```
# 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"])),
}
```

```
# Validate required scopes
for scope in SCOPES:
    if scope not in token_scopes:
        print(f"Missing required scope: {scope}")
        raise HTTPException(
            status_code=status.HTTP_403_FORBIDDEN, detail="Insufficient scope"
        )
# Returns a user object when all checks pass
print(f"User data: {user_data}")
return User(**user_data)
```

Scope check for both remote and local validation