Here’s an explanation of each **GunDB gun.user API method**, with examples and internal behavior:

**🔐 1. gun.user().create(username, password, callback)**

Creates a new user with a public key, and encrypts their private key with the password.

**Example:**

js

CopyEdit

gun.user().create('alice', 'secretpass', (ack) => {

if (ack.err) console.log('Error:', ack.err);

else console.log('User created:', ack);

});

**🔑 2. gun.user().auth(username, password, callback)**

Authenticates (logs in) a user with their credentials.

**Example:**

js

CopyEdit

gun.user().auth('alice', 'secretpass', (ack) => {

if (ack.err) console.log('Login failed');

else console.log('Logged in as:', ack);

});

✅ Internally:

* Gun decrypts the private key using the password.
* Once decrypted, it can sign/verifying data for that user.

**🚪 3. gun.user().leave()**

Logs out the user (removes the private key from memory).

**Example:**

js

CopyEdit

gun.user().leave();

**🧨 4. gun.user().delete(username, password, callback)**

Deletes the user account permanently (removes credentials from the system).

**Example:**

js

CopyEdit

gun.user().delete('alice', 'secretpass', (ack) => {

if (ack.err) console.log('Delete failed:', ack.err);

else console.log('User deleted:', ack);

});

**Important Note:**  
GunDB immediately deletes the user keys, unlike platforms like Instagram or Facebook that delay deletion by 30 days.

**🔁 5. gun.user().recall(options)**

Automatically remembers the user in localStorage and re-authenticates them.

**Example:**

js

CopyEdit

gun.user().recall({sessionStorage: true}); // or true for localStorage

✅ Enables persistent login across refreshes.

**❤️ 6. gun.user().alive(callback)**

Checks if a user is currently authenticated (logged in).

**Example:**

js

CopyEdit

gun.user().alive((ack) => {

console.log('Alive status:', ack);

});

**🔏 7. gun.user().trust(otherPublicKey, [callback])**

Marks another public key as trusted—mainly for access control systems.

**Example:**

js

CopyEdit

gun.user().trust('~otherPublicKey');

Used in secure apps to define roles or permissions.

**🔐 8. gun.user().grant(options)**

Gives another user access to some encrypted data.

**Example (AES key sharing):**

js

CopyEdit

gun.user().grant({

who: '~someOtherPubKey',

what: 'profile', // the node/key you want to grant

ok: (ack) => console.log("Granted access:", ack)

});

**🧬 9. gun.user().secret(data, callback)**

Encrypts sensitive data using the user's own private key.

**Example:**

js

CopyEdit

gun.user().secret("my private note", (ack) => {

console.log("Encrypted secret:", ack);

});

**TL;DR Summary Table**

| **Method** | **Purpose** | **Example Functionality** |
| --- | --- | --- |
| create() | Register user | New user creation |
| auth() | Log in user | Decrypts private key |
| leave() | Log out | Clears session |
| delete() | Permanently delete user | Removes keys & data |
| recall() | Persistent login | Auto-login after refresh |
| alive() | Check login status | Useful in UI logic |
| trust() | Trust another public key | Used in ACL/permission systems |
| grant() | Give access to encrypted data | For secure multi-user sharing |
| secret() | Encrypt private data | End-to-end encrypted fields |