1차 프로젝트 보고서

오출(오늘도출근)

통원/통근버스 예약 프로그램

2 팀

윤준호(조장), 김유진, 문수지, 손영석

- 목 차 -

1 프로젝트 개요

1.1 기획의도

도로 위 차량들은 국가 VIP 경호와 국가행사가 있을 경우 구간 도로 우회와 신호기 제어로 인한 교통 통제로 기존 정체구간에서 더 많은 시간을 소비합니다.

도로 아래 지하철은 최근 이어져온 2호선, 4호선, 5호선 그리고 9호선 장애인 차별철폐 탑승시위로 인하여 지하철 운행이 중단되는 문제를 야기하고 최대 80분이 넘는 시간을 지연시켰습니다.

이와 같이 사전에 예고되지 않은 교통통제는 대중교통 이용에 문제를 유발하고 출퇴근 시민들에게 큰 불편입니다.

1.2 주제선정

기존의 통근버스, 통원버스 등은 지정된 시간에 탑승지에서 탑승하는 구조로 운영되고 있습니다. 사전 고지 없는 국가 행사와 시위 그리고 기상이변으로 인해 대중교통 출퇴근이 어려워졌을 때, 통근버스에 대한 수요는 증가할 것이며, 소속 기관들은 운영 가능한 교통수단을 예약제를 통해 지원할 수 있는 시스템을 고안했습니다.

기관단체는 상황을 파악하고 수요에 맞는 운영 가능한 교통수단과 경로를 설정하고 제공합니다. 이용자는 자신에게 적합한 일정의 제공된 교통수단을 예약하고 이를 통해 출퇴근을 수월하게 해주는 예약 시스템을 구축하고자 합니다.

1.3 프로젝트 목표

이번 프로젝트의 목표는 그동안 학습한 것들을 최대한 구현하는 것과 팀프로젝트를 통한 협업 경험입니다.

1. 디자인 패턴 DAO, 싱글톤 등의 경험
2. JDBC를 통한 DB접근과 활용
3. 데이터베이스 모델링
4. SQL 활용 숙달과 단순조회 외, 그룹함수, join, 서브쿼리 등 다양한 SQL문 활용 시도
5. JavaFX와 CSS를 통한 화면 구현

위와 같은 요소들을 프로젝트 과정에 담도록 노력했습니다.

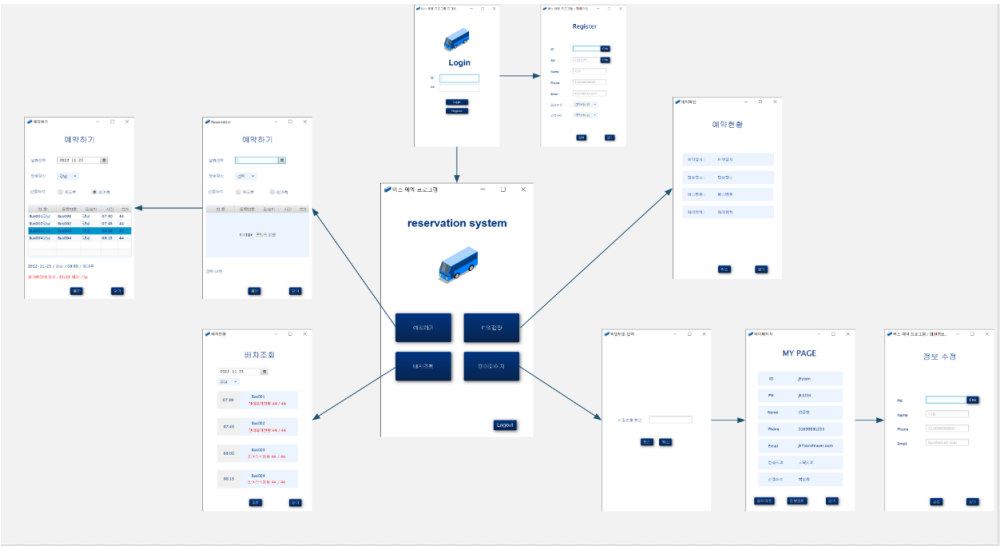
2. 프로젝트 구상

2.1 프로그램 전체 시나리오

단체 소속 회원 대상 운영 방식의 버스예약을 위해 로그인, 회원가입, 홈화면, 예약신청, 배차조회, 마이페이지를 핵심기능으로 구현

2.2 프로그램 전체 화면 구성

<그림. 전체 화면구성도>



2.3 DB설계

a. 활용 DB: Oracle18CEx

b. 테이블 구성 : 사용자정보(User), 예약정보(Reserve), 버스정보(Bus), 위치정보(Location) 총 4개의 테이블로 구성

c. 주요 특징

1) Foreignkey 활용 : 타 테이블을 참조하며 무결성이 중요하다고 생각되는 경우

2) 뷰(View)를 활용한 보안성 강화

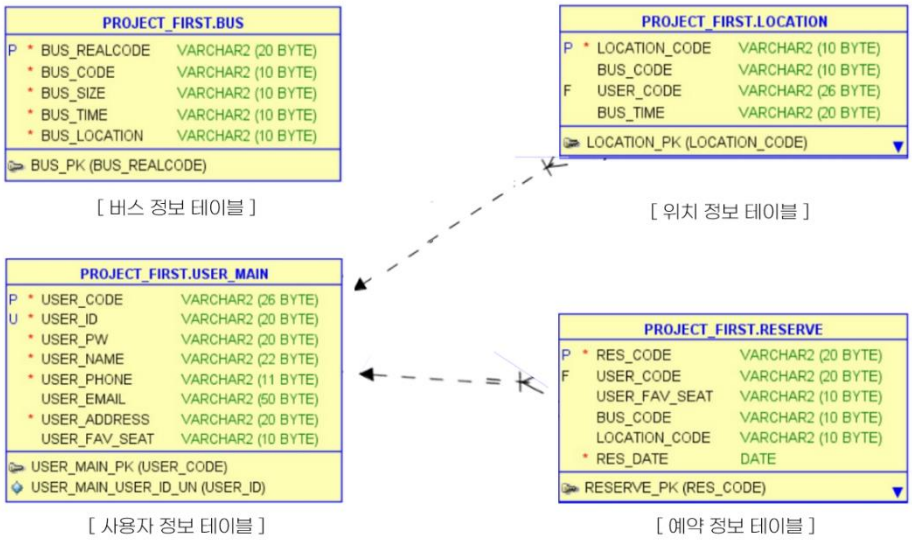
- 사용자 마이페이지 조회 시 사용자정보 테이블 전체를 불러오는 대신 필요한 정보들만으로 구성한 뷰를 통한 접근

3) 테이블별 접근권한 구분

- 임의의 유저를 생성 후 필요한 권한을 부여한 후 사용

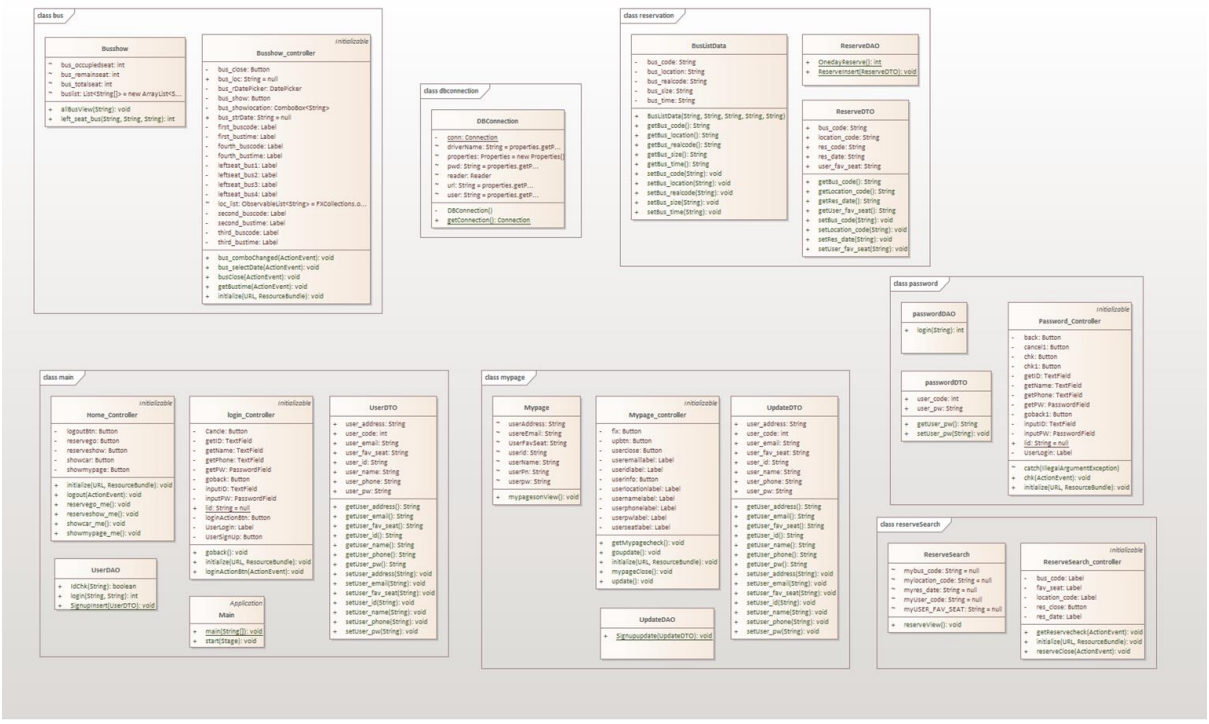
- 버스정보 테이블의 입력, 삭제 등의 경우 DBA 등 DB 직접접속 권한 필요

<그림. ER다이어그램(IE표기법)>



2.4 클래스 다이어그램

각 기능별 특성을 반영하여 총 7개의 패키지로 구성하였고, 특히 DB에 빈번하게 접속하게 되는 중요기능에 대해서는 DAO패턴을 활용



a. main 패키지(DAO패턴)

- 전체 실행

- 로그인 기능

- 홈화면 컨트롤

b. mypage 패키지(DAO패턴)

- 마이페이지(본인 가입정보) 조회

- 회원가입정보 수정

c. reservation 패키지(DAO패턴)

- 예약하기 기능

d. password 패키지(DAO패턴)

- password 검증 기능

e. reservesearch 패키지

- 예약조회 기능

f. bus 패키지

- 배차정보 조회 기능

g. dbconnection 패키지

- DB 연결정보 설정

- 외부 라이브러리(oracle.properties.txt / DB접속용 driver ,URL, user, password)를 작성해 DB접속 URL을 미리 설정해 활용

2.5 JDBC

JDBC를 통해 OracleDB와 JAVA 연결

<코드화면. 로그인/회원가입>

oracle.properties

```
driver=oracle.jdbc.driver.OracleDriver
url=jdbc:oracle:thin:@localhost:1521/xepdb1
user=project_first
password=1234
```

외부 라이브러리(oracle.properties)를 생성해 URL 저장

<코드화면. 로그인/회원가입>

```
public class DBConnection {

    private static Connection conn;

    private DBConnection() {
    }

    static {
        //환경 설정(db접속용) 파일을 읽어오기 위한 객체 생성
        Properties properties = new Properties();
        Reader reader;
        try {
            reader = new FileReader("lib/oracle.properties");
            properties.load(reader);
        } catch (FileNotFoundException e1) {
            System.out.println("예외: 지정한 파일을 찾을 수 없습니다. :" + e1.getMessage());
            e1.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }//end try

        String driverName = properties.getProperty("driver");
        String url = properties.getProperty("url");
        String user = properties.getProperty("user");
        String pwd = properties.getProperty("password");

        try {
            Class.forName(driverName);
            conn = DriverManager.getConnection(url, user, pwd);
            System.out.println("connection success");
        } catch (ClassNotFoundException e) {
            System.out.println("예외: 드라이버로드실패 :" + e.getMessage());
            e.printStackTrace();
        } catch (SQLException e) {
            System.out.println("예외: connection fail :" + e.getMessage());
            e.printStackTrace();
        }//end try
        }//end static

        //싱글턴 패턴 (한 번 접속되면 다른데서 사용 X)
        public static Connection getConnection() {
            return conn;
        }//end getConnection
}//end DBConnection
```

JDBC를 활용해 DB와 자바를 연결하였고 싱글톤 패턴을 사용

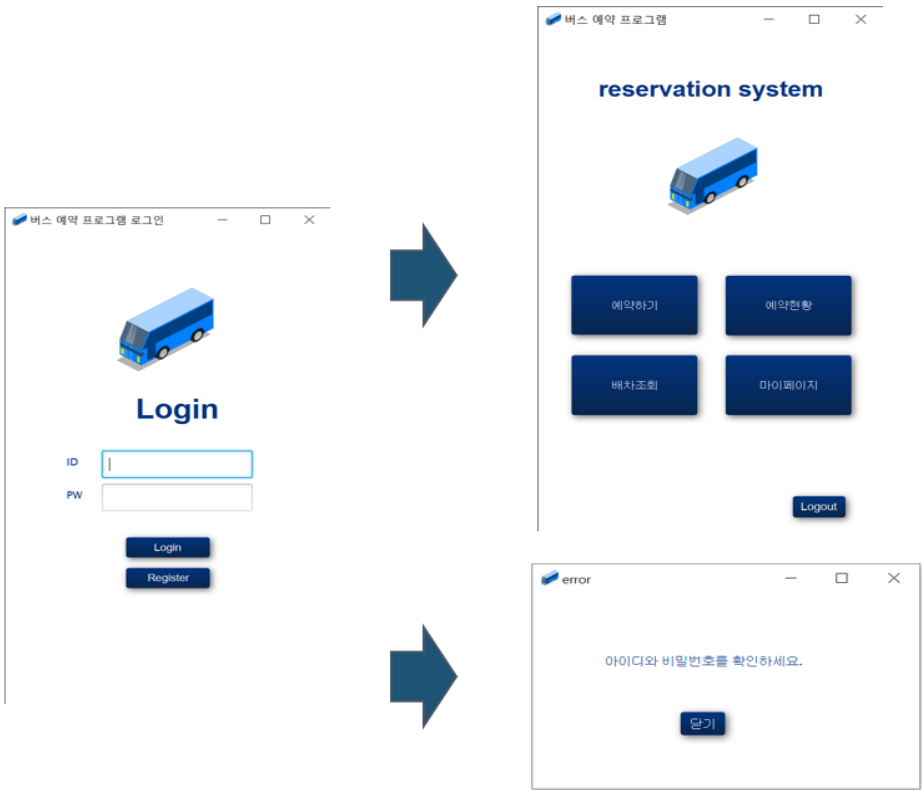
3. 프로젝트 구현

3.1 로그인/회원가입

3.1.1 로그인

ID와 비밀번호를 입력하고 로그인 버튼을 누르면 DB에 저장되어 있는 정보와 비교하여 일치할 경우 홈 화면으로 이동하고, 일치하지 않을 경우 오류 알림 창이 뜨도록 구현함

<그림. 로그인/회원가입 (로그인) 화면>



<코드화면. 로그인/회원가입>

```
//로그인 버튼
public void loginActionBtn(ActionEvent event) throws Exception {
    UserDTO uDTO = new UserDTO();   //객체 생성
    try {

        uDTO.setUser_id(inputID.getText());
        uDTO.setUser_pw(inputPW.getText());
        String getID = inputID.getText();
        String getPW = inputPW.getText().toString();
        UserDAO obj = new UserDAO();
        lid = uDTO.getUser_id();
        // 로그인하지 않고 버튼 눌렀을 때 오류 뜨기
        if (obj.login(getID,getPW)==1) {
            System.out.println("성공");

            try {
                Parent members = FXMLLoader.load(getClass().getResource("home_main.fxml"));
                Scene scene = new Scene(members);
                Stage primaryStage = (Stage) ((Node) event.getSource()).getScene().getWindow();
                //css 등록
                scene.getStylesheets().add(getClass().getResource("app.css").toString());
                //창 아이콘 이미지 삽입
                primaryStage.getIcons().add(new Image("file:lib/bus_image.png"));
                primaryStage.setTitle("버스 예약 프로그램");
                primaryStage.setScene(scene);
            } catch (Exception e) {
                e.printStackTrace();
            } else {
                System.out.println("실패");
                System.out.println(obj.login(getID, getPW));

                try {
                    Parent members = FXMLLoader.load(getClass().getResource("error.fxml"));
                    Scene scene = new Scene(members);
                    Stage primaryStage = (Stage) ((Node) event.getSource()).getScene().getWindow();
                    //css 등록
                    scene.getStylesheets().add(getClass().getResource("app.css").toString());
                    //창 아이콘 이미지 삽입
                    primaryStage.getIcons().add(new Image("file:lib/bus_image.png"));
                    primaryStage.setTitle("error");
                    primaryStage.setScene(scene);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
                System.out.println("contentID: " + getID);
                System.out.println("contentPW: " + getPW);
                System.out.println(getPW.getClass().getName());

            } catch (IllegalArgumentException e) {
                System.out.println("IllegalArgumentException caught"); // 예외처리 발생!!

            }
        }
```

Textfield로 아이디, passwordfield로 비밀번호 입력 값을 받고 DTO를 통해 get/set으로 User_main DB에 저장된 정보와 입력된 아이디, 비밀번호 대조 후 일치할 경우와 일치하지 않을 경우 다음 단계로 넘어가도록 설정

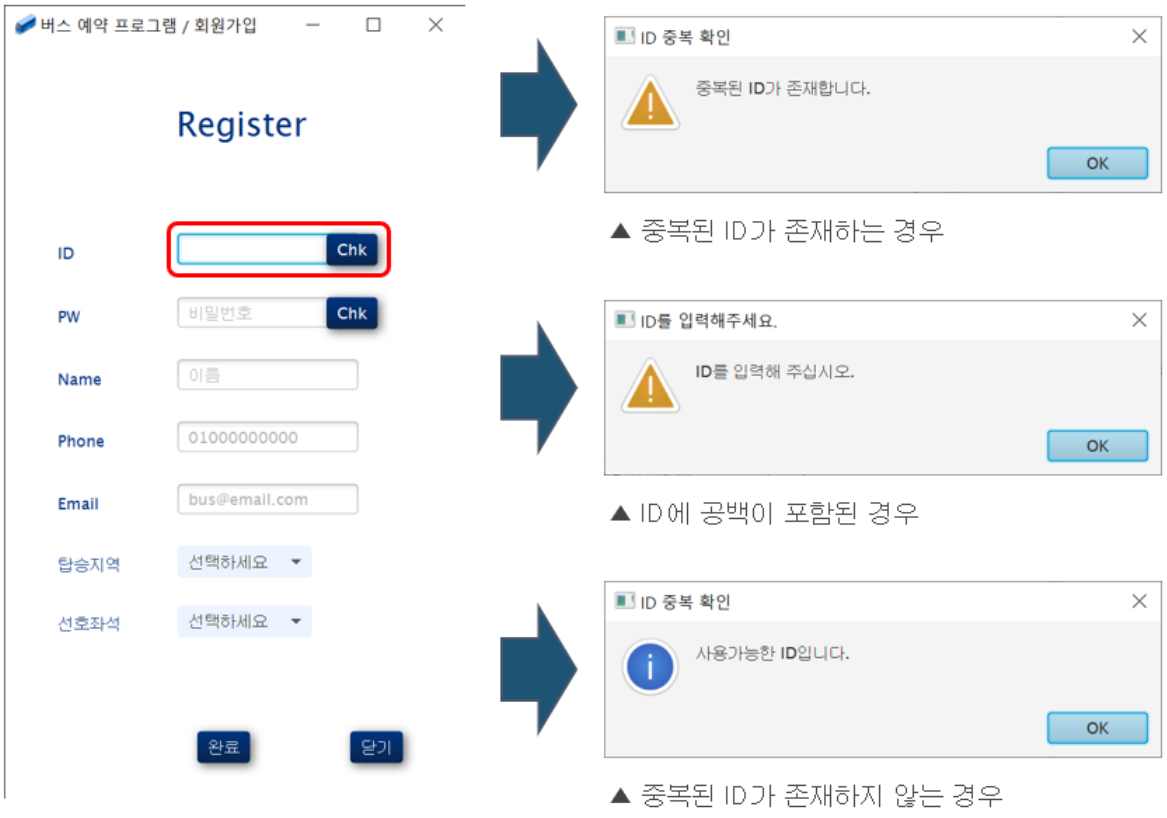
3.1.2 회원가입

로그인하기 위해 회원가입을 통해 계정을 생성해야 하며, 입력된 정보는 DB의 User_MAIN에 저장됨

추가적인 기능으로 ID 중복 체크 확인과 PW 조건 충족여부 확인 기능을 구현했으며 선호좌석을 제외한 모든 정보를 입력해야만 저장이 되도록 구현

a. ID 중복여부 확인

<그림. 로그인/회원가입 (회원가입) 화면>



<코드화면. 로그인/회원가입>

```
public boolean IdChk(String inputID) {
    String sql = "SELECT count(user_id) FROM "
            + "project_first.user_main WHERE user_id=?"; // 
    boolean isExist = false;

    try {
        Connection conn = DBConnection.getConnection(); // C
        PreparedStatement prst = conn.prepareStatement(sql);
        prst.setString(1, inputID); // 1번 물음표에 문자열로 입력받은
        ResultSet rs = prst.executeQuery();// 쿼리문 실행 결과를 
        rs.next(); //레코드가 있는 경우 읽음

        int result = rs.getInt(1);
        if(result == 1) {
            isExist = true; //중복된 ID 존재
        }else {
            isExist = false; //중복된 ID 없음
        } //end else
    }catch (Exception e) {
        e.printStackTrace();
    } // end catch
    return isExist;
} // end boolean
```

UserDAO 클래스에서 기존의 DB에 저장되어 있는 ID를 불러오기 위해 idChk메소드를 생성함

입력된 ID를 count하는 SQL문을 작성하여 count값이 1이면 중복된 ID가 존재하는 것이므로 IF문을 통해 true를 할당하였고, 존재하지 않는 경우는 false를 할당함

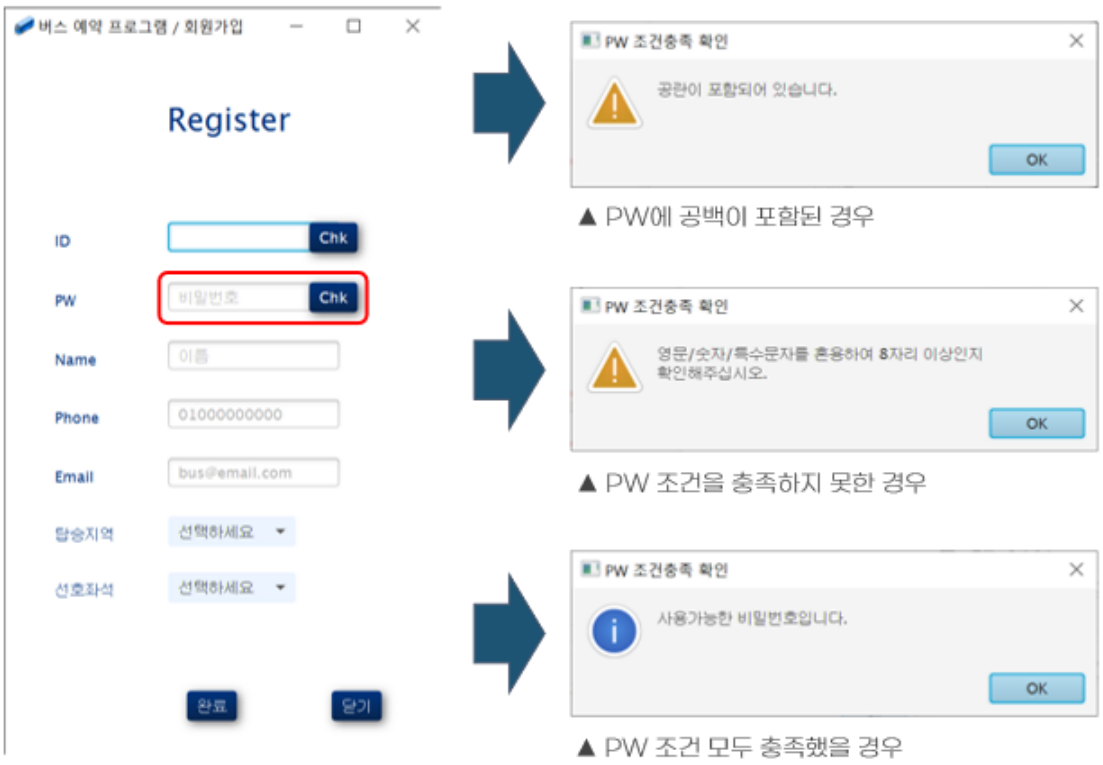
```
@FXML
public void idChk(ActionEvent event) throws Exception {
    UserDTO uDTO = new UserDTO();   //객체 생성

    try {
        uDTO.setUser_id(sid.getText()); // 입력한 ID값 setting
        getID = uDTO.getUser_id();  // 입력한 ID값 할당
        UserDAO obj = new UserDAO();    // 객체 생성

        if (getID.isEmpty() || getID.contains(" ")) { // (11/2

            Alert alert = new Alert(AlertType.WARNING); // WARI
            alert.setTitle("ID를 입력해주세요."); // 알림창 title 지정
            alert.setHeaderText(null); // 알림창 headertext 지정
            alert.setContentText("ID를 입력해 주십시오."); // 알림창에
            alert.show(); // 알림창 보여줌

            sid.clear(); // ID 입력창 비움

        }else if (obj.IdChk(getID)) {   // 입력한 ID값이 존재할 경우

            Alert alert = new Alert(AlertType.WARNING); // WARI
            alert.setTitle("ID 중복 확인"); // 알림창 title 지정
            alert.setHeaderText(null); // 알림창 headertext 지정
            alert.setContentText("중복된 ID가 존재합니다."); // 알림창
            alert.show(); // 알림창 보여줌

            sid.clear(); // ID 입력창 비움

        } else {    // 공백이 포함되지 않고 중복된 ID가 없을 경우 ID생성

            Alert alert = new Alert(AlertType.INFORMATION); //
            alert.setTitle("ID 중복 확인"); // 알림창 title 지정
            alert.setHeaderText(null); // 알림창 headertext 지정
            alert.setContentText("사용가능한 ID입니다."); // 알림창에
            alert.show(); // 알림창 보여줌
        } // end else
    } catch (IllegalArgumentException e) {
        System.out.println("IllegalArgumentException caught");
    } // end catch
} // end method
```

SignUp_Controller 클래스에서 idChk 이벤트 핸들러를 생성함

IF문을 통해 공백과 중복여부를 확인할 수 있도록 코드를 작성

b. PW 조건 충족여부 확인

<그림. 로그인/회원가입 (회원가입) 화면>



<코드화면. 로그인/회원가입>

```
public int validPW(String inputPW) {
    getPW = spw.getText();  // 입력한 PW값 할당
    int inspect; // 변수 선언

    String regExp = "^(?=.*[0-9])(?=.*[a-zA-Z])(?=.*\\W).{8,20}$";
    /*
    (?=.*[0-9]) >> 0~9까지 숫자 최소 한개
    (?=.*[a-zA-Z]) >> 영소대문자 a~z / A~Z 문자 최소 한개
    (?=.*\\W) >> 특수문자 최소 한개
    .{8,20} >> 선행 3가지 표현식(숫자, 영소문자, 특수문자)에 맞는 8~20자리 문자열
    */

    if (getPW.isEmpty() || getPW.contains(" ")) { // PW창에 공백이 있을
        inspect=0;  // 0반환
    } else if (getPW.matches(regExp)) { // 정규표현식 적용 결과 true >>
        inspect=2;  // 2반환
    } else { // 공백은 없으면서 정규표현식 조건을 충족하지 않을 경우
        inspect=1;  // 1반환
    } // end else
    return inspect;
} // end method
```

SignUp_Controller 클래스에서 PW조건을 확인할 수 있는 validPW 메소드를 생성함

Java 정규표현식을 사용하여 암호패턴을 만들었고 matches를 통해 일치 여부를 확인함

```
@FXML
public void pwChk(ActionEvent event) throws Exception {
    UserDTO uDTO = new UserDTO();   //객체 생성

    try {
        uDTO.setUser_pw(spw.getText()); //입력한 PW값 setting
        getPW = uDTO.getUser_pw();  // PW값 할당

        if (validPW(getPW)==0) {    // PW창에 공백이 있을 경우

            Alert alert = new Alert(AlertType.WARNING); // WARNING타입 알림창 생성
            alert.setTitle("PW 조건충족 확인"); // 알림창 title 지정
            alert.setHeaderText(null); // 알림창 headertext 지정
            alert.setContentText("공란이 포함되어 있습니다."); // 알림창에 뜰 문자 지정
            alert.show(); // 알림창 보여줌

            spw.clear(); // PW 입력창 비움

        } else if (validPW(getPW)==1){ // 공백은 없으면서 정규표현식 조건을 충족하지 않을 경우

            Alert alert = new Alert(AlertType.WARNING); // WARNING타입 알림창 생성
            alert.setTitle("PW 조건충족 확인"); // 알림창 title 지정
            alert.setHeaderText(null); // 알림창 headertext 지정
            alert.setContentText("영문/숫자/특수문자를 혼용하여 8자리 이상인지 확인해주십시오.");
            alert.show(); // 알림창 보여줌

            spw.clear(); // PW 입력창 비움

         } else { // 조건 충족할 경우

              Alert alert = new Alert(AlertType.INFORMATION); //
              alert.setTitle("PW 조건충족 확인"); // 알림창 title 지정
              alert.setHeaderText(null); // 알림창 headertext 지정
              alert.setContentText("사용가능한 비밀번호입니다."); // 알림
              alert.show(); // 알림창 보여줌
         } // end else
    } catch (IllegalArgumentException e) {
        System.out.println("IllegalArgumentException caught");
    } // end catch
}// end method
```

SignUp_Controller 클래스에서 pwChk 이벤트 핸들러를 생성하였고 IF문을 통해 validPW 메소드 결과값에 따라 알림창이 뜨도록 구현

c. 완료버튼 이벤트 (정보입력 미충족시)

<그림. 로그인/회원가입 (회원가입) 화면>

<코드화면. 로그인/회원가입>

```
@FXML
public void signupcompleteBtn(ActionEvent event) throws Exception {
    UserDTO uDTO = new UserDTO();   //객체 생성
    UserDAO obj = new UserDAO();
    uDTO.setUser_id(sid.getText());
    getID = uDTO.getUser_id();
    getPW = uDTO.getUser_pw();
    String getPhone = sphone.getText();

    if(sid.getText().isEmpty()||spw.getText().isEmpty()|| // (11/19 문
            sname.getText().isEmpty()||sphone.getText().isEmpty()||
            semail.getText().isEmpty()||strAddress.isEmpty()) {
        Alert alert = new Alert(AlertType.WARNING); // WARNING타입 알림창
        alert.setTitle("비어있는 정보가 있습니다."); // 알림창 title 지정
        alert.setHeaderText(null); // 알림창 headertext 지정
        alert.setContentText("정보를 모두 입력해주세요."); // 알림창에 뜰 문자 지정
        alert.show(); // 알림창 보여줌
    } else if (obj.IdChk(getID)) {  // 중복체크 없이 확인 버튼 눌렀는데, 중복ID일
        Alert alert = new Alert(AlertType.WARNING); // WARNING타입 알림창
        alert.setTitle("아이디 중복확인"); // 알림창 title 지정
        alert.setHeaderText(null); // 알림창 headertext 지정
        alert.setContentText("아이디 중복확인해주십시오."); // 알림창에 뜰 문자 지정
        alert.show(); // 알림창 보여줌
    } else if(validPW(getPW)==0 || validPW(getPW)==1) { // 비밀번호 조건 불
        Alert alert = new Alert(AlertType.WARNING); // WARNING타입 알림창
        alert.setTitle("비밀번호 조건충족 확인"); // 알림창 title 지정
        alert.setHeaderText(null); // 알림창 headertext 지정
        alert.setContentText("비밀번호 조건충족 확인해주십시오."); // 알림창에 뜰 문
        alert.show(); // 알림창 보여줌
    } else if (getPhone.length()>11) {  // (11/22 문수지) 휴대전화 번호가 11
        Alert alert = new Alert(AlertType.WARNING); // WARNING타입 알림창
        alert.setTitle("휴대전화 번호"); // 알림창 title 지정
        alert.setHeaderText(null); // 알림창 headertext 지정
        alert.setContentText("휴대전화 번호를 숫자 11자리 이하로 작성해주십시오.");
        alert.show(); // 알림창 보여줌
    } else {
```

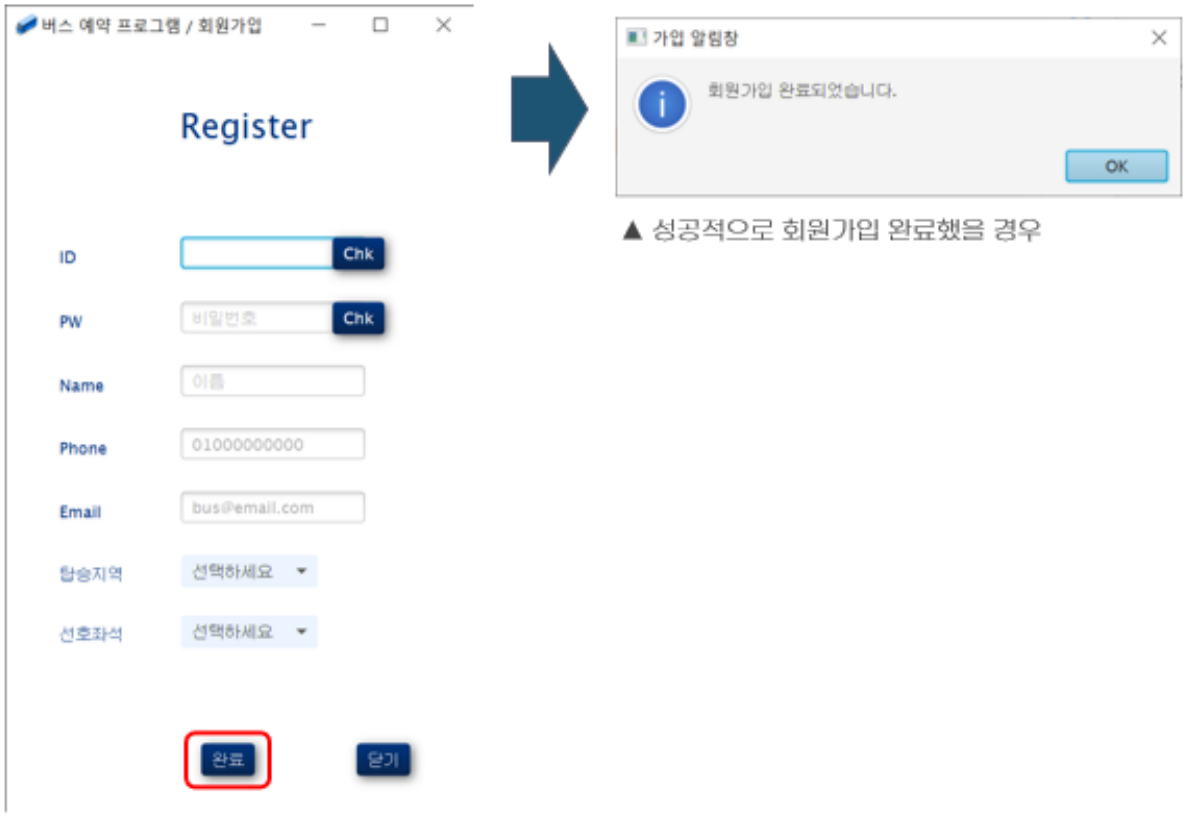
SignUp_Controller 클래스에서 완료버튼을 눌렀을 경우의 이벤트핸들러를 생성함

선호좌석을 제외한 정보들이 모두 입력되지 않았다면 알림창이 뜨도록 구현

중복된 ID를 입력 후 완료버튼을 눌렀을 경우, PW조건을 확인하지 않고 완료버튼을 눌렀을 경우, 휴대폰 번호를 11자리를 초과하여 작성했을 경우 각각에 맞는 알림창이 뜨도록 구현

d. 완료버튼 이벤트 (회원가입 성공)

<그림. 로그인/회원가입 (회원가입) 화면>



▲ 성공적으로 회원가입 완료했을 경우

<코드화면. 로그인/회원가입>

```
} else {

    try {
        uDTO.setUser_id(sid.getText());
        uDTO.setUser_pw(spw.getText());
        uDTO.setUser_name(sname.getText());
        uDTO.setUser_phone(sphone.getText());
        uDTO.setUser_email(semail.getText());
        uDTO.setUser_address(strAddress);
        uDTO.setUser_fav_seat(strFavseat);

        UserDAO.SignupInsert(uDTO); //(11.17 윤준호 작성 -
        //(11.17 문수지) 팝업 알림창 추가
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("가입 알림창"); // 알림창 title 지정
        alert.setHeaderText(null); // 알림창 headertext 지
        alert.setContentText("회원가입 완료되었습니다."); // 알
        alert.show(); // 알림창 보여줌

    try {
            Parent members = FXMLLoader.load(getClass().getResource("signup.fxml"));
            Scene scene = new Scene(members);
            Stage primaryStage = (Stage) ((Node) event.getSource()).getScene().getWindow();
            //(11/22 문수지) css 등록
            scene.getStylesheets().add(getClass().getResource("app.css").toString());
            //(11/21 문수지) 창 아이콘 이미지 삽입
            primaryStage.getIcons().add(new Image("file:lib/bus_image.png"));
            primaryStage.setTitle("버스 예약 프로그램 / 로그인");
            primaryStage.setScene(scene);
        } catch (Exception e) {
            e.printStackTrace();
        }
    } catch (IllegalArgumentException e) {
        System.out.println("IllegalArgumentException caught"); // 예외처리 발생!!
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

제약 조건들을 모두 만족했을 경우 회원가입이 성공적으로 완료되면 입력된 값을 UserDTO 의 setter 를 활용해서 해당 변수에 값을 할당하고 그를 알리는 알림창이 뜨도록 함

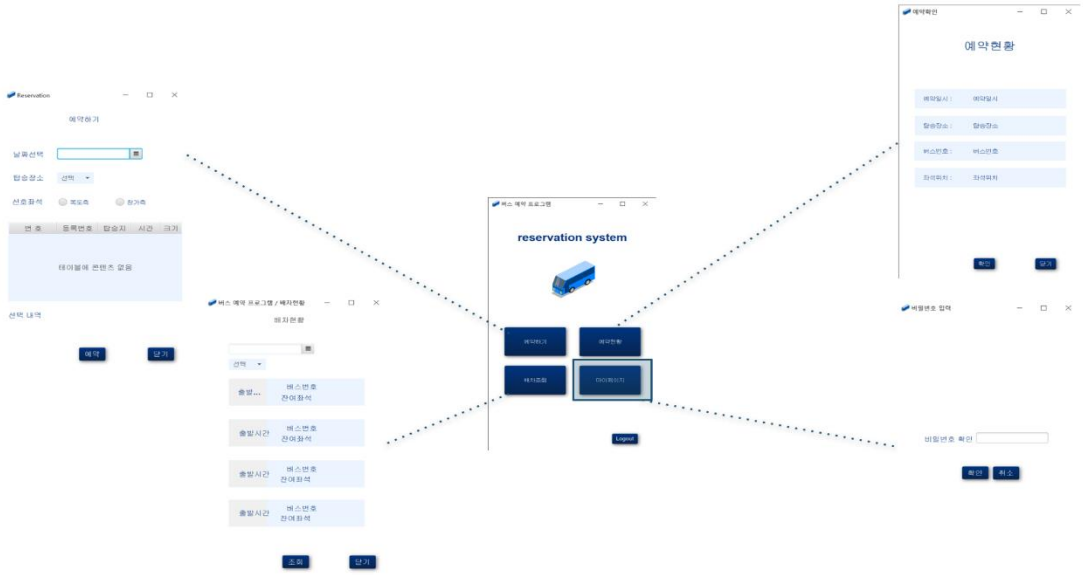
```
public static void SignupInsert(UserDTO uDTO) {

    // (11/21 문수지) user_code가 문자형으로 되어있기 때문에 to_number로 숫자형으로 변환
    String sql = "INSERT INTO user_main"
            + "(user_code, user_id, user_pw, user_name, user_phone, user_email, user_address, user_fav_seat)"
            + "VALUES((select 'user'||nvl(max(to_number(substr(user_code, 5)))+1,1) from user_main),?,?,?,?,?,?,?)";
    try{
            Connection conn = DBConnection.getConnection();
            PreparedStatement prst = conn.prepareStatement(sql);

            prst.setString(1, uDTO.getUser_id());
            prst.setString(2, uDTO.getUser_pw());
            prst.setString(3, uDTO.getUser_name());
            prst.setString(4, uDTO.getUser_phone());
            prst.setString(5, uDTO.getUser_email());
            prst.setString(6, uDTO.getUser_address());
            prst.setString(7, uDTO.getUser_fav_seat());
            prst.executeUpdate();

            System.out.println("정보 입력 성공");
    } catch(SQLException e) {
        System.err.format("SQL State: %s\n%s", e.getSQLState(), e.getMessage());
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

이때 입력한 정보들은 UserDAO 클래스의 SignupInsert메소드에서 JDBC를 사용해 DB와 연결하여 저장되도록 작성함

3.2 홈화면

홈화면에서는 예약하기, 예약현황 조회, 배차조회, 마이페이지 버튼을 클릭시 각 화면을 호출, 단 마이페이지 버튼은 바로 마이페이지를 보여주는 대신, 비밀번호 재검증 요구

<그림. 홈화면>



<코드화면. 홈화면>

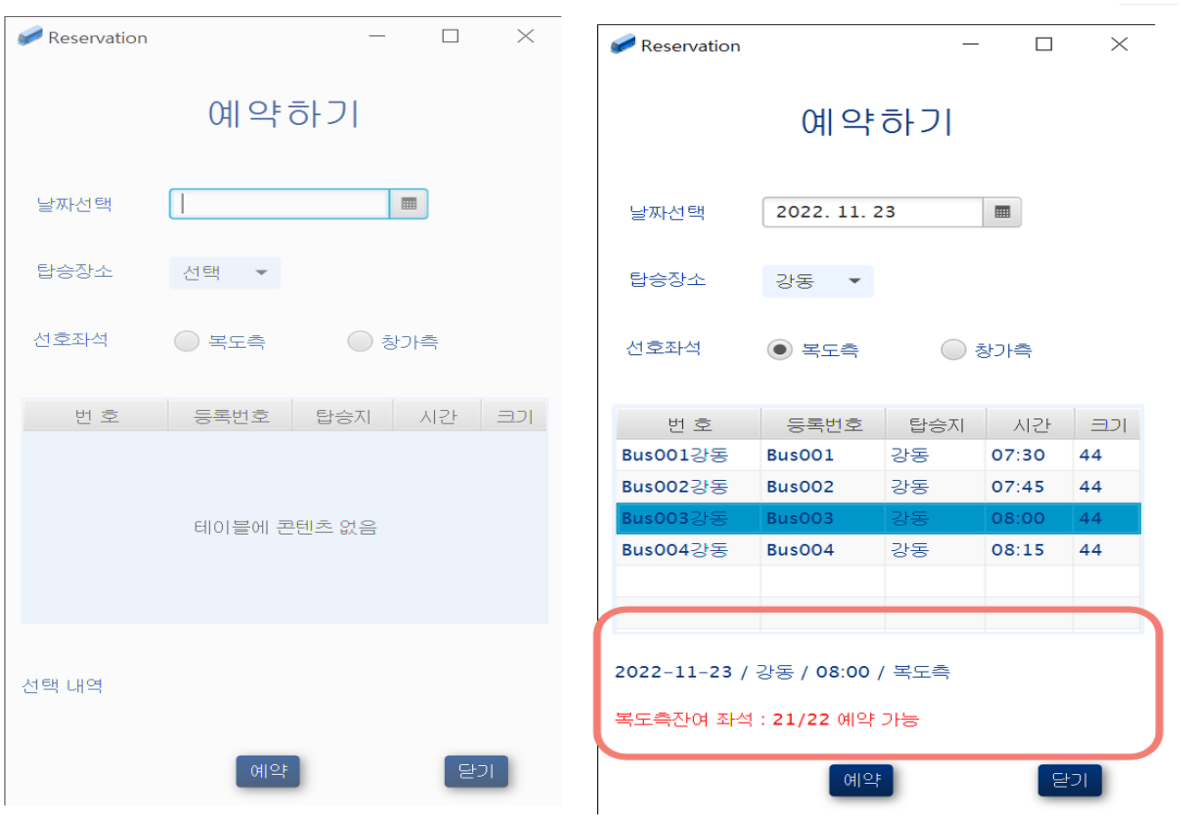
```
@FXML
private Button reservego, reserveshow, showcar, showmypage, logoutBtn;


// 예약하기(reservation.fxml) 화면으로 가기
public void reservego_me() throws Exception {
   Stage primaryStage = new Stage();
   Stage stage = (Stage) reservego.getScene().getWindow();

   Parent second = FXMLLoader.load(getClass().getResource("/reservation/reservation.fxml"));
   Scene sc = new Scene(second);
   //css 등록(11.22, 문수지 추가)
   sc.getStylesheets().add(getClass().getResource("app.css").toString());
   //창 아이콘 이미지 삽입(11.22, 문수지 추가)
   primaryStage.getIcons().add(new Image("file:lib/bus_image.png"));
   primaryStage.setTitle("Reservation");
   primaryStage.setScene(sc);
   primaryStage.show();
   stage.close();
}

// 예약현황(reserveChk.fxml) 화면으로 가기
public void reserveshow_me() throws Exception {
   Stage primaryStage = new Stage();
   Stage stage = (Stage) reserveshow.getScene().getWindow();

   Parent second = FXMLLoader.load(getClass().getResource("/reserveSearch/reserveChk.fxml"));
   Scene sc = new Scene(second);
   //css 등록(11.22, 문수지 추가)
   sc.getStylesheets().add(getClass().getResource("app.css").toString());
   //창 아이콘 이미지 삽입(11.22, 문수지 추가)
   primaryStage.getIcons().add(new Image("file:lib/bus_image.png"));
   primaryStage.setTitle("버스 예약 프로그램 / 예약확인");
   primaryStage.setScene(sc);
   primaryStage.show();
   stage.close();
}
```

```
// 배차조회(busshow.fxml) 화면으로 가기
public void showcar_me() throws Exception {
   Stage primaryStage = new Stage();
   Stage stage = (Stage) showcar.getScene().getWindow();

   Parent second = FXMLLoader.load(getClass().getResource("/bus/busshow.fxml"));
   Scene sc = new Scene(second);
   //css 등록(11.22, 문수지 추가)
   sc.getStylesheets().add(getClass().getResource("app.css").toString());
   //창 아이콘 이미지 삽입(11.22, 문수지 추가)
   primaryStage.getIcons().add(new Image("file:lib/bus_image.png"));
   primaryStage.setTitle("버스 예약 프로그램 / 배차현황");
   primaryStage.setScene(sc);
   primaryStage.show();
   stage.close();
}

// 마이페이지 조회를 위한 비밀번호 재확인화면(password.fxml) 가기
public void showmypage_me() throws Exception {
   Stage primaryStage = new Stage();
   Stage stage = (Stage) showmypage.getScene().getWindow();

   Parent second = FXMLLoader.load(getClass().getResource("/password/password.fxml"));
   Scene sc = new Scene(second);
   //css 등록(11.22, 문수지 추가)
   sc.getStylesheets().add(getClass().getResource("app.css").toString());
   //창 아이콘 이미지 삽입(11.22, 문수지 추가)
   primaryStage.getIcons().add(new Image("file:lib/bus_image.png"));
   primaryStage.setTitle("비밀번호 입력");
   primaryStage.setScene(sc);
   primaryStage.show();
   stage.close();
}


// 로그아웃(login.fxml) 화면으로 가기
public void logout(ActionEvent event) throws Exception {
     Parent members = FXMLLoader.load(getClass().getResource("login.fxml"));
     Scene scene = new Scene(members);
     Stage primaryStage = (Stage) ((Node) event.getSource()).getScene().getWindow();
     //css 등록(11.22, 문수지 추가)
     scene.getStylesheets().add(getClass().getResource("app.css").toString());
     //창 아이콘 이미지 삽입(11.22, 문수지 추가)
     primaryStage.getIcons().add(new Image("file:lib/bus_image.png"));
     primaryStage.setTitle("버스 예약 프로그램 / 로그인");
     primaryStage.setScene(scene);
    }
```

각 버튼별로 별도의 메소드로 구분하였고, 메소드 별 실제 구현 코드는 사실상 동일하며 매핑된 FXML 파일만 차이가 있다.

3.3 예약하기

<그림. 예약하기 화면>



<코드화면. 예약하기>

```
//DatePicker(11/17 윤준호 작성)
public void selectDate(ActionEvent event) {
    //선택된 날짜의 값을 문자열로 변환해서 문자열에 저장
    strDate = rDatePicker.getValue().toString();
}
```

선택된 날짜의 값을 문자열로 변환해 문자열에 저장

```
//탑승지 정보 콤보박스에서 선택할 때 이벤트(11/18 윤준호 작성 + 11/22수정)
 public void comboChanged(ActionEvent event){     //fxml 에서 연결

    table.getItems().clear();       //테이블 뷰 초기화(11/22 윤준호)
    loc = HopOnLocation.getValue();    //선택된 값 loc에 저장(11/18 윤준호)
    setCellTable();                 //테이블뷰 칼럼 설정(11/22 윤준호)
    BusDataView();                 //테이블뷰 데이터 출력(11/22 윤준호)
 }
```

사용자가 콤보 박스를 통해 탑승지를 선택할 때, 테이블 뷰를 출력해주기 위해 위와 같이 코드를 작성했다.

```
//테이블뷰의 칼럼 설정(11/22 윤준호 작성)
private void setCellTable() {
   columnbusRealcode.setCellValueFactory(new PropertyValueFactory<>("bus_realcode"));
   columnbusCode.setCellValueFactory(new PropertyValueFactory<>("bus_code"));
   columnbusLocation.setCellValueFactory(new PropertyValueFactory<>("bus_location"));
   columnbusTime.setCellValueFactory(new PropertyValueFactory<>("bus_time"));
   columnbusSize.setCellValueFactory(new PropertyValueFactory<>("bus_size"));
}
```

테이블 뷰의 칼럼을 설정해 준다.

```
//테이블 뷰에 세팅할 데이터 SQL DB에서 불러오기(11/21윤준호 작성)
public void BusDataView() {
    // sql문 작성
    String sql = "SELECT * FROM BUS WHERE Bus_location = '" +loc +"'";

    try {
       prst = con.prepareStatement(sql);   //sql문 실행
       rs = prst.executeQuery();          //sql문 실행결과 반환
       while (rs.next()) { //테이블뷰에 사용할 리스트 생성 리얼코드, 코드, 탑승지, 시간, 인승
          Blist.add(new BusListData(rs.getString(2),rs.getString(1),rs.getString(4)
                ,rs.getString(3),rs.getString(5)));

       }//end while
    } catch (Exception e) {
       System.out.println("DB 연결에 문제가 있습니다.");
       e.printStackTrace();
    } // end try
    table.setItems(Blist);
}// end method
```

DB 에 접속해 사용자가 선택한 탑승지의 버스 정보를 불러와 준다.

➔ 각 일자별로 버스를 구체화해 지정할 수도 있지만 현재 프로그램 설계상 각 탑승지의 버스들은 출발시간을 제외한 모든 정보가 동일하기에(통원/통근 버스) 일자별로 구분할 필요가 없다고 판단했다.

```
//최초 세팅(11/18 윤준호 작성 + 11/22 윤준호 추가)
@Override
public void initialize(URL location, ResourceBundle resources) {
   HopOnLocation.setItems(list);       //탑승지 콤보박스 리스트
   con = DBConnection.getConnection();    //DB접속
     Blist = FXCollections.observableArrayList();    //리스트초기화

     //테이블뷰 선택시 이벤트(11/22 윤준호 작성) -> 선택날짜, 선택 탑승지, 선택 시간, 선택 좌석 출력)
     table.setOnMouseClicked(new EventHandler<MouseEvent>() {
         @Override
         public void handle(MouseEvent event) {  //

             reserve_selected.setText(rDatePicker.getValue().toString()+" / " +
             table.getSelectionModel().getSelectedItem().getBus_location() + " / " +
             table.getSelectionModel().getSelectedItem().getBus_time() + " / " +
             seat);                                          //리스트뷰에서 선택된 값 라벨에 출력
             //버스코드 데이터 변수에 저장
             reserveBuscode = table.getSelectionModel().getSelectedItem().getBus_code();
             searchpossible();                               //잔여좌석 여부 확인 메소드 실행
             availableseat.setText(textAvailable);           //잔여좌석 표시
             availableseat.setTextFill(Color.web("FF0000")); //UI 디자인(11/22 문수지 추가)
         }
       });
}
```

Initialize 메서드에서 DB 접속을 하고 Combobox 데이터를 설정해주며 TableView 의 데이터를 초기화 해준다.

또, TableView 의 이벤트핸들러를 작성해 TableView 의 이벤트 처리를 정의해 줬다.

TableView 의 선택한 정보를 출력하기 위해 라벨에 해당 값들을 정의했고, 버스코드 데이터를 변수에 저장해 이후 잔여좌석 여부를 확인할 수 있도록 했다.

Searchpossible()메서드에서 상기 버스코드 데이터를 활용하며 여기서 도출된 결과를

Availableseat 라벨을 활용해 출력해 줬다.

```
//잔여 좌석 확인용 메서드
public void searchpossible() {
   String sql = "SELECT COUNT(*) FROM RESERVE WHERE BUS_CODE = '" + reserveBuscode + "' and LOCATION_CODE = '"
           + loc + "' and USER_FAV_SEAT = '" + seat +"'" + " and res_date = '" + strDate +"'";
   try {
      prst = con.prepareStatement(sql);   //sql문 실행
       rs = prst.executeQuery();         //sql문 실행결과 반환
       if(rs.next()) {
          reservedCount = rs.getInt(1);
          int counting = (22-reservedCount);
          if (counting >0) {
             textAvailable = seat + "잔여 좌석 : " +counting + "/22 예약 가능";
             available = true;
          } else {
             textAvailable = "예약 불가";
             available = false;
          }
       }
   }catch (Exception e) {
           System.out.println("DB 연결에 문제가 있습니다.");
          e.printStackTrace();
       } // end try
   }
```

잔여좌석 확인하기 위해 작성한 메서드

DB 접속해 상기 할당된 reserveBuscode 와 사용자가 선택한 데이터를 활용하여 Reserve 테이블의 해당정보와 동일한 예약횟수를 불러온다.

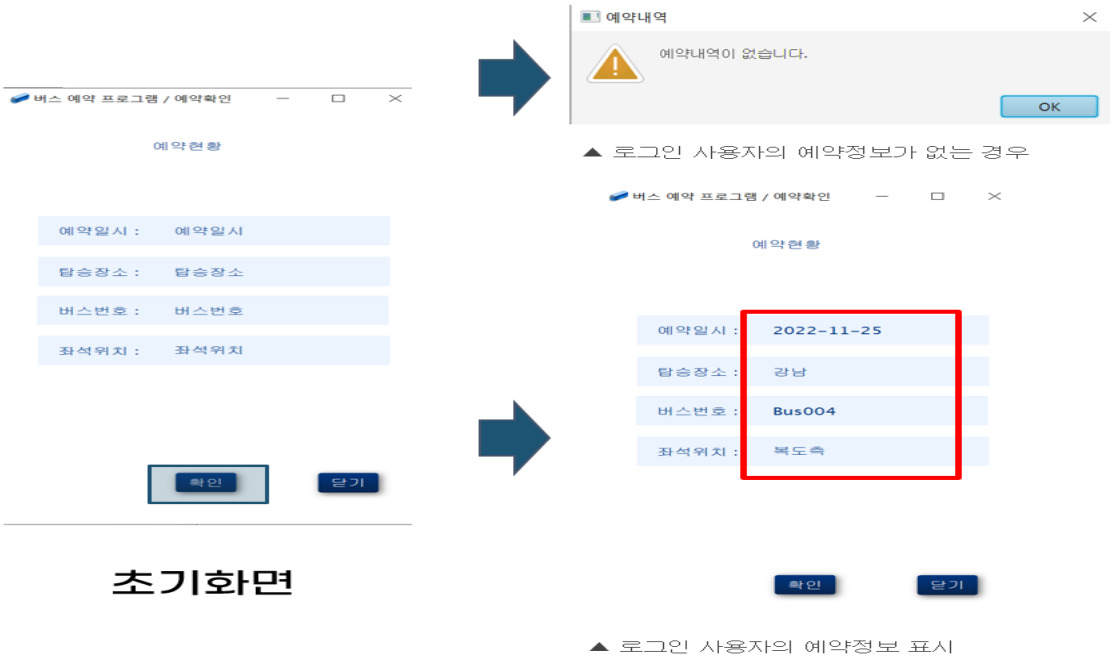
임의로 모든 버스는 44 인승으로 정의했기에 창측/복도측 각 22 인으로 설정해 22 에서 해당 예약횟수를 차감하는 것으로 잔여좌석을 보여준다.

➔ 버스 테이블에 창측/복도측 각각의 좌석을 지정해주어 해당 데이터를 활용하는 방식으로 변경해야 한다.

3.4 예약현황

예약확인 화면에서 확인버튼을 클릭할 경우 DB 내 RESERVE 테이블에서 로그인 된 사용자가 조회시점을 기준으로 가장 먼저 탑승하게 될 예약정보를 불러옴

<그림. 예약현황 화면>



<코드화면. 예약현황>

```
public void reserveView() {

  //sql문 작성
   String sql = "select to_char(res_date, 'YYYY-MM-DD') as res_date , location_code, bus_code, USER_FAV_SEAT from reserve "
              + " where user_code = (select user_code from user_main where user_id = ?) and (res_date > sysdate) order by res_date " ;
   String sql2 = "select user_code from user_main where user_id = ? " ;


   // 로그인한 user_id를 바탕으로 내부 DB에서 상응하는 user_code 가져오기
   try{
     //DB연결 및 실행
        Connection conn2 = DBConnection.getConnection();
        PreparedStatement prst2 = conn2.prepareStatement(sql2);

        prst2.setString(1, login_Controller.lid);     // 로그인한 id는 lid 변수에서 기저장, lid와 일치하는(즉 로그인id와 일치하는) user_code를 호출
        ResultSet rs2 = prst2.executeQuery();

        if (rs2.next()) {
        myUser_code = rs2.getString("user_code");
        }

   } catch(SQLException e) {
        System.err.format("SQL State: %s\n%s", e.getSQLState(), e.getMessage());
   } catch (Exception e) {
      e.printStackTrace();
   }
```

현재 로그인 사용자의 user_id정보만이 있는 상황에서 Reserve테이블에 바로 접근할 수 없고 User_main 테이블에서 user_id정보에 대응하는 user_code 정보를 먼저 받음은 후 Reserve테이블의 예약정보를 받아옴

```
// 위의 user_code를 활용하여 내부 DB의 예약 테이블에서 예약정보를 가져오기
   try{
     //DB연결 및 실행
       Connection conn = DBConnection.getConnection();
       PreparedStatement prst = conn.prepareStatement(sql);


      prst.setString(1, login_Controller.lid);     // lid 받아온것 활용
      ResultSet rs = prst.executeQuery();
   // 위에서 생성한 변수에 가져온 DB 예약정보를 저장
      if (rs.next()) {
         myres_date = rs.getString("res_date");
         mylocation_code = rs.getString("location_code");
         mybus_code = rs.getString("bus_code");
         myUSER_FAV_SEAT = rs.getString("USER_FAV_SEAT");
      }
```

```
//예약현황 조회를 위한 FXML(busshow.fxml) 컨트롤러(이하 클래스 별도 표기가 없을시 11.17, 손영석
public class ReserveSearch_controller implements Initializable {

    @Override
    public void initialize(URL location, ResourceBundle resources) {

    }

    //FXML 어노테이션 설정
    @FXML
    private Label res_date,location_code,bus_code,fav_seat;
    @FXML
    private Button res_close;


    // 예약현황을 가져와 화면표시하도록 하는 메소드
    public void getReservecheck(ActionEvent event) throws Exception {

        ReserveSearch obj_batch = new ReserveSearch();

        obj_batch.reserveView();
    // 예약내역이 없을시 경고창 출력 (11.18, 문수지 추가)
        if (obj_batch.myres_date==null) {
          Alert alert = new Alert(AlertType.WARNING);
          alert.setTitle("예약내역");
          alert.setHeaderText(null);
          alert.setContentText("예약내역이 없습니다.");
          alert.show();
        }
    // 예약한 일자, 장소, 버스번호, 선호좌석을 가져와 라벨에 입력
        else {
            res_date.setText(obj_batch.myres_date);
            location_code.setText(obj_batch.mylocation_code);
            bus_code.setText(obj_batch.mybus_code);
            fav_seat.setText(obj_batch.myUSER_FAV_SEAT);
        }
    }
```

FXML로 구현한 항목별 Label값에 Reserve테이블에서 가져온 텍스트정보를 입력

3.5 배차조회

배차조회 화면에서 사용자가 일자(Datepicker), 탑승장소(Combobox)를 선택하고 조회 버튼을 클릭할 경우 해당 일자-장소에 상응하는 모든 배차정보를 불러옴

<그림. 배차조회 화면>

▲ 버스 탑승날짜를 선택하지 않은 경우

▲ 일자-장소별 배차, 잔여좌석 정보 출력

<코드화면. 예약현황>

```
// 배차정보 및 잔여좌석현황을 가져와 화면표시하도록 하는 메소드
public void getBustime(ActionEvent event) throws Exception {

    Busshow obj_bus = new Busshow();
    obj_bus.allBusView(bus_loc);          //버스장소 String값을 입력받아 배차현황 자바파일에 저장된 버스

    //버스일자나 탑승장소가 공란일 경우 경고 메시지를 호출(11.22, 문수지 추가)
    if (bus_strDate==null) {
        Alert alert = new Alert(AlertType.WARNING);
        alert.setTitle("버스탑승날짜");
        alert.setHeaderText(null);
        alert.setContentText("버스 탑승 날짜를 선택해주십시오.");
        alert.show();
    } else if (bus_loc == null) {
        Alert alert = new Alert(AlertType.WARNING);
        alert.setTitle("버스탑승장소");
        alert.setHeaderText(null);
        alert.setContentText("버스 탑승 장소를 선택해주십시오.");
        alert.show();
    } else {
```

버스정보 조회를 위한 탑승일자, 탑승장소를 선택하지 않은 경우에는 오류메세지를 호출

```
    // 각 해당 라벨에 buslist 배열에 저장되어있던 값을 불러와 입력
    first_bustime.setText(obj_bus.buslist.get(0)[2]);//첫번째 버스 시간
    second_bustime.setText(obj_bus.buslist.get(1)[2]);//두번째 버스 시간
    third_bustime.setText(obj_bus.buslist.get(2)[2]);//세번째 버스 시간
    fourth_bustime.setText(obj_bus.buslist.get(3)[2]);//네번째 버스 시간

    first_buscode.setText(obj_bus.buslist.get(0)[0]);//첫번째 버스 번호
    second_buscode.setText(obj_bus.buslist.get(1)[0]);//두번째 버스 번호
    third_buscode.setText(obj_bus.buslist.get(2)[0]);//세번째 버스 번호
    fourth_buscode.setText(obj_bus.buslist.get(3)[0]);//네번째 버스 번호

    // 선택한 탑승장소, 일자에 해당하는 배차정보에서 이미 예약된 좌석수를 제외한 잔여좌석수를 가져옴, 잔여좌석수는 적색
    obj_bus.left_seat_bus("Bus001", bus_loc, bus_strDate);
    leftseat_bus1.setText("잔여좌석현황 " + Integer.toString(obj_bus.bus_remainseat) + " / "
    leftseat_bus1.setTextFill(Color.web("#FF0000"));
    obj_bus.left_seat_bus("Bus002", bus_loc, bus_strDate);
    leftseat_bus2.setText("잔여좌석현황 " + Integer.toString(obj_bus.bus_remainseat) + " / "
    leftseat_bus2.setTextFill(Color.web("#FF0000"));
    obj_bus.left_seat_bus("Bus003", bus_loc, bus_strDate);
    leftseat_bus3.setText("잔여좌석현황 " + Integer.toString(obj_bus.bus_remainseat) + " / "
    leftseat_bus3.setTextFill(Color.web("#FF0000"));
    obj_bus.left_seat_bus("Bus004", bus_loc, bus_strDate);
    leftseat_bus4.setText("잔여좌석현황 " + Integer.toString(obj_bus.bus_remainseat) + " / "
    leftseat_bus4.setTextFill(Color.web("#FF0000"));
    }
```

불러온 버스정보를 구현화면 Label 에 텍스트 값을 입력, 잔여좌석정보는 가시성을 고려하여 setTextfill 함수를 활용하여 적색으로 표기

```
//버스배차현황 및 잔여좌석 호출용도(이하 클래스 별도 표기가 없을시 11.21, 손영석 김유진 공동작성)
public void allBusView(String bus_loc) {

    // sql문 작성
   String sql = "SELECT * FROM bus where bus_location = ?";

 try {
     //DB연결
    Connection con = DBConnection.getConnection();
    PreparedStatement prst = con.prepareStatement(sql);
    //sql문 실행
    prst.setString(1, bus_loc);
    ResultSet rs = prst.executeQuery();
    //sql문을 통해 불러온 값들은 buslist 배열에 저장
    while (rs.next())
    {
       String[] arrStr = { rs.getString("bus_code"), rs.getString("bus_size"),rs.getString("bus_time"),

       buslist.add(arrStr);
```

```
// 버스잔여 좌석 가져오기
public int left_seat_bus(String bus_code, String bus_loc, String bus_date) {


   // sql문 작성
   String sql = "SELECT count(*) as reservecount FROM reserve where bus_code = ? and location_code = ? and res_date = ? ";
   String sql2 = "SELECT bus_size FROM bus where bus_code = ? and bus_location = ?  ";


    // 버스 총좌석수 호출
    while (rs2.next())
    {
       bus_totalseat = rs2.getInt("bus_size");
    }
    // 예약현황 좌석개수를 호출
    while (rs.next())
    {
       bus_occupiedseat = rs.getInt("reservecount");
    }
    //전체좌석수에서 예약된좌석수를 차감하여 잔여좌석수를 산출
    bus_remainseat = bus_totalseat - bus_occupiedseat;
```

버스잔여좌석 정보를 조회하기 위해서 Bus 테이블에서 버스크기(즉 총좌석수)를 가져오고, Reserve 테이블에서 사용자가 선택한 일자, 장소 등과 일치하는 예약정보의 개수를 가져와서 차감함으로써 잔여좌석을 구함

3.6 마이페이지

로그인한 사용자가 마이페이지에 접근하려는 경우 비밀번호를 재입력하도록 요구하여 검증하는 단계를 거치며, 본인의 회원가입정보를 조회하고, 필요시 이를 수정할 수 있는 기능을 구현

<그림. 마이페이지(비밀번호 재검증) 화면>

보안 절차 비밀번호 확인

비밀번호 일치

비밀번호 불일치

<그림. 마이페이지(회원가입정보 조회 및 수정) 화면>

로그인 회원 정보조회

회원 정보 수정

비밀번호 조건 충족

비밀번호 조건 불충족

<코드화면. 마이페이지(비밀번호 재검증)>_추가

```
//마이페이지 회원정보 확인 클릭시 보안상 비밀번호 요구 및 확인 11/21 손영석 작성
public void chk(ActionEvent event) throws Exception {
    System.out.println("성공1");
    passwordDTO pDTO = new passwordDTO();   //객체 생성
    try {
        pDTO.setUser_pw(inputPW.getText());
        String getPW = inputPW.getText().toString();     //비밀번호 입력창에 작성된 비밀번호 입력
        passwordDAO obj = new passwordDAO();
        lid = pDTO.getUser_pw();                         //로그인한 회원정보가 들어있는 전연변수, 로그인 비밀번호와 동일한지 비교
        if (obj.login(getPW)==1) {                       //로그인 비밀번호와 동일하면 성공
            try {
                Parent members = FXMLLoader.load(getClass().getResource("/mypage/Mypage.fxml"));
                Scene scene = new Scene(members);
                Stage primaryStage = (Stage) ((Node) event.getSource()).getScene().getWindow();
                //css 등록
                scene.getStylesheets().add(getClass().getResource("/main/app.css").toString());
                //창 아이콘 이미지 삽입
                primaryStage.getIcons().add(new Image("file:lib/bus_image.png"));
                primaryStage.setTitle("버스 예약 프로그램 / 마이페이지");
                primaryStage.setScene(scene);
            } catch (Exception e) {
                e.printStackTrace();
            }
        } else if (getPW.isEmpty() || getPW.contains(" ")) {
            System.out.println("PW 미입력");
            System.out.println(obj.login(getPW));
            Alert alert = new Alert(AlertType.WARNING);
            alert.setTitle("PW를 입력해주세요.");
            alert.setHeaderText(null);
            alert.setContentText("PW를 입력해 주십시오.");
            alert.show();
        }
        else {                                           //로그인 비밀번호와 틀릴시 실패
            System.out.println("실패");
            System.out.println(obj.login(getPW));

            try {
                Alert alert = new Alert(AlertType.WARNING);
                alert.setTitle("비밀번호 불일치");
                alert.setHeaderText(null);
                alert.setContentText("비밀번호가 틀렸습니다.");
                alert.show();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
            System.out.println("contentPW: " + getPW);
        } catch (IllegalArgumentException e) {
            System.out.println("IllegalArgumentException caught"); // 예외처리 발생!!
        }
```

마이페이지를 클릭하면 보안 절차로 비밀번호 확인창을 호출하며, gettext 로 입력받은 비밀번호를 DAO,DTO를 통해 로그인한 회원 정보가 들어있는 lid 전역변수와 비교하여 비밀번호 일치 여부를 확인

<코드화면. 마이페이지(회원정보 호출)>

```
//마이페이지 회원정보확인 11/21 손영석 작성
public void getMypagecheck() throws Exception {
    Mypage obj_batch = new Mypage();  //객체생성
    obj_batch.mypagesonView();
    //회원정보 호출
    useridlabel.setText(obj_batch.userid);                  //아이디
    userpwlabel.setText(obj_batch.userpw);                  //비번
    usernamelabel.setText(obj_batch.userName);              //이름
    userphonelabel.setText(obj_batch.userPn);               //핸드폰번호
    useremaillabel.setText(obj_batch.usereEmail);           //이메일
    userlocationlabel.setText(obj_batch.userAddress);       //지역
    userseatlabel.setText(obj_batch.UserFavSeat);           //선호좌석
}
```

마이페이지에 정보조회 버튼 클릭하면 userview를 활용하여 회원 정보를 호출

<코드화면. 마이페이지(회원정보 수정 업데이트)>

```
//회원정보 업데이트 버튼 수행 11/22 손영석 작성
try {

    uDTO.setUser_pw(aspw.getText());         //새로운 비밀번호 입력
    uDTO.setUser_name(asname.getText());     //새로운 이름 입력
    uDTO.setUser_phone(asphone.getText());   //새로운 번호 입력
    uDTO.setUser_email(asemail.getText());   //새로운 이메일 입력

    UpdateDAO.Signupupdate(uDTO);            //DB전달
      Alert alert = new Alert(AlertType.INFORMATION);
      alert.setTitle("가입 알림창");
      alert.setHeaderText(null);
      alert.setContentText("회원정보 수정 완료되었습니다.");
      alert.show();
} catch (IllegalArgumentException e) {
    System.out.println("IllegalArgumentException caught"); // 예외처리 발생!!
} catch (Exception e) {
    e.printStackTrace();
}
public class UpdateDAO {
    //회원 정보 페이지에 새롭게 입력된 정보를 DB에 업데이트 11/22 손영석 작성
    public static void Signupupdate(UpdateDTO uDTO) {
            String sql = "update user_main " //누락된 set값 발생오류--> sql문 작성시 띄어쓰기 안함으로 구문이 붙어 오류발생
                    + " SET user_pw = ?, user_name = ?, user_phone = ?, user_email = ? "
                    + " where user_id = ? ";
        try{
                Connection conn = DBConnection.getConnection();          //DB연결
                PreparedStatement prst = conn.prepareStatement(sql);     //SQL문 실행
                prst.setString(1, uDTO.getUser_pw());        //SQL문 1번째 ?에 새로입력받은 비밀번호 입력
                prst.setString(2, uDTO.getUser_name());      //SQL문 2번째 ?에 새로입력받은 이름 입력
                prst.setString(3, uDTO.getUser_phone());     //SQL문 3번째 ?에 새로입력받은 번호 입력
                prst.setString(4, uDTO.getUser_email());     //SQL문 4번째 ?에 새로입력받은 이메일 입력
                prst.setString(5, login_Controller.lid);     //SQL문 5번째 ?에 회원정보 전역변수 입력
                prst.executeUpdate();
        } catch(SQLException e) {
            System.err.format("SQL State: %s\n%s", e.getSQLState(), e.getMessage());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

업데이트 버튼을 누르면 회원 정보 수정이 가능하며, getText로 입력 받은 비밀번호는 조건을 충족하는지 체크버튼을 통해 재확인하고 오류가 없을시 입력받은 새로운 정보들로 수정된 회원 정보로 DB에 업데이트

4. 결론

4.1 향후 개선 가능사항

a. Version1.0 (버그 수정, 기능 개선)

- ID, PW 찾기 (DB의 다른 정보와 비교해 일치하는 항목을 통해 조회해서 반환해주는 구조)

- 예약현황 : 여러 항목을 스크롤을 통해 확인하는 기능 (Listview 로 변경 혹은 페이지 넘김으로)

날짜 선택 기능 (DatePicker 활용)

- 마이페이지 : PW 숨김 처리 (로그인 시 구현했던 동일 기능 추가)

- 예약하기 : 출발지 및 목적지를 설정(콤보 박스 활용)

- 관리자 페이지 추가 : 관리자용 예약통계, 데이터시각화 화면 구현, 노선 추가 등 (프로젝트 진행하며 다뤘던 기능들 통합한 페이지. DB연결, 수정, 차트화 수치 표현 등)

b. Version2.0

- 비밀번호 입력 순간순간 조건 충족 여부 알려주는 기능

- 예약 취소 기능

- 좌석 배치도를 통해 원하는 좌석을 선택하는 예약 기능 구현

c. Version3.0 (상용화 수준)

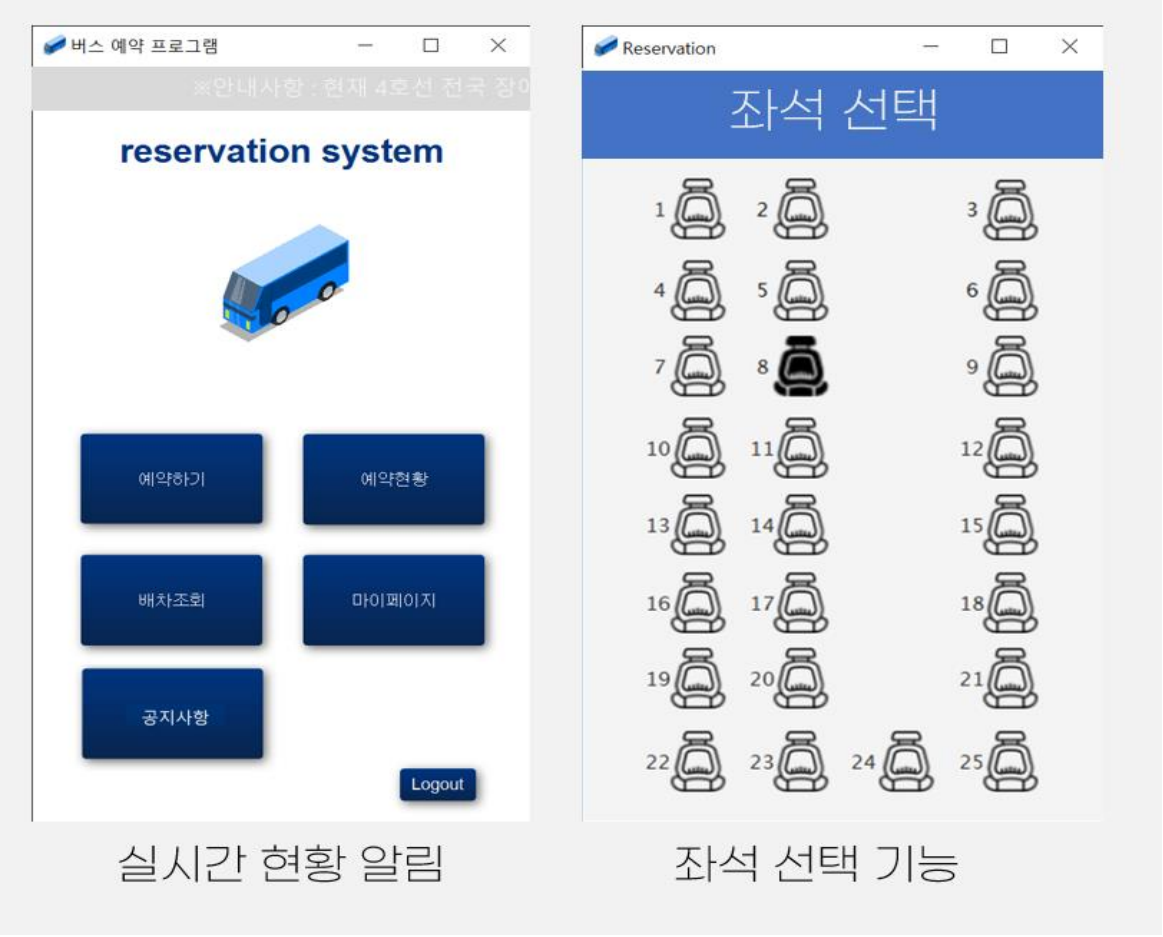
- 공지사항 게시판(시위현황, 업데이트 사항 등) 추가

- 실시간 사고현황, 지하철 지연 현황 등 표시해 줄 수 있는 기능(웹 크롤링 등의 지식 요할 것으로 예상)

- 회원신청한 가입자의 PW를 암호화하여 DB저장(데이터 변환을 통한 보안 강화)

- 결제 기능(향후 다른 방향의 프로그램으로 발전 가능)

<그림. 예상 구현화면>



4.2 팀원 역할분담 및 소감

초기 전체 코드의 통일성을 위해 가이드라인을 정한 뒤, 페어프로그래밍을 통해 프로젝트를 시작했다. 김유진, 손영석이 한 팀, 문수지, 윤준호가 한 팀으로 각각 로그인, 회원가입 기능을 구현했다. 해당 작업을 통해 향후, 프로젝트의 방향성 및 코드의 일관성을 유지하고자 했다.

이후 작업은 개인 역량에 따라 분담해 진행했고 진행 과정 중 작업속도 및 프로젝트 일정에 따라 세부 조율을 했다. 기능 추가, 일부 기능 변경 등의 작업을 거쳤고 제한된 일정에 맞춰 주요 기능이 들어간 프로그램을 완성했다. 프로토타입 완성 후 런 테스트를 거쳐 ID,PW 검증, 예약여부 확인, 잔여좌석 확인 등의 세부 기능들을 구현했다.

4.2.1 팀원 역할분담

이름	설계 단계	개발 구현 단계	개발 구현 세부 사항	개발 검증 / 발표
김유진	주제 선정 토의 참여	데이터베이스 모델링(SQL): 코드 작성(java)	코드 작성 홈화면, 예약조회, 배차조회 설계	시연 영상 녹화 및 편집
문수지	주제 선정 토의참여	코드 작성(java, css)	코드 작성 회원가입 화면 제약조건 세부 버튼들의 기능 설계(각 화면 별 알림 창)	발표 ppt 제작
손영석	주제 선정	코드 작성(java)	코드 작성 로그인, 마이페이지 설계	발표 스크립트 작성 및 발표 발표 영상 편집
윤준호	도출된 주제 통해 전체 일정 계획 전체 프로젝트 가이드라인	데이터 베이스 모델링(SQL) 코드 작성(java)	코드 작성 회원가입 DB저장 기능 구현 예약하기 페이지 설계	발표 ppt 제작

4.2.2 프로젝트 수행소감

윤준호

팀프로젝트의 성격상 초기 가이드라인을 설정할 때 구체화시키는 작업이 상당히 중요하다고 느꼈다. 첫 회의 간 막연하게 이야기한 사항들은 실제 구현할 때, 어려움이 있었고 구체적으로 의견을 나눴던 항목들은 비교적 빠른 시간에 구현이 되었다. 프로젝트 리더로서 팀원들의 의견을 취합해 프로젝트의 방향을 설정했는데, 초기 설계 단계에서 제법 많은 시간을 투자한 것이 성공적으로 프로젝트를 끝내는데 주요했다고 생각한다. 드로잉을 통해 화면을 구상했고 그 과정에서 빠르게 구현할 수 있는 기능, 시간이 필요한 기능들을 나눠, 향후 추가할 계획을 세워 우선순위를 정했고 그에 따라 일정을 조절해 늦춰지거나 딜레이 되는 일 없이 프로젝트를 끝마쳤다. 개인적으로 일정에 차질 없이 프로젝트를 끝내고 싶어 시간을 할애해 취합 및 분배를 내 선에서 처리했는데 이 또한, 작업간 어려움을 줄여줬던 것 같다. 작업 전에 패키지, 코드 작성, 주석 작성 등의 사항들에 대한 가이드라인, ppt 레이아웃 등을 제시했고 각 개인의 작업 후 취합시에 많은 시간이 필요하지 않았다. 또, 각 팀원들의 능력을 파악해 SQL 경험이 있던 김유진 팀원에게 DB에 조금 더 많은 양을, 디자인 경험이 있는 문수지 팀원에게 프로그램 레이아웃 작업을 우선 분배하고 그 외에 다른 작업을 나와 손영석 팀원이 분담해 프로젝트를 진행했는데, 이 과정이 원활하게 이루어져 큰 탈 없이 프로젝트를 끝냈다. 끝내고 난 뒤 프로그램을 보았을 땐, 부족한 점도 많이 보이

고 개선사항, 추가 사항 등 아쉬운 점이 있으나 첫 프로젝트인만큼 다음 프로젝트의 방향성이나 강도 등을 기억하며 더 잘 해낼 다짐을 하며 마무리했다.

코드 작성에 있어서 느꼈던 건 기본이 제일 중요하며, 사소한 사항들 또한 무시할 수 없다는 것이었다. 구글링, 스택오버플로우 등의 검색을 통해 해결하는 건 한계가 있고, 결국 스스로 코드의 구조를 알아야 문제의 명확한 원인을 파악하고 해결할 수 있었다.

값 할당 이후 출력을 해야 했던 메소드의 실행 순서를 잘못 설정해 에러가 났던 부분, SQL문 작성에 있어서 DB의 구조를 제대로 이해하고 있어야 효율적인 코드를 짤 수 있었던 부분 등 시간을 많이 뺏기지 않아도 될 부분들이 있었기에 보완의 필요성을 느꼈다.

김유진

실제 초기 구상과는 달리 미니 프로젝트 성과물을 만들어내는 과정에서 생각하지 못했던 어려움들을 경험하는 과정들을 통해 막연하던 개발과정이 조금은 더 구체화되었다고 느꼈음

이를 테면 DB접속 후 데이터를 받아오도록 하는 테스트과정에서, 자바소스 로직에는 문제가 없었지만 데이터베이스 샘플데이터 입력시 Insert후 commit 수행을 누락함으로써 데이터조회가 되지 않아 상당한 시간을 허비하였음 이처럼 사소할 수 있지만 기본적인 사항들부터 꼼꼼히 체크해 나가며 개발을 진행할 필요성을 느꼈음

문수지

강의시간에서 배운 내용들을 프로젝트를 통해 실제로 써볼 수 있어서 좋았습니다. 실제로 구현했을 때 발생하는 예상치 못한 오류를 잡기 위해 작성한 코드를 하나하나 확인하는 과정에서도 많은 것을 배울 수 있었습니다.

특히, DB와 자바 코드를 연결하는 과정에서 생기는 오류를 처음 접하였을 때는 시간이 다소 소요됐지만, 한번 접하고 난 뒤에는 비교적 원인을 쉽게 찾을 수 있었습니다.

또한, 막연하게 구상했던 코드를 작성하는 중에 풀리지 않을 때는 팀원들과 함께 해결할 수 있었습니다. 이번 프로젝트를 통해 배웠던 것들이 실제로 어떤 상황에서 쓰이는지 경험해볼 수 있었고 팀원들과 함께 작성한 코드의 결과물을 보며 뿌듯함을 느낄 수 있었습니다.

손영석

여러 웹페이지에서 사용하는 기능들을 직접 구현해보면서 실제로 이러한 기능들을 구현하는 과정

이 어떻게 진행되는지 직접 경험할 수 있는 좋은 기회였습니다.

교육과정에서 배운 지식을 토대로 복합적으로 응용하는 과정이 어려웠고 Java, Javafx, SQL 마다 더 높은 이해도와 공부가 필요하다는 걸 느꼈습니다.

코드를 만들면서 단순한 띄어쓰기나 오타로 인해 발생한 오류를 오랜 시간 발견하지 못하다 수정하면서 코드의 전문성도 중요하지만 사소한점도 발견할 수 있는 집중력과 섬세함도 필요하다는 것을 배웠습니다.

코드의 구성을 쉽게 찾지 못하고 오류를 해결하지 못할 때 팀원들의 도움을 받으며 해결해 나가면서 팀원들의 중요성을 느꼈습니다.