TAKNEEK PS - ZENITH

(100 Points)

TheDrive

Introduction :

With the rise of AI and growing privacy concerns in this data-driven world, it's high time to **revolutionise cloud storage systems**. These systems should be **smarter**, able to understand and answer queries based on the provided data, analyse it, and make changes in seconds. All of this while **ensuring secure cloud storage** solutions.

Task / Problem Statement :

Develop an **AI-powered cross platform file storage system**, similar to Google Drive, or One Drive to transform how users interact with their digital assets, **offering intelligent insights, explanations, and actions** while ensuring paramount **data privacy and security.**

1. **Multi-File Format Support:** The application must support diverse file formats, maintaining content integrity and layout awareness where applicable:
    a. **Document Formats:** .pdf, .docx, .txt, .pptx (PowerPoint presentations).
    b. **Spreadsheet Formats:** .csv, .xlsx (Excel spreadsheets)
    c. **Image Formats:** .jpg, .png
2. **Smarter Storage System (AI Integrations)**
    a. **Contextual Chat:** Users should be able to do
        i. Chat over **Whole Drive Context:** The user should be able to ask questions about the files present in the whole drive, be able to calculate aggregation metrics, ask questions about metadata, ask questions pertaining to multiple files, have context about the knowledge contents of multiple files so that if question is asked for a given file, then context can be extracted from other related files as well. Note that all chat interfaces should be well integrated with the drive UI. This means that there should not be a separate UI for the chat - optimize for user experience and convenience.
        ii. Chat over **Drive Folders:** The features that have been described for (i) should also be applicable at a drive folder level. This is to say when the user chats at a folder level, there will be no information leak from other folders and the context will be contained.
        iii. Chat over **Specific Documents**
        iv. **Highlight-Based** query Resolution, Explanation, Definition summarization, Elaboration/Expansion over pdfs and text files: The user should be able to:
            1. Highlight text, and an option for AI popup should appear. When the AI popup feature is selected, definition and

            explanation about the selected text should be provided. This can look like: the user highlighting a word and the AI popup feature providing its definition and usage.
        2. Select a portion of an image/presentation and description of the same will be provided by the AI popup feature
        3. Select a portion of a table, and aggregation metrics and information will be provided about that portion
        4. Subjective brownies for any other features and AI integrations the developers are able to come up with.
    v. **Cross reference** files/sections anywhere in storage / file.
        1. Whenever information is extracted from a file while answering a question, the source should be provided
        2. Wherever the citation is provided, the file location should be linked
        3. When the linked file is opened, the specific chunk from where the context is extracted to answer the question should be highlighted
        4. Subjective brownies for visual representation of knowledge connections in the form of interpretable structures like knowledge graphs, flowcharts etc.

  b. **Other Multimodal AI-Based Features**:
    i. **Image Understanding**: Analyze content within images (e.g., diagrams, charts, handwritten notes via OCR), allowing users to query visual data and answer questions about specific entities within the image
    ii. **Semantic Search**: Searching based on the meaning and context of queries across all indexed content.

3. **Mobile App and Web App**
    a. Deliver a **login-based, synchronized mobile and web application system** integrated with an Agentic AI system. (cross platform application)
    b. Ensure a **user-friendly interface** that enables seamless AI interaction.
    c. Design the interface to allow users to easily view, **search, filter, upload, download and view assets/files**.
4. **Secure Storage:**
    a. **End-to-End Encryption (E2EE)**: All user data (**files, metadata, chat histories, derived insights**) must be encrypted at rest and in transit. Remember "**admin can't access users' data**".
    b. The Agentic system should only be **given the data access if required after user's permission**, else it should be completely encrypted and stored.
    c. **Data Segregation**: Each user's data must be logically and physically isolated to prevent cross-user data leakage.

Deliverables:

1. **Android Application** (APK) and **Web Application** (hosted url and source code)
    a. An APK file for the mobile application (**size at most of 300MB**)
    b. A fully hosted Web application.
2. **Backend**
    a. A complete **API documentation** of backend on **Postman** for testing
    b. Backend **server hosted over on a cloud platform** (like AWS, Azure, etc.)
    c. A script to directly populate users and their data from a folder (you will be notified, if the date is updated) (**unable to provide one will result in direct disqualification**)
3. Clear & Precise **Documentation** hosted over GitHub pages explaining:
    a. The **project architecture**, **setup instructions** from scratch.
    b. User interface design and functionality
    c. **Data pipeline**, **Agentic, security** & **compression workflows** if used.
4. **Presentation**,
    a. Which you will be presenting later.
    b. Presentation time will be a **maximum of 10 minutes**.
    c. **At least 2 members** will be presenting.
5. **ZIP file** containing all the source code submitted in the GitHub Repo.
6. **Cloud Server Access:**
    a. Set up a role on the cloud server management console (such as an IAM)
    b. Submit the credentials of the user/role.

Deadline: 2200 hrs, 3rd September 2025

Evaluation Metrics :

Note: The weightage provided in front of each solution component is not used for objective scoring, but to only provide the participants a brief idea about the importance of various components. Metrics for which no weightage is provided are general recommendations and things we'll look out for during evaluation. These are not "application components", but "solution components" which are integral for the judges to be able to comprehensively evaluate the solution. Just because weightage is not provided for these components, it does not mean that they are unimportant. Also, the weightage is subject to change and extremely subjective - do not take it literally. The scoring and evaluation of the solutions would be completely subjective.

1. **Functionality (50%):**
   The ability of the system to accurately **answer user queries** described in the "**Smarter Storage System**" section.

a. **Method of judging AI response (30%)** :
Upon submission we will be **using the auto user generation script provided by the team and a fixed set of user queries** (same for all the teams).
The response will be judged upon the following criteria:

i. How well is the **chat over different folder structures (6%)**
ii. How well it **understands images (8%)**
iii. How well it works in with **tabular data in analysis (4%)**
iv. How well it works with the **highlight features (12%)**

b. **File Viewing system (15%)**

i. User interaction with different file formats over the platform. **(7.5%)**
ii. Comments, edits & highlights manually or by AI responses **(7.5%)**

c. **Security (5%)**
All stored data should be **encrypted** and only exposed to the agentic system when requested by the user with proper scope definition.

2. **User Interface and Experience (20%)**

a. **User Interface Design: (10%)**

i. Minimal and functional design for the users over both the platforms.
ii. Focus on the **ease of use of the AI solutions**.
iii. Ease of **importing, uploading & downloading** files.
iv. File Manipulation (move, rename, etc)

b. **Search and Filter Functionality: (10%)**

i. Basic search & advance AI search integration (as described earlier)
ii. Customizability of search and filter options
(e.g. by type, last modified, upload date etc).

3. **Documentation and Presentation (15%)**

a. Clarity and completeness of documentation **(3%)**

i. **Attention to detail:** How do you ensure polish and accuracy across all aspects of your work, code correctness, error handling, edge-case coverage, tests.
ii. **System architecture:** How well do you explain the basic system architecture concepts? Is your application architecture reasonable? What tech stack do you use and why was that specific selection made? How and where are you storing data? What's the distribution of client-side and server-side processes?
iii. **Trade-offs:** Have you explained basic architectural decisions and why you rejected alternative approaches? Do you conduct a trade-off analysis between various possible alternatives? Do you try out and compare multiple approaches, and benchmark them against standard methods? (this along with the System Architecture should be highlighted in the repository README, presentation and/or video demonstration)

b. A well-structured README file with clear instructions for set up **(2%)**

   c. Discussion of challenges encountered and solutions. **(2%)**
   d. We defined Postman Public WorkSpace for server testing **(2%)**
   e. Presentation **(6%)**

4. **Code Quality and Additional Details: (10%)**
   a. **Code quality:** How do you keep your code readable and modular, with meaningful names and minimal duplication?
5. **Dockerization (5%)**
   Both the parts, Web Application and The Backend Server.
6. **Question and Answer Round:** The question and answer round will be used to determine the scores for the first 5 points. This means that it will not have a separate marks allocation for QnA, but the responses of the QnA will be used to increase or deduct the scores in the first 5 sections.
   Example: If the UI is completely vibe coded and the developers are not able to answer the questions related to it in the QnA round, points will be reduced from the User Interface and Experience section.

Guidelines

Strict alterations in evaluation procedures will be taken in order to judge the problem statement on the merits of the developers and not the merits of the Large Language Model or the Vibe Coding platform. These shall, again be highly subject to the experience of the evaluator, but we shall try to list down the same in a structured fashion:

1. **Focus on knowledge during presentation:** If developers are not able to answer any question on a feature or component during the presentation, but the same is present in the codebase, the implementation of the feature shall be considered null and void.
2. **Focus on functionality over aesthetics:** Aesthetics of a platform are extremely important in order to increase the usability and make it look ‘classic’, but functional attributes like User Experience, Intuitiveness of the UI, convenience of accessing the features will be preferred over any flashiness in the UI.
3. **Clear description and understanding of system architectures and tradeoff considerations:** Large Language Models often come up with standard procedures, use standard tools, program functionalities in a certain fashion - and yes we are aware of all these patterns because we have also vibe-coded enough and we will be doing a thorough study of suggestions of LLMs for solving this problem statement. Think beyond the suggestion of the LLM, research about various optimizations, understand tradeoffs and do some research from ground up. Read blogs, understand why certain tools are preferred over the others in certain scenarios etc. These explanations should also be clearly mentioned in the README, and the participants should have a detailed understanding of these explanations. If we find that the explanations are AI-generated and participants are not able to answer questions, heavy penalties will be applied.

4. Lastly, even if the quality of one solution is objectively better than the other - we might choose to judge the objectively worse solution as the better one in case the developers of the prior are not able to explain their developments.
5. **On involvement of Y25s:** More than often - as observed in the previous iterations of the competition - freshers are not involved in the development of the platform, but explained specific sections of the applications just for the sake of presentation. In order to prevent this kind of deception, specific questions will be targeted towards freshers. These questions will not only be generic, but will also involve implementation based questions. In case it is observed that the concerned freshers were not involved in the development of the PS from scratch, the teams will be heavily penalized.

Rules and Team Composition :

1. You must use **only free-tier services** for hosting and development of the application. Disposable API keys should be provided to the judges for testing the application.
2. **Team Composition** : Each team should **have 8 members, with at least 3 Y25s and 3 Y24s.**
3. It is possible that both the APP and Web interface support different sets of features but the AI interaction should be the same over both the platforms.

For Any Queries, The Pool Captains and PS Leads are encouraged to ask in the Discord channel.