The core of the application is built using Flask, which is a lightweight web framework that allows us to easily handle HTTP requests and define the necessary routes in our application. Flask is used because of its simplicity and scalability—making it a good choice for a PoC. Alongside Flask is SQLAlchemy, which is used as the ORM layer to communicate with the SQLite database. SQLAlchemy simplifies working with the database, as it is easy to define models and to perform CRUD operations. The engine in this database is SQLite, which stores notes. It is a lightweight and serverless database. Being a very simple and self-contained SQL database engine, it works quite well for a PoC.

The fuzzywuzzy library is included for optimal search functionality. It is utilized for fuzzy string matching, so it will also support a small amount of natural language queries in such a way as it will look for a near match to the user query within the notes. The Python language has found its way through the entire project in the ecosystem of libraries and frameworks that make it possible for the development of sites and data processing.

The database layer can be abstracted into the Note model. The Note is just a SQLAlchemy model to hold the notes stored in the database. Each of those notes contains an id (primary key) and content (the text of the note). Interactivity with the application is governed by a few routes and endpoints. An /add_note endpoint posts a new note. An /get_recent_notes endpoint retrieves the most recent notes. The amount of notes that should be retrieved is passed in via a query parameter. The /search_notes endpoint searches for notes using natural language queries. It uses the function get_synonyms to expand synonyms and fuzzy matching to fetch the most similar notes in relevance. Lastly, the /clear_notes endpoint clears everything stored by deleting all the notes in the database.

Synonym expansion: Common queries are mapped to related terms to make the search more accurate. The parse_query function maps the natural language query to predefined keywords for better handling. The process.extract function from the fuzzywuzzy library performs fuzzy matching to get approximate matches of the search query with the stored notes.

The application was tried to be run on PythonAnywhere for deployment purposes. However, deployment issues were faced, as PythonAnywhere did not properly recognize the application. This one requires a further look in order to solve deployment problems. Future enhancements for the application include adding large language models (pre-built models provided by APIs like OpenAI's GPT, Google's Dialogflow )to increase the sophistication of natural language understanding and increase search precision, user authentication so multiple users can have their own note spaces, advanced searching features like filtering by date, category, or tags, and extension of the application to cover rich media content notes in terms of images, audio, and video.