Use case	**Weekend planner**
The application is able to make students' lives easier, reducing the number of hours spent in organizing material and studying for the exam. This is achieve by summarizing your resources, enabling real-time conversations with your documents, effortlessly organizing your materials, and generating personalized practice exams	
Test Objectives	
Informal definition of quality for the system	The effectiveness of the application is measured in terms of: - summary content fidelity and readability - precision of drive reorganization - ability of answer user questions about the material - create mock up exam that fulfill user request
Metrics	
Summarizer	
Content-Fidelity F1	***What*** → Overall faithfulness of the summary to its source. ***How*** → Generate *N* QA pairs on the source (using Transformers library). Answer each pair with (a) source, (b) summary. Compute token-overlap F1 and average. ***Target*** → **≥ 0.70** (block deploy if < 0.65).
QA-Precision	***What*** → Fraction of tokens in the summary-based answer that appear in the gold (source-based) answer. ***How*** → Same QA pipeline; Precision = TP / (TP + FP). ***Target*** → **≥ 0.65**.
QA-Recall	***What*** → Fraction of gold-answer tokens recovered by the summary answer. ***How*** → Recall = TP / (TP + FN). ***Target*** → **≥ 0.65**.
Flesch Reading Ease	***What*** → Ease of reading for undergraduates. ***Target*** → **≥ 60**.
Flesch–Kincaid Grade Level	***What*** → U.S. school grade needed. ***Target*** → **≤ 8**.
Gunning Fog Index	***What*** → Years of formal education implied by sentence length + complex words. ***Target*** → **≤ 12**.
Conciseness	***What*** → Compression ratio of summary to source. ***Target*** → **≈ 0.30 ± 0.05**.
LLM	Provide to a LLM (a powerful one) the original document, the summary and ask to evaluate the summary
Drive Organizer	
Intra-cluster similarity	***What*** → files within the same subfolder should be very similar in content. . ***Target*** → **≥ 0.70**
Low Inter-Cluster Similarity	**What** → files in different subfolders should be dissimilar **Target** → <=0.30

Silhouette Coefficient	**What** → this measures how similar a file is to its own cluster compared to other clusters **Target** → =>50
Rag	
Content-Fidelity F1	***What*** → Overall faithfulness of the summary to its source. ***How*** → Generate *N* QA pairs on the source (using Transformers library). Answer each pair with (a) source, (b) chunk retrieved from db. Compute token-overlap F1 and average. ***Target*** → **≥ 0.70** (block deploy if < 0.65).
Exam generator	
Correctness	**What** → evaluate the correctness of the generated exam **How** → Get embedded chunks from db, compute embedding for each question, compute cosine distance (question - chunks), take the maximum similarity score. **Target** → >=0.7
Relevance	**What** → evaluate the relevance of the generate exam respect to the topic covered on the documents **How** → extract top N key-phrases from document (TF-IDF), count how many key-phrases appear in the question, compute ratio **Target** → >=0.5
Experiment design	
Systems under test	Summarizer Drive Organizer RAG Exam Generator Orchestrator
End to end	The end-to-end evaluation involves analysing the performance of the orchestrator, since it is the only agent with which the user interacts directly. This is achieved by storing the trace of choices containing the agents that the orchestrator decided to call in order to fulfil a user request in the orchestrator's memory. If the agent was able to fulfil the request, nothing happens; otherwise, the memory is updated to suggest which agent should be called next. The orchestrator uses the memory to inform future choices. The orchestrator should achieve an accuracy of >0.9.

Memory and responsiveness to memory and context	Each agent has its own memory that it uses to better handle the user question over time. - Summarizer: the agent stores the user preferences about the summary. For instance the preferred length or the type of the summary. - Drive organizer: the agent stores preferences about how the user prefers the drive to be organized. - RAG: the agent stores the preferences about how the user prefers to receive answers. For instance the length of the answer, how much detail to insert in the answer. - Exam generator: the agents store the user preferences about the generated exam. For instance difficulty, length of the exam, type of question. - Orchestrator: the agent stores information about if it makes the correct choice while delegating the user answer to a specific agent. The agent has a long memory for each user.
Configuration / alternatives	
llms	Try bigger and better performance LLM for the agents
Communication protocols	The agents communicate to the others using an A2A protocon, while a simplified version of the MCP (Model Context Protocol) is used to enable the communication between the single agentes and their tools. So, each can decide which tool to run and use.
Datasets	
Type (golden dataset? Select on the fly – different each time?)	Keep one golden dataset on which we evaluate the agents
Is test data made up so that we can run controlled experiments	The dataset contains a set of topics on which we can run controlled experiments,
How do we make it up	Run the system and evaluate the metrics for each agent
Dataset creation workflow	The dataset contains a set of material types. For each material type there is a set of content types in order to evaluate the agents using a wide range of topics that might be covered in a university course.
Diversity and size criteria	
Personas	Students, Professors
Material type	- Lecture content - Textbook - Academic Journal Articlesù - Discussion Forum Threads - Assignment Description
Content Type	For each material type the dataset contains the following content type: - STEM - Humanities and Social sciences - Medial and Health sciences - Law and legal studies - Business, Management and Economics - AFAM (Fine arts, music and dance)
Size	A few hundred test data points with diverse data as described above

Is the ground truth in the golden dataset?	No
Eval workflow	When we want to evaluate an agent, we run the agent using the dataset and we observe the result
End and reporting	
End and reporting	Run the test and insert the result of the evaluation on the long memory of the agent so that in the future it is able to handle better the user request, If the performance of the agent is poor, consider updating the tool and debug what should be improved. Once the problem is found, fix it, and run the evaluation test again. Repeat this loop until the level of performance described above are reached.
Metrics in production	
Goal	Detect quality of the agents before users complain
What do we log?	• Request/response IDs • Full prompt, model output, token counts. • Agent chain trace (which agent, parameters, call order, retries). • Latency per micro-service. • Cost estimates from model provider.