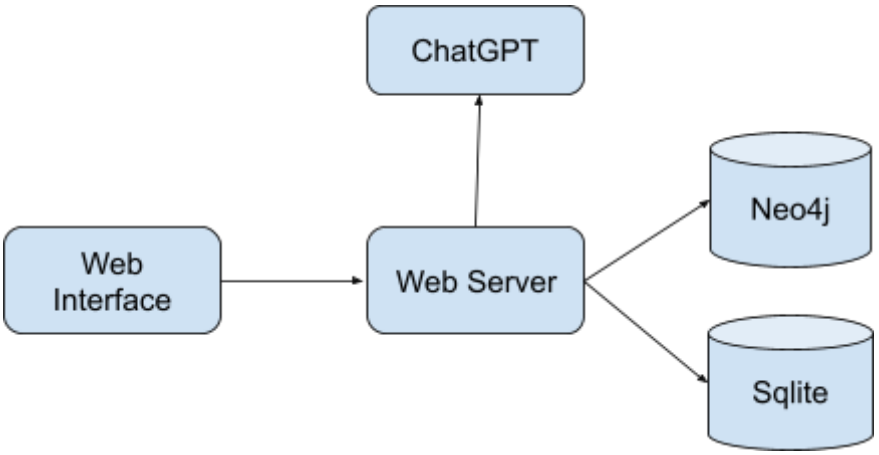
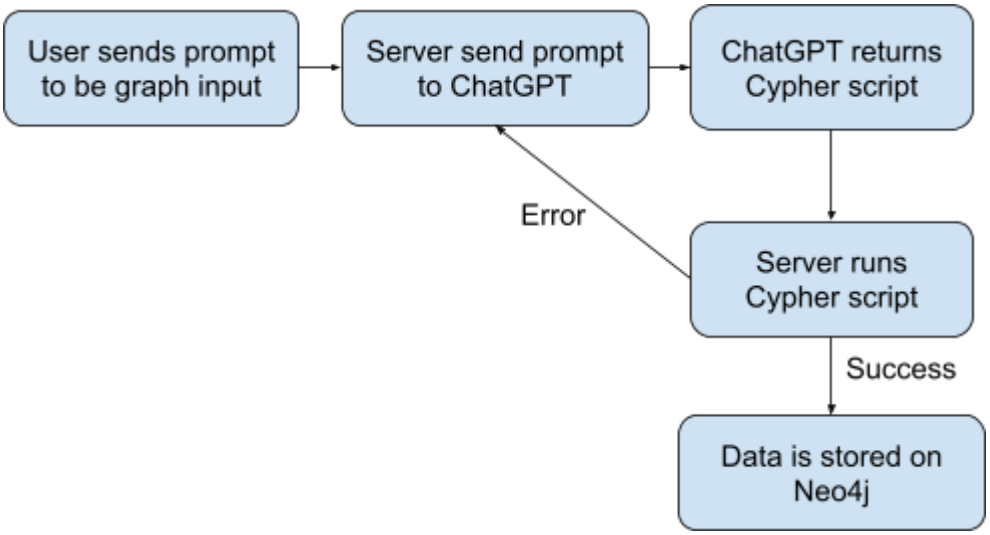
System Architecture



The system architecture for this project is simple, graph data is stored in Neo4j database and chat history data is stored on Sqlite. I am using SvelteKit for both the frontend and backend because I felt like separating the project would take more effort and because the server is quite simple. Even though the project is shared none of the server side credentials are secure because the user is only able to interact through the API.

Design Decisions



This is the process of generating nodes and relationships based on the prompt. All of the Cypher queries are generated by ChatGPT and to make sure that errors are minimum, this process is able to retry itself until a maximum of 5 times. This process is also auto improving; ChatGPT is given full context of the available schema and adjusts the query to prevent duplicates, ensuring the schema is optimised.

Design Limitations & Issues

Current design limitations include:

- Scalability: the current process of passing the whole schema context doesn't scale well into thousands of nodes, a way to improve this is to create embeddings of the schema.
- Current design only support one chat instance to make it simpler, this is able to change with some slight adjustments
- I was not able to find a graph layout algorithm that makes the graph readable, currently there is a lot of randomness on the graph placement.

Assumptions

- Assume that nodes with the same values should be merged together.
- The optimization for graph databases would be to merge similar nodes and to not reuse relationship types.
- Assume that the web interface is the only client.