Project Requirements: Instagram Scraper Web Application

Overview

The purpose of this web application is to scrape data from Instagram profiles, followers, followings, and posts/reels using multiple logged-in Instagram accounts. The scraper will handle high-volume data extraction (up to 30,000–40,000 items) while maintaining queue management, concurrency limits, and CSV export functionality.

The tool is **for internal use only** no external authentication or user login is required.

🔹 1. Functional Requirements

1.1 Core Features

1. **Multiple Instagram Account Login**
    - Ability to add, manage, and store multiple Instagram account credentials.
    - The system should rotate accounts automatically to distribute load and prevent rate limits.
    - Show the active/inactive status of each account.
2. **Scraping Options**
    - Input options on frontend:
        - Instagram **profile URL** → choose between "Followers" or "Following".
        - Instagram **post/reel URLs** → can input **multiple links** (via CSV upload or textbox).
    - System should queue each request and execute scraping automatically.
3. **Data Extraction**
    - Data format and fields will follow a **sample CSV file provided separately**.
    - Must support scraping of:
        - Followers list
        - Following list
        - Post/Reel details (caption, likes, comments, views, etc.)
    - Each data extraction job should generate a downloadable **CSV file**.
4. **Queue & Progress System**
    - Each scrape request must be added to a **backend queue**.
    - Frontend should display:
        - Queue position
        - Task status: Queued, In Progress, Completed, Failed

        - Real-time progress indicator (e.g., “Scraping 350/2000 followers…”)
    - Limit concurrent scrapes to avoid account bans and rate-limit issues.
5. **History Management**
    - Store scraping history (task name, target, date, status, and CSV download link).
    - Allow re-downloading of old CSVs from history section.
6. **CSV Output**
    - Each scrape should automatically generate a CSV file containing all data.
    - File naming format:
      scrapeType_target_username_date-time.csv
    - Provide “Download” button on frontend after completion.

---

2. Frontend Requirements

1. **Technology**
    - Developer’s choice of modern web framework (recommended: **React.js** or **Next.js**).
2. **Interface Sections**
    - **Account Manager Panel**
        - Add Instagram account (username, password)
        - View account status (active/inactive, blocked, logged in)
    - **Scrape Input Panel**
        - Option 1: Input Instagram profile URL + choose (“Followers” / “Following”)
        - Option 2: Input multiple post/reel URLs via textbox or upload CSV
        - Start button → sends request to backend queue
    - **Queue Status & History Panel**
        - Real-time display of queued and active tasks
        - Progress bar for ongoing tasks
        - Completed tasks table with CSV download buttons
    - **System Logs (optional)**
        - Section to display basic error logs or failed tasks
3. **UX**
    - Minimal dashboard-style interface (clean and data-focused)
    - No login screen required (for internal use only)

- **3. Backend Requirements**

1. **Technology**
    - Recommended: **Python (FastAPI or Flask)** + **Selenium**
    - Database: **PostgreSQL or MongoDB**
    - Task Queue: **Celery / RQ / custom job queue with Redis**
2. **Core Backend Features**
    - **Queue System**
        - All scraping requests should be queued.
        - Each job runs in background worker to prevent blocking.
    - **Concurrency Control**
        - Configurable number of concurrent jobs.
        - Automatically pause/switch accounts if Instagram rate limit is hit.
    - **Selenium Automation**
        - Use headless browser automation (ChromeDriver/undetected-chromedriver).
        - Handle login sessions for each account separately.
        - Must detect if an account gets blocked or requires re-login.
    - **Account Rotation**
        - Automatically switch to next available account when one hits a limit.
    - **Error Handling & Retry Logic**
        - Retry failed jobs up to N times.
        - Log errors in database.
3. **API Endpoints (Sample)**

Method	Endpoint	Description
POST	/api/login-account	Add new Instagram account
GET	/api/accounts	List all accounts + status
POST	/api/scrape/followers	Add profile followers scrape job
POST	/api/scrape/following	Add profile following scrape job
POST	/api/scrape/posts	Add post/reel scrape job

Method	Endpoint	Description
GET	/api/jobs	Get all queued/running jobs
GET	/api/job/:id	Get job status/progress
GET	/api/job/:id/download	Download CSV output

4. **Storage**
    - All scraped data stored temporarily in database or directly saved as CSV.
    - Keep CSV files on server (/downloads)

---

4. Scalability & Anti-Blocking

The scraper must support **high-volume scraping (30–40k profiles/posts)** efficiently.

Developer Decision Required:

- Suggest proxy rotation strategy to prevent bans.
    - Option 1: Residential proxies (e.g., Bright Data)
    - Option 2: Rotating datacenter proxies
- Suggest optimal Selenium configuration (headless, random delays, browser fingerprint rotation, etc.)

---

5. Deployment

1. **Hosting**
    - Application will be hosted on my private VPS/server.
    - Use **Docker** for isolated deployment (recommended).
2. **Ports**
    - Frontend and Backend can run on same domain via Nginx reverse proxy.

---

6. Security

- Store Instagram credentials encrypted in database (AES or bcrypt).
- Use environment variables for encryption key and DB credentials.
- No external access to API endpoints (restrict via IP whitelist or local network).

---

7. Developer Notes

- Sample CSV output file will be provided separately data fields and structure should follow exactly.

- Ensure modular code structure for easy extension (e.g., adding hashtag scraper in future).
- Include detailed logging for every job (start time, account used, total items scraped, failures).
- Avoid heavy JavaScript rendering or long page loads; focus on performance and stability.

🔹 **8. Deliverables**

1. Fully functional web-based scraper (frontend + backend)
2. Configuration guide (proxy setup, Selenium driver setup)
3. Docker deployment setup
4. Source code with comments and documentation
5. Short video walkthrough for UI and execution flow

---

9. Optional Future Enhancements

Feature	**Description**
Auto Re-login	Auto-detect Instagram re-login prompts & refresh cookies
Hashtag Scraper	Scrape posts under specific hashtags
Story Scraper	Download story views/comments
AI-based account health monitor	Detect which accounts are nearing limits