RoomSlice GraphQL API Guide

REQUESTS

USERS

Sign Up:

```
mutation {
  createUser(
    email:<email>,
    password: <password>,
    firstName: <first name>,
    lastName: <last name>
  ) {
    user {
        id
    }
  }
}
```

Sign In:

```
mutation {
    tokenAuth(email:<email>, password:<password>) {
        token
    }
}
```

Check if in household

- Token in header
- household is null if they aren't in one yet

```
query {
    me {
        household {
            id
        }
    }
}
```

Get Homepage info

- Token in header

```
query {
    homepage {
       ... on HouseholdType {
           name
       }

       ... on UserType{
           firstName
           status
       }
```

```
    }
}
```

Returns list in this order:

- household name
- logged-in user info
- user's roommates' info

Example of return

```
{
    "data": {
        "homepage": [
            {
                "name": "Household Name"
            },
            {
                "firstName": "Josh",
                "status": "sleeping"
            },
            {
                "firstName": "Jay",
                "status": "coding"
            },
            {
                "firstName": "Luis",
                "status": "AC:NH"
            }
        ]
    }
}
```

Update User Status

- Token in header

```
mutation {
    updateUser (userData: {status : <new status>}) {
        user {
            firstName
            status
        }
    }
}
```

Add user to a Household

- Token in header

```
mutation {
    updateUser (userData: {household: <household id>}) {
        user {
            household{
                id
            }
        }
    }
}
```

Delete User Account

- Token in header

```
mutation {
    deleteUser(email:"test1@test.com") {
            ok
    }
}
```

HOUSEHOLDS

Create a Household

- Token in header
- Adds logged-in user to Household
    - Load Home page after created

```
mutation {
    createHousehold(name: "New House") {
        household{
            id
        }
    }
}
```

Change Household name

- Token in header

```
mutation{
    updateHousehold(name : "New Name") {
        household {
            name
        }
    }
}
```

Delete a Household

```
mutation {
    deleteHousehold(hId: <household id>) {
        ok
    }
}
```

FREQUENCY tag: (for Tasks and Bills)

- `<code><number>`

Code	==
D	days

`W`	`weeks`
`M`	`months`
`Y`	`years`
`X0`	`none`

E.g.

- `"D2"` - repeat every 2 days
- `"W1"` - repeat every week
- `"M3"` - repeat every 3 months
- `"D38"` - repeat every 38 days
- `"X0"` - no repetition, one-time task/bill

DATE format

- When creating/updating a task/bill, use format: `"DDMMYYYY"`
  - `"21042020" == 4/21/2020`
- When request returns a dueDate, it's in format `"YYYY-MM-DD"`
  - `"2020-04-21"`

TASKS

Create Task

- Token in header

```
mutation {
    createTask(
        name: "Sweep floor",
        description: "use big broom",
        dueDate: "DDMMYYYY",
        frequency:"1W",
        current: 3,
        rotation: [3, 4, 5]) {
        task{
            name
            description
            dueDate
            frequency
            current {
                firstName
            }
            rotation{
                firstName
            }
        }
    }
}
```

Edit Task

- Token in header

```
mutation{
    updateTask(taskData: {taskId: <task id>, name: <new name>}) {
        task {
```

```
            name
            description
            dueDate
            frequency
            current{
                firstName
            }
            complete
            rotation{
                firstName
            }
        }
    }
}
```

- For editing **rotation**, specify if adding/removing and provide IDs
    - `taskData: {taskId: 1, addRotation: [1, 4]}`
        - Adds users 1 and 4 to rotation
    - `taskData: {taskId: 1, removeRotation: [3]}`
        - Removes user 3 from rotation

Complete a Task

- Adds task to completed tasks list
- If there's a rotation, assigns current to next roommate in list
- Sets next dueDate according to frequency
    - If frequency == "X0", deletes task

```
mutation{
    updateTask(taskData: {taskId: <id>, complete: true}) {
```

```
        task {
            id
        }
    }
}
```

Returns task id, but not needed

- Reload Task page after this is called to move task from "My Tasks" to "Completed Tasks"

Delete Task

```
mutation {
    deleteTask(taskId: 2) {
        ok
    }
}
```

Get Task Page data

- Token in header

```
query {
    tasks {
        id
        name
        description
        dueDate
```

```
        current
        {
            firstName
        }
        complete
        rotation {
            firstName
        }
    }
}
```

Returns: 2-d list with 2 lists of tasks

- First entry is logged-in user's tasks, second entry is other household tasks

Example of return

```
{
    "data": {
        "tasks": [
            [
                {
                    "id": 3,
                    "name": "Take out trash",
                    "description": "in back alley",
                    "dueDate": "2020-04-27",
                    "frequency: "D1",
                    "current": {
                        "firstName": "Josh"
                    },
                    "complete": false,
                    "rotation": [
                        {
```

```
                "firstName": "Luis"
            },
            {
                "firstName": "Josh"
            },
            {
                "firstName": "Jay"
            }
        ]
    },
],
[
    {
        "id": 1,
        "name": "Clean kitchen",
        "description": "stove, counters, table",
        "dueDate": "2020-06-09",
        "frequency": "W2",
        "current": {
            "firstName": "Jay"
        },
        "complete": false,
        "rotation": [
            {
                "firstName": "Josh"
            },
            {
                "firstName": "Luis"
            },
            {
                "firstName": "Jay"
```

```
                    }
                ]
            },
            {
                "id": 4,
                "name": "Cut grass",
                "description": "Lawnmower",
                "dueDate": "2020-04-30",
                "frequency": "M1",
                "current": {
                    "firstName": "Luis"
                },
                "complete": false,
                "rotation": [
                    {
                        "firstName": "Luis"
                    },
                    {
                        "firstName": "Josh"
                    },
                        "firstName": "Jay"
                    }
                ]
            }
        ]
    ]
}
}
```

Get Completed Tasks data

- Token in header
- Ordered by date (newest completed first)

```
query {
    completeTasks {
        name
        roommate {
            firstName
        }
        date

    }
}
```

BILLS

Create Bill

- Token in header
- Logged-in user assigned as Bill manager

- Creates Bill object that can be activated/deactivated each time there is a cycle, isActive set to false by default

```
mutation {
    createBill (name: <name>, dueDate: "30042020", frequency:<tag>, participants: [3, 4, 5]) {
        bill {
            name
            id
            manager {
                firstName
            }
            dueDate
            frequency
            isActive
            totalBalance
            participants {
                firstName
            }
        }
    }
}
```

Activate Bill (manager)

- Token not required
- Creates BillCycle objects for each participant with their share of the total balance
- Sets isActive to true

```
mutation {
```

```
    updateBill(billData: {billId: <id>, totalBalance: "63.89", isActive: true}){
        bill {
            name
            id
            manager {
                firstName
            }
            dueDate
            frequency
            totalBalance
            isActive
            numSplit
            participants {
                firstName
            }
            cycles {
                recipient {
                    firstName
              }
              amount
              isPaid
            }
        }
    }
}
```

Update Bill

- Token not required

- Arguments in billData & return fields
    - Only include what you need to update/see

- For editing **participants** of bill, specify if adding/removing and provide roommate IDs
    - `billData: {billId: 1, addParticipants: [1, 4]}`
        - Adds users 1 and 4 to rotation
    - `billData: {billId: 1, removeParticipants: [3]}`
        - Removes user 3 from rotation

```
mutation {
    updateBill(billData: {billId: <id>, name: <new name>, totalBalance: "999.9"}){
        bill {
          name
          id
          manager{
              firstName
          }
          dueDate
          frequency
          totalBalance
          isActive
          numSplit
          participants {
              firstName
          }
          cycles {
              recipient {
                  firstName
              }
              amount
              isPaid
```

```
            datePaid
          }

        }
    }
}
```

Delete Bill

- Token not required
- Returns ok, true if successful, false otherwise

```
mutation {
    deleteBill(billId: 1) {
        ok
    }
}
```

Get Bills Page data

- Token in header
- **Returns** two lists
    a. List of Bills that logged-in user is managing
    b. List of BillCycles that logged-in user hasn't paid yet to other roommates
        - Recipient
- "data" refers to each list

```
query {
    bills {
        ... on BillListType {
            data {
                name
                totalBalance
                isActive
                cycles {
                    recipient {
                        firstName
                    }
                isPaid
                }
            }
        }
        ... on CycleListType {
            data {
                bill {
                    name
                    manager {
                        firstName
                    }
                    dueDate
                }
            amount
            }
        }
    }
}
```

Example of return (with Josh logged-in):


```
{
    "data": {
        "bills": [
            {
                "data": [
                    {
                        "name": "Electricity",
                        "totalBalance": 57.12,
                        "isActive": true,
                        "cycles": [
                            {
                                "recipient": {
                                    "firstName": "Jay"
                                },
                                "isPaid": true
                            },
                            {
                                "recipient": {
                                    "firstName": "Luis"
                                },
                                "isPaid": false
                            }
                        ]
                    }
                ]
            },
            {
```

```
                "data": [
                    {
                        "bill": {
                            "name": "Gas",
                            "manager": {
                                "firstName": "Luis"
                            },
                            "dueDate": "2020-04-30"
                        },
                        "amount": 10.68
                    },
                    {
                        "bill": {
                            "name": "Water",
                            "manager": {
                                "firstName": "Jay"
                            },
                            "dueDate": "2020-04-30"
                        },
                        "amount": 5.42
                    }
                ]
            }
        ]
    }
}
```


Pay Bill Cycle

- Token in header
- Recipient of a bill can pay it to mark that BillCycle as paid

```
mutation{
    payBillCycle(billId: <id>) {
        cycle {
              recipient {
                  firstName
              }
              amount
              isPaid
              datePaid
        }
    }
}
```

Mark Bill as complete (manager)

- Token in header
- Bill Manager can mark their bill as complete when they see all participants have paid
- Updates Bill due date based on the frequency, sets isActive to false, sets totalBalance to 0.00
    - Bill can now be reactivated when next cycle occurs

```
mutation {
    updateBill(billData: {billId: <id>, isActive: false}) {
        bill {
            name
            manager {
```

```
                firstName
            }
            isActive
        }
    }
}
```

Get Bill History page data

- Token in header
- Returns list of all completed BillCycles in the household of logged-in user, sorted by datePaid (newest first)

```
query {
    completeBills {
        recipient{
            firstName
        }
        amount
        datePaid
        bill {
            name
            manager {
                firstName
            }
        }
    }
}
```

DEVELOPER

See all Users in database

```
query{
    users {
      id
      email
      firstName
      lastName
      status
      household {
          id
          name
      }
    }
}
```

See all Households in the database

```
query{
    households {
      id
      name
      users {
          id
```

```
            firstName
            lastName
            email
            status
        }
        tasks {
            id
            name
            description
            frequency
            dueDate
            current {
                firstName
            }
            rotation {
                firstName
            }
        }
        completeTasks {
            id
            name
            roommate {
                firstName
            }
            date
        }

    }
}
```