ChatGPT

Pokémon Card Investment Platform – Software Requirements Specification (SRS)

Introduction

Purpose: This Software Requirements Specification (SRS) outlines the comprehensive requirements for the **Pokémon Card Investment Platform**, a web-based application (with planned mobile app and Discord bot integration) that enables collectors, investors, and resellers of Pokémon trading cards to track card values, analyze market trends, and manage their collections like a financial portfolio. The purpose of this document is to detail what the final product will look like and the functionality it must deliver, serving as a reference for stakeholders and the development team through the entire product lifecycle.

Scope: The platform (initially a website) will provide a rich feature set similar in spirit to a Bloomberg terminal but for Pokémon cards – including a searchable card database with images and real-time price data, detailed analytics for each card (price history charts, volatility, buy/sell signals, etc.), portfolio management tools, watchlists and price alerts, and advanced analytical tools like backtesting for premium users. Future expansions (beyond the MVP) will include integration with Discord (a bot for retrieving prices/alerts in Discord communities) and native mobile applications (with features like card scanning via camera). All features will be designed with these future integrations in mind. Ultimately, this platform aims to be the **#1 destination** for tracking Pokémon card **collecting and investing**, providing data-driven insights and up-to-date information unparalleled by any existing solution.

Background & Motivation: In recent years the Pokémon card market has surged dramatically – for example, trading card sales on eBay climbed 142% in a single year, with Pokémon cards leading at an average of five cards sold per minute [1]. New tools and services have emerged for collectors, but none combine **real-time price tracking, advanced analytics, and portfolio management** into a single comprehensive platform. Existing apps like PokeData and TCGTrends offer pieces of this puzzle (e.g. price tracking and portfolio features) [2] [3], and startups like Collectbase/Cardbase have shown the demand for scanning and valuing card collections akin to stock portfolios [4]. This platform seeks to fill the gap by providing **a professional-grade, data-driven experience** for hobbyists and serious investors alike – bringing "quant hedge fund" level analysis to Pokémon cards. By leveraging machine learning and large-scale data collection (from marketplaces, grading services, and even social trends), the platform will deliver accurate price signals (buy/sell/hold) and forecasts to help users make informed decisions. The end goal is a one-stop solution where users can **search any card**, see its market trajectory and investment outlook, and manage their collection's value over time with confidence.

Intended Audience: This SRS is intended for the project stakeholders (product owners, investors) and the development team (developers, designers, data scientists) responsible for building the platform. It will also be useful to any future partners or third-party integrators to understand the platform's capabilities. The end-users of the platform include: - **Collectors/Hobbyists:** Individuals who collect Pokémon cards for personal enjoyment but want to track the value of their collection and perhaps make occasional trades. - **Investors/Traders:** Users who treat Pokémon cards as an investment asset class, actively buying/selling to

profit from market changes, requiring advanced analytics and signals. - **Resellers/Shop Owners:** Those who regularly sell cards (e.g., on eBay or stores) and need up-to-date price data, market trends, and inventory management insights. - **(Future) Mobile/Discord Users:** Community members who may interact via Discord or mobile devices for quick info and alerts on the go.

Definitions, Acronyms, and Abbreviations: (Key terms used in this SRS) - *Card*: A Pokémon trading card (assumed English unless specified otherwise). - *Set*: A specific release/set of the trading card game (e.g., *Base Set*, *Jungle*, *Evolving Skies*). - *Card Number*: The collector number on the card within its set. - *Raw*: An ungraded card in near-mint condition (used as a baseline for ungraded price). - *Graded*: A card graded by a professional service (e.g., PSA). Grading often ranges from 1 to 10; PSA 10 (Gem Mint) is typically most valuable. - *PSA*: Professional Sports Authenticator, a popular card grading company. In this document "PSA 10 price" refers to market price for a PSA grade 10 of that card. - *Portfolio*: A collection of cards a user owns, tracked with purchase values to monitor investment performance (analogous to a stock portfolio). - *Watchlist*: A list of cards a user is monitoring (but doesn't necessarily own) for price changes or potential future purchase. - *Buy/Sell/Hold Signals*: Indicators suggesting whether a card is trending upward (buy opportunity) or downward (maybe time to sell) or stable (hold), based on the platform's algorithms. - *Backtesting*: Analytical tool allowing users to test how a trading strategy or decision would have performed historically using past data. - *API*: Application Programming Interface – here referring to third-party services our platform will use (e.g., eBay API for sales data, Pokémon card database API, etc.). - *MVP*: Minimum Viable Product – the initial version of the product with core features.

Overall Description

Product Perspective

The Pokémon Card Investment Platform is envisioned as a **web application** that eventually evolves into a multi-platform ecosystem (web, mobile, and Discord). The web app will be the primary interface initially, functioning as a centralized hub for all data and user interactions. The design will follow a modern, modular approach: a robust back-end (with database and analytics engines) will power multiple front-ends (web UI, mobile app, Discord bot) through a set of APIs. This separation ensures that as new interfaces are added, they can leverage the same business logic and data.

The platform can be seen in context as a **specialized financial-analytics tool** applied to the collectible card domain. It draws inspiration from stock market analysis software (like Bloomberg Terminal) but tailored to Pokémon cards. In the current ecosystem, users often rely on disparate resources for information – e.g. checking TCGPlayer or eBay for prices, using spreadsheets for tracking, visiting forums for market sentiment [5]. Our platform consolidates these needs into one service: - It will include a **comprehensive card database** (sourced via Pokémon TCG APIs or an internal DB) with high-quality images and detailed metadata for each card. - It integrates **real-time pricing data from multiple marketplaces** (for accuracy and coverage) [6], providing up-to-the-minute market values and trends. - It introduces sophisticated **analytics and visualizations** (charts, signals, forecasting) that currently are not all found in one place for this hobby. - It allows users to **manage their collection as a portfolio**, similar to how investors manage stock portfolios [4], including tracking the value over time and computing gains/losses. - It leverages **machine learning and big data** to provide decision support (a unique selling point – making it feel as authoritative as tools used by quant hedge funds).

The product is data-driven at its core – without accurate and timely data, it cannot succeed. Therefore, a significant aspect of the system will be an underlying data pipeline that continuously aggregates and updates information (prices, sales, trends) from external sources. The platform's quality will be measured by the reliability of this data and the insights drawn from it.

This platform is **distinct from typical price-guide websites** by its depth of analysis and user-centric tools. While existing sites might let you look up a price or track a collection, **our platform aims to be an all-in-one market intelligence platform** for Pokémon cards, providing everything from basic lookup to advanced predictive analytics. It should become the go-to daily tool for serious collectors and investors – much like stock traders rely on terminals or financial websites for up-to-date information. The design must therefore prioritize **performance, accuracy, and a professional UX** to gain user trust and adoption.

Product Functions (Summary of Features)

In summary, the Pokémon Card Investment Platform will provide the following major functions to end-users (detailed requirements in the next section):

- **Card Database Search & Browse:** Users can search a vast database of Pokémon cards (by name, set, etc.) and browse results in an intuitive grid of card "tiles". Each result shows key info like card image, name/set, and quick indicators of its market trend (e.g., arrows for price direction). Filters and sorting will help users narrow results.

- **Detailed Card Analytics View:** For any specific card, the platform offers a detail page with comprehensive information: high-resolution images, card details, current market prices (raw and graded conditions), historical price charts (with adjustable time ranges), volume/sales data, population reports for graded cards, and AI-driven indicators (buy/sell/hold signals). This page also provides links to external marketplaces (with affiliate tracking) to buy/sell the card, and options to add the card to a personal portfolio or watchlist.

- **User Accounts & Profiles:** Users can create accounts to save their data and preferences. Profile management includes setting username, email, password, and possibly linking external accounts (e.g., Discord, in the future). User accounts enable access to personalized features like portfolio tracking and alerts. There will be different user tiers (e.g., Free vs. Premium) with corresponding access rights (premium features for subscribed users).

- **Portfolio Management:** A logged-in user can create a personal card portfolio representing their owned collection/investments. They can add cards (and sealed products) to their portfolio, specifying details such as quantity owned, purchase price, purchase date, and grading information. The platform will track the **current value of each item and the overall portfolio** in real-time, showing metrics like total portfolio value, daily or weekly changes, and profit/loss based on what the user paid [7]. Visualizations (graphs) will display portfolio value growth over time. Users can update holdings (e.g., if they sell a card, they can record a sale or remove it from the active portfolio without losing the financial history). The portfolio function essentially treats cards as assets and helps users monitor the performance of their collection as an investment.

- **Watchlists & Lists:** Users can maintain watchlists of cards they do not own but are interested in. This is similar to a stock watchlist – the user can quickly reference these items and see their current prices

and trends without including them in portfolio totals. Multiple lists may be allowed (for example, a user might create a "Wishlist" of cards they want to buy, or separate lists by theme). This feature keeps users engaged with cards they care about and facilitates easy tracking of potential investment opportunities.

- **Alerts & Notifications:** The platform will include a robust alert system. Users can set up custom alerts on specific cards or sets – for instance, "Notify me if the price of **Pikachu Illustrator (PSA 9)** drops below $X" or "Alert when **Charizard Base Set (Raw)** rises more than 5% in a day". Users can also subscribe to general market alerts, such as daily top movers (highest gainers/losers in value), or news alerts if applicable. Alerts will be delivered via user-preferred channels: email notifications, in-app notifications, and in the future via mobile push or Discord bot messages. This ensures users can react quickly to market changes (near real-time alerts for a trading-like experience).

- **Data Analytics & Backtesting Tools:** For advanced users (likely Premium members), the platform will offer analytical tools akin to those used in finance. One key tool is **backtesting**, which lets users simulate investment strategies or analyze historical performance. For example, a user could test "What if I had bought this card 6 months ago and sold today?" or "How would a monthly investment of $100 into vintage cards have performed vs. modern cards?". They might also backtest the platform's own signals (to evaluate their effectiveness). These tools will use historical data to output results like ROI, volatility, and other metrics, thereby helping users refine their approach. Other analytics might include comparing two cards side by side, correlation analysis between cards or sets, or even building a hypothetical "index" of Pokémon cards to compare against individual card performance.

- **Machine Learning & AI-driven Insights:** The platform will incorporate AI/ML algorithms to process the wealth of data and provide intelligent insights. This includes predictive models for price forecasting (short-term and long-term), and a rating system for cards (e.g., a score or classification as Buy/Sell/Hold). The AI might analyze trends from pricing history, volume of transactions, card grading population changes, and external factors like social media buzz or Google search trends. The output will be integrated into the UI (e.g., the colored arrows or a numeric "confidence score") giving users a quick indication of a card's momentum. Over time, these models will learn from new data to improve their accuracy. (Note: all AI suggestions will include disclaimers as needed since markets can be unpredictable.)

- **External Integrations:** To achieve the above, the platform will integrate with various external systems:

- **Market Data APIs:** e.g., **TCGPlayer API** for price data and possibly product catalog, **eBay API** for recent sale prices and listings (including sold listing scraping for real-world transaction data) [8], and others like PriceCharting or Mercari if available. Combining multiple sources ensures data accuracy and coverage (if one source lacks info on a card or is delayed, another can fill in) [6].
- **Card Database API:** e.g., the **Pokémon TCG API** (or a similar database) for card metadata and images. This provides the foundational info for each card (name, set, number, rarity, images) so we don't have to manually curate all that data.
- **Grading/Population Data:** integration with services like **PSA** (Professional Sports Authenticator) – for example, pulling *population reports* (how many cards have been graded at each grade) and

possibly price guides for graded cards [9]. If direct API is not available, data could be scraped or obtained via third-party datasets.

- **Analytics & Trends APIs:** e.g., **Google Trends** for search popularity of card names or related terms (to gauge interest), **social media APIs** (Twitter/X, Reddit) to track mention volume or sentiment on certain cards or the hobby overall. These data points can feed into the ML models for assessing hype or sentiment, which can precede price movements.
- **Affiliate Programs:** integration with **eBay Partner Network** and **TCGplayer affiliate** links. The platform will monetize by referring users to buy cards – e.g., when viewing a card, they might click "Buy on eBay" which uses our affiliate link so we earn a commission. The system needs to generate or maintain the correct URLs with tracking codes.

- **Payment Gateway:** (for premium subscriptions) integration with a payment provider like **Stripe** to handle subscription payments securely.

- **Reporting and Analytics (Admin & Internal):** The platform will also have an internal side (admin interface or scripts) to monitor system health and data integrity. Although not user-facing, it's worth noting: admins should be able to add/remove data (e.g., update if a new card set is released), moderate user content if any (though primarily there's no user-generated content except maybe usernames/portfolio names), and oversee subscription accounts. Analytics on user engagement (popular cards searched, etc.) might also be collected (within privacy bounds) to continuously improve the product.

User Classes and Characteristics

The platform targets a range of user types in the Pokémon card market, each with specific needs and characteristics:

- **Casual Collector:** This user primarily enjoys collecting Pokémon cards for personal enjoyment or nostalgia. They may not be extremely tech-savvy or finance-savvy. Their main needs are easy lookup of card values, a simple way to catalog what they have, and perhaps basic insights into how their collection's value changes. They might use the platform occasionally (when curious about a card's value or when adding new finds). For them, usability and simplicity are key – the interface should be friendly and not overwhelming. They might not pay for premium features early on, but if the collection grows or the hobby turns into an investment, they could upgrade.

- **Serious Collector / Investor:** This user treats their collection as an investment portfolio. They often keep track of market trends, are active in buying and selling, and are willing to pay for advanced features that give them an edge. They likely have familiarity with financial concepts. Their needs include real-time data, advanced analytics (charts, indicators), and tools to make or validate trading decisions. They will appreciate the platform's depth – e.g., being able to see a card's price history over years, set alerts for quick action, and use backtesting to test hypotheses. They are also likely to be active daily and will stress-test the platform's performance and data accuracy.

- **Card Reseller / Store Owner:** This user's goal is to maximize profit from buying and selling cards. They might manage a large inventory. They need up-to-date pricing (to price their sales competitively or spot underpriced buying opportunities) and possibly bulk portfolio management. Features like price alerts (for sourcing deals) and portfolio tools (to track inventory value) are very

useful. They may also use the platform's data to decide what cards to stock or to identify trends in demand. This class will value integration (maybe exporting data or using the platform's API if they want to incorporate it into their systems). They may also integrate the Discord bot into their community to show prices to customers in real-time.

- **Future Mobile Users:** Overlap with the above categories but using mobile app – likely want quick lookup and ability to scan cards. Mobile users could be at card trading meetups or stores scanning a card's barcode or image to get immediate pricing info from our platform. Thus, this class values speed and convenience. The mobile experience must cater to on-the-go usage with possibly spotty internet (caching important data).

- **Discord Community Members:** These could be any of the above but interacting via Discord. For example, a Discord server of card traders might add our bot to settle "What's this card worth right now?" questions quickly. These users need succinct text/image responses. They value the ability to get info without leaving their chat environment.

- **Administrators (Internal):** Not end-users, but the development/operations team who will maintain the system. They require tools to update card data (like adding a new set when released), manage user issues (reset passwords, handle fraudulent activity if any), and monitor system data flows (ensuring data from APIs is updated). This SRS primarily focuses on end-user functionality, but admin capabilities are recognized as needed (could be simple at first, e.g., direct database access or basic admin pages).

User Tier Considerations: Within the end-user classes, we will have **Free vs. Premium** users: - Free users get access to the essential features: searching cards, viewing basic card details (price chart, etc.), creating a portfolio (possibly with a limited number of items or limited history), and setting a small number of alerts. - Premium (subscribers) users unlock the full power: unlimited portfolio size, advanced analytics (like extended price history, more granular data, backtesting tool), more frequent or advanced alerts, and perhaps exclusive data (e.g., more AI insights or early access to new features). They help monetize the platform beyond affiliates, so the value proposition for premium must be strong (e.g., *"pro" level tools that serious investors need*).

Operating Environment

The Pokémon Card Investment Platform will operate in the following environments:

- **Web Application (Initial Product):** A responsive web app accessible via modern browsers (Chrome, Firefox, Safari, Edge) on desktop and mobile browsers. The web frontend will likely be built using a modern JavaScript framework (e.g., React, Angular, or Vue) to provide a dynamic, app-like experience. It should support all standard screen resolutions, with a design that can adjust to smaller screens (anticipating mobile usage, though a dedicated app will follow). The web app will run on a cloud platform or web server that can scale (for example, deploying on AWS, Azure, or similar, using load balancers and auto-scaling groups as needed).

- **Back-end Server and Database:** The core of the platform (database, server-side application, and analytics engines) will likely be hosted on cloud infrastructure. It might use a microservices architecture (for example, separate services for data collection, user management, analytics

calculations, etc.) or a modular monolith depending on design decisions. We anticipate using a robust database (SQL or NoSQL) to store card data, user data, and historical pricing. The environment should support scheduled jobs or real-time streaming for data updates (for example, a background worker pulling API data every few minutes). The back-end will also expose RESTful or GraphQL APIs to be consumed by the front-end and future mobile/Discord clients.

- **Mobile Application (Future):** Native apps for **iOS** and **Android** (likely built in a cross-platform framework like Flutter or React Native for efficiency, or natively if performance dictates). The mobile apps will interface with the same back-end via API. They'll leverage device hardware (camera for card scanning, local storage for caching, push notifications for alerts). They must run on reasonably up-to-date mobile OS versions (likely iOS 14+ and Android 10+ or as appropriate in the timeframe of development) and cover a wide range of device form factors.

- **Discord Bot (Future):** The Discord bot will run on a server (could be the same back-end or a separate service) and use the Discord API to interact in Discord servers. It will need network access to the platform's APIs and likely will be a lightweight process that can be hosted on cloud (maybe as a serverless function or small container). It must handle potentially many servers/requests concurrently if the bot becomes popular in communities.

- **External Systems:** The platform will interact with external environments including:

- **Third-party APIs** (e.g., eBay, TCGPlayer, Google Trends): These require internet connectivity and proper authentication (API keys). The environment must securely store API keys and handle HTTPS requests. Rate limiting policies of these APIs are a constraint on the environment – we may need caching servers or a data warehouse to minimize direct calls.
- **Affiliate Links:** The environment where affiliate redirects happen (user's browser going to eBay or TCGPlayer) must append required parameters. This doesn't impose heavy requirements on our environment except correctness and perhaps compliance with any terms (like showing an affiliate disclaimer if needed).

- **Email/SMS Notifications:** If sending emails (for alerts, verification, etc.), the back-end environment will integrate with an email service (like SendGrid) or SMS gateway for texts. Those external services must be reachable from the environment.

- **Security & Reliability:** The operating environment will enforce secure connections (TLS for all external traffic). We will maintain separate environments for development, testing, and production to ensure stability. The production environment should have redundancy (database backups, multiple server instances) to avoid downtime.

There are no specific OS/hardware requirements imposed on end users beyond having a device with a web browser or supported OS for mobile apps. On the server side, the system will likely run on Linux-based servers in the cloud. The architecture should be container-friendly (e.g., using Docker/Kubernetes) to facilitate scaling and deployment.

Design and Implementation Constraints

While designing this platform, a number of constraints and considerations must be kept in mind:

- **Real-time Data Constraints:** Achieving "real-time" or intraday updates on card prices is challenging due to reliance on third-party sources. We must work within the rate limits and update frequencies of APIs like eBay and TCGPlayer. For instance, if eBay's sold listings API has a limit of X calls per day, we might not get every single sale instantly. To mitigate this, we may need to maintain our own internal price model that updates incrementally and caches results. The system design must allow frequent data fetches and possibly use websockets or push notifications to update the frontend when new data comes in (for example, updating a price chart live if a new sale is recorded). If true real-time streaming is not possible, we will design for **frequent periodic updates** (e.g., every 5 minutes or every hour key data is refreshed).

- **Data Volume & Storage:** Storing historical price data for thousands of cards (each possibly daily or even transaction-level data) can become large. We should choose appropriate data storage (perhaps a time-series database or optimized SQL schema) for price history. Over years, this data grows; we need a plan for data retention (probably we will *not* throw away historical data, since it's valuable for analytics). Also, images for cards will be stored or cached (we may rely on external CDNs for card images via API to save space, but caching frequently viewed images on our server could improve performance).

- **Machine Learning Model Constraints:** Developing and integrating ML models (for price prediction or recommendations) will require historical data and possibly training infrastructure. We might need a separate environment or service for training (e.g., using Python scientific stack, possibly offline training jobs that then deploy a model to production). This implies a constraint that our architecture be flexible to include such components. Also, any AI-driven feature must be thoroughly tested – false or poor recommendations can hurt credibility. We need to implement it in a way that can be improved iteratively (the SRS will list it as a requirement, but actual model tuning will be ongoing).

- **Regulatory and Legal Constraints:** Using Pokémon intellectual property (card images, names) requires adherence to fair use and terms of use of sources. We must display disclaimers that we are not affiliated with The Pokémon Company (as TCGTrends and others do) [10]. Also, data from third-party APIs like eBay cannot be misused beyond their allowed scope (check API terms for storing data, etc.). If we implement payment subscriptions, we must comply with financial regulations (hence relying on a compliant payment processor). Privacy laws (GDPR, CCPA) will influence how we store and process user data – for example, EU users might need data stored in EU, or we need to allow data deletion on request, etc.

- **Compatibility Constraints:** The web app should function on standard modern browsers; we will avoid using bleeding-edge features that aren't widely supported. For the mobile app, we should consider a minimum OS version that covers the majority of users to not exclude too many. Discord bot must comply with Discord's terms (for example, not spamming, handling rate limits for bot messages, etc.).

- **Integration Constraints:** Each external integration may impose specific constraints:

- eBay API might require OAuth authentication and has specific query formats.
- TCGPlayer's API requires a key and possibly has a request limit per minute.
- Google Trends might not have an official API, requiring use of an unofficial one or scraping – that means possible instability. We might constrain such features to ensure the platform remains reliable even if a non-official data source fails.

- PSA data scraping might have to be throttled to not overwhelm their servers and stay within permissible use.

- **Scalability & Maintainability:** We intend to scale to a large user base (if this becomes the #1 platform). Therefore, design constraints include using scalable technologies (e.g., cloud databases that can handle high read/write, stateless web servers behind load balancers, etc.). We should also modularize features so that future developers (or current team as it grows) can maintain the code – e.g., keep the data collection logic separate from the presentation logic, so changes in an API don't break the web UI.

- **Mobile and Discord Early Design:** Even though the initial implementation is web, we constrain ourselves to design APIs and UI with mobile and bot in mind. For example, any function we build in the web UI should be backed by an API endpoint that a mobile client could also call. This likely means we will develop a REST API for core actions (search, get card details, manage portfolio) used by the web via AJAX – then later reuse these for mobile. We also plan the database with an idea that mobile might require offline caching or partial data sync (so perhaps provide endpoints to fetch all of a user's portfolio in one call, etc., to optimize). For Discord, we may constrain responses to be textual and image summaries (no complex interactive UI in Discord), which might shape what data we return in the API (e.g., perhaps an endpoint that returns a small text summary of a card given a name query, specifically for the bot's use).

- **Time-to-market vs. End-goal Features:** We must acknowledge not everything can be built at once. We will detail the end goal features here, but for implementation, we'll prioritize an MVP subset. This means some detailed requirements might initially be put behind feature toggles or placeholders. A constraint is to design the system such that adding those later doesn't require reworking core parts. For example, if backtesting isn't in MVP, we should still structure the data storage in a way that historical data is there for when we build backtesting. Or if Discord is later, ensure authentication system can handle API tokens for bots later. Essentially, early design decisions should not block future expansion.

Assumptions and Dependencies

Assumptions: - It is assumed that the third-party data sources (APIs for card data and pricing) will continue to be available and sufficiently comprehensive. For instance, we assume the Pokémon card API we choose covers all existing cards and is kept up-to-date with new releases, or we have a process to update it. - We assume that users are interested in a sophisticated tool and are willing to sign up and possibly pay for premium features if the value is demonstrated (market research suggests a subset of collectors treat this as an investment and spend money on tools/resources [11]). - We assume that the volume of data (both in terms of number of users and number of tracked cards) will grow steadily and we plan accordingly, but we also assume we won't overnight have a billion data points or users—i.e., we have time to scale up. - It is assumed that the platform will start with English Pokémon cards; support for other languages or other

trading card games can be added later if desired, but initially we focus on the main market (we design the data model with flexibility for expansion, but initial data will be primarily English set cards). - We assume all users have internet connectivity while using the platform (no offline mode planned except caching in mobile app for efficiency).

Dependencies: - **Pokemon TCG Database:** We depend on a reliable source for card definitions and images (like PokéAPI or Pokémon TCG Developers API). If this service is down or incomplete, our search feature and card info display are impacted. Mitigation: consider caching data in our own database and updating it ourselves when new sets come. - **Market Data Providers:** We heavily depend on eBay and TCGPlayer for pricing data. If eBay changes their API or terms, we may need quick updates. We might depend on scraping eBay's sold listings as a fallback (risky but sometimes done if API is limited). - **Affiliate Programs:** Our revenue partly depends on eBay Partner Network and others. If these programs change commission rates or rules, it can affect revenue. Also, we depend on being accepted into these programs. We'll assume acceptance; if not, that portion of monetization might be rethought (e.g., perhaps integrate our own marketplace or use ads instead). - **Payment Processor:** Dependency on Stripe (or similar) for subscriptions. If there are issues or if in some countries Stripe isn't available, that affects premium in those regions. We might assume initial user base is primarily in regions where these are available (e.g., US, Europe). - **Machine Learning Libraries/Services:** We will depend on certain ML libraries (TensorFlow/PyTorch, etc.) or possibly cloud services (like AWS Forecast or Google Cloud ML) for implementing predictive analytics. Those need to be compatible with our tech stack and licensing. - **Discord API & App Stores:** The future Discord bot depends on Discord's platform (which is stable but we abide by their policies). The mobile apps depend on Apple App Store and Google Play policies for distribution (e.g., any in-app purchases via subscriptions must use their IAP systems etc., which we'll comply with by possibly using a unified backend but platform-specific purchase flows). - **Open-Source Components:** Likely we will use open-source libraries (charting libraries for financial charts, UI frameworks, etc.). We depend on their reliability and community support. We will choose well-supported projects to mitigate risk.

If any of these dependencies fail (e.g., an API is discontinued or becomes pay-prohibitive), we will need to find alternatives (perhaps switch data providers or limit certain features). These assumptions and dependencies are considered in planning so that contingency plans can be made (for example, possibly partnering directly with some data providers or building web-scraping routines as backup, though respecting legal boundaries).

System Features and Requirements

The following sections detail the specific requirements of the system, organized by major feature areas. Each feature is described along with the functional requirements (what the system should do) and relevant non-functional requirements (performance, usability, etc.). The requirements are written in a way to guide design and testing – they use "shall" for mandatory requirements and "should" for desirable characteristics.

1. Card Database Search and Browse

Description: The platform shall provide a robust card search functionality that allows users to find Pokémon cards (and similarly, sealed products) by various criteria. This is the entry point for most users – e.g., a user types "Mew" into a search bar to find all Mew-related cards. Results will be displayed in an attractive **grid of card tiles** for easy browsing. The search/browse interface should be intuitive and fast, catering to hobbyists who might not recall exact set names or those who want to explore cards visually.

Functional Requirements:

- **Search by Name/Keyword:** The system shall allow users to search by a card's name or part of the name. For example, entering "Mew" should return cards like *Mew (multiple sets)*, *Mewtwo*, *Mewtwo & Mew GX*, etc. The search should be case-insensitive and accommodate partial matches and common synonyms (if any).

- **Search by Other Fields:** The system should support searches or filters by other attributes as well:

- Set Name (e.g., a user could filter to only show results from "Base Set" or "Evolving Skies").
- Card Number within set (e.g., "#25" if the user knows the card number).
- Card Type or Rarity (e.g., show only "Secret Rares" or only "Pokémon GX" type).
- If possible, by Pokémon character (perhaps the name search already covers this, but e.g., searching "Charizard" yields all Charizard cards).

- By year or release date range (to find vintage vs modern). These can be advanced search filters available via a search form or filter panel.

- **Tile Grid Display:** Search results shall be displayed as a grid of "card tiles". Each tile (for each card result) shall include at minimum:

- **Card Image:** A thumbnail image of the card (front face). This makes browsing visual and engaging.
- **Card Name and Details:** The name of the card, the set or expansion name, and the card's number in that set. For example, "**Mewtwo** – Base Set #10/102".
- **Current Price (Selectable):** By default, show either the current **raw price** or a chosen grade price. The tile shall have a UI element (e.g., a dropdown or toggle button) that lets the user switch the displayed price between *Raw* and a specific graded value (like PSA 10, PSA 9). For instance, a dropdown might say "Price: Raw ▼" and user can switch to "Price: PSA 10". The tile would then update to show the selected price value (like `$250` for PSA 10). If available, also show the short-term change next to the price (e.g., "$250 (+5.2%)" indicating recent change).
- **Trend Indicator:** Each tile shall display an icon or arrow indicating the platform's current signal/trend for that card: e.g., a **green upward arrow** meaning the card's value is rising or it's a recommended "Buy" (bullish signal), a **red downward arrow** meaning the value is falling or "Sell" signal, and a **gray/rightward arrow** or pause icon indicating a stable "Hold" situation. This provides at-a-glance insight without opening the details. (These signals are computed by the analytics engine, see section on ML and algorithms.)

- **Additional Info:** The tile can include any other brief info that adds value. For example:

    - The 24-hour or 7-day percentage change in price (especially if the arrow indicates trend, a numeric value could be shown on hover or in small text).
    - Perhaps a small icon if the card is on the user's watchlist or already in their portfolio (for logged-in users, to avoid duplication confusion).
    - Rarity or set symbol (some UIs show the set's logo or the rarity symbol on the card tile).
    - A quick action "+" button to add to portfolio/watchlist directly from the tile.

- **Result Sorting and Filtering:** The system should allow users to sort results by certain criteria (e.g., alphabetically, price high-to-low or low-to-high, newest set first, etc.). Filtering options should be

available (e.g., filter to show only graded vs raw, or only a specific grade like only PSA 10 results, or filter by rarity). This helps users narrow down especially if a search term returns many results.

- **Pagination/Scrolling:** If a search yields many results (e.g., searching "Pikachu" might have dozens of cards), the results should be paginated or use infinite scroll to load progressively. The UI should remain responsive and not attempt to load thousands of images at once.

- **No Results Handling:** If no card matches the search query, the system shall display a friendly message (e.g., "No cards found for 'XYZ'. Try adjusting your search terms.") and possibly suggestions (like did you mean... if a name is misspelled, though implementing a fuzzy search or suggestions is a nice-to-have).

- **Default Browse:** The system shall also allow browsing by set or category without a specific search query. For example, a user could click "Base Set" from an index of sets and see all cards in that set listed. Or browse by year or by card type. This encourages exploration. A "Sets" page listing all sets (with their icon and year) is desirable.

- **Performance:** Searching the card database should be optimized for speed. The system should retrieve and display results ideally within **2 seconds** for typical queries on a normal internet connection. This may involve indexing the card data for quick text search. The user experience of typing in the search box could be enhanced with typeahead suggestions (e.g., user starts typing "Chariz..." and a dropdown suggests "Charizard – Base Set" etc.), though this is an enhancement.

- **Data Accuracy:** The card information displayed (names, images, sets) shall come from the authoritative database. Any price displayed on the tile must be the latest available market price for that card in the selected condition. The trend indicator must reflect the latest algorithmic assessment (which might be updated daily or intraday as data changes).

- **Responsive Design:** The search and results page shall be mobile-friendly (even in the web version). On smaller screens, the grid might become a list (tiles stacking vertically) and the information might reflow. The design should ensure that the essential info (image, name, price, arrow) are visible even on a phone screen without excessive zooming or horizontal scrolling.

Rationale & References: This search feature is fundamental to allow users to quickly find the cards they care about. Similar platforms highlight the need for comprehensive and user-friendly search. For example, PokeData provides the ability to search sets and cards easily, building lists from those results [11]. Our approach ensures even a hobbyist can type a Pokémon's name and see *all versions of that Pokémon card* in one place, with visual appeal. The inclusion of price and trend on the result tile is akin to seeing stock tickers in a list – giving immediate context to the search results (as one Reddit user noted, having "everything you'd want" in one interface, from checking prices to building lists, is highly valued [11]).

2. Card Detail Analytics Page

Description: When a user selects a card (by clicking on a card tile from search or anywhere it's listed), the system shall display a **detailed analytics page** for that specific card. This page provides an in-depth view akin to a stock's detail page on a financial site or a mini Bloomberg terminal for the card. It combines card

specifics (identification info) with rich market data and analysis. The goal is to present both raw data (prices, charts) and synthesized insights (signals, comparisons) to the user.

Functional Requirements:

- **Card Overview Section:** At the top, the card's basic information shall be presented clearly:
- A large image of the card (front). If available, a toggle to view the back image or holo pattern might be nice-to-have (some collectors like to see back condition, but front is primary).
- Card name, set name, card number/total (e.g., **Charizard – Base Set #4/102**), rarity (e.g., *Rare Holo*).
- Year of release and any pertinent edition info (1st Edition, Shadowless, etc., if applicable).
- Perhaps the Pokémon type or other flavor details (not crucial, but could be shown for completeness).

- A small icon or note if the card is in the user's portfolio or watchlist (for logged-in users, maybe with an option to add/remove right there).

- **Price Summary:** Prominently display current market price information:

- **Raw Price:** The current estimated market price for an ungraded (raw, Near-Mint) version of the card. E.g., "Raw Market Price: **$85**".
- **Graded Prices:** A small table or list of key grades and their prices. For example:
    - PSA 10 (Gem Mint): $500
    - PSA 9 (Mint): $300
    - PSA 8: $150
    - (We might limit to most relevant grades, or allow the user to click "view all grades" to see a pop-up with more).
- Each of these prices should ideally also show a **daily or weekly change** (e.g., +2.5% or -$10 since last week).
- Include the date/time of last update for these prices so users know how fresh the data is.

- If available, show volume of sales or listings (e.g., "Avg Daily Volume: 10 cards/day" or "Last sold on eBay: $520 on 2025-08-01").

- **Price History Chart:** A core element of this page is an interactive chart showing the price of the card over time. Requirements for this chart:

- It shall default to a reasonable time frame (e.g., 1 year or All-Time if data length is not huge).
- Provide controls for the user to change the time range: e.g., 1W (one week), 1M, 3M, 6M, YTD, 1Y, All.
- The chart should visually plot price on the Y-axis and time on X-axis. If we have both raw and graded data, possibly allow the user to select which series to display (e.g., a toggle for "Raw vs PSA 10 vs PSA 9"). Alternatively, multiple series can be plotted concurrently with a legend (but that might clutter; perhaps one at a time via toggle is cleaner).
- The chart should include markers or volume bars to indicate transaction volume if data is available (e.g., small bars at bottom for number of sales per day/week, which helps interpret price jumps).
- Hovering or tapping on the chart shall show exact data values (price on that date).
- The chart should be high-quality and fast to load. For long term (years of data), it might aggregate by week or month; for short term (days, intraday) show granular data.

- (If feasible, incorporate candlestick style for intraday if multiple price points in a day – though for cards, daily average might suffice.)

- **Analytics Indicators:** Beside or overlaid on the chart, the system may show certain analytic overlays:

- Moving averages (e.g., 7-day or 30-day moving average lines) to smooth trends.
- An indicator of volatility or price range (maybe a band showing high-low range).
- Key events markers: if relevant, mark events like "New set release" or "Logan Paul stream" if they affected price (this is complex and likely a future addition; could tie in with news or custom annotations).

- For simplicity in MVP, maybe skip event markers, but design should allow adding them later.

- **Buy/Sell Signal & Rationale:** The platform's current recommendation for the card should be clearly displayed, for example:

- A section or badge that says **"Current Signal: BUY"** (with a green upward arrow icon), or SELL (red arrow down), or HOLD (gray hold icon).
- There should be a short explanation or data point summarizing why (especially for premium users who might get more detail). For example: "**Buy** – Price uptrend of +15% last quarter and increasing demand signals from online searches [5] ." Or "**Sell** – Price has spiked 50% above 6-month average; consider profit-taking."

- This gives transparency to the user that it's not arbitrary. However, detailed rationale might be a premium feature (basic users might just see the signal).

- **Population & Grading Data:** If the card is gradable, the system shall present known population stats:

- For example: "PSA Population: 1200 (PSA 10: 300, PSA 9: 500, PSA 8: 400, lower: ...)". This helps users gauge rarity of high-grade copies (a factor in value) [12] .
- If available, include a link or reference to the source (like PSA's public report).
- Possibly also show a chart of population over time if we have that data (e.g., how many PSA 10 existed each year – as more cards get graded, value can dilute [13]).

- If multiple grading companies data is available (PSA, Beckett, etc.), we might list all or focus on PSA initially (since it's most common for Pokémon).

- **Recent Sales or Listings:** The detail page shall show recent market activities for the card:

- **Recent Sold Items:** A list (table) of the last few sales (especially high-value or relevant condition). For example:
    - "2025-08-20: PSA 10 sold for $550 on eBay"
    - "2025-08-18: Raw NM sold for $90 on TCGPlayer"
    - "2025-08-15: PSA 9 sold for $300 on eBay"
- This gives real-world validation of the prices and also transparency on how we compute the market price.

- **Current Listings (Buy opportunities):** Optionally, show lowest current listing found (like "Lowest Buy It Now: $600 on eBay for PSA 10" with a link).

- These should be updated frequently via API. However, they may be limited by what data we can fetch (likely eBay's last sold and current listings which we can get via their API).

- **Affiliate Purchase Links:** For user convenience and monetization, the page shall include clear call-to-action buttons to purchase or sell the card:

- E.g., a **"Buy on eBay"** button and a **"Buy on TCGPlayer"** button. If possible, show a snippet like "Starting at $95 on eBay" if we have the data for current listings (to entice clicks) [8] .
- These buttons will open the respective site with our affiliate tracking. For selling, maybe a "Sell on eBay" link could lead the user to list an item (this is less straightforward with affiliate, but maybe an informational link).
- We might partner with other marketplaces too (if any official ones for Pokémon cards).

- Ensure these links are obvious but not overly intrusive (we want user trust; maybe mark them subtly as affiliate if needed by policy).

- **Add to Portfolio/Watchlist:** The detail page shall have buttons for the logged-in user to **"Add to Portfolio"** or **"Add to Watchlist"**. Clicking these should prompt for relevant details:

- If adding to portfolio: ask user to input quantity, condition/grade, purchase price, purchase date (default to today if new, or allow blank if they just want to track current value).
- If adding to watchlist: possibly prompt which watchlist or just add to a default one.

- If the user is not logged in, clicking these could prompt login/sign-up (with message like "Create an account to save this card to your portfolio").

- **Backtesting/Analysis Tools Access:** If the user has premium access, on the card page there might be an option like **"Analyze/Backtest"** – for example, "Backtest this card's performance…" which could open a tool allowing them to simulate trades (or it might be an entry to a more advanced analysis where they can compare this card's performance to another asset, etc.). For MVP this might not be implemented, but leave a placeholder or design the page with space for advanced tools.

- **Comparison Feature (Nice-to-have):** Possibly allow the user to compare this card to another: e.g., a button "Compare" that lets user add another card to see side by side stats or an overlay on the chart (like comparing two stocks). This might be beyond MVP, but design could foresee it.

- **UI/UX considerations:** The card detail page will contain a lot of information, so it should be well-organized with sections or tabs if necessary:

- Example layout: Left side showing the card image and basic info; right side top showing current prices and signal; below that the chart spanning full width; below chart, a 2-column section with on left: recent sales list, on right: population stats and maybe additional insights.
- Alternatively, use tabs like "Overview | Price History | Analytics | Market Listings" to separate concerns. However, a single-page scroll may be fine if sections are clearly delineated with headings.

- Ensure that on mobile, these sections stack properly (e.g., image and name at top, then a collapsible chart or a simpler chart for mobile, etc.).
- Use visual cues (colors for up/down, icons, etc.) consistently so users quickly grasp the state (like green text for positive changes, red for negative, etc., similar to financial apps).
- **Accuracy and Refresh:** The data on this page should update in near real-time. If the user keeps the page open, we should ideally auto-refresh key elements (like price or signal) whenever our backend gets new data (maybe via WebSocket or periodic AJAX refresh). If not possible initially, ensure that revisiting or manually refreshing will fetch the latest. The last updated timestamp for prices should be visible.
- **Error handling:** If data fails to load (e.g., third-party API issues), the page should handle gracefully, e.g., showing "–" for missing data or a message "Price data currently unavailable. Please retry." in that section, without crashing the entire page.

Rationale & References: This detailed view is where power users will spend a lot of time, akin to how a trader would study a stock's page. It needs to present a **complete picture** of the card's financial standing. Existing platforms hint at these needs: TCGTrends emphasizes seeing "value changes over time" with charts and multi-platform data [6]. PokeData shows portfolio performance graphs [14] and price changes, but our platform goes further with predictive signals and external indicators. By including population reports and recent sales, we echo what collectors often manually research (PSA populations, eBay sold listings) – bringing it in-app saves them time [9] [8]. The buy/sell signals turn raw data into actionable advice, leveraging our machine learning – a unique selling point if done well. Overall, the card detail page should make the user feel they have *Bloomberg-like power at their fingertips specifically tailored to their card*, which will set this platform apart.

3. User Accounts and Profiles

Description: The platform will support user account creation and login to enable personalized features (portfolio, watchlists, alerts, etc.). Account management must be secure and user-friendly. This section details how users register, authenticate, and manage their profile, as well as how the system differentiates access for different user roles or subscription tiers.

Functional Requirements:

- **User Registration:** The system shall allow a new user to create an account. At minimum, an email address and password will be required. The username could be either the email or a separate field if we want to display a public name (though there's no public social feature now, username might just be for personalization). The registration process should include:
- Validation of email format and password strength (password must meet minimum criteria like length, possibly a mix of characters for security).
- Optionally, integration with OAuth providers (e.g., "Sign up with Google" or Facebook) to lower friction – not mandatory for MVP but nice if possible.
- Upon submission, send a verification email with a link or code to confirm the email (to ensure we have a valid contact and reduce spam accounts).

- **User Login:** The system shall provide a secure login mechanism. Users enter email/username and password to authenticate. Support standard features:

- "Remember me" option to keep the user logged in (using secure cookies or token).
- Forgot Password flow: allow user to request a password reset email if they forget their password. The system emails a secure one-time link to reset.
- The login should start a session (JWT or server session) that persists across pages. Use HTTPS to protect credentials.

- Show helpful errors on failure (e.g., "Invalid credentials").

- **Account Security:** Passwords must be stored securely (hashed with a strong algorithm, salted). The system shall enforce best practices to prevent common vulnerabilities (SQL injection, XSS, etc. – likely handled by frameworks but must be tested). Potentially allow or plan for optional 2-Factor Authentication (2FA) for users who want extra security (e.g., via authenticator app or SMS). 2FA could be a later feature but should be kept in mind.

- **Profile Management:** Once logged in, the user should access a "Profile" or "Account Settings" page where they can:

- View and edit personal information: name, display name, email (with verification if changed), etc.
- Change password.
- Manage notification preferences (e.g., opt in/out of certain alert emails, choose if they want a weekly summary email, etc.).
- See their subscription status (free or premium) and if premium, maybe details like next billing date.
- Payment info if premium – though likely handled by the payment provider's portal, but maybe a link to "Manage Subscription".

- Possibly link external accounts: e.g., link Discord account (so the bot knows who is who) by generating a code to use with the bot, etc., or link Google for login.

- **User Roles & Permissions:** The system shall distinguish at least:

- **Standard User (Free):** Access to base features. If a free user tries to access a premium feature, the system should show a prompt or upgrade page.
- **Premium User:** Access to all features. The system should ensure premium-only features (like advanced analytics, unlimited alerts, backtesting) are gated.

- **Admin:** (internal use) accounts with administrative privileges. Admins can do things like manage the card database, view user list, perhaps moderate if needed. This could be implemented via a separate admin interface or simple checks in code. Not customer-facing, but included for completeness.

- **Subscription Management:** For premium subscriptions:

- The system shall integrate with a payment system to allow upgrading to premium. This might be via a "Upgrade to Premium" page which goes to a payment form (Stripe checkout, etc.).
- On successful payment, the user's account status is upgraded in our database.

- The system should handle renewal (likely the payment provider will send webhooks to our system on successful charges or if a subscription is canceled/expired).
- Provide a way for users to cancel their subscription or update payment, usually done via the payment provider's portal link.

- It's important that features downgrade gracefully if a subscription lapses (e.g., user keeps their data, but can't add new alerts beyond free limit, etc.).

- **Limits for Free vs Premium:** The system should enforce limits. Examples:

- Free user can create up to, say, 1 portfolio and 50 cards in it; premium can have unlimited or a very high cap.
- Free user can set 5 alerts; premium unlimited or 50, etc.
- Backtesting tool available only to premium (or allow a trial with limited functionality for free).
- Data granularity: maybe free sees price history at weekly granularity and premium can drill to daily or intraday.

- These specifics will be decided by business, but the system should be designed to easily handle such conditional access.

- **Privacy and Data Ownership:** The system shall allow users to delete their account (and associated personal data) if they wish (compliance with GDPR etc.). This could be through contacting support or an automated "Delete Account" button that wipes personal info (and perhaps anonymizes any aggregated data). The SRS notes this requirement even if MVP might not implement immediately, it should be possible.

- **Activity Logging:** The system should log important account activities for security/audit: e.g., login times, password changes, subscription changes. This is mostly internal but necessary if issues arise (not exposed to user except maybe "last login from X location" if we add that).

Non-functional/Usability: - The registration and login process should be quick and simple to encourage onboarding. Possibly include a brief tutorial or welcome modal on first login to guide new users (especially if the platform is complex). - All account pages should be mobile-friendly as well. - Security is paramount: ensure use of TLS, no sensitive info in URL or logs, protect against brute force (could add captcha after many failed logins). - The system should handle a scenario of many simultaneous sign-ups (e.g., after a marketing push) without crashing – so the architecture should scale, and any email sending for confirmation should use a reliable service to handle bursts.

Rationale: User accounts are the gateway to retaining users and offering advanced services. Since our platform is meant to be personalized and data-rich, having secure and well-managed accounts is essential. By implementing tiered accounts, we create a revenue model (premium subscriptions) beyond just affiliate links, which aligns with the instruction to include monetization methods. The ability for hobbyists to track their own collection's performance is a big draw, and that only works if they have accounts to save data. Security and privacy considerations build trust – important if we want to be seen as the Bloomberg for cards (financial platforms are expected to be secure and professional). Also, given the target audience might involve valuable collections, users will want assurance their data (and perhaps by extension knowledge of what they own) is safe and private.

4. Portfolio Management Dashboard

Description: The Portfolio feature enables users to track the value of their own Pokémon card collection or investment holdings. Similar to an investment portfolio tracker, this tool will let users input what cards (or sealed products) they own, and then monitor how the total value changes over time, as well as individual item performance. It's a core feature for collectors/investors, turning static collections into dynamic, informative assets.

Functional Requirements:

- **Creating a Portfolio:** Upon account creation, the system shall automatically provide each user a default portfolio (e.g., named "My Portfolio"). Users (especially premium) may be allowed to create multiple portfolios or sub-portfolios (e.g., one for "Vintage Collection" and one for "Modern Investments"), but MVP can start with one portfolio per user until multi-portfolio is needed. Each portfolio has its own list of items and aggregate stats.

- **Adding Items to Portfolio:**

- Users shall be able to add a card or sealed product to their portfolio through an "Add" function. This can be initiated from the card's detail page (as mentioned) or from within the portfolio management UI (e.g., an "Add Card" button that opens a search dialog).
- When adding, the user must specify at minimum: the card (by searching and selecting), the quantity they own, and optionally purchase details.
- If the card is graded, they should specify the grade (PSA 10, etc.) because that affects value. If raw, just mark as raw.
- Optional/advanced fields: purchase price per item (in their currency, default USD), total paid (if multiple quantity), purchase date, and any notes (like where they obtained it, lot number, etc., if they want).
- The system should allow adding items even without purchase price (not all know what they paid long ago). If no purchase price given, profit calculations will skip that item or treat cost as 0 for now.

- For sealed products, similar input (product name, perhaps edition or condition if relevant, quantity, price, date).

- **Portfolio List View:** Once added, the portfolio will display a list/table of all items the user has entered. For each item, show columns such as:

- Card/Product name (with possibly a thumbnail image).
- Details: for cards, include set and number, for sealed include product type.
- Quantity owned.
- **Current market value per item** (auto-fetched from our data, depending on raw/grade).
- **Total current value** (value per item * quantity).
- **Price paid (cost basis)** per item (if provided).
- **Unrealized Gain/Loss:** difference between current value and cost. This could be shown in both absolute and percentage. E.g., "+ $200 (+50%)" if the card's value rose by that much since purchase [7] .

- A field for "Last updated" or a refresh icon in case they want to ensure it's the latest price (but likely updates automatically).
- Possibly a flag if item is marked for sale or trade (future use).

- Allow sorting this list by any column (name, value, gain, etc.).

- **Portfolio Summary Metrics:** At the top of the portfolio page or section, the system shall show aggregate stats:

- Total number of items (or total cards) in portfolio.
- **Total current market value of portfolio** (sum of all current values).
- **Total cost basis** (sum of all purchase prices user entered).
- **Total Profit/Loss** (current value minus total cost) and percentage gain overall.
- Perhaps a breakdown like "% change in value in last 24h/7d" for the portfolio as a whole.

- If multiple portfolios, maybe an ability to switch which one is being viewed or see a combined view.

- **Portfolio Value Over Time Chart:** The system shall provide a chart for the portfolio's total value over time [14].

- This graph plots the sum value of all holdings on the Y-axis, over time on X-axis. Time could be daily or monthly points depending on data frequency.
- This requires capturing historical snapshots of the portfolio value. Implementation: the system can calculate the portfolio's value whenever underlying prices update or at least once daily and store that. Alternatively, calculate on the fly using historical price data for each item (which is more complex).
- The chart should allow the user to see how their collection's value has grown or declined. E.g., over the past year, maybe their $1000 collection grew to $1500.
- If the user adds or removes items, we should account for that – possibly treating an addition as injecting more "capital". We might visually mark when large additions happened (to differentiate growth by investment vs market gains).
- This is similar to investment portfolio charts that sometimes show "net deposits vs gains".

- For MVP, a simpler approach is fine: just show raw total value at each time point, even if jumps include new purchases.

- **Item Detail in Portfolio:** Clicking an item in the portfolio list could navigate to the card's detail page or show a mini detail popup. It might also show what the user paid and their gain, etc. Possibly for premium, allow a quick view.

- **Editing / Removing Items:** The user shall be able to edit or remove portfolio entries:

- Edit in case they made a mistake or want to add purchase price later, or change quantity (maybe they acquired more of that card).

- Removing an item: If the user has sold a card or no longer owns it, they might remove it.
    - **Important:** The system should ask if they are selling it (to record a sale) or just removing. If the user sells the card, we ideally want to record the sale price and date (if they want) and then mark that item as sold. Possibly we keep a transaction history (for advanced users to see realized gains).
    - If user chooses to just delete without sale info, we might assume they disposed of it at current value or unknown, but then we lose that from performance calcs. A better approach: mark as sold and move to an archive where the user can see past items and what profit made.
    - PokeData discussion suggests users want to track realized profit and not count sold cards as still in collection [15] [7]. So our system should support recording a sale event.
    - MVP could simply allow deletion and subtracting that item's cost/ value from portfolio, but ideally we implement a sold state.

- **Multiple Portfolios / Lists:** If implemented, user can create additional portfolios (maybe a "Create New Portfolio" button). Each portfolio can be named. This might be a premium feature (free users have 1, premium can have many). The UI should allow switching between them easily. This is useful if a user wants to separate, say, their personal collection vs inventory for sale, etc.

- **Watchlist vs Portfolio Distinction:** The portfolio is owned items. Watchlists (next section) are separate and should not count in portfolio value. Ensure that the UI makes it clear what's in your portfolio vs just a watch. Perhaps different pages or a tab switch between "Portfolio" and "Watchlist".

- **Data Refresh and Accuracy:** The system shall automatically update the values in the portfolio based on the latest price data available. This means if market prices changed overnight, the next time the user opens portfolio (or in real-time if they have it open), those current values and gains should recalc. Possibly highlight which items changed significantly (like + or - icons for the day).

- We can also allow user to manually trigger refresh (but likely not needed if automated well).

- If an item has no recent price data (maybe an extremely rare card with no recent sales), handle gracefully (could carry last known value or mark as "no recent data").

- **Export/Import:** A nice-to-have (especially for advanced users) is the ability to export their portfolio data as CSV (for backup or analysis). And possibly import a list of items (if someone has a spreadsheet of their collection, they might upload to add all at once). MVP might skip import due to complexity, but export is easier to do.

- **Privacy:** By default, a user's portfolio is private. In the future, we might allow sharing a portfolio via a public link (read-only) if a user wants to show off their collection or prove value (like some do in communities), but that would be opt-in. The system should ensure no unauthorized access to someone's portfolio data.

Non-functional/Usability: - The portfolio UI should be **clean and informative**. Possibly use visual cues like green/red coloring for gains or losses. - If the list is long, allow searching/filtering within portfolio (e.g., filter to a certain set or name in their portfolio). - The dashboard could show interesting things like "Asset

Allocation" – e.g., pie chart by set or by type, if data available (not core, but something to consider to make it feel like a real investment dashboard). - Performance: Even if a user has hundreds of items, the page should load quickly. Use efficient queries and possibly paginate the list or virtual scroll if extremely long. - On mobile, the portfolio page might collapse some columns (like maybe just show item name and current value, tapping expands details) to fit smaller screen.

Rationale & References: Managing one's collection as a portfolio is a central promise of this platform (and mentioned in the prompt akin to a stock portfolio) [4] . Tools like PokeData emphasize tracking collection value changes over time [14] and users find that incredibly useful. For example, a Reddit user praised that PokeData "tracks the value of your collection and the trend of each item" in a clean experience [11] . Our portfolio feature needs to surpass others by not only showing value but also integrating with our analytics (signals, alerts, etc.). By treating each card entry as an "investment line item" with cost basis and P/L, we help users quantify their hobby – turning what was once a static binder of cards into a dynamic financial asset in their eyes. This increases engagement (users will check in to see how their "Pokémon 401k" is doing) and provides a reason to keep using the platform long-term. It also sets the stage for professional investors to justify their actions (having a clear log of performance). Furthermore, implementing portfolio properly is crucial for premium upsell – advanced features like multiple portfolios, or analytics on the portfolio (like risk, diversification metrics) can be premium differentiators down the road.

5. Watchlists and Alerts

Description: The platform shall provide features for users to keep track of cards they are interested in (watchlists) and to be notified of important changes (alerts). These tools ensure that even when users are not actively using the platform, they remain engaged and informed about market movements relevant to their interests. They also mimic functionality from stock trading platforms (watchlists for stocks, price alerts) which target users will find familiar.

Watchlists (Follow Lists):

- **Creating and Managing Watchlists:** Users can maintain a list of cards (or products) they want to monitor closely, without adding them to their owned portfolio. The system may have a default watchlist (e.g., "Watchlist") and possibly allow multiple custom lists (like "Favorite Cards", "To Buy List").
- If multiple lists: user can create, rename, or delete watchlists as needed. This could be a premium feature (e.g., free has 1 watchlist, premium can categorize into many lists).
- Each watchlist functions similarly to a portfolio list but without the notion of ownership quantity or cost.
- **Adding Items to Watchlist:** From any card view (tile or detail), users should have an "Add to Watchlist" action. If multiple watchlists exist, user chooses which list to add to. The card is then listed in that watchlist. No need for purchase price or quantity (though user could optionally note a "target price" if we allow that for their own reference).
- **Watchlist Display:** The watchlist is essentially a simpler list showing:
- Card name, set, maybe image icon.

- Current price (in chosen condition or default raw/PSA10).
- Daily/weekly change or trend arrow (so the user can see what's moving).
- Perhaps a column for "target price" if the user set one (and indicator if current has reached it).

- The user can remove items from watchlist when no longer interested.

- **Integration with Alerts:** Watchlist items are prime candidates for alerts. The system could suggest, "You are watching Charizard; do you want to set an alert for when its price moves by 5%?" Possibly auto-add a basic alert.

- At minimum, allow user to manually set alerts (see below) on any watchlist item easily.

- **Use Case:** A collector might use watchlist for cards they want to buy if price drops, or an investor might watch cards they don't own but consider investing in if trends change. Resellers might watch certain cards to track when to restock or sell.

Alerts and Notifications:

- **Custom Price Alerts:** The system shall allow users to create custom alerts based on price conditions:
- A user can select a specific card (and grade/condition) and set a rule like:
    - "Alert me when price goes above $X" (good if they want to sell when it's high),
    - "Alert me when price falls below $Y" (for buying opportunity),
    - "Alert me when price changes by more than Z% in a day/week".

- These alerts should be flexible. For MVP, we can implement threshold and percentage change as options.

- **Market Movement Alerts:** Users may subscribe to general alerts such as:

- Daily or weekly digest of **Top Movers** – e.g., list the top 5 price gainers and losers across the market or within a category.
- **New Set Releases or News** – if a new set is added to our database or a notable event (maybe out of scope unless we incorporate news).
- Alerts if a card in their portfolio or watchlist hits a **record high or low** (e.g., "This card reached its highest value in 1 year").

- Social sentiment alert: if we integrate social signals and detect a sudden spike in mentions for a card they watch, alert them (advanced, possibly later).

- **Portfolio Alerts:** Users might want alerts related to their collection:

- "My portfolio value changed by more than X% in a day" or "dropped below/rose above a certain total".

- "One of your cards just had a sale at $Y, significantly impacting its value."

- **Setting Alerts UI:** There should be an interface to manage alerts:

- Possibly from the card detail page (like a bell icon "Set Alert" that opens options).
- And a central "Alerts" or "Notifications Settings" page where all alerts are listed, and user can add or remove them.
- When creating an alert, user specifies the condition as above and how they want to be notified (if multiple channels are available).

- For example, choose email or mobile push (when app is out) or Discord DM (if account linked and we have bot) – for MVP likely email is primary.

- **Notification Delivery:**

- **Email:** The system shall send an email to the user's registered email when an alert triggers. The email should clearly state what happened (e.g., "Charizard Base Set (PSA 9) dropped below $500, current price $480 [8] .") and possibly include a direct link to the card page.
- **In-App Notification:** The web app should have a notification center or at least an indicator (bell icon) for alerts. When triggered, the user logged in could see notifications in a dropdown or page.
- **Discord (future):** If user linked Discord and opted in, the system can send a message via our bot to them or a channel. E.g., the bot DM's the user or posts in a designated server channel if configured.

- **Mobile Push (future):** Mobile apps will use push notifications for instant alerts.

- **Alert Reliability:** The system must constantly monitor conditions for alerts. This likely means after each data update (e.g., price refresh), check all alert conditions and mark those that meet criteria.

- We'll implement efficient checking (maybe indexing alerts by card so when card price updates we quickly find relevant alerts).
- The system should avoid duplicate notifications (e.g., if price oscillates around threshold, ideally not spam multiple times – perhaps an alert auto-disables after firing unless user set it as recurring).

- The user could have an option to auto-expire the alert after it fires once (default) or keep it persistent (like "every time it drops below $X").

- **Frequency and Timeliness:** Alerts for price moves should be as real-time as possible – e.g., if a price update at 3:00pm triggers an alert, the notification should go out within a minute or two of that update. This is crucial for the platform's credibility as a timely tracker (intraday feel).

- For percentage change in a day, the check might be daily summary, but threshold crossing can be immediate.

- **Managing Alerts:** Users can view all their active alerts and delete or modify them. Also perhaps pause them (if they temporarily don't want notifications). Especially if someone sets many alerts, a management UI is needed.

- **Limitations for Free vs Premium:**

- Free users might have a cap (e.g., up to 5 active alerts).
- Premium users can have many more (50, unlimited, etc.) and perhaps access to more complex alert types (like the portfolio value ones or social sentiment ones might be premium).

• Also, premium might get faster alert delivery if we ever had tiered processing (but likely equal).

Non-functional Considerations: - The alert emails or messages must be clear and concise. Use a template that includes our branding and perhaps a disclaimer ("prices subject to change" etc. so we're not blamed for decisions). - The system's scheduler or trigger mechanism for alerts must be robust – e.g., ensure that if the system goes down and comes back, missed conditions might still be caught or at least not completely lost. - Privacy: ensure alert messages don't inadvertently leak sensitive info. For instance, a portfolio alert email should be sent only to that user and worded carefully (we wouldn't want to broadcast a user's portfolio specifics to others). - UX: Setting an alert should be straightforward – maybe present common options (above/below current price by X%) to make it easy, with a custom field as needed.

Rationale: Alerts create an engaging user experience by proactively bringing users back to the platform when something notable happens, which is vital for retention. They also align with the "trading" paradigm – investors rely on alerts to not miss opportunities. Since the platform aims to provide intraday/trading-like value, alerts are indispensable. From a monetization perspective, advanced alerts can justify premium subscription (e.g., an investor might pay to get numerous and instant alerts across many cards). The inclusion of watchlists and alerts means users don't have to constantly stare at the charts; they can trust the platform to inform them, which is a convenience factor that can set us apart from just passively checking prices on sites like TCGPlayer. Additionally, watchlists keep even non-owners engaged – a user might not own a 1st Edition Charizard, but they'll watch it out of interest. By catering to that, we keep them on our platform for all market intel, not just their collection.

6. Data Integration and Accuracy (Market Data Engine)

Description: (*This section details the behind-the-scenes requirements for data collection and integration, which ultimately support the user-facing features.*) The platform needs a robust data engine to fetch, aggregate, and maintain the vast array of data (card details, prices, sales, etc.) that powers all features. The quality of the product hinges on the quality of this data – it must be accurate, up-to-date, and comprehensive. Below are requirements for integrating external data sources and handling data within the system.

Functional Requirements:

- **Card Database Integration:**
- The system shall integrate with a Pokémon card database API to populate and update the card information library. For example, using the Pokémon TCG Developers API (if available) to get all card records (name, set, number, images, rarity, etc.).
- The system should initially import all existing cards (this could be >10,000 cards considering all sets) into an internal database for fast access.
- It should periodically check for new cards (e.g., when a new set is released, or we get update from the API) so our database stays current. This can be done via scheduled jobs (e.g., check monthly or when The Pokémon Company announces new sets).
- Each card record in our DB will have a unique ID, and fields for relevant metadata and a link to image (we might cache images on our server or use URLs from the API's CDN).
- If the external API has limitations, we might supplement it by manual data entry or community contributions for missing info, as a fallback.

- **Price Data Collection (Real-time Market Prices):**

- The system shall collect current market price data for cards from multiple sources to ensure accuracy:
    - **TCGPlayer:** This is a key source for card prices (market price, listed median, etc.) [16] . If TCGPlayer API is accessible, fetch prices for cards (they often provide a market price which is a rolling average of recent sales on their platform).
    - **eBay Sold Listings:** The system shall fetch recent sold listing prices from eBay for given card queries to gauge real-world transaction values [8] . For example, search eBay for "Charizard Base Set PSA 10" in sold items and get the last few sale prices. This can augment or validate other data.
    - **PriceCharting:** If accessible via API, this site aggregates prices from eBay and other sources. We can use it as a supplementary data source for some cards (especially vintage graded ones).
    - **Other Marketplaces:** If available, consider integration with other popular marketplaces like TrollAndToad, Mercari, StockX (if they do cards), or international markets (CardMarket in EU). These might be phase 2, but design to allow adding sources.
- The data engine should **aggregate these sources** to produce a single "market price" for each card+condition:
    - A strategy might be to prefer one source as primary (like TCGPlayer for raw card, PSA for graded?), or to calculate a weighted average or median of sources.
    - It should also flag discrepancies (e.g., if one source says $100 and another says $200 for same card, that's unusual – maybe due to condition differences – possibly highlight as volatility).
- **Refresh frequency:** The system shall update price data frequently. Ideally, for actively traded cards, updates could be as often as several times per day (intraday). For less active, daily might suffice. We can implement different frequencies based on card popularity (e.g., top 100 cards update every hour, others update daily to conserve API usage).
- The system shall support on-demand updates: e.g., if a user is viewing a card detail, we might fetch the latest live data from eBay at that moment (if not recently updated) to show the newest sale.

- We must adhere to API rate limits and use caching. Possibly maintain a queue of cards to update and cycle through.

- **Historical Data & Storage:**

- The system shall maintain a historical price database for each card (and per condition). This could be daily average prices, or even transaction-level data if available (e.g., store each eBay sale record with date and price).
- This data is crucial for charts and backtesting. We will accumulate it over time. We might also ingest any historical datasets available (some websites publish historical prices for certain cards).
- Each day (or defined interval), the system should record the price for each tracked category of each card. For example, end-of-day price for Raw, PSA10, PSA9 etc. If multiple sources, decide on one value (like last known market price).
- Store this in a way that can be queried quickly for chart generation (maybe a time-series DB or a table indexed by cardID and date).

- If storage is an issue, we can compress older data (but likely fine given number of cards is not astronomical).

- **Graded Population Data:**

- The system should fetch population reports from PSA or other grading companies for graded cards. PSA has an online pop report; we might need to scrape it unless they have an API.
- At least, occasionally update how many PSA10, PSA9 exist for key cards. These numbers change slowly (when new cards get graded), so updating monthly or so is fine.
- We will store these stats and could even show trends (pop increasing over time can affect price as noted in market analysis [13]).

- If possible, also fetch PSA's price guide (PSA website lists average prices for certain cards by grade [9]). This can validate our price data for graded cards.

- **Social/Trend Data Integration:**

- **Google Trends:** The system should be able to query Google Trends for certain keywords (e.g., "Charizard card" or "Pokémon Base Set") to get a popularity index. This could be done weekly. We can use an API or a third-party library to fetch trend data, which might give us a number (0-100 scale) of relative search interest.
- **Social Media:** If feasible, integrate Twitter API to track mentions of certain card names or related hashtags (e.g., #PokemonTCG). Or Reddit API to monitor relevant subreddits (like r/PokeInvesting, r/PokemonTCG) for keywords (card names or set names) and count mentions or sentiment (via simple NLP).
- This data should be processed into some metric (like sentiment score or mention count) that our ML algorithms can consume. It likely doesn't need to be shown raw to users except maybe in a premium analysis section.

- Frequency: daily scraping of social trends might suffice, or even event-driven when something spikes.

- **Data Accuracy and Clean-up:**

- The system must handle noisy data. For example, eBay listings might include lots or proxies (like 3 cards in one listing, or damaged cards). We should attempt to filter out outliers (like if one sale is way off trend because maybe it was a fake listing or a lot of 10 cards).
- Implement rules or manual oversight for anomalies (maybe flag when a price jumps too high and require verification).
- For naming mismatches (e.g., eBay listing titles might not match exactly our card name), our search queries need to be smart (perhaps include set and number when searching eBay to narrow results).

- The data engine should log when it updated what and any issues, for developers to review.

- **Affiliate Link Integration:**

- For each card, the system shall be able to generate a URL to eBay's search or TCGPlayer's page with affiliate tracking code embedded. Possibly pre-generate these links and store them, or generate on the fly.
- E.g., an eBay link might look like `https://ebay.com/sch/i.html?_nkw=Pikachu+Base+Set+Pokemon+Card&campid=AFFILIATEID...` (with proper keywords and affiliate ID).
- Ensure these links are up-to-date if affiliate programs change requirements.

- Possibly track clicks (just for our own analytics, not required for functionality).

- **APIs for Future Use (internal):**

- The data integration will essentially form internal APIs that our front-end uses. e.g., an endpoint like `/api/price/{cardId}` which returns current price and history. We should design these in tandem with how we store data.
- These will also be used by mobile app and Discord bot later, so they should be stable and well-documented.

Non-functional Requirements:

- **Timeliness:** The system should achieve near real-time updates for popular cards (e.g., within minutes of a major sale, our price reflects it). Less active items can tolerate slower updates. Define a schedule such that critical data is always fresh.
- **Scalability:** As user base and data volume grow, the data engine must scale. Using queue systems for data fetch tasks, and scalable storage (cloud DB) is important.
- **Fault Tolerance:** If a data source is temporarily unreachable (API down), the system should not crash; it should skip or retry later, and possibly use last known data in the meantime.
- **Data Security:** API keys for external services must be stored securely (not exposed in client code). Also any data we store from them might have usage restrictions (we should comply with e.g. not exposing raw eBay sold data beyond what's allowed).
- **Maintainability:** The integration code will need updates as APIs change. It should be written modularly (each integration separate) so that one failing doesn't break others. Also, easy to add new sources if needed (like if a new marketplace becomes key).

Rationale: This data engine is the backbone of the platform's value proposition – without it, everything else (search, portfolio values, alerts) fails or becomes "worthless" as the prompt says. The emphasis on multiple sources and real-time is to ensure **accuracy and up-to-dateness**, which are critical. For example, TCGTrends boasts gathering data across eBay and TCGPlayer to help make quick decisions [6]; we must at least match that. Also, by including social and trend data, we incorporate more **forward-looking indicators**, potentially giving our platform an edge in providing early warnings (in the Bloomberg spirit). Integrating affiliate links ties directly into monetization – and it's explicitly in the requirements to include that, as it's a revenue stream. All these data-related features operate largely behind the scenes, but they manifest in the user experience as reliable info and smart insights. In short, if the data side is executed well, users will trust and depend on the platform for their collecting/investing decisions, fulfilling our mission to be the #1 data-driven tool in this space.

7. Machine Learning & Analytics Engine

Description: A standout feature of the platform is its use of **machine learning (ML) and advanced algorithms** to analyze the data and provide actionable insights (like the buy/sell signals, price predictions, etc.). This section details the requirements for those intelligent features, ensuring they are integrated in the product in a meaningful way.

Functional Requirements:

- **Trend Analysis & Buy/Sell/Hold Signals:**

- The system shall analyze price movements and other factors to determine a simple recommendation for each card (Buy, Sell, or Hold).
- This could be based on a combination of short-term and long-term price trends (e.g., momentum indicators used in stock analysis), volume changes, and possibly external signals (hype, news).
- Implementation: We might create an algorithmic scoring system. For example:
    - Calculate % change over last week and last month.
    - If > X% increase and hitting new highs, that might signal momentum (Buy, unless it looks like a bubble?).
    - If price is dropping consistently or there's oversupply incoming (maybe via pop report increases), signal Sell.
    - If fluctuating within a range, signal Hold.
- The ML part could refine these thresholds or incorporate more variables.
- The output must be one of a limited set (Buy/Sell/Hold or maybe Strong Buy, etc., but keep it simple).

- This signal should be updated whenever new data comes in (so it's dynamic). On the UI, it's shown as arrows as described earlier, but here we define how it's decided.

- **Price Prediction (Forecasting):**

- The system should attempt to forecast future price movements for at least popular cards or generally. This could be a premium feature.
- For example, a model that predicts "Expected price in 1 month" or a probability distribution of where the price will be.
- Could use time-series forecasting models (ARIMA, Prophet, or ML regressors taking multi-factors).
- The requirement is that the platform provides some forward-looking insight, even if with uncertainty.
- Maybe show a dotted line on price chart indicating forecast, or a statement like "Projected 3-month trend: +10% (moderate confidence)".

- If confidence is low, we should be cautious in presentation (maybe only show for cards with enough data).

- **Comparative Analytics & Indices:**

- The system could create aggregate indices (like how S&P 500 indexes stocks). For example, an index for "Vintage cards" vs "Modern cards" or an overall Pokémon Card Market Index.
- This would involve weighting a basket of card prices. It's useful for giving context (e.g., if a card rose 5% but the market index rose 10%, it underperformed relative).
- Not a core user request, but a powerful analysis tool for advanced users. Possibly later phase.

- Still, we mention it as something the analytics engine could produce for dashboard (like showing overall market trend).

- **Backtesting Tool Implementation:**

- The system shall provide the backtesting functionality for premium users. This requires:

    - Access to historical data for the card(s) in question.

    - A way for user to define a strategy or input parameters. For MVP, perhaps provide a few preset strategies:
    - "Buy and hold from [date] to [date]" – then show ROI.
    - "Follow platform signals" – assume user bought when our signal changed to Buy, sold when it changed to Sell, over a timeframe (this tests our signals' efficacy).
    - Compare strategies: e.g., always hold vs. timed buy/sell.
    - The system then runs the calculation and outputs results. Because doing this dynamically could be heavy, maybe limit scope (e.g., one card at a time, or a small portfolio).
    - Results include metrics like total return (%), annualized return, volatility, max drawdown, etc., to mimic hedge fund style analysis.
    - It could also generate a chart of portfolio value over that period under that strategy.

- **AI-driven Recommendations:**

- The platform could have a feature to suggest cards to invest in or sell. This is different from the per-card signal; it's more like personalized advice:
    - For example, "Based on your portfolio or watch history, you might be interested in these cards which our model expects to rise in value."
    - Or a general list: "Top 5 cards to watch this week" based on some momentum algorithm.
- This could be delivered in a dashboard section or via a newsletter/alert for premium.
- Essentially, leverage ML to find patterns (maybe undervalued cards, or cards trending on social media that haven't seen price spike yet – indicating a potential upcoming rise).

- It might use anomaly detection (find cards whose price is lower than what indicators would predict, maybe due to lag or inefficiency).

- **Sentiment Analysis:**

- If we gather social media text data, the ML system can perform sentiment analysis (positive/negative tone about a card or market).
- This could contribute to signals (e.g., increasing positive sentiment might precede price upticks).
- Not necessarily exposed to user except perhaps a metric "Sentiment: 7/10 positive".

- Ensure any sentiment classification is correct often (not to mislead; maybe use known NLP services).

- **Continual Learning & Model Updates:**

- The ML models should improve over time. The system shall allow retraining of models with new data periodically (maybe monthly retraining of prediction model with the latest data).
- Also incorporate feedback: if our signals turn out wrong (e.g., said "Buy" but price fell), the model might adjust weight on factors.

- Possibly maintain a dataset of model predictions vs actual outcomes to evaluate performance.

- **Transparency and Override:**

- Because these are automated, there should be an ability for admin to override or adjust if something is clearly off. For instance, if a one-time event (like a sudden spike because of a celebrity purchase) fools the algorithm, the admin might mark that as anomaly so the model doesn't overreact.
- Also, for liability reasons, include disclaimers wherever predictions or signals are shown (like "Not financial advice, for informational purposes only").

Integration with UI: - The output of ML – primarily the **signal arrow** and any numeric ratings – must feed into the UI seamlessly (cards tile, card detail). - The predictions might show up as a number or chart extension on detail page. - Backtesting will be a separate UI (maybe a form and results page or overlay). - Recommendations might appear on user's dashboard or a "Insights" page.

Performance: - Any heavy ML computations (like training or large backtests) should be done server-side, possibly asynchronously or on dedicated threads, so as not to freeze the main user experience. - On-demand predictions (like if a user triggers a custom backtest) might take a few seconds; we should show a loading indicator and perhaps perform in a background task if it's long. - Real-time signal updates should be quick; computing a simple indicator for all cards might be done whenever prices update (which could be thousands of computations, but those can be optimized or spread out).

Rationale: The inclusion of ML and advanced analytics is what will differentiate this platform from simpler price trackers. It elevates the tool to something akin to a financial advisory tool. The user explicitly wanted "very good machine learning, algorithms, and custom PhD-level math formulas" to mirror what quant funds use – essentially making the platform feel authoritative and cutting-edge in its analysis. By delivering clear buy/sell signals and perhaps predictions, we cater to investors who may not have time or skill to analyze raw data themselves; it adds a layer of expert analysis at their fingertips. This is similar to how some stock apps provide analyst ratings or AI predictions for stocks. We must implement it carefully because bad or random advice can lose user trust quickly. However, if done well (even if just moderately accurate trend indicators), it gives users confidence in using the platform for decision-making. Moreover, features like backtesting serve two purposes: they give power users a sandbox to test their ideas, and they also validate our platform's signals (e.g., we can show historically our "algorithm" outperformed simple hold, if true). This can reinforce to users that the premium features are worth it. Overall, ML and analytics turn our large dataset into *actionable insights*, fulfilling the platform's promise of being data-driven and intelligent.

8. User Interface & Experience

(This section covers general UI/UX requirements not already mentioned in specific features, ensuring the product is user-friendly and has a logical design.)

Functional/Display Requirements:

- **Overall Design Aesthetic:** The platform's look and feel should convey **professionalism, data-centric design, and trustworthiness**, akin to financial dashboards, while also being appealing to the Pokémon collector community.
- Use a clean layout with a neutral or dark theme background (many financial tools use dark mode by default for better chart viewing, we can consider that).
- Use color-coding where it provides meaning (green for positive changes, red for negative, etc.) consistently across the site.

- Important metrics should be easily visible; avoid clutter by using progressive disclosure (show basic info, allow clicking or hovering for more details).

- **Navigation:**

- A clear navigation menu should be present, with sections such as **Home/Dashboard**, **Search** (or simply a prominent search bar on top), **Portfolio**, **Watchlist**, **Alerts**, **Insights** (if any blog or news), **Profile/Account**.
- Users should be able to get to key areas within 1-2 clicks. E.g., from the home they can search immediately, from any page they can navigate to portfolio.

- On mobile web/app, use a bottom nav or hamburger menu for the same sections.

- **Home Dashboard:** The homepage (after login, or even for guests with limited data) should provide an overview of the market and user's interests:

- For logged-in: Show a summary of their portfolio (e.g., "Portfolio Value: $X, up Y% today") at a glance, maybe a mini-chart [14] .
- Show top market movers: e.g., "Top 5 Gainers today: [Card name + % up]" and losers similarly.
- Possibly trending search terms or popular cards (to engage users to click on something).
- Maybe a news headline or announcement if we have a blog ("New set release coming next week…").
- Basically, a one-stop glance of what's happening.

- For not logged in users: show general top movers and a call to action to sign up to track their collection.

- **Responsiveness:** The UI shall be fully responsive:

- On desktop, it can use multi-column layouts (for example, portfolio page might show list and side widget).
- On mobile, those collapse to single column scroll. Charts should be swipe-friendly or at least viewable by pinch-zoom if needed.
- Use responsive charts libraries that can adapt or have simplified views for small screens.

- Ensure text is readable on smaller devices (use appropriate font sizes, avoid very wide tables that require side scroll).

- **Discord Bot UX Consideration:** Although Discord isn't part of the web UI, ensure the information we present can be summarized in text form as well. Possibly, when implementing the bot, we might reuse some formatted outputs. Keep message length and clarity in mind (for example, the one-line summary of a card: name, price, change, link).

- **Accessibility:**

- Use alt text for images (card images should have alt text like "Pikachu card image" for screen readers).
- Ensure color contrasts meet accessibility standards (since red/green colorblind users might not distinguish arrows, maybe include symbols or labels "Buy/Sell" text on hover).

- The site should be navigable via keyboard (for users who cannot use a mouse).
- Considering global audience: possibly plan localization later (not MVP, but design so text isn't hard-coded or concatenated poorly, etc.).
- **Error Messages & Feedback:**
- When an action succeeds or fails, the user should get clear feedback. E.g., "Card added to portfolio successfully" or "Failed to fetch data, please retry".
- Use modals or toast notifications for short messages.
- If a page fails to load data, show a partial page with an error section rather than a blank page.
- **Help and Onboarding:**
- Since the platform has advanced features, provide a quick onboarding tutorial for new users. Possibly highlighting where to search, how to add to portfolio, etc., with tooltips.
- Have a Help/FAQ section (maybe static pages) that explains key concepts (what is an alert, how do we calculate prices, etc.). This can reduce confusion and support requests.
- Provide info icons or tooltips near complex metrics. For example, a small "i" next to "Volatility" metric which when hovered explains "Volatility: a measure of price fluctuation...".
- **Integrations UI:**
- If users link their account to Discord or other services in future, the profile page should show status and give instructions (like "To link Discord, use command !link [code] in the bot" etc.).
- For mobile, ensure that whatever design we have can translate to native mobile components (we might do a similar layout or slightly adjusted in the app).

Non-functional:

- **Usability Testing:** We should aim to test the UI with a few actual users or at least developers pretending to be users to ensure it is intuitive. Any confusing workflow should be refined. For example, adding a card to portfolio should not require too many steps.
- **Performance (UI):** Pages should load within a couple seconds on typical broadband. Use techniques like lazy loading images (don't load all card images if not in view), compress assets, etc.
- **Consistency:** Maintain a consistent style across the app (fonts, button styles, spacing). Possibly create a style guide or use a UI framework to ensure uniformity.

Rationale: A detailed SRS often might not enumerate UI specifics, but given the importance of user adoption, we highlight key UI/UX considerations. We want a platform that is both **powerful and approachable**. The target users vary from casual collectors to data-savvy investors, so the interface must cater to both – meaning it should not intimidate newbies (hence, friendly search, tooltips, and clean design), but also not frustrate advanced users (hence, lots of data accessible when needed, customizable views, etc.). The UI plays a huge role in making the complex analytics digestible. If we present data like a raw spreadsheet, many users would be lost; but if we use visual cues (like arrow icons, colored tiles, charts), users can understand at a glance. This directly supports our goal of being the top platform – not just by

having data, but by presenting it in the most user-friendly manner, something competitors might lack. A professional look also builds credibility; for example, using a layout similar to known financial tools might subliminally signal to users that this platform is serious and reliable, increasing their trust in the information provided.

9. Security and Privacy

Description: The platform must ensure security of user data and privacy, especially since it deals with potentially valuable asset information (user collections) and integrates financial-like data. Users need confidence that their account is safe and their personal information will not be misused.

Security Requirements:

- **Authentication Security:** All login and sensitive account actions must occur over HTTPS. Passwords stored in the database must be hashed (with a strong hashing algorithm like bcrypt or Argon2) and salted. No plaintext passwords anywhere.
- Implement measures against brute-force (e.g., rate-limit login attempts, and/or use captcha after certain failed tries).
- Sessions or JWT tokens should be secure (if cookie-based, mark them HttpOnly and Secure to prevent XSS theft).
- **Access Control:** Ensure that user-specific data (portfolio, alerts, etc.) can only be accessed by that authenticated user. For example, API endpoints for getting portfolio must verify the session token corresponds to that user. No other user should be able to query another's data by manipulating IDs, etc.
- Admin-level functions (like altering data or viewing all users) should be restricted to admin accounts and possibly further protected (maybe IP restricted or separate interface).
- **Data Encryption:** Any sensitive personal data (like user email, if needed, or any billing info if stored) should be encrypted in transit (always via TLS) and at rest if possible. We might not store credit cards ourselves (Stripe handles that), but if we store even partial info or tokens, treat carefully.
- Portfolio data might not require encryption at rest, but if we anticipate targeted attacks (stealing collection info could be sensitive), we could consider it or at least ensure database is secure.
- **Privacy Compliance:** If serving international users, comply with laws:
- Provide a Privacy Policy detailing what data we collect (e.g., we collect user email, portfolio data, etc.) and how we use it (for analytics, improving ML, etc.) and how we don't (we won't sell it to third parties without consent).
- Allow users to delete their account (and thus personal data) upon request. This means designing a process to remove or anonymize their info in all systems (including removing them from analytics logs if needed).

- If a user is in EU or region with specific needs, ensure we allow them to download their data if requested (data portability).

- **Secure Integration with Third Parties:**

- API keys to external services (e.g., eBay, TCGPlayer) should not be exposed to the client. They should reside on the server and calls made server-side.
- When using webhooks (like Stripe for payment events), verify signatures to ensure the request is genuinely from Stripe.

- The Discord bot token should be kept secure on server; if compromised, someone could impersonate our bot.

- **Fraud Prevention & Integrity:** Although we are not a transaction platform, we should consider:

- If we provide any user-generated content in future (like comments or sharing portfolios), moderate to avoid spam or scams.
- Ensure our affiliate links cannot be hijacked (e.g., someone trying to replace our affiliate code).

- Monitor unusual activity on accounts (like a sudden huge number of alerts created could be a spam bot – maybe incorporate basic limits or admin alerts on such anomalies).

- **System Security:** The platform servers should be kept updated with security patches. Use firewalls, and possibly WAF for the web app. Protect against common web vulnerabilities (XSS, CSRF, SQL injection). E.g., use parameterized queries for DB, and implement CSRF tokens on forms or use same-site cookies.

- If storing any files (like user-uploaded images if we ever allow scanning upload via web), scan for malware.

- **Backups and Recovery:** Regularly back up the database and critical data. If a disaster or data corruption occurs, we need the ability to restore data to avoid losing user portfolios or price history.

- Keep backups encrypted and in a secure location. Test recovery procedures.

Rationale: Security is foundational; if there is a breach or loss of trust, users will abandon the platform, especially when it concerns financial-equivalent data. Even though we might not be dealing with actual money, users' collections can be worth thousands of dollars, so they may be secretive about what they own (to avoid theft etc.). Leaking that info or having accounts compromised could have real-world repercussions. Therefore, we treat security with high importance. Additionally, compliance with privacy laws is not only legal necessity but good for user trust (we want to be seen as professional and legitimate). By outlining these requirements, we set a standard that our development must follow to ensure user data and our data (like API keys) remain safe. This will help make our platform enterprise-grade, ready for potentially thousands of users, and perhaps even partnerships with companies that will expect strong security posture.

10. Future Expansion and Product Lifecycle

Description: This section outlines the envisioned product lifecycle – from MVP to full product – and additional features or integrations that may be pursued in the future. It also summarizes the end-goal vision to ensure we know what we're building towards, which will guide prioritization and design decisions. Understanding the full scope helps in delineating what the **Minimum Viable Product (MVP)** will include versus what can be phased in subsequently.

Product Roadmap Phases:

- **Phase 1: Minimum Viable Product (MVP) – Core Web Platform**
- **Key Features in MVP:**
  - Card search and browsing with basic tile view (card info, image, and maybe a simplified indicator like just price change % or trend arrow).
  - Card detail page with essential data: at least one price chart (maybe last 30 days or 1 year), current prices (raw and one graded like PSA10), and basic info. Possibly initial version of buy/sell signal (could be rudimentary, e.g., based on 7-day trend) [6].
  - User accounts (registration/login) and a simple portfolio tracker: user can add cards, see current value and total, and basic profit calculation [7]. Perhaps no fancy charts in MVP, or a simple portfolio chart if easy.
  - Watchlist (single list) and ability to set a basic alert (maybe one type: price above/below threshold) – or at least the infrastructure for alerts even if UI only exposes one or two types initially.
  - Data integrations: at least one reliable source for prices (maybe TCGPlayer's market price for raw and some reference for graded like a fixed dataset or initial scraping of eBay). The system must be seeding initial price data and updating daily.
  - Affiliate links integrated on card pages for monetization from day one (even if traffic is low initially, it's good to have).
- **Quality Bar for MVP:** The MVP should be functional and demonstrate the unique value (tracking and basic analytics) enough to attract early adopters (likely from the community). It might not have full ML sophistication yet, but at least the data should be correct and UI solid.
- The focus is on delivering the **core user journey**: search → view card → add to portfolio → see collection value.

- Machine learning in MVP might be minimal (perhaps use simple rules for signals), given training models might need more data/time. But design the system so ML can be plugged in later (i.e., separate a placeholder for signal generation logic).

- **Phase 2: Enhanced Analytics and Premium Features**

- **Expanded Features:**
  - Implement the full machine learning algorithms for buy/sell signals and price predictions. Begin incorporating external signals (social, Google Trends) into the models. The UI can now promote that our signals are "AI-driven" for marketing.
  - Introduce the backtesting tool for premium users. Also possibly add more alert types (percentage change alerts, portfolio change alerts).

    - Expand watchlist functionality (multiple lists) and allow sharing of watchlists or portfolios via link for those who want it.
    - Improve the card detail page with more charts (e.g., separate raw vs graded chart, volume chart) and perhaps a news or social feed snippet if relevant (like "5 mentions on Reddit today").
    - Add more data integrations if needed (e.g., pulling data from additional markets to cover holes).
    - Possibly start any content/community features (like a news blog or ability for users to comment on cards or something, if deemed useful to keep engagement).
- **Premium Subscription Launch:** Around this phase, start actively marketing premium tier with features that justify it: e.g., "Pro members get advanced analytics, unlimited alerts, and backtesting." Ensure the payment system is fully integrated and stable.

- **Scaling Consideration:** If user base is growing, move components to more scalable infrastructure as needed (e.g., separate database read replicas for heavy analytics, etc.). Also use feedback from MVP users to refine UI/UX.

- **Phase 3: Mobile App and Discord Integration**

- Now develop and release the **mobile apps** (iOS and Android). This involves:
    - Adapting the UI for smaller screens (if not already responsive) or creating native layouts.
    - Implementing the card scanning feature: integrate an image recognition service or build a model to recognize cards from camera input. This feature can be a big draw for casual users (scan a card to get its price, as mentioned in the Bloomberg article about scanning tech [4]).
    - Ensure all core features (search, view, portfolio, alerts) work in the app. Use push notifications for alerts.
    - Beta test with a small group for feedback before full launch on app stores.
- Develop the **Discord bot**:
    - The bot should handle commands like `!price [cardname]` returning a concise message with the card's latest price and arrow signal.
    - Also `!portfolio [username]` perhaps (if user links accounts, might DM them their summary).
    - Possibly `!top5` to list top movers, etc., to add value in a community chat.
    - The bot will help drive traffic to the platform (by being present in multiple servers, it's like free marketing).
    - Ensure scalability (if many queries at once, our API can handle it).

- **Marketing/Community:** By this phase, likely have a community of users – perhaps run a Discord server for our platform users, engage with them for feedback and promotion. Use social media to showcase cool analytics (like interesting card trends).

- **Phase 4: Full-Fledged Platform & Continuous Improvement**

- After initial releases, continue iterating with more sophisticated features:
    - Possibly add support for other collectibles (e.g., Magic: The Gathering, sports cards) reusing the same engine – this could exponentially grow user base but also needs more data. We'd only do this if Pokemon side is very successful and stable.

    - Build an **API for third-parties**: if our data is good, others might want to pull it (developers, or small apps). Could be a product in itself (with API keys, maybe a paid service for data).
    - Integrate with grading companies or marketplaces in deeper ways: e.g., maybe partner so users can submit cards for grading through the app, or list their cards for sale on eBay directly from their portfolio (with some streamlined process).
    - Use user data to create interesting aggregate insights (e.g., average portfolio value of collectors, popular cards in portfolios – sharing those stats anonymously as content).
    - Possibly introduce gamification or social features (users earn badges for collecting sets, or they can follow other public portfolios, etc.) to increase engagement, though this is optional depending on strategy.
- Keep refining the ML models. As more data accumulates (including user behavior data like what signals users acted on), fine-tune predictions. Perhaps incorporate new techniques (like deep learning if appropriate).

- Monitor performance and cost: as we have more features and users, optimize costs (maybe caching more to reduce API calls, etc.), and ensure high availability (maybe multi-region deployment if global user base).

- **Long-term Vision (End Goal):**

- The platform is the undisputed hub for Pokémon card market intelligence. Collectors and investors routinely use it to make decisions; it's cited in the community as the go-to source.
- It might even influence the market (if many follow its signals, it becomes like an index).
- We have a large user base with a healthy conversion to premium for advanced features, and sustainable revenue through both subscriptions and affiliates.
- Ideally, nothing like it exists outside – any competitor would be playing catch-up with our feature set and data quality. Our combination of **real-time data, powerful analytics, and ease-of-use** sets us apart.
- The product might expand to related domains (other TCGs, or even other collectibles like comics or coins) leveraging the same platform, thus growing the business, but even if just focusing on Pokémon, we cover from casual fun to serious investing.
- The platform could eventually attract partnerships (with grading companies or marketplaces) or acquisition offers due to its central role in the hobby.

MVP vs Full Feature Recap: To clarify, the end-goal includes everything described in this SRS: multi-source data, ML algorithms, mobile and Discord, etc. The MVP is a critical subset that can be built faster: - MVP will have: search, card pages with basic charts and data, user accounts, simple portfolio tracking, basic watchlist/alert, core integrations (at least one data source for now), and fundamental buy/sell indicator (even if simplistic to start). - The more complex elements like AI prediction, backtesting, extensive alert types, mobile app, Discord bot, etc., are part of the full product but will come in later iterations. - However, even in MVP design, we keep the final architecture in mind (for example, designing the database and API in MVP such that adding mobile later is smooth, or leaving hooks for additional data fields like sentiment scores even if not used yet).

By structuring development in phases and understanding the final destination, the project can focus on delivering immediate value while building the foundation for the comprehensive platform it will become. Each phase's feedback will inform the next, ensuring the final product truly meets user needs and achieves the vision of being the ultimate Pokémon card investment hub.

References

The requirements and features outlined in this SRS are informed by a combination of project vision and insights from existing market resources and user communities:

- A Bloomberg report on the Pokémon card boom highlighted the demand for tools that *"allow collectors to scan, price, organize and track the value of their card collection, similar to a stock portfolio"* [4], underscoring the need for portfolio tracking and card scanning capabilities.
- Community discussions praise platforms like PokeData for providing a clean experience to *"track your portfolio, build lists, search cards, check prices... it pretty much has everything you'd want."* [11] This validates features such as watchlists, price checking, and portfolio tracking as essential.
- Market intelligence blogs note that tools like **TCGPlayer, eBay sold listings, and PSA Price Guide** are indispensable for gathering price data [16] [8], which supports our integration of those data sources.
- Real-time pricing and portfolio features in existing tools (e.g., TCGTrends and PokeData) demonstrate the feasibility and demand for features like **price change analytics and portfolio value tracking** [2] [3]. TCGTrends in particular gathers data across multiple platforms (eBay, TCGPlayer) to help users *"make quick decisions and get the best deal on the fly"* [6], guiding our approach to multi-source data aggregation and timely alerts.
- The importance of graded card data is highlighted by the use of **PSA's Population Report and Price Guide** in the community [9], so our inclusion of graded price tracking and pop reports is well-founded.
- The value of **alerts and instant data** is reflected in user expectations of tools with *"real time pings faster than most other sources"* (as mentioned in community content), which reinforces our focus on real-time alerts and updates.
- Finally, broader market analysis suggests that informed collectors use a combination of resources to stay ahead [17]. Our platform aims to unify those resources, a goal that no single existing tool fully achieves, thus filling a gap and providing a unique value proposition to our target audience.

These references and community insights have been incorporated to ensure the SRS not only captures the project vision but also aligns with proven user needs and successful features in the domain. The end result is a detailed blueprint for a platform that is ambitious yet grounded in real-world demand, paving the way for a successful development and deployment.

---

[1] [4] [13] Pokemon card frenzy making collectors, start-ups rich - Taipei Times
https://www.taipeitimes.com/News/biz/archives/2021/07/11/2003760634

[2] [3] [5] [8] [9] [16] [17] Staying Informed: Top Resources for Pokémon Card Market Intelligence - BlockApps Inc.
https://blockapps.net/blog/staying-informed-top-resources-for-pokemon-card-market-intelligence/

[6] [10] [12] Pokemon Card Value Tracker & Collection Manager | TCG Trends
https://tcgtrends.com/

[7] [11] [15] You should be using Pokedata! : r/PokeInvesting
https://www.reddit.com/r/PokeInvesting/comments/1i8n0f5/you_should_be_using_pokedata/

14 PokeDATA - Track your Pokemon Collection and Monitor your Portfolio of Investments!
https://www.pokedata.io/portfoliolanding