Assignment #3

Overview Documentation

2020095178 최윤선

- **Metadata**

  Personalized Calendar API 최종

  - Java 파일과 Postgresql JDBC 연결

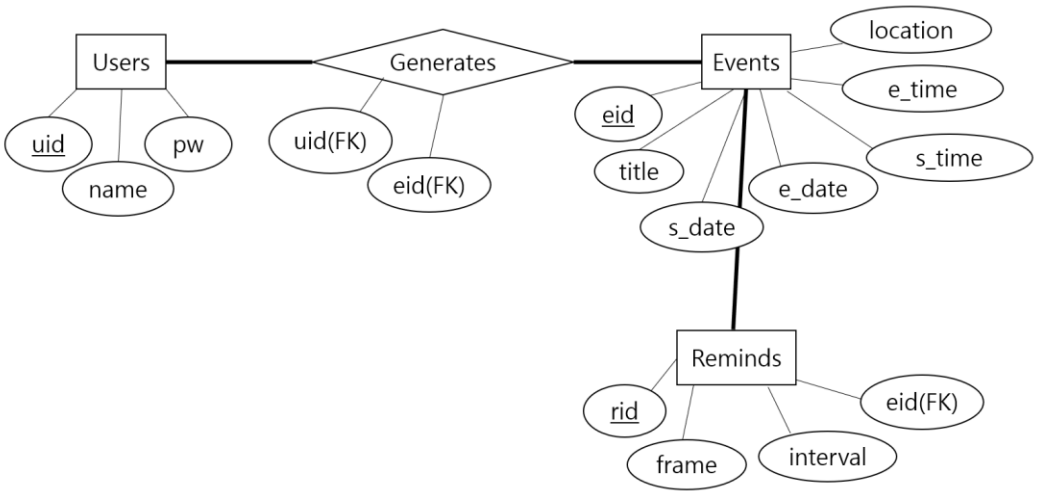
- **Summary**

  - 과제 요구 사항 변경으로 인한 스키마 및 UI 수정
  - JDBC 연결
    - Java code 와 psql 연결
    - Java UI 에서 입력한 텍스트를 받아 DB 에 입력할 수 있도록 data insertion 설계
    - 현재 로그인 된 사람의 이벤트만 접근할 수 있도록, sql select 문으로 user id 를 받아 작업할 수 있도록 설계

- **Specification**

  - Version: Ubuntu == 22.04.2 / PostgreSQL == 14
  - **FamilyCalendar.java 파일이 main 파일 → 해당 파일로 class 생성 및 실행**
  - Calendar.sql → 테이블 생성 sql
  - Calendar 구현을 위한 Requirements
    - Managing Account
      - ✓ Create a user account
      - ✓ Update a user account
      - ✓ Authenticate a user
    - Managing Events
      - ✓ Create an event
      - ✓ View an event
      - ✓ Update / Delete an event
      - ✓ Look up events in the calendar

- **Design and Implementation**
  - **스키마 설계 및 DB 설계**



위와 같이 스키마를 설계하였다. 테이블은 users, generates, events, reminds 총 4 개이다.

- Users: user 의 이름(name), id(uid), pw
- Events: 일정의 id(eid), 제목(title), 시작 날짜(s_date), 끝 날짜(e_date), 시작 시간(s_time), 끝 시간(e_time), 장소(location)
- Generates: 해당 user 가 만든 event 를 관리할 수 있는 relation 테이블
- Reminds: reminder id(rid), 해당 event 의 알림을 몇 분 전부터 줄 것인지(frame), 몇 분 간격으로 줄 것인지(interval)

Events 테이블의 eid 와 Reminds 테이블의 rid 는 자동 생성이 가능하도록 eid_seq 과 rid_seq 이라는 sequence 를 생성하였다. 따라서 event 와 remind 가 입력되면, eid 는 e1, e2, e3..., rid 는 r1, r2, r3...와 같이 자동 생성된다.

다음은 DB 를 설계한 sql 코드이다.

```
create table users (
uid varchar(50) primary key,
pw varchar(50) not null,
name varchar(100) not null);

create sequence eid_seq;

create table events (
eid varchar(10) primary key default 'e' || nextval('eid_seq')::text,
title varchar(50) not null,
s_date date not null,
e_date date not null,
s_time time,
e_time time,
location varchar(50));

create table generates (
uid varchar(50) not null,
eid varchar(10) not null,
foreign key (uid) references users(uid),
foreign key (eid) references events(eid));

create sequence rid_seq;

create table reminds (
rid varchar(10) primary key default 'r' || nextval('rid_seq')::text,
frame integer not null,
interval integer not null,
eid varchar(10) not null.
foreign key (eid) references events(eid))
```

- **Java UI 및 기능**

  회원가입 버튼, 계정 비밀번호 수정 버튼, 일정 검색 버튼을 추가하였고, 각각의 기능이 구현될 수 있도록 설계하였다.

  - **회원가입 구현**

```
signUpButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent a) {
        String sqlSignUp = "insert into Users (uid, pw, name) values (?, ?, ?)";
        try(Connection conn = DriverManager.getConnection(url:"jdbc:postgresql://127.0.0.1:5432/yschoi", user:"yschoi", password:"1008");
            PreparedStatement preparedStatement = conn.prepareStatement(sqlSignUp)) {
                preparedStatement.setString(parameterIndex:1, idField.getText());
                preparedStatement.setString(parameterIndex:2, new String(pwField.getPassword()));
                preparedStatement.setString(parameterIndex:3, nameField.getText());
                preparedStatement.executeUpdate();
                System.out.print(s:"record inserted successfully\n:");
            } catch (SQLException e) {
                System.out.print(e.getMessage());
            } catch (Exception e) {
                e.printStackTrace();
            }
        signUpFrame.dispose();
    }
});
```

  Sign up 버튼을 누른 후, 회원가입 창에 이름, id, password 를 입력하면, 연결된 database 의 users 테이블에 데이터가 입력된다.

  - **로그인 구현**

```
loginButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String id = idField.getText();
        char[] pw = pwField.getPassword();
        String sqlLogin = "select uid, pw from Users where uid=? and pw=?";

        try(Connection connection = DriverManager.getConnection(url:"jdbc:postgresql://127.0.0.1:5432/yschoi", user:"yschoi", password:"1008");
            PreparedStatement preparedStatement = connection.prepareStatement(sqlLogin)) {
            preparedStatement.setString(parameterIndex:1, id);
            preparedStatement.setString(parameterIndex:2, new String(pw));

            try(ResultSet resultSet = preparedStatement.executeQuery()) {
                if (resultSet.next()) {
                    uid = resultSet.getString(columnLabel:"uid");
                    loginBtn.setText(text:"Logout");
                    loginFrame.dispose();
                } else {
                    JOptionPane.showMessageDialog(parentComponent:null, message:"Failed Login!", title:"Login Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
});
```

  Users 테이블에 입력된 데이터를 바탕으로 로그인 창에 id 와 pw 를 입력했을 때, 해당 id 와 pw 가 users 테이블에 존재하면, 로그인에 성공한다. 이는 sql select 문을 통해 구현하였다. 만약 id 와 pw 가 맞지 않으면 "Failed Login!"이라는 경고창을 띄운다.

- **계정 정보 수정 구현**

```
modifyButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String id = idField.getText();
        char[] newPw = newPwField.getPassword();
        String sqlModify = "update users set pw=? where uid=?";

        if (!id.equals(uid)) {
            JOptionPane.showMessageDialog(parentComponent:null, message:"Wrong ID!", title:"Login Error", JOptionPane.ERROR_MESSAGE);
        }
        else {
            try(Connection connection = DriverManager.getConnection(url:"jdbc:postgresql://127.0.0.1:5432/yschoi", user:"yschoi", password:"1008");
            PreparedStatement preparedStatement = connection.prepareStatement(sqlModify)) {
            preparedStatement.setString(parameterIndex:1, new String(newPw));
            preparedStatement.setString(parameterIndex:2, uid);
            preparedStatement.executeLargeUpdate();
            loginBtn.setText(text:"Login");
            modifyFrame.dispose();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }
});
```

로그인이 되어 있는 상태에서 해당 user 가 비밀번호를 변경할 수 있도록 구현하였다. 정보 수정 직후 로그아웃 되므로, 바뀐 정보로 다시 로그인 해야한다.

- **Event 추가 구현**

```
String sqlAddEvent = "insert into Events (title, s_date, e_date, s_time, e_time, location) values (?, ?, ?, ?, ?, ?)";
String sqlAddRemind = "insert into Reminds (frame, interval, eid) values (?, ?, ?)";
String sqlEid = "select eid from events where title=? and s_date=?";
String sqlAddGenerates = "insert into Generates (uid, eid) values (?, ?)";

try(Connection connection = DriverManager.getConnection(url:"jdbc:postgresql://127.0.0.1:5432/yschoi", user:"yschoi", password:"1008");
    PreparedStatement preparedStatement = connection.prepareStatement(sqlAddEvent)) {
    preparedStatement.setString(parameterIndex:1, title);
    preparedStatement.setDate(parameterIndex:2, s_Date);
    preparedStatement.setDate(parameterIndex:3, e_Date);
    preparedStatement.setTime(parameterIndex:4, s_Time);
    preparedStatement.setTime(parameterIndex:5, e_Time);
    preparedStatement.setString(parameterIndex:6, location);
    preparedStatement.executeUpdate();

    try (PreparedStatement eidStatement = connection.prepareStatement(sqlEid)) {
        eidStatement.setString(parameterIndex:1, title);
        eidStatement.setDate(parameterIndex:2, s_Date);

        try (ResultSet generatedKeys = eidStatement.executeQuery()) {
            if (generatedKeys.next()) {
                eid = generatedKeys.getString(columnLabel:"eid");

                try (PreparedStatement remindsStatement = connection.prepareStatement(sqlAddRemind)) {
                    remindsStatement.setInt(parameterIndex:1, timeFrame);
                    remindsStatement.setInt(parameterIndex:2, interval);
                    remindsStatement.setString(parameterIndex:3, eid);
                    remindsStatement.executeUpdate();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }

                try (PreparedStatement generatesStatement = connection.prepareStatement(sqlAddGenerates)) {
                    generatesStatement.setString(parameterIndex:1, uid);
                    generatesStatement.setString(parameterIndex:2, eid);
                    generatesStatement.executeUpdate();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
        }
    }
} catch (SQLException ex) {
    ex.printStackTrace();
}
```

달력 상단의 "+" 버튼을 누르면 일정을 추가할 수 있는 창이 뜬다. 텍스트 필드에 일정의 제목, 시작 날짜, 끝나는 날짜, 시작 시간, 끝나는 시간, 장소, 몇 분 전부터 remind 를 할지(frame), 몇 분 간격으로 remind 를 할지(interval) 형식에 맞춰 입력한다. 입력된 텍스를 받아 events 테이블과 reminds 테이블에 데이터를 삽입할 수 있도록 구현하였다. 이때, 현재 로그인된 user 의 id 를 전역변수로 받아 generates 테이블에도 데이터가 삽입될 수 있도록 하였다.

- **일정 검색 기능 구현**

```
searchButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String searchDateStr = searchField.getText();
        Date searchDate = Date.valueOf(searchDateStr);

        String sqlSearch = "select title, s_date, s_time, e_date, e_time, location from events e, generates g where e.eid=g.eid and uid=? and s_date=?";

        try(Connection connection = DriverManager.getConnection(url:"jdbc:postgresql://127.0.0.1:5432/yschoi", user:"yschoi", password:"1008");
            PreparedStatement preparedStatement = connection.prepareStatement(sqlSearch)){
            preparedStatement.setString(parameterIndex:1, uid);
            preparedStatement.setDate(parameterIndex:2, searchDate);

            try (ResultSet resultSet = preparedStatement.executeQuery()) {
                if (resultSet.next()) {
                    String title = resultSet.getString(columnLabel:"title");
                    Date s_Date = resultSet.getDate(columnLabel:"s_date");
                    Time s_Time = resultSet.getTime(columnLabel:"s_time");
                    Date e_Date = resultSet.getDate(columnLabel:"e_date");
                    Time e_Time = resultSet.getTime(columnLabel:"e_time");
                    String location = resultSet.getString(columnLabel:"location");

                    String s_DateStr = s_Date.toString();
                    String s_TimeStr = s_Time.toString();
                    String e_DateStr = e_Date.toString();
                    String e_TimeStr = e_Time.toString();

                    resultPanel.removeAll();
                    resultPanel.add(new JLabel("Title: " + title));
                    resultPanel.add(new JLabel("Start Date: " + s_DateStr));
                    resultPanel.add(new JLabel("Start Time: " + s_TimeStr));
                    resultPanel.add(new JLabel("End Date: " + e_DateStr));
                    resultPanel.add(new JLabel("End Time: " + e_TimeStr));
                    resultPanel.add(new JLabel("Location: " + location));
                    resultPanel.revalidate();
                    resultPanel.repaint();

                    System.out.println(title);
                } else {
                    resultPanel.removeAll();
                    resultPanel.add(new JLabel(text:"No Event"));
                    resultPanel.revalidate();
                    resultPanel.repaint();
                }
            }catch (SQLException ex){
                ex.printStackTrace();
            }
        }catch (SQLException ex){
                ex.printStackTrace();
        }
    }
});
```

달력 상단의 search 버튼을 누르면 일정을 검색할 수 있는 창이 뜬다. Search 창 상단 텍스트 필드에 검색하고 싶은 날짜를 입력하면, 해당 날짜에 등록된 일정 정보가 뜨도록 구현하였다. 로그인 된 user id 와 시작 날짜(s_date)를 받아 해당 일정을 검색할 수 있도록 쿼리 문을 작성하였다. 만약 일정이 없는 날짜를 입력으로 받으면, "No Event"라는 문구가 나온다.

- **Event view 구현**

```
dayBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e){
        if(loginBtn.getText().equals(anObject:"Login")){
            JOptionPane.showMessageDialog(parentComponent:null, message:"Please Login First", title:"Login Error", JOptionPane.ERROR_MESSAGE);
        }else if(loginBtn.getText().equals(anObject:"Logout")){
            JFrame schFrame = new JFrame(title:"Schedule");
            schFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            JPanel schPanel = new JPanel(new BorderLayout());
            JPanel schAddPanel = new JPanel(new GridLayout(rows:6,cols:1));
            String sqlSchedule = "select title, s_date, s_time, e_date, e_time, location from events, generates where events.eid=generates.eid and uid=? and s_date=?";

            try(Connection connection = DriverManager.getConnection(url:"jdbc:postgresql://127.0.0.1:5432/yschoi", user:"yschoi", password:"1008");
                PreparedStatement preparedStatement = connection.prepareStatement(sqlSchedule)) {
                    preparedStatement.setString(parameterIndex:1, uid);
                    preparedStatement.setDate(parameterIndex:2, btnDay);

                    try (ResultSet resultSet = preparedStatement.executeQuery()) {
                        if (resultSet.next()) {
                            String title = resultSet.getString(columnLabel:"title");
                            Date s_Date = resultSet.getDate(columnLabel:"s_date");
                            Time s_Time = resultSet.getTime(columnLabel:"s_time");
                            Date e_Date = resultSet.getDate(columnLabel:"e_date");
                            Time e_Time = resultSet.getTime(columnLabel:"e_time");
                            String location = resultSet.getString(columnLabel:"location");

                            String s_DateStr = s_Date.toString();
                            String s_TimeStr = s_Time.toString();
                            String e_DateStr = e_Date.toString();
                            String e_TimeStr = e_Time.toString();

                            if (btnDay.equals(s_Date)) {
                                schAddPanel.add(new JLabel("Title: " + title));
                                schAddPanel.add(new JLabel("Start Date: " + s_DateStr));
                                schAddPanel.add(new JLabel("Start Time: " + s_TimeStr));
                                schAddPanel.add(new JLabel("End Date: " + e_DateStr));
                                schAddPanel.add(new JLabel("End Time: " + e_TimeStr));
                                schAddPanel.add(new JLabel("Location: " + location));
                            }
                        }
                    } catch (SQLException ex) {
                        ex.printStackTrace();
                    }
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
```

달력의 날짜를 각각 모두 버튼으로 만들어 일정을 보고 싶은 날짜의 버튼을 클릭하면 해당 날짜의 일정 정보를 띄우도록 구현하였다.

- **Event Update 구현**

```
saveBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String up_title = titleField.getText();
        String up_s_DateStr = s_DateField.getText();
        String up_e_DateStr = e_DateField.getText();
        String up_s_TimeStr = s_TimeField.getText();
        String up_e_TimeStr = e_TimeField.getText();
        String up_location = locationField.getText();
        String up_timeFrameStr = timeFrameField.getText();
        String up_intervalStr = intervalField.getText();

        Date up_s_Date = Date.valueOf(up_s_DateStr);
        Date up_e_Date = Date.valueOf(up_e_DateStr);
        Time up_s_Time = Time.valueOf(up_s_TimeStr);
        Time up_e_Time = Time.valueOf(up_e_TimeStr);

        Integer up_timeFrame = Integer.valueOf(up_timeFrameStr);
        Integer up_interval = Integer.valueOf(up_intervalStr);

        String sqlUpdateE = "update events set title=?, s_date=?, e_date=?, s_time=?, e_time=?, location=? where eid in (select e.eid from events e, generates g where e.eid=g.eid and uid=? and s_date=?)";
        String sqlUpdateR = "update reminds set frame=?, interval=? where eid in (select e.eid from events e, generates g where e.eid=g.eid and uid=? and s_date=?)";

        try(Connection connection = DriverManager.getConnection(url:"jdbc:postgresql://127.0.0.1:5432/yschoi", user:"yschoi", password:"1008");
            PreparedStatement preparedStatement = connection.prepareStatement(sqlUpdateE)) {
            preparedStatement.setString(parameterIndex:1, up_title);
            preparedStatement.setDate(parameterIndex:2, up_s_Date);
            preparedStatement.setDate(parameterIndex:3, up_e_Date);
            preparedStatement.setTime(parameterIndex:4, up_s_Time);
            preparedStatement.setTime(parameterIndex:5, up_e_Time);
            preparedStatement.setString(parameterIndex:6, up_location);
            preparedStatement.setString(parameterIndex:7, uid);
            preparedStatement.setDate(parameterIndex:8, btnDay);
            preparedStatement.executeUpdate();

            try(PreparedStatement preparedStatement2 = connection.prepareStatement(sqlUpdateR)) {
                preparedStatement2.setInt(parameterIndex:1, up_timeFrame);
                preparedStatement2.setInt(parameterIndex:2, up_interval);
                preparedStatement2.setString(parameterIndex:3, uid);
                preparedStatement2.setDate(parameterIndex:4, btnDay);
                preparedStatement2.executeUpdate();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }catch (SQLException ex) {
            ex.printStackTrace();
        }
        updateFrame.dispose();
    }
});
```

일정 정보를 나타내는 창의 하단에 Update 버튼을 클릭하면 일정을 수정할 수 있는 창이 뜬다. 그 창에 일정 정보를 수정하여 입력 후 save 버튼을 누르면, 해당 일정의 id(eid)는 유지한 채 데이터가 수정될 수 있도록 update 쿼리 문을 작성하였다.

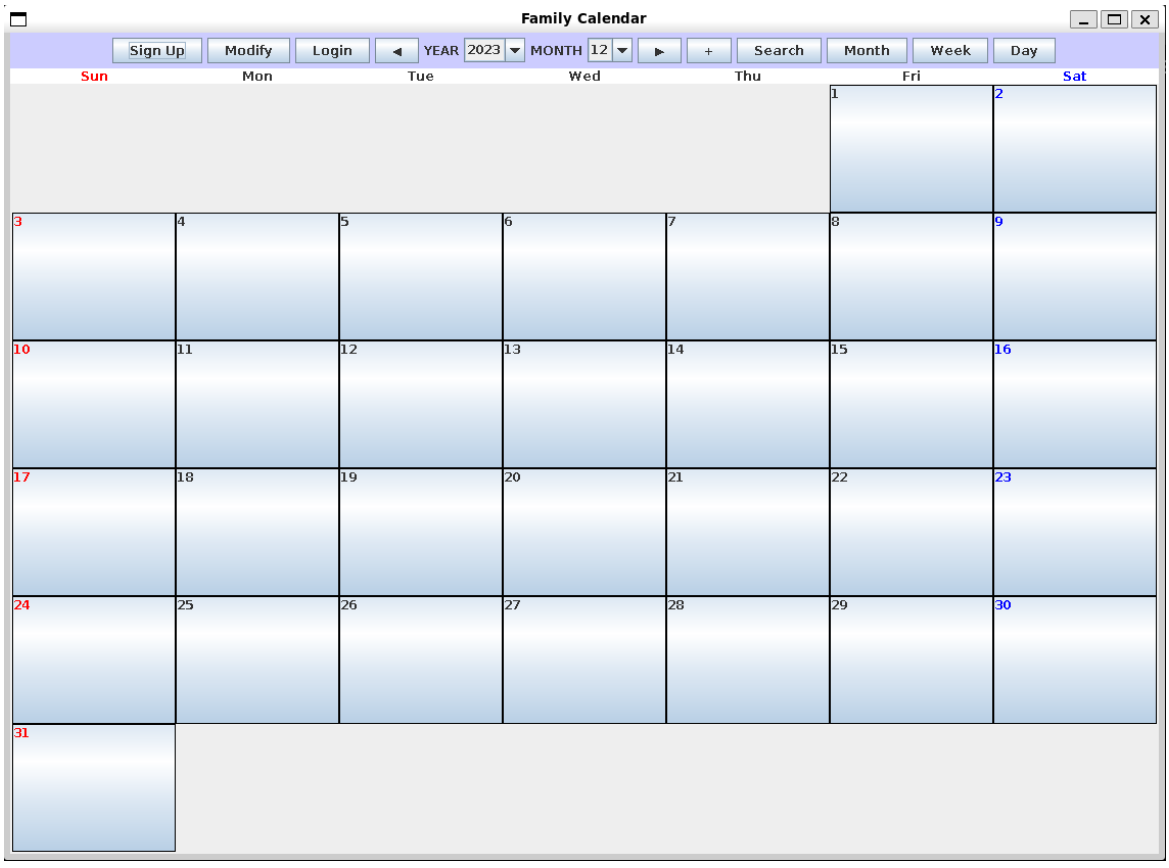
- **Event Delete 구현**

```
JButton deleteBtn = new JButton(text:"Delete");
deleteBtn.addActionListener(new ActionListener(){
    @Override
    public void actionPerformed(ActionEvent e){
        String sqlSelectDel = "select events.eid, generates.uid from events join generates on events.eid=generates.eid join reminds on events.eid=reminds.eid where generates.uid=? and events.eid in (select eid from events where s_date=?)";
        String sqlGeneratesDel = "delete from generates where uid=? and eid=?";
        String sqlRemindsDel = "delete from reminds where eid=?";
        String sqlEventsDel = "delete from events where eid=?";
        try(Connection connection = DriverManager.getConnection(url:"jdbc:postgresql://127.0.0.1:5432/yschoi", user:"yschoi", password:"1008");
            PreparedStatement preparedStatement = connection.prepareStatement(sqlSelectDel);
            PreparedStatement preparedStatement2 = connection.prepareStatement(sqlGeneratesDel);
            PreparedStatement preparedStatement3 = connection.prepareStatement(sqlRemindsDel);
            PreparedStatement preparedStatement4 = connection.prepareStatement(sqlEventsDel)) {
            preparedStatement.setString(parameterIndex:1, uid);
            preparedStatement.setDate(parameterIndex:2, btnDay);

            ResultSet resultSet = preparedStatement.executeQuery();

            if (resultSet.next()) {
                String eid = resultSet.getString(columnLabel:"eid");
                String delUid = resultSet.getString(columnLabel:"uid");

                preparedStatement2.setString(parameterIndex:1, delUid);
                preparedStatement2.setString(parameterIndex:2, eid);
                preparedStatement2.executeUpdate();

                preparedStatement3.setString(parameterIndex:1, eid);
                preparedStatement3.executeUpdate();

                preparedStatement4.setString(parameterIndex:1, eid);
                preparedStatement4.executeUpdate();
            }
        JOptionPane.showMessageDialog(parentComponent:null, message:"The schedule is deleted!", title:"Delete Schedule", JOptionPane.INFORMATION_MESSAGE);
        }catch (SQLException ex) {
            ex.printStackTrace();
        }
        schFrame.dispose();
    }
});
```

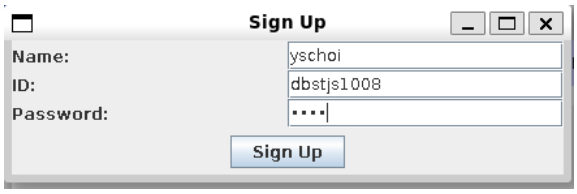
일정 정보를 나타내는 창의 하단에 delete 버튼을 누르면 해당 일정이 삭제되도록 쿼리문을 작성하여 구현하였다.

· Testing

· Java UI – 기본 달력 화면

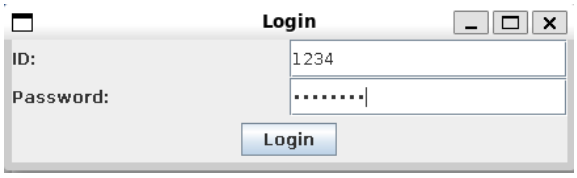


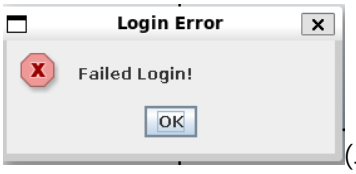
- **회원가입**




```
yschoi=# select * from users;
    uid     |  pw  |  name
------------+------+--------
 qudgns0211 | 0211 | bhchoi
 dbstjs1008 | 1008 | yschoi
 wlals1217  | 1217 | jmhan
 dmsgml0705 | 0705 | ehkim
(4 rows)
```

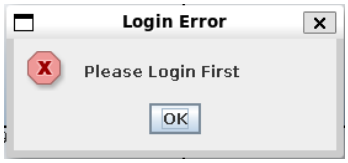
- **로그인**



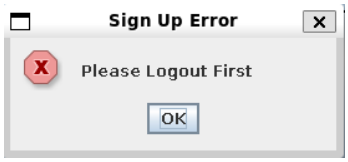


(로그인 실패)

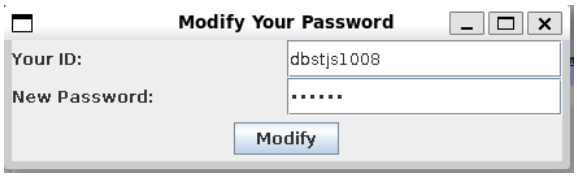
*로그인을 하지 않고 sign up 버튼 이외에 다른 버튼을 눌렀을 때



*로그인 상태에서 sign up 버튼을 눌렀을 때

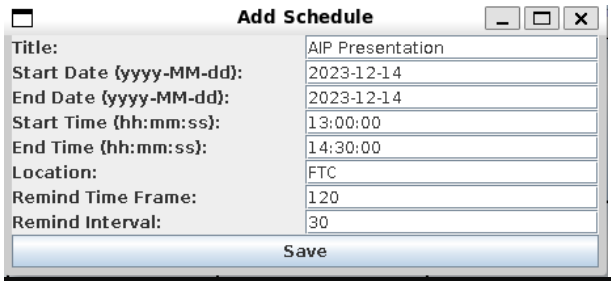


- **계정 정보 수정**




```
yschoi=# select * from users;
    uid     |   pw   |  name
------------+--------+--------
 qudgns0211 | 0211   | bhchoi
 wlals1217  | 1217   | jmhan
 dmsgml0705 | 0705   | ehkim
 dbstjs1008 | 123456 | yschoi
(4 rows)
```

- **Event 추가**

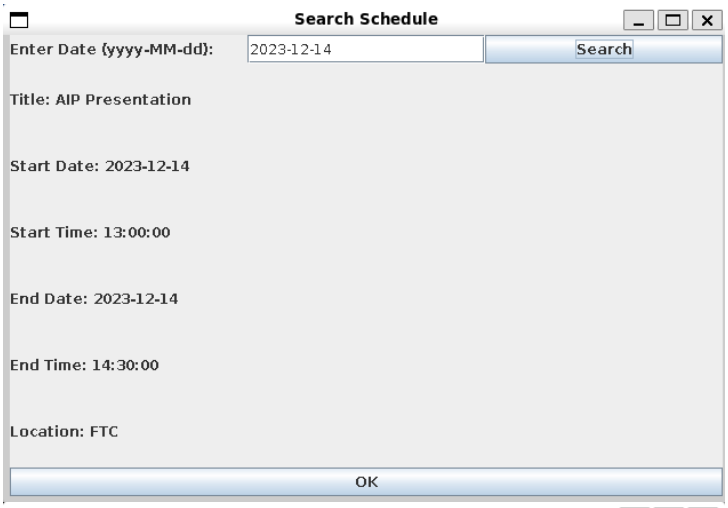


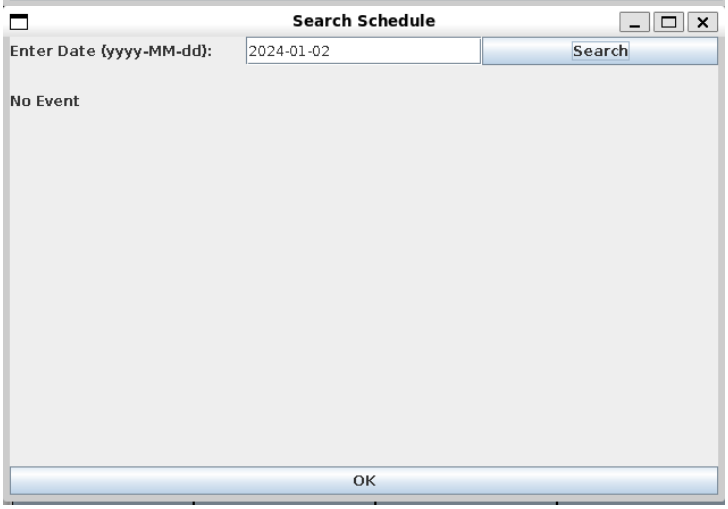

```
yschoi=# select * from events;
 eid |      title       |   s_date   |   e_date   |  s_time  |  e_time  | location
-----+------------------+------------+------------+----------+----------+-----------
 e2  | CSP Final Test   | 2023-12-18 | 2023-12-18 | 15:00:00 | 16:30:00 | FTC
 e3  | HCI Final Test   | 2023-12-19 | 2023-12-19 | 09:00:00 | 10:30:00 | FTC 5F
 e4  | Year-end party   | 2023-12-19 | 2023-12-19 | 18:00:00 | 23:59:59 | Gangnam
 e5  | Movie            | 2024-01-03 | 2024-01-03 | 19:00:00 | 21:20:00 | Lotte
 e6  | ACT meeting      | 2023-12-19 | 2023-12-19 | 17:00:00 | 22:00:00 | Wangsimni
 e7  | Travel           | 2024-01-05 | 2024-01-07 | 13:00:00 | 19:00:00 | Jejudo
 e9  | Movie            | 2023-12-31 | 2023-12-31 | 14:00:00 | 16:10:00 | CGV
 e10 | My Brithday      | 2023-12-17 | 2023-12-17 | 00:00:00 | 23:59:59 | DDP
 e11 | AIP Presentation | 2023-12-14 | 2023-12-14 | 13:00:00 | 14:30:00 | FTC
(9 rows)
```

```
yschoi=# select * from generates;
    uid     | eid
------------+-----
 dbstjs1008 | e2
 dbstjs1008 | e3
 qudgns0211 | e4
 qudgns0211 | e5
 dmsgml0705 | e6
 dmsgml0705 | e7
 dmsgml0705 | e9
 wlals1217  | e10
 dbstjs1008 | e11
(9 rows)
```

```
yschoi=# select * from reminds;
 rid | frame | interval | eid
-----+-------+----------+-----
 r2  |   180 |       60 | e2
 r3  |   120 |       60 | e3
 r4  |    60 |       15 | e4
 r5  |   120 |       30 | e5
 r6  |    60 |       15 | e6
 r7  |   180 |       60 | e7
 r9  |    60 |       30 | e9
 r10 |   120 |       60 | e10
 r11 |   120 |       30 | e11
(9 rows)
```

- **일정 검색 기능**



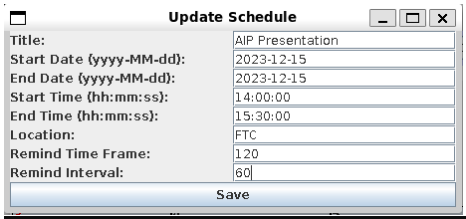


- **Event view**

(2023-12-14 버튼 클릭)

- **Event Update**

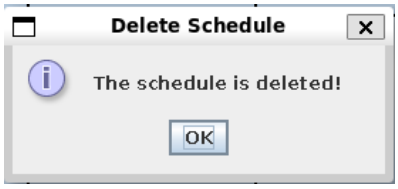
(2023-12-15 버튼 클릭)



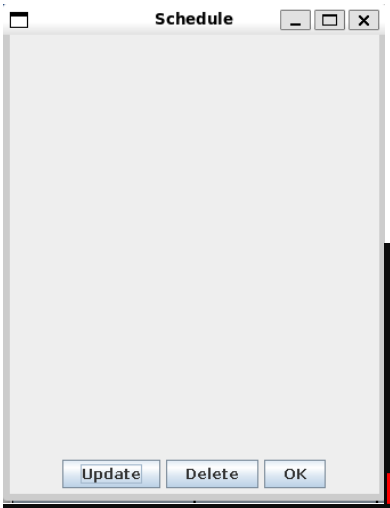

```
yschoi=# select * from events;
 eid |      title       |   s_date   |   e_date   |  s_time  |  e_time  | location
-----+------------------+------------+------------+----------+----------+-----------
 e2  | CSP Final Test   | 2023-12-18 | 2023-12-18 | 15:00:00 | 16:30:00 | FTC
 e3  | HCI Final Test   | 2023-12-19 | 2023-12-19 | 09:00:00 | 10:30:00 | FTC 5F
 e4  | Year-end party   | 2023-12-19 | 2023-12-19 | 18:00:00 | 23:59:59 | Gangnam
 e5  | Movie            | 2024-01-03 | 2024-01-03 | 19:00:00 | 21:20:00 | Lotte
 e6  | ACT meeting      | 2023-12-19 | 2023-12-19 | 17:00:00 | 22:00:00 | Wangsimni
 e7  | Travel           | 2024-01-05 | 2024-01-07 | 13:00:00 | 19:00:00 | Jejudo
 e9  | Movie            | 2023-12-31 | 2023-12-31 | 14:00:00 | 16:10:00 | CGV
 e10 | My Brithday      | 2023-12-17 | 2023-12-17 | 00:00:00 | 23:59:59 | DDP
 e11 | AIP Presentation | 2023-12-15 | 2023-12-15 | 14:00:00 | 15:30:00 | FTC
(9 rows)
```

```
yschoi=# select * from reminds;
 rid | frame | interval | eid
-----+-------+----------+-----
 r2  |   180 |       60 | e2
 r3  |   120 |       60 | e3
 r4  |    60 |       15 | e4
 r5  |   120 |       30 | e5
 r6  |    60 |       15 | e6
 r7  |   180 |       60 | e7
 r9  |    60 |       30 | e9
 r10 |   120 |       60 | e10
 r11 |   120 |       60 | e11
(9 rows)
```

- **Event Delete**



(2023-12-15 버튼 다시 클릭)




```
yschoi=# select * from events;
 eid |     title      |   s_date   |   e_date   |  s_time  |  e_time  | location
-----+----------------+------------+------------+----------+----------+----------
 e2  | CSP Final Test | 2023-12-18 | 2023-12-18 | 15:00:00 | 16:30:00 | FTC
 e3  | HCI Final Test | 2023-12-19 | 2023-12-19 | 09:00:00 | 10:30:00 | FTC 5F
 e4  | Year-end party | 2023-12-19 | 2023-12-19 | 18:00:00 | 23:59:59 | Gangnam
 e5  | Movie          | 2024-01-03 | 2024-01-03 | 19:00:00 | 21:20:00 | Lotte
 e6  | ACT meeting    | 2023-12-19 | 2023-12-19 | 17:00:00 | 22:00:00 | Wangsimni
 e7  | Travel         | 2024-01-05 | 2024-01-07 | 13:00:00 | 19:00:00 | Jejudo
 e9  | Movie          | 2023-12-31 | 2023-12-31 | 14:00:00 | 16:10:00 | CGV
 e10 | My Brithday    | 2023-12-17 | 2023-12-17 | 00:00:00 | 23:59:59 | DDP
(8 rows)
```

```
yschoi=# select * from generates;
    uid     | eid
------------+-----
 dbstjs1008 | e2
 dbstjs1008 | e3
 qudgns0211 | e4
 qudgns0211 | e5
 dmsgml0705 | e6
 dmsgml0705 | e7
 dmsgml0705 | e9
 wlals1217  | e10
(8 rows)
```

```
yschoi=# select * from reminds;
 rid | frame | interval | eid
-----+-------+----------+-----
 r2  |   180 |       60 | e2
 r3  |   120 |       60 | e3
 r4  |    60 |       15 | e4
 r5  |   120 |       30 | e5
 r6  |    60 |       15 | e6
 r7  |   180 |       60 | e7
 r9  |    60 |       30 | e9
 r10 |   120 |       60 | e10
(8 rows)
```

- **Known problems**

1. Reminder 를 구현하는 데 어려움이 있어, 완벽하게 구현하지 못했다.
2. Weekly view 와 Daily view 를 구현하는 데 어려움이 있어, 이 또한 완벽하게 구현하지 못했다.
3. DB 에는 같은 날짜에 여러 일정을 추가할 수 있지만, UI 에 view 를 띄울 때, 하나의 날짜 당 하나의 일정만 뜨는 문제가 있다.

- **Comments**

- 설계, 코드 작성, 과제에서 어려웠던 부분은 무엇인가?
  - UI 디자인은 Assignment#2에서 대부분 구현이 되었기 때문에 버튼 추가와 몇 개의 윈도우를 추가하는 것 이외에는 큰 시간을 들이지 않았다.

- Java 코드에 SQL에 넣고, 이 쿼리들이 원하는 대로 정상적으로 구현이 되도록 하는데 많은 시간이 걸렸다. 특히, 일정을 추가하여 DB에 insert하는 부분에 많은 시간이 할애되었다. 더하여, 텍스트 필드를 통해 데이터를 입력 받았기 때문에 이를 스키마 데이터 타입에 맞게 변경하는 데 있어 생각보다 많은 오류가 났다. 이를 디버깅하는데 많은 시간을 들였다.
- Reminder를 띄우는 것과 weekly, daily view를 만드는 데 어려움 있었다.