EduRanker: An Interface for Highschool Admissions Simulation and Education

Kora Stewart Hughes[1], **Tommaso Bonetti**[2], **and Martina Balducci**[3]

[1]**New York University, Center for Responsible AI**
[2,3]**Politecnico di Milano**

ABSTRACT

Institutional admissions are profoundly impactful, particularly when it comes to education. From shaping students' academic trajectories to influencing their future career opportunities, the admissions process holds immense sway over the lives of millions. Hence it is of paramount importance that such a process takes place in a fair manner and yields beneficial outcomes. Despite this, existing methods often fall short on both these axes - perpetuating disparities and limiting the potential of many talented individuals. To mitigate this issue, we present EduRanker: an interface that educates users on the inner workings of New York City's (NYC) high school admissions process and simulates admissions outcomes. Through an application of the Gale-Shapley algorithm, EduRanker brings to light the population-level effects of student and school ranking strategies.

Keywords: Simulation, Admissions, Ranking, Matching, UX

INTRODUCTION

0.1 Stable Matching

The NYC high school admission process can be modeled as an instance of the stable matching problem. This problem involves entities of two distinct types, represented by proposing entities $\{a_1, \ldots, a_N\} = \mathcal{A}$ and receiving entities $\{b_1, \ldots, b_M\} = \mathcal{B}$. Given that each entity a_n for $n \in \{1, \ldots, N\}$ has a preference for entities b_m for $m \in \{1, \ldots, M\}$, represented in a ranking, and vice versa, the problem entails creating a matching $\mathcal{M}$ such that every proposing entity is matched with exactly 1 receiving entity: $\mathcal{M}(a_n) = b_m$. In the ideal case, the set of rankings r^{a_i} is of length M and r^{b_i} is length N. This means that each entity provides a full ranking list of every member in the opposite set with respect to their criterion. With these list lengths as a prerequisite, it has been proven that a stable matching can be computed.

The simplest instance of the stable matching problem involves an equal number of participants $N = M$ from each set, where each unique proposing entity has a unique list of preferences for each receiving entity. In this setting, each proposing and receiving entity pair has a unique preference for each other: $\forall(a_i, b_j) \in (N, M)$ $r_1^{a_i} = b_j$ & $r_1^{b_j} = a_i$. From here, we can define the matching solution $\mathcal{M}$ to be $\mathcal{M}(a_i) = b_j$. Similarly, with a more complex profile of overlapping rankings r^{a_i}, the solution (matching) is a collection $\{(a_i, b_i)\}_{i=1}^{N}$ such that $\mathcal{A} = \{a_i : 1 \leq i \leq N\}$ and $\mathcal{B} = \{b_i : 1 \leq i \leq N\}$: in other words, each proposing entity appears exactly once in the matching, coupled with one of the receiving entities from the opposite set, each of which also appears exactly once.

Among its many possible practical uses, this formalism can easily be generalized to model institutional admissions, as was also suggested by Gale and Shapley.[3] In fact, this same algorithm (although with some variations) is also used to manage admissions to medical residency programs in France.

In the case of high school admissions, the two sets are those of the applicants $\mathcal{A} = \{a_1, ..., a_N\}$ and the schools $\mathcal{S} = \{s_1, ..., s_M\}$: it is immediately evident that $|A| \gg |S|$. As a result, each applicant will be matched to a single school,[1] but each school will generally be matched to multiple applicants—at most, as many applicants as it has available seats. It follows that the solution is a collection $\{(a_i, s_i)\}_{i=1}^{N}$ such that $\mathcal{A} = \{a_i : 1 \leq i \leq N\}$ and

$$\text{capacity}(s) \geq \sum_{i=1}^{N} \mathbf{1}\{s_i = s\}, \quad \forall s \in \mathcal{S}.$$

[1]Assuming the cumulative capacity of the schools is larger than the number of students (this held true for NYC admissions in 2023).

As the name suggests, the solution needs to be a stable matching. In the high school admissions scenario, this means that, given a matching $\mathcal{M}$ over applicants $\mathcal{A}$ and schools $\mathcal{S}$, there exists no pair $(\tilde{a}, \tilde{s})$ such that $(\tilde{a}, \tilde{s}) \notin \mathcal{M}$ and both $\tilde{a}$ and $\tilde{s}$ would rather be matched with each other than with their match according to $\mathcal{M}$.

0.2 Definitions

Given an admissions scenario with a set of schools $\mathcal{S}$ and a set of students $\mathcal{A}$, we define a few key terms with respect to our algorithms:

- Ranking: an ordered list of objects.
- Matching Algorithm: an algorithm that takes in a list of student rankings along with a list of school rankings, and allocates a certain number of students to each school – limited by each school's capacity.
- Placement: the order in which a school occurs on a student's ranking.
- $\mathcal{N}(\mu, \sigma)$ denotes a normal distribution with mean μ and standard deviation σ. The values of the parameters were set to the sample mean and sample standard deviation, computed from NYC's 2023 admissions cycle.[7]
- $\mathcal{U}(a, b)$ denotes a discrete uniform distribution over the integers between the lower bound a and the upper bound b, inclusive.

0.3 Gale-Shapley Algorithm in Admissions

High school admissions in NYC are managed using a modified version of the Gale-Shapley algorithm. This algorithm, also known as the deferred acceptance algorithm, is based on proposals: students (applicants) propose to schools in their preferred order. The school receiving the proposal, in turn, will only accept if they would rather be matched with the proposing applicant than at least one student they are currently matched with.

Applicants keep proposing moving down their preferences until all of them are matched to a school, at which point the algorithm is finished. Note that schools only agree to tentative matches: at any point in the algorithm, they can reject an applicant whose proposal was previously accepted to free a seat when an applicant they prefer proposes. When this happens, the rejected applicant will start proposing to other schools they are yet to propose to. Tentative matches are a crucial feature of Gale-Shapley, as they ensure that the final outcome is independent of the order in which the applicants are considered.

Additionally, the student-proposing Gale-Shapley algorithm always produces the optimal stable matching with respect to students. This means that, among all possible solutions to the stable matching problem given a set of applicants, a set of schools, and their preferences, the one found by this algorithm is the one in which each student a_j is matched to the highest-ranked possible school in r^{a_j}. This is a direct consequence of the fact that the applicants propose to the schools: in the opposite case, also known as school-proposing Gale-Shapley, the matching produced is optimal with respect to schools. Given this property, the algorithm is strategy-proof: the best strategy for the proposing participants, namely the students, is always to list the schools ranked from most to least preferred.

Lastly, the Gale-Shapley algorithm is Pareto-efficient. This means that, given the matching it produces, it is impossible to produce another stable matching that improves the result for a given student or school without making it worse for other students or schools.

0.3.1 Variant

Real-world applications of matching often introduce constraints that distort matching stability and introduce meta-strategies amongst rankings. In the case of NYC high school admissions, while every student is welcome to apply to any school, each student can only rank up to 12 schools in their final list. In other words, this variant of Gale-Shapley creates a student-ranking upper bound. Imposing this limit, however, has a critical consequence on the process: applicants will not propose to schools they have not explicitly ranked, hence all applicants who are not accepted by any of their top 12 school choices will be unmatched.

This variation has a very significant effect on the applicants' chances of admission, and as a result, the students have to change their strategy to avoid ending up unmatched. This often translates to adding "safety schools" to the applicants' ranking: schools with higher acceptance rates and lower competition, imposed to minimize the chances of receiving no match. While in isolation, strategies like this may positively affect one's chances of admission, across entire populations of students, widespread adoption of the same strategy can have undefined consequences such as deflated admissions rates. Throughout the development of our model, we seek to simulate and identify the population-level effects of these strategies and reduce the need for them.

0.4 Goals and Contributions

The goal of this project is to gain insights into the effects of different ranking and matching strategies in high school admissions by developing an interactive simulation of two-sided matching. By grounding our model in NYC's public high school admissions data, we hope to paint an accurate picture of the student and school ranking strategies existing in everyday matching scenarios and bring to light their population-level effects.

Additionally, the simulation will be accompanied by an adaptive "nutritional label" that explains its results and highlights expected outcomes for different types of applicants. Using these results we hope to better inform future parents, students, and applicants involved in these admissions processes on how to best craft their rankings as well. By integrating this label into an interactive website, we also hope to allow students and parents to learn more about the key components of high school admissions.

1 EXPERIMENT

We consider a set of M schools $\mathcal{S} = \{s_1, \ldots, s_M\}$ and a set of N applicants $\mathcal{A} = \{a_1, \ldots, a_N\}$. Each applicant $a_j \in \mathcal{A}$ specifies a ranking of $m_{a_j} \leq M$ schools, represented by $r^{a_j} = \{r_1^{a_j}, \ldots, r_m^{a_j}\}$. This results in preference profile $\mathcal{P} = \{r^{a_1}, \ldots, r^{a_N}\}$, which is a set of preferences of all applicants. Similarly, each school $s_i \in \mathcal{S}$ generates a ranking of $n_{s_i} \leq N$ applicants, represented by $r^{s_i} = \{r_1^{s_i}, \ldots, r_n^{s_i}\}$. Unlike m, which is a parameter defined at the instantiation of each applicant, n for school s_i can be computed as the sum of unique applicants $a_j \in \mathcal{A}$ such that $s_i \in r^{a_j}$.

Using this schema, we can define the placement of a school s_i on the ranking of a student a_j as $r^{a_j}(s_i)$, where $r^{a_j}(s_i)$ is an integer between 1 and m_{a_j} if $s_i \in r^{a_j}$ and -1 otherwise.

1.0.1 Experimental Setup

In our application, we initially set a uniform distribution of $m_{a_j} = 12$ for all applicants $a_j \in \mathcal{A}$. To mirror NYC's high school admission system, we also define our simulation constants based on NYC's 2023 admissions cycle. This means that $N = 71250$ students and $M = 440$ schools are considered. Additionally, each school's DBN, name, capacity, and number of applicants were extracted from the NYC Department of Education's website and used for database elements and policy logic.[7] A basic diagram of the information stored throughout simulations is shown in Figure 1.

1.1 Simulation

The main job of the simulation is threefold: create the student & school objects, derive their rankings, and pass these results through the matching algorithm.

For each generated student, $a_i \in \mathcal{A}$, the random state – stored in the id variable – derives the lottery number, selection policy, ranking policy, ranking length, and GPA. The lottery number ℓ is a sample from $\mathcal{U}(1, 2^{128})$ (a random selection of 128 bits), which is then converted into hexadecimal notation and cast to a string. The selection and ranking policies are independent samples from $\mathcal{U}(0, 1)$ (two single random bits). The selection policy p_{sel} denotes either a random selection of schools, if 0, or a *popularity* weighted sampling of schools, if 1. Similarly, the ranking policy p_{rank} denotes either a random ordering of schools, if 0, or a *likeability* based ordering of schools, if 1. The length of the ranking m_{a_j}, namely the number of schools the student applies to, is sampled from the distribution $\mathcal{N}(6.9, 2.2)$ truncated to fit the range $[1, 12]$, and subsequently rounded to the closest integer value.

The GPA g is sampled from $\mathcal{N}(68, 8.89)$. This value can be partitioned into buckets for the screen and EdOpt schools. In the case of the base admissions cycle used in our simulations,[6] the distribution cutoffs for the screen and EdOpt (seat) markers were $[94, 89.66, 82.75, 76.33]$ and $[88.25, 77.5]$ respectively; that is, a student with GPA g will have screen marker 1 if $g > 94$, screen marker 2 if $94 \geq g > 89.66$, and so on for the remaining markers.

In the case of the generated schools, the random state is seeded by the DBN and used to generate the student-ranking policy p_{adm}, which is a sample from $\mathcal{U}(1, 3)$ and denotes the method of ordering the students in the school's ranking $a_j \in r^{s_i}$:

- If the policy is 1, the school is considered an *Open* school and students are ordered randomly, seeded by the school's name.
- If the policy is 2, the school is considered an *EdOpt* school and students are placed in descending order of their seat marker: given the ranking r^{s_i} of an EdOpt school s_i, the EdOpt marker of $r_j^{s_i}$ is less than the EdOpt marker of $r_{j+1}^{s_i}$, for $j \in \{1, \ldots, n_{s_i} - 1\}$.

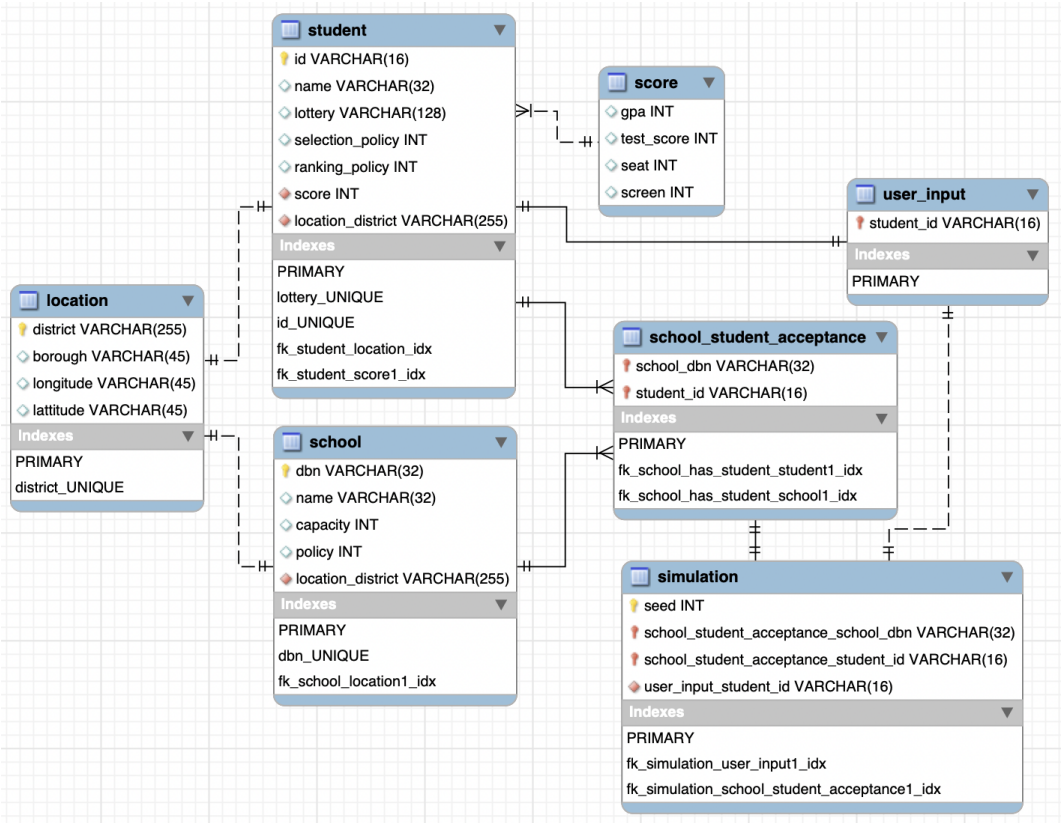


Figure 1. An example diagram for the database representing a simulation state with user input.

- If the policy is 3, the school is considered a *Screen* school and students are placed in descending order of their screen marker: given the ranking r^{s_i} of a Screen school s_i, the screen marker of $r^{s_i}_j$ is less than the screen marker of $r^{s_i}_{j+1}$, for $j \in \{1, \ldots, n_{s_i} - 1\}$.

The variables that inform the student's ranking policy, like the schools' *popularity* and *likeability*, are derived from data regarding NYC's 2023 admissions cycle.[5, 7]. Using this data, each school s_i is given a *popularity* equal to the number of applicants to the school and a *likeability*, set to the school's true applicant rate $\hat{\tau} = \frac{\tau}{n_{s_i}}$. Here, τ is the number of true applicants, namely the number of students who actually proposed to s_i during the matching.

In the case that a custom simulation is run and the number of schools is defined by the user, school information is randomly generated based on the NYC school's distribution as opposed to being used directly. These schools function in the same way as the NYC schools, with the notable difference that capacity, *popularity*, and *likeability* are randomly generated as well. Capacity c is sampled from $\mathcal{N}(145, 128.5)$. *Popularity* p is defined as the expected number of applicants: $p = c \cdot h$, where h is sampled from $\mathcal{N}(2.453505, 4.072)$. *Likeability* l is sampled from $\mathcal{U}(1, n_{s_i})$.

When a user supplies their own data (lottery number, school ranking, optionally GPA) to be added to the simulation and the matching, the schools in their ranking are real NYC high schools that can be selected through the website interface. Since these schools do not exist among those generated previously, they are mapped to the generated school they are most similar to in terms of popularity and likeability, and the resulting list of generated schools is fed as the input to the matching algorithm. The similarity between real and generated schools is computed in terms of the shortest edit distance between the string representation of the schools.

In the current instance of the simulation, location data and additional school attributes, such as AP courses offered or subject-matter specialization, are not considered. This is a main point of improvement for future iterations.

1.2 Stages

Based on the parameters used to simulate the applicants and schools, we define nine *stages*, aimed at reproducing different plausible real-world scenarios. The results obtained from each of these stages can then be compared to real data in order to evaluate which of them is better suited for approximating the real case. The parameters in question are the selection policy p_{sel}, ranking policy p_{rank} and list length m_{a_j} for applicants, and the admission policy p_{adm} for schools.

Stages 1–4 are all based on schools adopting an open admission policy ($p_{\text{adm}} = 1$) and applicants selecting as many schools as they can ($m_{a_j} = 12$), while the combinations of the selection and ranking policy of the applicants result in:

- Stage 1, where applicants select and order schools randomly ($p_{\text{sel}} = p_{\text{rank}} = 0$).
- Stage 2, where applicants select schools at random, but the list is then reordered based on *likeability* ($p_{\text{sel}} = 0, p_{\text{rank}} = 1$).
- Stage 3, where applicants select schools with probability proportional to their *popularity*, but the list is ordered randomly ($p_{\text{sel}} = 1, p_{\text{rank}} = 0$).
- Stage 4, where applicants select schools with probability proportional to their *popularity* and the list is reordered based on *likeability* ($p_{\text{sel}} = p_{\text{rank}} = 1$).

Stages 5–8 still have the applicants select as many schools as possible ($m_{a_j} = 12$), but both selection and ranking policies are sampled from $\mathcal{U}(0, 1)$ as described above. Here, different admission policies for the schools give rise to:

- Stage 5, where all schools are open ($p_{\text{adm}} = 1$).
- Stage 6, where all schools are EdOpt ($p_{\text{adm}} = 2$).
- Stage 7, where all schools are screen ($p_{\text{adm}} = 3$).
- Stage 8, where the admission policy is sampled from $\mathcal{U}(1, 3)$ as described above.

Lastly, in stage 9 the selection, ranking, and admission policies are all randomized, and the length of the ranking for each applicant, m_{a_j}, is sampled from $\mathcal{N}(6.9, 2.2)$ as described above.

1.3 Matching Algorithm

The goal of the matching algorithm is to compute a stable matching $\mathcal{M} : \mathcal{A} \to \mathcal{S}$ that assigns each student a_j to a single school $s_i \in r^{a_j}$, based on the school's placement on the student's list, $r^{a_j}(s_i)$, and the student's placement on the school's list, $r^{s_i}(a_j)$. This can be represented as a mapping $\mathcal{M}(a_j) = s_i$ for every student $a_j \in \mathcal{S}$. For our implementation, this is computed using the Gale-Shapley stable matching algorithm.

1.3.1 Initialization

The Gale-Shapley algorithm is implemented in a very simple way, starting from the base classes `Student` and `School`.

An instance of `Student` is created for each applicant $a_j \in \mathcal{A}$: it includes information such as the applicant's identifier, their lottery number, and their ranking r^{a_j} (an ordered list of school DBNs), as well as their current matching status. Similarly, an instance of `School` is created for all schools $s_i \in \mathcal{S}$, storing the ranking of the applicants r^{s_i} (an ordered list of student identifiers) along with a representation of the applicants currently accepted and an index that allows to retrieve the ranking of an applicant given their identifier.

The `School` class also has a `PrioritySchool` subclass that models schools with set-aside seats. Objects of this class have two separate lists for current seat holders (divided between set-asides and non-priority seats) and an index of the students with priority, namely those who have the right to claim a set-aside seat. Although set-aside seats are not considered in the scope of this project, this class could be useful for potential future expansions.

1.3.2 Proposals and matches

Once the applicants and schools have all been initialized, the matching is executed using a FIFO queue. If a student is yet to find a match, they will propose to their preferred school among those in their ranking they have not yet proposed to. On the school side, handling a proposal is very simple; if the school has any free seats then the proposing student is accepted. Otherwise, the student is only accepted if the school ranked them higher than the lowest-ranked student currently holding a seat, who is subsequently rejected from the school. Note that schools can only accept proposals on a provisional basis: this is an essential feature of the algorithm that ensures stability and student-optimality.

At first, all students are added to the queue (the order is irrelevant, as discussed in Section 0.3). Then, when a student is extracted from the queue they will propose to the first school on their ranking; if the school does not accept their proposal, they will then propose to the second school on the ranking, and the proposing student will keep moving down their school choices and proposing until one of the following occurs:

- One of the schools accepts the proposal, meaning the student is provisionally matched to that school.
- The student exhausts the options on their ranking without being accepted, meaning they are unmatched.

If the proposing student is unmatched, then their status cannot change: they already proposed to all their school choices, and each of them rejected the proposal. As a result, an unmatched student has their status permanently solidified, and will no longer be added to the queue of applicants waiting to be matched.

On the other hand, if the student is matched to a school, it is possible that the school had to reject a lower-ranked student who was previously holding a seat. If that is the case, the rejected student is added back to the queue and will have the chance to propose to the remaining schools on their ranking.

In either case, when the current student is done proposing, another student is extracted from the queue and the process starts over; this is repeated until the queue is empty. Since students without a match are added to the queue unless they are (permanently) unmatched, they are guaranteed to propose to every school among their choices before being removed from the matching. This also means that, when the queue is empty, the status of all matched students can be solidified: no new proposals that could change the current assignment will occur, and the matching is complete.

1.3.3 Optimizations

The performance of the basic implementation was improved thanks to a few optimizations, some of which were informed by Yehuda Gale's implementation of this algorithm.[4] Recall that n_{s_i} denotes the number of total applicants to school s_i, namely the number of students the school has ranked.

First, the sorted list of currently accepted students in the `School` class was replaced with a max-heap, indexed by the student's ranking. This still allows the retrieval of the lowest-ranked accepted student in $\mathcal{O}(1)$, enabling a quick comparison with the current proposing student, but most importantly it reduces the complexity of inserting a new student from linear to logarithmic in n_{s_i}. As Python's `heapq` library only implements a min-heap, the max-heap was simply realized by negating the student's rankings.

Second, during the instantiation of each `School` object, an index is created to retrieve an applicant's position in the ranking given their identifier. Although this slows down the creation of the object, it allows to fetch the ranking in $\mathcal{O}(1)$. Since retrieving the student's position from the sorted list representing the ranking has an average complexity of $\mathcal{O}(n_{s_i})$, this results in a significant speed-up of the function handling each proposal.

Lastly, an early check was implemented to identify whether a student's proposal would be rejected by the target school. This check runs in $\mathcal{O}(1)$ and allows the matching to avoid executing the costly proposal handler, which would take $\mathcal{O}(\log n_{s_i})$, in case of rejection.

1.3.4 Interface

The Gale-Shapley algorithm takes four inputs: the list of applicants, each with their ranking of schools r^{a_j}; the list of schools, each with their ranking of students r^{s_i}; a dictionary of student data, including their lottery numbers ℓ_{a_j}, and a dictionary of school data, including their capacities c_{s_i}.

At the end of the simulation, the output is comprised of three elements:

- The lottery numbers of all applicants, grouped by the placement of the school they were matched to, including a group containing the lottery numbers of the unmatched students. These lists are aggregated in a dictionary indexed by the placement.

- The specific outcome of each applicant, including the DBN of the school they were matched to and the school's placement on the student's preference profile. If the student was unmatched, the DBN and placement are both set to `null`. The outcomes are indexed by the identifier of the applicant.
- The specific outcome of each school, including the list of identifiers of the students matched to the school, the total capacity, and the number of true applicants. The outcomes are indexed by the DBN of the school.

There are two functions that allow to run the matching. One runs offline, loading a pre-saved input given the desired random state and saving the output to disk. The other runs online, requiring the input for the algorithm to be passed as a parameter and returning the output to the caller.

1.4 Application Logic

The application logic of the website is based on two modules: `oneshot.py`, which handles the simulation of the student and school data as detailed in Section 1.1, and `gale_shapley.py`, which executes the Gale-Shapley matching algorithm as explained in Section 1.3.

The modules are the building blocks of the `application_logic.py` script. When the script is invoked, passing the user-supplied data such as lottery number and school ranking, it simulates the student and school data through the one-shot pipeline. This data is then fed as the input to the Gale-Shapley code, as shown in Figure 2, and the output of the matching is passed back to the front-end to be visualized.[2]

In case the user does not supply any custom data, it is still possible to run the matching over the simulated applicants and schools. In order to speed up the website, however, the one-shot pipeline has been executed in advance to save the outcome of the matching with simulated data. These outcomes can simply be loaded from a file to be displayed immediately, without needing to execute the simulation in real time. Additionally, the pre-saved simulations are executed using 50 different random states: the specific simulation to be loaded is defined based on the timestamp at which the user has requested to display the simulation outcome, resulting in slightly different data being displayed at every execution. The user also has the possibility to disable this "shuffle" feature.

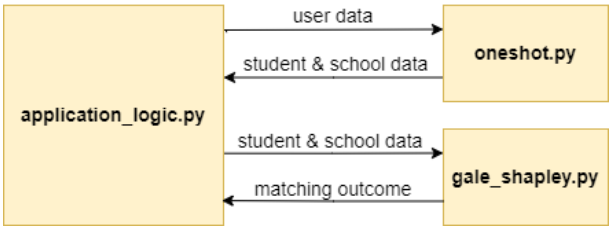


Figure 2. The interaction between the back-end modules.

2 INTERFACE DESIGN

2.1 Goal Definition

The primary aim of the website is to educate users about the complexity of the high school admission process in New York City. Through a combination of detailed explanations, data visualizations, and simulation runs, the interface wants to provide new high school students and their families with a clear comprehension of how the admission process works.

2.2 Structure of the website

To simplify the navigational challenges exhibited by complex website architecture and to minimize the cognitive load of users, the interface is composed of one main page: "Home", supported by two secondary pages — "About" and "Contacts". The main page is traversed linearly and allows users to explore each aspect of the admissions scenario before running a simulation. These sections can be broken down into an admissions overview, a description of the matching algorithm, an analysis of tentative matching, the relevancy of lottery numbers, and finally, a simulation of the algorithm.

[2] A more extensive overview of the project's architecture can be found in the ReadMe of the project repository: https://github.com/KoraSHughes/DataLife/blob/main/README.md

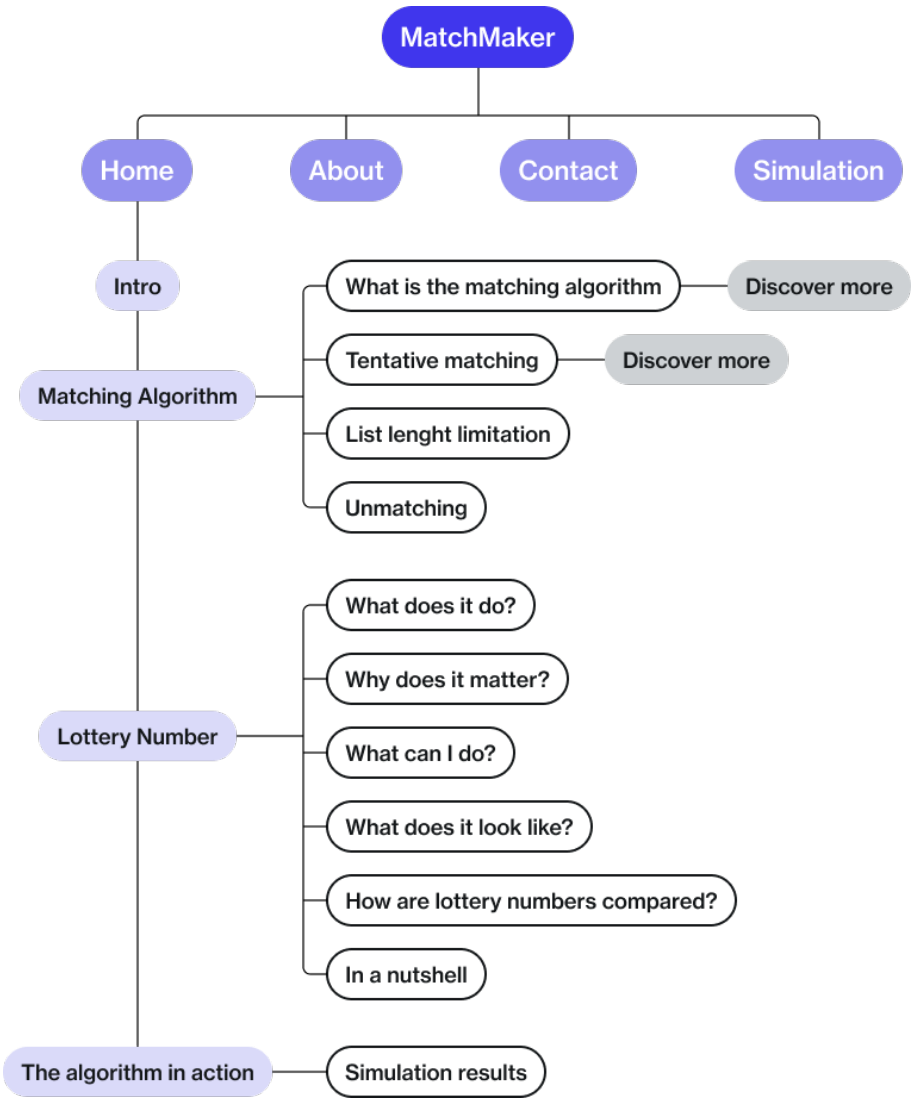


Figure 3. The structure of the website.

2.2.1 Intro

Users are introduced to the topic with a brief overview of the New York City high school admission process. This section provides a starting point for the readers to understand what follows.

2.2.2 Matching Algorithm

Here, users start deep-diving into the topic of the matching algorithm. This theme is analyzed in detail through different sections, each one focused on a specific aspect. Some animations have also been introduced to better convey the content in the text.

- **What is the matching algorithm?**: This section is all about the Gale-Shapley algorithm responsible for matching students with schools according to their preferences and the schools' admission criteria. With the help of animations, students and their parents can observe how the algorithm operates on a general level, understanding better its importance in the admission process. Furthermore, users can explore in-depth information about admission policies on a separate page.

- **Tentative matching**: Here the user can better understand how schools reserve seats for top-ranked students and how preferences are managed throughout the assignation process. With the support of a more detailed animation, users are led through all the steps of the process, increasing their overall comprehension.

Additionally, they have the opportunity to get more insights regarding tentative matching through a dedicated page.

- **List length limitation**: To ensure a comprehensive understanding of the matching process, users are informed about the practical limitations that come with restricting the length of the preference list. By looking into this topic, they can also understand the reasons why students are required to limit their list of preferred schools to just 12.

- **Unmatching**: After the algorithm's explanation, users are guided through the potential outcomes of students who remained unmatched in the admissions process. The goal is to gain awareness of the difficulties that the admission process can bring, encouraging careful thinking when creating preference lists to increase the chances of acceptance.

2.2.3 Lottery Number

The lottery number section plays a critical role in guiding users through the complexity of the admission process. Users, in fact, get insights into the lottery number system, including how it's composed and its role in student ranking. In terms of UI design, the goal of this section was to simplify this intricate aspect and provide users with a smooth and user-friendly navigation experience.

- **What does it do?**: This section sheds light on the role of the lottery number as a crucial determinant in student ranking. Through clear and concise explanations, users are guided to comprehend its function as a tie-breaking mechanism.

- **Why does it matter?**: Users are then led to comprehend one critical feature of the lottery number: once it is assigned to a student, it is unique and it can't be changed. By emphasizing its impact on student rankings, users can then recognize its relevance in shaping their ranking preferences.

- **What can I do?**: Moving forward with the discussion about the characteristics of the lottery number, this section encourages users to make informed decisions based on their newfound understanding. By providing some suggestions — on a separate page — on how to prepare their preference lists while considering the lottery number, users gain practical insights to optimize their approach.

- **What does it look like?**: The main goal of this section is to display the appearance of the lottery numbers in a user-friendly and understandable way. By using a very clear and simple animation, users can easily see how the hexadecimal number system works and the different ways in which the lottery number can present itself.

- **How are lottery numbers compared?**: This section is the core of lottery number explanation, clarifying the comparison process of lottery numbers based on the digits and their position. The design integrates visual aids and brief explanations to help users smoothly follow the comparison method.

Figure 4. The visual elements in the section "How are lottery numbers compared?".

- **In a nutshell**: This part provides an essential understanding of the importance of the lottery number and its key features in a very concise and easily understandable format. The design has been carefully crafted to highlight this section, achieved through the use of an accent color.

2.2.4 The Algorithm in action (simulation)

The page concludes with an engaging simulation, allowing users to directly observe how the algorithm works. This hands-on experience is a key tool in reinforcing users' understanding of the topics discussed earlier and encourages a more profound understanding of the complexity of the admission process.

As previously examined (1.2.4 Interface), there are two functions available for running the matching process. One function operates autonomously, requiring no user input and saving the output to disk; this pre-saved output is then loaded on the website to display the results. Even if not customized by the user, this matching simulation can provide students and their parents with an overview of the algorithm's operation.

Moreover, the system also offers another function, allowing users to input their own data and customize the simulation results. Users can provide their lottery number, GPA, and a list of schools they intend to consider, ranking them according to preference (with the option to include up to 12 preferences). This personalized input enables a more tailored simulation outcome, adapting directly to the individual circumstances and preferences of the user.

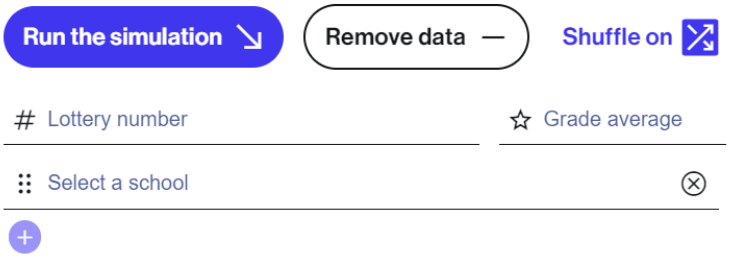


Figure 5. Input data feature to run the simulation.

2.3 User Interface Design

Influenced by current web aesthetics, the interface design aims to provide a visually appealing and user-friendly experience. Focusing on curved corners and neat lines, the goal is to give a feeling of smoothness and unity with a contemporary and welcoming look. Each section is structured and styled differently to reflect the topic and the hierarchical importance within the web page. The interface also incorporates a variety of button styles —primary, secondary, and tertiary— to strategically highlight different calls to action. These buttons serve as navigational signposts, guiding users towards desired actions or interactions within the web page.

2.3.1 Color palette

The color palette, designed with the intention of having a perfect balance between functionality and aesthetics, plays a pivotal role in shaping the visual identity of the interface. Furthermore, it helps users navigate and understand the information hierarchy better, making it easier for them to identify and interact with the different sections. The contrast of a clean white background (#FFFFFF) with sections highlighted in a delicate lilac shade (#EDECFC) creates a feeling of peace and elegance, while smaller parts are subtly defined with a soft grey hue (#FBFBFD). Vibrant blue accents (#4036ED) bring a sense of energy, providing visual prompts for users and giving importance to key elements on the interface.

Figure 6. The color palette of the website.

2.3.2 Typography

Typography was meticulously selected to ensure optimal readability and visual coherence throughout the interface. The choice of Neue Haas Grotesk, with its clean lines and simple look, as the typeface was made to infuse a modern and timeless aesthetic into the design. Furthermore, the decision to use a single typeface in various weights not only enhances readability but also fosters a sense of visual harmony creating a seamless flow across the interface. Through the use of varying font weights for headers, sub-headers, and main text, the design achieves a balanced hierarchy of information while maintaining a cohesive visual identity.

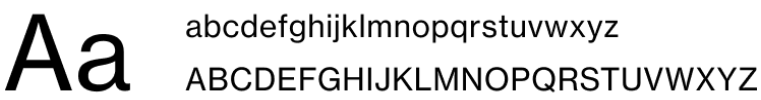


Figure 7. The Typography selected for the project: Neue Haas Grotesk, created by Monotype Studio.

2.3.3 Illustrations and Animations

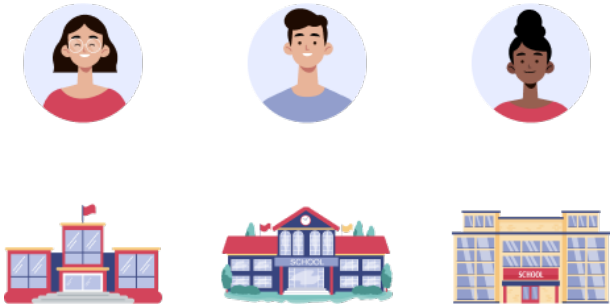


Figure 8. Representation of 3 students being matched to 3 schools.

Illustrations and animations are strategically incorporated into the interface not only to capture the attention of the user but also to facilitate their understanding of the topic discussed. These dynamic visual elements, seamlessly integrated into the user experience, are powerful tools in clarifying abstract and complex ideas, such as the matching algorithm and its functioning, presenting the users with tangible and accessible representations that enhance comprehension. Moreover, the use of personas contributes to providing contextual narratives that exemplify various user scenarios and needs to demonstrate the application of the matching algorithm without delving into individual user specifics.

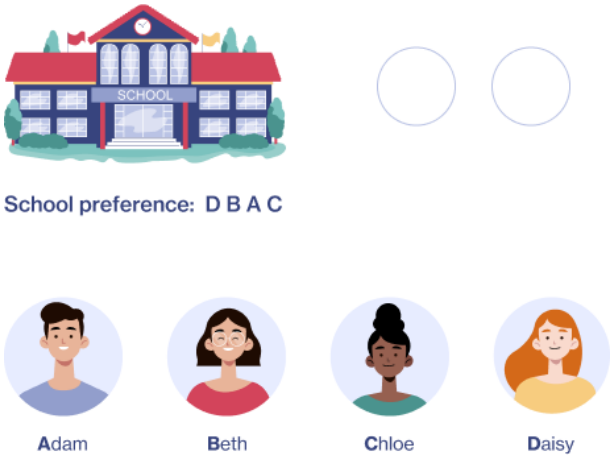


Figure 9. Representation of a school creating a ranking of 4 students.

2.4 Data visualizations

Data visualizations are essential tools in the simulation part, as they help to transform the complexity of the matching algorithm into practical insights, providing a visual explanation that complements the textual part.

2.4.1 Nutritional Label

The first element of explanation and visualization is a nutritional label, based on the one proposed in [8]. This widget gives the users an overview of the outcome of the matching process by displaying simple aggregate statistics, effectively communicating the results of the simulation to non-expert users who may be trying to grasp how this complex process works for the first time.

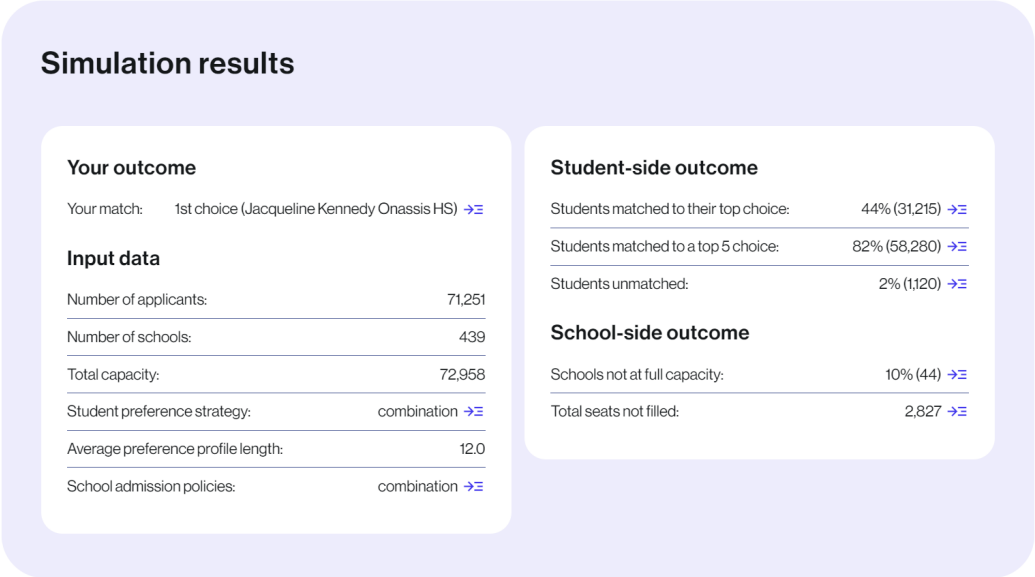


Figure 10. The nutritional label shown after the simulation is complete.

The nutritional label is divided into four sections:

- User outcome: this section is only displayed if the user has added their own data (lottery number, school ranking, optionally GPA) to the simulation, and it displays what school the user was matched to. It can be expanded to show more statistics about how the user stacks up compared to the applicants who were matched in the same school and (if applicable) in the schools the user ranked higher than their match, in terms of lottery number and GPA.
- Input data: this section contains information about how many applicants and schools participated in the matching, the combined capacity of all schools, and the strategies used by students and schools to rank each other. It can be expanded to show in more detail how many applicants adopted each combination of selection and ranking policy, and how many schools adopted each admission policy.
- Student-side outcome: this section features a summary of how many students were, respectively, matched to their top choice, matched to one of their top 5 choices, or unmatched. It can be expanded to show some statistics about the selected group of students, namely the median lottery number and average GPA, along with a plot representing the distribution of students by placement of their match. A further expansion allows users to analyze the distribution of lottery numbers for students matched with a specific placement, as well as the distribution by match placement of students whose number starts with a specific digit (see Figure 11).
- School-side outcome: this section highlights how many schools failed to fill all of the available seats and how many seats were left unfilled overall. It can be expanded to show how the schools that have unfilled seats compare to the others in terms of their popularity, which directly affects the likelihood of being included in the simulated applicants' lists.

An example of nutritional label is shown in Figure 10.

2.4.2 Other visualizations

Moving forward with the results of the simulation, the user will find other visualizations that focus on specific aspects, such as the number of students who were matched to their top choice and their lottery numbers (Figure 11). By presenting information in visually engaging formats such as charts, graphs, and diagrams, users can identify patterns and trends in the result, achieving a deeper comprehension of the dynamics of the matching algorithm and the admission process.

Visualizations are designed to be clean and easy to understand, removing any unnecessary element or intricate details, to effectively communicate essential insights in a clear and accessible way.

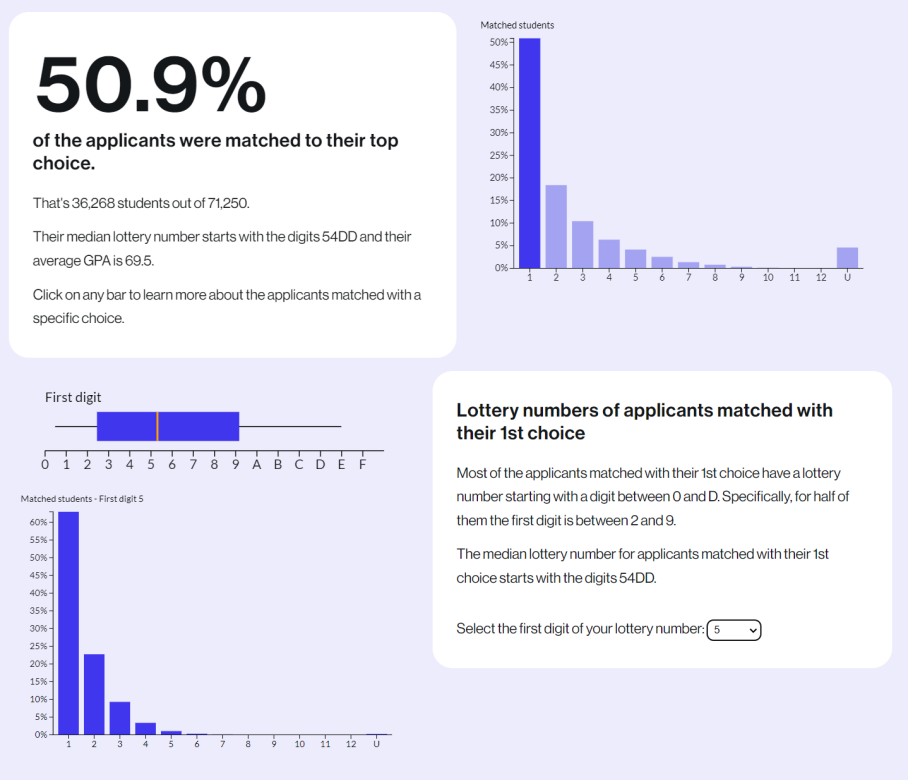


Figure 11. Detailed metrics and plots shown after the simulation is completed.

2.5 Website Architecture

The website is implemented using bare HTML and CSS. The interactive elements are managed in JavaScript, including the generation of the nutritional label and the plots displaying the aggregate statistics, created using the D3 library. [1]

Additionally, the front-end leverages PyScript to execute the `application_logic.py` script in the front-end, along with JavaScript. This allows the visual elements defined in JavaScript to communicate with the simulation results and processing done in Python. Using this pipeline, the PyScript worker executes the main simulation commands in the background, to avoid blocking the main thread when performing complex calculations. [2]

Specifically, the simulation and matching are triggered by an event listener placed in the Python code, in which the (optional) user-supplied data is passed to the simulation after being parsed by JavaScript. When the matching is complete the outcome is passed back to JavaScript through global variables set on the `window` object, at which point the Python code dispatches a predefined custom event indicating that the simulation and matching are complete. JavaScript subsequently handles this event, triggering the rendering of the nutritional label and the plots on the webpage.

This interaction between the front-end modules is shown in Figure 12.

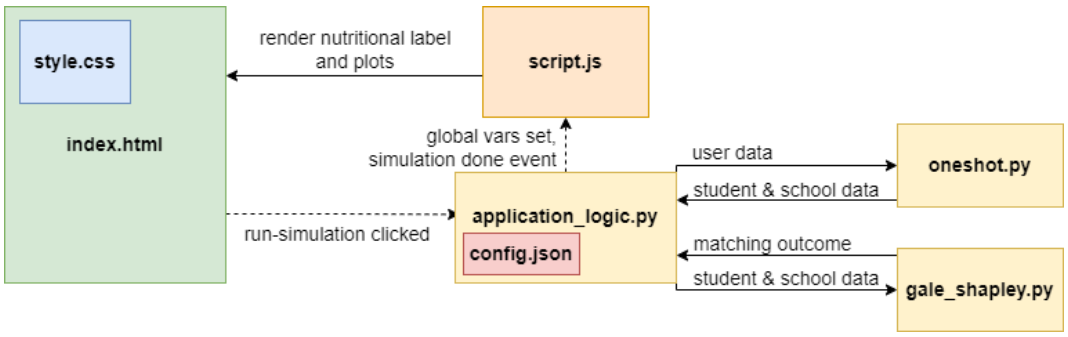


Figure 12. The interaction between HTML and front-end modules.

3 RESULTS

Though extensive user testing is required to validate the learnings students and parents can derive from this interface, there are a few key ideas that the simulation has shown.

Throughout the creation of this interface, many steps were taken to ensure its proximity to ground truth. To this extent, the overall population of students and schools and their ranking strategies were directly taken from real-world data. Similarly, the distributions of generated GPAs, capacity, and simulated behaviors were variants of the 2023 NYC high school admissions cycle and are therefore assumed to be successful replications.

Much of where this simulation's "realism" fails is in the nuance. The distribution of student and school strategies is arbitrarily uniform. A user study validating how these decisions are made is a major step forward. Additionally, hyperparameters such as the number of schools chosen or location-based matching were oversimplified leading to a larger population-level variance than real-world admissions would imply.

Despite these shortcomings, the preliminary evaluation suggests that both the formatting of the information presented and the simulation combined result in an educational experience that, at the very least, informs participants on the admissions scenario presented as a whole. More work is needed to validate which learnings are most significant and whether or not common misconceptions are properly corrected.

3.1 Evaluation Metrics

With respect to the simulation's results, there are a few key observations and metrics of evaluation that we propose with respect to a matching result $\mathcal{M} : \mathcal{A} \to \mathcal{S}$.

Given a school s_i and a student a_j in a matching where $\mathcal{M}(a_j) = s_i$, we can define $p_n \subseteq \mathcal{A}$ to be the set of applicants a_j such that $r^{a_j}(s_i) = n$. In other words, p_n is the set of students who were matched with a school at the n-th position on their ranking. We can also define p_0 to be the number of unmatched students $a_j \in \mathcal{A}$, namely those for which $\nexists s_i \in \mathcal{S}$ such that $\mathcal{M}(a_j) = s_i$. This implies that $\bigcup_{i=0}^{m} p_i = \mathcal{A}$, where m is the maximum ranking an applicant can give a school in a given simulation.

Using this definition of p_n, we can derive the number of students with placement n as $|p_n|$. Similarly, this implies $\sum_{i=0}^{m} |p_i| = N$ where N is the total number of students.

With these variables established, we can define our first metric, match ratio m_r, as

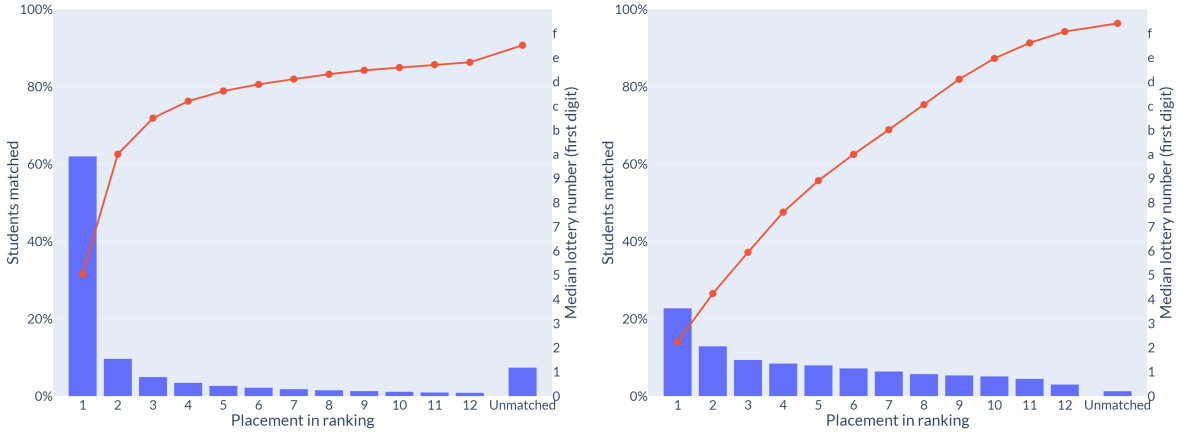
$$m_r = 1 - \frac{|p_0|}{N}.$$

The intent of this is to evaluate how many matches the simulation can create. This is particularly relevant for simulations where the sum of all school capacities is greater than the number of applicants – as often happens in real-world admissions scenarios. While this metric is a useful baseline for matching evaluation, it is not necessarily a good indicator of how satisfied students are in a simulation.

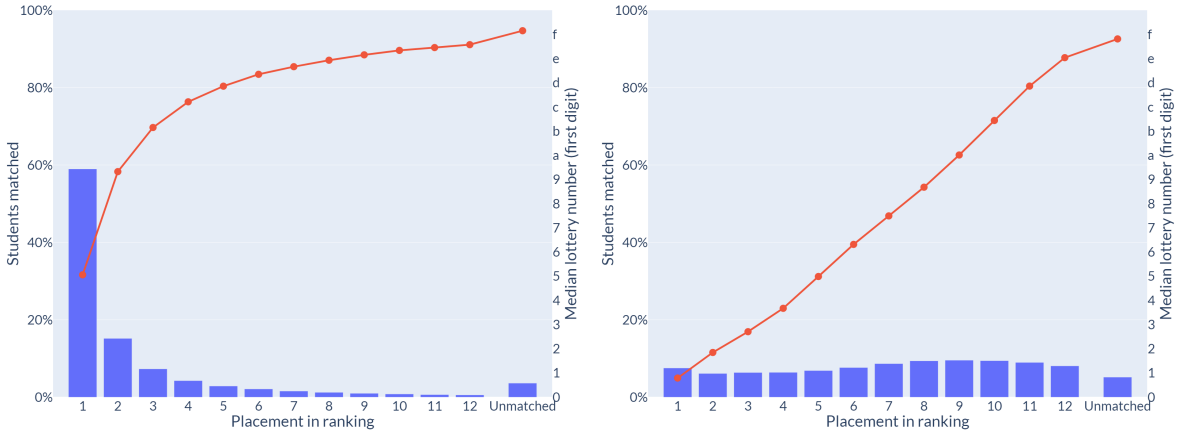
Therefore, we propose an additional metric $\bar{p}$ denoting the weighted average placement of all students in a simulation, where

$$\bar{p} = \sum_{i=1}^{m} i \cdot \frac{|p_i|}{N}.$$

The intent of this metric is to provide a rough estimation from 1 to m of what ranking school an applicant expects to get matched to in a given simulation. By choosing simulation states and ranking strategies that maximize



(a) Stage 1 **(b)** Stage 2



(c) Stage 3 **(d)** Stage 4

Figure 13. Match placement distribution and median lottery number for stages 1–4.

m_r and minimize $\bar{p}$, we hope to find the most beneficial state of high school admissions from an applicant perspective.

3.2 Observations

Stage	1	2	3	4	5	6	7	8	9
m_r	0.926	0.987	0.964	0.948	0.983	0.983	0.983	0.983	0.954
$\bar{p}$	2.022	3.860	1.532	6.221	2.165	2.163	2.165	2.167	1.607

Table 1. The values of m_r and $\bar{p}$ across the different stages.

We now analyze the values of the aforementioned metrics for different simulation stages, as defined in Section 1.2, averaging the results of simulations with different random seeds, to highlight the effect of various strategies on the result of the matching. Table 1 shows how the match ratio m_r and the average placement $\bar{p}$ change across the different stages.

We start by analyzing stages 1–4, with Figure 13 reporting the ratio of students matched and their median lottery number grouped by placement in the ranking. All schools are open, meaning students are only ranked based on their lottery number, while schools are ranked differently in each stage. In stage 1, applicants select the schools completely at random: this results in very few conflicts (students applying to the same school at the same placement), allowing more than half the students to match with their top choice. Stage 3 has a similar match placement distribution, but results in less than half as many unmatched students as stage 1 (3.6% vs 7.4%) and an average placement $\bar{p}$ that is 24.2% lower (1.532 vs 2.022). This is due to the fact that, in stage 3, schools are

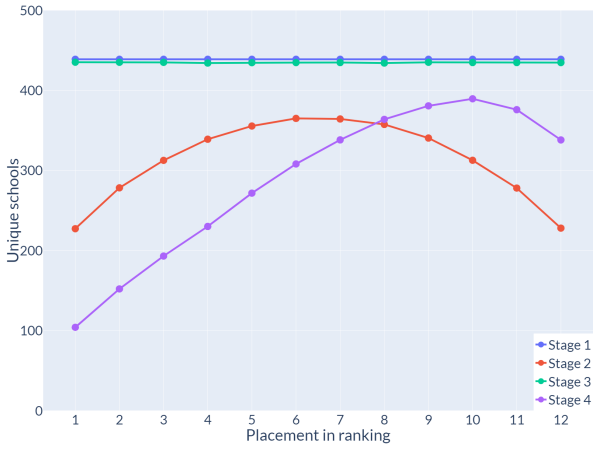


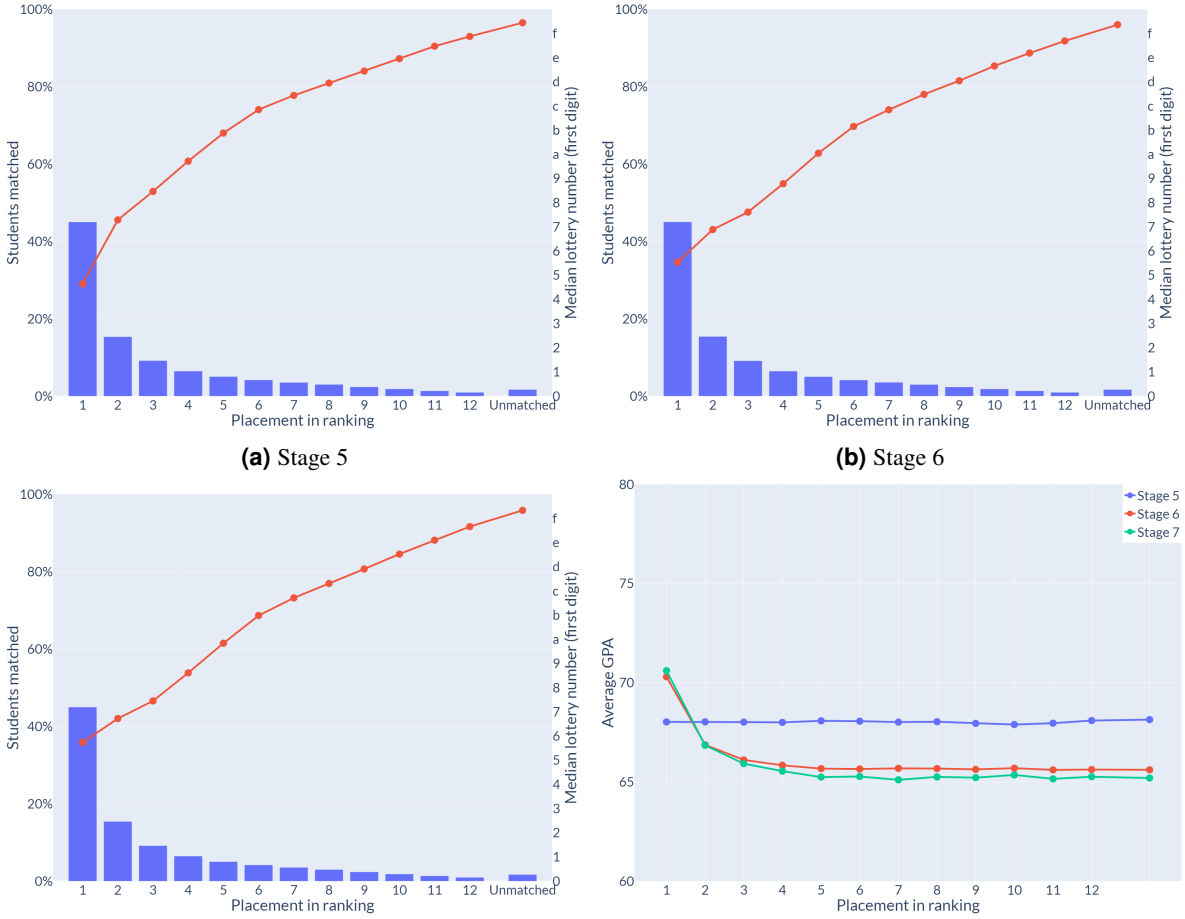
Figure 14. The number of unique schools selected at each placement across stages 1–4.

selected with probability proportional to their popularity: as this value is positively correlated with capacity, the schools to which students apply have more seats to fill (on average 284.5, as opposed to 165.7 in stage 1). Combined with the fact that the rankings are unsorted, which ensures there are still very few conflicts, this causes fewer rejections and therefore a higher m_r and a lower $\bar{p}$.

In stages 2 and 4, on the other hand, applicants reorder the schools in their ranking based on likeability: this creates a strong compartmentalization effect, meaning each school will tend to have a similar placement on the rankings of all students who selected it. The result is a stark increase in conflicts: in stage 2, only 227 unique schools are selected as the top choice (see Figure 14 for reference). Comparing the figure with stage 1, where all 439 schools are selected as the top choice by at least one student, it is easy to see why the average placement worsens so significantly, reaching the value of 3.860 (a 91% increase). It is interesting to notice, however, that stage 2 achieves the best overall match ratio m_r: the reason for this is likely the compartmentalization effect introduced by the reordering, which causes schools with average or low likeability to remain mostly empty after the applicants propose to their top choices, leaving more seats open when the bulk of the students apply and allowing to accommodate more applicants before needing to reject some. In stage 4, the conflicts increase further since the popular schools are much more likely to be selected: the number of unique schools by placement is lower than that reported for stage 2 up to the 7$^{\text{th}}$-ranked school, with only 104 unique top choices, while there is more variety among schools ranked 8–12 (refer to Figure 14). This distribution explains both the lower number of matches within the top half of the ranking placements and the higher number of matches within the bottom half, as well as the higher number of unmatched students (5.2% vs 1.3%, a 300% increase) and higher average placement (6.221 vs 3.860, up 61%).

Stages 5–7 (Figure 15) adopt a combination strategy for students, meaning each applicant is equally likely to select and rank schools based on any of the combinations used in stages 1–4, while the schools all adopt the same admission policy (open in stage 5, EdOpt in stage 6, screen in stage 7). The result for stage 5 is very close to an average between stages 1 through 4 in terms of match placement distribution, but it achieves very good values in terms of both match ratio m_r, at 98.3%, and average placement $\bar{p}$, at 2.165. Intuitively, the superposition of the effects of popularity-based selection and likeability-based ordering, which results in worse average placement values and more unmatched students when all students adopt the same strategy, does not affect the outcome as strongly when the strategies are mixed. Stages 6 and 7 achieve nearly identical values to stage 5 for the metrics as well as the match placement distribution, the only noticeable difference being the median lottery number, which takes worse values in the top two positions and better values in all others. Since, in both cases, the first admission criterion for students is their GPA instead of their lottery number, students with a good lottery number but a low GPA will be matched to worse-ranked schools compared to stage 5. Conversely, students with a high GPA and a bad lottery number will now be matched to better-ranked schools. This is also highlighted by the average GPA by match placement, as illustrated in Figure 15d: in stage 5 the GPA has little effect on admissions so its value stays constant, whereas in stages 6 and 7 it takes a higher value than the overall average for the top choice and a lower one for all remaining placements.

Lastly, the results for stages 8 and 9 are illustrated in Figure 16. In stage 8, school admission policies are sampled uniformly at random, creating a mix of stages 5–7. It is unsurprising, then, that both m_r and $\bar{p}$ take very



(a) Stage 5

(b) Stage 6

(c) Stage 7

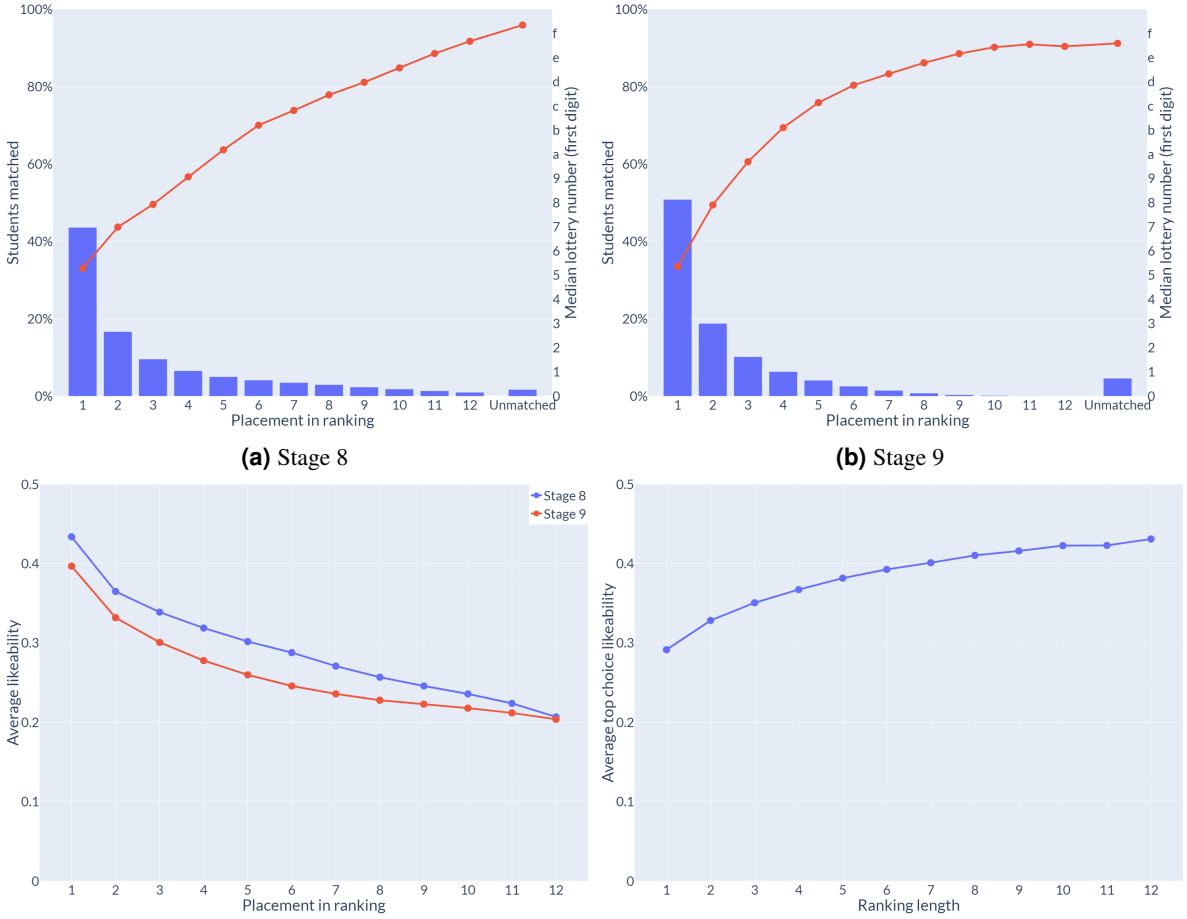
(d) Average GPA across the stages

Figure 15. Match placement distribution, median lottery number and average GPA for stages 5–7.

similar values to the previous stages, which already have nearly identical metrics; the same can be said about the match placement distribution, while the effect of the GPA on admissions, observed in stages 6 and 7, can be seen in this case as well. As for stage 9, the only difference is that the number of schools ranked by each applicant, m_{a_j}, is no longer a constant but rather it is sampled from $\mathcal{N}(6.9, 2.2)$. The result of this change is two-fold: the match ratio decreases and, at the same time, the top-choice match ratio grows. The change in match ratio, now 95.4% against 98.3% for stage 8, is a direct consequence of the change in list length: as in all previous stages applicants selected as many schools as they could (12), in stage 9 the number of schools they select naturally decreases, meaning the average applicant will need to be rejected by fewer schools than previously to end up unmatched and the ratio of unmatched students will grow. The increase in top choice match ratio is instead due to a decrease in conflicts: among students who reorder their ranking based on likeability, those who select fewer schools tend to have less likable schools as their top choice, as illustrated in Figure 16d. This is because selecting less schools decreases the probability of picking one with high likeability, as they are very rare: when the length of the ranking goes down to 1, the average likeability of the top choice coincides with the overall average likeability. In stage 9, then, the average likeability is lower than that in stage 8 for all placements in the ranking (see Figure 16c), creating fewer conflicts and ultimately resulting in more students being matched to their preferred school. Lastly, the increase in both unmatched students and top choice match ratio results in a better average placement $\bar{p}$, which decreases by over 25% compared to stage 8 and reaches the value of 1.607.

3.3 Education

This project was also carried out in partnership with the New York Hall of Science (NYSCI), an educational institution in Queens, NY that describes itself as “a hub for researchers, developers, and producers in education,



(a) Stage 8

(b) Stage 9

(c) Average likeability across the stages

(d) Average likeability by list length in stage 9

Figure 16. Match placement distribution, median lottery number, and average likeability for stages 8 and 9.

to create and study new ways of delivering equitable learning opportunities that are anchored in our Design Make Play approach to learning and STEM engagement." NYSCI develops and hosts full-scale science exhibits as well as hands-on smaller-scale interactive activities that aim to teach visitors - primarily children of different ages - about a range of Science, Technology, Engineering, and Mathematics (STEM) topics. Going forward we intend to bolster the educational aspect of this interface to improve readability for our target population: applicants and their families. Additionally, we hope to incorporate more interactive, activity-like elements into this interface and run demonstrations at NYSCI's headquarters to further gauge the usefulness of our nutritional labels and interface as a whole with an emphasis on collaborative design.

4 FUTURE WORK

In order to put this interface out into production, a few additional steps are required. A live pipeline should be set up such that the set of schools defined in the back-end is connected to the offerings of the present admissions cycle. More complex attributes such as location data (district) and AP course offerings should be included in the schools' data structure. Likewise, generated student agents ought to be modeled on a distribution of real-world location data and have district-centric preferences. More complex admissions policies, criteria, and set-asides should also be implemented to more closely mirror the complexities of ground-truth admissions.

From a simulation standpoint, there is also more work to be done. Theoretically, each applicant $a_j \in \mathcal{A}$ has a ranking $r^{a_j} = \{r_1^{a_j}, \ldots, r_m^{a_j}\}$ in which $m = M$. In plain terms, this means each student has the potential to rank all M schools. Since this is infeasible in real-world applications, matching algorithms take a partial ranking r^{a_j} that is constrained by m. Further experimentation is needed to evaluate the effect of ranking strategies and matching algorithms given an $m_{a_j} \neq 12$. Additionally, in unmatched cases, there is the potential for simulations

to infer $r^{a_j}_{m>12}$ given the characteristics of schools in $r^{a_j}_{m\leq 12}$.

Real-world matching also typically includes production-level errors such as schools that drop out after student rankings are collected or applications that are lost after the matching has taken place. Incorporating similar errors in our simulation would bring the overall system closer to ground truth and could present additional variance and bring to light unexplored challenges.

As for the education aspect, one improvement would be to gather a population of students whose pattern of school choice is similar to that of the current user. For instance, by finding overlaps in ranked schools and matching their respective likeability and popularity values, you can display the matching outcome of similar students with respect to the user. A similar feature has already been implemented in which users can compare their lottery number and GPA to those of students matched in the same school as them and or in schools the user ranked higher than their match. This can be leveraged to improve student awareness, allowing prospective applicants to better understand how good their lottery number really is, although it should be done with the necessary caution to avoid creating unrealistic expectations that could harm the students' chances of acceptance.

ACKNOWLEDGMENTS

Thank you to Andrew Bell, Prof. Julia Stoyanovich, Prof. Amélie Marian, Prof. Marco Brambilla, Prof. Stefano Ceri, and Prof. Gabriele Colombo for your guidance and feedback.

Thank you to the DataLife team for fostering such an amazing collaboration.

REFERENCES

[1] D3.js, 2024.

[2] PyScript (version 2024.3.2), 2024.

[3] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[4] Yehuda Gale. NYCSchoolDataGenerator. Technical report, https://github.com/yehudagale/NYCSchoolDataGenerator, 2023.

[5] New York City Department of Education. Fall 2023 Admission Outcomes.

[6] New York City Department of Education. NYC High School Screen Admissions.

[7] New York City Department of Education. NYC High Schools.

[8] Ke Yang, Julia Stoyanovich, Abolfazl Asudeh, Bill Howe, HV Jagadish, and Gerome Miklau. A nutritional label for rankings. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD/PODS '18. ACM, May 2018.