

Project2-week3

# Hospital Appointment

13.05.2024

Jingwen Li  
Youzhen Duan

# C O N T E N T S

01

---

Description

02

---

Tasks

03

---

Diagrams

04

---

Implemented  
functions

05

---

Demo

# Description

Part .01



## Description

1. During an appointment, bill of a patient is being generated. This bill will somehow influence patient's balance of insurance.
2. If a patient has insurance, then the insurance company will pay their medical cost according to their claim rate. If not, then the patient will pay all by herself or himself.

# Tasks

Part .02



## Tasks

1 generate bills

-Youzhen Duan

2 insurance claims

-Jingwen Li



## Previous tasks-Week1&2

1-1 register a new patient

-Jingwen Li

1-2 make an appointment

-Youzhen Duan

2-1 record prescriptions

-Jingwen Li

2-2 update medical records

-Youzhen Duan

2-3 retrieve patient's medical history

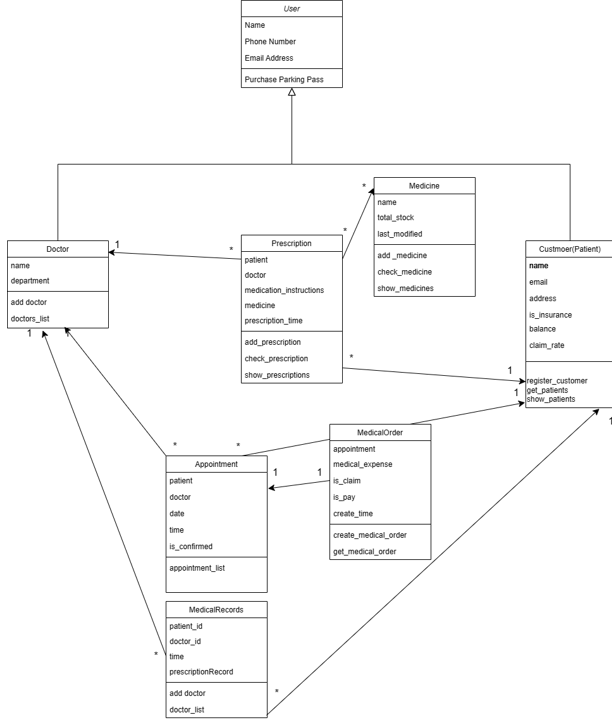
-Youzhen Duan

# Diagrams

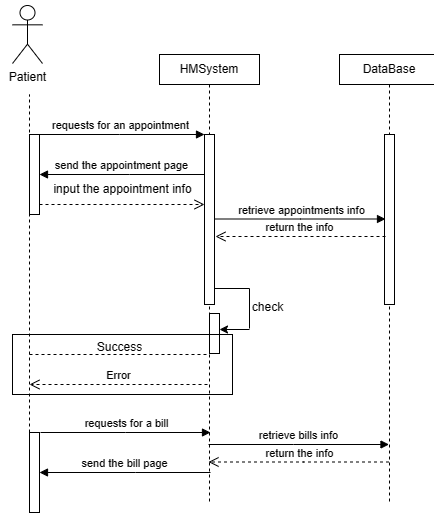
Part .03



# Class Diagram



# Sequence Diagram



# Implemented functions

Part . 04

# Implemented functions

createMedicalOrder

The screenshot displays a REST client interface with a POST request to `http://127.0.0.1:8000/updateMedicalOrder/`. The request body is raw text containing JSON data. The response status is 200 OK, and the response body is formatted JSON.

**Request:**

```
POST http://127.0.0.1:8000/updateMedicalOrder/

{
  "appointment_id": 6,
  "medical_expense": 100.0
}
```

**Response:**

```
{
  "code": 200,
  "message": "created successfully",
  "data": {
    "appointment_id": 6,
    "medical_expense": 100.0
  }
}
```

# Implemented functions

## getMedicalOrderByAppointmentId

The screenshot displays a REST client interface with the following details:

- Method:** GET
- URL:** http://127.0.0.1:8000/getMedicalOrderByAppointmentId/5/
- Buttons:** Send, Cookies
- Tabs:** Params, Authorization, Headers (7), Body, Scripts, Tests, Settings
- Query Params:** A table with 4 columns: Key, Value, Description, and Bulk Edit. The first row contains 'Key', 'Value', 'Description', and 'Bulk Edit'. The second row contains 'Key', 'Value', 'Description', and 'Bulk Edit'.
- Body Tab:** Status: 200 OK, Time: 18 ms, Size: 414 B, Save as example
- Response Format:** JSON
- Response Content:**

```
1  [
2    {
3      "id": 1,
4      "appointment_id": 5,
5      "medical_expense": 100.0,
6      "is_claim": false,
7      "is_pay": true,
8      "create_time": "2024-05-13T00:43:21.204271Z"
9    }
10 ]
```

## Implemented functions-pay

pay with claim\_tate=0 successfully and  
check balance

```
1 [
2   {
3     "id": 1,
4     "name": "Peter",
5     "email": "peter@163.com",
6     "address": "Shanghai",
7     "is_insurance": true,
8     "balance": 100,
9     "claim_rate": 0
10  },
11 ]
```

The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:8000/pay/`. The request body is a JSON object with `customer_id: 1` and `order_id: 4`. The response status is `200 OK` with a message: `"Your balance is 100.0."`.

**Request:**

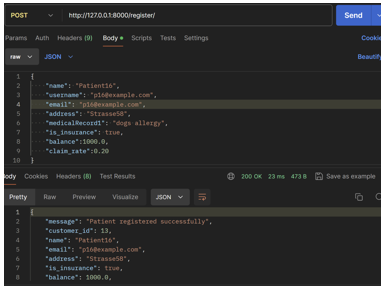
```
1 {
2   "customer_id": 1,
3   "order_id": 4
4 }
5 }
```

**Response:**

```
1 {
2   "code": 200,
3   "message": "Your balance is 100.0.",
4 }
```

## Implemented functions-pay

pay with claim\_tate=0.2 successfully  
and check balance



```
POST http://127.0.0.1:8000/register/

{
  "name": "Patient16",
  "username": "p16@example.com",
  "email": "p16@example.com",
  "address": "Strasse58",
  "medicalRecord1": "dogs allergy",
  "is_insurance": true,
  "balance": 1000.0,
  "claim_rate": 0.2
}

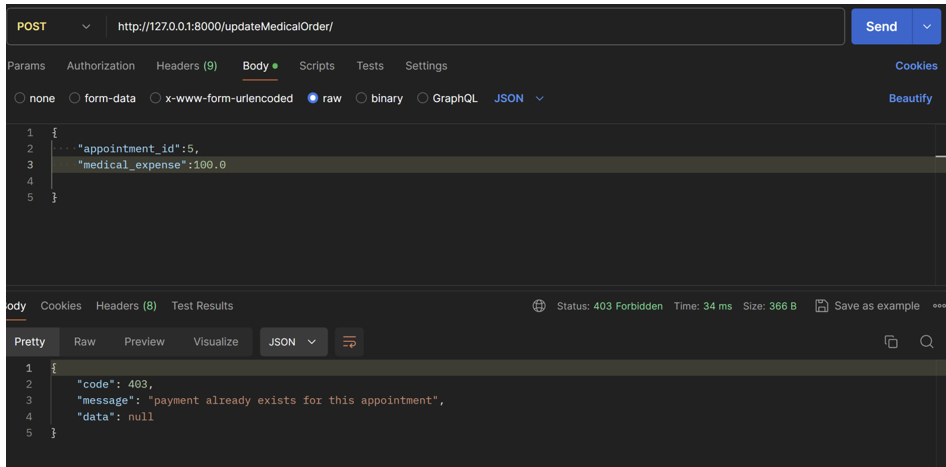
{
  "message": "Patient registered successfully",
  "customer_id": 13,
  "name": "Patient16",
  "email": "p16@example.com",
  "address": "Strasse58",
  "is_insurance": true,
  "balance": 1000.0
}
```

```
balance -= (1 - claim_rate)* expense
```

```
{
  "id": 10,
  "patient": {
    "id": 13,
    "name": "Patient16",
    "email": "p16@example.com",
    "address": "Strasse58",
    "is_insurance": true,
    "balance": 840,
    "claim_rate": 0.2
  },
  "doctor": {
    "id": 9,
    "name": "Doctor9",
    "department": "department"
  },
  "date": "28-03-2024",
  "time": "12:00",
  "is_confirmed": true
}
```

# Implemented functions-pay

payment has already made



The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:8000/updateMedicalOrder/`. The request body is a JSON object: `{ "appointment_id": 5, "medical_expense": 100.0 }`. The response status is `403 Forbidden` with a message: `"payment already exists for this appointment"`. The response body is a JSON object: `{ "code": 403, "message": "payment already exists for this appointment", "data": null }`.

**Request:**

```
POST http://127.0.0.1:8000/updateMedicalOrder/

{
  "appointment_id": 5,
  "medical_expense": 100.0
}
```

**Response:**

```
Status: 403 Forbidden Time: 34 ms Size: 366 B Save as example

{
  "code": 403,
  "message": "payment already exists for this appointment",
  "data": null
}
```



# Implemented functions-pay

pay failed

The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:8000/pay/`. The request body is a JSON object: `{ "customer_id": 9, "order_id": 3 }`. The response status is `200 OK` with a time of `15 ms` and a size of `337 B`. The response body is a JSON object: `{ "code": 502, "message": "insufficient balance!", "data": null }`.

POST `http://127.0.0.1:8000/pay/` Send

Params Authorization Headers (9) Body Scripts Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

```
1 {
2   "customer_id": 9,
3   "order_id": 3
4 }
5
```

Body Cookies Headers (8) Test Results Status: 200 OK Time: 15 ms Size: 337 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "code": 502,
3   "message": "insufficient balance!",
4   "data": null
5 }
```

# Demo

Part .05