CSY4022 – Computing Project Dissertation

Interim Report

Hubble: A unified, secure, and extensible knowledge-base software for teams and individuals

BSc. Software Engineering

(2024)

[1]

[1] **Disclosure**: This document makes use of Apple Intelligence (https://www.apple.com/uk/apple-intelligence) for proofreading and grammar checking.

Table of Contents

Hubble: A unified, secure, and extensible knowledge-base software for teams and individuals

1. Introduction

Can the rapid improvements and increasing availability of computing power, Large Language Models (LLMs), and hosting tools make it significantly easier for teams and individuals to efficiently and intelligently store, organise, and search data spread across multiple sources without tedious duplication, privacy, pricing, or vendor lock-in risks?

In most teams, and even in our day-to-day lives, we often have data spread across the tens or even hundreds of applications/services we use with no tangible way to truly make efficient use of that data; organise, "query", or sometimes even find them at all. There have been several applications and services built for this purpose, but for most organisations, they are often quite expensive (and in turn considered not worth it, especially for small to medium businesses), do not support all the required data sources, have privacy/pricing/vendor lock-in risks, simply do not allow continuing with your existing third-party services (i.e. they require a full switch), which is not feasible for most teams and companies, or are simply inextensible; leaving teams at the mercy of these third-party services for support.

With the recent breakthrough innovations and advancements in Artificial Intelligence like the more smaller and powerful transformer models (Vaswani *et al.*, 2023) enabling things like on-device embedding generation, text inference, etc., and easier access to more powerful and cheaper hardware (e.g. via Scaleway, Fly.io, etc.), it is even more feasible to meet all these requirements, while making other aspects like querying the data and even customer interaction more intelligent and easier to use for most teams (and individuals).

1.1. Rationale/Significance

While there have indeed been others that have identified this problem and even attempted to solve it, this project aims to approach the problem more differently and openly. This is to prove there have been enough advancements in the open-source world and hardware/hosting availability to allow a system like this to exist without the privacy or pricing risks that could occur from delegating data to a third-party whose terms are subject to change at any time (often making it impossible to migrate too).

Many projects have gone down the path of making it near impossible to self-host their "open source" version which they claim can be "easily" self-hosted due to their (over-)reliance on either paid SaaS products that are not optional, or a multitude of unnecessary services/dependencies that need to be deployed along with the software itself. For example, although not entirely related to this product, is OpenReplay which has about **22** dependencies/services; most of which are redundant and can be removed based on the author's personal experience trying to self-host the aforementioned software.

The project aims to use its open-source nature to deliver extensibility via user-land plugins that can be distributed as freely (or used internally only) as the software itself. Most of the attempts that have been examined have restricted the available data sources; if they even consider the existence of other data sources at all, to first-party integrations with the most common services, and often providing knee-capped APIs as an afterthought.

The open-source, extensible, and unified nature of the system aims to deliver more power and control into the hands of users while staying cost-effective and free from the various risks discussed earlier – this makes it unique from every other product in this category that has been SaaS-ified and takes away control from users while being even more expensive with several other drawbacks; with privacy still being the foremost concern in this age of un-consented data gathering and scraping for AI training purposes.

1.2. Limitations and Scope

A fully intelligent knowledge base system of this nature can be very large in scope but due to the time limitation and nature of the project as a dissertation project, in order to deliver a high-quality product regardless, the author has narrowed down the scope further down to the following points (also based on the proposal):

- The system will be online and web-only to make it significantly easier to deploy to all devices via the browser.
- Rather than designing for "infinite" (or large) scale as originally planned, the system will be designed to cater for a smaller pool. This will have two effects on the system's designs:
    - Components will be more coupled by **default**. In the initial proposal, the author aimed to deploy all essential services separately so that they can be independently and horizontally scaled to make the system suitable for even very large organisations. Although, it should be noted that essential modules *can* still be deployed independently with this model, there will simply be less doing more at a smaller scale.
    - Deploying the system will require even less dependencies, as existing components will be re-used to serve other purposes. For example, instead of deploying a separate vector database like Milvus purely for its negligible (in this case) performance benefits, that function will be performed by Postgres along with relational data management; significantly harder to scale both functions independently but still high functional and performant at the intended scale.
- While the system will provide multi-factor authentication, this will be limited to email (if enabled during deployment) and Time-based One-Time Passwords (TOTP).

2. Background & Literature Review

Companies and individuals store data in some form or another across every application or service in use and this leads to a severe fragmentation of data. This fragmentation leads to known and unknown inefficiencies, as there are no unified or intelligent ways to properly use the data.

There have been attempts to address this issue by various applications and products over the years but they are usually costly, lack comprehensive support for third-party data sources, pose privacy/vendor lock-in risks, and require complete transitions from existing systems.

A few comparable products that already exist today have been considered and fall short in one or more ways, and this section will contain a brief discussion of these products.

- **Zoho Suite**: Zoho (Zoho Corporation Pvt. Ltd, 2025) provides a variety of products with decent and somewhat unified search across all of them. However, it still demands manual effort and a significant amount of time to sift through the numerous products and often tens of hundreds or even thousands of threads, documents, and posts to obtain meaningful context or answer even simple or complex questions with answers rooted in the past. The offline capabilities for services like Cliq, which do have some of this feature, are severely limited and practically useless. Moreover, like every other platform, it presumes the existence or rather the non-existence of a previous product or service to import or link data from.

- **Docmost**: A self-hosted wiki and knowledge-base management software; comparable to Notion (Phil, 2024) and has the same drawbacks with fewer features to offer. It assumes the non-existence of previous data and provides no way to import them; search is based on keyword search which is less effective than natural language or even fuzzy search. There are also no Machine Learning or AI enhanced experiences to further interact with, sort or filter data.

- **Orama**: Orama is an "answer engine" that offers search and chat capabilities based on data sourced from diverse origins, including website scraping, API ingestion, JSON/CSV documents, and more. It exclusively supports textual data and is tailored for Business-to-Consumer (B2C) applications. Orama's primary objective is to streamline and enhance customer interactions

rather than serving as an internal knowledge base. Unlike other products, Orama recognises the existence of external data sources but has limited file format support and is primarily designed for client/customer-facing use cases.

- **Outline**: Outline (General Outline, 2025) is an open-source self-hostable “knowledge base” software, albeit it mostly compares itself to documentation applications like Confluence and Google Docs. Unlike the other ones previously discussed, Outline somewhat accounts for the existence of other knowledge/data sources and provides first-party support for the most popular services. However, these integrations are severely limited to mostly pasting links from these services and usually restricted - just like most products on this list - to textual data. The AI search and chat functionalities are locked to the Enterprise and Cloud versions only, which means that the software is incapable of taking advantage of users’ hardware to provide these enhanced experiences in a self-hosted instance.
- **Selectric**: Selectric describes itself as an AI assistant for knowledge workers (Selectric, 2025) and unlike the other products discussed already, it attempts to do as much work as possible locally with (local) AI-enhanced experiences available in both the free and paid versions. However, the software is not geared towards teams or a shared usage, making it unsuitable for teams and businesses. Additionally, it is a macOS-only application with built-in support for only a handful of services (Dropbox, Outlook, Slack, etc.), with no way to support more without contacting the developer(s) to build in a first-party integration.

The underpinning technology for solutions that have attempted to tackle this problem in the past were previously based solely or largely around semantic/keyword-based search. This mainly involved using inverted index structures that are then ranked using a ranking model like the BM25 algorithm (Robertson and Zaragoza, 2009) or enhanced variants of it, like the BMX algorithm (Li *et al.*, 2024). Synonyms and stemming are also used to further refine search results by exploring alternative variations of words in various tenses (e.g. burn, burnt, burning), and including related words in search results (e.g. “desktop” -> “PC”).

Ranking is an important process as methods like full-text or keyword search are bound to yield unnecessary results, and usually, a lot of them. Ranking search results before presenting them to users ensures that the most relevant search results are presented to the user first, but this can also be an expensive and time-consuming operation.

User errors like typos are inevitable, as are misspellings of variations of different words. To account for these issues and provide typo-resistant search experiences, solutions also integrate fuzzy matching or “approximate search” using algorithms like the Levenshtein distance (Po, 2020) to find near-matches.

3. Aims and Objectives

This project aims to design and develop a Web Application that will provide users (teams and individuals) with an extensible, intelligent, and secure platform to effectively build a shared knowledge base for both internal and external use. The final product should be easily deployable on new and existing hardware with minimal dependencies and requirements, and without reliance on any proprietary/external APIs or services.

The following objectives will be followed to achieve project completion:

- **Market Research**: Investigate existing solutions to identify their approaches, strengths, and weaknesses. The insight from this research will in turn be used to highlight opportunities for differentiation and improvements.
- **User Research**: Conduct interviews and/or conversations with potential users to understand their expectations, requirements, and technical information to ensure that the final solution aligns with the real-world demands.

- **Technology/Tools Evaluation**: Assess available libraries, models, and tools that can be utilised to deliver a lean software product that meets these expectations at the end.
- **Development Process**: Develop the technical foundations, implement feedback, and refine features and modules while adding new functionality as required to meet the goals laid out.

Upon completion of this process, software capable of the following functionalities will have been developed:

- Supports various media formats, including plain text, Microsoft Word document format, PDF, CSV, JSON, audio recordings, videos, and pollable feeds (e.g., RSS, JSONFeed).
- Offers comprehensive hybrid (full-text and vector) search capabilities across all or most supported data types.
- Enhances user experiences by integrating AI experiences utilising Retrieval Augmented Generation (Lewis *et al.*, 2021) for in-depth exploration.
- Enables extensibility through user-land code via easy-to-install plugins.
- Implements a Role-Based Access Control (RBAC) system to restrict access to potentially sensitive data.

3.1. Target Audience

The primary target audience for this project is teams and individuals with slightly technical knowledge who are looking to build a shared knowledge base for internal and external use. This includes but is not limited to:

- Solo entrepreneurs e.g. indie developers
- Small to medium businesses
- Researchers and academics

The system is designed to be as user-friendly as possible while remaining very hackable (as in, extensible) and secure. This is to ensure that it can be used by users of varying degrees of technical knowledge and expertise.

4. Methodology

The primary methodology for this project is the Agile methodology, specifically using the Scrum framework. This methodology and framework were chosen based on their iterative nature, allowing for the rapid development of a working system that can be tested and refined based on user feedback throughout the development process. This process is carried out in sprints that last for a fixed period and typically conclude with a review/retrospective to align the goals for the next sprint.

The Agile methodology is particularly well-suited for this project as the requirements are not fixed and must cater to the needs of various users across different domains. It provides the flexibility to adapt to changing requirements—a certainty over the course of the project—and allows for experimentation and exploration, both of which are crucial.

Agile methodology, particularly the Scrum framework, reduces much of the pre-planning involved in traditional software development lifecycle approaches. While some planning remains essential, the iterative nature of Agile minimises the need for exhaustive documentation and design before development begins. This approach is ideal for this project, where requirements and interactive elements are expected to evolve throughout development.

Alongside Scrum, the Kanban methodology is also employed to maintain order and visibility of the project's progress. Kanban provides a clear visualisation of completed tasks, ongoing work, archived tasks, and pending tasks. This visibility is particularly useful for the author to track progress, ensure alignment with goals, prioritise tasks effectively, and maintain a decent amount of project organisation.

While Scrum is primarily designed for teams, the author has found it to be effective for single-person projects when adapted appropriately and combined with Kanban. This combination maintains structure while preserving the flexibility and adaptability required for the project's success.

4.0.1. Considered Methodologies

Several other methodologies were considered before settling on this approach. Below is a brief overview of the ones evaluated:

- **Waterfall**: The Waterfall methodology was considered but quickly ruled out due to its rigid and non-iterative nature. It requires extensive upfront documentation, and potential issues are often discovered late in the development cycle, leading to costly redo of the earlier phases. This approach is not feasible for this project, which heavily relies on iterative development and user feedback to meet its goals.
- **Joint Application Development**: JAD relies on the collaboration of stakeholders, developers, and end-users to design and develop the system collectively. This methodology was rejected as it demands significant time and resources to gather stakeholders and reach consensus on system requirements and design upfront. Additionally, it involves extensive meetings and documentation, which would take time away from the actual development process.

Other methodologies were also considered but were found to suffer from similar issues, such as excessive planning requirements or a lack of flexibility. Ultimately, the Agile methodology, utilising the Scrum framework in combination with Kanban, was chosen as it provides the right balance tailored to the author's needs and the project's dynamic requirements.

4.1. Analysis of the requirements

Over the course of development and various refinements, the system's requirements have been broken down into the following categories:

- **Functional Requirements**: These requirements define the software's core functionalities and features. These include the following:
  - Authentication and Multi-Factor Authentication (MFA)
  - Extensibility using plugins written in JavaScript
  - Hybrid search capabilities across all supported data types
  - LLM-based intelligent chat and summarisation
  - Role-Based Access Control (RBAC) for data security
  - Support for text, audio, and video data types
- **Non-Functional Requirements**: These requirements define the system's performance, security, and usability characteristics. These include the following:
  - **Accessibility and Usability**: The system should be easy to use and navigate for all users, including those with disabilities, screen readers, etc. The user interface should be intuitive and responsive with minimal latency.
  - **Data Privacy and Security**: The system should prioritise data privacy and security by ensuring that data is properly encrypted at rest and in transit. It should also provide functionality to restrict access to sensitive data as required by administrators. Every action taken within and by the system should never call out to external services unless explicitly allowed by the user; for example, in the case of importing data via plugins.
  - **Performance and Scalability**: The system should deliver a relatively fast response time for all operations and be easily scalable to accommodate growing concurrent access and data as needed.
  - **Portability and Interoperability**: The system should be easy to deploy on a variety of platforms and with minimal resource demands and dependencies. It should also be able to interoperate with other compatible dependencies and services; for example, swapping out Badger DB for Etcd.

- **Reliability and Availability**: The system should be able to fail gracefully in the case of an error and recover quickly while providing a way to debug the system when required. It should be able to provide a high level of availability with the minimum amount of maintenance required.

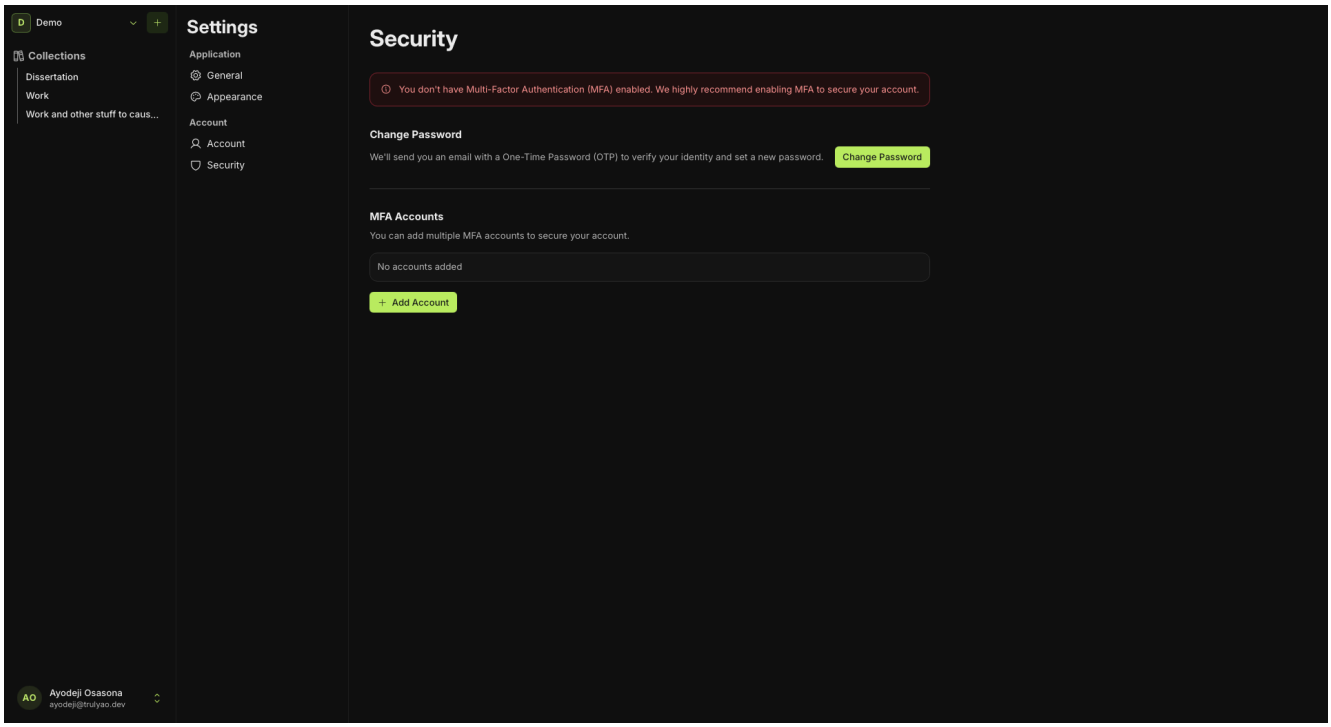
4.2. Design

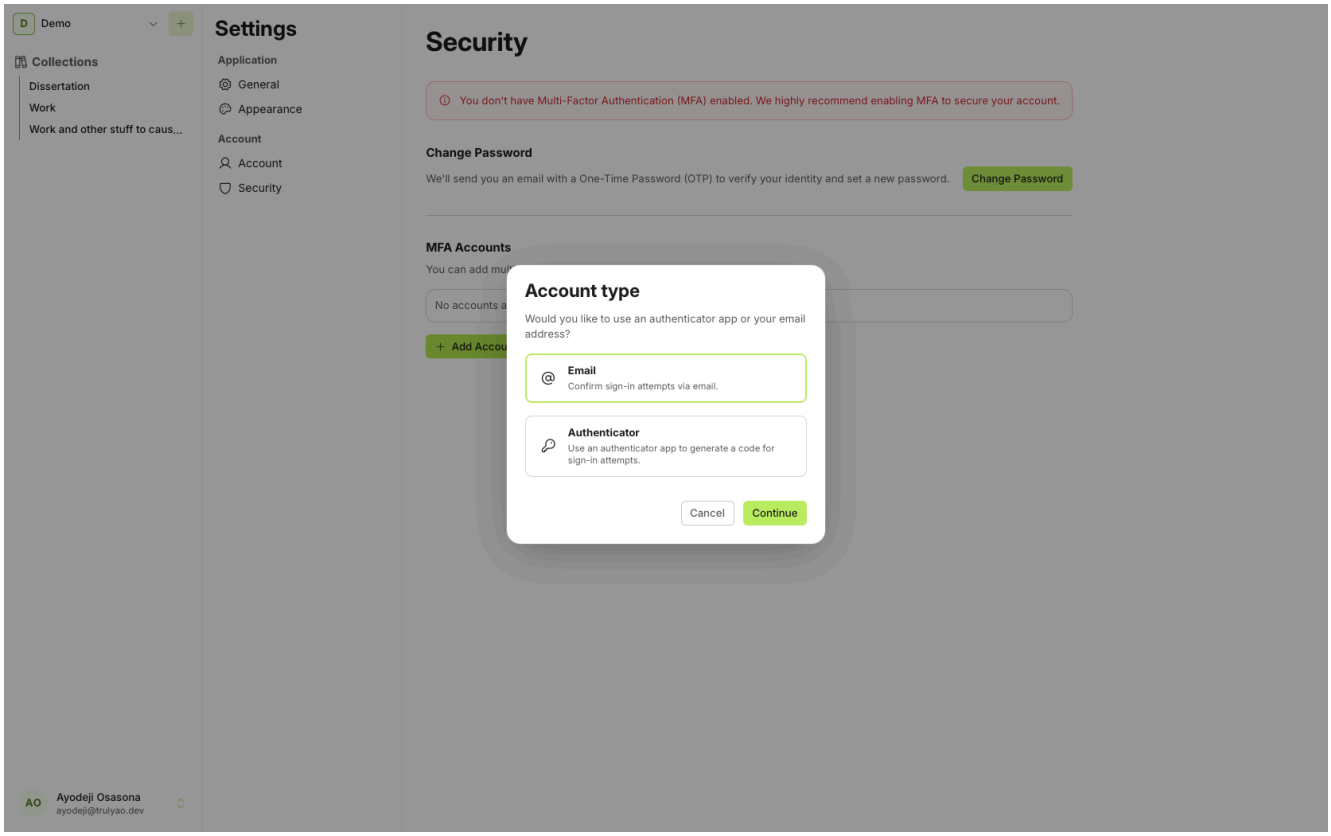
The system's user interface is designed to sport a modern and clean look, with elements that are fully responsive and accessible (keyboard, screen readers, etc). The interface, as with the rest of the system, presents users with customisation choices; albeit currently limited. Every part is designed to adapt to the user's preferred colour scheme to ensure that the system is as comfortable to use as possible, for example; users with vision impairments can choose a high-contrast color scheme and accent color.

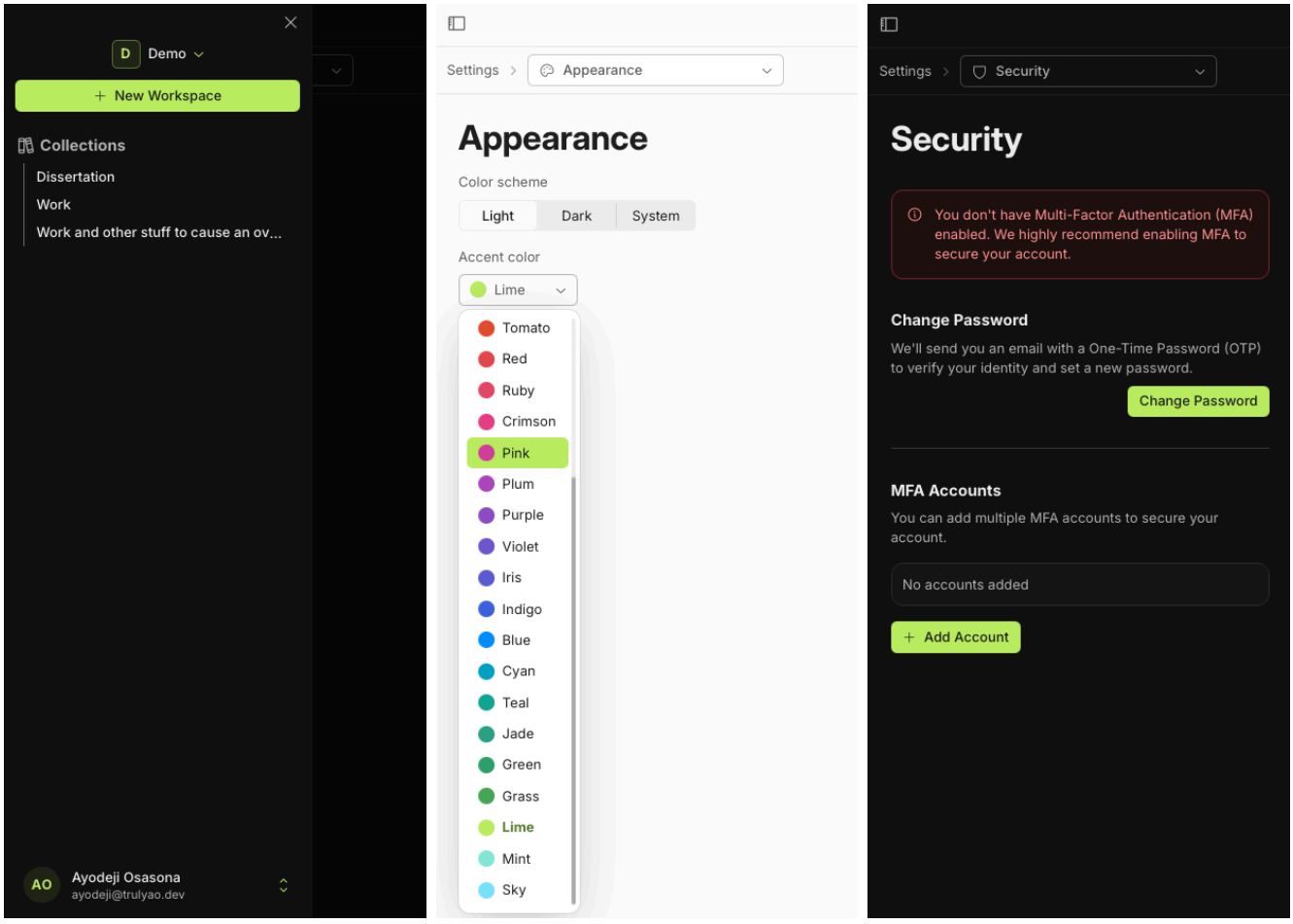
The system's design is based on the principles of simplicity and ease of use, this is to ensure that users can quickly, and easily navigate the system with little to no assistance required in the form of tutorials, documentations or developer support.

Actions are laid out in a logical manner than require the least amount of effort, while still ensuring that users are in control and conscious of these (destructive) actions.

4.2.1. Screenshots




Demo
Collections
Dissertation
Work
Work and other stuff to caus...
Settings
Application
General
Appearance
Account
Account
Security
Security
You don't have Multi-Factor Authentication (MFA) enabled. We highly recommend enabling MFA to secure your account.
Change Password
We'll send you an email with a One-Time Password (OTP) to verify your identity and set a new password.
Change Password
MFA Accounts
Account type
Would you like to use an authenticator app or your email address?
Email
Confirm sign-in attempts via email.
Authenticator
Use an authenticator app to generate a code for sign-in attempts.
Cancel
Continue
Ayodeji Osasona
ayodeji@trulyao.dev


Demo
New Workspace
Collections
Dissertation
Work
Work and other stuff to cause an ov...
Ayodeji Osasona
ayodeji@trulyao.dev
Settings
Appearance
Appearance
Color scheme
Light
Dark
System
Accent color
Lime
Tomato
Red
Ruby
Crimson
Pink
Plum
Purple
Violet
Iris
Indigo
Blue
Cyan
Teal
Jade
Green
Grass
Lime
Mint
Sky
Settings
Security
Security
You don't have Multi-Factor Authentication (MFA) enabled. We highly recommend enabling MFA to secure your account.
Change Password
We'll send you an email with a One-Time Password (OTP) to verify your identity and set a new password.
Change Password
MFA Accounts
You can add multiple MFA accounts to secure your account.
No accounts added
Add Account

5. Implementation

For the development of the system, Go is the language of choice as it strikes a nice balance between performance, concurrency, portability, and iteration speed, which are all essential requirements to deliver the high-performance & portable system as intended within the given time frame. For the API layer, it will make use of a new library; Robin, for client-server communication as it will boost developer productivity by getting rid of boilerplate code required for most CRUD actions that are present in every client-server architecture.

Essential components that have to be individually scalable like the models orchestrator (responsible for coordinating the various LLM models and actions) and hybrid search system will be built as embeddable and independent components such that they can be deployed and scaled independently of other components. For inter-service communication in the independent case where necessary, Cap'n Proto will be used for data serialisation, deserialisation, and remote procedure calls (Varda, 2024) as it is faster and more efficient than other alternatives like JSON or Protobuf.

5.1. Components/Dependencies

As much as one of the goals of this dissertation is to produce "lean" and portable software which might imply a single executable, that will not be the case; the system will contain a few external components due to its very nature and will be distributed via Docker which does provide a certain amount of portability of these different pieces, the various components are discussed further below.

- **PostgreSQL**: The main database in the system where all non-binary data is stored (including vectors using `pgvector` - a Postgres extension that adds vector support to PostgreSQL), it has been chosen based on its performance characteristics under write-heavy load, support for custom data types (extensibility in general) and Row Level Security (RLS) which would be useful for augmenting the Role-Based Access Control (RBAC) system.
- **Minio**: An S3-compatible object storage server designed to store and retrieve unstructured binary data (A.K.A objects) and can be used for a variety of use-cases like data archival, data lakes, etc. It will be used as the storage backend for assets like the images, documents, etc. imported into the system. It is also designed to be scalable and distributed with the ability to simply add more nodes when necessary as the data grows.
- **BadgerDB**: BadgerDB is an embeddable and persistent key-value store designed and built from the ground-up for speed in pure Go. It does not require running a separate server like Redis, Etcd, and other popular key-value stores. Badger will be used for operations that produce or require cached data or simple configuration data that do not need to be persisted for too long. A good use case in this system is to store One-Time passwords for email verification, password reset/change, etc.

5.2. Machine Learning and AI Workload

The Machine Learning and AI workload are powered by two distinct runtimes, depending on the operation's host platform and the use case. LocalAI (https://localai.io), a way to run local AI models with a front-facing OpenAI-compatible API, this is momentarily being used for multi-modal tasks. Although, the author aims to replace this with a leaner custom solution for more control and stability.

On the other hand, ONNX, a small yet highly efficient inference engine developed by Microsoft, specifically designed for mobile and cloud applications, will be used to run smaller models, primarily in the browser where applicable.

5.3. Authentication and Authorization

For authentication, the system supports email and password authentication, along with TOTP and email-based MFA. Although email support is optional to reduce a dependency on external services like the SMTP server, it is recommended to enable it as it makes it easier to perform operations like account recovery, identity verification, and workspace invitations among others.

While the system is a typical SPA plus API setup, the application uses “server-side sessions” for future requests after initial login. Sessions are persisted on the frontend using signed HTTP-only cookies (via the HMAC-SHA256 algorithm); this is to prevent users from modifying the cookies in any way, this is to ensure that they cannot be tampered with or accessed via JavaScript. And on the server side, valid sessions are stored with a time-to-live in Badger DB along with the user ID and other required metadata.

The advantage of this compared to JSON Web Tokens (JWTs) (Jones, Bradley and Sakimura, 2015) is that:

- Sessions can be listed for all users as they are well tracked.
- Sessions can be revoked en masse or individually since the expiration and validity are fully controlled by the system.
- It also simplifies the implementation, saves on storage, and removes the need to “purge” dead tokens as we do not need to keep track of “blacklisted” tokens; most JWT revocation systems do their job by keeping track of blacklisted tokens (keeping track of whitelisted ones would be nearly equivalent to just using session tokens).
- Sessions can even be associated with specific devices or IP addresses if needed
- Sessions can never hold stale data as they are looked up freshly before use

While most of these things can be reimplemented using JWT, they nullify the main benefits of JWTs, which is the statelessness and just bring their usage closer to this implementation. There are certain performance penalties to this approach, like having to look up the user data on every request, but compared to the risk of holding stale data in “stateless” tokens to avoid that lookup, it is a negligible cost to pay for better security.

5.4. User Interface

The user interface is built with React (https://react.dev) using themed Radix components (https://www.radix-ui.com), which provide excellent accessibility features out of the box. Additional styling not handled using the components’ props are handled using Tailwind v4 (https://tailwindcss.com) as it saves time and effort in the UI design by making it easy to test and iterate on different styles.

React and TypeScript have been chosen here due to the author’s familiarity with this particular SPA (Single Page Application) framework and the robust ecosystem support that comes with it, which would in turn boost productivity and enable the development of a responsive, accessible, and interactive user experience.

Global state management is handled using Valtio (https://valtio.dev) which is a tiny yet fast and reactive global state manager for React using Proxies. Valtio is heavily optimised to maintain render performance during state updates and removes a lot of boilerplate associated with other state management libraries like Redux.

5.5. Development and Deployment

Development is currently handled using Docker and Docker compose and environment variables are loaded using the docker compose `env_file` directive. This is to ensure that the system can be easily developed on any supported platform. For development outside Docker, the system also uses Mise to manage dependencies across all modules and for running tasks and scripts when required.

These methods ensure that it is very easy to get started with the development of the system without much overhead.

The current primary staging environment is deployed to Fly (https://fly.io) which is a Platform-as-a-Service (PaaS) that offers the deployment of applications into secure Firecracker micro-VMs while remaining compatible with the Docker ecosystem. It uses a managed Postgres instance (with pgvector installed) managed by Neon (https://neon.tech); an open source managed serverless PostgreSQL service.

6. Testing/Evaluation Strategy

For this project to be considered a success, it must meet the following criteria:

- Users should be able to onboard successfully with zero to minimal assistance from the author or any other party.
- Users should be able to import data from various sources with ease based on their preferred criteria.
- The system should be able to provide accurate search results for a significant percentage of queries.
- It should be significantly easy for a user with little to no technical knowledge to write and deploy a plugin to extend the system's functionality.
- It should be easy to self-host the system in its current state at the time with very little configuration required.
- Self-hosting the system should cost zero to a minimal amount of money to operate and maintain on own hardware.

The system will be broken down into smaller modules that can be unit-tested as required to ensure correctness and functionality. This approach will help to prevent the silent introduction of breaking changes as the system grows.

Potential users will be invited to test the system and provide feedback on their experience. This feedback will be used to gauge the system's usability, performance, and overall user experience. The feedback will be collected and analysed to determine the system's shortcomings and whether it has answered the research question posed by the dissertation.

The results of these various (user) testing will be objectively compared against similar experiences/products in the market to determine the system's effectiveness and usability in a real-world scenario. This will help to identify areas that need improvement, areas where the system excels, and areas that have been overlooked or require further research.

7. Professional, social, economic, and legal issues

Since the project is designed and developed to deal with sensitive and non-sensitive data, it brings up a couple of issues as to how the data can be used, who can access that data, when the data can be accessed, and other issues from a professional, social, economic, and legal standpoint.

7.1. Legal issues

The project will be designed to facilitate both internal and external access as the organisation or individual requires. This presents a couple of issues regarding data access privileges and data security. There are a set of guidelines and laws that are in place to protect the data of individuals and organisations. These include but are not limited to:

- **The General Data Protection Regulation (GDPR)**: A data privacy law put in place and enacted by the European Union (EU) to enhance the protection of personal data and privacy of individuals (Breen, Ouazzane and Patel, 2020) within the EU and the European Economic Area

(EEA). This is relevant as the system will be used, hosted, and accessed by individuals and organisations within the EU and EEA.

- **Data Protection Act 2018**: This is the UK's implementation of the EU's GDPR (Information Commissioner's Office, 2018). It was also put in place to dictate how personal data can be handled with an emphasis on the rights of the individual (fairness, transparency, and security).

7.2. Economic issues

Self-hosting a system with the proposed features and capabilities will undoubtedly demand a significant amount of resources to operate and maintain on both the software and hardware fronts, as well as human resources. This could pose a substantial challenge for small to medium businesses and individuals who may lack the necessary resources to maintain such a system or invest in additional hardware to accommodate the growing compute and storage requirements as usage increases.

To counter this, the system will be designed to be as lightweight and lean as possible to reduce the hardware requirements and the cost of operation. Tasks that require maintenance such as updates will be automated and simplified as much as possible to make the system more cost-effective and easier to maintain.

7.3. Professional & Social issues

A system like the one proposed in this project could require a significant amount of technical know-how to operate and fine-tune. This would pose a challenge for individuals and organisations that lack the necessary technical expertise to operate the system effectively. The system will be designed to be as user-friendly as possible to maintain a low learning curve, and to make it accessible to a wider range of users.

It also raises the question of individual privacy. For example, team members may be required to share data potentially containing personal information (e.g. chat logs), this could lead to a breach of privacy if not properly handled. This will require both a technological and social solution, with the technical part ensuring that the data imported can be controlled and restricted on a granular scale, and the social aspect involving a full understanding of what data is being collected and why the data is being collected.

Additionally, the system will implement a granular Role-Based Access Control (RBAC) authorization system (Bertino, 2003) to ensure that access to sensitive data can be restricted to only those who need it. This will help to prevent unethical use of the system and ensure that data is properly siloed or shared as required.

8. References

Bertino, E. (2003) "RBAC models — concepts and trends," *Computers & Security*, 22(6), pp. 511–514. Available at: https://doi.org/https://doi.org/10.1016/S0167-4048(03)00609-6.

Breen, S., Ouazzane, K. and Patel, P. (2020) "GDPR: Is your consent valid?," *Business Information Review*, 37(1), pp. 19–24. Available at: https://doi.org/10.1177/0266382120903254.

Evans, D. (2024) *Meet the new Notion AI. Get to know what it can do for you*. Available at: https://www.notion.so/blog/meet-the-new-notion-ai.

Faith Xu, B.L. (2023) *ONNX Runtime: performant on-device inferencing - Microsoft Open Source Blog — opensource.microsoft.com*. Available at: https://opensource.microsoft.com/blog/2023/02/08/performant-on-device-inferencing-with-onnx-runtime.

Fowler, M. (2005) *Event Sourcing — martinfowler.com*. Available at: https://martinfowler.com/eaaDev/EventSourcing.html.

General Outline, I. (2025) *Outline - Your team's knowledge base*. Available at: https://getoutline.com/.

Giacinto, E.D. (2023) *LocalAI: The free, Open source OpenAI alternative*. GitHub. Available at: https://github.com/go-skynet/LocalAI.

Information Commissioner's Office (2018) *An overview of the Data Protection Act 2018*. Available at: https://ico.org.uk/media/for-organisations/documents/2614158/ico-introduction-to-the-data-protection-bill.pdf.

Jones, M.B., Bradley, J. and Sakimura, N. (2015) *JSON Web Token (JWT). Request for Comments*, RFC Editor. Available at: https://doi.org/10.17487/RFC7519.

Kleppmann, M. (2021) "Thinking in Events: From Databases to Distributed Collaboration Software". Available at: https://martin.kleppmann.com/papers/debs21-keynote.pdf.

Kleppmann, M. and Beresford, A.R. (2017) "A Conflict-Free Replicated JSON Datatype," *IEEE Transactions on Parallel and Distributed Systems*, 28(10), pp. 2733–2746. Available at: https://doi.org/10.1109/TPDS.2017.2697382.

Kleppmann, M. *et al.* (2019) "Local-first software: you own your data, in spite of the cloud," in *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. Association for Computing Machinery (Onward! 2019), pp. 154–178. Available at: https://doi.org/10.1145/3359591.3359737.

Lewis, P. *et al.* (2021) *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. Available at: https://arxiv.org/abs/2005.11401.

Li, G. *et al.* (2011) "Efficient fuzzy full-text type-ahead search," *The VLDB Journal*, 20, pp. 617–640. Available at: https://api.semanticscholar.org/CorpusID:8942945.

Li, X. *et al.* (2024) *BMX: Entropy-weighted Similarity and Semantic-enhanced Lexical Search*. Available at: https://arxiv.org/abs/2408.06643.

Martinez-Gil, J. (2017) "Automated knowledge base management: A survey." OSF Preprints. Available at: https://doi.org/10.1016/j.cosrev.2015.09.001.

Phil (2024) "Docmost is here". Available at: https://docmost.com/blog/docmost-is-here.

Po, D. (2020) "Similarity Based Information Retrieval Using Levenshtein Distance Algorithm," *International Journal of Advances in Scientific Research and Engineering*, 6, pp. 6–17. Available at: https://doi.org/10.31695/IJASRE.2020.33780.

Rackauckas, Z. (2024) "Rag-Fusion: A New Take on Retrieval Augmented Generation," *International Journal on Natural Language Computing*, 13(1), pp. 37–47. Available at: https://doi.org/10.5121/ijnlc.2024.13103.

Ripley, B.D. (2005) "197 How computing has changed statistics," *Celebrating Statistics: Papers in honour of Sir David Cox on his 80th birthday*. Oxford University Press. Available at: https://doi.org/10.1093/acprof:oso/9780198566540.003.0011.

Rob Chahin (no date) *The SSO Tax*. Available at: https://sso.tax/ (Accessed: March 20, 2025).

Robertson, S. and Zaragoza, H. (2009) "The Probabilistic Relevance Framework: BM25 and Beyond," *Found. Trends Inf. Retr.*, 3(4), pp. 333–389. Available at: https://doi.org/10.1561/1500000019.

Saquib, N., Krintz, C. and Wolski, R. (2022) "Log-Based CRDT for Edge Applications," in *2022 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 126–137. Available at: https://doi.org/10.1109/IC2E55432.2022.00021.

Selectric (2025) *Selectic - AI assistant for knowledge workers*. Available at: https://selectric.io/.

Team, A.A. (2022) *Fast-forward — comparing a 1980s supercomputer to a modern smartphone*. Available at: https://blog.adobe.com/en/publish/2022/11/08/fast-forward-comparing-1980s-supercomputer-to-modern-smartphone.

Varda, K. (2024) *Cap'n Proto: RPC Protocol*. Available at: https://capnproto.org/rpc.html.

Vaswani, A. *et al.* (2023) *Attention Is All You Need*. Available at: https://arxiv.org/abs/1706.03762.

WebAssembly Core Specification (2018). Available at: https://www.w3.org/TR/wasm-core-2 (Accessed: December 18, 2024).

Xi, Y. *et al.* (2022) *Context-aware Reranking with Utility Maximization for Recommendation*. Available at: https://arxiv.org/abs/2110.09059 (Accessed: April 1, 2025).

Zhu, S. *et al.* (2023) “Intelligent Computing: The Latest Advances, Challenges, and Future,” *Intelligent Computing*, 2, p. 6. Available at: https://doi.org/10.34133/icomputing.0006.

Zoho Corporation Pvt. Ltd (2025) *Your life's work, powered by our life's work*. Available at: https://zoho.com/.