

송실대학교 학생식당 식자재 SCM 설계



산업정보시스템공학과 20192208 김형훈

산업정보시스템공학과 20211275 김진호

산업정보시스템공학과 20231342 박소윤

산업정보시스템공학과 20231351 안가은

과목명: 데이터베이스 설계 및 활용

분반: (가)반

프로젝트 그룹: 4조

Table of contents

1 주제 선정	2
1.1 주제	2
1.2 주제 선정 배경	2
1.3 과제 목표 및 범위	2
1.3.1 과제 목표	2
1.3.2 과제 범위	2
2 ER Modeling	3
2.1 E-R Diagram	3
2.2 Entity 및 Attribute	4
2.3 Entity Relationship	5
2.4 그외 요구사항	6
3 Database Design	7
3.1 Database Design	7
3.2 Database Design Schema	8
3.3 Column Property Specifications	8
3.4 SQL DDL	11
4 Implementation	13
4.1 구조	13
4.2 웹 서버 생성	13
4.3 데이터베이스 서버 생성	18
4.4 데이터베이스 설정 및 Application 실행	24
4.5 Application Code	29
4.5.1 Menu 페이지	29
4.5.2 Recipe 페이지	32
4.5.3 Ingredient 페이지	34
4.5.4 Ordering 페이지	37
4.5.5 Supplier 페이지	39
5 DB administration	42
5.1 User User Management	42
5.1.1 유저 생성	42
5.1.2 권한 부여	42
5.1.3 권한 적용	43

5.2 Backup 43

5.3 Recovery 44

5.4 Database Security 44

주제 선정

1.1 주제

송실대학교 학생식당 식자재 관리를 효율적으로 하기 위한 SCM 구축

1.2 주제 선정 배경

매일 수천 명의 학생들에게 식사를 제공하고 있는 송실대학교 학생식당은 효율적인 식자재 관리가 필수적이다. 재고 관리와 발주 과정을 시스템화하여 식자재 낭비를 줄이고, 신선도 관리를 강화하며, 이해관계자 간의 원활한 소통을 실현하고자 한다.¹ 또한 실제 운영되는 시스템을 가상으로 구현해봄으로써 데이터베이스 설계부터 구현까지의 전반적인 개발 과정을 학습하고, SCM(Supply Chain Management)의 실무적 이해를 높이고자 한다.

1.3 과제 목표 및 범위

과제 목표

1. 식자재의 효율적인 공급 및 재고 관리를 위한 통합 시스템 구축
2. 주방 및 부서 간의 원활한 의사소통 및 작업 흐름 관리
3. 공급망 최적화를 통한 운영 효율성 향상

과제 범위

예상 사용자별로 다음 기능들을 구현한다.

1. **주방 직원**
학생식당 메뉴별로 재료 사용량 및 재료 재고 확인
2. **구매 담당자**
재료 주문 생성 및 공급업체 관리
3. **공급업체**
식자재 제공 현황 및 거래 기록 확인
주문 상태 업데이트
4. **영양사**
학생식당 메뉴 등록/수정 및 재료별 영양 정보 확인

¹ Oracle의 [Symphony POS](#)와 같은 전문 시스템은 재고 및 메뉴 관리, 레스토랑 데이터 분석 등의 기능을 제공하여 대형 프랜차이즈의 운영 효율성을 크게 향상시켰다.

ER Modeling

2.1 E-R Diagram

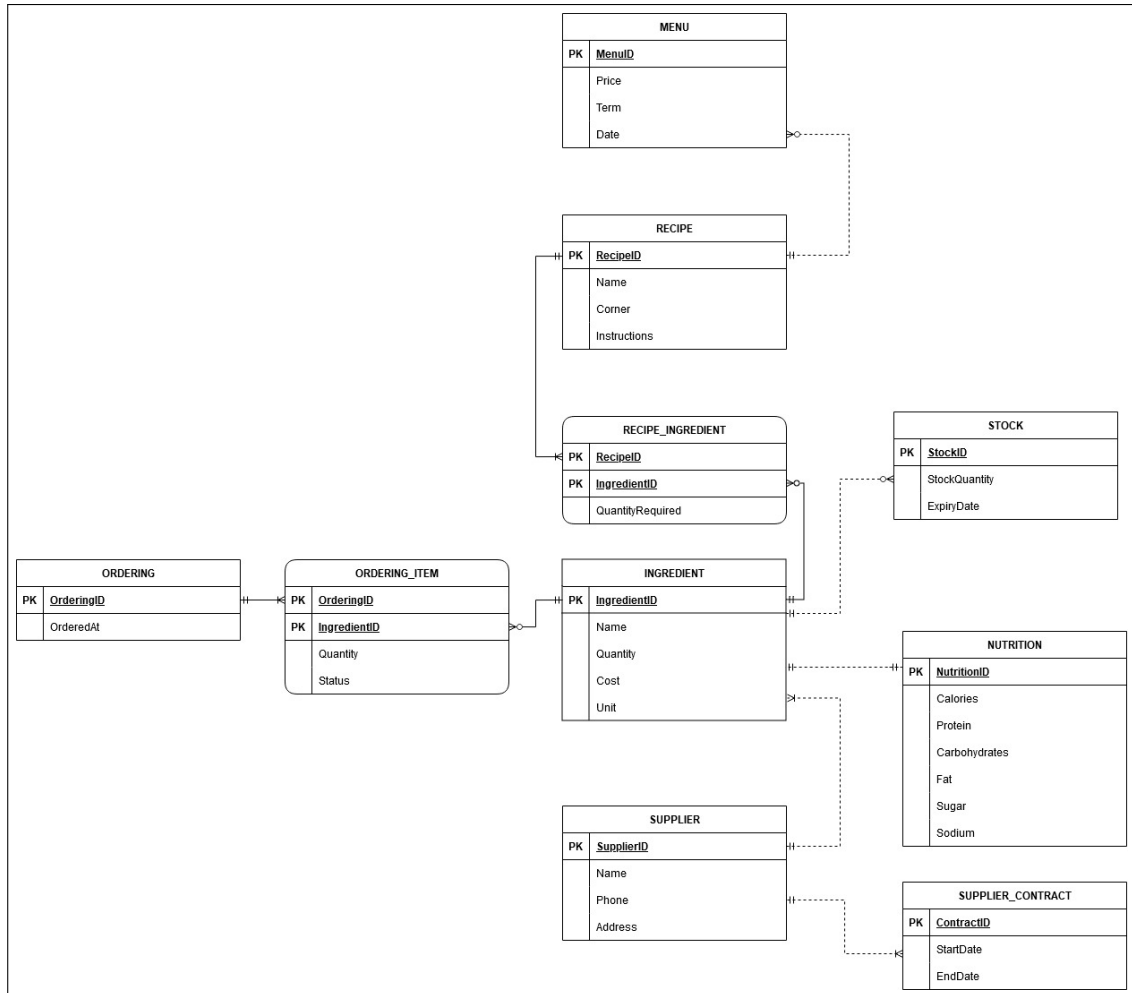


Figure 2.1: ER 다이어그램 전체 모습

2.2 Entity 및 Attribute

ORDERING

- 재료 주문의 기본 정보(주문 시간) 저장

ORDERING_ITEM

- 주문된 재료의 상세 정보(수량, 상태) 저장
- 상세 정보는 공급 업체가 수정 가능
- 상태값: Preparing, Shipping, Delivered, Cancelled, Returning, Returned

MENU

- 학생식당 메뉴 정보(가격, 제공 시간대, 날짜) 저장
- 시간대: Breakfast, Lunch, Dinner

RECIPE

- 요리 레시피 정보(이름, 조리법, 코너) 저장
- 코너: HotPot, RiceBowl, Western

RECIPE_INGREDIENT

- 레시피별 필요한 재료와 수량 정보 저장

INGREDIENT

- 공급업체로부터 구매 가능한 재료 정보(이름, 단가, 단위) 저장

STOCK

- 재료별 재고 정보(수량, 유통기한) 저장

NUTRITION

- 재료의 영양 정보(칼로리, 단백질 등) 저장

SUPPLIER

- 공급업체 정보(이름, 연락처, 주소) 저장

SUPPLIER_CONTRACT

- 공급업체와의 계약 정보(시작일, 종료일) 저장

2.3 Entity Relationship

ORDERING - ORDERING_ITEM - INGREDIENT

- ORDERING_ITEM은 weak entity
- ORDERING, INGREDIENT는 각각 고유한 PK를 가지므로 strong entity
- ORDERING → ORDERING_ITEM: 1:N (Mandatory One, Mandatory Many)
한 번의 주문에는 반드시 하나의 ORDERING_ITEM이 존재한다.
- ORDERING_ITEM ← INGREDIENT: N:1 (Optional Many, Mandatory One)
하나의 ORDERING_ITEM는 반드시 하나의 INGREDIENT에 의존하며, INGREDIENT를 여러번 주문할 수도, 안 할수도 있다.

MENU - RECIPE

- 둘 다 strong entity (non-identifying relationship)
- 1:N 관계 (Optional Many, Mandatory One) 하나의 MENU는 반드시 하나의 RECIPE에 의존하며, 하나의 RECIPE는 여러 MENU에 연결될 수도, 안될 수도 있다.

RECIPE - RECIPE_INGREDIENT - INGREDIENT

- RECIPE_INGREDIENT는 weak entity
- RECIPE, INGREDIENT는 각각 고유한 PK를 가지므로 strong entity
- RECIPE → RECIPE_INGREDIENT: 1:N (Mandatory One, Mandatory Many)
하나의 RECIPE_INGREDIENT는 반드시 하나의 RECIPE에 의존하며, 하나의 RECIPE는 하나 이상의 RECIPE_INGREDIENT가 존재한다.
- RECIPE_INGREDIENT ← INGREDIENT: N:1 (Optional Many, Mandatory One)
하나의 RECIPE_INGREDIENT는 반드시 하나의 INGREDIENT에 의존하며, 하나의 INGREDIENT는 여러 RECIPE_INGREDIENT에 연결될 수도, 안될 수도 있다.

INGREDIENT - STOCK

- 둘 다 strong entity (non-identifying relationship)
- 1:N 관계 (Mandatory One, Optional Many)
STOCK은 반드시 하나의 INGREDIENT에 의존하며, 재고에 하나의 INGREDIENT가 여러 번 존재할수도, 안 존재할 수도 있다. (같은 ingredient 여도 유통기한때문에 주문 날짜가 다르면 다르게 본다.)

INGREDIENT - NUTRITION

- 1:1 관계 (Mandatory One, Mandatory One)
INGREDIENT는 반드시 하나의 NUTRITION에 의존하며, 하나의 NUTRITION도 반드시 하나의 INGREDIENT에 의존한다.
- 둘 다 strong entity (non-identifying relationship)

SUPPLIER - INGREDIENT

- 1:N 관계 (Mandatory One, Mandatory Many)
하나의 INGREDIENT는 하나의 SUPPLIER에 의존하고, 하나의 SUPPLIER는 하나 이상의 INGREDIENT와 연결될 수 있다.
- 둘 다 strong entity (non-identifying relationship)

SUPPLIER - SUPPLIER_CONTRACT

- 1:N 관계 (Mandatory One, Mandatory Many)
하나의 SUPPLIER는 하나 이상의 SUPPLIER_CONTRACT와 연결될 수 있고, SUPPLIER_CONTRACT는 반드시 하나의 SUPPLIER와 연결된다.
- 둘 다 strong entity (non-identifying relationship)

2.4 그외 요구사항

1. MENU entity의 Term attribute에는 ('Breakfast', 'Lunch', 'Dinner') 중 하나의 값만 들어온다.
2. RECIPE entity의 Corner attribute에는 ('HotPot', 'RiceBowl', 'Western') 중 하나의 값만 들어온다.
3. ORDERING_ITEM entity의 Status attribute에는 ('Preparing', 'Shipping', 'Delivered', 'Cancelled', 'Returning', 'Returned') 중 하나의 값만 들어온다.

Database Design

MySQL을 사용할 것을 고려하여 데이터베이스를 설계하였습니다. 1:N의 경우 외래키를 N쪽에, 1:1의 경우 외래키를 둘 중 하나에 추가했습니다.

3.1 Database Design

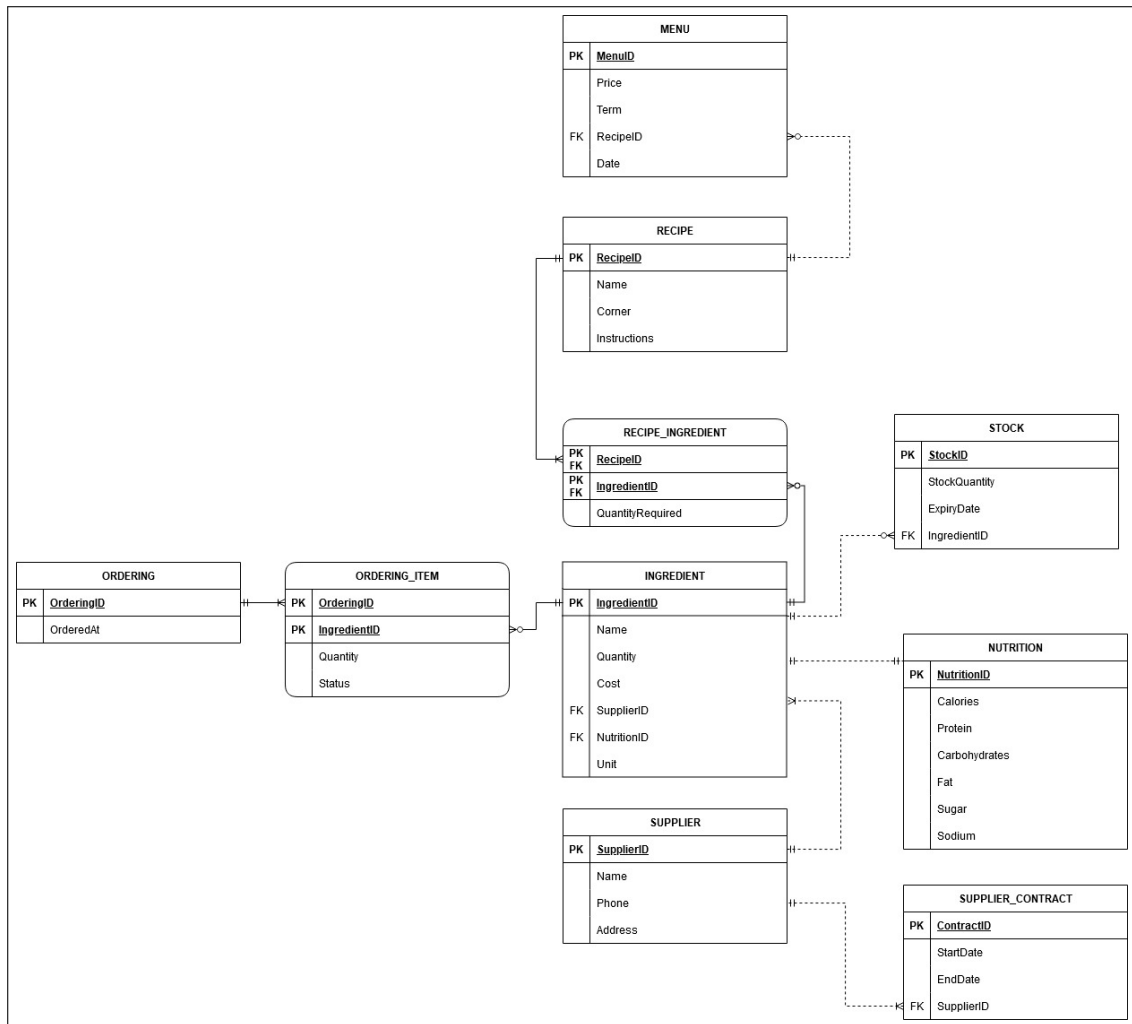


Figure 3.1: Foreign key를 추가한 모습

3.2 Database Design Schema

Entity	Attributes
MENU	<u>MenuID</u> , Price, Term, <i>RecipeID</i> , Date
RECIPE	<u>RecipeID</u> , Name, Corner, Instructions
RECIPE_INGREDIENT	<i>RecipeID</i> , <u>IngredientID</u> , QuantityRequired
INGREDIENT	<u>IngredientID</u> , Name, Quantity, Cost, <i>SupplierID</i> , <i>NutritionID</i> , Unit
STOCK	<u>StockID</u> , StockQuantity, ExpiryDate, <i>IngredientID</i>
SUPPLIER	<u>SupplierID</u> , Name, Phone, Address
SUPPLIER_CONTRACT	<u>ContractID</u> , StartDate, EndDate, <i>SupplierID</i>
NUTRITION	<u>NutritionID</u> , Calories, Protein, Carbohydrates, Fat, Sugar, Sodium
ORDERING_ITEM	<i>OrderingID</i> , <u>IngredientID</u> , Quantity, Status
ORDERING	<u>OrderingID</u> , OrderedAt

Primary key: 쿼은글씨, 밑줄

Foreign key: 기울임

3.3 Column Property Specifications

Table 3.1: MENU 테이블

ColumnName	DataType	Key	Required	DefaultValue	Remarks
MenuID	int	PK	YES	AUTO_INCREMENT	
Price	decimal(10,2)		YES		
Term	char(10)		YES		'Breakfast', 'Lunch', 'Dinner'
RecipeID	int	FK	YES		
Date	date		YES		

Table 3.2: RECIPE 테이블

ColumnName	DataType	Key	Required	DefaultValue	Remarks
RecipeID	int	PK	YES	AUTO_INCREMENT	
Name	varchar(255)		YES		
Corner	char(50)		YES		'HotPot', 'RiceBowl', 'Western'
Instructions	varchar(4095)		YES		

Table 3.3: RECIPE_INGREDIENT 테이블

ColumnName	DataType	Key	Required	DefaultValue	Remarks
RecipeID	int	PK, FK	YES		
IngredientID	int	PK, FK	YES		
QuantityRequired	decimal(15,10)		YES		

Table 3.4: INGREDIENT 테이블

ColumnName	DataType	Key	Required	DefaultValue	Remarks
IngredientID	int	PK	YES	AUTO_INCREMENT	
Name	varchar(255)		YES		
Quantity	decimal(10,2)		YES		
Cost	decimal(10,2)		YES		
SupplierID	int	FK	YES		
NutritionID	int	FK, UNIQUE	YES		
Unit	char(5)		YES		

Table 3.5: STOCK 테이블

ColumnName	DataType	Key	Required	DefaultValue	Remarks
StockID	int	PK	YES	AUTO_INCREMENT	
StockQuantity	decimal(10,2)		YES		
ExpiryDate	datetime		NO		
IngredientID	int	FK	YES		

Table 3.6: SUPPLIER 테이블

ColumnName	DataType	Key	Required	DefaultValue	Remarks
SupplierID	int	PK	YES	AUTO_INCREMENT	
Name	varchar(255)		YES		
Phone	char(15)		YES		
Address	varchar(255)		YES		

Table 3.7: SUPPLIER_CONTRACT 테이블

ColumnName	DataType	Key	Required	DefaultValue	Remarks
ContractID	int	PK	YES	AUTO_INCREMENT	
StartDate	date		YES		
EndDate	date		YES		
SupplierID	int	FK	YES		

Table 3.8: NUTRITION 테이블

ColumnName	DataType	Key	Required	DefaultValue	Remarks
NutritionID	int	PK	YES	AUTO_INCREMENT	
Calories	decimal(10,2)		NO		kcal
Protein	decimal(10,2)		NO		g
Carbohydrates	decimal(10,2)		NO		g
Fat	decimal(10,2)		NO		g
Sugar	decimal(10,2)		NO		g
Sodium	decimal(10,2)		NO		mg

Table 3.9: ORDERING_ITEM 테이블

ColumnName	DataType	Key	Required	DefaultValue	Remarks
OrderingID	int	PK, FK	YES		
IngredientID	int	PK, FK	YES		
Quantity	int		YES		
Status	char(20)		YES		'Preparing', 'Shipping', 'Delivered', 'Cancelled', 'Returning', 'Returned'

Table 3.10: ORDERING 테이블

ColumnName	DataType	Key	Required	DefaultValue	Remarks
OrderingID	int	PK	YES	AUTO_INCREMENT	
OrderedAt	datetime		NO		

3.4 SQL DDL

```
CREATE DATABASE IF NOT EXISTS SOONGSIL_STUDENT_CAFETERIA_SCM;
USE SOONGSIL_STUDENT_CAFETERIA_SCM;

CREATE TABLE `INGREDIENT` (
  `IngredientID` int NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `Name` varchar(255) NOT NULL,
  `Quantity` decimal(10, 2) NOT NULL,
  `Cost` decimal(10, 2) NOT NULL,
  `SupplierID` int NOT NULL,
  `NutritionID` int NOT NULL UNIQUE,
  `Unit` char(5) NOT NULL
);

CREATE TABLE `MENU` (
  `MenuID` int NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `Price` decimal(10, 2) NOT NULL,
  `Term` char(10) NOT NULL,
  `RecipeID` int NOT NULL,
  `Date` date NOT NULL,
  CONSTRAINT chk_term CHECK (`Term` IN ('Breakfast', 'Lunch', 'Dinner'))
);

CREATE TABLE `ORDERING_ITEM` (
  `OrderingID` int NOT NULL,
  `IngredientID` int NOT NULL,
  `Quantity` int NOT NULL,
  `Status` char(20) NOT NULL,
  PRIMARY KEY (`OrderingID`, `IngredientID`),
  CONSTRAINT chk_status CHECK (`Status` IN ('Preparing', 'Shipping', 'Delivered', 'Cancelled', 'Returning', 'Returned'))
);

CREATE TABLE `NUTRITION` (
  `NutritionID` int NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `Calories` decimal(10, 2),
  `Protein` decimal(10, 2),
  `Carbohydrates` decimal(10, 2),
  `Fat` decimal(10, 2),
  `Sugar` decimal(10, 2),
  `Sodium` decimal(10, 2)
); -- (kcal, g, g, g, g, mg)

CREATE TABLE `ORDERING` (
  `OrderingID` int NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `OrderedAt` datetime
);

CREATE TABLE `RECIPE` (
  `RecipeID` int NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `Name` varchar(255) NOT NULL,
  `Corner` char(50) NOT NULL,
  `Instructions` varchar(4095) NOT NULL,
  CONSTRAINT chk_corner CHECK (`Corner` IN ('HotPot', 'RiceBowl', 'Western'))
);

CREATE TABLE `RECIPE_INGREDIENT` (
  `RecipeID` int NOT NULL,
  `IngredientID` int NOT NULL,
  `QuantityRequired` decimal(15, 10) NOT NULL,
  PRIMARY KEY (`RecipeID`, `IngredientID`)
);

CREATE TABLE `STOCK` (
  `StockID` int NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `StockQuantity` decimal(10, 2) NOT NULL,
  `ExpiryDate` datetime,
  `IngredientID` int NOT NULL
);

CREATE TABLE `SUPPLIER` (
  `SupplierID` int NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `Name` varchar(255) NOT NULL,
  `Phone` char(15) NOT NULL,
  `Address` varchar(255) NOT NULL
);

CREATE TABLE `SUPPLIER_CONTRACT` (
  `ContractID` int NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `StartDate` date NOT NULL,
  `EndDate` date NOT NULL,
  `SupplierID` int NOT NULL
);

ALTER TABLE `INGREDIENT` ADD CONSTRAINT `INGREDIENT_NutritionID_fk` FOREIGN KEY (`NutritionID`) REFERENCES `NUTRITION` (`NutritionID`);
ALTER TABLE `INGREDIENT` ADD CONSTRAINT `INGREDIENT_SupplierID_fk` FOREIGN KEY (`SupplierID`) REFERENCES `SUPPLIER` (`SupplierID`);
ALTER TABLE `MENU` ADD CONSTRAINT `MENU_RecipeID_fk` FOREIGN KEY (`RecipeID`) REFERENCES `RECIPE` (`RecipeID`);
ALTER TABLE `ORDERING_ITEM` ADD CONSTRAINT `ORDERING_ITEM_IngredientID_fk` FOREIGN KEY (`IngredientID`) REFERENCES `INGREDIENT` (`IngredientID`);
ALTER TABLE `ORDERING_ITEM` ADD CONSTRAINT `ORDERING_ITEM_OrderingID_fk` FOREIGN KEY (`OrderingID`) REFERENCES `ORDERING` (`OrderingID`);
ALTER TABLE `RECIPE_INGREDIENT` ADD CONSTRAINT `RECIPE_INGREDIENT_IngredientID_fk` FOREIGN KEY (`IngredientID`) REFERENCES `INGREDIENT` (`IngredientID`);
ALTER TABLE `RECIPE_INGREDIENT` ADD CONSTRAINT `RECIPE_INGREDIENT_RecipeID_fk` FOREIGN KEY (`RecipeID`) REFERENCES `RECIPE` (`RecipeID`);
ALTER TABLE `STOCK` ADD CONSTRAINT `STOCK_IngredientID_fk` FOREIGN KEY (`IngredientID`) REFERENCES `INGREDIENT` (`IngredientID`);
ALTER TABLE `SUPPLIER_CONTRACT` ADD CONSTRAINT `SUPPLIER_CONTRACT_SupplierID_fk` FOREIGN KEY (`SupplierID`) REFERENCES `SUPPLIER` (`SupplierID`);
```

Figure 3.2: SQL DDL

Foreign key를 맨 뒤에 추가한 이유는, 테이블을 생성하는 동시에 외래키를 추가할 때, 참조하는 테이블이 존재하지 않을 수 있기 때문입니다.

Implementation

4.1 구조

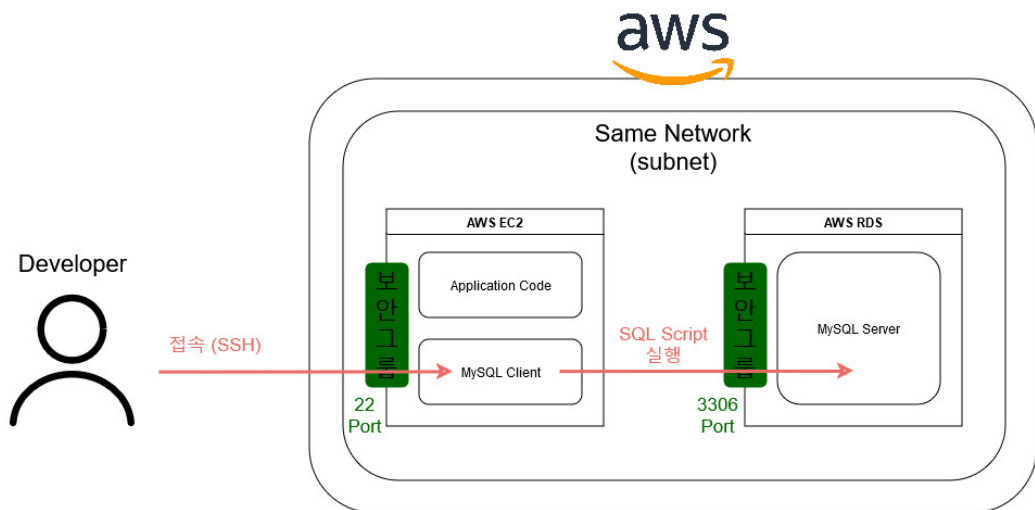


Figure 4.1: 개발 구조

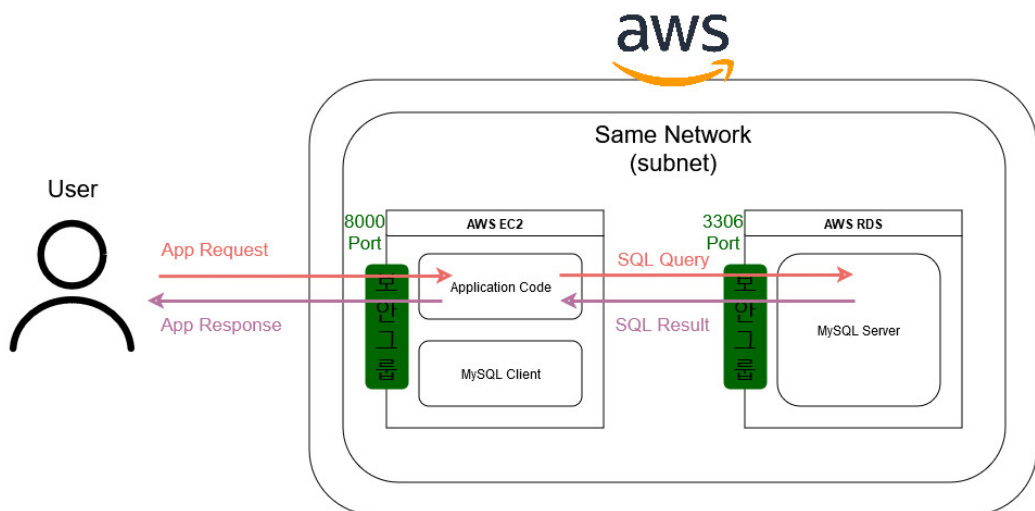


Figure 4.2: 사용자 접근 구조

4.2 웹 서버 생성

AWS Console에서 다음의 과정들을 거쳐 필요한 자원을 생성합니다.

먼저 Application Server이자 Mysql Client 역할을 해줄 컴퓨터(인스턴스)를 생성하겠습니다.

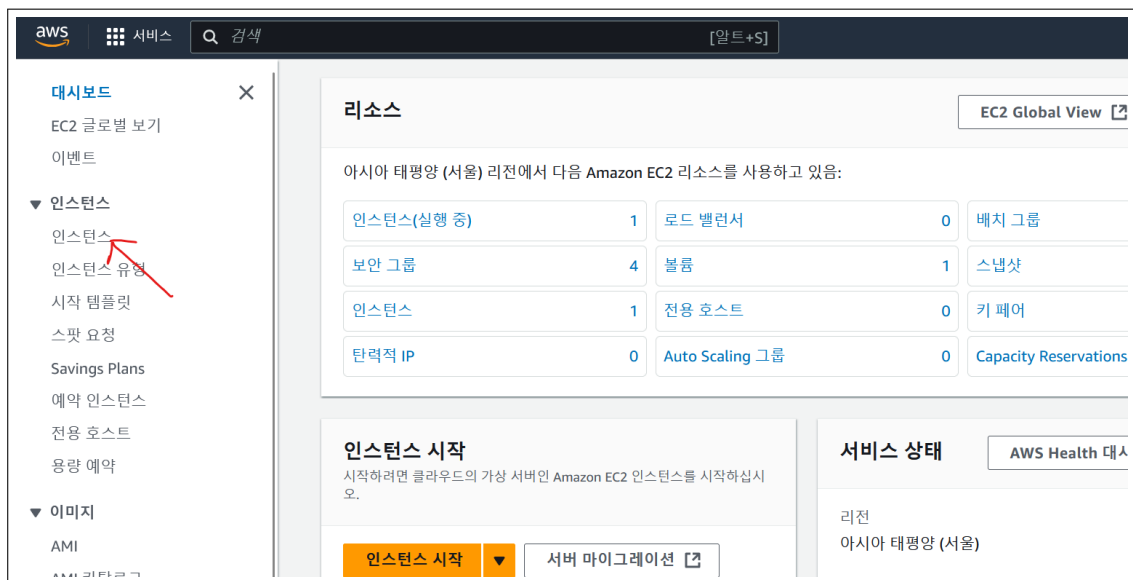
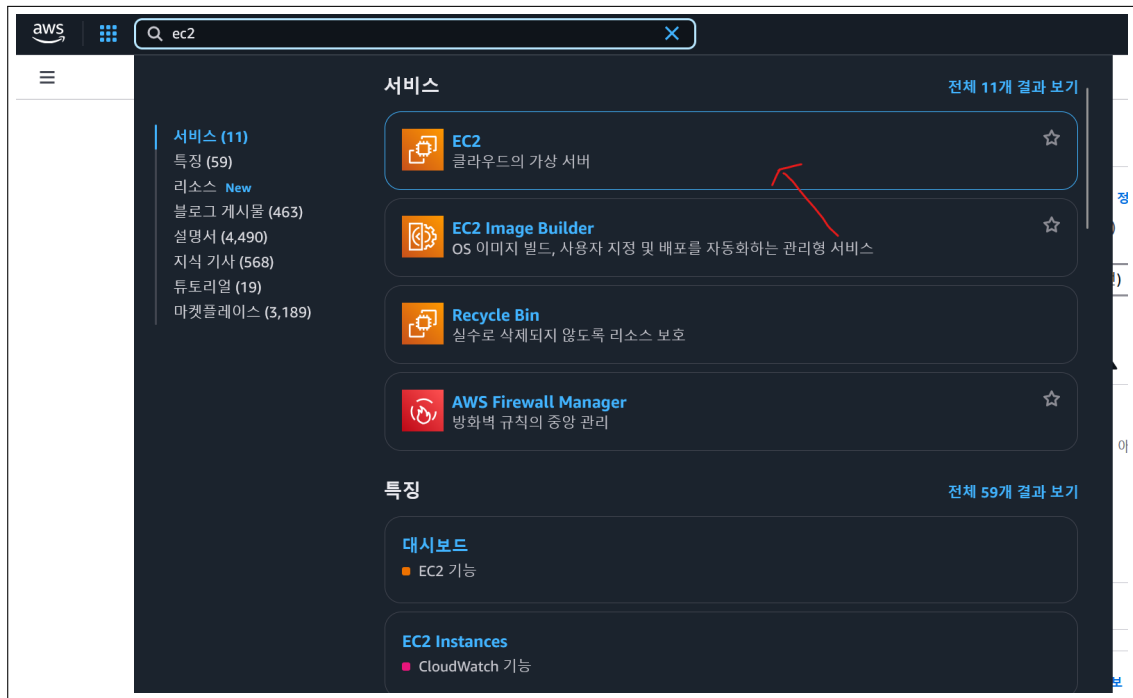
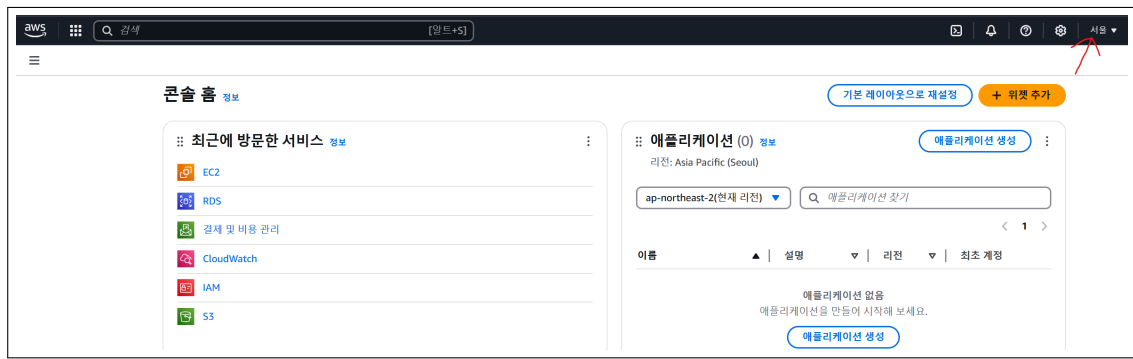


Figure 4.3: AWS console EC2 접속

제일 먼저 지역이 서울로 잘 설정되어 있는지 확인해줍니다.

그런 다음, EC2를 검색한 후 들어간 다음 왼쪽 탭에서 인스턴스를 선택합니다.

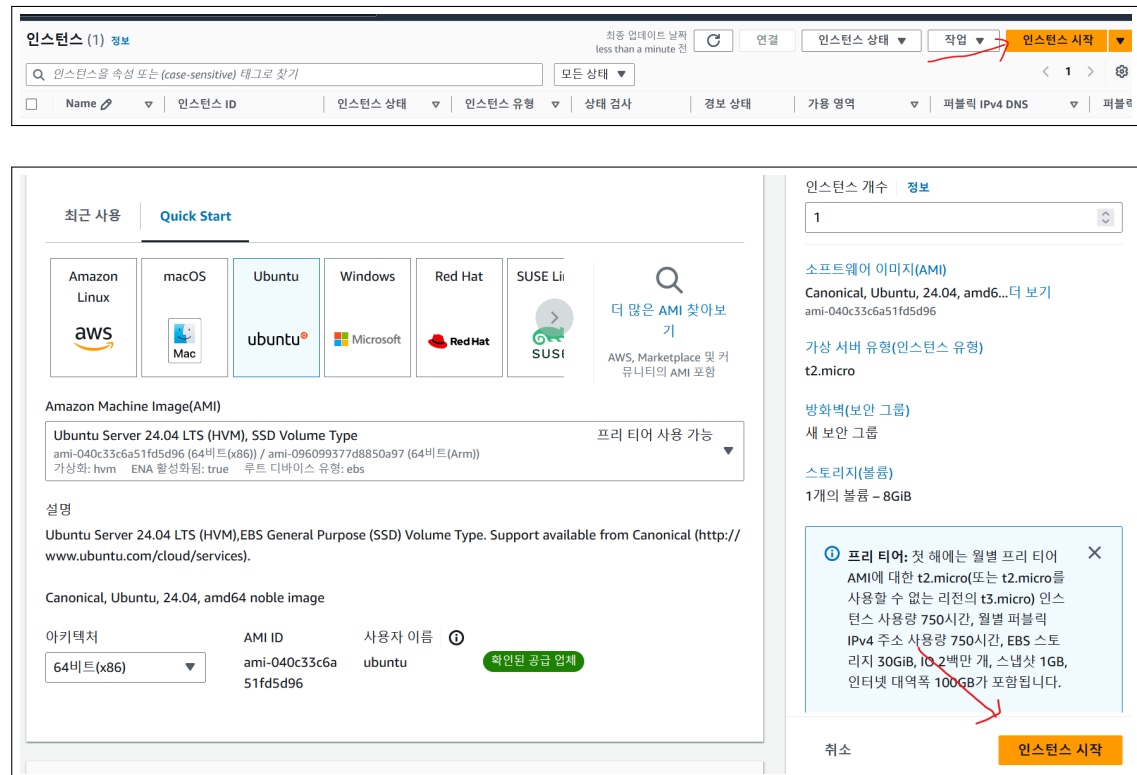


Figure 4.4: EC2 생성

인스턴스 시작을 눌러 사용할 운영체제만 설정 해줍니다.

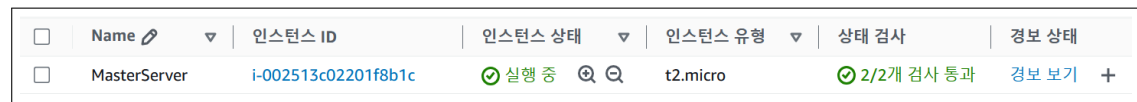


Figure 4.5: EC2 생성 완료

인스턴스가 정상적으로 생성된 모습을 확인해볼 수 있습니다.

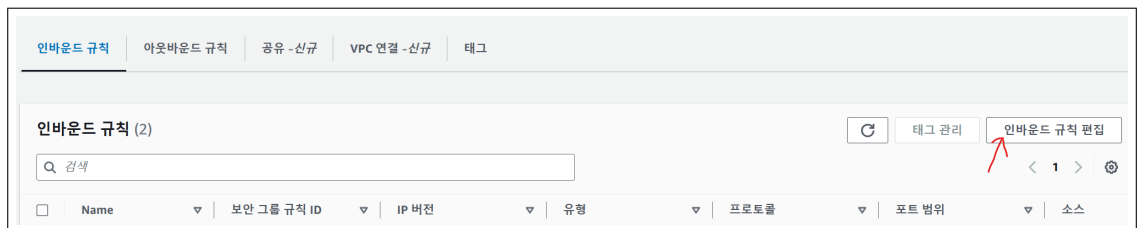
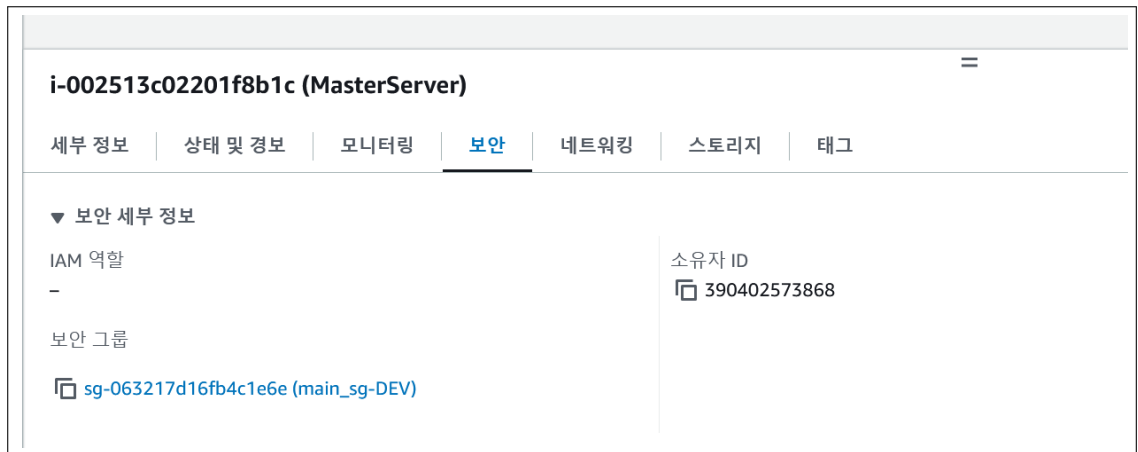
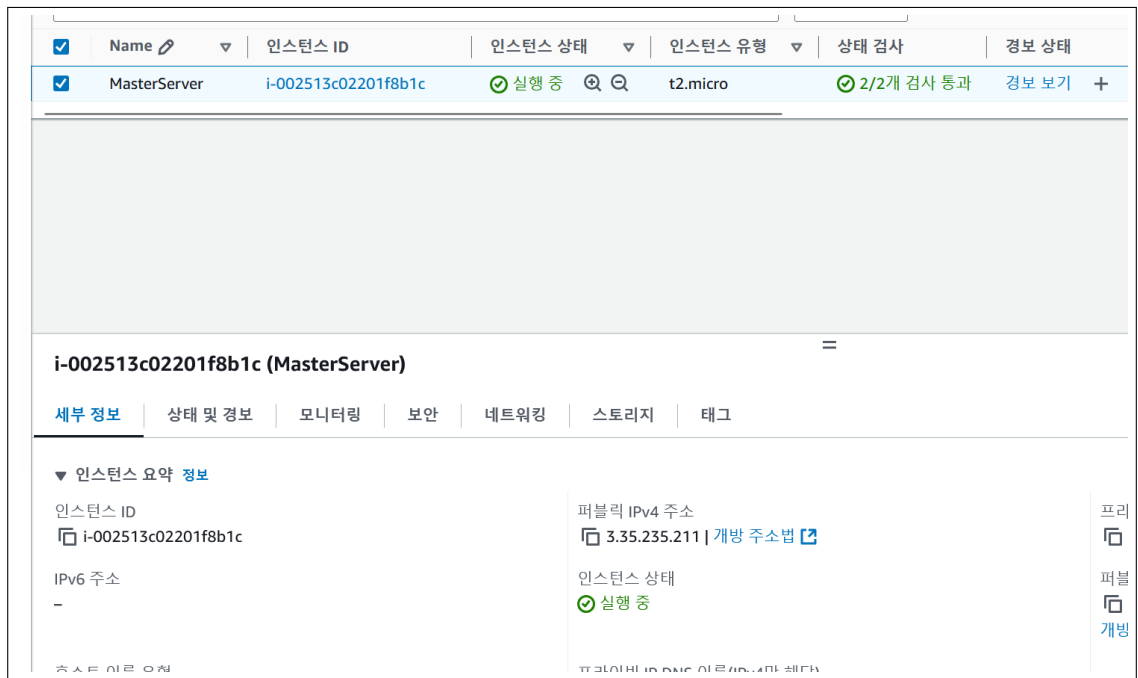


Figure 4.6: EC2 보안 그룹 설정

생성된 인스턴스 옆 체크박스를 선택하면 아래에 추가적인 정보가 보이는데, 여기서 보안을 눌러줍니다. 그런 다음 보안 그룹(firewall)의 'sg-...'으로 보이는 파란 글씨를 누른 다음 인바운드 규칙 편집을 눌러줍니다.

인바운드 규칙 정보

보안 그룹 규칙 ID	유형 정보	프로토콜 정보	포트 범위 정보	소스 정보	설명
sgr-0520ba5796eb1f5d1	SSH	TCP	22	사용자 지정	0.0.0.0/0
sgr-05a47005ebf30c9d7	사용자 지정 TCP	TCP	8000	사용자 지정	0.0.0.0/0

[규칙 추가](#)

sg-063217d16fb4c1e6e - main_sg-DEV

세부 정보

보안 그룹 이름 main_sg-DEV	보안 그룹 ID sg-063217d16fb4c1e6e
소유자 390402573868	인바운드 규칙 수 2 권한 항목

[인바운드 규칙](#) | [아웃바운드 규칙](#) | [공유 - 신규](#) | [VPC 연결 - 신규](#) | [태그](#)

인바운드 규칙 (2)

🔍 검색

☐	Name ▼	보안 그룹 규칙 ID ▼	IP 버전 ▼	유형
☐	-	sgr-0520ba5796eb1f5d1	IPv4	SSH
☐	-	sgr-05a47005ebf30c9d7	IPv4	사용

Figure 4.7: EC2 보안 그룹 설정

다음과 같이 보안그룹을 설정해줍니다.

22번 포트는 컴퓨터에 접속하기 위해, 8000번 포트는 사용자가 웹 서버에 접속하기 위해 열어줍니다.

설정을 마친 후, 보안 그룹 ID만 다른 곳에 복사를 해둡니다.

4.3 데이터베이스 서버 생성

이제 MySQL Server를 생성합니다.



Figure 4.8: AWS RDS 생성

RDS를 검색한 후 들어간 다음 왼쪽에 보이는 탭에서 데이터베이스를 선택합니다.

이후 데이터베이스 생성을 눌러줍니다.

RDS > 데이터베이스 생성

데이터베이스 생성 정보

데이터베이스 생성 방식 선택

☒ 표준 생성
가용성, 보안, 백업 및 유지 관리에 대한 옵션을 포함하여 모든 구성 옵션을 설정합니다.

☐ 손쉬운 생성
권장 모범 사례 구성을 사용합니다. 일부 구성 옵션은 데이터베이스를 생성한 후 변경할 수 있습니다.

엔진 옵션

엔진 유형 정보

☐ Aurora (MySQL Compatible)


☐ Aurora (PostgreSQL Compatible)


☒ MySQL


☐ PostgreSQL


☐ MariaDB

☐ Oracle

템플릿

해당 사용 사례를 충족하는 샘플 템플릿을 선택하세요.

☐ 프로덕션
고가용성 및 빠르고 일관된 성능을 위해 기본값을 사용하세요.

☐ 개발/테스트
이 인스턴스는 프로덕션 환경 외부에서 개발 용도로 마련되었습니다.

☒ 프리 티어
RDS 프리 티어를 사용하여 새로운 애플리케이션을 개발하거나, 기존 애플리케이션을 테스트하거나 Amazon RDS에서 실무 경험을 쌓을 수 있습니다. [정보](#)

Figure 4.9: RDS Configuration

MySQL 엔진 프리티어(무과금)를 기준으로 선택하겠습니다.

월별 추정 요금

Amazon RDS 프리 티어는 12개월 동안 사용할 수 있습니다. 매월 프리 티어를 통해 아래 나열된 Amazon RDS 리소스를 무료로 사용할 수 있습니다.

- 단일 AZ db.t2.micro, db.t3.micro 또는 db.t4g.micro 인스턴스에서 Amazon RDS를 750시간 사용.
- 20GB의 범용 스토리지(SSD).
- 20GB의 자동 백업 스토리지 및 사용자가 시작한 모든 DB 스냅샷.

[AWS 프리 티어에 대해 자세히 알아보세요.](#)

무료 사용이 만료되었거나 애플리케이션에서 프리 티어 사용량을 초과한 경우 [Amazon RDS 요금 페이지](#)에서 설명한 대로, 표준 종량 서비스 요금이 적용됩니다.

❗ 귀하는 AWS 서비스와 함께 사용하는 타사 제품 또는 서비스 일체에 대해 필요한 모든 권리를 보유할 책임이 있습니다.

취소

데이터베이스 생성

데이터베이스 (2)

그룹 리소스

수정

작업 ▾

S3에서 복원

데이터베이스 생성

Q 데이터베이스(들) 기준으로 필터링

< 1 > ⚙

DB 식별자 ▲

상태 ▾

역할 ▾

엔진 ▾

리전 및 ... ▾

크기 ▾

권장 사항 ▾

CPU ▾

[soongsil-student-cafeteria-scm](#)

🔄

생성 중

인스턴스

MySQL Co...

-

db.t4g.mi...

-

Figure 4.11: RDS 생성 완료

설정이 모두 마무리되면 데이터 베이스 생성을 클릭합니다. 그러면 정상적으로 생성된 모습을 확인해볼 수 있습니다.

이제 위의 보이는 화면에서 파란 글씨의 식별자를 선택합니다.

연결 및 보안 | 모니터링 | 로그 및 이벤트 | 구성 | 제로 ETL 통합 | 유지 관리 및 백업 | 데이터 마이그레이션 - 신규 | 태그

연결 및 보안

엔드포인트 및 포트	네트워킹	보안
엔드포인트 terraform-2024111601081838430000000 3.c3um20ke28jo.ap-northeast-2.rds.amazonaws.com 포트 3306	가용 영역 ap-northeast-2c VPC vpc-DEV (vpc-0fef6489f17246149) 서브넷 그룹 rds-subnet-group 서브넷 subnet-08c40b9f87c537c76 subnet-0b087b3d2649111ed 네트워크 유형 IPv4	VPC 보안 그룹 rds-sg-DEV (sg-068cdd7175246cba7)  활성 퍼블릭 액세스 가능 아니요 인증 기관 정보 rds-ca-rsa2048-g1 인증 기관 날짜 May 21, 2061, 02:28 (UTC+09:00) DB 인스턴스 인증서 만료 날짜 November 16, 2025, 10:10 (UTC+09:00)

보안 그룹 (1) 정보

Find resources by attribute or tag

[sg-068cdd7175246cba7](#) X Clear filters

☐	Name ▼	보안 그룹 ID ▼	보안 그룹 이름
☐	RDS-Security-Group	sg-068cdd7175246cba7	rds-sg-DEV

Figure 4.12: RDS 보안그룹

RDS 엔드포인트만 복사를 해둔 다음 보안 그룹을 선택합니다. 그런 다음 보안 그룹 ID를 선택해줍니다.

EC2 > 보안 그룹 > sg-068cdd7175246cba7 - rds-sg-DEV

sg-068cdd7175246cba7 - rds-sg-DEV

작업 ▼

세부 정보

보안 그룹 이름 rds-sg-DEV	보안 그룹 ID sg-068cdd7175246cba7	설명 Managed by Terraform	VPC ID vpc-0fef6489f17246149
소유자 390402573868	인바운드 규칙 수 1 권한 항목	아웃바운드 규칙 수 1 권한 항목	

인바운드 규칙 | 아웃바운드 규칙 | 공유 -신규 | VPC 연결 -신규 | 태그

인바운드 규칙 (1)

태그 관리 | 인바운드 규칙 편집

검색

Name	보안 그룹 규칙 ID	IP 버전	유형	프로토콜	포트 범위	소스

인바운드 규칙 정보

보안 그룹 규칙 ID: sg-0ca966325d9f13b60

유형: MySQL/Aurora

프로토콜: TCP

포트 범위: 3306

소스: 사용자 지정

규칙 추가

규칙 목록

규칙 ID	규칙 이름	유형	프로토콜	포트 범위	소스
sg-068cdd7175246cba7	rds-sg-DEV sg-068cdd7175246cba7	RDS-Security-Group	default sg-02d89f64998b41375	main_sg-DEV sg-063217d16fb4c1e6e	com.amazonaws.ap-northeast-2.ec2-instance-connect

규칙 저장

Figure 4.13: RDS 보안그룹

아까 복사해둔 보안 그룹 ID를 찾아서 선택해준 후 규칙을 저장합니다.

모든 자원할당 과정이 끝났습니다.

이제 EC2에 접속해서, 데이터베이스 설정과 application 실행 과정을 진행하겠습니다

4.4 데이터베이스 설정 및 Application 실행

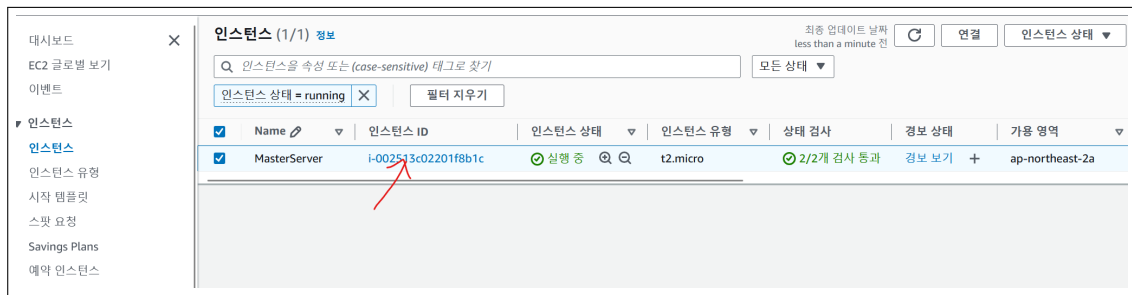


Figure 4.14: EC2 접속

위의 보이는 화면에서 인스턴스 ID를 선택합니다. 그러면 보이는 화면에서 퍼블릭 IPv4 DNS만 복사한 후, 연결을 눌러줍니다.

인스턴스에 연결

정보

다음 옵션 중 하나를 사용하여 인스턴스 i-002513c02201f8b1c (MasterServer)에 연결

EC2 인스턴스 연결

Session Manager

SSH 클라이언트

EC2 직렬 콘솔

인스턴스 ID

i-002513c02201f8b1c (MasterServer)

연결 유형

☒ EC2 Instance Connect을 사용하여 연결
 Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.

☐ EC2 인스턴스 연결 엔드포인트를 사용하여 연결
 프라이빗 IPv4 주소 및 VPC 엔드포인트가 있는 EC2 인스턴스 연결 브라우저 기반 클라이언트를 사용하여 연결합니다.

☒ 퍼블릭 IPv4 주소
 3.35.235.211

☐ IPv6 주소
 -

사용자 이름

인스턴스를 시작하는 데 사용되는 AMI에 정의된 사용자 이름을 입력합니다. 사용자 지정 사용자 이름을 정의하지 않은 경우 기본 사용자 이름인 ubuntu을(를) 사용합니다.

참고:

대부분의 경우 기본 사용자 이름 ubuntu은(는) 정확합니다. 하지만 AMI 사용 지침을 읽고 AMI 소유자가 기본 AMI 사용자 이름을 변경했는지 확인하세요.

취소

연결

```

Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1072-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Thu Nov 21 12:20:24 UTC 2024

System load:  0.0               Processes:            99
Usage of /:   31.4% of 7.57GB   Users logged in:     0
Memory usage: 26%              IPv4 address for eth0: 10.0.1.98
Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
 * compliance features.
 *
 * https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

5 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

2 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '22.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Nov 21 06:23:07 2024 from 39.118.248.251
ubuntu@ip-10-0-1-98:~$
  
```

Figure 4.15: EC2 접속

이어지는 화면에서 다시 연결을 선택하면 EC2 인스턴스에 접속할 수 있습니다. 이제 필요한 프로그램들을 설치합니다.

```
sudo apt update
sudo apt install -y mysql-client python-is-python3 python3-pip
pip install flask pymysql

git clone https://github.com/cryscham123/db_gimal_work.git work # source code

export RDS_ENDPOINT=<Copied RDS Endpoint>
export RDS_USERNAME=<RDS User Name>
export RDS_PASSWORD=<RDS Password>
# create database
mysql -h $RDS_ENDPOINT -u $RDS_USERNAME -p $RDS_PASSWORD < work/sql/create.sql
# create user
mysql -h $RDS_ENDPOINT -u $RDS_USERNAME -p $RDS_PASSWORD < work/sql/auth.sql
# insert data
mysql -h $RDS_ENDPOINT -u $RDS_USERNAME -p $RDS_PASSWORD < work/sql/insert.sql

python work/app/index.py # run application
```

다음 명령어를 실행해서 데이터베이스를 설정해준 후 어플을 실행합니다.

<Copied RDS Endpoint>는 복사해둔 RDS의 endpoint, 나머지 <RDS User Name>과 <RDS Password>는 RDS 생성시 설정한 것을 입력해줍니다.

☐	Table Name	Rows
☐	INGREDIENT	52
☐	MENU	84
☐	NUTRITION	52
☐	ORDERING	6
☐	ORDERING_ITEM	12
☐	RECIPE	30
☐	RECIPE_INGREDIENT	176
☐	STOCK	48
☐	SUPPLIER	6
☐	SUPPLIER_CONTRACT	11

Figure 4.16: 대략 이 정도의 데이터가 들어갑니다.

이제 브라우저에서 복사해둔 퍼블릭 IPv4 DNS의 8000번 포트로 접속하면, 아래와 같은 화면이 나옵니다.



Figure 4.17: HTTP 경고

네트워크 암호화 설정을 안해준 관계로 해당 경고창이 뜹니다.
그냥 HTTP 사이트로 계속 진행하겠습니다.

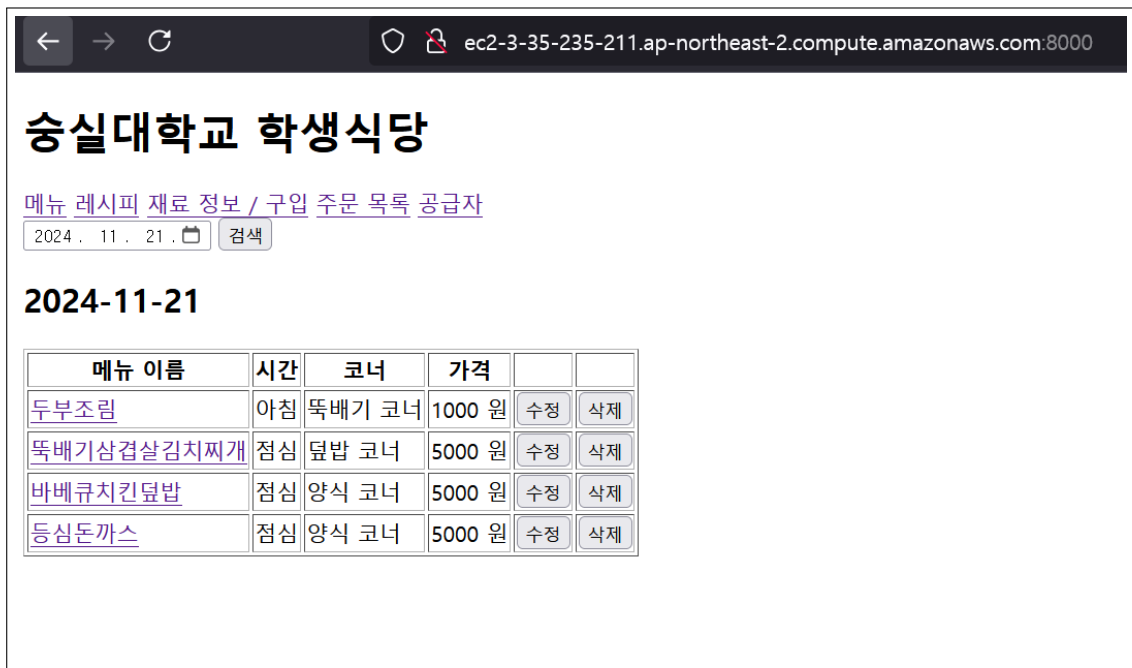


Figure 4.18: 브라우저 화면

정상적으로 실행되는 모습. 이 [페이지](#)에서 확인 가능합니다.

4.5 Application Code

Python의 flask 라이브러리를 사용하여 웹 서버를 구축했습니다.

구체적인 코드 구조는 [Remote 저장소](#)에서 확인 가능합니다.

Menu 페이지

이 페이지에서는 날짜별 학생식당 메뉴를 확인하고 수정할 수 있습니다.

The screenshot displays the Menu page in a web browser, the corresponding HTML template, and the Python controller code.

Web Browser View: The browser shows the URL `ec2-3-35-235-211.ap-northeast-2.compute.amazonaws.com:8000`. The page title is "송실대학교 학생식당". Below the title, there are links for "메뉴", "레시피", "재료 정보", "구입 주문 목록", and "공급자". The date "2024-11-26" is displayed. A table lists menu items with columns for "메뉴 이름", "시간", "코너", "가격", "수정", and "삭제".

메뉴 이름	시간	코너	가격	수정	삭제
뚝배기불고기	아침	뚝배기 코너	1000 원	수정	삭제
우삼겹두부찌개	점심	양식 코너	5000 원	수정	삭제
치킨가라아게동	점심	뚝배기 코너	5000 원	수정	삭제
등심돈까스	점심	양식 코너	5000 원	수정	삭제

app/templates/menu.html: This template defines the navigation links and the form for updating the menu date.

```
<h1 class='title'>송실대학교 학생식당</h1>
<nav>
  <a href="{{ url_for('home') }}">메뉴</a>
  <a href="{{ url_for('recipe') }}">레시피</a>
  <a href="{{ url_for('ingredients') }}">재료 정보 / 구입</a>
  <a href="{{ url_for('orders') }}">주문 목록</a>
  <a href="{{ url_for('suppliers') }}">공급자</a>
</nav>
<form action="/" method="GET">
  <input required type="date" id="dateInput" name="date" value="{{ date if date else '' }}">
  <button type="submit">검색</button>
</form>
```

app/template/menu.html: This template renders the menu items into an HTML table.

```
<table border="1">
  <tr>
    <th>메뉴 이름</th>
    <th>시간</th>
    <th>코너</th>
    <th>가격</th>
    <th>수정</th>
    <th>삭제</th>
  </tr>
  <tr>
    <td>{{ row['Name'] }}</td>
    <td>{{ row['Term'] }}</td>
    <td>{{ row['Corner'] }}</td>
    <td>{{ row['Price'] | round }} 원</td>
    <td>
      <form action="/" method="GET">
        <input type="hidden" name="modify" value="{{ row['MenuID'] }}">
        <input type="hidden" name="date" value="{{ row['Date'] }}">
        <button type="submit">수정</button>
      </form>
    </td>
    <td>
      <form onsubmit="deleteMenu(event, '{{ row['MenuID'] }}')">
        <button type="submit">삭제</button>
      </form>
    </td>
  </tr>
</table>
```

app/index.py: This code handles the GET request for the menu page, fetching the menu data from the database and rendering the template.

```
def get(self):
    try:
        target_date = request.args.get('date', date.today().strftime('%Y-%m-%d'))
        modify_id = request.args.get('modify', '')
        data = Database.fetch_data("""
            SELECT m.*, r.Name, r.Cornet
            FROM MENU m
            JOIN RECIPE r
            ON m.RecipeID = r.RecipeID
            WHERE Date = %s;
            """, (target_date,))
        recipe_data = Database.fetch_data("""
            SELECT RecipeID, Name FROM RECIPE;
            """)
        return render_template('menu.html',
                               date=target_date,
                               data=data,
                               modify=modify_id,
                               recipe_data=recipe_data), 200
    except Exception as e:
        return str(e), 500
```

Figure 4.19: 메뉴 화면(확대 시 선명하게 볼 수 있습니다)

메뉴 이름의 하이퍼링크를 클릭하면 해당 메뉴의 레시피 정보를 확인할 수 있습니다.

수정과 삭제버튼은 각각 record를 수정하고 삭제할 수 있는 기능을 제공합니다.

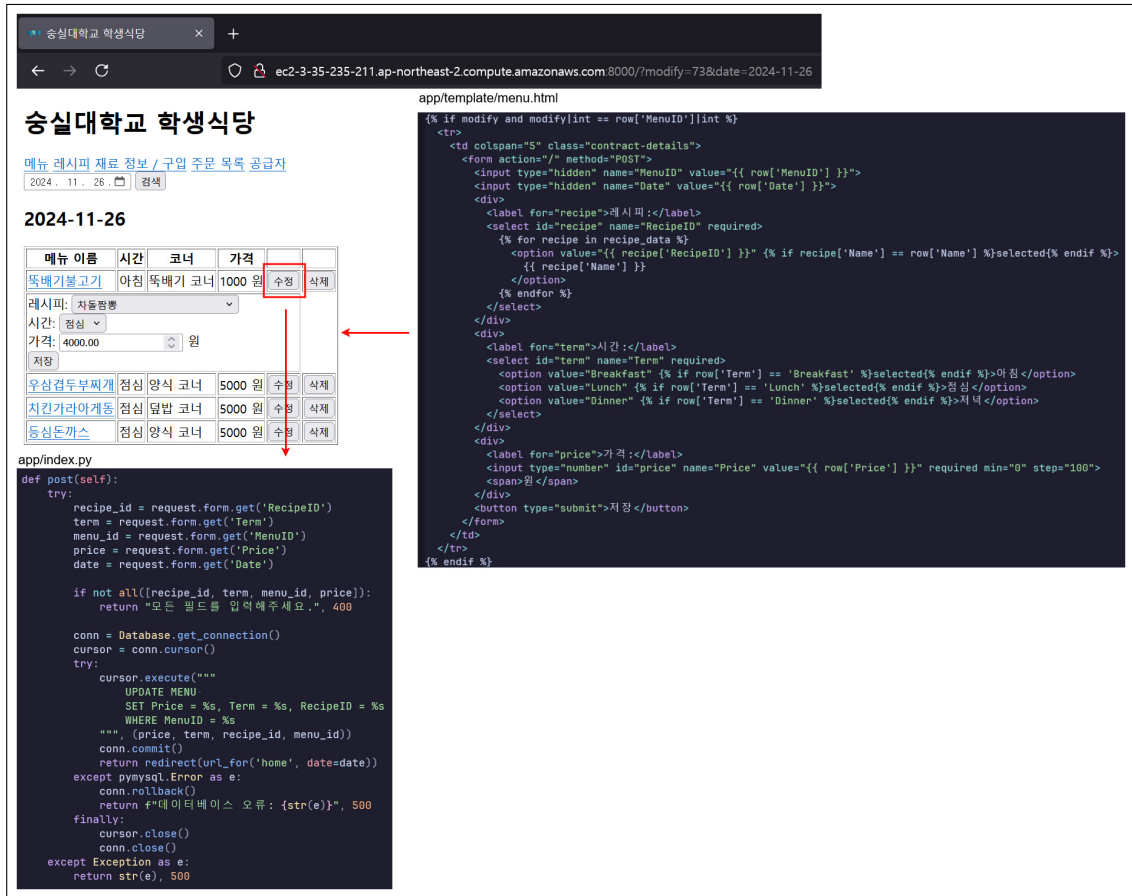


Figure 4.20: Update 로직

수정 버튼을 누르면 해당 레코드의 정보를 수정할 수 있는 form이 나타납니다. 원하는 값을 넣은 후, 저장을 누르면 수정이 완료됩니다.

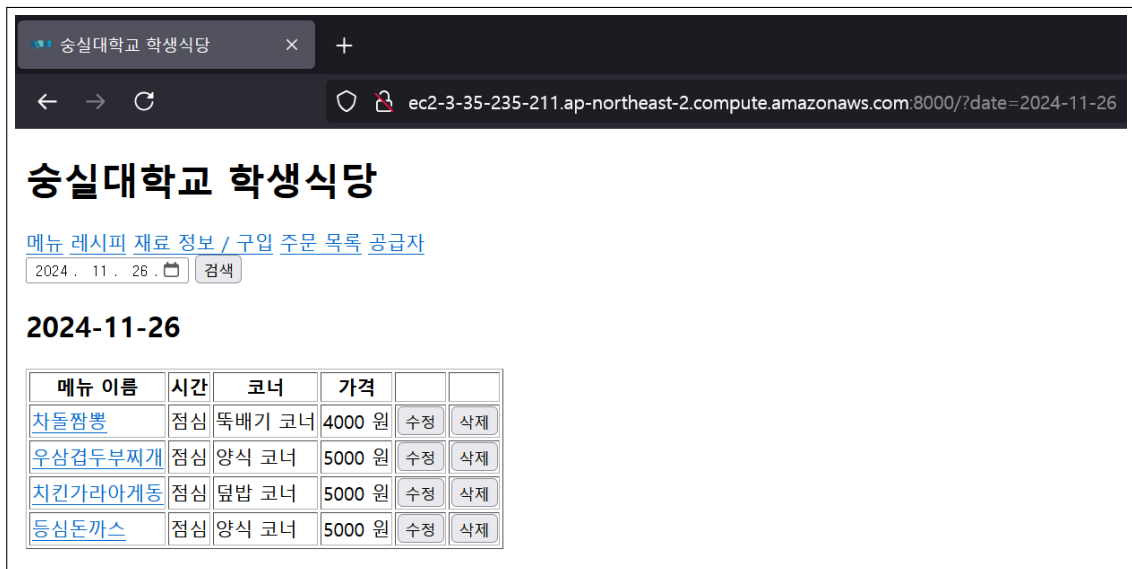


Figure 4.21: 수정이 완료된 모습. 저장을 눌러야 적용됩니다.

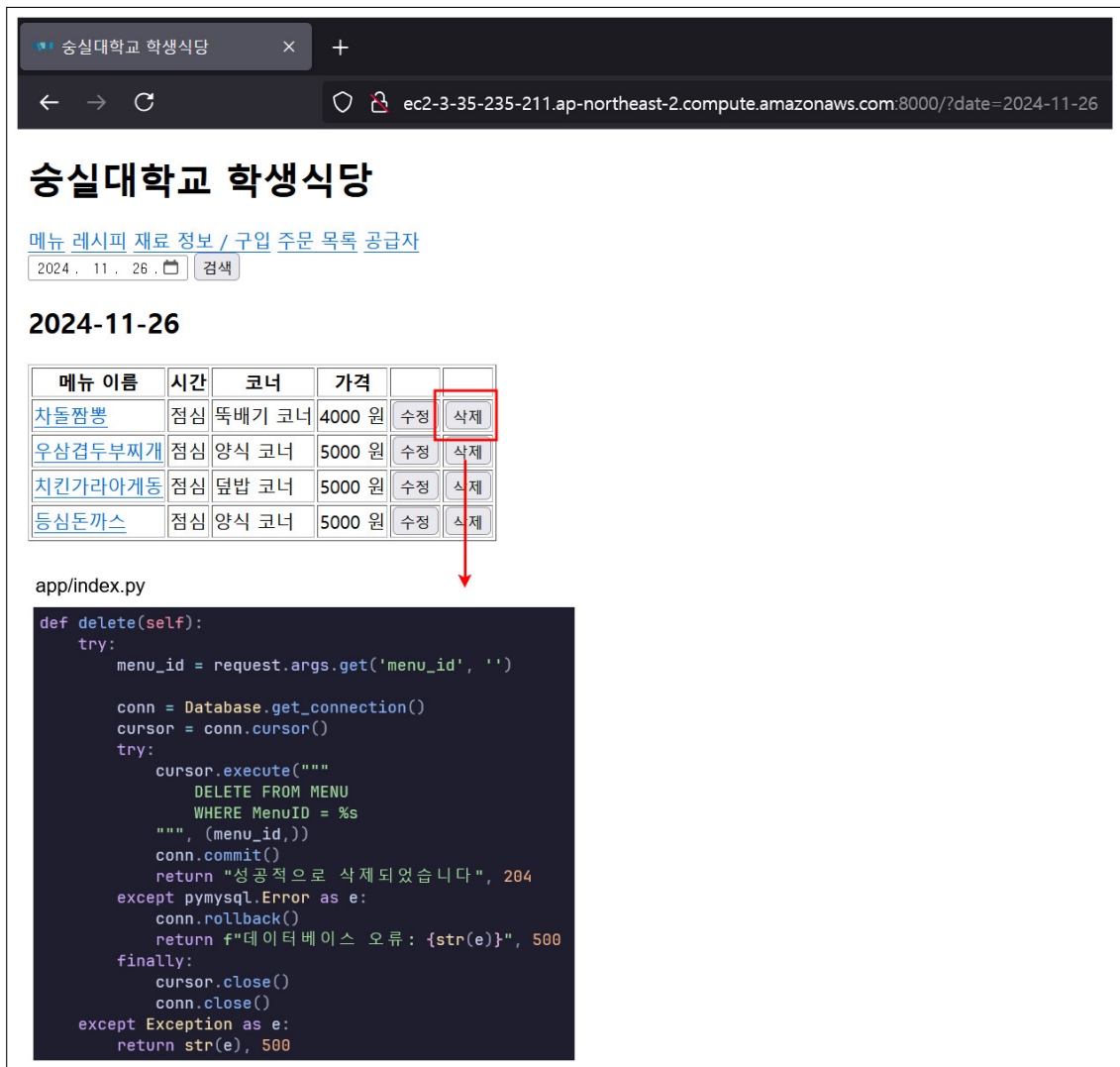


Figure 4.22: Delete 로직

삭제 버튼을 누르면 해당 레코드가 삭제됩니다.

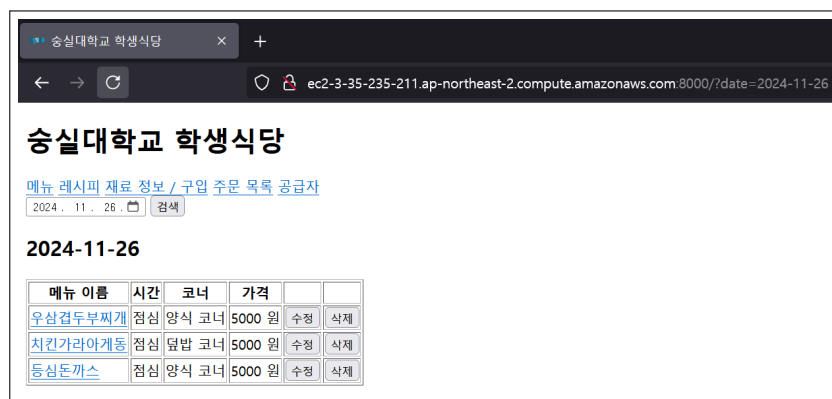
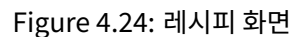


Figure 4.23: 삭제가 완료된 모습

이 페이지에서는 레시피 정보를 확인하고, 해당 레시피를 원하는 날짜의 메뉴에 추가할 수 있습니다.

확인 가능한 레시피 정보는 레시피 이름, instructions, 재료 사용량, 사용 재료 재고, 영양 정보입니다.



33

메뉴 등록하기

시간대: 점심 ▼

날짜: 2024 . 11 . 24 . 📅

가격: 5000 ▼

메뉴 등록

Figure 4.25: 메뉴 등록 폼

←

→

🔄

🔒

ec2-3-35-235-211.ap-northeast-2.compute.amazonaws.com:8000/?date=2024-11-24

승실대학교 학생식당

[메뉴 레시피](#)
[재료 정보](#)
[구입 주문 목록](#)
[공급자](#)

2024 . 11 . 24 . 📅

2024-11-24

메뉴 이름	시간	코너	가격		
새우까스	점심	양식 코너	5000 원	수정	삭제

Figure 4.26: 메뉴 등록 완료

Ingredient 페이지

이 페이지에서는 재료 정보를 확인하고 재료 구매 주문을 만들 수 있습니다.

승실대학교 학생식당

ec2-3-35-235-211.ap-northeast-2.compute.amazonaws.com:8000/ingredients

승실대학교 학생식당

메뉴 레시피 재료 정보 / 구입 주문 목록 공급자

주문 수량	재료 이름	수량	가격	공급 업체	사용 가능 재고	구매 가능 여부
☐	토마토	150 kg	200000 원	농협	4650.00 kg	구매 가능
☐	고추장	80 kg	200000 원	농협	0 kg	구매 가능
☐	깻잎	50 kg	200000 원	농협	21550.00 kg	구매 가능
☐	설탕	100 kg	200000 원	오뚜기	0 kg	구매 가능
☐	쌀	500 kg	200000 원	농협	240500.00 kg	구매 가능
☐	치즈	100 kg	200000 원	용주내 목장	20100.00 kg	구매 가능
☐	달걀	100 kg	200000 원	오뚜기	29100.00 kg	구매 가능
☐	닭고	150 kg	200000 원	농협	15150.00 kg	구매 가능
☐	참기름	50 L	200000 원	오뚜기	6000.00 L	구매 가능
☐	물엿	80 kg	200000 원	오뚜기	21280.00 kg	구매 가능
☐	마요네즈	80 kg	200000 원	오뚜기	0 kg	구매 가능
☐	돈까스 소스	100 L	200000 원	오뚜기	0 L	구매 가능
☐	후추	20 kg	100000 원	오뚜기	2940.00 kg	구매 가능
☐	토마토 소스	100 L	200000 원	오뚜기	41800.00 L	구매 가능
☐	소고기	300 kg	300000 원	용주내 목장	40200.00 kg	구매 가능
☐	치즈볶음	200 kg	300000 원	용주내 목장	8600.00 kg	구매 가능

app/template/ingredients.html

```

<table border="1">
  {% if data %}
    <tr>
      <th>주문 수량</th>
      <th>재료 이름</th>
      <th>수량</th>
      <th>가격</th>
      <th>공급 업체</th>
      <th>사용 가능 재고</th>
      <th>구매 가능 여부</th>
    </tr>
    {% for row in data %}
      <tr class="table_item">
        <td>
          {% if row['PurchaseStatus'] == 'Available' %}
            <input type="checkbox" name="{{ row['id'] }}" class="order-quantity" data-ingredient-id="{{ row['IngredientID'] }}" data-cost="{{ row['Cost'] }}" />
          {% endif %}
        </td>
        <td class="table_item_name">
          <a href="{{ url_for('ingredients', id=row['IngredientID']) }}">{{ row['IngredientID'] }}</a>
          <a href="{{ url_for('suppliers', id=row['SupplierID']) }}">{{ row['SupplierID'] }}</a>
        </td>
        <td class="table_item_quantity">{{ row['Quantity'] | round }} {{ row['Unit'] }}</td>
        <td class="table_item_cost">{{ row['Cost'] | round }} 원</td>
        <td class="table_item_price">
          <a href="{{ url_for('suppliers', id=row['SupplierID']) }}">{{ row['SupplierID'] }}</a>
        </td>
        <td class="table_item_total">{{ row['TotalStock'] | default(0, true) | round }} {{ row['Unit'] }}</td>
        <td class="table_item_corner">
          {% if row['PurchaseStatus'] == 'Available' %}
            구매 가능
          {% else %}
            구매 불가
          {% endif %}
        </td>
      </tr>
    </for>
  {% else %}
    <tr>
      <td colspan="7">구매 가능</td>
    </tr>
  {% endif %}
</table>

```

app/index.py

```

def get(self):
  try:
    ingredient_id = request.args.get('id', '')
    nutrition_id = request.args.get('nutrition', '')
    data = Database.fetch_data("""
      SELECT
        i.*,
        (
          SELECT NAME
            FROM SUPPLIER s
           WHERE s.SupplierID = i.SupplierID
        ) AS SupplierName,
        SUM(st.StockQuantity * i.Quantity) AS TotalStock,
        CASE
          WHEN EXISTS (
            SELECT SupplierID
              FROM SUPPLIER_CONTRACT sc
             WHERE i.SupplierID = sc.SupplierID
            AND NOW() BETWEEN sc.StartDate AND sc.EndDate
          ) THEN 'Available'
          ELSE 'Not Available'
        END AS PurchaseStatus
      FROM INGREDIENT i
     LEFT JOIN STOCK st
       ON i.IngredientID = st.IngredientID
     AND st.ExpiryDate > NOW()
     GROUP BY i.IngredientID
     ORDER BY PurchaseStatus;
    """)
    if ingredient_id == '' and nutrition_id == '':
      return render_template('ingredients.html', data=data), 200
    else:
      return render_template('ingredients.html', data=data), 200
  except:
    return render_template('ingredients.html', data=data), 200

```

Figure 4.27: 재료 화면

재료 이름을 클릭할 경우 재료의 영양 정보, 재고량을 확인할 수 있습니다.

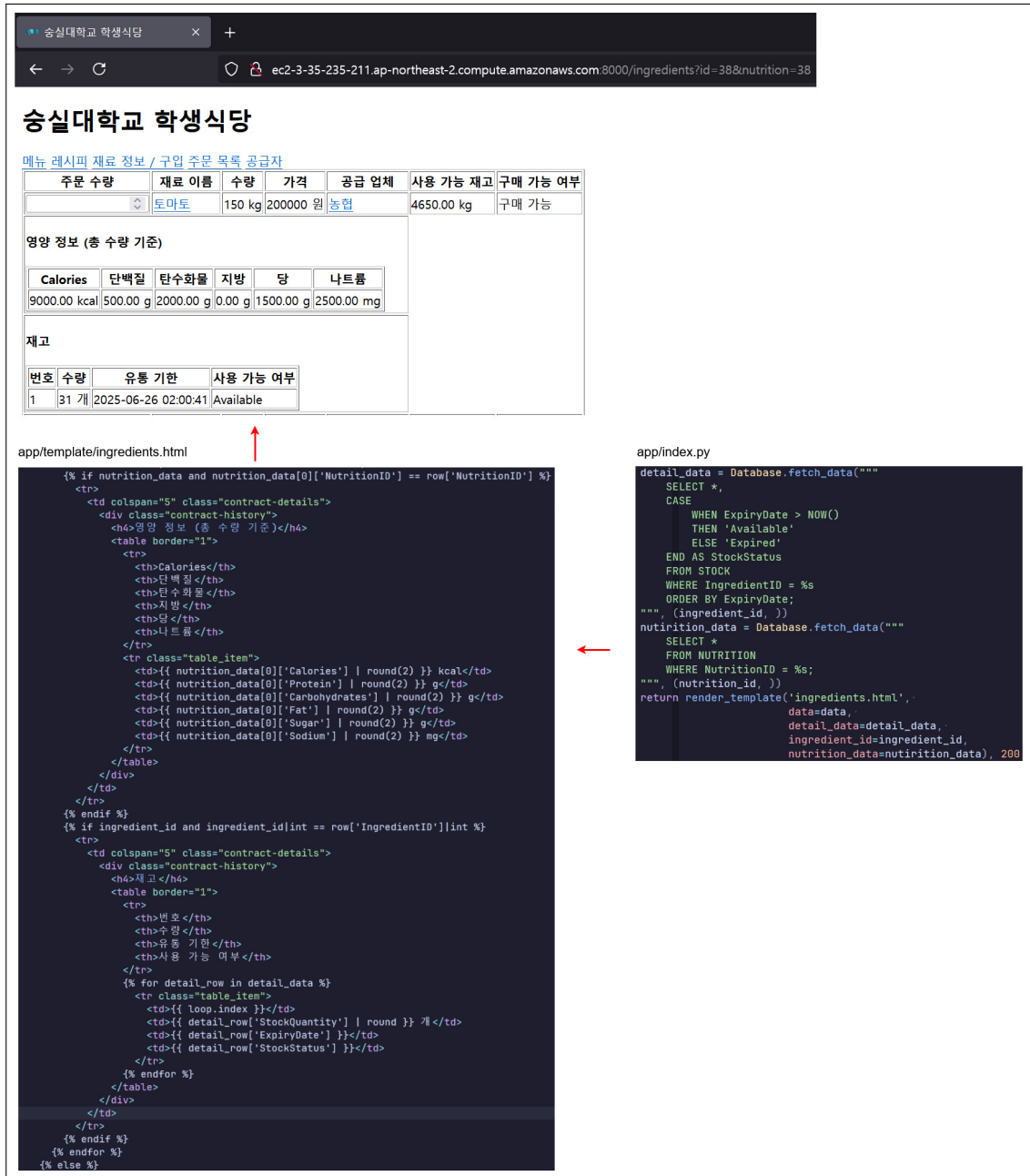


Figure 4.28: 재료 세부 정보

재료 주문을 만들어 보겠습니다. 재료의 좌측에 보이는 입력란에 원하는 수량을 입력한 후 주문 버튼을 누릅니다. 주문 버튼은 맨 아래에 있습니다. 계약이 만료된 공급업체의 재료는 입력란이 비활성화 되어 있습니다.

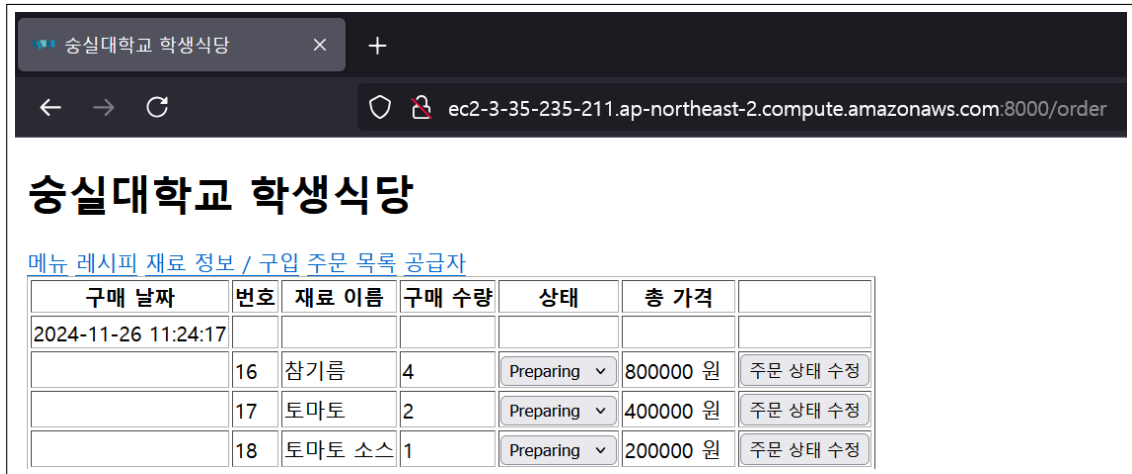


Figure 4.30: 주문 완료

Ordering 페이지

이 페이지에서는 주문 정보를 확인하고 주문 상태를 업데이트할 수 있습니다.

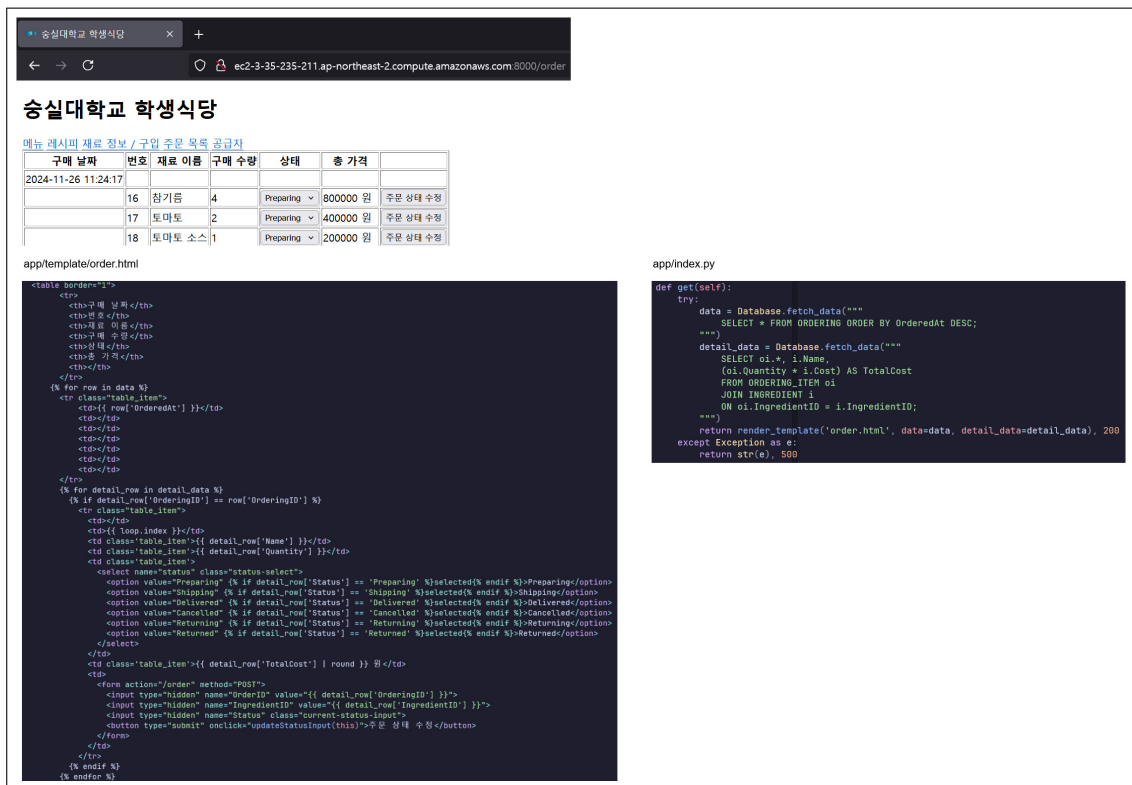


Figure 4.31: 주문 화면

주문 상태는 아래의 그림과 같이 드롭다운 메뉴를 통해 업데이트 할 수 있습니다.

송실대학교 학생식당 × +						
← → ↺ ec2-3-35-235-211.ap-northeast-2.compute.amazonaws.com:8000/order						
송실대학교 학생식당						
메뉴 레시피 재료 정보 / 구입 주문 목록 공급자						
구매 날짜	번호	재료 이름	구매 수량	상태	총 가격	
2024-11-26 11:24:17						
	16	참기름	4	Preparing ▾	800000 원	주문 상태 수정
	17	토마토	2	Preparing	400000 원	주문 상태 수정
	18	토마토 소스	1	Shipping	200000 원	주문 상태 수정
2024-11-22 22:25:24				Delivered		
	13	치즈	2	Cancelled	400000 원	주문 상태 수정
	14	참기름	3	Returning	600000 원	주문 상태 수정
	15	토마토	4	Returned	800000 원	주문 상태 수정
				Preparing ▾		
2023-11-15 02:44:56						

Figure 4.32: 주문 상태 업데이트

송실대학교 학생식당

×

+

←

→

↻

ec2-3-35-235-211.ap-northeast-2.compute.amazonaws.com:8000/order

송실대학교 학생식당

메뉴 레시피 재료 정보 / 구입 주문 목록 공급자

구매 날짜	번호	재료 이름	구매 수량	상태	총 가격	
2024-11-26 11:24:17						
	16	참기름	4	Delivered ▾	800000 원	주문 상태 수정
	17	토마토	2	Preparing ▾	400000 원	주문 상태 수정
	18	토마토 소스	1	Preparing ▾	200000 원	주문 상태 수정

app/index.py

```

def post(self):
    try:
        status = request.form.get('Status')
        order_id = request.form.get('OrderID')
        ingredient_id = request.form.get('IngredientID')

        if not all([status, order_id, ingredient_id]):
            return f"모든 필드를 입력해주세요.", 400

        conn = Database.get_connection()
        cursor = conn.cursor()
        try:
            cursor.execute("""
                UPDATE ORDERING_ITEM
                SET Status = %s
                WHERE OrderingID = %s
                AND IngredientID = %s;
            """, (status, order_id, ingredient_id))
            conn.commit()
            return redirect(url_for('order'))
        except pymysql.Error as e:
            conn.rollback()
            return f"데이터베이스 오류: {str(e)}", 500
        finally:
            cursor.close()
            conn.close()
    except Exception as e:
        return str(e), 500

```

Figure 4.33: 주문 상태 업데이트

Supplier 페이지

마지막 페이지입니다. 이 페이지에서는 공급업체 정보를 확인하고, 공급업체의 계약을 확인할 수 있습니다.

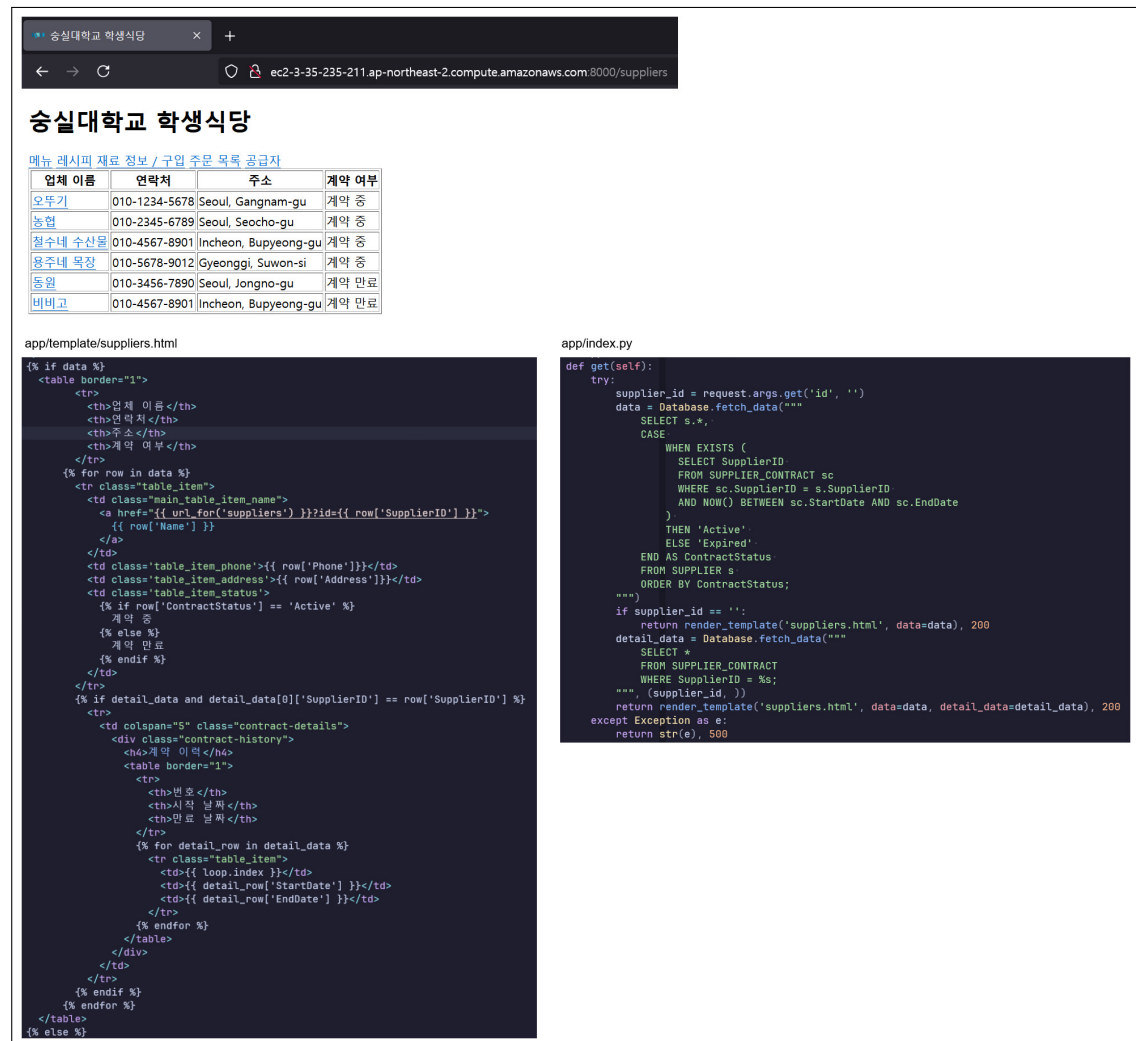
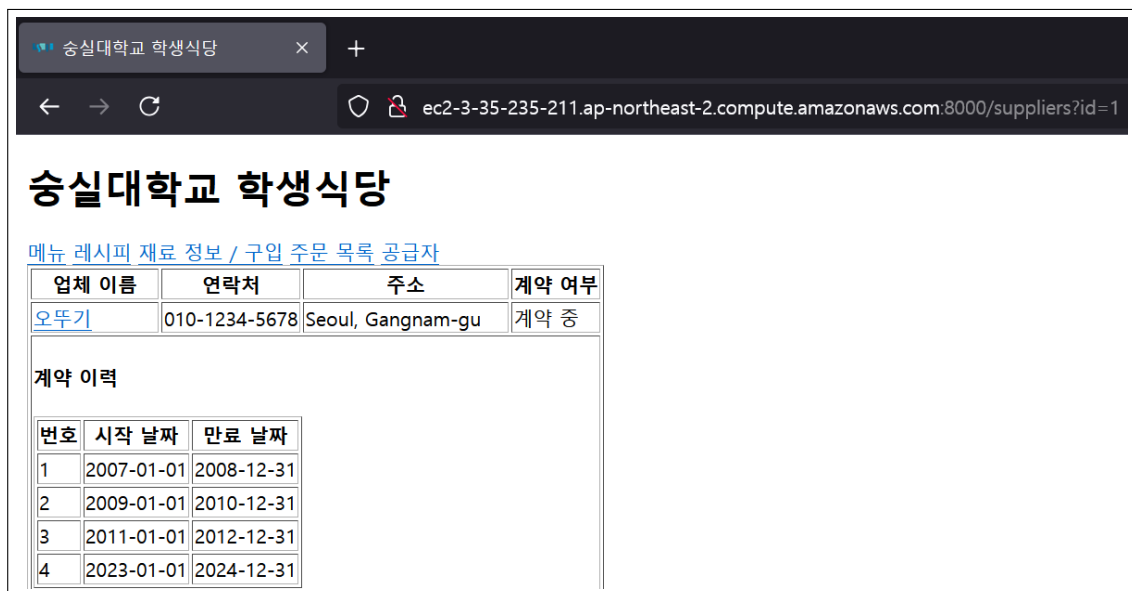


Figure 4.34: 공급 업체 화면

업체의 이름을 클릭하면 세부 계약 내역을 확인할 수 있습니다.



숭실대학교 학생식당

업체 이름	연락처	주소	계약 여부
오뚜기	010-1234-5678	Seoul, Gangnam-gu	계약 중
계약 이력			
번호	시작 날짜	만료 날짜	
1	2007-01-01	2008-12-31	
2	2009-01-01	2010-12-31	
3	2011-01-01	2012-12-31	
4	2023-01-01	2024-12-31	

Figure 4.35: 공급 업체 화면

DB administration

5.1 User User Management

유저 생성

```
CREATE USER 'kitchen_staff'@'%' IDENTIFIED BY 'secret123';
CREATE USER 'purchasing_manager'@'%' IDENTIFIED BY 'secret123';
CREATE USER 'supplier'@'%' IDENTIFIED BY 'secret123';
CREATE USER 'nutritionist'@'%' IDENTIFIED BY 'secret123';
```

목표한 사용자들을 생성합니다. 비밀번호는 'secret123'으로 설정해주었습니다.

권한 부여

```
-- kitchen staff
GRANT SELECT ON MENU TO 'kitchen_staff'@'%';
GRANT SELECT ON RECIPE_INGREDIENT TO 'kitchen_staff'@'%';
GRANT SELECT ON RECIPE TO 'kitchen_staff'@'%';
GRANT SELECT ON INGREDIENT TO 'kitchen_staff'@'%';
GRANT SELECT ON STOCK TO 'kitchen_staff'@'%';

-- purchasing manager
GRANT SELECT, INSERT ON ORDERING TO 'purchasing_manager'@'%';
GRANT SELECT, INSERT ON ORDERING_ITEM TO 'purchasing_manager'@'%';
GRANT SELECT ON INGREDIENT TO 'purchasing_manager'@'%';
GRANT SELECT ON STOCK TO 'purchasing_manager'@'%';
GRANT SELECT ON SUPPLIER TO 'purchasing_manager'@'%';
GRANT SELECT ON SUPPLIER_CONTRACT TO 'purchasing_manager'@'%';

-- supplier
GRANT SELECT ON ORDERING TO 'supplier'@'%';
GRANT SELECT, UPDATE ON ORDERING_ITEM TO 'supplier'@'%';
GRANT SELECT ON INGREDIENT TO 'supplier'@'%';
GRANT SELECT ON SUPPLIER_CONTRACT TO 'supplier'@'%';
GRANT SELECT ON SUPPLIER TO 'supplier'@'%';
```

```
-- nutritionist
GRANT SELECT ON RECIPE TO 'nutritionist'@'%';
GRANT SELECT ON RECIPE_INGREDIENT TO 'nutritionist'@'%';
GRANT SELECT ON NUTRITION TO 'nutritionist'@'%';
GRANT SELECT ON INGREDIENT TO 'nutritionist'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE ON MENU TO 'nutritionist'@'%';
```

권한 적용

마지막으로 FLUSH를 해서 모든 권한을 적용합니다.

```
FLUSH PRIVILEGES;
```

5.2 Backup

The screenshot displays the AWS Backup console interface. At the top, there are three summary cards: '백업' (Backup) showing '자동 백업' (Automatic Backup) with a 7-day retention, '최근 복원 시간' (Latest Restore Time) as November 22, 2024, 08:40 (UTC+09:00), and '리전으로 복제' (Replicate to Region) with a note about replicated automatic backups. Below these is the '스냅샷 (7)' (Snapshots (7)) section, which includes a search bar and a table of snapshots. The table has columns for '스냅샷 이름' (Snapshot Name), '스냅샷 생성 시간' (Snapshot Creation Time), '상태' (Status), and '스냅샷 유형' (Snapshot Type). All snapshots are in a '사용 가능' (Available) status and are of type '자동' (Automatic).

스냅샷 이름	스냅샷 생성 시간	상태	스냅샷 유형
rds:terraform-20241116010818384300000003-2024-11-16-01-11	November 16, 2024, 10:16 (UTC+09:00)	사용 가능	자동
rds:terraform-20241116010818384300000003-2024-11-16-07-06	November 16, 2024, 16:06 (UTC+09:00)	사용 가능	자동
rds:terraform-20241116010818384300000003-2024-11-17-07-06	November 17, 2024, 16:06 (UTC+09:00)	사용 가능	자동
rds:terraform-20241116010818384300000003-2024-11-18-07-06	November 18, 2024, 16:06 (UTC+09:00)	사용 가능	자동
rds:terraform-20241116010818384300000003-2024-11-19-07-06	November 19, 2024, 16:06 (UTC+09:00)	사용 가능	자동
rds:terraform-20241116010818384300000003-2024-11-20-07-06	November 20, 2024, 16:06 (UTC+09:00)	사용 가능	자동
rds:terraform-20241116010818384300000003-2024-11-21-07-06	November 21, 2024, 16:06 (UTC+09:00)	사용 가능	자동

Figure 5.1: AWS Backup 화면

RDS 기본 설정으로 백업이 설정되어 있습니다. 만약 백업 파일을 수동으로 생성하고 싶다면 EC2 인스턴스에서 다음 명령어를 실행합니다.

```
mysqldump -h $RDS_ENDPOINT -u $RDS_USERNAME -p$RDS_PASSWORD > backup.sql
```

만약 transaction log를 수동으로 생성하고 싶다면 다음의 단계를 따릅니다.

```
# set log retention period
CALL mysql.rds_set_configuration('binlog retention hours', 24);
SHOW BINARY LOGS; # check log file name
mysqlbinlog \
```

```

--read-from-remote-server \
--host=<RDS ENDPOINT> \
--port=3306 \
--user admin \
--password \
--raw \
--verbose \
--result-file=<path to store> \
<name of log file> # create log file

mysqlbinlog \
--base64-output=DECODE-ROWS \
--verbose \
<stored path> > binlog.sql # convert to sql format

```

우선 mysql에 접속해서 log파일의 보관기간을 수동으로 늘려줍니다. 그런 다음 log file의 파일명을 확인합니다. 그 후, mysqlbinlog 명령어를 통해 log file을 생성할 수 있습니다. log file은 사용하기 좋게 sql 형식으로 변환해줍니다.

5.3 Recovery

AWS 콘솔에서 복구를 선택할 수 있지만, 수동으로 생성한 백업 파일을 이용해서 복구할 수 있습니다. 다음의 명령어를 새로운 환경에서 실행합니다.

```

cat << EOF | mysql
CREATE DATABASE SOONGSIL_STUDENT_CAFETERIA_SCM;
EOF
mysql SOONGSIL_STUDENT_CAFETERIA_SCM < backup.sql
mysql SOONGSIL_STUDENT_CAFETERIA_SCM < binlog.sql

```

log file은 AWS RDS 환경이 아니면 에러가 발생할 수 있습니다. 하지만 AWS RDS 환경에서 log file을 직접적으로 사용할 권한이 없습니다. 결론적으로, 우리의 구현에서 log file은 사용할 수 없습니다.

5.4 Database Security

1. firewall

AWS 보안 그룹을 이용해서 RDS 인스턴스에 대한 접근을 제한했습니다.

2. Encryption store

암호화

☒ **암호화 활성화**
지정된 인스턴스를 암호화하려면 이 옵션을 선택합니다. AWS Key Management Service 콘솔을 사용하여 마스터 키 ID와 별칭이 생성된 후 해당 항목이 목록에 표시됩니다. [정보](#)

AWS KMS 키 [정보](#)

(default) aws/rds ▼

Figure 5.2: AWS RDS 생성 화면에서 암호화 설정

RDS 인스턴스 기본 설정으로 암호화 설정을 활성화했습니다.