

CS 6422 DBSI

EvaDB Project 2

Shreyas Kanjalkar

https://github.com/skanjalkar/evadb_resume_matcher_app



EvaDB

CONNECT AI TO YOUR DATABASE

Introduction

My project is using EvaDB to find the best job applications for a resume that the user will upload. The inspiration for this application was based on personal needs, as I am someone who is pursuing job applications and wanted to build an application that would help me streamline the process a bit more. This endeavor is not just about technological innovation; it's also a reflection of my empathy and understanding of the complexities and frustrations inherent in searching for a job in today's competitive landscape.

EvaDB

EvaDB is a cutting-edge database system specifically designed to empower the development of AI-powered applications. At its core, EvaDB serves as a robust query engine that seamlessly integrates with various data sources, such as PostgreSQL, S3 buckets, and

standard SQL database systems. This integration allows developers to build scalable AI applications over data stored in these systems using straightforward SQL queries. The system's design is highly conducive to the development of AI models for a range of purposes, including regression, classification, and time series forecasting. With EvaDB, software developers can create AI applications in just a few lines of code, significantly simplifying the development process for both structured and unstructured data.

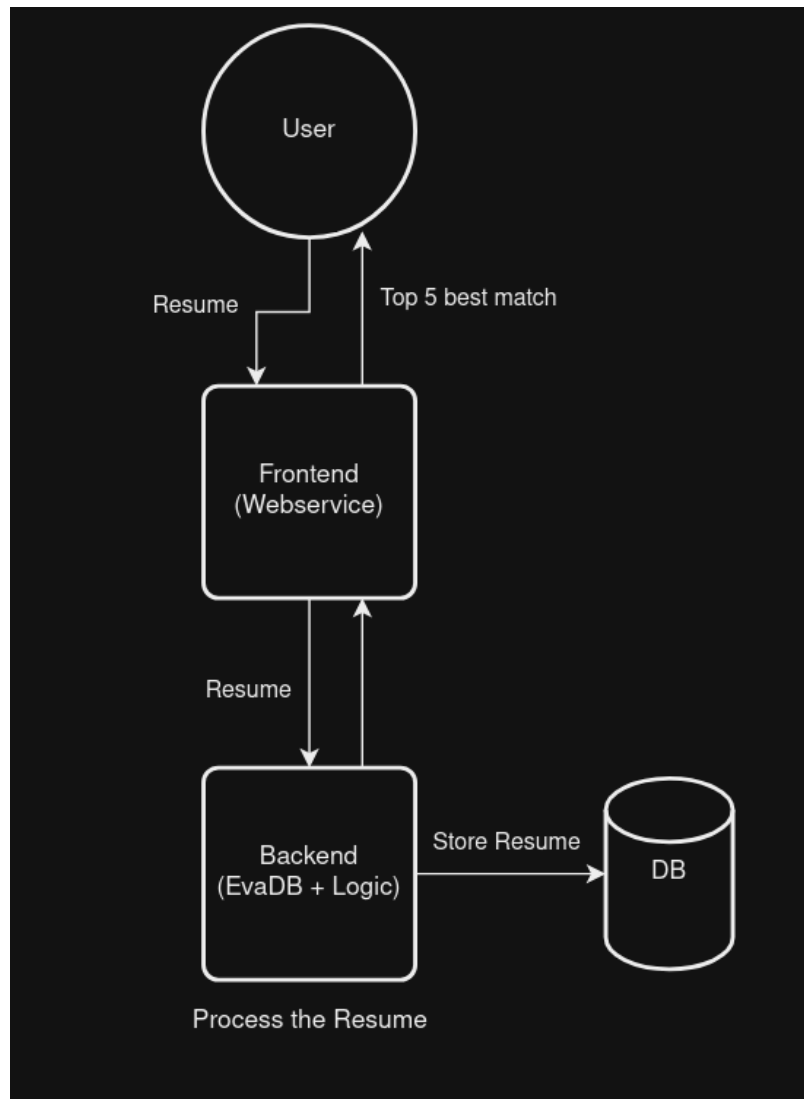
One of the standout features of EvaDB is its support for pre-trained AI models from renowned platforms like Hugging Face, OpenAI, YOLO, and PyTorch. This capability enables developers to query data using these sophisticated AI models, facilitating the incorporation of advanced AI functionalities into applications without the need to develop these models from scratch. Furthermore, EvaDB introduces its own query language, EvaQL, which is tailored for AI applications and derived from SQL. In EvaQL, AI models are integrated as simple function calls within the query, streamlining the process of leveraging AI in applications and data analysis. This unique approach makes EvaDB an invaluable tool for developers aiming to harness the power of AI in their software solutions, offering ease of use and flexibility in the rapidly evolving landscape of AI technology.

The Tech Stack:

My application has a simple microservice architecture as shown in the flowchart, and it consists of three things:

- Frontend
- Backend
- Database (WIP)

The Microservice architecture uses a webservice, which is more bandwidth oriented to serve more users and does not have much computing need. Therefore, the logical conclusion is to make the webservice more network oriented to support fast upload and communicate with the Backend.



Frontend:

The Frontend is made using Javascript. I used a bootstrap using ExpressJS framework to create a template. Then modifying that template, I created a Frontend UI which allows the users to upload a PDF. The webpage also explains how the process works at a high level. Figure 1 shows how the FrontEnd looks. Admittedly, I could have used the newer frameworks such as React, but as I am not comfortable with them, in the interest of time I decided to go with ExpressJS as I am more comfortable with that. The webservice uses **RESTFul** api design, to send data via POST request to the backend.

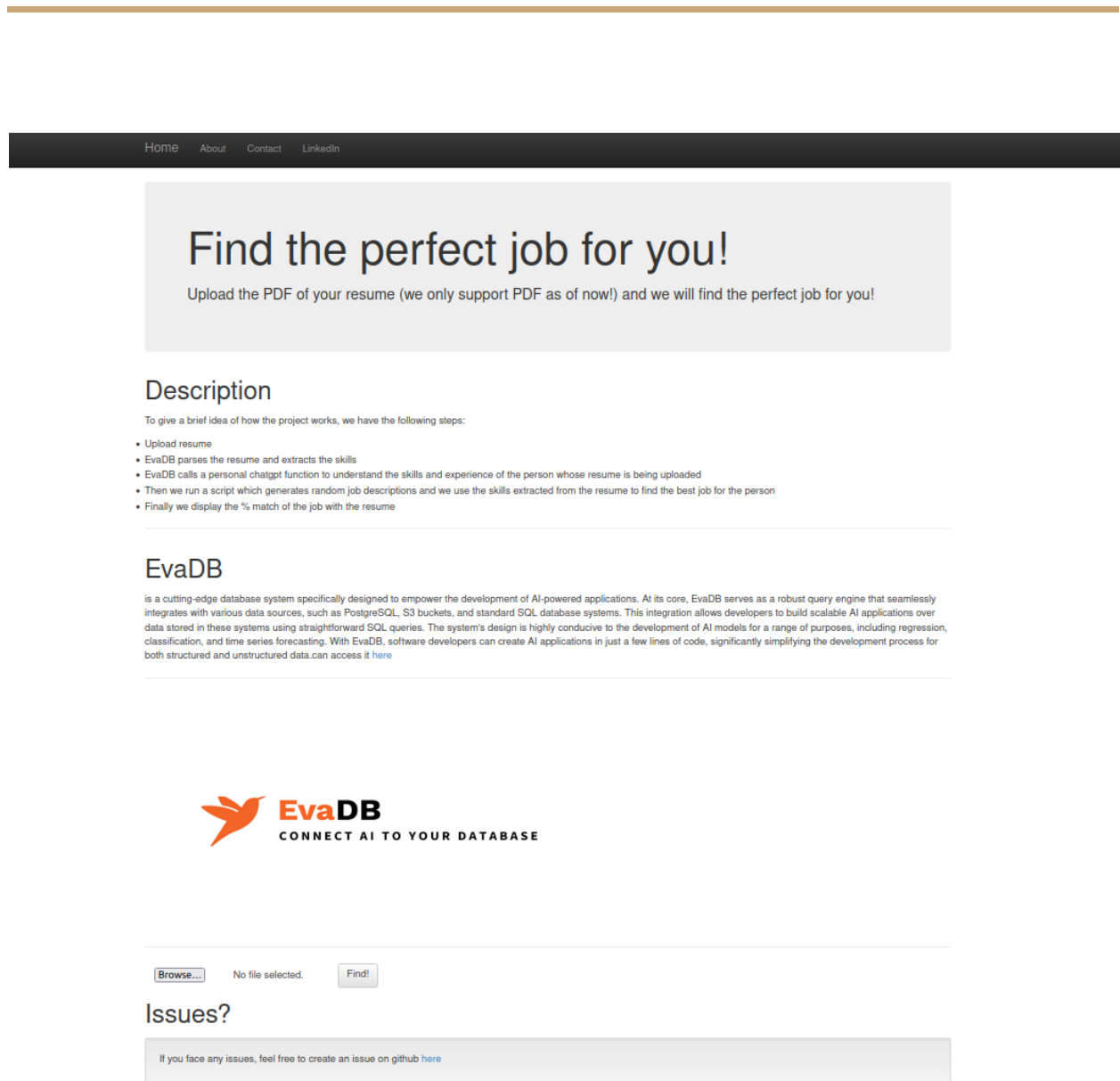
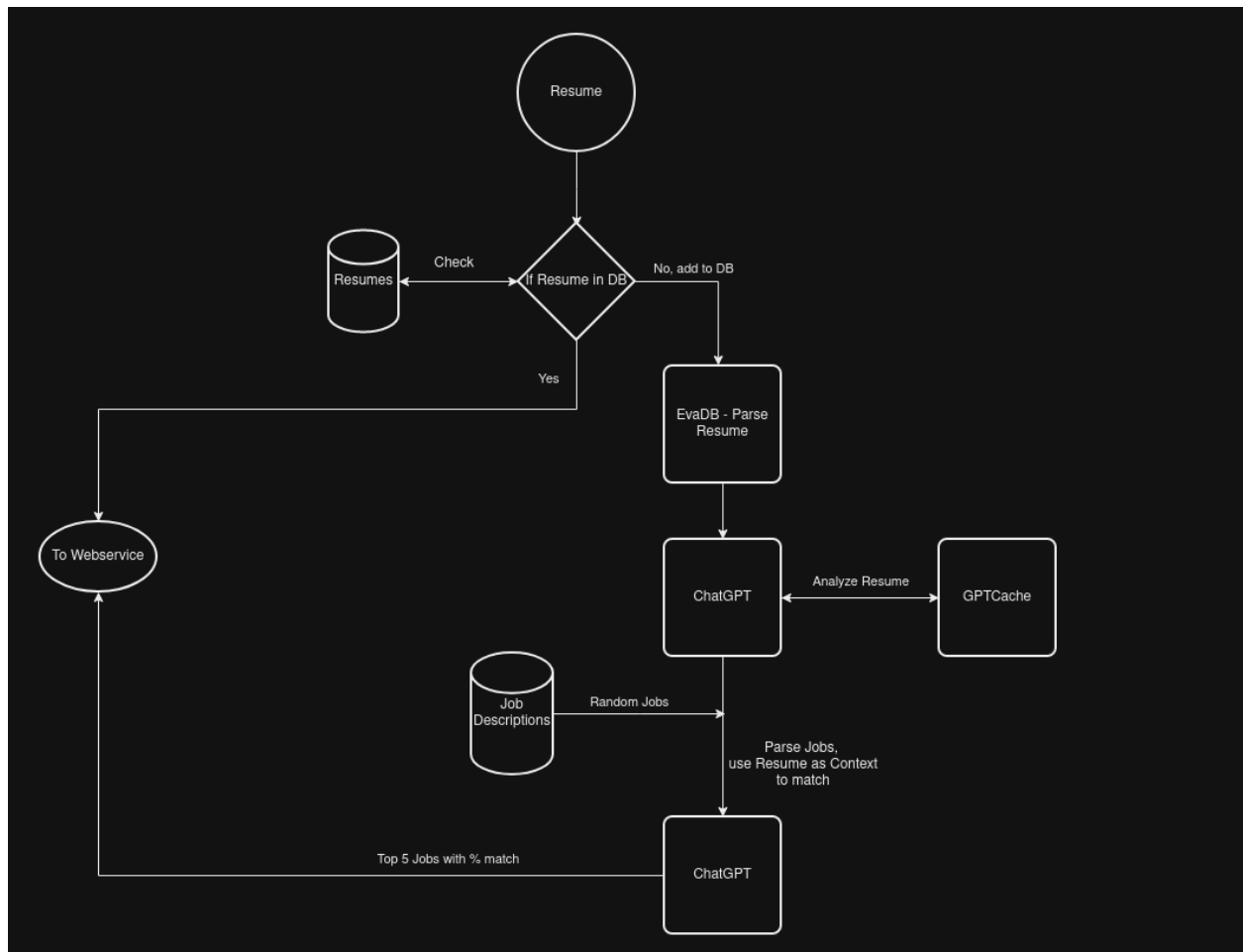


Figure 1: EvaDB-Frontend

Backend:

The Backend logic is shown below in the Flowchart. Previously, I was always passing the entire resume to chatgpt and asking it to summarize it.



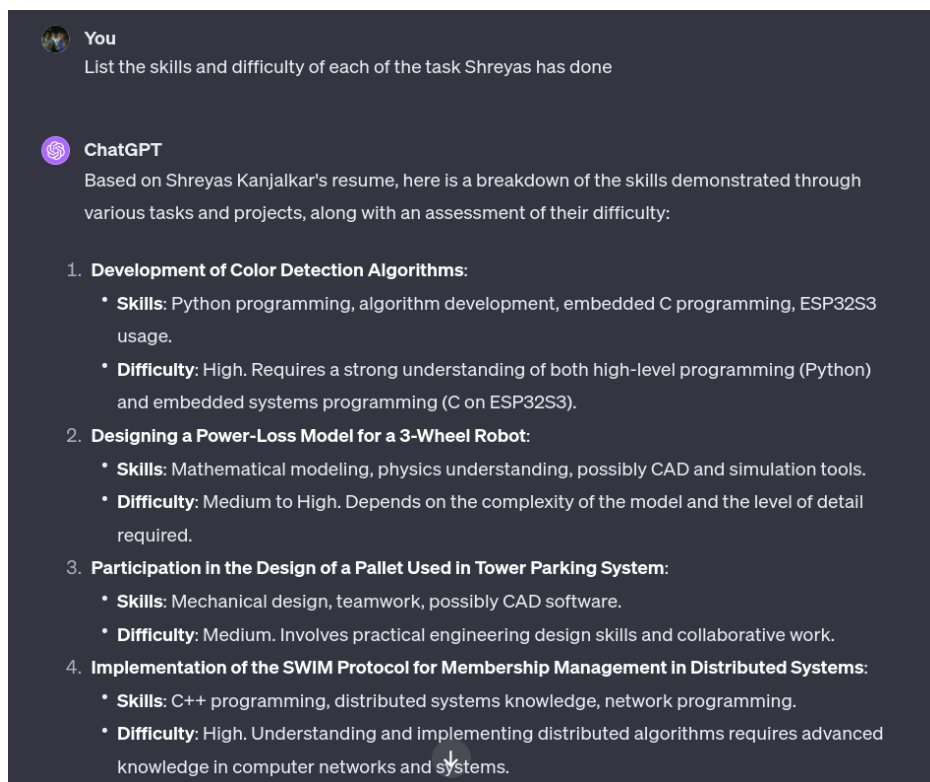
BackEnd Flowchart

The backend logic is shown in the flowchart, First the resume sent by the webservice is accepted by another javascript which is awaiting response using Restful design. The javascript spawns a python thread (we deal with multi threading here in order to deal with multiple people using the application at once). The python process takes the pdf, and checks if the resume exists in the database (this part is work in progress, hope to get it done soon). If it exists, then we just return the answer stored in the database. The database can be anything (SQLite or Postgres or Cassandra).

If the Resume doesn't exist, then we add it to the database. I've done some optimization from part 1 to improve the efficiency of the application. Now, in optimization I've done

some smart tricks to remove unnecessary information such as the Name or where the person stays to reduce the cost of tokens sent to openai. Another optimization I have done is storing the answer returned by chatgpt for the resume in gptcache. This optimization reduces the tokens sent to chatgpt (I am still unsure about this, have to look deeper into the implementation of gptcache, but I was getting responses faster). Another optimization I've done is prompt engineering, by sending over 20 different prompts to find out the best answer I have gotten consistently for resume parsing and matching with the job descriptions. The best prompt was to ask chatgpt itself to rank the difficulties of tasks done by the person on certain projects.

For example, consider the scenario where person A has worked on creating a distributed system fault tolerance logic for 10000 servers, and the same person A has also built an application which does webscraping from some website. Clearly, the first task is several times more complicated than the second task. This prompting, allowed chatpgt to give % weightage on the skillset of the person for each of the task and experience they have giving consistent results. I got this inspiration from playing around in chatgpt while updating my Resume, and I just asked it to rate my skill set and task difficulties. Here's what I got:



You
List the skills and difficulty of each of the task Shreyas has done

ChatGPT
Based on Shreyas Kanjalkar's resume, here is a breakdown of the skills demonstrated through various tasks and projects, along with an assessment of their difficulty:

- 1. Development of Color Detection Algorithms:**
 - **Skills:** Python programming, algorithm development, embedded C programming, ESP32S3 usage.
 - **Difficulty:** High. Requires a strong understanding of both high-level programming (Python) and embedded systems programming (C on ESP32S3).
- 2. Designing a Power-Loss Model for a 3-Wheel Robot:**
 - **Skills:** Mathematical modeling, physics understanding, possibly CAD and simulation tools.
 - **Difficulty:** Medium to High. Depends on the complexity of the model and the level of detail required.
- 3. Participation in the Design of a Pallet Used in Tower Parking System:**
 - **Skills:** Mechanical design, teamwork, possibly CAD software.
 - **Difficulty:** Medium. Involves practical engineering design skills and collaborative work.
- 4. Implementation of the SWIM Protocol for Membership Management in Distributed Systems:**
 - **Skills:** C++ programming, distributed systems knowledge, network programming.
 - **Difficulty:** High. Understanding and implementing distributed algorithms requires advanced knowledge in computer networks and systems.

-
5. **Development of Face Swap Technology:**
 - **Skills:** Python programming, OpenCV (Computer Vision), AWS EC2 deployment.
 - **Difficulty:** Medium to High. Combines software development with computer vision techniques and cloud deployment.
 6. **Programming Low-Latency Synchronization for Drones:**
 - **Skills:** Drone programming, synchronization algorithms, possibly real-time systems.
 - **Difficulty:** High. Requires knowledge of real-time systems and the ability to implement efficient synchronization mechanisms.
 7. **Implementing Collision-Free Movement Algorithm for a Quadrotor Swarm:**
 - **Skills:** Algorithm design, drone programming, understanding of kinematics and control systems.
 - **Difficulty:** Very High. Involves complex algorithmic work and understanding of dynamics and control systems for multiple interacting agents.
 8. **Data Analysis on Lichess Database:**
 - **Skills:** Database management, SQL, user interface design, data analysis.
 - **Difficulty:** Medium. While data analysis and database management are core skills, the complexity can vary based on the project's scope.

In summary, Shreyas has demonstrated a range of skills across different fields, with tasks ranging from medium to very high in difficulty. The variety and complexity of these tasks reflect a strong technical background and adaptability in various domains of computer science and engineering.

Using this as my starting point, I just had to fine tune the prompts till I got a good enough response from chatgpt consistently.

Database:

This part is still WIP, as of right now the resumes are just stored in the filesystem and deleted after a while. I plan to store the resumes in a database(maybe cassandra or mongodb) to store the resumes and their answers.

Challenges Faced:

I think without a doubt, the biggest challenge I faced was learning EvaDB. At first glance, using it seems simple, but there are nuances to it that one has to carefully give time to and debug it one step at a time. I think the slack community of EvaDB is great (special shoutouts to Andy for always helping me with the dumbest questions). I tried out bunch of other AI apps to help me do a proper resume matching with job descriptions using auto label, or using some external resume parser but none of them were as consistent as chatgpt so I stuck with it.

This project also allowed me to learn about how to understand a huge project with over thousands of lines of code, how it is implemented and what goes under the hood in such projects. I plan to contribute to some of the issues and bugs I found during this project on EvaDB during the winter break.

Lessons Learnt:

The biggest lesson I think, was learning to debug and github commits using a huge project. It's always a huge recursive issue where one line of code can cause your entire project to break down. Professor Joy always said to reduce the test case and minimize the scope. Although this approach is fine, I think I learnt another approach which is committing to github whenever I made a new change (reasonable enough such as a new feature or some refactoring), and if things broke down using **git diff** to figure out what went wrong. I truly appreciate the value of github using this.