어드벤처디자인 프로젝트 최종보고서

<table>
<tr><td>팀 명</td><td colspan="5">ANOPOS - 그냘 본 포스기의 이름을 우리는 아직 모른다</td></tr>
<tr><td>작 품 명</td><td colspan="5">ANOPOS</td></tr>
<tr><td>개발기간</td><td colspan="5">2021년 11월 11일 ~ 2021년 12월 10일</td></tr>
<tr><td>지도교수</td><td colspan="2">컴퓨터공학과</td><td colspan="3">윤 현 주</td></tr>
<tr><td>구 분</td><td>학년</td><td>학 번</td><td>성 명</td><td>휴대전화</td><td>E-mail</td></tr>
<tr><td>책임자(팀장)</td><td>2</td><td>20200123</td><td>김다은</td><td colspan="2" rowspan="2"></td></tr>
<tr><td>팀원</td><td>2</td><td>20180287</td><td>김준용</td></tr>
</table>

본인은 어드벤처 디자인 과목에서 수행한 프로젝트 최종보고서를 첨부와 같이 제출합니다.

2021년 12월 11일

팀 장 김다은 (서명)

컴퓨터공학심화프로그램

프로젝트 최종보고서

아노포스

1. 서론

판매 부분에서는 판매 테이블 선택, 구매할 메뉴를 선택, 계산, 영수증 출력, 총 판매 데이터 저장, 재고 추가를 가능하도록 구현한다. 통계 부분에서는 일별, 월별, 연도별 제품 판매량, 제품 판매수량, 사용자 판매량, 사용자 판매수량, 시간별 판매수량, 시간별 판매량 그리고 매출요약을 제공한다. 재고 관리 부분에서는 새로운 메뉴의 추가 재고의 추가가 가능하도록 구현한다. 가게에 상관 없이 포스기를 제공할 수 있도록 메뉴 csv 파일을 입력하면 그에 맞게 ui를 제공하도록 설계한다.

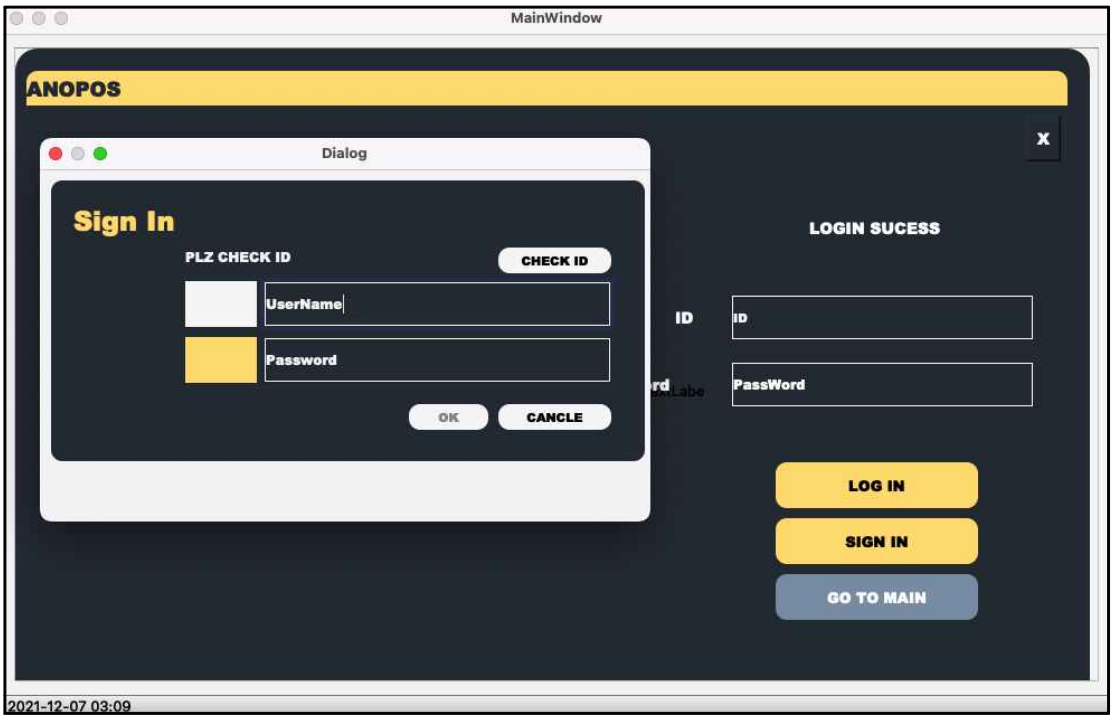
2. 작품 개요

■ 프로그램 구성

아노포스는 로그인 위젯, 메인 위젯, 판매 위젯, 재고 관리 위젯, 통계 위젯으로 구성되어있다.

■ 프로그램 실행

아노프스는 실행 시 **로그인 위젯**을 통해 시작한다. 로그인을 통하여 admin(관리자)유저가 사용자를 추가 재고를 관리할 수 있도록 구현하였고, 이외의 유저들은 교대할 때마다 자신의 아이디로 아노포스에 접근하여 최종적으로 유저의 총 판매 매출을 통계로 제공할 수 있도록 한다.



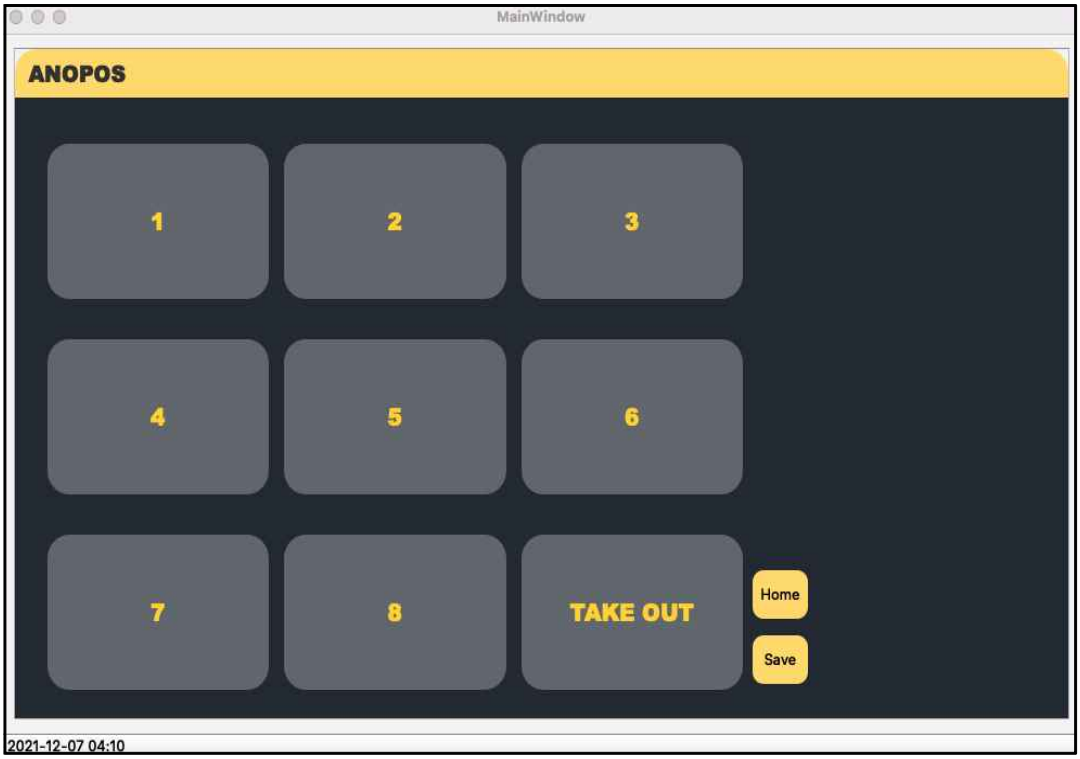
- **로그인 화면**

로그인 이후 실행되는 **메인 화면.** 각 기능마다 버튼을 통해 전환되도록 설계하여 UI를 더 직관적이고 편리하게 구현하였다.

버튼 별 기능	
로그아웃	로그인 화면으로 전환
영업 등록	테이블 관리 매출 등록 / 판매 화면으로 전환
영업 정보 관리	재고 추가와 신메뉴 추가 화면으로 전환
매출 요약	통계 화면으로 전환

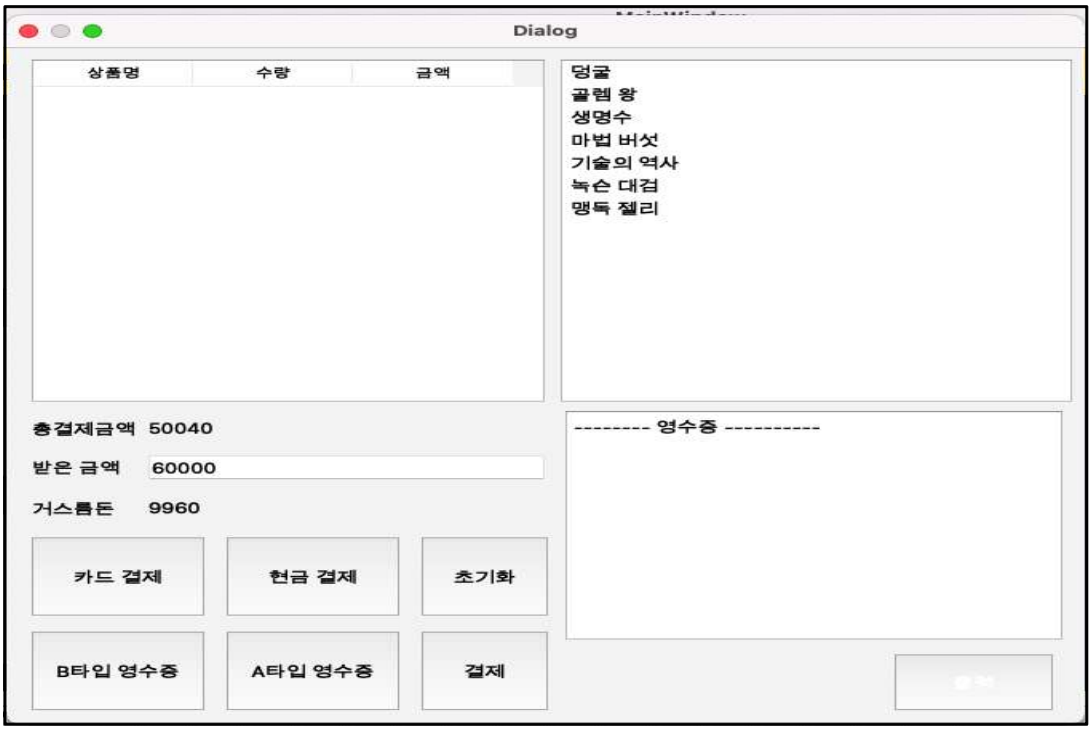
- **메인 화면**

메인 화면으로부터 영업등록 버튼을 클릭하여 전환 된 **영업 등록 화면.**
1~8번의 일반 테이블과 TAKE OUT 테이블을 분리되어, 각각 다른 품목들을 판매한다.



- **영업등록 화면(테이블)**

테이블 버튼을 클릭하면 제공되는 **판매 다이얼로그.** 품목을 등록하고, 카드 결제 및 현금 결제를 및 영수증 출력을할 수 있다. 중간에 결제하지 않고 다이얼로그를 종료하여도 초기화 또는 결제 전까지 해당 등록된 상품의 정보는 남아있다.

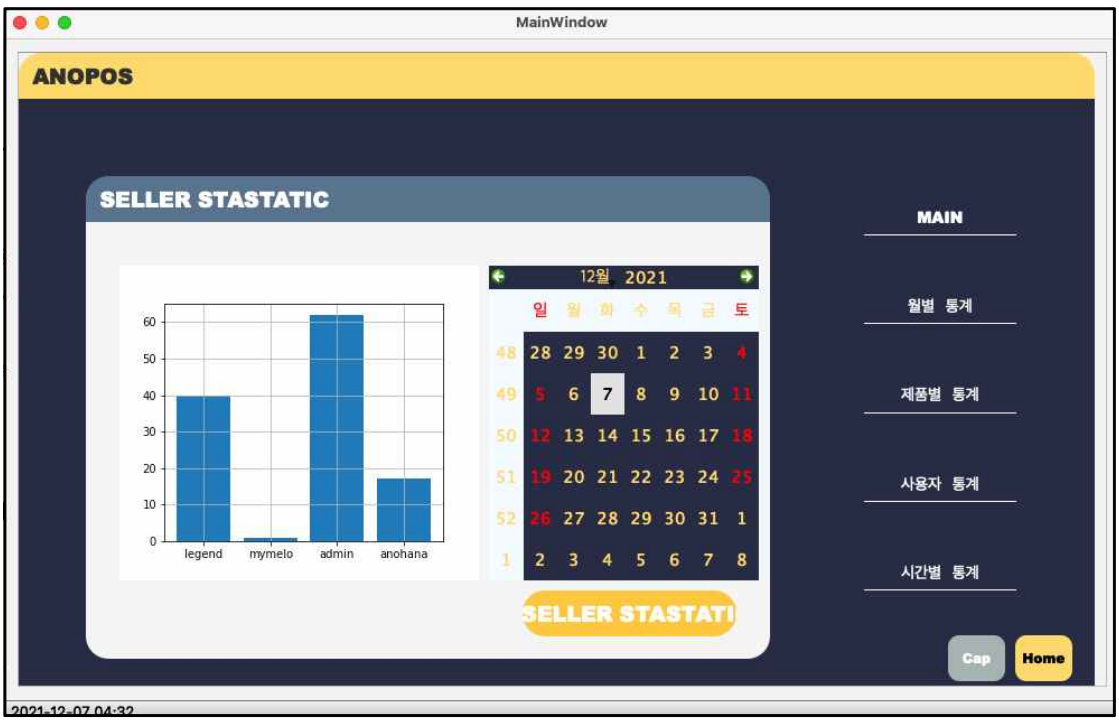


- **판매 다이얼로그**

메인 화면으로부터 영업 정보 관리 버튼을 클릭하여 전환 된 **재고 관리 화면.**
신메뉴를 추가하거나 부족한 재고를 추가할 수 있는 화면이다.

- **재고관리 화면**

메인 화면으로부터 매출요약 버튼을 클릭하여 전환 된 **통계 화면.**
제품별, 사용자별, 시간별 통계 정보를 확인할 수 있는 화면이다.



- **통계 화면**

3. 기능 및 요구사항 분석

3.1 입출력 정의

■ 입력 자료

	A	B	C	D	E	F	G	H
1	상품 코드	카테고리	판매가	품명	하위품목1	하위 품목2	하위 품목3	재고
2	1	골렘	2	덩굴	-1	-1	-1	5
3	2	골렘	3000	골렘 왕의	-1	-1	-1	6
4	3	숲	480	생명수	-1	-1	-1	4
5	4	숲	72	마법 버섯	-1	-1	-1	7
6	5	기술	50000	기술의 역사	-1	-1	-1	9
7	6	상인	4000	녹슨 대검	-1	-1	-1	6
8	7	상인	20	맹독 젤리	-1	-1	-1	9
9	8	테이블	500	용사 정식	용기	골렘의 땀	요정 가루	3
10	9	부재료	50	용기	-1	-1	-1	5
11	10	부재료	100	골렘의 땀	-1	-1	-1	6
12	11	부재료	30	요정 가루		-1	-1	8

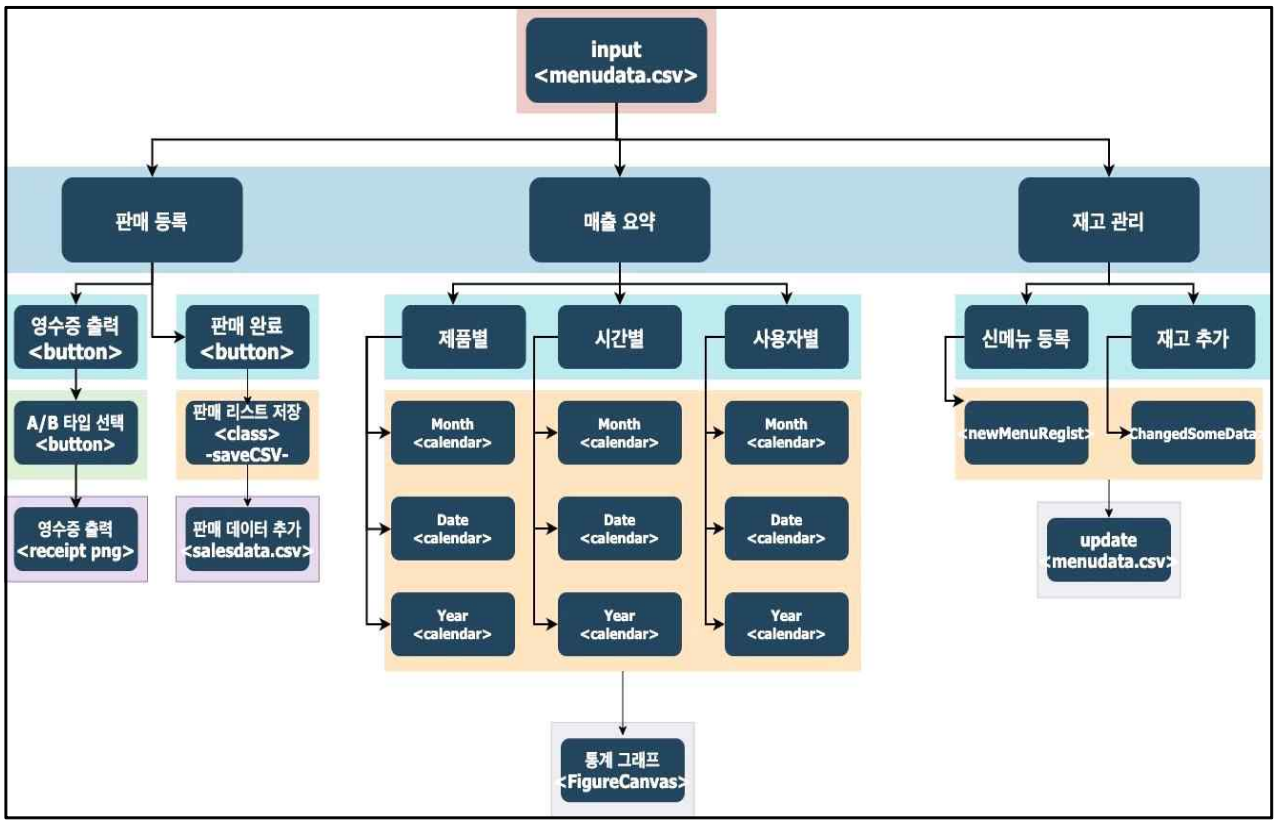
해당 파일은 menudata.csv 라는 이름으로 정의되어 있다. 사용자가 포스기를 이용하기 전에 메뉴에 대한 정보를 입력해 주어야 한다. 프로그램 실행 시 해당 메뉴 데이터를 토대로 판매 메뉴의 수량 및 가격을 설정한다.

■ 출력 자료

상품명	수량	상품금액	날짜	시간	판매자
jelly	2	400	2024-1-12	1:20	admin
mushrom	1	3200	2024-1-2	14:37	kuromi
party table	4	1000	2025-7-6	4:49	admin
courage	2	3000	2020-1-17	18:28	anohana
dinner	3	500	2020-10-12	2:09	mymelo
jelly	2	400	2025-10-5	19:58	legend
water	8	480	2020-1-29	8:12	legend
party table	6	1000	2024-7-13	7:03	legend
mushrom	10	3200	2024-12-2	23:31	kuromi
water	9	480	2020-7-13	5:02	admin
jelly	7	400	2024-1-12	21:39	legend
mushrom	4	3200	2025-4-10	21:24	admin
jelly	7	400	2022-1-12	10:18	legend
mushrom	8	3200	2020-1-2	3:36	mymelo
mushrom	4	3200	2020-1-1	16:23	legend
plant	3	300	2025-7-22	19:07	legend

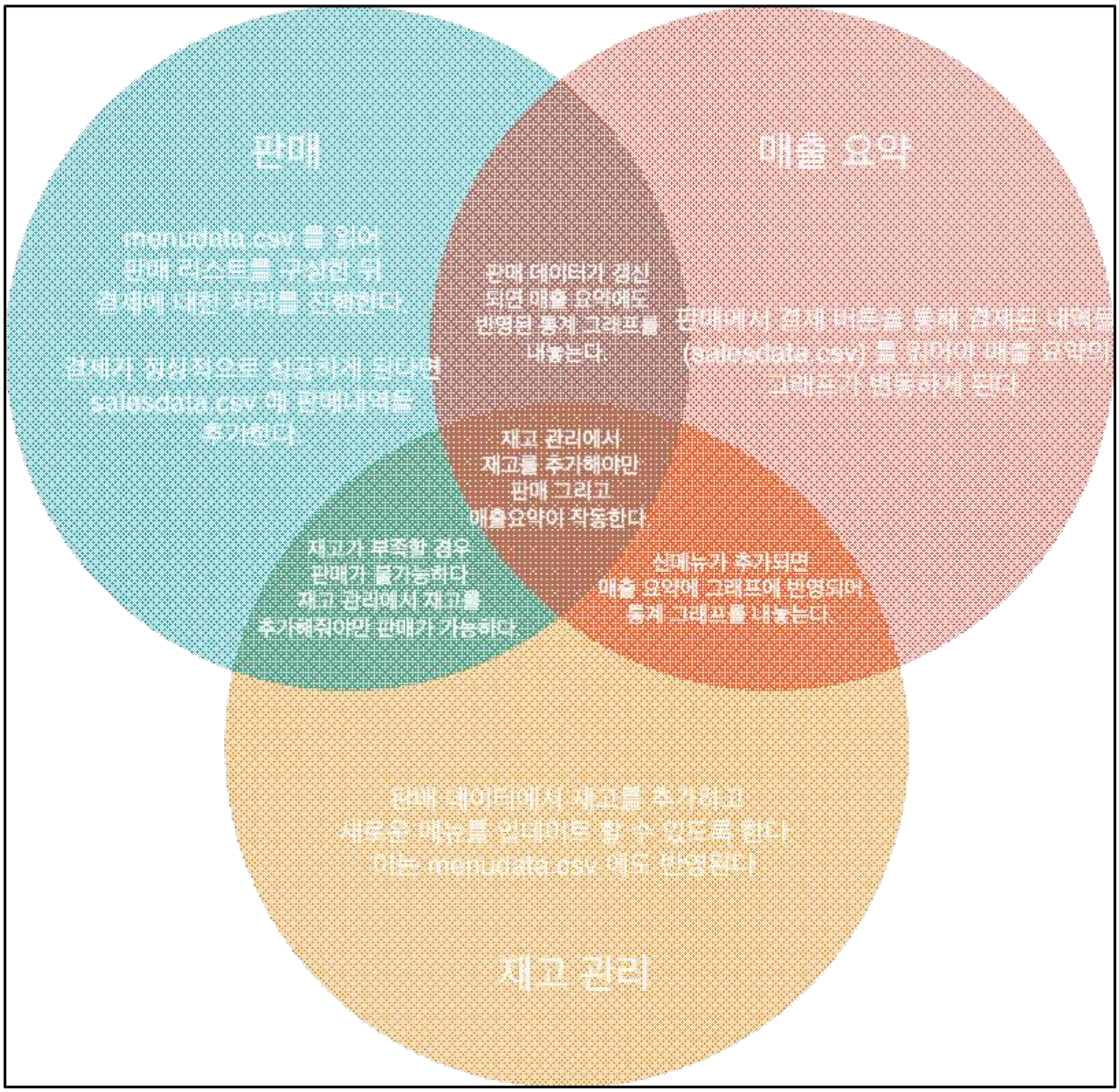
해당 파일은 salesdata.csv 라는 이름으로 정의되어 있다. 포스기에서 상품을 판매하면 update 된다.

■ 입출력 처리



판매등록, 매출요약, 재고관리에 대한 입출력 처리는 위의 그림과 같다. 판매등록에 대한 출력은 영수증출력, 판매 데이터 추가로 정의되고 매출 요약에 대한 출력은 통계 그래프, 재고 관리에 대한 출력은 menudata.csv 업데이트로 정의된다.

3.2 핵심 기능과 관계


판매
menudata.csv 를 읽어
판매 리스트를 구성한 뒤
결제에 대한 처리를 진행한다.
결제가 정상적으로 진행되면
salesdata.csv 에 판매내역을
추가한다.
매출 요약
판매에서 결제 버튼을 통해 결제된 내역을
(salesdata.csv) 를 읽어와 매출 요약의
그래프가 변동하여 된다.
재고 관리
판매 데이터에서 재고를 추가하고
새로운 메뉴를 입력하여 할 수 있도록 한다.
이는 menudata.csv 에도 반영된다.
관매 데이터가 갱신
되면 매출 요약에도
반영된 통계 그래프를
내놓는다.
재고 관리에서
재고를 추가해야만
판매 그리고
매출요약이 작동한다.
재고가 부족할 경우
판매가 불가능하다
재고 관리에서 재고를
추가해줘야만 판매가 가능하다.
신메뉴가 추가되면
매출 요약에 그래프에 반영되어
통계 그래프를 내놓는다.

3.3 요구사항 분석

<아노포스>

아노포스 요구 사항

날짜	버전	비고	작성자
2021-11-11	1.0	사용자 요구 1차 분석	김준용, 김다은
2021-11-20	2.0	사용자 요구 2차 분석	김준용, 김다은

서비스 개요

아노포스는 메뉴에 대한 정보 (menudata.csv) 를 입력받아 해당 메뉴들로 구성된 포스기를 제공하고 판매, 결제, 영수증, 현금결제, 카드결제 등을 처리할 수 있도록 한다. 또한, 매출 통계 및 매출 요약을 제공하여 연도, 월, 일별마다 제품, 시간, 사용자별 판매량/ 판매 수량을 그래프로 시각화하여 제공한다. 재고 추가 및 신메뉴 추가도 가능하도록 하여 사용자의 편의를 고려한다.

기능 요구사항

- 문제 해결 범위
  - 판매 품목을 사용자가 자유롭게 설정할 수 있도록 함
  - 테이블마다 판매를 진행
  - 테이블을 클릭하면 제공되는 다이얼로그를 통해 결제 상품 등록을 가능하도록 함
  - 재고 관리를 가능하도록 함
  - 신메뉴 추가를 가능하도록 함
  - 제품별, 사용자별, 시간별 통계를 제공하도록 함
  - 사용자 로그인 및 로그아웃을 가능하도록 함

대체 흐름

- 입력 오류
  - 숫자가 입력되어야 하는 edit 라벨에서 숫자가 아닌 한글이 입력될 경우 이에 대한 예외 처리를 해준다.
  - 회원가입시 id가 중복될 경우 이에 대한 예외 처리를 해준다.

- 동작 오류
  - 재고가 부족한데 결제를 진행하려고 하는 경우, 재고가 부족하다는 알림을 띄워준다.

사용 시나리오

[시나리오]
사용자가 상품을 주문한다. (식사 주문의 경우도 동일)

[액터]
사용자(POS기 사용자)

[액션 흐름]
1. 고객이 필요한 아이템을 주문한다.
2. 사용자가 메인 메뉴의 "명단 등록 메뉴"를 선택한다.
3. 시스템이 명단 등록 화면(테이블)로 화면을 전환 한다.
4. 사용자가 명단 등록 대화창에서 테이크아웃을 선택한다.
5. 시스템이 주문목록 대화창을 출력한다.
6. 사용자가 주문 받은 아이템을 추가한다.
7. 시스템이 해당 아이템의 잔여수량을 파악한다.
8. 시스템이 해당 아이템을 주문목록에 추가한다.
9. 고객의 요청에 따라, 사용자가 카드결제 혹은 현금결제를 선택한다.
10. 현금결제의 경우 시스템이 거스름 돈을 계산하여 출력한다.
11. 사용자가 고객에게 아이템을 제공한다.

[대체 시나리오]
주문한 수량보다 현재 재고가 부족한 경우

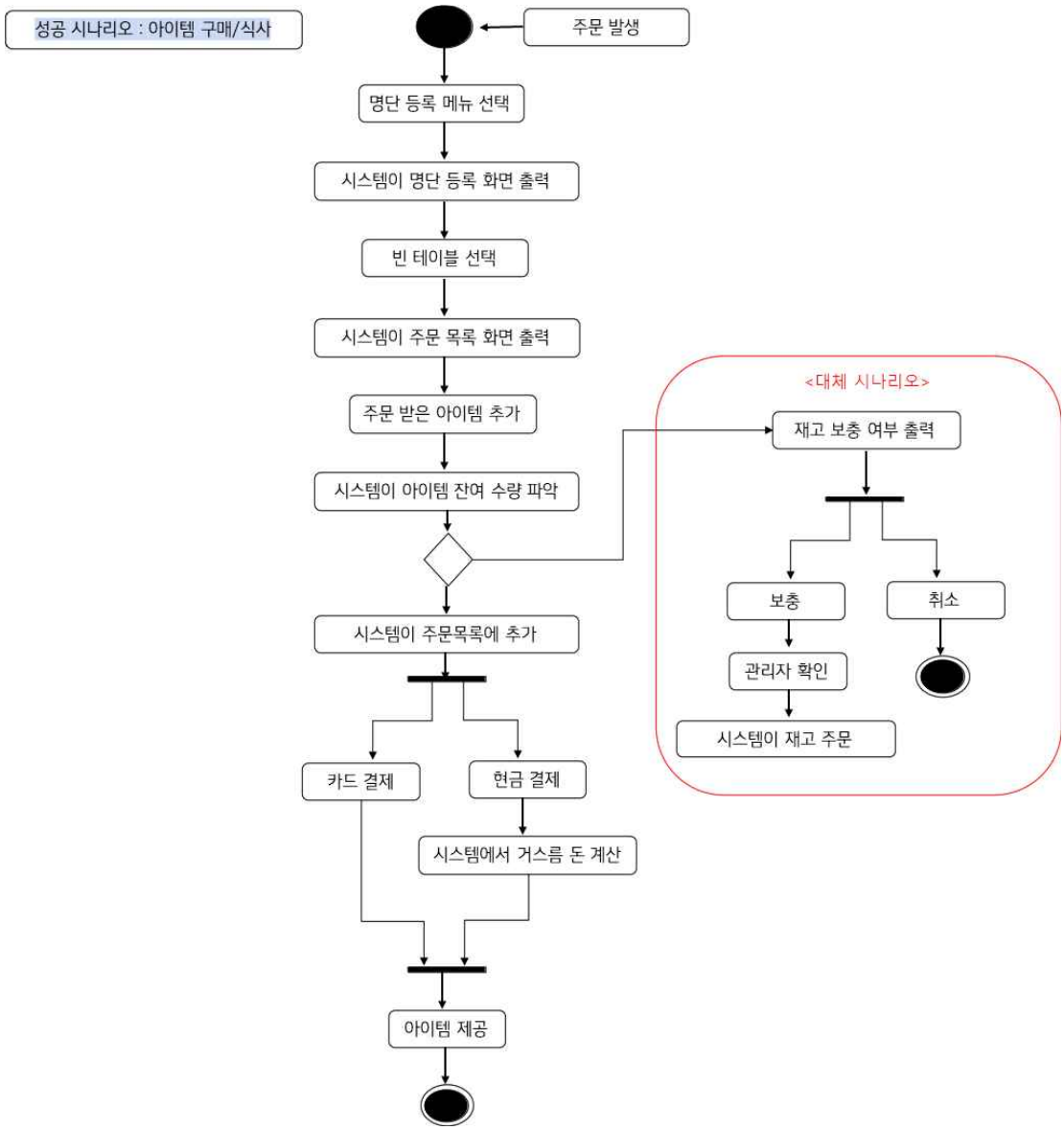
[액터]
사용자(POS기 사용자)

[액션 흐름]
1. 재고 보충 여부를 출력한다.

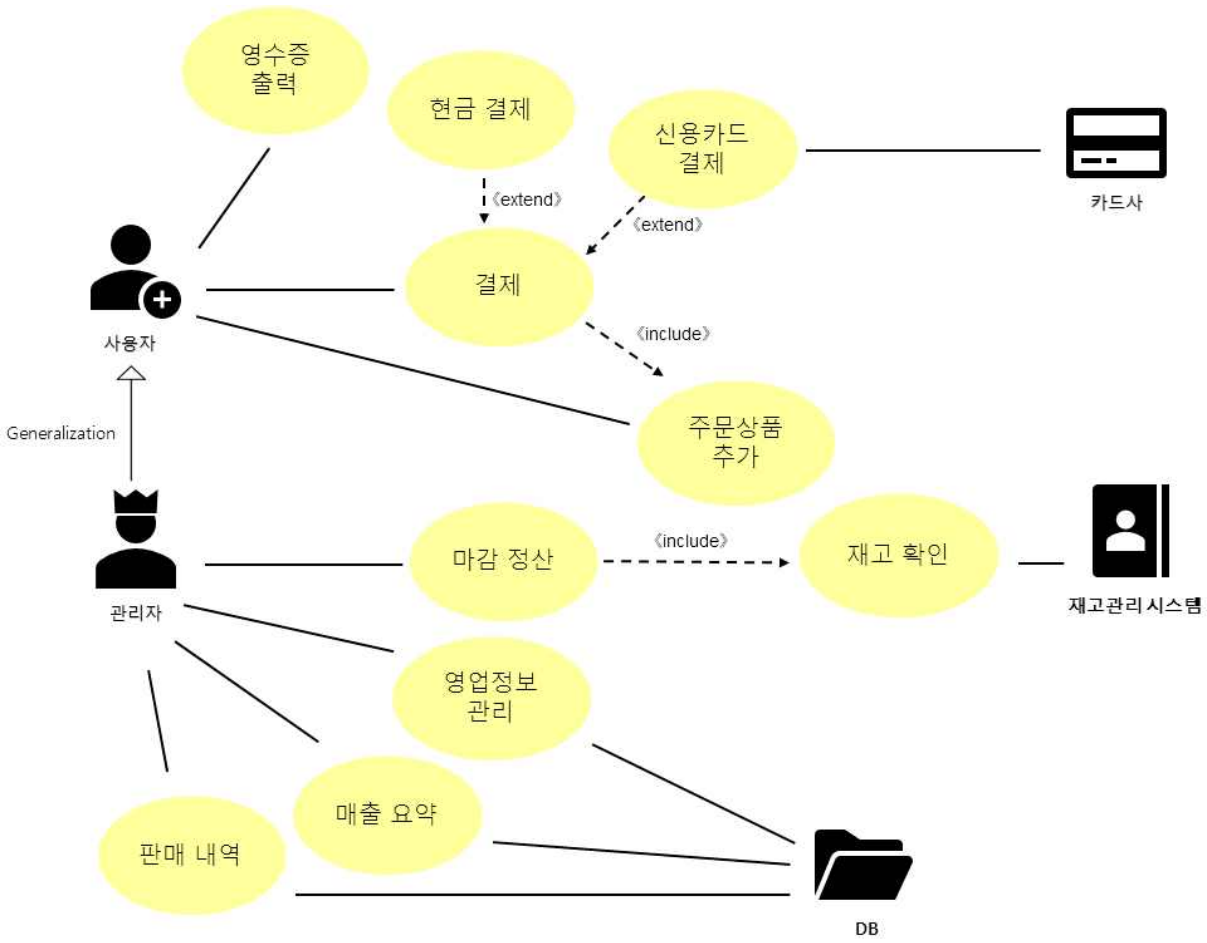
(case : 1 보충을 선택)
2. 사용자가 재고 보충을 선택한다.
3. 시스템이 재고 주문 권한이 있는지 확인한다.
4 시스템이 재고를 주문한다.

(case : 2 취소를 선택)
2. 취소를 선택할 경우 주문을 취소한다.

시나리오를 표현한 상태 차트

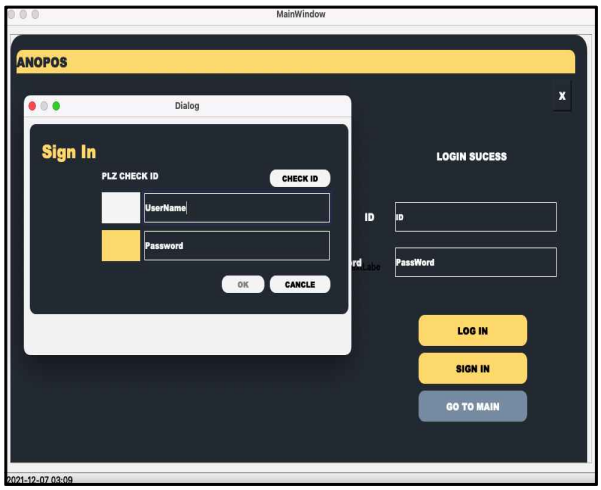


USE_CASE


영수증
출력
현금 결제
신용카드
결제
카드사
《extend》
《extend》
결제
사용자
《include》
주문상품
추가
Generalization
관리자
마감 정산
《include》
재고 확인
재고관리시스템
영업정보
관리
매출 요약
판매 내역
DB

3.4 화면별 설명

■ 로그인 화면



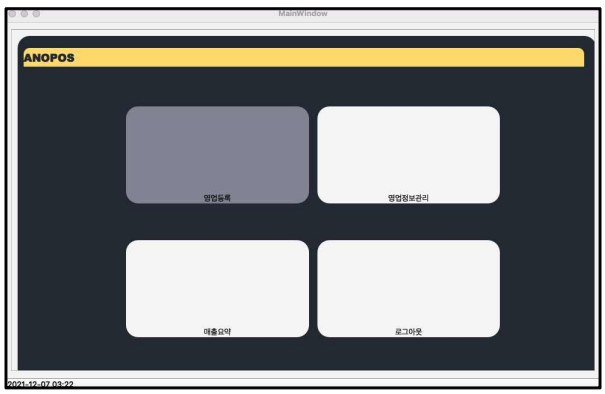
[로그인]

- 로그인 위젯은 맨 처음 프로그램이 시작될 때 보여지는 위젯이 된다.
- 로그인 위젯에서 login success 하기 전까지 메인 메뉴로 넘어가는 버튼이 활성화되지 않는다.
- 로그인을 클릭하여 success 한다면 로그인창 은 사라지고 메인 화면으로 넘어갈 수 있게 된다.
- User Name과 PassWord가 저장되어있는 pickle 파일과 일치해야 success 된다.
- id 와 비번이 없을경우 sign in을 통해 회원 가입을 진행한다.
- 초기설정 (User Name : admin, Password : 1234) 루트 권한으로 접근할 시에는 메인 메뉴에 모든 버튼이 enable 되며 signin 버튼이 활성화 된다.

[회원가입]

- 탐색을 통해 userName 이 중복되는지 검색
- 모든 입력이 완료될 시에 확인을 누르면 UserInfo.pkl 에 새로운 회원 정보가 입력된다.

■ 메인 화면

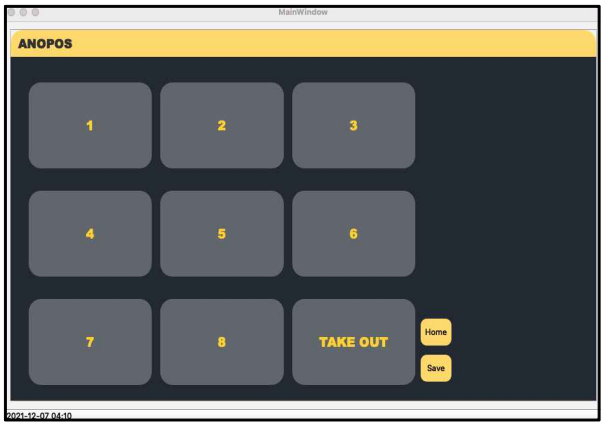


[메인 화면]

- 로그인 성공 시 뜨게 되는 화면
- 로그인한 유저의 권한에 따라 enable 되는 버 튼이 달라진다.

- 영업 등록 : 각 테이블 및 TAKE OUT 주문을 처리할 수 있는 화면으로 이동한다.
- 영업 정보 관리 : 전체 상품 목록에 대한 재고 현황을 살피고 재고 추가 및 새 상품 등록이 가능하다.
- 매출 요약 : 매출에 대한 통계 정보를 시각화하여 제공한다.

■ 판매 등록 화면



[영업 등록]

- 메인 화면에서 영업등록을 누르면 전환되는 화면
- 다이얼로그에서 결제를 누르면 salesdata.csv 에 품목이 추가되게 된다.
- 홈 버튼을 누르면 메인메뉴 화면으로 전환되게 된다.
- 숫자가 입력된 테이블에서는 테이블용 상품만 판매하도록 하고, TAKE OUT이 표기된 버튼에서는 TAKE OUT용 제품들만이 판매된다.
- status bar에는 현재 시각이 나타난다.

[판매 등록]

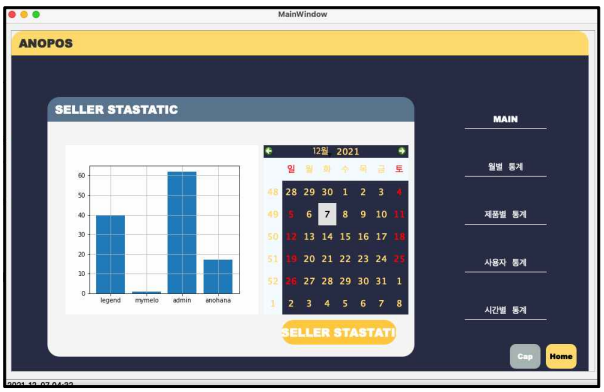
- 영업 등록 화면에 있는 버튼을 클릭할 시 출력되는 다이얼로그.
- 총 결제 금액과 받은 금액 거스름돈이 항상 업데이트 되게 된다.
- 영수증은 A타입과 B타입으로 나누어져 출력이 가능하다.
- 상품 등록 시 현재 존재하는 재고보다 등록하려는 재고가 더 많을 시 재고가 부족하다는 메시지 출력 및 해당 품목의 재고를 추가할 수 있는 다이얼로그를 생성한다.
- 결제를 누르기 전까지 각 버튼마다 다이얼로그는 현재 정보를 계속 유지하고 있게 된다.
- 상품 품목 리스트를 클릭하면 출력되는 다이얼로그는 수량을 추가할 수 있다.
- 카드 결제와 현금 결제를 선택할 수 있다.

■ 재고 관리 화면

[영업 정보 관리]

- 메인에서 영업 정보 관리 버튼을 누르면 전환되는 화면이다.
- 제품 등록 csv 파일을 읽어 테이블 위젯에 업데이트 해준다.
- 테이블 리스트에 품목을 클릭하면 상품 코드, 카테고리, 판매가, 품명이 입력되고 재고에 값 을 바꾸면 상품 목록의 재고가 업데이트 된다.
- 새로운 상품 코드가 입력되게 되면 새로운 상품이라고 판단하고 csv 파일 하단에 추가하여 준다.
- 홈 버튼을 누르면 메인으로 돌아갈 수 있다.

■ 통계 화면

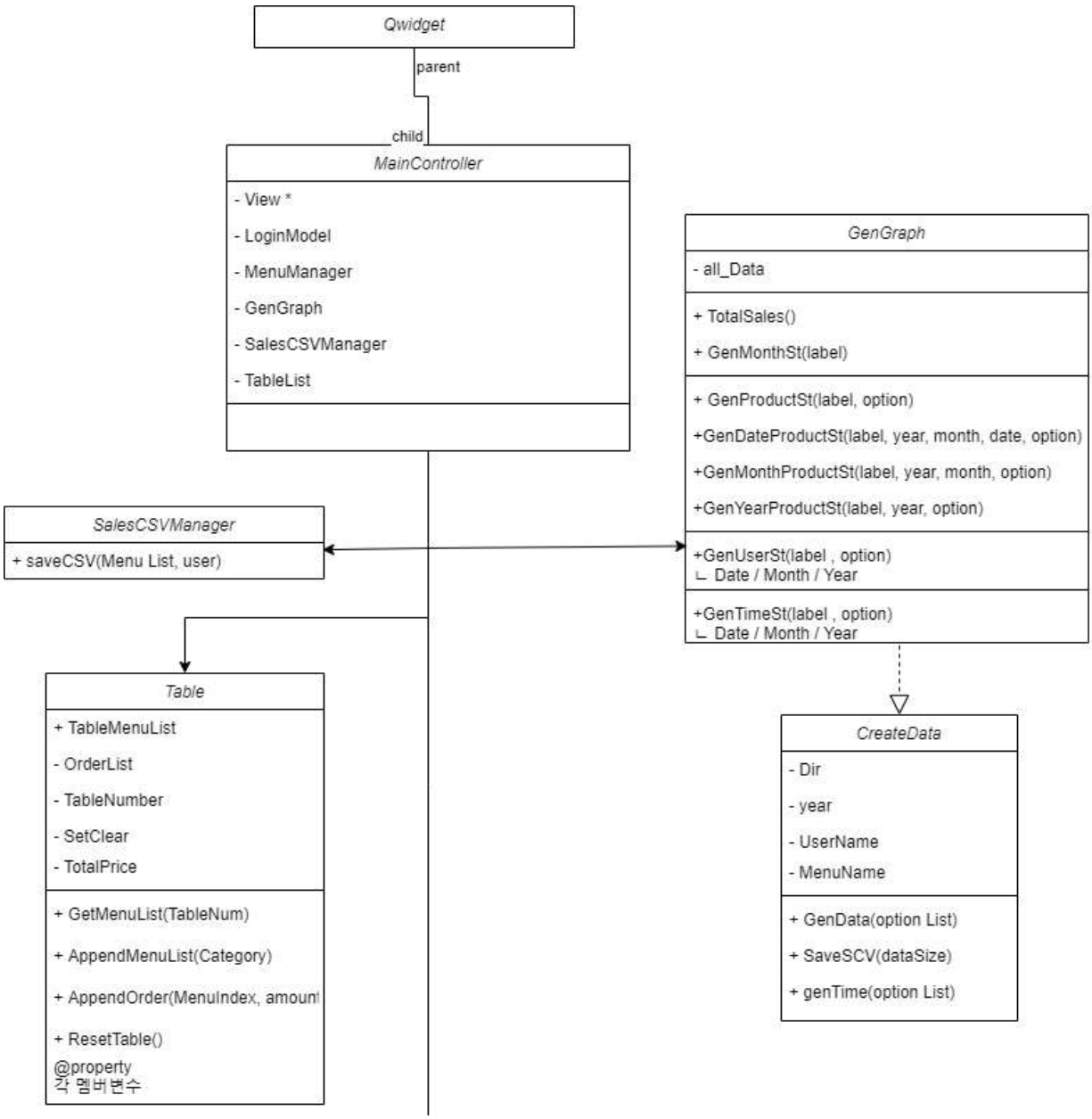


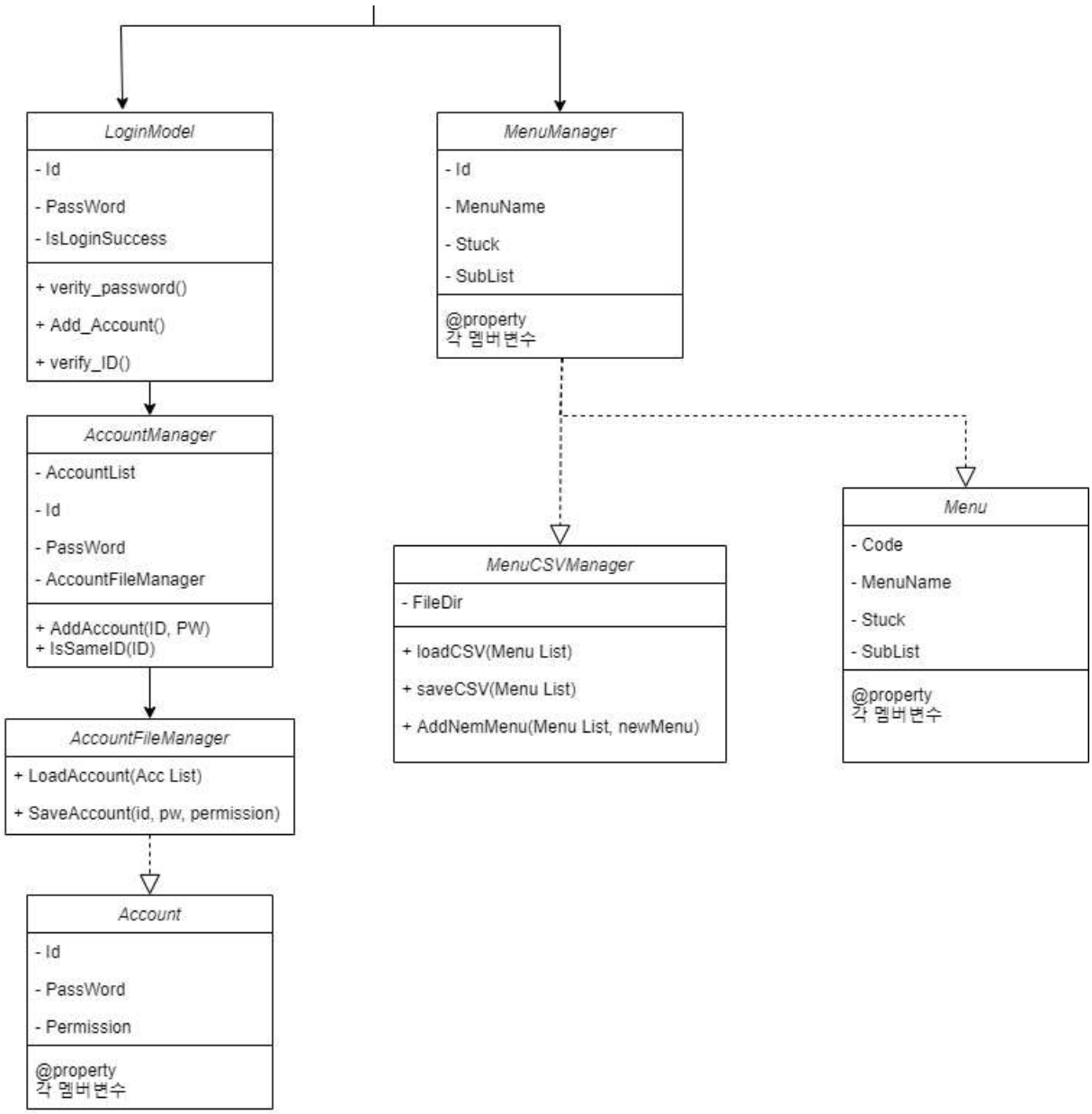
[매출 요약]

- 메인에서 매출 요약 버튼을 클릭하면 전환되는 화면이다.
- 달력에 기간을 클릭하면 해당 기간의 매출액을 시각화하여 확인할 수 있다.
- 매출수량/매출액 체크 버튼과 연도/월 토글을 위한 체크 박스가 존재한다.
- 연도와 달을 바꾸는 부분을 클릭하면 체크박스의 상태에 따라 그래프를 보여준다.
- 날짜를 클릭하면 해당 일자의 통계량을 보여준다.
- 판매량 체크박스의 상태에 따라 판매 수량 혹은 판매액을 보여준다.
- 판매 품목, 시간별, 유저 별 통계량을 확인할 수 있으며 버튼으로 해당 화면에 접근이 가능하다.
- 홈 버튼을 누르면 메인으로 돌아갈 수 있다.

4. 설 계

4.1 클래스 설계


Qwidget
parent
child
MainController
- View *
- LoginModel
- MenuManager
- GenGraph
- SalesCSVManager
- TableList
GenGraph
- all_Data
+ TotalSales()
+ GenMonthSt(label)
+ GenProductSt(label, option)
+GenDateProductSt(label, year, month, date, option)
+GenMonthProductSt(label, year, month, option)
+GenYearProductSt(label, year, option)
+GenUserSt(label , option)
ㄴ Date / Month / Year
+GenTimeSt(label , option)
ㄴ Date / Month / Year
SalesCSVManager
+ saveCSV(Menu List, user)
Table
+ TableMenuList
- OrderList
- TableNumber
- SetClear
- TotalPrice
+ GetMenuList(TableNum)
+ AppendMenuList(Category)
+ AppendOrder(MenuIndex, amount
+ ResetTable()
@property
각 멤버변수
CreateData
- Dir
- year
- UserName
- MenuName
+ GenData(option List)
+ SaveSCV(dataSize)
+ genTime(option List)


LoginModel
- Id
- PassWord
- IsLoginSuccess
+ verity_password()
+ Add_Account()
+ verify_ID()
AccountManager
- AccountList
- Id
- PassWord
- AccountFileManager
+ AddAccount(ID, PW)
+ IsSameID(ID)
AccountFileManager
+ LoadAccount(Acc List)
+ SaveAccount(id, pw, permission)
Account
- Id
- PassWord
- Permission
@property
각 멤버변수
MenuManager
- Id
- MenuName
- Stuck
- SubList
@property
각 멤버변수
MenuCSVManager
- FileDir
+ loadCSV(Menu List)
+ saveCSV(Menu List)
+ AddNemMenu(Menu List, newMenu)
Menu
- Code
- MenuName
- Stuck
- SubList
@property
각 멤버변수

4.2 알고리즘 설계

■ [Login - view]

```
class Login(QWidget, form_class):
    def __init__(self):
        super().__init__()
        // UI 셋업
        // 계정 관리 클래스 객체 생성
        // 계정의 정보를 가져옴
        // 로그인 버튼과 연결
        // 종료 이벤트 cancle 버튼과 연결
    def 로그인 슬롯
        // 텍스트 라인에 입력된 텍스트 정보 받기
        // account 에 존재하는 아이디인지 true false 반환하는 함수 실행
        // true 라면 // 비밀번호 텍스트 정보를 a 와 비교하여 만약 일치한다면 suceess
        // 실패한다면 fail massagebox 출력
    def 종료 슬롯
        // 다이얼로그 닫음
```

■ [Account class]

```
class Account:
    def __init__(self, _id="", _password="", _sudo=False):
        // 디폴트 _sudo false
        // 초기화

        # Property Setter 생성
```

■ [AccountProcess class]

```
# 사용자 정보를 담고있는 pickle 파일 처리 클래스 임포트
class 계정 관리자:
        계정을 딕셔너리로 관리
        picke 파일을 딕셔너리 형태로 변환
        def __del__(self):
                file_manager = AccountFileManager('./account.pkl')
                  file_manager.save(self.__account_list)    @property def account_list(self):
                return self.__account_list
        def 로그인(아이디, 비밀번호):
                if( 아이디를 킷값으로 조회 했을 때 아이디가 존재하지 않음 ):
                        false를 리턴
                else:
                        if 비밀 번호가 일치함 :
                                true를 리턴함
                        else:
                                false를 리턴함
 # 계정 추가
아이디가 중복되는지 검사 후
중복되지 않는다면 파일에 사용자 추가
# 계정 삭제
pickle 파일에 해당 아이디에 해당하는 사용자를 삭제
```

■ [GenGraph = model]

```
class GenGraph:
    def __init__(self):
        self.all_data->pd.read (sales data <csv>)
        // year, month, date 에 대해 tokenizing
        // dropna
        // 수량과 상품 금액 숫자로 전환
        // 판매량 = 수량 * 상품 금액
        // 시간은 데이트 타임에 맞게 변형한다.

        // 모든 함수(기능)에서 공통적으로 필요한 전처리는 init 에서 수행하였다.

    def TotalSales(self):
        리스트 변수에 추가 self.all_data['판매량'].sum()
        리스트 변수에 추가 self.all_data['수량'].sum()
        // 판매자 별 상품금액을 모두 더해 리스트에 append
        리스트 리턴
```

[제품별 처리]

```
        def GenMonthProductSt(self, label, year, month, option):
            mask 를 통하여 year과 month 에 해당하는 정보만 걸러낸다.
            groupby 상품명을 거친후 판매량에 대해 옵션별로 sum 을 해준다.
            옵션으로는 판매량 , 판매수량이 올 수 있다.
            x : 제품별 y : 옵션에 대한 수량 그래프를 생성한다.
            캔버스에 그린다.

        def GenYearProductSt(self, label, year, option):
            mask 를 통하여 year 에 해당하는 정보만 걸러낸다.
            groupby 상품명을 거친후 판매량에 대해 옵션별로 sum 을 해준다.
            옵션으로는 판매량 , 판매수량이 올 수 있다.
            x : 제품별 y : 옵션에 대한 수량 그래프를 생성한다.
            캔버스에 그린다.
```

[사용자별 처리]

```
## user 관련
def GenUserSt(self, label , option):
    saleman = self.all_data['판매자'].unique()
    판매자에 대해 unique 한 데이터를 추출하여 x 축으로 지정
    판매자에 대한 판매량, 판매금액을 y 축으로한다.
    bar 을 이용하여 그래프를 생성한다.
    그래프를 그린다.

def GenDateUserSt(self, label, year, month, date, option):
    마스크를 통해 입력 받은 연도, 월, 일에 해당하는 정보들을 추출한다.
    이후는 앞의 user 처리 과정과 동일하다.

def GenMonthUserSt(self, label, year, month, option):
    마스크를 통해 입력 받은 연도, 월에 해당하는 정보들을 추출한다.
    이후는 앞의 user 처리 과정과 동일하다.

def GenYearUserSt(self, label, year, option):
    마스크를 통해 입력 받은 연도에 해당하는 정보들을 추출한다.
    이후는 앞의 user 처리 과정과 동일하다.
```

[시간별 처리]

```
## TIME 관련
def GenTimeSt(self, label, option):
    result3 = self.all_data.groupby(['hour'])[option].count()
    시간에 대해 groupby 를 거친 뒤 option 에 해당하는 부분만 추출하여 각각을 count 한다.
    hour 에 대해 묶은 뒤 list comprehension 을 통해 시간 x 축을 생성한다.
    시간 x 축 option 에 대한 count 값 y 축인 그래프를 생성한다.
    그래프를 그린다.

def GenDateTimeSt(self, label, year, month, date, option):
    연도, 월, 일에 대해 mask 를 진행한다.
    앞의 date 에 대한 동일한 과정을 수행한다.

def GenMonthTimeSt(self, label, year, month, option):
    연도, 월에 대해 mask 를 진행한다.
    앞의 date 에 대한 동일한 과정을 수행한다.

def GenYearTimeSt(self, label, year, option):
    연도에 대해 mask 를 진행한다.
    앞의 date 에 대한 동일한 과정을 수행한다.
```

■ [Menu class]

```
class Menu:
    def __init__(self, __code, __category, __menuName, __price, __sublist, __stuck):
        [멤버 변수]
        //코드, 카테고리, 메뉴명, 가격, 부자재[리스트], 수량



    @property
     //메뉴 클래스의 멤버 변수 property
```

■ [Menu CSVManager]

```
//CSV 관리하는 Pandas 및 csv 모듈을 import
//Menu Class import
class MenuCSVManager
    [멤버 변수]
    // CSV 주소

    [메뉴 데이터 생성]
    def loadCSV(MenuList)
        // MemuManager에 존재하는 MenuList를 매개변수(참조)
        // CSV로부터 메뉴 데이터를 읽어온다.
          for i in range(데이터의 길이):
        // 데이터의 길이만큼 반복문을 수행하며 메뉴 데이터를 임시메뉴 데이터로 저장
        // 임시 메뉴 데이터 를 MenuList에 Append.
        // 메뉴의 수(데이터의 길이)만큼 반복적으로 추가

    [신메뉴 등록]
    def AddNewMenu(MenuList, NewList):
        // View에 입력된 새로운 메뉴 데이터를 Contoller를 통해 전달받음.
        // 해당 메뉴데이터를 통해 메뉴 객체 생성
        // MenuList에 Append.
```

■ [Menu Manager]

```
//MenuCSVManager Class import
    [멤버 변수]
    // 전체 메뉴 데이터가 저장된 리스트
    // CSV 메뉴 매니저
class MenuManager:

    @property
     //전체 메뉴리스트 반환

    [메뉴 추가]
    def AddMenu(NewList)     //새로운 메뉴 리스트를 인자로 전달 받음.
        for Menu in menuList
            if Menu.code == 새로운 메뉴 코드
                // 메뉴 코드 중복을 검사
                //  중복 시 return
            if Menu.menuName == 새로운 메뉴 명
                // 메뉴 이름의 중복을 검사
                //  중복 시 return
        // 중복 되지않은 경우 csv_manager를 통해 새로운 메뉴를 리스트에 추가

    [재고 추가]
    def AddStuck(code, stuck) // 메뉴 코드와 수량을 인자로 받음.
        for Menu in menuList:
            if 메뉴리스트에 현재 코드와 일치하는 메뉴 존재
                // 해당 메뉴의 수량을 인자로 전달받은 수량으로 변경
                // return을 통한 함수 종료
       // 코드가 일치하는 메뉴가 없을 시,
       // '해당 메뉴코드가 존재하지 않음.' 출력
```

■ [Table Class]

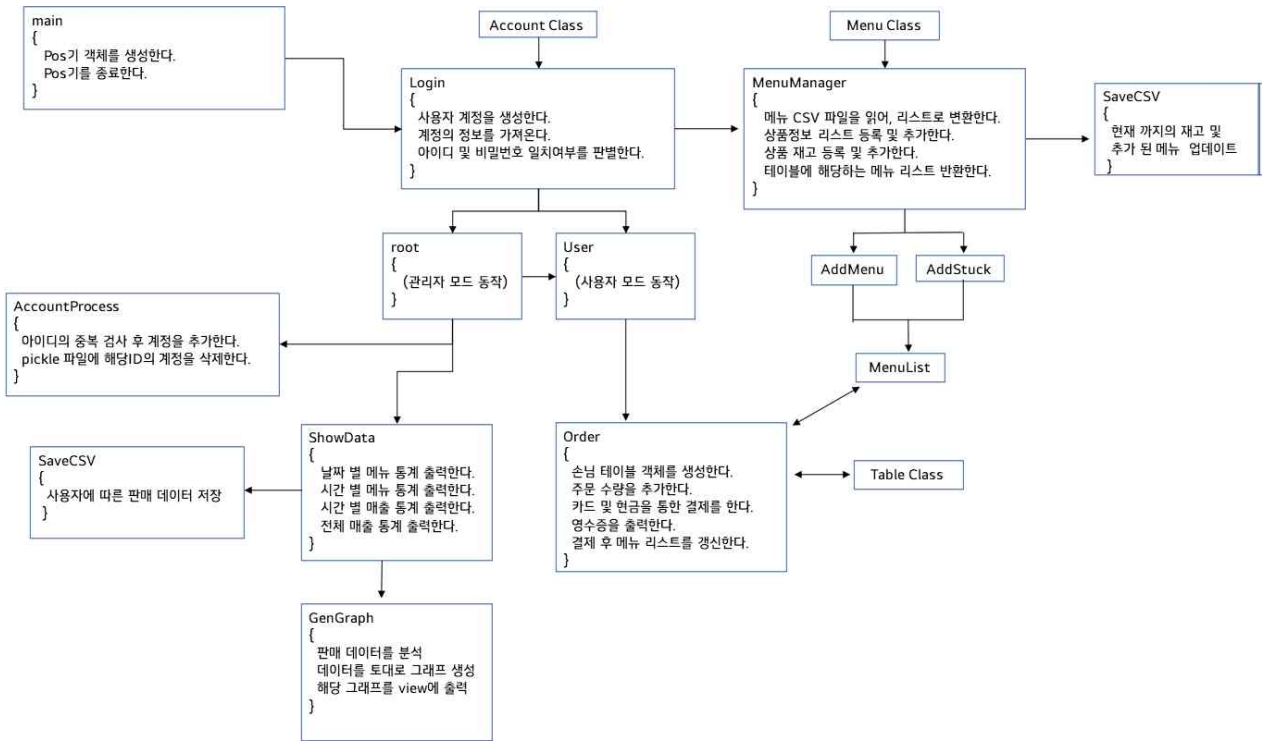
```
class Table:        // 각 테이블마다 존재하는 Table 객체, 0~7 : 매장 식사 8 : TakeOut
    [멤버 변수]
    // 메뉴 리스트 : 클래스 변수(공동 사용)
       - 전체 메뉴의 데이터(가격, 수량, 이름, 번호 등)가 저장된 리스트
    // 주문받은 리스트, 테이블 초기화 유무 변수 bool
    // 주문받은 총 가격,
    //주문 가능한 메뉴 리스트 (테이블에따라 다름)

    [테이블에 따른 메뉴 구분]
    -전체 메뉴리스트에서 매장내 식사 or 테이크 아웃에 따른 표시할 메뉴 항목 반환
    def GetMenuList(TableNumber):  // 테이블 번호를 인자로 받음.
        //  테이블 메뉴 리스트 생성
        if 테이블 번호가 8번 (테이크 아웃이라면)
            for Menu in Table._MenuList:
                    //Category 가 테이블인 경우만을 테이블 메뉴 리스트에 추가
       else  // 8번이 아니라면
              // Category 가 테이블인 경우를 제외하고 테이블 메뉴 리스트에 추가
       // 테이블 메뉴리스트를 반환

    [주문 추가]
    def AppendOrder(Menuindex, amount):
     // TalbelMenuList에 저장된 메뉴들의 순서와 Table Widget의 index순서는 동일하므로
     //  순서대로 메뉴의 이름, 수량, 가격, 을 저장
     //  수량과 가격을 곱하여 총 가격을 구함
     // 메뉴이름, 수량, 가격이 주문한 리스트 View 에 표시되어야 하므로 이를 하나로 리스트화 시킴
     // 해당 리스트를 orderList에 추가

      for Menu in Table._MenuList : //추가한 메뉴를 전체 메뉴 데이터 리스트에서 찾는다.
            if 메뉴이름과 일치하는 메뉴를 찾게 되면
                일치하는 메뉴의 데이터를 주문수량 만큼 -

    [주문 취소]
    def Reset():
        for orderMenu in self._orderList: // 주문한 메뉴 수만큼 반복
            for Menu in Table._MenuList:
                if // 메뉴 이름과 동일한 메뉴를 찾게 된다면
                  // 주문했던 수량만큼을 다시 +
      // 주문 넣어 뒀던 리스트 초기화
      // 결제할 총 가격도 0으로 초기화
```

■ [SalesList - View]

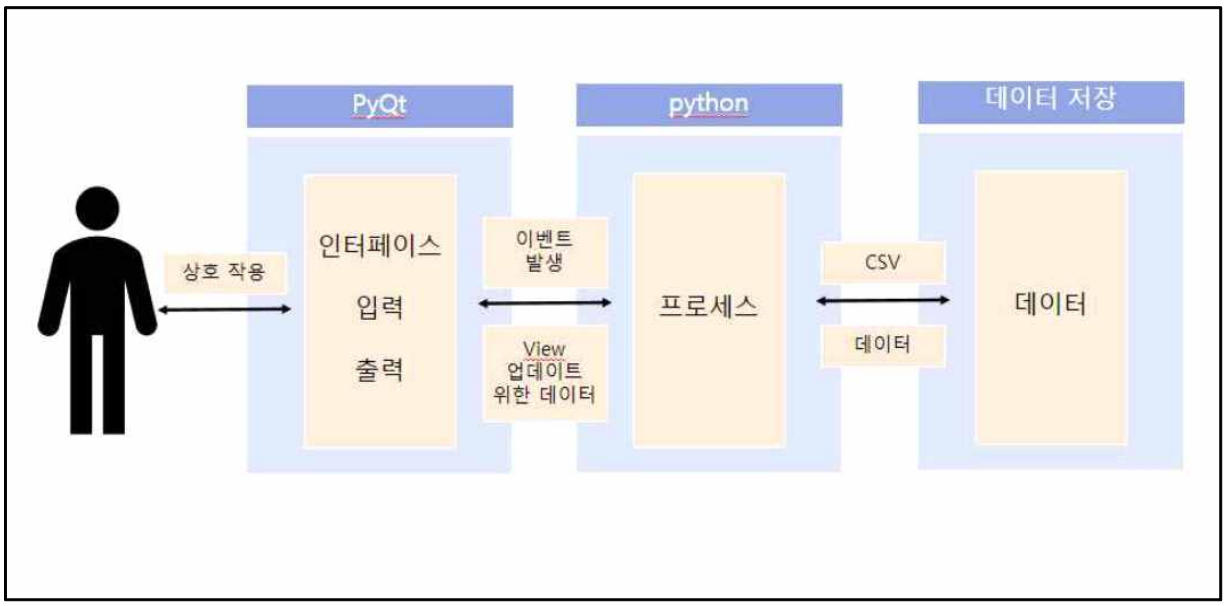
```
class SalesList(QDialog, dialog_class):
        if(카드 결제 버튼이 선택되었다면):
            // 현금 결제 버튼 비활성화
        else:
            // 카드 결제 버튼 비활성화
    [주문가능한 메뉴 리스트를 업데이트 – 테이블에 따라 다름]
    def ListUpdate(TableMenuList):
        for Menu in TableMenuList:
            // 추가할 메뉴 수 만큼 반복
            // View의 QsellList에 순서대로 해당 메뉴 이름을 추가

    [주문받은 메뉴를 주문 리스트에 업데이트]
    def OrderUpdate(TableOrderList):
        for Menu in TableOrderList:
          // 주문된 메뉴 수 만큼 반복
          // 테이블의 행렬을 고려하여 적절하게 해당 메뉴를 표에 입력
    [주문 초기화]
    def clear(self):
    // 주문한 표에 입력된 데이터를 전부 지움.
    // view에 표시된 총 가격을 0으로 초기화
    // view에 표시된 거스름 돈을 0으로 초기화

    [영수증 출력]
    def Areceipt(self, time, orderInfo, cgetmoney): # 메뉴명이 나오는 영수증
        if 카드가 활성화 되어있는 경우 -> 즉, 현금결제를 선택한 상황
            // 영수증 출력을 위한 위젯 생성 (메뉴명- TRUE, 'Cash' 메시지 전달)
            // 위젯 show (화면에 출력)
        else : (카드 결제를 선택)
            // 'Card' 메시지를 전달. 나머지는 동일
    def Breceipt(self, time, orderInfo ,cgetmoney): # 메뉴명 비공개되는 영수증
            // 비공개이므로 메뉴명 – False를 전달
class Label(QLabel):
    def paintEvent(self, e):
     // 영수증 이미지를 가져온다.
     // 해당 이미지에 전달 받은 정보를 통해 text 출력

class ReciptWidget(QWidget):
    def __init__(self,__dir,time,orderInfo,ismenu,pay,cgetmoney):
     // Label 생성, 생성 시 이미지에 출력 될 정보 전달
```

4.3 함수들 간의 호출 관계 또는 프로그램 구조

■ 프로그램 구조



■ 인터페이스 구조



■ 설계 구성 요소

[기존 설계 구성]

■ 주문 관리

- 테이블 별, 주문 별(테이크 아웃) 주문이 가능하도록 한다.
- 주문 현황판에 각 메뉴별 단가와 수량 기록이 되도록 한다.
- 고객의 요구에 따른 총액만 기록된 영수증 또는 메뉴명이 포함된 영수증을 따로 출력할 수 있도록 한다.
- 해당 영수증에는 현재 시각이 포함되도록 한다.
- 지불 수단에 따른 결제가 가능하도록 한다. 현금의 경우 고객의 지불 액에 따른 거스름돈을 반환한다 .

■ 매출 관리

- 일별/주별/월별/연도별 총 매출액, 각 기간별 메뉴별 매출액, 메뉴별 주문 수 등 시간을 고려한 통계 정보를 출력한다.

■ 재고 관리

- 판매 품목들과 메뉴에 들어가는 재료의 양을 정하고 품목 / 재료별 재고를 관리한다.
- csv 파일에 판매 품목의 정보를 저장하여 사용한다.
- 재고 상황을 항상 확인할 수 있도록 한다. 만약 재고가 없는 경우 재고가 보충되기 전까지 판매할 수 없도록 한다.

[실제 수행 결과 달라진 점]

■ 주문 관리

- 주문 현황판에 각 메뉴별 단가와 수량 기록이 되도록 한다. 주문현황판에 기록된 메뉴의 정보에 따라 받은금액, 거스름 돈이 계산되어 출력된다.
- 영수증을 이미지화하여 출력되도록 구현하였다.

■ 매출 관리

- 일별/주별/월별/연도별 총 매출액, 각 기간별 메뉴별 매출액, 메뉴별 주문 수 등 시간을 고려한 통계 정보를 출력한다.

■ 재고 관리

- 메뉴에 들어가는 부재료의 경우, 데이터의 관리가 너무 어려워져 부재료의 사용을 삭제하였다.

■ 설계 제한 요소

[기존 설계 제한]

■ 성능

- 메인 메뉴에서 각 기능선택에 의한 화면 전환까지 1초 이내로 전환 되도록 한다. - 정산 버튼을 누르면 anopos.csv 에 판매 목록이 추가되고, 해당 날짜와 시간 담당자의 이름이 담긴 제목으로 반영된 정산 csv 파일이 생성된다. ex.) 20211110_1043_김다은.csv

■ 신뢰성

- 기능 및 성능에서 오류 없이 동작하도록 신뢰도 100%를 목표로 설계한다.

■ 안정성/내구성

[안전성]

- 포스기 프로그램 내 데이터 파일의 입출력에 있어, PC 내 다른 파일 혹은 프로그램에 피해나 위협이 되지 않도록 한다.
- 포스기 프로그램 내 판매 내역과 같이 중요한 기능은 root 권한을 가진 사용자만이 접근할 수 있도록 한다.
- 관리자의 부주의 혹은 잘못된 조작으로 인하여, 재고 주문(음수 값 혹은 너무 큰 값)에 오류가 발생할 시, 이를 예외처리한다.

[내구성]

- 포스기의 장기적인 실행에도 타 응용프로그램과의 시스템 충돌이 발생하지 않도록 한다. 동적 메모리를 사용할 경우, 사용 후 즉시 해제를 통해 프로그램 내 메모리 누수가 발생하지 않게 설계한다.

[실제 수행 결과 달라진 점]

■ 성능

- 정산 시 포스기 사용자 및 판매한 내역을 전체 판매 CSV데이터에 추가한다. 해당 날짜의 판매 데이터의 경우 통계에서 확인할 수 있으므로 따로 csv파일로는 출력되지 않도록 하였다.

■ 안전성/내구성

[안전성]

- 예외처리로 아이디가 일치하는 경우, 금액 및 수량의 데이터가 숫자가 아닌 경우, 잔여수량보다 많은 양을 주문 할 경우, 수량이 아닌 다른 데이터를 수정하려고 했을 경우, 메뉴의 추가 시 중복되는 경우를 고려하였다

.

5. 구 현

5.1 클래스 및 함수 코드

■ genGraph - model

```
class GenGraph:
    def __init__(self):
        모든 함수에 공통으로 필요한 전처리 수행
    def TotalSales(self):
        판매 요약 리스트 return

    def GenProductSt(self, label, option):                          전체 물품 그래프
    def GenYearProductSt(self, label, year, option):                연도별 물품 그래프
    def GenMonthProductSt(self, label, year, month, option): :      달별 물품 그래프
    def GenDateProductSt(self, label, year, month, date, option):   일별 물품 그래프

    def GenUserSt(self, label , option):                            유저 전체 그래프
    def GenDateUserSt(self, label, year, month, date, option):      일별 유저 그래프
    def GenMonthUserSt(self, label, year, month, option):           월별 유저 그래프
    def GenYearUserSt(self, label, year, option):                   연도별 유저 그래프


    def GenTimeSt(self, label, option):
전체 시간 그래프
    def GenDateTimeSt(self, label, year, month, date, option):      일별 시간 그래프
    def GenMonthTimeSt(self, label, year, month, option):           달별 시간 그래프
    def GenYearTimeSt(self, label, year, option):                   연도별 시간 그래프
```

■ Controller

■ 이벤트 연결

```
## 제품별 통계 버튼 클릭
self._mw.product_st.clicked.connect(self.GenProductGraph)
# 사용자 통계 버튼 클릭
self._mw.user_st.clicked.connect(self.GenUserGraph)
# 시간별 통계 버튼 클릭
self._mw.time_st.clicked.connect(self.GenTimeGraph)

## 월별 연도별 일별 통계를 위해 캘린더 페이지 이벤트 추가
self._mw.seller_cal.currentPageChanged.connect(self.SellercalendarPageChanged)
self._mw.seller_cal.selectionChanged.connect(self.SellercalendarSelectionChanged)

self._mw.product_cal.currentPageChanged.connect(self.ProductcalendarPageChanged)
self._mw.product_cal.selectionChanged.connect(self.ProduccalendarSelectionChanged)

self._mw.time_cal.currentPageChanged.connect(self.TimecalendarPageChanged)
self._mw.time_cal.selectionChanged.connect(self.TimecalendarSelectionChanged)
```

■ 제품 그래프

```
def initGraph(self): # 초기화
def GenMainInfo(self): # 매출 요약

def GenProductGraph(self): # 전체 제품에 대해 (처음 버튼 클릭할시 생성되는 그래프)
def GenMonthProductGraph(self, year, month): # 월별 제품
def GenYearProductGraph(self, year): # 연도별 제품
def ProductcalendarPageChanged(self): # 연도별 , 월별에 대한 이벤트 처리
def ProductcalendarSelectionChanged(self): # date에 대한 이벤트 처리
```

■ 사용자 그래프

```
def GenUserGraph(self):
def GenMonthUserGraph(self, year, month):
def GenYearUserGraph(self, year):
def GenDateUserGraph(self, year, month, date):

def SellercalendarPageChanged(self):
def SellercalendarSelectionChanged(self):
```

■ 시간 그래프

```
def GenTimeGraph(self):
def GenMonthTimeGraph(self, year, month):
def GenYearTimeGraph(self, year):
def GenDateTimeGraph(self, year, month, date):

def TimecalendarPageChanged(self):
def TimecalendarSelectionChanged(self):
```

■ MenuCSVManger - model

```
class MenuCSVManager:
    def __init__(self):
        전체 메뉴데이터가 저장된 CSV 파일 주소 전처리 수행
    def loadCSV(self,MenuList):
        해당 주소로부터 CSV파일 데이터를 읽어온다.
        for i in range(len(data)):
            dlist = list(data.iloc[i])
                        재료1   재료2   재료3 (부재료)
            sublist = [dlist[4], dlist[5], dlist[6]]
                                코드    카테고리    메뉴명    가격   부재료    수량
            tempMenu = Menu(dlist[0], dlist[1], dlist[2], dlist[3], sublist, dlist[7])
            ㄴ순서에 따른 메뉴데이터를 가져와 메뉴객체를 생성한다.
            MenuList.append(tempMenu)
            ㄴ메뉴리스트에 생성한 메뉴를 추가한다.

    def AddNewMenu(self, MenuList, NewList):
    신메뉴 추가 -> 사용자가 새로운 메뉴 데이터를 입력 후 등록 버튼을 통해 이벤트를 발생시킨다.
    사용자가 View를 통해 입력한 데이터를 Controller에서 NewList형태로 전달하고 이를 사용한다.

        tempMenu = Menu(NewList[0], NewList[1], NewList[2], NewList[3], NewList[4],
        NewList[5])
        ㄴNewList에 저장된 새로운 메뉴데이터 값을 통해 메뉴객체를 생성한다.

         MenuList.append(tempMenu)
         ㄴ메뉴리스트에 생성한 신메뉴를 추가한다.
```

6. 결 론

6.1 결과

[요약]

기존 설계단계에서 개발하고자 하였던 목표를 대부분 달성하였다.

[분석]

MVC 패턴을 사용하여 각 구성요소를 독립시킴으로써 각 팀으로 하여금 맡은 부분의 개발에만 따로 집중할 수 있게 하여 개발의 효율성을 높였고, 개발 완료 후에도 유지보수성과 확장성을 보장할 수 있었다. 또한 명확하고 깔끔한 UI 인터페이스를 통해 사용자의 편의성을 향상시켰다.

[결론]

팀 프로그래밍을 통해서, 혼자 할 때보다 큰 시간 단축의 효과를 보았으며 코드의 분석을 통해 효율적인 코드 사용 및 올바른 흐름으로 코드를 작성할 수 있었다. 설계부터 시작하여 전 과정에 걸쳐 계속해서 의견공유와 토론과정을 통해 좀 더 좋은 방향으로 나아갈 수 있었다.

6.2 향후 개선 과제

■ 부재료 데이터의 추가 사용.

[요약]

부재료에 관한 부분을 초기 설계 당시, 부재료의 사용을 생각하였으나, 시간적 한계로 인하여 부재료에 관한 부분을 제외하고 메뉴데이터를 처리하는 방향으로 변경하였다.

[분석]

부재료를 사용하게 될 경우, 메뉴 판매에 따른 재고 관리에 있어서 처리하게 되는 데이터와 재고의 추가주문 오류 처리 등 고려해야 할 여러 문제가 발생하게 된다. 기존 부재료의 사용을 염두해두고 현재 부재료를 저장하는 부분을 개발과정에서 지우지 않고 남겨두었기에, 부재료의 사용을 추가함에 있어 기존 데이터 구조를 완전히 변화시키지는 않아도 될 것으로 분석된다.

[결론]

알고리즘 자체가 어려운 부분은 없기에, 언제든 시간만 충분하다면 충분히 해결할 수 있는 과제라고 생각된다. 따라서 시험 기간 이후 개선할 예정이다.

6.3 기대 효과

- 포스기 사용을 통해, 편리한 재고 확인 및 관리 가능.
- 시계열 데이터를 활용한 판매 및 매출 예측 통해 가게 운영의 효율성 증가.
- 매출 판매 통계를 바탕으로 효율적인 직원관리 가능.

7. 자체 평가서

7.1 자체 분석과 평가

김준용	짝 프로그래밍을 통해 얻은 이점이 많았다. 기존에는 설계나 구조 측면에서 별다른 고려사항 없이, 데이터를 추가하고 View로부터의 직접적인 값을 저장하는 등을 아무렇게나 사용했었다. 이러한 문제점은 2단계의 결과보고서에 "잘 되긴 하나 복잡하고 유지보수가 어렵다" 라고 작성할 만큼 해결해야 할 과제였는데, 이번 프로젝트에는 사용 용도에 따라 각 모듈을 분리하고 MVC 패턴의 역할에 맞게 설계하여 잘 짜인 프로그램을 제작할 수 있었다. 또한, 배운 지 오래된 파이썬은 무지한 수준이었는데, 함께한 팀원분께서 파이썬의 설치부터 학습 및 설계까지 도와주셔서 한걸음 씩 문제를 해결해 나갈 수 있었다. 개인 프로젝트라면 대충했을 부분도, 코드의 병합을 위해서는 결국 상호 간 이해가 되어야 하기에 어떻게 하면 좀 더 쉽고 명확할까를 고민하며 코드를 작성하게 되었다. 결과적으로 몇 가지 구현하지 못한 부분이 있지만 만족스럽다. 이번 프로젝트를 통해 여러모로 많이 배우고 성장할 수 있었다. 감사하다.
김다은	수행 과정에서 우리 팀은 상호 소통을 위해 주석이나 코드를 최대한 가독성 좋게 작성하려고 노력했다. 코드도 훨씬 깔끔해졌고 수행 결과도 ux 나 ui 다 발전된 것 같다. 문제 해결을 위해 내부의 model 도 잘 구성한 것 같다. 1, 2 단계를 할때 중복되는 코드를 함수로 묶는 것 리팩토링은 어느 정도 익숙해졌지만, 유지보수를 하기에는 클래스 간의 의존도가 너무 커 하나를 수정하려면 모든 클래스의 함수들을 수정해야 하는 등의 문제가 있었다. 후에 이러한 문제점을 인식하고 3단계는 조원분과 함께 디자인 패턴이나 클린 코드를 신경 써서 작성해야겠다고 생각했다. 그 중 MVC 패턴을 공부하여 적용하여 보았는데 그때 당시에는 이게 맞다라고 확신을 가졌던 부분도 코드를 짜다 보니 더 좋은 방법이 존재한다는 것을 깨달았다. 이렇게 계속 방식을 조금씩 개선해 나가는 과정이 존재하여 다행이라는 생각도 들었다. 혼자서 이 프로젝트를 진행하였다면 내가 생각하고 있는 것이 맞는지 어떻게 구현해야 할지 고민하는 것에 시간을 많이 들였을 것 같다. 하지만 팀원분의 적극적인 의견 제시와 조언을 주셔서 금방 해결 방안에 가까워질 수 있었다. 또 현재 주어진 범위에서 어떻게 하면 문제를 쉽고 빠르게 해결할 수 있는지 빠르게 접근하시는 걸 보고 같이 프로젝트를 진행하면서 많은 걸 배운 것 같다. 클린 코드를 위해서 오류처리 같은 것을 미리 처리해 두는 것 MVC 패턴에 대한 이해가 높아진 것이 제일 기억에 남는다.

7.2 팀원 역할 및 기여도

■ 역할

김준용	김다은
로그인 및 회원가입 구축	판매 구현
메인 화면 구축	GUI 구성 및 디자인
예외 처리 및 사용자 유즈 케이스 고려 설계	CSV 파일 생성 및 판매 통계 구현
영업 정보 구현	전체 코드 병합

■ 전체 기여도

김준용	김다은
50	50

8. 부록

■ 진행 일정

(수행 기간 : 2021년 11월 13일 ~ 2021년 12월 10일)					
구분 / 수행 내용	1 주	2 주	3 주	4 주	주
POS 회원 관리 구현					
POS 테이블 버튼 다이얼로그 로직 구현 및 테이블 뷰 변경과 정산 기능					
판매량 통계 구현					
재고 관리 및 상품 추가 구현					
영수증 출력과 판매 구현					

■ 회의록

아노포스 회의록

날짜	주제	작성자
2021-11-23	구현 계획	김다은
2021-11-25	구현 상황 정리	김준용
2021-11-26	진행 상황 및 수정 사항	김다은
2021-12-01	영수증 처리 및 최종 점검	김준용

구현 계획

1. 판매 등록 신메뉴 추가 및 세부적인 오류 잡기 (재고 관리에 대해 조금 추가)
2. 판매 완료가 되었을 때 나온 CSV 파일 처리 후 통계 제공
3. 최종 점검 필요

판매 내역은 해당 당일의 내역이나 연도별로 보여준 뒤 매출 요약은 통계를 고려하여
어떻게 하면 매출 관리에 핵심적인 정보를 한눈에 보일 것인가에 대해 고려

시각화를 위해 matplotlib 을 사용

초기화에 대하여 : deep copy를 활용하여 원본데이터를 가지고 다시 original 데이터로 접근하여
초기화시켜주는 것을 계획하였으나 원하는 대로 작동하지 않음 (보류)

재고 추가에 대한 것이니 초기화 버튼을 누르면 재고만 -로 값을 처리해 주어 다시 재고에 추가되는
방식에 대해 제안

-CSV 파일 처리에 대한 고민

- 영업등록 Save는 단순 CSV 데이터 업데이트 마감정산은 정산 내역 김다은_211123.CSV 로 한다.
- 마감정산을 다루는 파일 하나를 만들어 계속 추가한다.

결제가 되지 않은 테이블은 값을 날림, CSV 파일에 입력하지 않는 것으로 구현
판매는 다이얼로그에 결제가 진행되면 시간대 정보와 함께 입력

save를 눌러 1~8 테이블 전체 초기화 시, 현재 코드상 판매 다이얼로그로 상품이 입력되면 결제하지
않아도 주문 수량만큼 수량이 차감되게 되는데 초기화 시, 주문 수량 다시 되돌려주는 게 사실상 필수
구현되어야 하는가에 대해 고려

아직 결제하지 않아도, 일단 판매 다이얼로그에 물품이 등록되는 순간 해당 물품은 그 시간부로 판매된
거로 처리되도록 하는가에 대해 논의

구현 상황 정리

1. 판매기능 구현 완료

2. 통계 자료를 위한 데이터
계절에 대한 처리
어떤 판매자가 제일 판매량이 많은지
어떤 시간대에 제일 많은지
어떤 월에 어떤 물품이 제일 많은지
에 대해 데이터 생성 코드
Mat plot로 이미지 띄우기
GUI 연결

부재료를 없애고 계절에 대한 정보를 추가하는 것에 대해 논의

계절 별로 그리고 어떤 때 가장 사용이 많은가를 통계로 보여줌
미리 준비된 데이터 없이 (어떤 계절에 어떤 물품에 대한 판매량이 증가한다 이런 부분) 그냥 matplot 에 대한 것만 준비되어있다면 통계 부분이 제대로 구현되어 있는지에 대한 검증이 제대로 이루어지지 않으리라 판단.

Dataset 만드는 것도 추가로 요소마다 가중치 다르게 해서 여러 개의 데이터 셋을 만들 수 있도록 코드를 작성하는 방향을 제시

사용 횟수를 수작업으로 늘리는 것은 시간상 불가능하다고 판단.
매출 분석을 한다는 게 좀 애매하게 되므로 검증을 위해 데이터셋을 생성하기로 결정
포스기 사용자가 계절, 시간에 맞춰 아마 이런 걸 샀을 것이다 가정하고 만들어지는 데이터일 뿐이니까 괜찮으리라 판단

해결 문제 제시

1. 매출이 가장 좋은 달 그달에 얼마를 벌었으며 제일 높은 판매 물품은 무엇이었는지 요약할 수 있도록

2. 가장 많은 제품을 판매한 사람은 누구인지 알 수 있도록

3. 고객의 제품 구매 가능성을 극대화하기 위해 시간별 그래프를 확인할 수 있도록

함께 판매되는 제품과 가장 많이 팔린 제품에 대해 고려

진행 상황 및 수정 사항

self.__menu = ["덩굴", "골렘 왕", "생명수", "마법 버섯", "기술의 역사", "녹슨 대검", "맹독 젤리", "용기", "골렘의 땀", "요정 가루", "금가루"]
        # 골렘의 땀을 여름철 특선 메뉴로 index 8
        # 마법 버섯을 겨울철 특선 메뉴로 index 3
        # 요정 가루를 특정 날의 특선 메뉴로 index 9
이러한 방법으로 가중치를 두어 특정 달의 해당 메뉴 판매 가중치를 높임

여름 계절 메뉴의 경우 겨울에는 판매하지 않도록 조건문을 통하여 통제

판매량 예측 시도를 위해 rnn 과 lstm 에 대해 고려

gui 에 그래프를 띄우는 부분에 대한 오류
qpixel 에서 pixmap 에 이미지를 덮어씌우는 방법을 선택했으나 문제가 발생

1. 판매
- 초기화

2. 영업관리메뉴
-메뉴 재고 수정
(QwidegetTable이 textEdit처럼 직접 수정이 가능해서, 직접수정 후 엔터로 등록)
(숫자만 입력 가능, 재고 제외 다른 거 건드릴 시 오류처리)
-신메뉴 등록

영수증 처리 및 최종 점검

```
In [7]: a
Out[7]: 3

In [8]: a = [1,2,3]

In [9]: b = a

In [10]: b.append(1)

In [11]: print(a)
[1, 2, 3, 1]

In [12]: B = 1

In [13]: A = B

In [14]: id(A)
Out[14]: 140717273786160

In [15]: id(B)
Out[15]: 140717273786160

In [16]: A = 3

In [17]: B
Out[17]: 1

In [18]: A is B
Out[18]: False

In [19]: A == B
Out[19]: False

In [20]: id(A)
Out[20]: 140717273786224

In [21]: id(B)
Out[21]: 140717273786160
```

- 파이썬 변수에 대한 논의

a = 3을 해주었을 때 3 도 하나의 객체이므로
a = b를 하였을 때 이 둘의 아이디는 같으나
a = 3을 해주게 되면 a 에 3이라는 정수의 객체의 아이디를 참조하게 되니 그림과 같은 결과가 나옴

리스트에 append() 수행 해당하는 객체에 숫자를 더하게 되는 것이므로 객체 자체의 아이디는 변하지 않음

참조된 것이 아니라 현재 변수에 참조된 리스트에 단순히 값이 추가된 것이기 때문

1이라는 객체를 a b에 넣어놓으면 둘 다 같아짐 이에 대해서 고민

1. 메인 컨트롤러
2. salesListviw.py
3. table.py
4. 이미지 (현금결제, 카드결제)

추가 구현 및 수정 완료

메뉴 데이터 입력 영어로 변환 -> 그래프 출력 시 문제

```
for TableNum in range(9):
        self._origTableList.append(Table(self._menuManager.MenuList, TableNum)) #전체
self._TableList = copy.deepcopy(self._origTableList)

# error
```

메뉴 리스트를 저장해둔 것을 받아서 반환 해주는 부분
메뉴 매니저에 함수 이름이 잘못 입력되어있는데 돌아가는 오류 발생
pycache 에 저장되어있는 기록 때문에 발생한 문제라고 판단
pycache 전부 삭제 후 재실행 -> 정상 동작

그래프 show에서 메뉴 데이터가 고루 들어가지 못한 부분은 그래프 데이터가 보여지지 않음
이에 대해서 처리는 따로 할 필요 없이 그래프를 띄우지 않는 것으로 정함

영수증은 jpeg나 png에서 pixmap으로 그리는 것을 통해 구현

9. 참고 문헌

- 공공 데이터 및 제공되는 판매 데이터 참고
- 파이썬 쿡북 및 책
- PyQt 공식 문서
- MatPlotlib 공식 문서
- python 공식 문서
- 디자인 패턴 책 (mvc 패턴)