# **Discord Server Scraper Documentation**

## **1. Overview**

### **Objective**

This script automates data extraction from **Discord servers** using **Selenium WebDriver**. It efficiently scrapes:

1. **Server Information** (name, ID)
2. **Channel Details** (text & voice channels)
3. **Member List** (grouped users, online members)
4. **Messages from Channels** (username, timestamp, message content, attachments)
5. **Last Active Status** (status tracking of all members)

## **2. Process Flow**

### **Step 1: Logging into Discord**

* The login\_discord function automates the login process.
* It opens **Discord Canary**, enters credentials, and verifies login success.
* It uses **explicit waits** to ensure elements load properly.
* A **failure screenshot** is captured in case of login issues.

### **Step 2: Extracting Server Information**

* The script extracts the **server name** and **server ID**.
* Finds the **server sidebar** and identifies available **text and voice channels**.

### **Step 3: Extracting Online Members**

* The extract\_groups\_and\_online\_members function scrolls through the **member list**.
* **Online, Idle, Do Not Disturb, and Offline members** are classified.
* Uses **incremental scrolling** to ensure all users are detected.

### **Step 4: Extracting Messages**

* Messages from each **channel URL** are scraped.
* Extracted data includes:
  + **Username**
  + **Timestamp**
  + **Message Content**
  + **Attachments** (images, links)
* Uses **scrolling and retries** to load full conversation history.

### **Step 5: Extracting Last Active Status**

* Tracks the **last online status** for all users.
* Implements **DOM tracking & scrolling** for complete member list retrieval.

### **Step 6: Saving Data**

* All extracted data is **stored in a JSON file** using save\_to\_file().
* The format follows a structured **server, members, and messages hierarchy**.

## **3. Data Model**

### **Input Data**

| **Parameter** | **Description** |
| --- | --- |
| **Credentials** | Discord email and password (stored in .env). |
| **Server URLs** | List of Discord server URLs to scrape. |
| **Channel URLs** | Optional: List of channels to extract messages from. |

### **Output Data Structure (JSON)**

| **Field** | **Description** |
| --- | --- |
| **server\_name** | Name of the Discord server. |
| **server\_id** | Unique identifier for the server. |
| **channels** | List of extracted text & voice channels. |
| **members** | Extracted groups and members. |
| **messages** | Messages from specific channels (optional). |
| **last\_active** | Last seen status of members. |

## **4. Technical Solution**

### **Technology Stack**

* **Programming Language**: Python
* **Web Automation**: Selenium WebDriver
* **Data Storage**: JSON format

### **Key Components**

#### **1. Selenium WebDriver**

* Automates browser interactions for **login, scrolling, and extraction**.
* Uses **ChromeDriver with stealth configurations** to bypass bot detection.

#### **2. Error Handling & Retry Logic**

* Implements **timeouts and exception handling** for:
  + Login failures
  + Stale DOM references
  + Element loading delays

#### **3. Scroll-Based Data Extraction**

* Uses **incremental scrolling** for:
  + Member list retrieval
  + Message history extraction

## **5. How to Run the Script**

### **Prerequisites**

1. Install Python 3.x on your system.
2. Install dependencies:  
    pip install selenium webdriver-manager python-dotenv

### OR

pip install -r ./requirements.txt

### **Steps to Execute**

**Set up credentials** in a .env file:  
 DISCORD\_EMAIL=your\_email

DISCORD\_PASSWORD=your\_password

1. Add Discord Server URLs in the server\_urls list.
2. Add Channel URLs in the channel\_urls list.
3. Run the script:  
    python discord\_scraper.py
4. The extracted data will be saved in **discord\_data.json**.

## **6. Test Plan & Results**

### **Test Plan**

#### **Objective**

* Validate that the script correctly logs in, navigates to servers, extracts data, and handles errors.

#### **Test Scenarios**

| **Test Case** | **Expected Outcome** |
| --- | --- |
| **Login with valid credentials** | Successfully logs into Discord. |
| **Login with incorrect credentials** | Captures failure screenshot and logs error. |
| **Navigate to servers** | Loads server page successfully. |
| **Extract server details** | Fetches server name, ID, and channels. |
| **Extract member list** | Scrolls & extracts online members. |
| **Extract messages from channels** | Retrieves messages with timestamps & attachments. |
| **Handle offline users** | Correctly stores last active status. |

### **Sample Test Output (JSON Format)**

{

"server\_name": "Example Server",

"server\_id": "123456789",

"channels": [

{

"category": "General",

"channels": [

{

"name": "announcements",

"id": "c1",

"url": "https://discord.com/channels/123/announcements",

"type": "text"

}

]

}

],

"members": {

"groups": [

{"group": "Online", "count": 10},

{"group": "Idle", "count": 5},

{"group": "Offline", "count": 30}

],

"online\_members": [

{"id": "user1", "username": "JohnDoe"}

]

},

"messages": [

{

"username": "Admin",

"timestamp": "2025-01-30T10:00:00Z",

"content": "Welcome to the server!",

"attachments": []

}

],

"last\_active": {

"user1": {

"username": "JohnDoe",

"status": "online",

"last\_seen": "2025-01-30T12:30:00Z"

}

}

}

## **7. Conclusion**

* This **Discord Scraper** automates data collection from **servers, channels, and members**.
* Uses **Selenium & dynamic scrolling** for reliable extraction.
* **Error handling & retry logic** ensure stability.
* **Stores structured JSON output** for easy analysis.