ChatGPT

Roadmap for T2.2: Visualization, Interaction & Analysis of Educational Data

Introduction

T2.2 of the GAIMHE project is dedicated to developing tools for **visualizing, interacting with, and analyzing large-scale educational data**. This involves both robust statistical analysis methods and an intuitive visualization platform. The ultimate goal is to enable **researchers, teachers, and developers** to explore and debug student learning traces from adaptive learning platforms (e.g. Adaptiv'Math) in order to gain actionable insights. In practice, this means creating libraries and software to navigate big datasets of student interactions and to visualize various data types – from individual exercise content and student responses to whole curriculum graphs and student progression paths [1]. Key statistical techniques (like dynamic student ability estimation and exercise difficulty modeling with ELO) will be employed alongside modern web technologies to turn raw data into meaningful stories for stakeholders [2].

To structure this roadmap, we break it into two main components:

1. **Statistical and Analytical Methods** – the data analysis techniques and metrics we will implement, including best practices and pitfalls in learning analytics.
2. **Visualization Platform** – the front-end and back-end design of the toolset, including interactive UI components (curriculum graph, trajectory explorer, dashboards, etc.) and considerations for usability and performance.

Each section below is organized with clear subtopics and examples (especially from EdTech and adaptive learning contexts) to illustrate how these methods and tools come together.

Statistical and Analytical Methods

This component outlines **what analyses and metrics** our tools will provide to summarize and probe the educational data. We emphasize established best practices in learning analytics, note common pitfalls, and describe relevant statistical techniques. The focus is on understanding student behavior and learning outcomes through data. Key analytical areas include descriptive metrics of learning progression, sequence and trajectory analysis, group comparisons, content usage patterns, difficulty estimation models, and exploratory discovery of bottlenecks or anomalies. Where possible, we include examples from educational technology research to ground these ideas.

Descriptive Analytics: Student Progression, Mastery & Retention

Descriptive analytics provide core insights into *what* has happened in the learning data. We will implement metrics and visualizations to track student progress over time, their mastery of skills, and their retention or dropout rates. Some focal techniques and considerations:

- **Student Progression Metrics:** Track how students advance through the curriculum or content. This can include counts of modules completed, time spent, or achievement levels reached by

each student or on average. Trend lines or progress distributions help identify if students are moving at the expected pace or getting stuck. It's important to present these in an intuitive way (e.g. a simple timeline of completed topics or a progress index over weeks).

- **Mastery Curves:** Use *learning curves* to visualize how student performance on a skill improves with practice. A typical learning curve plots the error rate or time to complete versus the practice opportunity count, often showing a decreasing trend as mastery develops [3]. For example, as a student practices a particular math skill, we expect to see fewer errors or hints needed – a mastery curve can highlight this. Best practice is to use smoothing or appropriate scales (logarithmic x-axis for attempt count, etc.) to clearly show the diminishing returns or plateaus in learning. Mastery curves can be per-student or aggregated per skill. They help in identifying if additional practice yields improvement or if a student has likely reached mastery.

- **Survival Analysis for Retention:** Borrowing from biostatistics, *survival analysis* examines the time until a certain event – here, typically student dropout or disengagement. In an EdTech context, we treat "dropping out" (e.g. stopping usage of the platform before finishing the curriculum) as the event of interest. Survival analysis techniques like Kaplan-Meier curves or Cox proportional hazards can model the probability of a student "surviving" (remaining active) past a certain lesson or time point [4]. This accounts for **censored data** (students who have not dropped out by the end of observation) [4], which traditional logistic regression would ignore. Implementing a survival analysis allows us to answer not just *who* is likely to disengage, but *when* this tends to happen [5] [4]. For instance, we might find that after 4 weeks or after failing 3 exercises in a row, dropout probability sharply increases – a signal for intervention.

When performing descriptive analytics, we must be mindful of **common pitfalls**. One pitfall is focusing on aggregate trends and missing individual variability – our tools should allow drilling down from an average curve to individual student traces. Another pitfall is **limiting the scope** to historical data without linking to future outcomes [6]. For example, describing that "80% of students mastered skill X by week 8" is useful, but even better is connecting it to what that implies (perhaps those who mastered earlier performed better later). We should also avoid **survivor bias** – only looking at students who completed the program and ignoring those who dropped out [7]. Valuable insights come from analyzing the ones who struggled or left, not just the successful learners. Additionally, mixing **qualitative context** with quantitative data leads to better interpretation; for example, a dip in progression might be explained by a school vacation (context a teacher could provide) – purely quantitative dashboards might mislead if taken alone [8]. Ensuring our descriptive analytics are contextualized, easily interpretable, and not overly cluttered will be key.

Learning Trajectories and Path Clustering

Beyond static summaries, we need to analyze **sequential data** – the paths students take through exercises and content. Each student's interaction log forms a trajectory (a sequence of states: e.g. which exercise they attempted, whether they succeeded, what they did next, etc.). By reconstructing these trajectories, we can answer questions like: *What common paths do students follow? Where do they diverge? Are there distinct strategies or missteps?*

Our plan is to apply methods for **trajectory reconstruction and clustering** of student paths:

- First, represent each student's path through the content. For an intelligent tutoring system, this could be the sequence of exercise nodes traversed in the curriculum graph along with outcomes (pass/fail, hints used). We might enrich these sequences by aggregating certain steps (e.g. treat a cluster of retries on the same exercise as one state "struggled on Exercise 5" to simplify the

sequence). This kind of state abstraction prevents the path analysis from becoming too fine-grained.

- Next, use clustering or visualization to find patterns across these sequences. One approach is to model the sequence data as a **state transition graph** and then depict flows through it. For example, **PathViewer** (a tool by Wang et al.) visualizes the intermediate steps students take in solving problems by constructing a graph of states and using a Sankey diagram to show how students *flow* from one state to another [9] [10]. Each node in the graph is a distinct state (a particular solution attempt or concept), and each directed edge is a transition taken by some students. The width of the edge represents how many students followed that transition, immediately highlighting the most common pathways [10]. This visual flow technique allows clustering implicitly – the dominant paths emerge as thick flows, while less common paths are thinner [11].

- We will incorporate similar **flow visualization and clustering**. By converting raw event logs into a state-flow representation, we can identify prototypical trajectories. For instance, we might discover a cluster of students who all follow a "remedial loop" – bouncing between a foundational exercise and its hints – versus another cluster that speeds through to advanced content. In PathViewer's case, the Sankey visualization revealed that many students did **not** progress linearly but looped back and forth between states when struggling with a coding problem [12]. This loop detection is crucial: it immediately flags *learning bottlenecks* where students are stuck cycling without making forward progress. Our analysis will similarly look for loops or long detours in learning paths, as these often indicate confusion or poor guidance.

- We can also apply more formal clustering: e.g., treating each student's path as a sequence and using sequence clustering algorithms (like Levenshtein distance or dynamic time warping between sequences, followed by k-medoid clustering). Another technique is to identify key pivot points or sub-sequences (analogous to n-grams in text) and cluster students by those features. The **n-gram approach** in PathViewer, for example, lets an analyst focus on sequences of a fixed length to see common short patterns (like a 3-step loop many students execute) [13] [14]. We will adopt a similar capability, enabling analysts to zoom into sequence patterns of interest (e.g., "show me the most frequent sequence of 3 actions students take after getting an answer wrong").

Best practices here include being careful with **state definition** (too granular leads to spaghetti paths; too coarse might hide important differences) and providing interactive filtering. We plan to allow filtering by metadata (for example, view trajectories only for students who eventually mastered a skill vs. those who didn't, or compare trajectories of different classrooms). This helps in identifying if, say, one class's students all follow a suboptimal path due to a teacher's assignment order, whereas another class has more varied and maybe more effective paths.

One pitfall is the sheer complexity of sequence data – visual overload can happen if we try to show all paths at once. Our roadmap includes **interactive visual analytics** to mitigate this: for instance, an interface where the user can select a particular branch in a Sankey diagram to highlight that subset of students, or click on a cluster to drill down to individual student examples. Human-in-the-loop pattern finding is important because algorithms might cluster paths that are statistically similar but a teacher might notice a small difference in one path that is pedagogically significant. Providing the means to inspect and compare specific student pathways is thus crucial.

As an example of applying trajectory analysis, consider identifying **misconception patterns**: We might find a subset of students who all bounce between a fraction addition exercise and a multiplication

exercise – indicating they might be conflating the two concepts. By spotting this trajectory and grouping those students, an instructor can intervene with a targeted clarification. Research shows that such visual mining of student paths can uncover unexpected strategies or misconceptions [15] that would be hard to detect from summary statistics alone.

Comparisons Across Classrooms and Cohorts

Educational data is often naturally grouped (by classroom, school, demographic cohort, etc.). Our tools will support **comparative analytics** to examine group effects – for example, how does one class's performance differ from another's? Do students from different socioeconomic status (SES) backgrounds show different usage patterns or outcomes in the platform?

Key methods to implement for group comparisons include:

- **Aggregated Group Metrics:** For any metric computed (progress, mastery level, average attempts per question, etc.), allow slicing by group. A teacher could compare her current class's mastery curve to last year's class. Or researchers could compare urban vs. rural schools in terms of engagement time. Visualizations like side-by-side histograms or comparative line charts (with confidence intervals) can show these differences clearly. If data permits, statistical tests or effect size calculations (t-tests, ANOVA, Cohen's d) can be provided to quantify differences between groups.

- **Classroom Dashboards:** Each teacher (or school) should have a dashboard summarizing their class and enabling comparison to a broader baseline. For example, a dashboard might show that *"Class A's median skill mastery is 70% which is slightly below the district average of 75%"*, or *"Your class has 5 students at risk of falling behind compared to 2 in the average class"*. This contextualization helps interpret data in a meaningful way. We will incorporate benchmarking features so that group data isn't viewed in isolation.

- **Cohort Trend Analysis:** Looking at year-over-year or cohort progress. This is useful for researchers or administrators. For instance, did the curriculum change introduced in 2025 result in improved outcomes compared to the 2024 cohort? This could be analyzed via **cohort survival curves** (to see if retention improved) or overlaying mastery progress charts of different cohorts. Cohort analysis is a powerful way to prove if interventions are making a difference by providing a built-in comparison group (earlier cohorts as control) [16] .

When doing group comparisons, it's important to follow best practices of **fair analysis**. Differences observed might be due to underlying factors (confounders). A known pitfall is **selection bias** – e.g., one class might have more struggling students to begin with [17], so their lower performance on Adaptiv'Math doesn't necessarily mean the teacher or method was worse; it might reflect initial placement. Wherever possible, our tools will allow analysts to control for prior knowledge or pre-test scores when comparing outcomes (if such data is available). Even simply providing rich filtering (compare classes with similar average pre-test, or filter to a subset of content both classes covered) can help avoid drawing false conclusions.

Another caution is **confirmation bias** – the risk that an analyst might cherry-pick comparisons that confirm their belief (e.g., "I expect boys do better on this topic than girls" and only looking for evidence of that) [18] . To counteract this, the UI will encourage exploring multiple breakdowns and perhaps even highlight comparisons that *might* be of interest (for instance, if there's a notable gap in performance by

SES or gender, the system could flag that neutrally for further investigation, rather than the user only checking what they expect).

Despite these concerns, group analysis can yield actionable insights. For example, we might discover **classroom effects**: if one teacher's class consistently outperforms others on certain skills, we can investigate that teacher's approach (perhaps their use of the "playlist" feature is particularly effective – more on that next). Or we could identify equity issues, such as students from a certain demographic group having less practice time on the platform, prompting us to delve into causes (access issues at home, etc.). The aim is to equip educators and researchers with comparative lenses in a responsible way – shining light on patterns while reminding them to seek causes beyond the data.

Playlist Usage and Pedagogical Strategies

Adaptiv'Math (and similar EdTech systems) often allow creation of **playlists** or sequences of exercises, either algorithmically generated or manually curated by teachers. These playlists reflect pedagogical strategies – e.g., a teacher might assign a remedial playlist for struggling students or an enrichment playlist for advanced learners. Analyzing how playlists are used and their impact is a key component of our roadmap. We will develop features for **playlist usage analysis and pedagogical strategy evaluation**:

- **Usage Analytics:** We'll track which playlists are used, how often, and by whom. For example, if the platform offers preset playlists (like "Fractions Basics" or "Challenge Problems on Algebra"), we can report how many students or classes engaged with each playlist, and completion rates. If teachers can create custom playlists, our tool will help them see *"which playlist did I assign most, and did students finish them?"*. A simple metric is a **funnel**: of those who start a playlist, what percentage complete it? If we visualize this as a step-by-step drop-off chart, teachers might notice, for instance, that many students abandon a certain playlist halfway – perhaps indicating it's too long or difficult.

- **Flow Visualizations of Paths:** A playlist is essentially a mini-curriculum, so we can apply flow diagrams here too. For example, if a playlist has optional branching (or if students can navigate non-linearly), a Sankey diagram could show how students flow through the playlist tasks. Even in a fixed sequence playlist, a Sankey diagram can be useful by showing where students deviated (e.g., skipped an exercise, or repeated it). **Sankey diagrams have been recognized as powerful for visualizing student pathways through courses or content** [19]. They highlight the *sequences and how effectively students navigate them* [20]. In our context, a Sankey for a playlist might highlight, say, that **70% of students** smoothly go from Exercise 1 -> 2 -> 3, but a significant branch of **30%** repeat Exercise 2 twice before moving on (a potential bottleneck). Such visualizations turn raw log data into an immediate picture of the playlist's effectiveness.

- **Pedagogical Strategy Comparison:** We aim to help answer questions like *"What strategies yield better learning outcomes?"*. For example, consider two teachers: one always gives a review playlist before an exam, another uses only new material. If we have outcome data (exam scores, or mastery gains), we can compare the strategies. This might involve a form of A/B testing analysis or at least observational comparison controlled for student level. Our tool can't automatically run experiments, but it can present data to facilitate comparisons. We might show that classes which used the "review playlist" had a 10% higher mastery gain on average than those that didn't, prompting further investigation. However, we must caution that correlation != causation; the goal is to generate hypotheses for educators to consider, not definitive judgments.

- **Content Sequence Analysis:** We will analyze not just *if* a playlist was used, but *how* it was used. Did teachers modify it? Did students take it in the intended order? Did some skip certain items? For adaptive sequences generated by the platform, we can study their adaptivity: e.g., "Adaptive Playlist A adjusts to student level; do students following it have less need for teacher intervention than those on a fixed sequence?" These analyses combine logs and outcomes. Techniques like **sequence mining** can find frequent patterns of content usage. For example, using association rule mining on sequence data might reveal *"Students who struggled in topic X often were assigned playlist Y next."* If that pattern correlates with improvement, it's a potentially effective strategy.

A best practice in analyzing pedagogical strategies is to involve the educators themselves. Our tools should allow teachers to annotate or tag their playlists with a goal (e.g., "reinforcement practice" vs "extension activity"). This metadata can then be used to compare strategy categories. Perhaps "reinforcement" playlists are generally more effective at improving quiz scores than "extension" ones, except for top quartile students. These are the kind of nuanced insights we want to extract.

One common pitfall is **over-interpreting small sample sizes** – a particular teacher's innovative playlist might show great results in one class, but with n=20 students it could be luck of the draw. We will provide indications of sample size and variability (confidence intervals, or at least a note like "based on only 20 students") so users gauge reliability. Another issue is that playlists rarely operate in isolation (a teacher who uses them might also have other strengths), so again, caution with attribution. Despite these challenges, shedding light on playlist usage is valuable. It helps validate the design of these sequences and can inspire knowledge sharing: if data shows *Playlist Z* consistently leads to strong engagement, it could be promoted as a best practice for all teachers. Conversely, if a popular playlist has poor completion rates, perhaps it needs redesign.

In terms of visualization for this analysis, aside from Sankey flows, we will incorporate comparison charts. For instance, a **bar chart of average score improvement** for classes that used each type of playlist strategy can be made, with error bars. Also, timeline views might show *when* playlists are assigned during the school year and if timing influences efficacy (maybe playlists given right before holidays see low completion). The platform's flexibility will let analysts pose these kinds of exploratory questions easily.

Difficulty Modeling (ELO, IRT, and Calibration)

Modeling exercise difficulty and student ability is at the core of adaptive learning analytics. In GAIMHE, a specific aim is to develop dynamic evaluation methods for student level and item difficulty, notably using ELO-type algorithms [21]. Our roadmap includes implementing and experimenting with such difficulty modeling, as well as classical approaches like Item Response Theory (IRT). The insights from these models will feed both into adaptation (in the product) and into analysis dashboards (for researchers to understand content quality and student growth).

Key points and steps in this area:

- **ELO Rating Algorithm for Difficulty & Ability:** The ELO rating system, originally developed for chess ranking, can be adapted to education. In our context, each student and each exercise (or question) can have a rating; when a student attempts an exercise, their ratings update based on whether the student is correct (meaning the student was likely under-rated or the item was easier than thought) or incorrect (student over-rated or item harder than estimated). Over many interactions, the ratings converge to represent student proficiency and item difficulty. **Research has shown that the ELO algorithm provides accurate and stable estimates of task difficulty**

and learner ability in online learning environments, often outperforming simpler metrics like proportion-correct [22]. It works especially well because it updates on-the-fly with each attempt and does not require a fixed test setting [23] [22]. We will implement an ELO-based model and include tools for researchers to adjust its parameters (such as the K-factor, which controls how fast ratings change) and to monitor its calibration. For example, a dashboard might display the ELO-derived difficulty of each exercise along with confidence intervals, and highlight exercises whose difficulty rating is still uncertain (perhaps due to low data).

- **IRT (Item Response Theory):** IRT is a more theory-driven approach (e.g., the Rasch model or 2PL model) that estimates item difficulty and discrimination, and person ability, by fitting a logistic model to response data. IRT is powerful in that it provides interpretable parameters and a solid theoretical foundation. However, it traditionally assumes student ability is static during the test and often requires batch processing of data (not real-time updating). In an adaptive platform where learning happens over time with multiple attempts, standard IRT can be challenging to apply. One known issue is that **IRT models can be computationally heavy for large datasets and frequent updates**, making them less practical for real-time adaptation [24]. We will likely use IRT in an offline analysis mode: e.g., periodically fit an IRT model to accumulated data to validate our ELO model or to conduct post-hoc analysis of content. This could reveal if certain exercises have poor discrimination (i.e., they don't actually differentiate high vs low ability students as expected) which might suggest those items need revision. Our roadmap includes comparing the IRT results to ELO results: if they broadly agree, that boosts confidence in using the simpler ELO online; if they diverge for some items, those are worth a closer look.

- **Calibration and Drift:** A critical part of difficulty modeling is checking **calibration over time**. Content difficulty might change (for example, if hints are added to an exercise, it effectively becomes easier). Students also evolve. We'll incorporate analytics to detect *calibration mismatches* – cases where the model's predicted performance doesn't match actual data. For instance, if an exercise is rated as "easy" but a majority of students are getting it wrong, that's a mismatch. It could indicate that the content is misleading or that the student population has a specific gap. Our tool will flag such anomalies. One concrete method: plot the expected success probability (from the model) vs. the observed success rate for each item over recent attempts. Ideally, these align; significant deviation means the model might need to adjust or the item needs review. This ties into **bottleneck detection** as well – an overestimated easy item that students fail is certainly a bottleneck.

- **Dynamic Difficulty (Personalized):** In adaptive systems, difficulty is not one-size-fits-all; an item might be easy for an advanced student and hard for a novice. Beyond global difficulty, we can analyze *personalized difficulty metrics*. For example, we might use the ELO model to trace each student's ability over time as they learn (this is akin to a **learning trajectory in ability space**). Graphing a student's ELO rating progression can show their learning curve in a single metric – ideally a rising trend as they master more content. Comparing those trajectories between students or groups can yield insights (like detecting if some students' ability scores stagnate, meaning they aren't benefiting from the system).

- **Pitfalls and Best Practices:** A known challenge in difficulty modeling is the **cold start problem** – new exercises or new students have no data, so initial difficulty or ability estimates are uncertain. We plan to implement sensible defaults (e.g., starting every new item at a medium difficulty rating and then letting data refine it) and show the uncertainty. Also, if using ELO, choosing the right K-factor (the rate of update) is important: too high and estimates jump around with noise, too low and the model is slow to adapt. Recent research even looks at **dynamic K-values** to

balance stability vs. flexibility [25]. We will consult best practices and perhaps allow an adaptive K (e.g., higher K initially, lower K as confidence grows).

Another issue is **multi-dimensional ability** – a student isn't one number if the content covers different skills. In the long run, we might extend to a vector of skill proficiencies (like a separate ELO per skill tag, or a full knowledge tracing model). Initially, a unidimensional model is simpler to implement and explain. We will however design the data pipeline such that extending to skill-specific difficulty is possible (for example, each exercise is tagged with a skill, and we maintain separate difficulty ratings per skill domain).

From a **learning analytics best practice** perspective, difficulty models should be validated. We will use holdout data or retrospective analyses to ensure our difficulty estimates actually correlate with student outcomes. If we say an item is of difficulty 750 (ELO) and a student's rating is 700, the expected success chance is ~0.35 (depending on formula); we can check if in reality 35% succeed. Such validation will be part of the research outputs and can be presented in the interface for transparency.

On the **user interface** side (detailed more in the next section), presenting difficulty info must be done carefully. For teachers, we might display difficulty in relative terms (e.g., "This exercise is rated 4/5 stars in difficulty" or "Difficulty: Hard" based on threshold) rather than raw ELO numbers. For researchers, we can present the numeric values and even allow export of the data for external analysis. We will also include visualizations like **difficulty-progress plots**, where one axis is item difficulty and the other is student proficiency – points on or off the diagonal indicate overchallenge or underchallenge zones. This can guide content adjustments (e.g., if many points lie above the diagonal, it means many students are being given items above their ability – maybe the sequencing is pushing too fast).

In summary, our roadmap in difficulty modeling is to implement a **robust, real-time ELO-based system** for ongoing difficulty estimation (with allowances for later multi-skill expansion), complement it with periodic **IRT analyses** for deeper insight, and integrate the results into both the adaptation logic and the analytics dashboards. This will help answer exploratory questions such as: *Which exercises are too easy or too hard? How reliable is the difficulty estimate for each content piece? Are students appropriately challenged?* Addressing these questions ensures the educational platform remains effective and well-calibrated to learners' needs.

Exploratory Insights: Bottlenecks, Mismatches & Effective Paths

This final analytical category is more open-ended: using the tools to discover **emergent patterns, anomalies, and answer "why" questions** about the learning process. Given the rich dataset of student traces, we want to empower users to explore for insights like identifying learning bottlenecks, detecting calibration mismatches (as touched on above), and finding which learning paths are most effective. Here we outline some of these exploratory analyses and how we'll support them:

- **Learning Bottlenecks:** A bottleneck is a concept or exercise where a large portion of students get stuck or progress unusually slowly. To find bottlenecks, we can combine several analyses: look for exercises with high failure rates or many repeated attempts, identify states in the trajectory graphs that have a lot of incoming flow but little outgoing flow (many students enter that state and linger or loop) [12], and use teacher feedback (e.g., anecdotally teachers might know "fractions simplification" is hard – our data can validate that). Once identified, bottlenecks should be visualized clearly. For instance, we might highlight the node in the curriculum graph with a red border if it's a bottleneck, or have a "Trouble Spots" report listing the top 5 bottleneck skills. Pinpointing bottlenecks is the first step; the next is helping address them. Our analytics can track if changes are made (say, adding an instructional video to that section) and then later

show if the bottleneck eased (improved metrics). This closes the loop from insight to intervention.

- **Calibration Mismatches:** As discussed under difficulty modeling, a calibration mismatch means something in the system's expectations is off. This could manifest as *too easy or too hard content relative to assumptions*, or *student self-assessment vs actual ability diverging* (if we have survey/confidence data). Another example: if the adaptive system predicted a student had mastered a skill (perhaps by reaching a threshold in practice), but on an external assessment the student failed questions on that skill – that's a calibration problem in the mastery estimation. We plan to incorporate **cross-checks** wherever possible. For example, compare the platform's internal mastery status with an independent test's results, if available. Even without external tests, we can use later performance as a yardstick: if a student "mastered" all grade 4 content but struggles immediately on early grade 5 content, maybe the bar for mastery was set too low. Our tool could generate alerts like *"30% of students flagged as mastered in Fractions still made errors in review questions"*. Internally, we saw a form of calibration issue in PathViewer's study – *some incorrect solutions passed all the system's test cases* [26]. Translating that generally: the system thought the students were correct (tests passed) but in reality their solution was flawed, indicating the assessment criteria missed something. Spotting such cases is gold for platform developers, as it shows where the content or evaluation logic can be improved. We'll enable querying for anomalies (e.g., filter for events where a student success was followed by failure in a conceptually similar task – could indicate luck or a flaw in evaluation).

- **Effective Learning Paths:** One of the most exciting questions is *"What paths lead to the best outcomes?"*. Since Adaptiv'Math is adaptive, students might take different routes through content. Some routes might be more efficient or produce deeper learning. By analyzing the data, we aim to find patterns of success. Concretely, we can take the top X% performing students (however defined – maybe final assessment scores, or fastest mastery) and see what is common in their usage. Do they tend to do optional exercises that others skip? Do they have fewer repeated attempts (indicating they learned it right the first time)? We can compare that to the lower performers' paths. This is essentially **association analysis** between path features and outcomes. Another approach is to use predictive modeling: train a decision tree or sequence model to predict final success from early path choices. For example, such a model might learn that *"Students who revisit foundational skills early on tend to do better later"*, suggesting that an effective strategy is to ensure mastery of basics before advancing. We should present any such findings carefully – likely as guidance rather than hard rules. A visual way to show effective vs less-effective paths is to overlay two flow diagrams or highlight the "successful path" in a distinct color on the trajectory graph. Perhaps the Sankey diagram could be filtered by outcome: one could toggle to view the flow of students who achieved mastery versus those who didn't, and differences would pop out (maybe the non-mastery group had a huge stream going through a particular detour that the masters avoided). This aligns with research where **learner trajectory networks were used to identify prototypical pathways and provided evidence of what content sequences were effective** [27] [28]. Our platform will allow such comparisons so that researchers and designers can glean what sequence of activities yields the best learning gains.

- **Root Cause Exploration:** With bottlenecks or anomalies identified, the tool should support digging deeper. Suppose we find a bottleneck exercise – we'd want to see *why* students are struggling. Are they all making the same mistake (which we could see via the distribution of wrong answers)? The data collected include the wrong answers students gave [29], so in an exercise viewer one might observe that 60% chose option C, a particular misconception. That immediately suggests a fix (address that misconception with feedback). Or maybe the bottleneck is because of a **group factor** – maybe primarily one class or one region's students struggled

(which could indicate a gap in prior knowledge taught). Thus, linking the analyses: the user could click the bottleneck skill and then see a breakdown by class or demographics. Our integrated approach with interactive dashboards will facilitate these multi-dimensional queries (drill-downs).

- **Open-Ended Queries:** We will also provide a way for power users to do custom queries on the data (perhaps via a simplified query builder or a Python notebook interface for advanced researchers). This is because no pre-built dashboard can anticipate every question. For example, a researcher might wonder: *"Is time-of-day of practice related to performance?"* or *"Do students who engage in forum discussions (if any) have better retention?"*. Such questions are beyond the immediate scope, but by making the data accessible (with proper anonymization), we empower deeper research. In the context of a **roadmap**, we acknowledge these possibilities so that our system is built extensibly.

To summarize this section: the platform's analytical tools will not only answer known questions (progress, comparison, etc.) but also enable **discovering new insights**. By visualizing data in interactive ways and combining statistical methods (like survival analysis, clustering, correlation analysis), we anticipate the system will uncover non-obvious patterns. These could lead to improvements in content (fixing bottlenecks), better alignment of difficulty (closing calibration gaps), and sharing of best practices (amplifying effective paths and strategies). The aim is to turn raw learning traces into a form of **educational knowledge base** that continually informs teaching and learning design.

(In practice, we will iterate these analyses with actual data, validating findings with educators to ensure they make pedagogical sense. The roadmap sets the stage for building these capabilities, with the understanding that flexibility and user feedback are key as we refine the tools.)

Visualization Platform

Complementing the analytical methods, the project will develop a **visualization and interaction platform** to make these insights accessible. This platform comprises a front-end application (for interactive exploration and display of data) and a back-end infrastructure (for efficient data querying and computation). The design must be **modular, performant, and interpretable**, catering both to technical users (researchers, data analysts) and non-technical users (teachers, educators). In this section, we outline the technology stack choices and the key UI components, with guidance on design and successful patterns from existing systems.

Frontend and Backend Technology Stack

Our chosen stack for the front end is **Svelte** (a modern reactive web framework) combined with **D3.js** (a powerful data visualization library). On the back end, we plan to use **Python** with **DuckDB** and **Polars** for data handling, operating on Parquet files that store the collected traces. Here's why these choices were made and how they will work together:

- **Svelte (Frontend):** Svelte is a component-based framework that compiles away (no virtual DOM overhead), resulting in very fast and small web apps. This is ideal for potentially complex dashboards that need to run smoothly in a browser. Using Svelte, we can create reactive UI components – for example, a component for a time-series chart that automatically updates when new data is loaded or filters change. Svelte's reactivity and component encapsulation will help us build a **modular interface**: each visualization (like the curriculum graph, or a student profile panel) can be a component that can be developed and tested independently, then

combined in various dashboard pages. Moreover, Svelte's simplicity in managing state and events will be useful for interactions (like brushing and linking between charts, or toggling filters).

- **D3.js (Frontend Visualization):** D3 is the de-facto standard for bespoke web visualizations, giving fine-grained control over DOM or Canvas elements and data-driven rendering. We will leverage D3 for drawing complex visuals such as graphs, Sankey diagrams, and animated transitions. For instance, the **Sankey diagram** we envision for student trajectories or playlist flows can be implemented using D3's Sankey layout utility [11]. We might use existing D3 plugins like **d3-sankey** or **d3-force** (for network layouts) to jump-start development. The combination of Svelte + D3 is powerful: Svelte manages the overall UI state (which dataset is selected, which student is highlighted, etc.) and D3 handles the heavy lifting of rendering shapes and paths based on data. This approach has been demonstrated in tutorials and projects where Svelte provides structure and D3 handles the visualization specifics, yielding interactive charts with relatively little code [30] [31]. One important design decision is to keep visualizations **lightweight and efficient** – we will use canvas or WebGL via D3 for very large data rendering if needed (e.g., plotting thousands of points), but many education datasets (like one class's data) can be handled in SVG with careful optimization.

- **Python Backend with DuckDB and Polars:** For data analysis and serving, Python is convenient given the ecosystem of libraries and the data science expertise in the team. **DuckDB** is an in-process analytical SQL engine that excels at querying columnar data (it's often called "SQLite for analytics"). It can directly query **Parquet files** stored on disk as if they were database tables, with very impressive speed on large datasets [32]. This means we don't need to import all data into memory or a separate database; we can run SQL queries on the raw data efficiently. For example, if the front end needs "average score on skill X by week for class Y," the back end can run a DuckDB SQL query on the relevant Parquet, aggregating on the fly. DuckDB's performance on large files (gigabytes of data) is outstanding – benchmarks show it often outperforms Pandas by orders of magnitude for aggregation and filtering, even beating in-memory DataFrame operations by using smart vectorized execution [33] [32]. This choice ensures that even as our data scales (say to millions of rows of student logs), queries remain snappy, which is crucial for interactive exploration.

Polars is a newer DataFrame library (written in Rust with Python bindings) that provides an API similar to Pandas but with much better performance and parallelism. We plan to use Polars especially for more complex transformations or when Python-side logic is needed. For instance, computing an ELO rating update might not be straightforward in pure SQL; we could use Polars to efficiently group data by student and iterate. Polars also integrates with Apache Arrow memory format, which pairs well with DuckDB (they can exchange data efficiently). In our workflow, DuckDB can handle heavy aggregations and joins (especially ones that benefit from SQL optimization), and Polars can handle tasks where a Pythonic approach is easier. This dual approach is recommended by experts: *use DuckDB to query large files or to act as glue between sources, and use Polars for in-memory computations* [34]. Both can consume Parquet files directly, meaning our back end can remain **stateless** to a degree – simply reading from the Parquet data lake for each query, which avoids having to maintain a separate database state.

- **Serving Data to Frontend:** We will likely implement a lightweight API (possibly using a Python web framework or even DuckDB's REST interface if available). The front end could request data in chunks (for example, ask "give me data for chart X for class Y"), and the back end runs the query and returns JSON or Arrow data. Because of DuckDB/Polars efficiency, this can happen on-demand for many queries. We should still design with caching in mind: if certain queries (like a particular class dashboard) are repeatedly requested, caching those results in memory could

further speed things up. However, given that *"almost everybody's data fits on a laptop"* with modern columnar tools [35], we expect even on-the-fly queries to be quite fast, especially since educational datasets, while large, are not at the petabyte scale. DuckDB running locally can scan tens of millions of records in sub-second time on modest hardware [36].

- **Modularity and Integration:** The entire stack is chosen for modularity. Svelte components can be developed and even packaged as a library of reusable edu-visualization components. On the back end, the use of SQL and DataFrame operations means each analysis (e.g., computing survival curve data, or ELO model outputs) can be written as a standalone function or query, which we can unit test. We might create a library of "analytics endpoints" corresponding to each analytical method in the roadmap. For example, a function `get_mastery_curve(class_id)` might run a DuckDB query to get error rate vs attempt for that class and return the data ready for plotting. This modular approach ensures we can maintain and extend the platform (for instance, adding a new metric later is just a new query and front-end component, not a massive refactor).

- **Scalability:** Using Parquet + DuckDB means we can handle increasing data by partitioning files (by date, by school, etc.) and DuckDB will prune and query only what's needed. If usage grows, we could move to a client-server mode (DuckDB can be embedded in a server process) or even consider DuckDB's multi-threading and upcoming multi-user capabilities. Since this is a roadmap, suffice to say the stack is chosen with an eye to scale *and* ease of development – it lets us start simple (files on disk, local queries) and grow to more distributed scenarios if needed without changing core components.

To ensure the **interface is performant**, we'll leverage these tech strengths: heavy computations done in DuckDB/Polars (so minimal load on the browser), and efficient updates in the UI (Svelte's reactivity means only minimal DOM updates happen when state changes). We'll also consider using web workers for any expensive client-side computations (e.g., rendering a very large visualization off the main thread). As a result, even complex interactive visuals (like dragging a slider to filter date and updating multiple charts) should feel smooth, given the underlying optimizations.

Key User Interface Components

A variety of UI components will be developed to address different analysis needs. Below we describe the key components, including how they function and design considerations for each. We also cite successful visualization designs from existing platforms and literature as inspiration.

- **Curriculum Graph Viewer:** This component visualizes the structure of the curriculum or content graph (data type D2 in GAIMHE). Think of this as an interactive knowledge map: nodes represent exercises or learning units, and edges represent prerequisite or recommended sequences (the "graph of exercises" that defines possible learning paths). The graph viewer will likely display a directed acyclic graph (if prerequisites) or a network if there are cross-links. A good design pattern is a **node-link diagram** with nodes colored by some property (e.g., difficulty or average success rate) and perhaps scaled by number of attempts. Edges can be arrows indicating the flow. We will ensure the graph is **zoomable and pannable**, as these graphs can be large. Users (especially researchers or curriculum designers) can use this to get a high-level overview of the content structure. We will incorporate features like **highlighting**: hovering on a node could highlight its connected prerequisites and dependents, allowing a teacher to see the immediate relationships for a given exercise. Clicking a node might open a detailed panel (the Exercise Viewer, described below). From a design perspective, we might use a force-directed layout or a hierarchical layered layout (if there's a natural ordering like grade levels or modules). Existing

systems like Khan Academy's knowledge map or other adaptive learning systems' content maps serve as inspiration. One study on MOOC course design used *tree visualizations to represent course hierarchical structure and sequences* of modules [37] – similarly, our curriculum graph can be thought of as a tree/dag showing the progression structure. We will ensure this visualization is **interpretable** to non-technical users by possibly offering a simplified view (like collapsing modules into bigger nodes, or switching to a list view for those who prefer it). But for researchers, the full graph with all connections is valuable to analyze curriculum complexity and identify potential choke points (which might align with bottlenecks if many edges converge on one node).

- **Student Trajectory Explorer:** This is an interactive tool to explore individual student paths (data type D3: "parcours des élèves") through the curriculum graph and over time. It ties closely with the trajectory analysis discussed earlier. The UI might have two modes: **per-student timeline** and **aggregated flow**. In per-student mode, a teacher or researcher can select a particular student and see a chronological timeline of their actions: e.g., a sequence of exercise attempts with timestamps, outcomes (perhaps green for correct, red for incorrect), maybe dots or icons on a horizontal timeline. This helps in "debugging" a student's learning process – one can spot if the student was stuck retrying one problem for an hour, or if they skipped around. We will include controls to overlay events like interventions (e.g., teacher gave hint at this point, if that data is available). In aggregated mode, one can switch to view something like the **Sankey diagram of state transitions** for a group of students. As described in the analysis section, this would show the most common pathways taken. For example, a Sankey might show that out of 100 students, 50 followed path A (ex1 → ex2 → ex3), 30 diverged after ex1 to an alternate ex4, etc. The **flows' widths make it easy to see the dominant paths** [11]. We will make this Sankey interactive – clicking on a flow could filter to the subset of students who took that path, and then the UI could enable drilling into their outcomes (maybe show their average score or time). A successful design in research is PathViewer's Sankey-based path visualization, which used width-encoded flows and even an n-gram focus to highlight loops [10] [14]. We plan to implement something similar, using D3's Sankey module. For interpretability: Sankey diagrams can be a bit overwhelming at first, so we'll add legends and maybe tooltips that explain "width represents number of students". In fact, one advantage of Sankey for educators is that it transforms a spreadsheet of sequences into an **intuitive flow picture**, essentially "watching the story unfold" of student movement through content [38] [39]. Our trajectory explorer aims to deliver that "aha" visual moment when, for example, a teacher sees a big fat flow going into a dead-end node – instantly revealing many students are getting stuck in the same way.

- **Classroom & Cohort Dashboard:** This is a composite dashboard intended for teachers (for their class) or administrators (for a cohort of classes). It aggregates key information at a group level in an easily digestible format. Typical components of a classroom dashboard could be: a **summary panel** (e.g., "Class Mastery: 75% average, Range: 50–95%"), a list or chart of **students at a glance** (each student's progress bar or an emoji/status indicating who needs help), and charts of class-wide distribution (like a histogram of number of skills mastered, or a boxplot of time spent). We will design it such that a teacher can quickly identify outliers (who is far behind or ahead) and overall class strengths/weaknesses. A **class mastery timeline** could show how the class's average mastery grew over the semester, perhaps compared to last year's class (if data available). For cohort or school dashboards, similar displays can show comparisons between classes or highlight equity gaps. We take inspiration from mastery-based dashboards like **MasteryTrack**, which provides *"clear displays of student learning progress"* and allows teachers to *"analyze and compare student learning progress over time"* [40] [41]. Our dashboard will incorporate both current snapshot and trend over time, since teachers need to monitor day-to-day but also see growth. Another useful visualization is a **heatmap** of students vs skills: each cell colored by

mastery or recent performance. This lets a teacher spot, for example, that a particular skill column is red for many students (a problematic topic for the class), or a particular student row is red across many topics (an at-risk student). A simplified version of this was popularized by some LMS dashboards where each row is a student and columns are course modules – we can implement a more interactive version (clicking a cell could bring up that student's attempts on that skill).

Design for dual audiences: researchers might want to use the dashboard to compare metrics precisely (so we'll include tables or the ability to download data), whereas teachers want an at-a-glance view. We resolve this by layering detail: by default, show high-level visuals and allow a "details" toggle or hover that shows numbers. For example, a bar chart of mastery by skill might by default just show approximate bars, but hovering a bar shows the exact % mastered and perhaps how that compares to another group. We will also emphasize **actionable visuals** – e.g., use alert icons or colors to draw attention to important things (if a student hasn't logged in for X days, or if the class average on last topic dropped, etc.). The idea is not just to display data but guide the user to where action might be needed.

- **Exercise/Content Viewer:** When a user (especially a teacher or content developer) wants to delve into a specific exercise or piece of content (data type D1: individual exercise data), this component comes into play. It will have two facets: **content rendering** and **analytics**. On one side, we may integrate a simple version of the exercise player – for instance, showing the question text, options, correct answer, maybe the hint or solution. This gives context about what the item is. On the other side (or below it), we present analytics about that exercise. This can include: difficulty rating (from our model) – perhaps shown as "Difficulty: 600 (Medium)" with an explanation; item analysis stats – e.g., *overall success rate: 64%*, *average attempts per student: 1.7*, *discrimination: 0.4* (if using IRT); and **common wrong answers** (for multiple-choice, a bar chart of how many chose A, B, C, etc., or for open response, maybe a word cloud of common incorrect responses). Since the GAIMHE data includes wrong answer distributions [29], we can surface that clearly: e.g., "Most common wrong answer: `0.5` (mistaking 1/2 for 1/4) chosen by 30% of students [29]." This kind of insight is extremely useful for content debugging – it tells the author where misconceptions lie. We can also show **temporal data** for the exercise: if it's been used over months, a line chart could show if it's getting easier (perhaps after a curriculum change) or if a particular cohort struggled more. Another visualization could be the **position in curriculum graph**: highlighting where this exercise sits (prerequisites and post-requisites), which can help explain things (maybe it's hard because a prior skill was weak).

For interpretability, we'll avoid overwhelming the user with statistical jargon. Teachers might not know what "discrimination" means in IRT, so we could hide advanced stats by default or provide tooltips ("discrimination: how well the question differentiates stronger vs weaker students; higher is better"). What teachers will care about is "Is this question hard for my students? Are many making the same mistake? Should I revisit that concept?" – our viewer will aim to answer those. We might implement a simple indicator like "Misconception alert" if a wrong answer is chosen by a large fraction (meaning it's a tempting distractor). For researchers/content teams, the raw stats will be available and exportable.

A successful example to emulate is the item analysis in many LMS or assessment systems which shows option statistics. Our edge is that we tie it with adaptive data: e.g., we can show not just final answers but also if students required hints on that item, or how timing on the item correlates with success. With Polars/DuckDB on the back end, we can answer queries like "for those who got it wrong, what percentage eventually learned the concept?" or "what next exercise did most students do after failing this one?" – essentially connecting item performance with trajectory. Visually, we might show a small Sankey or flow from this exercise to what happens after (did they retry, did they drop out, etc.). If, say,

20% of students who fail this exercise then exit the platform, that's a sign it's frustrating or poorly placed. Representing that as a mini-flow diagram could be very insightful (a bit like a funnel emanating from the exercise node).

- **Playlist Analyzer:** For examining playlists and sequences (whether a teacher-assigned sequence or an automatically recommended path), this component will provide both summary and flow visuals. At a summary level, it can list all playlists (or a selected one) and show usage stats (e.g., assigned to 5 classes, completion rate 60%, average score improvement +5%). We'll include a **step-by-step dropout chart**: a bar for each step in the playlist showing how many students reached that step. This effectively forms a funnel chart – wide at step 1 (everyone starts) narrowing down to the final step (those who finished). If a particular drop between steps is large, that might suggest a problem at that transition (maybe step 3 was too difficult causing many to quit). Additionally, we can show the **performance per step**: perhaps a color on each bar indicating average success at that step.

The more advanced visualization would be a **Sankey diagram or flow chart** showing alternate paths in the playlist. For example, if the playlist is not strictly linear (some students skip or get an alternate exercise via adaptivity), a Sankey will illustrate those branches. Prior work like *CoursePathVis* used Sankey diagrams to visualize how students progress through a curriculum with the ability to group and filter the flows [42]. In our case, the "curriculum" is a short playlist, but the principle is the same: **flows make it easy to see where students go and where they exit**. We might augment this with a "funnel" concept similarly to CoursePathVis [43] [44] – for example, designate a critical exercise as a funnel point and see how the flow looks before and after it. This can reveal if that exercise is a gatekeeper that filters out students.

Another view might be **comparison of multiple playlists**. If teachers have options (say two different playlists teaching the same standard), we could produce a comparative chart. Imagine two side-by-side funnels or a combined chart where each playlist is a line on a plot showing mastery outcomes of students who took it. If one consistently leads to better outcomes, that's worth noting. However, as mentioned, we must be cautious: different playlists might be given to different levels of students. So, our tool could allow filtering by initial student level to make a fair comparison.

For teachers, the playlist analyzer will help answer: "Is the playlist I'm assigning effective? Where do my students struggle in it?" For researchers: "How do different content sequences compare? Are there patterns in usage?" We will draw inspiration from learning design tools that visualize learning pathways. The BERA blog example we cited earlier demonstrated using Sankey for entire program curricula to great effect, highlighting how students navigate and where improvements can be made [20] [45]. We aim to bring that level of insight to micro-curricula (playlists) as well.

- **Teacher/Researcher Mode Toggles:** While not a visualization per se, it's worth mentioning we will likely implement different "views" or modes for different users. A teacher logging in may, for instance, by default see their classroom dashboard and simplified options, whereas a researcher might see a data overview and a menu to dive into advanced analyses. The underlying components are the same, but the navigation and defaults can be customized. We'll also ensure **access control** – e.g., a teacher should only see their students' data, while a researcher might see de-identified aggregate data across the platform. This is crucial for privacy and also for reducing clutter (a teacher doesn't need global stats typically, and a researcher doesn't need individual student names).

- **Interpretability Aids:** Across all components, we plan to include UX features that aid interpretation: concise **textual summaries** above visuals (e.g., "This chart shows the flow of

students through the playlist. The width of each arrow is proportional to number of students [11].”), **tooltips** on hover that translate data into plain language (“20 students (40%) chose this wrong answer.”), and possibly a **guided tour** or tutorial mode for new users. Since one of our goals is serving non-technical users, these touches will help ensure the data stories are understood. In the Medium article about Sankey charts, the author noted that once you understand the basics (nodes, flows, widths), *“it’s a breeze”* to interpret [46]. We want to bring our users to that point of understanding quickly. So any novel visualization we introduce will come with an explanation. This might be done via an info icon or an overlay that can be toggled.

In implementing these UI components, we will make use of various libraries to speed up development: for instance, **D3.js** (and extensions like d3-sankey, d3-scale for color scales, d3-axis for chart axes), possibly **Component libraries** for Svelte (if any suitable ones exist for layouts or basic charts that we can customize), and for certain charts like heatmaps or timelines, we might integrate existing open-source components (e.g., a calendar heatmap for activity over time). The emphasis is on not reinventing the wheel for standard charts, but giving ourselves flexibility to create custom visualizations where needed for educational data nuances.

Design Considerations: Modularity, Performance, and User-Centered Interface

To ensure the platform serves both researchers and teachers, we will adhere to a few guiding principles in the design:

- **Modularity:** As mentioned, each component is modular. This also means the interface can be configured to some extent. A researcher might assemble a custom dashboard with a subset of components (like a research workspace focusing only on trajectories and difficulty analysis), whereas a teacher interface might combine the classroom dashboard with quick links into individual student profiles. Technically, modularity in Svelte/D3 means clear separation of components and using a shared state management where necessary (for example, a store for “current class selected” that multiple components subscribe to). This ensures changes in one part propagate appropriately.

- **Performance:** We anticipate potentially large data on the front end if, say, a researcher pulls in thousands of student traces at once. We will use virtualization for any long lists (only render what's visible), and aggregate data on the server as much as possible before sending. Using DuckDB to pre-aggregate means the browser typically receives data already summarized (e.g., 100 points for a curve, not 100k raw records). D3 can handle a few thousand SVG elements comfortably; beyond that we might switch to canvas. We will test with realistic data sizes and ensure interactions (like hovering or filtering) don’t lag. The backend will exploit multi-threading in DuckDB/Polars to handle concurrent queries (for example, loading two charts at once). If needed, we might implement *pre-fetching*: e.g., when a teacher opens the dashboard, also fetch data for each student in the background so that clicking a student is instantaneous. These are the kinds of optimizations that can make the experience seamless.

- **Interpretability & Simplicity:** The interface should communicate insights, not just data. We will follow **data visualization best practices**: using consistent color schemes (e.g., a particular color for correct vs incorrect, or a color palette for difficulty levels), labeling axes and legends clearly, and avoiding clutter. For a teacher, seeing a complicated graphic without guidance can be a turn-off, so every view will be accompanied by at least a title and subtitle explaining what’s being shown. We will also incorporate some design from **learning analytics dashboards (LAD) research**, which emphasizes that dashboards should drive *action*. For example, if a teacher sees in the dashboard that a certain skill is low for many students, we might include a suggested

action like “Consider reviewing Skill X in class.” While building that in might be phase 2 (beyond initial development), keeping the end-goal in mind will shape our design towards practical utility.

- **Responsive and Accessible Design:** Teachers may access the dashboards on various devices, including tablets or projectors in class. We will ensure the layout is responsive (Svelte makes it fairly easy, and we can design components to resize). Also, we aim to follow accessibility guidelines (color choices with sufficient contrast, support screen readers by providing alt text or aria labels on graphs where possible). Inclusive design will widen the tool’s adoption and is simply good practice (and mentioned in analysis pitfalls – neglecting accessibility skews data use [47]).

- **Inspired by Real Use Cases:** We are looking at examples like the **ION K-12 analytics** (as described in a Medium post) where they integrated Sankey charts in their product because *“educators need more than static data – they need stories, movement, and a clear path to understanding how students are progressing”* [48]. This sentiment guides our UI design: every visualization should tell a story at a glance. We will emulate such success stories – e.g., using color to indicate improvement vs regression on a Sankey (the Medium example mentioned using green for improvement flows and red for regressions [49]). Those small touches can make data immediately actionable (teacher sees red flow = regression, she knows to focus there).

Finally, the platform will support an **iterative design process** with users. We plan to involve a few pilot users (teachers and researchers) to try early prototypes and give feedback. This will ensure the interface is understandable and actually addresses their needs. For example, a teacher might say “I really want to be able to click a student and see where exactly they got stuck, easily,” which could lead us to refine the student trajectory view to be one-click from the class dashboard. Researchers might request the ability to export charts or data – we can include export buttons (CSV downloads or even direct links to open the data in a Jupyter notebook for further analysis).

In summary, the visualization platform is being built with **cutting-edge yet pragmatic technology** (Svelte, D3, DuckDB, Polars) to achieve interactivity and speed. Its UI components cover the full spectrum from high-level overviews (dashboards) to deep dives (trajectory explorer, item viewer). By studying effective designs (like Sankey diagrams for pathways, mastery dashboards, etc.) and prioritizing clarity, we aim to create a tool that not only displays data but truly enables understanding and data-driven decision-making for education. Each user – be it a teacher tweaking their instruction or a researcher testing a hypothesis – should find the interface empowering and insightful.

Conclusion

This roadmap outlined the vision for T2.2 of the GAIMHE project: a comprehensive toolkit for analyzing and visualizing educational data from adaptive learning platforms. On the analytical side, we detailed methods ranging from basic descriptive stats to advanced modeling, emphasizing how to extract meaningful insights (while avoiding pitfalls) about student progression, behavior patterns, group differences, content difficulty, and more. On the visualization side, we proposed an interactive platform using modern web technologies to bring those analyses to life for end-users through intuitive dashboards, graphs, and exploration tools. The combination of **robust analytics** and **user-friendly visualization** is crucial – one without the other would not fully serve the stakeholders. Researchers need the depth and rigor of the analyses, and teachers need the accessibility and intuitiveness of a well-designed interface; our roadmap strives to deliver both in a balanced manner.

By following best practices from learning analytics research and learning from existing EdTech visualizations, we increase the chances of creating a successful system. For instance, using Sankey diagrams to visualize learning paths has proven effective in both research and practical tools [20] [48], and we plan to leverage that. Employing ELO-based difficulty modeling has shown strong results in prior studies [22], giving us confidence in that approach for dynamic assessment. Throughout the roadmap, a recurring theme is **iteration and feedback** – both the data analysis and the interface will be refined continuously with input from actual use. This is aligned with agile development and the fact that as soon as the tools are in the hands of educators, we'll learn new things that can shape further improvements.

In conclusion, the roadmap for T2.2 sets a path to transform raw data from Adaptiv'Math and similar platforms into **actionable intelligence**. It will enable a researcher to cluster hundreds of learning trajectories and uncover new pedagogical insights, and equally enable a teacher to glance at a dashboard and pinpoint which student needs help on which topic. By investing in this dual capability of analysis and visualization, GAIMHE will deliver a product that not only **collects** learning traces but truly **illuminates** them – helping to debug issues, personalize learning, and ultimately improve student outcomes through data-informed decisions.

<small>*Sources cited above illustrate relevant research findings, best practices, and examples that informed this roadmap. All visualization examples and techniques referenced (e.g., Sankey diagrams, mastery dashboards) are grounded in recent successful implementations in education analytics literature and industry.*</small>

Sources: [1] [2] [3] [4] [6] [7] [9] [10] [12] [11] [14] [15] [50] [19] [20] [11] [38] [39] [40] [41] [22] [23] [24] [25] [26] [27] [28] [29] [32] [33] [34] [30] [31] [48] [49]

---

[1] [2] [21] [29] GAIMHE.pdf
file://file_0000000070c871f480c591c70225f3f9

[3] [PDF] Learning Curves for Problems with Multiple Knowledge Components
https://www.educationaldatamining.org/EDM2016/proceedings/paper_84.pdf

[4] [5] CIKM2016.pdf
https://par.nsf.gov/servlets/purl/10021820

[6] [7] [8] [17] [18] [47] [50] 5 eLearning Data Pitfalls and How to Avoid Them
https://elearningindustry.com/elearning-data-analysis-pitfalls-and-how-to-expertly-avoid-them

[9] [10] [11] [12] [13] [14] [15] [26] PathViewer: Visualizing Pathways through Student Data
https://www.cs.cornell.edu/~eland/papers/chi2017_pathviewer.pdf

[16] How to Prove Education's Impact with Cohort Analysis - Intellum
https://www.intellum.com/resources/blog/how-to-prove-educations-impact-with-cohort-analysis

[19] [20] [45] Transforming curriculum planning with Sankey diagrams | BERA
https://www.bera.ac.uk/blog/transforming-curriculum-planning-with-sankey-diagrams

[22] [23] [24] E-mentor - Article: Elo Rating Algorithm for the Purpose of Measuring Task Difficulty in Online Learning Environments - E-mentor | e-learning, lifelong education, knowledge managemnet, e-business, economic curriculum
https://www.e-mentor.edu.pl/eng/article/index/number/82/id/1444

[25] Balancing stability and flexibility: investigating a dynamic K value ...
https://pmc.ncbi.nlm.nih.gov/articles/PMC12682724/

27 28 37 Visualizing learner engagement, performance, and trajectories to evaluate and optimize online course design - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC6502341/

30 Interactive Data Visualizations with Svelte and D3
https://datavisualizationwithsvelte.com/

31 Intro to Web-Based Visualization with Svelte + D3.js | Data Vis 2024
https://dig.cmu.edu/datavis-fall-2024/lectures/intro-svelte-d3.html

32 Pandas vs. Polars vs. DuckDb. Who "wins"? - The Pipe & The Line
https://thepipeandtheline.substack.com/p/pandas-vs-polars-vs-duckdb-who-wins?utm_medium=email&action=share

33 Pandas vs Polars vs DuckDB - Performance Showdown on Flight Data
https://blog.lordpatil.com/posts/pandas-polars-duckdb-performance-comparison/

34 Pandas vs. Polars vs. DuckDB: Fastest Analytics in 2025
https://medium.com/@ThinkingLoop/pandas-vs-polars-vs-duckdb-fastest-analytics-in-2025-44c3162e5f73

35 PyArrow, Polars, Pandas 3.0, and DuckDB: The Small-Data Revolution
https://mohammeda.li/blog/pyarrow-polars-pandas-duckdb-small-data-revolution

36 650GB of Data (Delta Lake on S3). Polars vs. DuckDB vs. Daft vs ...
https://news.ycombinator.com/item?id=45920881

38 39 46 48 49 Sankey Charts: The Unsung Hero of Student Data Analysis | by Kyle Holder | Medium
https://medium.com/@kyle.holder/sankey-charts-the-unsung-hero-of-student-data-analysis-b8aefe12a082

40 41 Mastery-based dashboards from MasteryTrack | Blended & Personalized Learning Practices At Work
https://practices.learningaccelerator.org/strategies/mastery-based-dashboards-from-masterytrack

42 43 44 academicweb.nd.edu
https://academicweb.nd.edu/~cwang11/papers/vda22-cpvis.pdf