SER421 Lab **GraphQL APIs**

Activity 1 (30 points): Construct GraphQL Queries

SpaceX is a commercial rocket and spacecraft manufacturer. Launcher, and satellite communications company. They provide a public GraphQL API that you can query using a GraphIQL-like Explorer toolhere: https://studio.apollographql.com/public/spacex-l4uc6p/variant/main/explorer

Note that when you go into the Explorer the first time you will see an Example Query:

```
query ExampleQuery {
  company {
    ceo
  }
  roadster {
    apoapsis_au
  }
}
```

Which you can run to get the result by clicking on the blue button in the upper right. I like this API and Explorer tool because the Documentation on the left is easy to follow. However, you can also use the GraphiQL electron app and the URL https://main--spacex-l4uc6p.apollographos.net/graphql.

Using the documentation and the tool, write GraphQL queries that generate the EXACT responses shown below. For your solution, copy and paste just your Operation (the center pane of the Explorer, the Query), into your solution. Please name your query "QueryX" where X corresponds to the number below. For example, "Query1" would replace "ExampleQuery" above. I recommend as you work that you use the "+" sign to create separate tabs for each query until you are done.

1. (8) Response:

```
{
  "data": {
    "company": {
      "name": "SpaceX",
      "founded": 2002,
      "coo": "Gwynne Shotwell",
      "cto": "Elon Musk",
      "employees": 9500
    }
  }
}
```

2. (10) Response:

```
{
  "data": {
    "launchesPast": [
      {
        "launch_date_local": "2006-03-25T10:30:00+12:00",
        "links": {
          "article_link": "https://www.space.com/2196-spacex-inaugural-falcon-1-rocket-lost-launch.html"
        },
        "rocket": {
          "rocket_name": "Falcon 1",
          "rocket_type": "rocket"
        }
      },
      {
        "launch_date_local": "2007-03-21T13:10:00+12:00",
        "links": {
          "article_link": "https://www.space.com/3590-spacex-falcon-1-rocket-fails-reach-orbit.html"
        },
        "rocket": {
          "rocket_name": "Falcon 1",
          "rocket_type": "rocket"
        }
      }
    ]
  }
}
```

3. (12) Response:

```
{
  "data": {
    "roadster": {
      "details": "Elon Musk's Tesla Roadster is an electric sports car that served as the dummy payload for the
February 2018 Falcon Heavy test flight and is now an artificial satellite of the Sun. Starman, a mannequin
dressed in a spacesuit, occupies the driver's seat. The car and rocket are products of Tesla and SpaceX. This
2008-model Roadster was previously used by Musk for commuting, and is the only consumer car sent into space.",
      "launch_date_utc": "2018-02-06T20:45:00.000Z"
    },
    "rockets": [
      {
        "name": "Falcon 1",
        "boosters": 0,
        "company": "SpaceX",
        "landing_legs": {
          "material": null
        }
      },
      {
        "name": "Falcon 9",
        "boosters": 0,
        "company": "SpaceX",
        "landing_legs": {
          "material": "carbon fiber"
        }
      },
      {
        "name": "Falcon Heavy",
        "boosters": 2,
        "company": "SpaceX",
        "landing_legs": {
          "material": "carbon fiber"
        }
      }
    ]
  }
}
```

Submission for Part 1: Please put your queries in a file named LabGraphQL_Act1_<azurite>.txt. Please use a text file as we intend to simply cut-and-paste your queries to test them.

Activity 2 (40 points): Extend a GraphQL API in Spring

In the course public repo there is a GraphQL example for the Booktown example we have been using for various server-side concepts (apps, APIs). *Please do the following before starting this coding exercise*:

1. Read the Readme.md in that folder
2. Read the GraphQLReadme.md in that folder
3. Build and run the example
4. Play around with the available APIs
5. COPY the folder out of a git-enabled repo and into a clean folder! (Do this TWICE, see Activity 3!)
   a) In other words, if you cloned the class repo (as you should have), do not develop your lab inside that folder.
   b) Copy the code to a clean working area for this lab, and make sure it is not under git anymore, which basically means remove the .git folder.
   c) I do recommend managing your lab code under git (local, not GitHub) always, so you can then do a "git init" and establish a new local repo to track your changes.
   d) If you really know what you are doing with git, you could do something different for b & c and manage your local repo while maintaining a commit chain off of mine. But if that sounds like gobblygook to you then don't worry about it. If it makes sense then you are a git expert and can go ahead 😊

Tasks for this activity (8 points each):

1. Add an API to get a list of Books based on an authorId
2. Add an API to get a list of Authors based on a lastName
3. Add an API to update an Author's firstName based on the Author's id. If successful return the old first name, else return null.
4. Add an API to delete a Book given it's ISBN. If successful return the ISBN otherwise return null.
5. Add an API to get a list of all Book titles written by Authors with a given first name.

Activity 3 (30 points): Convert the GraphQL API to use a JPA backend

Note: Please work from the same starting code for Activities 2 and 3. That is do NOT include your Activity 2 solution with Activity 3! Therefore, do step 5 TWICE, once for Activity 2 and again for this Activity 3!

The GraphQL example uses hardcoded dummy data as a mock for having a real database backend (see the static initialization blocks in AuthorRepository.java and BookRepository.java respectively). For Activity 3, remove the static initialization blocks in favor of a Spring JPA-enabled database. You must use the built-in H2 database, same as our example for `grocerydemojpa` in the class GitHub repo.

Full Submission Instructions:

- You should be working in a clean folder, with two subfolders *activity2* and *activity3* which have your source code dev trees for these activities respectively.
- Important! Your source code dev trees should be *clean*! That is, you should only have the gradle, configuration, and source code files, not build files when you create your submission package. *If you fail to do this your submission file may be unnecessarily large and rejected by Canvas. This will not be grounds for an appeal!*
- As always you can put a Readme.txt in the root of your dev trees to tell us anything we need to know before grading.
- Put your LabGraphQL_Act1_<azurite>.txt file in the parent folder
- Zip it all up into LabGraphQL_<azurite>.zip for submission to Canvas.

ASCII art for your submission package:

```
ser421lab (clean folder)
      |── LabGraphQL_Act1_<azurite>.txt
      |
      |── Activity2 (folder)
      |         → All Activity 2 stuff in here
      |── Activity3 (folder)
      |         → All Activity 3 stuff in here
```