Project 3 Report:

Web-Based Questionnaires for Keeping Track of The Severity of The Hearing Problem

Team 7 member:

Minh Ngo

Tran Dang

CS157A - Section 03 - Professor: Dr. Katarzyna Tarnowska

I. Problem Statement

Design a web-based application to enter and store data from questionnaires collected at the visits. The questionnaires are used to evaluate the severity of the hearing problem: Tinnitus Handicap Inventory (THI) and Tinnitus Functional Index (TFI). The application adds new interview for a specific visit, either THI or TFI, collect data, and store in the given database. The application must also be able to list and select visits for adding interview. On server-side, the application connects to a given database. On client-side, the application must be easy to access and operate.

II. Analysis and Design

A. Database

Since the task is storing data for THI and TFI, we created two relations: THI and TFI The Description relation is used for entering result of THI and TFI with code number and description

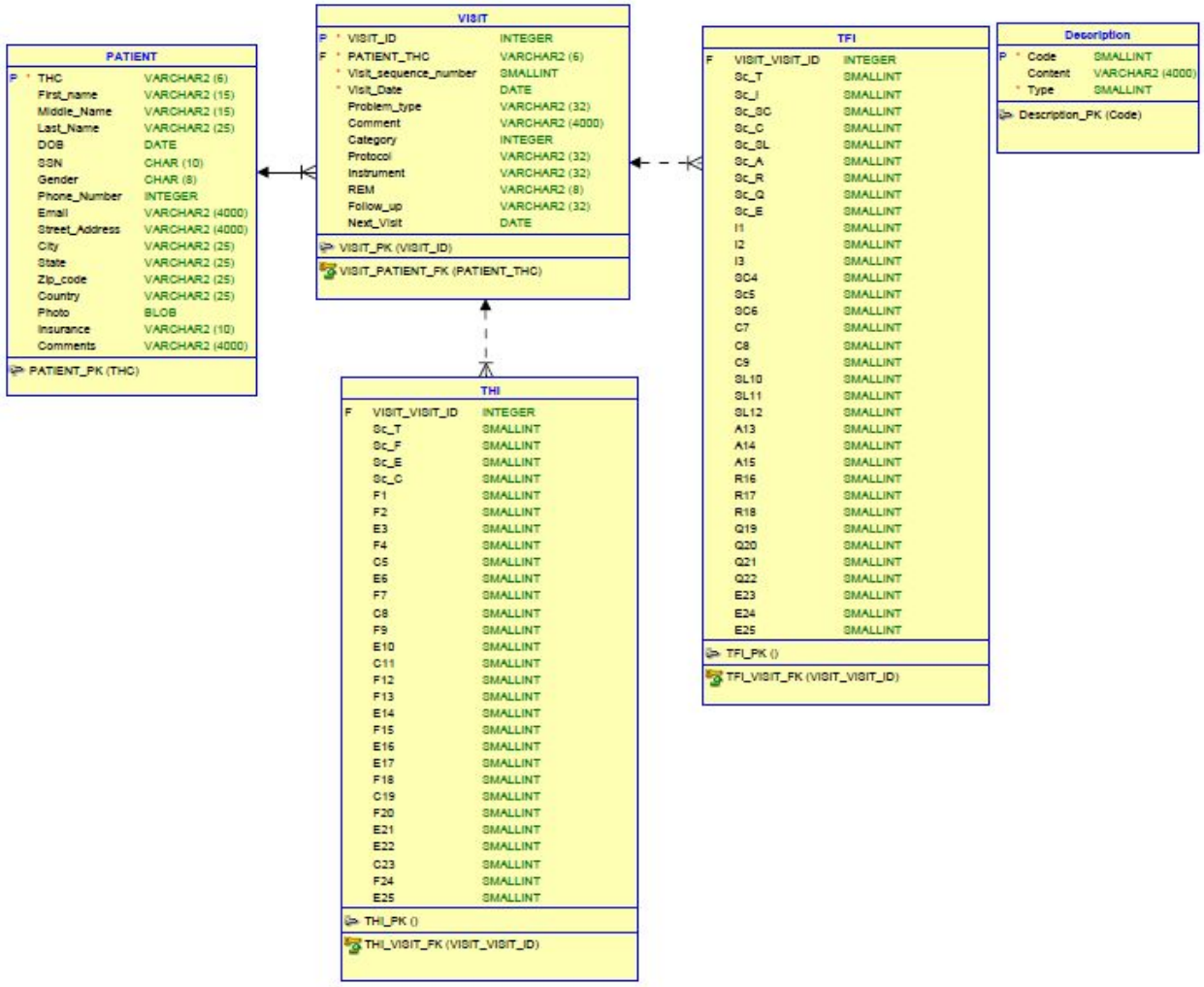
Patient and Visit relations are created for manipulating data from patients and their visits. Since our project is about THI and TFI, we hard-coded information of the patients for easier testing.

Query

On THI, TFI and Descriptions relations, the application must be able to use certain query to store the answer from the questionnaires. We used INSERT INTO, DELETE, and SELECT to save, cancel and find the data.

On Visit and Patient relations, the query is used to find and select a certain visit based on patient's given THC number or by looking up patients by name/social security number. We used SELECT and NATURAL JOIN for searching and sorting data. INSERT INTO is also used

to add new visit data to the Visit relation. The visit ID is auto-generated, and the visit sequence number is also auto-generated but unique for each patient.



B. Platforms

To design a web-based application, we used Javascript for client-side code and MySQL Workbench for server-side. Embedded Javascript (ejs) and Express Javascript is also used to help render the visual of the application on web-browser.

Implementation

The Javascript file uses Express to connect to EJS files (ejs files are html files with Javascript code embedded). The application will ask user for connection information. Since each database on each computer is different. The code will store input data and used to connect to the given database.

```
app.post('/connect',function(req,res){
    let myhost = req.body.host
    let myuser = req.body.user
    let mypass = req.body.password
    let myport = req.body.port
    let mydatabase = req.body.database
    const db = mysql.createConnection ({
        host: myhost,
        user: myuser,
        password: mypass,
        port: myport,
        database: mydatabase
    });
```

Navigation in the application is implemented by routing. Each page has different routes, such as /home, /addvisit, /editvisit, /thi, /tfi, etc.

```
app.post('/connect',function(req,res){···
})

app.post('/home',function(req,res){···
  });
app.get('/home',function(req,res){···
  });
//thc visit form
app.post('/thc', function(req,res){···
});
app.post('/newvisit', function(req,res){···
});
```

The query is declared as a string, and is passed into MySQL by query(query, function (err,result){...}. The result from MYSQL is passed to the variable result in the function. The result is an array of tuples and is rendered into ejs files. If there is an error during the query, the application is then redirect to /home, which is the homepage.

```
let query = "SELECT * FROM patient";

    // execute query
    db.query(query, (err, result) => {
        if (err) {
            res.redirect('/home');
        }
        res.render('LookUp.ejs', {
            title: "lookup",
            patients: result,
        })
    })
```

After user click add new visit, new data tuple is inserted into Vist relation with given patient's THC number and Date (current date). The Visit ID is unique and is auto-generated (auto-increment so that no visit ID will be the same). The Visit_sequence_number is generated by Javascript code. The code will find the largest visit sequence number of a patient (select from

Visit relation based on THC number). The new visit sequence number is the increment of the largest visit sequence found and insert into the data.

```
let query1 = "SELECT Visit_sequence_number FROM visit WHERE PATIENT_THC ='"+thc+"';"
db.query(query1, (err, result) => {
    if(err){
        return res.status(500).send(err)
    }
    for(let i = 0;i<result.length;i++){
        if (result[i].Visit_sequence_number >= sequence) sequence = result[i].Visit_sequence_number
    }
    sequence++
```

The THI and TFI answers are stored as arrays and insert to the relations as new tuples. When a user submit the answers, the application generates and displays the result. The data is also inserted into the tables as new data. If the user click “Submit”, then the data is kept. If the user click “Cancel”, the data is then deleted using query DELETE FROM

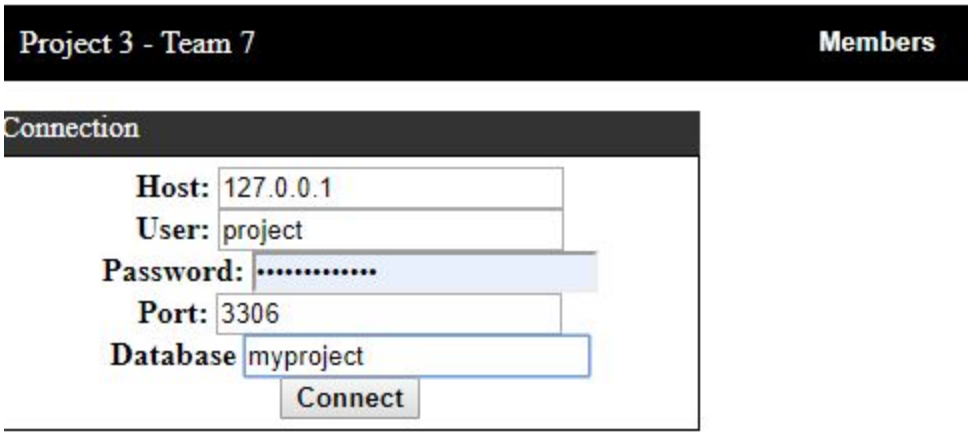
```
query = "INSERT INTO thi(F1,F2,E3,F4,C5,E6,F7,C8,F9,E10,C11,F12,F13,E14,F15,E16,E17,F18,C19,F20,E21,E2

db.query(query, [question_result], function (err, result) {
    if (err) {
        res.redirect('/home');
    }
    console.log("Number of records inserted: " + result.affectedRows);
})
```

```
let query2 = "DELETE FROM thi WHERE VISIT_VISIT_ID = " + visitid
db.query(query2, (err, result) => {
    if (err) {
        res.redirect('/home');
    }
    console.log("Number of records deleted: " + result.affectedRows);
})
```

An stylesheet (appStyle.css) is used for the application visual as well as align the data in the data in the listed tables.

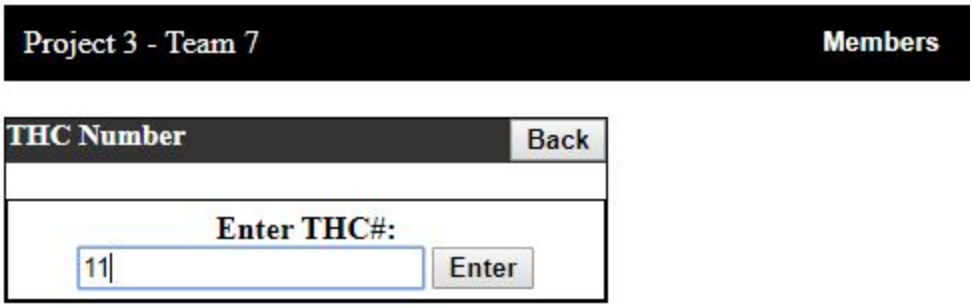
Testing

Beginning prompt asks for connection information

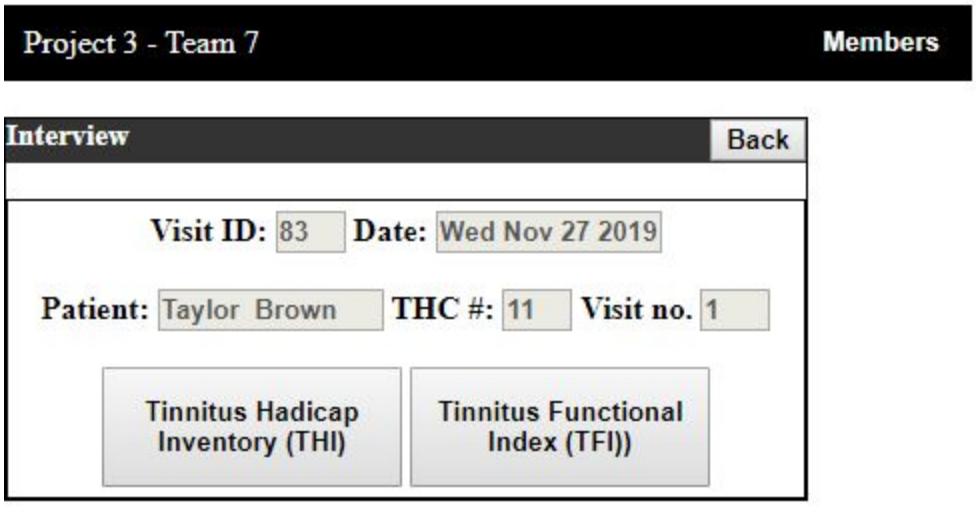


After connected to the database, main menu of the application shows up

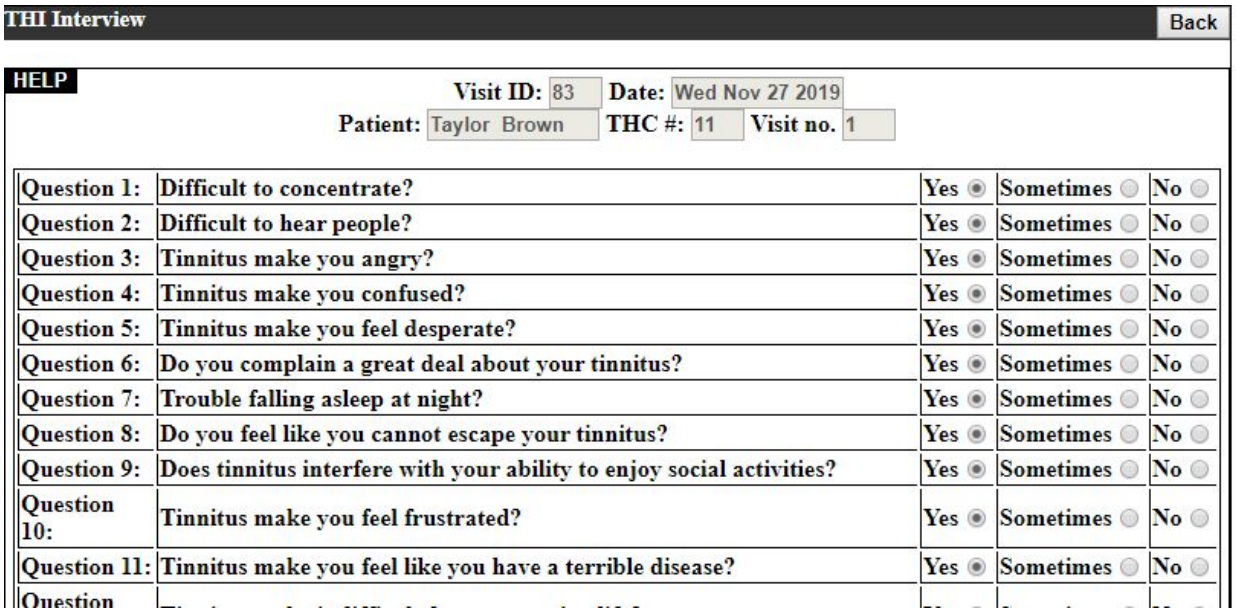
In this test, we will choose Add New Visit option, the prompt will ask user to enter THC number



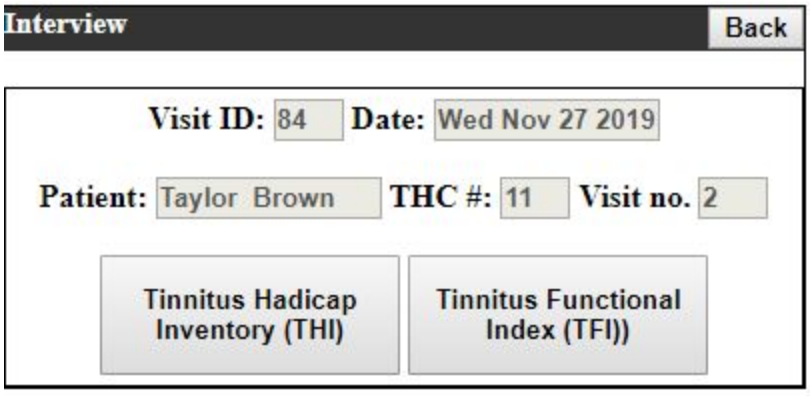
After enter THC number, the application will generate new visitID, current date, name of the patient and visit number



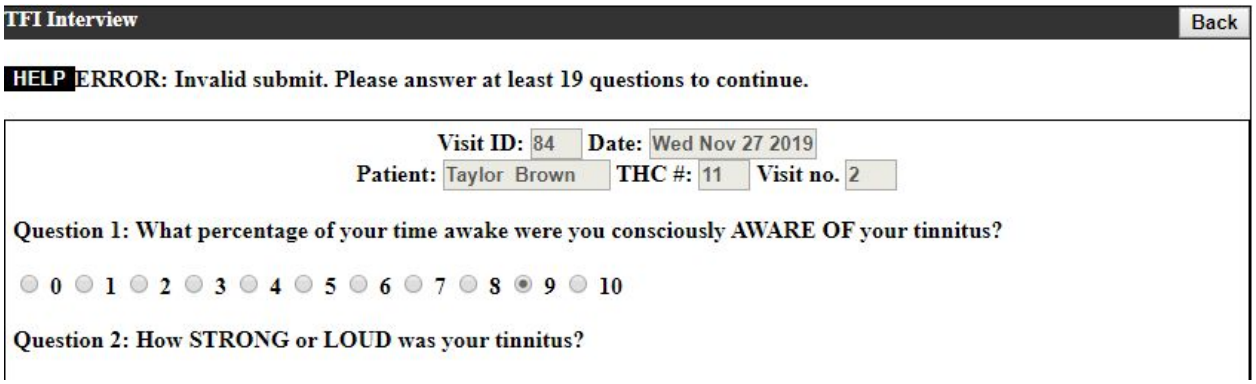
After the user choose THI option, the question page will show and option for user to choose



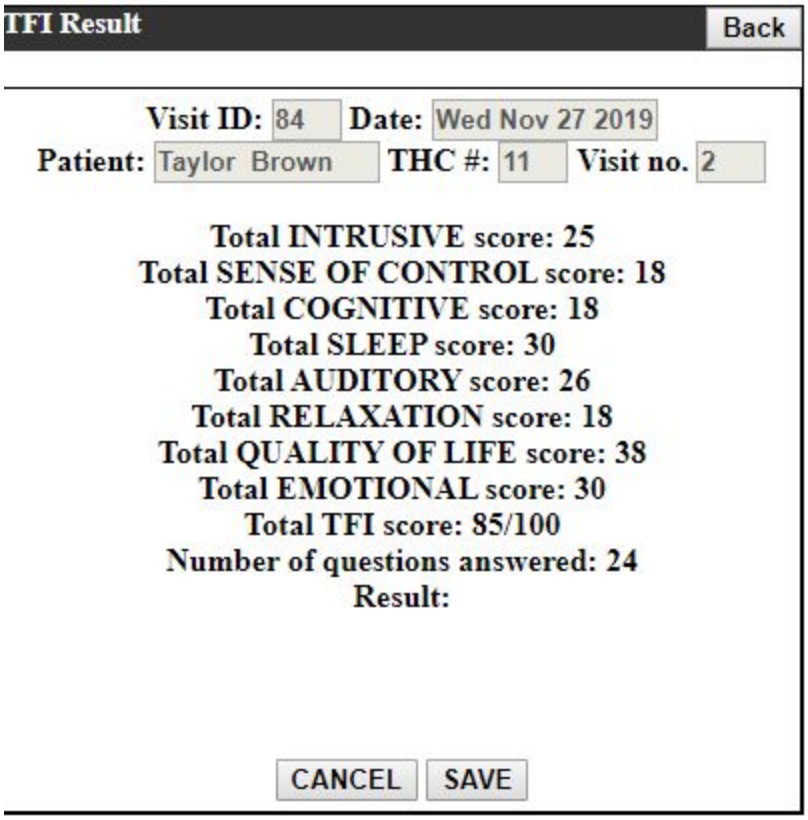
For testing, if we go back and choose Add New Visit and enter same THC number again, the application will create new data, as below. The visit ID is different and visit number is increased to 2



When user choose TFI interview option and left blank more than 19 questions, the application will not insert the data and display error.



After the user complete the questions, the result is displayed and the user has option to cancel or save the result.



III. Conclusion

The project is completed with a complete running application that is accessible to MySQL database. There are parts that need improvement such as: making the Back button more complex instead of returning back to home page and making Help button more specific for each question. However, these problems are minor and does not affect the usability of the application.

IV. Contributions

Minh Ngo:

-created relational and logical model

-helps in creating ejs files

-created sql scripts
-codes THI and TFI parts of the application
-created configuration instruction
-helps in final report
-helps in cleaning and documentation the code

Tran Dang:

-helps in creating relational and logical model
-helps in creating ejs files
-codes Add Visit, Edit Visit, Connection and Lookup parts of the application
-created css file for the application
-helps in final report
-helps in cleaning and documentation the code