E-4 조 / 주행 1

에스프레소 카페 주문 서비스 로봇

R&R

윤지원

- 5nside@naver.com

- 기획서 및 화면 설계
- UI 구현

권수빈

suprmo8007@gmail.com

- DB 설계
- 주문 기능 구현

배정빈

jnbn135@naver.com

- 로봇 제어
- 맵 제작

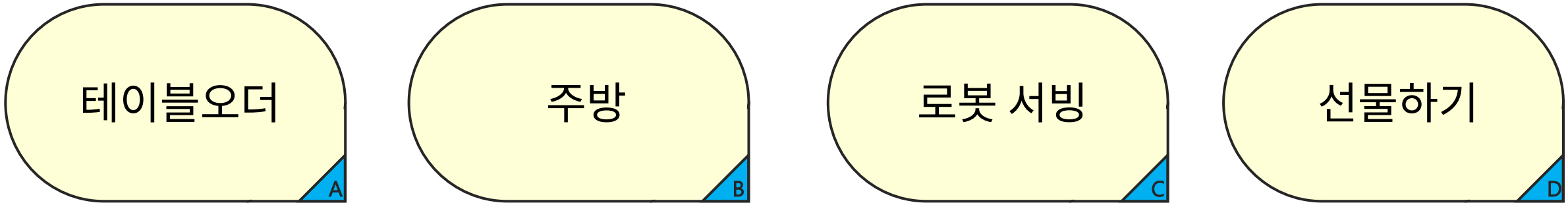
목차

1-1) 프로젝트 개요

프로젝트 명 : 에스프레소 카페 주문 서비스 로봇

목적 : 자동화된 주문 - 서빙 시스템 구축

주요기능 기획



테이블에서의
키오스크를
이용한 주문

주방에서의 실시간
주문 관리

로봇을 통한
자동서빙

테이블간
선물하기 기능

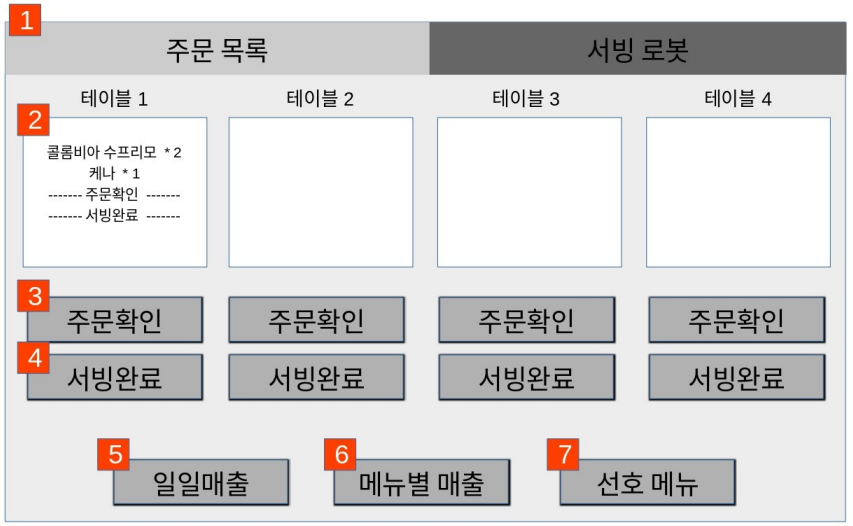
1-2) 화면

GUI 이름	화면 이름
테이블오더 (2/3)	에스프레소 주문화면

1. 버튼 클릭시 2 번 창에 표시 됩니다
2. 사용자가 주문한 에스프레소의 종류와 개수가 표시되는 창입니다
3. 합계 금액이 표시되는 창입니다
4. 주문한 내용을 초기화 하는 버튼 입니다
5. 클릭시 주방 주문목록 화면으로 정보가 넘어갑니다

< 테이블 노드 화면 설계 >

GUI 이름	화면 이름
주방 (1/2)	주문목록 화면



1. 탭을 클릭하면 원하는 화면으로 이동할 수 있습니다
2. 사용자가 주문한 목록이 표시되는 창입니다
3. **주문확인 버튼**
   사용자가 주문한 목록 완료시 주문목록 표시 창에 완료로 표시합니다
4. **서빙완료 버튼**
   서빙완료시 주문목록 표시 창에 완료로 표시합니다
5. **일일매출 버튼**
   일일 매출을 표시해줍니다
6. **메뉴별매출 버튼**
   일일 매출을 표시해줍니다
7. **선호 메뉴 버튼**
   가장 많이 주문한 선호 메뉴를 표시해줍니다

< 주방 노드 화면 설계 >

UI/UX 디자인 화면설계 진행

1-3) 시스템 구조

ROS2 노드 구성

customer_v4.py 4 × kitchen_v4.py 9+

home > yjw > Desktop > example_project_ws > src > example_project > example_project > customer_v4.py > ...

```python
import sys
from PyQt5.QtWidgets import (QApplication, QMainWindow, QWidget, QVBoxLayout, QMessageBox,
                             QHBoxLayout, QLabel, QPushButton, QTextEdit, QStackedWidget, QSpinBox, QLineEdit)
from PyQt5.QtCore import Qt, pyqtSignal
from PyQt5.QtGui import QPixmap, QImage
from ament_index_python.packages import get_package_share_directory
import rclpy
from rclpy.node import Node
from std_msgs.msg import String
import threading
import signal
import os
# ---------- DB --------------
import psycopg2
from db_utils import get_connection, return_connection

class TableOrderNode(Node):
    def __init__(self, table_number):
        super().__init__(f'table_order_node_{table_number}')
        self.publisher = self.create_publisher(String, 'table_orders', 10)

        self.table_number = table_number
        # beans_data를 메서드 호출로 초기화
        self.beans_data = self.get_beans_info()

    def send_order(self, order_data):
        msg = String()
        msg.data = f"테이블 {self.table_number}: {order_data}"
        self.publisher.publish(msg)
        self.get_logger().info(f'Published order: {msg.data}')

    # def send_order(self, order_data):
    #     # order_data는 딕셔너리 형태로 전달
    #     msg = String()
    #     # 딕셔너리를 문자열로 변환하여 전송
    #     msg.data = f"{self.table_number}|{order_data['total_price']}|{order_data['orders']}"
    #     self.publisher.publish(msg)
    #     self.get_logger().info(f'Published order: {msg.data}')

    def kitchen_message_callback(self, msg):
        if self.emit_signal is not None:
            self.emit_signal(msg.data)

    def set_emit_signal(self, emit_func):
        self.emit_signal = emit_func
```

< 테이블 노드 >

customer_v4.py 9+ × kitchen_v4.py 9+

home > yjw > Desktop > example_project_ws > src > example_project > example_project > customer_v4.py > ...

```python
import sys
from PyQt5.QtWidgets import *
from PyQt5.QtCore import Qt, pyqtSignal
import rclpy
from rclpy.node import Node
from std_msgs.msg import String
import threading
import signal
from datetime import datetime
import sqlite3
# ---------- DB --------------
import psycopg2
import json
import ast
from db_utils import get_connection, return_connection

class KitchenOrderNode(Node):
    def __init__(self):
        super().__init__('kitchen_order_node')
        self.subscription = self.create_subscription(
            String,
            'table_orders',
            self.order_callback,
            10)
        self.publisher = self.create_publisher(String, 'kitchen_messages', 10)
        self.emit_signal = None

        # ---------- DB --------------
        # 카페 테이블 정보 가져오기
        self.tables_data = self.get_cafe_tables_info()
        self.get_logger().info(f'Cafe_tables data: {self.tables_data}')

    def order_callback(self, msg):
        self.get_logger().info(f"Received message: {msg.data}")
        if self.emit_signal is not None:
            try:
                # 테이블 번호와 주문 정보 분리
                table_info, order_data = msg.data.split(':', 1)
                table_number = table_info.replace('테이블 ', '').strip()

                # 주문 정보를 안전하게 파싱
                order_dict = ast.literal_eval(order_data.strip())

                # 주문 정보에 테이블 번호 추가
                order_dict['table_number'] = int(table_number)
```

< 주방 노드 >

1-3) 시스템

< 테이블 노드 >

[Node 1]

topic

[Node 1]

GUI

< 주방 노드 >

[Node 2]

topic
action

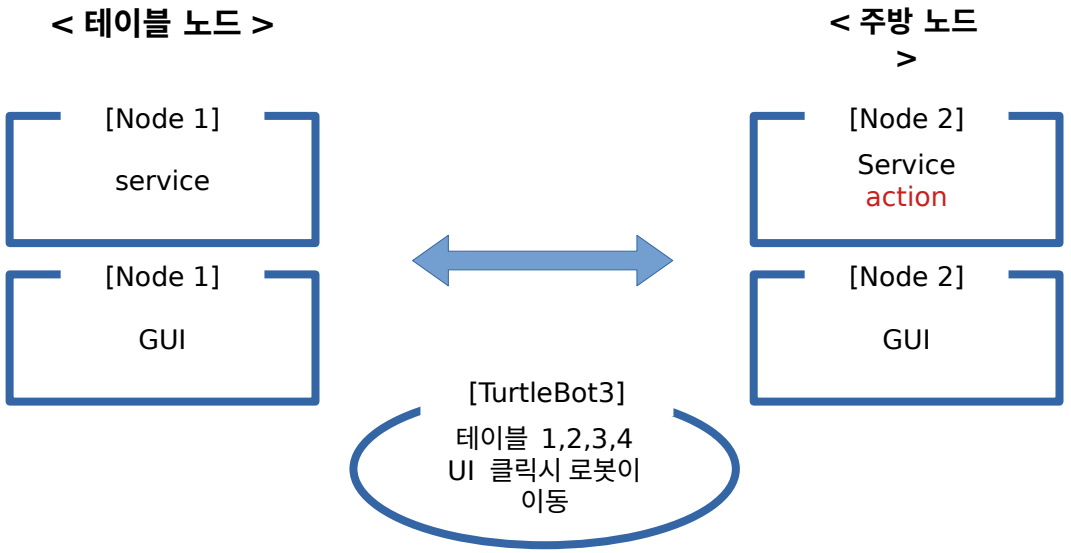
[Node 2]

GUI

[TurtleBot3]

테이블 1,2,3,4
UI 클릭시 로봇이 이동

1-3) 시스템

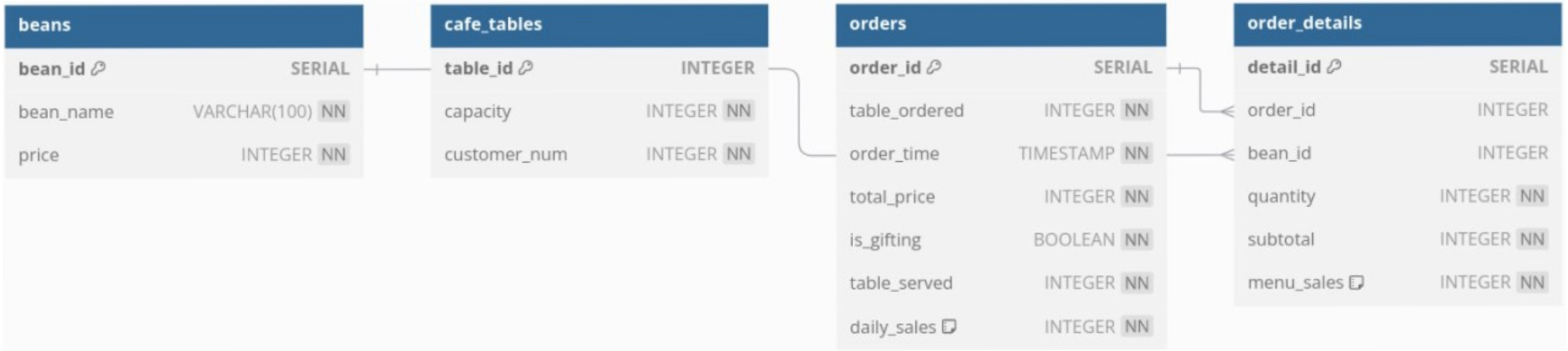


ROS2 노드 구성

1. 테이블 노드 (4 개 UI)
   주문 정보 발행
   UI 인터페이스 제공
   주문 상태 구독

2. 주방 노드 (1 개 UI)
   주문 정보 구독
   DB 연동 및 데이터 저장
   주문 상태 관리

1-3) 시스템 구조 -

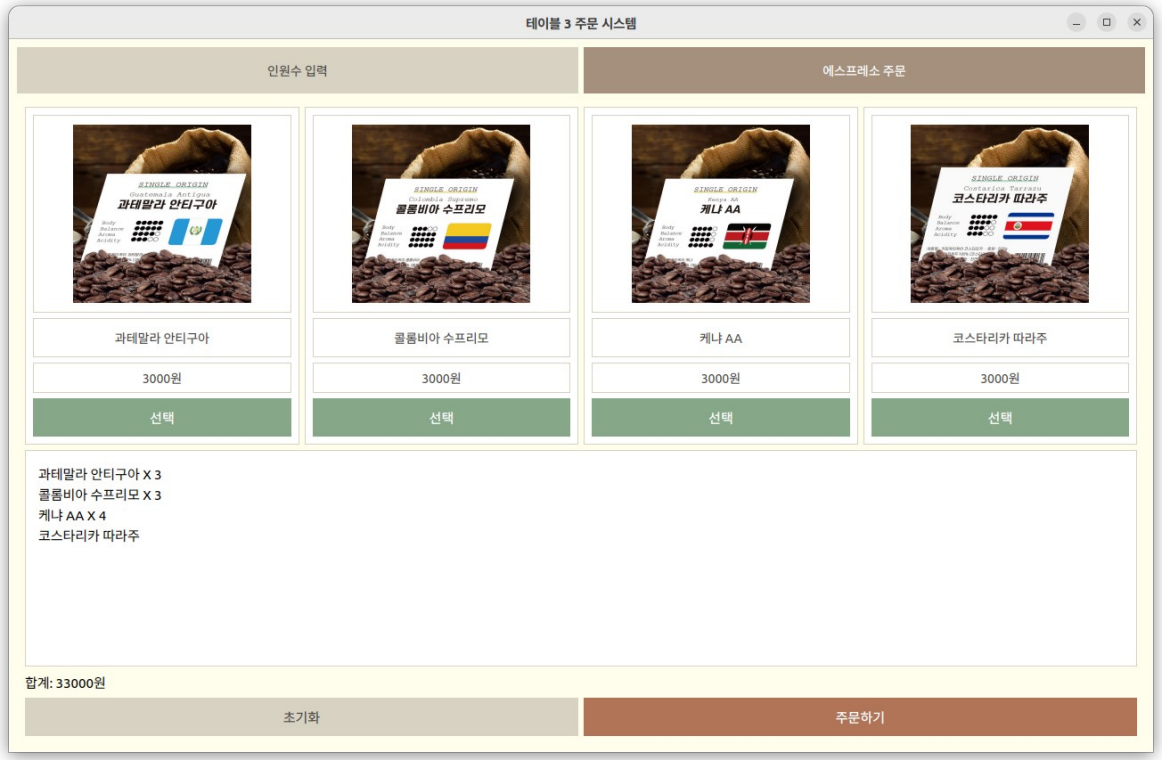

beans
bean_id SERIAL
bean_name VARCHAR(100) NN
price INTEGER NN
cafe_tables
table_id INTEGER
capacity INTEGER NN
customer_num INTEGER NN
orders
order_id SERIAL
table_ordered INTEGER NN
order_time TIMESTAMP NN
total_price INTEGER NN
is_gifting BOOLEAN NN
table_served INTEGER NN
daily_sales INTEGER NN
order_details
detail_id SERIAL
order_id INTEGER
bean_id INTEGER
quantity INTEGER NN
subtotal INTEGER NN
menu_sales INTEGER NN

2. 기획 및 화면 설계

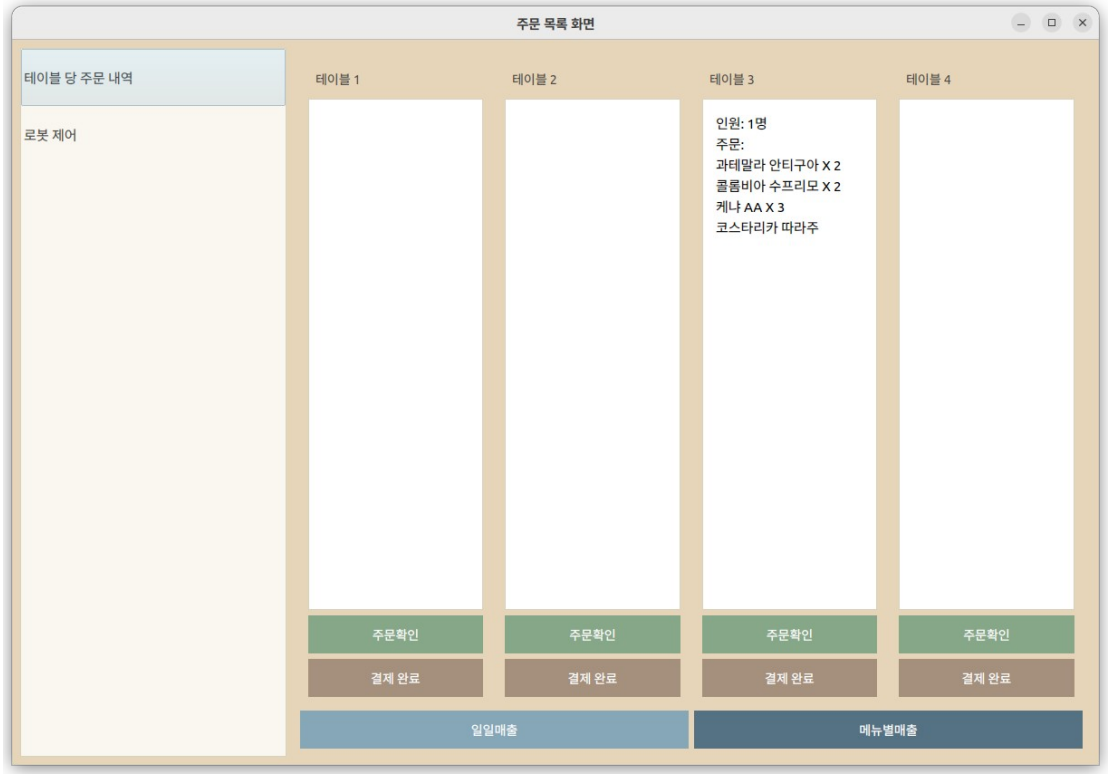
1) UI / UX 디자인

2) 시스템 구조

2-1) UX/UI 디자인

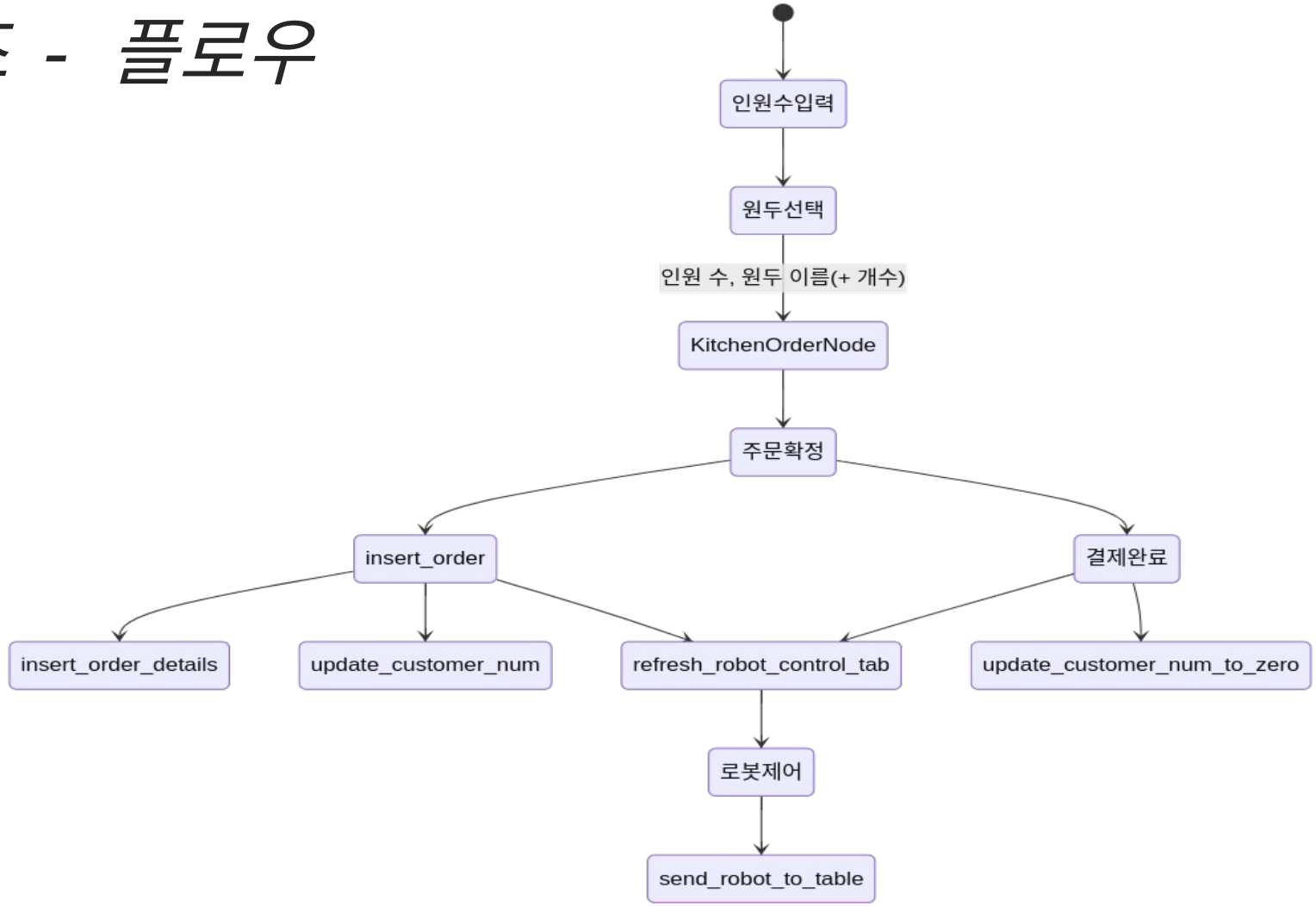


< 테이블 노드 >



< 주방 노드 >

2-2) 시스템 구조 - 플로우


인원수입력
원두선택
인원 수, 원두 이름(+ 개수)
KitchenOrderNode
주문확정
insert_order
결제완료
insert_order_details
update_customer_num
refresh_robot_control_tab
update_customer_num_to_zero
로봇제어
send_robot_to_table

2-2) 시스템 구조 – Table Order Node

1. 인원 수 입력 및 원두 선택

인원 수 , 원두 이름 , 개수 , 테이블당 총 금액 → Kitchen Order Node 로 보냄 (**Topic**)

```
def send_order(self, order_data):
    msg = String()
    msg.data = f"테이블 {self.table_number}: {order_data}"
    self.publisher.publish(msg)
    self.get_logger().info(f'Published order: {msg.data}')
```

2-2) 시스템 구조 – Kitchen Order Node

2. 주문 확정 (Service)

1) insert_order()

주문 번호 , 주문한 테이블 번호 , 주문 시각 , 원두 이름 ,

테이블 당 총 금액 , 선물용 주문 (T/F), 서빙받을 테이블 번호

→ orders 테이블로 insert

2-2) 시스템 구조 – Kitchen Order Node

```python
# orders ----------------------------------------------------------------------
def insert_order(self, order_data):
    conn = get_connection()
    try:
        with conn.cursor() as cur:
            cur.execute("""
                INSERT INTO orders (table_ordered, order_time, total_price, is_gifting, table_served)
                VALUES (%s, %s, %s, %s, %s)
                RETURNING order_id
            """, (
                order_data['table_number'],
                datetime.now(),
                order_data['total_price'],
                False,  # is_gifting
                order_data['table_number']  # table_served
            ))
            order_id = cur.fetchone()[0]
            conn.commit()
            return order_id
    except Exception as e:
        self.log_error(f"Error in insert_order: {e}")
        conn.rollback()
        return None
    finally:
        return_connection(conn)
```

2-2) 시스템 구조 – Kitchen Order Node

2. 주문 확정 (Service)

2) insert_order_details()

세부주문 번호 , 주문 번호 , 원두 번호 , 수량 , 소계

→ order_details 테이블로 insert

2-2) 시스템 구조 – Kitchen Order Node

```python
# order_details ---------------------------------------------------------------
def insert_order_details(self, order_id, orders):
    conn = get_connection()
    try:
        with conn.cursor() as cur:
            for item in orders:
                self.get_logger().info(f"Processing order item: {item}")  # 로그 추가
                try:
                    # 'X' 또는 'x'로 분리 시도
                    if ' X ' in item:
                        bean_name, quantity = item.split(' X ')
                    elif ' x ' in item:
                        bean_name, quantity = item.split(' x ')
                    else:
                        self.get_logger().error(f"Invalid order format: {item}")
                        continue

                    quantity = int(quantity)

                    # beans 테이블에서 bean_id와 price 조회
                    cur.execute("""
                        SELECT bean_id, price
                        FROM beans
                        WHERE bean_name = %s
                    """, (bean_name,))

                    bean_info = cur.fetchone()
                    if bean_info:
                        bean_id, price = bean_info
                        subtotal = price * quantity

                        cur.execute("""
                            INSERT INTO order_details (order_id, bean_id, quantity, subtotal)
                            VALUES (%s, %s, %s, %s)
                        """, (order_id, bean_id, quantity, subtotal))
                except ValueError as e:
                    self.get_logger().error(f"Error processing item {item}: {e}")
                    continue
            conn.commit()
    except Exception as e:
        self.get_logger().error(f"Error in insert_order_details: {e}")
        conn.rollback()
    finally:
        return_connection(conn)
```

2-2) 시스템 구조 – Kitchen Order Node

2. 주문 확정 (Service)

3) update_customer_num()

주문한 테이블 번호 , 입력받은 인원 수 → cafe_tables 테이블로 insert

2-2) 시스템 구조 – Kitchen Order Node

```python
def update_customer_num(self, table_id, customer_num):
    conn = get_connection()
    try:
        with conn.cursor() as cur:
            cur.execute("""
                UPDATE cafe_tables
                SET customer_num = %s
                WHERE table_id = %s
            """, (customer_num, table_id))
            conn.commit()
            self.get_logger().info(f"Updated customer_num for table {table_id} to {customer_num}")
    except psycopg2.Error as e:
        self.get_logger().error(f"Error updating customer number: {e}")
        conn.rollback()
    finally:
        return_connection(conn)
```

2-2) 시스템 구조 – Kitchen Order Node

2. 주문 확정 (Topic)

4) refresh_robot_control_tab()

customer_num(착석 중인 인원), capacity(수용 가능 인원)

← cafe_tables 로 부터 받아옴

2-2) 시스템 구조 – Kitchen Order Node

```
def update_table_buttons(self):
    for i, button in enumerate(self.table_buttons, 1):
        table_info = self.node.get_table_info(i)
        if table_info:
            customer_num, capacity = table_info
            button.setText(f"테이블 {i}\n({customer_num}/{capacity})")
            button.setStyleSheet("""
                QPushButton {
                    text-align: center;
                    padding: 10px;
                }
            """)

def refresh_robot_control_tab(self):
    self.update_table_buttons()
```

2-2) 시스템 구조 - DB

3. 결제 완료 (Service)

1) update_customer_num_to_zero()

인원수 = 0

→ cafe_tables 로 update

```python
def update_customer_num_to_zero(self, table_id):
    conn = get_connection()
    try:
        with conn.cursor() as cur:
            cur.execute("""
                UPDATE cafe_tables
                SET customer_num = 0
                WHERE table_id = %s
            """, (table_id,))
            conn.commit()
            self.get_logger().info(f"Updated customer_num for table {table_id} to 0")
    except psycopg2.Error as e:
        self.get_logger().error(f"Error updating customer number: {e}")
        conn.rollback()
    finally:
        return_connection(conn)
```

2-2) 시스템 구조

3. 결제 완료 (Service)

2) refresh_robot_control_tab()

customer_num(착석 중인 인원), capacity(수용 가능 인원)

← cafe_tables 로 부터 받아옴

- 로봇 제어 탭의 버튼 UI 에 표시되는 착석 중인 인원 수를 갱신
- 인원 수에 따라 서비스 쿠키 제공

```
def update_table_buttons(self):
    for i, button in enumerate(self.table_buttons, 1):
        table_info = self.node.get_table_info(i)
        if table_info:
            customer_num, capacity = table_info
            button.setText(f"테이블 {i}\n({customer_num}/{capacity})")
            button.setStyleSheet("""
                QPushButton {
                    text-align: center;
                    padding: 10px;
                }
            """)

def refresh_robot_control_tab(self):
    self.update_table_buttons()
```

2-2) 시스템 구조

4. 로봇 제어 (Action)

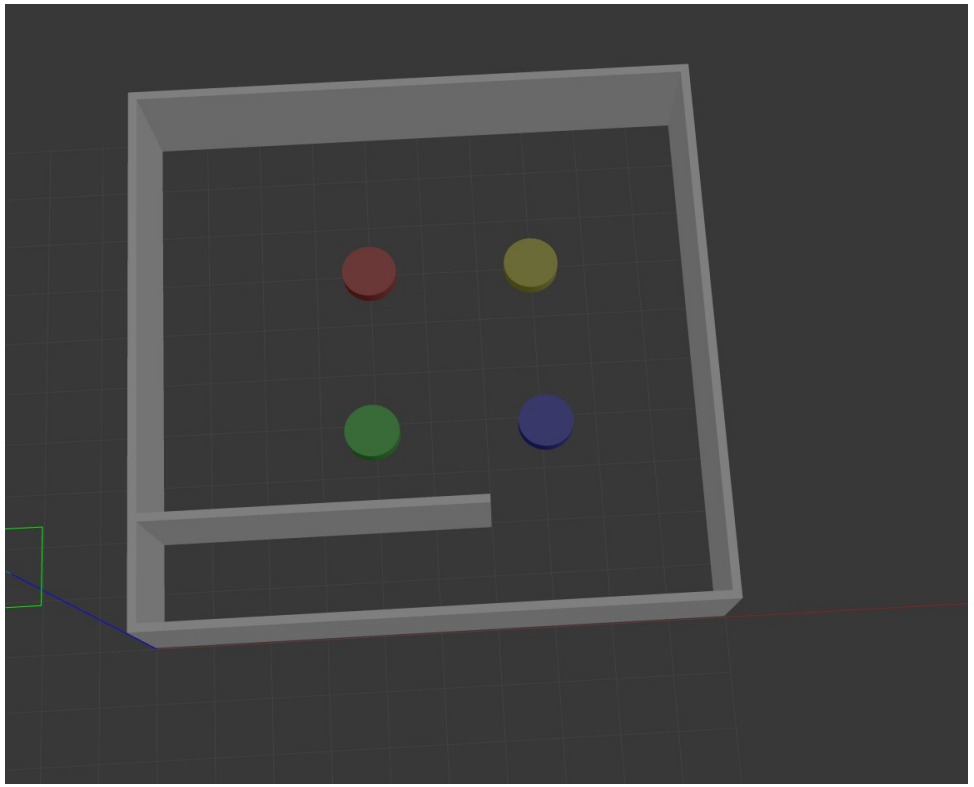
send_robot_to_table()

- nav2 를 이용해서 테이블 좌표로 이동시킴 (커피 및 쿠키 서빙)

```python
def send_robot_to_table(self, table_number):
    table_coordinates = {
        1: (1.0, 4.0, 0.0),
        2: (1.0, 7.0, 0.0),
        3: (4.0, 4.0, 0.0),
        4: (4.0, 7.0, 0.0)
    }

    if table_number not in table_coordinates:
        self.get_logger().error(f"Invalid table number: {table_number}")
        return

    x, y, z = table_coordinates[table_number]

    goal_pose = PoseStamped()
    goal_pose.header.frame_id = 'map'
    goal_pose.header.stamp = self.get_clock().now().to_msg()
    goal_pose.pose.position.x = x
    goal_pose.pose.position.y = y
    goal_pose.pose.position.z = z
    goal_pose.pose.orientation.w = 1.0

    goal_msg = NavigateToPose.Goal()
    goal_msg.pose = goal_pose

    self.nav_client.wait_for_server()
    self.get_logger().info(f"Sending robot to table {table_number}")
    self.nav_client.send_goal_async(goal_msg)
```

2-3) 맵 구현

GAZEBO 상 Building Editor 를 통해 기본적인 외형 건축물 설치



2-3) 맵 구현

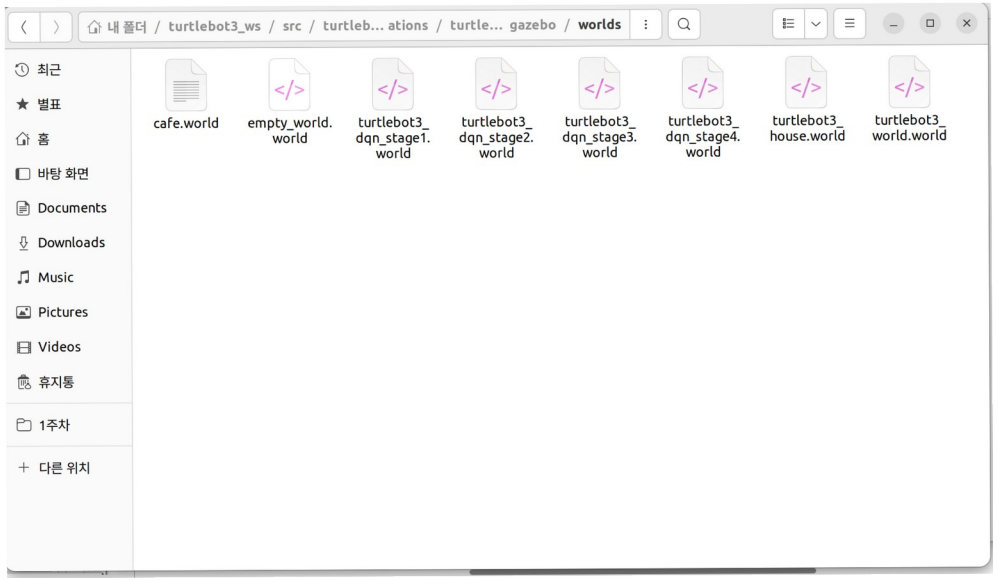
```
          <bounce/>
          <friction>
            <torsional>
              <ode/>
            </torsional>
            <ode/>
          </friction>
        </surface>
      </collision>
      <visual name='base_visual'>
        <pose>-0.032 0 0 0 -0 0</pose>
        <geometry>
          <mesh>
            <uri>model://turtlebot3_common/meshes/burger_base.dae</uri>
            <scale>0.001 0.001 0.001</scale>
          </mesh>
        </geometry>
      </visual>
      <self_collide>0</self_collide>
      <enable_wind>0</enable_wind>
      <kinematic>0</kinematic>
    </link>
    <link name='imu_link'>
      <sensor name='tb3_imu' type='imu'>
        <always_on>1</always_on>
        <update_rate>200</update_rate>
        <imu>
          <angular_velocity>
            <x>
              <noise type='gaussian'>
                <mean>0</mean>
                <stddev>0.0002</stddev>
              </noise>
            </x>
            <y>
              <noise type='gaussian'>
                <mean>0</mean>
```

일반 텍스트 ⌵ 탭 너비: 8 ⌵ 332행, 9열 ⌵ 삽입

장애물 인식을 위한 충돌 조건 추가

>> collision

2-3) 맵 구현



해당 파일을 < 아무이름 >.world 라고 이름을 변경한 후 ~/turtlebot3_ws/src/turtlebot3_simulations/turtlebot3_gazebo/worlds 에 위치

2-3) 맵 구현

ros2 pkg create --build-type ament_python move_robot

패키지 안에 launch file 생성 후 cafe.launch.py
생성

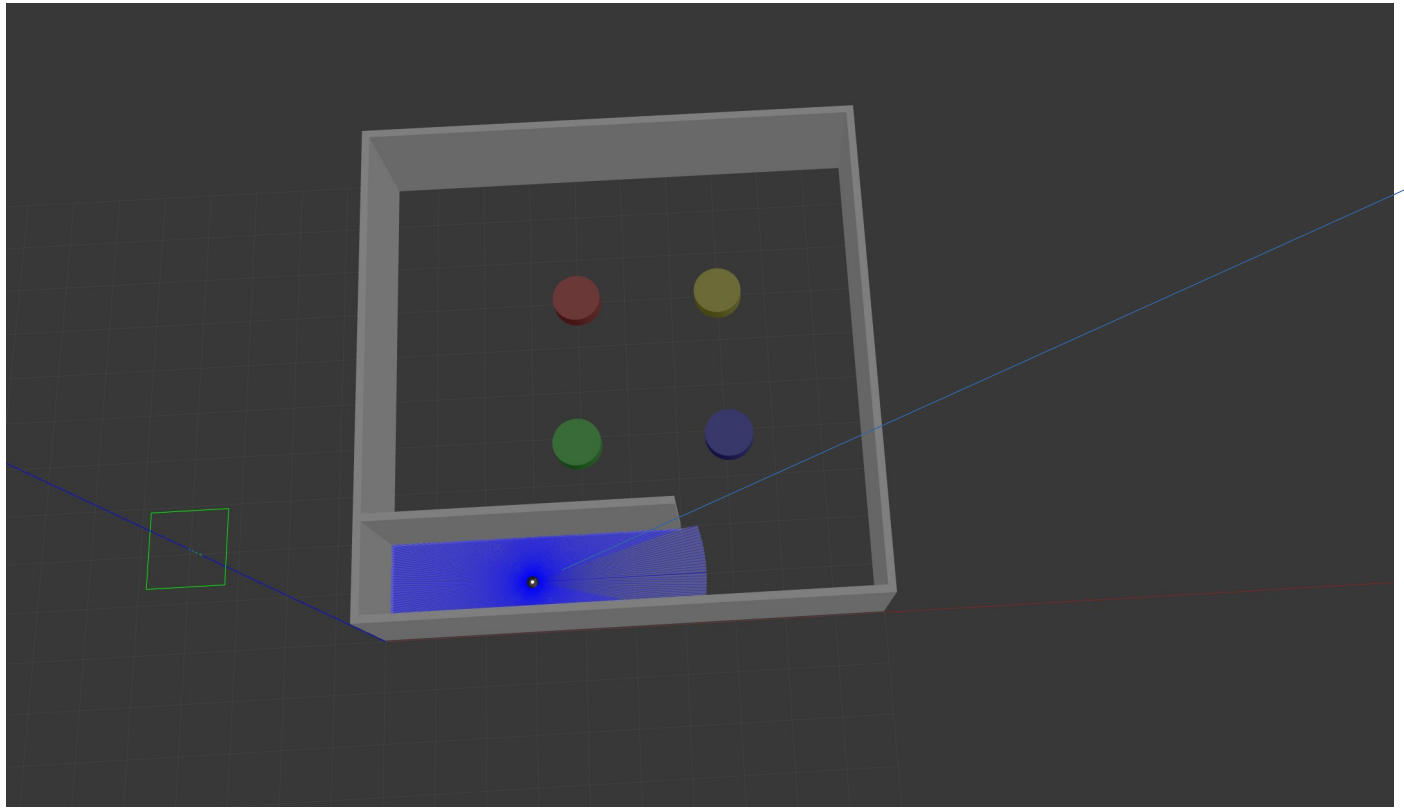
~/turtlebot3_ws/src/turtlebot3_simulations/turtlebot3_gazebo/launch
안에 아무 파일이나 선택한 후 해당 내용을 cafe.launch.py 를 복사 붙여넣기

39 번째 줄을 < 아무이름 >.world 로
변경

33 번 , 34 번째 줄에 x,y 좌표를 원하는 좌표로
변경

2-3) 맵 구현

ros2 launch move_robot cafe.launch.py



입력한 해당 좌표로 터틀봇 생성

3) 향후 개선 방향

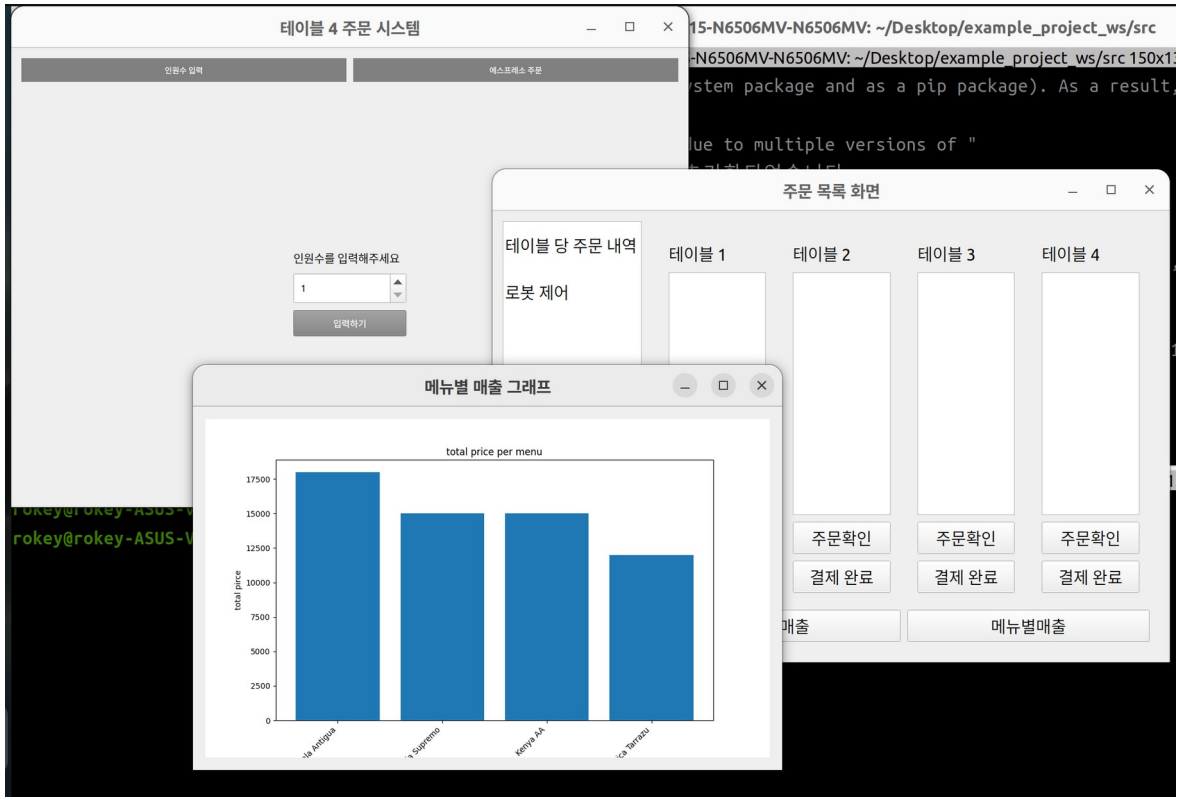
1. 다른 테이블로 선물하기

선물할 테이블 선택 후 주문 대신하기 기능 : orders 테이블의 is_gifting 컬럼 사용

3) 향후 개선 방향

2. 매출 데이터 시각화

메뉴별 매출과 선호 메뉴를 그래프로 시각화



4) 주요 기능 시연

감사합니다

E-4 조