AGILE 2

Neurotrade

2025.02.03

발표자 : 박현준

Contents

AGILE 2

타임라인

단위 프로젝트

2025년 3월 4일 화요일	36	데이터 분석과 머신러닝, 딥러닝	단위 프로젝트
2025년 3월 5일 수요일	37	데이터 분석과 머신러닝, 딥러닝	단위 프로젝트

AGILE 2

방향

너무 필요한 것 (2차 에자일까지)

1. 화면 구현 및 기능 끝내기 (1주차)

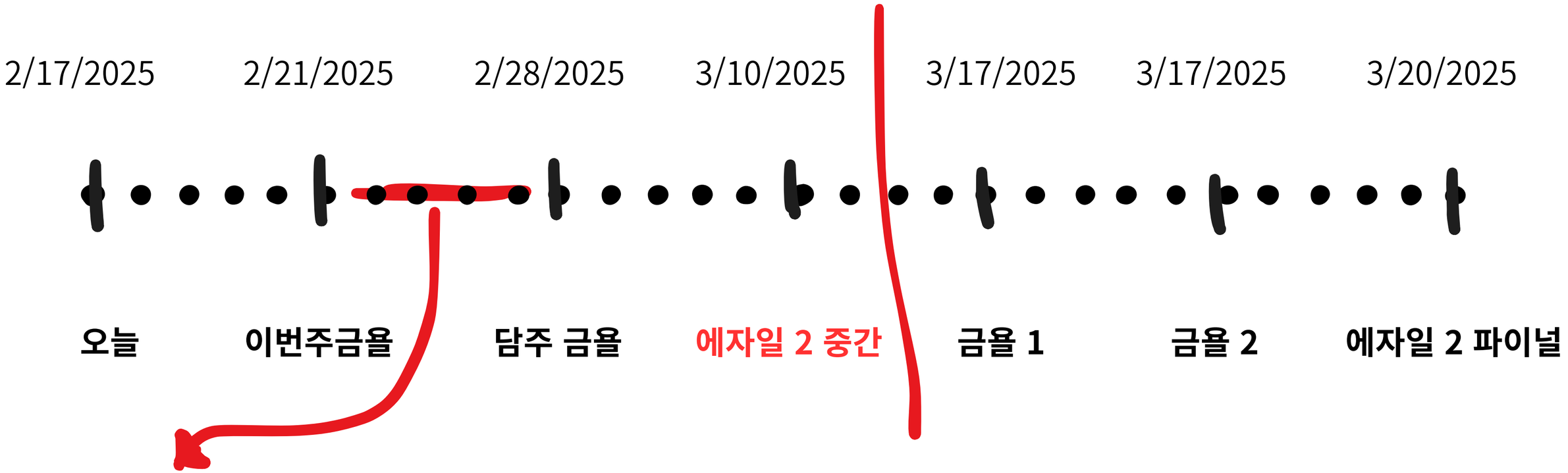
1. 여러 전략 백테스팅 모델 구현 + AI (2주차)

1. 각 모델 별 실제 거래 구현 (중간보고 이후)

1. ~~실제 거래를 총 모니터링하고 실행하는 메인 평션 구현 및 시뮬레이션~~

1주차

- 모든 화면 + 기능 끝내기 (이미 거의 다 됨)



2주차

- 백테스팅 모델 구현
  (프론트엔트에 결과 쏘는 것까지)

3주차

- AI 맛보기(실험) 및 단위 프로젝트 준비

AGILE 2

지금 되어있는 것

+

안되어있는 것

전략

- 만들어진 전략 불러오기 가능 (DB에서)

- (구독할)전략 선택 -> 유저랑 연결 안되어있음
  - if (전략 선택 됨 (전략 클릭)){ 전략 선택되었다는 확인 메세지 및 유저 모델에 업데이트}
  - if (유저.premium user){ 프리미엄 전략 보이게 하기}

- 구독버튼 및 유저 연결
  - If (구독버튼 클릭) {구독확인 메세지 및 유저 모델에서 premium == 1로 바꾸기}

- 봇 통계자료 실제 trade model에서 업데이트 해주기

백테스팅

- 다른 전략 별 각각 백테스팅 “실행”가능

- 다른 전략들도 쓸 수 있는 “중앙” 백테스팅 함수 구현 (이미 거의 구현되어있음)
  - def run_backtesting (전략 1 <- 인풋으로 받기)
    {결과 json으로 프론트에 쏘기 (중요한 그래프 테이블 다)}

- Backtesting/ TA library 에서 Vectorbt로 변환
  - TA library/ Backtesting library 전부 vectorbt 라이브러리로 변환

- 커스텀 백테스팅 (유저 인풋) -> 후순위
- 백테스팅 자료 깡통 (업비트 캔들 1000개 보다 더 확실한 기준 필요) -> 후순위

대쉬보드

- 개인정보 및 기타 지표 디스플레이 가능
- 회원정보 수정 불가 (업비트 api key, secret 업데이트 불가) -> 수정 구현
- api 암호화 -> fernet 사용해서 풀었다 잠궜다하기

AGILE 2

까지가 AGILE 2 1주차


Trades (0)
λ(C)
1.600e+8
1.580e+8
1.560e+8
1.540e+8
1.520e+8
1.500e+8
1.480e+8
1.460e+8
Volume
800
600
400
200
0
Fri, 31 Jan
Sun, 02 Feb
Tue, 04 Feb
Thu, 06 Feb
Sat, 08 Feb
70
60
50
40
30
λ(C)

에자일 2 2주차

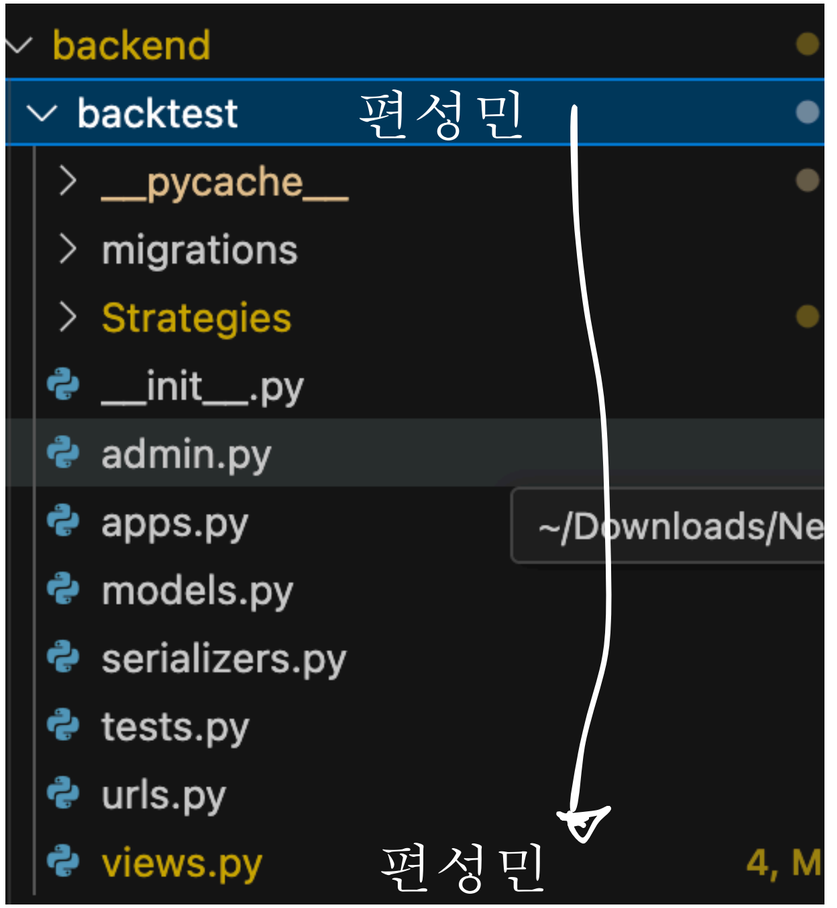
AGILE 2

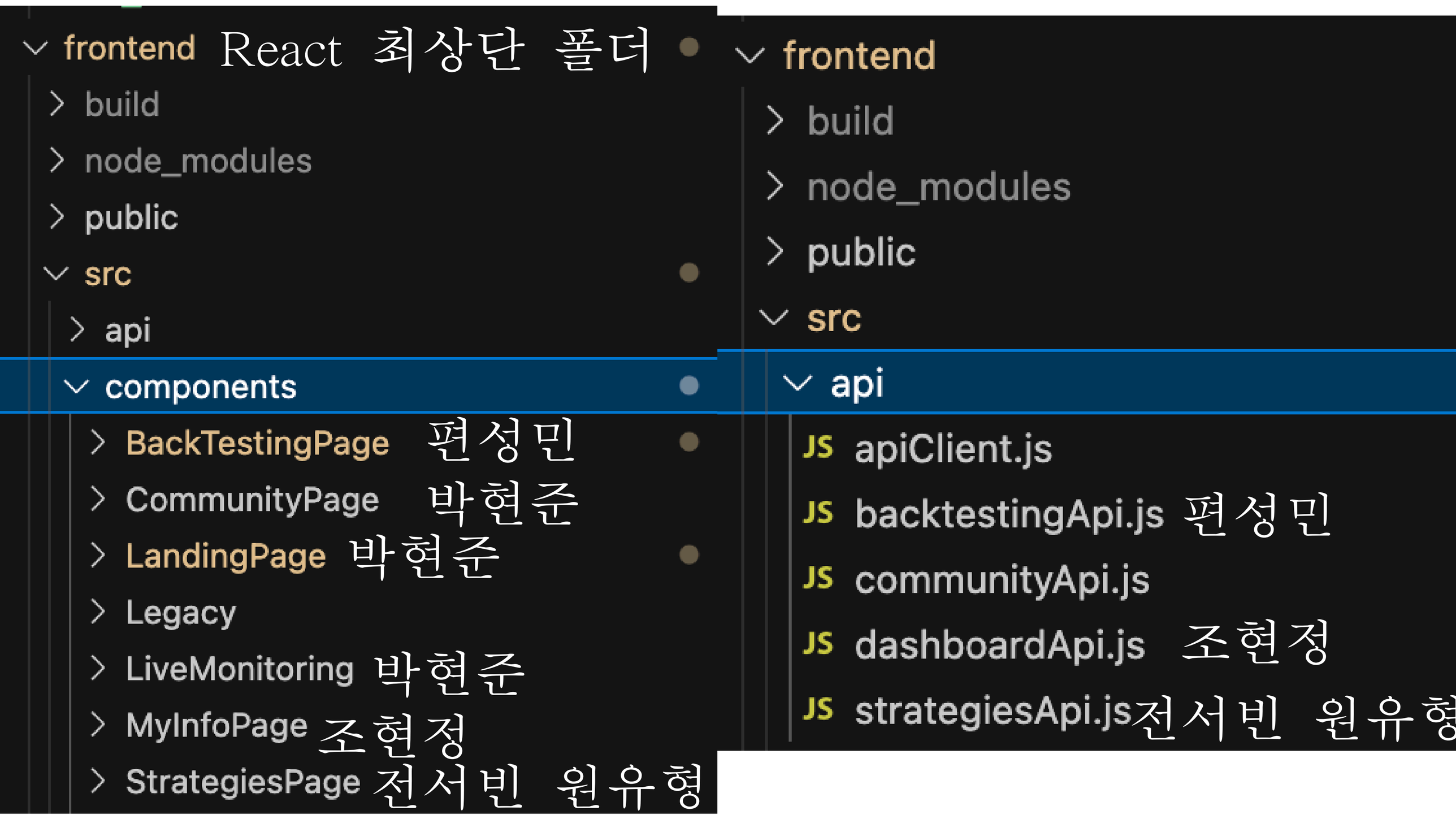
협업 및 REACT 기본

NEUROTRADE-1

backend Django 최상단 폴더

- backtest 편성민
- community 박현준
- dashboard 조현정
- NeuroTrade 박현준
- strategies 전서빈 원유형
- users 박현준

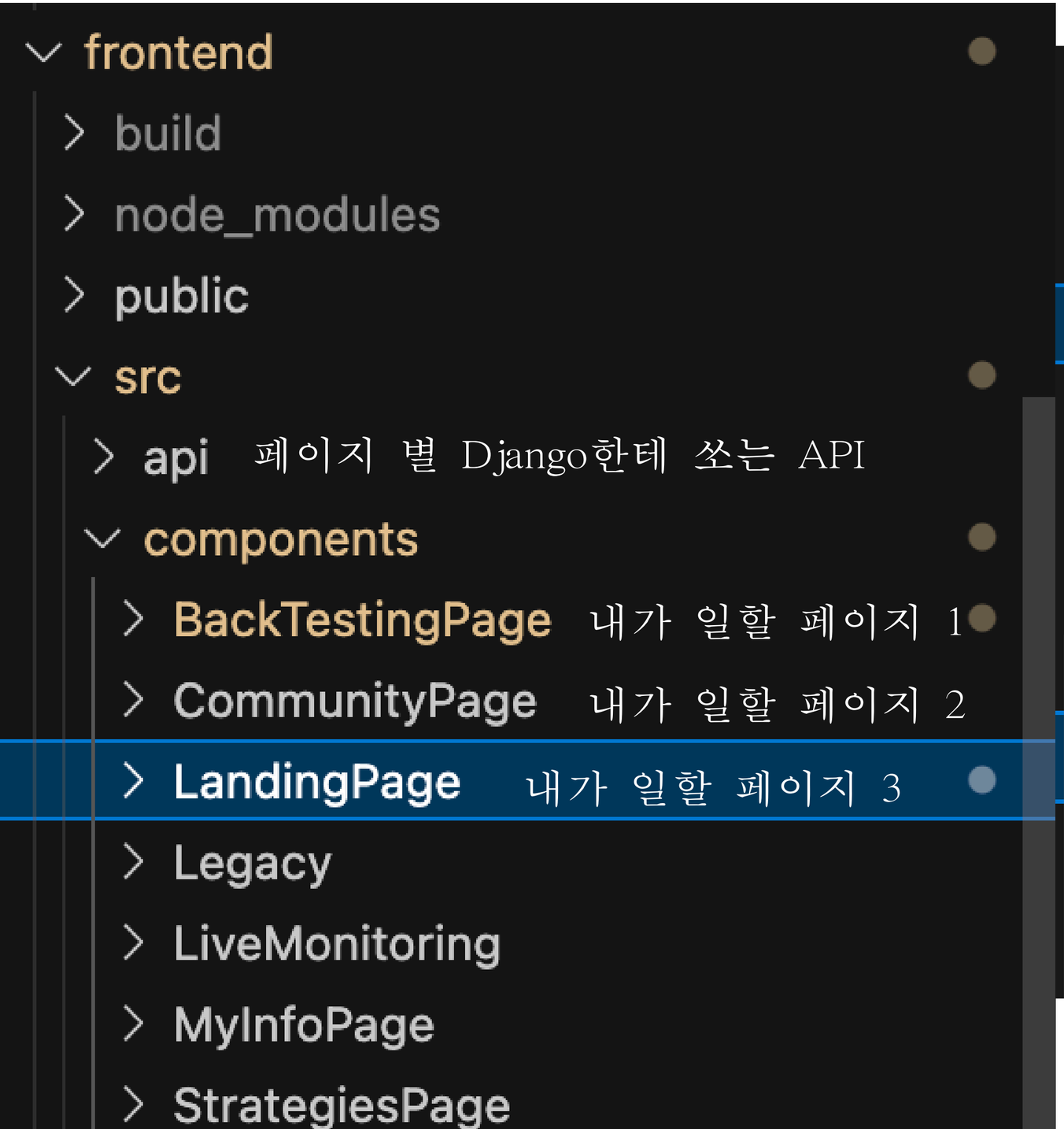


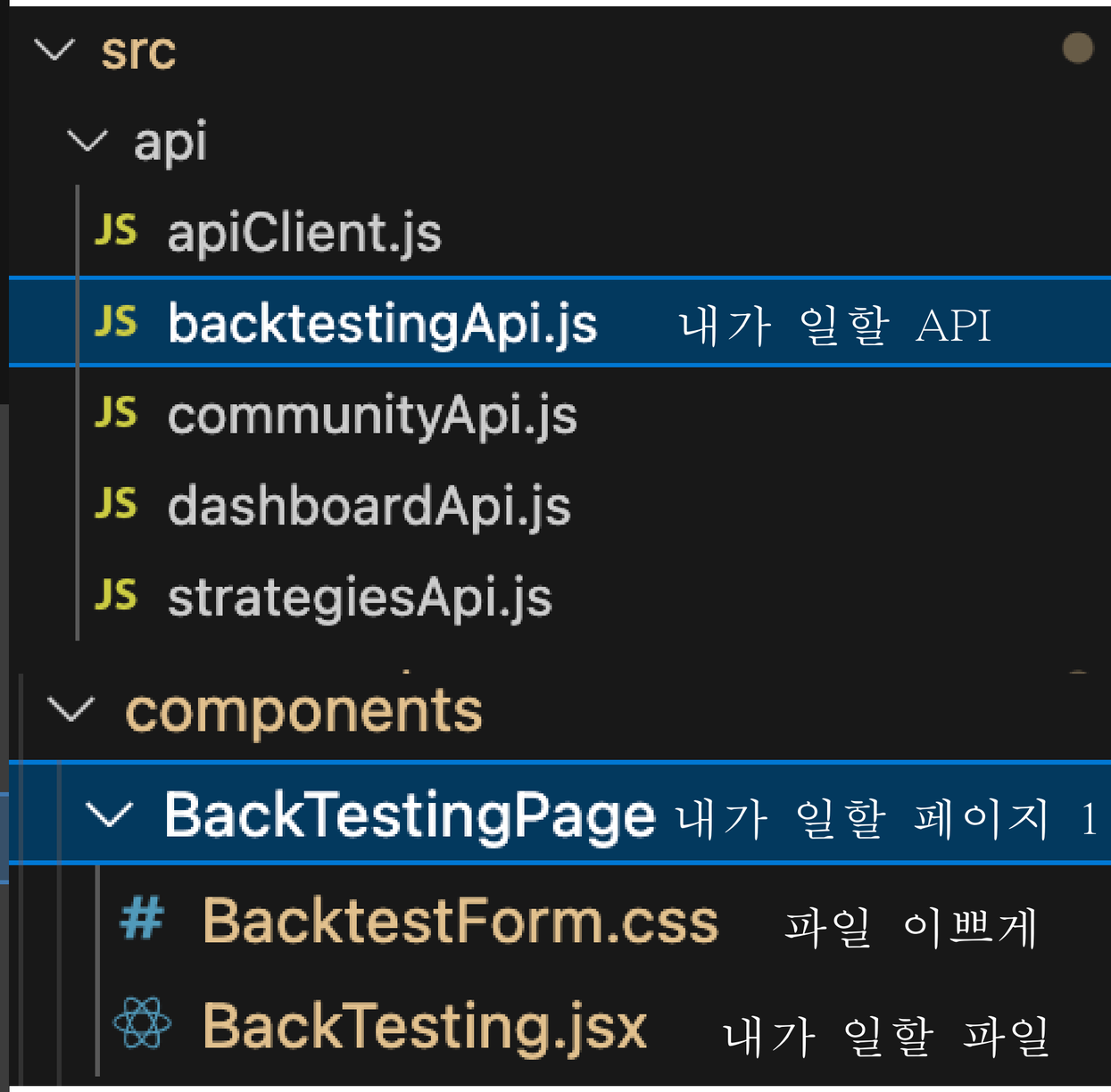

frontend React 최상단 폴더
build
node_modules
public
src
api
components
BackTestingPage 편성민
CommunityPage 박현준
LandingPage 박현준
Legacy
LiveMonitoring 박현준
MyInfoPage 조현정
StrategiesPage 전서빈 원유형
frontend
build
node_modules
public
src
api
JS apiClient.js
JS backtestingApi.js 편성민
JS communityApi.js
JS dashboardApi.js 조현정
JS strategiesApi.js전서빈 원유형

자기 파트만 건드리기

남의 파트 건드려야하면 그 사람에게 공지하기

REACT Component 구성


frontend
build
node_modules
public
src
api 페이지 별 Django한테 쏘는 API
components
BackTestingPage 내가 일할 페이지 1
CommunityPage 내가 일할 페이지 2
LandingPage 내가 일할 페이지 3
Legacy
LiveMonitoring
MyInfoPage
StrategiesPage


src
api
apiClient.js
backtestingApi.js 내가 일할 API
communityApi.js
dashboardApi.js
strategiesApi.js
components
BackTestingPage 내가 일할 페이지 1
BacktestForm.css 파일 이쁘게
BackTesting.jsx 내가 일할 파일

import ...

- API
- 기타 라이브러리
- CSS

const BacktestPage = () => {

- useState + HandleSubmit
- useEffect

return(

- 버튼
- 그래프
- 등등

);

}

Backtesting.jsx

```jsx
import React, { useState, useEffect } from "react";
import { Line, Bar } from "react-chartjs-2"; // ✅ Imported Bar
import "chart.js/auto";
import Header from "../LandingPage/Header";
import { runBacktest } from "../../api/backtestingApi";
import "./BacktestForm.css";
import { fetchStrategies } from "../../api/strategiesApi";
```

REACT 기본 펑션 useeffect/ usestate
기타 제 3자 라이브러리

backtestingAPI.js 에서 사용할 API 불러오기

backtestingAPI.js 에서 사용할 API 불러오기

```jsx
const BacktestPage = () => {
```

진짜 BacktestingPage 펑션 시작

```jsx
useEffect(() => {
  const loadStrategies = async () => {
    try {
      const strategiesData = await fetchStrategies();
      setStrategies(strategiesData);
      console.log(strategiesData);
```

여러가지 변수 및 상황 핸들링

```jsx
return (
  <div>
    <Header />
    <div className="backtest-container">
      <div className="backtest-container">
        <h2 className="backtest-title">전략 선택</h2>
```

return 에서 정말로 화면에서
보여줘야할 것들 렌더링

App.js

```
return (
    <AnimatePresence mode="wait">

    <Routes location={location} key={location.pathname}>
      <Route path="/" element={<PageTransition><LandingPage /></PageTransition>} />
      <Route path="/myinfo" element={<PageTransition><MyInfo /></PageTransition>} />  {/* Route for MyInfo */}
      <Route path="/landingpage" element={<PageTransition><LandingPage/></PageTransition>} />  {/* Route for MyInfo */}
      <Route path="/backtesting" element={<PageTransition><Backtesting/></PageTransition>} />  {/* Route for MyInfo */}
      <Route path="/strategies" element={<PageTransition><Strategies/></PageTransition>} />  {/* Route for MyInfo */}
      <Route path="/livemonitoring" element={<PageTransition><LiveMonitoring /></PageTransition>} />
      <Route path="/community" element={<PageTransition><Community /></PageTransition>} />
    </Routes>

    </AnimatePresence>
);
```

import ...

- API
- 기타 라이브러리
- CSS

```
const BacktestPage = () => {
  • useState + HandleSubmit
  • useEffect

  return(
  • 버튼
  • 그래프
  • 등등
  );
}
```

뭔가 계속 실시간으로 바뀌어야 할 때

API 불러올때

나오는 화면이 이상할때

디버깅 + CHAT GPT한테 물어볼때

예시: 벡테스팅 결과 요청하고 반환 받기

Console.log(“” or varible) 활용 (react가 문제일때)

print(“” or variable) 활용 (django가 문제일때)

전략

- (구독할)전략 선택 -> 유저랑 연결 안되어있음
  - if (전략 선택 됨 (전략 클릭)){ 전략 선택되었다는 확인 메세지 및 유저 모델에 업데이트}
  - if (유저.premium user){ 프리미엄 전략 보이게 하기}

1. 전략 클릭 후 “확인” 버튼 만들기 (1분)
2. 확인 버튼 누르면 api쏴서 현재 로그인 된 유저랑 선택된 봇이랑 연결하기 (1분~1시간)
3. 어떤 전략 선택했는지 팝업창 띄우기 (1분~1시간)
4. 프리미엄 유저인지 확인 -> 프리미엄이면 프리미엄 전략 선택 가능 (프리미엄 전략 임의로 만들어서 확인) (1분~1시간)

- 구독버튼 및 유저 연결
  - If (구독버튼 클릭) {구독확인 메세지 및 유저 모델에서 premium == 1로 바꾸기}

1. 구독버튼 만들기 -> 구독버튼 누르면 현재 유저 isSubcribed = True로 바꿔주기 (1분 ~ 1시간)
2. 이미 구독중이면 구독하기 버튼 텍스트 구독 중으로 표시하기 (1분)

- 봇 통계자료 실제 trade model에서 업데이트 해주기

1. 봇 스태티스틱스 모델 확인 -> 모델 값을 trade 모델에서 받아와서 그 봇에 관한 트레이드 통계 평균 or 총 합으로 봇 통계 업데이트 해주기 (1분~1시간)

5시간 ~ 7 시간

백테스팅

- 다른 전략들도 쓸 수 있는 “중앙” 백테스팅 함수 구현 (이미 거의 구현되어있음)
  - def run_backtesting (전략 1 <- 인풋으로 받기)
    {결과 json으로 프론트에 쏘기 (중요한 그래프 테이블 다)}

1. 프론트에서 선택된 전략 받아오기 (이미 구현 0분)
2. 백테스팅 결과 확인할 수 있도록 최소한 1개 트레이드는 걸리는 데이터로 확인 (1시간)
3. run_backtesting함수가 반환할 데이터 확인 (어떤 통계자료를 쏠까? 어떤 그래프를 보여줄까? -> 맡기겠음 (1시간)

- Backtesting/ TA library 에서 Vectorbt로 변환 (이게 먼저)
  - TA library/ Backtesting library 전부 vectorbt 라이브러리로 변환

1. 현재 모든 백테스팅 위 라이브러리로 실행중 -> vectorbt사용한 백테스팅으로 변환 (~2시간)
2. 프론트에 백테스팅 결과 쏘기 (2시간)
3. 이 함수는 모든 사람들이 앞으로 사용할 것이기 때문에 하드코팅 X

- 커스텀 백테스팅 (유저 인풋) -> 후순위
- 백테스팅 자료 깡통 (업비트 캔들 1000개 보다 더 확실한 기준 필요) -> 후순위

5시간 ~ 7 시간

대쉬보드

- 개인정보 및 기타 지표 디스플레이 가능

- 회원정보 수정 불가 (업비트 api key, secret 업데이트 불가) -> 수정 구현

1. 수정 버튼 만들기 (1분)
2. 수정 버튼 만들면 팝업 or 모달 -> 인풋 받고 유저 모델에 보내서 수정 (1시간)

- api 암호화 -> fernet 사용해서 풀었다 잠궜다하기

1. 처음 api / api secret 받을 때 fernet으로 암호화해서 db에 저장 (1시간)
2. 나중에 api쓸때는 암호화 풀어서 업비트에 쏘기 -> 데이터 받기 (계좌정보 등) (1시간)

5시간

저한테 맞게 물어보는 법

여기서 뭐하신 건지 (구조/세팅)한번만 설명해주세요

이걸 원하시는게 맞는지 확인해주세요

이거해보고 저거 해봤는데 잘 안돼서 한 번 봐주실래요?

저한테 맞는 법

이거 어떻게 해요? API 이렇게 쏘는 거 맞아요?

다 맞게 했는데 왜 안되는지 모르겠어요 / 이거 안돼요

= 아몰랑 해줘

Feature branch 생성좀

Develop 밑에..

NEUROTRADE

팀장 박현준