Coding Walkthrough

1. Give overview of provided code
   1. Connection setup
   2. Method design
   3. Each line they need to write has a comment describing it
2. Describe how this could be generalized to any deterministic piece of work that gets repeated
3. Start code walkthrough
   1. Have them change # Your Name Here to their name
   2. ‘Try to get the query result from redis’
      1. string jsonString = redisDB.StringGet(query);
      2. Basic way to get data from redis.
      3. Similar methods for other data types.
   3. ‘Check if a result was returned (the query IS in the cache)’
      1. String.IsNullOrEmpty(jsonString)
      2. Basic string method. If the key does not exist in redis, redis returns Null
   4. ‘Parse JSON string into JObject and return’
      1. return JsonConvert.DeserializeObject<dynamic>(jsonString);
      2. Converting the string into the desired type
      3. Must be able to do this if you want to cache data
   5. ‘Issue graphQL query’
      1. var graphQLResponse = graphQL.PostQueryAsync(query).GetAwaiter().GetResult();
      2. Standard way to issue graphQL request
      3. Slow because uses HTTP
      4. Redis lets us avoid this
   6. ‘Convert graphQL response to JSON string’
      1. jsonString = graphQLResponse.Data.ToString();
      2. Extract jsonString from the response
      3. Format that can be stored in redis
   7. ‘Conditionally store result in redis cache’
      1. jsonString.Length \* sizeof(char) < 1024 \* 1024 \* 10
      2. Could be any condition about the data you want to store
      3. Here doing size because we don’t want large queries cached
      4. Mention keys and values limited to 512MB
   8. ‘Add to cache’
      1. redisDB.StringSetAsync(query, jsonString);
      2. Way to add key-value pair to redis
      3. Key must be string
      4. Similar methods for other data types
      5. Async and synchronous methods exist
   9. ‘return JObject’
      1. return graphQLResponse.Data;
      2. This gets the JObject from the response
      3. Default format for this data
4. Take questions
   1. Where will the bottleneck be?
      1. If on same network, then the network
      2. If on same device/host, Redis processing
   2. How easy is it to configure?
      1. Very, the config is reasonably documented, and the file has lots of info
   3. What makes it good for caching?
      1. In memory, so very fast
      2. Key-value store with hashing to find values, so O(1) lookup (I THINK)
      3. Supports multiple eviction strategies, we want allkeys-lru
   4. What is the best way to store data?
      1. As far as I can tell, strings
      2. Look into storing as binary data cast to a string