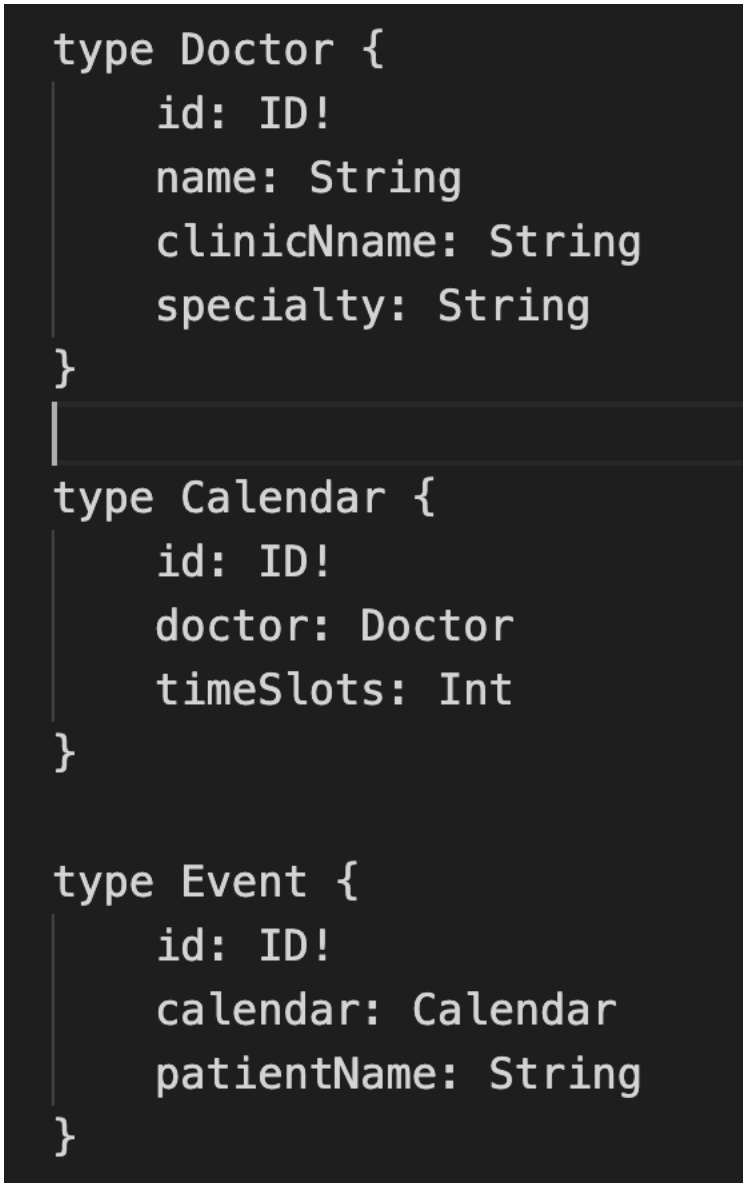
A2.2 Reflection

What were some of the alternative schema and query design options you considered? Why did you choose the selected options?

I considered the old schema as below.

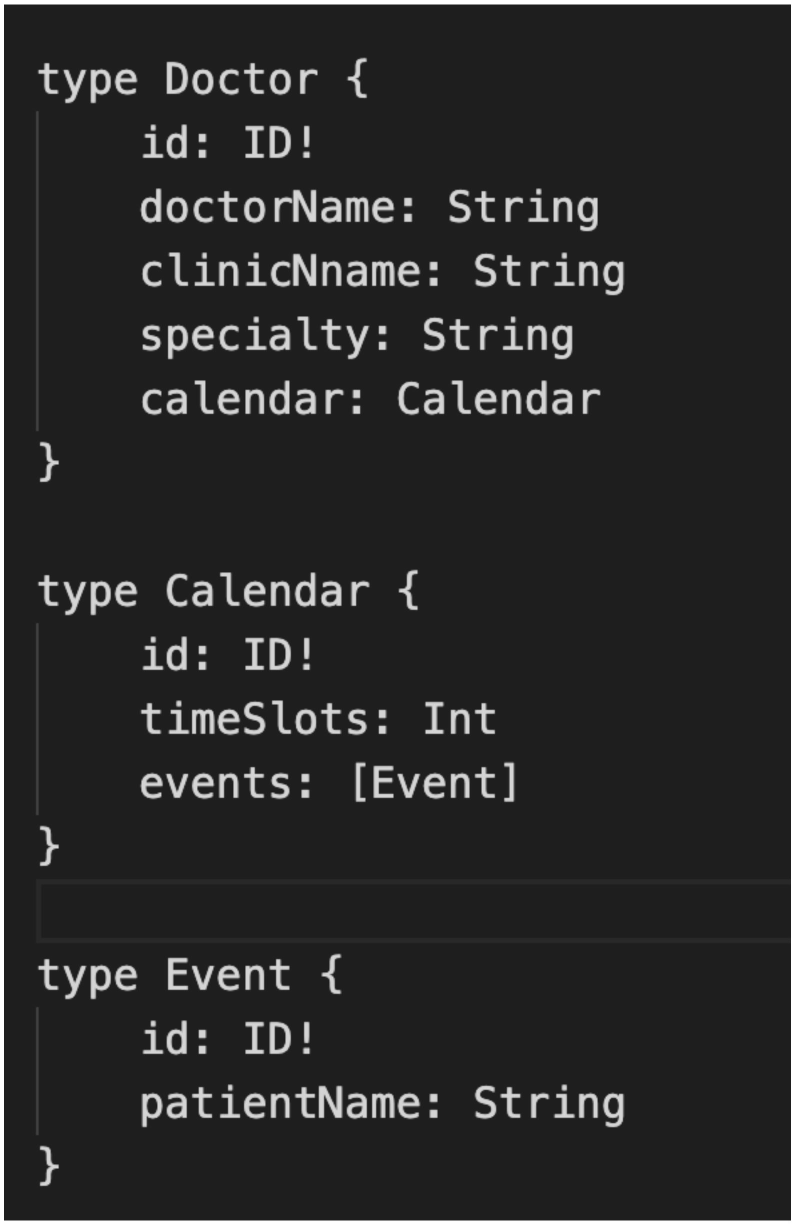


```
type Doctor {
    id: ID!
    name: String
    clinicNname: String
    specialty: String
}

type Calendar {
    id: ID!
    doctor: Doctor
    timeSlots: Int
}

type Event {
    id: ID!
    calendar: Calendar
    patientName: String
}
```

Then I found out if I structured it this way, when I want to use calendar and event related information, I need to do query first to find which calendar or event I'm referring to, instead of directly linking them together. Therefore I referred to our lecture slides and made some modification according to the student, university example and person post example. So my final schema looks like this:



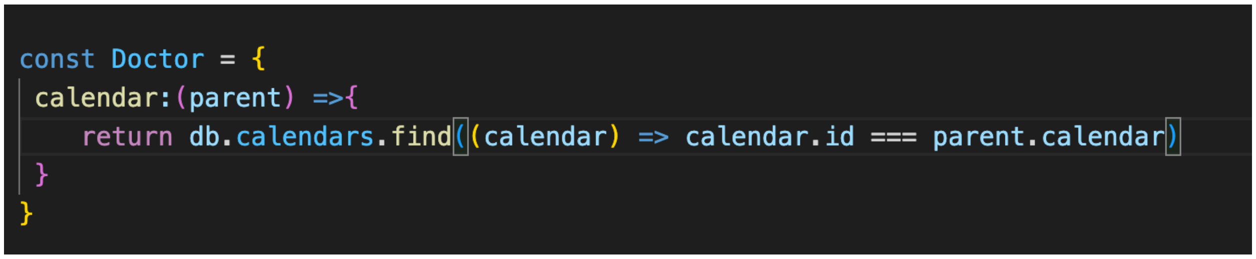
```
type Doctor {
    id: ID!
    doctorName: String
    clinicNname: String
    specialty: String
    calendar: Calendar
}

type Calendar {
    id: ID!
    timeSlots: Int
    events: [Event]
}

type Event {
    id: ID!
    patientName: String
}
```

Also, another alternative query design I made at the beginning is to have two queries, one for doctor, the other for the calendar, but after I made changes to the schema, there is no need to have the second query. Because the doctor query can cover it all and make it very simple.

Finally, I also encountered difficulty when I work on the resolver because the mentioned changes. After modification, the calendar is linked to the doctor, not the calendar id. Therefore my previous idea of joining calendar id together is wrong. When I do API design afterwards, I would remember to make corresponding changes when I modify schema or query design.



```
const Doctor = {
 calendar:(parent) =>{
    return db.calendars.find((calendar) => calendar.id === parent.calendar)
 }
}
```

Consider the case where, in future, the ‘Event’ structure is changed to have more fields e.g reference to patient details, consultation type (first time/follow-up etc.) and others.

○ What changes will the clients (API consumer) need to make to their existing queries (if any).

The clients just need to include what new fields they are interested in when they do queries, if any. But for the existing queries, there should not be any change.

○ How will you accommodate the changes in your existing Schema and Query types?

First of all, I need to include the new fields in the schema under Event type, of course. Then link new fields that are related to other types, if any. Then for the existing queries, there is no modification needed.

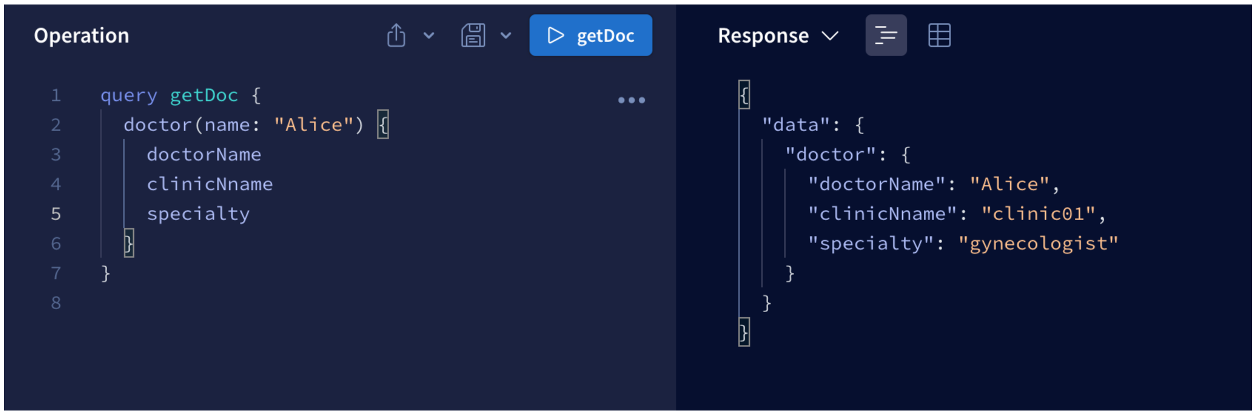
Describe **two** GraphQL best practices that you have incorporated in your API design.

First, when I design schemas, I always keep in mind that I should pay attention to non-nullable fields. For example the id of each type, to prevent possible errors manipulating the data. Also, I care about the nesting objects. As I mentioned above, I didn’t do well at the beginning and then realize type doctor and type calendar should be nested structure so that additional query is not requested to get calendar details.

Besides, when I design mutations, I stick to the rule that queries should be self-explanatory and mutations should be verb. The mutations are named bookAppointment, cancelAppointment and updateAppointment. And all the arguments are put into the input type.

A2.2 Testing

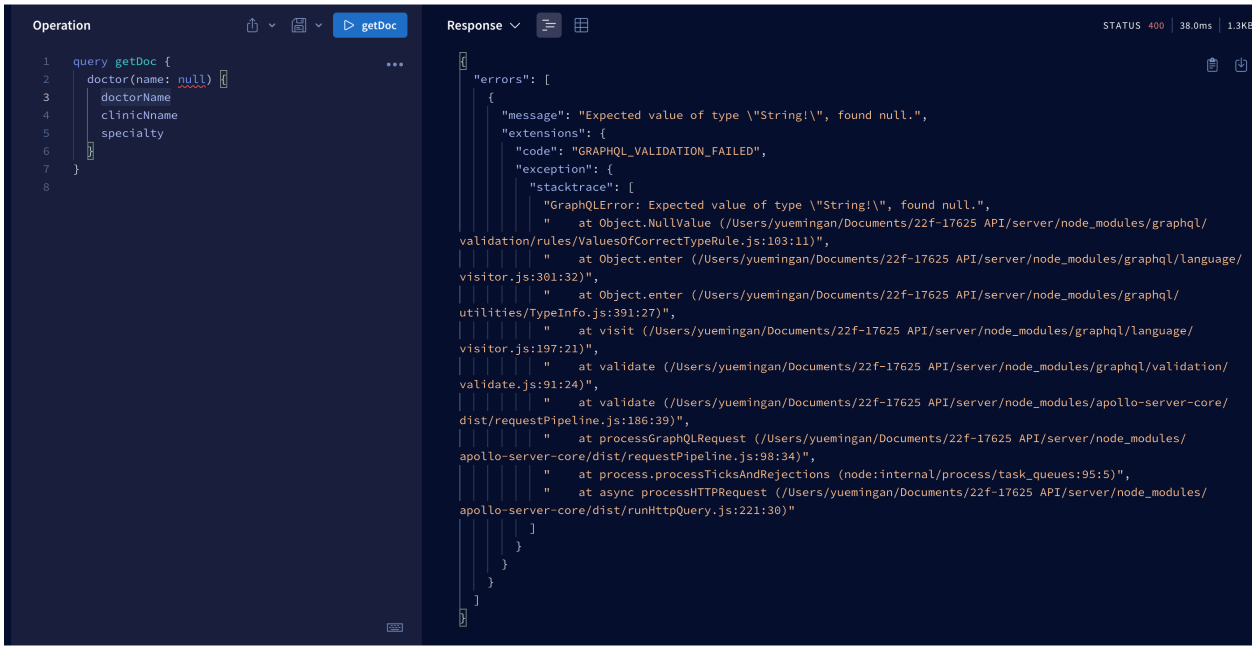
Q1.1



```graphql
query getDoc {
  doctor(name: "Alice") {
    doctorName
    clinicNname
    specialty
  }
}
```

```json
{
  "data": {
    "doctor": {
      "doctorName": "Alice",
      "clinicNname": "clinic01",
      "specialty": "gynecologist"
    }
  }
}
```

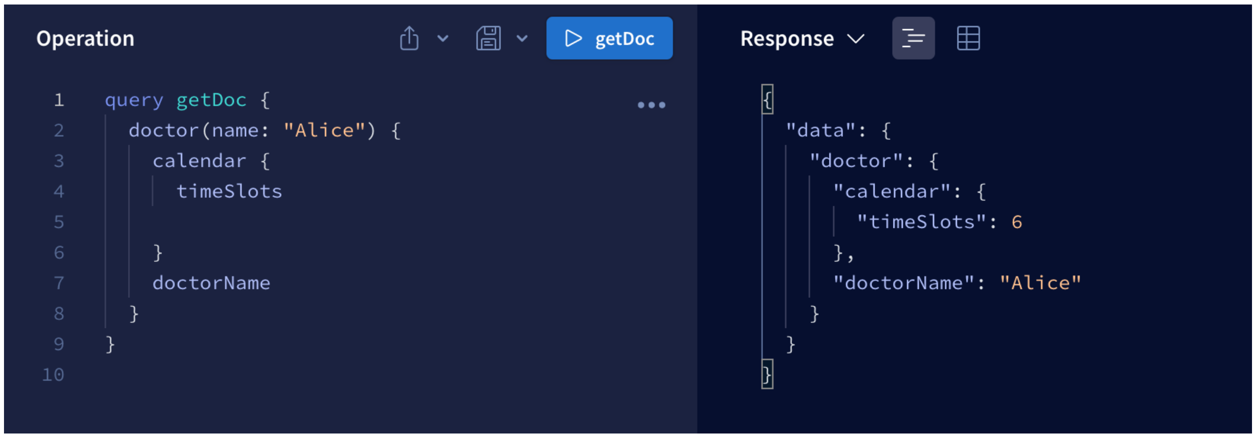
Q1.2



```graphql
query getDoc {
  doctor(name: null) {
    doctorName
    clinicNname
    specialty
  }
}
```

```json
{
  "errors": [
    {
      "message": "Expected value of type \"String!\", found null.",
      "extensions": {
        "code": "GRAPHQL_VALIDATION_FAILED",
        "exception": {
          "stacktrace": [
            "GraphQLError: Expected value of type \"String!\", found null.",
            "    at Object.NullValue (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/validation/rules/ValuesOfCorrectTypeRule.js:103:11)",
            "    at Object.enter (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/language/visitor.js:301:32)",
            "    at Object.enter (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/utilities/TypeInfo.js:391:27)",
            "    at visit (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/language/visitor.js:197:21)",
            "    at validate (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/validation/validate.js:91:24)",
            "    at validate (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/apollo-server-core/dist/requestPipeline.js:186:39)",
            "    at processGraphQLRequest (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/apollo-server-core/dist/requestPipeline.js:98:34)",
            "    at process.processTicksAndRejections (node:internal/process/task_queues:95:5)",
            "    at async processHTTPRequest (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/apollo-server-core/dist/runHttpQuery.js:221:30)"
          ]
        }
      }
    }
  ]
}
```

Q2.1

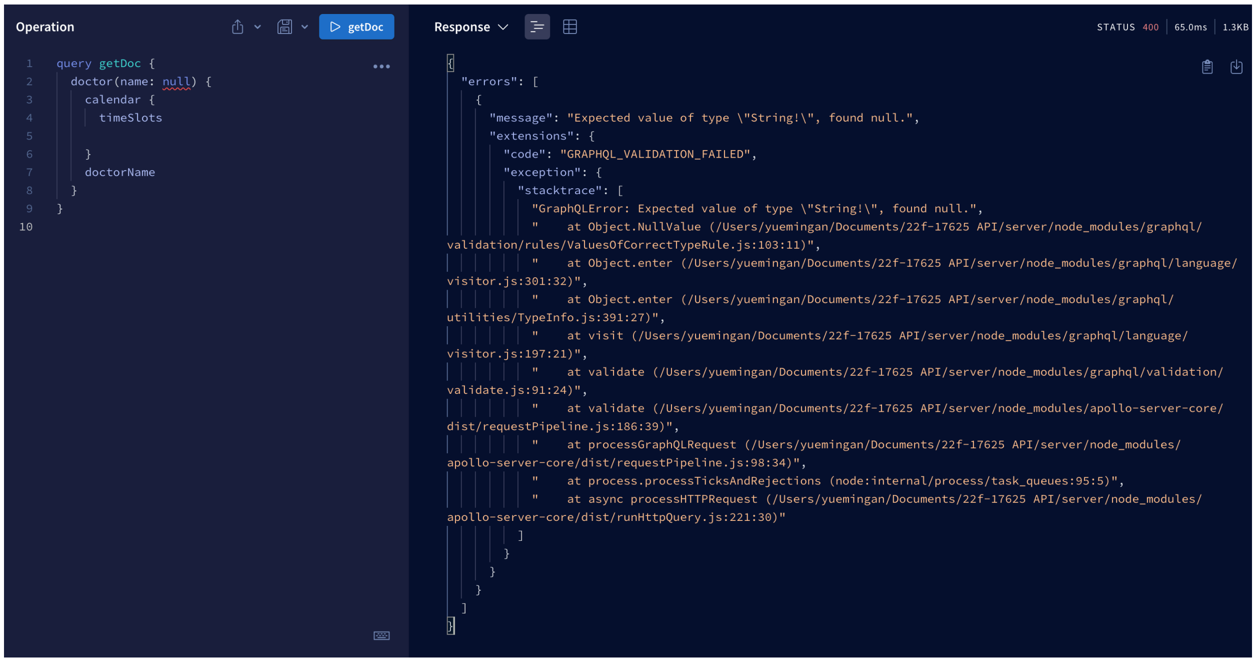


```graphql
query getDoc {
  doctor(name: "Alice") {
    calendar {
      timeSlots

    }
    doctorName
  }
}
```

```json
{
  "data": {
    "doctor": {
      "calendar": {
        "timeSlots": 6
      },
      "doctorName": "Alice"
    }
  }
}
```

Q2.2

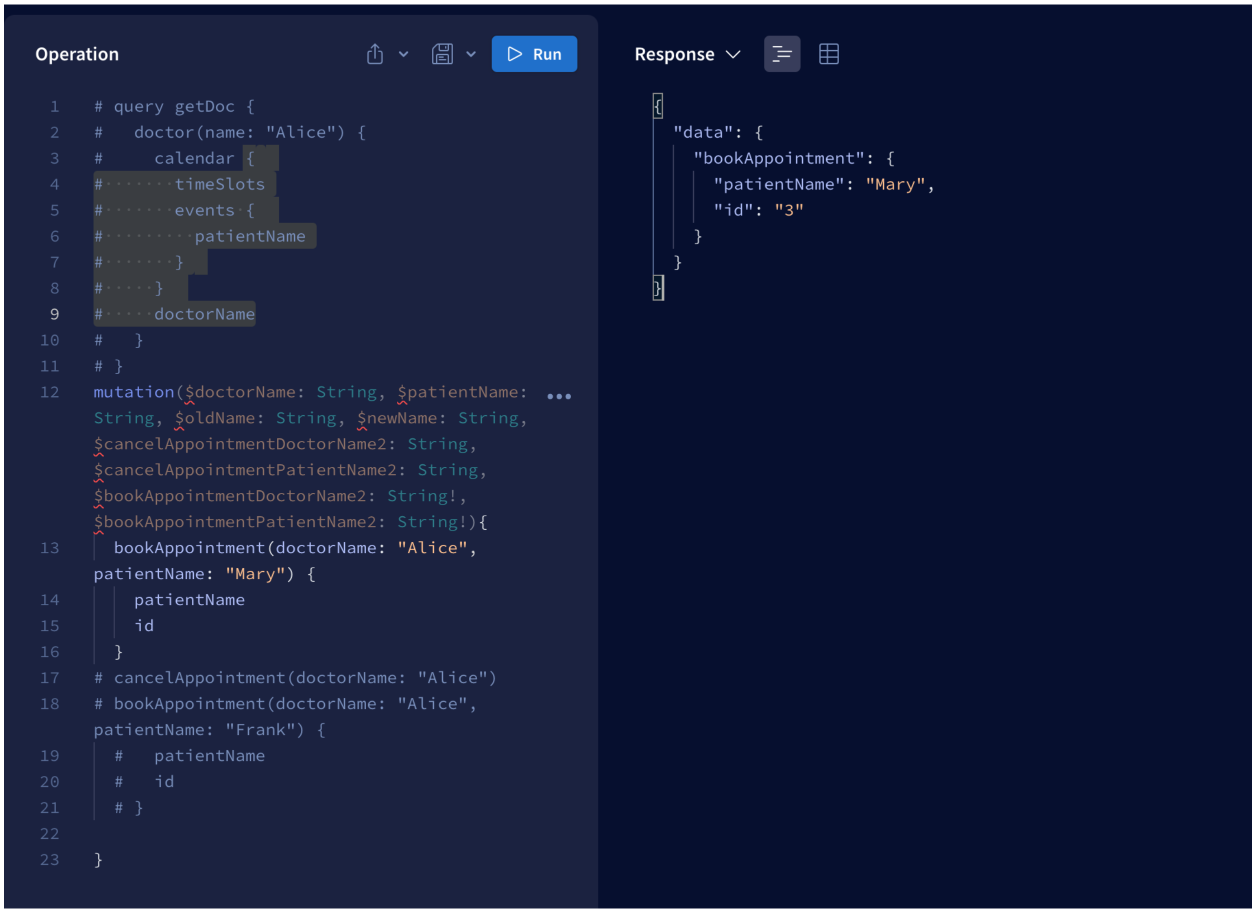


```graphql
query getDoc {
  doctor(name: null) {
    calendar {
      timeSlots

    }
    doctorName
  }
}
```

```json
{
  "errors": [
    {
      "message": "Expected value of type \"String!\", found null.",
      "extensions": {
        "code": "GRAPHQL_VALIDATION_FAILED",
        "exception": {
          "stacktrace": [
            "GraphQLError: Expected value of type \"String!\", found null.",
            "    at Object.NullValue (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/
validation/rules/ValuesOfCorrectTypeRule.js:103:11)",
            "    at Object.enter (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/language/
visitor.js:301:32)",
            "    at Object.enter (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/
utilities/TypeInfo.js:391:27)",
            "    at visit (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/language/
visitor.js:197:21)",
            "    at validate (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/validation/
validate.js:91:24)",
            "    at validate (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/apollo-server-core/
dist/requestPipeline.js:186:39)",
            "    at processGraphQLRequest (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/
apollo-server-core/dist/requestPipeline.js:98:34)",
            "    at process.processTicksAndRejections (node:internal/process/task_queues:95:5)",
            "    at async processHTTPRequest (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/
apollo-server-core/dist/runHttpQuery.js:221:30)"
          ]
        }
      }
    }
  ]
}
```

M1.1

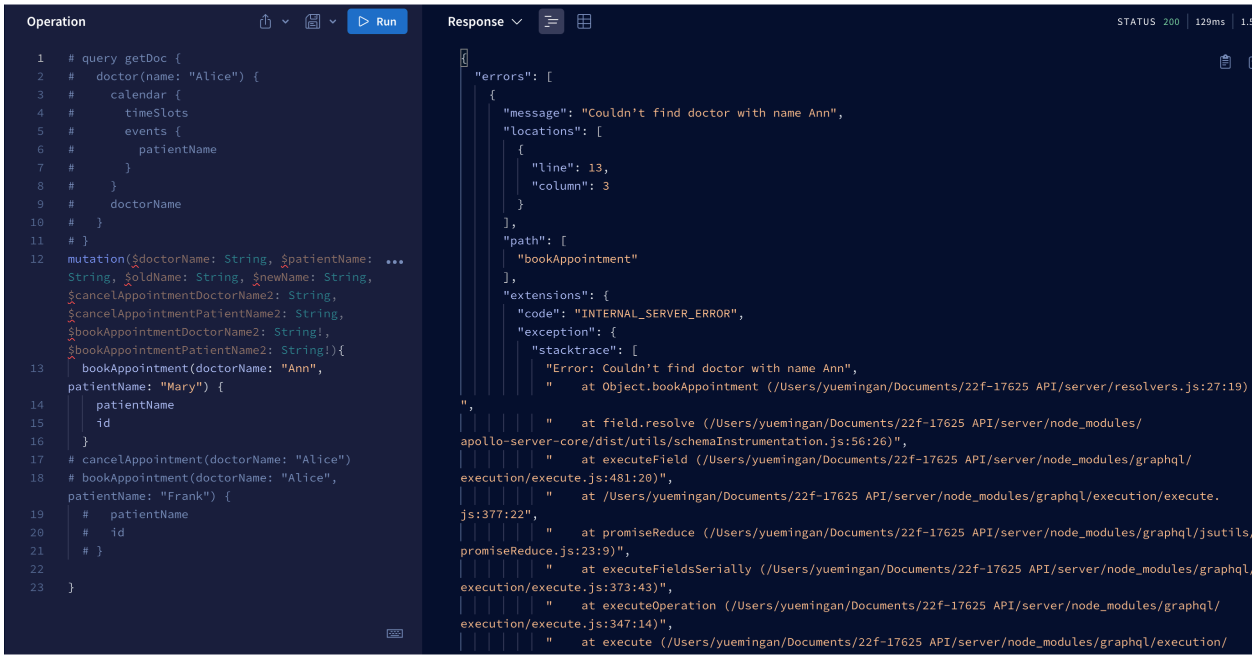


```graphql
# query getDoc {
#   doctor(name: "Alice") {
#     calendar {
#       timeSlots
#       events {
#         patientName
#       }
#     }
#     doctorName
#   }
# }
mutation($doctorName: String, $patientName: String, $oldName: String, $newName: String, $cancelAppointmentDoctorName2: String, $cancelAppointmentPatientName2: String, $bookAppointmentDoctorName2: String!, $bookAppointmentPatientName2: String!){
  bookAppointment(doctorName: "Alice", patientName: "Mary") {
    patientName
    id
  }
# cancelAppointment(doctorName: "Alice")
# bookAppointment(doctorName: "Alice", patientName: "Frank") {
  #   patientName
  #   id
  # }

}
```

```json
{
  "data": {
    "bookAppointment": {
      "patientName": "Mary",
      "id": "3"
    }
  }
}
```

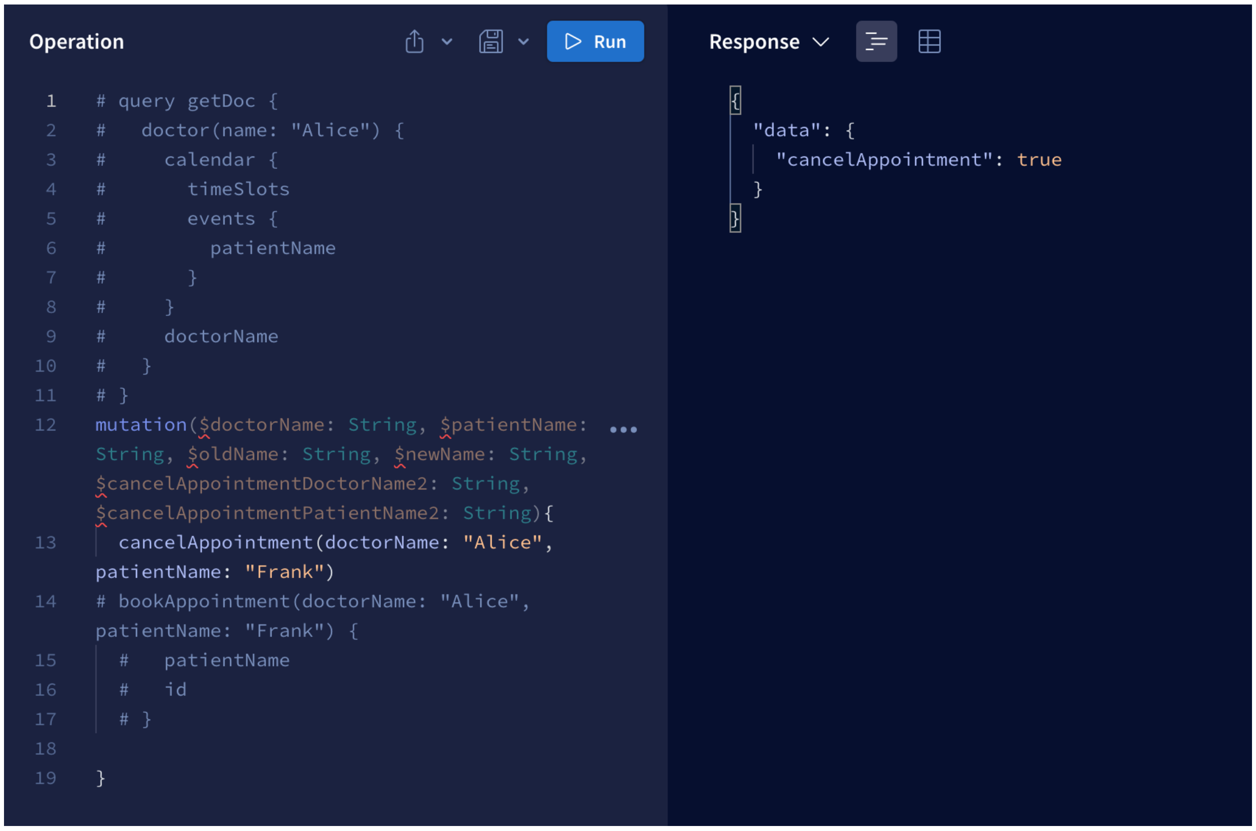
M1.2



```
1   # query getDoc {
2   #   doctor(name: "Alice") {
3   #     calendar {
4   #       timeSlots
5   #       events {
6   #         patientName
7   #       }
8   #     }
9   #     doctorName
10  #   }
11  # }
12  mutation($doctorName: String, $patientName:
    String, $oldName: String, $newName: String,
    $cancelAppointmentDoctorName2: String,
    $cancelAppointmentPatientName2: String,
    $bookAppointmentDoctorName2: String!,
    $bookAppointmentPatientName2: String!){
13    bookAppointment(doctorName: "Ann",
    patientName: "Mary") {
14      patientName
15      id
16    }
17  # cancelAppointment(doctorName: "Alice")
18  # bookAppointment(doctorName: "Alice",
    patientName: "Frank") {
19    #   patientName
20    #   id
21    # }
22
23  }
```

```
{
  "errors": [
    {
      "message": "Couldn’t find doctor with name Ann",
      "locations": [
        {
          "line": 13,
          "column": 3
        }
      ],
      "path": [
        "bookAppointment"
      ],
      "extensions": {
        "code": "INTERNAL_SERVER_ERROR",
        "exception": {
          "stacktrace": [
            "Error: Couldn’t find doctor with name Ann",
            "    at Object.bookAppointment (/Users/yuemingan/Documents/22f-17625 API/server/resolvers.js:27:19)
",
            "    at field.resolve (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/
apollo-server-core/dist/utils/schemaInstrumentation.js:56:26)",
            "    at executeField (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/
execution/execute.js:481:20)",
            "    at /Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/execution/execute.
js:377:22",
            "    at promiseReduce (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/jsutils/
promiseReduce.js:23:9)",
            "    at executeFieldsSerially (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/
execution/execute.js:373:43)",
            "    at executeOperation (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/
execution/execute.js:347:14)",
            "    at execute (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/execution/
```

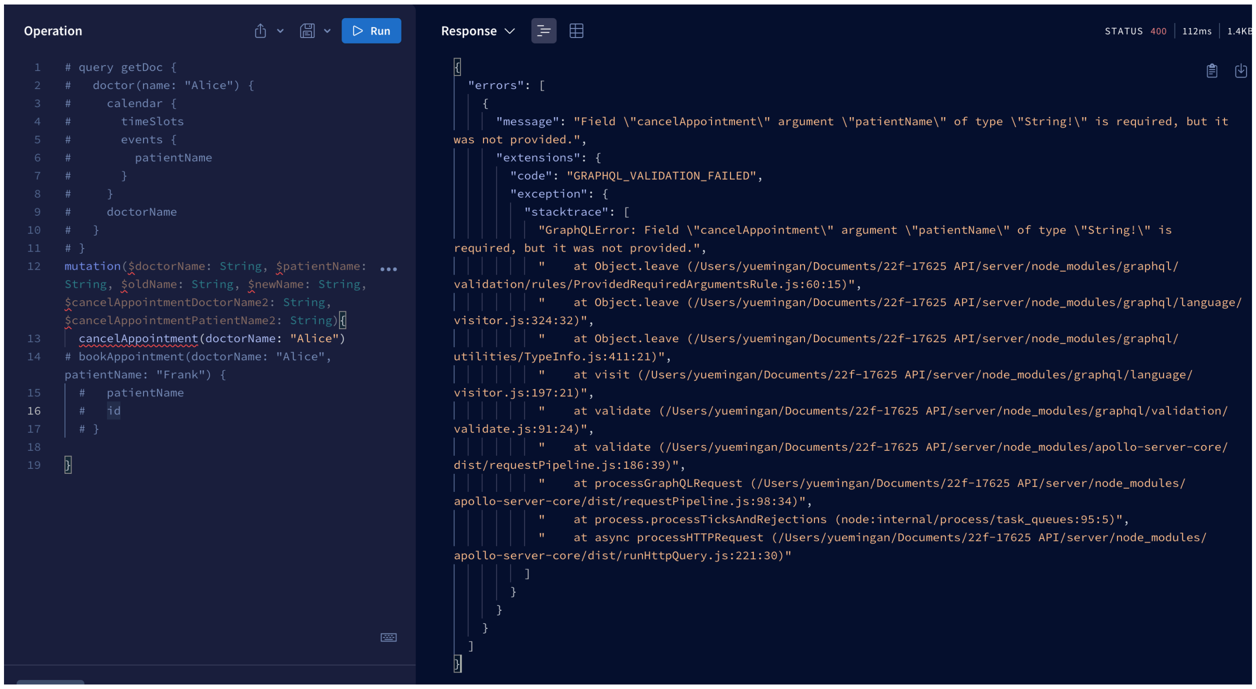
M2.1



```
1   # query getDoc {
2   #   doctor(name: "Alice") {
3   #     calendar {
4   #       timeSlots
5   #       events {
6   #         patientName
7   #       }
8   #     }
9   #     doctorName
10  #   }
11  # }
12  mutation($doctorName: String, $patientName:
    String, $oldName: String, $newName: String,
    $cancelAppointmentDoctorName2: String,
    $cancelAppointmentPatientName2: String){
13    cancelAppointment(doctorName: "Alice",
    patientName: "Frank")
14  # bookAppointment(doctorName: "Alice",
    patientName: "Frank") {
15    #   patientName
16    #   id
17    # }
18
19  }
```

```
{
  "data": {
    "cancelAppointment": true
  }
}
```

M2.2

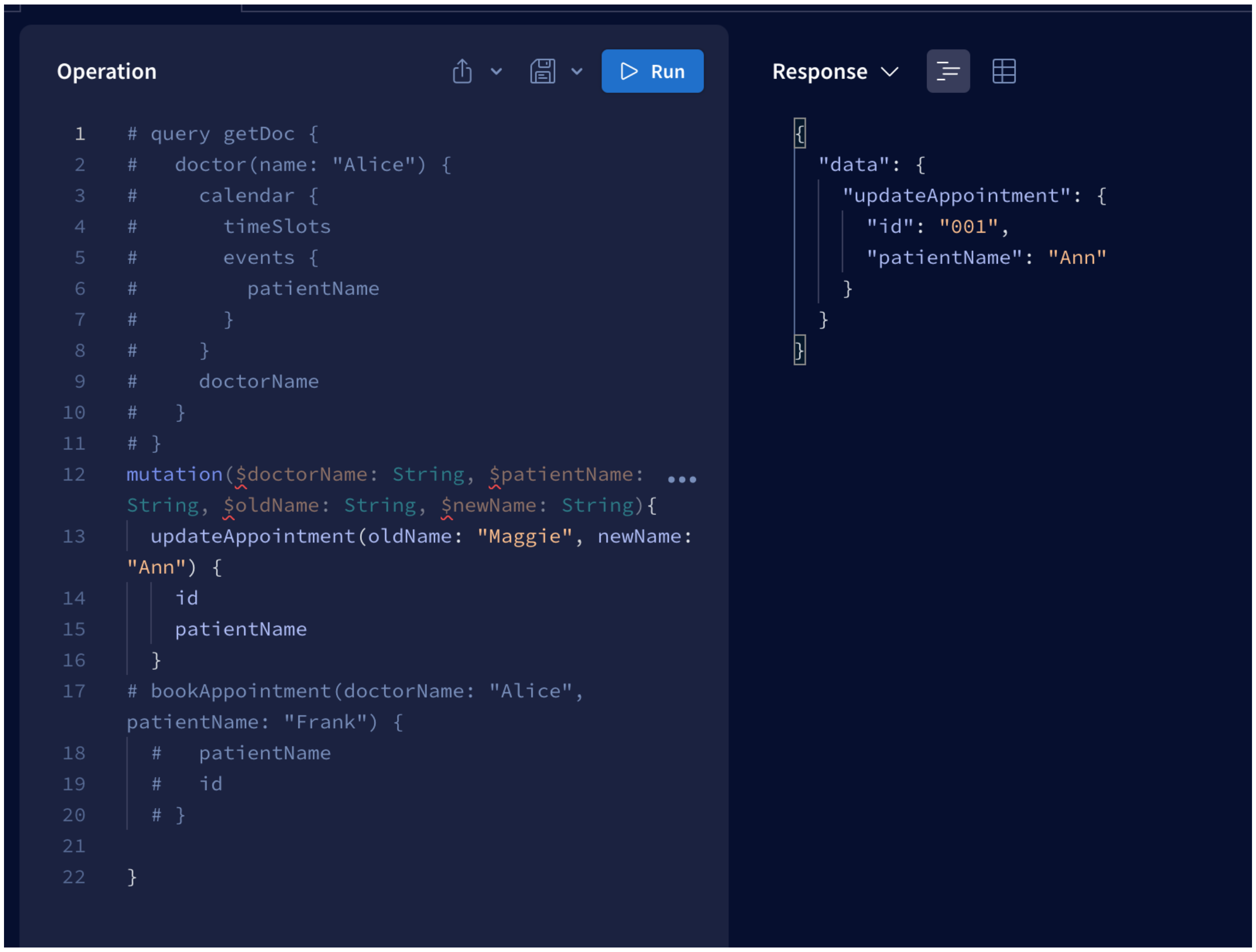


```
# query getDoc {
#   doctor(name: "Alice") {
#     calendar {
#       timeSlots
#       events {
#         patientName
#       }
#     }
#     doctorName
#   }
# }
mutation($doctorName: String, $patientName: String, $oldName: String, $newName: String, $cancelAppointmentDoctorName2: String, $cancelAppointmentPatientName2: String){
  cancelAppointment(doctorName: "Alice")
# bookAppointment(doctorName: "Alice", patientName: "Frank") {
  #   patientName
  #   id
  # }

}
```

```
{
  "errors": [
    {
      "message": "Field \"cancelAppointment\" argument \"patientName\" of type \"String!\" is required, but it was not provided.",
      "extensions": {
        "code": "GRAPHQL_VALIDATION_FAILED",
        "exception": {
          "stacktrace": [
            "GraphQLError: Field \"cancelAppointment\" argument \"patientName\" of type \"String!\" is required, but it was not provided.",
            "    at Object.leave (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/validation/rules/ProvidedRequiredArgumentsRule.js:60:15)",
            "    at Object.leave (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/language/visitor.js:324:32)",
            "    at Object.leave (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/utilities/TypeInfo.js:411:21)",
            "    at visit (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/language/visitor.js:197:21)",
            "    at validate (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/validation/validate.js:91:24)",
            "    at validate (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/apollo-server-core/dist/requestPipeline.js:186:39)",
            "    at processGraphQLRequest (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/apollo-server-core/dist/requestPipeline.js:98:34)",
            "    at process.processTicksAndRejections (node:internal/process/task_queues:95:5)",
            "    at async processHTTPRequest (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/apollo-server-core/dist/runHttpQuery.js:221:30)"
          ]
        }
      }
    }
  ]
}
```

M3.1

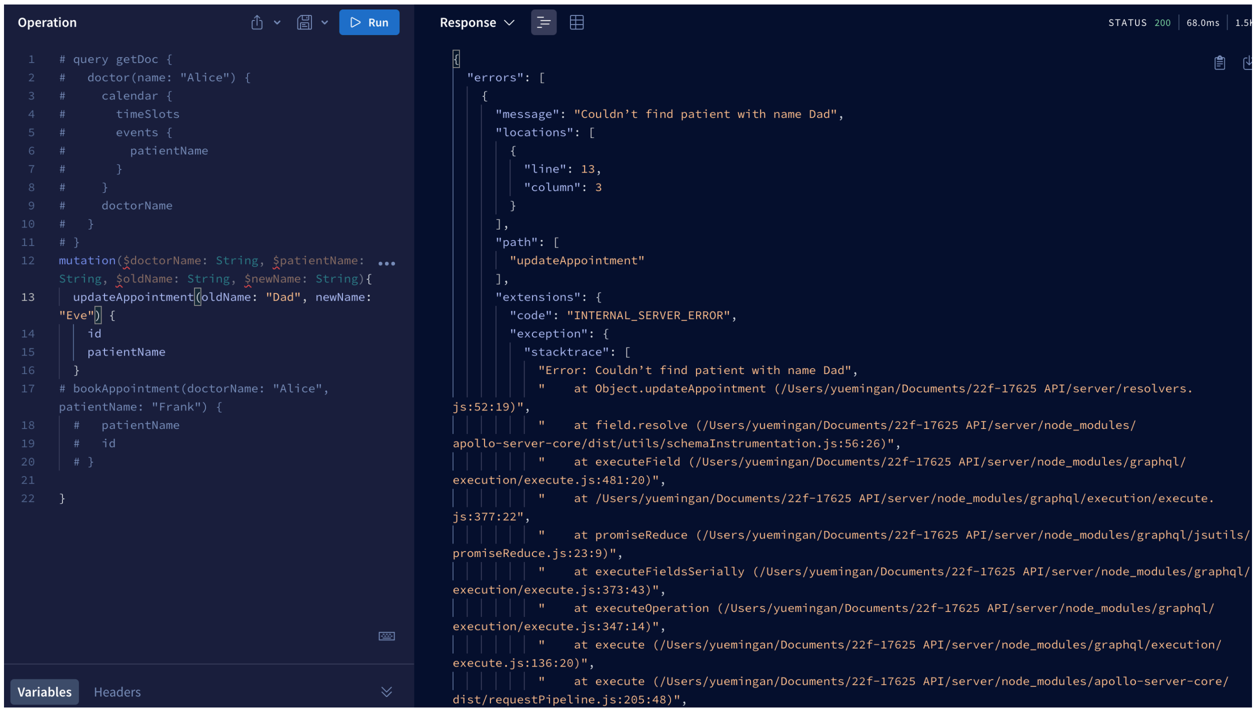


```
# query getDoc {
#   doctor(name: "Alice") {
#     calendar {
#       timeSlots
#       events {
#         patientName
#       }
#     }
#     doctorName
#   }
# }
mutation($doctorName: String, $patientName: String, $oldName: String, $newName: String){
  updateAppointment(oldName: "Maggie", newName: "Ann") {
    id
    patientName
  }
# bookAppointment(doctorName: "Alice", patientName: "Frank") {
  #   patientName
  #   id
  # }

}
```

```
{
  "data": {
    "updateAppointment": {
      "id": "001",
      "patientName": "Ann"
    }
  }
}
```

M3.2


Operation ▷ Run

```
1   # query getDoc {
2   #   doctor(name: "Alice") {
3   #     calendar {
4   #       timeSlots
5   #       events {
6   #         patientName
7   #       }
8   #     }
9   #     doctorName
10  #   }
11  # }
12  mutation($doctorName: String, $patientName:
    String, $oldName: String, $newName: String){
13    updateAppointment(oldName: "Dad", newName:
    "Eve") {
14      id
15      patientName
16    }
17  # bookAppointment(doctorName: "Alice",
    patientName: "Frank") {
18    #   patientName
19    #   id
20    # }
21
22  }
```

Variables Headers

Response STATUS 200 | 68.0ms | 1.5k

```
{
  "errors": [
    {
      "message": "Couldn't find patient with name Dad",
      "locations": [
        {
          "line": 13,
          "column": 3
        }
      ],
      "path": [
        "updateAppointment"
      ],
      "extensions": {
        "code": "INTERNAL_SERVER_ERROR",
        "exception": {
          "stacktrace": [
            "Error: Couldn't find patient with name Dad",
            "    at Object.updateAppointment (/Users/yuemingan/Documents/22f-17625 API/server/resolvers.
js:52:19)",
            "    at field.resolve (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/
apollo-server-core/dist/utils/schemaInstrumentation.js:56:26)",
            "    at executeField (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/
execution/execute.js:481:20)",
            "    at /Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/execution/execute.
js:377:22",
            "    at promiseReduce (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/jsutils/
promiseReduce.js:23:9)",
            "    at executeFieldsSerially (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/
execution/execute.js:373:43)",
            "    at executeOperation (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/
execution/execute.js:347:14)",
            "    at execute (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/graphql/execution/
execute.js:136:20)",
            "    at execute (/Users/yuemingan/Documents/22f-17625 API/server/node_modules/apollo-server-core/
dist/requestPipeline.js:205:48)",
```