Web Scraping, Frontend, and Backend Overview
Web Scraping (Python)

I utilized Python for web scraping, employing Beautiful Soup for HTML parsing and Pandas for data manipulation. The target website was Amazon.in, from which I extracted four attributes for each book: book name, picture link, publication year, and category. Using Pandas, I organized this information into a data frame, exporting the final result as OutputFile.csv in CSV format.

Frontend (TypeScript with Next.js)

In frontend development, I opted for TypeScript and Next.js. I crafted a Home page and a Form Page using Next.js. The Form Page includes input fields for gathering client data. This data is transmitted to the backend, which responds with JSON data utilized by the Result.tsx component. Additionally, I implemented state management using useContext.

Backend (Node.js with TypeScript and MongoDB)

For the backend, I integrated MongoDB as the database and utilized Node.js with TypeScript. All data from outputFile.csv is stored in a MongoDB collection, with the schema specified in the booksSchema.ts file. Before adding data, the backend checks if the collection is empty. If so, all entries from the CSV file are added to the collection using the addBooksintoBD.ts file. If not, the process is skipped. Upon receiving client information (name, category, age, and year), the backend triggers the compareVectors function. This function uses a vector difference technique, considering the year of publishing, age, and name as coordinates. The output is an array of documents indicating similarity. The backend then selects an index from this array by applying the modulus between the array's length and the length of the client-provided name. The output is retrieved and sent back to the client.

Note: The backend response may sometimes experience delays, so please be patient during the loading process.