Design Specification

Wizard of Order

제출일	2018.05.06	**그룹**	Group 6
과목	소프트웨어공학개론	**담당교수**	이은석 교수님
이름	김윤성	**학번**	2015312606
이름	박종원	**학번**	2014314516
이름	이지훈	**학번**	2016314088
이름	이혜리	**학번**	2014314853
이름	조건봉	**학번**	2016313167

Contents

1. Preface

1.1. Objective

이 섹션에서는 이 문서의 대상 독자들과 문서의 전체 구조, 문서 각 장의 내용에 대해 서술한다. 또한, Version History를 제시하여 버전 관리 정책, 버전 변경 기록, 문서의 변경 사항들을 기술한다.

1.2. Readership

해당 설계 명세서는 목표 시스템의 개발 및 유지 보수를 담당하는 모든 사람들을 대상 독자로 하여 작성되었다. 실제로 개발을 담당하는 소프트웨어 엔지니어, 시스템을 설계하는 시스템 아키텍처와 고객 기술 지원을 위한 서비스 팀 모두가 본 문서의 독자로 설정되어 있다.

1.3. Document structure

A. Preface

Preface에서는 본 문서의 대상 독자들을 제시하고, 문서의 전체 구조와 각 장의 역할 및 그 내용을 제시한다. 또한, 버전 관리 정책과 이에 따른 Version History를 표 형태로 기록하고, 추가적인 문서의 변경 사항을 서술한다.

B. Introduction

Introduction 에서는 시스템의 설계에 사용한 UML(Unified Modeling Language)의 다양한 다이어그램 작성을 위해 사용한 개발 툴을 소개한다.

C. System Architecture

System Architecture에서는 목표 시스템에 대한 높은 수준의 개요와 시스템 기능의 전체적 분포를 보여준다. 이 과정에서 재사용되는 컴포넌트는 강조하여 표현하였다. 전체 시스템의 구조는 Block Diagram을 통하여 도식화 하였으며, 서브 시스템들의 관계와 실제 배포 형태를 Package Diagram 및 Deployment Diagram을 사용하여 표현하였다.

D. Order create system

Order system은 식당을 방문하는 고객과 관련된 시스템으로, Select Menu, Confirm Order, Record Order, Final Check와 같은 4개의 sub-system으로 나뉜다. 이러한 Order system에 대하여 Class diagram, Sequence diagram과 State Diagram을 통해 User Management System의 구조를 표현하고 설명한다.

E. Restaurant system

Restaurant system은 고객 주문을 전달받는 식당 직원 입장과 관련된다. Restaurant system은 Kitchen sub-system, Hall sub-system, Order History sub-system과 같은 3가지 서브시스템으로 나뉜다. 이러한 restaurant system에 대하여 Class Diagram, Sequence Diagram, State Diagram 을 통하여 세부 구성 컴포넌트를 표현하고 이에 대하여 설명한다.

F. Protocol Design

Protocol Design 에서는 서브 시스템들이 상호 통신하기 위하여 필수적으로 준수해야 하는 프로토콜에 대하여 서술한다. 통신하는 메시지의 형식과 용도, 의미를 설명한다.

G. Database Design

Database Design 에서는 요구사항 명세서에서 기술하였던 요구사항을 기반으로 한 데이터베이스를 설계하고 이에 대하여 설명한다. 요구사항에 기반한 데이터베이스의 ER Diagram을 제시하고, 이에 대한 Relation Schema를 서술한다. Normalization 과정을 통하여 데이터베이스 간에 존재할 수 있는 중복을 방지하고, SQL DDL을 작성한다.

H. Testing Plan

Testing Plan 에서는 요구사항 명세서에서의 관리자 및 사용자 시나리오를 바탕으로 한 전체 시스템의 테스트 케이스를 설명하고, 관련 테스트를 수행하기 위한 테스팅 정책을 제시한다.

I. Development Environment

Development Environment 에서는 실제 시스템 개발을 위하여 필요한 개발 환경과 코딩 규칙을 설명한다. 개발 과정에서의 버전 관리 도구를 제시하고, 프로그래밍 과정에서 기반이 되는 규칙들에 대하여 서술한다.

J. Index

Index에서는 문서의 인덱스들을 정리한다. 알파벳 순서의 단어 인덱스를 기반으로, 다이어그램

인덱스 및 기능 인덱스를 포함한다.

1.4. Version of the Document

A. Version Format

버전 번호는 major.minor[.maintenance] 형태로 표현하며, 본 문서의 버전은 0.1부터 시작한다.

B. Version Management Policy

본 명세서가 수정될 때마다 버전을 업데이트한다. 다만 변경 간의 간격이 1 시간 이내 일 때에는 버전 번호를 추가로 업데이트하는 대신 하나의 업데이트로 간주한다. 이미 완성된 부분에 대한 변경의 경우 minor number를 변경하며, 새로운 부분을 추가하거나 문서의 구성이 예전에 비해 괄목할 변화가 있을 경우 major number를 변경한다. 이미 작성한 부분에 대해서 오타를 수정하거나, 문서의 구조를 변경할 경우 maintenance number를 추가하거나 변경한다.

C. Version Update History

Version Modified date Explanation

Table 1. Version update history

0.1	**2015-05-10**	**Preface, Introduction, System Architecture 작성**
1.0	2015-05-13	Sub-system 작성, 문서의 초안과 표지 작성
1.1	2015-05-17	System Architecture 하위 항목 수정
2.1	2015-05-19	다이어그램 추가 및 Glossary, Database design 작성
3.0	2015-05-20	Protocol Design, Testing Plan, Development Environment, Index 작성, 캡션 추가 및 레이아웃 조정

2. Introduction

2.1. Objective

Introduction 에서는 시스템의 설계에 사용한 UML(Unified Modeling Language)의 다양한 다이어그램 작성을 위해 사용한 개발 툴을 소개한다.

2.2. Applied Diagram

A. UML

Figure 1. UML logo

UML(Unified Modeling Language, 통합 모델링 언어)는 소프트웨어 공학에서 사용되는 범용 모델링 언어이다. UML은 객체 지향 소프트웨어 시스템을 설계할 때 시스템의 구조를 표준화된 시각적 모델로 명시하여 개발자를 포함한 이해당사자간 의사소통을 원활히 하는데 목적이 있다. UML은 이를 처음 고안한 OMG(Object Management Group, 객체 관리 그룹)에서 관리하고 있으며 소프트웨어 산업의 실질적인 표준 역할을 하고 있다. 객체지향의 3인방이라 불리는 그래디 부치(Grady Booch), 제임스 럼버(James Rumbaugh), 이바 야콥슨(Ivar Jacobson)이 고안하고 설계한 UML의 초안은 1997년 UML컨소시엄에서 HP, Microsoft, Oracle등 산업의 중추적인 역할을 하는 기업들이 중심이 되어 버전 1.0을 만들어 내게 되고 그 이후 꾸준히 산업의 변화에 따라 OMG에서 체계적으로 관리, 개선되어 왔다. 현재는 총 13개의 UML이 있으며, 우리의 프로젝트에서는 4개의 UML을 사용하게 된다.[1]

B. Class Diagram

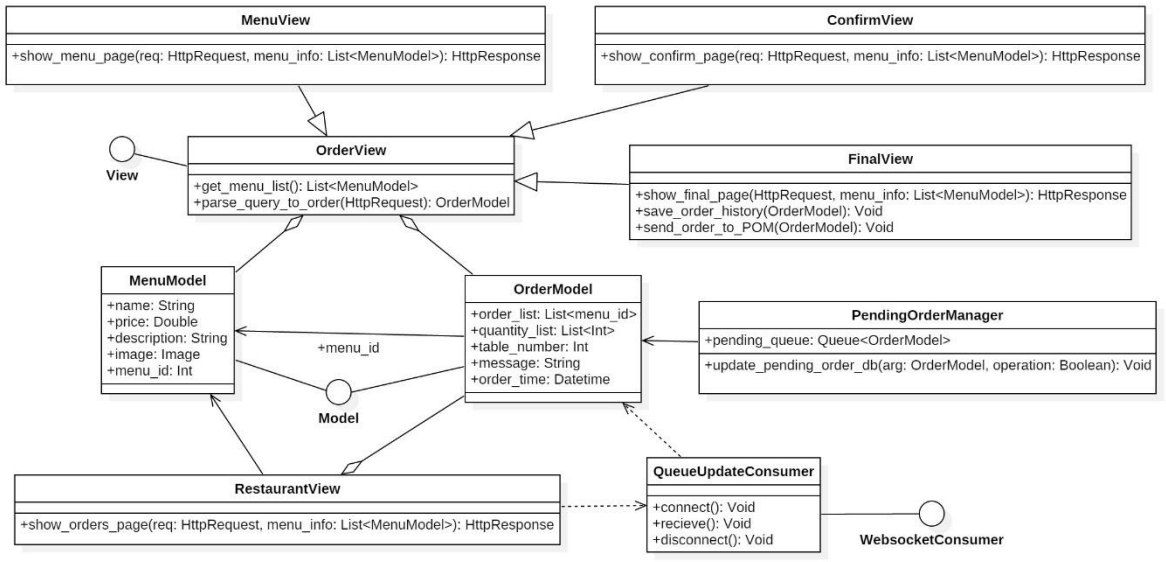

Diagram 1. Class diagram example

Class diagram은 객체지향 설계에서 널리 사용되는 diagram으로 시스템의 각 클래스간 정적인 논리적구조를 나타낸 것이다. 한 클래스는 직사각형 모양으로 나타내어지며, 그 안에 클래스 이름, 속성(attribute), 그리고 함수(method)들이 들어가게 된다. 각 클래스의 상속 등의 관계를 표현할 수 있다. Implementation 단계에서 직접적인 코드 작성에 필수적이다. [2]

C. Sequence Diagram

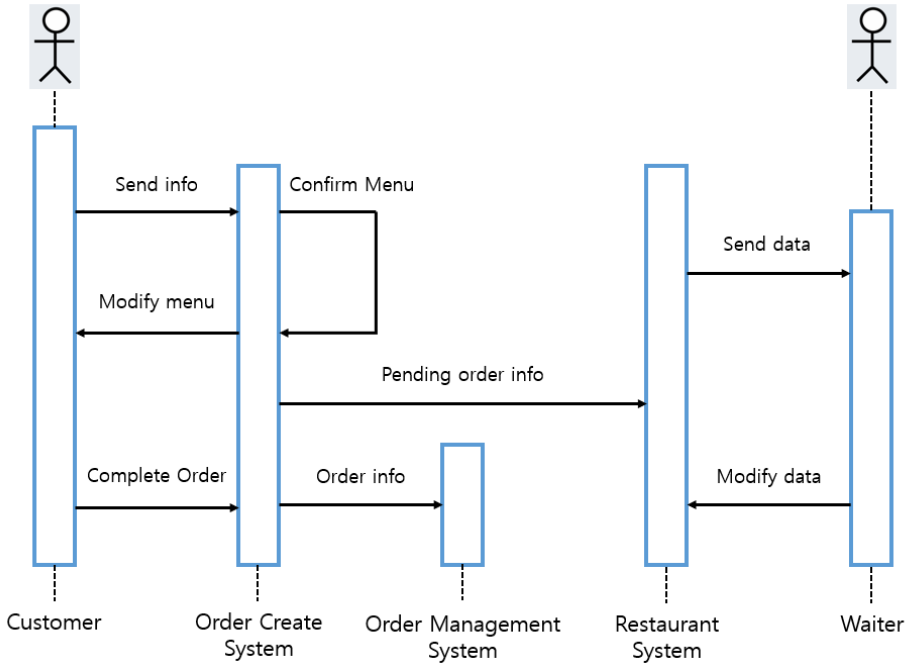

Diagram 2. Sequence diagram example

Sequence diagram은 시간 순서로 시스템, 객체, 또는 클래스들의 상호 작용을 보여준다. 시나리오와 관련된 객체와 클래스와 시나리오의 기능 수행에 필요한 객체 간에 교환되는 메시지의 순서를 설명한다. 시퀀스 다이어그램은 일반적으로 개발 중인 시스템의 logical perspective에서 use case들을 실현하는 것과 연관되어 있다. Sequence diagram은 event diagram 또는 event scenario라고도 한다. Sequence diagram은 병렬 수직선 또는 직사각형(라이프 라인)으로 동시에 존재하는 다양한 프로세스, 시스템 또는 개체와 수평 화살표로, 메시지가 발생하는 순서대로 서로 교환되는 메시지를 보여준다. 이를 통해 시각적인 방식으로 단순 runtime 시나리오를 지정할 수 있다.[3]

D. State Diagram

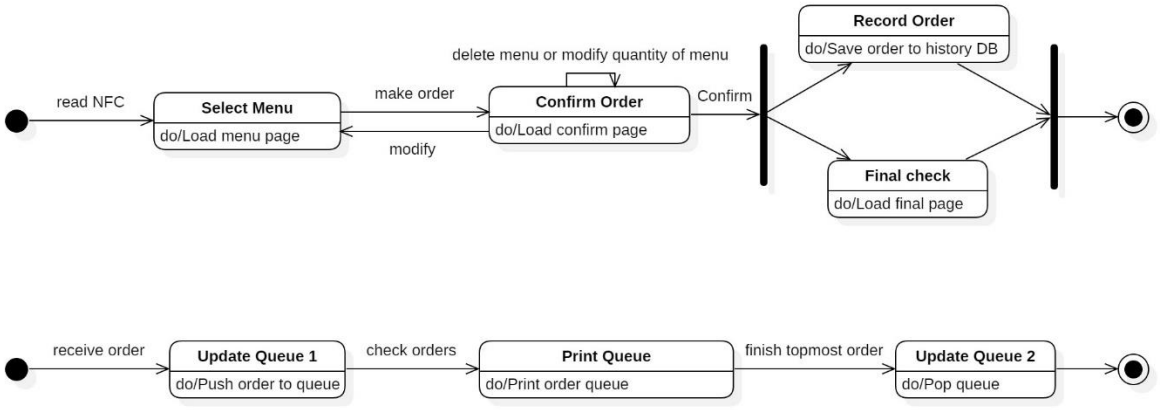

Diagram 3. State diagram example

State diagram은 시스템의 동작을 설명하기 위해 컴퓨터 과학 및 관련 필드에 사용되는 다이어그램의 일종이다. 프로그램의 상태와 행동을 표현하기 적적하여 UML에 채택되어 사용되고 있다. State diagram은 시스템의 동작에 대한 추상적인 설명을 제공하는 데 사용되는데 이 동작은 분석되며 하나 이상의 가능한 state에서 발생할 수 있는 일련의 이벤트로 표시된다. 각 다이어그램은 일반적으로 단일 클래스의 객체를 나타내며 시스템을 통해 해당 객체의 여러 상태를 추적하게 된다.[4]

E. ER Diagram

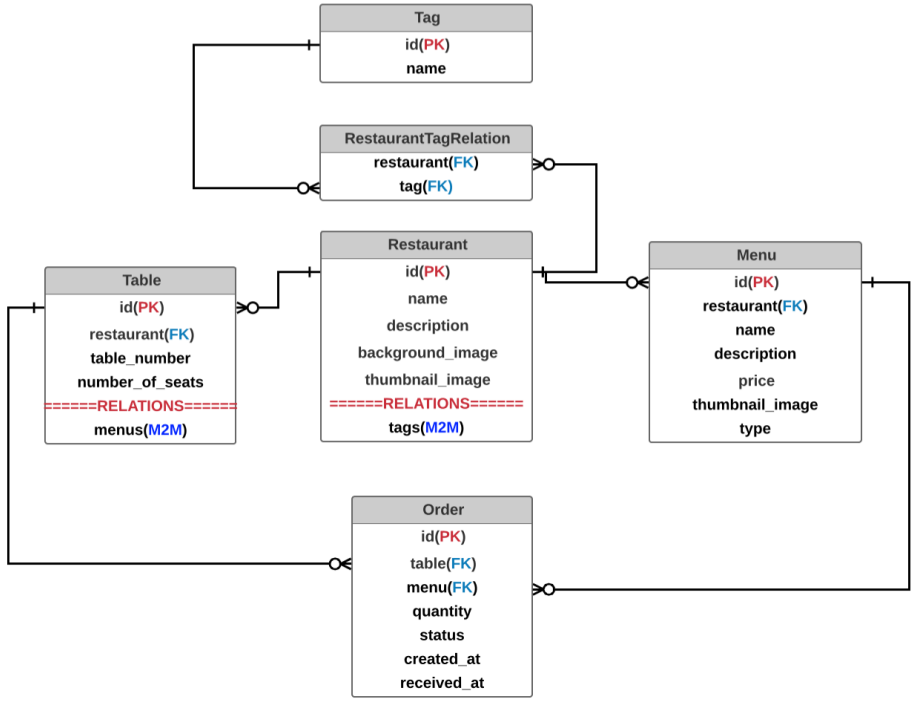

Diagram 4. ER diagram example

데이터 모델링 분야에서 ER diagram(Entity-Relationship diagram, 개체-관계 모델)이란 구조화된 데이터에 대한 일련의 표현이다. 구조화된 데이터를 저장하기 위해 데이터베이스를 쓴다. 이 데이터의 구조 및 그에 수반한 제약 조건들은 다양한 기법에 의해 설계될 수 있다. 그 기법 중 하나가 개체-관계 모델링이다. 데이터 모델링 과정은 데이터 모델을 그림으로 표현하기 위해 표시법을 필요로 한다. ERD는 개념적 데이터 모델 혹은 semantic 데이터 모델의 한 타입이다.[5]

2.3. Applied Tool

A. Online Diagram Software

- ER diagram: Lucidchart(https://www.lucidchart.com/)
- State diagram, class diagram : StarUML(http://staruml.io/)
- FlowChart: Sketch (https://www.sketchapp.com/)

B. Front-End: VSCode

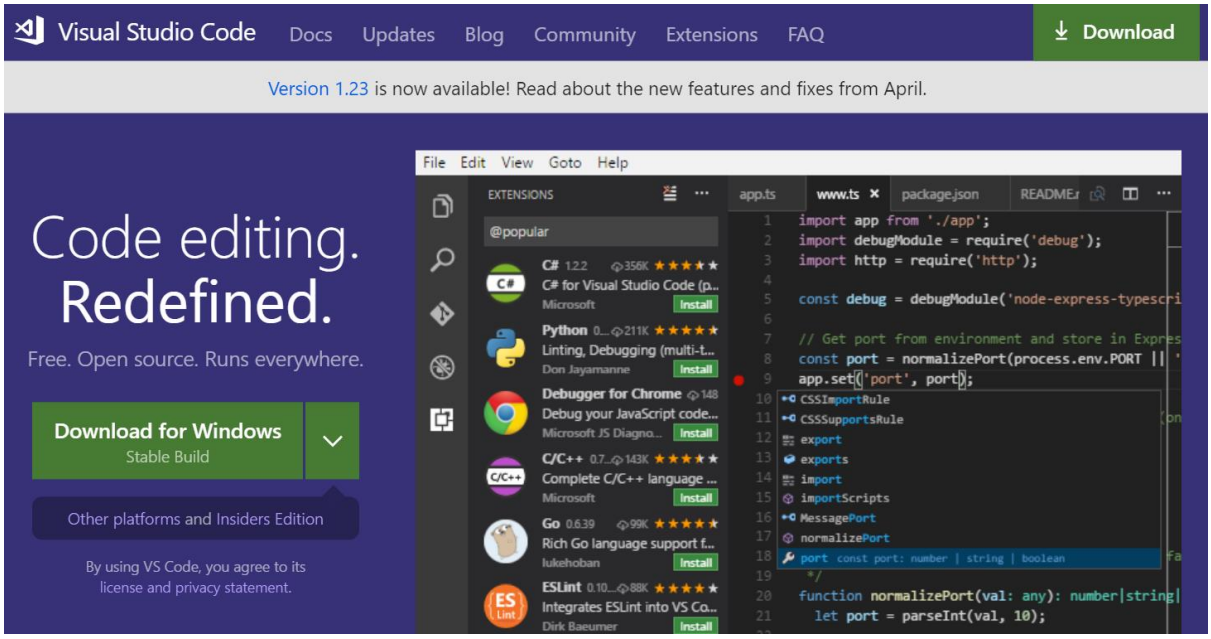

Figure 2. VSCode

- 라이브러리/프레임워크: React

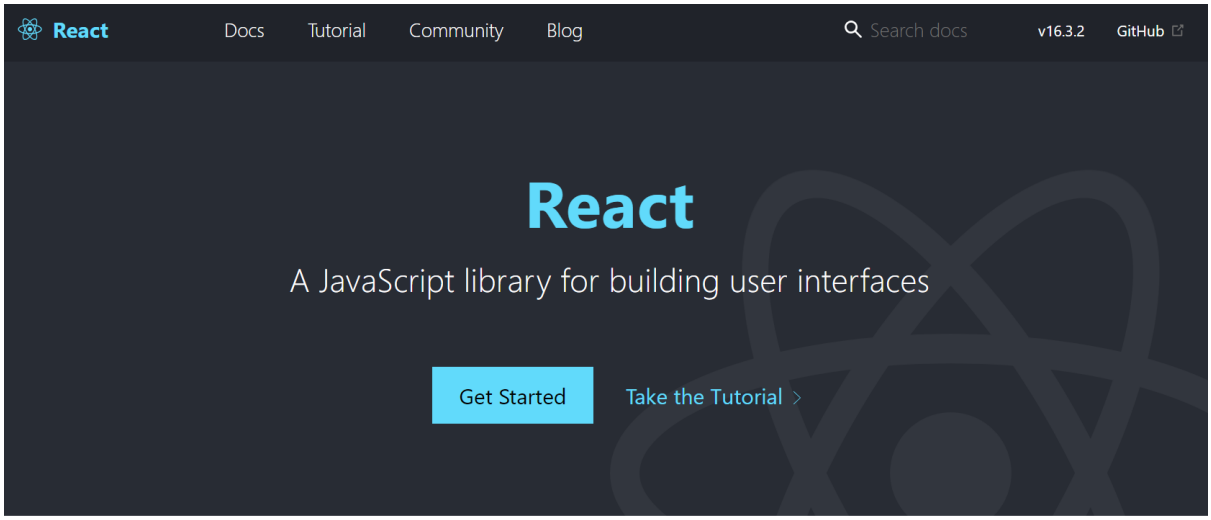

Figure 3. React

- 패키지 매니저 : yarn

C. Back-End: Pycharm

Figure 4. Pycharm

- 라이브러리/프레임워크: Django
- 패키지 매니저 : pip

D. 서버: AWS 사용

Figure 5. AWS

- 인스턴스: AWS EC2 사용

- 데이터베이스: AWS RDS 사용(postgresql)
- 정적 파일 serve: AWS S3 사용

2.4. Project Scope

Wizard of Order(오더의 마법사) 시스템은 기존 식당 주문 시스템의 단점을 보완하여 손님들의 이동을 줄이고, 직원들의 수고를 덜어주기 위해 고안된 모바일 웹 시스템이다. 오더의 마법사 시스템은 크게 두가지 시스템으로 나뉠 수 있다.

첫째로, Order system이 있다. Order system은 식당을 방문하는 고객과 관련된 시스템으로, Select Menu, Confirm Order, Record Order, Final Check와 같은 4개의 sub-system으로 나뉜다. 식당을 방문한 고객은 NFC태그를 통해 모바일 웹에 접속하고, 이때 Select Menu sub-system으로 들어가게 된다. Select Menu sub-system은 고객의 모바일 창에 주문가능한 메뉴를 보여주는 시스템으로, 고객이 메뉴를 선택하면 Confirm Order sub-system으로 넘어가게 된다. Confirm Order sub-system은 고객이 선택한 메뉴에 대하여 수량과 더불어 부가적인 주문사항을 묻는다. 선택한 메뉴 이외의 다른 추가적인 주문이 필요하거나 수정사항이 필요한 경우에는 Select Menu sub-system으로 돌아가거나 Confirm Order sub-system을 재가동 할 수 있다. 고객이 원하는 메뉴와 수량, 부가적인 주문까지 입력을 받고 나면 Final check sub-system으로 넘어간다. Final check sub-system은 요청한 주문사항에 대하여 확인작업을 하는 단계이다. 이와 동시에 Record Order sub-system을 통해 해당 주문을 Order history database에 입력한다. 이때 입력된 데이터는 추후에 통계적인 분석을 위해 사용된다. Order system의 구조는 다음과 같다.

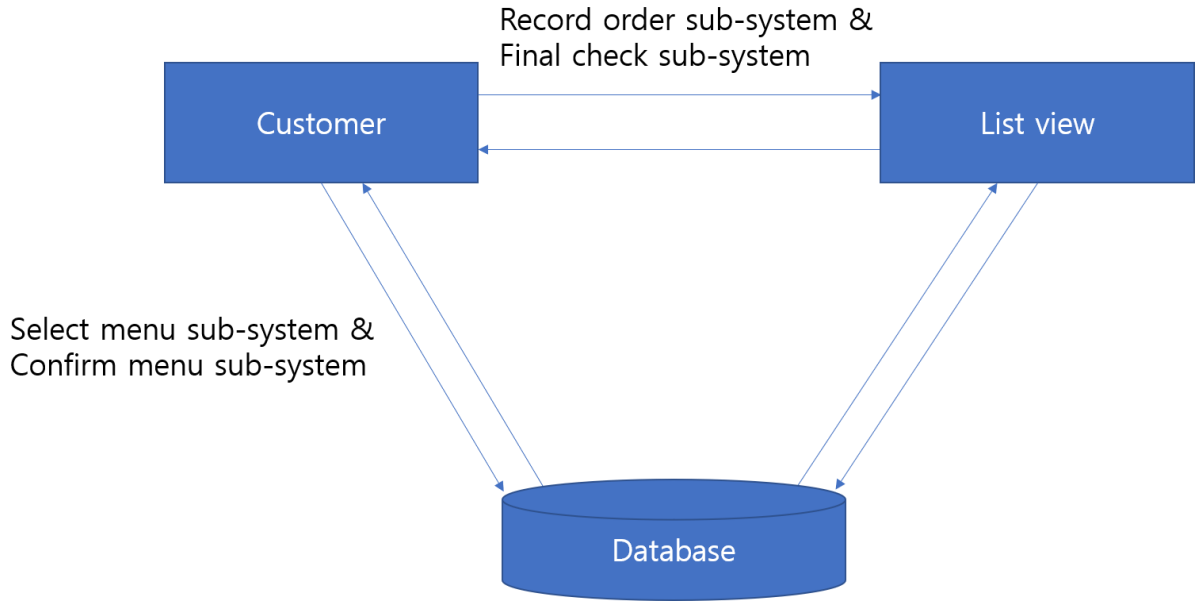

Figure 6. Order system의 구조

다음으로, 오더의 마법사를 구성하는 또 다른 시스템으로 Restaurant system을 들 수 있다. Restaurant system은 고객 주문을 전달받는 식당 직원 입장과 관련된다. Restaurant system은 Kitchen sub-system, Hall sub-system, Order History sub-system과 같은 3가지 서브시스템으로 나뉜다. Kitchen sub-system은 조리자가 현재 주문의 진행사항을 확인할 수 있는 페이지를 반환해주는 시스템이다. Hall sub-system은 홀에서 서빙을 하는 담당자가 현재 주문 및 조리의 진행사항을 확인할 수 있는 페이지를 반환해주는 시스템이다. Order History sub-system은 입력 받은 주문을 Order History Database에 저장해주는 시스템이다. Restaurant system의 구조는 다음과 같다.

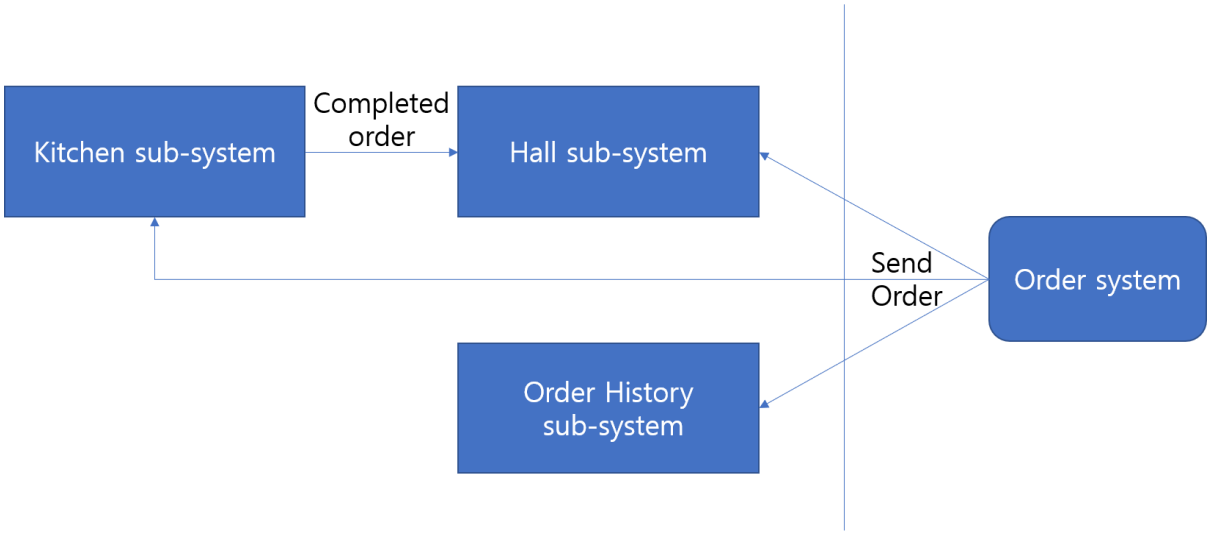

Figure 7. Restaurant system의 구조

3. System Architecture

3.1. Objective

System Architecture에서는 목표 시스템에 대한 높은 수준의 개요와 시스템 기능의 전체적 분포를 보여준다. 이 과정에서 재사용되는 컴포넌트는 강조하여 표현하였다. 전체 시스템의 구조는 Block Diagram을 통하여 도식화 하였으며, 서브 시스템들의 관계와 실제 배포 형태를 Package Diagram 및 Deployment Diagram을 사용하여 표현하였다.

3.2. System Organization

A. Order create system

Order create system은 식당의 손님이 이용하는 시스템으로, NFC태그를 스마트폰을 스캔하여 menu page를 불러오는 것부터 메뉴들을 선택 후 confirm page로 이동하는 것, 마지막으로 주문을 완료하여 final page를 불러오는 것 까지를 아우른다. NFC에서 URL configuration을 하여 menu page를 불러올 때 Menu view가 Food DB와 Menu template을 참조한다. Menu page에서 메뉴 선택 완료 후 장바구니 담기 버튼을 클릭하면 Confirm view가 Confirm template을 참조한다. 마지막으로 Submit order를 하게 되면 Final view가 Final template을 참조하여 Final page를 불러오게 된다.

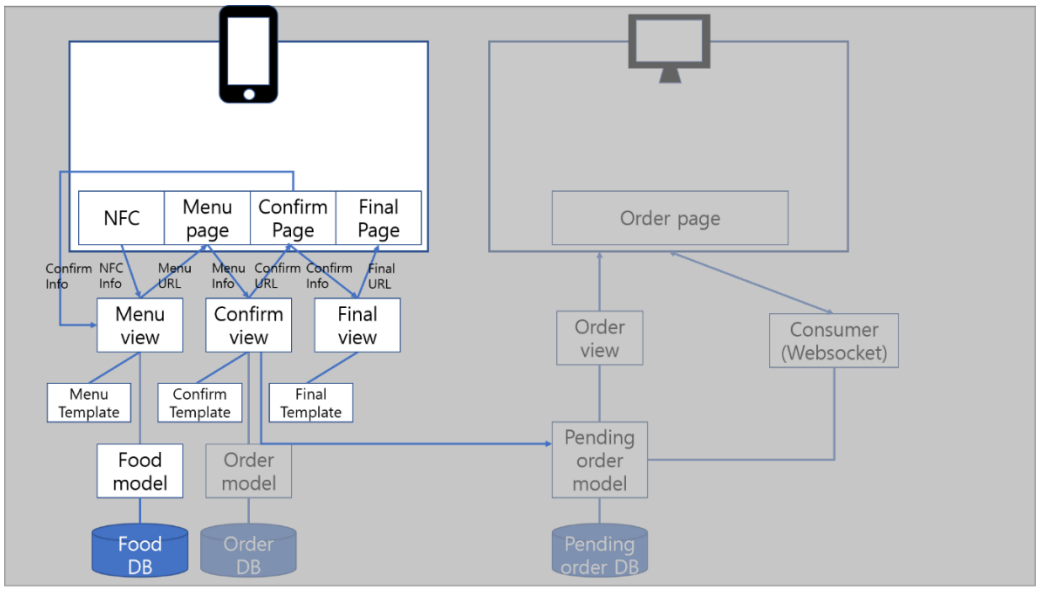

Diagram 5. Order create system

B. Restaurant System

Restaurant System은 매장의 주문 관리와 관련된 시스템으로, 주방에서 현재 조리 대기 중인 주문을 관리하는 Kitchen Sub-system, 각 테이블의 주문 상황과 현재 서빙 해야 하는 음식을 관리하는 Hall Sub-system, 그리고 주문 역사(?)를 확인하는 Order History Sub-system으로 나뉜다.

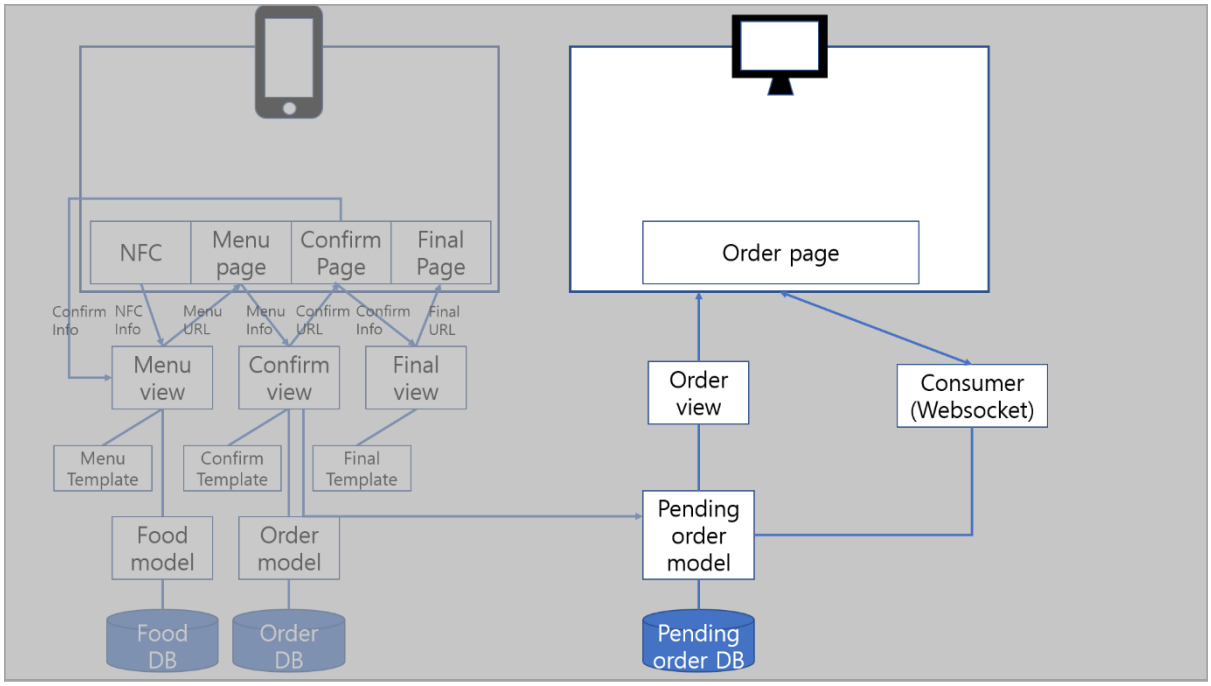

Diagram 6. Restaurant system

4. Order create system

4.1. Objective

Order system은 식당을 방문하는 고객과 관련된 시스템으로, Select Menu, Confirm Order, Record Order, Final Check와 같은 4개의 sub-system으로 나뉜다. 이러한 Order system에 대하여 Class diagram, Sequence diagram과 State Diagram을 통해 User Management System의 구조를 표현하고 설명한다.

4.2. Class Diagram

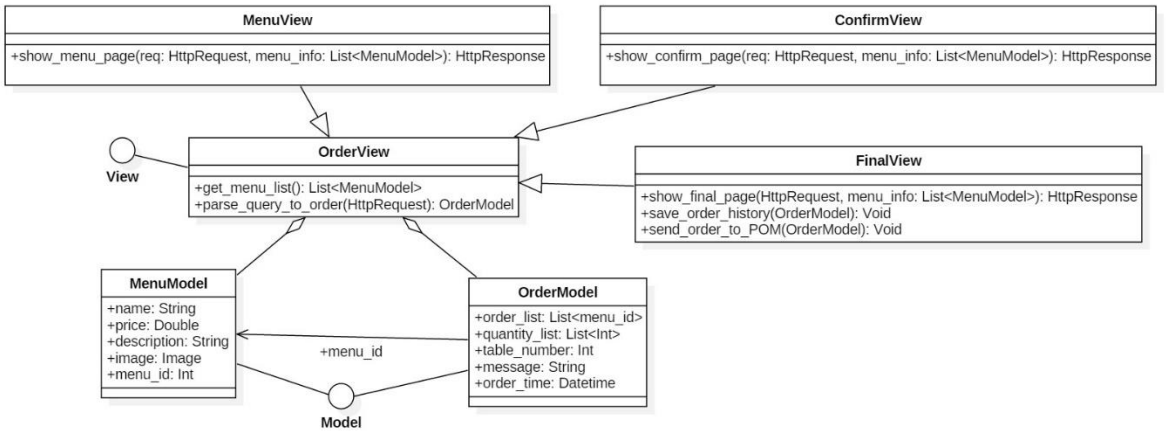

Diagram 7. Create order system class diagram

A. Menu View

A.1. Attributes

해당사항 없음

A.2. Methods

- HttpResponse show_menu_page(HttpRequest, List<MenuModel>):

  메뉴 DB에서 음식 정보를 받아 메뉴판 페이지를 생성한다. 그리고 해당 HttpResponse를 사용자 단말기에 전송한다.

B. Confirm View

B.1. Attributes

해당사항 없음

B.2. Methods

- HttpResponse show_confirm_page(HttpRequest, List<MenuModel>):

  HttpRequest 정보 안에 주문한 음식 내용을 바탕으로 메뉴 DB에서 해당 음식 사진을 받아 주문서 확인 페이지를 생성한다. 그리고 해당 HttpResponse를 사용자 단말기에 전송한다.

C. Order View

C.1. Attributes

해당사항 없음

C.2. Methods

- List<MenuModel> get_menu_list():

  메뉴 DB로부터 데이터를 읽어 메뉴 리스트를 생성해 반환한다.

D. Final View

D.1. Attributes

해당사항 없음

D.2. Methods

- HttpResponse show_final_page(HttpRequest, List<MenuModel>):

  주문 완료 페이지를 생성해 HttpResponse로 사용자 단말기에 전송한다.

- void save_order_history(OrderModel):

최종 주문서 내용을 메뉴 DB에 튜플로 저장한다.

- void send_order_or_POM(OrderModel):

  최종 주문서 데이터를 식당 내 주문 관리 시스템에 전송한다.

E. Menu Model

E.1. Attributes

- String name: 메뉴의 이름
- Double price: 메뉴의 가격
- String description: 메뉴의 상세설명
- Image image: 메뉴의 사진
- Int menu_id: 각 메뉴를 구별하는 primary key

E.2. Methods

해당사항 없음

F. Order Model

F.1. Attributes

- List<menu_id> order_list: 주문한 음식들의 리스트
- List<Int> quantity_id: 각 음식의 수량
- Int table_number: 음식을 주문한 테이블의 넘버
- String message: 음식 주문시 요청사항
- Datetime order_time: 음식을 주문한 시간

F.2. Methods

해당사항 없음

4.3. Sequence Diagram

A. Select menu

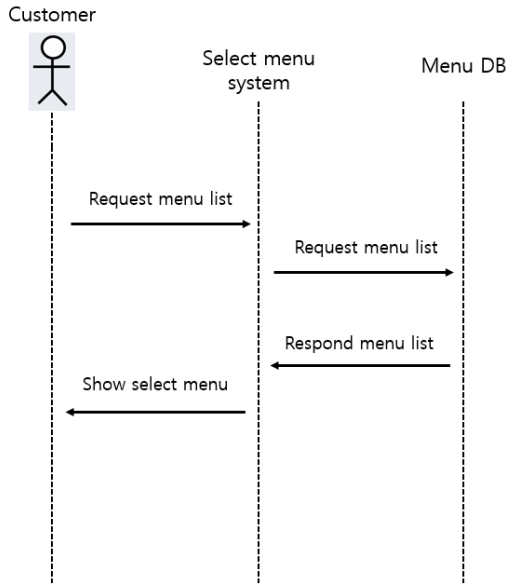

Diagram 8. Select menu sequence diagram

B. Confirm order and modify order

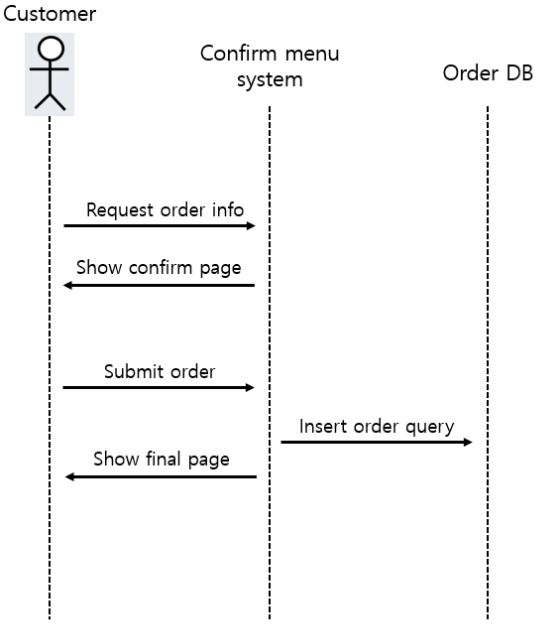

Diagram 9. Confirm and modify order sequence diagram

C. Submit order

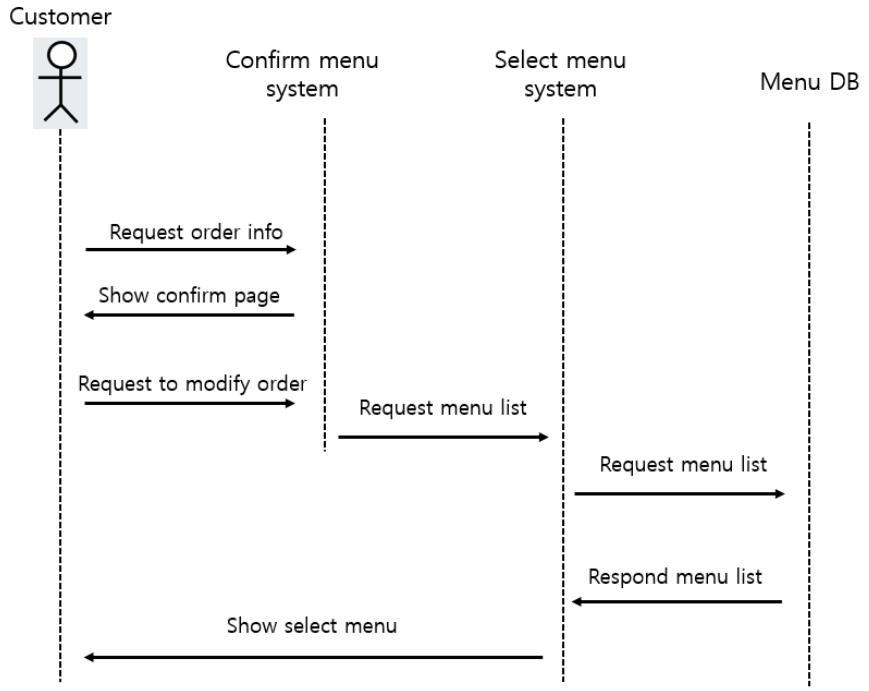

Diagram 10. Submit order sequence diagram

4.4. State diagram

A. Select menu

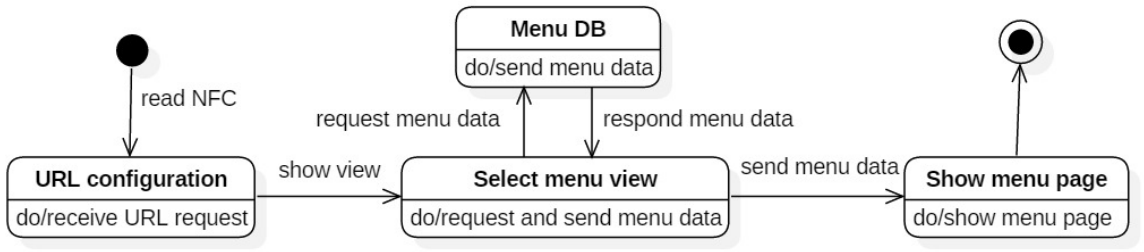

Diagram 11. Select menu state diagram

B. Confirm menu and final page

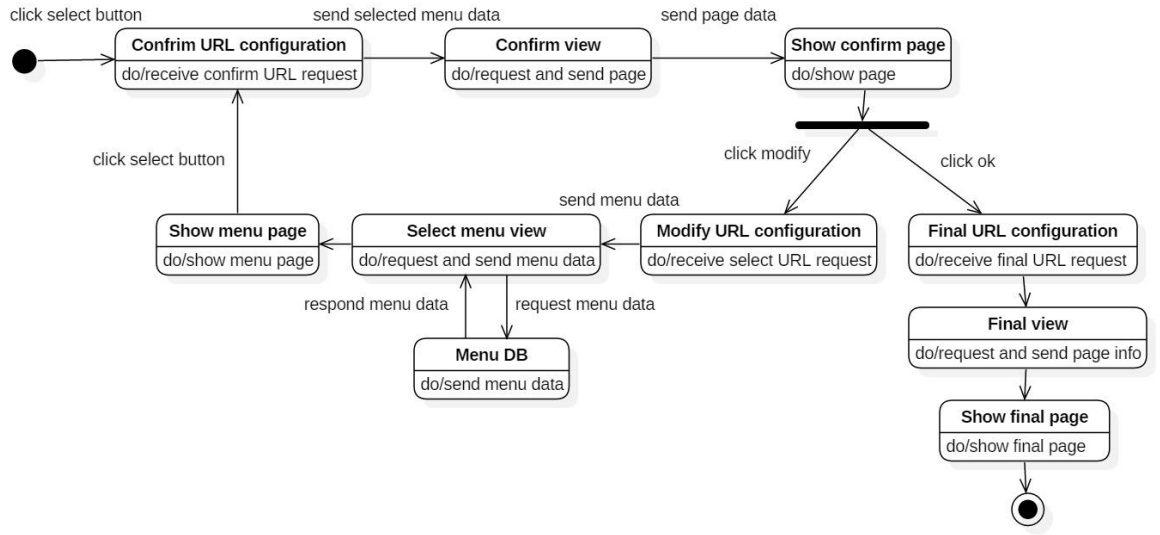

Diagram 12. Confirm menu and final page state diagram

5. Restaurant System

5.1. Objective

식당 종업원이 사용할 시스템의 설계를 설명한다. Restaurant System은 3개의 서브 시스템을 가지고 있다. 주방에서 조리 대기 중인 주문을 관리하는 Kitchen System, 홀에서 각 테이블의 주문 정보와 서빙 해야 하는 주문을 관리하는 Hall System, 그리고 추후에 분석하기 위해 주문 정보를 저장하는 Order History System이 있다. Class diagram, sequence diagram과 State Chart diagram을 통해 Restaurant System의 구조를 표현하고 설명한다.

5.2. Class Diagram

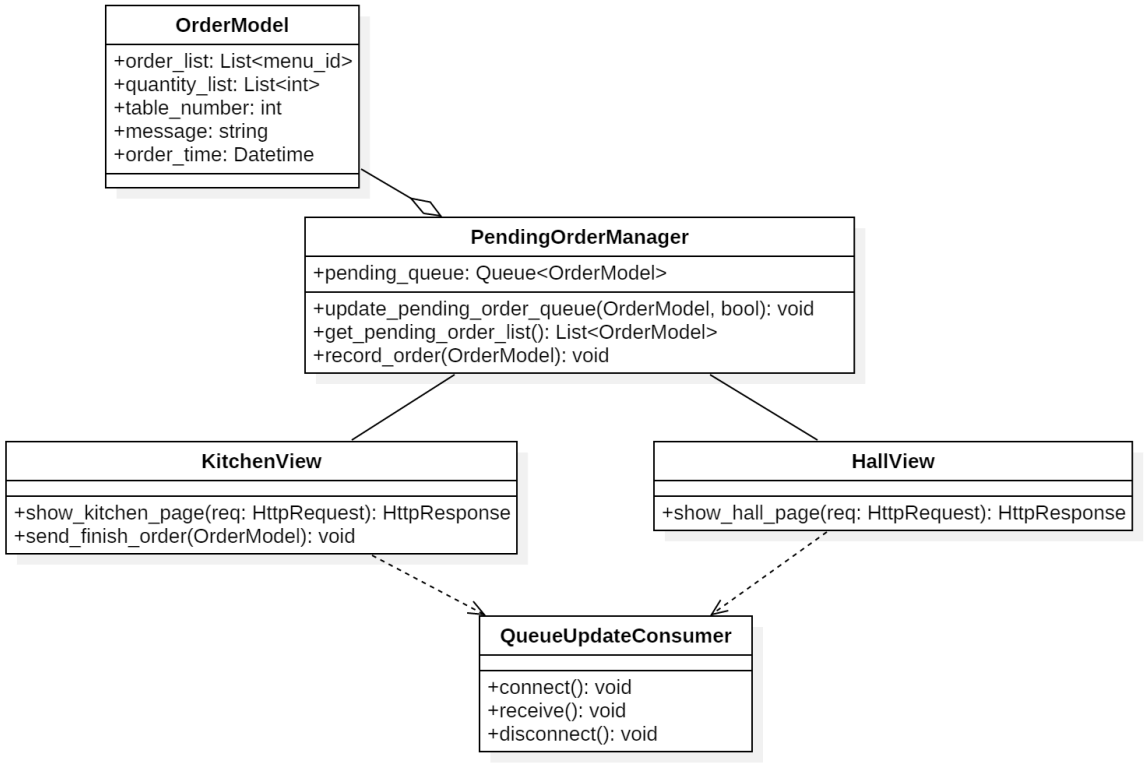

Diagram 13. Restaurant system class diagram

A. OrderModel

A. 1. Attributes

- order_list: 주문한 메뉴 리스트
- quantity_list: 주문한 메뉴들의 수
- table_number: 주문한 테이블의 번호
- message: 주방 전달 사항
- order_time: 주문 시간

A.2. Method

해당사항 없음

B. PendingOrderManager

B.1. Attributes

- Pending_queue: 현재 주문 대기열

B.2. Operations

- Update_pending_order_queue(OrderModel, bool):

  대기열에 새로운 주문을 push하거나 완료된 주문을 pop함

- Get_pending_order_list():

  현재 주문 대기열을 리스트 형식으로 반환

- Record_order(OrderModel):

  주문을 Order History DB에 저장함

C. KitchenView

C.1. Attributes

해당사항 없음

C.2. Operations

- Show_kitchen_page(HttpRequest):

  Http 요청을 받아서 주문 대기열 상황을 출력하고 수정할 수 있는 Kitchen Page를 반환.

- Send_finish_order(OrderModel):

  어떤 주문의 조리가 끝났음을 시스템에 알림

D. HallView

D.1. Attributes

해당사항 없음

D.2. Operations

- Show_hall_page(HttpResponse):

  Http 요청을 받아서 서빙 해야 하는 주문을 알려주고, 식당의 주문 상황을 출력하고 수정할 수 있는 Hall Page를 반환

5.3. Sequence Diagram

A. Kitchen System

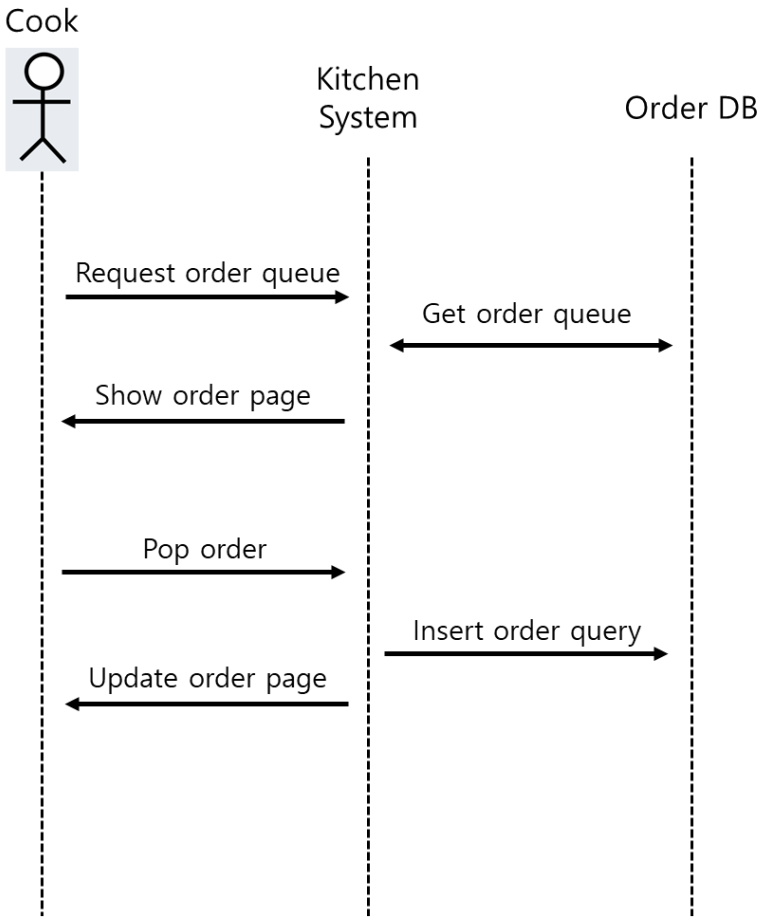

Diagram 14. Kitchen system sequence diagram

B. Hall System

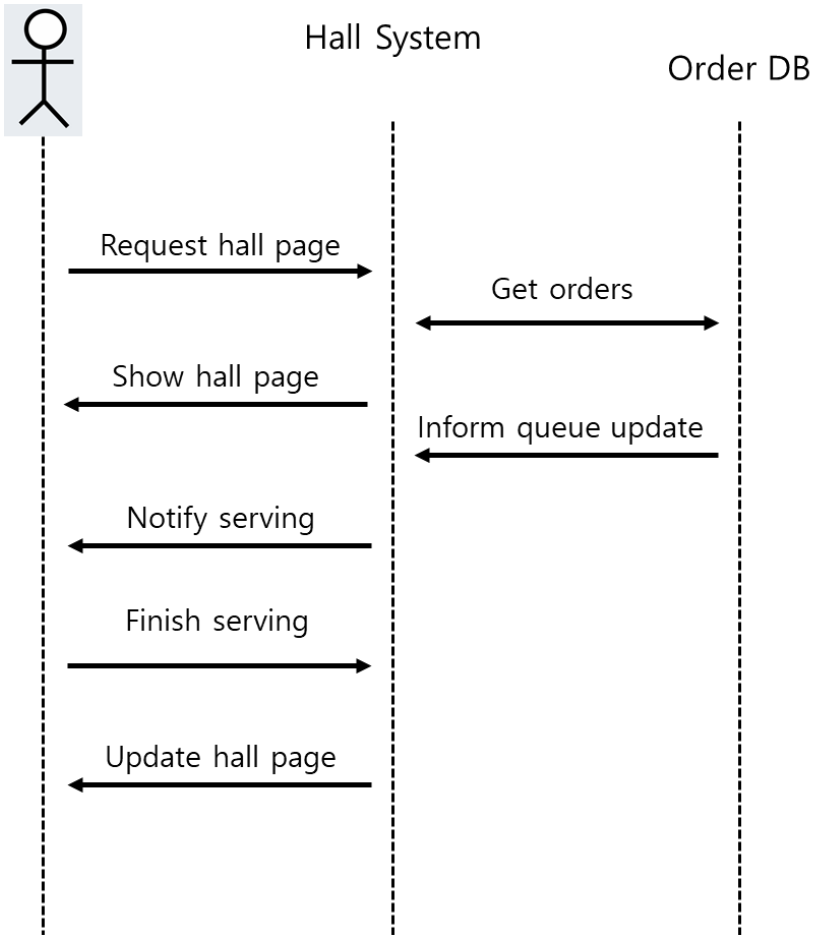

Diagram 15. Hall system sequence diagram

C. Order History System

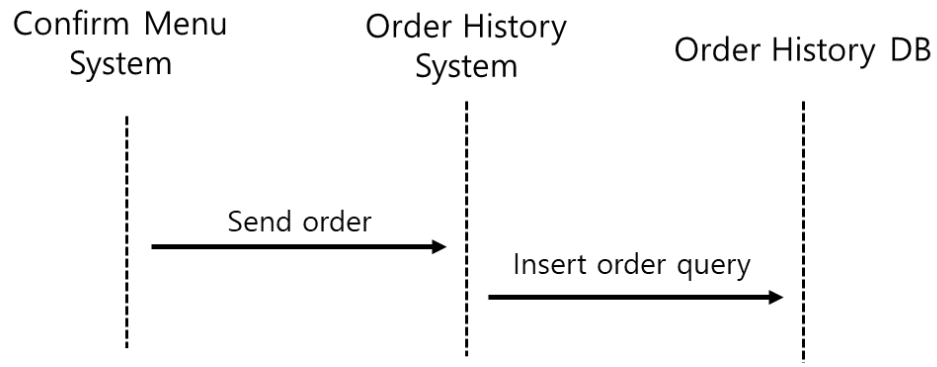

Diagram 16. Order history system sequence diagram

6. Protocol Design

6.1. Objective

Protocol Design 에서는 서브 시스템들이 상호 통신하기 위하여 필수적으로 준수해야 하는 프로토콜에 대하여 서술한다. 통신하는 메시지의 형식과 용도, 의미를 설명한다.

6.2. HTTP POST

Hypertext Transfer Protocol (HTTP)는 분산된 hypermedia 정보 시스템에 대한 애플리케이션 프로토콜이다. HTTP는 WWW상에서 주로 HTML 문서를 주고 받는데 쓰인다. HTTP는 클라이언트와 서버 사이에 이루어지는 요청/응답(request/response) 프로토콜이다. POST는 HTTP에서 request를 통해 server에 인수를 전달하는 방식으로, HTML body 안에 <key>value</key> 형식으로 전달된다.

6.3. WebSocket

WebSocket은 단일 TCP 연결을 이용해 전이중 통신을 지원하는 컴퓨터 통신 프로토콜이다. WebSocket은 낮은 오버헤드로 웹 client와 웹 server간의 상호작용을 가능하게 한다. Attribute: Value 형식으로 표현되는 JSON등을 사용해 복잡한 데이터를 편리하게 보낼 수도 있다.

6.4. Protocol Description

A. Overview

HTTP 통신에서 client와 server 사이에서 전송되는 메시지의 형태를 정의한다. Client의 요청과 server의 응답에 포함될 내용을 설명한다.

B. Menu Request Protocol

B.1. Request

해당 없음

B.2. Response

Table 2. Menu Request Protocol

Key	Value
menu_ids	각 메뉴의 menu_id
menu_prices	각 메뉴의 가격
menu_descriptions	각 메뉴의 설명

C. Order Submit Protocol

C.1. Request

Key	Value
order_ids	주문한 메뉴들의 menu_id
order_quantites	주문한 메뉴들 각각의 수량
order_descriptions	주문한 메뉴들에 대한 전달 사항
table_number	주문한 테이블의 번호

Table 3. Order Submit Request Protocol

C.2. Response

Key	Value
order_success	주문 전달 성공 여부

Table 4. Order Submit Response Protocol

D. Order Queue Update (WebSocket)

D.1. Server to Client

Attribute	Value
new_order_id	대기열에 추가된 주문들의 order_id
new_order_menus	대기열에 추가된 주문들의 메뉴들의 menu_id
new_order_quantites	대기열에 추가된 주문들의 메뉴들의 수량
new_order_time	대기열에 추가된 주문들의 주문 시간

Table 5. Order queue update, Server to Client

D.2. Client to Sever

Attribute	Value
finished_order_id	조리를 완료한 주문의 order_id

Table 6. Order queue update, Client to Sever

E. Table System Update (WebSocket)

E.1. Server to Client

Attribute	Value
table_orders	각 테이블의 주문들의 order_id
table_orders_menus	각 테이블의 주문들의 메뉴들의 menu_d
wating_orders	서빙 대기 중인 주문들의 order_id

Table 7. Table System Update, Server to Client

E.2. Client to Server

Attribute	Value
finished_order_id	서빙을 완료한 주문의 order_id

Table 8. Table System Update, Client to Server

7. Database Design

7.1. Objectives

Database Design 에서는 요구사항 명세서에서 기술하였던 요구사항을 기반으로 한 데이터베이스를 설계하고 이에 대하여 설명한다. 요구사항에 기반한 데이터베이스의 ER Diagram을 제시하고, 이에 대한 Relation Schema를 서술한다. Normalization 과정을 통하여 데이터베이스 간에 존재할 수 있는 중복을 방지하고, SQL DDL을 작성한다.

7.2. ER Diagram

데이터베이스는 MySQL 구조를 사용하였으며, 어플리케이션은 서버의 PHP 파일을 통하여 DB에게 요청을 보내고 데이터를 전송 받게 된다. 전체적인 데이터베이스의 구조는 다음과 같다.

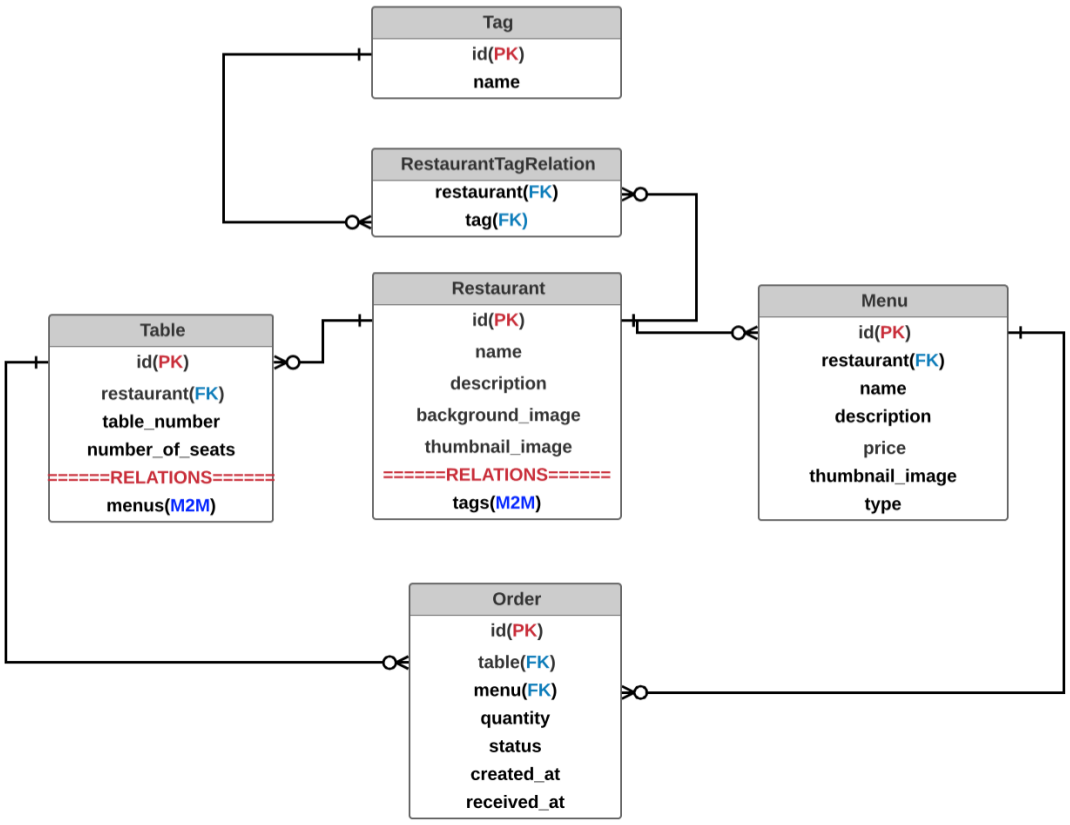

Diagram 17. Database ER diagram

7.3. Relational Schema

A. RESTAURANT

RESTAURANT 테이블의 Relational Schema는 다음과 같다.

RESTAURANT				
ID	**Name**	**Description**	**Background_image**	**Thumbnail_image**

- Primary Key: ID
- Functional Dependency: ID → {Name, Description, Background_image, Thumbnail_image}

설명: 식당 정보를 저장하는 테이블이다. Primary Key는 ID이며, ID가 나머지 attribute를 결정하는 functional dependency를 가지고 있다. Name은 식당의 이름을 저장하고 Description은 위치정보와 같은 부가적인 정보를 저장한다. Background_image와 Thumbnail_image는 식당의 사진을 저장한다.

```
CREATE TABLE RESTAURANT (
        ID                  VARCHAR(10)     NOT NULL,
        Name                VARCHAR(30)     NOT NULL,
        Description         VARCHAR(256)    NOT NULL,
        Background_image    LONGBLOB,
        Thumbnail_image     LONGBLOB
        PRIMARY KEY(ID),
        UNIQUE(ID)
);
```

Background_image와 Thumbnail_image는 꼭 필요하지는 않기 때문에 NOT NULL을 추가하지 않는다.

B. ORDER

ORDER table의 Relational Schema는 다음과 같다.

ORDER						
ID	**Table**	**Menu**	**quantity**	**status**	**Created_at**	**Received_at**

- Primary Key: ID
- Foreign Key: {Table, Menu}
- Functional Dependency: ID → {Table, Menu, quantity, status, created_at, received_at}

```
CREATE TABLE ORDER (
        ID              VARCHAR(10)     NOT NULL,
        Table           INT (10)        NOT NULL,
        Menu            VARCHAR(30)     NOT NULL,
        Quantity        INT(10)         NOT NULL
        Status          INT(3)          NOT NULL
        PRIMARY KEY(ID),
        FOREIGN KEY(Table) REFERENCES TABLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        FOREIGN KEY(Menu) REFERENCES MENU
        ON UPDATE CASCADE
        ON DELETE CASCADE
        UNIQUE(ID)
);
```

C. MENU

MENU table의 Relational Schema는 다음과 같다.

MENU						
ID	**Restaurant**	**Name**	**Description**	**Price**	**Thumbnail_image**	**Type**

- Primary Key: ID
- Foreign Key: Restaurant
- Functional Dependency: ID → {Restaurant, Name, Description, Price, Thumbnail_image, Type}

```
CREATE TABLE MENU (
        ID                  VARCHAR(30)     NOT NULL,
        Restaurant          VARCHAR(10)     NOT NULL,
        Name                VARCHAR(10)     NOT NULL,
        Description         VARCHAR(256)
        Price               INT(10)         NOT NULL
        Thumbnail_image     LONGBLOB
        Type                VARCHAR
        PRIMARY KEY(ID),
        FOREIGN KEY(Restaurant) REFERENCES RESTAURANT
        ON UPDATE CASCADE
        ON DELETE CASCADE
        UNIQUE(ID)
);
```

D. TABLE

Table 테이블의 Relational Schema는 다음과 같다.

TABLE			
ID	**Restaurant**	**Table_number**	**Number_of_seats**

- Primary Key: ID
- Foreign Key: Restaurant
- Functional Dependency: ID → {Restaurant, Table_number, Number_of_seats}

```
CREATE TABLE TABLE (
        ID                  INT(10)         NOT NULL,
        Restaurant          VARCHAR(10)     NOT NULL,
        Table_number        INT(10)         NOT NULL,
        Number_of_seats     INT(10),
        PRIMARY KEY(ID),
        FOREIGN KEY(Restaurant) REFERENCES RESTAURANT
        ON UPDATE CASCADE
        ON DELETE CASCADE
        UNIQUE(ID)
);
```

8. Testing Plan

8.1. objectives

시스템이 구상한 대로 잘 실행되는지 확인하고, 발생할 수 있는 오류들을 사전에 발견하기 위해 테스트를 진행한다. 본 testing plan는 이를 설계하는 단계이다. 각 testing plan에서는 testing policy와 test case에 대해 설명한다.

8.2. Testing policy

본 'Wizard of Order'의 testing은 총 3단계로 구성된다. Developing testing, release testing 그리고 user testing이다. User testing의 경우, 다시 component testing, integrate testing, system testing 그리고 acceptance testing 총 4단계로 나뉜다.

A. Developing testing

시스템 개발 과정에서 진행하는 테스트이다. Component testing은 각 시스템 단위에서 component를 구성하는 요소들이 제대로 동작하는지 확인한다. Integrate testing은 system testing은 테스트가 검증된 컴포넌트들을 점진적으로 합치면서 테스트를 진행한다. 새롭게 발견되는 오류가 없는지 확인한다. 그리고 acceptance testing은 최종적으로 통합된 시스템이 사용자의 요구사항을 잘 반영하고 있는지 테스트해 확인한다.

B. Release testing

Product 혹은 service를 출시 전에 최종적으로 완성된 system에 대해서 테스트를 진행한다. 사전에 계획하고 명시한 사용자의 요구사항이 잘 반영되었는지 확인한다.

C. User testing

사용자가 직접 제품 및 서비스를 이용하면서 진행하는 테스트이다.

8.3. Test Case

A. Order Create System

A.1 Select menu

1) User: NFC 태그를 통해 웹 사이트 접속, 메뉴를 골라 주문서를 작성한다.

2) System operation:

Menu DB로부터 메뉴 정보를 불러와 사용자에게 웹 페이지를 생성해 전송한다.

2-1) 메뉴 선택

시스템 알림: "'~'메뉴를 장바구니에 담았습니다."

시스템 동작: 사용자가 메뉴를 하나씩 고를 때마다 이를 주문서 데이터에 갱신하고, 사용자가 이를 인식할 수 있게 한다.

2-2) 작성 완료

시스템 동작: 작성된 주문서 데이터를 confirm order and submit order단계의 시스템 view에 전송한다.

A.2 Confirm order and submit order

1) User: 작성한 주문서를 확인하고, 주문을 완료할지, 수정할지 결정한다.

2) System operation:

주문서 데이터를 바탕으로 menu DB에서 사진 정보를 받아오고, 사용자가 자신이 주문한 음식들을 확인할 수 있는 웹 페이지를 생성해 단말기에 전송한다.

2-1) 메뉴 수정

시스템 동작: 메뉴 수정을 위해, 주문서 데이터를 select menu단계의 view에 전달하고 다시 주문서를 작성하기 위해 전 단계로 돌아간다.

2-2) 주문 완료

시스템 알림: "주문이 접수되었습니다."

시스템 동작: 주문 완료되었음을 알리는 페이지를 생성하고 이를 단말기에 전송한다. 최종 주문서 데이터는 본 시스템의 'Order History DB'와 restaurant system 내 'Order DB' 테이블에 저장한

다.

A.3 Modify order

1) User: 주문서를 수정하기 위해, 다시 메뉴를 고르며 주문서를 작성한다.

2) System operation:

Menu DB로부터 메뉴 정보를 불러와 사용자에게 웹 페이지를 생성해 전송한다. 이전에 작성한 주문서 데이터는 유지되어야 한다.

2-1) 메뉴 선택

시스템 알림: "'~'메뉴를 장바구니에 담았습니다."

시스템 동작: 사용자가 메뉴를 하나씩 고를 때마다 이를 주문서 데이터에 갱신하고, 사용자가 이를 인식할 수 있게 한다.

2-2) 작성 완료

시스템 동작: 작성된 주문서 데이터를 confirm order and submit order단계의 시스템 view에 전송한다.

B. Restaurant System

B.1 Kitchen system

1) User: 손님이 주문한 주문서들을 리스트 형식으로 확인하고, 서빙이 완료되거나 취소된 주문은 리스트에서 제거한다.

2) System operation:

2-1) 주문서 불러오기

시스템 동작: Order DB에 저장된 주문서 데이터를 조리실에서 사용하는 웹 서비스 상에 표시한다. 각 주문서마다 테이블 번호와 만들어야할 음식들을 명시한다.

2-2) 주문서 삭제

시스템 동작: 웹 페이지에 표시된 주문서를 화면상에서 제거하고, order DB에서도 제거한다.

B.2 Hall system

1) User: 손님이 주문한 주문서들을 리스트 형식으로 확인하고, 서빙을 할 테이블을 확인하고, 음식을 전달한다.

2) System operation:

시스템 동작:

Order DB에 저장된 주문서 데이터를 음식점 홀에서 사용하는 웹 서비스 상에 표시한다. 각 주문서마다 테이블 번호와 서빙을 할 음식들을 명시한다.

B.3 Order history system

1) System operation:

시스템 동작: 최종 주문서 데이터를 'Order History DB' 테이블에 저장한다.

9. Development Environment

9.1. Objectives

Development Environment에서는 개발자의 환경에 대해 설명한다. 사용한 프로그래밍 언어와 SQLite에 대해 서술한다.

9.2. Programming Language & SQLite

A. Programming Language

Figure 8. django logo

Django는 Python으로 작성된 오픈 소스 웹 애플리케이션 프레임워크로, 모델-뷰-컨트롤러(MVC) 패턴을 따르고 있다. 고도의 데이터베이스 기반 웹사이트를 작성하는 데 있어서 수고를 더는 것이 장고의 주된 목표이다. 장고는 컴포넌트의 재사용성(reusability)과 플러그인화 가능성(pluggability), 빠른 개발 등을 강조하고 있다.

B. SQLite

Figure 9. SQLite logo

SQLite는 클라이언트 응용 프로그램에 임베디드 되어 동작하는 오픈 소스 DBMS의 일종이다. 안드로이드, iOS, macOS에 기본적으로 포함되어 있다. 작명은 가벼운 SQL이라는 의미로 SQL + Lite(Light) 이다.

10. Development Plan

10.1. Objectives

본 프로젝트의 개발 일정을 서술한다. Gantt chart를 사용하여 전체적인 개발의 흐름을 한눈에 살펴보고 계획할 수 있게 한다.

10.2. Gantt chart

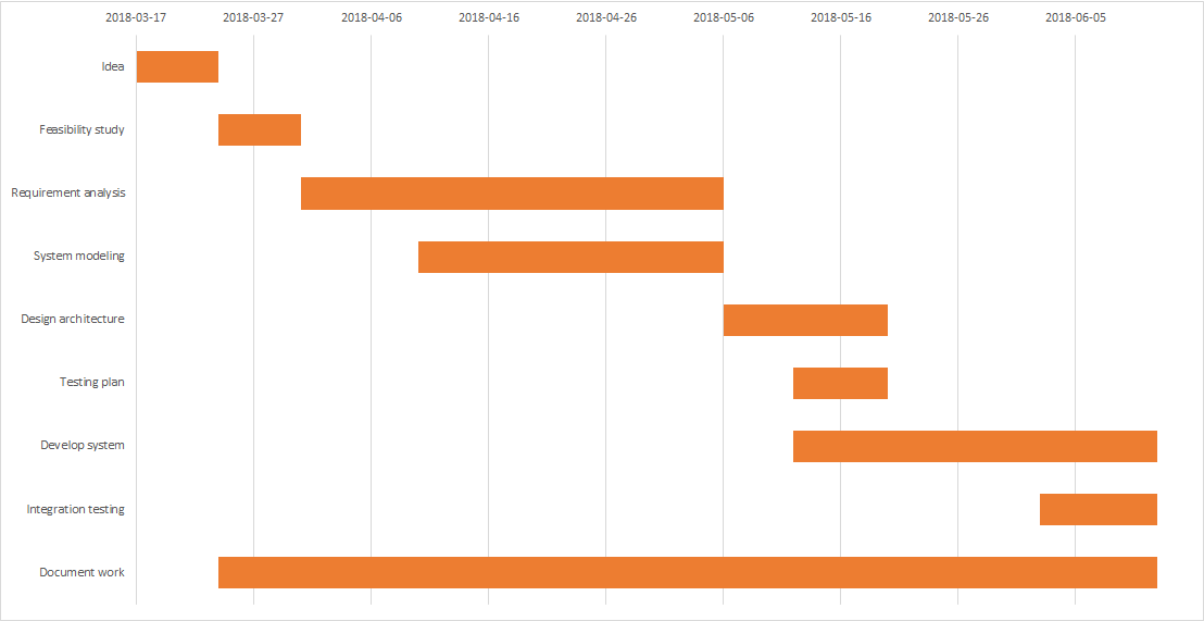

Diagram 18. Gantt Chart

A. Conceptual design

기존 주문 자동화 시스템의 한계점을 발견하고, 이를 개선하기 위한 'Wizard of Order' 웹 서비스란 아이디어를 도출했다. 기존 시제품이 갖는 문제점을 분석하고, 이를 해결하기 위한 본 서비스에서 제공하는 기능들을 정의했다.

B. Requirement engineering

지금껏 수업 시간에 다룬 내용들을 바탕으로 본 서비스 시스템을 개발하기 위한 요구사항명세서를 작성했다. 제안서 작성 때 정의한 요구사항들을 명확하게 분류하여, 유저의 입장에서 필요한 functional requirement와 non-functional requirement로 나누었다. 그리고 이를 바탕으로 시스템을 구성하는 architecture를 고수준의 추상화된 다이어그램으로 구상했다. 그리고 여기서 시스템 개발에 요구되는 functional requirement와 non-functional requirement로 분류해 정리했다. 또한, 여러 UML 모델 작성을 통해 서비스의 기능적인 흐름을 정리하여 전체 시스템을 구성하는 요소들 사이의 상호작용을 명시했다. 마지막으로 본 시스템의 추후 유지보수 때 발생할 수 있는 변화에 대해 미리 조사해 이에 적응 가능한 시스템 개발을 도모했다.

C. Design specification

소프트웨어 개발자 및 엔지니어의 관점에서 본 시스템의 전반적인 기능적 요소들을 정의했다. 전체 시스템을 구성하는 하위 시스템들 그리고 이들을 구성하는 요소들을 여러 다이어그램을 활용하여 시각화하고 추후의 개발에 도움이 될 수 있도록 했다. 개발에 필요한 개발 환경과 툴을 살피고 이를 기반으로 본 웹 서비스를 구성하는 클래스와 오브젝트들을 정리하고 시각화했다.

11. Index

11.1. Objective

Index에서는 본 문서에서 사용된 용어와, 다이어그램, 기능에 대한 인덱스를 나타낸다.

11.2. Table Index

11.3. Figure Index

11.4. Diagram Index

12. Reference

[1] 위키피디아 UML, (2018.05.20)

https://ko.wikipedia.org/wiki/%ED%86%B5%ED%95%A9_%EB%AA%A8%EB%8D%B8%EB%A7%81_%EC%96%B8%EC%96%B4

[2] 위키피디아 Class Diagram, (2018.05.20)

https://en.wikipedia.org/wiki/Class_diagram

[3] 위키피디아 Sequence Diagram, (2018.05.20)

https://en.wikipedia.org/wiki/Sequence_diagram

[4] 위키피디아 State Diagram, (2018.05.20)

https://en.wikipedia.org/wiki/State_diagram

[5] 위키피디아 ER Diagram, (2018.05.20)

https://ko.wikipedia.org/wiki/%EA%B0%9C%EC%B2%B4-%EA%B4%80%EA%B3%84_%EB%AA%A8%EB%8D%B8