Company Graph v2.0

## **Functional Requirement**

* Relationship Query

1. Query and display extracted relationship
   1. support partial visualization of competition relationships:
      1. Users can choose or search a company and view its competitors
      2. Users can choose how many nodes or companies to visualize
2. Categorize different types of competition relationships
   1. Each type of competition relationship should contain different label

## **Non-Functional Requirement (list use cases)**

1. High availability
   1. The web app should provide service 99% of the time(service level agreement)
2. QPS
3. Latency
   1. UI
   2. Backend
4. Reasonable response time (latency)
   1. The web app should present data and visualization within 500ms on average
5. Storage independence(Data Persistence Layer)
   1. The web app itself can use different types of storage solutions: Graph DB, Relational DB, and other NoSQL databases

## **Scale**

Max 100 queries per second (QPS) for retrieving company relationship

Storage: 750,000 filings, estimated 750k\*20=1.5MM nodes in the graph, similar scale on the number of edges.

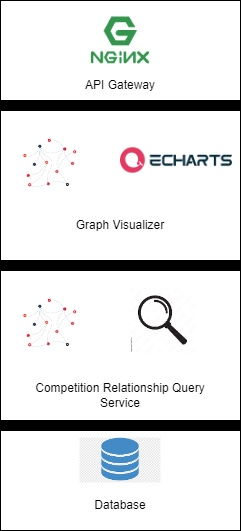
## **Web API**

The design of web API follows the RESTful standard.

GET /competitors/<company\_name>

GET /competition-graph?center-node-id={node\_id: int}&max-layers={max\_expand\_layers: int}

## **Architecture**

****

## **Key UX Design**

Check out the UI

## **Frontend Project Structure**

TODO: Describe how modules are organized

## **Backend Project Structure**

TODO: Describe how modules are organized

## **DB Design**

### **Graph Files**

DB Choice: 3 Files on Hard Drive

Schema:

* train2id: (e1, e2, rel)
* relation2id: (relation\_name, id)
* entity2id: (entity, id)

## **Assumptions/Important tradeoff**

Trade off 1:

Proposal:

Option 1:

Pros:

Cons: