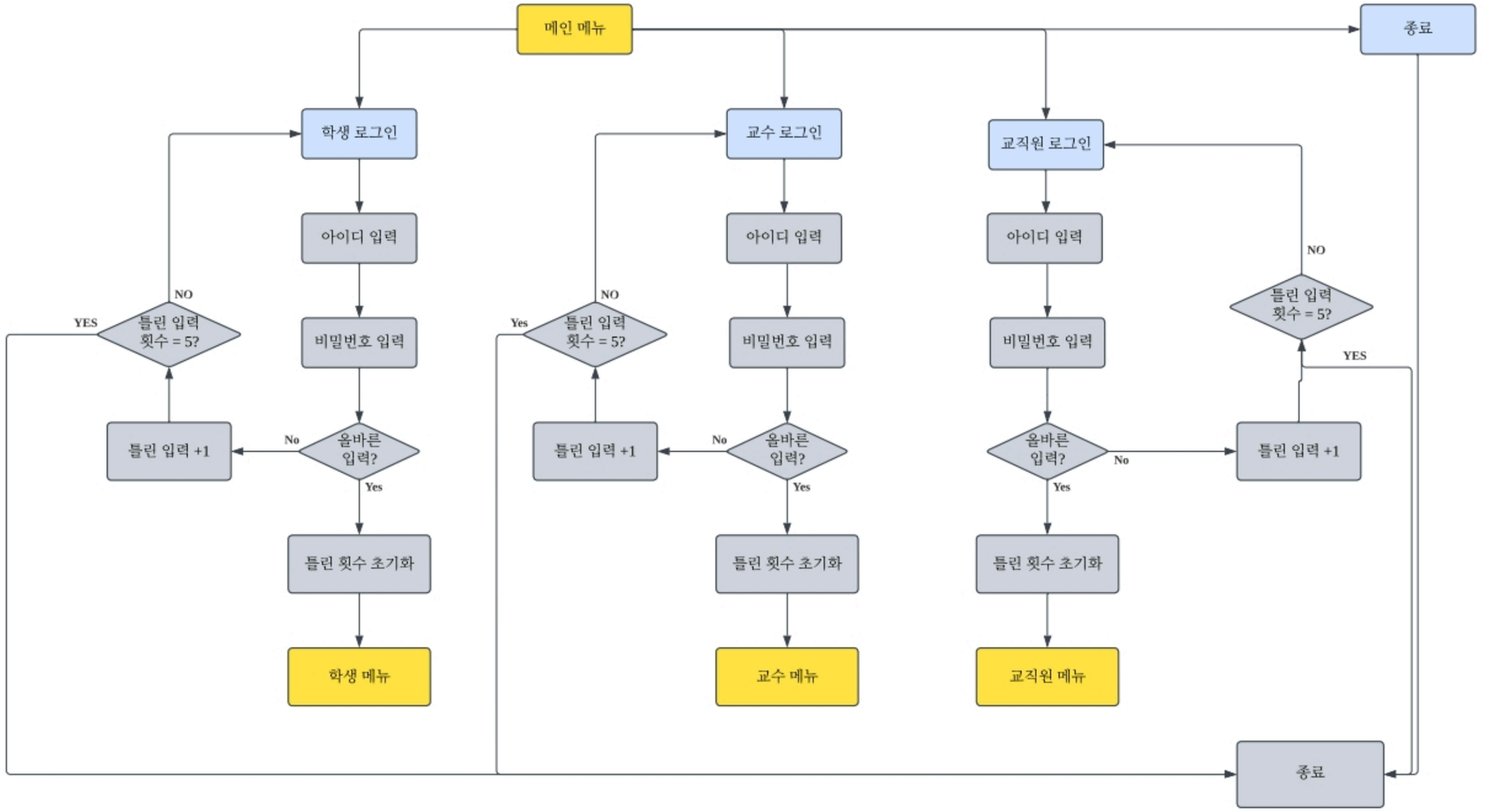
학사관리 시스템

20230B234 최호수(조장)
20210B___ 이_우
20230B___ 정_원
20220B___ 김_형

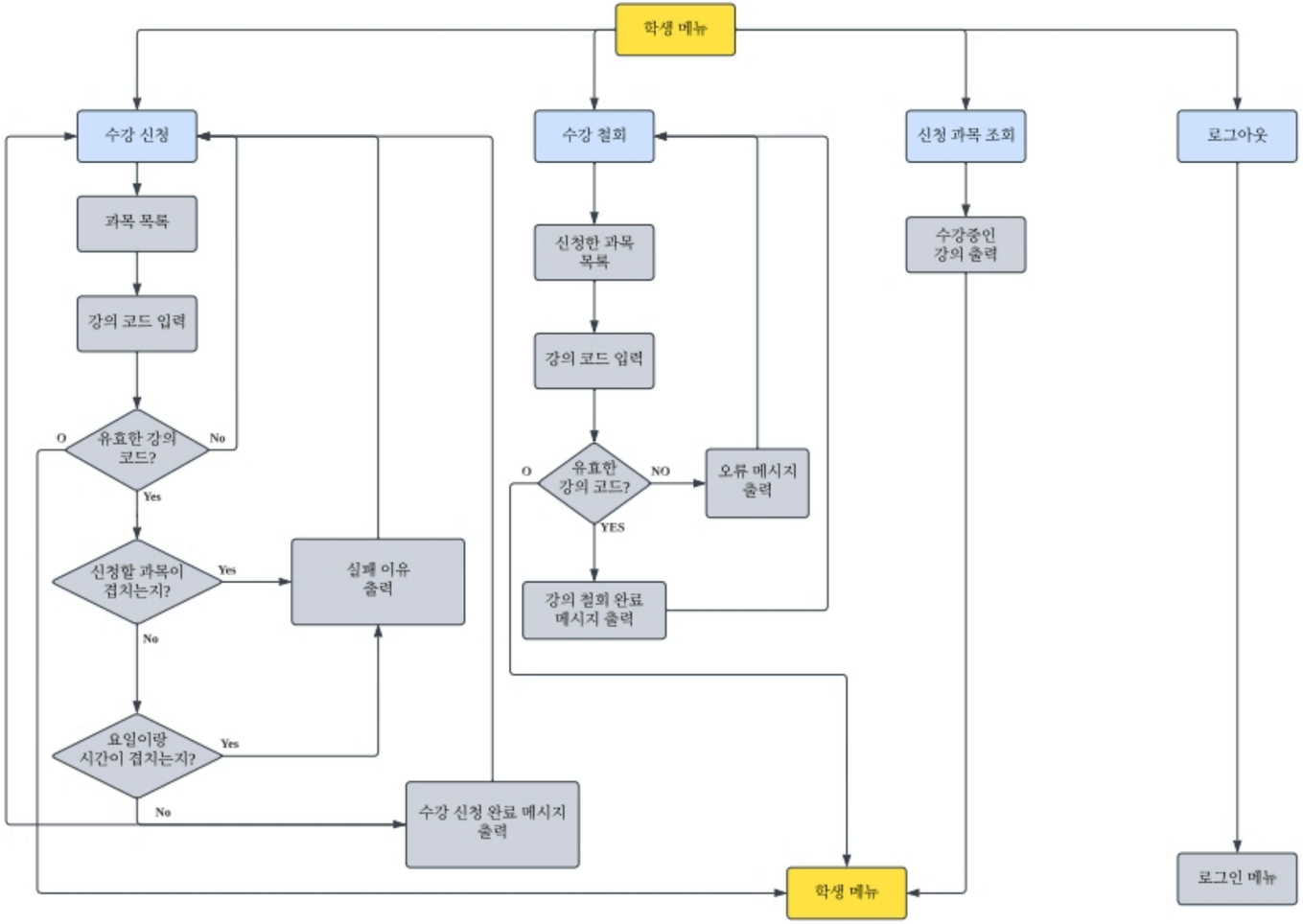
목차

CONTENTS

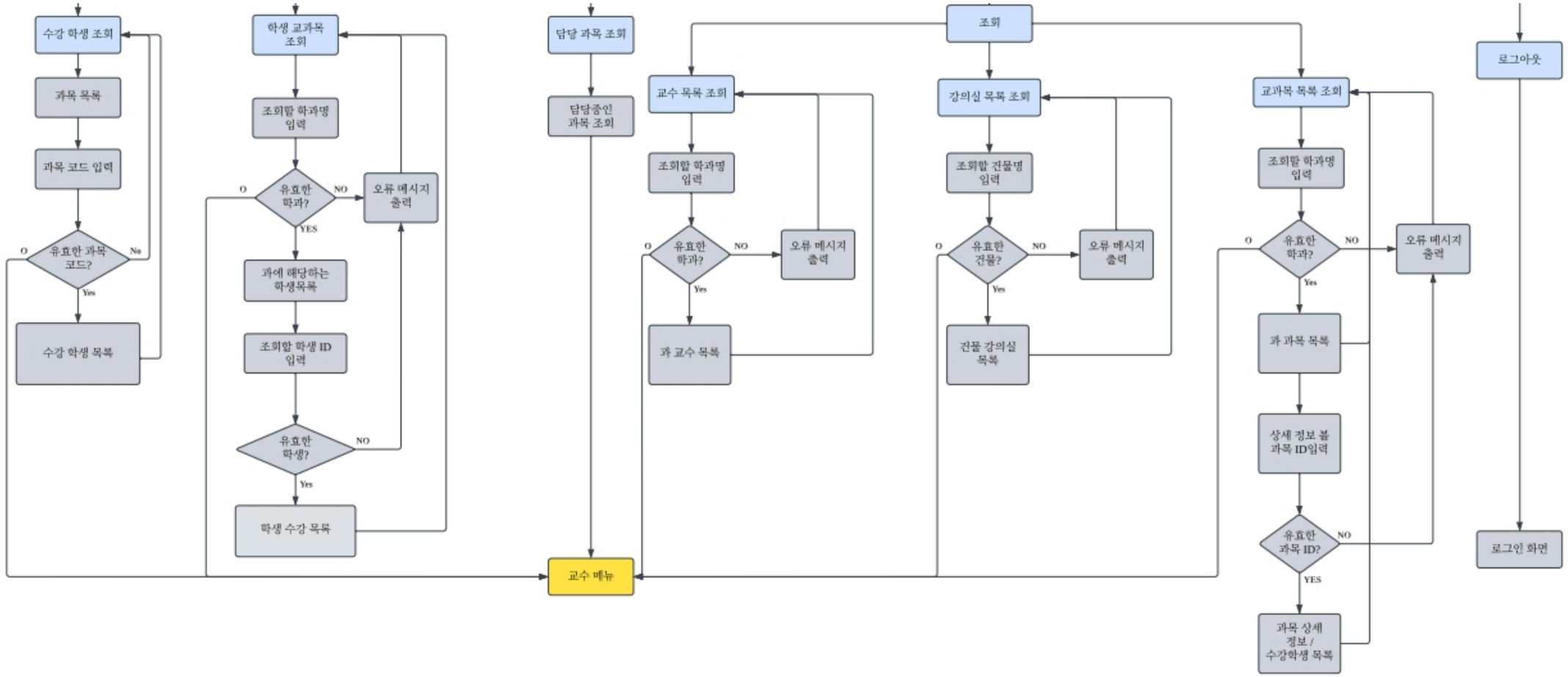
구현기능(플로우차트-메인)



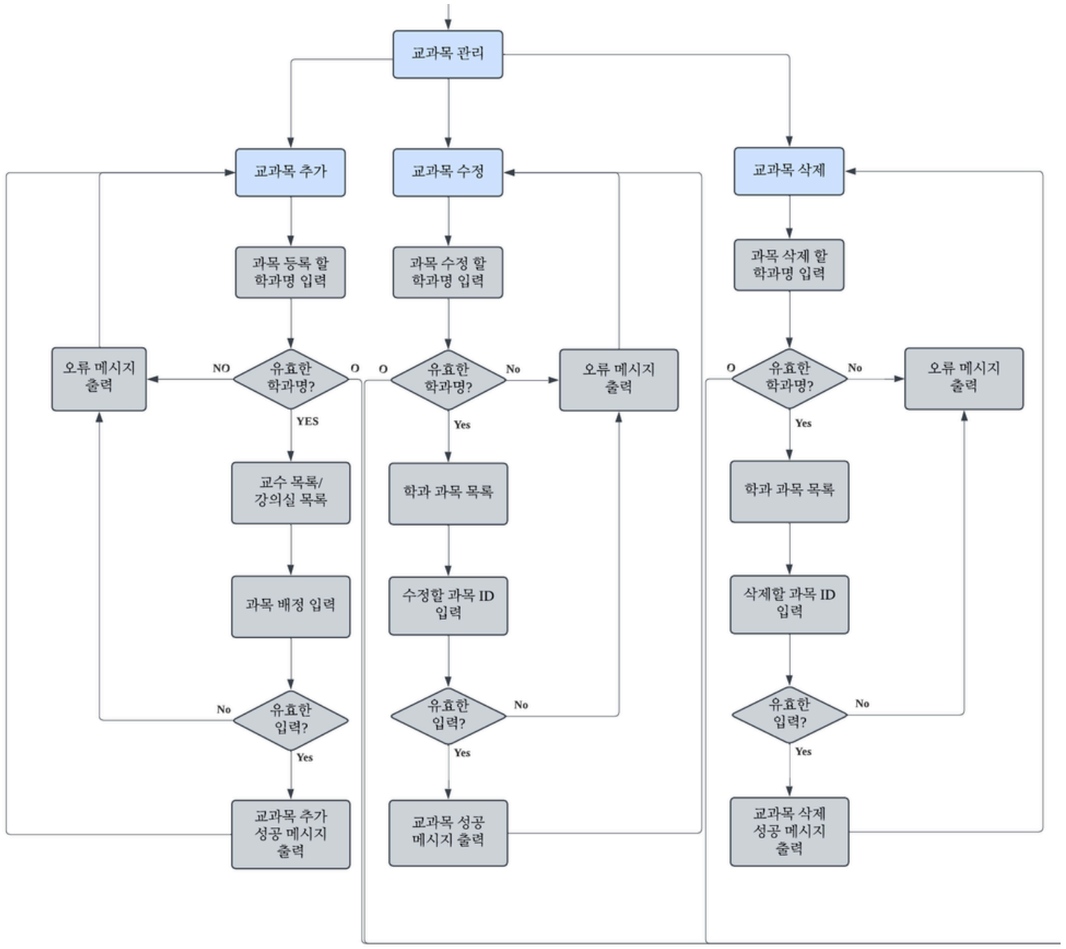
구현기능(플로우차트-학생)



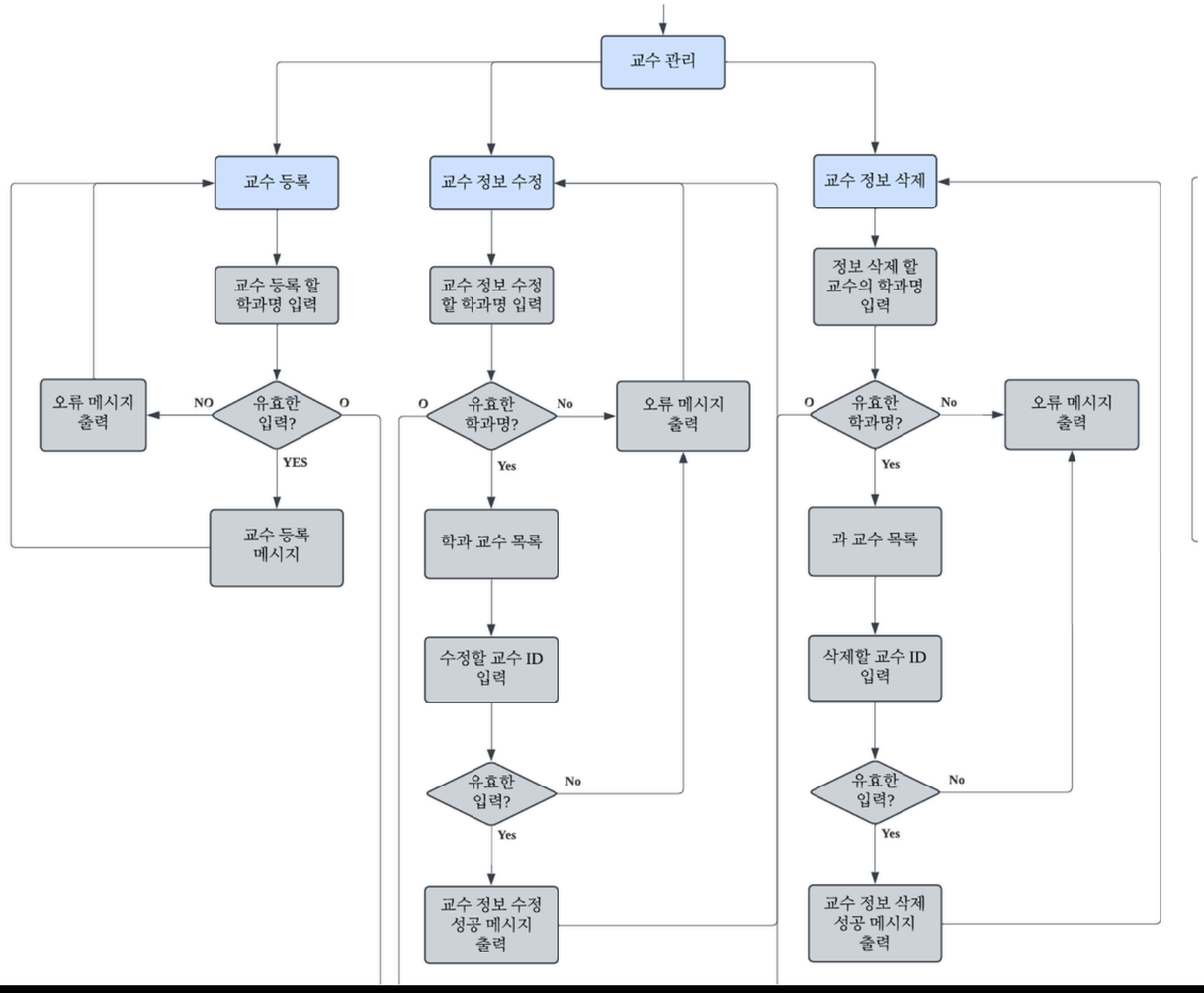
구현기능(플로우차트-교수)



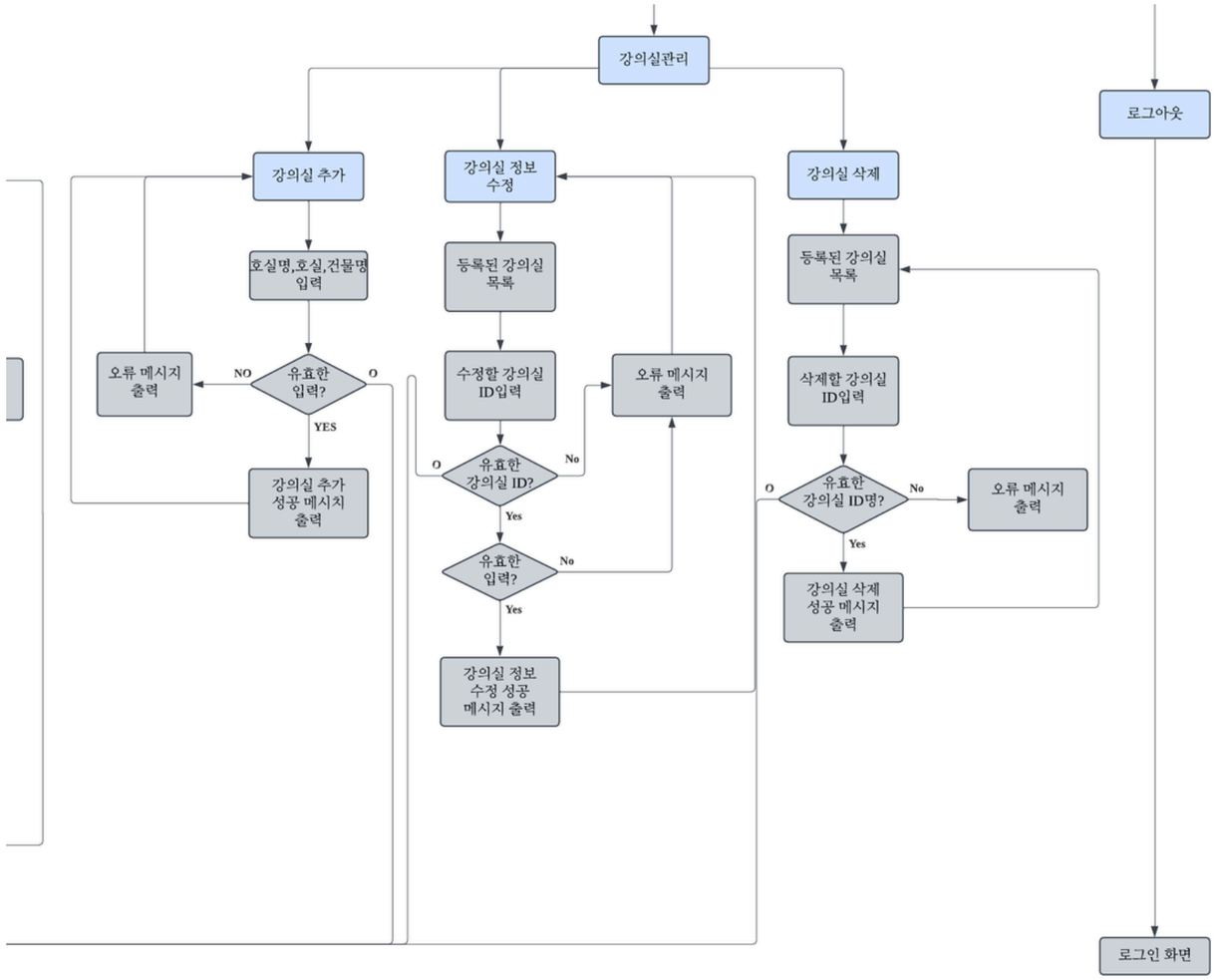
구현기능(플로우차트-교직원)



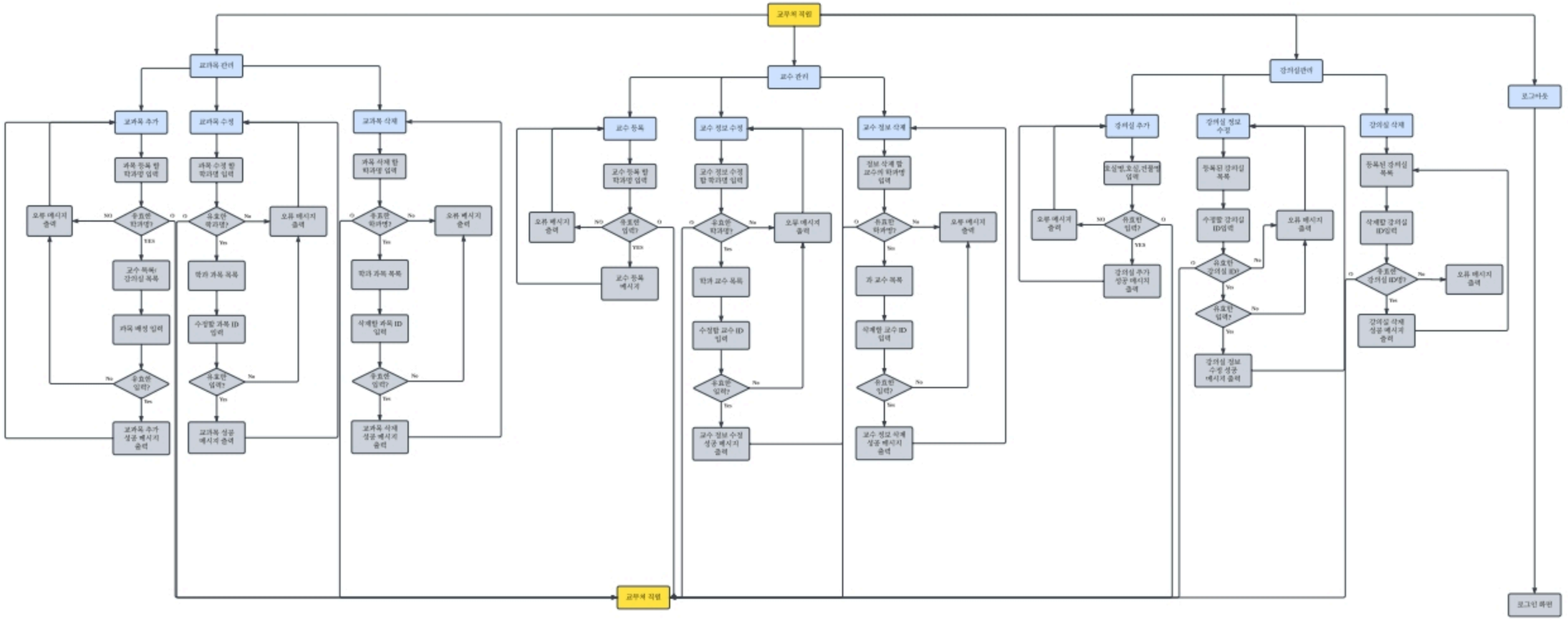
구현기능(플로우차트-교직원)



구현기능(플로우차트-교직원)



구현기능(플로우차트-교직원)



구현기능(앱)

1. 앱 처음 실행 시 나오는 화면

2 . 로그인, 회원가입

3. 학생으로 로그인 했을 때 화면

3-1. 수강 신청과 취소 버튼

3-2. 현재 신청한 교과목을 볼 수 있는 창

구현기능(앱)

1. 교수 메인 화면

1-1. 교수가 담당하는 과목 목록

1-2. 담당하는 과목을 눌렀을 때 과목을 수강하는 학생 목록

1-3. 교과목 조회 시 그 학생의 교과목 목록을 볼 수 있음

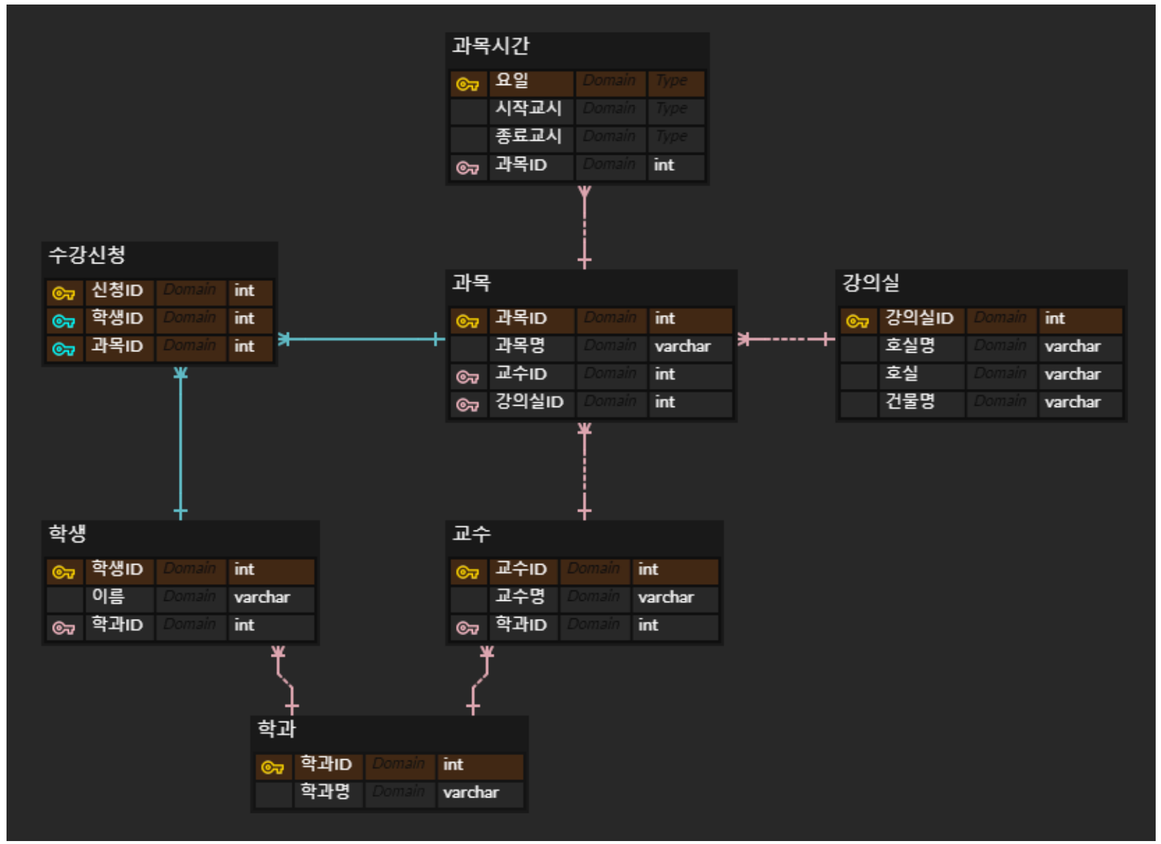
2. 교수 목록, 강의실 목록, 학과 교과목 목록 버튼

3. 교직원 메인 메뉴

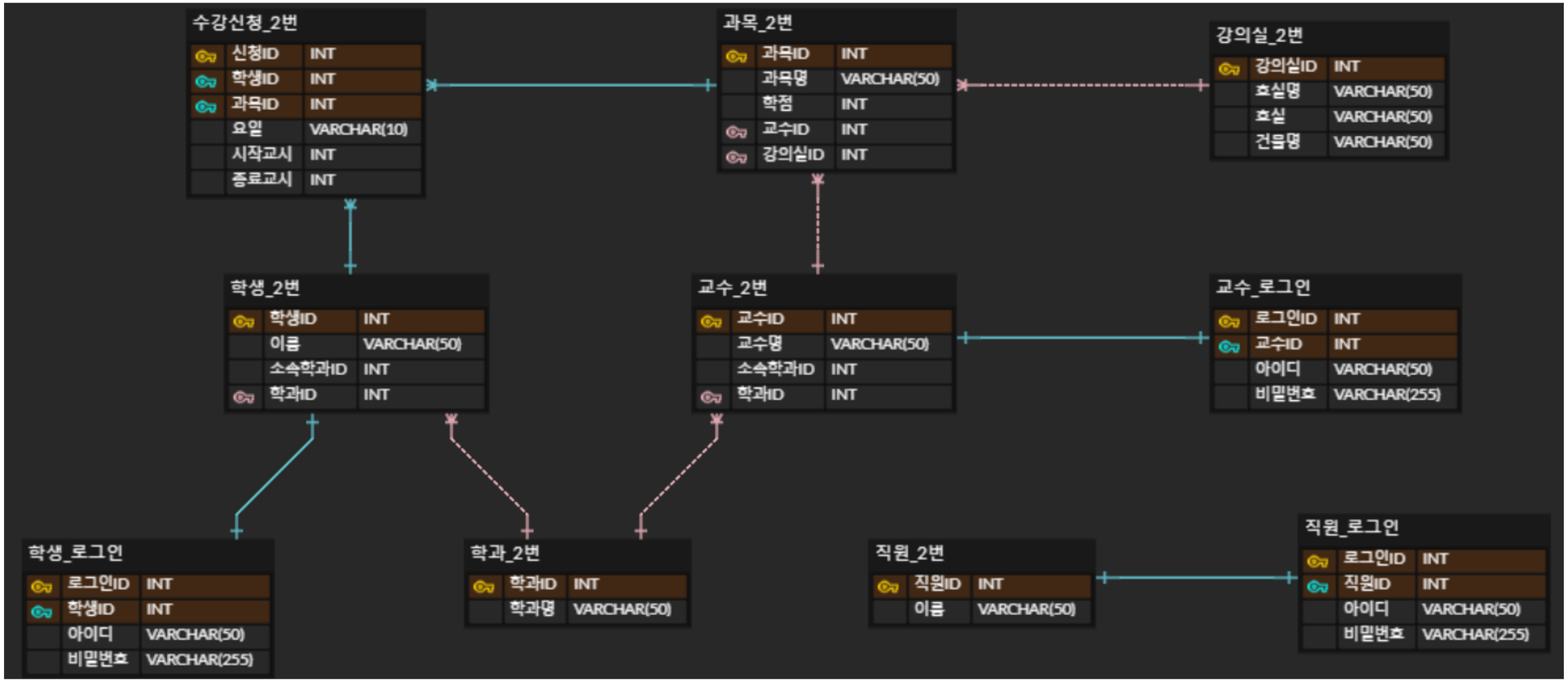
3-1. 교직원 관리 메뉴

3-2. 교직원 조회 메뉴

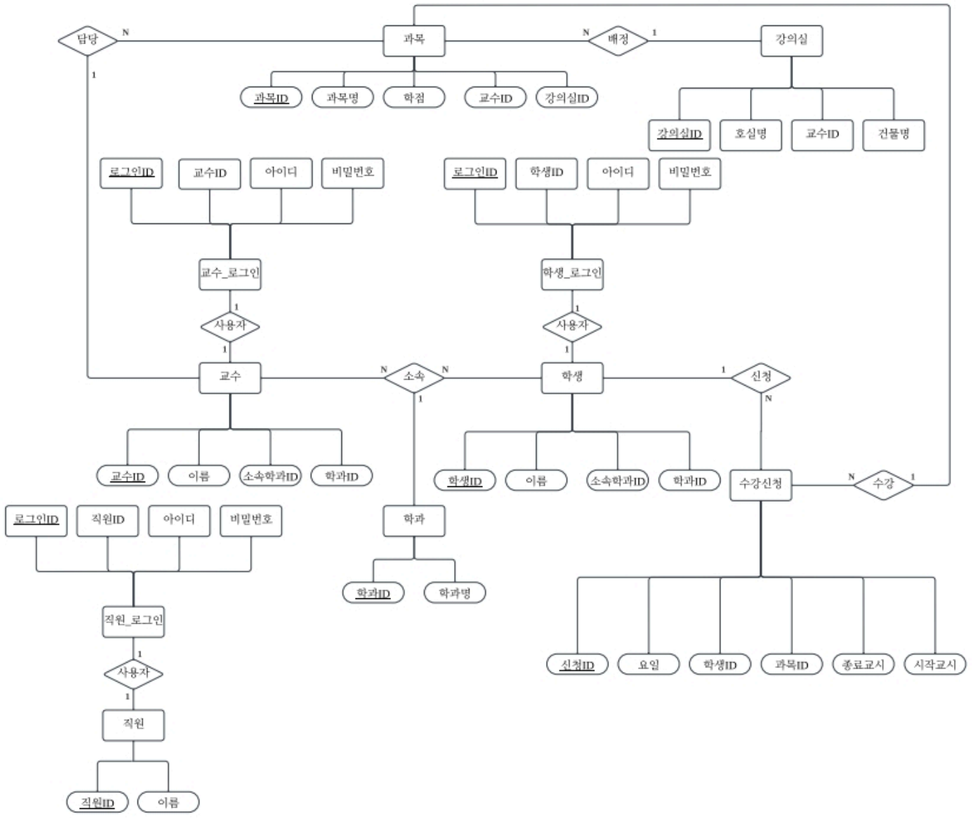
구현기능(IE)


과목시간
요일 Domain Type
시작교시 Domain Type
종료교시 Domain Type
과목ID Domain int
수강신청
신청ID Domain int
학생ID Domain int
과목ID Domain int
과목
과목ID Domain int
과목명 Domain varchar
교수ID Domain int
강의실ID Domain int
강의실
강의실ID Domain int
호실명 Domain varchar
호실 Domain varchar
건물명 Domain varchar
학생
학생ID Domain int
이름 Domain varchar
학과ID Domain int
교수
교수ID Domain int
교수명 Domain varchar
학과ID Domain int
학과
학과ID Domain int
학과명 Domain varchar

구현기능(IE)


수강신청_2번
신청ID INT
학생ID INT
과목ID INT
요일 VARCHAR(10)
시작교시 INT
종료교시 INT
과목_2번
과목ID INT
과목명 VARCHAR(50)
학점 INT
교수ID INT
강의실ID INT
강의실_2번
강의실ID INT
호실명 VARCHAR(50)
호실 VARCHAR(50)
건물명 VARCHAR(50)
학생_2번
학생ID INT
이름 VARCHAR(50)
소속학과ID INT
학과ID INT
교수_2번
교수ID INT
교수명 VARCHAR(50)
소속학과ID INT
학과ID INT
교수_로그인
로그인ID INT
교수ID INT
아이디 VARCHAR(50)
비밀번호 VARCHAR(255)
학생_로그인
로그인ID INT
학생ID INT
아이디 VARCHAR(50)
비밀번호 VARCHAR(255)
학과_2번
학과ID INT
학과명 VARCHAR(50)
직원_2번
직원ID INT
이름 VARCHAR(50)
직원_로그인
로그인ID INT
직원ID INT
아이디 VARCHAR(50)
비밀번호 VARCHAR(255)

구현기능(피터첸)


담당
과목
배정
강의실
과목ID
과목명
학점
교수ID
강의실ID
강의실ID
호실명
교수ID
건물명
로그인ID
교수ID
아이디
비밀번호
로그인ID
학생ID
아이디
비밀번호
교수_로그인
학생_로그인
사용자
사용자
교수
소속
학생
신청
교수ID
이름
소속학과ID
학과ID
학생ID
이름
소속학과ID
학과ID
수강신청
수강
로그인ID
직원ID
아이디
비밀번호
학과
학과ID
학과명
직원_로그인
사용자
직원
직원ID
이름
신청ID
요일
학생ID
과목ID
종료교시
시작교시

프로그램 구조(1)

```
if __name__ == "__main__":
    import sys

    app = QApplication(sys.argv)  # PyQt5 애플리케이션 객체 생성
    main_window = MainWindow()    # 메인 윈도우 객체 생성
    main_window.show()            # 메인 윈도우 표시
    sys.exit(app.exec_())         # 애플리케이션 실행
```

1. 프로그램 시작점

- QApplication 객체를 사용해 PyQt5 GUI 애플리케이션을 실행
- MainWindow 클래스가 프로그램의 시작 화면

프로그램 구조(2)

```
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("수강 시스템 프로그램")
        self.setGeometry(100, 100, 600, 400)
        self.initUI()

    def initUI(self):
        self.central_widget = QWidget()
        self.setCentralWidget(self.central_widget)
        layout = QVBoxLayout()

        self.title_label = QLabel("수강 시스템 프로그램", self)
        layout.addWidget(self.title_label)

        self.login_button = QPushButton("로그인", self)
        self.signup_button = QPushButton("회원가입", self)
        self.exit_button = QPushButton("종료", self)

        layout.addWidget(self.login_button)
        layout.addWidget(self.signup_button)
        layout.addWidget(self.exit_button)

        self.login_button.clicked.connect(self.show_login)
        self.signup_button.clicked.connect(self.show_signup)
        self.exit_button.clicked.connect(self.close_application)

        self.central_widget.setLayout(layout)

    def show_login(self):
        login_window = LoginWindow()
        login_window.exec_()

    def show_signup(self):
        signup_window = SignupWindow()
        signup_window.exec_()

    def close_application(self):
        cursor.close()  # 데이터베이스 커서 종료
        db.close()      # 데이터베이스 연결 종료
        QApplication.quit()
```

2. 메인창

- 프로그램의 시작 화면으로, 로그인, 회원가입, 종료 버튼을 제공

프로그램 구조(3)

```
class LoginWindow(QDialog):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("로그인")
        self.setGeometry(150, 150, 400, 300)
        self.layout = QVBoxLayout()

        self.user_type_label = QLabel("사용자 유형을 선택하세요:", self)
        self.layout.addWidget(self.user_type_label)

        self.user_type_combo = QComboBox(self)
        self.user_type_combo.addItems(["학생", "교수", "교직원"])
        self.layout.addWidget(self.user_type_combo)

        self.id_label = QLabel("아이디:", self)
        self.layout.addWidget(self.id_label)
        self.id_input = QLineEdit(self)
        self.layout.addWidget(self.id_input)

        self.password_label = QLabel("비밀번호:", self)
        self.layout.addWidget(self.password_label)
        self.password_input = QLineEdit(self)
        self.password_input.setEchoMode(QLineEdit.Password)
        self.layout.addWidget(self.password_input)

        self.login_button = QPushButton("로그인", self)
        self.layout.addWidget(self.login_button)
        self.login_button.clicked.connect(self.handle_login)

        self.setLayout(self.layout)
```

3. 로그인 창

- 로그인 창으로 사용자 유형(학생, 교수, 교직원)에 따라 인증을 처리

프로그램 구조(4)

```
class SignupWindow(QDialog):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("회원가입")
        self.setGeometry(150, 150, 400, 300)
        self.layout = QVBoxLayout()

        self.user_type_label = QLabel("회원가입 유형을 선택하세요:", self)
        self.layout.addWidget(self.user_type_label)

        self.user_type_combo = QComboBox(self)
        self.user_type_combo.addItems(["학생", "교수", "교직원"])
        self.layout.addWidget(self.user_type_combo)

        self.signup_button = QPushButton("회원가입 진행", self)
        self.layout.addWidget(self.signup_button)
        self.signup_button.clicked.connect(self.handle_signup)

        self.setLayout(self.layout)

    def handle_signup(self):
        user_type = self.user_type_combo.currentText()

        if user_type == "학생":
            self.student_signup()
        elif user_type == "교수":
            self.professor_signup()
        elif user_type == "교직원":
            self.staff_signup()
        else:
            QMessageBox.warning(self, "오류", "올바른 사용자 유형을 선택하세요.")
```

4. 회원가입 창

- 사용자 유형(학생, 교수, 교직원)에 따라 별도의 회원가입 프로세스를 제공

프로그램 구조(5)

```
class 학생메뉴(QDialog):
    def __init__(self, 학생ID, 이름):
        super().__init__()
        self.학생ID = 학생ID
        self.이름 = 이름
        self.setWindowTitle(f"{이름}님의 학생 메뉴")
        self.setGeometry(200, 200, 600, 400)
        self.layout = QVBoxLayout()

        self.title_label = QLabel(f"환영합니다, {이름}님!", self)
        self.layout.addWidget(self.title_label)

        self.view_courses_button = QPushButton("수강 과목 조회", self)
        self.enroll_button = QPushButton("수강 신청", self)
        self.cancel_button = QPushButton("수강 취소", self)
        self.logout_button = QPushButton("로그아웃", self)

        self.layout.addWidget(self.view_courses_button)
        self.layout.addWidget(self.enroll_button)
        self.layout.addWidget(self.cancel_button)
        self.layout.addWidget(self.logout_button)

        # 버튼 클릭 이벤트 연결
        self.view_courses_button.clicked.connect(self.view_courses)
        self.enroll_button.clicked.connect(self.enroll_course)  # 연결 문제 수정
        self.cancel_button.clicked.connect(self.cancel_course)
        self.logout_button.clicked.connect(self.logout)

        self.setLayout(self.layout)
|
```

5. 학생 메뉴

- 학생이 로그인한 후 접근할 수 있는 메뉴

프로그램 구조(6)

```
class 교수메뉴(QDialog):
    def __init__(self, 교수ID, 교수명):
        super().__init__()
        self.교수ID = 교수ID
        self.교수명 = 교수명
        self.setWindowTitle(f"{교수명} 교수님 - 교수 메뉴")
        self.setGeometry(200, 200, 600, 400)
        self.layout = QVBoxLayout()

        self.title_label = QLabel(f"환영합니다, {교수명} 교수님!", self)
        self.layout.addWidget(self.title_label)

        self.view_courses_button = QPushButton("담당 과목 조회", self)
        self.view_students_button = QPushButton("수강 학생 조회", self)
        self.view_professors_button = QPushButton("교수 목록 조회", self)
        self.view_rooms_button = QPushButton("강의실 목록 조회", self)
        self.view_subjects_button = QPushButton("교과목 목록 조회", self)
        self.view_student_subjects_button = QPushButton("학생별 교과목 조회", self)
        self.logout_button = QPushButton("로그아웃", self)

        self.layout.addWidget(self.view_courses_button)
        self.layout.addWidget(self.view_students_button)
        self.layout.addWidget(self.view_professors_button)
        self.layout.addWidget(self.view_rooms_button)
        self.layout.addWidget(self.view_subjects_button)
        self.layout.addWidget(self.view_student_subjects_button)
        self.layout.addWidget(self.logout_button)

        self.setLayout(self.layout)
```

6. 교수 메뉴

- 교수만 접근 가능한 메뉴

프로그램 구조(7)

```
class 직원메뉴(QDialog):
    def __init__(self, 이름):
        super().__init__()
        self.이름 = 이름
        self.setWindowTitle(f"{이름}님 - 교무처 직원 메뉴")
        self.setGeometry(200, 200, 800, 600)
        self.layout = QVBoxLayout()

        self.title_label = QLabel(f"환영합니다, {이름}님!", self)
        self.layout.addWidget(self.title_label)

        self.view_students_button = QPushButton("학생 목록 조회", self)
        self.view_rooms_button = QPushButton("강의실 목록 조회", self)
        self.view_subjects_button = QPushButton("교과목 목록 조회", self)
        self.manage_courses_button = QPushButton("교과목 관리", self)
        self.manage_professors_button = QPushButton("교수 관리", self)
        self.manage_rooms_button = QPushButton("강의실 관리", self)
        self.logout_button = QPushButton("로그아웃", self)

        self.layout.addWidget(self.view_students_button)
        self.layout.addWidget(self.view_rooms_button)
        self.layout.addWidget(self.view_subjects_button)
        self.layout.addWidget(self.manage_courses_button)
        self.layout.addWidget(self.manage_professors_button)
        self.layout.addWidget(self.manage_rooms_button)
        self.layout.addWidget(self.logout_button)

        self.setLayout(self.layout)
```

7. 교직원 메뉴

- 교직원이 사용할 수 있는 메뉴

테이블 구조

```sql
-- 1. 학과 테이블 생성
CREATE TABLE IF NOT EXISTS 학과_2번 (
    학과ID INT AUTO_INCREMENT PRIMARY KEY,
    학과명 VARCHAR(50) NOT NULL
);

-- 2. 직원 테이블 생성
CREATE TABLE IF NOT EXISTS 직원_2번 (
    직원ID INT AUTO_INCREMENT PRIMARY KEY,
    이름 VARCHAR(50) NOT NULL
);

-- 3. 학생 테이블 생성 (학과 참조)
CREATE TABLE IF NOT EXISTS 학생_2번 (
    학생ID INT AUTO_INCREMENT PRIMARY KEY,
    이름 VARCHAR(50)  NOT NULL DEFAULT '이름없음',
    소속학과ID INT NOT NULL,
    FOREIGN KEY (소속학과ID) REFERENCES 학과_2번(학과ID) ON DELETE CASCADE
);

-- 4. 교수 테이블 생성 (학과 참조)
CREATE TABLE IF NOT EXISTS 교수_2번 (
    교수ID INT AUTO_INCREMENT PRIMARY KEY,
    교수명 VARCHAR(50) NOT NULL,
    소속학과ID INT NOT NULL,
    FOREIGN KEY (소속학과ID) REFERENCES 학과_2번(학과ID) ON DELETE CASCADE
);

-- 5. 강의실 테이블 생성
CREATE TABLE IF NOT EXISTS 강의실_2번 (
    강의실ID INT AUTO_INCREMENT PRIMARY KEY,
    호실명 VARCHAR(50) NOT NULL,
    호실 VARCHAR(50) NOT NULL,
    건물명 VARCHAR(50) NOT NULL
);
```

테이블 구조

```sql
-- 6. 과목 테이블 생성 (교수 및 강의실 참조) - 학점 컬럼 추가
CREATE TABLE IF NOT EXISTS 과목_2번 (
    과목ID INT AUTO_INCREMENT PRIMARY KEY,
    과목명 VARCHAR(50) NOT NULL,
    학점 INT NOT NULL DEFAULT 3,
    교수ID INT NOT NULL,
    강의실ID INT,
    FOREIGN KEY (교수ID) REFERENCES 교수_2번(교수ID) ON DELETE CASCADE,
    FOREIGN KEY (강의실ID) REFERENCES 강의실_2번(강의실ID) ON DELETE SET NULL
);
```

```sql
-- 7. 학생_로그인 테이블 생성 (학생 및 학과 참조)
CREATE TABLE IF NOT EXISTS 학생_로그인 (
    로그인ID INT AUTO_INCREMENT PRIMARY KEY,
    학생ID INT NOT NULL,
    아이디 VARCHAR(50) NOT NULL UNIQUE,
    비밀번호 VARCHAR(255) NOT NULL,
    이름 VARCHAR(50) NOT NULL,
    소속학과ID INT NOT NULL,
    FOREIGN KEY (학생ID) REFERENCES 학생_2번(학생ID) ON DELETE CASCADE,
    FOREIGN KEY (소속학과ID) REFERENCES 학과_2번(학과ID) ON DELETE CASCADE
);
```

테이블 구조

```sql
-- 8. 교수_로그인 테이블 생성 (교수 참조)
CREATE TABLE IF NOT EXISTS 교수_로그인 (
    로그인ID INT AUTO_INCREMENT PRIMARY KEY,
    교수ID INT NOT NULL,
    아이디 VARCHAR(50) NOT NULL UNIQUE,
    비밀번호 VARCHAR(255) NOT NULL,
    FOREIGN KEY (교수ID) REFERENCES 교수_2번(교수ID) ON DELETE CASCADE
-- 9. 직원_로그인 테이블 생성 (직원 참조)
CREATE TABLE IF NOT EXISTS 직원_로그인 (
    로그인ID INT AUTO_INCREMENT PRIMARY KEY,
    직원ID INT NOT NULL,
    아이디 VARCHAR(50) NOT NULL UNIQUE,
    비밀번호 VARCHAR(255) NOT NULL,
    FOREIGN KEY (직원ID) REFERENCES 직원_2번(직원ID) ON DELETE CASCADE
);
-- 10. 과목 시간 테이블 생성 (과목 참조)
CREATE TABLE IF NOT EXISTS 과목_시간_2번 (
    시간ID INT AUTO_INCREMENT PRIMARY KEY,
    과목ID INT NOT NULL,
    요일 VARCHAR(10) NOT NULL,
    시작교시 INT NOT NULL,
    종료교시 INT NOT NULL,
    FOREIGN KEY (과목ID) REFERENCES 과목_2번(과목ID) ON DELETE CASCADE
);
```

```sql
-- 11. 수강신청 테이블 생성 (학생 및 과목 참조)
CREATE TABLE IF NOT EXISTS 수강신청_2번 (
    신청ID INT AUTO_INCREMENT PRIMARY KEY,
    학생ID INT NOT NULL,
    과목ID INT NOT NULL,
    요일 VARCHAR(10) NOT NULL,        -- 수업 요일 추가
    시작교시 INT NOT NULL,            -- 수업 시작 교시 추가
    종료교시 INT NOT NULL,            -- 수업 종료 교시 추가
    FOREIGN KEY (학생ID) REFERENCES 학생_2번(학생ID) ON DELETE CASCADE,
    FOREIGN KEY (과목ID) REFERENCES 과목_2번(과목ID) ON DELETE CASCADE,
    UNIQUE (학생ID, 과목ID)
);
```

시간중복체크 함수

```sql
-- 시간중복체크 함수: 특정 학생의 시간 중복 여부를 확인
CREATE FUNCTION 시간중복체크(
    입력학생ID INT,            -- 수강 신청하려는 학생 ID
    입력요일 VARCHAR(10),      -- 신청 요일
    입력시작교시 INT,          -- 신청 시작 교시
    입력종료교시 INT           -- 신청 종료 교시
)
RETURNS BOOLEAN
BEGIN
    DECLARE 중복횟수 INT;  -- 중복된 시간의 개수를 저장하는 변수

    -- 수강신청 테이블에서 시간 중복 여부를 확인
    SELECT COUNT(*) INTO 중복횟수
    FROM 수강신청_2번
    WHERE 학생ID = 입력학생ID                  -- 같은 학생
      AND 요일 = 입력요일                      -- 같은 요일
      AND (시작교시 <= 입력종료교시            -- 기존 수업이 끝나는 시간 >= 새로운 수업의 시작 시간
           AND 종료교시 >= 입력시작교시);  -- 기존 수업이 시작하는 시간 <= 새로운 수업의 끝나는 시간

    -- 중복된 시간이 존재하면 TRUE 반환, 그렇지 않으면 FALSE 반환
    RETURN 중복횟수 > 0;
END;
```

상세 코드(수강신청)

```
-- 수강신청 프로시저: 학생이 수강신청을 처리하는 절차를 정의
CREATE PROCEDURE 수강신청(
    IN 입력학생ID INT,          -- 수강신청을 하는 학생 ID
    IN 입력과목ID INT,          -- 신청하려는 과목 ID
    IN 입력요일 VARCHAR(10),    -- 신청 요일
    IN 입력시작교시 INT,        -- 신청 시작 교시
    IN 입력종료교시 INT         -- 신청 종료 교시
)BEGIN
    DECLARE 과목학점 INT;       -- 신청 과목의 총 학점
    DECLARE 남은학점 INT;       -- 과목의 남은 신청 가능 학점
    DECLARE 신청학점 INT;       -- 신청 학점을 계산하는 변수
    DECLARE 중복여부 BOOLEAN;   -- 시간 중복 여부를 확인하는 변수
    -- 1. 과목 학점 조회: 신청하려는 과목의 학점을 가져옴
    SELECT 학점 INTO 과목학점
    FROM 과목_2번
    WHERE 과목ID = 입력과목ID;
```

```
-- 1. 과목 학점 조회: 신청하려는 과목의 학점을 가져옴
SELECT 학점 INTO 과목학점
FROM 과목_2번
WHERE 과목ID = 입력과목ID;
-- 과목이 존재하지 않으면 오류를 발생시킴
IF 과목학점 IS NULL THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = '해당 과목ID가 존재하지 않습니다.';
END IF;
-- 2. 신청 학점 계산: 종료 교시에서 시작 교시를 뺀 값 + 1
SET 신청학점 = 입력종료교시 - 입력시작교시 + 1;

-- 시작 교시와 종료 교시가 올바르지 않으면 오류 발생
IF 신청학점 <= 0 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = '잘못된 교시 입력입니다.';
END IF;
```

상세 코드(수강신청)

```
    -- 3. 남은 학점 계산: 신청 가능한 학점을 확인
    SELECT 과목학점 - IFNULL(SUM(종료교시 - 시작교시 + 1), 0)
    INTO 남은학점
    FROM 수강신청_2번
    WHERE 학생ID = 입력학생ID AND 과목ID = 입력과목ID;
    -- 남은 학점보다 신청 학점이 크면 오류 발생
    IF 신청학점 > 남은학점 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '신청 학점 초과: 남은 학점을 초과했습니다.';
    END IF;
    -- 4. 시간 중복 확인
    SET 중복여부 = 시간중복체크(입력학생ID, 입력요일, 입력시작교시, 입력종료교시);
    -- 시간 중복이 발생하면 오류 발생
    IF 중복여부 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '시간이 중복됩니다.';
    ELSE
        -- 중복이 없으면 수강 신청 정보를 삽입
        INSERT INTO 수강신청_2번 (학생ID, 과목ID, 요일, 시작교시, 종료교시)
        VALUES (입력학생ID, 입력과목ID, 입력요일, 입력시작교시, 입력종료교시);
    END IF;
END //
DELIMITER ;
```

상세 코드(뷰)

```sql
-- 학과별 교수 목록 뷰: 특정 학과에 소속된 교수 정보를 출력
CREATE OR REPLACE VIEW 학과별_교수목록 AS
SELECT
    g.교수ID,        -- 교수의 고유 식별자
    g.교수명,        -- 교수 이름
    d.학과명         -- 소속 학과 이름
FROM 교수_2번 g
JOIN 학과_2번 d ON g.소속학과ID = d.학과ID; -- 교수와 학과를 연결
```

```sql
-- 강의실 목록 뷰: 강의실의 상세 정보를 출력
CREATE OR REPLACE VIEW 강의실목록 AS
SELECT
    강의실ID,        -- 강의실의 고유 식별자
    호실명,          -- 강의실 이름
    호실,            -- 강의실 번호
    건물명           -- 강의실이 위치한 건물 이름
FROM 강의실_2번;
```

상세 코드(뷰)

```sql
-- 학과별 교과목 목록 뷰: 특정 학과에서 개설된 교과목 정보 출력
CREATE OR REPLACE VIEW 학과별_교과목목록 AS
SELECT
    c.과목ID,          -- 과목의 고유 식별자
    c.과목명,          -- 과목 이름
    c.학점,            -- 과목 학점
    p.교수명,          -- 담당 교수 이름
    d.학과명           -- 과목이 속한 학과 이름
FROM 과목_2번 c
JOIN 교수_2번 p ON c.교수ID = p.교수ID          -- 과목과 담당 교수 연결
JOIN 학과_2번 d ON p.소속학과ID = d.학과ID;     -- 교수와 학과 연결
```

```sql
-- 학생별 신청 과목 목록 뷰: 특정 학생이 신청한 과목 정보 출력
CREATE OR REPLACE VIEW 학생별_신청과목목록 AS
SELECT
    s.학생ID,          -- 학생의 고유 식별자
    st.이름 AS 학생명,  -- 학생 이름
    c.과목ID,          -- 과목의 고유 식별자
    c.과목명,          -- 과목 이름
    c.학점,            -- 과목 학점
    p.교수명,          -- 담당 교수 이름
    r.건물명,          -- 강의실이 위치한 건물 이름
    r.호실명,          -- 강의실 이름
    s.요일,            -- 수업 요일
    s.시작교시,        -- 수업 시작 교시
    s.종료교시         -- 수업 종료 교시
FROM 수강신청_2번 s
JOIN 학생_2번 st ON s.학생ID = st.학생ID        -- 학생과 수강신청 연결
JOIN 과목_2번 c ON s.과목ID = c.과목ID          -- 과목과 수강신청 연결
JOIN 교수_2번 p ON c.교수ID = p.교수ID          -- 교수와 과목 연결
JOIN 강의실_2번 r ON c.강의실ID = r.강의실ID;   -- 강의실과 과목 연결
```

상세 코드(뷰)

```
-- 과목별 수강 학생 목록 뷰: 특정 과목에 수강 신청한 학생 정보 출력
CREATE OR REPLACE VIEW 과목별_수강학생목록 AS
SELECT
    c.과목ID,          -- 과목의 고유 식별자
    c.과목명,          -- 과목 이름
    s.학생ID,          -- 학생의 고유 식별자
    st.이름 AS 학생명,  -- 학생 이름
    s.요일,            -- 수업 요일
    s.시작교시,        -- 수업 시작 교시
    s.종료교시         -- 수업 종료 교시
FROM 수강신청_2번 s
JOIN 과목_2번 c ON s.과목ID = c.과목ID          -- 과목과 수강신청 연결
JOIN 학생_2번 st ON s.학생ID = st.학생ID;       -- 학생과 수강신청 연결
```

기능 시연

P-TECH 융합전자통신과		1 (09:00~10:00)	2 (10:00~11:00)	3 (11:00~12:00)	4 (12:00~13:00)
1학년	1반	과목명 : 커뮤니케이션 능력 교수명 : 김영선 강의실 : 창의융합통신강의실2(D0420)		과목명 : 스마트기기개론 교수명 : 김재목 강의실 : IoT 창의적 종합 설계실(D0301)	점심 시간
	2반	과목명 : 커뮤니케이션 능력 교수명 : 김재연 강의실 : 창의융합통신강의실1(D0416)		과목명 : 정보통신기기 아날로그회로설계 교수명 : 신익조 강의실 : 통신응용실험실(D0401)	
2학년	1반	과목명 : 컴퓨터활용2 교수명 : 강성인 강의실 : 기초회로실험실(D0317)		과목명 : 외부평가대비 하드웨어개발총론 교수명 : 김영포 강의실 : 안테나 실험실(D0417)	
	2반	과목명 : 컴퓨터활용2 교수명 : 김재목 강의실 : IoT 창의적 종합 설계실(D0301)		과목명 : 스마트기기네트워크 교수명 : 강성인 강의실 : 기초회로실험실(D0317)	

프로젝트 결론

PROJECT CONCLUSION

수강신청 앱

효율적인 인적 관리

•프로젝트 구조에 맞춘 자원 배분(학생, 교수, 교직원 별 메뉴 제공).
• 사용자의 역할과 기능에 따른 자원 분배(플로우차트 활용).

데이터 분석 / 파악

• 중복 수강 및 학점 초과 문제의 실시간 확인.
• 강의실 및 교수 정보 연결로 의사 결정 지원 .
• 데이터 베이스 뷰(View)를 통한 효율적인 정보 조회.

최적화된 프로세스

• 회원가입, 로그인, 수강 신청 프로세스를 단계별로 최적화.
• GUI 애플리케이션을 사용한 인터페이스 간소화.
• 시간 중복 체크 및 학점 관리 함수로 데이터 처리 속도 개선.

참고 문서

https://zorba91.tistory.com/29 [mySQL] 프로시저 만들기(DECLARE, SET, IN, IF, ELSEIF 등)
https://spiderwebcoding.tistory.com/5 [MySQL] 데이터 타입(data_type) 정리
https://github.com/joy3968/PyQt_project PyQt를 활용한 간단 수강신청 프로그램 만들기

궁금한 사항이 있으면
편하게 질문해 주시기 바랍니다.

감사합니다.