**Question 5 – Operational aspects**

1. *Explain GraphQL schemas and show how GraphQL can be used.*

**GraphQL Schemas & Usage**

Scalar types

* Define the leaves of your schema and include:
  + **Int, Float, String, Boolean, ID** (as unique string)

Type definitions

* Declare object shapes:
  + **!** marks non-nullable fields
  + Lists (**[…]**) represents arrays

Entry points

* **Query** for reads, **Mutation** for writes

**Using GraphQL in Apollo Server**

1. Load SDL into Apollo:  
   const typeDefs = fs.readFileSync("./schema.graphql", "utf8");  
   const server = new ApolloServer({ typeDefs, resolvers, context });
2. **Resolvers** map schema fields to data operations, using three main args:
   * **parent** (for nested fields)
   * **args** (the query/mutation inputs)
   * **context** (shared services, e.g. database clients)
3. **Sample query** asking for only needed fields
4. **Sample mutation** to add a blog
5. *Compare the GraphQL schemas of assignment 2 and 3. Demonstrate your MongoDB Cloud deployment. Include triggers.*

**Comparing the GraphQL Schemas of Assignment 2 & 3**

**Document (Assignment 2)**

Backing Store

* Data stored & fetched in **MongoDB Atlas Cluster** through GraphQL API
* **MongoDB Realm’s managed GraphQL** **service** to expose an endpoint

SDL Style

* Plain SDL loaded into Apollo Server (no schema directives)

Triggers

* For the **orders** collection
* Defined in **Atlas UI**
* Checks **changeEvent.operationType** to distinguish inserts & updates
* **changeEvent.fullDocument**: after insert/update, realm gives you the entire document as it looks after the write
* **changeEvent.updateDescription**: MongoDB provides breakdown of exactly **which fields changed,** and  **which fields were removed**.
* All entries go into a separate **orderLog** collection

**Graph (Assignment 3)**

Backing Store

* Neo4j graph database

SDL Style

* SDL with schema-directives: **@node, @relationship, @cypher**
  + @node & @id auto-generate CRUD types & resolvers
  + @relationship wires GraphQL fields to Neo4j relationships
  + @cypher lats you create custom queries for advanced cases

Apollo + Neo4jGraphQL Bootstrap

* **getSchema()** reads the SDL with the directives and auto-wires all Cypher under the hood

Sample Query

* will invoke the created Cypher in the SDL, retuning each matching **Book** node and its connected **Author** nodes