```
from fastapi import Depends, HTTPException, status
from fastapi.security import OAuth2PasswordBearer
from auth.jwt_handler import verify_access_token

oauth2_scheme = OAuth2PasswordBearer(tokenUrl="/user/signin")

async def authenticate(token: str = Depends(oauth2_scheme)) -> str:
    if not token:
        raise HTTPException(
            status_code=status.HTTP_403_FORBIDDEN,
            detail="Sign in for access"
        )

    decoded_token = verify_access_token(token)
    return decoded_token["user"]
```

이 코드는 다음과 같은 의존 라이브러리를 임포트한다.

- **Depends**: oauth2_scheme을 의존 라이브러리 함수에 주입한다.
- **OAuth2PasswordBearer**: 보안 로직이 존재한다는 것을 애플리케이션에 알려준다.
- **verify_access_token**: 앞서 정의한 토큰 생성 및 검증 함수로, 토큰의 유효성을 확인한다.

그런 다음 OAuth2를 위한 토큰 URL과 authenticate() 함수를 정의한다. authenticate() 함수는 토큰을 인수로 받는다. 토큰이 유효하면 토큰을 디코딩한 후 페이로드의 사용자 필드를 반환하고 유효하지 않으면 verify_access_token() 함수에 정의된 오류 메시지를 반환한다.

라우트에 보안 적용을 위한 의존 라이브러리를 만들었다. 다음으로 라우트를 수정해서 인증 처리를 적용하고 authenticate() 함수를 이벤트 라우트에 주입해보자.

7.3 애플리케이션 변경

이번에는 라우트를 수정해서 새롭게 작성한 인증 모델을 적용해보자. 또한 이벤트 추가용 POST 라우트를 변경해서 사용자 레코드에 이벤트 필드를 추가해보자.