Ministry of Higher Education

Herat University

Computer Science Faculty

Department (Software Engineering)

Agile vs Waterfall: Choosing Appropriate Project Management Methodology For Herat Computer Science Faculty Archive Management System

Monograph Documentation by Nesar Ahmad Morady

Supervisor: Mr. Abdul Khaleq Herawi

Date: 2024

Herat University

Computer Science Faculty

Software Engineering Department

Acknowledgment of Thesis Defense

It is at this moment confirmed that the thesis of the honorable Nesarahmad the son of Mohammad Omer titled Agile Vs Waterfall: Choosing Appropriate Project Management Methodology For Herat Computer Science Faculty Archive Management System under the supervision of the respected teacher, Abdul Khaleq Herawi, was written and the student mentioned above has successfully defended his thesis on 2024 / /.

- Signature of the supervisor

1- Signature of the jury

2- Signature of the jury

3- Signature of the jury

With respect,

Abdul Khaleq Herawi

Head of Software Engineering Department

Abstract

The selection of an appropriate project management methodology is crucial to the success of software development projects. This study presents a comparative analysis of Agile and Waterfall methodologies within the context of developing an archive management system for the Computer Science Faculty at Herat University. Agile, known for its iterative and flexible approach, and Waterfall, characterized by its structured, linear process, were both used to manage the development process and risk management aspects of the project. Using Trello boards to manage tasks, risks, and timelines, the study evaluates the effectiveness of each methodology based on task completion rates, average time per task, and risk mitigation strategies.

The results reveal that while Waterfall had a higher task completion rate (87% vs. Agile's 83.3%) and proved more efficient in terms of average time per task (2.7 hours vs. 5 hours), Agile demonstrated higher flexibility in adapting to evolving requirements and managing high-impact risks. Both methodologies identified and mitigated a similar number of risks, but Agile's iterative nature enabled more proactive and timely responses, minimizing delays. This research provides valuable insights into the applicability of Agile and Waterfall methodologies, offering practical recommendations for selecting the most suitable approach based on project complexity, stakeholder needs, and risk management requirements.

By comparing these methodologies in a real-world academic setting, this study contributes to the ongoing discourse on project management in software engineering and guides institutions facing similar decisions in software project execution.

Keywords: Agile, Waterfall, Project management

Acknowledgment

First and foremost, I express my deepest and sincerest gratitude to Almighty Allah, whose countless blessings have guided and sustained me throughout my academic journey. Without His divine support, none of this would have been possible. I owe an immeasurable gratitude to my beloved family, whose unwavering love, prayers, and sacrifices have been the cornerstone of my success. I am forever grateful to my siblings for their encouragement and for always standing by me. I also want to extend my deepest appreciation to my esteemed supervisor, Mr. Abdul Khaleq Herawi. His insightful guidance has been pivotal in the completion of this monograph. To all those who have played a part in this endeavor, I extend my heartfelt thanks. May Allah bless you all.

Table of Contents

Table of Figures

Table of Tables

Chapter 1: Introduction

1.1 Introduction:

The way we develop software has changed a lot over the years, leading to different methods aimed at improving project results and keeping stakeholders happy. Two of the most well-known approaches are Agile and Waterfall, which are very different from each other. Agile is flexible, allowing teams to work in cycles and adjust as they go, while Waterfall follows a step-by-step process with clear plans and stages (Thesing, Feldmann, & Burchardt, 2022)

Recently, many organizations have been trying to figure out which method works best for their projects and specific needs. Knowing the pros and cons of Agile and Waterfall in real-world situations can help project managers, developers, and other stakeholders make better decisions about how to run their projects (Awad, 2005)

Waterfall, for example, is a traditional method that starts by gathering all requirements, followed by detailed design and development. Because of its structured nature, it's often called "heavyweight." However, as the world of software development changed, new methods like Agile were created to handle the unpredictability and constant changes in projects. Agile, which focuses on improving in small steps, first came into play in 1975 (Awad, 2005)

This study will compare Agile and Waterfall, focusing on two important areas: how they handle the development process and manage risks. To make this comparison more practical, we look at a real example—developing an archive management system for the Computer Science Faculty at Herat University. The system's purpose is to make it easier to store, find, and manage academic records, improving both efficiency and access.

The results of this comparison, based on real data, will show how Agile and Waterfall work in an academic project setting. This research will also give useful tips on which method to choose, based on actual evidence and insights from the case study.

By the end of this study, readers will have a better grasp of the key differences between Agile and Waterfall, helping them choose the best approach for their projects, depending on their needs and working environments.

1.2 Problem Statement:

Developing software projects like an archive management system should be efficient, seamless, and fully meet stakeholder requirements. The chosen project management methodology should support smooth development processes, and powerful risk management to ensure timely delivery and high-quality outcomes.

However, project managers and developers often struggle to select the most appropriate methodology between Agile and Waterfall. Both offer distinct advantages and disadvantages. The lack of experiential data comparing these methodologies in academic settings creates uncertainty and potential inefficiencies in project execution. This uncertainty can prevent informed decision-making, leading to suboptimal project performance, increased risks, and stakeholder dissatisfaction.

Consequently, this gap in knowledge and indecision can result in significant issues, such as delays, increased costs, and compromised project quality. Inadequate stakeholder interaction and poor risk management can further intensify these problems, leading to projects that fail to meet their objectives.

1.3 Research Objective:

1.3.1 Main Objective:

Comparing agile and waterfall in the archive management system of Herat computer science faculty.

1.3.2 Sub Objective:

Evaluating the development process in both methodologies

Comparing e-risk identification and mitigation strategies in both methodologies.

1.4 Research Question:

1.4.1 Main Question:

How to compare waterfall and agile in the Herat computer science faculty archive management system?

1.4.2 Sub Question:

1- How to Evaluate the development process in both methodologies?

2- How to compare risk identification and mitigation strategies in both methodologies?

1.5 Research Significance:

1.5.1 Effective risk mitigating strategy:

The evaluation of risk management strategies in Agile and Waterfall methodologies will identify specific practices that effectively mitigate project risks in academic contexts.

1.5.2 Academic Practice:

Findings from this study will inform academic practice by providing empirical evidence on project management methodologies within computer science education.

1.5.3 Institutional Decision-Making:

The research outcomes will support institutional decision-making processes related to project management methodologies, by providing evidence-based recommendations.

1.5.4 Software Engineering Practices:

Beyond its immediate academic focus, the research will contribute valuable insights to software engineering practices. Industry practitioners can improve these insights to adapt and refine their project management approaches.

1.5.5 Guidance for Methodology Selection:

This research by rigorously comparing Agile and Waterfall methodologies in the context of developing an archive management system, will offer practical guidance on selecting the most suitable approach.

1.6 Research methodology and material:

1.6.1 Case study:

A case study is developing an archive management system and evaluating risk management, development process, and stakeholder interaction in this system. Using Trello for managing agile and waterfall methodology for this system.

1.7 Research structure:

1.7.1 Introduction:

Overview and introduction of topics like comparing agile and waterfall in different areas.

1.7.2 Literature Review:

Reviewing some previous Studies related to the research.

1.7.3 Method and material:

Upon what methods and material is this research performed?

1.7.4 Result:

Showing the results in tables or graphs.

1.7.5 Discussion:

Interpretation of research findings after developing the system.

1.7.6 Conclusion:

Summary of the key findings of the research.

Chapter 2: Literature Review

(Petersen & Wohlin, 2010) Have done several comparative studies to evaluate the effectiveness of Agile and Waterfall methodologies in different contexts. For instance, a study compared Agile and Waterfall in terms of productivity and quality in software projects. Their findings suggest that Agile methods generally lead to higher productivity and better quality outcomes, particularly in projects with changing requirements.

(Awad, 2005)compared the two methodologies in terms of customer satisfaction and project success rates. The study found that Agile projects tend to have higher customer satisfaction due to continuous involvement and feedback mechanisms, while Waterfall projects are more successful in environments with stable and well-defined requirements.

(Thesing T. F., 2021) have done research (based on 15 expert interviews) and an experiential survey (carried out as semi-structured, guided interviews with interview partners drawn from various sectors, company sizes, and age groups.) with the result of the decision model, no procedural model is a "silver bullet" for all types of projects. Each procedural model is particularly well suited for particular project types with defined criteria. If these project criteria are not met or are met only to a limited extent, the approach is likely to fail.

(Gaborov, et al., 2021) Studied an IT system and Stated that there is no single recipe for project management and leadership due to the wide range of problems and requirements that arise. There is no best methodology to use. Depending on the nature of the problem, the appropriate methodology will be selected. If the project has clearly defined requirements and goals, some of the traditional methodologies will be used, but for projects with unstable requirements, some will always be used agile methodologies or combinations because they are adaptable SO, IT was concluded that the agile Scrum methodology is mostly used in IT companies, with the combination of several methodologies often appearing, due to the need for projects, to eliminate the shortcomings of each methodology.

(Andrei, Casu-Pop, Gheorphe, & Boiangiu, 2019) Proposed a practical study by analyzing the results of a survey designed to capture the experience of developers with the agile and waterfall methodologies. They focused on the Scrum and Kanban Agile methods and Waterfall to analyze the findings of the study and concluded that there is no silver bullet solution when it comes to choosing the methodology for a project, as numerous factors need to be accounted for. The waterfall will be a better solution for small projects that have well-defined requirements that will

not change, while Agile is preferred when continuous delivery and feedback are important, requirements are not well-defined and time to market is more important than releasing a full feature version.

(Maassen, 2018) analyzed two of the main product development business model trends in the IT field, namely the Waterfall model and the Agile model, the latest being an adapting strategy to increased customer requirements and the changing business environment. To ensure a practical approach, the case study was based on the analysis of their implementation within the company Avira Soft S.R.L. The results of the study emphasized the benefits of using the Agile model at Avira Soft S.R.L starting in 2011 in comparison to the previous model of Waterfall product development.

(casteren & Wilfer van, 2017) Investigate both methodologies agile and waterfall and he Stated that both models have their uses, advantages, and disadvantages. Small projects are almost always suitable for an Agile approach and rarely for a Waterfall approach. A big and complex project with multiple teams working simultaneously on different parts of the application is almost always a Waterfall project.

(Fortaleza, 2023) stated that A hybrid methodology is a possibility to join the best of both worlds, and it could be achieved by following as suggested in the new PMI's PMBOK Version 6, in which the agile execution will assume an essential advantage for project managers in the construction industry within the Macau environment. As a future work of this research, to have a broader point of view, it would be necessary to have a larger population of Macau's hotel construction industry as part of the survey.

(Itamar, Michel dos, Joseph , Jan van, & Jos, 2010) Have done research by comparing two case studies one small project and another large project they considered the development process in both projects to evaluate which methodology is more suitable for the project's process management they concluded that in the small project, the RUP methodology is the best and FDD methodology is in second position in another hand they concluded that in large project XP and scrum suit the best.

Chapter 3: Method and Material

3.1 Research methodology:

3.1.1 Instrumental Case Study:

An instrumental case study is a qualitative research method used to gain a deeper understanding of a particular issue or phenomenon through investigating a specific case, unlike intrinsic case studies, which focus on the case itself because of its unique features, instrumental case studies are conducted when the case is of secondary interest and serve primarily as a tool to provide insight into an issue or to refine a theory.

The key idea behind an instrumental case study is that the researcher examines a case not just for its own sake, but to gain insights that can be generalized or applied to other cases. This method is particularly valuable when the researcher wants to understand a bigger issue, such as the effectiveness of a few practices or methodologies, by closely examining a specific instance.

The instrumental case study methodology was chosen for this research because it provides the opportunity to deeply examine the development and comparison of Agile and Waterfall methodologies in a controlled, real-world environment by developing an archive management system and focusing on this specific case, the study aims to generate wisdom that can be generalized to issues in software development, such as risk management, stakeholder involvement, and the effectiveness of different project management approaches. The instrumental case study allows for a detailed examination of the archive management system's development process, and risk management, for understanding the strengths and weaknesses of Agile and Waterfall methodologies. By documenting and analyzing the project's improvement using Trello boards for different aspects of project management. This in-depth analysis helps address the research problem by offering practical insights and evidence-based conclusions that can inform future projects and decision-making.

3.1.2 Process of conducting the research methodology:

3.1.2.1 Case Selection:

The archive management system was selected as the case for this study due to its relevance to the research question and its potential to provide valuable insights into software development methodologies.

3.1.2.2 Development Process:

I developed the system using both Agile and Waterfall methodologies simultaneously. Trello boards were used to manage the different aspects of the project. Figure 1 shows the development process board for the waterfall It contains all the phases of the waterfall and each phase has different cards that show the task that should be done in the project.

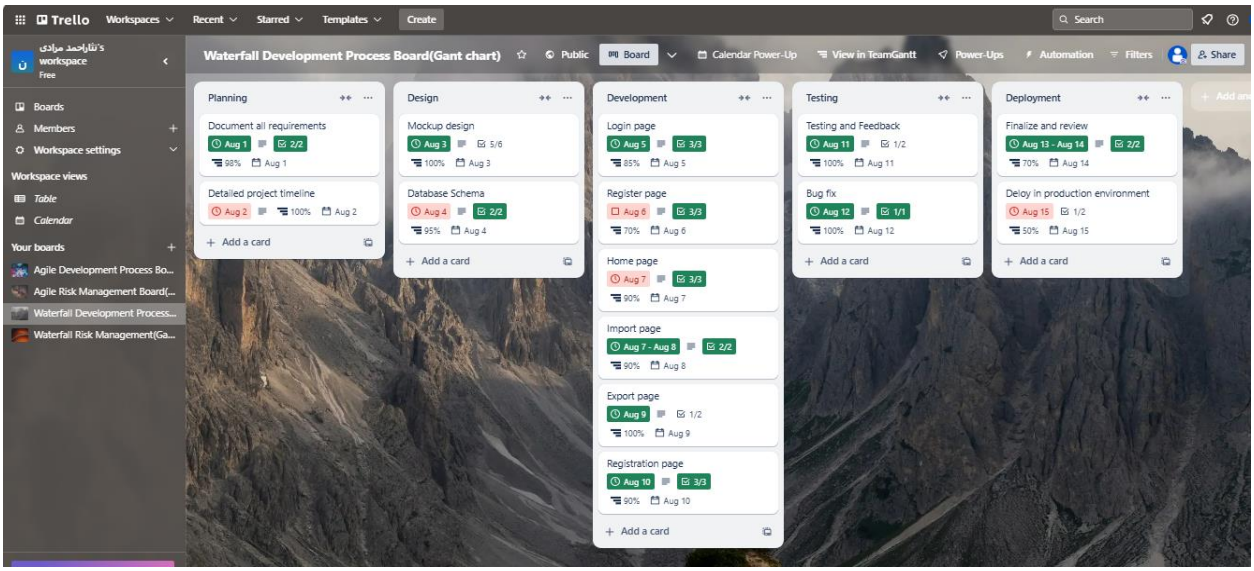

Figure 1waterfall board

figure 2 shows the Agile Trello board it contains all the sprints and tasks that stakeholders gave after meetings every checkbox shows one task

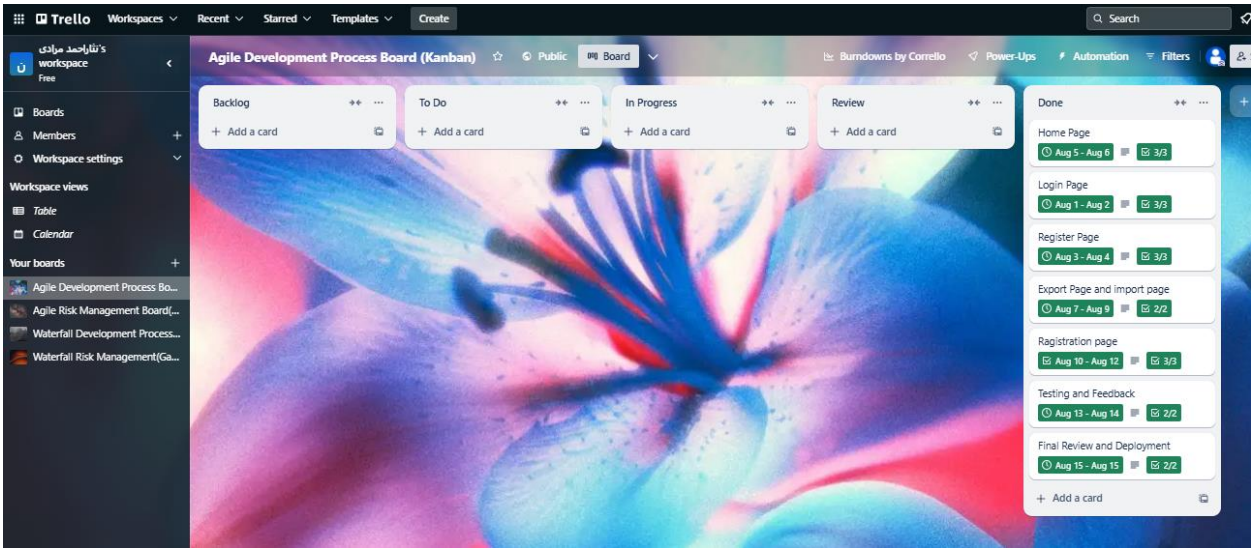

Figure 2: Agile Trello board

3.1.2.3 Risk Management in Agile:

All the risks that have a probability of happening were listed and risks gathered by helping a few developers in the area of programming and they registered in Trello as they are shown in Figure 3 the Trello board risk management for agile has three lists 1- Identified: Shows all the risks that were identified. 2- In progress: Shows all the risks that were in progress of mitigation. 3-Mitigated: Shows all the risks that were mitigated and they will not create problem any more

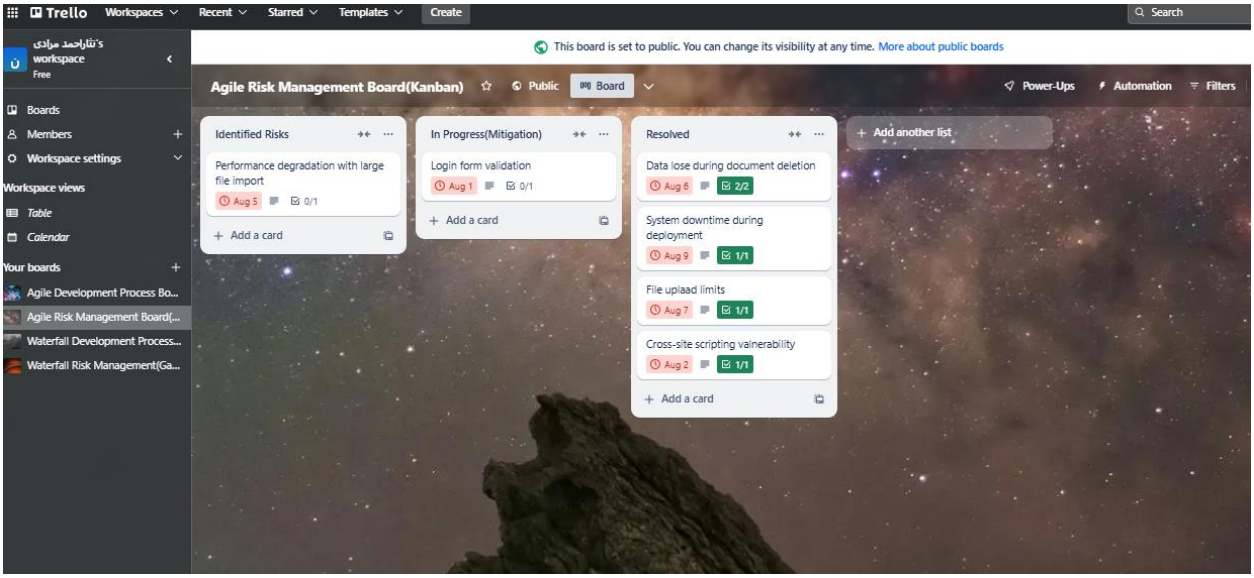

Figure 3: Agile Risk Management Board

3.1.2.4 Risk Management in Waterfall:

As I explained in risk management in the agile subject the way of finding risks I found the probable risk for the waterfall in the same way Figure 4 shows the Waterfall risk management board. It also has three lists: 1- Planning risks: Shows all the risks that were identified in the planning phase of the waterfall methodology. 2- Development risks: Shows all the risks identified in the development phase of the waterfall methodology. 3- Tasting risks: Shows all the risks that were identified in the tasting phase of the waterfall methodology.

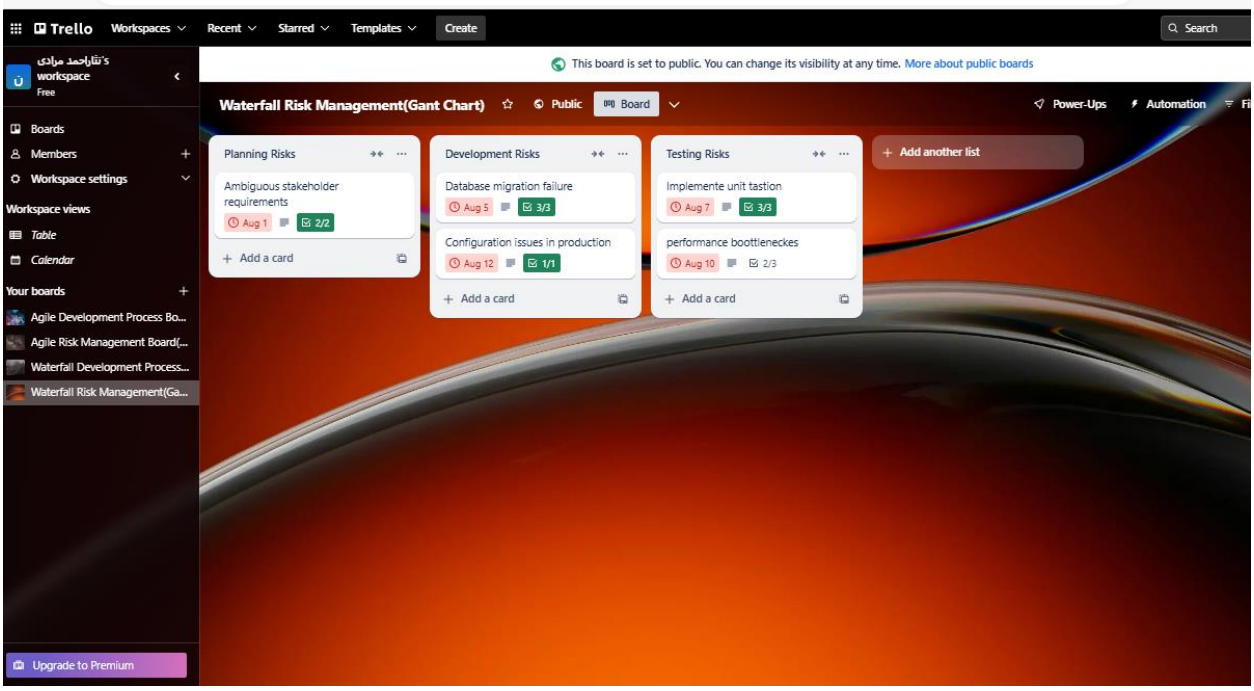

Figure 4: Waterfall Risk Management Board

3.1.2.5 Gantt Chart Board for Waterfall:

This was a power-up in the Trello board that used to see the task completion rate in the Waterfall, the average time per task, and the timetable for the Waterfall development process board. Figure 5 shows the Gantt chart board for the waterfall.

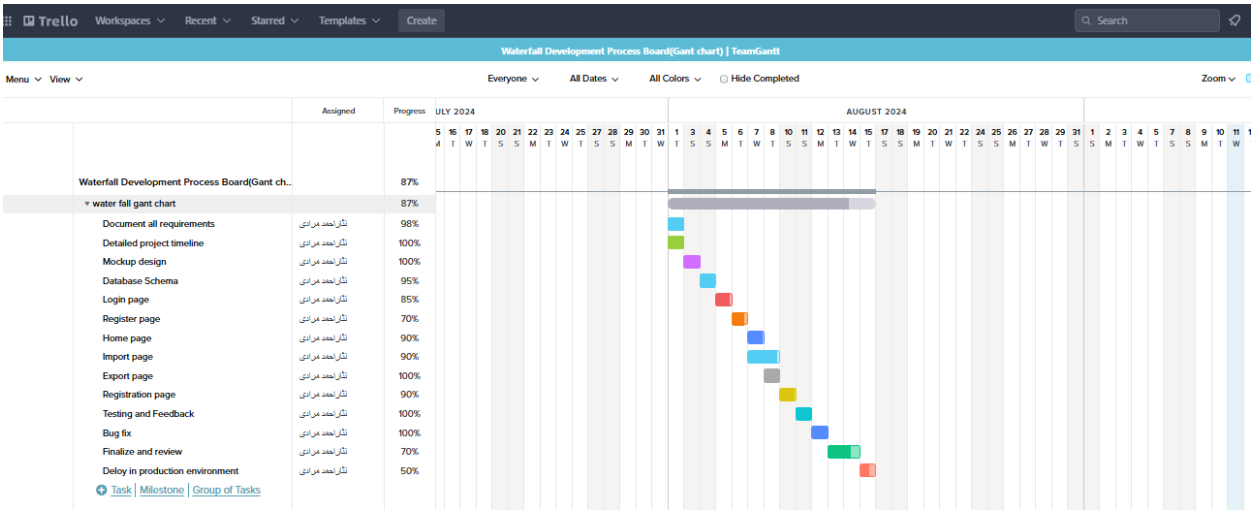

Figure 5: Gant Chart For Waterfall

3.1.2.6 Kanban Board for Agile:

Used to visualize tasks and workflow in Agile, allowing for flexibility and continuous delivery of project components Figure 6 shows the Kanban board for Agile with all sprints and backlogs.

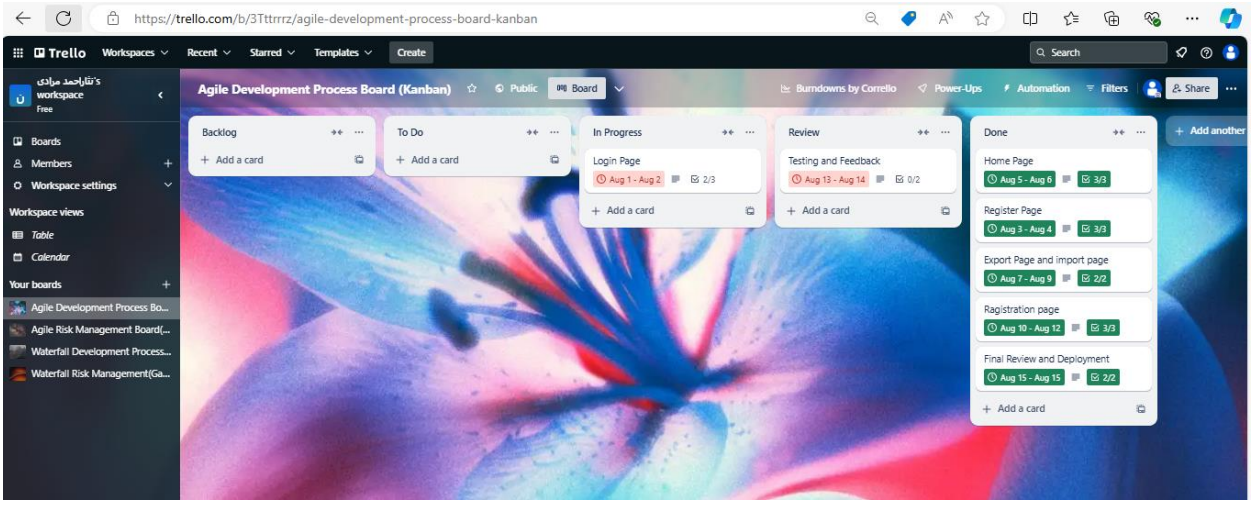

Figure 6: Kanban board for agile

3.1.2.7 Data Collection:

Data was collected throughout the development process, including project timelines, task completion rates, risk occurrences, and stakeholder feedback, by using Trello boards. This data was systematically documented and categorized for analysis.

3.1.2.8 Analysis:

The data was analyzed to identify patterns and differences in how Agile and Waterfall's methodologies addressed various aspects of the project, such as efficiency, risk management, and development process. The findings were contextualized within the broader discussion of software development best practices.

3.2 Materials:

3.2.1 Waterfall:

The Waterfall model was initially introduced in the 1960s as a structured approach to software development for large-scale systems, especially in industries like defense and space, which involved long-term, multi-year projects. This model was particularly appropriate for large computer systems, such as mainframes and minicomputers, which evolved slowly. The concept

was modeled after hardware design, where engineers could often predict system interactions with relative certainty, but without challenges. Winston Royce is frequently credited with formally describing the Waterfall model in 1970, though his original paper criticized the method as inadequate for most software projects, emphasizing its flaws rather than advocating for its use (Royce, 1970)

The Waterfall model performs well in stable environments where incremental changes are made to an already established system. It is effective when project managers can control changes in design details from start to finish, with minimal unplanned adjustments. However, this approach struggles in dynamic environments, such as the personal computer software market, where rapid changes in hardware and customer requirements occur. In these cases, the linear, sequential nature of Waterfall—progressing from design to coding, followed by testing—becomes a limitation.

When specifications are defined too early, and unexpected changes occur, the team may deliver a product that no longer meets user needs or is obsolete upon release. Attempts to adapt during development can lead to poorly integrated components, requiring significant rework during the testing phase. As a result, many projects encounter delays, exceed budgets, and suffer from quality issues due to difficulties in integrating evolving pieces of the system (Cusumano & Smith, August 16, 1995)

3.2.1.1 Characteristics of Waterfall:

Predictive Approach: The Waterfall model is based on a linear, sequential process, where each phase is planned.

Comprehensive Documentation: Detailed documentation is created at the start, defining all project requirements.

Process-Oriented: The model emphasizes strict adherence to a predefined set of processes.

Tool-Oriented: It relies heavily on tools for documentation, planning, and managing the stages.

3.2.1.2 Advantages:

Simplicity and Ease of Use: The step-by-step nature of Waterfall makes it simple, especially for teams familiar with structured project management.

Manageability: The rigid phase structure makes managing the project easier, as each phase has clear deliverables and a review process.

Sequential Completion: Tasks are processed and completed one by one, ensuring that the previous stage is fully finalized before moving to the next.

Best for Small Projects: The Waterfall model is ideal for smaller projects where the requirements are well understood and unlikely to change.

Organizational Structure: It provides a well-defined structure that helps in organizing and controlling the development of software projects.

3.2.1.3 Disadvantages:

Lack of Flexibility: Once the project is underway, changing the scope can be extremely challenging and may lead to project failure.

Delayed Results: The Waterfall model often produces working software only late in the development cycle, limiting early feedback and adjustments.

High Risk and Uncertainty: Due to its rigidity, the Waterfall model involves significant risks, especially when uncertainties or unforeseen issues arise.

Unsuitability for Complex Projects: It is not ideal for large, complex, or object-oriented projects, which require more flexibility and adaptability.

Ineffective for Long-Term Projects: The model struggles with ongoing projects that require iterative improvements and continuous updates.

Changing Requirements: The Waterfall model performs poorly when project requirements are subject to frequent changes, making it unsuitable for dynamic environments.

3.2.1.4 Waterfall Lifecycle:

Requirements Gathering: In this initial phase, all project requirements are collected and documented in detail. This step is critical, as the Waterfall model does not easily accommodate changes. Project managers must ensure that every requirement is captured accurately.

Design: This phase focuses on designing the user interface (UI) and the overall system architecture based on the gathered requirements. This helps in laying out the blueprint for development.

Development: Once the design is finalized, developers begin coding the software according to the specifications set during the design phase.

Testing: The software is rigorously tested to ensure that it functions as expected and that any bugs or issues are addressed.

Deployment: In the final stage, the fully developed and tested software is delivered to the customer or deployed for use.

3.2.2 Agile:

Agile software development is based on principles outlined in the Agile Manifesto, established in 2001 by a group of software developers. The group later formed the Agile Alliance, a nonprofit organization dedicated to promoting and refining the philosophies of Agile development (Beck et al., 2001). Agile is centered around incremental and iterative development, where each phase of the software life cycle is revisited multiple times to ensure continuous improvement. Customer feedback plays a crucial role in driving iterations, which helps in progressively refining the solution. Unlike traditional, plan-driven models, Agile emphasizes adaptability and flexibility, dividing the development process into smaller units called "increments" or "iterations." Each iteration encompasses all the conventional phases of software development, including design, coding, and testing (Leau, Khong Loo, Tham, & Tan, 2012)

According to the Agile Manifesto, four key principles are foundational to Agile practices:

1- Early and Continuous Customer Involvement

2- Iterative Development

3- Self-organizing Teams

4- Adaptation to Change

There are various methodologies under the Agile umbrella, including Scrum, Extreme Programming (XP), Feature-Driven Development (FDD), Crystal Methods, Lean Software Development, and Dynamic Software Development Method (DSDM).

3.2.2.1 Extreme Programming (XP):

XP focuses on improving software quality and responsiveness to evolving customer needs. A key practice in XP is the daily stand-up meeting, where progress is reviewed, and potential issues are addressed immediately. The iterative nature of XP ensures that project velocity is constantly monitored and any deviations are corrected promptly (Beck, 2004).

3.2.2.2 Scrum:

Scrum is one of the most widely adopted Agile frameworks, especially in software development. It is a project management framework that emphasizes short development cycles, known as sprints, typically lasting 2-4 weeks. Scrum allows teams to manage complex projects by breaking them into smaller, manageable tasks, making it easier to adapt to changing requirements. Teams in Scrum are typically small (5-9 members), though Scrum can scale to larger organizations and projects (Schwaber & Sutherland, 2013).

3.2.2.3 Feature-Driven Development (FDD):

FDD is a client-centric methodology focusing on defining and refining the project's architecture throughout its lifecycle. The process involves building a detailed model of the system and continuously refining it to meet evolving requirements (Felsing & Palmer, 2002)

3.2.2.4 Crystal Method:

Developed by Alistair Cockburn, Crystal Methods prioritizes the human elements of software development over tools or rigid processes. Crystal is designed to be flexible and scalable, with more comprehensive frameworks required for large or safety-critical projects, while smaller, less critical projects can follow simpler methods (Cockburn, 2004)

3.2.3 Characteristics of Agile Methodologies:

As noted by Highsmith and Cockburn, Agile is distinct not for its practices but for the mindset it brings to software development. The core idea is that people, not processes, are the driving force behind successful projects. Agile combines flexibility, simplicity, and collaboration to enhance responsiveness and maneuverability (Highsmith & Cockburn, 2001)

The key characteristics that differentiate Agile from traditional methods include:

1- Adherence to Reality

2- Balancing Flexibility and Planning

3- Empirical Process Control

4- Decentralized Decision-Making

5- Simplicity

6- Collaboration

7- Small, Self-Organizing Teams

3.2.4 Advantages of Agile:

Flexibility and Adaptability: Agile allows teams to respond to changes rapidly.

Customer Involvement**:** Continuous feedback from customers ensures the product aligns with their expectations.

Faster Delivery: Short iterations lead to quicker delivery of functional software.

Improved Quality: Frequent testing during iterations helps in identifying and fixing issues early.

Enhanced Team Collaboration: Agile fosters a collaborative environment where teams work closely together.

Higher Productivity: Breaking work into smaller chunks helps teams maintain focus and complete tasks efficiently (Leau, Khong Loo, Tham, & Tan, 2012)

3.2.5 Disadvantages of Agile:

Lack of Predictability: Agile can be unpredictable, especially when scope and requirements change frequently.

Scope Creep: Without careful management, Agile projects may suffer from scope creep as requirements evolve.

Resource Intensive: Agile requires significant team involvement and dedication, which can be resource-intensive.

Fragmented Documentation: Agile focuses more on working software than on comprehensive documentation, which can create gaps.

Team Dependency: Agile success heavily depends on strong collaboration between all team members, making it challenging for teams not used to self-organization.

Management Challenges: Managing Agile teams requires a different skill set, focusing on facilitation rather than control.

Not Suitable for All Projects: Agile may not be the best choice for projects with fixed requirements or regulatory constraints (Cockburn, 2004)

3.2.6 Computer:

Dell, 6th Generation with 8GB RAM and 500GB HDD this computer provided a solid foundation for my software development work. The 8GB RAM ensured smooth multitasking and the ability to run demanding Integrated Development Environments (IDEs) like VSCode and heavy-duty frameworks like Spring Boot. The 500GB HDD offered enough storage for software libraries, databases, and development tools. The Dell system's specifications were sufficient for handling the workload of developing an archive management system, managing tasks, and running database operations.

3.2.7 Archive management system:

The development of this system was chosen as a case study for this research it has 6 pages and each one of them does a separate work

1- login page: On this page, users can log in and go to the home page if they do not have they will be redirected to the registration page and register themselves then come to the log-in page enter their username and password, and go to the home page, Figure 7 shows the login page of the system.

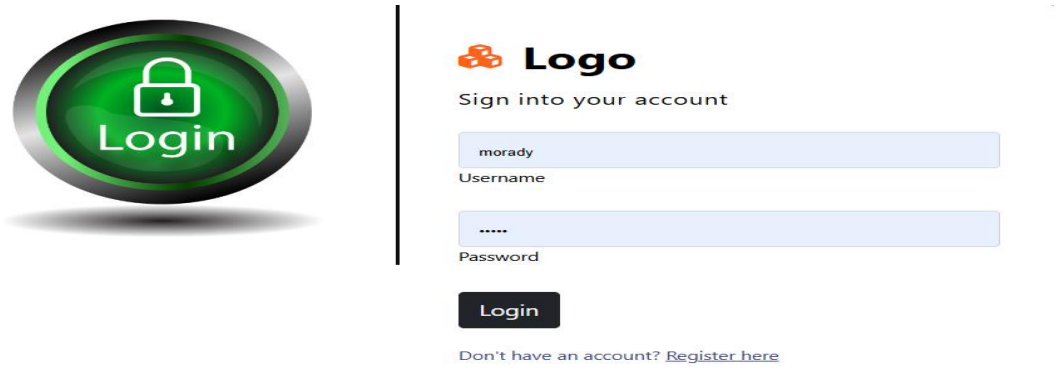

Figure 7: Login page of the system

2-Registration page: On this page, users can register with a user ID and password. When they register themselves they will be redirected to the login page, Figure 8 shows the registration page.

Figure 8:Register page of the system

3- Home page: On this page, users can see the information about the system and go to all other pages by using the links that exist in the app bar of the system, Figure 9 shows the home page of the system.

Figure 9: Home page of the system

4- Import document page: This page shows all the import documents and documents can be deleted and downloaded on this page Figure 10 shows this page.

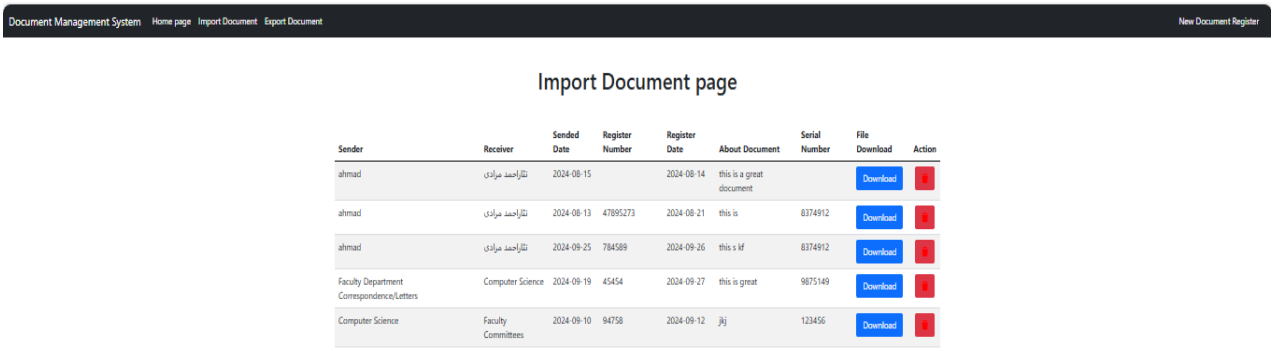

Figure 10: Import document page

5- Export document page: This page shows all export pages like the import page and Figure 11 shows the export documents page.

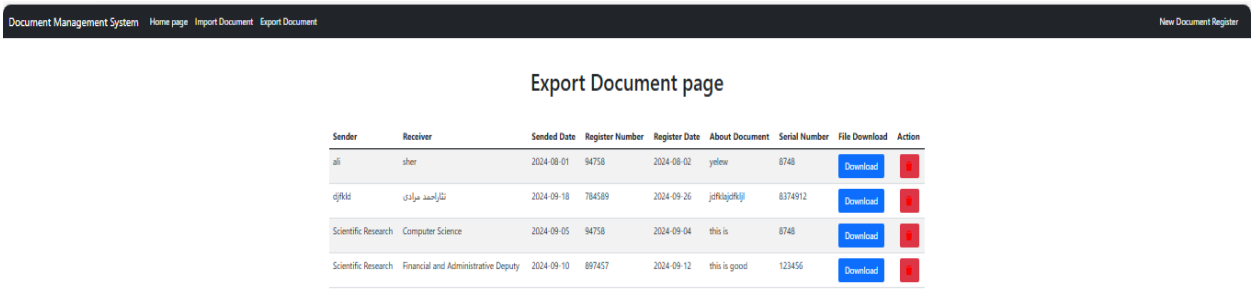

Figure 11:Export document page

6- Registration document page: On this page, users can save a new document of every type that they want and it will save the document into an export document if its type is exported, and if the type of document is imported it will save the document in import page Figure 12 shows the registration page document.

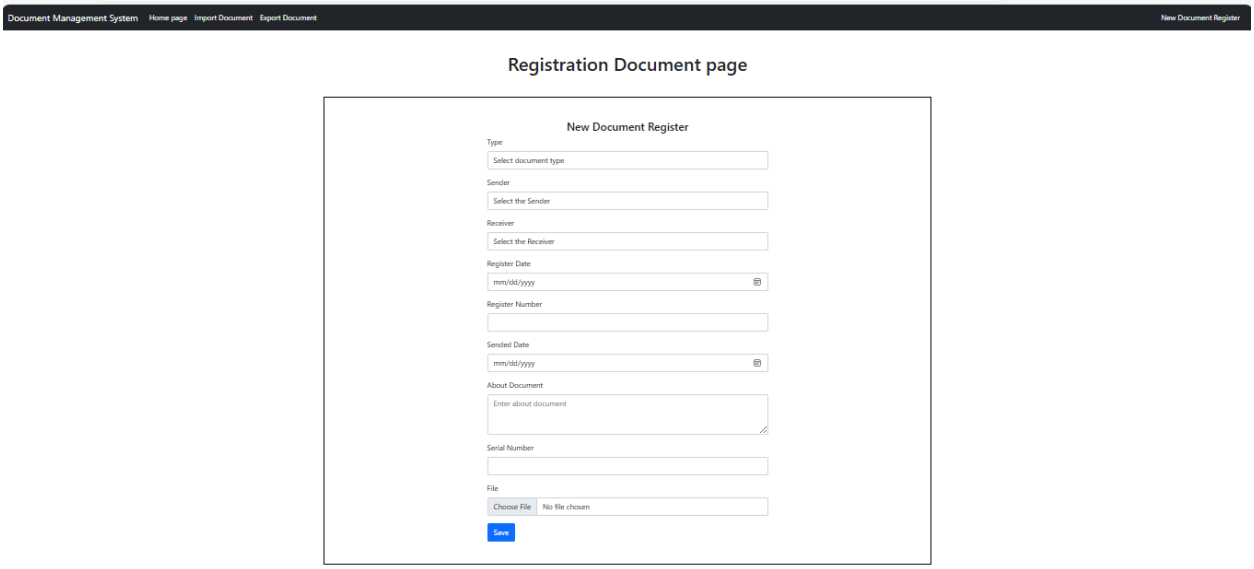

Figure 12: Registration document page

3.2.8 Visual Studio Code (VSCode):

VSCode was my primary development tool, known for its lightweight performance and extensive customization options. It supports a wide range of programming languages and has various extensions that streamline coding, debugging, and collaboration. I utilized specific extensions such as:

3.2.9 Java Extension Pack:

This extension includes tools like Maven, Java Debugger, and Java Test Runner, which were essential for writing, testing, and debugging Java code in my project.

3.2.10 Spring Boot Extension Pack:

Spring Boot was the framework used for my project, and this extension provided vital tools like Spring Initializer for bootstrapping my application and support for Spring-specific annotations and configurations. This made it easier to develop, test, and run your web application efficiently.

3.2.11 Spring boot initializer:

This was the Grates web page that helped me to initialize the first time my Archive management system and I was able to install all dependencies and choose the version of Java, write the name of the project, choose Cradle or Maven for my system, and all other necessary things.

3.2.12 Google Scholar:

Google Scholar is a necessary research tool, Google Scholar was used a lot to access scholarly articles, research papers, and other academic resources. I used this platform to study related works, gather theoretical foundations, and validate your research methodology on Agile and Waterfall. By reading almost 100 articles, Google Scholar provided a wide range of literature, allowing me to thoroughly analyze project management methodologies, risk management strategies, and case studies, which contributed to the academic validity of my research.

3.2.13 Research Gate:

Research Gate offered a valuable platform to access academic articles and research material. I used this platform to improve my understanding of Agile and Waterfall methodologies, gaining insights into their applications in different contexts. Research Gate also provided a means of accessing practical case studies and communicating with other researchers in the field.

3.2.14 YouTube:

YouTube is known for its vast range of educational content, YouTube is a critical learning resource, especially when studying new concepts and methodologies. It provided tutorials on Agile and Waterfall project management, and research methodologies that helped shape my method Additionally, I used YouTube to troubleshoot technical challenges during the system development phase, such as tutorials on Spring Boot, MySQL integration, and debugging.

3.2.15 AI (Artificial Intelligence):

I used AI tools, such as coding assistants and error fixers, to solve problems encountered during development. AI was especially useful for error handling during the development of my system.

3.2.16 MySQL:

MySQL was the relational database management system used in my project for storing, retrieving, and managing data related to the archive management system. MySQL allowed me to:

1- Create databases and tables to organize system data.

2- Perform CRUD operations create, read, update, and delete to manage records for documents, users, and system logs.

3- Ensure data unity and scalability, as MySQL is known for handling large datasets efficiently.

3.2.17 Spring Boot:

This Java-based framework was central to developing my archive management system. Spring Boot simplifies the development of RESTful web services and enterprise-level applications, allowing rapid application development with minimal configuration. The key dependencies I used included:

3.2.18 JPA (Java Persistence API):

For managing data persistence, allowing interaction with the MySQL database using object-relational mapping (ORM).

3.2.19 Web Dependency:

To develop RESTful web services, which enabled the system to handle HTTP requests and serve data to the front end.

3.2.20 Dev Tools:

For hot-reloading and faster development, which streamlined the process of testing and deploying changes.

3.2.21 MySQL Connector:

Integrated MySQL with my Spring Boot application, enabling seamless database interaction.

Spring Boot's powerful annotations, auto-configuration, and embedded server capabilities allowed me to focus on building business logic without worrying about infrastructure setup.

3.2.22 Trello:

Trello (Kanban and Gantt Chart Power-Up): Trello was used as a project management tool to track and manage tasks for both Agile and Waterfall methodologies. Two key features were utilized:

Chapter 4: Result

This chapter presents the comparative analysis of Agile (Kanban) and Waterfall (Gantt Chart) methodologies based on the development of an archive management system for the Computer Science Faculty at Herat University. The analysis focuses on two key aspects: Development Process and Risk Management. The data is presented in tables for clarity.

4.1 Comparative Results Table:

After collecting data from both methodologies, this is a comparative table to visualize differences and insights.

Metric	Agile(kanban)	Waterfall(gant chart)
Total tasks	18	33
Total task completed	15	29
Total task planed	19	33
Task completion rate	83.3%	87%
Average time per task	5 h	2.7 h
Total risk identified	6	5
Total risk mitigate	4	4
High impact risks	3	2
Risk frequency	3	3

Table 1: Comparing Agile and waterfall

4.2 Development Process Metrics:

The development process was measured by tracking the time and completion of tasks across both Agile and Waterfall methodologies. The six major tasks in the project (Login Page, Register Page, Home Page, Import Page, Export Page, Registration Page) were used to compare the efficiency and flexibility of each methodology.

Task	Methodology	Estimated Time (hrs.)	Actual Time (hrs.)	Sprint/Phase	Completion Date
Login Page	Agile	6	4	Sprint 1	Aug 2
Register Page	Agile	5	6	Sprint 1	Aug 4
Home Page	Agile	2	3	Sprint 2	Aug 6
Import Page	Agile	3	4	Sprint 2	Aug 8
Export Page	Agile	3	5	Sprint 2	Aug 10
Registration Page	Agile	5	8	Sprint 2	Aug 12
Login Page	Waterfall	3	2.5	Development	Aug 2
Register Page	Waterfall	3	1	Development	Aug 4
Home Page	Waterfall	4	1.5	Development	Aug 6
Import Page	Waterfall	1	2	Development	Aug 8
Export Page	Waterfall	2	1.5	Development	Aug 10
Registration Page	Waterfall	3	5	Development	Aug 12

Table 2: Development process metrics

4.3 Risk Management Metrics:

Risk management metrics were tracked to measure how each methodology handled the identification and mitigation of risks. The key risks involved in the project included form validation issues, file upload limits, bugs, and scope creep.

Risk	Methodology	Impact	Frequency	Phase	Impact on Timeline (days)	Mitigation Date
Login Form Validation	Agile	Medium	1	Development	0	Aug 2

File Upload Limits	Agile	High	2	Import	0	Aug 8
Bugs in the Registration Page	Agile	Medium	1	Testing	2	Aug 12
Unexpected Bugs on the Export Page	Waterfall	High	1	Development	5	Aug 12
Scope Creep in Import Page	Waterfall	High	1	Planning	3	Aug 8
Slow Testing Process	Waterfall	Medium	1	Testing	2	Aug 14

Table 3: risk management metrics

Chapter 5: Discussion

This chapter contains the comparative analysis of Agile and Waterfall methodologies, considering their application in developing the archive management system for the Herat Computer Science Faculty. It will discuss two critical aspects of the project management development process and risk management.

5.1 Development Process:

The approach of the software development process is different in agile and waterfall. The Agile methodology, characterized by its iterative nature, allowed the team to accept continuously changing requirements. Agile's strength is its flexibility and ability to make changes at any stage of development. For example, Agile's ability to introduce tasks in the middle of development, respond to feedback, and revisit unfinished work highlights its adaptability. In contrast, the Waterfall model emphasizes a more structured and linear approach, where each phase is completed before moving to the next. This method is beneficial for projects with well-defined requirements but struggles when changes arise.

As shown in Table 1, In this study, Agile completed 15 out of 18 tasks, while Waterfall completed 29 out of 33 tasks, showing that Waterfall achieved a slightly higher task completion rate (87% vs. Agile's 83.3%). However, Agile proved to be more efficient in responding to stakeholder feedback and adapting to issues as they arise, which reflects the dynamic nature of the archive management project.

Waterfall surpasses in its time efficiency, with an average task completion time of 2.7 hours compared to Agile's 5 hours per task. This suggests that Waterfall is more appropriate for projects with clear and stable requirements, as it enforces a more rigid timeline with defined phases. However, in projects like the archive management system, where requirements evolved and needed frequent adjustments, Agile's flexibility provided a better fit despite the longer task duration.

5.2 Risk Management:

Risk management is another area where Agile and Waterfall methodologies have significant differences. Agile's iterative approach encourages continuous risk identification and mitigation. Throughout the project, as shown in Table 3 Agile identified six risks and mitigated four, with high-impact risks being managed effectively with minimal disruption to the timeline. Agile's risk management board, organized into identified, in-progress, and mitigated categories, allowed for real-time tracking of issues and quick responses to potential challenges.

Waterfall, on the other hand, identified five risks, mitigating four. However, Waterfall faced more significant delays due to high-impact risks, such as unexpected bugs and scope creep. These delays, especially when issues occurred late in the development cycle, proved more challenging to address within Waterfall's linear framework. For instance, one high-impact risk caused a 5-day delay, underscoring the limitations of Waterfall when unplanned events or issues arise.

Agile's strength in risk management is its ability to continuously reassess risks and make adjustments. In contrast, Waterfall's risk management tends to be more reactive, as risks are often identified and addressed in later phases, making it difficult to correct course without substantial delays.

5.3 Suitability for Project Types:

This comparative study shows that both Agile and Waterfall have their strengths depending on the project context. For projects with well-defined requirements and minimal expected changes, Waterfall's structured approach provides clear advantages in time management and task tracking. However, for projects like the archive management system, which involved evolving requirements and the need for constant stakeholder feedback, Agile's adaptability was more beneficial.

The archive management system, as a project, required frequent updates and changes based on stakeholder input. Agile's iterative nature allowed the project team to incorporate these changes without disrupting the overall workflow. Waterfall, though efficient in completing tasks once the plan was set, struggled with accommodating unanticipated changes, leading to delays and additional resource allocation.

Chapter 6: Conclusion

This monograph discussed the challenge of choosing the best project management methodology for developing an archive management system for the Computer Science Faculty at Herat University. The study emphasized project managers' and developers' difficulties when deciding between Agile and Waterfall methodologies. Both approaches have their benefits, but the lack of comparative data. This knowledge gap often leads to less effective decisions, resulting in delays, higher costs, and lower project quality. The main goal of this research was to compare Agile and Waterfall methodologies in the context of the archive management system. Two key areas are the development process and the effectiveness of risk identification and mitigation strategies.

Agile completed 15 out of 18 tasks, while Waterfall completed 29 out of 33 tasks. Despite Waterfall's higher task completion rate (87% vs. 83.3%), Agile demonstrated greater flexibility in adjusting task timelines. However, Waterfall was more efficient in terms of average time per task, taking 2.7 hours compared to Agile's 5 hours. This suggests Waterfall may be more suitable for tasks with well-defined requirements, while Agile provides better adaptability during the development process, especially when facing changes or uncertainties.

Both methodologies identified and mitigated a similar number of risks (6 identified in Agile, 5 in Waterfall), and both managed to mitigate 4 risks. However, Agile encountered more high-impact risks (3 compared to 2 in Waterfall) but handled them more efficiently with minimal impact on the project timeline. Waterfall faced more significant delays, particularly with high-impact risks like unexpected bugs and scope creep, which delayed the project by up to 5 days. Agile's iterative process allowed for quicker risk identification and mitigation, minimizing disruptions.

This study found Agile's flexibility and proactive risk management made it more appropriate for this project, especially in managing changes and addressing risks effectively. Waterfall, with its structured approach, proved more efficient in terms of time per task but struggled with adaptability and managing unexpected issues. For projects requiring adaptability, like the archive management system, Agile presents a more suitable option, while Waterfall is preferable for projects with stable, clearly defined requirements.

References

- Andrei, B. A., Casu-Pop, A. C., Gheorphe, S. C., & Boiangiu, C. A. (2019). A study on using waterfall and agile methods in software project management. *Journal of Information Systems & Operations Management, 125-135*, 11.
- Awad, M. (2005). A comparison between Agile and traditional software development. *The University of Western Australia*, 84.
- Beck, K., Beedle, M., VanBennekum, A., Cockburn, A., Cunningham, W., Fowler, M., & Jeffries, R. (2001). *agilemanafasto.com*. Retrieved from http:/manafasto.com
- Begel, A., & Nagappan, N. (2007). Usage and Perceptions of Agile Software Development in an Industrial Context. *First International symposium on empirical software*, 6.
- Boehm, B., & Turner, R. (2023). Using risk to balance Agile and plan-driven methods. *Computer, 36(6), 57-66*, 11.
- Broy, M. (2006). Challenges in automotive software engineering. *Proceedings of the 28th International Conference on Software Engineering*, 10.
- casteren, & Wilfer van. (2017). The Waterfall Model and Agile Methodologies: A comparison by project characteristics. *Open university Nederland*, 7.
- Cockburn, A. (2004). *Crystal Clear: A Human-Powered Methodology for Small Teams.*
- Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research, 20(3), 329-354*, 11.
- Cusumano, M., & Smith, S. (August 16, 1995). Beyond the Waterfall: Software Development at Microsoft. *MIT Sloan School of Management International Business Machines*, 34.
- Dingsøyr, T., Dybå, T., & Moe, N. B. (2010). Agile software development: Current research and future directions. *Springer Science & Business Media*, 9.
- Felsing, J. M., & Palmer, S. R. (2002). *A Practical Guide to Feature-Driven Development.* Prentice Hall.

- Fortaleza, R. G. (2023). EFFECTIVENESS ANALYSIS OF WATERFALL AND AGILE PROJECT MANAGEMENT METHODOLOGIES – A CASE STUDY FROM MACAU'S CONSTRUCTION INDUSTRY. *University of Saint Joseph, Macau, SAR China (v. 12, n. 1, p. 23-38*, 16.
- Gaborov, M., Karuović, D., Kavalic, M., Radosav, D., Milosavljev, S., Stanisaljev, S., & Bushati, J. (2021). Comparative analysis of agile and traditional methodologies in IT project management. *Journal of Applied Technical and Educational Sciences, 11(4), 1-ArtNo*, 24.
- Highsmith, J., & Cockburn, A. (2001). *Agile Software Development: The Business of*.
- Hoda, R., Noble, j., & Marshall, s. (2011). The impact of inadequate customer collaboration on self-organizing Agile teams. *Information and Software Technology, 53(5), 521-534*, 8.
- Itamar, S., Michel dos, S., Joseph , B., Jan van, d., & Jos, V. (2010). A DECISION FRAMEWORK FOR SELECTING A SUITABLE. *Delft University of Technology, Jaffalaan 5, 2628 BX Delft, The Netherlands*, 10.
- Leau, Y., Khong Loo, W., Tham, W., & Tan, S. (2012). Software Development Life Cycle AGILE vs Traditional Approaches. *School of Engineering and Information Technology Universiti Malaysia Sabah, Malaysia*, 6.
- Maassen, M. A. (2018). Product development models in the IT sector Waterfall to Agile Project Management models in the case of AVIRA SOFT SRL. *In Proceedings of the International Conference on Business Excellence (Vol. 12, No. 1, pp. 568-578)*, 11.
- Petersen, K., & Wohlin, C. (2010). The effect of moving from a plan-driven to an incremental. *Empirical Software Engineering,15(6), 654-693*, 40.
- PK.Raghunath, S.Velmourougan, P. Davachelvan, S.Kayalvizhi, & R.Ravimohan. (January 2010). Evolving A New Model (SDLC Model-2010) For Software Development Life Cycle (SDLC). *IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.1*, 8.

- Royce, W. W. (1970). Managing the Development of Large Software Systems. *Proceedings of EEE WESCON*, *1-9*, 11.
- Schwaber, K., & Sutherland, J. (2013). AGILE AND WATERFALL CHOOSE A DEVELOPMENT PROCESS. *The Scrum Guide. Scrum.org.*, 6.
- Sommerville. (2016). *Software Engineering*. Chicago: Pearson.
- Thesing, T. F. (2021). Agile versus waterfall project management: decision model for selecting the appropriate approach to a project. *Procedia Computer Science, 181,* , 11.
- Thesing, T., Feldmann, C., & Burchardt, M. (2022). A comparison between Agile and traditional software development methodologies. *Procedia Computer Science,181,746-756.*, 6.
- Yu Beng Leau, Wooi Khong Loo, Wai Yip Tham, & Soo Fun Tan. (2012). Software Development Life Cycle AGILE vs Traditional Approaches. *School of Engineering and Information Technology Universiti Malaysia Sabah, Malaysia*, 6.

Terminology

JPA (Java Persistence API): A Java specification that simplifies database operations by managing relational data as Java objects. It supports object-relational mapping (ORM) and is commonly used in Java-based frameworks like Spring Boot for database interactions.

CRUD (Create, Read, Update, Delete): Refers to the four basic operations that can be performed on persistent data. These operations are fundamental in database management systems, enabling users to create, retrieve, modify, and delete records.

UI (User Interface): The point of interaction between a user and a digital system or application. It includes all visual elements, such as buttons, icons, and menus, and plays a key role in ensuring the system is intuitive and easy to navigate.

XP (Extreme Programming): An Agile software development methodology that focuses on improving software quality and adaptability by promoting frequent releases, close collaboration with customers, and iterative development cycles.

FDD (Feature-Driven Development): A client-centric Agile methodology that breaks down software projects into small, manageable features. Each feature is developed incrementally, ensuring continuous progress and frequent delivery of working software components.

REST (Representational State Transfer): An architectural style for designing networked applications, where HTTP is used for communication between client and server. RESTful APIs allow interaction with web services using standard HTTP methods like GET, POST, PUT, and DELETE.

ORM (Object-Relational Mapping): A technique that allows developers to interact with a database using object-oriented programming. ORM frameworks, such as JPA or Hibernate, map database tables to classes and records to objects, simplifying database management and reducing the need for SQL queries.

IT (Information Technology)**:** IT refers to the use of computers, networks, and systems to store, retrieve, transmit, and manipulate data. It encompasses a wide range of technologies, including hardware, software, databases, and telecommunications. IT is essential for modern businesses,

enabling efficient operations and decision-making. It plays a critical role in both academic and corporate environments.

RUP (Rational Unified Process)**:** RUP is a software development framework that provides a structured approach to large-scale software projects. It is iterative, meaning that software is developed in cycles, and incorporates key best practices like risk management, proper documentation, and stakeholder involvement. RUP focuses on reducing project risks through clear phase definitions, including inception, elaboration, construction, and transition.

چکیده

ا نتخاب یک روش مدیریت پروژه مناسب برای موفقیت پروژه های توسعه نرم افزار بسیار مهم است. این مطالعه تحلیل تطبیقی روش‌های agile و Waterfall را در چارچوب توسعه یک سیستم مدیریت آرشیو برای پوهنحی کامپیوتر ساینس در پوهنتون هرات ارائه می‌کند. Agile که به دلیل رویکرد تکرار شونده و انعطاف پذیر شناخته می شود و Waterfall که با فرآیند ساختاری و خطی مشخص می شود، هر دو برای مدیریت فرآیند توسعه و جنبه های مدیریت ریسک پروژه مورد استفاده قرار گرفتند. این مطالعه بااستفاده از بورد های Trello برای مدیریت وظایف، ریسک‌ها و جدول‌های زمانی، اثربخشی هر روش را بر اساس نرخ تکمیل کار، میانگین زمان هر کار، و استراتژی‌های کاهش ریسک ارزیابی می‌کند.

نتایج نشان می‌دهد که در حالی که waterfall نرخ تکمیل کار بالاتری داشت (87٪ در مقابل چابک 83.3٪) و از نظر میانگین زمان برای هر کار (2.7 ساعت در مقابل 5 ساعت) کارآمدتر بود، Agile انعطاف‌پذیری بیشتری در انطباق با الزامات در حال تکامل نشان داد. و مدیریت ریسک های با تاثیر بالا هر دو روش تعداد مشابهی از خطرات را شناسایی و کاهش دادند، اما ماهیت تکرار شونده Agile پاسخ‌های فعال‌تر و به موقع‌تر را امکان‌پذیر کرد و تاخیرها را به حداقل رساند. این تحقیق بینش‌های ارزشمندی در مورد کاربرد روش‌های چابک و آبشار ارائه می‌کند و توصیه‌های عملی برای انتخاب مناسب‌ترین رویکرد بر اساس پیچیدگی پروژه، نیازهای ذینفعان و الزامات مدیریت ریسک ارائه می‌دهد.

این مطالعه با مقایسه این روش‌ها در یک محیط آکادمیک در دنیای واقعی، به گفتمان جاری در مورد مدیریت پروژه در مهندسی نرم‌افزار کمک می‌کند و مؤسساتی را که در اجرای پروژه‌های نرم‌افزاری با تصمیم‌های مشابه روبرو هستند، راهنمایی می‌کند.