GraphQL Data Mocking at Scale with LLMs and @generateMock

Michael Rebello

👋🏽 I’m Michael Rebello

Staff Engineer @ Airbnb

What % of your development time goes to writing and maintaining mock data when building a feature?

“Honestly, in some cases, messing with the manual data mocks can be ~30% of a task”

— Senior Software Engineer

`@generateMock`
solves the key challenges with
GraphQL data mocking

Agenda

- The Pains of GraphQL Data Mocking
- @generateMock: Schema + Context + LLMs = Magic
- @respondWithMock: Unblocking Client Development
- Schema Evolution: Keeping Mocks Truthful
- In the GraphQL community

Agenda

- The Pains of GraphQL Data Mocking
- @generateMock: Schema + Context + LLMs = Magic
- @respondWithMock: Unblocking Client Development
- Schema Evolution: Keeping Mocks Truthful
- In the GraphQL community

3 key challenges with GraphQL data mocking

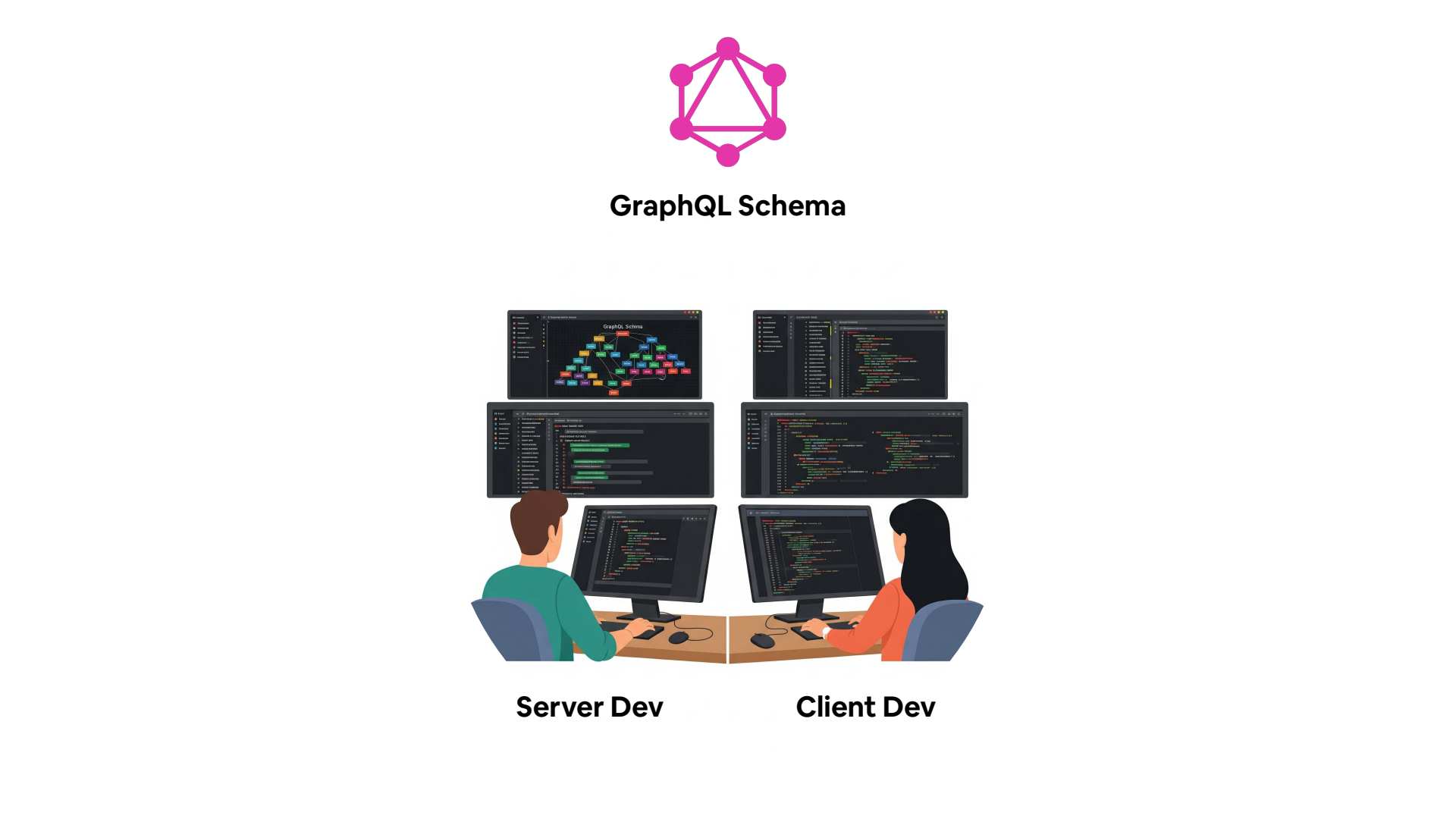
#1

Manually creating mocks is very time consuming

```
{
   "paymentPlanOptions":[
      {
         "amountSubtitle":null,
         "depositInfo":null,
         "learnMoreLink":null,
         "localizedAmount":"$2,123.99",
         "paymentsDepositUpsellData":null,
         "paymentPlanOption":{
            "depositOption":null,
            "displayString":"Pay in full",
            "paymentPlanType":"FULL_PAYMENT",
            "paymentPlanSubtype":"FULL_PAYMENT",
            "klarnaOption":null
         },
         "paymentPlanType":"FULL_PAYMENT",
         "subtitle":null,
         "title":"Pay $2,123.99 now",
         "savingsDetail": null
      },
      {
         "amountSubtitle":null,
         "depositInfo":{
            "lastChargeAmount":"$1,699.19",
            "lastChargeDate":"Oct 17, 2024"
         },
         "learnMoreLink":{
            "content":{
               "closeButtonText":"Cancel",
               "sections":[
                  {
                     "text":"Confirm your reservation by paying a portion of the total amount.",
                     "title":"Pay part of the total now",
                     "url":null,
                     "sectionHtml":null
                  },
                  {
                     "text":"Your original payment method will be charged on the second payment date.",
                     "title":"Pay the rest before check-in",
                     "url":null,
                     "sectionHtml":null
                  },
                  {
                     "text":"You don't have to worry, we'll send a reminder 3 days before the next payment.",
                     "title":"Payment is automatic",
                     "url":null,
                     "sectionHtml":null
                  },
                  {
                     "text":"Terms Apply",
                     "title":"",
                     "url":"/help/article/1696/pay-less-upfront-terms",
                     "sectionHtml":null
                  }
               ],
               "subtitle":"You can pay for part of this reservation now, and the rest later. No additional fees.",
               "title":"Pay part now, part later",
               "termsAndConditionsHtml":null,
               "headerImage":null

            ... 200 more lines
```

#2

Prototyping & demoing clients without the server is hard


GraphQL Schema
Server Dev
Client Dev

#3

Mocks get out-of-sync with GraphQL queries over time

```
query InboxMessages($userId: ID!) {
  inbox {
    # Existing fields
    threads {
      ...
    }
  }
}
```

```
query InboxMessages($userId: ID!) {
  inbox {
    # Existing fields
    threads {
      ...
    }
    # New fields
    emojiReactions {
      id
      emoji
      count
    }
  }
}
```

These are persistent challenges across the industry

Agenda

- The Pains of GraphQL Data Mocking
- @generateMock: Schema + Context + LLMs = Magic
- @respondWithMock: Unblocking Client Development
- Schema Evolution: Keeping Mocks Truthful
- In the GraphQL community

#1

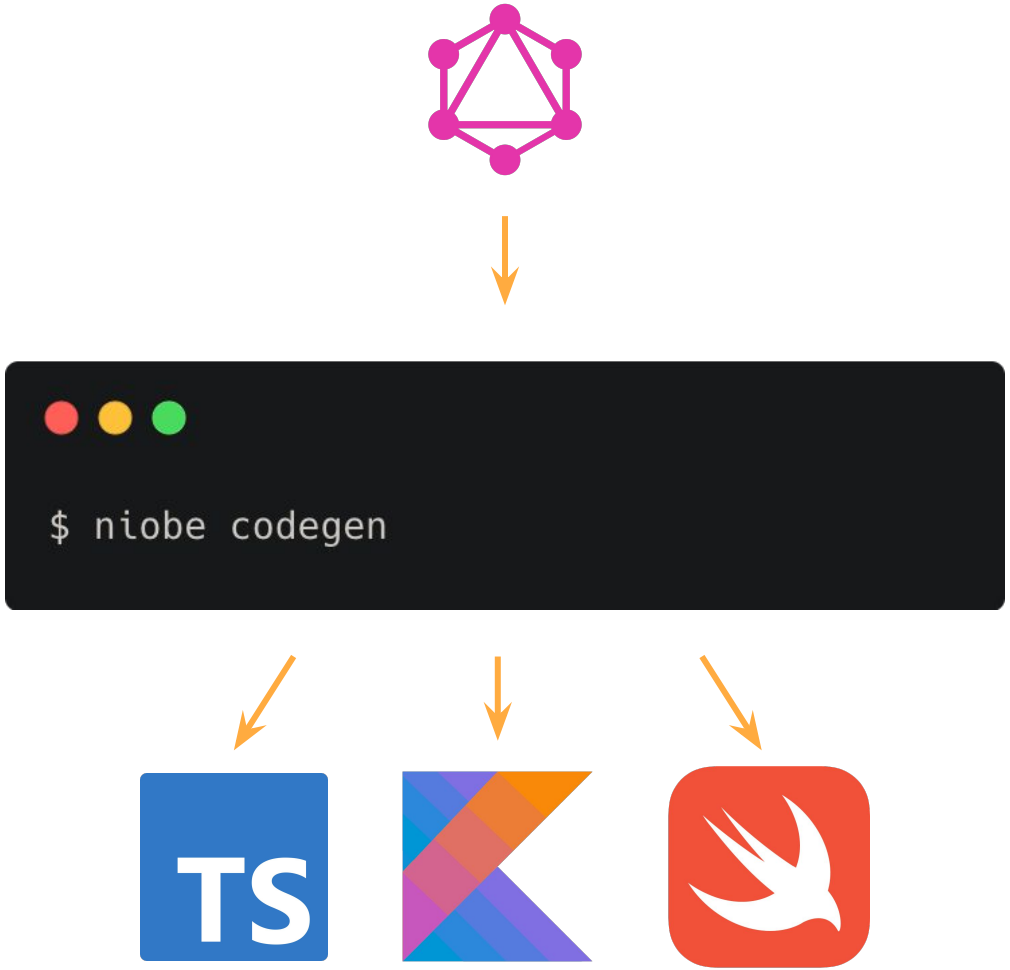
Eliminate the need to
hand-write mock data

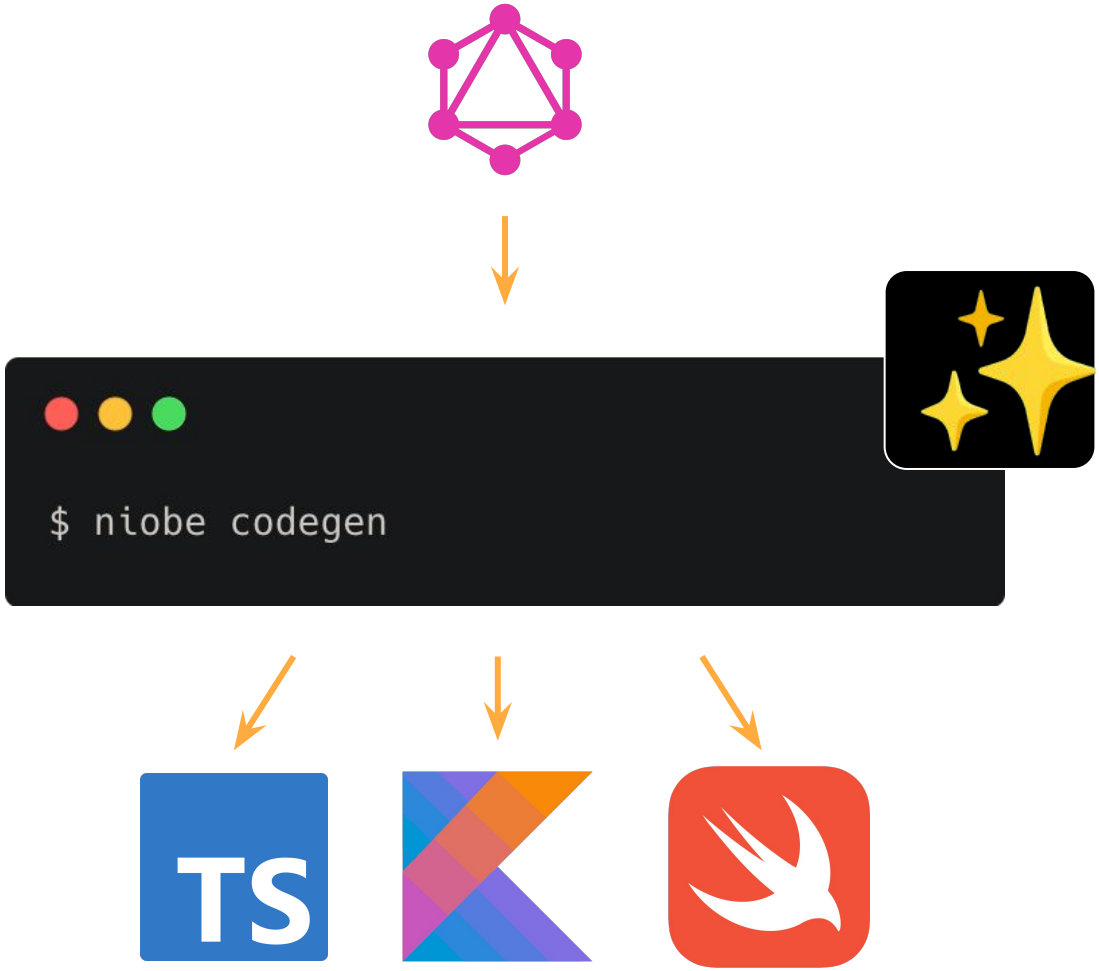
#2

Mock data should be
highly realistic

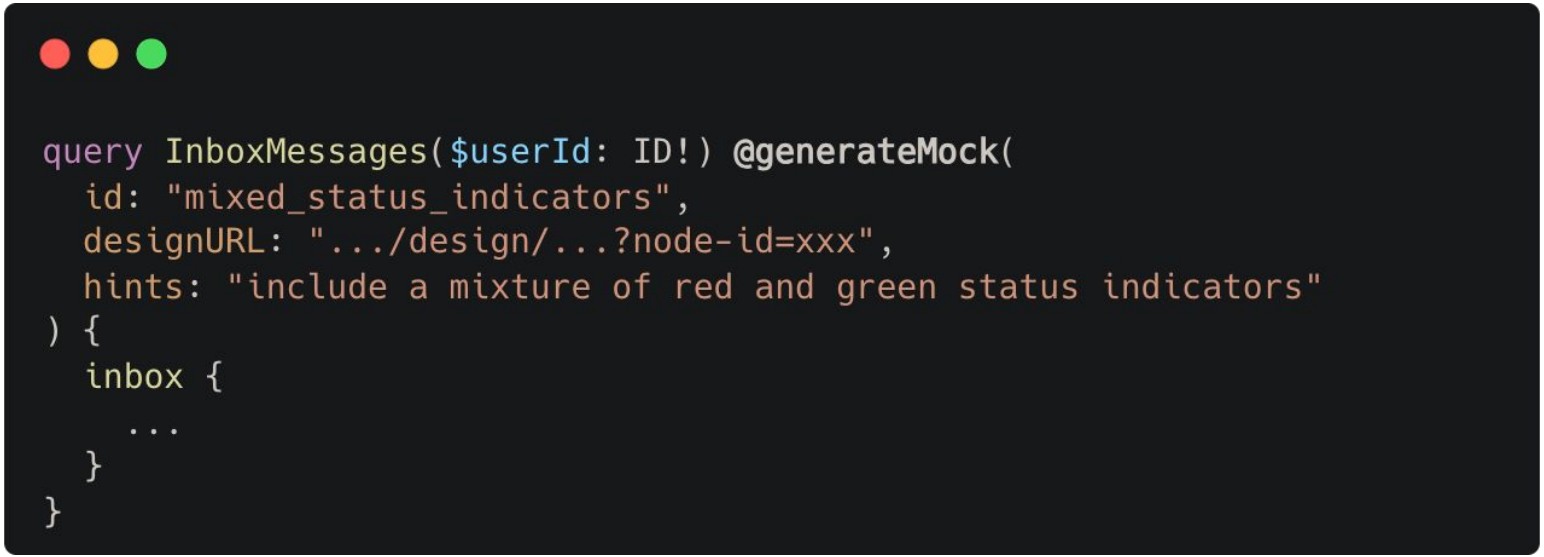
#3

Keep engineers in their local focus loop

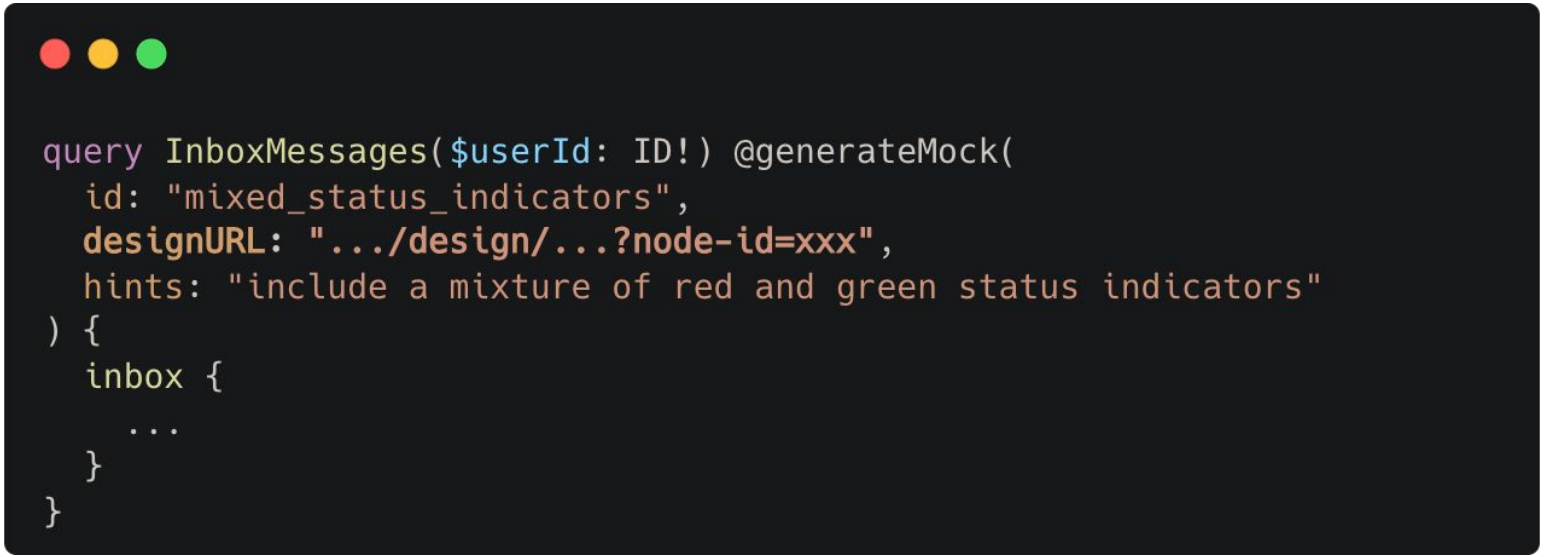

$ niobe codegen
TS


$ niobe codegen
TS

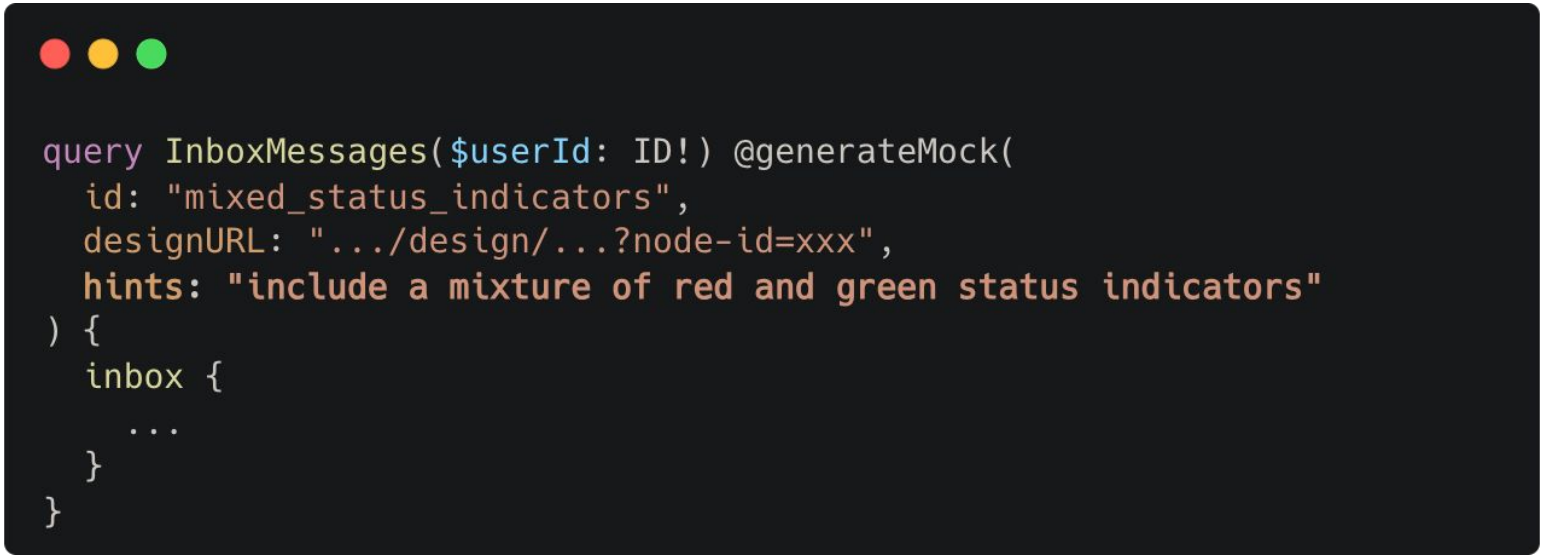
@generateMock



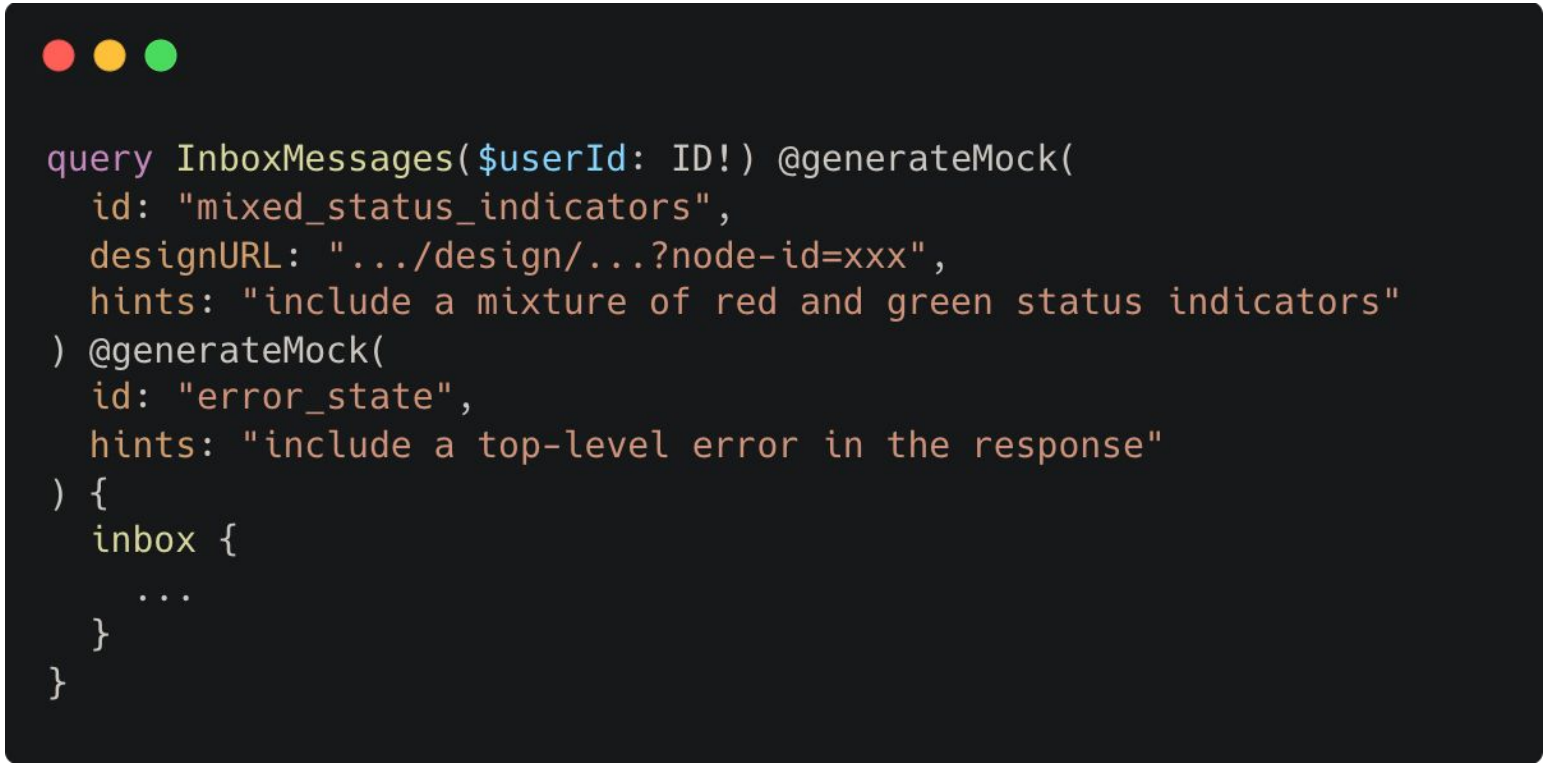
```
query InboxMessages($userId: ID!) @generateMock(
  id: "mixed_status_indicators",
  designURL: ".../design/...?node-id=xxx",
  hints: "include a mixture of red and green status indicators"
) {
  inbox {
    ...
  }
}
```



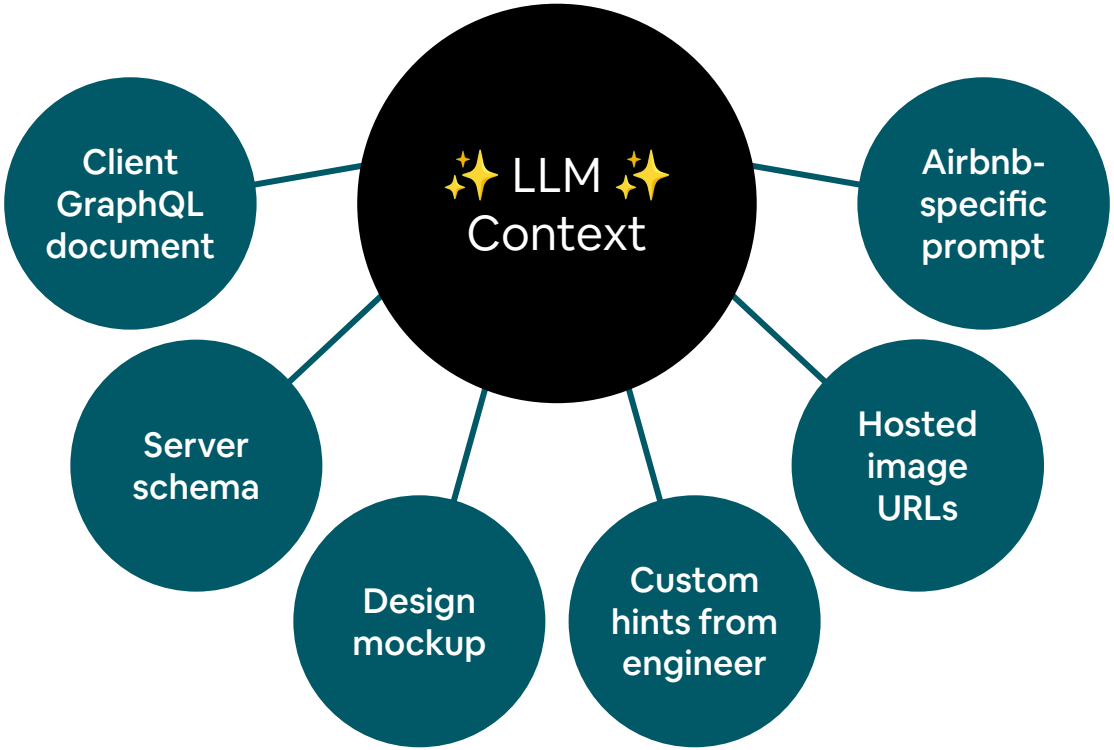
```
query InboxMessages($userId: ID!) @generateMock(
  id: "mixed_status_indicators",
  designURL: ".../design/...?node-id=xxx",
  hints: "include a mixture of red and green status indicators"
) {
  inbox {
    ...
  }
}
```



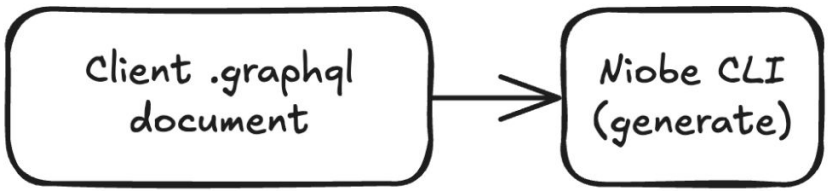
```
query InboxMessages($userId: ID!) @generateMock(
  id: "mixed_status_indicators",
  designURL: ".../design/...?node-id=xxx",
  hints: "include a mixture of red and green status indicators"
) {
  inbox {
    ...
  }
}
```

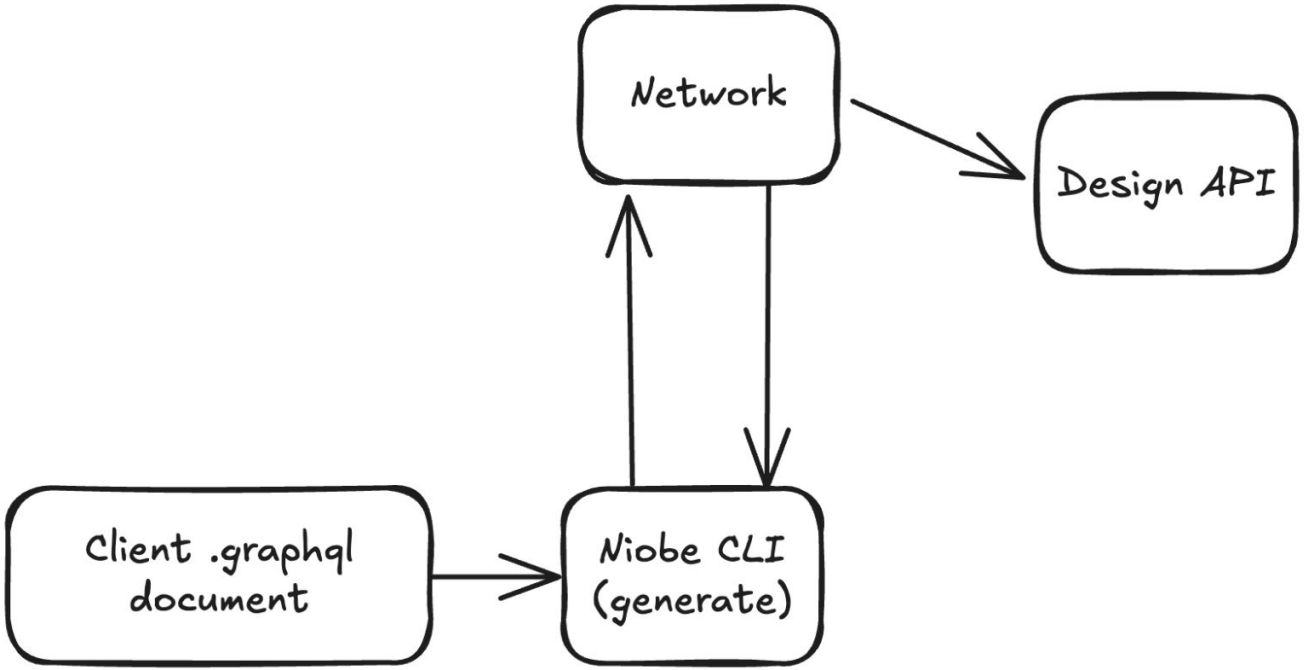


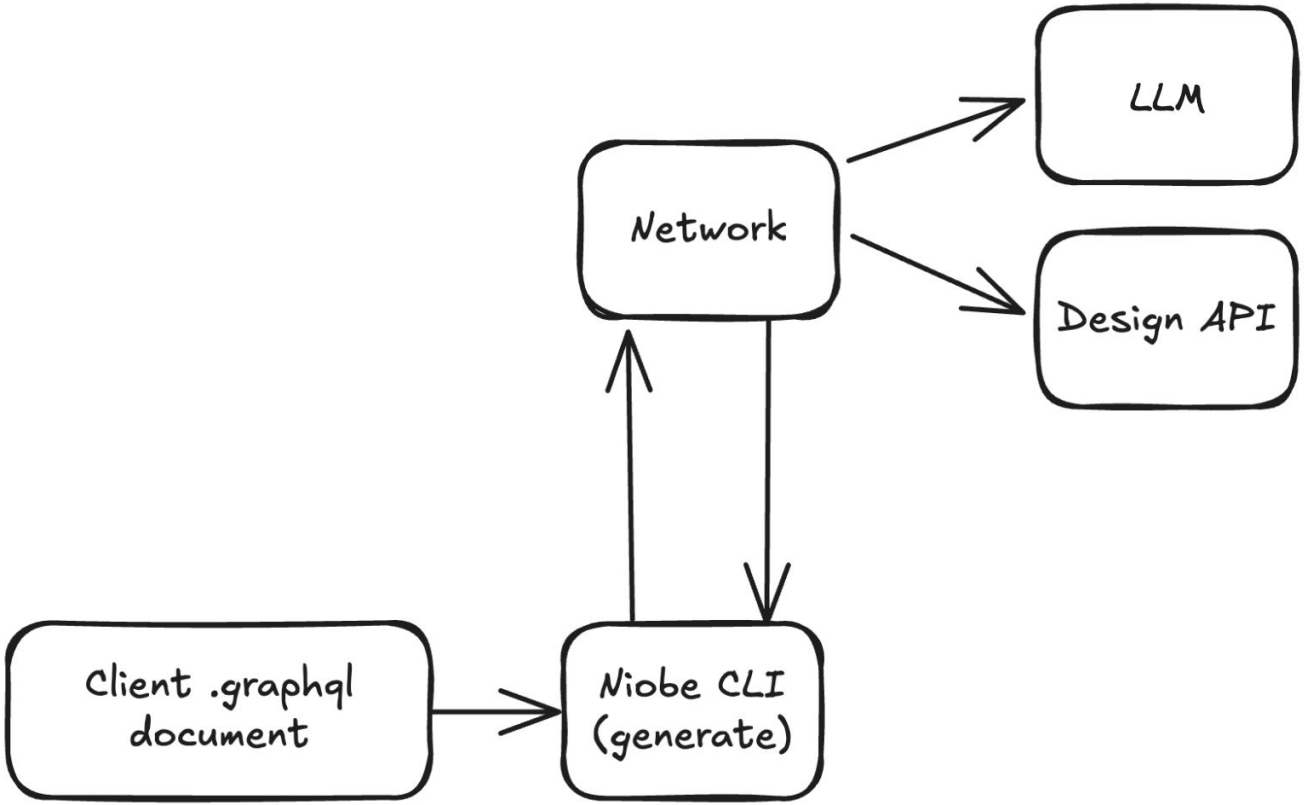
```
query InboxMessages($userId: ID!) @generateMock(
  id: "mixed_status_indicators",
  designURL: ".../design/...?node-id=xxx",
  hints: "include a mixture of red and green status indicators"
) @generateMock(
  id: "error_state",
  hints: "include a top-level error in the response"
) {
  inbox {
    ...
  }
}
```

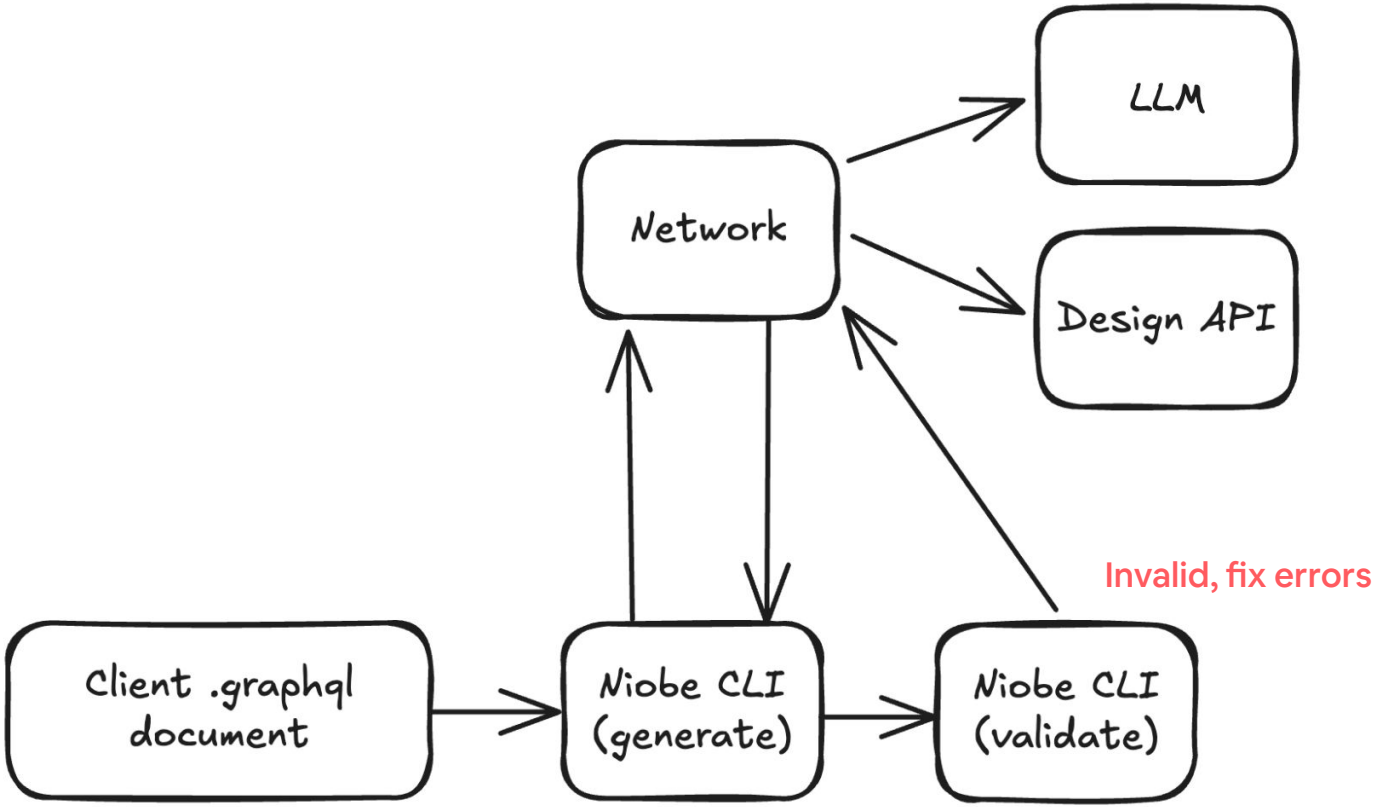

✨ LLM ✨
Context
Client GraphQL document
Airbnb-specific prompt
Server schema
Hosted image URLs
Design mockup
Custom hints from engineer


Client .graphql
document

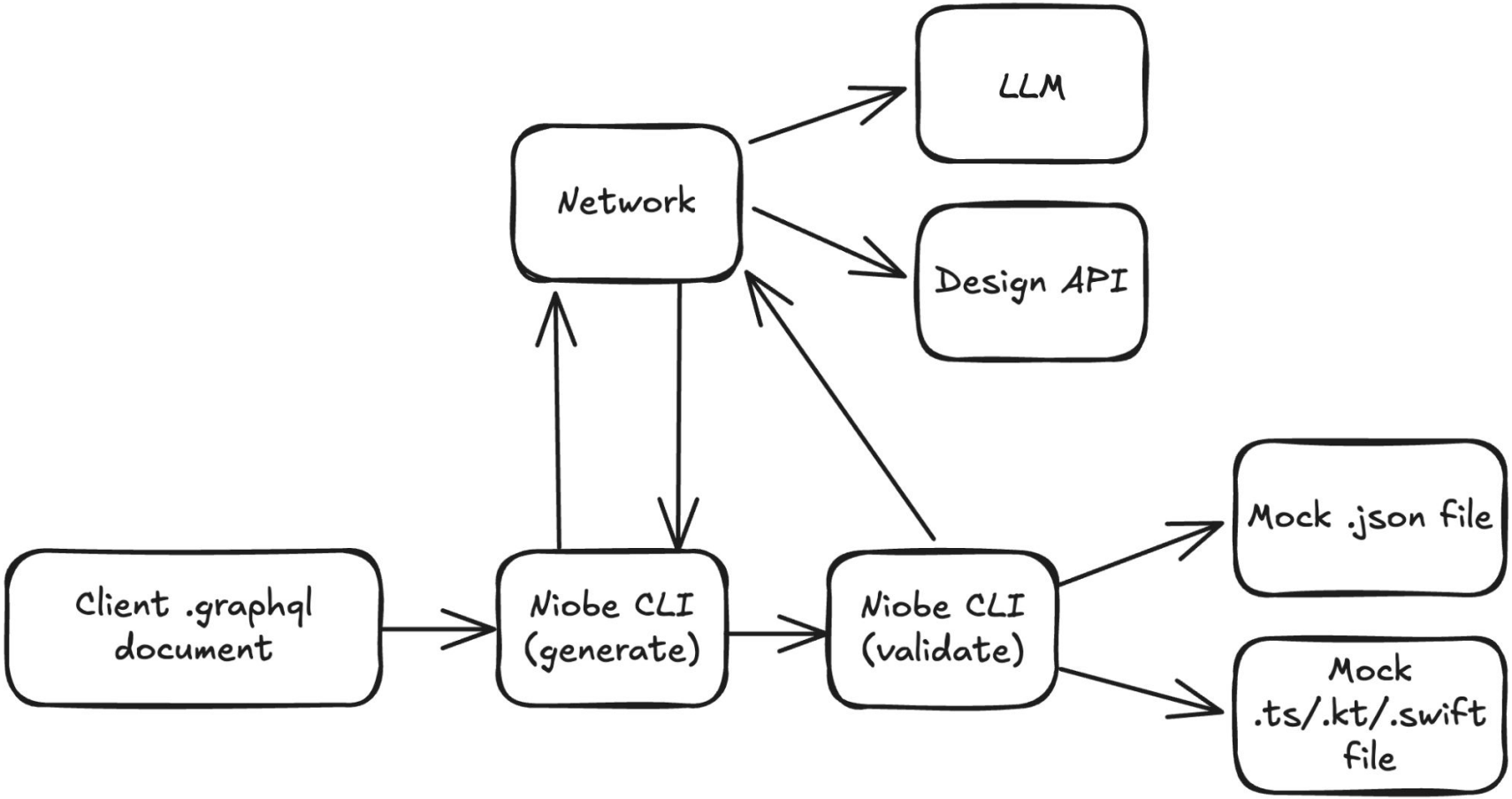

Client .graphql
document
Niobe CLI
(generate)

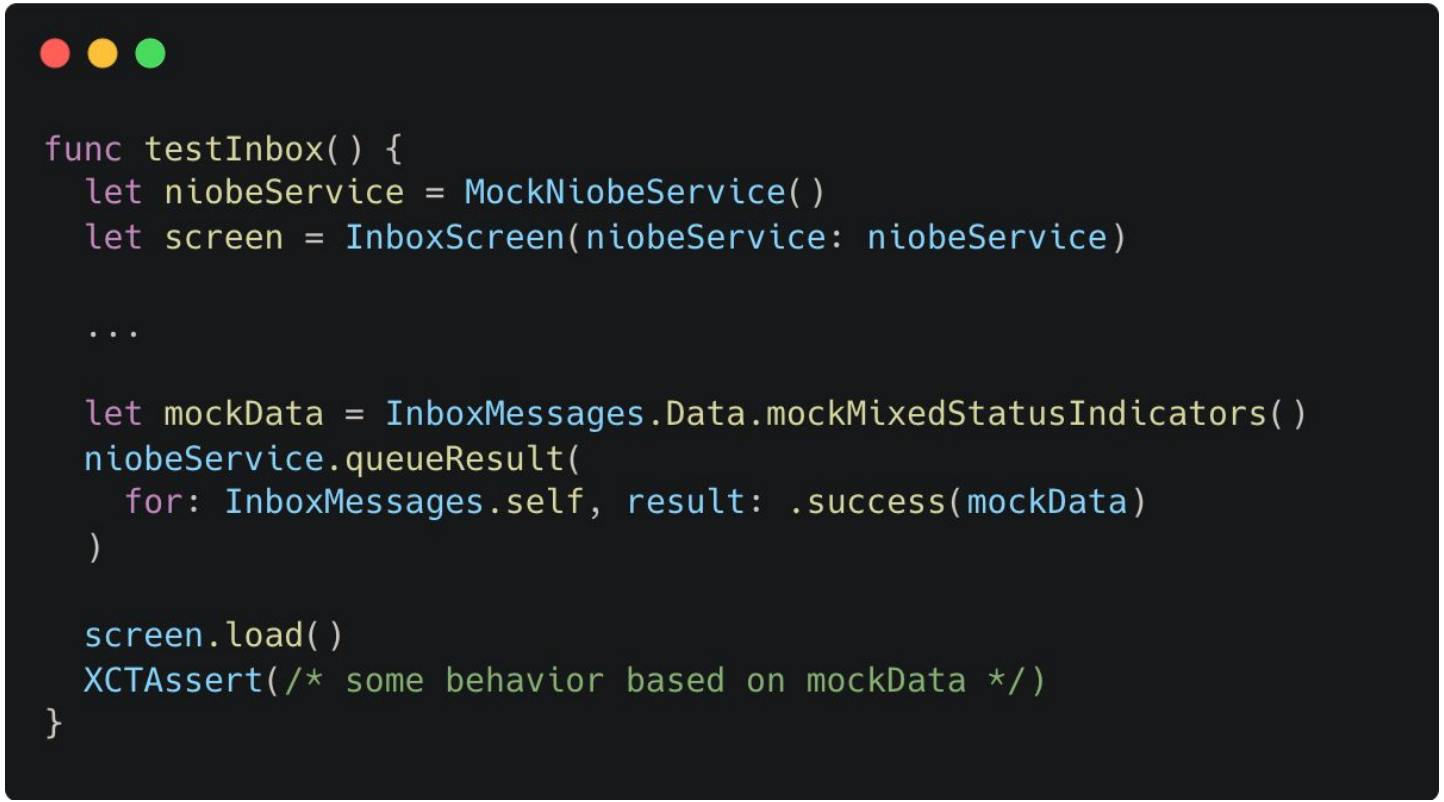

Network
Design API
Client .graphql
document
Niobe CLI
(generate)


LLM
Network
Design API
Client .graphql
document
Niobe CLI
(generate)


LLM
Network
Design API
Invalid, fix errors
Client .graphql document
Niobe CLI (generate)
Niobe CLI (validate)


Guardrails (Validation)
GraphQL Infra
GraphQL Infra
LLM


LLM
Network
Design API
Client .graphql document
Niobe CLI (generate)
Niobe CLI (validate)
Mock .json file
Mock .ts/.kt/.swift file

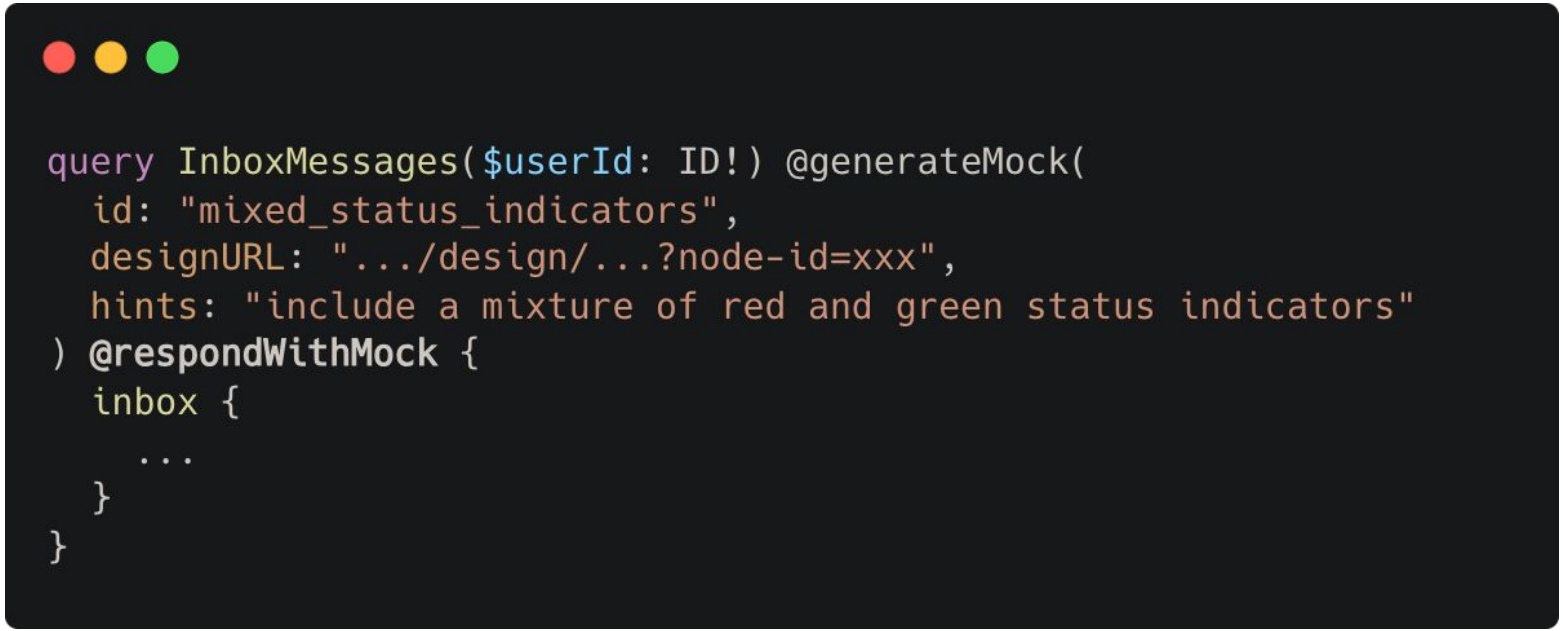


```
func testInbox() {
  let niobeService = MockNiobeService()
  let screen = InboxScreen(niobeService: niobeService)

  ...

  let mockData = InboxMessages.Data.mockMixedStatusIndicators()
  niobeService.queueResult(
    for: InboxMessages.self, result: .success(mockData)
  )

  screen.load()
  XCTAssert(/* some behavior based on mockData */)
}
```

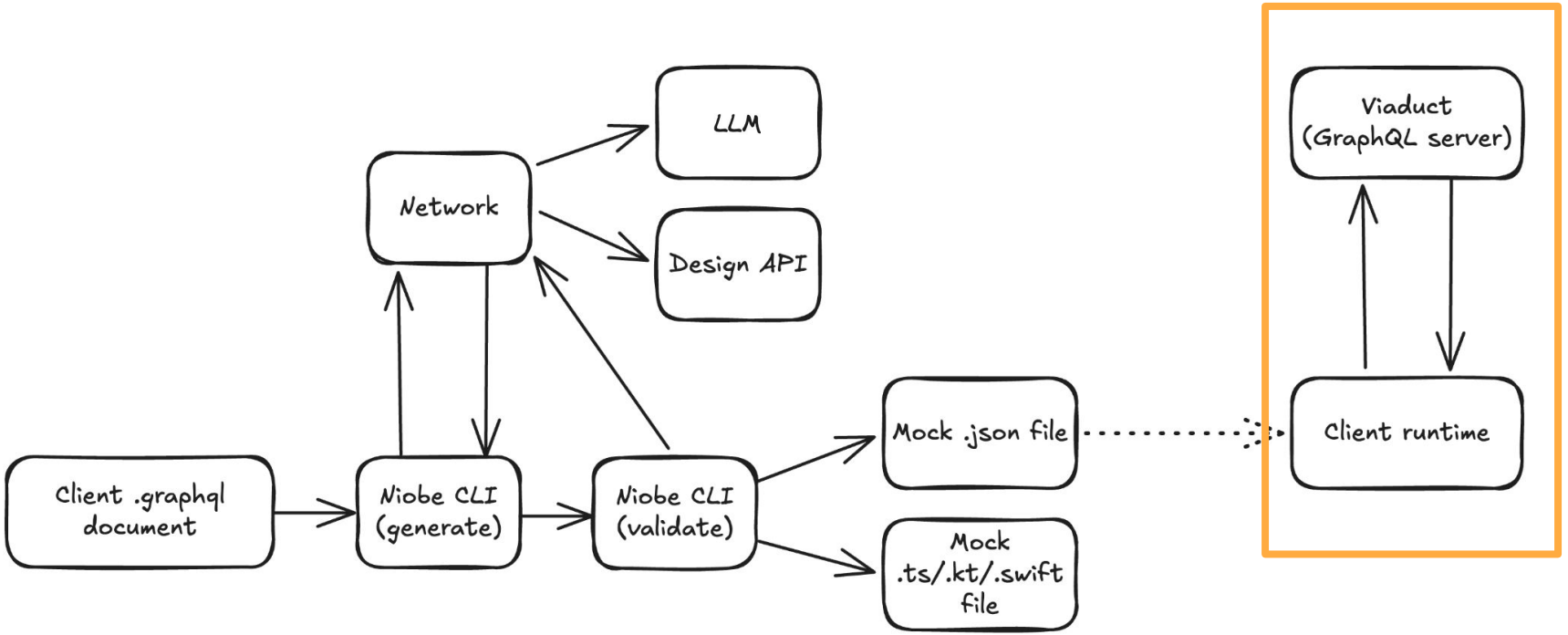
Agenda

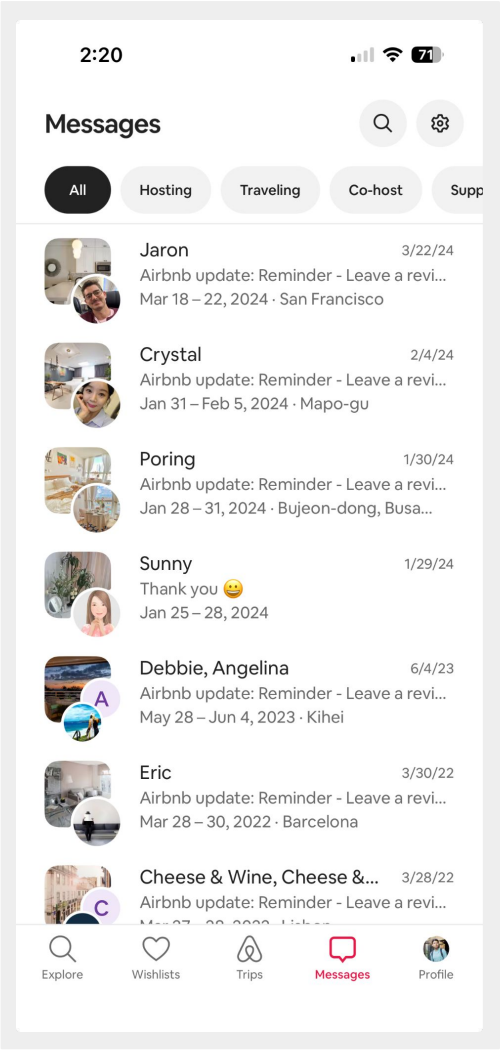
- The Pains of GraphQL Data Mocking
- @generateMock: Schema + Context + LLMs = Magic
- @respondWithMock: Unblocking Client Development
- Schema Evolution: Keeping Mocks Truthful
- In the GraphQL community

@respondWithMock



```
query InboxMessages($userId: ID!) @generateMock(
  id: "mixed_status_indicators",
  designURL: ".../design/...?node-id=xxx",
  hints: "include a mixture of red and green status indicators"
) @respondWithMock {
  inbox {
    ...
  }
}
```

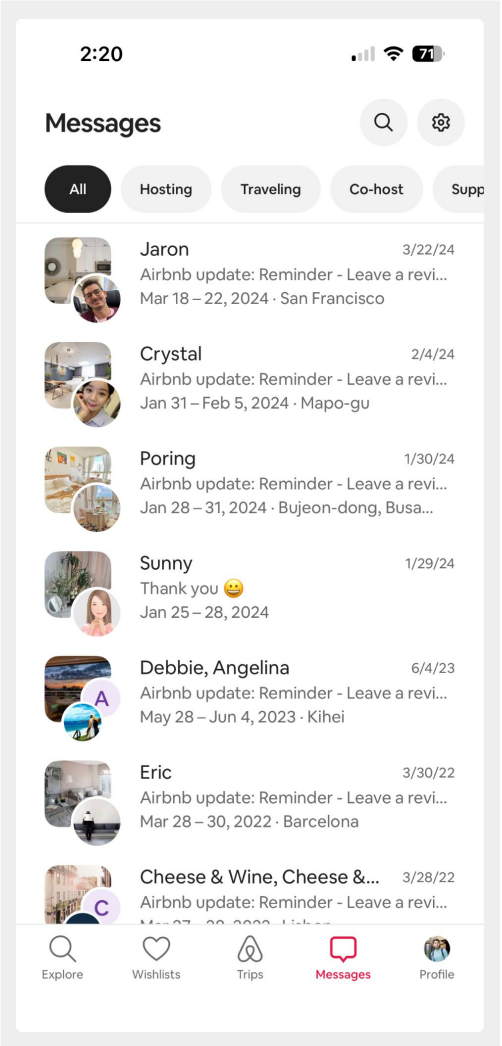

LLM
Network
Design API
Viaduct
(GraphQL server)
Mock .json file
Client runtime
Client .graphql
document
Niobe CLI
(generate)
Niobe CLI
(validate)
Mock
.ts/.kt/.swift
file



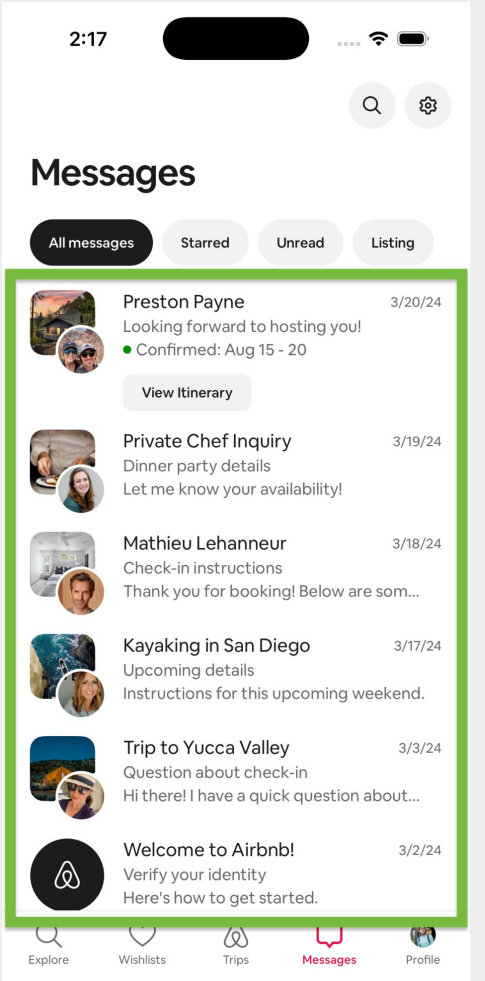

```
query InboxMessages($userId: ID!) @generateMock(
  id: "mixed_status_indicators",
  designURL: ".../design/...?node-id=xxx",
  hints: "include a mixture of red and green status indicators"
) @respondWithMock {
  inbox {
    ...
  }
}
```

```
$ niobe codegen
```

Design

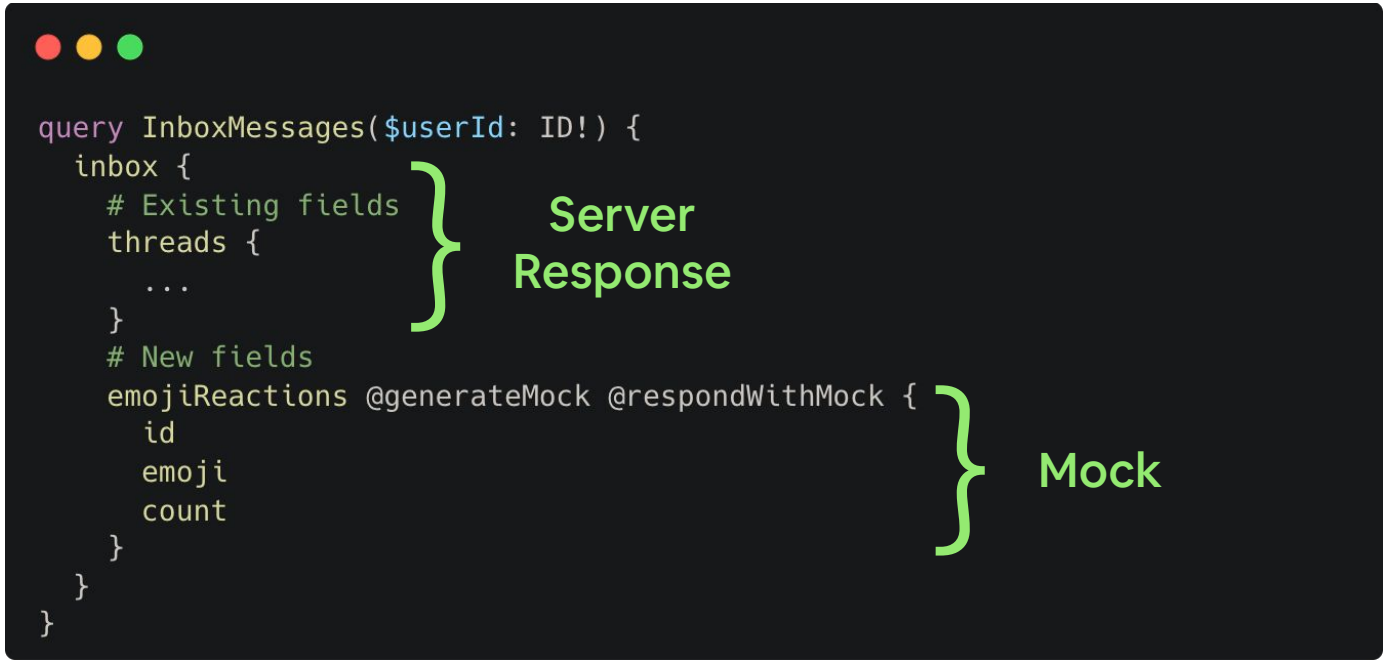


Generated



Add directive → generate → run

```
query InboxMessages($userId: ID!) {
  inbox {
    # Existing fields
    threads {
      ...
    }
    # New fields
    emojiReactions {
      id
      emoji
      count
    }
  }
}
```


query InboxMessages($userId: ID!) {
inbox {
# Existing fields
threads {
...
}
# New fields
emojiReactions @generateMock @respondWithMock {
id
emoji
count
}
}
}
Server Response
Mock

Conditional mocking

```
query InboxMessages($userId: ID!) {
  inbox {
    # Existing fields
    threads {
      ...
    }
    # If $userId == "id123", uses mock reactions_with_feature1.
    # If $userId == "id456", uses mock reactions_with_feature2.
    # Otherwise, falls back to mock default_fallback since it does not have any conditionals.
    # Note: If there was no unconditional @respondWithMock specified, the server response would be used instead.
    emojiReactions @generateMock(
      id: "reactions_with_feature1", hints: "..."
    ) @respondWithMock(
      id: "reactions_with_feature1", if: { id: $userId }, matches: { id: "id123" }
    ) @generateMock(
      id: "reactions_with_feature2", hints: "..."
    ) @respondWithMock(
      id: "reactions_with_feature2", if: { id: $userId }, matches: { id: "id456" }
    ) @generateMock(
      id: "default_fallback", hints: "..."
    ) @respondWithMock(
      id: "default_fallback"
    ) {
      id
      emoji
      count
    }
  }
}
```

Agenda

- The Pains of GraphQL Data Mocking
- @generateMock: Schema + Context + LLMs = Magic
- @respondWithMock: Unblocking Client Development
- Schema Evolution: Keeping Mocks Truthful
- In the GraphQL community

```
query InboxMessages($userId: ID!) {
  inbox {
    # Existing fields
    threads {
      ...
    }
    # New fields
    emojiReactions {
      id
      emoji
      count
    }
  }
}
```


{
"__niobe_mock_client_entity_hash": "XXXXX",
"__niobe_mock_directive_hash": "YYYYY",
/* Mock data */
}


query InboxMessages($userId: ID!) {
inbox {
threads {
...
}
+ emojiReactions @generateMock @respondWithMock {
+ id
+ emoji
+ count
+ }
}
}

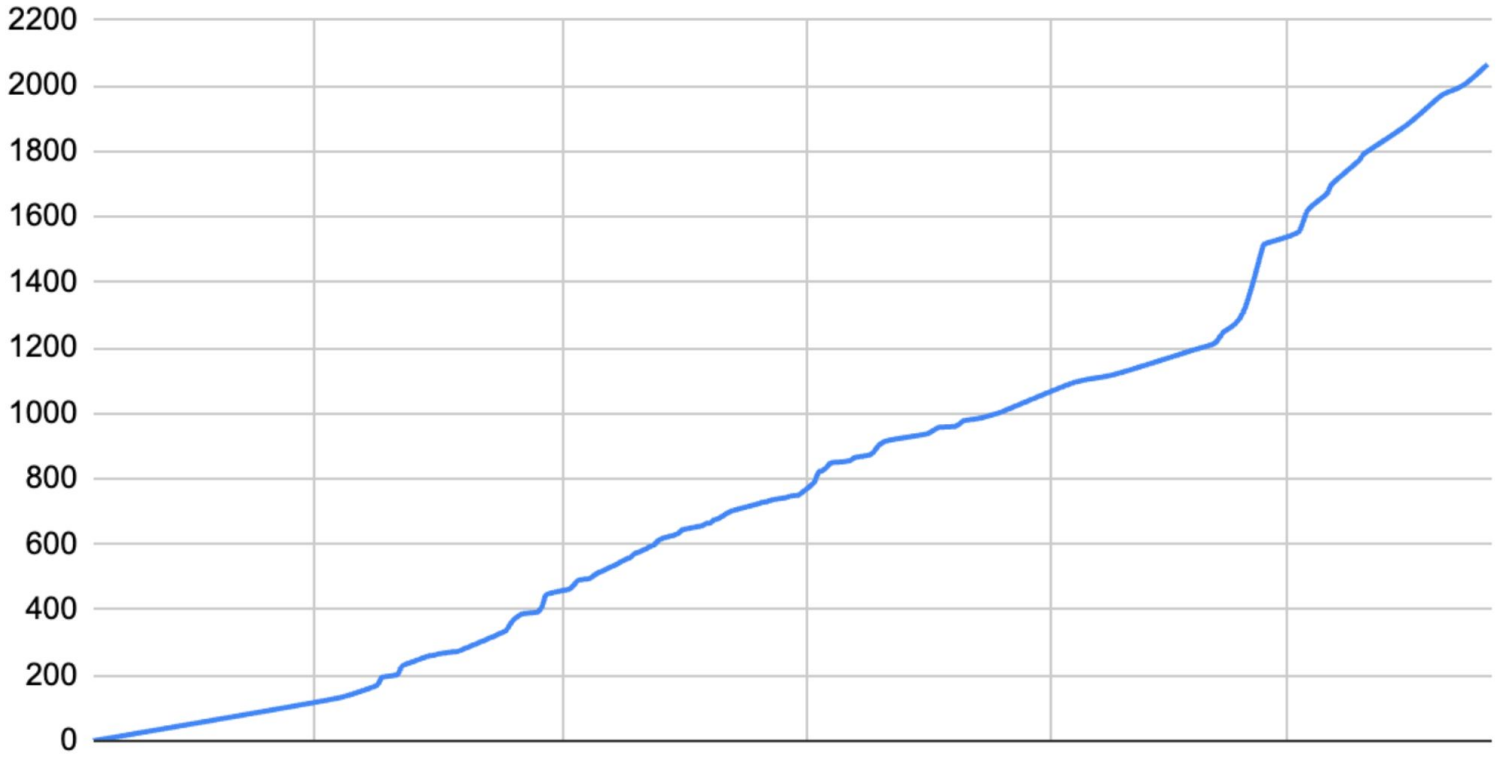
```
[
  {
    "type": "SET", # or DELETE
    "path": ["data", "inbox", "emojiReactions"],
    "value": [
      {
        "id": "RW1vamlSZWFjdGlvbjoxMDI=",
        "emoji": "👍",
        "count": 12
      },
      {
        "id": "RW1vamlSZWFjdGlvbjoxMDM=",
        "emoji": "❤️",
        "count": 7
      },
      {
        "id": "RW1vamlSZWFjdGlvbjoxMDQ=",
        "emoji": "😂",
        "count": 3
      }
    ]
  }
]
```

Mocks are guaranteed to stay in sync over time

We covered

- ✓ The Pains of GraphQL Data Mocking
- ✓ @generateMock: Schema + Context + LLMs = Magic
- ✓ @respondWithMock: Unblocking Client Development
- ✓ Schema Evolution: Keeping Mocks Truthful

2000+ mocks generated



“@generateMock has significantly sped up my local development and made working with local data much more enjoyable.”

— Senior Software Engineer

In the GraphQL Community

add `@mock` spec #10

Open magicmark wants to merge 23 commits into `graphql:main` from `magicmark:mock_spec`

Conversation 58 | Commits 23 | Checks 1 | Files changed 4

magicmark commented on Mar 8 • edited — Contributor

see graphql/ai-wg#79

- 🚀 **Rendered Spec:** https://public.larah.me/~mark/MockSpec.html
- ✨ **Demo:** https://mock-spec-demo.larah.me/

Prior art! https://medium.com/airbnb-engineering/graphql-data-mocking-at-scale-with-llms-and-generatemock-30b380f12bd6

gaps.graphql.org/GAP-10

Thank you!