HELLENIC OPEN UNIVERSITY

“School of Social Sciences”

Postgraduate Dissertation

“Design and Evaluation of an Agentic AI Prototype for Task Prioritization in Software Project Management”

“Aristeidis Kantas”

Supervisor: “Thomas Dasaklis”

Patras, Greece, “June” “2026”

HELLENIC OPEN UNIVERSITY

“Design and Evaluation of an Agentic AI Prototype for Task Prioritization in Software Project Management”

“Aristeidis Kantas”

Supervising Committee

Supervisor:
“Thomas Dasaklis”

Co-Supervisor:
“Athanasios Mihiotis”

Patras, Greece, “June” “2026

Abstract

This dissertation presents the design, implementation and evaluation of a lightweight Agentic Artificial Intelligence (AI) prototype intended to automate and support key aspects of task management within software project workflows. The proposed system leverages the capabilities of large language models (LLMs) to read and interpret software project backlogs, perform task assignments, recommend deadlines, generate sprint-level summaries, and prioritize work items.

The implemented prototype operates as a prompt-engineered, AI-assisted reasoning pipeline within predefined boundaries. It demonstrates limited agentic characteristics, such as contextual interpretation, goal-oriented task analysis, and structured recommendation generation, but it should not be understood as a fully autonomous AI agent. Instead of completely replacing human action, the system is designed to assist managers and team leaders in planning. The prototype integrates agentic behaviors such as contextual reasoning, goal awareness, and adaptive prioritization, allowing it to respond dynamically to different project environments and backlog characteristics.

To ensure ethical compliance, the system was evaluated using synthetically constructed backlogs inspired by real-world software projects, without relying on proprietary or sensitive corporate data. The evaluation focuses on functional accuracy, responsiveness, adaptability, and consistency of outputs across multiple project scenarios.

Beyond technical implementation, we examine the managerial implications of deploying agentic AI systems in engineering management contexts. Key considerations include their potential to enhance decision quality, optimize resource allocation, and improve team coordination, alongside challenges related to trust, transparency, accountability, and effective human–AI collaboration. The findings contribute to both the technical feasibility and the strategic governance discussion surrounding the integration of autonomous AI agents into modern software project management practices

Keywords

Agentic AI, Generative AI, Large Language Models, Software Project Management, Backlog Prioritization, Engineering Management, Human-AI Collaboration, Decision Support Systems

Περίληψη

Η παρούσα διπλωματική εργασία παρουσιάζει τον σχεδιασμό, και την αξιολόγηση ενός ελαφρού πρωτοτύπου Τεχνητής Νοημοσύνης με στόχο την αυτοματοποίηση και υποστήριξη βασικών διαδικασιών διαχείρισης έργων λογισμικού. Το προτεινόμενο σύστημα αξιοποιεί τις δυνατότητες μεγάλων γλωσσικών μοντέλων (LLMs) για την ανάγνωση και ερμηνεία backlogs έργων λογισμικού, την ανάθεση εργασιών, την πρόταση προθεσμιών, τη δημιουργία συνοπτικών αναφορών sprint και την ιεράρχηση εργασιών με βάση την εκτιμώμενη επιχειρηματική αξία και τους περιορισμούς του έργου.

Το υλοποιημένο πρωτότυπο δεν αποτελεί ένα πλήρως αυτόνομο AI agent σύστημα με την αυστηρή τεχνική έννοια. Αντίθετα, λειτουργεί κυρίως ως ένα prompt-engineered, AI-assisted reasoning pipeline, το οποίο ενσωματώνει περιορισμένα agentic χαρακτηριστικά, όπως ερμηνεία πλαισίου, προσανατολισμό σε συγκεκριμένο στόχο και παραγωγή δομημένων προτάσεων σχεδιασμού. Δεν αποσκοπεί στην αντικατάσταση της ανθρώπινης κρίσης, αλλά στη λειτουργία ως υποστηρικτικό σύστημα λήψης αποφάσεων για engineering managers και team leads. Οι προτάσεις που παράγονται από το σύστημα παραμένουν ελεγχόμενες και αναθεωρήσιμες από τον ανθρώπινο χρήστη.

Η αξιολόγηση του συστήματος πραγματοποιείται κυρίως με χρήση συνθετικών backlogs εμπνευσμένων από πραγματικά έργα λογισμικού, ενώ δημόσια διαθέσιμα δεδομένα από GitHub αποθετήρια εξετάζονται ως κατάλληλη και ηθικά ασφαλής πηγή για πρόσθετη ή μελλοντική αξιολόγηση. Με αυτόν τον τρόπο αποφεύγεται η χρήση ή έκθεση εμπιστευτικών εταιρικών δεδομένων. Η μελέτη εξετάζει την ακρίβεια λειτουργίας, την πληρότητα των αποτελεσμάτων, την απόκριση, την προσαρμοστικότητα και τη συνέπεια των παραγόμενων προτάσεων.

Παράλληλα, η εργασία διερευνά τις διοικητικές επιπτώσεις της υιοθέτησης agentic AI σε περιβάλλοντα engineering management, αναδεικνύοντας τόσο τα οφέλη στη λήψη αποφάσεων και την κατανομή πόρων, όσο και τις προκλήσεις που σχετίζονται με την εμπιστοσύνη, τη διαφάνεια και τη συνεργασία ανθρώπου–AI.

Λέξεις – Κλειδιά

Τεχνητή Νοημοσύνη με Πρακτορικά Χαρακτηριστικά, Γενετική Τεχνητή Νοημοσύνη, Μεγάλα Γλωσσικά Μοντέλα, Διαχείριση Έργων Λογισμικού, Ιεράρχηση Backlog, Διοίκηση Μηχανικών, Συνεργασία Ανθρώπου–Τεχνητής Νοημοσύνης, Συστήματα Υποστήριξης Αποφάσεων

Table of Contents

Table of Figures

1 Introduction

1.1 Background and Motivation

In recent years software project managers face more and more difficulties. It is common for systems to have more interconnected parts and team members often work in different geographic locations. As a result companies require that staff deliver software in less time while they maintain the same level of performance. On that account many teams use agile or hybrid methods. By using those processes, workers complete tasks in small increments plus change their plans if the situation requires it - but the methods still require that managers and team leaders organize the work but also make many choices.

Many project management activities still depend on human judgment. Backlog refinement, task prioritization, sprint planning, workload balancing, and progress reporting are usually done manually. As projects become larger, this work becomes more difficult and time-consuming. Decisions may also be affected by incomplete information, personal opinion, delivery pressure, or too much project data. This creates a need for intelligent tools that can support managers, while keeping the final responsibility on the human side.

At the same time, artificial intelligence has improved a lot, especially through large language models. These models can work with natural language and produce structured information from text. This is useful in software project management, because backlog items, user stories, bug reports, sprint notes, and technical comments often contain important information that is not always written in a clear format.

These developments have also supported the rise of Agentic AI systems. Unlike simple automation tools that follow fixed rules, Agentic AI systems can work towards a goal and use the available context. They can understand a task, follow steps, and produce outputs within specific limits. This makes them useful for workflows that involve reasoning, prioritization, and decision support.

The motivation of this dissertation comes from this connection. Software project management is becoming more complex, while AI systems are becoming more capable of supporting text interpretation and structured planning. However, many existing AI approaches in project management focus mainly on predictions, reports, or dashboards. Less

focus has been given to lightweight AI agents that support everyday activities such as backlog interpretation, task prioritization, role suggestions, and sprint planning summaries. For this reason, this dissertation examines whether an Agentic AI prototype can support these activities in a useful and transparent way, while the human manager keeps control of the final decisions.

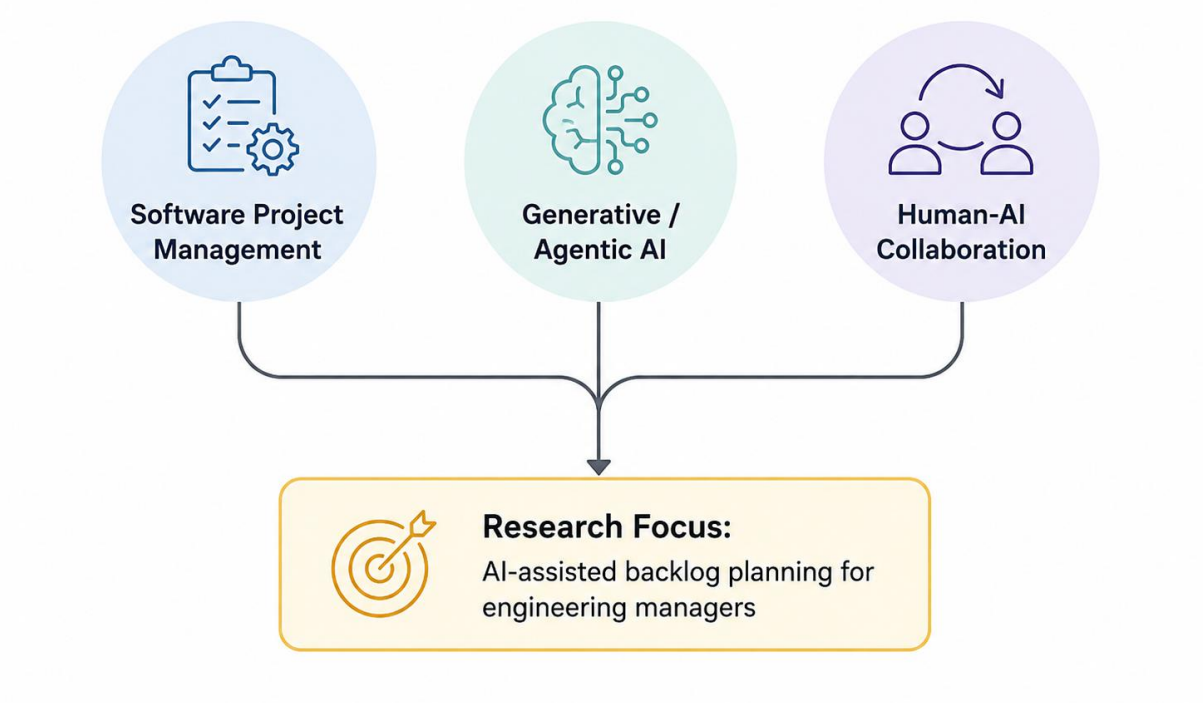

Figure 1: Conceptual Foundation of the Study

1.2 Problem Statement

There are many advanced project management platforms available today. However software teams still face important difficulties in their everyday planning work. Also it is difficult to assign priority on all of the tasks. In addition, it is hard to find the connection between the technical work with the business goals of the project. Most of the time backlog items are prioritized based on limited information or personal judgment. Sometime urgent requests and the pressure to deliver something quickly can also affect the priority assignment. This can lead to decisions that are not always the best for the project. It can also make it harder for the rest of the team to understand why some tasks were selected over others.

AI tools have also started to appear in the field of project management, but most of them still work mainly as advisory systems. They can provide forecasts, reports, or recommendations, but they usually do not take an active role in the actual planning process.

Because of this, the project manager still has to interpret the results, connect them with the real project context, and decide what should happen next. This limits the extent to which such tools can actually reduce managerial workload. Another important issue is that many of these systems are not able to fully understand unstructured backlog descriptions, unclear requirements, changing priorities, or dependencies that appear as the project evolves.

Based on this problem, the present dissertation examines whether a lightweight Agentic AI system can support core task management activities in software development projects. More specifically, it investigates whether such a system can read and interpret project backlogs, prioritize tasks using more than one criterion, suggest suitable roles and deadlines, and produce useful managerial summaries. At the same time, the system should remain transparent and controllable, so that it supports the work of the project manager without replacing human judgment or responsibility.

1.3 Research Objectives

The primary objective of this dissertation is to design, implement, and evaluate an Agentic AI prototype that supports task management within software project workflows.

The specific objectives of the study are to:

- Design a lightweight AI-assisted planning architecture with limited agentic characteristics, tailored to the requirements of software project management
- Implement a prototype capable of interpreting structured and semi-structured backlog data
- Enable AI-assisted task prioritization recommendations based on business value, urgency, and contextual factors
- Generate task assignment and deadline suggestions consistent with agile project constraints
- Produce sprint-level summaries that support managerial situational awareness
- Evaluate the system's functional accuracy, responsiveness, and adaptability across diverse project scenarios
- Analyze managerial implications related to trust, transparency, accountability, and human–AI collaboration

1.4 Research Questions

To achieve the above objectives, the study is guided by the following research questions:

1. How can a lightweight AI-assisted reasoning pipeline with limited agentic characteristics be designed to support task management in software projects while operating within clearly defined managerial constraints?
2. To what extent can such a system accurately interpret project backlogs and prioritize tasks in a manner that aligns with managerial expectations and business objectives?
3. What opportunities and risks arise from the introduction of Agentic AI systems into engineering management practices, particularly with regard to trust, transparency, and human oversight?

1.5 Significance of the Study

This dissertation is expected to contribute both to academic research and to managerial practice. From an academic point of view, it does not examine artificial intelligence in project management only as a theoretical idea. Instead, it focuses on the design, implementation, and evaluation of a working Agentic AI prototype. In this way, the study adds a more practical perspective to the existing literature, by showing how such a system can be structured, limited, and tested within a software project management context.

From a practical point of view, the dissertation can help engineering managers and technology leaders better understand how Agentic AI may support their work. The goal is not to present AI as a replacement for the manager, but as a tool that can assist with specific planning activities. For example, the system may help make task prioritization more consistent, reduce some of the manual effort involved in backlog review, and improve visibility over project status, dependencies, and risks.

At the same time, the study also looks at the risks and responsibilities that come with the use of AI in managerial work. AI recommendations should not be accepted without checking them first. They need to be understandable and easy to review. Also, it is important that they are connected with the decision-making process, so that the human manager can use them before making the final decision. This is why the dissertation also discusses transparency and accountability. It also examines who is responsible when AI is used to support project management decisions. In the end, human judgment remains very important, because AI can support the process, but it cannot fully understand every business, technical, or team-related factor by itself.

1.6 Scope of the Study

The scope of this dissertation is limited to task management activities within software development projects, with particular emphasis on backlog interpretation, task prioritization, assignment suggestions, deadline estimation, and sprint-level summarization. The study focuses on the design, implementation, and evaluation of a lightweight AI-assisted backlog planning prototype. The prototype is evaluated primarily through controlled testing with synthetic backlog data and is intended to demonstrate practical feasibility rather than production-grade deployment.

2 Literature Review

This chapter reviews the literature relevant to the development of an Agentic AI prototype for software project task management. The purpose of the chapter is to establish the theoretical and practical foundation of the dissertation by examining existing knowledge in software project management, decision-making in engineering environments, artificial intelligence in project management, large language models, Agentic AI systems, and human–AI collaboration. In doing so, the chapter provides the academic basis for the system design and implementation that follow in later chapters.

The topic of this dissertation is connected with two main areas. The first one is software project management. This field focuses on how software systems are planned, coordinated, monitored, and finally delivered. In this process, project managers need to take into account many limitations, such as time, cost, project scope, quality, and the people that are available to work on the project. The second area is artificial intelligence. More specifically, this dissertation focuses on the recent progress of large language models and agentic AI systems. These two areas are becoming more connected, because modern software projects produce a lot of information in textual and operational form. At the same time, project managers and engineering leaders are expected to make fast and consistent decisions, even when the available information is not complete.

Software project management has changed a lot during the last decades. In the past, many approaches were based on sequential planning and strict control. The work was usually planned from the beginning and then followed step by step. Today, approaches such as Agile

and Scrum give more importance to adaptability and collaboration. They also focus on delivering work in smaller iterations instead of waiting until the end of the project. However, even in Agile environments, many important project management activities still depend on human judgment. For example, the team still needs to prioritize the backlog, estimate the effort, assign tasks, prepare the sprint, and report the progress. These activities are not simple mechanical steps. They require understanding of the project context and the ability to compare different options. They also require trade-off analysis, because one decision may affect time, quality, cost, or team workload. This is why these activities are difficult to automate with traditional rule-based systems.

At the same time, artificial intelligence has progressed from narrow predictive models toward systems that can interpret natural language, generate structured outputs, and support complex reasoning tasks. Prompting techniques such as chain-of-thought prompting (Wei et al., 2022) have further shown that LLMs can improve performance on reasoning-oriented tasks when intermediate reasoning is elicited through carefully designed prompts. Large language models are especially relevant to project management because many project artifacts are text-based. User stories, issue descriptions, bug reports, requirements, sprint notes, and technical documentation often contain valuable information, but this information is not always structured in a way that can be easily processed by traditional tools. LLMs create new opportunities for extracting meaning from such data and transforming it into managerial insights.

This literature review therefore aims to answer a central question: what does existing research suggest about the possibility of using Agentic AI to support software project management, and what gaps remain? The chapter does not simply describe existing literature. Instead, it critically connects the literature to the specific problem addressed by this dissertation: the design and evaluation of a lightweight Agentic AI system that can interpret software project backlogs, prioritize tasks, suggest assignments and deadlines, and generate sprint-level summaries.

Figure 2: Literature Review Synthesis

The structure of the chapter follows a logical progression. It begins with software project management as the problem domain, then moves to decision-making in engineering management as the managerial context. It then discusses the role of artificial intelligence in project management, before narrowing the focus to large language models and Agentic AI. Finally, it considers human–AI collaboration and identifies the research gap addressed by this dissertation.

2.1 Software Project Management

Software project management has a lot of aspects. There are a lot of principles, methods and tools that are used in order to be able to plan, organize and deliver software systems that are robust. It may share some similar points with other engineering projects however in the case of the latter the final deliverable is more visible and the requirements are mostly set from the get go, whereas in software, requirements change so frequently during the development and in many cases decisions depend on the knowledge and experience of the team. This suggest that software project management is a difficult activity, especially when teams work in a fast-moving technical and business environments.

In the past, software development was strongly connected with sequential lifecycle models (Royce, n.d.) . Royce's work is one of the most well-known examples. His model later became associated with the Waterfall approach, where requirements, design, implementation, testing, and deployment follow separate phases. However, Royce also recognized that a fully linear process can create problems when important issues appear late in development. This shows that concerns about strict planning and late feedback existed from the early years of software engineering.

Traditional project management can provide structure and stability, but it works better when requirements are clear and the delivery process is well understood. In software projects this is not always the case. Customer expectations, technical uncertainty, and market conditions can change during the project. This is why a plan made only at the beginning can quickly become outdated (Boehm, 1988).

Agile and Scrum help teams respond to change (Kent Beck, n.d.), but they do not make project management simple. Backlog management still requires continuous refinement and prioritization. A backlog may include user stories, bugs, technical debt, infrastructure work, documentation, and security tasks. It is not always clear what should be done first, because some tasks have business value, while others may block future work.

This creates a prioritization challenge (Achimugu et al., 2014). Agile frameworks provide useful principles, but the product owner, engineering manager, or team must still evaluate tasks, understand dependencies, estimate complexity, and handle changing requirements. These decisions require judgment and cannot always be solved with simple scoring.

Team coordination is also important (Moe et al., 2010) . Research shows that Agile teamwork depends on communication, leadership, coordination, and shared understanding. Moe, Dingsøyr and Dybå found that coordination and leadership issues can limit team effectiveness even when Agile practices are used. In larger organizations, Agile becomes even more complex (Stettina & Hörz, 2015)because teams must coordinate across products and departments. Stettina and Hörz also showed that Agile portfolio management affects routines, structures, and values at the portfolio level.

Software project management is therefore both technical and managerial. It includes dependencies, uncertainty, changing requirements, communication, prioritization, resource allocation, and decisions under pressure. This makes it a suitable area for AI-supported

tools. However, these tools must be designed carefully. A simple automation system may not understand the project context, while a system with too much autonomy may create risks without human control.

For this reason, this dissertation focuses on a lightweight Agentic AI prototype and not on a fully autonomous project management system. The goal is to examine whether an AI agent can support time-consuming activities, such as interpreting backlog items, identifying priorities, suggesting roles, estimating deadlines, and creating sprint summaries.

2.2 Decision-Making in Engineering Management

An engineering manager spends a large portion of their time when they make decisions. In software projects, the manager determines which features are the first to be developed, which features occur later and which people perform specific tasks - those choices are complex because the manager does not possess every detail when they begin a project. Technical risks exist, deadlines create pressure, and stakeholders have various goals for the outcome. For those reasons, managers base many choices on the data they have at a particular time, rather than on facts that are certain.

Traditional theories of decision-making often assume that a person can examine all possible options and then choose the best one. However, this does not describe very well how decisions are made in real projects. Herbert Simon's idea of bounded rationality is important here, because it explains that managers have limited information and limited time. Because of this, they cannot always find the perfect solution. In many cases, they choose a solution that is good enough for the current situation. In software project management, this happens when managers and teams prioritize tasks without knowing every technical detail, every dependency, or every future risk.

Decision-making becomes even more difficult because of cognitive bias and information overload. Kahneman explains that people often use mental shortcuts (Kahneman, 2011) , and these shortcuts can sometimes lead to mistakes. For example, a recent customer issue may receive more attention than a deeper architectural problem. A team may also underestimate a task because a similar task seemed easy in the past. At the same time, modern software teams produce a lot of data from tickets, commits, pull requests, bug

reports, tests, logs, alerts, and customer feedback. This data can help managers, but it can also make it harder to understand what is actually important.

In this context, decision support systems and artificial intelligence are relevant tools (Jarrahi, 2018). By using artificial intelligence, engineering managers receive assistance when the system analyzes project information. It finds patterns, creates summaries plus suggests actions that are possible. It is necessary that the system explains why it makes suggestions. It must also follow the goals and limits that the manager sets. If the system provides recommendations but does not provide a reason, the manager might not trust the output. Risks exist if the system acts but also does not have limits. On that note, the prototype in this dissertation supports specific decisions - those tasks include when the manager prioritizes tasks, suggests roles, estimates deadlines, and summarizes sprints. It does not attempt to make decisions that are perfect as well as it does not replace the manager. Its purpose is to ensure that the process is structured, clear, and repeatable. And the final responsibility is with the human manager.

2.3 Artificial Intelligence in Project Management

Artificial intelligence has started to receive more attention in project management, because it can support activities such as forecasting, planning, risk identification, resource allocation, and decision-making. Traditional project management depends a lot on human experience, historical information, and manual reporting. With AI, it becomes possible to analyze larger amounts of project data and identify useful patterns that may not be easy for a manager to find manually (Perkusich et al., 2020).

Early uses of AI in project management were mostly focused on prediction and optimization. Machine learning methods have been used to estimate project duration, detect risks, and support project control. For example, Wauters and Vanhoucke used a similarity-based method to forecast project timelines in the context of earned value management (Wauters & Vanhoucke, 2017). More recent studies also focus on explainable AI. This is important because managers do not only need accurate results. They also need to understand why a system produces a specific result, so that they can use it in a practical way.

In general, AI in project management can support several areas. It can be used for predictive analytics, such as estimating delays or cost overruns. It can also help with resource allocation, risk detection, and project reporting. However, many existing AI tools depend on structured data, such as schedules, cost records, time logs, or resource lists. This creates a limitation in software project management, because a lot of important information is written in text. Backlog items, bug reports, sprint notes, and technical comments may contain useful information, but this information is not always organized in a standard format.

Another limitation is that many AI systems remain passive. They may produce forecasts, dashboards, or warnings, but they do not take an active role in the project's workflow. For example, a system may predict that a project will be delayed, but it may not explain which tasks should be prioritized next. A dashboard may show that a team has too much work, but it may not create a practical sprint plan. This difference between passive analysis and active support is important for this dissertation.

There are also issues of trust and responsibility. A project manager may not trust an AI recommendation if the system does not explain its logic. On the other hand, if users trust the system too much, they may accept its suggestions without proper review. This creates the risk of automation bias. For this reason, AI systems in project management should be technically useful. They should also be understandable and suitable for managerial work.

Digitalization is also changing project management in a wider way, by introducing new tools, new ways of collaboration, and more data-driven practices. Because of this, Agentic AI should be examined not only as a technical tool, but also as a management tool. This dissertation follows this direction by focusing on a lightweight system that is connected to task management. Instead of building a large enterprise platform, the study examines a focused problem: how backlog data can be transformed into priorities, role suggestions, deadline estimates, and summaries.

The move from predictive AI to Agentic AI is important. Predictive AI mainly answers questions such as "What is likely to happen?". Agentic AI can support questions such as "What should be done next?" and "How should this information become action?". This does not mean that AI should make the final decision. It means that AI can prepare structured suggestions that a human manager can review, adjust, and use.

2.4 Large Language Models (LLMs)

Large language models are one of the most important recent developments in artificial intelligence. Most modern LLMs are based on the Transformer architecture, which improved the way models process language and relationships inside text (Vaswani et al., 2017). Unlike older natural language processing systems that were usually created for one specific task, LLMs are trained on very large amounts of text. Because of this, they can support many language-related tasks, such as summarization, classification, question answering, information extraction, and structured output generation.

Brown et al. showed that larger language models (Brown et al., 2020) can perform well in tasks even when they are given only instructions or a small number of examples. This is important for project management because software tasks are not always the same. A backlog may include bugs, features, infrastructure work, testing tasks, documentation, or refactoring. Creating a different traditional model for every type of task would be difficult and inefficient. LLMs provide a more flexible solution because they can interpret different types of text through prompting.

The main value of LLMs in software project management is that they can work with natural language. Backlog items are not always written in a standard way. Some are detailed, while others are short or unclear. Some explain the business impact, while others only imply it. An LLM can help interpret this information and turn it into more structured output. For example, a task about users not being able to log in can be understood as urgent, because it affects authentication and a core user flow. This kind of interpretation is useful because important project information is not always available in numeric fields.

LLMs can also support reporting and summarization. A manager often needs to collect many task updates and turn them into a clear sprint summary. This can take time, especially in larger projects. An LLM can help by summarizing completed work, blockers, risks, and possible next steps. It can also produce outputs in structured formats, such as tables, JSON, or predefined schemas. This is important for the proposed prototype, because priorities, suggested roles, deadlines, and rationales need to be displayed and evaluated in a consistent way.

However, LLMs also have limitations. Bender et al. warn that language models may reproduce patterns from training data without real grounded understanding (Bender et al.,

2021). They may also produce biased or incorrect outputs. Another important risk is hallucination. In project management, this could mean that the model invents dependencies, exaggerates urgency, or assumes information that is not included in the backlog. For this reason, the system must work within clear limits and should base its output only on the provided data and decision criteria.

Transparency is also important. A priority label is not enough by itself. The system should also explain why this label was selected, so that the manager can review the recommendation. LLMs are also sensitive to prompt design, because different instructions may lead to different outputs. In this dissertation, the prompt defines the role of the agent, the decision rules, the priority logic, and the expected output format.

For this reason, LLMs are not treated as autonomous managers in this dissertation. They are treated as reasoning components inside a controlled system. The model helps with interpretation, while the system around it provides structure, limits, and output validation. This distinction is important because the agentic behavior comes from the combination of the model, the prompt, the data pipeline, and the decision workflow.

2.5 *Agentic AI Systems*

Agentic AI systems are a step beyond simple automation tools. An AI agent can be understood as a system that receives information from its environment, processes it, and acts toward a specific goal (Wooldridge, 2009). This idea is not completely new, since agent theory existed before modern large language models. However, it has become more important today because LLMs can now be used as reasoning components inside agentic workflows (Yao et al., 2022).

The main difference between an agentic system and a traditional software tool is the way it works. A traditional tool usually waits for the user to give a command and then executes a fixed action. An agentic system can be given a goal, interpret the available context, decide what steps are needed, and then produce an output that supports this goal. In more advanced cases, such systems may also call external tools, retrieve information, update records, or interact with other platforms (Park et al., 2023).

In this dissertation, the agentic system is intentionally lightweight. It does not try to control a full project management platform or make final decisions on behalf of the manager.

Instead, it focuses on a specific planning activity. The system reads backlog data, interprets task descriptions, applies decision criteria, suggests priorities, recommends suitable roles, estimates deadlines, and creates a sprint summary. These actions are agentic because they are connected to a clear goal: supporting software task management.

Recent research on LLM-based agents also shows how reasoning and action can work together (Wang et al., 2024). For example, the ReAct framework proposed by Yao et al (Yao et al., 2022). combines reasoning steps with task-specific actions. This is relevant to project management because planning is not only about understanding information. It is also about organizing that information into actions that can be used in a real workflow. A useful AI assistant should not only describe the backlog but also help the manager understand what should happen next.

Agentic AI is useful in cases where traditional automation is too limited. A simple automation script can work well when the rules are clear, such as moving a ticket after a pull request is completed. However, it is less effective when the task requires interpretation, such as understanding the seriousness of a bug, identifying blockers, or summarizing project risks. In these cases, an LLM-based agent can be more flexible because it can process natural language and take context into account.

At the same time, agentic systems create risks if they are given too much freedom. In project management, an AI system that changes deadlines, assigns work, or communicates decisions without human approval could create confusion if its output is wrong. For this reason, the prototype in this dissertation is designed as a decision-support tool. It produces recommendations and summaries, but the final decision remains with the human manager.

This distinction between autonomy and control is very important. A system can still be agentic without being fully autonomous. In fact, a controlled agent may be more useful in an organization because it combines AI support with human oversight. The proposed prototype follows explicit decision rules, produces structured outputs, and gives short rationales for its suggestions. This makes its behavior easier to review and reduces the risk of uncontrolled action.

It is important to clarify that the prototype developed in this dissertation is not a fully autonomous AI agent in the strict technical sense. It does not independently plan and execute multi-step actions across external systems, maintain long-term memory, call tools

dynamically, or modify project-management platforms without human involvement. Instead, it is better understood as a prompt-engineered LLM workflow or AI-assisted reasoning pipeline that demonstrates limited agentic characteristics. The term "agentic" is therefore used in this dissertation to describe goal-oriented reasoning, contextual backlog interpretation, and structured planning support, rather than full operational autonomy.

2.6 Human-AI Collaboration

The use of AI in management should not be understood only as a question of automation. It is also a question of collaboration between humans and machines. In organizational settings, AI systems rarely operate in isolation. Their outputs are interpreted, accepted, rejected, modified, or challenged by human users. Therefore, the success of an AI system depends not only on technical capability but also on how effectively it supports human work.

Davenport and Kirby argue that the future of work should not be framed only as human replacement by machines, but also as augmentation (Davenport & Kirby, 2016), where technology helps humans work more effectively. This perspective is highly relevant to engineering management. The goal of the proposed AI agent is not to remove the engineering manager from the process. Rather, it is to reduce administrative burden, improve consistency, and provide structured decision support.

In various scenarios, people and artificial intelligence work together to manage projects. An AI system is often helpful to managers when it summarizes data, identifies potential problems, creates draft plans, or suggests which tasks are most important. It is then the responsibility of the human manager to assess those results - considering the specific organization, the way team members interact, what stakeholders expect, and long-term goals. On that account, the work is divided so that the AI handles data processing while the human addresses organizational politics, informal team knowledge plus ethical concerns.

To make this cooperation work, it is necessary for users to have a specific level of confidence in technology. If users lack confidence, they might ignore suggestions that are actually helpful. If they have too much confidence, they might accept errors without checking the work carefully. Because of this balance, it is a goal to support calibrated trust so that users are aware of what the system can and cannot do (Shneiderman, 2020). In the

prototype, there are rationales provided so that users can understand why a task has a certain priority or deadline (Amershi et al., 2019).

And because trust is necessary, the system must be explained. As Doshi Velez & Kim state, interpretability is a requirement (Doshi-Velez & Kim, 2017) when machine learning is used in areas where safety, fairness and the quality of decisions are important - but they also note that there is a need for more precise definitions of interpretability. In the field of project management, it is important to explain how the system works because its outputs change workloads, delivery schedules, but also what stakeholders expect. As a result, a task priority is not useful if it seems random - it is necessary for a clear explanation to be present.

By addressing the issues, we must also consider accountability. If an AI system recommends that a task is delayed and this action causes a loss for the business, there is a question about who is at fault. It could be the developer, the manager who agreed with the suggestion or the organization that used the software. As systems become more independent, those questions are more serious. To avoid the problems, the prototype is a support tool and is not the final decision maker. It provides suggestions but the people who use it are responsible for the final choices.

Any tool of usability is a factor that determines success - even if a system is technically advanced, it is not successful if it does not fit into regular work habits. It is unlikely that engineering managers will use a tool if they must prepare data manually for a long time or if the results are hard to read. The prototype is built with simple inputs as well as outputs. Backlog data is in a structured file, and the system shows priorities, roles, deadlines, reasons, and summaries. On a practical level, this simplicity makes it easier for individuals to use or test the tool.

There is also a cultural aspect to consider - it is possible that software teams will doubt AI suggestions if they think the technology is taking away their independence. Since agile teams often prefer to talk directly and organize themselves, an AI system must be introduced with care. It is better if the system is a help for team discussions instead of something that forces a decision. For instance, a priority list from the AI is a starting point for planning a sprint and is not a finished plan.

As those tools are used, the manager's job is likely to change. If AI is able to do regular analysis next to status reports, managers are free to spend less time on paperwork and more time on coaching, talking to stakeholders and making judgments - this is consistent with the idea of augmentation. In this model, the AI is responsible for tasks that are repetitive or involve a lot of data, while humans are responsible for tasks that involve people plus social context.

But it is not certain that using AI is always a good thing - if an AI system is designed poorly, it can cause confusion, hide how decisions are made or repeat existing prejudices. If the system always gives security tasks a low priority because they are not visible to the business, it is possible that long term risks will increase. If it creates deadlines that are impossible to meet, it is likely to put stress on the team. To prevent this, it is necessary to check the system for its relevance and how reasonable its suggestions are, rather than just checking if the output is complete.

In this dissertation, human–AI collaboration provides the governance frame for the prototype. The agent is not evaluated as an independent manager but as a tool that supports human managerial work. Its outputs must therefore be understandable, reviewable, and aligned with practical project management reasoning. This perspective is important because it prevents dissertation from making unrealistic claims about AI autonomy.

Theme	Main focus in the literature	Implication for this thesis
Software Project Management	Agile planning, backlog management, coordination, dependencies	Defines the project context and planning tasks to be supported
Decision-Making in Engineering Management	Bounded rationality, information overload, managerial judgment	Justifies the need for AI-assisted decision support
AI in Project Management	Forecasting, analytics, risk identification, decision support	Shows current AI use but also its limited practical scope
Large Language Models	Text interpretation, summarization, structured generation	Provides the technical basis for backlog analysis
Agentic AI Systems	Goal-oriented reasoning, multi-step support, constrained autonomy	Frames the prototype as a lightweight agentic planning assistant
Human-AI Collaboration	Trust, explainability, accountability, human oversight	Supports the human-in-the-loop design of the prototype
Research Gap	Limited evaluated prototypes for backlog planning in software projects	Motivates the design and evaluation of the proposed prototype

Figure 3: Literature Synthesis Matrix

2.7 Research Gap

The literature reviewed in this chapter shows that software project management still depends on many human decisions. Agile methods and tools such as Scrum help teams organize their work, but they do not remove the need for judgment. Managers and teams still have to prioritize backlog items, plan sprints, assign tasks, and report progress. These activities are not simple mechanical steps, because they require context, discussion, and comparison between different options. The literature on decision-making also shows that managers do not always have complete information. In many cases, project data is spread across different tools and written in an unstructured way. This creates a need for systems that can help managers process information and make decision criteria clearer.

In recent studies about artificial intelligence in project management, experts find that machine learning and predictive models are helpful when people forecast outcomes, identify risks or control projects - but those systems are often passive because they require data that is organized in specific formats. It is possible that a system predicts a delay or indicates a risk, but it does not always assist a person who must decide on the next step for a plan. Large language models are different because they can read items in a backlog, descriptions of

issues, and notes from a sprint. As the models process information, they summarize text and generate outputs that are structured. At the same time those models are risky because they sometimes generate false information, produce results that are not the same every time, or operate in a way that people cannot see. Due to these factors, users might trust the technology more than is safe. On that account, it is necessary that individuals set clear limits for artificial intelligence in project management and provide oversight.

This is the research gap that this dissertation addresses. Existing literature discusses Agile project management, AI analytics, LLMs, and agentic systems, but there is less practical work showing how these ideas can be combined into a small, tested prototype for software backlog management. This dissertation therefore designs and evaluates a lightweight Agentic AI prototype focused on task management. The system does not try to automate an entire project. Instead, it examines whether AI can read backlog items, prioritize work, suggest roles, estimate deadlines, and create sprint summaries in a way that supports managers. This contribution is both technical and managerial, because it shows how an LLM-based agent can be structured, while also examining how its output can support decision-making without removing human control.

3 Research Methodology

This chapter presents the research methodology followed for the design, implementation, and evaluation of the AI-assisted backlog planning prototype developed in this dissertation. The purpose of the methodology is to explain how the research problem was approached, how the system was designed, what type of data was selected for testing, how the experimental setup was organized, and which criteria were used to evaluate the usefulness and limitations of the proposed solution.

The dissertation investigates the use of Generative AI in software project management, with a particular focus on backlog analysis, task prioritization, role recommendation, deadline estimation, sprint planning, and risk identification. These activities are highly relevant to Project Managers and Engineering Managers because they directly affect planning quality, delivery predictability, team coordination, and decision-making under uncertainty.

The implemented system is a lightweight prototype named **Agentic AI Backlog Planner**. The current implementation is not a fully autonomous project management agent. Instead,

it is a prompt-driven AI decision-support tool that accepts a backlog in CSV format and produces structured planning recommendations. The system uses a Streamlit interface, Python-based backend logic, an OpenAI model, carefully designed project-management prompts, and structured Pydantic output models. The final output is displayed to the user and exported as CSV and JSON files.

The methodological approach is based on the development and evaluation of a working prototype. This approach was selected because the research question is practical and application oriented. The dissertation does not only discuss Generative AI theoretically; it also examines whether such a system can be implemented and whether its output can support realistic software planning activities. Therefore, the research methodology combines system design, prototype development, controlled testing, and qualitative evaluation of the generated recommendations.

3.1 Research Approach

The research follows an applied, design-oriented approach (Hevner et al., 2004) . The central objective is not to test a purely theoretical model, but to design and evaluate a functional artifact that demonstrates the potential of Generative AI in software project management. This makes the research closer to a design science perspective, where knowledge is produced through the creation and assessment of an artifact.

The artifact in this dissertation is the AI-assisted backlog planning prototype. The prototype is used to explore how an AI model can transform semi-structured backlog data into a more organized planning output. The system takes as input a list of software development tasks and returns priorities, suggested owner roles, deadline estimates, rationales, blockers, risks, a sprint goal, and a sprint summary.

The research is also exploratory in nature. Generative AI and agentic AI systems are still developing rapidly, and their use in software project management is an emerging topic. For this reason, the study does not attempt to prove that AI can replace human project managers. Instead, it explores how AI can assist them in specific planning activities. The emphasis is placed on decision support, not decision replacement.

In this description, the research approach follows a specific sequence. For the first step, a software prototype was designed and implemented that is ready for use. To test the tool the team applies it to data from a backlog that contains typical features of the field. And when the results are assessed, standards that relate to how people manage software projects are used. On the next stage, the findings are the basis for a discussion about where AI-supported planning systems are helpful or where they face constraints. With those results, it is also possible to describe how the systems might improve in the future.

3.2 Research Design

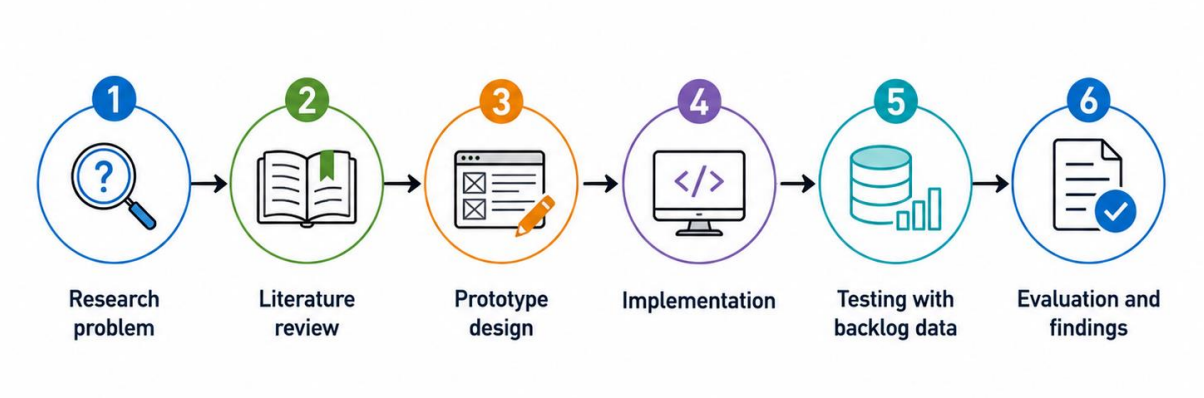

Figure 4: Research Methodology

The research design is organized around the development and assessment of a Minimum Viable Product, following the logic of design science research in which an artifact is built, demonstrated, and evaluated (Hevner et al., 2004). The Minimum Viable Product (MVP) approach is used for this research because this method ensures that the dissertation concentrates on the primary features that are necessary to show the core idea. By using this strategy, it is avoided adding technical details that are not needed. If a project management platform is built for full professional use, it must have features where people manage user accounts and verify identities. To be functional in that context, the system is also required to connect with various other software tools and apply rules for what different users can see. And it is necessary for the software to save data permanently plus function in a live environment. These features would increase the scope significantly without necessarily improving the core research contribution.

The development team is working on a Minimum Viable Product that addresses the planning of a backlog. It is common for teams to use this specific process when they develop software through agile methods. Because this task requires people to make technical choices, manage resources and rank tasks by importance, it is a useful example for the project. In a typical backlog, the items include text that describes features, notes on how tasks rely on each other, data about financial worth, levels of time sensitivity and predictions of how difficult the work is. As those inputs consist of written information, a Generative AI system is able to process them. With this data the model is able to understand the language and create recommendations in a structured format.

The research design consists of five main stages. The first stage is the identification of the planning problem and the definition of the research scope. The second stage is the design of the system architecture and the prompt strategy. The third stage is the implementation of the prototype. The fourth stage is the preparation of test data, using synthetic backlog data and the possibility of publicly available GitHub issue data. The fifth stage is the evaluation of the system output according to functional, managerial, and usability criteria.

This design supports both technical and managerial analysis. From a technical point of view, the dissertation examines whether such a prototype can be implemented with a lightweight architecture. From a managerial point of view, it examines whether the generated output is meaningful for project planning and decision support.

3.3 System Design Principles

The developers built the system according to principles that align with the technical goals of the dissertation and the managerial goals of the prototype.

As a first principle, the system is simple - it is necessary for users to operate the software without difficulty, show its functions clearly and describe its logic easily. In this case, the code uses a limited set of technologies - Python, Streamlit, pandas, Pydantic, dotenv plus the OpenAI API. By choosing those tools, the system avoids complex infrastructure and keeps the focus on how the planning workflow operates.

On the second principle, the system creates structured data - if a manager needs to review, compare, export or reuse recommendations for a project, text that lacks a specific format is not sufficient. In this implementation, the software uses predefined models to organize task

recommendations, sprint plans and the analysis of the backlog. With the models, the system converts the response from the AI into tables but also files.

To follow the third principle, the system requires humans to make decisions. It is not the purpose of the software to choose project actions automatically. It generates a draft of a plan for a manager to check. If a manager makes a decision, they must consider facts that are not in the backlog file, like the capacity of the team, what stakeholders expect, the dates of deadlines, technical limits, organizational dynamics, and what the business prioritizes.

For the fourth principle, the system shows its reasoning - there is a brief explanation for each task that the system recommends. It is important for a manager to know why the system prioritizes a task, suggests a specific role, or identifies risks and blockers. And while the explanation is not a guarantee that the data is correct, it helps a user to judge the recommendation.

By the fifth principle, the system has limited autonomy - this version of the prototype is restricted by design. It does not create GitHub Issues, change project boards, assign users or update other systems. Due to those limits, the risk is lower as well as the system is appropriate for an academic study. If the system were to act with more independence, it would be a future update and is not part of the current product.

In the sixth principle, the system allows for reproducibility. It includes a backlog for testing and saves results in standard formats. If a researcher uses the files, they can repeat the experiment many times or compare what the system generates.

3.4 Prototype Development Methodology

This iterative approach is consistent with design science methodology in software engineering, where practical artifacts are refined through cycles of design, implementation, and evaluation (Wieringa, 2014). The initial focus was to create the minimum functionality required to test the research idea: reading a backlog, sending it to an AI model, and receiving planning recommendations. Once this basic flow was established, additional functionality was added to make the prototype more usable and suitable for demonstration.

The first implementation step was the creation of the backlog input format. CSV was selected because it is simple, widely understood, and easy to edit using tools such as Excel

or Google Sheets. It also allows the prototype to remain independent from a specific project management platform.

The second step was the creation of the AI analysis module. This module is responsible for sending the backlog to the OpenAI model and requesting a structured response. The system prompt was designed to instruct the model to act as a software project management assistant. The prompt includes rules for prioritization, role assignment, deadline estimation, dependency handling, and risk identification.

The third step was the definition of structured output models. The system uses Pydantic models to define the expected response format. This ensures that the AI output includes specific fields, such as priority, suggested owner role, rationale, blockers, and risks. The use of structured output is important because it allows the system to process the response reliably instead of depending on manually interpreted text.

The fourth step was the development of the Streamlit interface. The interface allows the user to upload a backlog, view the input, run the analysis, and inspect the generated recommendations. It also provides download buttons for the generated CSV and JSON files.

The fifth step was the creation of a sample backlog and output files. The sample backlog contains realistic software development tasks related to authentication, registration, testing, security, deployment, and documentation. This data allows the prototype to be tested even when no external backlog is available.

3.5 Data Sources

In this dissertation, two categories of data were used which are synthetic backlog data and publicly available GitHub data.

There is a synthetic backlog file in the current prototype - to represent a software development scenario that occurs in practice; this backlog was created manually. It includes tasks where developers implement a login API, create a login UI page, set up a database schema, create a CI/CD pipeline, perform a security audit, deploy staging, and write end-to-end tests. As those tasks occur frequently in software engineering projects plus have dependencies that are significant, they are in the file.

By using synthetic backlog data, it is easy to maintains control over the testing scenario. It is possible to define the number of tasks, how complex they are, the dependencies, how

urgent they are, and indicators of business value. Due to these factors, it is easy to evaluate if the AI system follows a logical process. If a database schema blocks a registration API and that API blocks a registration UI, the AI should identify that developers must prioritize backend work before frontend work.

On GitHub, the data is relevant to the methodology because repositories contain issues, pull requests, labels, milestones, discussions, but also activity from software projects. To test the prototype against software project examples that are not controlled, this date is used. For the prototype, public GitHub data is useful because it provides descriptions of tasks, reports of bugs, requests for features, tasks for documentation, and discussions about technical subjects.

The reason for selecting publicly available GitHub data is also ethical and practical. Public repositories can be studied without exposing confidential corporate information. This is particularly important for a dissertation, because proprietary company backlogs may contain sensitive information, internal priorities, security issues, customer names, employee data, or business strategy. By using public GitHub data, the research avoids these privacy and confidentiality risks.

At the same time, public GitHub data has limitations. Public issues may be incomplete, inconsistent, or not written in a format suitable for direct input into the prototype. Some issues may lack explicit business value, urgency, or complexity estimates. Therefore, GitHub data may require preprocessing before it can be used in the same format as the synthetic CSV backlog.

For this reason, the synthetic backlog is used as the primary controlled test dataset, while public GitHub data is treated as a suitable source for additional testing and future evaluation.

3.6 Rationale for Using Synthetic Backlogs

Synthetic backlogs are used to test the system under controlled conditions. In software projects that exist in reality, items in a backlog are often inconsistent or incomplete. It is common for the organizational context to affect those items in ways that the data does not show. While the conditions are realistic, they make the initial evaluation of a system difficult.

By using a synthetic backlog, clearer dependencies are defined, and the behavior of the planning is specified. As an example, one can design tasks for authentication so that the login API restricts the login UI. And the database schema restricts user registration, while the CI/CD pipeline restricts the deployment to staging. It is then possible to evaluate if the AI recommendations follow a logic for planning that is reasonable.

With synthetic data, it is also safer to present and publish the prototype. Because the backlog does not include information from an actual company, there is no risk that the program will expose data that is confidential - this is relevant because the dissertation is for academic purposes and others will review, share or archive the work.

And another advantage is that the process is repeatable - to compare the outputs of the model; one uses the same synthetic backlog multiple times. If the user modifies the prompt, they reuse the same input to observe how the recommendations change.

But synthetic data has limitations - it is often more structured and has fewer errors than data from an actual project. On a real backlog, descriptions are often vague, and issues are duplicated. Priorities are frequently outdated; dependencies are missing, and the criteria for acceptance are not clear. Due to those factors, synthetic backlogs are useful for testing that is controlled, but they are not a substitute that is complete for data from the real world.

3.7 Experimental Setup

The experimental setup is designed to evaluate whether the prototype can successfully transform backlog data into structured planning recommendations.

The system is executed locally. The user starts the Streamlit application, uploads a backlog CSV or uses the included sample backlog, selects the model, enters a project name, and runs the analysis. The application then sends the backlog to the AI model and receives a structured response. The result is displayed in the interface and saved as CSV and JSON output files.

The main experiment uses the synthetic sample backlog. This backlog contains multiple task types, including backend development, frontend development, testing, DevOps, security, documentation, and performance optimization. It also contains dependencies between tasks, which allows the system's ability to identify blockers and prioritize foundational work to be evaluated.

The experimental process can be described in five stages. First, the backlog data is prepared in CSV format. Second, the backlog is loaded into the application. Third, the AI model analyzes the backlog using the system prompt and structured output schema. Fourth, the recommendations are displayed and exported. Fifth, the output is reviewed according to the evaluation criteria.

The experiment does not attempt to measure productivity improvement in a live software team. It is an evaluation of if the prototype creates suggestions that are logical, organized and helpful for people who manage projects in a setting that is controlled. For the scope of the dissertation, this approach is appropriate because the goal is to show that the system can function and to assess how well the prototype plans tasks - but the objective is not to put the system into use within a live environment where software is produced.

3.8 Evaluation Criteria

The evaluation criteria are designed to assess both the technical behavior of the system and the managerial usefulness of its output.

The first criterion is functional correctness. The system should successfully load a CSV backlog, display the input data, send the backlog to the AI model, receive a structured response, display the recommendations, and export the output files. If the application cannot complete this workflow, it cannot be considered a successful prototype.

The second criterion is output completeness. For every backlog item, the system should return a priority, suggested owner role, deadline estimate, rationale, blockers, and risks. At sprint level, it should return a sprint goal, summary, top priorities, major blockers, and major risks.

The third criterion is planning relevance. The recommendations should make sense from a software project management perspective. Tasks with high business value, high urgency, critical user impact, security relevance, or blocking dependencies should generally receive higher priority.

The fourth criterion is dependency awareness. The system should identify when one task blocks another. For example, a frontend login page depends on the login API, and deployment to staging depends on the CI/CD pipeline. Dependency awareness is important because poor sequencing can delay delivery.

The fifth criterion is risk identification. The system should identify meaningful risks, such as security issues, integration problems, environment configuration failures, test fragility, or late discovery of vulnerabilities. Risk identification is important because managers need to anticipate problems before they affect delivery.

The sixth criterion is role suitability. The suggested owner role should be appropriate for the task. Backend tasks should generally be assigned to backend engineers, UI tasks to frontend engineers, test-related tasks to QA engineers, pipeline work to DevOps engineers, and security reviews to security engineers.

The seventh criterion is explainability. Each recommendation should include a short rationale that helps the manager understand why the recommendation was made. This is important for trust and human review.

The eighth criterion is usability. The system should be easy to operate. The user should be able to upload a backlog, run the analysis, understand the results, and download the output without needing advanced technical knowledge.

The ninth criterion is practically useful. The output should be useful as a planning draft. It does not need to be perfect, but it should help the manager start a planning discussion more quickly and with better structure.

3.9 Human Evaluation of the Output

In this process, people assess the quality of the results - it is suitable to use this method because the system generates suggestions for how to manage projects. As there is often more than one correct way to rank items or assign tasks, different managers might organize work in various ways - those differences depend on the strategy of the business, the capacity of the team, the promises made to customers or the risks related to technology.

For those reasons, the assessment checks if the results are logical, and does not contradict themselves and helps the user. To do this, the system compares the generated priorities with the information that exists in the backlog. It also reviews if the suggested roles fit the tasks. By checking the blockers and risks, people ensure that the points relate to the project. On the final step, the evaluation confirms if the summary of the sprint provides a clear direction for planning.

Due to those factors, human judgment is a component of the method - this aligns with how individuals intend to use the system. It is not a requirement for the AI assistant to create final decisions without help - but the system is expected to assist the human manager when it creates a first draft that follows a specific structure.

3.10 Ethical and Practical Considerations

In this methodology, data from private companies is not used. It is a decision based on ethics because real project backlogs often include information that is restricted or private. As an example, a backlog can show the strategy for a product, problems faced by customers, vulnerabilities in security, dates for internal deadlines, tasks assigned to employees, or business priorities that are not public.

By using synthetic data and data from GitHub that is open to the public, the risks are lowered. Synthetic data is free of confidential details besides GitHub data exists already in the public domain - but it is necessary to manage even public data with care. To protect privacy, the research does not display personal information when it is not required. And the analysis describes the characteristics of project management instead of focusing on the people who contribute.

As another part of the ethical framework, the way that AI functions during the process of making decisions is addressed. It is not accurate to describe the system as a substitute for the judgment of a human. Because recommendations from AI are sometimes incomplete, biased, or wrong, the prototype is built as a tool that requires a human in the loop. The manager is the person who must review and approve the final plan.

There is also a limitation in practice that relates to how the model behaves. If inputs are similar, generative AI models can still create different outputs. Because recommendations change based on the words in a prompt, the version of the model and how good the input is, the system is a tool for decision support. It is not a planning engine that produces a single, fixed result.

3.11 Summary

This chapter described the research methodology used in the dissertation. The research follows an applied, design-oriented approach based on the development and evaluation of a working AI-assisted backlog planning prototype. The methodology combines system

design, prototype implementation, controlled testing with synthetic backlog data, and the use of publicly available GitHub data as a realistic and ethically appropriate source for additional testing.

The system was designed according to principles of simplicity, structured output, human-in-the-loop decision-making, transparency, bounded autonomy, and reproducibility. The experimental setup evaluates whether the system can transform backlog data into useful planning recommendations. The evaluation criteria include functional correctness, output completeness, planning relevance, dependency awareness, risk identification, role suitability, explainability, usability, and practical usefulness.

Overall, the methodology supports the central objective of the dissertation: to investigate how Generative AI can assist Project Managers and Engineering Managers in software development planning. The prototype demonstrates that AI can provide structured planning support, while the methodology also recognizes the need for human oversight, ethical data selection, and careful interpretation of AI-generated recommendations.

4 System Implementation

This chapter presents the implementation of the AI-assisted backlog planning prototype developed as part of this dissertation. The purpose of the system is to demonstrate how Generative AI can support software project management activities, especially during backlog review, prioritization, task assignment, deadline estimation, and sprint planning. The implementation was designed as a practical Minimum Viable Product rather than a complete enterprise project-management platform. This approach allowed the development effort to remain focused on the main research objective: examining how an AI-based assistant can help a Project Manager or Engineering Manager transform a raw backlog into a structured planning recommendation.

The implemented system is a Python-based web application built with Streamlit. The source code of the prototype is publicly available in the project's GitHub repository (Kantas, 2026). It allows the user to upload a backlog file in CSV format, review the input tasks, run the AI analysis, and receive structured recommendations. These recommendations include task priorities, suggested owner roles, estimated deadlines in working days, rationales, blockers, risks, a recommended sprint goal, and a sprint-level planning summary. The system also

provides downloadable output files in CSV and JSON format, making the results usable outside the application.

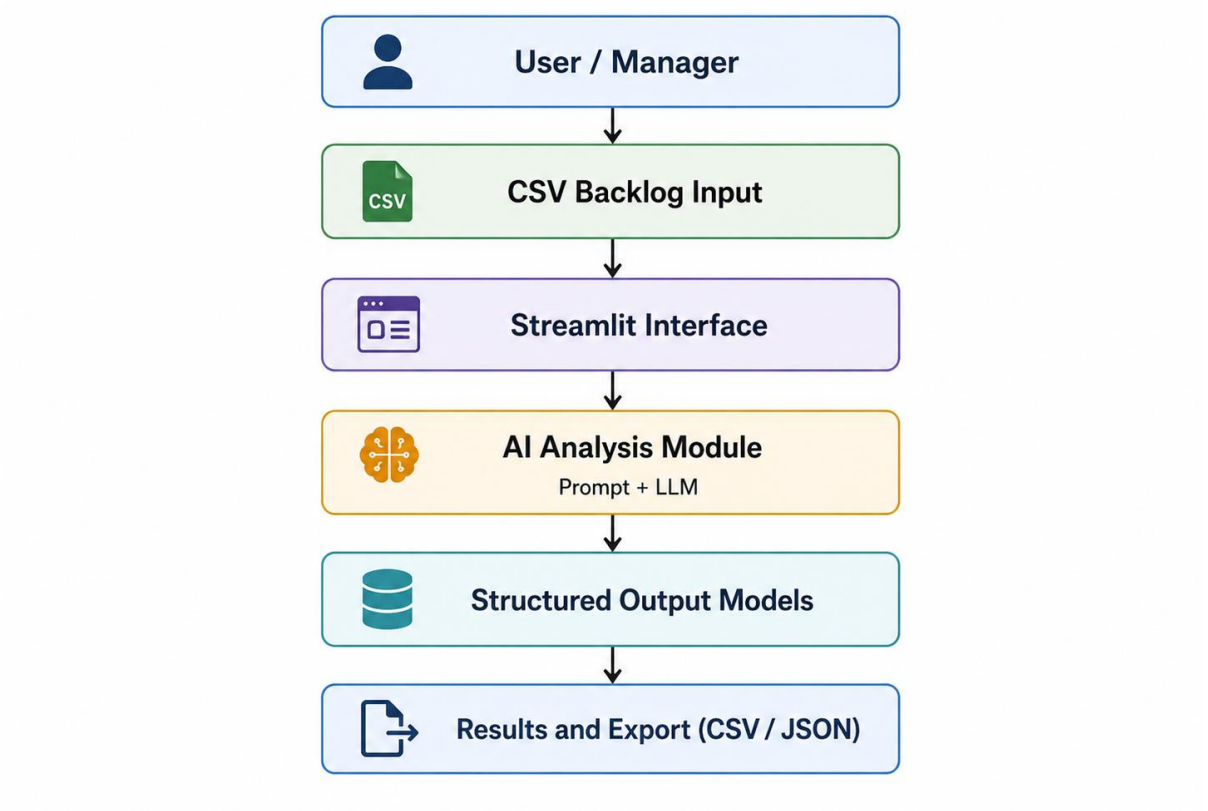

Figure 5: System Architecture of the Prototype

Although the application carries the name "Agentic AI Backlog Planner", the implemented system should be understood primarily as an AI-assisted reasoning pipeline. It uses a carefully designed prompt, structured input data, and predefined output models to generate planning recommendations. Therefore, its agentic character is limited and bounded. The system does not independently execute actions, maintain long-term memory, call external tools dynamically, or update project-management platforms. Instead, it demonstrates how LLM-based reasoning can support backlog interpretation, prioritization, role suggestion, deadline estimation, and sprint summarization under human supervision. By using an artificial intelligence model, the system examines information from a backlog and produces a result for planning that has clear boundaries - but the system is not able to complete sequences of actions on its own, like when it would need to change GitHub Issues, adjust a sprint board, contact external tools or keep information in its memory between separate

sessions - this difference is significant so that the limits of the current version are precise plus the capabilities are not described as greater than they are.

As a tool, the system is useful because it is easy to understand, shows its processes clearly, and relates directly to the work of managing software. It is not an attempt to take the place of a person who manages projects. It assists a manager when it creates an initial version of a plan. For the user, this plan is something they can examine, fix and use when they talk about the project.

4.1 Technology Stack

The development of the prototype happened using a set of technologies that are easy to use. Python is the primary language because developers use it often for tasks that involve artificial intelligence, data manipulation, scripts, and the creation of early models. In addition, Python contains established sets of code for communication with external interfaces, organized tables, settings for the local environment, and the verification of data formats.

To create the part of the system that users see Streamlit was used. It is helpful because it allows for the construction of interactive pages without the need for complex code for the visual layout. For this academic work, this choice is relevant because the core objective is the process for planning with artificial intelligence rather than the construction of visual interfaces. With Streamlit, the program can show the data from the backlog, provide settings for the user, execute the evaluation by the model plus display the findings in a web browser.

Pandas is the tool that performs the reading and management of files in the comma separated values format. Since the data in the backlog exists in rows and columns, Pandas is a functional option to load the file but also change the suggested tasks into a grid. By using this tool, the results are also simple to save in other formats or to review.

Pydantic is the library that establishes the specific format for the data that the model returns - this is a significant technical decision for the project. Instead of the system receiving only unorganized words, precise templates were created for the suggestions and evaluation of the development cycles. Because of this structure, the results are more consistent, and the computer can process them with more accuracy.

To send as well as receive data from the model, the system uses the library from OpenAI. When the program runs, it transmits the backlog and the instructions for project management to the model so that it can ask for a response in a set format. And to protect the secret access code, the system uses python dotenv to retrieve it from a separate file on the computer. By doing this the code does not contain sensitive information directly.

On the whole the selection of those tools ensures that the prototype stays small, clear and easy to run on one computer - this is suitable for the requirements of a thesis because the purpose is to show that the idea works or to display the process rather than to create a system for many users.

4.2 Repository Structure

The implementation is organized into a small number of files, each with a clear responsibility. The main user interface is implemented in app.py. This file contains the Streamlit application, including the page configuration, sidebar controls, file upload logic, input backlog display, execution button, results display, and download buttons.

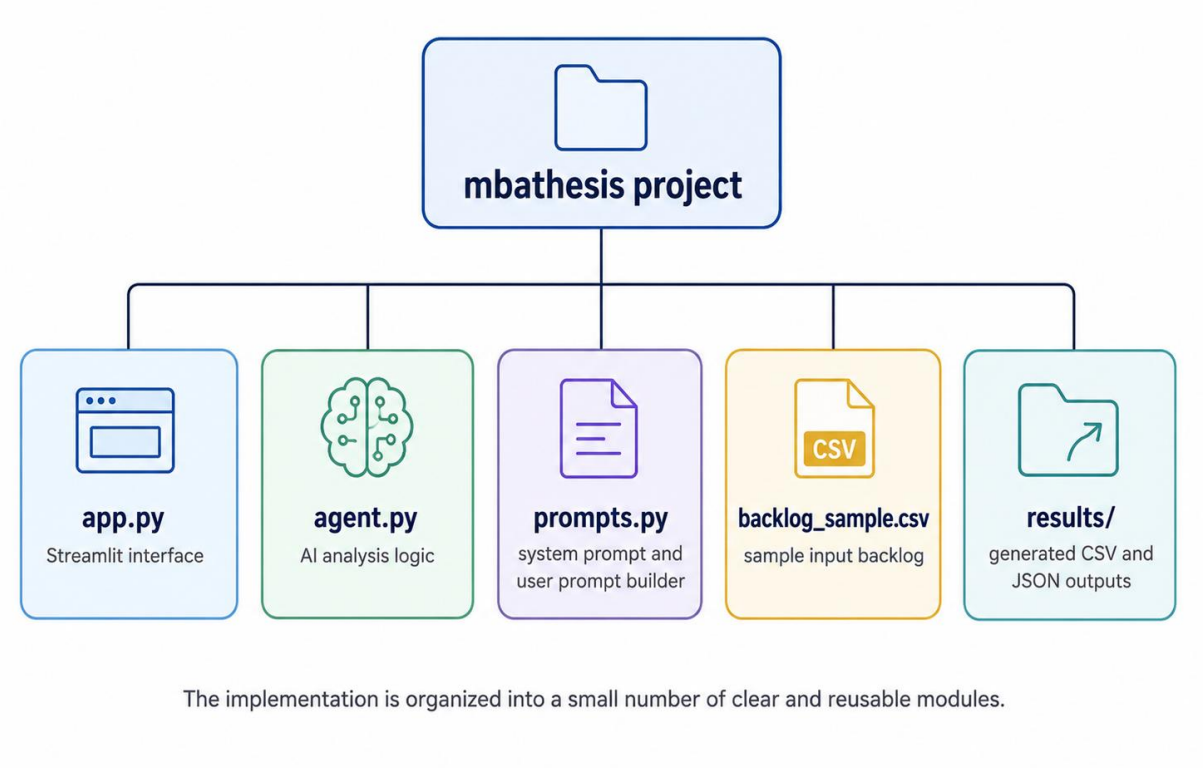

Figure 6: Repository Structure Overview

The main AI logic is implemented in agent.py. This file contains the structured output models, the BacklogAgent class, the method that sends the backlog to the AI model, the function that converts the result into a DataFrame, and the function that saves the output files. In practice, agent.py acts as the backend logic of the application.

The file prompts.py contains the prompt used to guide the AI model. It includes the system prompt, which defines the behavior of the AI assistant, and a helper function that builds the user prompt by inserting the backlog of CSV text. This separation makes the implementation easier to maintain because the prompt can be improved without changing the rest of the application logic.

The file backlog_sample.csv contains the default backlog used by the application when the user does not upload a custom file. This sample backlog includes tasks from a realistic software development scenario, mainly around authentication, registration, testing, deployment, security, and documentation. The sample data is important because it allows the application to be demonstrated immediately after setup.

The results folder contains generated output files. The JSON output preserves the complete structured result, including the sprint-level analysis. The CSV output contains the task-level recommendations in a tabular format, which can be opened in spreadsheet software or included in a report.

This simple repository structure supports the objective of the prototype. It keeps the project easy to understand, while still separating the interface, AI logic, prompt design, input data, and output data.

4.3 User Interface Implementation

The user interface was implemented in Streamlit through the app.py file. When the application starts, it loads environment variables, configures the page, and displays the main title, "Agentic AI Backlog Planner". The page caption explains that the application is a lightweight thesis MVP for backlog prioritization, assignment suggestions, and sprint summaries.

The sidebar contains the main configuration options. The user can select the model, enter the project name, and upload a backlog CSV file. This design keeps the configuration separate from the main results area and makes the application easier to use. The model selection option allows the user to choose between a faster, lower-cost model and a stronger reasoning model. The project name field is used to label the analysis and make the generated output more specific.

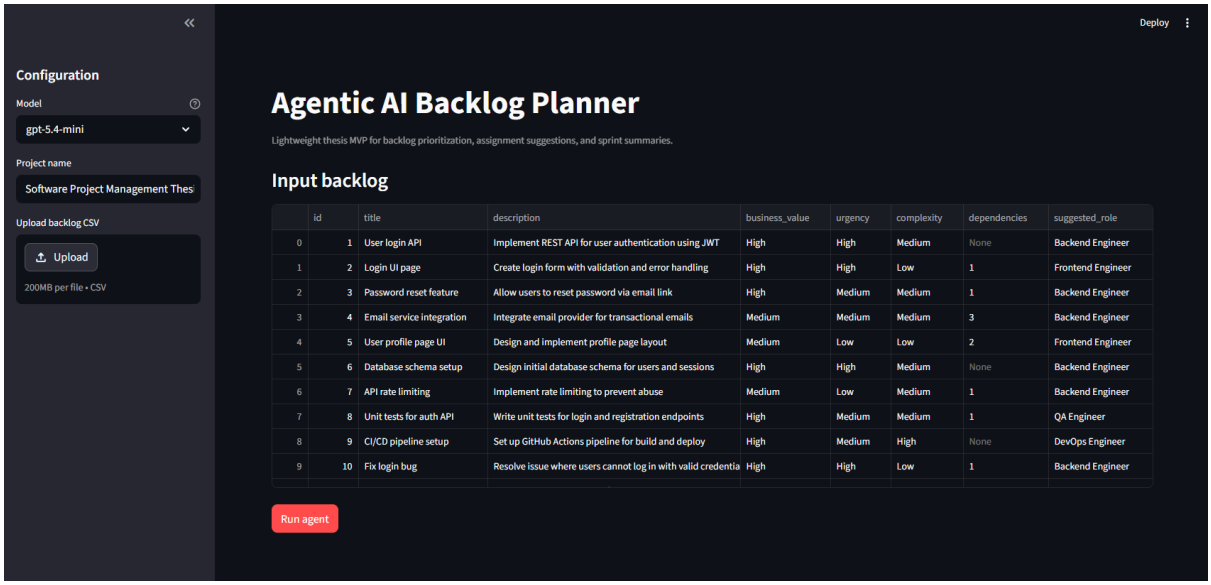

Figure 7: Agentic AI Backlog Planner UI

The file upload component accepts CSV files. If the user uploads a file, the application reads that file with pandas. If the user does not upload a file, the system automatically loads the included sample backlog. This fallback behavior is useful because it ensures that the application can always be demonstrated, even when no external input file is available.

After the backlog is loaded, it is displayed in the main area of the application as an input table. This step is important from a usability perspective because it allows the user to inspect the backlog before running the AI analysis. In a real project environment, this would give the manager an opportunity to confirm that the input data is correct.

The analysis is triggered by the "Run agent" button. When the button is clicked, the application creates an instance of BacklogAgent, converts the backlog DataFrame into CSV text, sends it to the AI model, receives the structured analysis, converts the task recommendations into a table, and saves the output files. If the process completes successfully, the application displays a success message.

The output is displayed in two main sections. The first section presents the recommended task plan in a table. This table includes the task ID, title, priority, suggested owner role, deadline estimate, rationale, blockers, and key risks. The second section presents the sprint-level recommendations, including the sprint goal, sprint summary, top priorities, major blockers, and major risks.

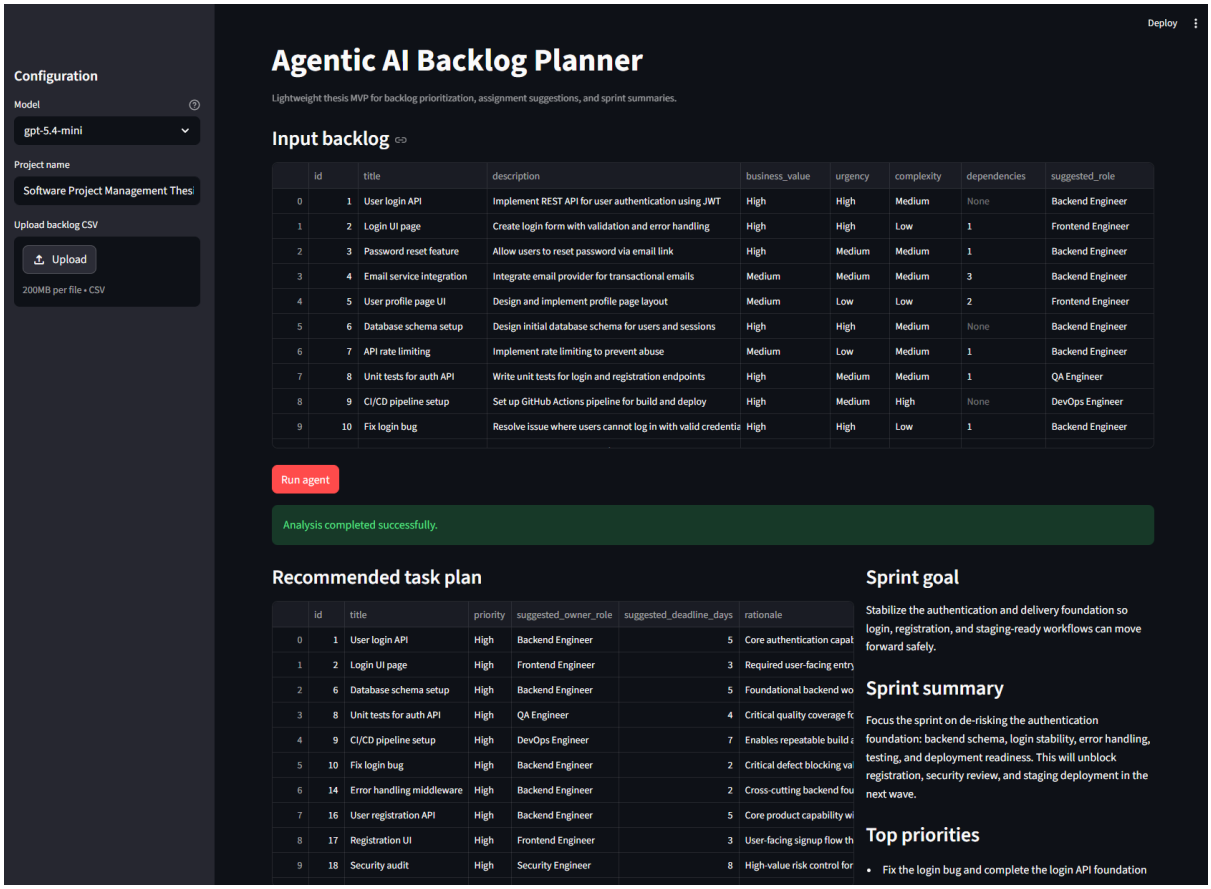

Figure 8: Agentic AI Backlog Planner: Successfull Analysis

Finally, the application provides download buttons for both CSV and JSON outputs. This is a useful feature because the generated recommendations do not remain limited to the web interface. They can be saved, reviewed, shared, or included in the thesis evaluation.

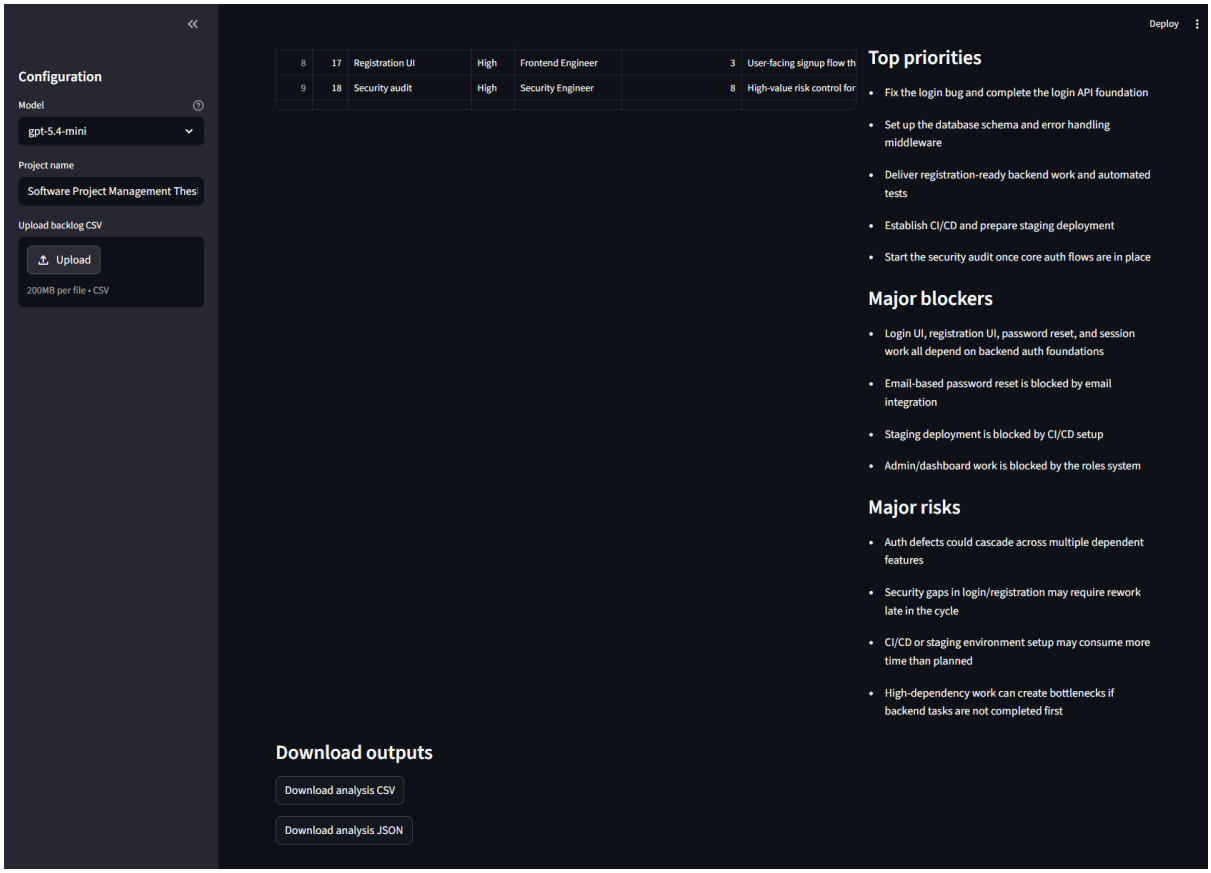

Figure 9: Agentic AI Backlog Planner: Results

4.4 AI Planning Logic

The `BacklogAgent` class contains the primary logic for planning. It connects the application to the OpenAI model and it provides a structured analysis of the backlog.

When an instance of `BacklogAgent` is created, it checks if the `OPENAI_API_KEY` environment variable exists. If the key is not present, the system provides a message that describes the error. For the application to function, access to the OpenAI API is required. With this check the process is more transparent to the user because the message describes the specific problem.

The `analyze_backlog_text` method is the primary function of the class. It accepts the backlog in a comma separated format and it also accepts the project name. By combining the user instructions with the system instructions, the method sends a complete request to the model. To ensure the response is formatted correctly, the method requests data that follows the `BacklogAnalysis` structure.

In this process the model does not produce unstructured text. It returns data that adheres to a specific schema. Because of this design, the output is easy to verify and use in further steps. The application is more consistent as a result, since the other parts of the code can rely on the presence of specific data fields.

There is also a method named `analyze_backlog_file` - this method retrieves a file from storage and it then executes the `analyze_backlog_text` method. By using this structure, the logic remains functional without the Streamlit interface. As an example a user can run the agent from a command line interface with a sample file.

And so the code separates the analysis logic from the parts that the user sees. Because of this organization, the same logic is available for use in different contexts, like a web service or a project management tool.

4.5 Structured Output Models

A central part of the implementation is the use of structured output models. These models are defined with Pydantic and describe the exact format that the AI response must follow.

The first model is TaskRecommendation. It represents the recommendation for a single backlog item. Each task recommendation includes the task ID, task title, recommended priority, suggested owner role, suggested deadline in working days, a short rationale, blockers, and key risks. The priority field is restricted to three possible values: High, Medium, and Low. This restriction makes the output more consistent and easier to compare.

The second model is SprintPlan. This model represents the overall sprint-level recommendation. It includes a concise summary, a list of top priorities, a list of major blockers, and a list of major risks. This allows the system to move beyond individual task analysis and provide a more managerial view of the backlog.

The third model is BacklogAnalysis. This is the top-level output model. It includes the project name, the recommended sprint goal, the list of task recommendations, and the sprint plan. In this way, all generated planning information is grouped into a single structured object.

Using structured models is important because it reduces the uncertainty normally associated with AI-generated text. In a typical prompt-only interaction, the model might return a long

paragraph, a bullet list, or an inconsistent table. In this implementation, the model is guided to return data that can be parsed, displayed, sorted, and exported.

This approach is particularly useful for software project management because the output must be actionable. A Project Manager needs clear task priorities, ownership suggestions, deadlines, blockers, and risks. The structured schema helps transform the AI response into something closer to a practical planning artifact.

4.6 Prompt Implementation

The prompt is implemented in the prompts.py file. The system prompt defines the role and expected behavior of the AI assistant. It instructs the model to act as an AI assistant for software project management and to produce planning outputs for an engineering manager.

The prompt gives the model a clear set of responsibilities. These include interpreting backlog items, assigning priority levels, suggesting suitable owner roles, estimating deadlines, explaining the rationale behind each decision, producing a sprint summary, and identifying blockers, dependencies, and risks.

The decision policy in the prompt is also important. The model is instructed to consider business value, urgency, complexity, dependencies, and task description together. This reflects how real backlog planning works. A task should not be prioritized only because it has high business value, or only because it is urgent. The manager must consider several factors at the same time.

Figure 10: Backlog Planning Pipeline

The prompt also includes specific guidance for software projects. For example, it states that bug fixes affecting authentication, access, payment, security, or core user flows should

generally be prioritized highly. This is a practical rule because such issues often affect user trust, system reliability, and delivery risk.

The priority rubric defines what High, Medium, and Low priority mean. High priority is associated with strong business impact, urgency, blocking dependencies, critical bugs, security relevance, or enablement of several downstream tasks. Medium priority is used for important but not immediately blocking tasks. Low priority is used for useful but non-critical work, polish, documentation, or tasks with low urgency.

The deadline guidance gives the model a simple estimation framework. Low-complexity tasks are usually estimated at one to three working days, medium-complexity tasks at three to five working days, and high-complexity tasks at five to ten working days. This does not replace professional estimation techniques, but it helps the AI produce more consistent and realistic suggestions.

4.7 Data Processing and Export

After the model returns the structured analysis, the application converts the task recommendations into a pandas DataFrame. This is done by the analysis_to_dataframe function. The function loops through the list of task recommendations and creates one row for each task.

The generated table includes the most important planning fields: ID, title, priority, suggested owner role, suggested deadline, rationale, blockers, and key risks. Blockers and risks are converted into text fields so that they can be displayed clearly in the Streamlit table and saved in CSV format.

The function also sorts the tasks by priority. High-priority tasks appear first, followed by medium-priority and low-priority tasks. This is a small but useful implementation detail because it makes the output more immediately useful to a manager. In backlog planning, the most important information is usually what should be handled first.

The save_outputs function writes the results to the results folder. It saves the full structured analysis as JSON and the task-level recommendations as CSV. These two formats serve different purposes. The JSON file preserves the complete structure of the AI output, including the sprint-level plan. The CSV file is easier to open in spreadsheet tools and can be used for further review, filtering, or reporting.

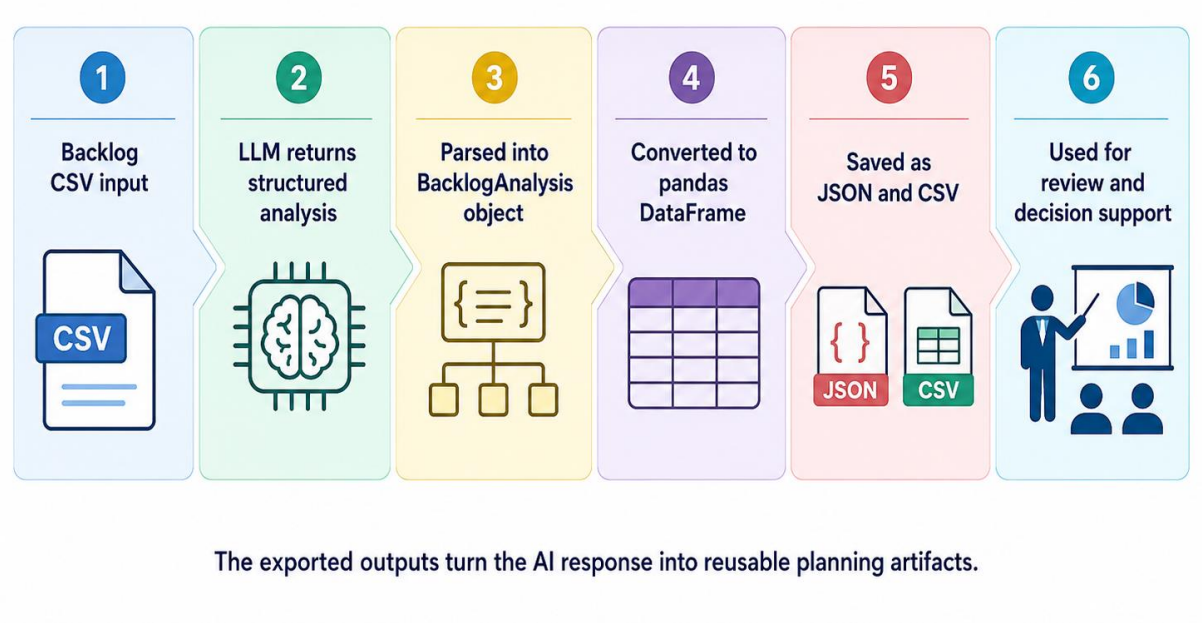

Figure 11: Output Generation and Export Flow

This export functionality makes the prototype more practical. The user does not have to manually copy results from the interface. Instead, the generated planning output becomes a reusable artifact.

4.8 Example Output and Interpretation

The included sample backlog produces a planning result focused on authentication and delivery foundations. The generated sprint goal recommends stabilizing authentication, preparing the delivery pipeline, and shipping the core login and registration flow with basic quality and deployment readiness.

This result is reasonable because the sample backlog contains several foundational tasks. For example, the User login API is connected to the login UI, password reset, tests, session handling, and security review. The database schema setup is necessary before user registration, and roles can be implemented. The CI/CD pipeline is necessary before deployment to staging. The security audit depends on the core authentication and registration features being available.

The generated output prioritizes tasks such as the login API, database schema setup, CI/CD pipeline setup, login bug fix, registration API, registration UI, security audit, deployment to

staging, and end-to-end tests. These recommendations show that the model is not only reading individual tasks but also considering dependencies and delivery risk.

The sprint-level summary also provides a useful management perspective. It recommends focusing the sprint on stabilizing the core authentication flow, establishing the database and delivery pipeline, and validating the critical flows through tests. At the same time, it suggests deferring lower-value polish or optimization tasks until the main flow is stable.

From a management point of view, this is one of the most useful parts of the system. The output does not only say which tasks are a high priority. It also explains why the sprint should focus on areas and what risks could affect delivery.

4.9 Running the application

The application can be run locally after installing the required dependencies and configuring the API key. On Windows, the user creates a virtual environment, activates it, installs the dependencies from requirements.txt, copies .env.example to .env, and adds the real OpenAI API key.

After setup, the web application is started with the command:

streamlit run app.py

When the command runs successfully, Streamlit opens the application in the browser. The user can then upload a backlog or use the default sample backlog and run the analysis.

The backend can also be tested without the Streamlit interface by running:

python agent.py

In this mode, the system reads the sample backlog, sends it to the AI model, prints the sprint goal and sprint summary, and saves the generated output files. This is useful for quick testing and for confirming that the AI logic works independently of the user interface.

4.10 Possible Future Extension Toward Agentic Behavior

A natural future improvement would be to make the system more agentic. In the current version, the AI analyzes the backlog and returns recommendations. A more agentic version would be able to perform a sequence of actions based on the analysis.

For example, after analyzing the backlog, the system could identify missing information and ask the user follow-up questions. It could validate dependencies through a separate deterministic function. It could propose a sprint plan, wait for human approval, and then create GitHub Issues automatically. It could also add labels, assign roles, and generate a sprint summary in a project-management tool.

Such an extension would increase the value of the system, but it would also increase implementation complexity. External integrations would require authentication, API handling, error management, permission control, and safeguards to prevent the AI from making unwanted changes. For this reason, the current implementation keeps the AI in a decision-support role and leaves final control to the human manager.

This design choice is appropriate for the dissertation because it demonstrates the core concept without introducing unnecessary technical risk. At the same time, it provides a clear direction for future research and development.

4.11 Summary

The implemented prototype demonstrates how Generative AI can be applied to a practical software project management task. The system accepts a backlog in CSV format, analyzes it through a project-management prompt, returns structured recommendations, displays the results through a Streamlit interface, and saves the output as CSV and JSON files.

The implementation combines Python, Streamlit, pandas, Pydantic, dotenv, and the OpenAI API in a simple but effective architecture. The use of structured output is one of the strongest parts of the design because it makes the AI-generated recommendations easier to validate, display, and export.

The prototype is not a complete project-management platform and should not be presented as such. It is better understood as a focused AI planning assistant that supports backlog analysis and sprint preparation. Its main contribution is that it shows how AI can assist managers with prioritization, ownership suggestions, risk identification, and sprint-level planning, while still preserving the need for human judgment and final approval.

5 Evaluation and Results

The evaluation examines whether the system can process backlog data and produce structured planning suggestions for software project management. The main focus is on the correctness of the generated output, the responsiveness of the system, and its ability to work under different project conditions.

The system evaluated in this chapter is the prototype described in the previous sections. It is a Streamlit application that accepts backlog data in CSV format. The application sends this data to an OpenAI model through a project management prompt and returns structured recommendations. These recommendations include task priorities, suggested owner roles, deadline estimates, rationales, blockers, and possible risks. The system also produces a sprint goal and a sprint summary. The results are displayed in the interface and can also be exported in CSV and JSON format.

It is important to clarify how the term "accuracy" is used in this evaluation. In traditional software testing, accuracy can often be measured against a fixed expected output. In backlog planning, however, there is rarely one objectively correct answer. Different project managers may prioritize tasks depending on team capacity, business strategy, stakeholder pressure, technical risk, and organizational context. For this reason, accuracy in this chapter is understood as planning accuracy. This means that the generated recommendations are evaluated based on whether they are complete, consistent with the input data, aligned with common project-management reasoning, and useful for human review.

The evaluation does not claim that the prototype replaces a human Project Manager. Instead, it examines whether the system can produce a reliable first planning draft. This is consistent with the intended purpose of the implementation: to support managerial decision-making rather than automate it completely.

Figure 12: Evalutation Framework

5.1 Testing Procedure and Evaluation Setup

The evaluation was carried out using the working prototype and the backlog structure supported by the application. The main test case included a sample backlog, which contains twenty-five software development tasks. These tasks represent a realistic software project scenario focused on authentication, user registration, testing, deployment, security, and documentation. The backlog includes different types of work, such as backend development, frontend implementation, QA activities, DevOps tasks, and security review.

The testing procedure followed the same workflow that a user would follow in practice. First, the application was started locally through the Streamlit interface. Then, the backlog was loaded into the system. If no external file was uploaded, the application used the default sample backlog. The input data was displayed in the interface so that it could be reviewed before analysis. After that, the AI analysis was executed using the "Run agent" button. The system then generated the task-level and sprint-level recommendations, displayed them in the interface, and saved the results in CSV and JSON format.

The evaluation considered both functional and managerial aspects. Functional testing examined whether the system operated correctly from an implementation perspective. This included checking whether the backlog could be loaded, whether the model returned

structured output, whether the recommendations were displayed correctly, and whether the export files were created. Managerial evaluation examined whether the recommendations were meaningful from a software project management perspective.

The main evaluation questions were the following:

Table 1: Evaluation Questions

Evaluation Area	Main Question
Functional correctness	Does the system complete the full workflow without errors?
Output completeness	Does every task receive a complete recommendation?
Planning accuracy	Do the priorities and deadlines make sense based on the backlog?
Role suitability	Are tasks assigned to suitable engineering roles?
Dependency awareness	Does the system identify important blockers and dependencies?
Risk identification	Does the system identify realistic delivery and technical risks?
Responsiveness	Is the system practical to use during planning?
Adaptability	Can the system handle different types of software project backlogs?

The evaluation was mainly qualitative, supported by quantitative indicators extracted from the generated output. This mixed approach is suitable for this type of prototype because backlog planning is partly structured and partly judgment based. The quantitative results show whether the system produces complete and well-formed output. The qualitative analysis examines whether the recommendations were useful and reasonable.

5.2 Quantitative Results

The first part of the evaluation focuses on measurable results from prototype execution. The system successfully processed the sample backlog and generated structured recommendations for all twenty-five tasks. Each task received a priority, suggested owner role, deadline estimate, rationale, blockers, and risks. The system also generated one sprint goal and one sprint-level plan.

The following table summarizes the main functional results of the evaluation.

Table 2: Main Funtional Results of the Evaluation

Metric	Result	Interpretation
Input tasks analyzed	25 / 25	All backlog items were processed successfully
Task recommendations generated	25 / 25	Every task received a structured recommendation
Sprint goal generated	1 / 1	The system produced a project-level sprint objective
Sprint summary generated	1 / 1	The system produced an overall planning summary
CSV export generated	Yes	The task-level output was saved in tabular form
JSON export generated	Yes	The full structured analysis was saved successfully
Output structure	Valid	The response matched the expected schema

The priority distribution of the generated task recommendations was also analyzed. Out of the twenty-five tasks, twelve were classified as High priority, four as Medium priority, and nine as Low priority.

Table 3: Priority Results

Priority	Number of Tasks	Percentage
High	12	48%
Medium	4	16%
Low	9	36%
Total	25	100%

This distribution is reasonable for the selected backlog. The backlog describes a project where several foundational tasks are required before the core user journey can be delivered. Tasks such as the login API, database schema setup, CI/CD pipeline, registration API, security audit, and end-to-end tests are naturally important because they either support core functionality or reduce delivery risk. At the same time, tasks such as documentation, performance optimization, profile page UI, and admin dashboard UI were treated as lower priority because they are useful but less urgent for the initial sprint goal.

The suggested owner roles were also evaluated. The generated role distribution was as follows:

Table 4: Role Assignment Results

Suggested Owner Role	Number of Tasks
Backend Engineer	13
Frontend Engineer	5
QA Engineer	3
DevOps Engineer	2
Security Engineer	1
Total	25

The data shows how the sample backlog is structured - there is a large volume of backend work in the backlog, specifically for authentication, registration, session management, database schema, rate limiting, logging, error handling besides API documentation. For the frontend tasks, focus on the login, registration, profile, responsiveness, and admin dashboard UI. In the QA category, tasks involve unit testing and end-to-end testing, while DevOps tasks link to CI/CD plus staging deployment. By assigning the security audit to a Security Engineer, the system follows appropriate procedures.

To provide a schedule, the system generated time requirements for all tasks. As calculated, the deadlines are between two and seven working days and the average is approximately 3.7 working days for each task. With those results, the output is consistent with the instructions, as tasks that are not complex take fewer days but also tasks that are complex take more days. It is important that the system did not create durations of zero days or durations that are too long to be functional - this indicates that the rules and the design of the output are effective for keeping the data within a useful range.

On the whole the numbers show that the system is able to finish the workflow. It is evident that the system handled all input tasks, created structured output and moved the data to an external format. And the patterns in the recommendations are the same as the information in the backlog.

5.3 Qualitative Analysis of Planning Accuracy

The second part of the evaluation examines the quality and usefulness of the generated planning recommendations. Since backlog planning does not have one mathematically

correct answer, the output was reviewed based on managerial reasoning and consistency with the input data.

The strongest result of the prototype is that it correctly identified the core theme of the backlog. The generated sprint goal focused on stabilizing authentication and delivery foundations before shipping the login and registration flow. This is a reasonable interpretation of the backlog because many tasks are related to authentication, user access, testing, and deployment. A human Project Manager reviewing the same backlog would likely identify authentication and registration as central parts of the project.

The system also showed good dependency awareness. For example, the login UI depends on the login API, while the registration UI depends on the registration API. The registration API depends on the database schema setup, and deployment to staging depends on the CI/CD pipeline. These relationships are important because they affect task sequencing. If the frontend work is started before the required backend APIs are available, delivery may be delayed, or integration problems may appear later. The system's recommendations correctly emphasized foundational work before dependent work.

The prioritization output was generally logical. Tasks related to core user flows, authentication, registration, testing, CI/CD, staging deployment, and security were placed in the high-priority group. This is appropriate because these tasks either deliver essential functionality or reduce major project risk. For example, fixing a login bug was treated as a high priority because it affects the ability of users to access the system. The security audit was also treated as high priority because authentication and user management features are sensitive areas where vulnerabilities can have serious consequences.

Medium-priority tasks were also reasonable. Password reset, email service integration, mobile responsiveness, and user roles were treated as important but not necessarily the first items to complete. This reflects practical planning logic. These tasks are valuable, but they depend on the core authentication and registration flow being stable.

Low-priority tasks included items such as API documentation, performance optimization, database indexing, session timeout handling, frontend unit tests, and admin dashboard UI. Some of these tasks are still important, but they are not necessarily required before the first stable delivery of the login and registration flow. For example, performance optimization is

useful, but it may be premature before the main functionality and realistic load patterns are known.

The system also produced useful risk descriptions. Examples of risks included security flaws in authentication, API integration mismatch, test fragility, environment configuration issues, deployment failures, and late discovery of vulnerabilities. These are realistic risks in software delivery. They are not generic about business risks; they are connected to the technical nature of the backlog.

The rationales were concise and understandable. This is important because a Project Manager needs to understand why a recommendation was made. The output did not only assign a priority; it also explained the reasoning behind it. This improves transparency and makes the result easier to review.

In certain areas the system has constraints - it is possible that users need to adjust some recommendations. As an example the software suggests dates - calculating how complex tasks are and how they relate to each other - but it does not have information regarding how fast the team works, which engineers are available, when holidays occur, how long the sprint lasts or which other tasks are important. It is able to suggest roles like Backend Engineer or QA Engineer. By contrast, it does not have data about the specific abilities or the current amount of work for each person on the team. The results are helpful when people create a first version of a plan - but users should not use the output as a finished sprint plan.

As shown by the qualitative evaluation, the system creates recommendations that are logical and helpful for people who manage projects. To identify the first tasks, recognize frequent problems in software delivery and explain the main goal of a sprint are the functions where the system works best. Then again, the system does not have specific information about how a particular company operates. And this lack of context is expected because the prototype is a simple version of the software.

5.4 Responsiveness and Usability Results

Responsiveness was evaluated from the perspective of practical use. The system is designed to support a manager during backlog review, so it must be simple enough to use without creating additional planning overhead.

The Streamlit interface performed well for the intended scope of the prototype. The user can load the backlog, inspect the input table, execute the analysis, and review the output on the

same page. The layout is straightforward and does not require the user to understand the internal implementation. This is important because the system is intended for project-management support, not only for technical experimentation.

The application also provides immediate visual feedback. The input backlog is displayed before execution, and the generated task recommendations are displayed in a structured table after execution. The sprint-level output is shown separately, which helps distinguish between detailed task planning and higher-level management interpretation.

Another positive usability feature is the ability to download the output as CSV and JSON. The CSV file is useful for spreadsheet review, while the JSON file preserves the complete structured analysis. This makes the prototype more practical because the generated results can be used outside the Streamlit interface.

The responsiveness of the system depends mainly on the AI model response time. The local application itself is lightweight. Loading the CSV, displaying the table, converting the output to a DataFrame, and saving the files are not computationally heavy operations. The main waiting time comes from the external model call. For the scope of an academic prototype, this is acceptable. In a production environment, however, response time would need to be measured more formally and optimized if the system were used with larger backlogs or multiple users.

From a usability perspective, the system has three main strengths. First, the workflow is easy to understand. Second, the output is presented in a format that resembles familiar project-management planning tables. Third, the user remains in control because the system does not automatically apply changes to an external project board.

The main usability limitation is that the system currently requires the backlog to be prepared in CSV format, which requires some manual work. A more advanced version could connect directly to GitHub Issues, Jira, Azure DevOps, or another project-management tool. However, this would also introduce more complexity, authentication requirements, and governance concerns.

Overall, the responsiveness and usability results are positive for an MVP. The system is practical for demonstration and suitable for controlled thesis evaluation. It is not yet a production-ready planning platform, but it successfully supports the intended AI-assisted planning workflow.

5.5 Adaptability Across Project Scenarios

Adaptability refers to the ability of the system to handle different types of backlog content and still produce useful recommendations. The implementation is not hardcoded for a specific project. It accepts CSV backlog data and analyzes the task descriptions, business value, urgency, complexity, and dependencies provided by the user. This makes the system adaptable to different software project scenarios, if the backlog is structured clearly.

The main tested scenario was an authentication and user-management project. In this scenario, the system performed well because it recognized the importance of core user access, security, database foundations, testing, and deployment. This is a common software project scenario and therefore suitable for demonstrating the prototype.

In another project scenario, the same system design is applicable. As an example, the system ranks production bugs, regression risks, monitoring improvements and reliability tasks when it analyzes a backlog for maintenance. To support feature development, the system ranks user functionality with high value, dependencies for future work, and tasks for release of readiness. In an infrastructure or DevOps context, the system ranks deployment pipelines, environment stability, observability, plus security configuration.

By using a prompt design and a structured schema, the system remains adaptable. It is not limited to rules for a single sample backlog. It defines project management principles where the model considers business value, urgency, complexity, dependencies, and task descriptions at the same time. As a result, the same implementation works with different types of backlogs.

But this adaptability has limits - the quality of the recommendations is dependent on the quality of the input data. If a backlog item has a title that is not specific, no description, no urgency but also no dependency information, the model has less context to produce a recommendation. If a project has constraints for a specific domain, like regulatory deadlines, customer commitments or technical risks, those constraints must be in the input. It is not possible for the system to infer context when the data is not provided.

On GitHub, public data is available to test further adaptability. GitHub issues include real bug reports, feature requests, technical debt items, documentation improvements, and release tasks. The issues require preprocessing before they are used in the prototype. They

often lack explicit business value, urgency, complexity, or dependencies. On that account, the public data is useful when it is transformed into the CSV format for the system.

With this evaluation, it is clear that the prototype is adaptable during input and reasoning but not during integration. It can analyze different backlog scenarios when they are provided as CSV files, but it is not connected to external project management systems - this is a characteristic of the current implementation as well as a direction for future work.

5.6 Summary of Findings

The evaluation shows that the implemented prototype successfully demonstrates the core idea of AI-assisted backlog planning. The system was able to load backlog data, process all tasks, generate structured recommendations, display the results, and export the output in CSV and JSON format. From a functional perspective, the prototype completed the expected workflow.

The results are consistent with the requirements for numerical data. To process the information, the system examined all twenty-five input tasks and produced recommendations for every task. It assigned twelve tasks to the category for high priority, four tasks to the category for medium priority, and nine tasks to the category for low priority. By using this process, the system matched tasks to roles that are appropriate for the work, like Backend Engineer, Frontend Engineer, QA Engineer, DevOps Engineer besides Security Engineer. If the deadlines are examined, they are in a range that users can achieve in a professional environment. On average, the time for each task is 3.7 working days.

The qualitative evaluation also showed positive results. The system correctly identified the main planning theme of the sample backlog: stabilizing authentication and delivery foundations before completing the login and registration flow. It recognized important dependencies, highlighted realistic risks, and produced rationales that were understandable from a managerial perspective.

Responsiveness and usability were appropriate for an academic MVP. The Streamlit interface was simple, the workflow was clear, and the export functionality made the results reusable. The main response-time dependency was the external AI model call, while the local application remained lightweight.

The system also showed adaptability potential. Because it accepts CSV input and uses general project-management reasoning rules, it can be applied to different backlog scenarios. However, its adaptability depends on the quality of the input data, and it does not yet integrate directly with GitHub, Jira, Azure DevOps, or similar platforms.

The main limitation is that the system is still a prompt-driven planning assistant rather than a fully autonomous agentic system. It does not execute actions, create issues, update sprint boards, or maintain memory across sessions. Nevertheless, this limitation is acceptable for the scope of the dissertation. The prototype demonstrates the feasibility and value of AI-supported backlog planning while preserving human control.

Overall, the evaluation supports the conclusion that Generative AI can assist Project Managers and Engineering Managers in software development planning. The system does not replace managerial judgment, but it can provide a structured first draft that helps accelerate backlog review, improve visibility of dependencies and risks, and support more organized sprint planning.

6 Discussion and Managerial Implications

This chapter discusses the findings of the evaluation and interprets their meaning from a software project management perspective. While the previous chapter focused on the testing procedure and the results produced by the prototype, this chapter examines what those results imply for Project Managers, Engineering Managers, and software development teams. The purpose is not only to determine whether the implemented system works technically, but also to understand how such a tool could influence planning practices, managerial decision-making, team coordination, and human-AI collaboration.

As a result of the implemented prototype, it is clear that Generative AI is able to convert a software backlog into a structured planning output - those outputs include task priorities, suggested owner roles, deadline estimates, blockers, risks and a sprint level summary. Because those items relate to common software project management activities, they are functional - but the results require careful interpretation. It is important to note that the system does not substitute for the judgment, experience, or responsibility of a human manager. It is a mechanism to support decisions. To use it is to accelerate backlog reviews plus create a first planning draft.

In this chapter, the opportunities and limitations of AI-assisted planning are explored. On the one hand, there are benefits shown by the prototype like higher planning efficiency but also better visibility of dependencies. And there is clearer communication and more structured decision making. Then again there are concerns about trust, transparency, accuracy, as overreliance. There are also issues with data quality and the requirement for human oversight - those factors are relevant in software projects. In the projects, decisions depend on technical uncertainty or team capacity. They also depend on organizational priorities and stakeholder expectations.

The discussion is organized around four main areas - first, the chapter interprets the evaluation results within the framework of software project management. It describes the benefits for managers when they use AI as a planning assistant. It examines the limitations next to the risks of the current implementation. In this section, there are details about trust and transparency. There is a discussion on the broader implications of human-AI collaboration. And it considers how the prototype might change toward more agentic behavior in the future

6.1 Interpretation of the Results in Software Project Management

In the evaluation of the implemented prototype, results show that Generative AI is a source of support for the management of software projects - this support is present during the review of backlogs, the setting of priorities, the preparation for sprints and the awareness of risks. By using the system, a user transforms a raw CSV backlog into a structured output for planning - this output includes priorities, roles for owners, deadlines, reasons for decisions, blockers, risks, goals for the sprint and summaries. As the prototype is limited in its scope, the data indicates that a tool with AI assistance helps a Project Manager or Engineering Manager when they organize information in a backlog.

To manage a software project in a practical way, a person does not perform a mechanical task. It is a process where a manager uses their judgment and experience to find a balance between the value for a business, the level of urgency, the level of technical complexity, the dependencies and the risks. On that accounting process, the prototype applies a policy for decisions based on prompts to every item in the backlog. When the system generates recommendations, it identifies tasks that form a foundation, like APIs for authentication, the setup of database schemas, the configuration for CI/CD and reviews for security - those

tasks are of high priority because they affect the delivery of other items and are able to become blockers if delays occur.

By using the prototype, the system also shows that structured output from an AI is useful. Instead of a long answer in free text form, the system creates a format that is consistent and available as a table, a CSV file or a JSON file. From the perspective of management, this is useful because managers must review, compare, share and reuse the outputs of planning. For a Project Manager, an opinion is not enough, as they require a document that supports discussions and decisions. Due to the structured format, the output from the AI is similar to a document for planning.

On the topic of system performance, the AI is most effective when the backlog has descriptions of tasks that are clear, indicators for business value, levels of urgency, estimates for complexity and data about dependencies. If the input data is of high quality, the AI is useful for project management. When a backlog is prepared well, the AI generates recommendations that are coherent. If a backlog is not complete or is vague, the quality of the output is lower. And this is a lesson for managers, as tools for AI do not stop the requirement for documentation that is disciplined. They increase the value of project data that is clear and structured.

But the results are subject to careful interpretation - it is not a fact that AI is able to manage a software project in full capacity. To be precise, the prototype shows that AI is an assistant for one activity in planning. Although the system suggests priorities and risks, it is not able to understand the factors in an organization that influence decisions. As an example, it is not aware if an engineer has too much work, if a customer asks for a feature with urgency, if a deadline is in a contract or if a team changes its architecture. If those factors are not provided as input, they are outside the model.

The interpretation of the results is that the system is a mechanism for the support of decisions. It is not a maker of decisions that act on its own. With this tool, a manager accelerates the first stage of a backlog review and has a starting point that is structured. The final decisions for planning are the responsibility of the human manager.

6.2 Managerial Benefits of AI-Assisted Planning

In the first instance, the prototype makes the planning process more efficient for managers. On many software teams, people must spend a lot of manual effort to review a backlog. It

is necessary for a manager to read every task, know what the task is for and look at how it relates to other tasks. And the manager must find dependencies, calculate how urgent tasks are and assign owners. For a backlog of a medium size, this process lasts for a long duration. To lower this work, the system automatically creates a structured recommendation. The manager is able to review, fix and improve the plan instead of beginning with nothing.

By using the assistant, the team also achieves more consistency. When humans set priorities, they are often affected by recent events, pressure from stakeholders, or how well they know specific tasks. Or they may not fully understand how tasks depend on each other - but the AI assistant uses the same logic for every item in the backlog. It looks at business value, urgency, complexity, dependencies and risk - following the rules in the prompt. If the AI is not always right, it still makes the initial review process more uniform. As a result, managers are able to see when tasks are ignored or when people think tasks are easier than they are.

With this prototype, blockers and risks are more visible - in software projects; delays happen frequently because individuals do not understand dependencies well. As an example, a frontend task is often dependent on a backend API that does not exist yet. Because of this a task that seems simple can be difficult to complete. By highlighting those blockers, the prototype helps managers find problems with the order of tasks early.

For cross functional coordination, the system provides further support. It suggests roles like Backend Engineer, Frontend Engineer, QA Engineer, DevOps Engineer besides Security Engineer. Because software delivery involves many different roles, this feature is useful. If the system maps tasks to roles, the manager is able to see which team members have the most work. When many urgent tasks are for backend engineers, there is a risk that the workers have too much to do. If testing happens at the end of a plan, the manager is able to move those tasks to an earlier time.

To improve communication, the system creates a sprint goal and a summary - those provide a short description of the backlog. Because managers must explain plans to developers and leaders, this description is helpful. A list of tasks does not always show why a sprint is organized in a certain way - but a summary is able to show the focus and the most important risks. In this way, the AI is a tool for both planning and talking to others.

And the system makes the management of a backlog more analytical. If every task is checked for priority, role, deadline, and risk, the planning becomes more thorough - this helps teams use clear reasons for their choices instead of using informal methods.

As a final point the advantages are greatest when the AI is an assistant and not a final authority. It is best if the manager treats the output as a draft. The system is valuable because it makes the planning process faster and more organized, not because it makes final decisions without a person.

6.3 Trust, Transparency, and Risks

The evaluation results indicate that software project management benefits when AI-assisted backlog planning creates structured priorities, role suggestions, deadline estimates, blockers, risks and sprint level summaries - but such systems also create risks for managers - those risks are technical and involve how people trust the system, how clear the processes are, who is responsible for outcomes, how accurate the data is and how humans supervise the work.

Trust is a central factor - if a project manager uses the system, they must rely on the output as a planning help without accepting the suggestions without a review. To achieve this the manager must maintain calibrated trust. The system is a decision support tool that creates a draft but it is not an authority that makes the final decisions for the project. For this to work the interface and interaction design must help users see what the system can do, inspect the outputs and keep control over final decisions.

Transparency is also necessary - when a manager understands why the system produced a recommendation, that recommendation is more useful. On that account the prototype provides reasons for every task recommendation - those explanations allow the user to see if the system prioritized a task because of its business value, urgency, dependencies, security relevance or delivery risk - but an explanation is not a guarantee that the recommendation is correct. In some cases the AI misses context about the organization that does not exist in the backlog.

Another risk is automation bias - as a result of this bias, users might use the automation in ways that are not appropriate or rely on it too much. If the AI-generated priorities, deadlines

or role suggestions look structured and professional, users might accept them - this is a problem when the backlog is not complete or when the data lacks important business constraints. Because decisions affect workload, delivery commitments and what stakeholders expect, human review is necessary.

Accountability must also be clear - when an AI recommendation results in a plan that is not effective, the human cannot transfer the responsibility to the system. By validating adjusting and approving the final plan, the human manager remains responsible. On that account the prototype uses a human-in-the-loop approach where the AI supports the planning but the manager is accountable for the decisions.

Data quality is an additional concern - because the system uses the backlog as input, it depends on the quality of that data. If the descriptions of tasks are vague, if dependencies are not there or if priorities are old, the recommendations are likely to be weak. AI-assisted planning is not a replacement for disciplined backlog management. It makes clear and structured project data more important.

Use in a professional setting requires that the organization pays attention to how it governs data and keeps it confidential. In software backlogs, there is often sensitive information like customer issues, product strategy, security weaknesses or internal deadlines. Because of this any deployment needs clear rules about data privacy, who can access the data and how to use AI-generated recommendations in a responsible way.

By showing that Generative AI can support backlog planning, the prototype also confirms that human judgment is essential. Transparency, accountability and governance are the conditions that allow people to use AI-assisted planning tools in a responsible manner within software project management.

6.4 Human-AI Collaboration and Future Managerial Implications

By this dissertation, the main implication for management is that managers should treat AI as a tool for collaborative planning and not as a tool that replaces the person who manages projects. It is most useful when the prototype supports how a manager thinks. As the system reads the backlog, it proposes a plan, identifies dependencies, highlights risks plus summarizes the direction of the sprint. When the manager receives this output, they evaluate the data, apply context from the organization, adjust what the system recommends and make the decision.

Figure 13: Human-AI Collaboration in Planning

To change how people perform project planning, this collaboration between humans besides AI is necessary. Instead of spending time when they organize backlog information by hand, managers are able to spend more time when they evaluate alternatives, discuss trade offs but also align the team. Because of this process, the AI performs the analytical preparation and the manager focuses on judgment, communication, negotiation as well as leadership.

For Engineering Managers, the system is a way to detect when technical bottlenecks exist. If a role has many tasks with a high priority, a capacity issue is likely. If tasks for security, testing or deployment are blockers, the manager can address those risks early. On this basis the AI output supports how people plan tasks, resources and the management of delivery risks.

For Project Managers, the system is a support for sprint planning or for when they communicate with stakeholders. To explain why certain tasks are first, the manager uses the generated sprint goal and summary. In the situations, planning discussions are structured next to there is less risk that decisions are based only on what individuals feel or on pressure from urgent tasks.

For organizations, the implication is that AI can improve how mature the planning process is if the integration is responsible. With AI tools, there is more structured backlog data, task descriptions are clear, dependencies are explicit and risk documentation is better - but it is necessary for managers to understand where AI-generated output is strong plus where it is limited.

In a future version, the system could act more like an agent - if it connects to GitHub Issues or Jira, it retrieves tasks, detects when information is missing, asks questions, validates dependencies, proposes a sprint plan and creates issues or labels after a human gives approval. And while this makes the system more powerful, the complexity but also risk are also higher. To have more autonomy, it is necessary to have safeguards, permission controls, audit logs and mechanisms for approval.

By following the most realistic direction, the goal is controlled autonomy as well as not full automation. In this model the AI prepares recommendations and drafts changes but the human manager approves them before they change the project environment. It is then possible to keep the manager accountable while the project benefits from the speed or analysis of the AI.

To conclude the prototype shows that Generative AI supports the management of software projects in a way that is practical and easy to understand. Its value is not that it makes decisions without errors but that it helps managers when they structure the planning process. It is a way to lower manual effort, improve visibility next to support discussions that are based on information. At the same time it is necessary to use the system with care. If AI tools for project management are used in organizations then trust, transparency, oversight by humans and data governance are essential.

7 Conclusion and Future Work

This dissertation examined the use of Generative AI as a decision-support tool for software project management, with a specific focus on backlog analysis, task prioritization, role recommendation, deadline estimation, sprint planning, and risk identification. The central aim was to investigate whether an AI-assisted prototype could help a Project Manager or

Engineering Manager transform a raw software backlog into a structured and useful planning output.

To address this aim, a lightweight prototype was designed and implemented. The system, named Agentic AI Backlog Planner, was developed as a Python and Streamlit application. It accepts backlog data in CSV format, sends the backlog to an OpenAI model through a project-management prompt, and returns structured recommendations. The output includes task priorities, suggested owner roles, deadline estimates, rationales, blockers, risks, a sprint goal, and a sprint-level planning summary. The results are displayed in the web interface and exported as CSV and JSON files.

The implementation showed that Generative AI can be applied effectively to a practical project-management workflow. The prototype was able to process the sample backlog, identify important tasks, highlight dependencies, suggest suitable roles, and produce a coherent sprint direction. In particular, the system correctly emphasized foundational work such as authentication, database setup, CI/CD configuration, testing, security review, and staging deployment. These recommendations were consistent with common software project management reasoning, where blocking dependencies, core user flows, and delivery risks must be considered carefully during planning.

In this dissertation, the conclusion is that artificial intelligence provides value when it improves how a process for planning is structured. A backlog often contains many tasks which possess different degrees of importance for a business, technical difficulty or time sensitivity. And those tasks frequently have relationships where one item depends on another. By reviewing the data points without automation, a person uses a large amount of time and produces results that vary in quality. To address this the developed prototype shows that artificial intelligence is able to organize those data into a format that people use more easily. It is not a replacement for the decisions that managers make but it is a tool that reduces how much work a person must do to create a first version of a plan.

Another important conclusion is that structured output is essential for making AI useful in project-management contexts. A simple free-text response would not be sufficient for practical use. Project managers need information that can be reviewed, compared, exported, and discussed. By using structured models for task recommendations and sprint planning,

the prototype produces output that is closer to a real planning artifact. This improves usability and makes the system more suitable for further development.

The dissertation also concludes that the current implementation should be understood as a prompt-driven AI planning assistant rather than a fully autonomous agentic system. Although the prototype uses the term "Agentic AI" in its title, the current version does not independently execute multi-step actions, update GitHub Issues, modify sprint boards, or maintain memory across sessions. This distinction is important. The system supports planning, but it does not take control of the project management process. This makes the prototype safer, simpler, and more appropriate for a thesis implementation.

The evaluation showed positive results in terms of functionality, output completeness, planning relevance, responsiveness, and adaptability. The system successfully analyzed all tasks in the sample backlog and generated complete recommendations. It also produced a sprint goal and summary that reflected the overall direction of the project. At the same time, the evaluation made clear that the output depends heavily on the quality of the input data. When backlog items are clear and structured, the AI can provide stronger recommendations. When the backlog is vague or incomplete, the output may require more human correction.

Overall, the dissertation supports the view that Generative AI can assist software project managers, especially during early planning and backlog refinement. Its greatest value is not in replacing human decision-making, but in supporting it. The system can provide a first draft, identify risks, expose dependencies, and help managers start planning discussions from a more organized position.

7.1 Key Contributions

The first contribution of this dissertation is the development of a working AI-assisted backlog planning prototype. The project goes beyond theoretical discussion by providing an implemented system that can be executed, tested, and demonstrated. This is important because many discussions about AI in management remain abstract. The prototype shows one concrete way in which Generative AI can be used in a software project management workflow.

The second contribution is the design of a focused AI planning workflow. The system follows a clear process: backlog input, AI analysis, structured response, visual display, and output export. This workflow is simple, but it is also realistic. It reflects how a Project Manager might use such a tool during backlog review or sprint preparation.

The third contribution is the use of structured AI output. The prototype defines specific output models for task-level and sprint-level recommendations. This makes the system more reliable than a simple chatbot-style interaction. The generated output can be transformed into tables, saved as files, and reviewed systematically. This contribution is particularly relevant because many business uses of Generative AI require outputs that are not only readable, but also reusable.

The fourth contribution is the application of AI to managerial reasoning in software engineering. The prototype does not generate code or perform technical debugging. Instead, it supports planning decisions. It considers business value, urgency, complexity, dependencies, role suitability, deadlines, blockers, and risks. This demonstrates that Generative AI can support not only technical software engineering tasks, but also management activities around coordination and decision-making.

The fifth contribution is the identification of practical limitations and governance concerns. The dissertation does not present AI as a perfect or autonomous solution. It recognizes issues such as trust, transparency, overreliance, data quality, lack of organizational context, and the need for human review. This balanced view is important because AI adoption in management must be handled carefully.

Finally, the dissertation contributes by showing a realistic path from a prompt-driven prototype toward a more agentic system. The current implementation provides a foundation that could later be extended with tool use, memory, GitHub or Jira integration, dependency validation, and human approval workflows.

7.2 Practical Implications

The findings are useful for people who manage software projects in multiple ways.

It is possible for tools that use artificial intelligence to help people plan backlogs so they spend less time when they create a draft for a plan. Instead of a manager who looks at every item in a list by hand from the start, the manager is able to use the system to create a version

of the plan that has a structure. It is true that this does not remove the need for individuals to check the work but the process of checking is able to be more rapid and directed.

The system is able to make it easier for people to see things that rely on each other and things that might cause problems. In projects for software, people often experience delays when they find those relationships between tasks after too much time has passed. The prototype is helpful because it finds things that stop progress and things that are risky during the start of the planning process. As a result individuals are able to put tasks in a better order and plan sprints with more information.

The system is able to help when technical and non technical people talk to each other. As an example the goal for the sprint and the summary that the system creates are ways to explain what the sprint is about and the reasons for it - this is helpful when Project Managers describe why they made certain choices to developers, product owners, senior managers or other people involved in the project.

Planning that uses artificial intelligence is able to make individuals keep better backlogs. Because the system is more effective when people define descriptions of tasks, value for the business, how soon things are needed, how difficult things are and how tasks rely on each other, teams are likely to keep data in the backlog that has more structure. In this way the use of the tools is an indirect way to make people more disciplined when they write project documents.

The prototype is evidence that individuals should use those tools as systems where humans and machines work together. If a Project Manager uses the system, they should not take the result without looking at it. The manager is supposed to treat the suggestion from the system as a draft. It is the responsibility of the manager to use what they know about the business, what they know about the team, and their professional ability to make decisions.

Due to the points, the message is that artificial intelligence is useful in the management of software projects - but it is meant to be a helper, and it is not a thing that takes the place of a manager who is responsible for the results.

7.3 Limitations

This dissertation has several limitations that must be acknowledged. These limitations are methodological, technical, evaluative, and organizational in nature. They do not undermine

the value of the prototype, but they define the boundaries within which the findings should be interpreted.

In the first limitation, the scope of this research is narrow. The dissertation examines specific tasks that people perform to manage software development projects - those activities include when people interpret backlogs, prioritize tasks, recommend roles, estimate deadlines, identify blockers, identify risks and summarize sprints. It does not measure how large organizations adopt AI for project management, how teams behave over long periods, how much money the system saves or how entire companies change. A reader should view those findings as a sign that the system works and is useful for planning. By design the study does not prove that the system changes a whole organization.

As a second limitation, the prototype is a simple system - the software is lightweight and is not a version that a company can use for daily operations. It lacks features for when users log in, store data permanently, collaborate with others, control access, log audits, deploy software, or secure data. To use this in a real company, a developer would need to add the features. For this thesis those elements were not part of the plan. The goal was to show how AI helps with planning rather than to create a commercial product.

And a third limitation is that the system responds to prompts instead of acting as a full agent. Although the text discusses Agentic AI, the software follows one main process. In this process the system takes backlog data, sends the data to an AI model with a specific prompt and provides a list of suggestions. It does not use external tools on its own, create GitHub Issues, update boards, assign tasks to individuals or perform actions across different systems. Because of this the prototype has a low level of autonomy - but this choice makes the system more secure and easier for a researcher to test. With this design the prototype is suitable for an academic project.

The fourth limitation concerns the use of synthetic backlog data as the main evaluation dataset. Synthetic data was useful because it allowed the testing scenario to be controlled, safe, repeatable, and free from confidential company information. However, synthetic backlogs may not fully represent the complexity of real software project environments. Real backlogs often contain vague descriptions, incomplete acceptance criteria, duplicate issues, outdated priorities, missing dependency information, stakeholder pressure, changing business objectives, and team-specific constraints. As a result, the system's performance

with synthetic data should not be assumed to transfer directly to all real-world project settings without further testing.

The fifth limitation is the absence of live organizational context. The prototype analyzes the information provided in the backlog, but it does not have access to actual team capacity, developer availability, historical velocity, individual skill levels, holidays, business deadlines, contractual commitments, or stakeholder priorities. These factors often influence real planning decisions. For this reason, the generated deadlines, priorities, and owner-role suggestions should be understood as initial planning recommendations rather than final managerial decisions.

The sixth limitation concerns the evaluation approach. The evaluation was mainly qualitative, supported by descriptive quantitative indicators such as the number of tasks processed, the completeness of recommendations, priority distribution, owner-role distribution, and output export success. This approach is suitable for an exploratory MBA dissertation and for evaluating an early-stage prototype. However, future research could apply a more formal quantitative evaluation framework. For example, AI-generated recommendations could be compared against rankings produced by experienced Project Managers or Engineering Managers. Future studies could also measure planning time reduction, user satisfaction, trust, perceived usefulness, and consistency across repeated runs.

The seventh limitation relates to AI reliability. Generative AI models may produce outputs that are incomplete, inconsistent, biased, or contextually incorrect. Although the use of structured output models reduces some of this risk, it does not eliminate it entirely. The system may still misunderstand a task, overestimate or underestimate priority, suggest an unsuitable role, or identify risks that require human validation. This is especially important because well-formatted AI output can appear more authoritative than it actually is. Human review remains necessary before any recommendation is accepted or applied.

The eighth limitation concerns dependency analysis. The prototype relies mainly on the AI model's interpretation of task descriptions and dependency fields. It does not include a deterministic dependency graph engine that formally calculates dependency chains, identifies circular dependencies, or validates task sequencing. As a result, dependency awareness exists at the reasoning level of the AI output, but not as a separate algorithmic

verification mechanism. A future version could combine LLM-based interpretation with deterministic dependency graph logic to improve reliability.

In the ninth limitation, the system does not integrate with external tools for project management. It uses files in CSV format for input and produces files in CSV besides JSON formats for output. With this design the user can run, show and test the prototype easily but the user must also prepare the backlog data by hand. For a more advanced version, the system would link directly to GitHub Issues, Jira, Azure DevOps, Trello, Linear or other platforms that perform similar tasks. By adding those links, the system would become more useful in a practical sense - but the links would also make the system more complex because they require methods for identity verification, access rights, errors in the interface, privacy of data plus processes for approval.

The last limitation is about how the system governs and protects data. Real project backlogs often include details that are private, like the names of customers, weaknesses in security, dates for internal deadlines, plans for product strategy, tasks for employees and the order of business goals. If a user sends this data to a model for artificial intelligence that is external, it creates risks for privacy, legal rules but also safety. To avoid those risks, this dissertation uses data that is artificial or information from projects that is open to the public. If an organization uses this system in a real setting, it must use stronger methods for governance, which include systems to label data by type, tools to control who can see data, ways to track who changed data and rules about which data a system for artificial intelligence can process.

On the whole, the limitations prove that the system is a specific demonstration of a concept and not a finished tool for autonomous project management. It is clear from the prototype that generative artificial intelligence helps with planning backlogs as well as making decisions for managers. And it is also clear that humans must still watch the system, the data must be accurate, the logic must be visible, and the governance must be responsible. As a result those limitations establish a basis for future research - this research can focus on systems that act more independently, ways to test the system that are more rigorous, tests in real environments and the careful linking of the system with platforms for project management.

7.4 Future work

To improve the system, the developer can first make the process more agentic. In a future version, the system is able to perform a workflow for multi step planning instead of a single analysis that is based on a prompt. As an example the tool first analyzes the backlog and identifies information that is missing. By asking follow up questions and validating dependencies, it proposes a plan for the sprint. It then requests a person to give approval before it creates the final output. As a result the tool is more interactive and acts like an assistant that helps with planning.

The second improvement is the integration of the system with tools for project management. At this time the prototype is dependent on CSV input, which is a method that is simple but requires manual work. In a future version, the tool connects directly to GitHub Issues, Jira, Azure DevOps, Trello, Linear or Notion. With those connections, the system is able to read items from the backlog automatically. And it might create labels, update how it ranks priorities or draft plans for sprints - but the integrations are only possible if the developer pays attention to permissions, security and mechanisms for approval.

The third direction is the addition of a workflow where a human provides approval. If the developer does not allow the system to apply changes automatically, the tool generates actions it proposes and asks a manager to approve or reject them. To illustrate the tool suggests that it should create labels in GitHub or update the priority of an issue - but it only executes those changes after a person gives an explicit confirmation. Because of this design, the system is balanced between processes that are automatic and control that a manager maintains.

The fourth improvement would be better dependency analysis. The current system relies mainly on the AI model to interpret dependencies from the backlog. A future version could include deterministic dependency graph logic. This would allow the system to calculate dependency chains, detect circular dependencies, identify blocked tasks more reliably, and visualize the order in which tasks should be completed.

The fifth direction would be the inclusion of team capacity and historical delivery data. If the system had access to sprint velocity, developer availability, role capacity, and previous delivery performance, it could produce more realistic deadline and ownership recommendations. This would make the planning output more useful in real project environments.

The sixth improvement would be a stronger evaluation framework. Future research could compare the AI-generated recommendations with those of experienced Project Managers or Engineering Managers. The study could measure agreement rates, planning time reduction, user satisfaction, perceived usefulness, and trust in the system. This would provide stronger evidence of the practical value of AI-assisted planning.

Another possible direction would be testing the system with public GitHub repositories. Public GitHub data could provide real software issues and feature requests without exposing confidential company data. These issues could be transformed into the CSV format required by the current prototype or used directly through an API integration in a future version.

Finally, future work could examine the governance of agentic AI in project management. As systems become more autonomous, questions around accountability, transparency, security, and trust become more important. Further research could investigate how organizations should define approval rules, audit AI decisions, protect sensitive data, and ensure that human managers remain responsible for final decisions.

7.5 Final Remarks

In this dissertation, the author shows that a lightweight Generative AI prototype assists software project management when it turns backlog data into recommendations for structured planning. It is important to note that the system does not replace a Project Manager and it does not act as a fully autonomous agent but - but the results demonstrate that AI provides practical utility when it functions as a planning assistant.

The prototype is valuable because it can organize information in the backlog, identify which tasks are most important, suggest specific roles and calculate when work will finish. For software projects, the system highlights potential problems plus summarizes the direction of a sprint. By creating a first draft that follows a specific structure, the system helps managers so they spend less time on tasks, see plans more clearly and talk with their teams while they have more information.

At the same time the research confirms that humans must still make the final judgments. As managers receive recommendations from the AI, they must review the data according to how much work the team can perform, what the business needs most, the limitations of the technology and the specific situation of the organization. To describe the most likely future, managers do not lose their jobs to AI but instead they use AI tools so they can make

decisions that are more accurate, occur more quickly but also are easy for others to understand.

To conclude this research supports the idea that Generative AI has a significant capacity to help with software project management. There is a practical foundation in the prototype for more development and future research. And this work moves toward planning systems that can act more independently, fit into existing workflows and remain under the control of humans.

Bibliography

Achimugu, P., Selamat, A., Ibrahim, R., & Mahrin, M. N. (2014). A systematic literature review of software requirements prioritization research. Information and Software Technology, 56(6), 568–585. https://doi.org/10.1016/j.infsof.2014.02.001

Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S., Bennett, P. N., Inkpen, K., Teevan, J., Kikin-Gil, R., & Horvitz, E. (2019). Guidelines for human-AI interaction. Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, 1–13. https://doi.org/10.1145/3290605.3300233

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). Manifesto for Agile Software Development. https://agilemanifesto.org/

Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots. Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, 610–623. https://doi.org/10.1145/3442188.3445922

Boehm, B. W. (1988). A spiral model of software development and enhancement. Computer, 21(5), 61–72. https://doi.org/10.1109/2.59

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). Language models are few-shot learners. ArXiv Preprint arXiv:2005.14165.

Davenport, T. H., & Kirby, J. (2016). Only humans need apply: Winners and losers in the age of smart machines. Harper Business.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. ArXiv Preprint arXiv:1702.08608.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. MIS Quarterly, 28(1), 75–106. https://doi.org/10.2307/25148625

Jarrahi, M. H. (2018). Artificial intelligence and the future of work: Human-AI symbiosis in organizational decision making. Business Horizons, 61(4), 577–586. https://doi.org/10.1016/j.bushor.2018.03.007

Kahneman, D. (2011). Thinking, fast and slow. Farrar, Straus and Giroux.

Kantas, A. (2026). Agentic AI Backlog Planner [Source code]. GitHub. https://github.com/AristidisKantas/mbathesis

Moe, N. B., Dingsøyr, T., & Dybå, T. (2010). A teamwork model for understanding an agile team: A case study of a Scrum project. Information and Software Technology, 52(5), 480–491. https://doi.org/10.1016/j.infsof.2009.11.004

Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, 1–22. https://doi.org/10.1145/3586183.3606763

Perkusich, M., Chaves e Silva, L., Costa, A., Ramos, F., Saraiva, R., Freire, A., Dilorenzo, E., Dantas, E., Santos, D., Gorgônio, K., Almeida, H., & Perkusich, A. (2020). Intelligent software engineering in the context of agile software development: A systematic literature review. Information and Software Technology, 119, 106241. https://doi.org/10.1016/j.infsof.2019.106241

Raisch, S., & Krakowski, S. (2021). Artificial intelligence and management: The automation–augmentation paradox. Academy of Management Review, 46(1), 192–210. https://doi.org/10.5465/amr.2018.0072

Royce, W. W. (1970). Managing the development of large software systems. Proceedings of IEEE WESCON, 1–9.

Shneiderman, B. (2020). Human-centered artificial intelligence: Reliable, safe and trustworthy. International Journal of Human-Computer Interaction, 36(6), 495–504. https://doi.org/10.1080/10447318.2020.1741118

Stettina, C. J., & Hörz, J. (2015). Agile portfolio management: An empirical perspective on the practice in use. International Journal of Project Management, 33(1), 140–152. https://doi.org/10.1016/j.ijproman.2014.03.008

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. ArXiv Preprint arXiv:1706.03762.

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., Zhao, W. X., Wei, Z., & Wen, J.-R. (2024). A survey on large language model based

autonomous agents. Frontiers of Computer Science, 18(6), 186345. https://doi.org/10.1007/s11704-024-40231-1

Wauters, M., & Vanhoucke, M. (2017). A nearest neighbour extension to project duration forecasting with artificial intelligence. European Journal of Operational Research, 259(3), 1097–1111. https://doi.org/10.1016/j.ejor.2016.11.018

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. V., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. ArXiv Preprint arXiv:2201.11903.

Wieringa, R. J. (2014). Design science methodology for information systems and software engineering. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-43839-8

Wooldridge, M. J. (2009). An introduction to multiagent systems. John Wiley & Sons.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). ReAct: Synergizing reasoning and acting in language models. ArXiv Preprint arXiv:2210.03629.

Author's Statement:

I hereby declare that, in accordance with article 8 of Law 1599/1986 and article 2.4.6 par. 3 of Law 1256/1982, this thesis/dissertation is solely a product of personal work and does not infringe any intellectual property rights of third parties and is not the product of a partial or total plagiarism, and the sources used are strictly limited to the bibliographic references.