**✅ Step-by-Step API Flow with JSON**

**1. 👤 POST /login**

Creates a user with some default data.

**URL:** http://127.0.0.1:5000/login  
**Body (raw JSON):**

{

"username": "monish",

"password": "1234"

}

**2. 🧑‍💼 POST /add\_partner**

Registers a partner (e.g., PartnerX) for that user.

**URL:** http://127.0.0.1:5000/add\_partner  
**Body:**

{

"username": "monish",

"partner": "PartnerX"

}

--to view profile

### ✅ View Profile

**GET** http://127.0.0.1:5000/profile?username=monish

Expected:

json

CopyEdit

{

"username": "monish",

"partners": ["PartnerX"],

"data": {

"salary": 85000,

"credit\_score": 735,

"account\_balance": 42350.75,

"transaction\_history": ["Paid $50 to Netflix", "Received $200 from client"]

}

}

**3. ✅ POST /grant\_consent**

Grants consent for a partner to access a specific data type for a limited time.

**URL:** http://127.0.0.1:5000/grant\_consent  
**Body:**

{

"username": "monish",

"partner": "PartnerX",

"data\_type": "salary",

"duration\_minutes": 5

}

📝 Supported data\_type: salary, credit\_score, account\_balance, recent\_txns

✅ Response will include:

"token": {

"token\_id": "abc-123-xyz", ← COPY this for next steps

...

}

**4. 🔓 POST /use\_token**

Partner uses token to access the data (if valid).

**URL:** http://127.0.0.1:5000/use\_token  
**Body:**

{

"token\_id": "paste-the-token-id-here",

"partner": "PartnerX"

}

🟢 If token is valid and not expired or revoked → returns the data  
🔴 Else → returns error (expired, revoked, or unauthorized)

**5. ⛔ POST /revoke\_token**

Revokes a token, even if it's not yet expired.

**URL:** http://127.0.0.1:5000/revoke\_token  
**Body:**

{

"username": "monish",

"token\_id": "paste-the-token-id-here"

}

✅ Response: { "message": "Token xyz revoked" }  
🔴 If token not found for the user → error

**(Optional) 6. 📜 Coming Soon: /audit\_log**

You’ll be able to see:

* Active tokens
* Revoked tokens
* Expired tokens

**🧪 Tips for Postman:**

* Use Headers tab → set:
  + Content-Type: application/json
* Set method to POST for all
* Save each request in a **collection** for easy reuse
* Use **environments** to store token values dynamically later

Let me know if you want to:

* Add audit logs next
* Persist users/tokens using JSON or SQLite
* Or move to frontend setup later

We're on track to having a full-featured privacy vault! 🔐💡

* Steps for get tokens and get partners----------------------------------------------------------
* **Method:** POST
* **URL:** http://127.0.0.1:5000/get\_tokens
* **Headers:**
  + Content-Type: application/json
* **Body (raw JSON):**

json

CopyEdit

{

"username": "monish"

}

**✅ Response Example:**

json

CopyEdit

{

"tokens": [

{

"token\_id": "1234-5678",

"username": "monish",

"partner": "PartnerX",

"data\_type": "salary",

"expires\_at": "2025-07-05T18:00:00",

"revoked": false,

"expired": false

},

{

"token\_id": "abcd-efgh",

"username": "monish",

"partner": "PartnerY",

"data\_type": "credit\_score",

"expires\_at": "2025-07-05T17:00:00",

"revoked": true,

"expired": false

}

]

}

**✅ 2. POST /get\_partners**

**🧠 Purpose:**  
Returns all partners that a user has registered. Useful to populate a dropdown when granting consent.

**✅ Postman Setup:**

* **Method:** POST
* **URL:** http://127.0.0.1:5000/get\_partners
* **Headers:**
  + Content-Type: application/json
* **Body (raw JSON):**

json

CopyEdit

{

"username": "monish"

}

**✅ Response Example:**

json

CopyEdit

{

"partners": [

"PartnerX",

"PartnerY"

]

}