Cosmic Trajectories and the Challenge of Financial Data Management

EPPS 6354 - Information Management

Dr. Karl Ho

Joseph Martinez

May 11, 2025

Introduction & Motivation

Today, investment advisor representatives (IARs) use a plethora of tools to handle the data ecosystem needed to run a registered investment advisory firm (RIA). A mix of sales software for data collection and analysis of the whole financial well-being of RIAs and their clients has become mainstream, such as Arch, Salesforce, and Addepar. Meanwhile, market analysis tends to be done using market analysis software such as Bloomburg, Thinkpipes, or Koyfin. Only after both the market and business/client data are collected are they finally analyzed in tandem, typically through workbook software such as Airtable or Excel.

This means in order for IARs to truly analyze a client's portfolio, they must pull information from at least two separate software programs into a workbook that is usually driven entirely by the IAR leading the case. There are advantages to this model. Workbooks are incredibly accessible; although hard to master, they are easy to learn. This means clients and IARs without a technical background can easily work with the end deliverable of an IAR running a portfolio analysis. The software used for client/RIA data collection can usually also run the reports needed for the RIA to run their operations effectively and with confidence in the direction of the business (as long as data entry remains clean and factual). However, this system is also limited from more advanced modeling that could save IARs time-value and maximize the output of the RIA's more technically skilled employees.

This is why the idea of *Cosmic Trajectories* was initially inspired. Cosmic Trajectories' end goal would be to become a one-stop shop for client portfolio analysis that can become an essential part of an RIA's data ecosystem. It would handle all the market data management and analysis the IARs need after being fed client portfolio data from the RIA's sales software—replacing both the market analysis software and the workbooks.

Essential to Cosmic Trajectories would be quality data management. Data must be easy to extract and run so that new tools can be designed with ease. Cosmic Trajectories will mostly act as a tools library IARs can use to quickly run reports on client portfolios or market data in general. This was an ambitious design, so for this class, only a pilot tool was set for completion.

However, the challenge of the schema needed in order to have a database that was not only usable by the end user but also secure to protect financial information was a challenge. Being an amateur at the time in SQL and data management, a patchwork schema was organically grown to handle the creation of a usable output. Due to schema creep, the resulting output had a fundamentally different schema than the original ambition set out to accomplish but with a strong and flexible foundation that can be built upon to either become a tool for RIAs or retail investors.

Original Project Ambition

The original ambition of the project was to create a schema pulling real market data from TradingView widgets and advisor-provided client portfolio data through a data entry tool. Client portfolios must be cleanly segregated from each other and other funds. Holdings of mutual funds and exchange-traded funds would also be categorized under the fund, including the percentage of holdings. Two tools would be built: a portfolio allocation tool and a correlation analysis tool.

The ambition was for the portfolio allocation tool to allow investment advisors to set target allocations and modify the asset classes for target allocations, which will be as follows: public equities, public credit (fixed income), public real estate, private equity, private credit, and private real estate. Each asset class can have a target allocation of 0-100% of the portfolio. All allocations must equal 100% when added together. Each security should be assigned an asset class so portfolios can accurately show their current allocation.

A correlation dashboard would also be built for IARs to use to compare portfolios, funds, and financial securities to set indexes. The dashboard will return a correlation's beta, alpha, and a few visualizations to provide a more complete picture.

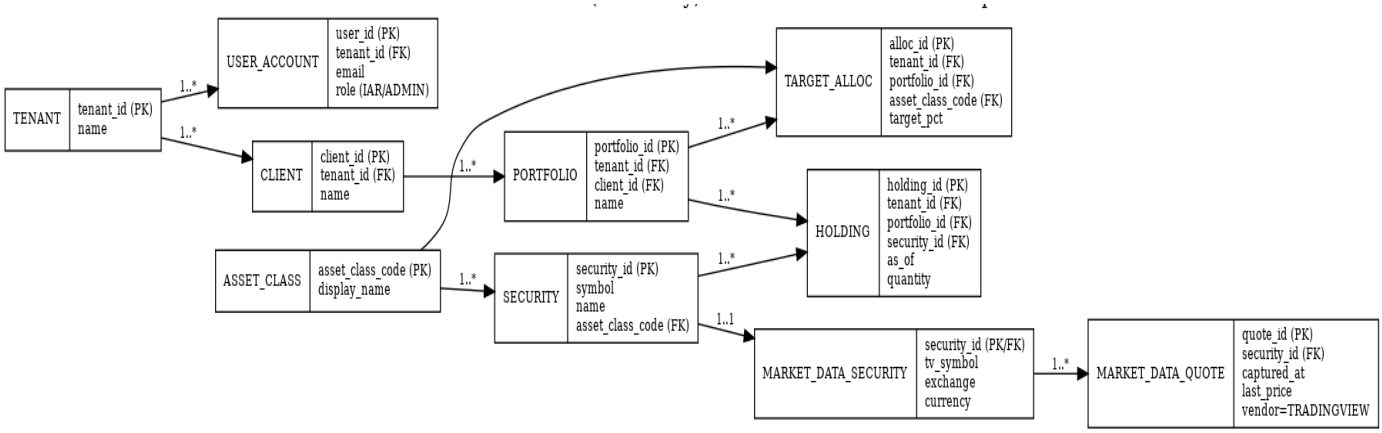

Image I - Proposed Schema: *The original proposed schema for the project.*

Initial Wins, Challenges, & Branding

The first win was the creation of a website titled "Cosmic Trajectories" and an associated application embedded into the website titled "Trajecta." The website, although an unnecessary step in the project, was used as a creative outlet to build out a brand that took inspiration from the brands and logo designs of the University of Texas at Dallas (UT Dallas) and Texas Instruments due to the mass influence Texas Instruments has on UT Dallas's brand, identity, and history. The name "Cosmic Trajectories" was chosen to combine the *fintech* concept behind the project with the "Comets" mascot of UT Dallas. The color scheme of black, white, and a deep burnt orange was used to create sharp contrasts. To create a fintech brand that spoke to the user, "orbital finance," Orbitron and Inter were settled on as fonts.

Image II - Brand Inspirations: *The banners of UT Dallas and Texas instruments for reference of ibrand inspirations.*

Image III - Brand Inspirations: *The banner logo for Cosmic Trajectories. Showcasing both the inspiration from Texas Instruments and UT Dallas, alongside*

After using the website design as a sandbox to build out a brand identity, a draft application titled "Trajecta" was designed, named as such to be consistent with the orbital finance theme. This initial design came with massive early wins. The stock data, which was sourced from TradingView, seamlessly flowed into the app. Allowing real-time data to provide up-to-date stock market information to the user. Additionally, the application clearly worked in its ability to import portfolio data from a spreadsheet on a user's computer into the application. Not only that, but the user could then seamlessly export a report based on the imported data. The user could also manually enter portfolio information as well. The application also allowed the user to filter through several graphs showing the historic performance of the individual positions (such as stock or exchange-traded funds) inside the portfolio.

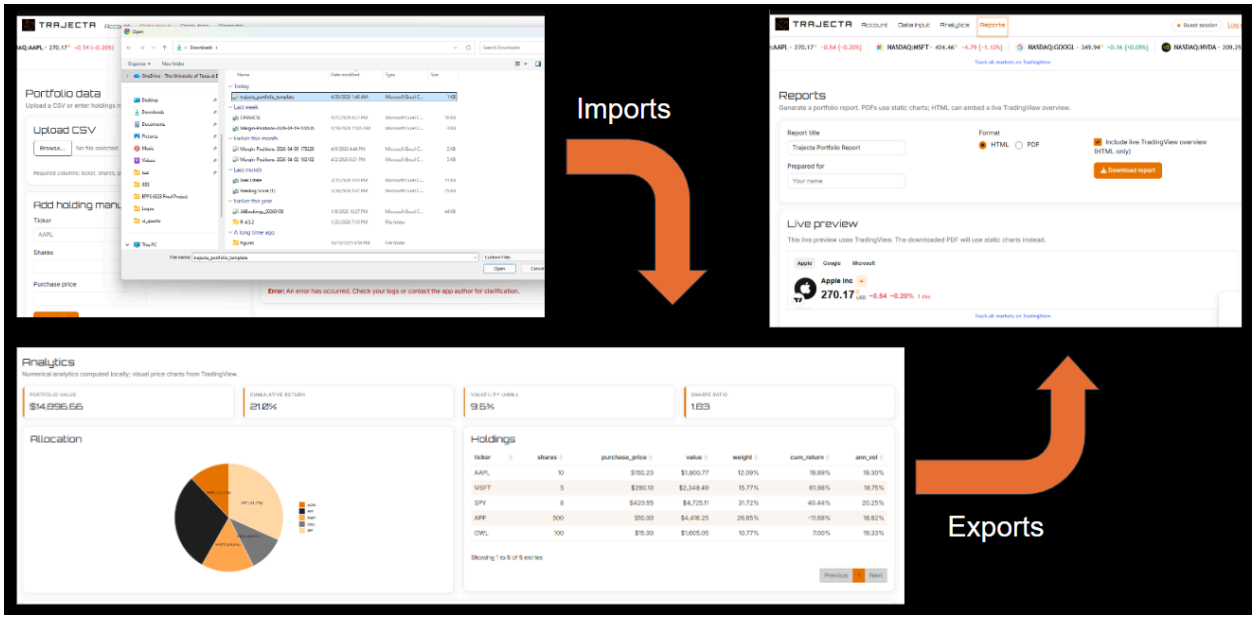

Image IV - Rough Draft of Application: *The photo shows the initial success of creating an application that can receive imported data from a user and export outputs using said data. The application at the time was unable to save outputs to a specified user. All work by a user would be forgotten by the end of the session.*

However, immediately the application was exposed to two problems. Firstly, some of the outputs the application would produce were inaccurate. For example, the application would showcase a

portfolio market value that was calculated by multiplying the cost basis of the portfolio by the size of the portfolio's position—which is the calculation of the portfolio's initial purchase price (also known as cost basis), not the market price of the portfolio. More importantly, the user's interaction with the application would reset at the end of each use. Loosing all progress made whenever the user closed out of the application. There was a desperate need for the application's database to be able to be updated by the application's user in such a way that each user's portfolios had clean separations while maintaining usable relationships. In the end result, user's must be able to have the application link them to their portfolios, their portfolios to securities, and securities to market data records without overlap or ambiguity

Adaptive Solutions

The Cloud Server

In order to tackle the main challenge of users having their interactions with the application updated into the database, offloading the database's schema from a local server to a cloud server was paramount. The solution had to be affordable, so after analyzing the cost-benefit analysis on various providers, Neon was chosen. After creating an account, I created a project on neon.tech titled "Trajecta" with a server based out of Ohio, USA.

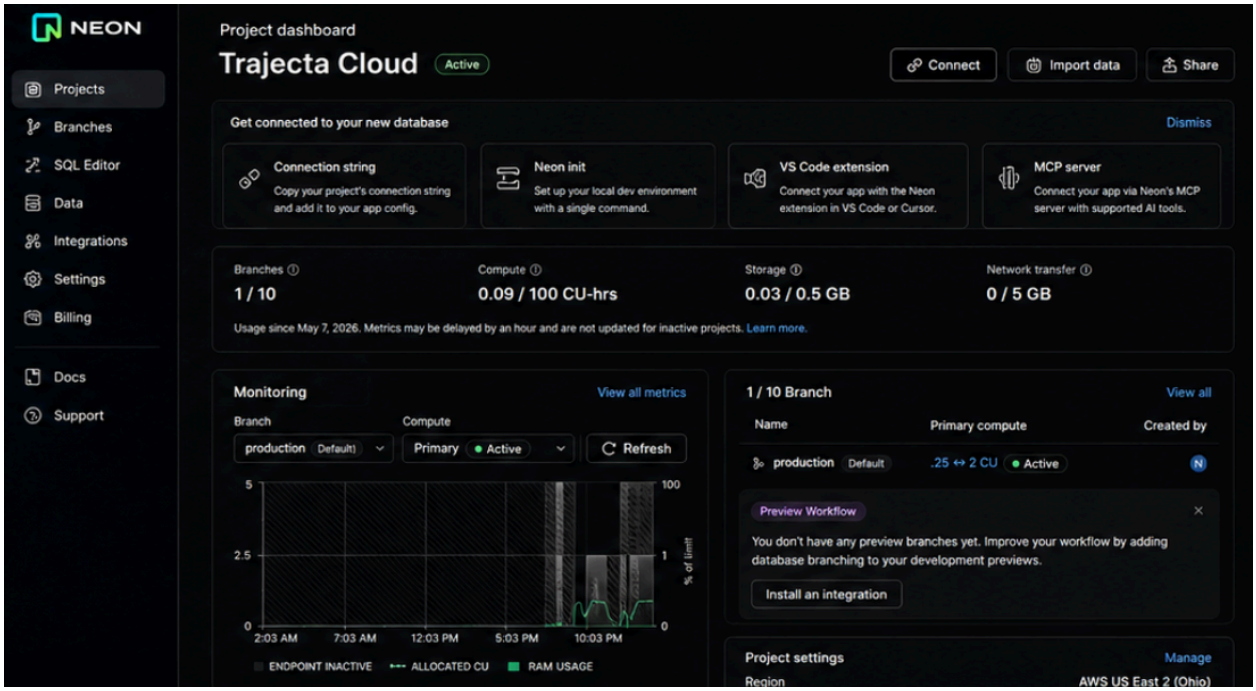

Image V - Neon.Tech Project Dashboard: *An example of the dashboard used by neon.tech. The project allows up to 0.5 GB of data storage and up to 100 user hours for the free version. Which was perfect for the scale of the project during the duration of the semester.*

After setting up the Neon project, a new pgAdmin4 cloud server had to be created. To keep the larger project's various software easy to keep up with, the server was named "Trajecta Cloud."

Then, the pgAdmin4 server was offloaded onto neon.tech by making the neon.tech project the owner of the pgAdmin4 server.

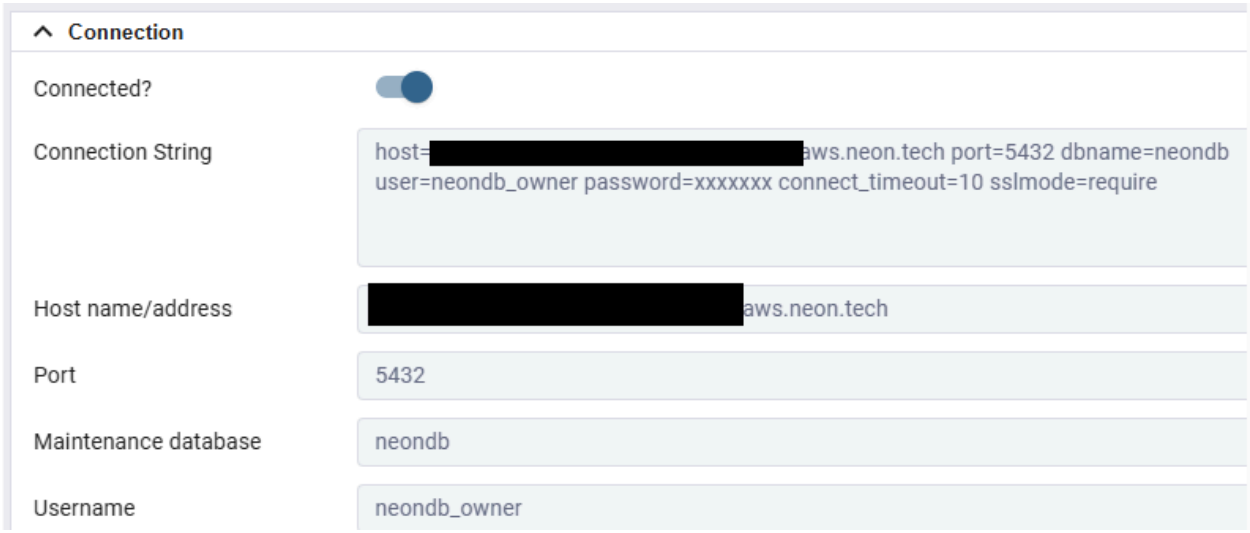

Image VI - pgAdmin4 Connection: *The image shows the neon.tech "Trajecta" project's ownership over the pgAdmin4 project "Trajecta Cloud." It is this ownership that allows users to update the database and only affect the parts of the database in relation to them.*

The Patchwork Schema

Once this was set up, the schema could be adjusted through trial and error. The schema had to shift away from only modeling financial securities and toward solving the more practical problem of user accounts, saved portfolios, and persistent app data. In that sense, the schema became a patchwork schema that was organically grown through mistakes, revision, and practical problem-solving rather than created all at once from a perfect original design.

At the center of the schema is the users table, which acts as the anchor for the entire application. This table stores the core account information needed for registration and login, including "user_id," "email," "password_hash," "is_active," "created_at," "updated_at," and "last_login_at." The use of email "CITEXT NOT NULL UNIQUE" is important because it helps prevent duplicate accounts based only on capitalization differences in an email address. The schema also includes a basic email format check, showing that even though the design is practical and patchworked, it still attempts to create guardrails around the quality of user data. This table is the foundation for the rest of the schema because almost every major part of the application eventually needs to connect back to a specific user.

The "user_sessions" table builds on the user account system by creating a place to track login sessions. Its relationship to the user table is made clear through the line "user_id BIGINT NOT NULL REFERENCES users(user_id) ON DELETE CASCADE." In plain terms, this means each session belongs to a specific registered user, and if that user is deleted, the related sessions are deleted as well. The schema also includes fields such as "expires_at," "revoked_at," "user_agent," and "ip_address," which means the system was became designed not only to let users log in, but also to eventually support more complete session management, audit trails, or

remember me features. This is a good example of the schema's organic growth: the immediate need was login, but the table leaves room for stronger session controls later.

The "password_resets" table shows that the idea of password recovery is present in the schema, but this part has not yet been tested. The table includes a "token_hash," "created_at," "expires_at," and "used_at," which are the pieces needed for a reset-token system. The schema also notes that only the hash of the reset token is stored, while the raw token should only be shown once through the user interface or email. This is the correct general direction for a safer password recovery design, but it should not be presented as fully functional. At this stage, the schema contains the structure for password recovery, but the feature itself should be described as planned or drafted rather than complete.

The "application_data" table is one of the clearest examples of the patchwork but flexible nature of the schema. Instead of creating a separate table for every possible user setting or app state, the schema creates a more open-ended storage space using "data_content JSONB NOT NULL DEFAULT '{}'::jsonb." This allows the application to save flexible information, such as the user's open tab, most recent portfolio, preferred settings, or other app-state details. This design probably emerged because the application needed to remember what the user was doing without requiring the developer to perfectly predict every future feature. It adds some complexity because JSONB data can become messy if not managed carefully, but it also gives the project room to keep growing without constantly rebuilding the database structure.

The financial side of the schema is mainly handled through the portfolios and holdings tables. The portfolios table creates one saved portfolio name per user, using "UNIQUE (user_id, portfolio_name)" to make sure a single user cannot accidentally create duplicate portfolios with the same name. The holdings table then stores the actual positions inside each portfolio, including "ticker," "shares," "purchase_price," and "purchase_date." This creates a clean chain where users connect to portfolios, and portfolios connect to holdings. That structure directly responds to the earlier problem that the application needed to "link them to their portfolios, their portfolios to securities, and securities to market data records without overlap or ambiguity." While the schema does not fully return to the original, more ambitious securities-and-market-data design, it creates a workable foundation for separating one user's financial information from another's.

The schema also includes several practical maintenance features that make it more usable as an actual application database. Indexes such as "idx_portfolios_user" and "idx_holdings_portfolio" are included to make common lookups faster, especially when the app needs to find a user's portfolios or the holdings inside a selected portfolio. The shared trigger function "trg_set_updated_at()" automatically refreshes the "updated_at timestamp" when certain records are changed. This matters because the application is not just storing static information; it is trying to preserve an evolving user workspace. The view "v_latest_portfolio" also supports this goal by making it easier to identify the most recently updated portfolio for each user. These features show that the schema is not only storing data, but also beginning to support the user experience of leaving the app and later returning to it.

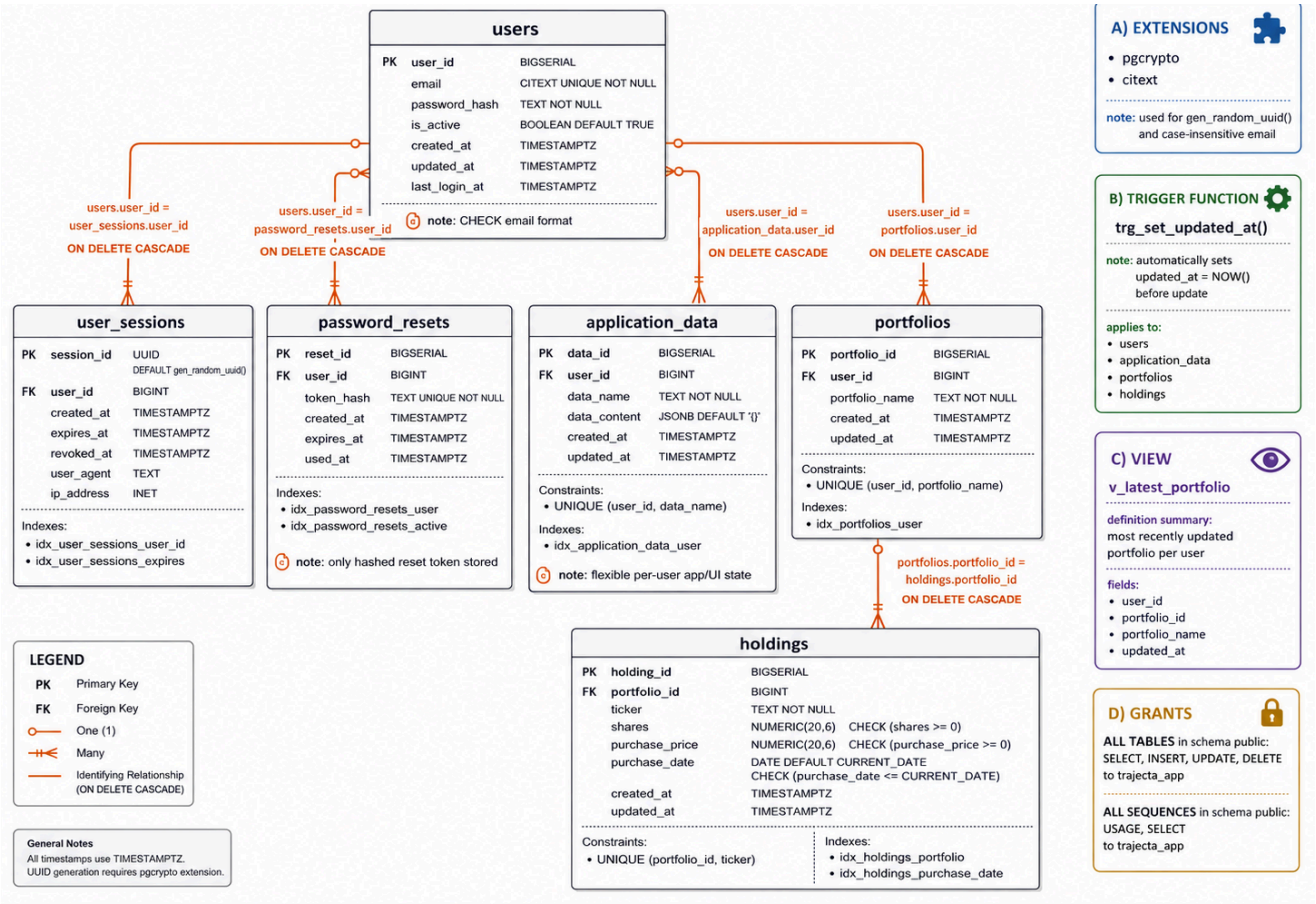

Image VII - Trajecta, PostgreSQL Schema: *This schema map illustrates the relational backbone of Trajecta, showing how user authentication, session management, portfolio storage, and holdings data connect within a centralized PostgreSQL database architecture.*

Divergence from Initial Ambition

As you can see, this schema represents a major divergence from the original ambition, but not necessarily a failure of it. The first ambition was a cleaner and more complete financial data model, but the actual development process revealed that user persistence, account separation, and saved portfolio data had to come first. This created schema creep and added complexity, but it also forced the project to build a stronger practical base. The final result is still patchwork, but it is a useful patchwork: the schema now supports registered users, user sessions, flexible app-state storage, saved portfolios, and portfolio holdings. It is not the final version of Cosmic Trajectories, but it gives Trajecta a foundation that can continue to be cleaned, normalized, tested, and expanded upon.

While this schema creates a useful foundation for user accounts, saved application activity, portfolios, and holdings, it does not yet fully support the kind of account structure that registered investment advisors would need in order to manage client accounts professionally. At this stage, the schema is centered primarily around the individual “user_id,” which works well for a single user saving their own application activity, but it is not yet designed around advisor-client

relationships, household-level financial planning, compliance workflows, or a formal book-of-business structure. This limitation should not be understood as a failure of the schema, but rather as evidence that the project is still developing and that its future direction has not been fully locked into one audience. If Trajecta continues toward the RIA model, the existing foundation could be expanded with additional relationship-based layers such as "advisor," "firm," "client," or "portfolio" or other identifiers that more clearly connect advisors, firms, clients, and portfolios. At the same time, the schema could also evolve in a different direction and become a standalone tool for retail investors, where the individual user remains the center of the database and the application focuses on helping that user better understand how their finances, portfolios, and investment decisions are working over time. In this sense, the schema's incompleteness is also one of its opportunities: because it was built organically, it remains flexible enough to be shaped toward RIAs, retail investors, or some hybrid of both.

The Broader Shifting Industry Paradigm & Cosmic Trajectories' Future Direction

Over the past several years, registered investment advisors and broker-dealers have been moving away from purely manual, records-centered information management and toward more integrated systems built around artificial intelligence, automation, cloud infrastructure, and stronger data governance. This shift has not happened evenly across both types of firms. RIAs have tended to use technology around fiduciary advice workflows, client questionnaires, portfolio management, rebalancing, digital disclosures, and personalized planning tools. Broker-dealers, by contrast, have often had a broader operational burden, using data systems to manage customer account records, communications review, supervisory trails, trading records, market-access controls, and surveillance obligations. In large part, this contrast is pushed by the differences in regulations between the two types of firms—with RIAs being overall less regulated than broker-dealers. The broader trend is that firms are having AI layered into existing compliance, recordkeeping, client service, and portfolio analysis workflows.

However, these tools also create new responsibilities around accuracy, privacy, recordkeeping, model reliability, and misleading AI claims. The practical lesson is that AI does not remove the old duties of financial firms; it makes the information-management environment more complex and makes strong data infrastructure more important. In summary, the industry is moving from basic record retention to a fuller information governance, where client information, market data, advisor communications, model outputs, and compliance documentation are organized in ways that are usable, reviewable, and secure in the new AI paradigm.

Nevis Wealth appears to represent the latest possible innovation for this broader industry movement. The self-proclaimed "AI platform for wealth management," Nevis has built a product meant to help financial advisors grow, reduce administrative work, and spend more time with clients. The platform's listed capabilities include meeting summaries, automated task generation, smart meeting preparation, AI search across client information, client-ready emails, and account opening support. This shows a clear industry direction: advisory technology is increasingly being built around integrated workflows, organized client data, and AI-supported tools that help advisors operate more efficiently while still keeping the human advisor at the center of the relationship.

These industry developments create several possible future directions for Cosmic Trajectories and its pilot project, Trajecta. The original ambition of Cosmic Trajectories was to become a tool that could help organize client portfolio analysis inside an RIA data ecosystem, reducing the need to pull information from separate software systems into workbooks before meaningful analysis could begin. Trajecta could continue moving in that direction by becoming an RIA-facing platform that helps advisors manage client portfolios, organize financial data, and eventually build a clearer book-of-business structure and compete with the likes of Nevis. At the same time, the project could also develop into a standalone tool for retail investors who want to better understand their own portfolios, financial decisions, and long-term investment outcomes.

The most forward-looking possibility would be to eventually incorporate an AI assistant that helps retail investors in real time as they build and evaluate portfolios. Such an assistant could draw from modern portfolio theory and endowment-style investing to help users think through diversification, risk, asset allocation, and long-term portfolio construction. This is not yet a completed feature, but it is a natural future possibility for a project already built around the connection between financial analysis, user data, and the practical problem of making investment information easier to understand. All while carving out an industry niche that has been historically too costly for RIAs to provide services for: the working-class investor.

Appendix:

Links to Project Outputs:

Link to Cosmic Trajectories Website: https://martinezjoseph0517.github.io/cosmictrajectories/

Link to Shiny Application (Trajecta): https://martinez2k18.shinyapps.io/Trajecta-App-V1/

Links to Products Used:

Neon: https://neon.com/

pgAdmin4: https://www.pgadmin.org/download/pgadmin-4-windows/

RStudio: https://posit.co/downloads

Shiny Applications: https://www.shinyapps.io/

Final Schema:

```
-- ================================================
-- Trajecta — PostgreSQL schema
-- ------------------------------------------------------------------------
CREATE EXTENSION IF NOT EXISTS pgcrypto;
CREATE EXTENSION IF NOT EXISTS citext;

-- ------------------------------------------------------------------------
-- USERS
```

```
-- ------------------------------------------------------------------------
CREATE TABLE IF NOT EXISTS users (
   user_id       BIGSERIAL PRIMARY KEY,
   email         CITEXT      NOT NULL UNIQUE,
   password_hash  TEXT        NOT NULL,
   is_active      BOOLEAN     NOT NULL DEFAULT TRUE,
   created_at     TIMESTAMPTZ NOT NULL DEFAULT NOW(),
   updated_at     TIMESTAMPTZ NOT NULL DEFAULT NOW(),
   last_login_at  TIMESTAMPTZ,
   CONSTRAINT users_email_format
      CHECK (email ~* '^[^@[:space:]]+@[^@[:space:]]+\.[^@[:space:]]+$')
);

-- ------------------------------------------------------------------------
-- USER SESSIONS (optional; useful for "remember me" / audit trails)
-- ------------------------------------------------------------------------
CREATE TABLE IF NOT EXISTS user_sessions (
   session_id   UUID        PRIMARY KEY DEFAULT gen_random_uuid(),
   user_id      BIGINT      NOT NULL REFERENCES users(user_id) ON DELETE
CASCADE,
   created_at   TIMESTAMPTZ NOT NULL DEFAULT NOW(),
   expires_at   TIMESTAMPTZ NOT NULL,
   revoked_at   TIMESTAMPTZ,
   user_agent   TEXT,
   ip_address   INET
);
CREATE INDEX IF NOT EXISTS idx_user_sessions_user_id   ON user_sessions(user_id);
CREATE INDEX IF NOT EXISTS idx_user_sessions_expires   ON user_sessions(expires_at);

-- ------------------------------------------------------------------------
-- PASSWORD RESET TOKENS
-- Only the *hash* of the token is stored; the raw token is shown to the user
-- exactly once (in the UI/email) and never persisted in plaintext.
-- ------------------------------------------------------------------------
CREATE TABLE IF NOT EXISTS password_resets (
   reset_id     BIGSERIAL PRIMARY KEY,
   user_id      BIGINT      NOT NULL REFERENCES users(user_id) ON DELETE
CASCADE,
   token_hash   TEXT        NOT NULL UNIQUE,
   created_at   TIMESTAMPTZ NOT NULL DEFAULT NOW(),
   expires_at   TIMESTAMPTZ NOT NULL,
   used_at      TIMESTAMPTZ
);
CREATE INDEX IF NOT EXISTS idx_password_resets_user    ON password_resets(user_id);
CREATE INDEX IF NOT EXISTS idx_password_resets_active
   ON password_resets(user_id, expires_at) WHERE used_at IS NULL;
```

```
-- ------------------------------------------------------------------------
-- APPLICATION_DATA — flexible per-user UI / state bag
-- One row per (user, data_name).  data_content is JSONB so we can store
-- arbitrary settings, the most-recent portfolio name, the open tab, etc.
-- ------------------------------------------------------------------------
CREATE TABLE IF NOT EXISTS application_data (
   data_id      BIGSERIAL PRIMARY KEY,
   user_id      BIGINT      NOT NULL REFERENCES users(user_id) ON DELETE
CASCADE,
   data_name    TEXT        NOT NULL,
   data_content  JSONB       NOT NULL DEFAULT '{}'::jsonb,
   created_at    TIMESTAMPTZ NOT NULL DEFAULT NOW(),
   updated_at    TIMESTAMPTZ NOT NULL DEFAULT NOW(),
   UNIQUE (user_id, data_name)
);
CREATE INDEX IF NOT EXISTS idx_application_data_user   ON application_data(user_id);

-- ------------------------------------------------------------------------
-- PORTFOLIOS (normalized—one row per saved portfolio name)
-- ------------------------------------------------------------------------
CREATE TABLE IF NOT EXISTS portfolios (
   portfolio_id    BIGSERIAL PRIMARY KEY,
   user_id        BIGINT      NOT NULL REFERENCES users(user_id) ON DELETE
CASCADE,
   portfolio_name  TEXT        NOT NULL,
   created_at      TIMESTAMPTZ NOT NULL DEFAULT NOW(),
   updated_at      TIMESTAMPTZ NOT NULL DEFAULT NOW(),
   UNIQUE (user_id, portfolio_name)
);
CREATE INDEX IF NOT EXISTS idx_portfolios_user        ON portfolios(user_id);

-- ------------------------------------------------------------------------
-- HOLDINGS — one row per (portfolio, ticker)
-- ------------------------------------------------------------------------
CREATE TABLE IF NOT EXISTS holdings (
   holding_id      BIGSERIAL PRIMARY KEY,
   portfolio_id    BIGINT       NOT NULL REFERENCES portfolios(portfolio_id) ON
DELETE CASCADE,
   ticker          TEXT         NOT NULL,
   shares          NUMERIC(20,6) NOT NULL CHECK (shares       >= 0),
   purchase_price  NUMERIC(20,6) NOT NULL CHECK (purchase_price >= 0),
   purchase_date   DATE         NOT NULL DEFAULT CURRENT_DATE,
   created_at      TIMESTAMPTZ  NOT NULL DEFAULT NOW(),
   updated_at      TIMESTAMPTZ  NOT NULL DEFAULT NOW(),
   UNIQUE (portfolio_id, ticker),
```

```
    CONSTRAINT holdings_purchase_date_not_future
        CHECK (purchase_date <= CURRENT_DATE)
);
CREATE INDEX IF NOT EXISTS idx_holdings_portfolio      ON holdings(portfolio_id);
CREATE INDEX IF NOT EXISTS idx_holdings_purchase_date  ON
holdings(purchase_date);

-- ------------------------------------------------------------------------
-- updated_at maintenance trigger (shared)
-- ------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION trg_set_updated_at()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = NOW();
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DO $$
DECLARE
    t TEXT;
BEGIN
    FOREACH t IN ARRAY ARRAY['users','application_data','portfolios','holdings'] LOOP
        EXECUTE format('
            DROP TRIGGER IF EXISTS set_updated_at ON %I;
            CREATE TRIGGER set_updated_at
                BEFORE UPDATE ON %I
                FOR EACH ROW EXECUTE FUNCTION trg_set_updated_at();', t, t);
    END LOOP;
END$$;

-- ------------------------------------------------------------------------
-- Convenience view: most-recently updated portfolio per user
-- ------------------------------------------------------------------------
CREATE OR REPLACE VIEW v_latest_portfolio AS
SELECT DISTINCT ON (user_id)
    user_id, portfolio_id, portfolio_name, updated_at
FROM portfolios
ORDER BY user_id, updated_at DESC;

GRANT SELECT, INSERT, UPDATE, DELETE
  ON ALL TABLES    IN SCHEMA public TO trajecta_app;
GRANT USAGE, SELECT
  ON ALL SEQUENCES IN SCHEMA public TO trajecta_app;
```

A.I. Disclosure:

Artificial intelligence (AI) software is used in the development of Cosmic Trajectories, including, but not limited to, debugging code, drafting marketing material, and brainstorming. All final product art will be manmade. Claude, in particular, has played a role in this project primarily as a development assistant rather than as an embedded feature of the application itself. Both Claude and ChatGPT were used to develop user authentication systems, PostgreSQL database schemas, cloud database integrations, and financial analytics features while accelerating troubleshooting and interface development. Additionally, ChatGPT was consulted to brainstorm branding and design by assisting with logo concepts, website themes, and user-interface ideas that aligned with the project's orbital fintech identity.

References

Anthropic. (2025). Code assistant for drafted scripts, bug detection, and debugging, Sonnet 4.6 & Opus 4.7 [Generative AI chat]. Desktop application.

Basole, R. C., & Patel, S. S. (2018). Transformation through unbundling: Visualizing the global FinTech ecosystem. Service Science, 10(4), 379–396. https://doi.org/10.1287/serv.2018.0210

Golden Door Asset. (2026, February 27). Navigating the RIA technology stack: Why firms average 8 vendors and how to optimize your ecosystem. https://www.goldendoorasset.com/benchmark/content/navigating-the-ria-technology-stack-why-firms-average-8-vendors-and-how-to-optim

Iannarone, N. G. (2018). Computer as confidant: Digital investment advice and the fiduciary standard. Chicago-Kent Law Review, 93(1), 141–163. https://scholarship.kentlaw.iit.edu/cklawreview/vol93/iss1/5/

International Organization of Securities Commissions. (2021, September 7). The use of artificial intelligence and machine learning by market intermediaries and asset managers: Final report (FR06/2021). https://www.iosco.org/library/pubdocs/pdf/IOSCOPD684.pdf

Moran, G. (2017). The SEC's data dilemma: Addressing a modern problem by encouraging innovation, responsibility, and fairness. Nebraska Law Review, 96(2), 446–509. https://digitalcommons.unl.edu/nlr/vol96/iss2/9

Neon. (n.d.). Neon serverless Postgres: Ship faster. Retrieved May 11, 2026, from https://neon.com/

Nevis. (n.d.). Nevis: The AI platform for wealth management. Retrieved May 11, 2026, from https://www.neviswealth.com/

OpenAI. (2025). Schema, prompt, and branding optimization, ChatGPT [Generative AI chat]. https://chatgpt.com/g/g-p-69add56011bc819198a7ccba83f9474e-cosmic-trajectories/project

Securities Industry and Financial Markets Association, & SIFMA Asset Management Group. (2025, October 15). Modernizing communications and record retention rules for broker-dealers, investment advisers, and security-based swap dealers.

https://www.sifma.org/advocacy/letters/modernizing-communications-and-record-retention-rules-for-broker-dealersinvestment-advisers-and-security-based-swap-dealers-sifma-and-sifma-amg

Financial Industry Regulatory Authority. (2025, December). 2026 FINRA annual regulatory oversight report. https://www.finra.org/sites/default/files/2025-12/2026-annual-regulatory-oversight-report.pdf

Office of the Federal Register, & U.S. Government Publishing Office. (n.d.). Title 17—Commodity and securities exchanges. Electronic Code of Federal Regulations. Retrieved May 11, 2026, from https://www.ecfr.gov/current/title-17

Tiwari, S., Ramampiaro, H., & Langseth, H. (2021). Machine learning in financial market surveillance: A survey. IEEE Access, 9, 159734–159754. https://doi.org/10.1109/ACCESS.2021.3130843

U.S. Securities and Exchange Commission. (2024, October 21). Examination priorities: Fiscal year 2025. https://www.sec.gov/files/2025-exam-priorities.pdf