Prague University of Economics and Business

Faculty of Informatics and Statistics

Sharepoint Agent VŠE

Popeyes Research Division

Technical Report – 4IT500

2025-06-08

Table of contents

1 Solution Architecture

1.0 Background

Students often struggle to locate essential information, such as enrolment procedures, thesis-submission rules or exam schedules, because the university intranet runs on SharePoint and is split across many separate faculty, department and information-type sites. In practice it is rarely obvious where to search. The goal of this project is therefore to design and test a search chatbot that accepts natural-language questions (Czech and English) and returns relevant answers instantly, removing the need to navigate the intranet manually.

1.1 Purpose and Scope

This chapter defines the target architecture for introducing Microsoft Copilot Studio as that chatbot layer. The solution must run entirely inside the university's Microsoft 365 tenant, respect individual permissions, satisfy EU data-residency requirements, and appear natively inside Microsoft Teams and SharePoint.

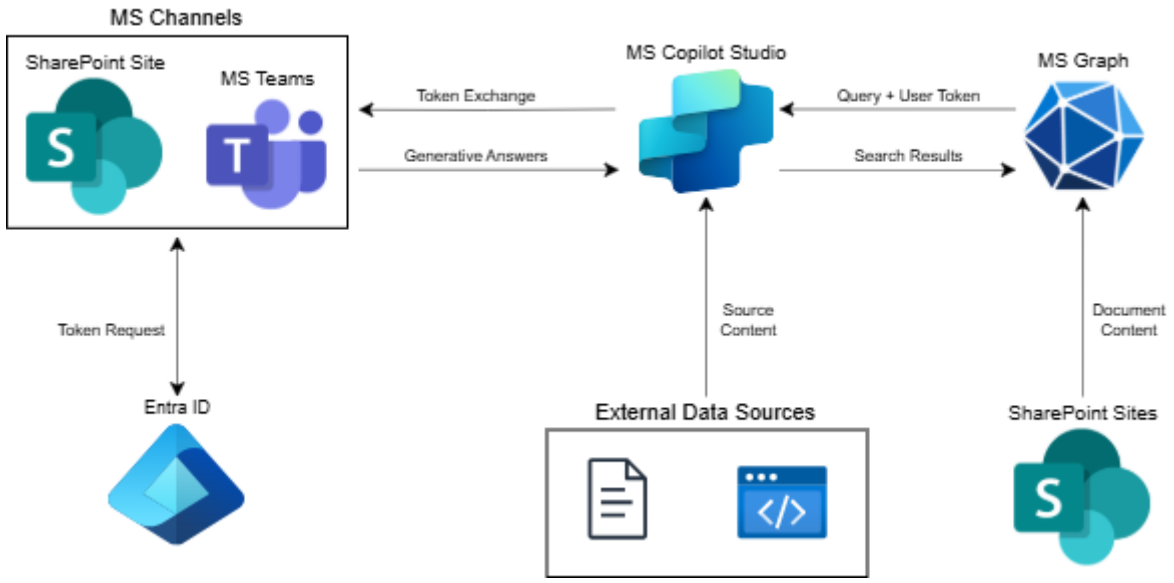

Figure 1 Solution Architecture Overview

1.2 Architectural Overview

At the center of the solution is Microsoft Copilot Studio, provisioned in a dedicated Power Platform Developer environment anchored in the Western Europe Azure region. End-users interact with the chatbot through Microsoft Teams, in personal chat, and where available, via SharePoint web parts deployed on the intranet. Upon submission of a user prompt, the client retrieves an Entra ID (formerly Azure AD) access token representing the user's identity and permissions, and securely transmits both the token and the prompt text to Copilot Studio.

Copilot Studio orchestrates a retrieval-augmented generation pipeline. In the recall phase, the service queries Microsoft Graph using the user's delegated token, thereby enforcing access boundaries identical to those present in SharePoint or OneDrive. Simultaneously, Copilot Studio queries its own internal semantic index, which is constructed and maintained automatically from registered knowledge sources. In practice, this index is populated by the managed SharePoint connector, which performs periodic crawls of the specified document libraries and web sources. The frequency and depth of these refresh operations are governed by Microsoft's backend infrastructure and are not directly configurable by tenant administrators. As a result, there may be a variable delay between updates in SharePoint and their reflection in the chatbot's available knowledge base.

During the reasoning phase, Copilot Studio grounds the question using the most relevant snippets returned from Microsoft Graph and the semantic index, synthesizing a grounded prompt for the large language model. The LLM then generates an answer, always accompanied by citations referencing the underlying source documents. The completed response is streamed token-by-token back to the user in Teams, or in supported configurations displayed as a message within the SharePoint web part interface.

1.3 Component Description

Microsoft Channels deliver the conversational interface. Because both Teams and SharePoint support single sign-on, no additional client-side code is required beyond the Copilot canvas that renders chat messages and citations. Native deployment of Copilot as a SharePoint web part is a relatively recent addition, and requires that each user be assigned an appropriate Microsoft 365 Copilot license. Other, non-native deployment methods (such as embedding via iFrame, SPFx custom web parts, or third-party widgets) may be technically possible, but do not offer the same level of seamless integration, support, or SSO experience.

Microsoft Entra ID issues OAuth 2.0 delegated tokens per user session. These tokens are used to scope all Graph queries and knowledge retrievals, ensuring that Copilot Studio only has access to content the caller could otherwise access directly through the university's Microsoft 365 environment.

Copilot Studio orchestrates the overall query flow. The agent's configuration specifies all active knowledge sources, grounding strategies, and any custom actions. Registered sources can include SharePoint document libraries, uploaded PDFs, and public URLs. The managed SharePoint connector automatically crawls and embeds these sources to populate Copilot Studio's internal semantic index. While the connector generally captures all new or updated documents within several hours, the exact refresh cadence is managed by Microsoft and not exposed to the tenant administrator; manual or forced refresh is not currently supported.

Microsoft Graph acts as the federated search layer, providing content discovery across SharePoint, OneDrive, and (in future) other Microsoft 365 workloads. Copilot's use of the caller's delegated token ensures that all permission boundaries are strictly enforced, with no risk of information over-exposure.

External knowledge sources presently include two PDF academic calendars, and two public university websites.

1.3.1 Knowledge Sources

Source	Type	Location	Notes
Intranet-VSE-root	SharePoint	tenant	Primary intranet root.
Intranet-studenti-VSE	SharePoint	tenant	Shared student resources.
Intranet-studenti-FIS	SharePoint	tenant	Faculty of Informatics & Statistics.
Intranet-studenti-FMV	SharePoint	tenant	Faculty of International Relations.
Harmonogram-AR-2025-2026_ZS_v240125.pdf	PDF	SharePoint library	Academic calendar (winter semester).
Harmonogram_2025_2026_kalendar-1.pdf	PDF	SharePoint library	Full academic-year calendar.
https://fis.vse.cz	Website	public	Faculty microsite; registered as public knowledge source.

https://vse.cz	Website	public	University portal; registered as public knowledge source.

1.4 Operational Data Flow

When a student asks a question in Teams, the text of the query, together with the Entra ID (Azure AD) access token, is transmitted to the Copilot endpoint. Copilot then issues a content query to Microsoft Graph, constrained by the user's delegated permissions, and in parallel performs a semantic search over its own managed internal vector store (which indexes all registered knowledge sources). Microsoft Graph returns ranked results including document snippets, titles, and URLs; Copilot merges these with any semantic matches, constructs a grounded prompt for the large language model (LLM), and generates an answer. After generation, Copilot automatically injects source citations, formats the response as text, and returns it to the user through Teams.

1.5 Knowledge Source Management

Knowledge source indexing and refresh are managed by the Copilot Studio SharePoint connector. Each registered SharePoint site or library is periodically crawled and indexed; the scheduling and depth of these operations are handled by Microsoft as part of the managed service and are not directly configurable by tenant administrators. Copilot Studio's dashboard provides visibility into the status and health of each knowledge source, including alerts for failures or stale content. Manual intervention or forced reindexing is not currently supported.

1.6 Security, Compliance and Governance

Identity and access control are enforced end-to-end through the Entra ID token. All traffic between channels, Copilot Studio and Microsoft Graph is encrypted in transit with TLS 1.2 or higher, and all data at rest in Azure is protected with AES-256. Copilot Studio logs every interaction, including the user prompt, the grounding snippets and the generated answer, into the tenant's audit pipeline so that compliance officers can review or export them to the central SIEM. The Power Platform environment is anchored in the Western Europe region, fulfilling EU data-residency mandates. Existing Microsoft 365 sensitivity labels travel with documents into Copilot answers, and Data Loss Prevention policies can block or mask sensitive strings before the answer is surfaced.

1.7 Reasons for Selecting Microsoft Copilot

This decision wasn't based on a singular factor, but rather a long series of explorations and dead ends in pursuit of a solution that could meet both functional and operational requirements. It's worth noting that our very first approach was, in fact, to prototype with Microsoft Copilot Studio. At the time, the platform's configurability, limited flexibility for custom search flows, and perceived lack of "real" development control left us unimpressed. We initially set Copilot Studio aside, believing we could deliver a richer, more tailored experience by engineering our own retrieval pipelines using Azure AI Search, Microsoft Graph, and external orchestration. What followed was a long series of experiments, each theoretically promising, each ultimately destroyed by rate limiting, security gaps, or operational complexity. After many rounds of false starts and dashed hopes (see sections 1.7.1–1.7.4.1), we eventually returned to Copilot Studio, no longer out of excitement, but because it was the only viable solution left standing. The irony was not lost on us: we circled back to the very tool we originally dismissed, now understanding exactly why its tradeoffs exist, and why, for all its quirks, it actually works in the real world.

1.7.1 Azure AI Search with SharePoint Online Indexer

Our initial foray into this project led me to Azure AI Search, using the SharePoint Online Indexer. It seemed like an obvious choice, especially given the natural integration between Microsoft's search platform and SharePoint. However, we quickly ran into significant roadblocks. The indexer required very specific formatting of queries, as well as extensive configuration via Azure CLI, an added complexity that made it challenging to deploy.

Despite these challenges, we persisted, and attempted to set up the necessary app registration with delegated permissions. While this made the solution technically possible, the complexity involved made it clear that this approach wasn't practical for a university-scale chatbot solution. It was also clear that relying on this method would likely result in significant delays and operational headaches.

1.7.2 Exploring Microsoft Graph API

Next, we briefly experimented with Microsoft Graph API, specifically aiming to retrieve the actual contents of documents, rather than just their locations. This was possible, or so I thought, because Microsoft Teams' search bar, for example, returns exactly the content of the documents. We spent several hours trying to find a similar way to retrieve the content directly from Graph API, but after extensive trial and error, we failed to find a reliable query for this.

This search for a way to extract document content led me back to Azure AI Search, as we believed it was the more promising solution at that point.

1.7.3 Rediscovering Azure AI Search and SharePoint Indexing Limitations

After further exploration of Azure AI Search, we realized that the SharePoint indexer, while powerful, ignores security access scopes. This presented a significant issue: We couldn't be sure that the data indexed by SharePoint would respect individual user permissions.

The common workaround in such cases is to dump SharePoint contents into Azure Blob Storage, which retains security access scopes, and then run Azure AI Search on that blob. However, this presented another problem: We'd either need to use one of our accounts, which would make the solution limited to the FIS Student SharePoint or we'd have to request a some kind of a "global admin" account for the project, an incredibly complicated, lengthy, insecure and uncertain proposition.

At that point, we realized that we were left with a hard choice: either limit the system to one faculty and its student available sources or request permissions that would be difficult to justify, especially given the potential security risks. Either way, it seemed like a significant tradeoff, and it wasn't a viable long-term solution.

1.7.4 Back to Microsoft Graph API

Reluctantly, we returned to Graph API. It was clear that Graph would be the best option for respecting user permissions, but the problem of indexing and data retrieval persisted. Our goal was to avoid storing and managing every file locally, since the associated write/read operations would introduce unwanted latency and potentially create performance bottlenecks, especially problematic for a chatbot expected to respond in real-time.

To circumvent the need for heavy IO, we devised a solution to stream files as bytestreams directly into memory rather than saving them to disk. We then used a specialized library (markitdown) to convert these in-memory byte streams (regardless of the original file type) into Markdown, which we could pass straight into the chatbot. This streamlined pipeline minimized disk operations and kept everything as fast and lightweight as possible.

However, we quickly discovered that not every file download URL from Graph was reliable, sometimes, for reasons still unclear, the URL generation would simply fail. We compensated for this by programmatically retrying for only failed file download URLs using their file-id.

To further optimize, we incorporated a Redis cache to store and reuse answers for similar queries, hoping this would reduce the overall number of calls to Graph. Nevertheless, even with these optimizations, every session still required a burst of API requests to fetch and process files dynamically.

1.7.4.1 The Redis Caching Idea and Rate Limiting Nightmare

This architecture, while clever, was fatally flawed in practice. Microsoft's undocumented rate limits on Graph API became the real project killer. After only a handful of questions, often by just the second query, the entire chatbot would grind to a halt as requests started getting throttled. For all its scalability and technical elegance, the whole solution was effectively rendered unusable for more than a handful of questions.

Any other attempts to extend or upgrade this solution, ran into the exact same existential bottleneck: every user interaction meant more Graph API calls, each subject to the same punishing rate limits. Regardless of how we stored, cached, or indexed the content, the fundamental constraint remained: any solution requiring dynamic, per-session access checks against Microsoft Graph would inevitably collapse under the weight of rate limiting before ever reaching a usable scale.

1.7.5 The Pivot to Microsoft Copilot Studio

At this point, we had exhausted every avenue we could think of. We realized that we needed a solution that could address all of the challenges we faced: document access control, scalability, compliance with EU data residency requirements, and simplicity of integration.

This is where Microsoft Copilot Studio became the only viable solution. Copilot Studio handled access control seamlessly via Entra ID, automatically enforced user permissions, and integrated well with SharePoint and Microsoft Teams, two core systems already in use at the university.

Moreover, Copilot Studio's integration with Microsoft's broader ecosystem ensured that it would scale automatically, with updates handled on the backend, without requiring additional configuration or intervention from our IT team.

Additionally, the fact that Copilot Studio updates monthly and its stable release cycles provided assurance that the system would remain robust and aligned with the university's ever-evolving needs. Microsoft's strategic push behind Copilot, which was becoming ubiquitous in its product suite, further reinforced the long-term value of investing in this solution.

2 Solution Documentation

2.0 Agent Configuration and Setup

The SOVA chatbot agent is provisioned and managed through Microsoft Copilot Studio, where all core settings, knowledge sources, and user access policies are defined. The agent is designed to deliver dynamic, context-aware answers through advanced generative AI, while maintaining strong guardrails for academic integrity, compliance, and user privacy.

During deployment, we enabled Generative AI orchestration, which allows SOVA to generate answers dynamically, leveraging both curated university data sources and, where permitted, general foundational knowledge. Deep reasoning (Preview) is active, enabling the agent to perform advanced synthesis, multi-step reasoning, and more sophisticated answer construction, particularly useful for complex or multi-part student queries. Connected agents are currently disabled, keeping the SOVA agent as a standalone service, but this can be revisited if workflow orchestration across agents becomes a requirement in the future.

The primary response model is set to GPT-4o (default), selected for its broad language capabilities and superior contextual reasoning. Responses follow the agent's default formatting, with future upgrades possible as more advanced formatting controls are released.

User feedback collection is enabled. Students may rate responses with thumbs-up or thumbs-down and leave optional comments, with all feedback being logged for internal review and continuous agent improvement. This feedback does not leave the university tenant or go to Microsoft.

The agent's knowledge configuration allows the use of general foundational knowledge, ensuring that broad, non-university-specific questions can be addressed, but web search via Bing is explicitly disabled. All answers are grounded in the university's own knowledge sources, as defined in Section 1.3.1, guaranteeing control over cited content. File processing is enabled for image uploads (PNG, WEBP, JPEG, non-animated GIF, up to 15MB), though this capability is not currently a core use case.

Tenant graph grounding with semantic search is enabled, providing improved search and retrieval performance on Microsoft 365 content for users with Copilot licenses. Document freshness is maintained by automatically filtering on upload date (using the Text(Now(), DateTimeFormat.UTC) function in configuration).

A summary of the current agent configuration is presented below:

Parameter	Setting / Value	Notes
Orchestration	Generative AI	Dynamic, context-aware responses leveraging available knowledge
Deep Reasoning	Enabled (Preview)	Allows advanced multi-step reasoning and synthesis
Connected Agents	Off	Standalone agent, no cross-agent workflow
Primary Response Model	GPT-4o (default)	State-of-the-art, broad language capabilities
Response Formatting	Default	Will update as new controls become available
Moderation	High	Maximum content safety for student/institutional compliance
User Feedback	On	Thumbs up/down + comments, logged internally only
Disclaimer	Not set	No custom disclaimer currently configured
General Knowledge	On	Permits broad knowledge in responses, grounded in curated sources
Use Information from the Web	Off	Web search disabled, ensures only university sources are cited
File Processing (Image Uploads)	On (15MB, .png, .webp, .jpeg, .gif)	Not a core use case yet; capability present
Search: Tenant Graph Grounding	On (Premium)	Improved semantic search for M365 Copilot tenants, depends on license availability
Date/Time Freshness Filter	Text(Now(), DateTimeFormat.UTC)	Ensures latest documents are indexed and used for retrieval

2.1 Operational Data Flow in Production

In production, the SOVA chatbot functions as a tightly integrated component within the university's Microsoft 365 environment. When a student initiates a query via Microsoft Teams, the agent authenticates the user in real time using Entra ID (Azure AD). The delegated access token includes the user's faculty affiliation and group memberships, which are leveraged to dynamically filter all downstream data retrieval to only those knowledge sources the student is permitted to access.

Once authenticated, Copilot Studio orchestrates parallel retrieval actions: a Microsoft Graph query scoped by delegated permissions, and a semantic search against the agent's managed internal vector store. The Graph query enforces strict document-level access, ensuring content is never exposed beyond what a student could access natively through SharePoint. The semantic index, maintained automatically by Copilot Studio's SharePoint connector, provides additional relevance scoring and snippet matching from registered sources.

Results from both retrieval paths are merged and filtered by Copilot Studio's logic, ensuring deduplication, recency, and contextual fit. The resulting snippets are then grounded into a prompt for the GPT-4o model, which synthesizes a contextually aware, citation-backed response. This answer is rendered as an adaptive card within Teams, where the user can provide immediate feedback. All data transformations and retrievals are logged in the university tenant's SIEM pipeline for operational and compliance monitoring.

2.2 Knowledge Source Management and Quality Assurance

All knowledge sources are managed through the Copilot Studio interface and configured as described in Section 1.3.1. Each SharePoint site is registered as a discrete source, with access mapped to faculty or group via Entra ID claims. The SharePoint connector is responsible for crawling and indexing the selected document libraries and files; however, the refresh schedule and crawl depth are not user-configurable and are fully managed by Microsoft. The PDF documents and web portals are essentially public, meaning anyone using the copilot is able to access these sources.

Quality assurance is a combination of automated and manual processes. Copilot Studio's dashboard provides near-real-time status on the health and currency of each knowledge source. Alerts are triggered in case of crawl errors or stale content. During the pilot phase, the admin team manually verified document availability by issuing test queries after new uploads or major updates. In observed operation, new documents typically became accessible via the chatbot within a few hours, though occasional lags were noted.

For critical updates (such as new academic calendars or urgent policy changes), a direct verification is recommended using both dashboard checks and live queries.

2.3 Frontend Outputs and User Experience

The only frontend currently in use is the Microsoft Teams chat interface. Students access the SOVA agent through a conversational UI embedded in Teams, which presents a familiar, intuitive user experience. On first interaction, users are shown prompt tiles (in Czech) to help illustrate the chatbot's capabilities. All responses are returned as adaptive cards containing the answer text, explicit source citations (with direct links to documents), and thumbs-up/down controls for feedback.

Feedback data is logged internally and reviewed regularly to drive ongoing improvement of both the agent's response quality and the underlying knowledge base. All frontend logic and UI elements leverage Microsoft's native Teams components, with no custom code or client-side scripting required.

Integration with SharePoint as a native web part remains on the roadmap and is expected once product maturity and university licensing permit.

2.4 Implementation Trade-offs and Lessons Learned

The move to Copilot Studio required accepting several trade-offs. Most notably, while Copilot Studio provides a managed, scalable, and secure foundation, it limits the degree of backend and retrieval customization available to the development team. Advanced or custom retrieval logic, fine-tuning of LLM prompts, and direct control over document refresh intervals are not supported; all of these are managed by Microsoft's backend and subject to change without notice.

Previous attempts at more custom/robust solutions; custom retrieval pipelines, caching layers, or direct Microsoft Graph API calls, were blocked by strict rate limiting, security enforcement, and operational complexity (see Section 1.7). As a result, all retrieval and reasoning is now handled through Copilot Studio's managed workflow.

Despite these constraints, the solution offers robust compliance, seamless Microsoft 365 integration, horizontal scalability, and rapid onboarding for new data sources and users. The experience confirmed the value of prioritizing operational reliability, compliance, and user privacy over maximum technical flexibility.

3 User Testing Results

3.0 User Testing Procedure

User testing was conducted as a targeted pilot involving six participants: five students from the Faculty of Informatics and Statistics (FIS), the primary intended user group and one student from the Faculty of International Relations (FMV), providing an additional perspective from a different faculty. All testing was performed in Microsoft Teams using the SOVA chatbot. Each participant completed five question-and-answer tasks, where of them were predefined and two were chosen freely by each student.

While the sample size was limited, this approach enabled rapid collection of detailed feedback and surfaced core usability and accuracy issues for the most relevant demographic.

3.1 Tasks and Scenarios

The predefined test questions used in all sessions were as follows:

- **Kdy přesně začíná a končí zimní semestr?**
  (When exactly does the winter semester begin and end?)
- **Co musím udělat před tím, než začnu psát bakalářskou práci?**
  (What must I do before I start writing my bachelor's thesis?)
- **Jak se přihlašuje na státní závěrečnou zkoušku?**
  (How does one register for the state final examination?)

Participants were then encouraged to ask two additional questions of their own choosing relevant to their studies or general university procedures.

3.2 Testing Criteria

The chatbot was evaluated according to the following criteria, each rated on a 1 (poor) to 5 (excellent) scale:

- **Ease of Use:** How intuitive and accessible the chatbot interface was for new users.
- **Response Accuracy:** The correctness and factual reliability of answers provided by the chatbot.
- **Performance:** The speed and consistency with which responses were delivered.
- **User Satisfaction:** Overall satisfaction with the answers and the chatbot's utility.

3.3 Documented Problems

Several recurring issues were identified during the user testing phase:

- The chatbot occasionally misunderstood or misinterpreted the intent of user questions, sometimes providing answers to a different (but related) question.
- There were occasional technical difficulties, including minor response delays and occasional inaccuracies in the retrieved content.
- Some students reported the initial prompts to be confusing and chatbot responses longer than expected.
- It was also noted by several students that certain answers were excessively verbose, providing more context or background than was required.

3.4 Quantitative Results

The testing produced the following quantitative results:

- **Ease of Use:** All students rated the chatbot's ease of use as 5 (excellent).
- **Response Accuracy:** The FMV student rated accuracy as 4, while FIS students gave ratings ranging from 3 to 4.
- **Performance:** All participants rated performance between 2 and 3.
- **User Satisfaction:** Satisfaction scores ranged from 3 to 5, with an overall average of 4.

3.5 Quanlitative Feedback

Qualitative feedback collected from the students highlighted several key themes:

- All participants agreed that the chatbot's interface was highly intuitive, easy to use and actually accessible.
- Some students noted that answer accuracy could be improved, particularly in cases where the chatbot provided information that did not exactly match the intent of their questions.
- A recurring suggestion was to upgrade to a more capable language model as soon as such an option becomes available, as this is expected to address many of the observed accuracy and relevance issues.

3.6 Visual showcase of answering questions.

The following image illustrates the initial user experience when the SOVA chatbot is first opened in Microsoft Teams. The central area displays six sample prompt tiles in Czech, guiding users in the types of questions they can ask.