Nottingham Trent University

School of Science and Technology

**Breaking Down Silos: Developing a Knowledge Sharing Platform for Internal Business**

by

Callum White

in

2023

Project report in part fulfilment

of the requirements for the degree of

Bachelor of Science with Honours

Digital & Technology Solutions Professional

Software Engineer/Cyber Security Analyst/Network Engineer/Data Analyst

I hereby declare that I am the sole author of this report. I authorize the Nottingham Trent University to lend this report to other institutions or individuals for the purpose of scholarly research.

This work is covered for confidentiality in accordance with the contract signed by Nottingham Trent University and my employer and should not be reproduced without permission of my employer.

Signature

Callum John White

Abstract

Enter abstract here. The abstract should include:

a) Brief statement of problem

b) Brief exposition of methods and procedures and

c) Brief summary of findings.

After the title, the abstract is the next level of detail regarding your work. The scientific community relies on 'The Abstract' as the main means of communicating research interests. There are many collections of abstracts published from which interested parties can trace the original work. They form the first step in a researcher's quest for up to date information regarding work being carried out worldwide.

The abstract is no more than a one A4 page, 1.5 Line Spacing, retrospective account of the achievements, techniques and conclusions of the report. On no account are you to exceed one page. Do not go into a great long explanation of the general area, be very precise and stick to what you have achieved.

Online databases will use the abstracts of published research work to help readers identify the basic content of papers, and decide if the research is of interest to them. They will know about the general subject area and its problems; what they want to know is whether you have found a solution. Be sure to include as much relevant information as possible on that page.

The abstract is very much an impersonal, factual, retrospective account of your finished project as might be written by somebody else. The tone of the English might be 'The work introduces the general area of..... and then investigates an apparently new method for..... The method is partially successful in that it...... '.

Acknowledgements

Enter acknowledgements here. It is usual to acknowledge those that have assisted you in your work and will normally include your main project supervisor. The order of acknowledgments (most important first) and their respective length indicates their relative importance to you.

Table of Contents

[Abstract ii](#_Toc132055144)

[Acknowledgements iii](#_Toc132055145)

[Table of Contents iv](#_Toc132055146)

[List of Figures x](#_Toc132055147)

[List of Tables xi](#_Toc132055148)

[CHAPTER 1 1](#_Toc132055149)

[Introduction 1](#_Toc132055150)

[1.1 Introduction 1](#_Toc132055151)

[1.2 Opportunity 3](#_Toc132055152)

[1.3 Gap Analysis 3](#_Toc132055153)

[1.4 Feasibility Analysis 4](#_Toc132055154)

[1.4.1 Operational 4](#_Toc132055155)

[1.4.2 Technical 4](#_Toc132055156)

[1.4.3 Schedule 4](#_Toc132055157)

[1.4.4 Economic 5](#_Toc132055158)

[1.5 Risk Analysis 5](#_Toc132055159)

[CHAPTER 2 7](#_Toc132055160)

[Background 7](#_Toc132055161)

[2.1 Introduction 7](#_Toc132055162)

[2.2 Team on MS Teams 8](#_Toc132055163)

[2.2.1 Limitations 8](#_Toc132055164)

[2.3 Dedicated SharePoint site 8](#_Toc132055165)

[2.3.1 Limitations 9](#_Toc132055166)

[2.4 Miro board 9](#_Toc132055167)

[2.4.1 Limitations 10](#_Toc132055168)

[2.5 Nuclino 11](#_Toc132055169)

[2.5.1 Limitations 11](#_Toc132055170)

[2.6 Gap in Existing solutions 12](#_Toc132055171)

[2.7 Similar Tools/Software/Approaches 12](#_Toc132055172)

[2.8 Commercial Context 13](#_Toc132055173)

[2.9 Research Studies 13](#_Toc132055174)

[2.9.1 Deloitte's knowledge-sharing platform 13](#_Toc132055175)

[2.9.2 Knowledge Sharing in Digital Platform Ecosystems: A Textual Analysis of SAP's Developer Community 15](#_Toc132055176)

[2.9.3 Social Media for Knowledge-Sharing: A Systematic Literature Review 16](#_Toc132055177)

[2.9.4 Comparison 16](#_Toc132055178)

[2.10 Summary 17](#_Toc132055179)

[2.11 Analysis of Legal, Social, Ethical and Professional Issues 18](#_Toc132055180)

[2.11.1 Legal Issues 18](#_Toc132055181)

[2.11.2 Social Issues 20](#_Toc132055182)

[2.11.3 Ethical Issues 20](#_Toc132055183)

[2.11.4 Professional Issues 22](#_Toc132055184)

[CHAPTER 3 23](#_Toc132055185)

[Project Definition 23](#_Toc132055186)

[3.1 Introduction 23](#_Toc132055187)

[3.2 Aims & Objectives 23](#_Toc132055188)

[3.2.1 Objectives 23](#_Toc132055189)

[3.3 Requirement analysis 24](#_Toc132055190)

[3.3.1 Research and Discovery 24](#_Toc132055191)

[3.4 Software Development Methodologies 26](#_Toc132055192)

[3.4.1 Waterfall 26](#_Toc132055193)

[3.4.2 Agile/Scrum 26](#_Toc132055194)

[3.4.3 Chosen Development Method 27](#_Toc132055195)

[3.5 Functional and Non-Functional requirements 29](#_Toc132055196)

[3.5.1 Functional Requirements 29](#_Toc132055197)

[3.5.2 Non-functional Requirements 30](#_Toc132055198)

[3.5.3 Conclusion 31](#_Toc132055199)

[3.6 Use Case Analysis 31](#_Toc132055200)

[3.6.1 Use Case #1 32](#_Toc132055201)

[1.1.1 Use Case #2 33](#_Toc132055202)

[1.1.1 Use Case #3 34](#_Toc132055203)

[3.7 MoSCoW 35](#_Toc132055204)

[3.7.1 Must-have: 35](#_Toc132055205)

[3.7.2 Should-have: 35](#_Toc132055206)

[3.7.3 Could-have: 36](#_Toc132055207)

[3.7.4 Won't-have: 36](#_Toc132055208)

[3.8 Project management 36](#_Toc132055209)

[3.9 Scope 36](#_Toc132055210)

[3.10 Schedule plan 37](#_Toc132055211)

[3.11 Resource planning 37](#_Toc132055212)

[3.12 Gantt Chart & PERT Diagram 38](#_Toc132055213)

[3.13 Risk Assessment 39](#_Toc132055214)

[CHAPTER 4 41](#_Toc132055215)

[Design 41](#_Toc132055216)

[4.1 Introduction 41](#_Toc132055217)

[4.2 High-level architecture design 41](#_Toc132055218)

[4.2.1 Web application framework 43](#_Toc132055219)

[1.1.1 Open ID Single Sign On (SSO) 44](#_Toc132055220)

[4.2.2 Azure container instances 44](#_Toc132055221)

[4.3 Graph database design 46](#_Toc132055222)

[4.3.1 Node properties 47](#_Toc132055223)

[4.4 UI design 49](#_Toc132055224)

[CHAPTER 5 51](#_Toc132055225)

[IMPLEMENTATION 51](#_Toc132055226)

[5.1 Introduction 51](#_Toc132055227)

[5.2 Coding standards 51](#_Toc132055228)

[5.2.1 SOLID 51](#_Toc132055229)

[5.2.1 Design patterns 52](#_Toc132055230)

[5.3 NET Project layout (Layered architecture) 53](#_Toc132055231)

[5.4 Automated deployment to Azure 55](#_Toc132055232)

[5.4.1 Github workflow 56](#_Toc132055233)

[CHAPTER 6 57](#_Toc132055234)

[RESULTS 57](#_Toc132055235)

[6.1 Introduction 57](#_Toc132055236)

[CHAPTER 7 58](#_Toc132055237)

[CONCLUSIONS / FUTURE WORK 58](#_Toc132055238)

[7.1 Conclusions 58](#_Toc132055239)

[7.2 Future work 58](#_Toc132055240)

[Reference 59](#_Toc132055241)

[Bibliography 60](#_Toc132055242)

[Appendix A 61](#_Toc132055243)

[Appendix B 62](#_Toc132055244)

List of Figures

[Figure 1 - Architecture diagram 42](#_Toc132055140)

[Figure 2 - Graph database design 46](#_Toc132055141)

[Figure 3 - Main display design wireframe 49](#_Toc132055142)

[Figure 4 - .NET project layout 55](#_Toc132055143)

List of Tables

[Table 1 - Gap Analysis 4](#_Toc132055134)

[Table 2 - Use case #1 32](#_Toc132055135)

[Table 3 - Use case #2 33](#_Toc132055136)

[Table 4 - Use case #4 34](#_Toc132055137)

[Table 5 - Resource skill 38](#_Toc132055138)

[Table 6 - Risk Assessment 40](#_Toc132055139)



Introduction

Introduction

A software system known as a knowledge sharing platform has both contemporary features that enable users to communicate information in real time and as a knowledge base. Users from around the organisation can post content or pose questions using this technology and, subject matter experts, can publish their comments in response.

Items are often shared between departments on an ad-hoc, informal basis, which can lead to duplication of effort or (unintended) exclusion of departments from useful findings.

Departments include:

* Product Design – who need to understand the needs and challenges of customers, what competitors offer, market trends etc through means such as desk research, webinars, conferences, and customer events/feedback.
* UX – who need to understand how to present the requirements of product features in the UI, using desk research, user testing and customer events/feedback.
* Research – who look towards the horizon of new technology and its applications, through desk research, webinars, conferences, early access demos, in-house trials, and developing prototypes.
* Marketing – who need to understand MHR’s place in the market, the needs and challenges of potential customers and the competitor space, though means such as desk research, webinars, conferences, and customer events/feedback, as well as access to research institutions such as Gartner and IDC.

Other departments around the company will also do their own research, such as DPO for keeping up to date on data privacy regulations and their effect on product development, but for the scope here, we will focus on the above four departments.

Items discovered/produced because of the research take on different forms, such as:

Links (to articles, solutions, competitor websites, upcoming webinars, etc)

Papers (academic papers, whitepapers produced by research organisations and/or competitors, etc)

Books (several in the Research bookcase, and more dotted around the company)

Videos (webinar recordings, YouTube videos, Teams meeting recordings, etc)

Summary write ups/word documents/blogs/presentations produced (e.g. as a result of an investigation or of the findings from a conference)

Excel spreadsheets of analysis (e.g. competitor analysis)

Items are shared between departments on an ad-hoc and informal basis. This sometimes leads to duplication of effort or (unintended) exclusion of departments from useful findings.

**Means of ad-hoc sharing currently include:**

* Email and Teams messaging
* SharePoint and OneDrive
* People First communities
* Miro
* Ad-hoc conversations

## Opportunity

MHR is a leading HR service provider who strives for continuous evolution of its product, which requires mass research be shared across the business. Currently departments are siloed between each other, causing harm to company culture, and decreasing efficiency. The opportunity to bring in a revolutionary way to share knowledge between departments would be a huge asset to the business. Initial improvements would be seen in the reduced duplication of effort, and as time advances it would encourage the reduction of silos by increased collaboration, sharing of knowledge, and between department idea generation.

## Gap Analysis

Table 1 shows a gap analysis which identifies and examines two primary areas that this proposed initiative addresses: two business procedures requiring development.

We are investigating a new method for between department idea generation and sharing of such ideas and knowledge, whilst allowing users to search and comment on posted content.

|  |  |  |
| --- | --- | --- |
| Current State | Desired State | Action to state |
| Content is currently shared between departments on an ad-hoc, informal basis, which can lead to duplication of effort or (unintended) exclusion of departments from useful findings. | An application for departments to manage and tag content. | Develop a cloud application where departments can share and tag content for other departments in a social feed. |
| Presently there is no access to globally search across the business for knowledge content, which causes increase of time due to not knowing content location. | Simplistic searching of content to allow users to use subjects. | Tagging of content and provide search functionally to filter down content. |
| Content has no managed structure and data is shared in different formats, making it difficult to analyse and report on. | Provide basic content structure and types to allow for manageable knowledge resources. | The system should allow users to enter content via forms which has strict content types, links, word documents, images, etc. |
| Content shared around the business has no feedback feed for employees to share thoughts on the knowledge resources. | Have collaborative feedback feature to allow for employees to feedback. | Have comments to be posted on the content for the author to review and reply. |

Table 1 - Gap Analysis

## Feasibility Analysis

### Operational

Departments will need to ensure that their staff will use this system in part of the process of knowledge sharing. It will become integrated into day-to-day business operations of sharing information. This system will be trialled on selected departments before being rolled out to the rest of the business.

### Technical

This project will use the identical technology stack as other products within the business, meaning there will not be a need for project participants to receive additional training. The stack is Microsoft.NET for backend services, Angular for front end application and SQL database for persistence layer. The code will be deployed and maintained on the same cloud platform, Azure.

### Schedule

A single team could take on this project, or run it in parallel with a current project, as there are several development teams operating within the company, provided they have redundancy and multiple developers per discipline.

### Economic

As this is intended to be an internal tool, rather than a sellable product or to incorporate into another product, the project’s return on investments will be modest. Although it will not generate sales or outside income, it reduces duplication of effort around the departments and increase idea generation between departments.

## Risk Analysis

Table 2 shows the risk analysis for the system, identifying risk itself, the probability, impact, and how we can mitigate or reduce the risk.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#ID** | **Risk** | **Likelihood** | **Impact** | **Mitigation** |
| 1 | Requirements are not fit | 3 | 5 – Requirements gathered does not match what the stakeholders want out of the system. | Complete business cases and provide questions for project stakeholders and document everything. |
| 2 | Scope Creep | 5 | 5 - When scope creep results in software development projects missing deadlines and taking longer than expected, risks can result. | Get the management approval before putting the project scope down in a Project Initiation Document. Throughout the project, refer to it and compare all modifications to it to ensure that they are in line with the Business Case. |
| 3 | Poor risk management | 3 | 4 – Poor risk management can lead to disruptions. These setbacks frequently result in one or more serious failures of a poorly thought-out software development project. | Create an in-depth risk assessment for the project which covers all risk and its mitigations. |



# Background

Introduction

Knowledge sharing platforms are important tools that enable organizations to share and leverage knowledge for better decision-making, innovation, and problem-solving. In this technical literature review, an overview will be provided of existing technical solutions and identify gaps that need to be addressed. Also a review of similar tools/software/approaches will be created that can be used in the proposed solution and their limitations.

There are several existing knowledge sharing platforms that organizations can use. Some of the most popular solutions include SharePoint, Miro board, Teams, Slack, and Nuclino. These platforms provide a range of features such as document management, workflow management, project management, social collaboration, messaging, and analytics. While these platforms are effective, they also have some limitations. For example, SharePoint and Nuclino can be expensive and complex to implement and maintain, while Teams and Slack are primarily designed for social communication and may not provide robust document management or workflow capabilities.

I picked out 3 existing solutions and did a more in-depth review to understand their capabilities and limitations.

## Team on MS Teams

MS Teams is the best messaging app on the market, for businesses. Microsoft (2019) says it offers a workspace for real-time collaboration, meetings, file and app sharing, and chat.

Currently, MS Teams is being used as a day-to-day file-sharing application for siloed departments to share knowledge content. Its benefits of being a real-time messaging system and file streaming service allow for quick sharing of information between users. Also, MS teams is integrated with Azure Active Directory, which secures its services by SSO, single sign-on.

The business has setup teams so that each user is assigned to their departments, which allow them to access certain channels.

### Limitations

MS Teams does not provide, in isolation, services required to meet the business’ desired state as described in the above gap analysis (Table 1). MS Teams has a static library structure which makes arranging and organising of knowledge difficult and near to impossible without SharePoint. This does not fit our need to have an application that enables users to manage and tag content. Additionally, MS Teams does not provide a solution for simple content sharing as you must search between each channel and chat manually meaning there is no solution for global search of files or information.

## Dedicated SharePoint site

Microsoft SharePoint is a document management and collaboration platform that allows corporations to manage archives, documents, reports, and other content that is essential to its operational procedures.

It will be simpler for staff who are already accustomed to using SharePoint for daily document management and business activities to switch to a full knowledge management solution set up within their current environment.

### Limitations

SharePoint has a search capability that, when used with wide query parameters, can be sufficient for obtaining a limited number of files, although the performance quickly falters when used with larger data sets and more specific search criteria.

Below are some limitations on searching –

* Without further customization, the only factor that can be used to filter search results is the document's age.
* The user's current site collection is the only one that may be searched.
* How administrators configured the functionality has a significant impact on the user search experience.

*(Brooks & Ryan Brooks, 2020)*

The above limitations impact the suitability of SharePoint, as a knowledge platform, due to it not achieving the desired state of global searching content. Searches are frequently for popular articles or rules, amongst others content. Although, several items have similar search phrases, there is frequently only one item that will provide the query’s answer.

## Miro board

Miro positions itself as a collaborative digital whiteboard that is simple to use. You can arrange objects, make notes, and connect via online chat or embedded videocalls all within the software. Miro is currently used by the business as a whiteboard session workspace, allowing team members to share and collaborate on feature projects.

If we were to use Miro as a knowledge platform it will primarily aid most as a visual knowledge platform, alongside the additional uses listed below.

To implement "Navigation" across the board, use internal linking. If the navigation is done right, this will prove to be a vital element to allow for content linking.

You can generate the content for the actual knowledge base by utilising different shapes, text, connection lines, and pen tools.

Miro board also provides access through SSO (Single-Sign on) which allows employees to access the service without using external emails outside of the business.

### Limitations

An issue with using Miro, is that if it is not managed it can become unwieldy and become hard to go back to existing boards. Users have limited structure or templates to fill in content within Miro board, which does not fit in with our desired state manageable, knowledge resources. Furthermore, this solution would also make it difficult to analyse and report on current knowledge content, shared within the system.

Additionally, Miro board doesn’t not have searching capabilities, other than the name of the board, which further does not fit with the desired state of content searching, meaning users must open multiple boards before finding the resources they require.

## Nuclino

Teams may share and edit documents in real-time with the aid of the knowledge management and collaboration platform Nuclino. It has a variety of capabilities, such as visual editors and third-party connectors, and is quite simple to use.

By creating material from scratch or using a template from their extensive library, it is easy to adapt to team requirements. This fits with our desired state of having a manageable knowledge resource structure and would allow the business to analyse knowledge resources across the business.

Furthermore, the effective search function rapidly displays pertinent results, and the phrases are highlighted in the results, making it simpler to locate desired information (Hero, 2021).

The system also supports real-time collaboration and commenting, by showing group changes in real time. This also achieves our desired state of collaborative feedback, by allowing employees to comment and share thoughts on knowledge resources shared by the author (Hero, 2021).

### Limitations

A limitation of Nuclino is that it is a novel tool outside of the business infrastructure, meaning knowledge is hosted outside of the business environment, which would limit any internal confidential content being shared.

Nuclino uses a pricing structure (£4.00 per user, per month) which could be a limitation due to budgetary constraints, going against the desired need for the knowledge platform to be accessible companywide. If the business were to implement Nuclino, it would cost £40,800.00 per year on licenses (£4.00 x 850 employees).

## Gap in Existing solutions

One of the gaps in existing solutions is the lack of efficient searching capabilities for knowledge repositories. While some platforms offer keyword-based search, this method can be limited in providing context-specific results. Therefore, there is a potential market for knowledge sharing platforms that utilize graph databases and tagging to improve searching efficiency. Graph databases can allow for more complex relationships between data points and enable the platform to suggest relevant information based on those relationships. Additionally, tagging can improve search results by allowing for more granular categorization of information. By incorporating these features, knowledge sharing platforms can better support the organization's knowledge management efforts and enhance decision-making processes.

## Similar Tools/Software/Approaches

Graph databases and tagging have been utilized in other knowledge management tools, such as Neo4j and Ontotext, to improve searching efficiency and provide more accurate recommendations. However, these tools are often targeted towards developers and require technical expertise to implement. There is a potential market for knowledge sharing platforms that incorporate graph databases and tagging in a user-friendly manner, providing non-technical users with the ability to leverage these features for more efficient knowledge management. By providing easy-to-use tagging and search functionalities, knowledge sharing platforms can improve the accuracy and relevance of search results, enabling users to quickly access the information they need. Incorporating graph databases into these platforms can further enhance the user experience by enabling the platform to suggest relevant information based on complex relationships between data points. Overall, the use of graph databases and tagging in knowledge sharing platforms has the potential to revolutionize the way organizations manage their knowledge and drive innovation.

## Commercial Context

Knowledge sharing platforms are widely used in various industries such as healthcare, finance, and technology. According to a report by MarketsandMarkets, the knowledge management software market is expected to grow from $6.1 billion in 2020 to $16.3 billion by 2025, with a compound annual growth rate (CAGR) of 21.5%. This growth is driven by the increasing need for organizations to leverage knowledge for better decision-making and innovation. The proposed solution can address the gaps in existing solutions and provide organizations with a competitive advantage by enabling personalized and context-specific recommendations and incentivizing knowledge sharing and collaboration.

## Research Studies

Below are three case studies done on Knowledge sharing platforms and what results and impact they had.

### Deloitte's knowledge-sharing platform

D Street is a digital platform that enables Deloitte's employees to collaborate and share knowledge across the company's various business units and geographic locations. The platform was launched in 2012 and has since become a critical tool for Deloitte's knowledge management strategy.

Some of the key features of D Street include:

1. Search functionality: D Street allows users to search for relevant content using keywords, filters, and advanced search options. The platform uses algorithms to suggest relevant content based on the user's search history and activity on the platform.
2. Content curation: D Street's content team curates and organizes content to ensure that it is accurate, up-to-date, and easily accessible. The team also creates new content based on user feedback and analytics.
3. Collaboration tools: D Street includes tools that enable users to collaborate on projects and share knowledge in real-time. These tools include discussion forums, instant messaging, and video conferencing.
4. Analytics: D Street uses data analytics to track user activity and measure the effectiveness of the platform. The analytics help Deloitte's leadership team identify areas for improvement and make data-driven decisions about the platform's development.

Since its launch, D Street has had a significant impact on Deloitte's knowledge management strategy. Some of the benefits of the platform include:

1. Increased knowledge-sharing: D Street has led to a 400% increase in the number of users accessing the platform, which has helped to break down silos and encourage collaboration across the company.
2. Improved content quality: D Street's content team has been able to improve the quality and relevance of the content on the platform, which has led to better decision-making and problem-solving across the company.
3. Reduced duplication: D Street has helped to reduce the duplication of efforts and resources, as employees can now easily find and build on existing knowledge within the company.

Overall, D Street has become a critical tool for Deloitte's knowledge management strategy, and has helped the company to improve collaboration, productivity, and innovation across its various business units and geographic locations.

### Knowledge Sharing in Digital Platform Ecosystems: A Textual Analysis of SAP's Developer Community

The paper titled "Knowledge Sharing in Digital Platform Ecosystems: A Textual Analysis of SAP's Developer Community" explores the concept of knowledge sharing within a digital platform ecosystem using SAP's developer community as a case study. The study uses textual analysis to identify the factors that promote and inhibit knowledge sharing within the community and examines how the community members' characteristics impact their engagement in knowledge sharing.

The study found that the community members' motivation and willingness to share knowledge, as well as the trust and support within the community, are key factors that promote knowledge sharing. In contrast, factors such as complexity and ambiguity of the platform, lack of time, and absence of recognition for contributions inhibit knowledge sharing. The study also identified that different types of community members, such as core developers and peripheral members, exhibit different patterns of knowledge sharing.

Overall, the paper provides insights into the complex dynamics of knowledge sharing within digital platform ecosystems and highlights the importance of understanding the factors that influence knowledge sharing to enhance the effectiveness of such communities.

### Social Media for Knowledge-Sharing: A Systematic Literature Review

The paper titled "Social Media for Knowledge-Sharing: A Systematic Literature Review" provides a comprehensive analysis of the existing literature on the use of social media platforms for knowledge sharing. The authors conducted a systematic literature review of 39 studies published between 2008 and 2017, and identified the various ways in which social media platforms can enhance knowledge sharing.

The paper finds that social media platforms can provide a collaborative environment for sharing ideas, facilitate informal learning, and enable access to a wide range of expertise. The success of social media for knowledge sharing depends on several factors such as organizational culture, technology adoption, and user motivation. The authors suggest that organizations need to develop strategies for implementing social media platforms and integrating them into their knowledge-sharing practices to achieve their full potential.

The paper highlights the need for further research in this area to better understand the role of social media in knowledge sharing. In summary, this paper provides valuable insights for organizations looking to leverage social media platforms for knowledge sharing, as well as for researchers seeking to explore this topic further.

### Comparison

Some differences between the three papers –

1. Focus and scope: The three papers focus on different platforms and contexts for knowledge-sharing. Deloitte's knowledge-sharing platform, D Street, is designed for internal use within Deloitte, while the other two papers examine knowledge-sharing platforms in the context of digital platform ecosystems and social media.
2. Methodology: The papers use different research methods. Deloitte's case study is based on internal data and analysis, while the other two papers use textual analysis and literature review methodology to analyse the content and trends in digital platforms and social media.
3. Audience and stakeholders: The three papers have different intended audiences and stakeholders. Deloitte's case study is primarily aimed at Deloitte's leadership and employees, while the other two papers are targeted at researchers, practitioners, and industry professionals interested in knowledge-sharing and digital platforms.

Overall, these studies demonstrate the importance of digital platforms for knowledge-sharing in organizations. While the studies focus on different platforms and contexts, they all highlight the importance of collaboration, innovation, and continuous learning in promoting effective knowledge-sharing. The studies also highlight the importance of features such as search functionality, content curation, and analytics in facilitating knowledge-sharing and improving organizational performance.

## Summary

In conclusion, knowledge sharing platforms are essential for organizations to manage their knowledge and drive innovation. However, existing solutions have limitations, such as the lack of efficient searching capabilities and the need for technical expertise to implement more advanced features. Graph databases and tagging have been utilized in other knowledge management tools to improve searching efficiency and provide accurate recommendations, but they are often targeted towards technical users. There is a potential market for knowledge sharing platforms that incorporate graph databases and tagging in a user-friendly manner, providing non-technical users with the ability to leverage these features for more efficient knowledge management. By incorporating these features, knowledge sharing platforms can improve the accuracy and relevance of search results, enabling users to quickly access the information they need. Furthermore, with the growing knowledge management software market, the proposed solution has the potential to offer organizations a competitive edge by enabling efficient knowledge sharing and collaboration while providing personalized and context-specific recommendations. By leveraging machine learning techniques and gamification, such as NLP, social network analysis, and machine learning algorithms, organizations can enhance their decision-making processes and drive innovation.

## Analysis of Legal, Social, Ethical and Professional Issues

### Legal Issues

**Licensing**

Currently the project is only for internal use and is not required to be licenced. If the business wants to transition the project to customer consumption, the product will require licensing. The rights of the creator and user of a piece of software are outlined in the software licence. It outlines the software's usage restrictions and how to pay for it. *(Lutkevich & Lebeaux, 2021)*.

If the project would become open source and require no official software licensing, it can provide a MIT licence.

**Intellectual property of content**

Intellectual property infringement damages can range from an injunction to a judgement for several million pounds. The knowledge platform can be legally taken down if it violates a patent, trademark, copyright, trade secret, or trade dress.

A simple prevention for this is all employees must reference any knowledge shared on the repository if it came from an external source outside of the business.

**Data ownership**

The product provides the feature of allowing a user to enter in information and content, because of this it falls under the GRPDR act. Data collection and storage must adhere to the regulation, and software should only request and handle the data essential for the software to operate.

Any violations must be disclosed to individuals impacted, failure to comply with this can result in significant fines and a negative company reputation. *(Art. 5 GDPR – principles relating to processing of personal data 2021)*

**Source control**

All the code of the project will be stored away within source control, Azure DevOps. It’s very important to ensure we use a verified source control provider as we want to code to be secure. The implementation is not open source, meaning no external individual will be given read or write access to the repository. This will protect the IP (Intellectual Property) of the product.

Only verified personnel within the business, using Azure Active Directory, will have permission on the repository.

### Social Issues

**Free text content**

A feature of the product is allowing the user to enter free text as content to share. This could lead to users posting insensitive texts and content around the system that would offend other users which will lead to social issues within the business. Social issues are guarded by company policies on communication and interaction of software between employees.

Content admins are allowed to manage all content around the system, this will allow the admin to review all free text shared on the system and remove any inappropriate content.

**Content sharing**

Another feature of the product is allowing users to share external content within the system. This could lead to users sharing inappropriate work content across the system, which again could lead to users being offended and social issues.

A prevention for this is content admins will have a list of verified domains, and users can only share information from a domain which has been verified.

### Ethical Issues

**Commenting for relative feed (gamification)**

It is possible that users will try and force other users to comment on their knowledge content so that their post is more likely to come up in other users’ feeds. This is very unethical as users will try and exploit the commenting system for gamification. For this reason, commenting has no effect on the order of posts on other users’ feeds.

**BCS (Code of conduct)**

Software developers are required to abide by a code of ethics and a set of guidelines when creating, designing, and maintaining software.

Below are some examples from BCS Code of conduct for making software for everyone *(BCS Code of conduct)* –

* have respect for the environment, other people's well-being, their privacy, security, and public health.
* respect the legal rights of others as they deserve.
* conduct your professional duties without prejudice based on a person's gender, sexual orientation, marital status, race, colour, nationality, ethnicity, religion, age, or disability, or any other condition or requirement.
* encourage equal access to the advantages of IT and work to advance the participation of all societal segments whenever chances present themselves.

**Plagiarism**

With the sharing content feature, a user might try and plagiarise knowledge content as their own. Copying someone else’s work is unethical because it does not give credit to the author which can lead to false impression about people skills and callabilities.

Content admins can verify knowledge content around system to ensure it’s not been plagiarised. An extra feature, if in scope, is to introduce a Turnitin like software which will check the content automatically as a part of the post feature.

### Professional Issues

**Business related content**

The content for this system should be limited to business or business skills related information to aid employees for upskilling and training purposes. Other content is not desired as this is a business tool for internal use.



Project Definition

Introduction

This report outlines the aims and objectives, requirements, planning, and success criteria for the knowledge sharing project. The purpose of this chapter is to provide an overview of the project and its goals.

## Aims & Objectives

The aim of this project is to create a knowledge sharing platform that will promote collaboration, information sharing, and transparency within the company, while breaking down silos between departments. The platform will provide a centralized location for employees to share their knowledge and expertise, access information and resources, and collaborate with colleagues across different teams and departments.

### Objectives

* To create a user-friendly and intuitive platform that encourages active participation and engagement from all employees.
* To promote a culture of transparency and knowledge sharing by providing easy access to relevant information and resources.
* To break down silos between departments and encourage collaboration by providing a centralized location for sharing knowledge and expertise.
* To improve the efficiency and effectiveness of internal communication by providing a platform for real-time feedback and discussion.
* To increase productivity and reduce duplication of effort by providing employees with access to a wide range of resources and expertise.
* To ensure that the platform is secure and compliant with relevant data privacy regulations, such as GDPR or CCPA.
* To integrate the platform with existing internal systems and workflows to maximize adoption and usability.
* To provide analytics and reporting features that allow admins to track user engagement and content performance, and to use this information to continuously improve the platform.

## Requirement analysis

### Research and Discovery

Requirements will be acquired through a joint application design (JAD) session to match the business needs found in the GAP analysis. As this is a fresh project, with no existing system to replace, the JAD session will concentrate on developing new needs without comparison to an existing system. The JAD session participants can communicate and provide a better grasp of the aims and goals, relative to their abilities and knowledge. This should clear up uncertainties regarding the deliverable’s ultimate goals, objectives, and expectations.

In addition to a JAD session, surveys will be created and provided to selected individuals within the business, in each siloed department. Each survey will gather information of key stakeholders regarding how knowledge is shared within their departments and team, with the aim of analysing variations and similarities. This information will be used in brainstorming and workshop sessions to better understand system users, and to produce applicable solutions which allow all types of knowledge to be shared companywide.

Some questions to ask people to understand the current situation from their points of view and add more to this section:

* Where do you currently store notes, papers, links etc
* When you attend an event, do you share your notes? With who?
* Do you share any knowledge (links/papers/documents) with other departments?
* What would be useful to you if you could search in another department’s notes/documents?
* When people share articles/videos with you, do you have the time to read/watch them?

If these requirement elicitation techniques are achieved, it will reduce the ‘#1 - Requirements are not fit’ risk, within the risk analysis.

The JAD session will be divided into three days. Each survey will be given out on a conduct analysis, to create appropriate questions.

* Day 1 – Survey feedback review and use cases.
* Day 2 – Requirements analysis, brainstorming and workshop.
* Day 3 – Specification design session

A senior stakeholder will be present within the JAD sessions, who will be a project representative and have final say on related decisions. Said stakeholder could be the CEO, CIO, or project manager (Davidson, E.J., 1999). Attendees will be made to feel as if they are a part of the process and the project objectives stated. Should initial worries be voiced, the stakeholder should strive to alleviate them.

## Software Development Methodologies

In this section there are two different approaches presented that are potential methods for competition of this project.

### Waterfall

A traditional model used in the system development life cycle to design a system with a linear and sequential approach is the waterfall model. Because the model progresses methodically from one phase to the next in a downward direction, it is known as a waterfall model. The output of one phase is utilised as the input for the following phase in this model, which is broken up into several phases. There is no overlap between phases; each must be finished before the next one begins.

Since each step is pre planned in a linear order, one drawback of the waterfall project management method is that it is not very adaptable. Any alteration in stakeholder objectives or demands will throw off the arrangement and necessitate a revision or maybe a completely new blueprint. For knowledge-based tasks, like computer programming, waterfall project management performs less well. However, waterfall workflows can be simply duplicated for upcoming, comparable activities, making up for its lack of flexibility. *(What is Waterfall Project Management? 2016).*

### Agile/Scrum

Agile methodology employs the "Waterfall" style of software development and focuses on communication and results. The concept focuses on self-management, short timescales, and flexible planning, enabling system improvements over time.

It eliminates the possibility of timelines that span months or years and lessens the likelihood that projects would fail because of errors made earlier on. Each phase is kept brief—under 4 weeks—which enables the system to advance quickly. Even though there is less documentation, the project places a strong emphasis on putting the key stakeholders' needs first by allowing teams and employees to collaborate closely with them.

Scrum is the component of Agile approach that we will focus on. Most firms utilise this model since it offers a simple framework for processes. In Scrum, a product owner creates a prioritised “wishlist” known as the product "backlog." From this backlog, teams, which are often small, will take bits of the backlog and produce a sprint. A sprint is a period (between 2-4 weeks) during which this item on the backlog will be evaluated, created, and integrated into the larger system. The teams will gather each day for a meeting called a "daily scrum," which tracks progress on the sprint and is run by a Scrum Master to keep the team on task. When the sprint is finished, they review and deploy the changes, then a new sprint is started.

An agile strategy might be used to increase the project's adaptability to changes and start delivering work sooner so that stakeholders could inspect it and provide input, ultimately giving end consumers a richer result. *(Denning, 2022).*

### Chosen Development Method

The Agile methodology makes use of short, fixed-length iterations that concentrate on a small number of components of change. Each iteration is produced, tested, and distributed using prototypes, which allows for quick user feedback to inform current and subsequent revisions, this will allow to understand and create the correct solution for internal use.

This internal system is very heavily prototyped, allowing the requirements to be abstracted and divided into smaller timeboxes with a minimum number of features that would benefit flexibility of change and adapting of features. *(Alberto Sillitti, 2011)*

One of the main principles of Scrum is that the business gives the scrum team authority to complete the task. The scrum team develops into a self-sufficient division of the company. This will allow the team more control on the work, not rushing each task to a deadline.

Although Agile approaches first appear to result in greater development costs and extended project timeframes due to the crucial role conversations across teams/companies play, this gradual process empowers users and promotes collaboration.

Requesting software specifications and requirements from key stakeholders using the waterfall process meant they had to specify exactly what they needed up front. Their specifications were outlined in a document that the development team and testers used to make sure they delivered exactly what the client had requested, without adding or removing anything.

This strategy assumed that customers knew what they wanted from the start. However, this was infrequently the case, and customers frequently requested changes to the specification after changing their minds, this increases our risk of “requirements to not fit.” The impact on the delivery date and who would be responsible for the cost of the change were then discussed.

The project will be using the Agile Scrum structure to plan, research, and create this system. Agile is a great lightweight framework for small, straightforward projects with compressed timelines.

## Functional and Non-Functional requirements

Below is the analysis of the requirements given by the identified stake holders, these requirements will be split into Functional and Non-Functional requirements. Functional consist of features that the system or data must contain. Non-Functional is aspects of the system which is general behaviours that the system needs.

Below is the detail of each requirement and reference from the requirement.

### Functional Requirements

**User Authentication and Authorization**

* The platform shall require users to authenticate themselves before accessing any content.
* The platform shall grant different levels of authorization to different types of users, such as admins and regular users.
* The platform shall integrate with Active Directory for Single Sign-On (SSO) to simplify user authentication and authorization.
* The platform shall authenticate users against the company's Active Directory domain.
* The platform shall provide users with a seamless login experience by using their existing Active Directory credentials.
* The platform shall allow admins to manage user access and permissions through Active Directory groups and roles.

**Content Creation and Management**

* The platform shall allow users to create and publish new content, including articles, open texts, and books.
* The platform shall allow admins to manage content, including reviewing and approving new posts before they are published, and removing or archiving posts that are outdated or no longer relevant.

**Tagging and Search Functionality**

* The platform shall include a robust tagging system that allows users to tag their posts with relevant keywords and categories.
* The platform shall include a powerful search function that allows users to easily find content based on specific keywords or categories.

**User Engagement and Collaboration**

* The platform shall encourage user engagement and collaboration through features such as commenting and rating.
* The platform shall allow users to connect with other users who share similar interests or expertise.

**Analytics and Reporting**

* The platform shall provide analytics and reporting features that allow admins to track user engagement and content performance, such as views, likes, and comments.
* The platform shall allow admins to export data for further analysis.

**Security and Data Privacy**

* The platform shall be secure and comply with relevant data privacy regulations, such as GDPR or CCPA.
* The platform shall include features such as encryption, user access controls, and data retention policies.

**Integration with Other Systems**

* The platform shall be able to integrate with other systems used by the company, such as an internal communication or project management system.
* The platform shall allow for easy sharing and collaboration between different teams and departments.

### Non-functional Requirements

**Performance**

* The platform shall be able to handle a large volume of users and content without significant degradation in performance.
* The platform shall be responsive and load quickly.

**Usability**

* The platform shall be easy to use and navigate.
* The platform shall include user-friendly interfaces for content creation and management.

**Reliability**

* The platform shall be always available and accessible to users.
* The platform shall include backup and disaster recovery mechanisms to ensure data integrity.

**Compatibility**

* The platform shall be compatible with a variety of devices and browsers.
* The platform shall comply with relevant web standards and protocols.

### Conclusion

The knowledge sharing platform will be an important tool for internal communication and collaboration within the company. By following the requirements outlined in this software requirement specification, the platform will provide a secure and efficient way for employees to share knowledge and expertise, while also facilitating user engagement and collaboration.

## Use Case Analysis

Use case analysis is an important tool for understanding how a software system will be used by its users and stakeholders. It helps to identify user requirements, define system behaviour, and specify system functionality. This information is critical for developing an effective testing strategy that ensures the system meets the needs of its users and stakeholders.

The system's primary functionality is represented by the three use cases listed below.

### Use Case #1

|  |  |
| --- | --- |
| **Use Case Name** | A user adds knowledge content |
| **ID** | 1 |
| **Priority** | High |
| **Actor** | Subject matter expert (user) |
| **Description** | The user selects content template to fill (links, papers, books, free texts). Once form is completed, the user will click save and publish, this will then persist this knowledge resource in a database. |
| **Trigger** | The user wants to use this system to share a piece of knowledge information with the rest of the business. |
| **Preconditions** | The application has verified and approved the user credentials. |
| **Normal Course** | * User clicks on add content button. * User selects content template from dropdown. * User fills out content form. * User submits and publishes the content |
| **Postconditions** | * Content is persisted within the database and shown on other user’s content feeds. * User logs out of application. |
| **Exceptions** | * User credentials are not valid or timed out. * Profanity filter picks up words. |

Table 2 - Use case #1

### Use Case #2

|  |  |
| --- | --- |
| **Use Case Name** | A searches knowledge content |
| **ID** | 2 |
| **Priority** | High |
| **Actor** | Content consumer (user) |
| **Description** | The user should be able to search for the content with the system via headers and tags. |
| **Trigger** | The user wants to use this system to learn different knowledge around the business. |
| **Preconditions** | The application has verified and approved the user credentials.  Content has already been added in the system with tags. |
| **Normal Course** | * User click on navigation icon on menu bar * Text field appears for user to write into * User press enter with text field filled * List of content results are returned |
| **Postconditions** | * User can click content. * The user logs out of the system. |
| **Exceptions** | * The user's login credentials are not valid. * No content in the system was added. |

Table 3 - Use case #2

### Use Case #3

|  |  |
| --- | --- |
| **Use Case Name** | Add comments to content |
| **ID** | 3 |
| **Priority** | Medium |
| **Actor** | Users |
| **Description** | The user should be able to comment and reply on content. |
| **Trigger** | The user wants to use this system to provide, or be given feedback on knowledge shared around the business. |
| **Preconditions** | The application has verified and approved the user credentials.  Content has already been added in the system with tags. |
| **Normal Course** | * User navigates to content * User enters comment text field and press enters * User enters reply comment text fields and press enters (for reply to threads) |
| **Postconditions** | * The user logs out of the system. |
| **Exceptions** | * The user's login credentials are not valid. * No content in the system was added. |

Table 4 - Use case #4

## MoSCoW

Below are the identified functional and non-functionals requirements of MoSCoW.

The term MoSCoW stands for must-have, should-have, could-have, and won't-have (or will not have right now) for the project.

### Must-have:

* User authentication and authorization with Active Directory integration.
* User profile creation and management.
* Content creation, including articles, books, and open texts, with the ability to tag and search for content.
* Admin dashboard for managing users, content, and platform settings.
* Recommended post feed for users based on their interests and engagement.
* User feedback and discussion features for each post.
* Analytics and reporting features for admins to track user engagement and content performance.

### Should-have:

* Social features, such as liking and sharing posts.
* Comment moderation tools for admins.
* The ability to follow users and subscribe to topics.
* Notification system for updates and activity on posts and subscriptions.
* Mobile-responsive design for easy access from different devices.
* Integration with existing internal systems and workflows.
* Customizable email templates for notifications and alerts.

### Could-have:

* Integration with third-party tools, such as project management or collaboration platforms.
* Content creation templates for standardising formatting and structure.
* User gamification features, such as badges or leaderboards.
* Machine learning algorithms to improve the accuracy of recommended posts.
* Integration with external knowledge sources, such as industry publications or news sites.
* Integration with external social media platforms for sharing content.

### Won't-have:

* Support for outdated web browsers or devices.
* Integration with external payment systems for purchasing content.
* Native mobile applications for iOS or Android.

## Project management

This section will outline the project planning, including the timeline and the resources that will be required. This will also include an overview of the project management approach that will be taken.

## Scope

This project is to develop a cloud-based central knowledge that reduces duplication of effort, increases collaboration, sharing of new ideas/knowledge, and idea generation between departments. The central repository will allow users to manage, feedback and share content between employees of MHR.

## Schedule plan

The project will begin at the start of November 2022 with an estimated end date mid-April 2023, a total project time of 4 months. The project timescales include additional time to allow for contingency of work items, if in case additional tasks are needed or existing tasks need more time to be completed. Work items will be planned into each timebox before it starts, following the agile pattern with a two-week timebox. This way, the capacity of each team member for the following sprint is known, and enough work can be planned with contingency time for issues or bugs that may prevent the team from progressing the work items for the project.

## Resource planning

The project plan tools details and outline the tasks and scope of the project with each team member assigned to it. The engineers, within the business, already possess the necessary technical expertise and skills for the project, both analysis and use of technology, this means no outside resources will be required to complete this project. It is crucial that at least one executive level participant attend the design sessions of the project.

In table 5 describes the development project team, the team only consist of 5 members –

|  |  |
| --- | --- |
| **Job titles** | **Skills** |
| Lead Full-stack Engineer | * Lead project manager for team * Full stack (Angular & C#) * Architecture design |
| Software Engineer (.Net) | * NET (C#) * SQL Database * Microservices design * CI/CD pipelines (backend services) * Automated testing |
| Frontend Web developer | * Angular (Typescript) * HTML/CSS3 * CI/CD pipelines (frontend application) * Automated testing |
| UX Designer | * Design UX for requirements |
| Business analyst | * Details requirements * User acceptance testing * Senior Stakeholder |

Table 5 - Resource skill

## Gantt Chart & PERT Diagram

A Gantt chart resource plan has been created and can be accessed within the file *‘COMP37005\_Project\_Management\_Plan\_GANNT.mpp’*, which can be found in annex 1.

A PERT diagram can be also found within the fille *‘COMP37005\_Project\_Management\_Plan\_PERT.mpp’* via annex 1, shows the dependencies of the project.

The Gantt chart project plan details the critical tasks with the project, such as initial analysis and requirement gathering phase, foundations for identifying the specifications for scope, and then the development timeline for phase 1, which consists of 3 timeboxes. Tasks have been mapped out with their dependencies which has created the critical path for the project. Some tasks can be started and finished in parallel with others, such as timebox development work. Within timebox 1, tasks 4.1.2.1, 4.1.2.2, 4.1.2.3 can be run in parallel as its solo dependency Is the refinement of the work (4.1.1.1) and are not dependent on each other and run in isolation with different team members. Another example of parallel tasks is 2.6.1 and 2.6.1, the two modelling tasks, these can be run in parallel because are not dependent on each other as well and requirements of the project being defined before beginning.

## Risk Assessment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description | Likelihood | Impact | Severity | Owner | Mitigation |
| Scope of project might change | Medium | Medium | Medium | Business Analyst | Ensure all scope is defined in detail via the design workshops and all stakeholders are involved. |
| Lack of communication leads to unclear thinking and misunderstanding. | Low | Medium | Medium | Team | Create a communication plan that details the timing, purpose, and target audience for each communication. Timebox reviews will allow the team time to feedback on the timebox. |
| Deficient testing leads to large post go live bug list . | High | High | High | Lead Engineer | Make sure the tester creates test cases and quality checks, and that they guard the window for testing and quality assurance. |
| Small team absences/sickness might delay project. | Medium | High | High | Lead Engineer | Additional time buffer added to guard any unplanned absences or sickness. |
| Undefined project needs and purposes | Medium | High | High | Business Analyst | Finish the business case and make sure the project charter's purpose is clearly stated. |
| Unexpected tasks that need to be accommodated | Low | Medium | Medium | Lead Engineer | Verify each plan and each quantity survey. Record all planning assumptions before the project begins. |

Table 6 - Risk Assessment



Design

Introduction

The purpose of this chapter is to provide a detailed analysis of the design aspects of the project. The chapter will include a justification of the design choices made, as well as a summary of the initial design completed prior to development. In addition to the summary, the chapter will also include full design documentation as appendices. Depending on the type of project, this documentation may include UI design mock-ups, system diagrams, and diagrams from formal design methodologies such as UML. If an iterative methodology is being used, analysis of feedback received, and changes made based on this feedback will also be included in the chapter.

## High-level architecture design

It's good to have a high-level architecture design for several reasons:

* Clarity: A high-level architecture design provides a clear picture of the system and its components. It helps in understanding the overall structure, the roles, and responsibilities of each component, and how they work together.
* Scalability: A high-level architecture design provides a framework for scaling the system. It helps in identifying the areas that need improvement and the ones that can be scaled horizontally or vertically.
* Reusability: A high-level architecture design promotes reusability of components. It helps in identifying the components that can be reused in different systems and the ones that need to be built from scratch.
* Maintainability: A high-level architecture design makes the system more maintainable. It provides a roadmap for maintenance activities and helps in identifying the areas that need attention.

Figure 1 is the architecture diagram for a containerized Web application with SSO and a Neo4j database as its persistence layer.

Diagram

Description automatically generated

Figure 1 - Architecture diagram

The architecture for this application would consist of the following components:

* Presentation Layer: The presentation layer would consist of the Web application that would be containerized using Docker. This layer would be responsible for rendering the user interface and communicating with the other layers.
* Business Logic Layer: The business logic layer would consist of the services that would implement the business logic of the application. These services would communicate with the persistence layer to retrieve and persist data. They would also communicate with the presentation layer to receive and send data.
* Persistence Layer: The persistence layer would consist of the Neo4j database that would be containerized using Docker. This layer would be responsible for storing and retrieving data. It would communicate with the business logic layer to provide data access.
* Single Sign-On Layer: The SSO layer would be responsible for authentication and authorization of users. It would authenticate users and generate access tokens that would be used by the application to access the resources.
* Container Orchestration Layer: The container orchestration layer would be responsible for managing the containers running the application and the database. It would ensure that the containers are running, healthy, and accessible.

### Web application framework

For the web application, this project will be using .NET Blazor, this is a web framework developed by Microsoft that allows developers to create web applications using C# and .NET instead of JavaScript.

It allows you to build rich, interactive user interfaces using a combination of HTML, CSS, and C#. Blazor provides two hosting models - server-side and client-side. In server-side Blazor, the application runs on the server, and the UI is rendered in the browser using SignalR. In client-side Blazor, the entire application is downloaded to the client browser and runs in WebAssembly. *(Microsoft. 2022).*

Blazor's lightweight design is one of its main advantages. Applications built using Blazor can be compact and quick because they don't need a lot of JavaScript or other complicated dependencies. Instead, the application's logic can be written in C# and either run on the server or compiled to WebAssembly. Blazor's lightweight design makes it particularly advantageous for projects that move swiftly since it enables programmers to produce applications quickly and with few dependencies.

### Open ID Single Sign On (SSO)

A standards-based, open-source authentication system called Azure OpenID enables you to authenticate users across numerous apps and services. It is a component of Microsoft's Azure Active Directory (Azure AD), a solution for cloud-based identity and access management. You can offer single sign-on (SSO) to your applications using Azure OpenID, enabling users to access numerous applications using a single set of credentials. Blazor supports several different authentication services, including Azure AD. This will make it easy for us to integrate the Blazor app with the company's AD.

### Azure container instances

Azure Container Instances (ACI) is a serverless container hosting solution that allows you to run your Docker containers quickly and easily without the need to manage servers or infrastructure. ACI is a great choice for hosting containerized Blazor applications with SSO and a Neo4j database as its persistence layer because it provides the following benefits:

* Easy Deployment: ACI makes it easy to deploy containerized applications without having to manage infrastructure or servers. With ACI, you can deploy your containers with a simple command or through the Azure portal.
* Scalability: ACI allows you to scale your containerized application easily, without having to manage the underlying infrastructure. You can scale up or down your container instances based on your workload requirements, and ACI will automatically manage the resources needed to run your application.
* Cost-effectiveness: With ACI, you only pay for the resources you consume while running your container instances. This means you can run your application cost-effectively without having to pay for unused resources.
* Security: ACI provides built-in security features such as network isolation, encryption at rest, and role-based access control (RBAC). This helps ensure that your application and data are secure.

By using Azure Container Instances to host your containerized Blazor application with SSO and a Neo4j database as its persistence layer, you can take advantage of the benefits of serverless container hosting and quickly deploy and scale your application with minimal effort.

## Graph database design

Figure 2 represents the graph database design diagram.

Diagram

Description automatically generated

Figure 2 - Graph database design

The graph database design consists of four types of nodes: **Department**, **Person**, **Post**, and **Tag**.

**Department** nodes have a one-to-many relationship with **Person** nodes, representing the fact that a department can have many employees. They also have a one-to-many relationship with **Tag** nodes, representing the fact that a department can have multiple interests or preferences.

**Person** nodes have a one-to-many relationship with **Post** nodes, representing the fact that a person can write multiple posts. They also have a one-to-many relationship with **Tag** nodes, representing the interests or preferences of the person.

**Post** nodes have a one-to-many relationship with **Tag** nodes, representing the fact that a post can have multiple tags.

**Tag** nodes represent the interests or preferences of a department, person, or post. Each **Tag** node is connected to one or more **Department**, **Person**, or **Post** nodes via a "LIKES" or "HAS\_TAGGED" relationship, respectively.

This graph database design is good for quick searching of tags for posts and recommendations of posts based on users' liked tags because it models the relationships between posts, tags, and users in a highly connected graph structure. For example:

* To search for posts that are tagged with a certain tag, you can simply traverse the graph from the Tag node to the **Post** nodes, returning all posts that are associated with the given tag.
* To recommend posts to a user based on their liked tags, you can first traverse the graph from the **Person** node to the Tag nodes representing the user's liked tags. Then, you can traverse from each **Tag** node to the Post nodes, returning all posts that are associated with the user's liked tags.

Overall, this graph database design enables efficient querying and searching of the data, making it ideal for applications that require quick access to data based on relationships between entities.

### Node properties

The node bodies in the graph database design represent entities such as Department, Person, Post, and Tag, and are designed to capture specific attributes and properties of these entities.

**Department**

* name: a string representing the name of the department.
* description: a string representing a description of the department.
* location: a string representing the location of the department.

**Person**

* name: a string representing the name of the person.
* age: an integer representing the age of the person.
* email: a string representing the email address of the person.

**Post**

* title: a string representing the title of the post.
* body: a string representing the body of the post.
* timestamp: a datetime object representing the time when the post was created.
* type: a string which represents the type of post.

**Tag**

* name: a string representing the name of the tag.
* description: a string representing a description of the tag.

The **Post** node will consist of a property called “type”. This will allow for different types of posts to be added, such as Linked content, Article content, etc. This will provide a more templated approach for posts. The .NET project will initialise the correct object based on the type.

## UI design

Web design is an essential aspect of building a successful online presence. A well-designed website can not only help attract visitors but also improve the user experience and encourage engagement with the content. This section will describe the UI designs that been created to allow for the platform to be visually appealing, easy to navigate, and functional.

Graphical user interface, application

Description automatically generated

Figure 3 - Main display design wireframe

1. Left-hand side navigation: This is a common UI element that is used to provide users with quick access to different sections of the application. By placing the navigation on the left-hand side of the screen, it is easy for users to access the different sections of the application, regardless of what page they are on.
2. Create post button: This button provides users with a clear call to action to create new content within the application. By placing this button in a prominent location, users are more likely to engage with the application and create new content.
3. Profile and manage, only if admin: This section provides users with access to their profile and the ability to manage their account settings. Additionally, if the user is an admin, they will have access to additional management features. By providing this functionality, users can easily manage their account and perform any necessary tasks related to their role within the application.
4. Top-hand navigation with a search bar: This navigation provides users with quick access to the search functionality of the application. By placing the search bar in a prominent location, users are more likely to use this feature and find the content they are looking for.
5. Main display with a scrollable feed for posts: This is the primary display for the application and contains a scrollable feed of posts. By making this display prominent, users are more likely to engage with the content and spend more time on the application. Additionally, the scrollable feed allows users to browse through the content quickly and easily without having to navigate to different pages.

Overall, this design is good because it provides users with easy access to the different features of the application, encourages engagement with the content, and makes it easy for users to manage their account and find the content they are looking for.



IMPLEMENTATION

Introduction

This section will include discussing the coding standards we used to ensure high-quality code that is easy to maintain and modify. We will also provide an overview of the project layout, including the organisation of the codebase.

Next, we will outline some of the issues we encountered during development, including bugs, integration problems, and performance issues. We will describe the steps we took to address these issues and the lessons we learned from them.

## Coding standards

Coding standards assist in the construction of less complex software programmes, which reduces. errors. If software engineering programming standards are followed, the code is consistent and easy. to maintain. This is since anyone may comprehend it and change it at any time.

### SOLID

Single responsibility principle, open-closed principle, Liskov substitution principle, interface segregation principle, and dependency inversion principle are all acronyms for five essential design principles.

Single Responsibility Principle - The SRP is based on the idea that each class, module, or function in a programme should have a single responsibility or purpose.

Open Closed Principle - Classes should be open to extension but closed to modification, according to the Open-Closed Principle.

Liskov Substitution Principle - Subclasses should be interchangeable with their base classes, according to the Liskov Substitution Principle.

Interface Segregation Principle - The Interface Segregation Principle is about keeping the interfaces separate, and segregation involves keeping things apart.

Dependency Inversion Principle - The Dependency Inversion Principle emphasises that instead of concrete classes and functions, our classes should rely on interfaces or abstract classes.

### Design patterns

In software engineering design patterns are common solutions that solves occurring problems in software design. A design pattern is not a finished design that can be used within every implementation of a piece of software but it’s a guide/template that can help you solve your problem (GeeksforGeeks. 2021.).

**Factory Pattern –**

The Factory Method design pattern is a creational design pattern that provides an interface for producing objects in a superclass while allowing subclasses to choose the type of objects created.

**Adapter Pattern –**

The adapter pattern acts as a link between two interfaces that are incompatible. This design pattern is classified as a structural pattern since it integrates the capabilities of two separate interfaces.

A single class is responsible for joining the capabilities of separate or incompatible interfaces in this design.

**Provider Pattern -**

Provider pattern allows for implementation of components to be introduced easily within a system. The provider object will have abstract logic or implementation that can be genetically referenced by other objects.

## NET Project layout (Layered architecture)

The .NET solution name will be called “KnowledgeShare”.

Creating a .NET project with a layered architecture consisting of KnowledgeShare.Web, KnowledgeShare.Core, and KnowledgeShare.Persistence is a good design because it promotes separation of concerns, improves maintainability, and enhances testability of the application.

Here's how each layer contributes to the overall design:

* KnowledgeShare.Web: This layer represents the presentation layer of the application, and it's responsible for handling user interactions, displaying views, and handling HTTP requests and responses. By keeping this layer separate from the other layers, we can easily swap out the front-end technology or framework without affecting the rest of the application.
* KnowledgeShare.Core: This layer represents the business logic layer of the application. It contains the core domain objects, services, and interfaces that encapsulate the application's behaviour. By separating this layer from the presentation and persistence layers, we ensure that the business logic is not tightly coupled to any specific technology or data storage mechanism.
* KnowledgeShare.Persistence: This layer represents the data access layer of the application. It contains the code that interacts with the database or other data storage mechanisms. By separating this layer from the other layers, we can easily switch to a different data storage technology or implement caching without affecting the rest of the application.

By using a layered architecture, we can achieve the following benefits:

* Separation of concerns: Each layer is responsible for a specific set of tasks, and there is clear separation between the layers. This makes it easier to reason about the application's behaviour and helps to prevent "spaghetti code."
* Improved maintainability: With a layered architecture, changes in one layer can be made without affecting the other layers, making it easier to maintain the application over time.
* Enhanced testability: Each layer can be tested independently, allowing for more comprehensive testing of the application. This also helps to identify issues earlier in the development process when they are easier and less costly to fix.

Overall, using a layered architecture with KnowledgeShare.Web, KnowledgeShare.Core, and KnowledgeShare.Persistence is a good design because it promotes separation of concerns, improves maintainability, and enhances testability, which are all important factors in building robust, scalable, and maintainable software applications.

Graphical user interface, text

Description automatically generated

Figure 4 - .NET project layout

## Automated deployment to Azure

The ability to deliver updates and new features more frequently and with fewer errors makes automated deployments crucial in the software development process. An automated deployment process can automatically build, test, and deploy changes to the production environment rather than requiring manual deployment of code changes. By doing so, the risk of human error is diminished, and it guarantees that the new changes are tested and validated before being made available to end users.

The software delivery process can be automated with the help of a set of procedures called continuous integration and continuous deployment (CI/CD). Using CI/CD pipelines, developers can automatically create, test, and deploy code updates to live environments. Developers can more quickly find and fix problems by implementing CI/CD, which leads to shorter time to market and shorter software development cycles.

### Github workflow

GitHub Workflow is a feature that allows developers to automate their software development processes using workflows. GitHub workflows can be triggered by events such as code pushes, pull requests, and issue comments. Workflows consist of a set of jobs, which are composed of one or more steps. Each step runs in its own environment and can be used to perform a specific task.

For the knowledge sharing platform CI/CD, a Github workflow was created that includes the following jobs –

Check out the code from the repository.

1. Build a Docker image for the .NET application and tag it with "mhr-ksp:latest".
2. Log in to Azure Container Registry using a service principal.
3. Push the Docker image to the Azure Container Registry.
4. Deploy the Docker image to Azure Container Instances.

The workflow will automatically trigger whenever a commit is pushed to the master branch. This will build and deploy the .NET application to a container in Azure Container Instances, enabling a faster and more reliable deployment process.

We can also create triggers to run a new set of jobs for deploying to production when a commit is pushed to a release branch. Having a release branch for deployment to production is a common best practice in software development, as it provides a stable and reliable environment for deploying code changes to production. Brown, A. (2019).



RESULTS

Introduction

This chapter include analysis of the performance and success of your product, covering the success criteria stated in the ‘Project Definition’ chapter.

You should give a clear view of the final product.

You should include testing of your final product using appropriate methods to measure against your success criteria. You can include a plan for further testing where testing hasn’t been possible eg after release/publication if your product is not yet live.



CONCLUSIONS / FUTURE WORK

## Conclusions

Whatever it was that your results showed should be summarised here. Your project or may or may not have achieved all that you set out to at the start.

This is your opportunity to conclude whether the project was a ‘success’ and how it might have been tackled differently in hindsight. This should include how you planned and managed the project, and how you addressed the PSEL issues identified in the ‘Background’ chapter.

## Future work

In either case there should be some reference to future work, either to forward and expand on the successful outcome or to test ways of overcoming the shortfall in your ideas that didn't work out quite as expected but there should be something that shows you can see further implications of what you have achieved.

Reference

References are a list that includes the essential bibliographical details for each item to which you have referred in the body of your paper. It should ONLY include items to which you have made direct reference. A direct reference is where you have quoted/reproduced text or diagrams from another author or mentioned/referred to the work of another author in your report. That is quoted directly what they have said about something or mentioned their views or conclusions in your report. You can use either the Harvard or the Scientific Numeric referencing system.

Bibliography

A Bibliography is a list of published materials that you have read or consulted for general information in the preparation of your work, concerning the subject of your Project, but have not made any direct reference to in your report i.e. 'background reading'.

You should always provide a Reference List. **A Bibliography is optional but when provided it should include all items in your Reference List as well as any additional items consulted in preparation of your work.**

Appendix A

You must include an appendix which demonstrates how your project meets the specialism KSBs from the apprenticeship standard. A minimum would be cross-referencing to the relevant sections, but you may need to add some additional context to make it clear why that section meets the requirements.

Templates will be provided for you to use.

Appendix B

The content of other appendices will differ with the different types of project. Any design and analysis charts/diagrams will be included here in full.

You may need to provide some form of user documentation to explain how to use the software/hardware produced. A researcher following up the work may wish to utilise the work of the original author and an appendix laying out the format of input files and how to interpret the output is required.