```
else:
    # Perform local verification
    jwks_data = await get_jwks()
    rsa_key = None
    for key in jwks_data["keys"]:
        rsa_key = JsonWebKey.import_key(key)
        break
    # Decode the token
    payload = jwt.decode(
        token, rsa_key, claims_options={"exp": {"essential": True}}
    )
    # Validate the 'aud' claim
    if payload.get("aud") != config('OKTA_AUDIENCE'):
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED, detail="Invalid audience"
        )
    # Validate the 'iss' claim
    if payload.get("iss") != AUTHORIZATION_SERVER_URL:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED, detail="Invalid issuer"
        )
```

Local validation with Authlib

```
# Validate the 'exp' claim
if "exp" in payload:
    expiration = datetime.fromtimestamp(payload["exp"], tz=timezone.utc)
    print(f"Token expiration time: {expiration}")
    if expiration < datetime.now(timezone.utc):
        print("Token has expired")
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED, detail="Token has expired"
        )
# Get scopes
token_scopes = payload.get("scp", [])
if not isinstance(token_scopes, list):
    token_scopes = token_scopes.split()

user_data = {
    "sub": payload["sub"],
    "username": payload.get("username", payload.get("preferred_username",
payload["sub"])),
}
```