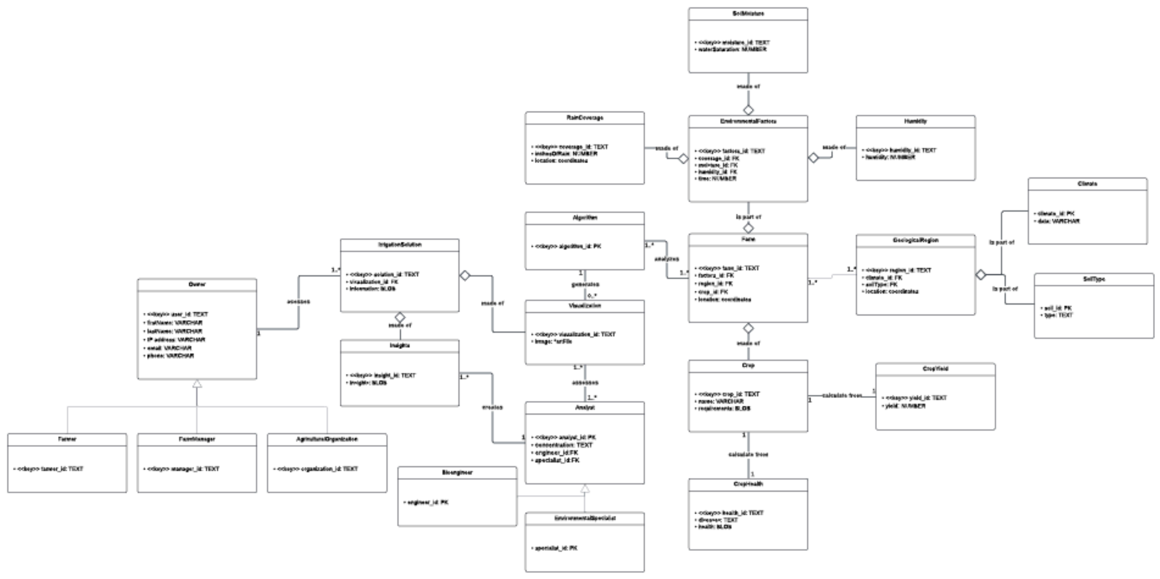

1. Functionalities selected to be used as an in-memory key-value storage: Will store the farms and their corresponding Irrigation solutions. All the farm information minus the extraneous ids originally in SQL will be implemented in the mongo database and used in the web application. This includes:

```
        Farm_id,
        Solution {
                insights, visualization {
                        image, analyst {
                                analyst_id, analyist_type, concentration
                                }
                        },
                        information
                },
        user {
                user_id, firstName, lastName, IP_address, email, phone, userType
        },
        factors {
                coverage {
                inchesOfRain, location
                },
                waterSaturation, humidity, time
                }
        region_id {
                        {
```

```
                region_id, soilType
                } climate {
                climateType, data, location
                }
            }
        Crop {
            Name, Requirements, health {
                diseases, health
                }
            yield,
            location
        }
```

2. Redis data structures

To implement the key and farm + Irrigation solution, I will use a Redis string key:value with key "farm_ID", and the value as a json document with the farm information on it as well as the respective irrigation solution. I will be using the caching system with redis to have a quicker retrieval system and use a hash (key: value) to store farms.

3. Redis commands:

I will return a hash set of farms with the same climate type (Savanna Climate) as well as subsequent irrigation solutions. Therefore I need the following redis commands:

- hSet: puts farms into hash for the cache (puts farms in cache).
  - key: `` `farm:${climateType}:${farm.id.$oid}` `` *(the farm_id)*
  - Value: `Object.entries(farm).flatMap(([key, value]) => [key, JSON.stringify(value)])`
    - For each farm entry, it will flatten the set and stringify it to json.
- rPush: pushes farm (key: value) pairs into the hash.
- set: sets the hash cache item with its key and sets an expiration time for the cache (for this application, it is 60 seconds)
- lRange: gets all farms within the range, (from 0, -1, in other words, all the farms)
- hGetAll: gets all the farm keys
- FlushAll: cleans cache after expiration time (60 seconds in this case)