지하철 노선도 구현 코딩 프로젝트

개요

역할 분담

목표

사용 툴

디자인
레이아웃

코드 구현

피드백

역할 분담

전승원 [프론트 엔드]
ui / ux 디자인 설계 및 구현

김다인 [프론트 엔드]
ui / ux 디자인 구현

문희성 [백 엔드]
데이터 베이스 설립

강성빈 [백 엔드]
알고리즘 구현

목표

선정 지하철

대구 지하철 1,2,3 호선

DTRO 대구교통공사 Daegu Transportation Corporation

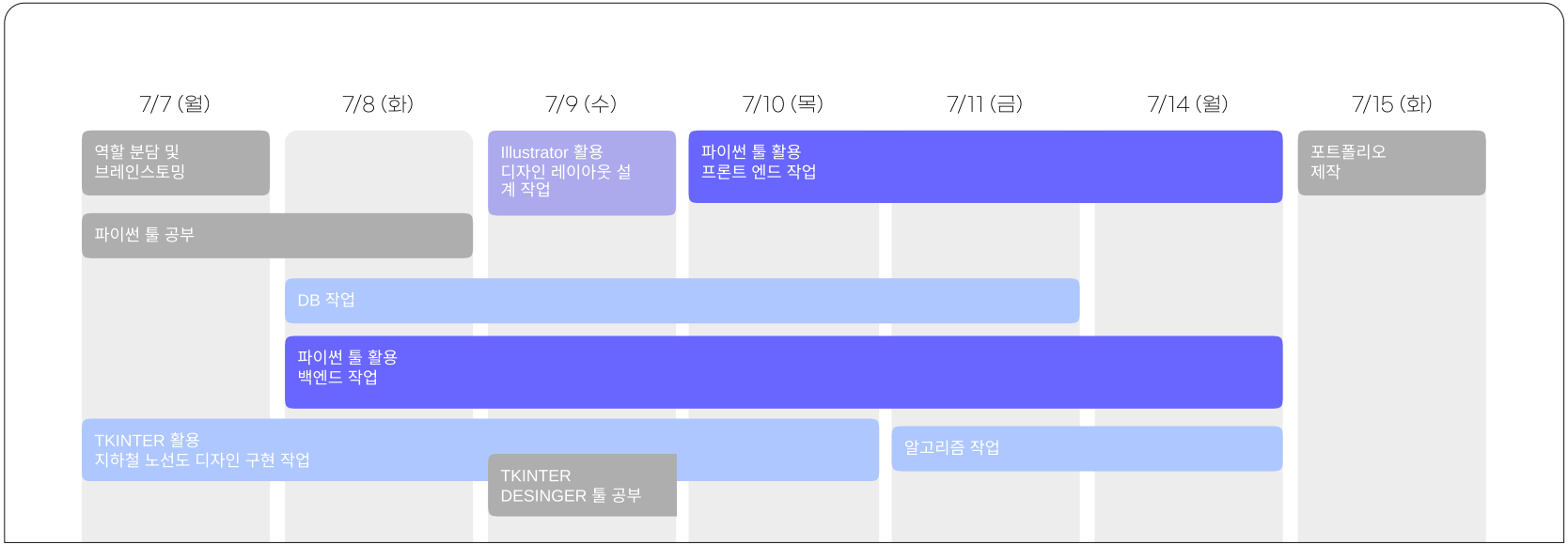
선정 이유

노선이 가장 간편해 보였고, 접근하기 쉬웠다.
또한 구현할 수 있는 기능이 다양해 도전해볼 만하다고 생각했다.

일정 계획

프로젝트 기간

7/7 (월) - 7/15 (화)



사용 툴

파이썬

프로그래밍 언어

비주얼 스튜디오 코드

코드 편집기

커서

AI 코드 에디터

어도비 일러스트레이터

그래픽 디자인 툴

피그마

UI 디자인 툴

깃허브

파일 저장 협업 플랫폼

디자인 레이아웃

사용한 컬러

#6A66F

#004098

#00B4ED

사용한 폰트

맑은 고딕

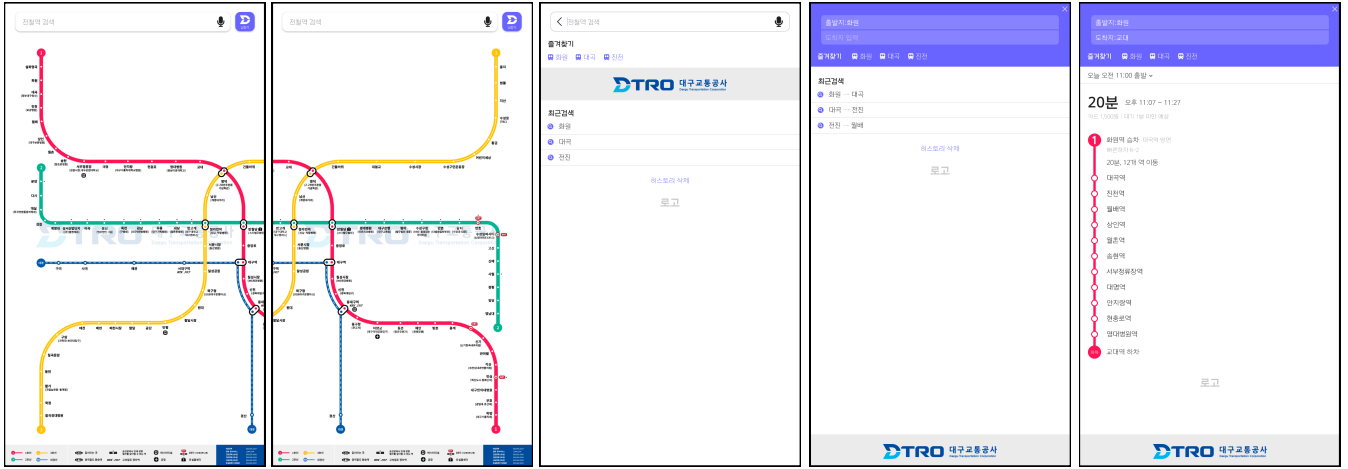
가나다

ABCabc

디자인 레이아웃

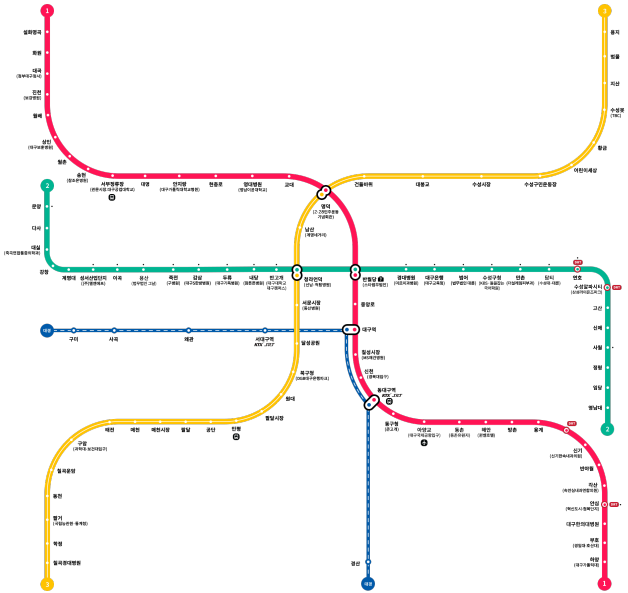
로고
프로토타입

디자인
레이아웃



디자인 레이아웃

대구 지하철
노선도 일러스트화



코드 구현

[main]

메인 노선도 및 역 선택,
팝업, 즐겨찾기 추가 등
메인 인터랙션 담당

```python
"""
[MainPage 클래스 구조 안내]
- 상태 변수 정의
- UI 생성 및 배치
- 핵심 메서드(이벤트 핸들러, 주요 로직 등) 정의
"""
# MainPage: 메인 노선도 및 역 선택, 팝업, 즐겨찾기 추가 등 메인 인터랙션 담당
class MainPage(tk.Frame):
    def __init__(self, parent, controller):
        super().__init__(parent)
        self.controller = controller
        self.configure(bg="#FFFFFF")
        self.popup_window = None
        self.highlight_circle = None
        self.stations = [
            {"name": "설화명곡", "x": 74, "y": 66, "lines": ["1호선"]},
            {"name": "화원", "x": 74, "y": 93, "lines": ["1호선"]},
            {"name": "대곡", "x": 74, "y": 122, "lines": ["1호선"]},
            {"name": "진천", "x": 74, "y": 149, "lines": ["1호선"]},
            {"name": "월배", "x": 75, "y": 178, "lines": ["1호선"]},
            {"name": "상인", "x": 86, "y": 208, "lines": ["1호선"]},
            {"name": "월촌", "x": 105, "y": 234, "lines": ["1호선"]},
            {"name": "송현", "x": 130, "y": 251, "lines": ["1호선"]},
            {"name": "서부정류장", "x": 162, "y": 259, "lines": ["1호선"]},
            {"name": "대명", "x": 206, "y": 260, "lines": ["1호선"]},
            {"name": "안지랑", "x": 251, "y": 260, "lines": ["1호선"]},
            {"name": "현충로", "x": 300, "y": 260, "lines": ["1호선"]},
            {"name": "영대병원", "x": 350, "y": 260, "lines": ["1호선"]},
            {"name": "교대", "x": 399, "y": 260, "lines": ["1호선"]},
            {"name": "명덕", "x": 446, "y": 281, "lines": ["1호선", "3호선"]},
            {"name": "반월당", "x": 487, "y": 388, "lines": ["1호선", "2호선"]},
            {"name": "중앙로", "x": 487, "y": 431, "lines": ["1호선"]},
            {"name": "대구역", "x": 487, "y": 465, "lines": ["1호선", "대경선"]},
            {"name": "칠성시장", "x": 487, "y": 499, "lines": ["1호선"]},
            {"name": "신천", "x": 494, "y": 530, "lines": ["1호선"]},
            {"name": "동대구역", "x": 513, "y": 559, "lines": ["1호선", "대경선"]},
            {"name": "동구청", "x": 538, "y": 578, "lines": ["1호선"]},
            {"name": "아양교", "x": 579, "y": 589, "lines": ["1호선"]},
            {"name": "동촌", "x": 625, "y": 590, "lines": ["1호선"]},
            {"name": "해안", "x": 662, "y": 590, "lines": ["1호선"]},
            {"name": "방촌", "x": 697, "y": 590, "lines": ["1호선"]},
            {"name": "용계", "x": 731, "y": 589, "lines": ["1호선"]},
            {"name": "신기", "x": 795, "y": 620, "lines": ["1호선"]},
            {"name": "반야월", "x": 813, "y": 647, "lines": ["1호선"]},
            {"name": "각산", "x": 820, "y": 674, "lines": ["1호선"]},
            {"name": "안심", "x": 820, "y": 700, "lines": ["1호선"]},
            {"name": "대구한의대병원", "x": 820, "y": 726, "lines": ["1호선"]},
            {"name": "부호", "x": 820, "y": 751, "lines": ["1호선"]},
            {"name": "하양", "x": 820, "y": 777, "lines": ["1호선"]},
            {"name": "문양", "x": 75, "y": 300, "lines": ["2호선"]},
            {"name": "다사", "x": 75, "y": 330, "lines": ["2호선"]},
            {"name": "대실", "x": 75, "y": 359, "lines": ["2호선"]},
            {"name": "강창", "x": 83, "y": 380, "lines": ["2호선"]},
```

코드 구현

[second page]

출발/도착지 선택,
즐겨찾기,
최근검색 관리 및 표시

```python
"""
[SecondPage 클래스 구조 안내]
- 상태 변수 정의
- UI 생성 및 배치
- 핵심 메서드(이벤트 핸들러, 주요 로직 등) 정의
"""
# SecondPage: 출발/도착지 선택, 즐겨찾기, 최근검색 관리 및 표시
class SecondPage(tk.Frame):
    def __init__(self, parent, controller):
        super().__init__(parent)
        self.controller = controller
        self.recent_searches = [] # 최근검색 리스트
        self.canvas = tk.Canvas(self, bg="#FFFFFF", height=960, width=540, bd=0, highlightthickness=0, relief="ridge")
        self.canvas.place(x=0, y=0)
        self.canvas.create_rectangle(0.0, 0.0, 540.0, 128.0, fill="#6A6FFF", outline="")
        self.input_origin_img_tk = ImageTk.PhotoImage(Image.open("build/images/input-origin.png").resize((int(515.0-25.0), int(52.0-23.0))))
        input_origin_img_id = self.canvas.create_image(25.0, 23.0, anchor="nw", image=self.input_origin_img_tk)
        self.input_destination_img_tk = ImageTk.PhotoImage(Image.open("build/images/input-destination.png").resize((int(515.0-25.0), int(84.0-55.0))))
        input_destination_img_id = self.canvas.create_image(25.0, 55.0, anchor="nw", image=self.input_destination_img_tk)
        self.footer2_img_tk = ImageTk.PhotoImage(Image.open("build/images/footer2.png").resize((540, 67)))
        self.canvas.create_image(0, 893, anchor="nw", image=self.footer2_img_tk)
        self.canvas.create_text(15.0, 98.0, anchor="nw", text="즐겨찾기", fill="#FFFFFF", font=("Malgun Gothic", 14 * -1))
        self.fav_icon_img_tk = ImageTk.PhotoImage(Image.open("build/images/subway-icon.png").resize((14, 15)))
        self.fav_items = []  # 즐겨찾기 캔버스 아이템 id 리스트
        self.departure_text_id = self.canvas.create_text(33.0, 29.0, anchor="nw", text="출발지 입력", fill="#FFFFFF", font=("Malgun Gothic", 14 * -1))
        self.arrival_text_id = self.canvas.create_text(33.0, 61.0, anchor="nw", text="도착지 입력", fill="#FFFFFF", font=("Malgun Gothic", 14 * -1))
        def go_to_main_departure(event=None):
            if controller.departure_station and controller.arrival_station:
                controller.arrival_station = None
                self.update_arrival_text()
            controller.is_selecting_departure = True
            controller.is_selecting_arrival = False
            controller.show_frame("MainPage")
        self.canvas.tag_bind(input_origin_img_id, '<Button-1>', go_to_main_departure)
        self.canvas.tag_bind(self.departure_text_id, '<Button-1>', go_to_main_departure)
        def go_to_main_arrival(event=None):
            if controller.departure_station and controller.arrival_station:
                controller.departure_station = None
                self.update_departure_text()
            controller.is_selecting_arrival = True
            controller.is_selecting_departure = False
            controller.show_frame("MainPage")
        self.canvas.tag_bind(input_destination_img_id, '<Button-1>', go_to_main_arrival)
        self.canvas.tag_bind(self.arrival_text_id, '<Button-1>', go_to_main_arrival)
        def go_to_main_normal(event=None):
            controller.is_selecting_arrival = False
            controller.is_selecting_departure = False
            controller.show_frame("MainPage")
        self.canvas.create_text(15.0, 151.0, anchor="nw", text="최근검색", fill="#000000", font=("Malgun Gothic", 16 * -1))
        self.recent_y_start = 186
        self.recent_items = []
        search_icon_img = Image.open("build/images/search-icon.png").resize((18, 18))
        self.search_icon_img_tk = ImageTk.PhotoImage(search_icon_img)
```

코드 구현

[third page]

상태 변수 정의,
ui 생성 및 배치,
핵심 메서드
(이벤트 핸들러,
주요 로직 등) 정의

```python
"""
[ThirdPage 클래스 구조 안내]
- 상태 변수 정의
- UI 생성 및 배치
- 핵심 메서드(이벤트 핸들러, 주요 로직 등) 정의
"""

import tkinter as tk
from tkinter import messagebox, simpledialog
from datetime import datetime, timedelta
from PIL import Image, ImageTk
from route_finder import find_best_route, split_path_by_line
from path_utils import infer_direction

class ThirdPage(tk.Frame):
    def __init__(self, parent, controller):
        super().__init__(parent)
        self.controller = controller
        self.canvas = tk.Canvas(self, bg="#FFFFFF", height=960, width=540, bd=0, highlightthickness=0, relief="ridge")
        self.canvas.place(x=0, y=0)
        self.canvas.create_rectangle(0.0, 0.0, 540.0, 128.0, fill="#6A66FF", outline="")
        self.input_origin_img_tk = ImageTk.PhotoImage(Image.open("build/images/input-origin.png").resize((490, 29)))
        input_origin_img_id = self.canvas.create_image(25.0, 23.0, anchor="nw", image=self.input_origin_img_tk)
        self.input_destination_img_tk = ImageTk.PhotoImage(Image.open("build/images/input-destination.png").resize((490, 29)))
        input_destination_img_id = self.canvas.create_image(25.0, 55.0, anchor="nw", image=self.input_destination_img_tk)
        self.footer_img_tk = ImageTk.PhotoImage(Image.open("build/images/footer.png").resize((540, 46)))
        self.canvas.create_text(15.0, 98.0, anchor="nw", text="즐겨찾기", fill="#FFFFFF", font=("Malgun Gothic", 14 * -1))
        self.fav_icon_img_tk = ImageTk.PhotoImage(Image.open("build/images/subway-icon.png").resize((14, 15)))
        self.fav_items = []
        x_icon_img = Image.open("build/images/x-icon.png").resize((10, 10))
        self.x_icon_img_tk = ImageTk.PhotoImage(x_icon_img)
        x_icon_id = self.canvas.create_image(520, 10, anchor="nw", image=self.x_icon_img_tk)
        def go_to_second_page(event=None):
            controller.is_selecting_arrival = False
            controller.is_selecting_departure = False
            controller.show_frame("SecondPage")
        self.canvas.tag_bind(x_icon_id, '<Button-1>', go_to_second_page)
        self.departure_text_id = self.canvas.create_text(33.0, 29.0, anchor="nw", text="출발지 입력", fill="#FFFFFF", font=("Malgun Gothic", 14 * -1))
        self.arrival_text_id = self.canvas.create_text(33.0, 61.0, anchor="nw", text="도착지 입력", fill="#FFFFFF", font=("Malgun Gothic", 14 * -1))
        def go_to_main_departure(event=None):
            if controller.departure_station and controller.arrival_station:
                controller.arrival_station = None
                self.update_arrival_text()
                controller.frames["SecondPage"].update_arrival_text()
            controller.is_selecting_departure = True
            controller.is_selecting_arrival = False
            controller.show_frame("MainPage")
        def go_to_main_arrival(event=None):
            if controller.departure_station and controller.arrival_station:
                controller.departure_station = None
                self.update_departure_text()
                controller.frames["SecondPage"].update_departure_text()
```

코드 구현

[fourth page]

상태 변수 정의,
ui 생성 및 배치,
핵심 메서드
(이벤트 핸들러,
주요 로직 등) 정의

```python
"""
[FourthPage 클래스 구조 안내]
- 상태 변수 정의
- UI 생성 및 배치
- 핵심 메서드(이벤트 핸들러, 주요 로직 등) 정의
"""

import tkinter as tk
from tkinter import messagebox
from PIL import Image, ImageTk

class FourthPage(tk.Frame):
    def __init__(self, parent, controller):
        super().__init__(parent)
        self.controller = controller
        self.recent_searches = []
        self.canvas = tk.Canvas(
            self,
            bg="#FFFFFF",
            height=960,
            width=540,
            bd=0,
            highlightthickness=0,
            relief="ridge"
        )
        self.canvas.place(x=0, y=0)
        self.search_icon_img_tk = ImageTk.PhotoImage(Image.open("build/images/search-icon.png").resize((18, 18)))
        input_search_img = Image.open("build/images/input-search2.png").resize((496, 48))
        self.tk_img = ImageTk.PhotoImage(input_search_img)
        input_search_img_id = self.canvas.create_image(22, 13, anchor="nw", image=self.tk_img)
        self.search_entry = tk.Entry(self.canvas, font=("Malgun Gothic", 14), bd=0, relief="flat", fg="#222222")
        self.search_entry.place(x=60, y=22, width=350, height=32)
        self.search_entry.bind("<Return>", self.on_search_enter)
        self.search_entry.bind("<KeyRelease>", self.on_search_typing)
        self.suggestion_items = []
        self.arrow_img_tk = ImageTk.PhotoImage(Image.open("build/images/arrow-icon.png").resize((10, 20)))
        arrow_img_id = self.canvas.create_image(37, 27, anchor="nw", image=self.arrow_img_tk)
        def go_to_main(event=None):
            controller.show_frame("MainPage")
        self.canvas.tag_bind(arrow_img_id, '<Button-1>', go_to_main)
        self.canvas.create_rectangle(1.0, 151.70, 540.0, 215.08, fill="#EEEEEE", outline="")
        self.footer2_png = Image.open("build/images/footer2.png").resize((540, 64))
        self.footer2_png_tk = ImageTk.PhotoImage(self.footer2_png)
        self.canvas.create_image(0, 151.7, anchor="nw", image=self.footer2_png_tk)
        self.canvas.create_text(23.37, 82.20, anchor="nw", text="즐겨찾기", fill="#000000", font=("Malgun Gothic", 15 * -1))
        self.canvas.create_text(23.37, 225.41, anchor="nw", text="최근검색", fill="#000000", font=("Malgun Gothic", 15 * -1))
        self.fav_icon_img2_tk = ImageTk.PhotoImage(Image.open("build/images/subway-icon2.png").resize((14, 15)))
        self.fav_items = []
        self.render_favorites()
        self.render_recent_searches()
        self.canvas.bind('<Button-1>', self.on_fav_click)
```

실행 결과

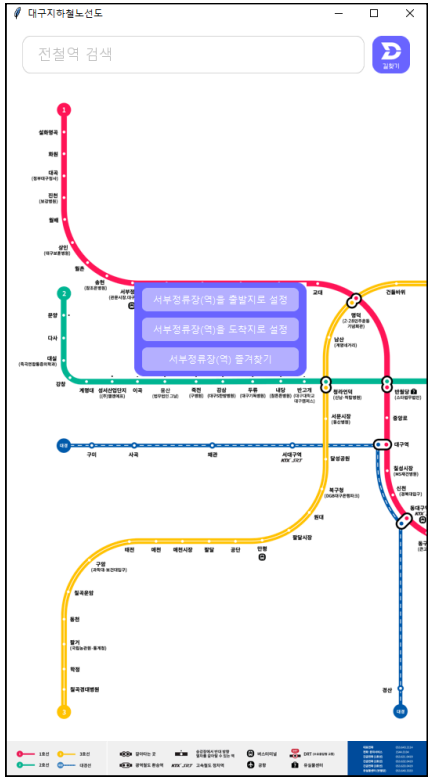
[gui 구조 안내]

- 상태 변수 정의
- ui 생성 및 배치
- 핵심 메서드(이벤트 핸들러, 주요 로직 등) 정의

실행 결과

[main]

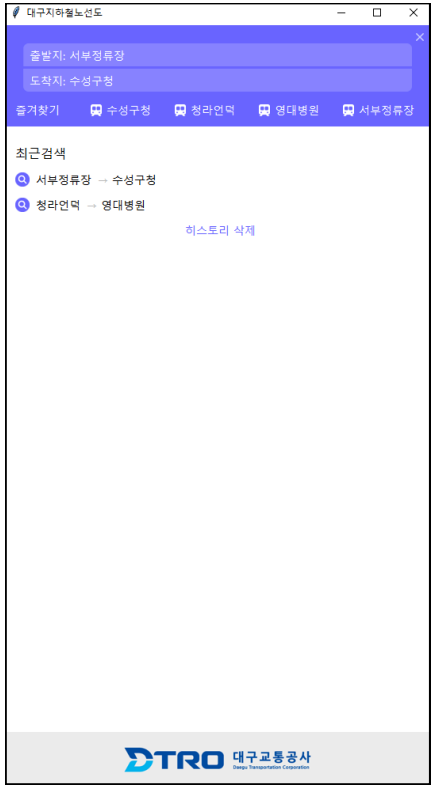
- 전철역 검색창 클릭 시 fourthpage (텍스트 검색창)로 이동
- 길찾기 버튼 클릭 시 secondpage (도착지/출발지 설정창)로 이동
- 노선도(이미지 캔버스)는 확대, 축소, 이동이 가능하며 전철역을 클릭 시 출발지/도착지/즐겨찾기 설정이 가능함



실행 결과

[second page]

- 출발지/도착지를 설정 가능하며 출발지/도착지 둘 다 설정됐을 때 thirdpage로 이동
- 출발지/도착지 검색창 클릭 시 mainpage로 이동하며, 노선도에서 전철역을 클릭하여 지정 가능
- 즐겨찾기 리스트 기능
- 최근검색 리스트 기능



실행 결과

[third page]

- 백엔드 부분과 연결된 핵심 기능 페이지
- 출발지와 도착지가 설정되면 아래의 기능들을 사용 가능

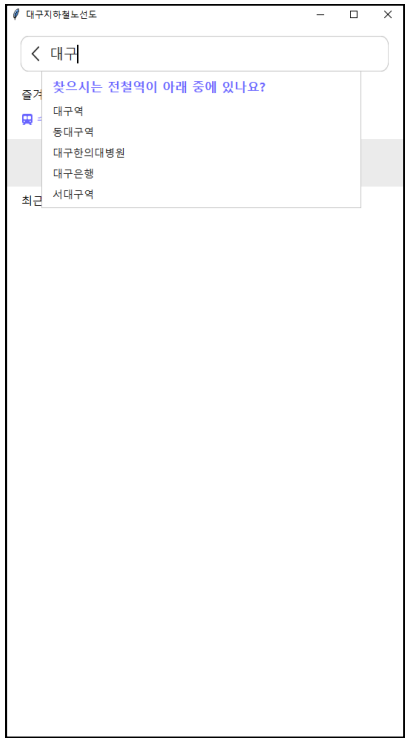
1. 출발 시간 설정 기능
2. 이동 시간 출력
3. 가장 빠른 승차시간을 불러와 예상 시간 출력
4. 시각적 레이아웃으로
승하차/경유/환승역 표현
5. 경유역 개수 출력

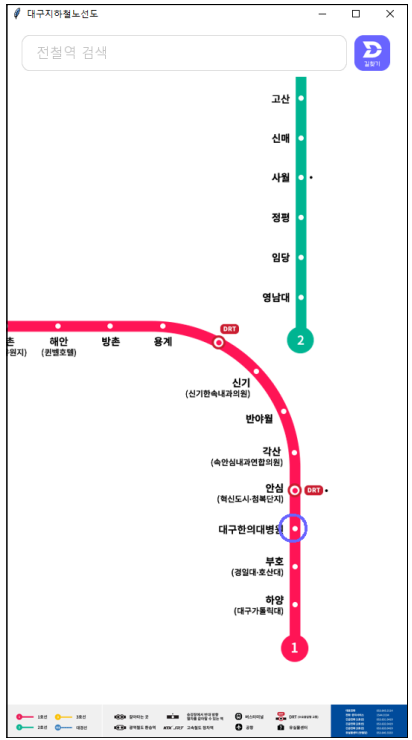
- 즐겨찾기 리스트 기능으로
출발지/도착지 지정 시 secondpage로 이동

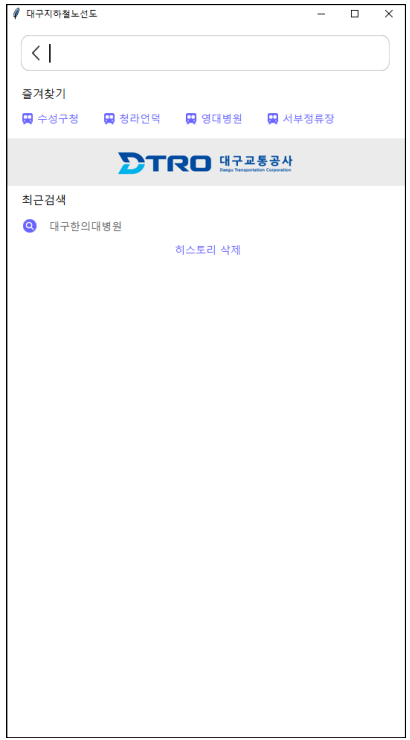
실행 결과

[fourth page]

- 2글자 이상 검색하여 일치하는 전철역 발견 시 팝업에 리스트형태로 출력
(2글자 이하 경고/일치하지 않는 단어 입력 시 경고)
- 팝업 리스트 요소(전철역)클릭 시 mainpage로 이동하며 노선도에서 전철역의 좌표로 줌인
- 즐겨찾기 리스트 기능
- 최근검색 리스트 기능







피드백

전승원

[프로젝트 느낀 점]

python 첫 개발이었지만
언어의 확장성과 유연성 덕분에
전반적인 프로젝트 진행에는
큰 어려움이 없었다.
다만 ui 개발 부분에서는 여러 시행착오를 겪었고,
그로 인해 시간이 많이 소비되어 아쉬움이 남았다.

[프로젝트 개선점]

tkinter designer는
간단한 레이아웃을 구현할 때는 용이했지만,
복잡한 gui를 구현하기에는 적합하지 않았다.
웹에서는 간단하게 배치할 수 있는 요소들이
좌표 계산과 수식 코드로 복잡하게
구성되어 있었기 때문이다.
따라서 python으로 ui를 구현할 때는
개발자 친화적인 템플릿을 먼저 찾는 것이 중요하다고
판단했다.

김다인

[프로젝트 느낀 점]

이번 프로젝트를 통해 파이썬 툴, tkinter, tkinter designer 모듈을 공부할 수 있었고,
ui/ux 측면에서도 많은 것을 배울 수 있었다.
사용자 친화적인 방식으로 프로그램을 제작하는 과정에서
개발자로서 갖추어야 할 중요한 역량이라는 걸
알게 되었고,
그런 부분을 직접 경험해볼 수 있어서
의미 있는 시간이었다.

[프로젝트 개선점]

역할 분담과 커뮤니케이션 부분에서는
다소 아쉬움이 있었다.
회의 시간을 충분히 가졌더라면
각자의 역할을 더 명확하게 정할 수 있었고,
서로의 업무에 대한 이해도
높일 수 있었을 거라고 생각한다.
또한, 익숙하지 않은 툴을 새롭게 배우는 과정이 있었기에
프로젝트 내 역할 수행에 다소 어려움을 겪기도 했다.

문희성

[프로젝트 느낀 점]

공공데이터를 활용해 직접 데이터를 불러오고 분석해보면서 실무에서의 데이터 처리 과정을 체험할 수 있었다.
경로 탐색 알고리즘을 적용하고,
실제 상황에 맞게 테스트해보는 과정이 흥미로웠다.
특히, 다양한 경우의 수를 고려하며
로직을 설계하는 경험이 큰 도움이 되었다.

[프로젝트 개선점]

데이터를 수집하고 전처리하는 과정에서
라이브러리의 사용법을 익히는 데 다소 시간이 걸렸다.
특히 데이터의 구조를 명확히 이해하지 않고
분석을 진행하다 보니
초기 분석 결과가 왜곡되거나 부정확한 경우가 있었다.
앞으로는 데이터 분석 전에
충분한 탐색적 데이터 분석을 통해
데이터의 패턴과 이상치를 먼저 파악하는 습관이 필요하다고 느꼈다.

강성빈

[프로젝트 느낀 점]

다익스트라 알고리즘의 원리를 이해하고, 강사님과 타인이 작성한 예제를 참고하며 익힐 수 있었다.
챗gpt를 활용해 오류 원인을 빠르게 검토하고
수정할 수 있었던 점이
프로젝트 진행에 큰 도움이 되었다.
공공데이터를 직접 읽고 정제하여
그래프를 구성하고, 다양한 모듈을 통해
기능을 세분화하면서 실질적인 데이터 기반 경로 탐색 시스템을 구현할 수 있었다는 점에서
의미 있는 경험이었다.

[프로젝트 개선점]

timetable_query.py의 기능 분리를
완전히 마무리하지 못한 점이 아쉽다.
또한 공공데이터의 총점역 시각이
너무 이른 문제를 해결하기 위한 예외처리 로직(출발 시각 + 예상 소요 시간 보간)도 구현을 끝내지 못했다.
gui와 연동하기 위해 작성한 main.py도
완성도가 낮아 아쉬움이 남는다.
다음에는 전체 구조를 더 명확히 설계하고,
예외처리나 시각 정렬 등 세부적인 완성도를 높이는 데 더 집중해야겠다고 느꼈다.

감사합니다