# COMP 353
# Main Project Report

**xvc 353_4**

TianMing Chen 40058543
Zirui Qiu 40050008
Kaixin Dai 40070072
Jingyi Lin 40044719

## E/R diagram:

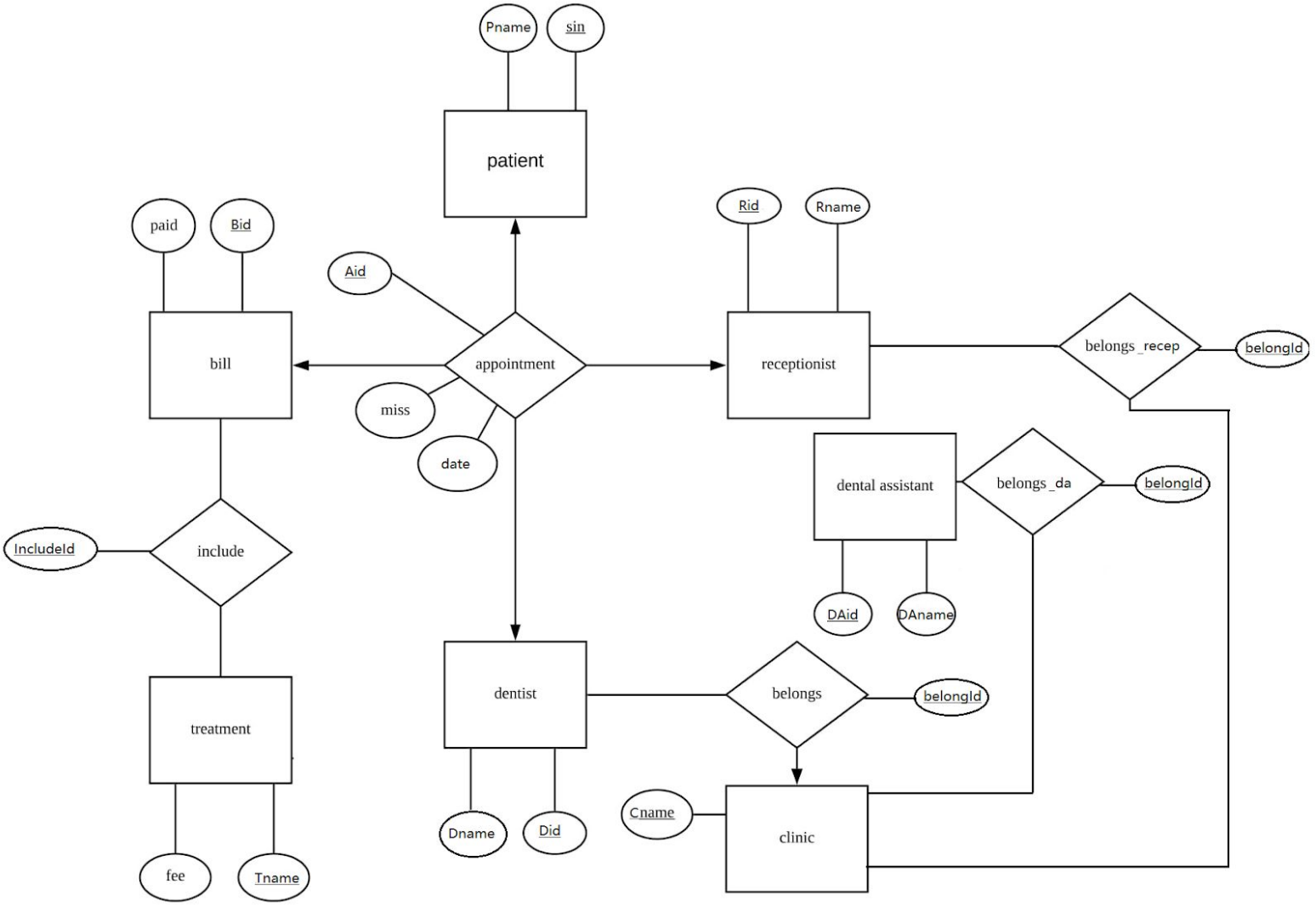



Figure 1: ER diagram of the system

## Database Schemas:

clinic(Cname)

belongs(belongsId,Cname*, Did*)

belongs_da(belongsId,Cname*, DAid*)

belongs_recep(belongsId,Cname*, Rid*)

dentist(Did, Dname)

dental_assistant(DAid, DAname)

receptionist(Rid, Rname)

patient (Pname, Sin)

bill (Bid, paid)

appointment(Aid,Sin*, Did*, Rid*, Bid*, miss, data)

treatment(Tname, fee)

include(included,Bid*, Tname*)

## Functional Dependencies:

**patient:**
Sin→Pname

**appointment:**
Aid→Bid, SIN, Rid,Did

**bill:**
Bid →paid

**include:**
includeId→name, Bid

**treatment:**
name→fee

**receptionist:**
Rid→Rname

**dentist:**
Did→Dname

**belongs:**
belongsId→Cname,Did

**belongs_da:**

belongsId→Cname,DAid

**belongs_recep:**

belongsId→Cname,Rid

**dental assistant:**
DAid→DAname

## SQL declarations of the relations, the implementation code, relation instances

### Table structure

The following section represents the formation of the table through code, the added key and constraint is showing right under the table creation. The description of the table is also included for clarification. The auto_increment section is added right after all tables.

### table appointment

The appointment table includes its own primary key called Aid and several needed foreign key such as Sin from patient, Did (dentist ID) from dentist, Rid from receptionist, Bid from bill. The data miss represent if the patient attended for this appointment or not in the form of “boolean”, 0 for false, and 1 for true.

```
CREATE TABLE `appointment` (
  `Aid` int(4) NOT NULL,
  `Sin` int(4) NOT NULL,
  `Did` int(4) NOT NULL,
  `Rid` int(4) NOT NULL,
  `Bid` int(4) NOT NULL,
  `miss` tinyint(1) NOT NULL,
  `date` date NOT NULL
);

ALTER TABLE `appointment`
  ADD PRIMARY KEY (`Aid`),
  ADD KEY `Did` (`Did`),
  ADD KEY `Rid` (`Rid`),
  ADD KEY `Sin` (`Sin`),
  ADD KEY `Bid` (`Bid`);
```

```
ALTER TABLE `appointment`
  ADD CONSTRAINT `appointment_ibfk_1` FOREIGN KEY (`Bid`) REFERENCES
`bill` (`Bid`),
  ADD CONSTRAINT `appointment_ibfk_2` FOREIGN KEY (`Did`) REFERENCES
`dentist` (`Did`),
  ADD CONSTRAINT `appointment_ibfk_3` FOREIGN KEY (`Rid`) REFERENCES
`receptionist` (`Rid`),
  ADD CONSTRAINT `appointment_ibfk_4` FOREIGN KEY (`Sin`) REFERENCES
`patient` (`Sin`);
```

---

### table belongs

Table belongs represents the relation between dentist and clinic, in which dentist belongs to the clinic. It has the primary key BelongId. Cname is the clinic name from table clinic. Did is the dentist ID from table dentists.

```
CREATE TABLE `belongs` (
  `BelongId` int(11) NOT NULL,
  `Cname` varchar(50) NOT NULL,
  `Did` int(11) NOT NULL
);

ALTER TABLE `belongs`
  ADD PRIMARY KEY (`BelongId`),
  ADD KEY `Cname` (`Cname`),
  ADD KEY `Did` (`Did`);

ALTER TABLE `belongs`
  ADD CONSTRAINT `belongs_ibfk_1` FOREIGN KEY (`Cname`) REFERENCES
`clinic` (`Cname`),
  ADD CONSTRAINT `belongs_ibfk_2` FOREIGN KEY (`Did`) REFERENCES `dentist`
(`Did`);
```

---

### table belongs_da

Table belongs_da represents the relation between dentist assistant and clinic, in which dentist assistant belongs to the clinic. It has the primary key BelongId. Cname is the clinic name from table clinic. DAid is the dentist assistant ID from table dental_assistant.

```
CREATE TABLE `belongs_da` (
```

```
  `BelongId` int(11) NOT NULL,
  `Cname` varchar(50) NOT NULL,
  `DAid` int(11) NOT NULL
);

ALTER TABLE `belongs_da`
  ADD PRIMARY KEY (`BelongId`),
  ADD KEY `Cname` (`Cname`),
  ADD KEY `DAid` (`DAid`);

ALTER TABLE `belongs_da`
  ADD CONSTRAINT `belongs_da_ibfk_1` FOREIGN KEY (`Cname`) REFERENCES
`clinic` (`Cname`),
  ADD CONSTRAINT `belongs_da_ibfk_2` FOREIGN KEY (`DAid`) REFERENCES
`dental_assistant` (`DAid`);
```

---

### table belongs_recep

Table belongs_da represents the relation between receptionist and clinic, in which receptionist belongs to the clinic. It has the primary key BelongId. Cname is the clinic name from table clinic. Rid is the receptionist ID from the table receptionist.

```
CREATE TABLE `belongs_recep` (
  `BelongId` int(11) NOT NULL,
  `Cname` varchar(50) NOT NULL,
  `Rid` int(11) NOT NULL
);

ALTER TABLE `belongs_recep`
  ADD PRIMARY KEY (`BelongId`),
  ADD KEY `Cname` (`Cname`),
  ADD KEY `Rid` (`Rid`);

ALTER TABLE `belongs_recep`
  ADD CONSTRAINT `belongs_recep_ibfk_1` FOREIGN KEY (`Cname`) REFERENCES
`clinic` (`Cname`),
  ADD CONSTRAINT `belongs_recep_ibfk_2` FOREIGN KEY (`Rid`) REFERENCES
`receptionist` (`Rid`);
```

---

### table bill

Table bill represents the bill created after each appointment. Bid is the primary key for table bill and Paid represents “boolean” in which represents whether the bill is paid or not. The sum of bill will be calculated in query because a lot a treatment can be include to a single bill (more detail in the treatment table)

```
CREATE TABLE `bill` (
  `Bid` int(4) NOT NULL,
  `Paid` int(8) NOT NULL
);
ALTER TABLE `bill`
  ADD PRIMARY KEY (`Bid`);
```

---

#### table clinic

Table clinic is a simple table which the name of the clinic is represented using Cname. This table is mostly used for relation belongs, belongs_da and belongs_recep to see which person is a part of a given clinic.

```
CREATE TABLE `clinic` (
  `Cname` varchar(50) NOT NULL
);

ALTER TABLE `clinic`
  ADD PRIMARY KEY (`Cname`);
```

---

### table dental_assistant

Table dental_assistant represents the dental assistant, it has its own ID and a name.

```
CREATE TABLE `dental_assistant` (
  `DAid` int(4) NOT NULL,
  `DAname` varchar(50) NOT NULL
);

ALTER TABLE `dental_assistant`
  ADD PRIMARY KEY (`DAid`);
```

---

### table dentist

Table dentist represents the dentists, it has its own ID and a name.

```
CREATE TABLE `dentist` (
  `Did` int(4) NOT NULL,
  `Dname` varchar(50) NOT NULL
);

ALTER TABLE `dentist`
  ADD PRIMARY KEY (`Did`);
```

---

### table include

Table include represents the relation in which the treatment is included into a single bill. As we all know, a bill can include more than one treatment, therefore a many to one relation is needed in this case. The sum of the bill can be easily calculated by doing the sum of all treatment fees. The include consists of a Tname for treatment name and a Bid for bill ID.

```
CREATE TABLE `include` (
  `Tname` varchar(4) NOT NULL,
  `Bid` int(4) NOT NULL
);

ALTER TABLE `include`
  ADD PRIMARY KEY (`IncludeId`),
  ADD KEY `Tname` (`Tname`),
  ADD KEY `Bid` (`Bid`);

ALTER TABLE `include`
  ADD CONSTRAINT `include_ibfk_1` FOREIGN KEY (`Bid`) REFERENCES `bill`
(`Bid`),
  ADD CONSTRAINT `include_ibfk_2` FOREIGN KEY (`Tname`) REFERENCES
`treatment` (`Tname`);
COMMIT;
```

---

### table patient

Table patient is the instance of patient in which consists of Pname (patient name) and Sin (social insurance number) as primary key.

```
CREATE TABLE `patient` (
  `Pname` varchar(50) NOT NULL,
  `Sin` int(9) NOT NULL
);

ALTER TABLE `patient`
  ADD PRIMARY KEY (`Sin`);
```

---

### table receptionist

Table receptionist represents an instance of a receptionist who works at a clinic. It contains an ID, Rid as primary key and a name as Rname.

```
CREATE TABLE `receptionist` (
  `Rid` int(4) NOT NULL,
  `Rname` varchar(50) NOT NULL
);

ALTER TABLE `receptionist`
  ADD PRIMARY KEY (`Rid`);
```

---

### table treatment

Table treatment describes the name of the treatment and necessary fee. It includes TName as varchar and fee as int type.

```
CREATE TABLE `treatment` (
  `TName` varchar(50) NOT NULL,
  `fee` int(4) NOT NULL
);
```

```
ALTER TABLE `treatment`
  ADD PRIMARY KEY (`Tname`);
```

### -- Indexes for dumped tables

Include in table formation

---

## AUTO_INCREMENT

The following part contains the continuation of the table creation query. We decide to move this part after the main table creation query because it is repetitive because it will only apply for every primary key of each table.

```
ALTER TABLE `appointment`
  MODIFY `Aid` int(4) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=50;

ALTER TABLE `belongs`
  MODIFY `BelongId` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=15;

ALTER TABLE `belongs_da`
  MODIFY `BelongId` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

ALTER TABLE `belongs_recep`
  MODIFY `BelongId` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

ALTER TABLE `bill`
  MODIFY `Bid` int(4) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

ALTER TABLE `dental_assistant`
  MODIFY `DAid` int(4) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

ALTER TABLE `dentist`
  MODIFY `Did` int(4) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=8;

ALTER TABLE `include`
  MODIFY `IncludeId` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;

ALTER TABLE `receptionist`
  MODIFY `Rid` int(4) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;
```

**SQL scripts for the queries and transactions(Results show in Web Interface):**

The GUI is made of vanilla HTML for the frontend, and a php REST API for the backend. The following query represents the query we use to fulfill each requirement from a to g as well the extra required features.

All the query is running inside api/models/post.php
Please notice, the screenshot might be hard to read, please visit “our” website to see the result more clearly and play around the wonderful GUI we made.
The query will be represent in the plain text and how it looks like in the code (php)

### a) Get details of all dentists in all the clinics.

```
select *
from dentist, belongs
where belongs.Did = dentist.Did;
```

```
SELECT * FROM dentist, belongs WHERE belongs.Did = dentist.Did
```


| Did | Dname | Cname |
|---|---|---|
| 1 | Tom | clinic_1 |
| 2 | Jack | clinic_2 |
| 3 | Mary | clinic_3 |
| 4 | Sally | clinic_4 |
| 7 | Pony | clinic_5 |
| 2 | Jack | clinic_1 |

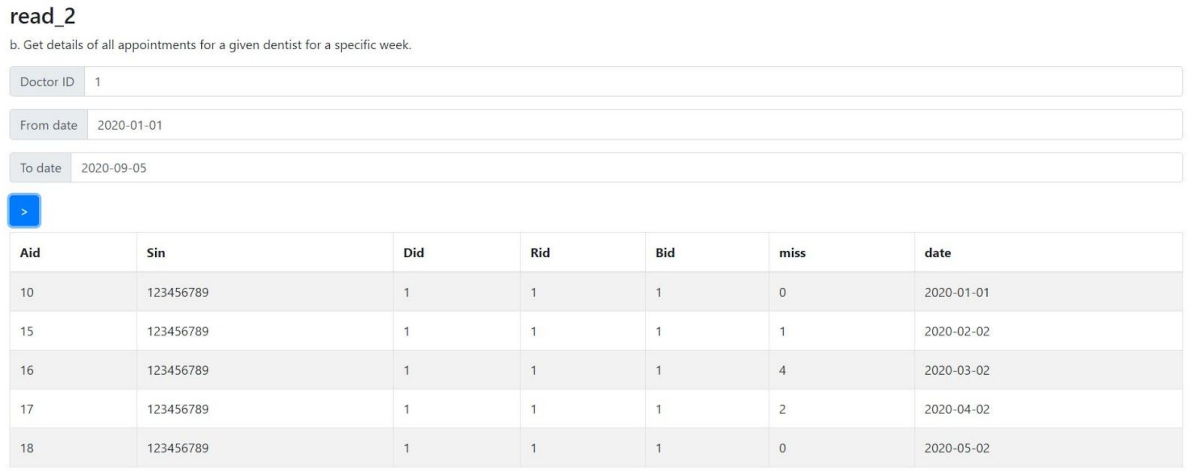
[Figure 2. all dentist displayed upon clicking on the button ]

### b) Get details of all appointments for a given dentist for a specific week.

```
select *
from appointment
where Did = "given dentist id"
```

```
AND data >= "startDate"
AND data <= "endDate" ;
```

```
SELECT *
                FROM appointment
                WHERE Did = ' . $Did . '
                AND date >= \'' . $d1 . '\'' . '
                AND date <= \'' . $d2 .'\'
```




| Aid | Sin | Did | Rid | Bid | miss | date |
|---|---|---|---|---|---|---|
| 10 | 123456789 | 1 | 1 | 1 | 0 | 2020-01-01 |
| 15 | 123456789 | 1 | 1 | 1 | 1 | 2020-02-02 |
| 16 | 123456789 | 1 | 1 | 1 | 4 | 2020-03-02 |
| 17 | 123456789 | 1 | 1 | 1 | 2 | 2020-04-02 |
| 18 | 123456789 | 1 | 1 | 1 | 0 | 2020-05-02 |

[Figure 3. Display of tuples given Doctor ID = ‘1’ and date from ‘2020-01-01’ to ‘2020-09-05’ ]

---

### c) Get details of all appointments at a given clinic on a specific date.

```
select *
from appointment, belongs
where appointment.Did = belongs.Did AND
        appointment.data = "given date" AND
        belongs.Cname = "given clinic name";
```

```
SELECT * from appointment, belongs
           where appointment.Did = belongs.Did
           AND  appointment.date = \'' . $date . '\'
           AND belongs.Cname = \'' . $clinic .'\''
```

read_3

c. Get details of all appointments at a given clinic on a specific date.

Clinic name: clinic_1

Date: 2020-01-01

>

| Aid | Sin | Did | Rid | Bid | miss | date | BelongId | Cname |
|---|---|---|---|---|---|---|---|---|
| 10 | 123456789 | 1 | 1 | 1 | 0 | 2020-01-01 | 9 | clinic_1 |
| 11 | 987654321 | 2 | 2 | 2 | 0 | 2020-01-01 | 14 | clinic_1 |
| 12 | 333333333 | 3 | 3 | 3 | 1 | 2020-01-01 | 15 | clinic_1 |
| 13 | 444444444 | 4 | 4 | 4 | 1 | 2020-01-01 | 16 | clinic_1 |
| 14 | 555555555 | 7 | 5 | 5 | 1 | 2020-01-01 | 17 | clinic_1 |

[Figure 4. Details of all appointments given a clinic name = ‘clinic_1’ and Date =’2020-01-01’]

---

## d) Get details of all appointments of a given patient.

```
select *
from appointment
where Sin = "given sin";
```

```
SELECT *
            from appointment
            where Sin = \'' . $sin . '\'
```

read_4

d. Get details of all appointments of a given patient.

Patient SIN: 123456789

>

| Aid | Sin | Did | Rid | Bid | miss | date |
|---|---|---|---|---|---|---|
| 10 | 123456789 | 1 | 1 | 1 | 0 | 2020-01-01 |
| 15 | 123456789 | 1 | 1 | 1 | 1 | 2020-02-02 |
| 16 | 123456789 | 1 | 1 | 1 | 4 | 2020-03-02 |
| 17 | 123456789 | 1 | 1 | 1 | 2 | 2020-04-02 |
| 18 | 123456789 | 1 | 1 | 1 | 0 | 2020-05-02 |

[Figure 5. all appointments of a patient given Pname = ‘12345678’]

---

## e) Get the number of missed appointments for each patient (only for patients who have missed at least 1 appointment).

```
select Sin, count(*) as count
from appointment
where miss = 1
group by Sin
having count(*) > 0;
```

```
SELECT Sin, count(*) as count
            FROM appointment
            where miss = 1
            group by Sin
            having count(*) > 0
```

read_5

e. Get the number of missed appointments for each patient (only for patients who have missed at least 1 appointment).

>

| Sin | count |
|---|---|
| 123456789 | 1 |
| 333333333 | 1 |
| 444444444 | 1 |
| 555555555 | 1 |
| 987654321 | 1 |

[Figure 6. the number of missed appointments for each patient]

---

**f) Get details of all the treatments made during a given appointment.**

```
select appointment.Bid, treatment.Tname, treatment.fee
from appointment, include, treatment
where Aid = "given appointment id"
AND  appointment.Bid = include.Bid
AND  include.Tname = treatment.Tname;
```

```
SELECT appointment.Bid, treatment.Tname, treatment.fee
            from appointment, include, treatment
            where Aid = \'' . $Aid . '\' AND
                    appointment.Bid = include.Bid AND
                    include.Tname = treatment.Tname
```

The result table show all treatments include inside the requested appointment

| Bid | Tname | fee |
|---|---|---|
| 1 | braces | 7000 |
| 1 | examination | 30 |
| 1 | extractions | 80 |
| 1 | fillings_and_repairs | 30 |
| 1 | root_canals | 120 |
| 1 | teeth_cleaning | 65 |
| 1 | teeth_whitening | 75 |
| 1 | fillings_and_repairs | 30 |

[Figure 7 all treatments from clinic]

---

### g) Get details of all unpaid bills.

```
select bill.Bid, Sum(treatment.fee) as sum, Pname
from bill, include, treatment, appointment, patient
where paid = 0 and bill.Bid = include.Bid and treatment.Tname =
include.Tname
and bill.Bid = appointment.Bid and appointmnet.SIN = patient.SIN
group by Bid;
```

```
SELECT bill.Bid, Sum(treatment.fee) as sum, Pname
            from bill, include, treatment, appointment, patient
            where paid = 0 AND bill.Bid = include.Bid AND treatment.Tname =
include.Tname AND bill.Bid = appointment.Bid AND appointment.SIN = patient.SIN
            group by Bid
```

| Bid | sum | Pname |
|---|---|---|
| 1 | 59440 | patient_1 |
| 2 | 30 | patient_2 |
| 3 | 80 | patient_3 |
| 4 | 7000 | patient_4 |
| 5 | 75 | patient_5 |

[Figure 8. all unpaid bills]

## Other required features

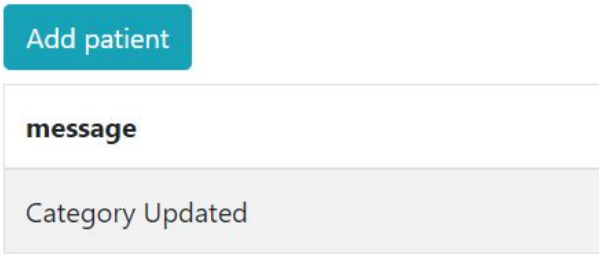
The following screenshots show some additional functions other than queries of our web application with step of procedure.

### Add a Patient

A patient can be registered by inputting his name and SIN. Here is how the section looks like.


[Figure 9. UI for adding a patient ]

After clicking on the button “Add patient” the patient will be added to the database, we can verify that by clicking the “See all patient” button. Once we add the patient, if action executes with success, we can see a message.




[Figure 10. message displayed when modification is successful]

We can verify if the patient is successively added to db by clicking the “See all patient” button.

| Pname | Sin |
| --- | --- |
| patient_1 | 123456789 |
| New patient | 123541241 |
| patient_3 | 333333333 |
| patient_4 | 444444444 |
| patient_5 | 555555555 |
| patient_6 | 666666666 |
| patient_2 | 987654321 |

[Figure 10.All existing patient with the new patient added]

Query for adding a patient.

```
INSERT INTO patient
                VALUE (\'' .$Pname . '\',\'' . $Sin . '\')
```

---

### Schedule a Appointment

Here is an overview of the section


[Figure 11.UI for schedule an appointment]

The user should first click on “1. See all clinic” button to get a list of all clinics.


[Figure 12. all clinics displayed]

Then underneed the clinic list, the user can input the clinic he/she wants to go by input the name of the clinic (in this case, clinic_1). Finally the user can schedule an appointment based on the dentist ID, his SIN and the date he wants. If the user does not know which dentist to pick or the date he wants. System will automatically select the first dentist from the clinic he wants to go to today’s date.

Query to see the list of dentists by given clinic name.

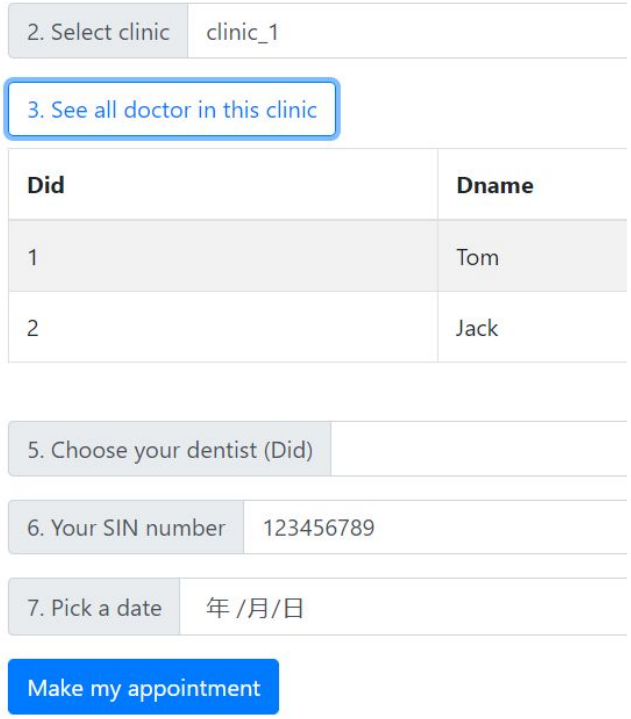
```
SELECT dentist.Did, dentist.Dname, Cname from dentist, belongs
            WHERE belongs.Cname = \'' . $Cname . '\'
            AND belongs.Did = dentist.Did
```

Query for selecting the first dentist from the clinic (if the user does not choose a dentist).

```
SELECT dentist.Did from dentist, belongs
            WHERE belongs.Cname = \'' . $Cname . '\'
            AND belongs.Did = dentist.Did
            LIMIT 1
```

Query for add the appointment.

```
INSERT INTO appointment(Did, Sin, date, miss, Rid, Bid)
                VALUE (\'' .$Did . '\',\'' . $Sin . '\',\'' . $date . '\',
1,1,1 )
```




[Figure 13. doctors displayed in clinic 1 upon on clicking the 'See all doctor in this clinic' ]

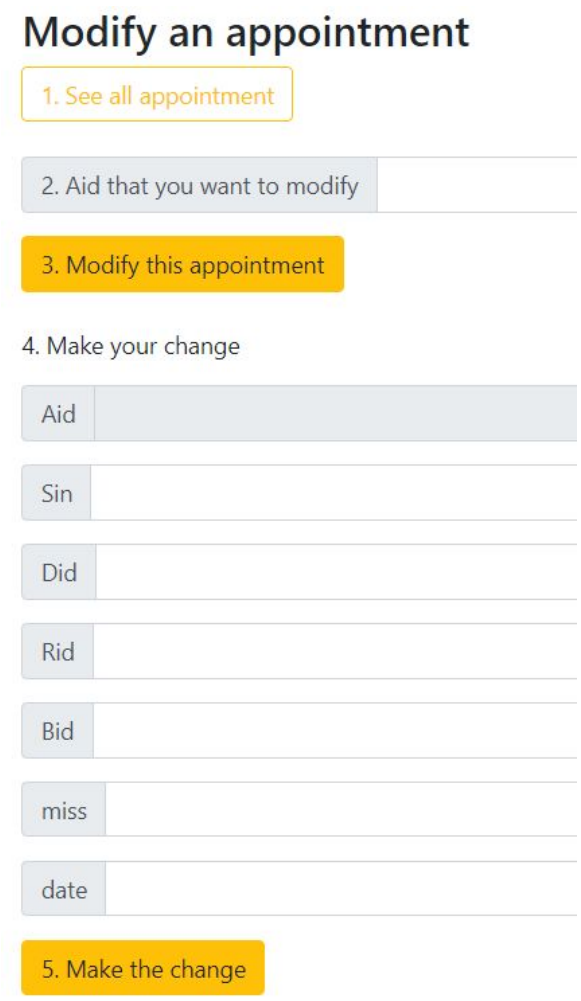
Here is an appointment we just made from clinic_1. Did is 1 from the clinic_1. Date is picked today. Miss is 1 because the user has not yet attended the appointment. We assign a receptionist and a bill ID for him.

| Aid | Sin | Did | Rid | Bid | miss | date |
|---|---|---|---|---|---|---|
| 51 | 123456789 | 1 | 1 | 1 | 1 | 2020-04-13 |

[Figure 14. new appointment made and displayed ]

**Modify an Appointment**

Here is an overview of the section. First, we can see a list of appointments by clicking on button “1. See all appointment”. Then select the ID of appointment we want to modify. Automatically, by clicking on “3. Modify this appointment”, all information of the appointment will be auto filled into the following input section. Finally, the user can make the change he wants (but not for Aid, that is why it is disabled)

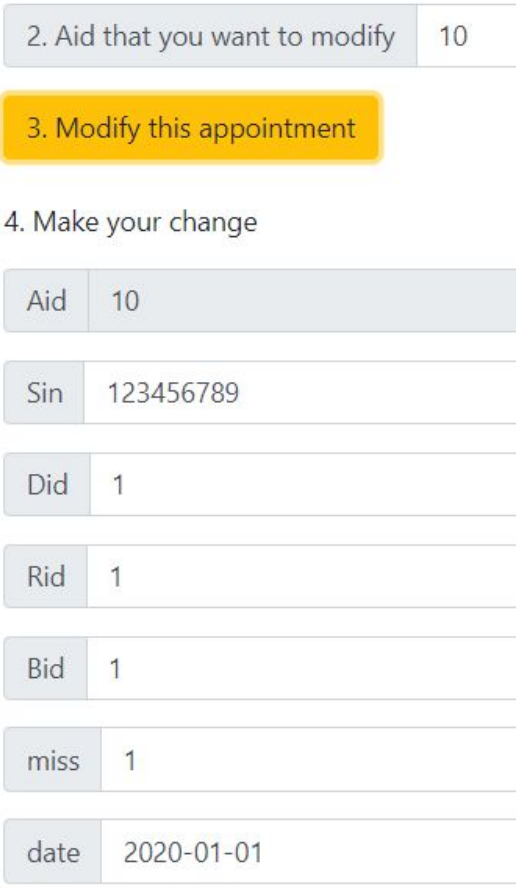



[Figure 15. UI for modifying an appointment]

If we pick appointment Aid = 10 to be modified.

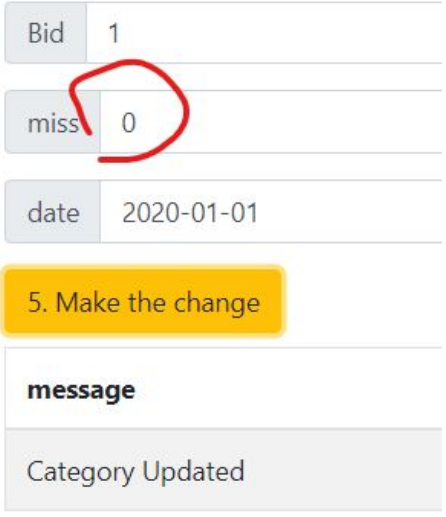

[Figure 16. an appointment displaying matching Aid = ‘10’’]

After input 10 and click to select this to modify, system autofill everything from this appointment.



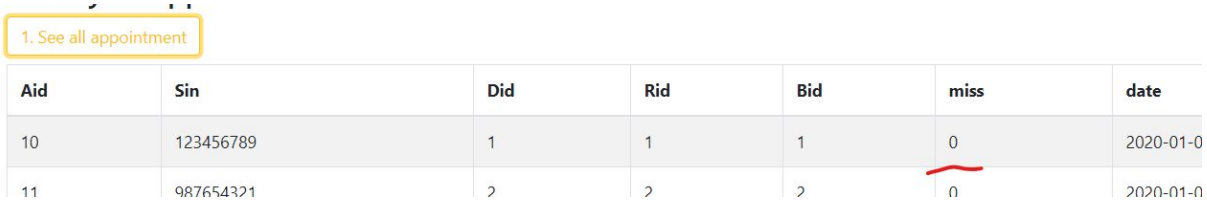

[Figure 17. autofill the input with original tuple which is Aid = ‘10’’]

For demonstration, we change the data miss of this tuple from 0 to 1.




[Figure 18.change attribute value ‘miss’ from ‘0’ to ‘1’]

We can see all appointments again and see the change is made.




[Figure 19. changed result displayed]

Query for read everything from a single appointment (result will be handle by JS at frontend)

```
SELECT * from appointment WHERE Aid = \'' . $Aid . '\'
```

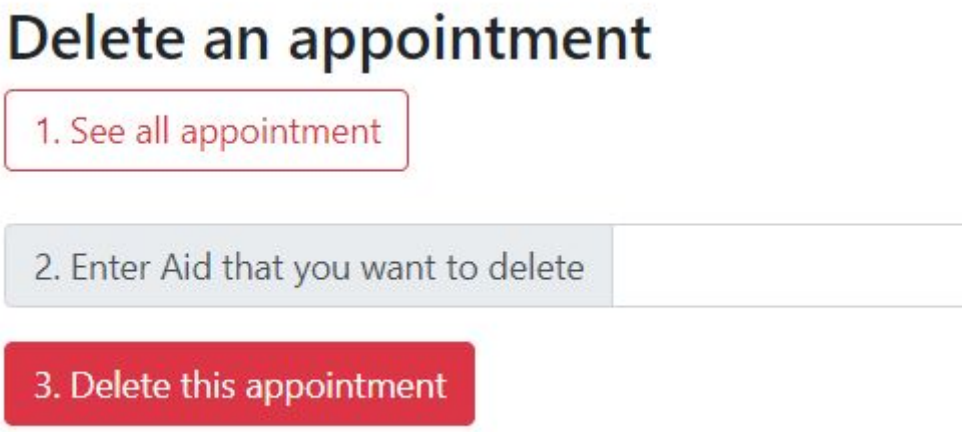
Query to modify the appointment tuple, given every data of the appointment.

```
UPDATE appointment
                SET Sin = \''. $Sin . '\',
                    Did = \''. $Did . '\',
                    Rid = \''. $Rid . '\',
                    Bid = \''. $Bid . '\',
                    miss = \''. $miss . '\',
                    date = \''. $date . '\'
            WHERE Aid = \''. $Aid . '\'
```

---

**Delete an Appointment:**

Here is the section where you can delete an appointment.

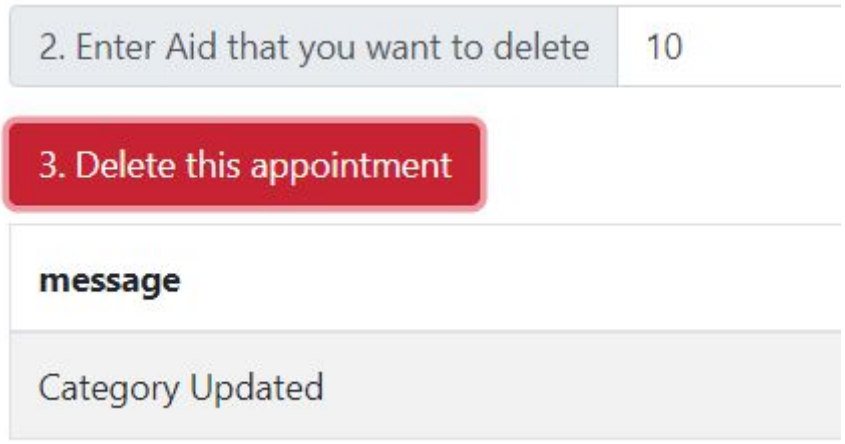



[Figure 20. UI for deleting an appointment]

Users can see the list of appointments and enter the appointment ID (Aid) that the user wants to be deleted. For example, if I want to delete an appointment with Aid = 10, I need the input 10 inside the input box then click on button “3. Delete this appointment”.


[Figure 21.display appointments upon clicking ‘see all appointment’]

With a confirmed message. The appointment with Aid = 10 is deleted from the table.




[Figure 22. message indicating successful delete]

Here is the list of all appointments after clicking again. Appointment with Aid = 10 is no longer there.


[Figure 23. update in the table(delete successful)]

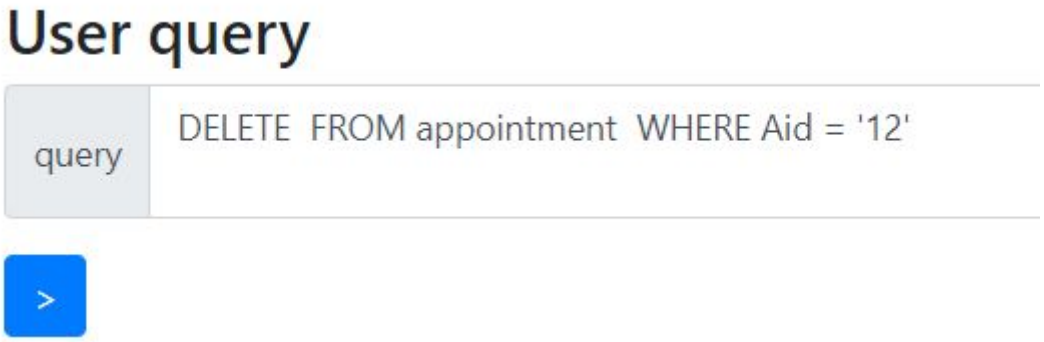
Query for deleting the appointment, given Aid

```
DELETE
                FROM appointment
                WHERE Aid = \''. $Aid . '\'
```

---

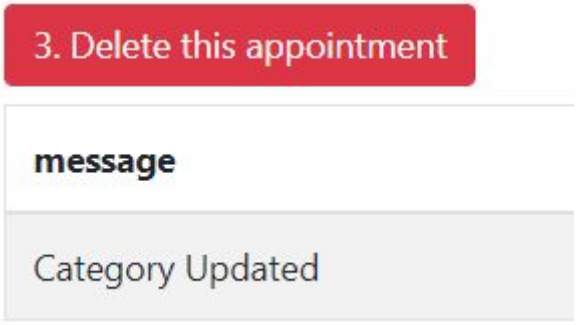
**User self-defined query**

This is the section where the user can submit its own query to modify the database. From the image below, users submit a query to delete the appointment with Aid = 12.




[Figure 24.UI for user defined query and an user input ]

A message that confirms the execution of the query is complete with success.




[Figure 25. message indicating successful delete with user-defined query]

The appointment with Aid = 12 is no longer in the database.


[Figure 26. display of the table after executing user-defined query upon on clicking 'See all appointment']

Segment of code

```
public function user_query($query){
            $stmt = $this->conn->prepare($query);
            if($stmt->execute()) {
                return true;
            }
            //printf("Error: $s.\n", $stmt->error);
            return false;
        }
```

## Other information

The website is made of vanilla html supported with CSS framework Bootstrap connecting with a php REST API to the interaction between the frontend and database.

In the developer environment, we are using XAMPP for local databases and the server for handling php REST API.

After we make sure everything is completed, we transfer everything onto the school server.

https://xvc353.encs.concordia.ca
Username: xvc353_4
Password: Comp3534

To execute the website locally, clone down everything from https://github.com/Ming424/comp353_main_project to XAMPP virtual server folder **htdocs**.
Open Apache and MySQL module on XAMPP.
Inside phpmyadmin, create a table called comp353 then import comp353.SQL from source code.
Then finally, open test.html (index.html is for school server) to access the website locally.