C-AIR

서채원

1 프로젝트 개요

a. 프로젝트명 :

- CNU Airline Reservation "C-AIR"

b. 목적 :

- 웹 기반 인터페이스를 통해 사용자가 항공기 검색, 항공권 예약/취소, 예약/취소 내역 조회를 쉽고 직관적으로 수행할 수 있는 시스템 개발

c. 주요 기능 (자세한 요구사항은 "TP_문제정의서.PDF" 참고) :

- 회원 정보 관리 및 로그인
- 항공기 정보 검색
- 항공권 예약
- 항공권 취소
- 예약/취소 내역 조회
- 관리자 전용 통계 정보 검색

d, 제외사항 :

- 회원/항공기 등록, 변경, 삭제
- 결제/환불 및 좌석 번호 배정

2 개발 환경

a. LG Laptop (Windows11)

- DB : Oracle XE 21c
- IDE : SQL Developer

b. MacBook Air 15" M2 (MacOS Sequoia)

- Web Server :
  - -- Python 3.11.11
    - --- Python Flask 3.1.1
    - --- python-oracledb 3.1.1 Thin Mode (DB Connection)
    - --- SQLAlchemy 2.0.41 (ORM)

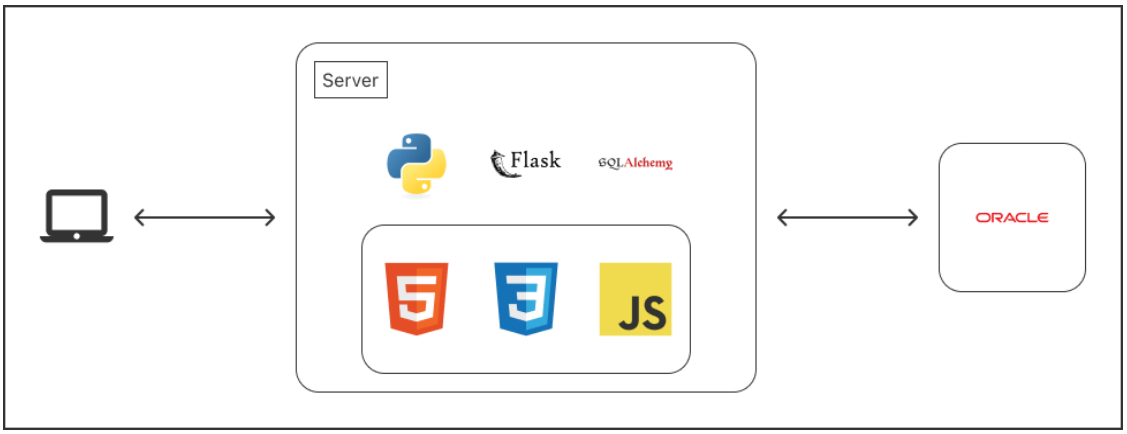
- Front-End :
  - -- Figma (UX/UI Design)
  - -- HTML, CSS, JavaScript
  - -- Jinja2 Template 3.1.6

- API :
  - -- Naver STMP Mail

- IDE :
  - -- VSCode
  - -- Cursor

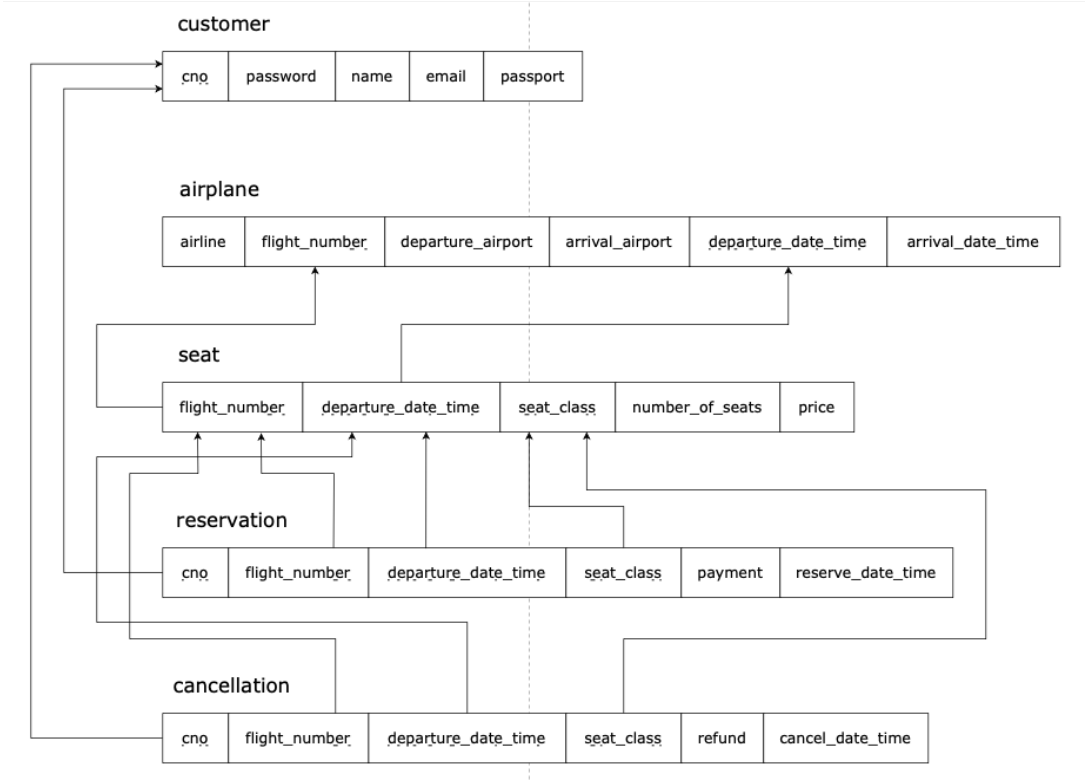
3 시스템 설계

a. 전체 아키텍처



b. 데이터베이스 설계

- Relational Schema :



c. UI 설계

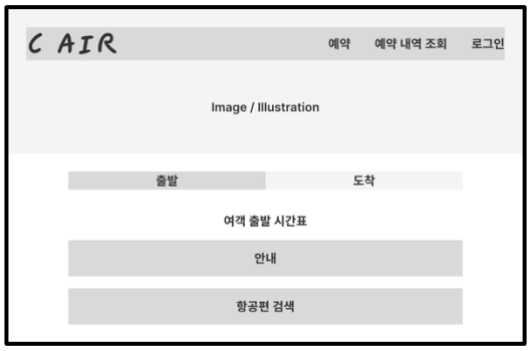
- 인천공항 & 제주공항 공식 홈페이지 참고

- Wireframes :

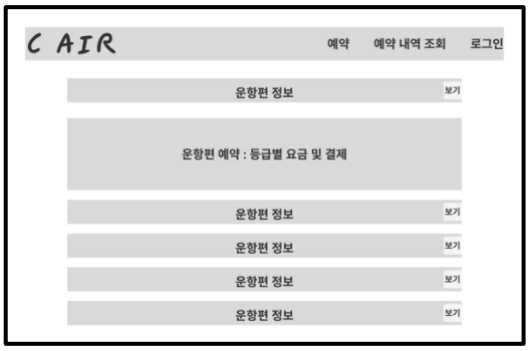
-- 홈 화면 ::

-- 로그인 화면 ::

-- 항공기 검색 화면 ::

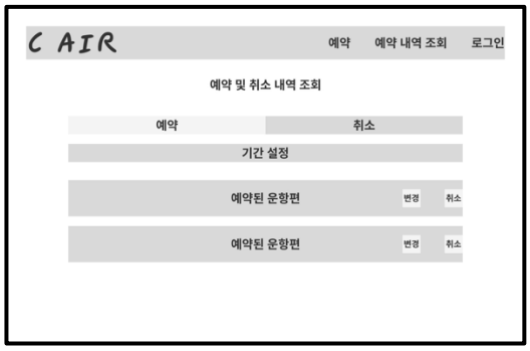


-- 항공기 예약 화면 ::

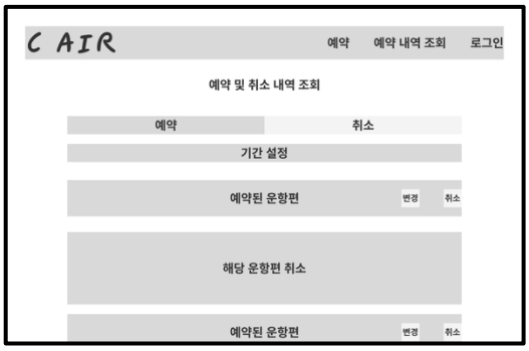


--- 검색 결과 >> 예약

-- 예약/취소 내역 조회 화면 ::



-- 예약 취소 화면 ::



--- 예약/취소 내역 조회 >> 예약 취소

-- 관리자 전용 통계 화면 ::

--- 관리자 로그인 >> [통계 정보 확인] 이동 >> 통계 정보 조회 가능

4 세부 구현 내용

a. 전체 프로젝트 구조 및 역할

tree-maker > project-tree.md

```markdown
. C-AIR
└── __pycache__/
├── .env
├── app.py
├── config.py
└── models/
|   └── __pycache__/
|   ├── __init__.py
|   ├── airplane.py
|   ├── cancellation.py
|   ├── customer.py
|   ├── reservation.py
|   ├── seat.py
|   ├── statistics.py
└── routes/
|   └── __pycache__/
|   ├── __init__.py
|   ├── api.py
|   ├── main.py
└── static/
|   └── assets/
|   └── css/
└── templates/
|   ├── cancellation-complete.html
|   ├── cancellation.html
|   ├── history-result.html
|   ├── history-search.html
|   ├── index-login.html
|   ├── index.html
|   ├── login.html
|   ├── payment-complete.html
|   ├── payment.html
|   ├── reservation.html
|   ├── search.html
|   ├── statistics.html
```

- app.py :: Flask 애플리케이션 진입점 ::: DB 연결 및 라우트 등록, 서버 실행

- config.py :: 프로젝트 환경 설정 관리

- .env :: 프로젝트 환경 변수 저장

- models/ :: ORM 모델(클래스) 및 비즈니스 로직 정의 저장 폴더

- static/ :: 정적 파일(아이콘/이미지/로고 및 CSS 파일) 저장 폴더

- templates/ :: HTML 템플릿 파일 저장 폴더

b. 주요 로직

- Oracle DB Connection :

-- config.py ::

```
class Config:

    # Oracle DB 설정 (환경변수에서 값 읽어옴)
    DB_USER = os.getenv("DB_USER")
    DB_PASSWORD = os.getenv("DB_PASSWORD")
    DB_HOST = os.getenv("DB_HOST")
    DB_PORT = os.getenv("DB_PORT")
    DB_SERVICE = os.getenv("DB_SERVICE")

    # Oracle DB 연결 문자열 (SQLAlchemy 용, Thin 모드)
    SQLALCHEMY_DATABASE_URI =
    f"oracle+oracledb://{DB_USER}:{DB_PASSWORD}@{DB_HOST}:{DB_PORT}/?service_name={DB_SERVICE}"
    SQLALCHEMY_TRACK_MODIFICATIONS = False   # SQLAlchemy 이벤트 시스템 비활성화(권장)
```

- ORM Mapping & Logic :

-- models/airplane.py ::

```
class Airplane(db.Model):
    __tablename__ = "AIRPLANE"

    # 공항 코드와 공항명 매핑 (검색 시 한글/영문 모두 지원)
    AIRPORT_MAPPING = {
        "ICN": ["인천", "인천공항", "ICN"],
        "JFK": ["뉴욕", "JFK", "JFK 공항"],
    }

    # 항공편 정보 컬럼 정의
    airline = Column("AIRLINE", String(100), nullable=False)
    flight_number = Column("FLIGHT_NUMBER", String(20), primary_key=True)
    departure_date_time = Column(
    "DEPARTURE_DATE_TIME", DateTime, primary_key=True
    )
    departure_airport = Column(
    "DEPARTURE_AIRPORT", String(100), nullable=False
    )
    arrival_date_time = Column(
    "ARRIVAL_DATE_TIME", DateTime, nullable=False
    )
    arrival_airport = Column("ARRIVAL_AIRPORT", String(100), nullable=False)

  def __repr__(self):
      return f"<Airplane {self.flight_number} {self.departure_date_time}>"
```

+ 검색 편의를 위한 공항 코드 리스트 반환 / 항공편 검색 메소드 정의

-- models/cancellation.py ::

```
class Cancellation(db.Model):
  __tablename__ = "CANCELLATION"

  cno = Column("CNO", String(20), primary_key=True)
  flight_number = Column("FLIGHT_NUMBER", String(20), primary_key=True)
  departure_date_time = Column("DEPARTURE_DATE_TIME", DateTime, primary_key=True)
  seat_class = Column("SEAT_CLASS", String(20), primary_key=True)
  refund = Column("REFUND", Integer, nullable=False)
  cancel_date_time = Column(
  "CANCEL_DATE_TIME", DateTime, default=datetime.now, nullable=False
  )

  # Foreign Key 관계 설정
  customer = db.relationship(
    "Customer", foreign_keys=[cno], primaryjoin="Cancellation.cno == Customer.cno"
  )
  airplane = db.relationship(
    "Airplane",
    foreign_keys=[flight_number, departure_date_time],
    primaryjoin="and_(Cancellation.flight_number == Airplane.flight_number, "
    "Cancellation.departure_date_time == Airplane.departure_date_time)",
  )

  def __repr__(self):
    return (
      f"<Cancellation {self.cno} {self.flight_number} {self.departure_date_time}>"
    )
```

+ 수수료 정책을 적용한 환불 금액 계산 메소드 정의
+ 취소 메소드 정의
+ 검색 편의를 위한 취소 내역 조회 / 취소 통계 조회 메소드 정의

-- models/customer.py ::

```
class Customer(db.Model):
    __tablename__ = "CUSTOMER"

# 고객 정보 컬럼 정의
cno = Column("CNO", String(20), primary_key=True)   # 회원번호
password = Column("PASSWORD", String(60), nullable=False)   # 비밀번호
name = Column("NAME", String(60), nullable=False)   # 이름
email = Column("EMAIL", String(100), unique=True, nullable=False)   # 이메일
passport = Column("PASSPORT", String(20), unique=True, nullable=False)   # 여권번호

def __repr__(self):
    return f"<Customer {self.cno}>"
```

-- models/reservation.py ::

```
class Reservation(db.Model):
    __tablename__ = "RESERVATION"

    # 예약 정보 컬럼 정의
    cno = Column("CNO", String(20), primary_key=True)
    flight_number = Column("FLIGHT_NUMBER", String(20), primary_key=True)
    departure_date_time = Column(
    "DEPARTURE_DATE_TIME", DateTime, primary_key=True
    )
    seat_class = Column("SEAT_CLASS", String(20), primary_key=True)
    payment = Column("PAYMENT", Integer, nullable=False)
    reserve_date_time = Column(
    "RESERVE_DATE_TIME", DateTime, default=datetime.now, nullable=False
    )

    # Foreign Key 관계 설정 (고객, 좌석, 항공편)
    customer = db.relationship(
        "Customer", foreign_keys=[cno], primaryjoin="Reservation.cno == Customer.cno"
    )
    seat = db.relationship(
        "Seat",
        foreign_keys=[flight_number, departure_date_time, seat_class],
        primaryjoin="and_(Reservation.flight_number == Seat.flight_number, "
        "Reservation.departure_date_time == Seat.departure_date_time, "
        "Reservation.seat_class == Seat.seat_class)",
    )
    airplane = db.relationship(
        "Airplane",
        foreign_keys=[flight_number, departure_date_time],
        primaryjoin="and_(Reservation.flight_number == Airplane.flight_number, "
        "Reservation.departure_date_time == Airplane.departure_date_time)",
    )

    def __repr__(self):
        return (
            f"<Reservation {self.cno} {self.flight_number} {self.departure_date_time}>"
        )
```

+ 새 예약 생성 메소드 정의
+ 검색 편의를 위한 예약 조회 메소드 정의

-- models/seat.py ::

```
class Seat(db.Model):
    __tablename__ = "SEAT"

    # 좌석 등급 매핑 (한글/영문 모두 지원)
    SEAT_CLASS_MAPPING = {
        "Business": ["비즈니스", "비즈니스석", "Business", "business"],
        "Economy": ["이코노미", "이코노미석", "Economy", "economy"],
    }

    # 좌석 정보 컬럼 정의
    flight_number = Column("FLIGHT_NUMBER", String(20), primary_key=True)
    departure_date_time = Column(
        "DEPARTURE_DATE_TIME", DateTime, primary_key=True
    )
    seat_class = Column("SEAT_CLASS", String(20), primary_key=True)
    number_of_seats = Column("NUMBER_OF_SEATS", Integer, nullable=False)
    price = Column("PRICE", Integer, nullable=False)

    # Foreign Key 관계 설정 (Airplane 과 연결)
    airplane = db.relationship(
        "Airplane",
        foreign_keys=[flight_number, departure_date_time],
        primaryjoin="and_(Seat.flight_number == Airplane.flight_number, "
        "Seat.departure_date_time == Airplane.departure_date_time)",
    )

    def __repr__(self):
        return f"<Seat {self.flight_number} {self.seat_class}>"
```

+ 좌석 등급 입력값을 DB 좌석 등급으로 변환하는 메소드 정의
+ 특정 항공편/좌석등급의 가용 좌석 수 조회 메소드 정의
+ 좌석 수 업데이트 메소드 정의

-- models/statistics.py ::

--- GROUP Function 01 :: 운항편별 총 예약 금액 및 평균 예약 금액 조회

```
def get_group01_stats():
    q = (
        db.session.query(
        Reservation.flight_number.label("flight_number"),
        Reservation.departure_date_time.label("departure_date_time"),
        func.sum(Reservation.payment).label("total_amount"),
        func.avg(Reservation.payment).label("avg_amount"),
        func.count().label("reservation_count"),
        )
        .group_by(Reservation.flight_number, Reservation.departure_date_time)
        .order_by(func.sum(Reservation.payment).desc())
    )
    df = pd.read_sql(q.statement, db.engine)
    df.columns = [
        "flight_number",
        "departure_date_time",
        "total_amount",
        "avg_amount",
        "reservation_count",
    ]
    # 컬럼 순서 변경
    df = df[
            [
                "flight_number",
                "departure_date_time",
                "reservation_count",
                "total_amount",
                "avg_amount",
            ]
    ]
    return df
```

--- GROUP Function 02 :: 고객별 총 취소 환불액 및 최대 환불액 조회

```
def get_group02_stats():
    q = (
        db.session.query(
            Cancellation.cno.label("cno"),
            Customer.name.label("name"),
            func.sum(Cancellation.refund).label("total_refund"),
            func.max(Cancellation.refund).label("max_refund"),
            func.count().label("cancel_count"),
        )
        .join(Customer, Customer.cno == Cancellation.cno)
        .group_by(Cancellation.cno, Customer.name)
        .having(func.sum(Cancellation.refund) > 0)
        .order_by(func.sum(Cancellation.refund).desc())
    )
    df = pd.read_sql(q.statement, db.engine)
    df.columns = ["cno", "name", "total_refund", "max_refund", "cancel_count"]
    # name 컬럼은 내부적으로만 사용, 반환 시에는 제거
    df = df[["cno", "name", "cancel_count", "total_refund", "max_refund"]]
    return df
```

-- WINDOW Function 01 : 항공편별 누적 예약 금액 및 예약 금액 순위

```
def get_window01_stats():
    q = db.session.query(
        Reservation.flight_number.label("flight_number"),
        Reservation.departure_date_time.label("departure_date_time"),
        Reservation.payment.label("amount"),
        func.sum(Reservation.payment)
        .over(
            partition_by=[Reservation.flight_number, Reservation.departure_date_time],
            order_by=Reservation.reserve_date_time,
        )
        .label("cumulative_amount"),
        func.rank()
        .over(
            partition_by=[Reservation.flight_number, Reservation.departure_date_time],
            order_by=desc(Reservation.payment),
        )
        .label("amount_rank"),
    )
    df = pd.read_sql(q.statement, db.engine)
    df.columns = [
        "flight_number",
        "departure_date_time",
        "amount",
        "cumulative_amount",
        "amount_rank",
    ]
    # 컬럼 순서 변경
    df = df[
        [
            "flight_number",
            "departure_date_time",
            "cumulative_amount",
            "amount",
            "amount_rank",
        ]
    ]
return df
```

--- WINDOW Function 02 ::: 고객별 총 예약 금액 순위

```
def get_window02_stats():
    sql = """
    SELECT
        c.cno,
        c.name,
        SUM(r.payment) AS total_amount,
        RANK() OVER (ORDER BY SUM(r.payment) DESC) AS total_rank
    FROM customer c
    JOIN reservation r ON c.cno = r.cno
    GROUP BY c.cno, c.name
    ORDER BY total_rank
    """
    df = pd.read_sql(text(sql), db.engine)
    df.columns = ["cno", "name", "total_amount", "total_rank"]
    df = df[["cno", "name", "total_amount", "total_rank"]]
    return df
```

c. UX/UI Design

- 전체 :

-- Main Color ::

--- 검정색 (#333333) ::: 텍스트 :::: 높은 가독성

--- 주황색 (#F26D4F) :::: 긍정적 인상 & 높은 접근성/주목성

-- 인천공항 & 제주공항 공식 홈페이지 UX/UI 참고

-- 상단 네비게이션 바 ::: "C-AIR" 로고 & 메뉴

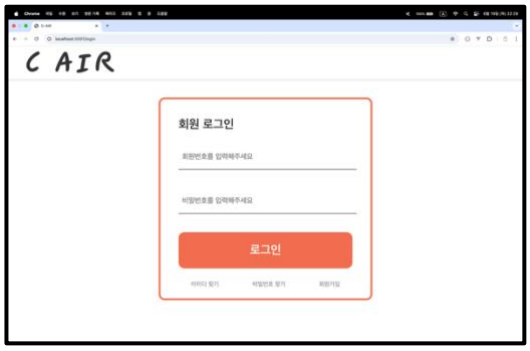
- 홈 화면 :

-- Hero Image :: 메인 주황색 들어간 비행기 일러스트레이션 사용

-- 빠른 검색 :: 화면 이동 없이도 주요 기능 [운항편 검색] 가능

--- 검색 조건 입력 필드 - 아이콘 및 플레이스홀더 사용 :::: High Usability

- 로그인 화면 :



-- 회원번호/비밀번호 입력 필드 - 플레이스홀더 :::: High Usability

-- [로그인] CTA 버튼 :: 큰 사이즈 & 메인 주황색 ::: 시각적 강조

- 검색 화면 :

-- 상단 네비게이션 바 [예약] 버튼 클릭 > 이동

-- [빠른 검색] 기능과 동일하게 동작

-- 홈 화면과의 차이:: Hero Image 변경 및 운항편 검색 안내 문구 추가

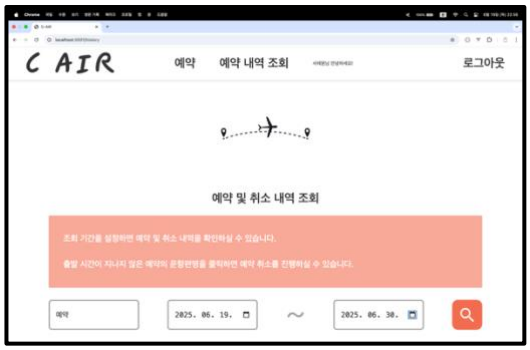
- 검색 결과 화면 :

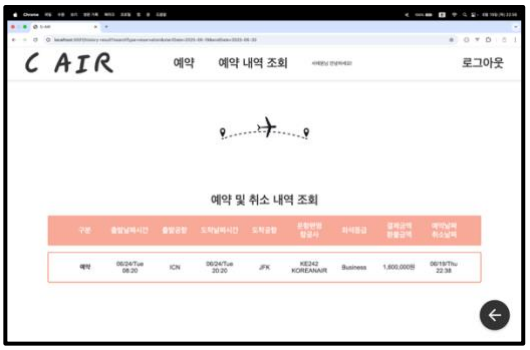
-- 각 운항편 내용을 설명하는 안내바 :: 투명도 낮춘 메인 주황색 사용

-- 각 운항편 컴포넌트 클릭 >> 해당 운항편 예약 화면으로 이동

-- 시간순 / 요금순 오름차순 및 내림차순 정렬 가능

- 예약/취소 내역 조회 화면 :



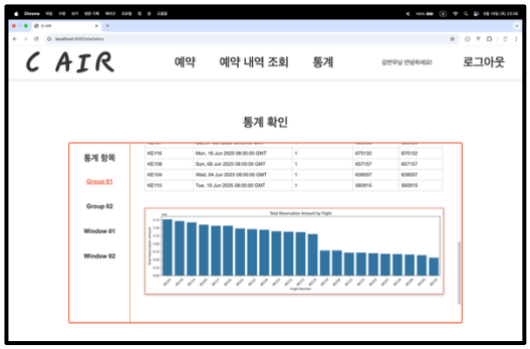


-- 원하는 검색 조건에 따른 예약/취소 내역 조회 가능

--- 검색 조건 :: 전체/예약/취소 타입 & 조회 기간

-- 취소 내역 중 출발날짜시간이 아직 지나지 않은(취소 가능한) 운항편 클릭 >> 해당 운항편 취소 화면으로 이동

- 관리자 전용 통계 화면 :



-- 관리자 로그인 O 상태에서만 조회 가능

-- 사이드바의 [함수] 버튼 클릭

>> 각 함수를 사용한 통계 질의에 대한 결과 조회 가능 (표 & 도표 제공)

d. 사용자 매뉴얼

- 회원/비회원 :
  -- 비회원은 운항편 검색만 가능, 예약/결제/내역조회/취소는 불가능 (로그인 필요)
  -- 회원은 로그인 후 검색/예약/내역조회/취소 등 모든 기능 사용 가능

- 기능별 설명 및 화면 동작 :
  -- 로그인/로그아웃 ::
    --- 로그인 ::: 회원번호, 비밀번호 입력 --> 성공 시 메인 화면 이동
    --- 로그아웃 ::: 우측 상단 메뉴에서 [로그아웃] 클릭

  -- 운항편 예약 ::
    --- 운항편 검색 --> 원하는 운항편 클릭 --> 최종 확인 및 결제 --> 예약 완료
    --- 예약 완료와 동시에 회원 이메일로 예약 내역 자동 발송
    --- 운항편 예약은 반드시 회원 로그인 필요

  -- 예약 내역 조회 ::
    --- 내역 검색 --> 전체/예약/취소 내역 각각 조회

  -- 예약 취소 ::
    --- 내역 검색 --> 예약 내역 조회 결과 중 출발날짜시간이 아직 지나지 않은
        운항편 클릭 --> 최종 확인 및 취소 --> 예약 취소 완료
    --- 취소 위약금 정책에 따라 환불 금액 계산 수행
    --- 취소 완료 시 환불 금액 안내 및 취소 내역 반영 + 항공기 좌석수 복구

  -- 관리자 전용 통계 ::
    --- 관리자 로그인 시에만 [통계] 메뉴 활성화
    --- 항공편별/고객별 예약/취소/환불 통계 조회 가능
    --- 표 & 도표 활용 시각화

5 테스트 및 결과

a. 테스트(데모) 평가 리스트

2025-1학기 DB Term Project Demo Check List (총 300점)

분류	항목	배점
가) 회원 정보 관리 및 로그인	회원번호와 비밀번호를 이용하여 로그인 가능하다. ⭕	10
	로그인을 하기 전과 로그인을 한 후의 상태를 구분할 수 있다. ⭕	10
나) 항공기 정보 및 검색	출발공항, 도착공항, 출발날짜, 좌석 등급을 입력하면, 정확한 조건의 항공기를 검색하는가 ⭕	15
	조회 결과에 요구된 모든 항목이 포함되어 있는가? 남은 좌석 수가 stored procedure로 정확히 계산되는가? ⭕ (조회 결과 항목 : 항공사명, 운항편명, 출발공항, 도착공항, 출발날짜시간, 도착날짜시간, 요금, 남은 좌석 수)	25
	조회 결과에 대해 요금 기준으로 정렬이 되어 있는가? ⭕ (요금이 저렴한 것에서부터 비싼 순으로)	10
	사용자의 선택에 따라 다른 정렬 기준(예:출발날짜시간 등)으로 결과값을 볼 수 있다. ⭕	15
다) 항공기 예약	로그인한 회원이 자신만의 항공권만을 예약 할 수 있도록 처리되는가 ⭕	10
	남은 좌석이 0이거나 유효하지 않은 운항편에 대해 예외 메시지 또는 구매 불가 처리가 되는가? ⭕	10
	예약 완료 후, 탑승권이 회원에게 이메일로 전송된다. ⭕ 남은좌석수 1인 운항편 예약 --[좌석수-1]--> 다시 검색 :: 검색X (구매 불가 처리)	20
라) 항공기 취소	출발 전인 예약 항공권들을 검색할 수 있다. ⭕ (조회 결과 항목 : 항공사명, 운항편명, 출발공항, 도착공항, 출발날싸시간, 도착날짜시간, 요금)	10
	취소 위약금이 정확하게 계산되고, 환불 금액(= 요금 - 위약금)이 정확히 표시되는가 ⭕	15
	취소 기능이 정상적으로 동작하는가 ⭕	15
마) 최종 예약/취소 내역 조회	예약과 취소 내역을 개별 조회 및 동시 조회할 수 있다. ⭕	15
	사용자 지정 기간 필터링이 정확하게 동작하는가 (예: 시작일, 종료일 범위 지정) ⭕	25
	조회 결과에 요구된 모든 항목이 포함되어 있는가? ⭕ (조회 결과 항목 : 항공사명, 운항편명, 출발공항, 도착공항, 출발날짜시간, 도착날짜시간, 결제 금액 또는 환불 금액, 예약 또는 취소된 날짜 시간)	10
바) 통계 정보 검색	통계 정보를 볼 수 있는 관리자 계정으로 접속이 가능하다. ⭕	10
	그룹 함수 사용 질의가 실제로 관리자에게 유용한 통계적 정보를 제공하는가 ⭕	15
	윈도우 함수 사용 질의가 순위, 누적합 등 관리자에게 중요한 분석을 수행하는가 ⭕	15
사) 예외 처리	잘못된 입력, 조건 불충족 시 시스템이 에러 없이 적절하게 대응하는가 (예 : 필수 항목 미입력 시 "모든 항목을 입력해 주세요" 등)	20
아) UI 편의성	사용자 입장에서 사용하기 쉬운가 ⭕ (예 : 예약 내역이 표 형식으로 정렬되어 있음, 정렬 기능은 버튼 형태로 선택 가능 등)	25
총점		300

-- 모든 체크 리스트 항목 통과⭕

-- DB 연결 ✕ 문제 :

--- 원인 >> MacBook Server <-> Windows Laptop Oracle DB 연결 불안정

6 느낀 점 및 향후 개선 방향

a. 프로젝트 수행 중 배운 점

- Python oracledb 라이브러리를 활용한 Oracle DB 원격 접속 방법
- Python SQL Alchemy ORM 을 활용한 Query 작성 방법

b. 한계점

- 원격 DB 연결 불안정 문제
- FE Template :: 반응형 ✕ (MacBook Air 15" M2 Chrome 전용)
- 문제 및 요구사항 정의 상의 기능 :: 실제 항공기 예약 시스템과 다름 (더 간단함)

c. 보완점

- Web Server 개발환경인 MacBook Air 15" M2 에 DB 이식 (Docker 활용 재구축)
- FE Template :: 반응형 웹 구현
- 실제 항공기 예약 시스템과 동일하게 문제 및 요구사항 정의 --> 구현