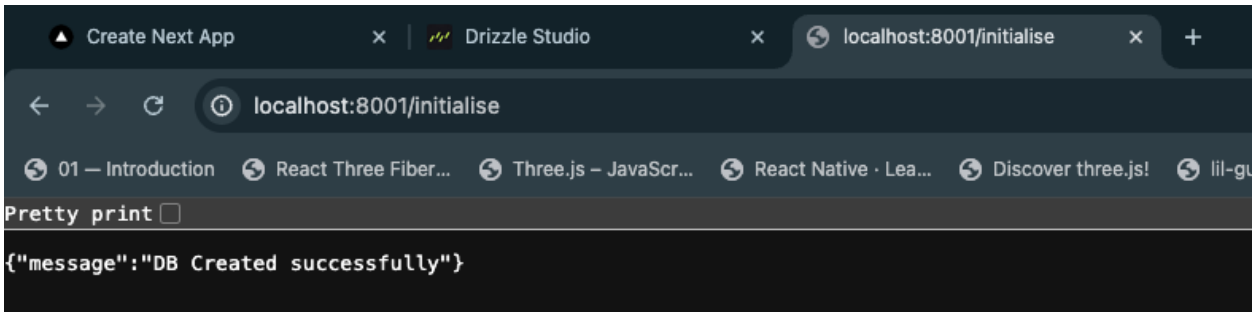
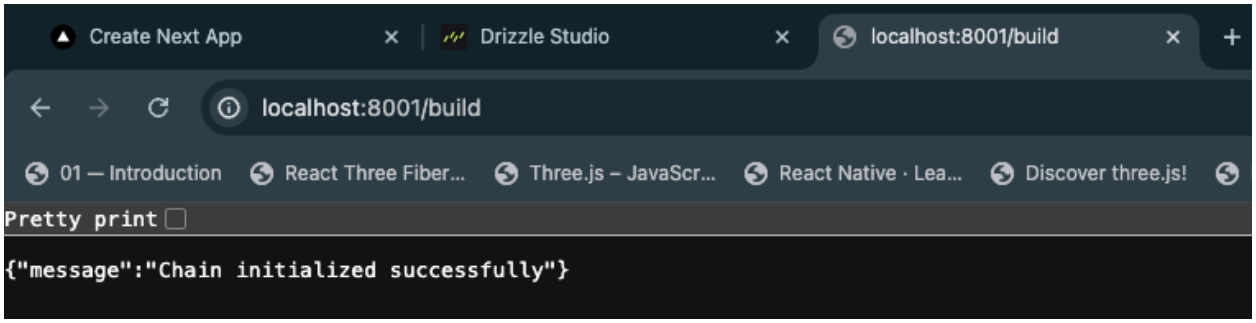
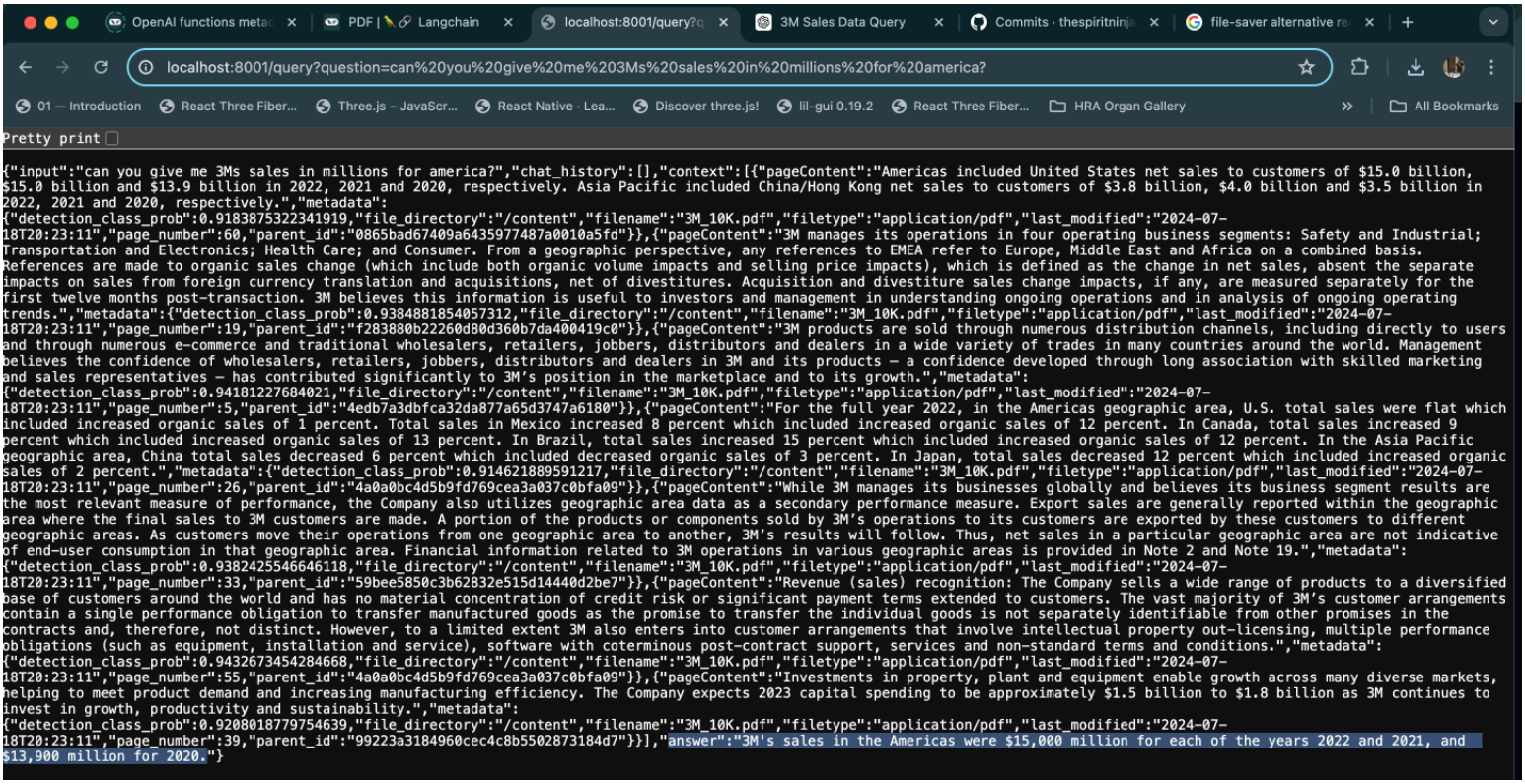
:Example Usage and Journey Document:

Backend Development:

- For the application to interact with I created a RESTful Backend service on NodeJS and used chromadb to store the pdf data as chunks for context info.
- The query was being processed correctly and I did get responses from the RAG-Model that I had designed primitively for the usecase. [PFB Screenshots]



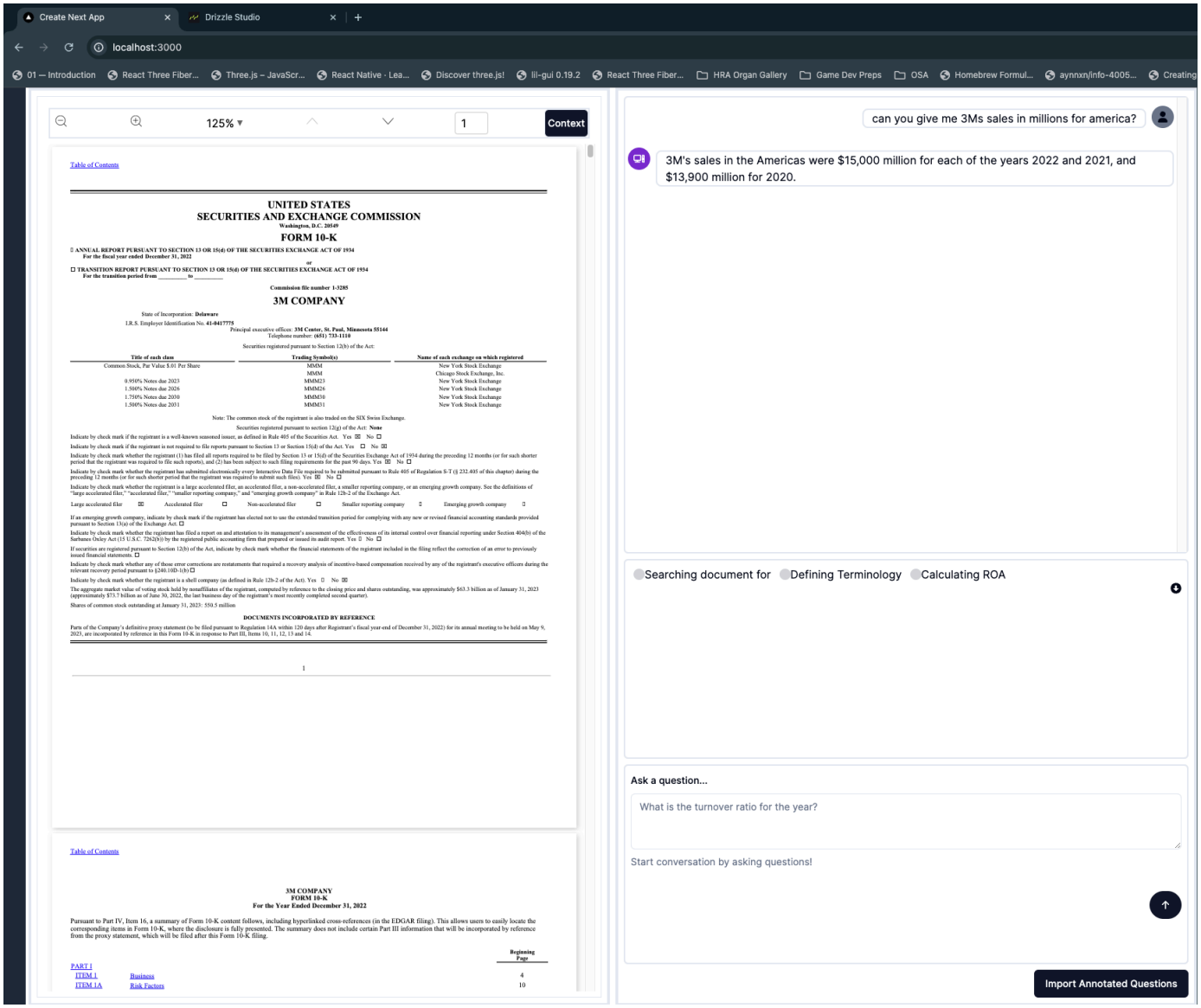




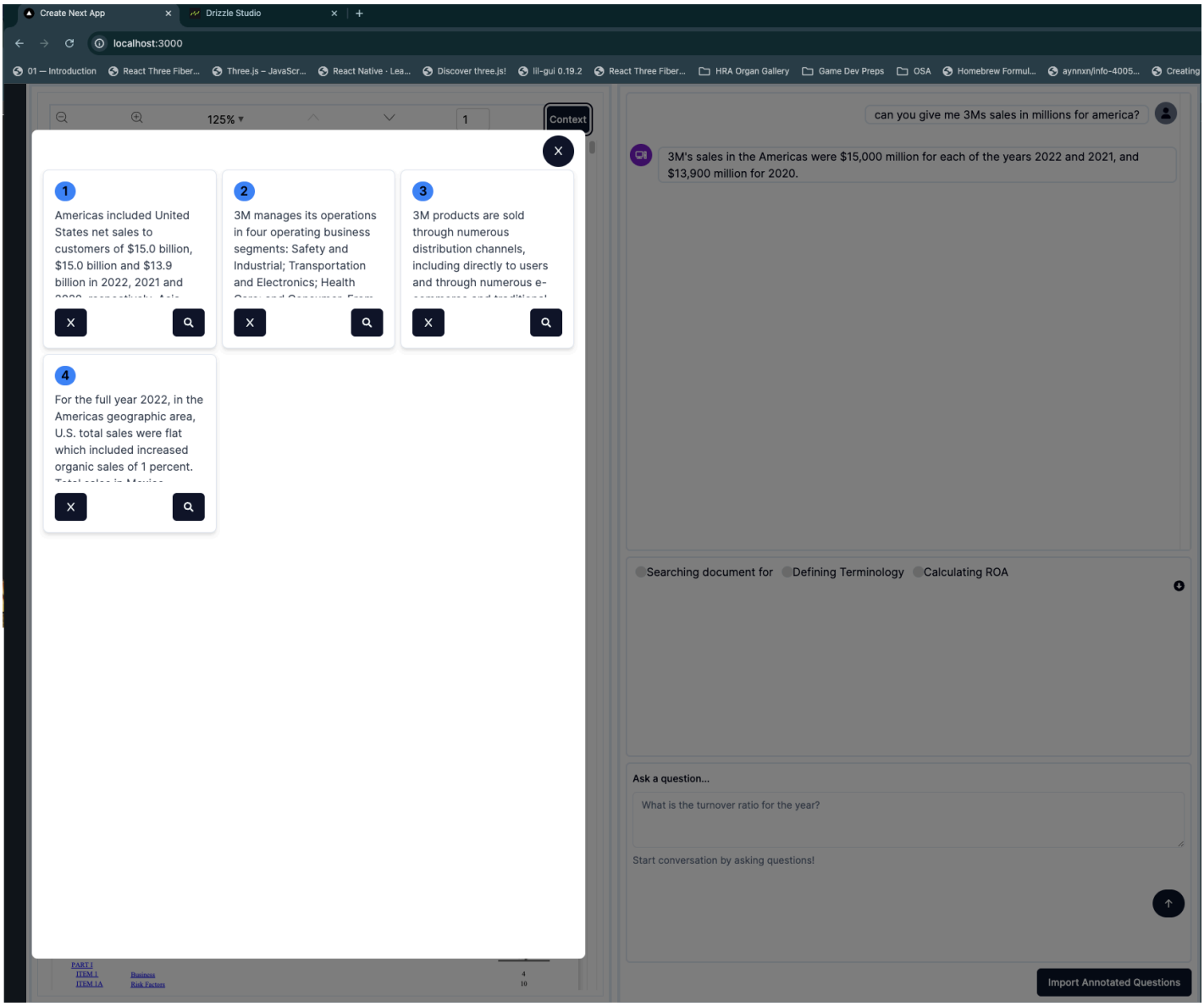
- However, for some reason I found out that the chroma db was not storing the page context coordinates [i.e numericals] in the vectorDB
- I needed the page coordinates as highlighting the context was a big use case from my end that I wanted to have implemented.
- I did not want to spend more than a day rectifying the issue as I had to respect the 48HR timeline. Also, the query response data structure is bound to change and has to be adapted to your model response in the future.
- I decided to proceed with a sample JSON query response data [attached with the email] to build and display on the application.
- Please feel free to go through the backend server code, I'd appreciate your reviews on it as well.

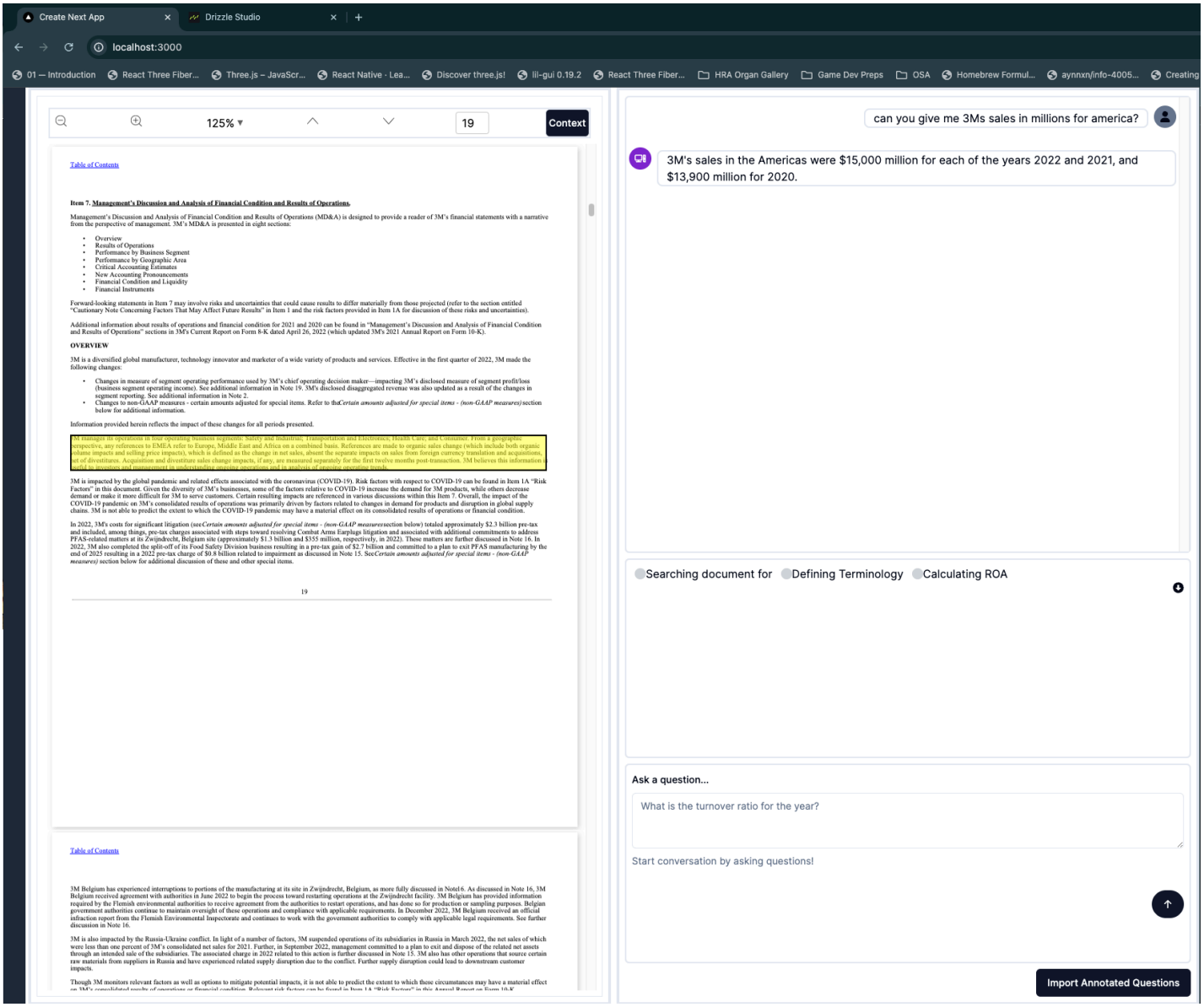
Frontend Development:

- I've built the frontend using NextJS and have integrated a Postgres DB using Drizzle ORM to store and retrieve questions data.
- This is how the frontend looks like:



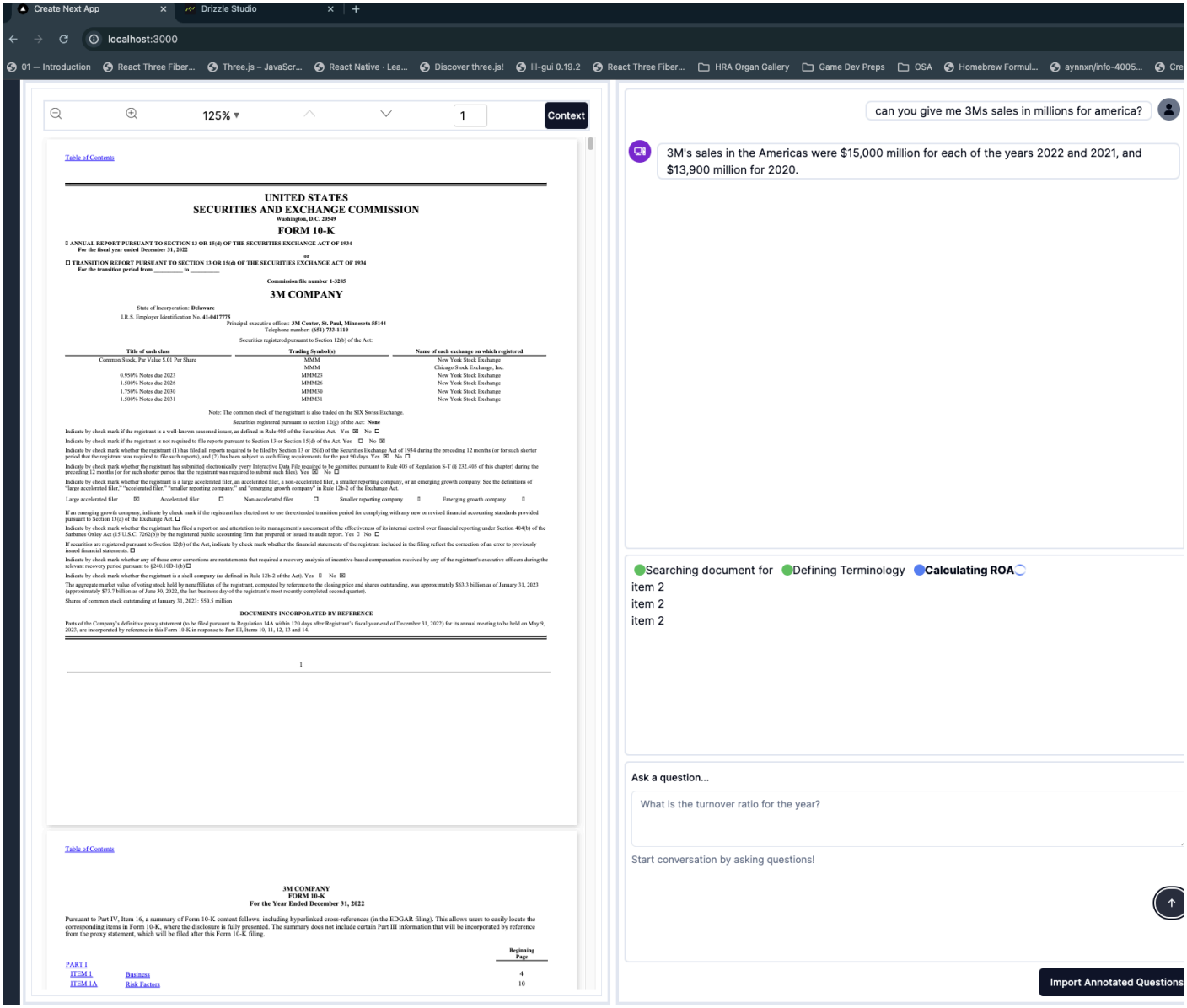
- Please note that I've used a sample response(static) data which I expected to work with and the frontend is not connected to the backend as I couldn't get the coordinates details returned from the server.
- The left part of the screen has a PDF Viewer and is interactable, also the context button on the top right is supposed to be populated/updated w.r.t every response you receive for the question asked.
- Upon clicking the context button a modal is shown and when you click on the search button it takes you to the page and shows the context highlighted as shown below. The highlighted context is only possible through the coordinates.



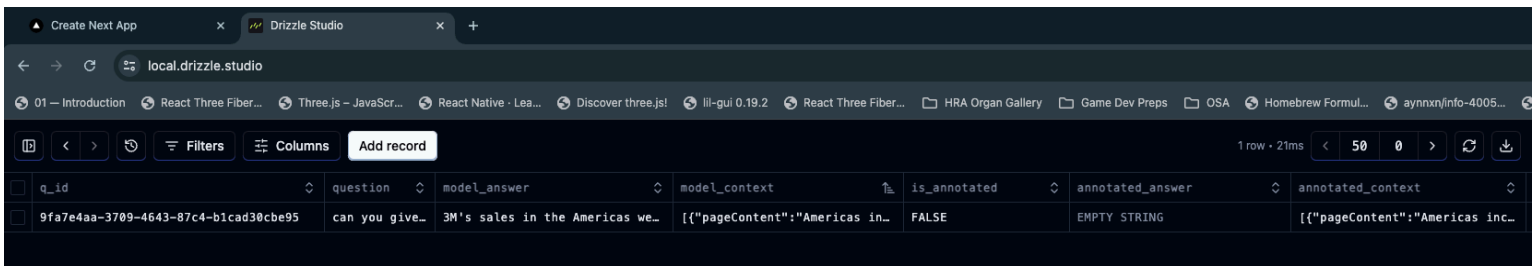


[I clicked search button on the second card]

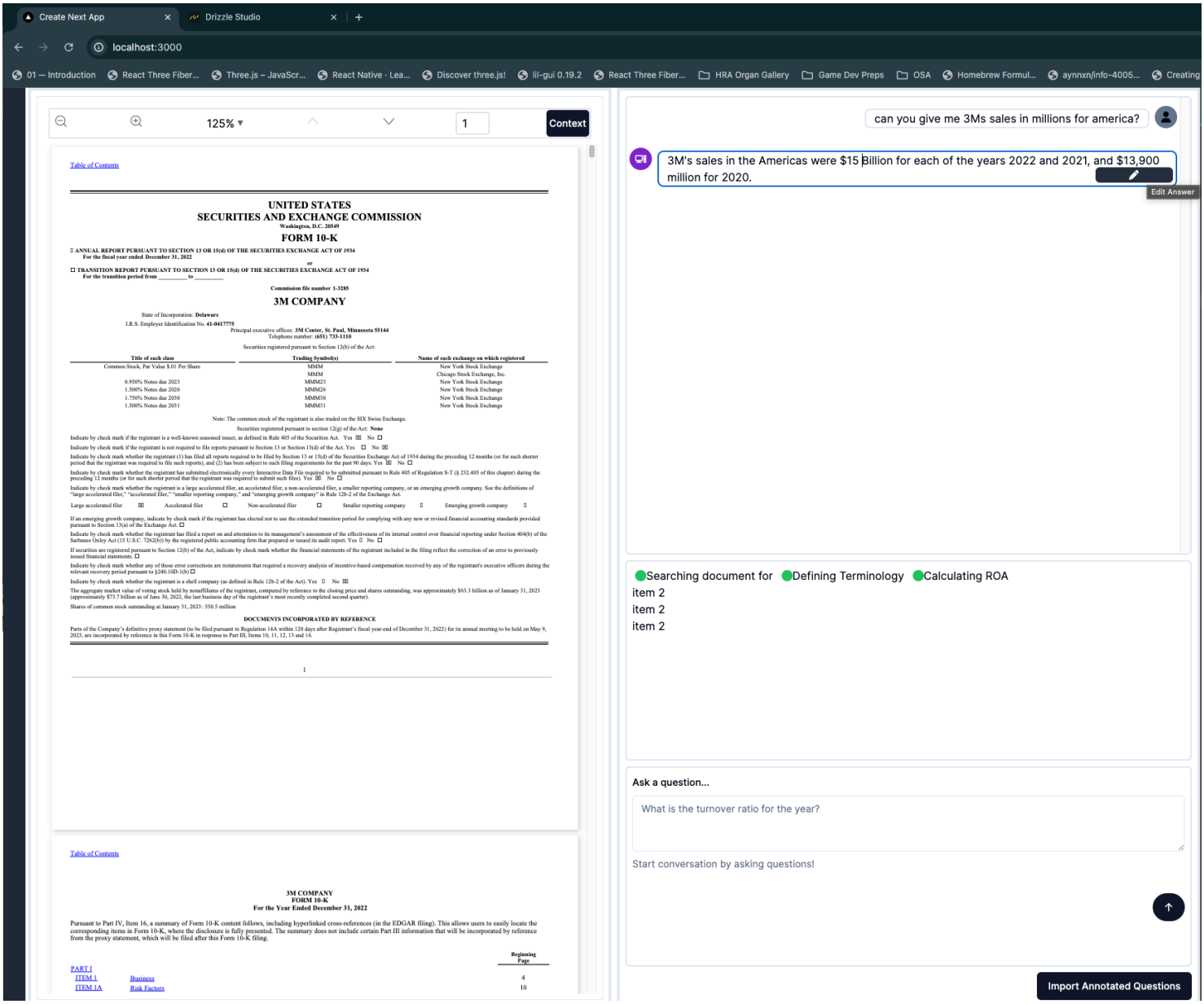
- The right hand panel is divided into 3 different sections:
  1] The first is the chat-window which is static by virtue of the sample data, this is supposed to be added as per the response keeps updated. For the assessment purpose I've only kept one entry.
  2] The second section has the loading timeline, for now it's only a place holder and is ready to be populated w.r.t the responses that I get from your model.



3] Lastly we have the text input which will query the question to the backend, the submit button for now only stores the data in the postgres db with a unique q_id, the response from the model and its context and defaults annotated to false. [PFB Screenshot]



- As mentioned in the assessment requirement we also wanted a feature for the user to annotate the response and be able to retrieve the annotated responses as well
- The response section in the chat window is editable and user can interact with it. Upon editing the answer the user can click on the button and it saves the response in the postgres DB. You can also have the context removed from the context modal before hitting save and it will retain the info.



- Finally to import the annotated response, you can click the Import button on bottom right and it will get all the annotated response in the db. This can be later reconfigured per our requirements and we can add a modal which asks for all the dates, user and other filters.

- I've also uploaded the Excel file with the email.

Finishing Thoughts:

1. These were the threshold goals that I wanted to achieve in the 48Hr timeline that was set for completing the frontend. I hope you'll take this build into consideration. I thoroughly enjoyed building this interface.
2. I would love to hear back your views on the interface and hope to have an opportunity to continue working on this, as I believe there is definitely room for improvements.
3. That being said I genuinely believe this would meet your expectations as a preliminary V1 build of the interface.