“CommPass” Project

The "CommPass" project is designed to revolutionise meeting efficiency and decision-making within corporate environments. By converting traditional meeting transcripts into structured, easily navigable visual maps, "CommPass" provides a powerful tool that enhances meeting clarity and actionable outcomes. The key benefits are time savings, improved clarity, enhanced decision-making, and actionable outcomes.

CommPass link: https://commpass-graph-visual.vercel.app/

GitHub resources link: https://github.com/aadityachapagain/Capstone_project

1. Frontend

1.1 Overview

The user interface for the "CommPass" project is built using React, specifically version 18.2.0. We are using Node.js version 20.4.0 and npm version 9.7.2.

1.2 Getting Started

a. Tech Stack

- Reactjs
- React-Router-Dom
- Axios
- D3js
- React-Table

b. Development Mode

To run the app in development mode, go to the frontend project directory and execute:

npm start

Open http://localhost:3000 to view it in your browser.

The page will reload when you make changes.
You may also see any lint errors in the console.

c. Production Build

To create a production build of the application, go to the frontend project directory and execute:

npm run build

To build the app for production to the build folder.
It correctly bundles React in production mode and optimizes the build for the best performance.

For our project we have deployed the build file to Vercel. Vercel is a cloud platform for static sites and serverless functions that enables developers to deploy websites and applications quickly and efficiently.

2. Backend

2.1 Overview

The project "Commpass" backend service is built using Python and focuses on providing RESTful APIs through the FASTAPI framework. It utilizes MongoDB for Non-Relational database management, which stores audio and transcription files,graphs JSON and metadata, Docker for containerization, and AWS Lambda for deployment. Additionally, it integrates AI models from Hugging Face and ChatGPT for sophisticated functionalities.

2.2 Tech Stack

- Python [Programming Language][version==3.11.4]
- MongoDB [Database]
- FastAPI [RestFul API]
- Docker [Containerization]
- AWS Lambda [Deployment]

2.3 AI Models

- GPT-4 LLM [Graph Generation]
- Claude LLM [Text Summarization]
- Hugging Face [Name Entity Recognition, & Punctuation parser and predictor]

2.4 Installation and Deployment

- Clone the GitHub repository: `git clone https://github.com/aadityachapagain/Capstone_project`
- Create a virtual environment: `python3 -m venv <venv_name>`
- Go to the backend directory: `cd Capstone_project/app`
- Create an .env file with following API tokens:
  `claude_token="<CLAUDE-API-TOKEN>"`
  `db_username="<MongoDB-USER-NAME>"`
  `db_password="<MongoDB-Password>"`
  `llm_key="<OPENAI-TOKEN>"`
- Build the docker Image: `docker build -t <IMAGE_NAME>:<IMAGE_VERSION> .`
- Run the docker Image: `docker run --name <CONTAINER-NAME> --env-file .env -t -p 8080:8000 <IMAGE_NAME>:<IMAGE_VERSION>`
- Deploy serverless through AWS Lambda: `aws deploy`

3. Testing

3.1 Overview

The CommPass project employs a robust testing strategy to ensure the reliability and performance of the application. Our testing approach includes:

3.2 Frontend Testing Case

- Functionality Testing
- Compatibility Testing
- Performance Testing

3.3 Backend Testing Case

- Unit Testing
- Integration Testing
- Database Testing
- Performance testing

3.4 Testing Tools Used

While most test cases were executed manually, we also utilized specialized tools for backend integration and frontend performance testing.

- Backend Integration Testing: Conducted using Postman to ensure seamless communication and functionality between various backend services.
- Frontend Performance Testing: Performed using Google Lighthouse to optimize and verify the application's speed, responsiveness, and overall user experience.

Contributors

Aaditya Chapagain, Ali Aamir Khan, Anish Pandey, Burak Cem Boga, Sunil Bista, Xiaoming Mo, Xuerong Ji