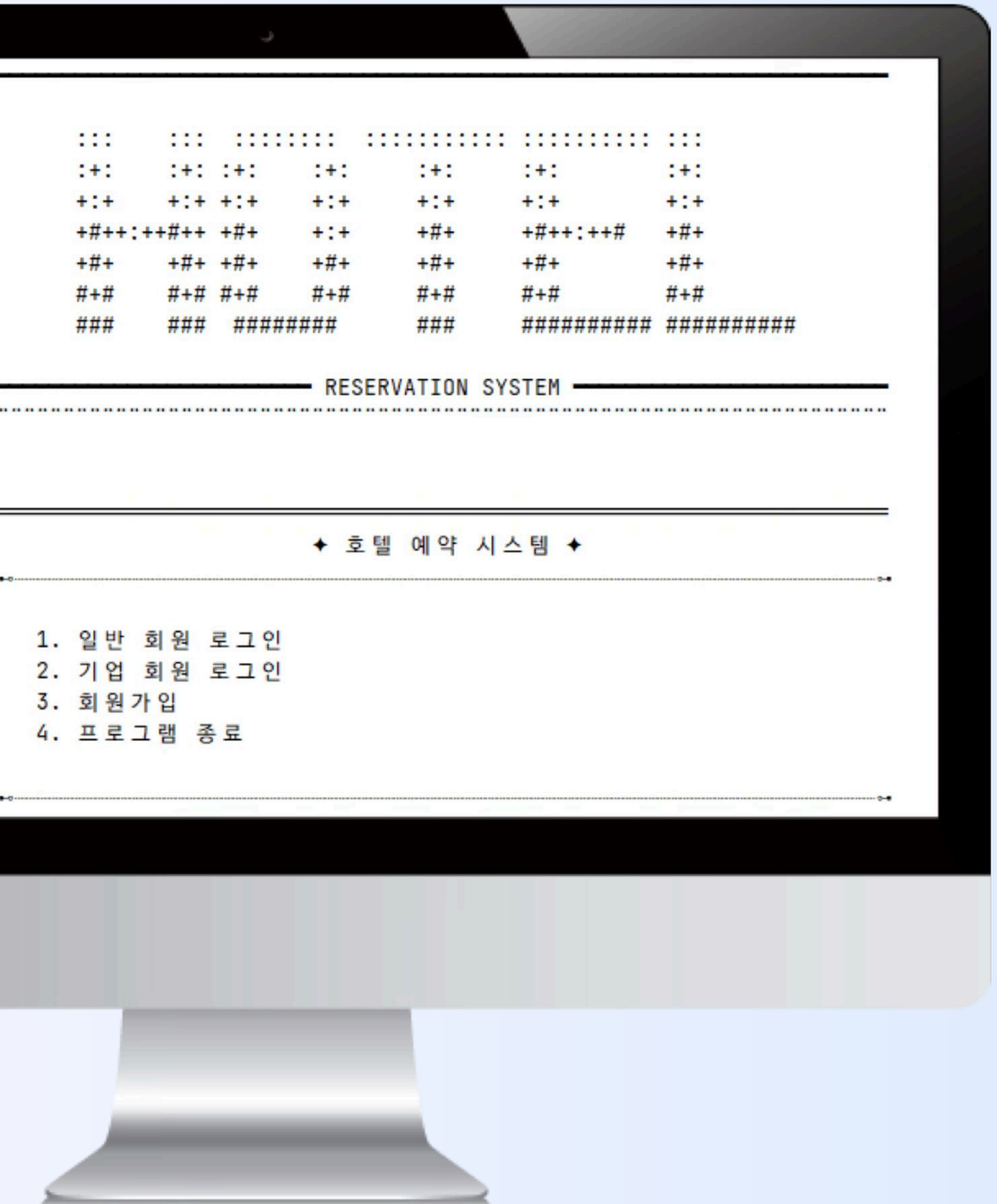
ConsoleProject

호텔관리시스템

파일 입출력 기반의 데이터 처리 Java ConsoleProject



호텔 예약 시스템

전재만, 김정섭, 홍선민, 김서진

25.07.01 ~ 25.07.14(14일)

사용자 중심의 통합 예약 플랫폼

개발환경

SYSTEM :
OS: Windows 11(x64)
Proccesser: intel® Core™ i7-13700(2.10 Ghz)
Memory: 16GB

Java Ver: JDK 17

IDE: Eclipse

사용 기술

파일 입출력 :
BufferedReader
BufferedWriter
FileReader
FileWriter

사용자 입력:
Scanner

컬렉션:
ArrayList<T>
List<T>
Comparator

개요&목적

개요:
1. 호텔을 검색하고 예약할 수 있는 프로그램
2. 호텔정보를 관리할 수 있는 프로그램

목적:
1. 사용자 중심의 통합 예약 플랫폼 구축
2. 사용자편의성과 신뢰도 향상
3. 사용자 분류를 통한 기능 분화
4. 호텔 운영의 효율성과 자율성 확대

담당 업무

공통 업무 :
데이터 구조 설계
패키지 및 클래스 설계

전재만:
더미데이터 생성
로그인
회원가입 및 회원정보 CRUD
예약 확인

김정섭:
더미데이터 생성
예약 관련 CRUD
호텔 관리 CRUD

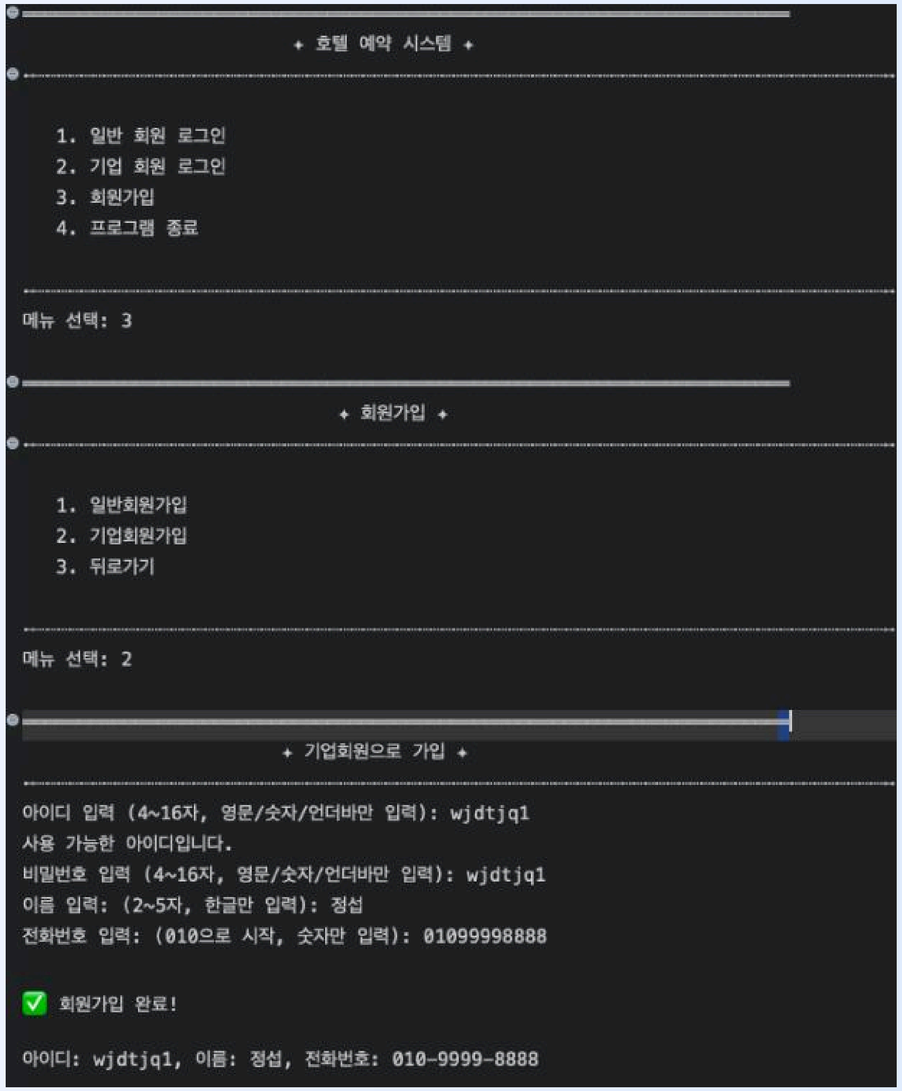
홍선민:
총괄
순서도
호텔 검색 기능
예약신청

김서진:
화면설계
리뷰상세보기
방문조회
리뷰 작성

담당업무

ConsoleProject

담당업무-회원가입



배정된 업무

- 회원가입(일반회원 / 기업회원)
- 로그인(일반회원 / 기업회원)
- 더미데이터(테스트 계정·샘플 데이터) 정리

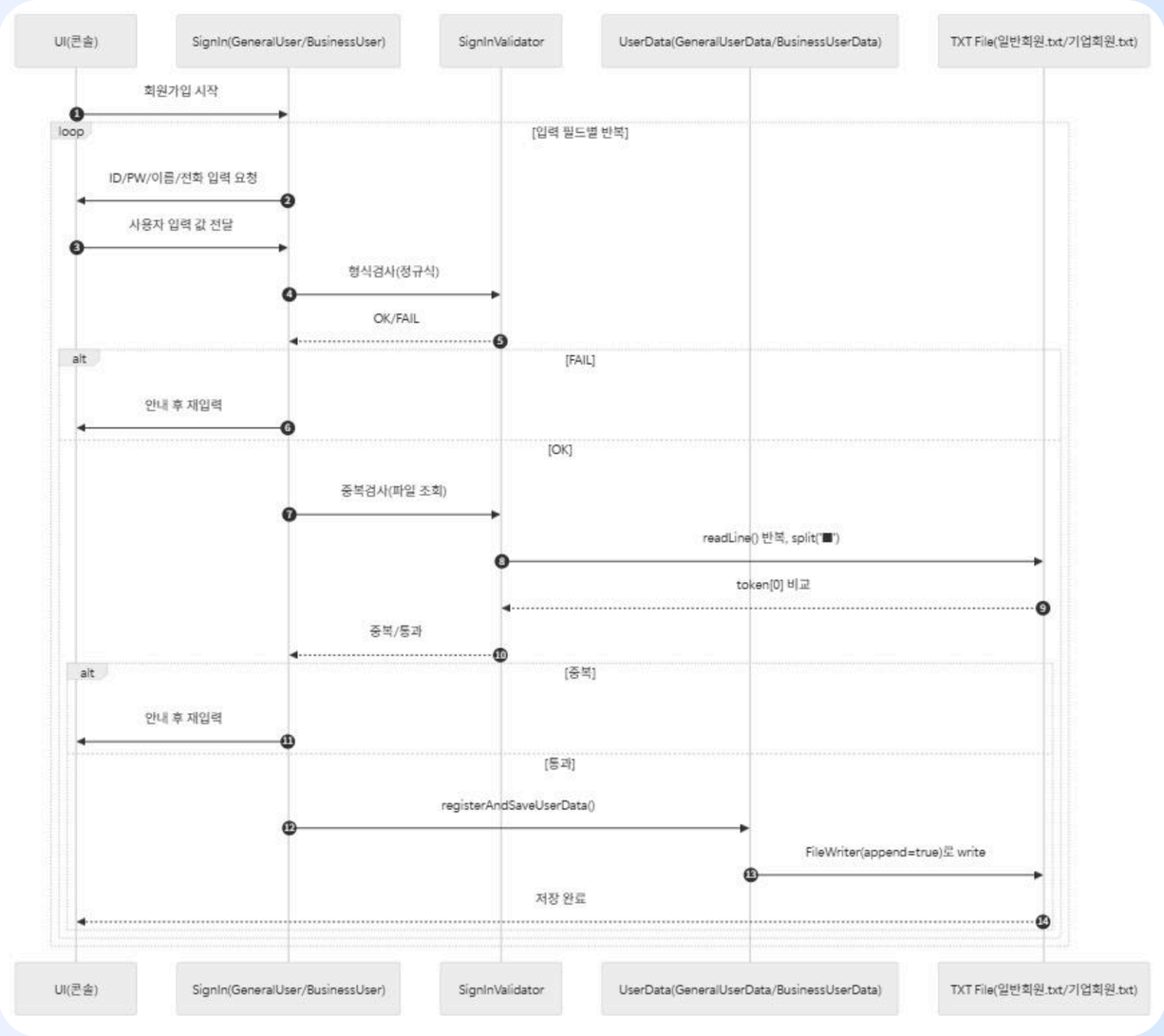
설계 목표

- 콘솔 입력 실수가 잦은 환경에서 검증 → 재입력 → 저장 흐름을 안정적으로 유지
- 파일 기반 저장 구조에서 중복/불일치 데이터가 생기지 않도록 선제 차단

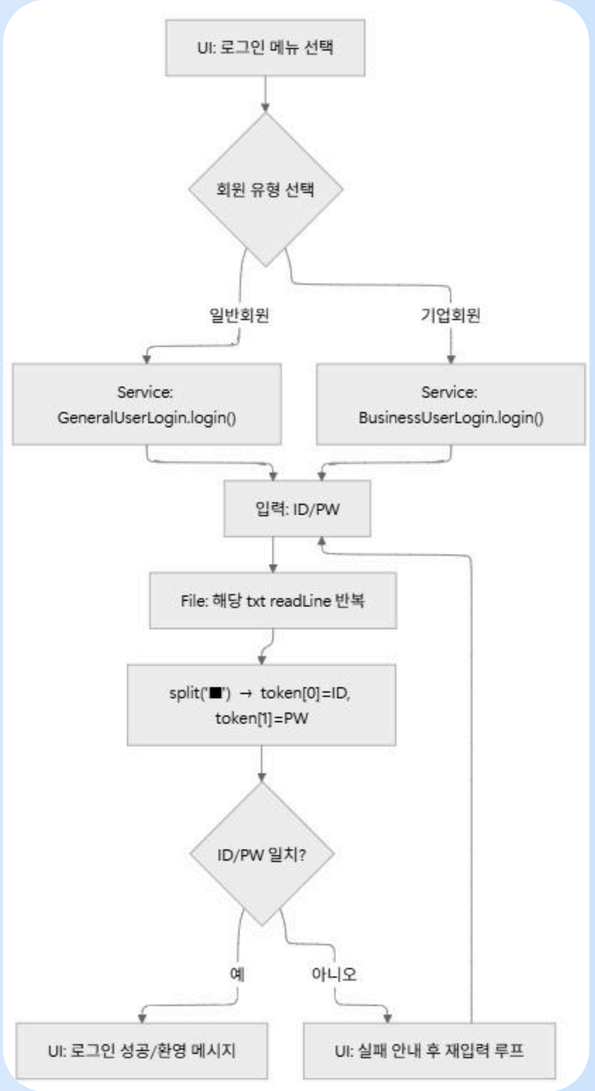
ConsoleProject

담당업무-회원가입

sequenceDiagram



Login Flow



ConsoleProject

SignInValidator.java

SignInValidator는 회원가입 입력값을 다음 순서로 검증

- 형식 검증(정규식)
  - ID / PW / 이름 / 전화번호 입력 규칙을 정규식으로 체크
- 중복 검증(파일 기반)
  - 회원유형(일반/기업)에 따라 텍스트 파일을 읽어
  - 한 줄씩 split("■") 후 tokens[0](ID) 중복 여부 확인
- 최종 판단
  - “형식 OK + 중복 없음”이면 사용 가능한 ID로 승인

입력 규칙 (검증 기준)

- D: ^[a-zA-Z0-9_]{4,16}$
- 4~16자, 영문/숫자/언더바만 허용
- PW: ^[a-zA-Z0-9_]{4,16}$
- 4~16자, 영문/숫자/언더바만 허용
- 이름: ^[가-힣]{2,5}$
- 한글 2~5자
- 전화번호: ^010\\\\\\\\d{8}$
- 010 + 숫자 8자리 (총 11자리)

```
// 1) 형식 검증(Regex)
isValidId(id)       -> "^[a-zA-Z0-9_]{4,16}$"
isValidPassword(pw) -> "^[a-zA-Z0-9_]{4,16}$"
isValidName(name)   -> "^[가-힣]{2,5}$"
isValidPhone(phone) -> "^010\\\\d{8}$"

// 2) 중복 검증(File)
isDuplicateId(id, filePath):
  reader = open(filePath)
  while (line = reader.readLine()) != null:
    tokens = line.split("■")
    if tokens[0].equals(id):
      return true
  return false

// 3) 최종 판단(회원유형별 ID 체크)
generalCheckId(id):
  if !isValidId(id) -> print("올바른 형식이 아닙니다."); return false
  if isDuplicateId(id, ".//dat//일반회원.txt") -> print("중복된 아이디입니다."); return fa
lse
  print("사용 가능한 아이디입니다."); return true

businessCheckId(id):
  if !isValidId(id) -> print("올바른 형식이 아닙니다."); return false
  if isDuplicateId(id, ".//dat//기업회원.txt") -> print("중복된 아이디입니다."); return fa
lse
  print("사용 가능한 아이디입니다."); return true
```

ConsoleProject

Project Review

프로젝트는 원래 5인을 기준으로 요구사항과 작업 범위를 산정했지만,

진행 중 한 명이 **이탈**하면서 4명이서 동일한 볼륨을 소화해야 했습니다.

일정을 지키기 위해 역할 재분배가 필요했고, 기존에 맡은 영역 외에도 다른 파트의 기능을 흡수해 구현해야 했다.

특히 팀 협업이 처음인 상태에서 데이터 포맷, 파일 입출력 규칙을 맞추는 과정이 예상보다 큰 난관이었습니다.

“기능 완성도”를 중심으로 우선순위를 재정렬하고 여유가 있다고 판단되는 팀원에게 분배하였습니다.

그 결과 인원 감소로 인한 리스크가 있었지만, 최종적으로는 기획했던 기능을 목표한 형태로 완성했고,

협업 경험이 부족한 상태에서도 결과물을 만들어냈다는 점이 가장 큰 성취였습니다.

기술적으로는 새로운 라이브러리나 프레임워크를 많이 익혔다기보다,

Java 기반 객체지향 설계(OOD)를 실제로 적용하면서 깊게 이해하게 된 것이 가장 큰 수확이었습니다.

예를 들어 입력 **검증/중복 검증/파일 조회처럼 반복되는 책임을 클래스로 분리하면서 “역할과 책임”이 명확해질수록 유지보수성이 좋아진다**는 걸 체감했습니다.

또한 프로젝트를 마무리하면서 도메인 중심으로 패키지를 정리하는 방법(기능별 분리 vs 도메인 기준 분리)을 학습했고, 코드가 커질수록 **패키지 구조가 협업 속도와 품질에 직접 영향을 준다는 것을 경험으로 알게 되었습니다**.