



SortSwift: Documentation & Technology Overview

Overview of SortSwift

SortSwift is an all-in-one software platform designed for trading card game (TCG) and hobby store operations. It offers integrated solutions for managing card inventory, pricing, point-of-sale (POS), marketplace syncing, and especially **high-speed card scanning and identification** ¹. The system is tailored to help users (like card shop owners or large collectors) **scan and catalog large volumes of trading cards quickly and accurately**, then organize and list those cards for sale. In short, SortSwift streamlines the once-manual processes of sorting, identifying, pricing, and uploading cards into a mostly automated workflow.

Some key features of SortSwift include:

- **Bulk Card Scanning & Recognition:** Fast image-based identification of cards across many TCGs, with **AI-driven accuracy** (including detection of card variants like foils or edition markings) ².
- **Inventory Management:** Tools to organize and track cards (support for most major TCGs, with front/back images, storage locations, etc.)
- **Auto-Pricing and Catalog Data:** Automated price updates using market data, and a comprehensive card catalog/database covering multiple games.
- **Marketplace Integration:** Syncing inventory and sales with platforms like TCGplayer, CardTrader, Shopify, etc. ³.
- **Point of Sale & Other Tools:** An in-store POS system, CSV import/export utilities, buylists, and more to support card shop operations.

Overall, SortSwift serves as a **central hub for TCG businesses**, combining hardware and software to dramatically speed up how cards are sorted, identified, and listed for sale.

Architecture: Front-End, Back-End, and Mobile

Front-End: The main user interface for SortSwift is a web-based application (accessible via app.sortswift.com) that requires login. This web front-end is where users manage their inventory, adjust pricing, review scan results, print labels/picklists, and perform other operations. The interface is likely built with modern web frameworks (the marketing site references Webflow/React) and is optimized for desktop use, since managing thousands of cards or running a POS is easiest on a full screen. The web app provides dashboards and tools for all the core features (inventory, scanning, POS, etc.) in one place.

Back-End: SortSwift's back-end is a cloud-based service that handles data storage, business logic, and integration with external APIs. This includes:

- A **database** of cards and inventory records (often called the "**Catalog**" feature in SortSwift) which contains card details for each supported game (name, set, edition, etc.) and possibly references to pricing. SortSwift maintains an internal catalog covering **9+ TCGs (now expanded)**

to 20+ games), ranging from Magic: The Gathering, Pokémon, Yu-Gi-Oh! to newer games like Disney Lorcania, One Piece, Flesh and Blood, etc. This internal database was likely seeded from major sources (e.g. TCGplayer's catalog or open databases) and is kept updated. Having its own copy of card data allows SortSwift to quickly identify cards and list them without constantly calling external APIs.

- **Image recognition services** for card scanning (discussed in detail below) – likely implemented as a specialized microservice or module in the back-end that processes uploaded card images using AI models to return identifications.
- **Integration modules** that communicate with external platforms. SortSwift can sync inventory and orders with third-party marketplaces: for example, it connects to TCGplayer and CardTrader to push stock or fetch pricing, to Shopify for web store integration, etc. ³. These modules use the APIs provided by those platforms (such as TCGplayer's API) to send and receive data (e.g. updating quantities, fetching current market prices, etc.). This way, any cards scanned into SortSwift can be automatically listed or updated on multiple sales channels.
- **Business logic** for features like auto-pricing (which likely uses rules or algorithms to adjust prices based on market data), buylist management, and other tools. The back-end performs scheduled tasks such as daily price updates (if subscribed), generating reports/exports, and handling user management and billing for the subscription plans.

In terms of technology stack, specifics aren't publicly documented, but one can infer that SortSwift's back-end is built to handle real-time image processing and database transactions at scale. It likely uses a combination of cloud services (for storage of images and data) and possibly machine learning frameworks (for the card recognition). The mention of "AI Card Scans" suggests a deep learning model is involved on the server side ². The back-end must also be highly reliable, as it interfaces with physical hardware (scanners/sorters) and needs to update inventory records accurately as cards are processed.

Mobile: In addition to the desktop-oriented web app, SortSwift introduced a **mobile scanning application** for on-the-go use. This was initially launched as a mobile web app (accessible via scanner.sortswift.com in a smartphone browser), and they have also announced native iOS/Android apps (as of a 2023 update) pending app store approval. The mobile app isn't a full version of the SortSwift management interface; rather, it's focused on **card scanning via the phone's camera** and basic list/collection building. Essentially, the mobile app lets users quickly scan cards with their phone, identify them, and build lists or add to inventory from anywhere. Key benefits of logging in with a SortSwift account on the mobile scanner include syncing those scans to your collections/inventory on the cloud, saving scan sessions, etc.

On the technical side, the mobile scanning app likely uses the phone's camera feed and either runs a lightweight on-device recognition model or (more likely) uploads the images to SortSwift's server for identification. Given that SortSwift monetizes scans and emphasizes AI accuracy, the workflow is probably: the phone app captures card images (or a live video feed frame), sends them to the back-end, which returns the identified card info to the app. This ensures the same robust recognition engine is used on mobile as with the high-speed scanners. The mobile interface then allows the user to confirm or correct the identifications and save them. By offering a web-based mobile app first (PWA) and then native apps, SortSwift ensures broad availability – users can just visit the URL to start scanning, which lowers entry barriers. (The native apps likely will wrap the same functionality with possibly some offline features or camera optimizations.)

In summary, **SortSwift's architecture** comprises a powerful cloud back-end (with the database and AI services), a feature-rich web front-end for management, and a specialized mobile/web front-end for

camera scanning. This multi-front-end approach means users can input cards either through **bulk scanner hardware or simply a phone camera**, all feeding into the same back-end system.

Card Scanning & Recognition Mechanism

One of the most critical and innovative parts of SortSwift is its **card scanning and recognition system**. The goal is to take a physical trading card (or many cards) and identify exactly which card it is (including game, set, edition, etc.) with minimal manual effort. SortSwift accomplishes this using a combination of **specialized hardware, high-resolution imaging, and AI-driven computer vision** to recognize cards.

Supported Hardware and Setup

For high-volume scanning, SortSwift leverages professional document scanners. The recommended model is the **Ricoh (Fujitsu) fi-8170** high-speed document scanner ⁴ (though many Fujitsu-branded scanners are supported). These scanners can rapidly feed and scan cards (similar to how a sheet-fed scanner works) at a rate of dozens of cards per minute. SortSwift provides a hardware kit called the “**Simple Sifter**” which includes the scanner and a small controller: a **Raspberry Pi 4** mini-computer that acts as the “brain” of the scanning station ⁵. The Raspberry Pi connects to the scanner and runs SortSwift’s scanning client software.

In this setup, when you load cards into the scanner hopper, the Pi will control the scanner to pull each card, capture its image, and then automatically upload those images to the SortSwift cloud for recognition. This “**Direct-to-Web**” integration means you don’t have to manually drag image files or use a PC interface – the hardware pipeline feeds cards directly into your online inventory via SortSwift’s software ². The Pi is pre-configured with a SortSwift image (software) that customers install on an SD card (SortSwift provides a “*Simple Sifter*” OS image and tutorial for setup). Once running, it essentially turns the scanner into a smart IoT device for cards.

For lower volumes or more casual use, SortSwift also supports simpler capture devices: in earlier versions, they provided a **desktop application** that you could install on a PC to use a standard webcam as a scanner. In that mode, a user could hold cards up to a webcam and the software would detect and identify them (this is slower and one-card-at-a-time, but requires only a webcam). The company’s new **mobile app** similarly uses the phone camera to snap cards one by one. These options make scanning accessible without investing in the full scanner setup, though at a cost of speed. (The **fastest throughput** is achieved with the dedicated document scanner – feeding a stack of cards – as opposed to manually positioning cards in front of a camera.)

Image Recognition Technology

Regardless of capture method, the core of SortSwift’s scanning system is the **image recognition AI** that identifies cards from their photos. While the exact algorithms are proprietary, we can outline how it works and what capabilities it has:

- **AI Card Identification:** SortSwift’s system uses advanced computer vision (likely deep learning models trained on card images) to recognize a card’s identity from the image. It doesn’t rely on barcodes or QR codes; instead, it “sees” the card’s face and matches it to a known catalog entry. SortSwift advertises extremely high accuracy in identification – reportedly the system remains accurate **even if a large portion of the card is obscured or cropped** (for example, even if 2/3 of a Yu-Gi-Oh! card was covered, it could still correctly identify it, according to their demonstrations). This implies the model has learned to focus on distinctive visual features of the

card (artwork, layout, text patterns) and can tolerate partial views or poor lighting. They describe the accuracy as “*insane*”, with the software being very confident in its matches. The recognition works for **20+ different TCGs** (as of 2024) – it initially launched with about 9 games and expanded to more than twenty, including all major ones like MTG, Pokémon, Yu-Gi-Oh!, Flesh and Blood, One Piece, Dragon Ball Super, Digimon, Magic *Lorcania*, etc., and continues to add support for new games. Each game’s cards have been incorporated into the training data and catalog.

- **Card Variants & Attributes:** The recognition engine doesn’t just identify the card name – it often can pinpoint the exact edition or variant of the card. For example, SortSwift’s scanning can **distinguish foil (holographic) cards vs. non-foil**, detect a “**1st Edition**” stamp or other edition markings on cards (important for games like Yu-Gi-Oh! or early Pokémon), and even recognize different **languages** for certain games ⁶. This is quite sophisticated: detecting a 1st edition might involve OCR (optical character recognition) to read text like “1st Edition” on the card, and foil detection might involve analyzing the lighting/reflection on the card surface. The system likely uses a combination of image classification and targeted detection – e.g. it might first classify the card’s identity by artwork, then examine specific regions (like where a edition symbol or collector number is) to refine which printing it is. Not all games have such variants, but SortSwift tries to account for those that do. (Their marketing notes that foil/edition detection is available for “select games,” meaning it’s implemented where applicable.)
- **Front/Back Scanning:** The platform supports **back-image scanning** ², meaning if a card has important information on the back, the system can handle it. In practice, for most TCGs the card back is generic (not needed for ID), but this feature is useful for games or scenarios where the back matters (e.g. double-faced cards in Magic: The Gathering – the scanner can scan both sides in one pass, and SortSwift will recognize a double-faced card properly by its front/back combination). It could also help verify things like card condition or authenticity if needed, but the main use is likely double-faced cards and perhaps sports cards (front/back difference).
- **Cloud Processing & Speed:** When a batch of card images is scanned and sent to the server, the recognition happens very quickly. SortSwift’s AI is designed for speed and scale – one testimonial and demo showed **102 Magic cards being uploaded, identified, and verified in under 1 minute** (using a high-speed scanner) which gives a sense of performance. The heavy lifting is done server-side on presumably GPU-accelerated processes. SortSwift’s scanning subscription plans actually meter the number of scans (e.g. a plan might include 20,000 scans/month) – this suggests each image processed has a cost, likely due to the computational load of the AI. (On the highest-tier plan, if you use the **Simple Sifter hardware**, they allow *unlimited* scans ⁷, likely because a serious store with that hardware will be scanning huge volumes and they price the subscription accordingly.)
- **Accuracy and Error Handling:** The system strives for near 100% accuracy but also provides a workflow for *error correction*. SortSwift indicates there is “instant error correction” and ways to catch mistakes ². In practice, after the software identifies each card, the results are presented to the user for verification. Cards that the AI is very confident about might be automatically confirmed, whereas any uncertain identifications can be flagged for review. SortSwift likely has a confidence rating internally (similar to TCGplayer’s Roca vision which labels matches as Good/Fair/Poor confidence ⁸ ⁹). If a card is not confidently recognized (or not recognized at all), it will appear in an “unidentified” list for the user to manually review. The user interface provides quick tools (like dropdown suggestions or search) to correct any mis-identified card. Notably, SortSwift built keyboard shortcut (“hotkey”) support to speed up this verification step – experienced users can go through a list of scanned cards and press keys to confirm or adjust

entries very rapidly. One shared workflow showed a user verifying ~100 cards in about 5–6 minutes using keybinds, which is far faster than typing each name ¹⁰. This human-in-the-loop step ensures that any errors are caught immediately, and it's "instant" in the sense that as soon as you correct a card's identity, it's updated in the system. (It's possible that SortSwift's system even learns from corrections or at least tracks them to improve the AI over time.)

- **Alerts and Data Enrichment:** Once cards are identified, SortSwift can automatically attach additional data. For example, it has **High-Value card alerts** – if a scanned card is worth over a certain threshold or is on a known reserve list (for Magic), the system will alert you ¹¹. This is useful so you don't accidentally put a very expensive card in the bulk pile. Also, since the cards are matched to specific catalog entries, things like the card's current market price, set code, etc., are now linked and can be used for pricing or listing immediately.

After identification and any user confirmation, the cards can be **automatically added to inventory** with all their details. SortSwift essentially turns a pile of mixed cards into a digital list of specific card records in seconds, ready for sale. Users can then set quantities, conditions, or pricing rules and push the listings online.

Performance and Scaling

With the combination of proper hardware and the AI, SortSwift achieves very high throughput: A single **Simple Sifter (scanner+Pi)** can process on the order of **hundreds of cards per minute**. The upcoming "**Super Sorter**" machine (a more advanced automated sorter developed in partnership with MTech Cave) pushes this to the next level – it not only scans cards but also physically sorts them into 29 output bins. The Super Sorter uses SortSwift's recognition engine as well. The prototype of the machine runs about *1,200 cards per hour* and is expected to reach **2,500–3,600 cards per hour** with optimizations ¹¹. That's roughly **40–60 cards per minute** continuously sorted, including the mechanical actions. The final production target is around **3,000 cards/hour (with 29 bins)** in real use ¹² ¹³. This demonstrates that SortSwift's software can keep up with very rapid scanning – as cards are fed in and images captured, the system identifies each almost in real-time, fast enough that the bottleneck becomes the mechanical sorting speed, not the recognition.

The **accuracy** of the recognition remains a highlight. In the context of the Super Sorter, it's noted that SortSwift's software provides "*unparalleled accuracy in card scanning*", which is essentially the "heartbeat" of the whole machine ¹⁴. Without reliable identification, an automated sorter wouldn't function correctly. SortSwift's tech is what empowers these machines to correctly route each card to the right bin or inventory category. The tight integration of hardware and software here also shows how the back-end can communicate in real-time with devices (for example, the sorter likely queries the cloud service for each card image to decide which bin corresponds to that card's category, and does so extremely fast).

To summarize, the scanning mechanism in SortSwift works as follows:

1. **Capture** – Cards are scanned via a high-speed document scanner (or photographed via camera) and sent to the SortSwift system. In automated setups, a Raspberry Pi or similar controller handles this upload seamlessly ⁵.
2. **Recognition** – SortSwift's cloud service receives each image and runs it through an AI model to identify the card's game, name, set, and any special attributes (foil, edition, etc.). This is done using computer vision algorithms trained on a vast image dataset of TCG cards, yielding very high accuracy even under suboptimal conditions.

3. **Result & Verification** – The identified card entries are returned to the user interface. The user can then quickly verify and correct any that are uncertain. Thanks to confidence metrics and a refined UI with shortcuts, this step is streamlined (“instant error correction” means the user can fix an ID with one click or keypress and move on) ².
4. **Integration** – The confirmed cards can then be auto-organized: added into the user’s inventory with correct details, grouped for export, or queued for listing on marketplaces. The system can also prompt actions like printing labels for the cards or alerting the user to high-value finds. All images are **saved to cloud storage** under the user’s account for record-keeping or later review ² (for example, a store owner could refer back to the scan images if needed to verify condition or proof). Additionally, if using the physical sorter, the software’s identification is used in real-time to direct cards into physical bins sorted by whatever criteria (set, game, etc. as configured).

This end-to-end pipeline significantly cuts down the labor involved in processing large card collections. What used to take hours of manual lookup and data entry can now happen in minutes with SortSwift. The combination of **AI recognition** and cleverly integrated hardware is what makes this possible, and SortSwift’s technology is at the forefront of this in the TCG industry (comparable to, and by some accounts exceeding, the capabilities of other solutions like TCGplayer’s Roca Vision or Dragon Shield’s scanning app).

Data Management and Integration (APIs & Database)

Since the user specifically is interested in using an API and eventually building a personal database of cards, it’s important to understand how SortSwift handles data. SortSwift essentially sits between the raw card data (provided by external sources) and the user’s inventory, acting as an aggregator and enhancer.

- **Internal Card Database (“Catalogs”):** SortSwift maintains its own internal catalog of cards for each supported game. This includes card names, variations, set information, likely unique IDs linking to marketplace listings, etc. This database is what the scanning AI uses to match images to the correct entry. SortSwift’s catalog was likely built by pulling data from established sources – for instance, TCGplayer’s catalog API provides a full list of card objects for games they support, and other games might use open databases (like Scryfall for Magic: The Gathering, or official card lists). By aggregating these, SortSwift can have a comprehensive reference. Over time, this internal database can be expanded and corrected, becoming a proprietary asset. (The pricing page even mentions “Additional Catalog Sources” as a future add-on ¹⁵, including sports cards, coins, etc., indicating SortSwift plans to broaden the types of collectibles in its database.) For a developer building their own app, the takeaway is that having a robust **card data repository** is crucial – initially you might rely on an external API, but eventually you’ll want your own database so you’re not dependent on others and can add custom fields or handle offline queries.
- **Pricing Data and Autopricing:** SortSwift does not operate its own marketplace, so it uses external market data to price cards. It integrates with multiple **pricing sources** – notably TCGplayer (a major marketplace) and CardTrader, and possibly others like eBay for certain cards (one user testimonial mentioned they used to parse eBay and other sources manually, which SortSwift made unnecessary ¹⁶). SortSwift’s **Autopricing** feature allows users to automatically update their card prices based on rules (for example, match TCGplayer market price or slightly undercut it, etc.). To do this, SortSwift’s back-end regularly pulls pricing info via APIs (e.g., TCGplayer’s API provides daily market price, low price, etc., for each card). SortSwift can then adjust the user’s inventory prices and even push those updates to their online store. This is a powerful feature for sellers with large inventory – ensuring prices stay competitive without manual effort. It also implies that **SortSwift’s database stores price fields** for cards (updated

perhaps daily or in real-time) so that when a card is scanned or viewed, you can immediately see its current value.

- **Inventory & Sync:** When a user adds cards to their inventory in SortSwift (either by scanning or by import), those records reside in SortSwift's database (with quantities, condition, etc. as input by the user). The **Syncing** modules then take that data and send it to other platforms the user connects. For example, if a user has a TCGplayer seller account, SortSwift can use the TCGplayer API to push the newly added cards (with correct set and card IDs) and quantities to their TCGplayer inventory, rather than the user having to upload a CSV manually. SortSwift supports syncing with **TCGplayer (semi-automated)**, CardTrader, Shopify (for web stores), and even a platform called ManaPool ³. "Semi-automated" for TCGplayer likely means TCGplayer requires a file upload or confirmation step (as TCGplayer's platform sometimes doesn't allow direct full automation for inventory updates, so SortSwift might prepare the data and the user clicks confirm). In any case, the integration ensures that once the data is in SortSwift, it doesn't have to be re-entered elsewhere – the software acts as the central point of truth.
- **Exports and API access:** SortSwift also provides **CSV/PDF export** capabilities ². This means users can export their inventory or scan results in standard formats if they want to use them elsewhere. For instance, after scanning, you could generate a CSV list of all identified cards with quantities and prices. This is helpful if a user wants to import the data into another system or just keep records. While not a public API in the traditional sense, these export tools give flexibility in accessing the data outside the app. (SortSwift likely does not offer a public developer API for third parties, since it's a commercial product, but the user's need here is more about internal use of data.)
- **Database building considerations:** The user mentioned they currently rely on an external API and plan to build their own DB. SortSwift's approach shows a typical migration path: leverage external data sources to fill your database initially, then maintain that database internally with periodic updates. Over time, your DB becomes authoritative for your application, and external APIs might only be used for things like price refresh or adding new set releases. SortSwift essentially sits on a constantly updated trove of card data. The inclusion of features like custom remarks, storage locations, etc., in their Inventory module suggests they extend the base data with user-specific info. So, architecturally, having a **central database** that links to multiple external systems (one-to-many) is key. SortSwift can pull *in* data from sources (like prices from TCGplayer) and push *out* data (like inventory listings to marketplaces) – acting as a bridge.

From a technology standpoint, maintaining data consistency across these integrations is non-trivial. SortSwift likely uses unique identifiers (like a TCGplayer product ID or a universal card ID) to map cards between its database and external platforms. For example, when the AI identifies a card, it probably attaches the TCGplayer catalog ID for that card, so that if you sync to TCGplayer it knows exactly which item to update. Similarly, for CardTrader, it would map to their ID system. For a custom app, one would need to decide on a canonical ID system (maybe use an existing one or create your own keys and store mappings to others). SortSwift's success in syncing *both* TCGplayer and CardTrader simultaneously ¹⁷ suggests they manage multiple IDs per card internally.

Finally, **security and scaling**: Since images and data are uploaded to SortSwift's cloud, they must handle security (likely using user auth tokens, etc.). The Raspberry Pi scanner software probably uses API keys or login credentials to connect to the account. The images stored could be sizable, so cloud storage (like AWS S3 or similar) is probably used, and they might clean up images after some time or keep them for a certain period (not explicitly stated, but "Automatic Cloud Storage" implies they keep your scans, which could be useful for audit or if you need to verify what was scanned). The ability to

handle spikes of thousands of images in a short time means the back-end likely has auto-scaling services and load balancers, especially as their client base grows.

Conclusion and Notable Points

SortSwift marries hardware and software to tackle a once labor-intensive task in the TCG industry: sorting and listing cards. Technologically, it stands on a foundation of **computer vision (AI)** for image recognition, a robust **central database** of card information, and a rich set of integration capabilities (APIs to marketplaces, etc.), all delivered through user-friendly front-ends (web dashboard and mobile app). The front-end/back-end split allows heavy computation and data work to happen on the server, while users get responsive tools to manage their collections.

To recap some important technical aspects:

- The **front-end** includes a web application for full functionality and a dedicated mobile/web app for scanning via camera, ensuring cross-platform access.
- The **back-end** handles AI card identification, data storage, and sync – essentially the brains of the operation – and leverages external data sources for card info and pricing while building an independent, comprehensive card database.
- The **scanning mechanism** uses high-speed document scanners (or cameras) plus SortSwift's **patent-pending** recognition software to achieve extremely fast and accurate card identification. SortSwift's software can identify cards from partial images, differentiate variations like foils and languages, and currently supports over twenty different TCGs – a breadth that showcases the extensibility of their AI model and database ¹⁴ ¹⁸. It integrates tightly with physical devices (like the Simple Sifter station or the 29-bin Super Sorter), essentially automating end-to-end card processing.
- The system provides for **verification and error correction**, acknowledging that no AI is perfect but making it easy for users to fix any misread card on the spot. The design philosophy is to minimize the user's effort while maximizing throughput and accuracy.
- SortSwift uses a **subscription model**, where heavy users can scan tens of thousands of cards per month. The highest tiers allow unlimited scanning when using their hardware, indicating that part of their strategy is to offer an integrated hardware-software solution for power sellers.
- On the data side, SortSwift is moving towards being a one-stop ecosystem (their term "Ultimate TCG Ecosystem" fits) – they want to hold all the necessary data (card details, inventory, prices) so that a card shop doesn't need any other software. They've even branched into things like event management, loyalty programs, etc., but those are ancillary. At its core, the **card recognition tech and inventory management** are what differentiate SortSwift from generic inventory systems.

Notably, SortSwift emerged as an alternative to other solutions like BinderPOS or TCGplayer's tools, and customers report it being *faster, more accurate, and more responsive* to feature requests ¹⁹. Technologically, this means SortSwift is leveraging modern AI and agile development to outperform older competitors. The founder (Mikah Walters) and team iteratively improve the software, as evidenced by rapid updates (for example, adding new games support, improving sorting algorithms, etc.). They even have a **patent-pending** on their sorting machines, which suggests unique innovations in how the system handles cards.

For someone building their own application: the SortSwift case study highlights the importance of combining **scanning hardware, AI-based image recognition, and a well-structured database**, plus offering both **desktop and mobile interfaces** for flexibility. Each of these components in SortSwift is finely tuned – from using a Raspberry Pi to offload scanning tasks, to training AI models for each game,

to syncing with multiple APIs. The result is a highly efficient system that can take a pile of unsorted cards and quickly turn it into organized digital inventory ready to sell. The technical documentation and marketing materials of SortSwift emphasize these points, and the success of the platform indicates that such an integrated approach is very effective for the problem at hand.

Sources: SortSwift Official Website (features & pricing) [20](#) [4](#) [5](#); MTech Cave – Super Sorter collaboration notes [14](#) [21](#); TCGplayer Help (for context on scanning tech) [8](#). (Additional information inferred from SortSwift product demos and user testimonials for completeness.)

[1](#) [2](#) [3](#) [7](#) [15](#) [20](#) Pricing

<https://www.sortswift.com/pricing>

[4](#) [5](#) Hardware

<https://www.sortswift.com/hardware>

[6](#) [10](#) Here is a Scanning Workflow that I use using the Simple Sifter You ...

<https://www.facebook.com/61561097535885/posts/heres-a-scanning-workflow-that-i-use-using-the-simple-sifter-youreable-to-scan/122134568816369917/>

[8](#) [9](#) How Scan & Identify Technology Works – TCGplayer.com

<https://help.tcgplayer.com/hc/en-us/articles/27303183354007-How-Scan-Identify-Technology-Works>

[11](#) [12](#) [13](#) [14](#) [21](#) The Super Sorter — MTech Cave

<https://mtechcave.com/pages/the-super-sorter>

[16](#) [17](#) [19](#) SortSwift – All-in-One Game Store Management Software

<https://www.sortswift.com/>

[18](#) We covered 2/3 of a Yu-Gi-Oh! card... and SortSwift STILL scanned ...

<https://www.facebook.com/61561097535885/videos/we-covered-23-of-a-yu-gi-oh-card-and-sortsift-still-scanned-it/655584790442214/>