WHALES Bulk Sender - Official Documentation

1. Introduction

WHALES (WhatsApp Human-like Automation & Lead Engagement System) is a powerful, self-hosted application designed for sending automated bulk messages on WhatsApp to unsaved contacts. Born from the need for a free and private alternative to paid API services, WHALES is built with one core philosophy at its center: **account safety through advanced human simulation.**

This tool is designed for small businesses, marketers, and individuals who need to communicate with a large audience on WhatsApp without the tedious process of saving contacts manually. It provides a rich set of features through a simple web-based interface that runs entirely on your own computer, ensuring your data remains completely private.

2. Core Philosophy: Risk Reduction

The biggest challenge with WhatsApp automation is avoiding detection and the subsequent banning of your account. WHALES is engineered from the ground up to mitigate this risk by mimicking the chaotic, unpredictable, and imperfect behavior of a real human user. Every feature is designed not just for functionality, but to make the bot's activity signature on WhatsApp's servers appear as natural as possible.

3. Key Features

The application is packed with features designed for power, control, and safety.

Campaign Management

- **Contact Groups & Management:** Upload and save contact lists (CSV/Excel) as named groups on the "Contacts" page. View the contents of any group and manage your lists easily.
- **Automatic Resume:** The system automatically tracks the sending progress for each contact group. If a campaign is stopped, the next time you run it for that group, it will intelligently start from the last unsent number, preventing duplicate messages. Progress can be reset manually.
- **Real-Time Dashboard:** Monitor campaigns with a visual progress bar and live statistics (Sent, Failed, Total). The dashboard also displays lifetime and daily sending stats.
- **Pause & Resume:** Manually pause any running campaign directly from the dashboard and resume it later from the exact point you left off.
- **Batch Sending:** Break large campaigns into smaller, more natural batches. The system

waits for your manual click before starting the next batch, adding a genuine user interaction event.

Human Simulation & Security

- **Adaptive Typing Simulation:** The bot calculates a realistic typing duration based on the length of the message, bounded by your custom min/max settings. This avoids fixed, robotic typing times.
- **Copy/Paste Simulation:** Randomly chooses between "typing" a message and "pasting" it, adding another layer of unpredictability to the bot's behavior.
- **Randomized Delays:** Waits for a random duration (e.g., 30-90 seconds) between each message to break predictable sending patterns.
- **"Warm-Up" Mode:** Protect new or inactive accounts by automatically starting with a low daily sending limit and gradually increasing it over several days. You have full control over the starting amount, increment, and duration.
- **"Simulate Reading" Activity:** Programmatically pauses between batches to open and "read" other chats, mimicking real user behavior and making the session appear less focused on just sending.

Data & Template Management

- **Advanced Template Engine:** Create and save reusable message templates with a full emoji picker.
- **Spintax Support:** Use spintax format (e.g., {Hello|Hi|Greetings}) in your templates to generate unique message variations for every contact, drastically reducing spam filter detection.
- **Media Library:** A central library for all your images, PDFs, and documents, complete with image previews and file renaming capabilities. Renaming a file automatically updates all templates that use it.
- **Interactive Reports:** View detailed reports for completed campaigns, filter results by status (Sent, Failed, etc.), and download professional PDF summaries.
- **Confirmation Modals:** All destructive actions (like deleting templates, media, or reports) require user confirmation to prevent accidents.
- **Full Data Privacy:** The entire application and all your data (contacts, templates, reports) are stored and run 100% locally on your machine.

4. End-to-End Workflow

This section describes the typical journey of using the WHALES application from start to finish.

1. **Preparation (One-Time Setup):**
    - **Upload Media:** The user navigates to the "Media Library" and uploads all necessary images, PDFs, or documents.
    - **Create Contact Groups:** On the "Contacts" page, the user uploads CSV or Excel files, which are saved as named groups.

    - **Design Templates:** On the "Templates" page, the user creates message templates, utilizing spintax for variation, personalization tags like {name}, the emoji picker, and attaching files from the Media Library.
2. **Starting the Application:**
    - The user runs the npm start command in their terminal.
    - The Node.js server starts, and the user accesses the UI at http://localhost:3000.
    - The backend launches a headless Chromium browser and attempts to log into WhatsApp. If not logged in, a QR code is generated and sent to the UI.
    - The user scans the QR code with their phone, authenticating the session. The session data is saved locally for future runs.
3. **Configuring a Campaign:**
    - On the Dashboard, the user selects a Contact Group or uploads a new list.
    - They choose a pre-made Template from the dropdown.
    - They configure the sending parameters: daily limits, batch sizes, and the min/max delays between messages.
    - They fine-tune the human simulation settings: typing speed, simulation style (typing/paste/random), and whether to enable "Warm-Up" mode or "Simulate Reading."
4. **Execution & Monitoring:**
    - The user clicks "Start Campaign."
    - The backend reads the contact list, noting any saved progress for that group, and begins the sending loop from the correct starting index.
    - For each contact, the bot performs its simulation (waits, simulates typing, etc.), processes the spintax in the template, and sends the message.
    - The user monitors the progress in real-time on the dashboard via the progress bar and the live log.
    - After each batch is complete, the system pauses and waits for the user to click "Start Next Batch." The user can also manually Pause/Resume at any time.
5. **Completion & Analysis:**
    - Once the campaign is finished (or stopped), a detailed report is saved as a CSV file in the app/data folder.
    - The progress for the contact group is automatically reset to zero, ready for a future campaign.
    - The user navigates to the "Reports" page to view a list of all generated reports, view the data in a table, filter it by status, and download a PDF summary for their records.

5. Project File Structure

The project is organized into a clean and logical directory structure to separate concerns.

```
/whatsapp-bulk-sender/
|
|-- /app/                -- Core backend logic
```

```
|   |-- /contacts/          -- Stores saved contact group files (CSV/Excel)
|   |-- /data/              -- Stores JSON data like templates, stats, and reports
|   |-- /media/             -- Stores uploaded images, PDFs, etc.
|   |-- /session/           -- Stores your WhatsApp login session to avoid re-scanning
|   |-- server.js           -- The main Express web server file.
|   `-- whatsapp-client.js   -- All automation logic using whatsapp-web.js.
|
|-- /node_modules/          -- All downloaded libraries (created by `npm install`)
|
|-- /public/                -- Frontend assets accessible by the browser
|   |-- /css/
|   |   `-- style.css       -- All visual styling for the application.
|   `-- /js/
|       `-- main.js         -- All frontend interactivity and communication logic.
|
|-- /uploads/               -- Temporary storage for uploaded contact files.
|
|-- /views/                 -- EJS template files that create the HTML pages
|   |-- /partials/          -- Reusable UI components (header, footer, nav)
|   |-- contacts.ejs        -- The "Contacts" page.
|   |-- dashboard.ejs       -- The main "Dashboard" page.
|   |-- media.ejs           -- The "Media Library" page.
|   |-- reports.ejs         -- The "Reports" page.
|   `-- templates.ejs       -- The "Templates" page.
|
|-- .wwebjs_cache/          -- Cache for the whatsapp-web.js library.
|
|-- package.json            -- Defines project dependencies and scripts.
|-- package-lock.json       -- Records exact versions of dependencies.
`-- README.md               -- This documentation file.
```

6. Technology Stack & System Architecture

The application operates on a three-layer architecture:

1. **Frontend (User Interface):** The visual interface you interact with in your browser (http://localhost:3000).
    - **Technology: EJS (Embedded JavaScript)** for generating dynamic HTML, **CSS** for all styling, and **client-side JavaScript** for all user interactivity.
    - **Key Libraries:** jsPDF for PDF exporting, emoji-picker-element for the emoji menu.
2. **Backend (Web Server):** The central "brain" of the application that runs on your

computer.
    - **Technology: Node.js** as the runtime environment and **Express.js** as the web framework.
    - **Responsibilities:** It serves the frontend pages, manages a RESTful API for all data operations (CRUD for templates, media, reports, contacts), and acts as the command center for the automation logic.
3. **Automation Core (The Bot):** The layer that performs the actual WhatsApp automation.
    - **Technology:** The powerful **whatsapp-web.js** library, which programmatically controls a headless instance of the **Chromium** browser via Puppeteer.
    - **Responsibilities:** It is this hidden browser that logs into WhatsApp Web, maintains the session, and executes all the automated actions like finding chats, typing messages, attaching files, and clicking "send".

Real-time communication between the backend and frontend (for the QR code, live log, and campaign progress) is handled by **Socket.io**.

7. Risk Reduction Strategy in Detail

This section outlines the primary risks associated with using this application and the specific features built into WHALES to mitigate them. The impact of any violation is always **High** (temporary or permanent account ban), so the goal of these features is to significantly lower the **probability** of detection.

Risk Category	Specific Risk	Impact	Mitigation in WHALES
Account Safety	**Violation of WhatsApp's Terms of Service**	High	This is an inherent risk of any automation. WHALES is designed for **educational purposes**, and all subsequent features are built to minimize the chances of a violation being detected.
Account Safety	**Algorithmic Detection of Robotic Behavior**	High	This is the main threat. WhatsApp's systems look for

			non-human patterns. WHALES counters this with a suite of features: **Random Delays**, **Adaptive Typing Speed**, **Random Copy/Paste Simulation**, **Batch Sending**, and **Simulated Reading Activity**. Together, these create a chaotic, unpredictable pattern that is hard to distinguish from a human.
Account Safety	**Sending Too Much, Too Soon**	High	A new or inactive account suddenly sending hundreds of messages is a major red flag. The **"Warm-Up" Mode** directly mitigates this by starting with a low, user-configurable daily limit and gradually increasing it over several days, mimicking a natural usage pattern.
Account Safety	**Sending Repetitive Content (Spam Flag)**	High	Sending the exact same message to many users is a classic spamming pattern. The **Spintax Support**

			feature allows you to create message templates with variations (e.g., `{Hi
Account Safety	**User Reports**	High	If recipients report your messages as spam, your account will be flagged. This is a non-technical risk that depends on your content and contact list. WHALES helps by allowing for personalization ({name}) and varied messages (Spintax) to make your content feel more genuine and less like spam.
Technical	**WhatsApp Web Updates**	Medium	If WhatsApp changes its website code, the bot may stop working. WHALES mitigates this by using a popular, actively maintained library (whatsapp-web.js). Performing a "Clean Build" and reinstalling dependencies often resolves these issues quickly.
Operational	**Data Integrity Issues**	Low	Incorrectly formatted contact

			files (e.g., Excel's scientific notation) can cause campaigns to fail. The application includes robust error checking to detect if a contact file is empty or unreadable and will alert you in the log instead of failing silently.

8. Getting Started

Follow these steps to set up and run the application.

1. **Prerequisites:** Ensure you have **Node.js** installed on your computer.
2. **Clean Build:** For the best results, start with a clean slate by deleting the node_modules, app/session, .wwebjs_cache, and uploads folders, as well as the package-lock.json file.
3. **Install Dependencies:** Open a terminal in the project folder and run the command: npm install. This will download all required libraries, including the Chromium browser.
4. **Run the Application:** Once the installation is complete, run the command: npm start.
5. **Access the UI:** Open your web browser and navigate to **http://localhost:3000**. You will be greeted with a QR code to begin.

9. Disclaimer

This application is provided for **educational and research purposes only**. Automating WhatsApp services is a direct violation of their Terms of Service. The developers of this tool are not responsible for any consequences of its use, including but not limited to the temporary or permanent banning of your WhatsApp account. **Use at your own risk.**