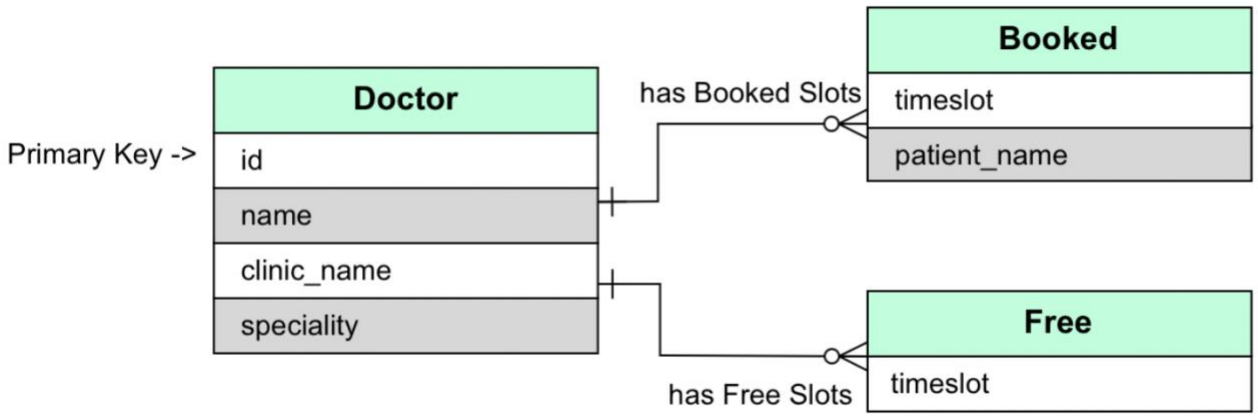
API Design
Homework A2

API Design Task

1.1 Design Subtask:

Schema of the GraphQL API:



Queries:

```
type Query {
    #Query to get details of all the doctors
    doctors: [Doctor!]!
    #Query to get the doctors details
    doctor(id: ID!): Doctor
    #Query to get the free slots of doctor
    doctor_freeslots(id: ID!): Doctor
}
```

Query 1:
Name: doctors
Input: null
Output: List of all the doctors and their details.
Description: This API is designed to check the details of all the doctors present in the DoctorList.

Query 2: (Capability 1 for the assignment)
Name: doctor
Input: id of type ID
Output: doctor details(name, clinic_name, speciality) of a particular "id" that is sent as input.
Description: This API is designed to get the details of doctor based on the id. Doctor details include name, clinic_name and speciality.
Query 3: (Capability 2)
Name: doctor_freeslots
Input: id of type ID
Output: doctors available free timeslots for a particular "id" that is given as input.
Description: This API is designed to get the details of free time slots of a doctor. Doctor is selected based on the input "id" that is passed as argument.

Mutations:

```
type Mutation {
    #Mutation to add a doctor
    create_doctor(input: CreateDoctor): Doctor
    #Mutation to schedule an appointment
    schedule_appointment(id: ID!, input: ScheduleAppointment): Doctor
    #Mutation to update the name of a patient
    update_appointment(oldName: String, newName: String): Doctor
    #Mutation to cancel an appointment
    cancel_appointment(id: ID!, patient_name : String!): Doctor
}
```

Mutation 1:
Name: create_doctor
Input: "id" of type ID, "name" of type String, "clinic_name" of type String and "speciality" of type String
Output: Doctor with the variables id, name, clinic_name and speciality that are passed in the input are pushed to the DoctorList and Output is the new doctor details that are created with variables passed.
Description: This API is designed to add new doctor into the DoctorList with basic details.

Mutation 2: (Capability 3)
Name: schedule_appointment
Input: "id" of type ID, "timeslot" of type String and "patient_name" of type String
Output: Doctor details and booked timeslot details of the ID sent as input argument
Description: This API is designed to schedule a new appointment, eventually adding a new object in booked with timeslot and patient_name as passed in the input.

Mutation 3: (Capability 4)
Name: cancel_appointment
Input: "id" of type ID and "patient_name" of type String

Output: Updated doctor details with booked timeslots which would not have the cancelled appointment.
Description: This API is designed to cancel a scheduled appointment and return the updated doctor details.

Mutation 4: (Capability 5)
Name: update_appointment
Input: “oldName” of type String and “newname” of type String
Output: Updated doctor details with booked timeslots which inturn will have new patient_name and timeslot
Description: This API is designed to update the name of a patient in scheduled appointments

All the capabilities that are stated in the assignment are divided into 5 different APIs, which are described above with additional APIs for the ease of usage.

Endpoint of the GraphQL API: http://localhost:4000/

API Design
Homework A2

1.2 Testcases design subtask:

Testcase Identifier	Testcase Description	Inputs	Expected Output	Remarks
DoctorByIdHappyTest	Tests the doctor API which returns details of doctor based on ID by passing an existing id	id	Doctor details: id name clinic_name speciality	Details of the doctor with the given id are returned and tested
DoctorByIdErrorTest	Tests the doctor API which returns details of doctor based on ID by passing a non-existing id	id	Error	As the id is not present in the doctors list, the test throws an error

DoctorFreeSlotsHappyTest	Tests the doctor_freeslots API which returns the freeslots of a doctor by passing an existing id	id	Doctor details:
			Id
			name
			timeslots_available{
			timeslot
			patient_name
			}
				Details of the doctor and freeslots with the given id are returned and tested

DoctorFreeSlotsErrorTest	Tests the doctor_freeslots API which returns the freeslots of a doctor by passing a non-existing id	id	Error	As the id is not present in the doctors list, the test throws an error

ScheduleAppointmentHTest	Tests the schedule_ appointment API which adds a new appointment to the existing id	Id
		timeslot
		patient_name
			Doctor details:
			Id
			name
			timeslots_available{
			timeslot
			patient_name
			}
				Details of the doctor with the given id are returned with new appointment in the list

ScheduleAppointmentETest	Tests the schedule_ appointment API which adds a new appointment to the non-existing id	Id timeslot patient_name	Error	As the id is not present in the doctors list, the test throws an error

API Design
Homework A2

CancelAppointmentHTest	Tests the cancel_appointment API which deletes a scheduled appointment of the existing id	Id
		patient_name
			Doctor details:
			Id
			name
			timeslots_available{
			timeslot
			patient_name
			}
				Details of the doctor with the given id are returned with no cancelled appointment in the list

CacnelAppointmentETest	Tests the cancel_appointment API which deletes a scheduled appointment of the non-existing id	Id patient_name	Error	As the id is not present in the doctors list, the test throws an error

UpdateAppointmentHTest	Tests the update_appointment API which updates the name of an existing patient in booked appointments to a new name	old_name
		new_name
			Doctor details:
			Id
			name
			timeslots_available{
			timeslot
			patient_name
			}
				Details of the doctor with the given id are returned with modified appointments patient name in the list

UpdateAppointmentETest	Tests the update_appointment API which updates the name of a non-existing patient in booked appointments to a new name	old_name new_name	Error	As the id is not present in the doctors list, the test throws an error

For each of the queries and mutations designed to meet the capabilities, a happy test and an error test are defined above.