17625 A2 Part 1

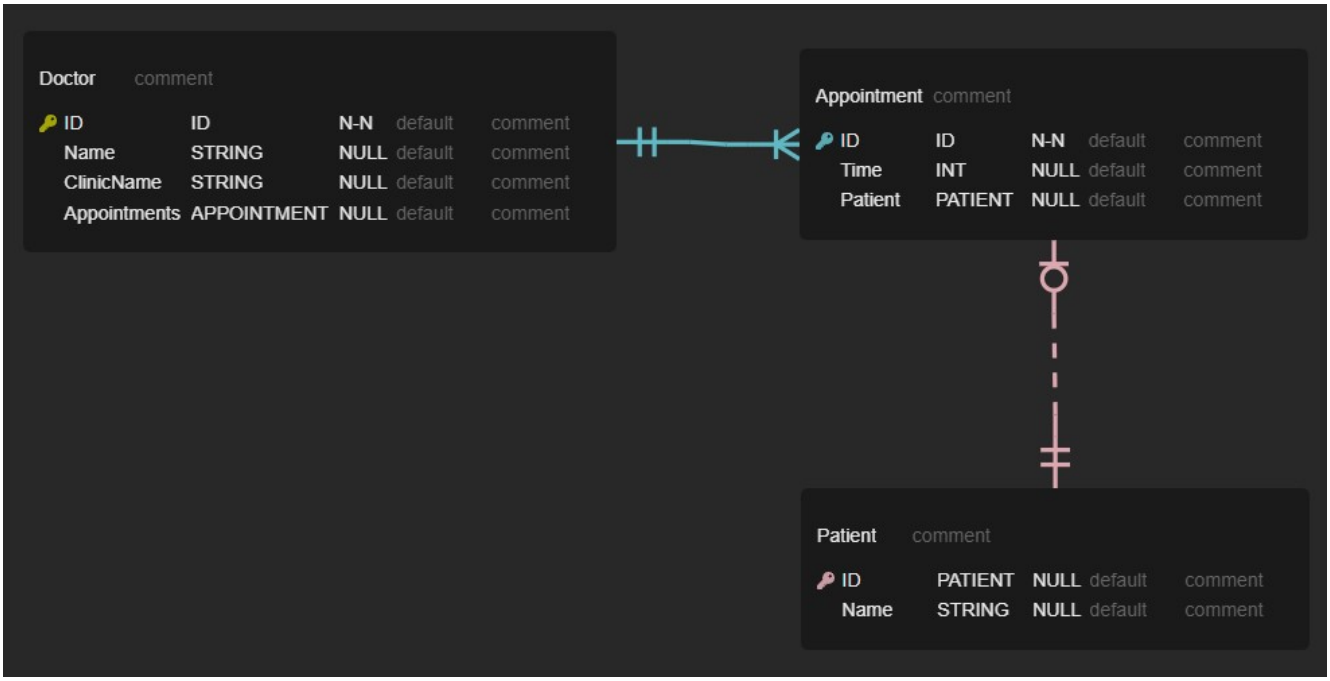
by Yucen Xu

API Design Task

1.1 Design subtask

1.1.1 Schemas

The schema for the API is as follows.




```
type Appointment {
  id: ID!
  time: Int
  patient: String
  doctor: Doctor
  patient: Patient
}

type Doctor {
  id: ID!
  name: String
  clinicName: String
  appointments: String
  appointmentList: [Appointment!]!
}

type Patient {
  name: String
  appointment: Appointment
}
```

1.1.2 Queries

a. Get Doctor Details

Description: Get name, clinic of a doctor.

Input:

```
type Query{
  doctorByID(uid: ID): Doctor
}
```

Output:

```
{
  doctor
  {
    name
    clinic name
    appoitments {
        id
        time
        patient {
            id
            name
        }
    }
  }
}
```

b. Get Available Timeslots

Description: Get doctor's available timeslots for current day. Timeslot is a number from 1 to 48 (each number represents a 30 min slot, starting from 12 am)

Input:

```
type Query{
  timeslotByID(uid: ID): [int]
}
```

Output:

```
{
  timeslots
}
```

1.1.3 Mutations

a. Add Appointment

Description: Book an appointment with doctor for today

Input:

```
type Mutation{
  createAppointment(input: CreateAppointmentInput!): ID
}

input CreateAppointmentInput {
    doctorID: ID!
    patient: Patient
}
```

Output:

```
{
  appointmentID
}
```

b. Cancel Appointment

Description: Cancel an appointment

Input:

```
type Mutation{
  cancelAppointment(input: CancelAppointmentInput!): ID
}

input CancelAppointmentInput {
    appointmentID: ID!
}
```

Output:

```
{
  appointmentID
}
```

c. Update Appointment

Description: Update name of the patient for an appointment

Input:

```
type Mutation{
  updateAppointment(input: UpdateAppointmentInput!): ID
}

input UpdateAppointmentInput {
    appointmentID: ID!
    patientName: String
}
```

Output:

```
{
  appointmentID
```

```
}
```

1.1.3 Endpoints

There will be only one endpoint for the system, since in GraphQL, the query is based on types. For local test, the endpoint will be localhost:8000.

1.2 Testcases

Identifier	Description	Inputs	Expected Output	Remarks

Get_Doctor_Success	Get details of a doctor with valid ID	Query {
		doctorByID("D_1")
		}
			{
			"data":{
			"name":"doctor_xxx"
			"clinic name":"xxx"
			"appointments":[
			{"id":"A_1"
			"time":16
			"patient":{
			"id":"P_1"
			"name":"patient_xxx"}
			}
			]
			}
			}	

Get_Doctor_Fail	Get details of a doctor with invalid ID	Query {
		doctorByID(null)
		}
			{
			"message":"Invalid Parameter"
			}
				Server should handle error

Get_Time_Success	Get available timeslots of a doctor with valid ID	Query {
		doctorByID("D_1")
		}
			{
			"data":{
			"timeslots":[1,2,3]
			}
			}	

Get_Time_Fail	Get available timeslots of a doctor with invalid ID	Query {
		doctorByID(null)
		}
			{
			"message":"Invalid Parameter"
			}
				Server should handle error

Add_Appointment_Success	Add appointment with valid docter ID and input	Mutation {
		CreateAppointment("D_1", "P_1")
		}
			{
			"data":{
			"appointment":"A_1"
			}
			}	

Add_Appointment_Fail	Add appointment with invalid input	Mutation {
		CreateAppointment(null)
		}
			{
			"message":"Invalid Parameter"
			}	

Identifier	Description	Inputs	Expected Output	Remarks
Cancel_Appointment_Success	Cancel appointment with valid appointment ID	Mutation { CancelAppointment("A_1") }	{ "data":{ "appointment":"A_1" } }	
Cancel_Appointment_Fail	Cancel appointment with invalid appointment ID	Mutation { CancelAppointment(null) }	{ "message":"Invalid Parameter" }	
Update_Appointment_Success	Update appointment with valid appointment ID and input	Mutation { UpdateAppointment("A_1","Patient_Newname") }	{ "data":{ "appointment":"A_1" } }	
Update_Appointment_Fail	Update appointment with invalid input	Mutation { UpdateAppointment(null) }	{ "message":"Invalid Parameter" }	