# 11a [Individual/Paris/Group] Auth Integration

Step-by-Step Guide Using Python and Poetry

This guide walks you through setting up a simple Python web app using **FastAPI**, **Uvicorn**, and **Google OAuth 2.0** to authenticate users via **Google login**, with dependencies managed using **Poetry**.

The project consists of a simple backend and a styled frontend that welcomes the user after login.

## Content

## Step 1: Install and Set Up Poetry

1. If Poetry is not installed, install it via the official script:

```
curl -sSL https://install.python-poetry.org | python3 -
```

   Alternatively, check: [Python-Poetry/installation]

2. Create a new Poetry project:

```
poetry new 11a._Auth_Integration
cd 11a._Auth_Integration
```

3. Configure Poetry to use your Python version (optional but recommended):

```
poetry env use python3.11.9
```

4. Open the pyproject.toml file and update the project info if needed.

## Step 2: Add Dependencies with Poetry

Install required packages:

```
poetry add fastapi uvicorn python-dotenv jinja2 google-auth google-auth-oauthlib
google-api-python-client
```

## Step 3: Configure Google Cloud OAuth Credentials

1. Visit [Google Developers Console]()
2. Create or select a project.
3. Navigate to **APIs & Services > OAuth Consent Screen** and complete the form:
    a. fill out app info and support email
    b. Select External and click **Next**
    c. Fill out the required fields (App name, email, etc.)
    d. And lastly click **Create**
    e. Add your own email under test users under the **Audience** tab in the left side of the screen.
4. Go to **Credentials > Create Credentials > OAuth Client ID**: (If you have a hard time finding the credentials tab you can use the search bar in the top middle of the screen)
    a. Choose **Web application**
    b. Under **Authorized Redirect URIs**, add:

       http://localhost:8000/auth/google/callback

    c. You can leave **Authorized JavaScript origins** blank – it's not need for FastAPI
    d. Click **Create**, and copy the **Client ID** and **Client Secret**

## Step 4: Set Up the Project Structure

```
11a._Auth_Integration /
│
├── app/
│   ├── main.py
│   ├── .env
│   ├── static/
│   │   ├── index.css
│   │   └── index.js
│   └── templates/
│       └── index.html
└── .gitignore
```

## Step 5: Create Environment File

In .env, store your sensitive credentials:

```
CLIENT_ID=your_google_client_id
CLIENT_SECRET=your_google_client_secret
REDIRECT_URI=http://localhost:8000/auth/google/callback
```

## Step 6: Create the FastAPI App (app/main.py)

```python
from dotenv import load_dotenv
import os
import json
from fastapi import FastAPI, Request
from fastapi.responses import RedirectResponse, HTMLResponse
from fastapi.staticfiles import StaticFiles
from google_auth_oauthlib.flow import Flow
from google.oauth2.credentials import Credentials
import googleapiclient.discovery
from fastapi.templating import Jinja2Templates
from fastapi.responses import Response

load_dotenv(dotenv_path=os.path.join(os.path.dirname(__file__), ".env"))
app = FastAPI()
app.mount("/static", StaticFiles(directory="static"), name="static")
templates = Jinja2Templates(directory="templates")

CLIENT_ID = os.getenv("CLIENT_ID")
CLIENT_SECRET = os.getenv("CLIENT_SECRET")
REDIRECT_URI = os.getenv("REDIRECT_URI")
SCOPES = ["https://www.googleapis.com/auth/userinfo.profile"]

def get_google_auth_flow() -> Flow:
    flow = Flow.from_client_config(
        {
            "web": {
                "client_id": CLIENT_ID,
                "client_secret": CLIENT_SECRET,
                "auth_uri": "https://accounts.google.com/o/oauth2/auth",
                "token_uri": "https://oauth2.googleapis.com/token",
                "redirect_uris": [REDIRECT_URI]
            }
        },
        scopes=SCOPES
    )
    flow.redirect_uri = REDIRECT_URI
    return flow

@app.get("/", response_class=HTMLResponse)
async def homepage(request: Request):
    return templates.TemplateResponse("index.html", {"request": request})
```

```python
@app.get("/auth/google/consent")
async def google_consent():
    flow = get_google_auth_flow()
    auth_url, _ = flow.authorization_url(access_type='offline',
include_granted_scopes='true')
    print(auth_url)
    return RedirectResponse(auth_url)

@app.get("/auth/google/callback")
async def google_callback(request: Request):
    try:
        code = request.query_params.get("code")
        flow = flow = get_google_auth_flow()
        flow.fetch_token(code=code)
        credentials = flow.credentials
        tokens = {
            "token": credentials.token,
            "refresh_token": credentials.refresh_token,
            "token_uri": credentials.token_uri,
            "client_id": credentials.client_id,
            "client_secret": credentials.client_secret,
            "scopes": credentials.scopes
        }
        tokens_json = json.dumps(tokens)
        return RedirectResponse(url=f"/?tokens={tokens_json}")
    except Exception as e:
        print("Callback Error:", e)
        return HTMLResponse(content="Error during authentication", status_code=500)

@app.get("/auth/whoami")
async def whoami(tokens: str):
    try:
        tokens_dict = json.loads(tokens)
        creds = Credentials(
            token=tokens_dict["token"],
            refresh_token=tokens_dict.get("refresh_token"),
            token_uri=tokens_dict["token_uri"],
            client_id=tokens_dict["client_id"],
            client_secret=tokens_dict["client_secret"],
            scopes=tokens_dict["scopes"]
        )
        service = googleapiclient.discovery.build("oauth2", "v2", credentials=creds)
        user_info = service.userinfo().get().execute()
        return user_info
    except Exception as e:
        print("Whoami Error:", e)
        return HTMLResponse(content="Error fetching user info", status_code=500)

@app.get("/favicon.ico", include_in_schema=False)
async def favicon():
    return Response(status_code=204)  # No Content
```

## Step 7: Create the index.html (app/templates/index.html)

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Login Test App</title>
  <link rel="stylesheet" href="/static/index.css" />
</head>
<body id="body">
  <header class="header">
    <a href="/auth/google/consent" class="btn">Login with Google</a>
  </header>

  <div class="main">
    <div id="user">
      <h1>Welcome Guest</h1>
      <p>Please login to continue.</p>
    </div>
  </div>

  <script src="/static/index.js"></script>
</body>
</html>
```

## Step 8: Create CSS (app/static/index.css)

```css
body {
    font-family: Arial, Helvetica, sans-serif;
    margin: 0;
}

h1 {
    margin: 0;
}

.btn {
    display: inline-block;
    padding: 10px 20px;
    background-color: #007bff;
    color: #fff;
    text-decoration: none;
    border-radius: 5px;
}

.header {
    padding: 20px;
    background-color: #f4f4f4;
    display: flex;
    align-items: center;
    justify-content: space-between;
}

.main {
    padding: 20px;
}

.profile-img {
    width: 40px;
    height: 40px;
    border-radius: 50%;
    border: 1px solid #333;
    box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);
}
```

## Step 9: Create JS (app/static/index.js)

```
const queryString = window.location.search;
const urlParams = new URLSearchParams(queryString);
const tokens = urlParams.get('tokens');

if (tokens) {
    (async () => {
        const response = await fetch(`/auth/whoami?tokens=${tokens}`, {
            method: 'GET',
        });

        const data = await response.json();

        if (data.error) {
            console.error(data.error);
            return;
        }

        document.getElementById('body').innerHTML = `
            <header class="header">
                <img class="profile-img" src="${data.picture}" alt="${data.name}"/>
                <a href="/" class="btn">Logout</a>
            </header>
            <div class="main">
                <div id="user">
                    <h1>Welcome ${data.name}</h1>
                </div>
            </div>
        `;
    })();
}
```

## Step 10: Add .gitignore

```
.env
__pycache__/
*.pyc
*.pyo
*.pyd
venv/
```

## Step 11: Run the Project

1. Start the virtual environment:

   poetry shell

2. Run the server from the root of the project with:

   uvicorn app.main:app --reload

3. Open your browser and go to:

   http://localhost:8000

4. Click "**Login with Google**" and you'll be greeted with your name and profile image.