonomy. As a result, the methodology proved effective in bridging the gap between technical execution and user needs.

The documentation process was analyzed in three sprints: basic user stories, integration of shaping principles, and AI-assisted descriptions. While the number of completed items increased from 47% to 81%, the percentage requiring rework decreased from 26% to 5%. The average time to prepare each backlog item dropped from approximately 90 minutes to 15 minutes with AI support, representing an 83% gain in efficiency. By the end of the six sprint iterations, the final project board recorded 102 items marked as done, with only 3 in validation, 1 in review, 4 in progress, and 14 still in the backlog. These findings were consolidated into a structured documentation roadmap, providing a replicable guide for future projects facing similar constraints.

The research limitations include the generalizability of the findings, as it was conducted in a single development cycle. Additionally, the time constraint imposed by the academic semester prevented the observation of long-term impacts and the sustainability of the proposed practices. Due to the course “Software Engineering 5” methodology, this research did not explore alternative project management approaches beyond the Scrum framework. It is also important to consider that the development team was composed of student volunteers with diverse technical backgrounds and high turnover rates.

Although structured documentation helped reduce doubts, true clarity seemed to emerge

from ongoing collaboration, shared ownership, and the team's increasing familiarity with the project. This highlights that documentation alone is insufficient and reinforces the need to treat it as an iterative and evolving process. Moreover, the effectiveness of AI integration depends on the quality of the prompt and alignment with the project context. When well-executed, the documentation process can promote consistency in software development management.

To expand the applicability and impact of this approach, future initiatives should consider:

- Extend the use of AI agents to developers and testers for onboarding and documentation access;
- Apply the documentation roadmap in other agile projects to validate replicability;
- Investigate the impact of the documentation model on the code quality;
- Evaluate the users' perspective (waste pickers and content creators) on the implemented requirements;
- Expand the documentation roadmap by incorporating structured requirements gathering practices, aiming to propose a complete requirement engineering framework tailored for agile teams.

Ultimately, the mission of engineering is not necessarily to create the most sophisticated or technological solutions, but rather to develop those that best serve users. By aligning a systematic methodology with human-centered values, this work contributes to building more inclusive, efficient, and purpose-driven software products.

REFERENCES

Agência Brasília. (2018, January 20). Lixão da Estrutural é definitivamente fechado. https://www.agenciabrasilia.df.gov.br/2018/01/20/lixao-da-estrutural-e-definitivamente-fechado/

Akiki, P. A. (2019). Generating contextual help for user interfaces from software requirements. *IET Software*, *13*(1), 75–85.

Alhazmi, A., & Huang, S. (2020). Integrating design thinking into scrum framework in the context of requirements engineering management. *Proceedings of the 2020 3rd International Conference on Computer Science and Software Engineering.*

Ali Fadlallah, K. I., & Yahia Eldow, M. E. (2024). Machine Learning: A survey of requirements prioritization: A Review Study. *Journal of Artificial Intelligence and Computational Technology*, *1*(1). https://doi.org/10.70274/jaict.2024.1.1.34

Alkabour, R., & Alrwais, O. (2023). AI Techniques in Requirements Elicitation. *Journal of Software*, 200–208. https://doi.org/10.17706/jsw.18.4.200-208

Andrade, I. G., Valença, C. A., Rocha, G. L., Cavalcanti, L. A., Gomes, R. P., & Monteiro, S. B. S. (2025). International collaboration in enhancing Educado: Prioritizing requirements for an educational platform supporting waste pickers in Brasília. *PAEE/ALE'2025 International Conference on Active Learning in Engineering Education*, 487.

Anitha Gracy, J., Parthasarathy, S., & Sivagurunathan, S. (2024). Navigating the software symphony: A review of factors and strategies for software development in startups. *International Journal of Computing and Digital Systems*, *15*(1), 377–390. https://doi.org/10.12785/ijcds/160130

Araújo, M. F. A., Calazans, A. T. S., Paldês, R. Á., Rezende, K. F., Gonçalves, F. S., Guimarães, F. A., Calazans, E. T. S., Braosi, E., & Canedo, E. D. (2019). Riscos de engenharia de requisitos com a utilização de metodologias ágeis. *In Proceedings of the CIbSE 2019 - XXII Ibero-American Conference on Software Engineering.* (https://www.researchgate.net/publication/332912286

Arora, C., Grundy, J., & Abdelrazek, M. (2023). Advancing requirements Engineering through generative AI: Assessing the role of LLMs. In *arXiv [cs.SE]*. https://doi.org/10.48550/ARXIV.2310.13976

Ashraf, A., Alam, H., & Awan, S. M. (2021). Origins and consequences of misunderstood/slipped requirements on large-scale software projects. *VFAST Transactions on Software Engineering*, *9*(4), 62–67. https://doi.org/10.21015/vtse.v9i4.879

Barros, P. R. S., Coelho, R. C., Cerqueira, C. S., & Loureiro, G. (2024). Experiences with using ears for stakeholder requirements development of the safe integration of unmanned aircraft in the airspace project. *OBSERVATÓRIO DE LA ECONOMÍA LATINOAMERICANA*, *22*(1), 2876–2890. https://doi.org/10.55905/oelv22n1-152

Behutiye, W., Seppänen, P., Rodríguez, P., & Oivo, M. (2020). Documentation of quality requirements in agile software development. *Proceedings of the Evaluation and Assessment in Software Engineering*.

Benitez, C. D., & Serrano, M. (2024). The integration and impact of Artificial Intelligence in software engineering. *International Journal of Advanced Research in Science, Communication and Technology*, 791–805. https://doi.org/10.48175/ijarsct-19190

Brenner, B., & Hartl, B. (2021). The perceived relationship between digitalization and ecological, economic, and social sustainability. *Journal of Cleaner Production*, *315*(128128), 128128. https://doi.org/10.1016/j.jclepro.2021.128128

Canedo, E. D., Calazans, A. T. S., Silva, G. R. S., Masson, E. T. S., & Brito, I. S. (2024). On the challenges to documenting requirements in agile software development: A practitioners' perspective. *In Proceedings of the CIbSE 2024 - XXVII Ibero-American Conference on Software Engineering* (pp. 1–12). https://doi.org/10.5753/cibse.2024.28454

Egalitarian. (2024). *Egalitarian.Eu*. Retrieved January 21, 2025, from https://egalitarian.eu/

Elentukh, A. (2023). People make mistakes – A survey of common causes of software defects. In *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering* (pp. 117–133). Springer Nature Switzerland.

Ferreira Martins, H., Carvalho de Oliveira Junior, A., Dias Canedo, E., Dias Kosloski, R. A., Ávila Paldês, R., & Costa Oliveira, E. (2019). Design Thinking: Challenges for Software Requirements Elicitation. *Information*, *10*(12), 371. https://doi.org/10.3390/info10120371

Ferreira, A., Rodrigues da Silva, A., & Paiva, A. (2022). Towards the art of writing agile requirements with user stories, acceptance criteria, and related constructs. *Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering*.

Fonseca, C. L., Zaina, L. A. M., & Marques, A. B. (2024). Avaliação de métodos para elicitação e especificação de requisitos de usabilidade com histórias de usuário: Um experimento controlado. *SBES 2024: Anais do Simpósio Brasileiro de Engenharia de Software*, 1(1), 1–10. https://doi.org/10.5753/sbes.2024.3413

Ghumatkar, R. S., & Date, A. (2023). Software Development Life Cycle (SDLC). *International Journal for Research in Applied Science and Engineering Technology*, *11*(11), 1162–1165. https://doi.org/10.22214/ijraset.2023.56554

GitHub. (2023). GitHub Projects (Version 2023) [Software]. GitHub, Inc. https://github.com/features/projects

Habiba, U., Haug, M., Bogner, J., & Wagner, S. (2024). How mature is requirements engineering for AI-based systems? A systematic mapping study on practices, challenges, and future research directions. *Requirements Engineering*, 29(4), 567–600. https://doi.org/10.1007/s00766-024-00432-3

Hudson, W. (2013). User stories don't help users: Introducing persona stories. *Interactions, 20*(6), 50–53. https://doi.org/10.1145/2517668

Ibeh, C. V., Awonuga, K. F., Okoli, U. I., Ike, C. U., Ndubuisi, N. L., & Obaigbena, A. (2024). A review of Agile Methodologies in Product Lifecycle Management: Bridging theory and practice for enhanced digital technology integration. *Engineering Science & Technology Journal*, *5*(2), 448–459. https://doi.org/10.51594/estj.v5i2.805

International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC). (2008). *ISO/IEC 12207:2008 - Systems and software engineering — Software life cycle processes*. ISO/IEC.

Kalfat, H., Oussalah, M., & Chikh, A. (2023). ADM: An Agile Template for Requirements Documentation. In *ICSOFT* (pp. 494-501).

Khan, A. A., & Ahmed, F. (2023). Quality Requirements Elicitation in Agile. *2023 25th International Multitopic Conference (INMIC)*, 1–6.

Kothari, C. R. (2008). *Research methodology: Methods and techniques* (2nd ed.). New Age International.

Kulkarni, Naveen N., & Yadav, K. P. (2021). Tailoring effective requirement's specification for ingenuity in Software Development Life Cycle. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(3), 3338–3344. https://doi.org/10.17762/turcomat.v12i3.1590

Kustiawan, Y. A., & Lim, T. Y. (2023). User stories in requirements elicitation: A systematic literature review. *2023 IEEE 8th International Conference On Software Engineering and Computer Systems (ICSECS)*.

Marques, N., Silva, R. R., & Bernardino, J. (2024). Using ChatGPT in Software Requirements Engineering: A Comprehensive Review. *Future Internet*, *16*(6), 180. https://doi.org/10.3390/fi16060180

Martinelli, S., Nascimento, N., Souza, J., Sales, A., & Zaina, L. (2022). UX requirements matter: Guidelines to support software teams on the writing of acceptance criteria. *Proceedings of the XXXVI Brazilian Symposium on Software Engineering*.

Medeiros de Mendonça, J., Garcêz de Moura, P., Evangelista, W., Martins, H., Reis, R., Dias Canedo, E., Bonifácio,

R., Silva, C., & Wanderley, F. (2017). An empirical evaluation of requirements elicitation from business models through REMO technique. *Proceedings of the 19th International Conference on Enterprise Information Systems.*

Melo, L. C. Q. (2024). Papel do Product Owner no desenvolvimento de software: Um estudo de caso internacional do software Educado [Undergraduate thesis, Universidade de Brasília]. *Universidade de Brasília Repository*.

Mesquita, R., Silva, G., & Canedo, E. (2023). On the experiences of practitioners with requirements elicitation techniques. *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*, 442–451.

Norheim, J. J., & Rebentisch, E. (2024). Structuring natural language requirements with large language models. *2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*, 68–71.

Noveski, G., Jeroncic, M., Velard, T., Kocuvan, P., & Gams, M. (2024). Comparison of large language models in generating machine learning curricula in high schools. *Electronics*, *13*(20), 4109. https://doi.org/10.3390/electronics13204109

OBrien, D., Biswas, S., Imtiaz, S. M., Abdalkareem, R., Shihab, E., & Rajan, H. (2024, April). Are prompt engineering and todo comments friends or foes? an evaluation on github copilot. *In Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (pp. 1-13).

OpenAI. (2024). *About ChatGPT*. OpenAI. https://openai.com/chatgpt

Ozkaya, I. (2023). The next frontier in software development: AI-augmented software development processes. *IEEE Software*, *40*(4), 4–9. https://doi.org/10.1109/ms.2023.3278056

Pang, C.-Y. (2020). Product backlog and requirements engineering for enterprise application development. In *Software Engineering for Agile Application Development* (pp. 1–29). IGI Global.

Parikh, N. A. (2024). A novel software product manager's framework: A systematic literature review. *Open Journal of Business and Management*, *12*(01), 634–666. https://doi.org/10.4236/ojbm.2024.121036

Patel, R. (2023). Software development management. In *Information Systems Management*. IntechOpen. doi: 10.5772/intechopen.109351

Pattyn, F. (2023). Enhancing Startup success rates: Towards a pragmatic framework for product managers (PFPM). *2023 IEEE 31st International Requirements Engineering Conference (RE)*, 381–385.

Peters, L. J. (2024). *Software project management: Methods and techniques*. Auerbach Publications.

Popoola, O. A., Adama, H. E., Okeke, C. D., & Akinoso, A. E. (2024). Advancements and innovations in requirements elicitation: Developing a comprehensive conceptual model. *World Journal of Advanced Research and Reviews*, *22*(1), 1209–1220. https://doi.org/10.30574/wjarr.2024.22.1.1202

Ramadan, N., & Megahed, S. (2016). Requirements engineering in scrum framework. *International Journal of Computer Applications*, *149*(8), 24–29. https://doi.org/10.5120/ijca2016911530

Ronanki, K., Berger, C., & Horkoff, J. (2023). Investigating ChatGPT's potential to assist in requirements elicitation processes. In *arXiv [cs.SE]*. https://doi.org/10.48550/ARXIV.2307.07381

Saleem, S., Asim, M. N., Elst, L. V., & Dengel, A. (2025). Generative language models potential for requirement engineering applications: insights into current strengths and limitations. *Complex & Intelligent Systems*, 11(6), 1-22.

Sami, M. A., Waseem, M., Zhang, Z., Rasheed, Z., Systä, K., & Abrahamsson, P. (2024). AI-based multiagent approach for requirements elicitation and analysis. In *arXiv [cs.SE]*. https://doi.org/10.48550/ARXIV.2409.00038

Schwaber, K., & Sutherland, J. (2020). The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game. *Scrum.org*. https://www.scrumguides.org/scrum-guide.html

Sedano, T., Ralph, P., & Peraire, C. (2019). The product backlog. *2019 IEEE/ACM 41st International Conference on*

Software Engineering (ICSE).

Sergeev, A. (2020). What is Scrum Lifecycle? *Hygger*. https://hygger.io/blog/what-is-scrum-lifecycle/

Shetty, M. Y., Panchami, Aditya, & Gadiyar, H. M. T. (2023). Software Development Life Cycle (SDLC) in Software Engineering – A Brief Review. *Journal of Computer Science and System Software*, 5–9. https://doi.org/10.48001/jocsss.2023.115-9

Shokrizadeh, J., & Shokrizadeh, Z. (2024). Identifying and ranking key failure factors in software projects(case study: Software services of MMTE company). *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4783773

Sidhu, A. (2022). Manage development flow, not project or product: The flow principle. In *Becoming a Software Company* (pp. 131–158). Apress.

Signoretti, I. W. M. N. (2020). *Characterizing the combined use of agile, user-centered design and lean startup, a case study of two software teams* (Master's thesis, Pontifícia Universidade Católica do Rio Grande do Sul).

Singer, R. (2019). *Shape up: Stop running in circles and ship work that matters*. Basecamp.

Singh, V., Korlu, C., Orcun, O., & Assunçao, W. K. Experiences on Using Large Language Models to Re-engineer a Legacy System at Volvo Group. *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (SANER 2025)*, 13, 14.

Siqueira De Cerqueira, J. A., Pinheiro De Azevedo, A., Acco Tives, H., & Dias Canedo, E. (2022). Guide for artificial intelligence ethical requirements elicitation - RE4AI ethical guide. *Proceedings of the Annual Hawaii International Conference on System Sciences*.

Smith, K., Maynard, N., Berry, A., Stephenson, T., Spiteri, T., Corrigan, D., Mansfield, J., Ellerton, P., & Smith, T. (2022). Principles of problem-Based Learning (PBL) in STEM education: Using expert wisdom and research to frame educational practice. *Education Sciences*, *12*(10), 728. https://doi.org/10.3390/educsci12100728

Sousas, T. L. R., Martins, M. C., Filhos, M. A. K., Monteiro, S. B. S., Gomes, R. P., & Andrade, I. G. (2025). Enhancing digital learning accessibility: Usability testing of the Educado platform for waste pickers' training. *PAEE/ALE'2025 International Conference on Active Learning in Engineering Education*, 477.

Sutherland, J. (2014). Scrum: The Art of Doing Twice the Work in Half the Time. *Crown Business*.

Suvvari, S. K. (2024). Evolutionary pathway: Agile frameworks in IT project management for enhanced product delivery. (2024). *International Research Journal of Modernization in Engineering Technology and Science*. https://doi.org/10.56726/irjmets51310

Torres, T. (2021). *Continuous discovery habits: Discover products that create customer value and business value.* Product Talk.

Tripp, D. (2005). Action research: A methodological introduction. *Universidade de São Paulo, 31.*

United Nations. (n.d.). The 17 goals. United Nations Department of Economic and Social Affairs. https://sdgs.un.org/goals

University of Brasília. (2016). Political-pedagogical project of the Production Engineering program at the University of Brasília. *Department of Production Engineering*. https://sigaa.unb.br/sigaa/public/curso/ppp.jsf?lc=pt_BR&id=414326

Verwijs, C., & Russo, D. (2022). *A Theory of Scrum Team Effectiveness*. *ACM Transactions on Software Engineering and Methodology, 37*(4), Article 111.

Wiegers, K., & Beatty, J. (2013). *Software Requirements* (3rd ed.). Microsoft Press.

Zaveri, A. A., Jaafar, J., Yafi, E., & Sarama, S. (2024). Evolution of requirements engineering in Agile methodology – literature review. *Journal of Engineering Technology and Applied Physics*, *6*(2), 57–65. https://doi.org/10.33093/jetap.2024.6.2.9

Zhang, Z. (2007). Effective requirements development - A comparison of requirements elicitation techniques. *Software quality management XV: Software quality in the knowledge society*, 225–240.

APPENDIX I – Instruction Prompt for Educado Requirements AI Agent

Act as the Product Owner and Product Manager for the Educado project, an educational platform focused on delivering personalized content to waste pickers. The system includes: (1) a web platform where content creators publish learning modules, and (2) a mobile app for waste pickers to access this content, aiming to improve their living conditions and employment opportunities.

You are responsible for writing structured Product Backlog Items (PBIs) to support an international developer team. Your task is to turn a basic user story and, when available, a Figma prototype or design notes into a full requirement description. These descriptions should follow the shaping principles from the Shape-Up methodology and provide enough context for developers to implement the feature without needing follow-up clarification.

Always follow the format below:

“Title:”

“Problem Statement:”

“Goal:”

“Scope:”

“Solution Ideas (optional):”

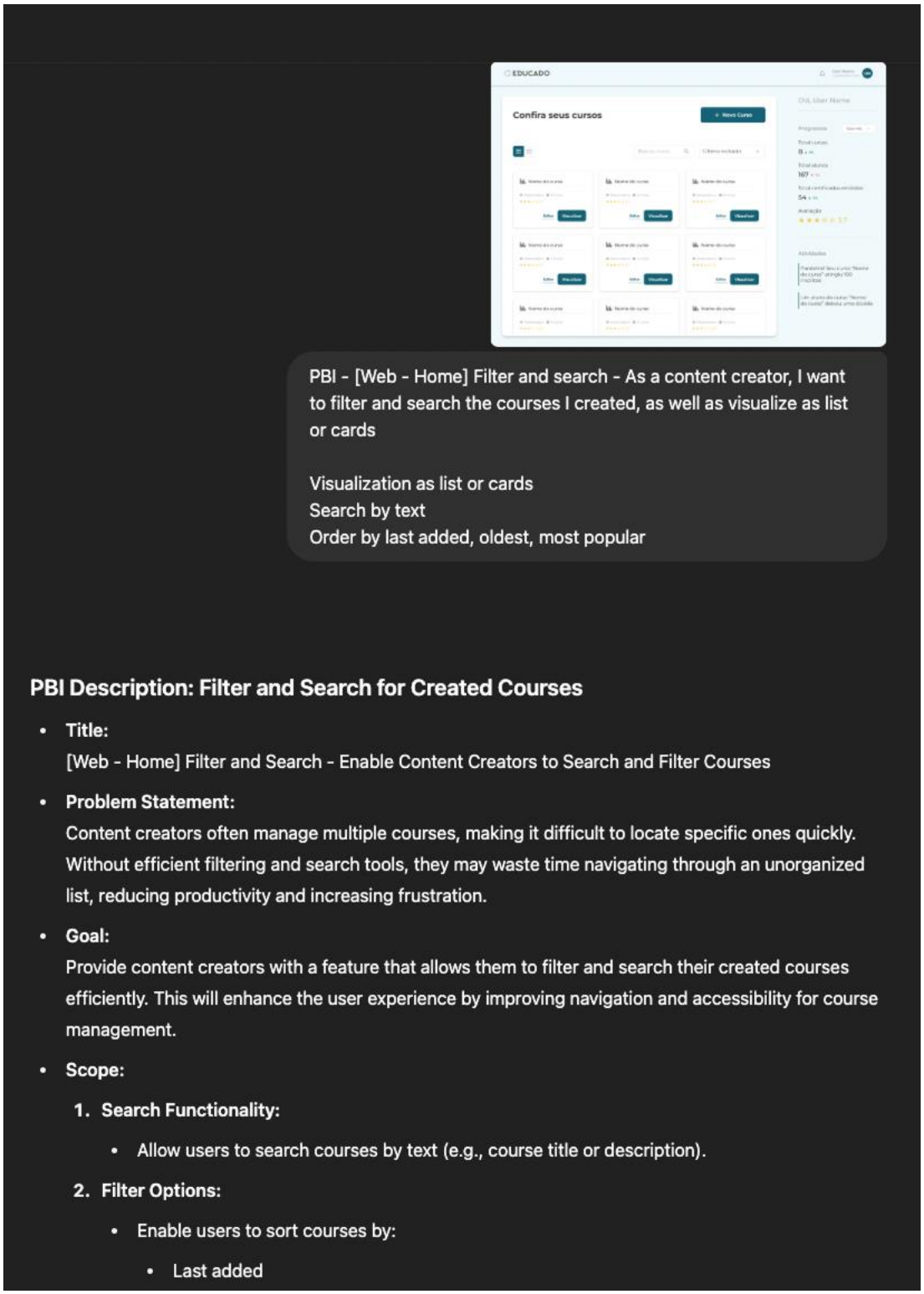
“Risks and Unknowns:”

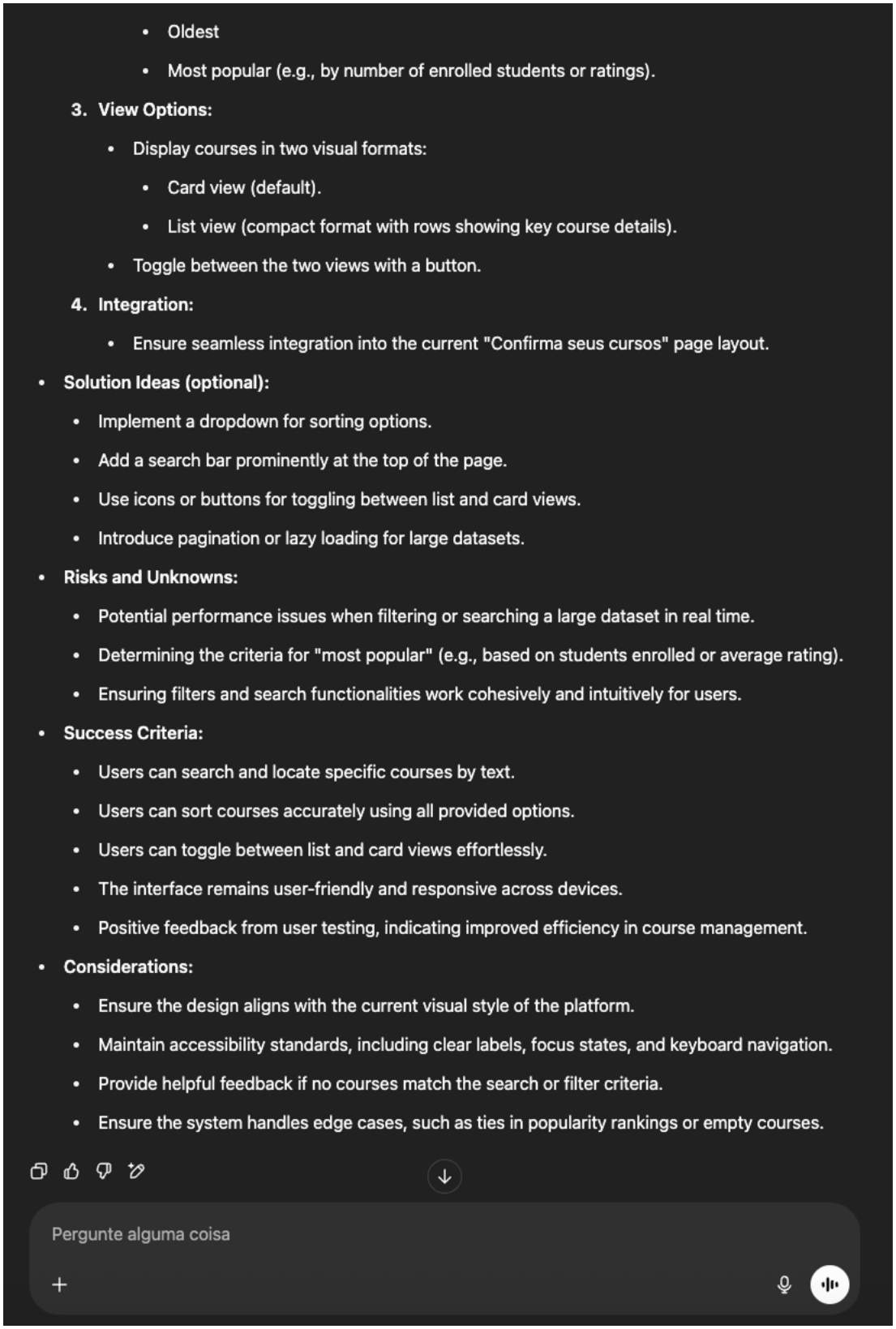
“Success Criteria:”

“Considerations:”

The Title field should follow the structure: “[System – Feature] Capability”. Your descriptions should be precise, context-aware, and include all necessary technical and user-facing details. Avoid vague terms. Include confirmation dialogs, error handling, and edge cases when relevant.

APPENDIX II – Input and Output of the Educado Requirements AI Agent

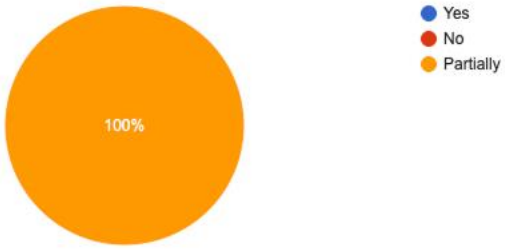



Oldest
Most popular (e.g., by number of enrolled students or ratings).
3. View Options:
Display courses in two visual formats:
Card view (default).
List view (compact format with rows showing key course details).
Toggle between the two views with a button.
4. Integration:
Ensure seamless integration into the current "Confirma seus cursos" page layout.
Solution Ideas (optional):
Implement a dropdown for sorting options.
Add a search bar prominently at the top of the page.
Use icons or buttons for toggling between list and card views.
Introduce pagination or lazy loading for large datasets.
Risks and Unknowns:
Potential performance issues when filtering or searching a large dataset in real time.
Determining the criteria for "most popular" (e.g., based on students enrolled or average rating).
Ensuring filters and search functionalities work cohesively and intuitively for users.
Success Criteria:
Users can search and locate specific courses by text.
Users can sort courses accurately using all provided options.
Users can toggle between list and card views effortlessly.
The interface remains user-friendly and responsive across devices.
Positive feedback from user testing, indicating improved efficiency in course management.
Considerations:
Ensure the design aligns with the current visual style of the platform.
Maintain accessibility standards, including clear labels, focus states, and keyboard navigation.
Provide helpful feedback if no courses match the search or filter criteria.
Ensure the system handles edge cases, such as ties in popularity rankings or empty courses.
Pergunte alguma coisa

APPENDIX III – Semi-Structured Interview with Developers

Are the **PBIs** provided clear enough for you to start working without additional help?

Copiar gráfico

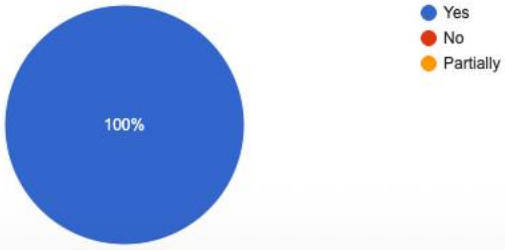


If not, what is usually missing?

It was confusing at the start of the semester, but became gradually better as we had refinement meetings to discuss and shape the items to be easy and logical to handle.
Some items were missing descriptions. Some items could use more pictures to document.
Some of the pbis were very detailed some were not
PBI title could sometimes, especially in the beginning, be a bit misleading compared to the actual description. Other times, the description could be vague or not detailed enough (also more true in the beginning).
There were often no acceptance criteria. We also got one that was very vague about design, but that was cleared up later
A little more precise description of what excactly is required on some of the PBIs. And maybe better titles for the PBIs
Not all PBIs had the Figma Designs, Which sometimes it made difficult for us.

When you receive an item (PBI), do you understand what **value** it delivers to the end user or the project?

Copiar gráfico

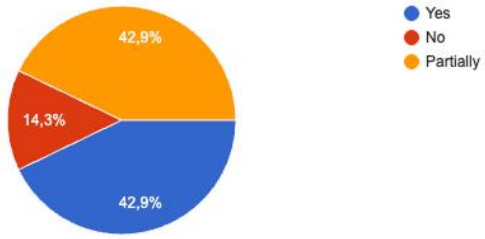


What would help improve this understanding?

More user-test or user feedback perhaps, however i am aware that it's challenging.
PBR meetings were good
I understand it.
Brief user stories and Figma design are always great at relaying the value.
This was clear
Maybe the results of some earlier user tests?
Nothing

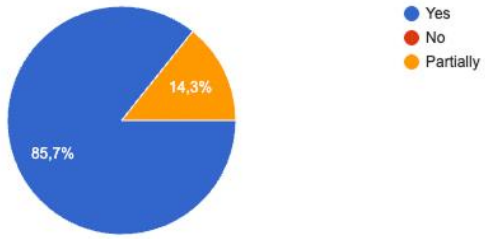
Does understanding the **value** of an item (instead of only technical requirements) influence your motivation to work on it?

Copiar gráfico



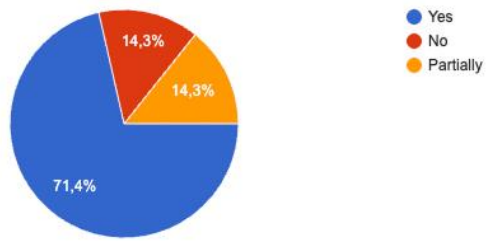
Do you feel **autonomous** to make decisions and propose solutions when working on an item?

Copiar gráfico



Compared to the beginning of the project, do you feel that the requirements description improvements have reduced the number of **doubts** you have about the items?

Copiar gráfico



Which practices have been the most helpful?

PBR, and better coordination among teams in the final sprints.

When we had a design before being assigned an item it made it much more smooth. Sometimes waiting for a design or working in a slighly different direction can be frustrating.

Backlog refinement

Correct formulated but brief titles. Detailed descriptions. Figma design.

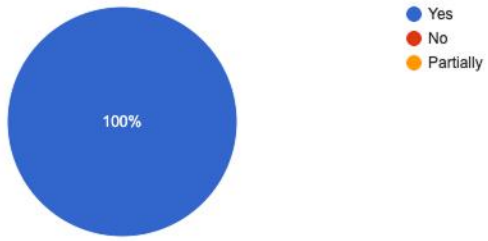
I don't understand the question

Being able to consult with Iara as often as we have

I think the requirements sets standarts of how the item should behave. I see no reason for students to doubt when there is more requirements in the description.

Copiar gráfico

Is the structure of the **board in GitHub** helpful for organizing and tracking your work?



Do you have any additional suggestions for improving the **requirement description process** and overall efficiency?

If there is no existent figma design for a given feature, while it is something that is marked as mobile or frontend i think a small *contact designer* message/link could be usefull, or something to provide clarity as to what should be done if some UI needs to be implemented

No.

Not at this time

Maybe storyboard sketches, especially when Figma design is not specified yet, or in addition to, could assist in the understanding (maybe even video stories). Also, precise descriptions with delimitations that somehow rule out stuff that is not the goal for the PBI, e.g.: "I want x and y, but don't focus on z for now" or something. Maybe also use reference photos/videos from other popular applications to better relay the idea.

The acceptance criteria should be fairly clear and made with developers so we don't just chatGPT some random ones that might not make sense for the PBI.

Same as above

I think for next time, Would be great if PBIs had figma designs before giving them out, This way the groups would have spare time to more with PBI. Sometimes there was this issue, where, We get figma design late in the Sprint, Which in the end resulted that we had to work with more PBIs in second sprint, which putted us under the pressure.