Kazuya Erdos, Karina Sargent, Justin Wang

Computer Science

10 May 2021

# Rkhive Software Design Document

## Statement of Goals

The software-oriented goals of Rkhive are mainly focused around the creation of a versatile web interface which retains a smooth, secure, and easily updatable flow of data from user input to publishing on the webpage.

## Functional Description – Minimum Viable Product

Rkhive includes two key features in its minimum viable product.

The app contains an interactive yearbook, with the ability to include information about past and current students at Mass Academy. Specifically, name, year of graduation, section, favorite class, personal description, quote, and image of any individual is able to be shown on the web page arranged in alphabetical order by last name and categorized by year of graduation. Users input data through a form, which also enables editing of student information after initial publishing, although the timetable for editing is limited to before the student's graduation.

The app contains an interactive alumni page, which for each student is directly mapped to their yearbook profile. Upon graduation, each student is permitted to create and maintain an alumni profile, containing personal information pertaining to college and career, opportunities or advice to current students, and contact information.

Taken together, these features create a holistic illustration of students at Mass Academy, establishing a valuable resource for future students, and develop a closer sense of connectivity within the Mass Academy community.

**Data Input**

A significant amount of programming is integrated into the spreadsheet itself. User-inputted data is captured from Google Forms with a structured question-response layout, in its raw state. That data is then sorted alphabetically to the data administration sheet, which is the main interface of the manager. Sheet functions like COUNTIF, SORT, FILTER, IMPORTRANGE, and ISNA are used to move, filter, and sort data all around the sheets.

In addition to the functions within the sheet, an auxiliary Google Apps Script was created to manipulate the data in more powerful ways as well as to transfer the data from the spreadsheet to the realtime database. A method called initialize() (provided by Google) converts the entirety of the sheet (all of the tabs) into keyed JSON data, organized by tabs and rows. Then, a trigger occurs when a cell is modified within the sheet, causing the realtime database to also update. However, current Google triggers only sense an update when a value is manually updated by a human, so formula updates, Apps Script copy-pasting, etc. will not update the database. Thus, a button was created within the sheet that calls another Apps Script function that re-initializes the entire sheet, thus populating the entire Rkhive database with all of its current data. In its current state, we found that RKhive can be updated and the data can be redeployed in around 10 to 15 seconds.

## Data Storage

Our data is stored in a Realtime Database from the service Firebase. The software dynamically updates JSON data that is fed directly into our Javascript logic, allowing for on-the-fly updates to the page without need for maintenance periods. The website is designed to only process data snapshots upon refresh, allowing for a smooth user experience, even while data is being changed, removed, or updated. The preliminary layer between the user and the realtime database is the combination Google Forms and Google Sheets, the former acting as the main method of input and the ladder acting as a data management layer for Rkhive managers to easily monitor incoming data. The Google Sheet includes a button for updating the entirety of the website, ensuring stable, controlled states of the website, while also avoiding unwanted or malicious content. The dynamic yearbook of a class shall stay in Google Sheets while the class is active at the academy, until the data is grandfathered into the database once the class graduates. This prevents over-cluttering of the sheet, as it is only needed for current data.

With this system of data storage, the contents of the HTML files themselves include only shells for data and the structural and information aspects of the site, meaning that a singular HTML file can house the entire yearbook and every alumni page, with the ability to add new class pages very easily. This is accomplished through Javascript functions which store variables on the class (2022, 2021, etc.) and mode (Yearbook, Alumni) based on user input. The script then populates the innerHTML content of automatically-generated HTML elements, based on indexed IDs. The metadata is stored in the realtime database, in a location called “statistics”, keeping tabs on data such as the population counts of each yearbook page.

As an example, assume the user wishes to access the yearbook page for the class of 2022. Upon clicking the “2022” and “Yearbook” buttons, the script will assign those values to variables tracking the year and mode currently employed by the website. Then, the software detects that both the variables are populated, so it calls methods to auto-generate the boxes that will house the student information. It knows how many boxes to generate from the ‘Statistics’ root of the database, and is pointing at the key specifically for ‘2022 yearbook.’ With that number of boxes (for example, 15), the script will populate a data houser div with 15 copies of HTML code, adding specific IDs following a standard naming convention to each element to ensure the data is routed properly where it belongs. For example, on the third box it creates, the paragraph tag containing the description for a student will be generated with the ID ‘yb-description3’ (Yearbook Description 3). The script will then be accessing the key of the third user in the 2022 yearbook root of the database (with an indexed for loop) and inject the description into that specific ID. This process is repeated for all data, instantaneously. If the user clicks the ‘2021’ button, the system will trigger an event, clear the HTML in the data houser, then repeat the process for the 2021 yearbook. As we are working with a realtime database, this all happens somewhat instantly, and yearbooks can be clicked through very quickly with almost no interruptions (in the state of testing data that we are currently in).

Currently, images are accepted through the input Google Form and placed into a google drive folder. The links to said images are saved in the sheet, and a Google Apps Script parses through these links and replaces ‘open’ with ‘uc’ to allow  for HTML to have access to and process the images from the links. This is not the ideal system for image storage, and we wish to move to Firebase storage. Firstly, Google Forms does not allow users to edit their file uploads once they upload and submit the form. In addition, Google Drive is not meant to be a database, and it is more for personal storage of files Firebase storage is a more ideal storage method as it is tightly integrated with the realtime database, and persists independent of a Google account. Also, there is more storage available through Firebase’s system, and it is easily upgradable (charged by the GB, not monthly).

We are confident that the change from Google Drive to Firebase storage will be smooth, and it is one of our top future priorities.

## Data Security

As briefly mentioned before, our HTML files themselves contain no critical data within them, so backups of those can be created, so that in the event of those files being lost or corrupted they can be replaced with no loss of data. Firebase, from our research and experience, is very unlikely to lose data, so we can confidently trust the service to store any grandfathered data. For data that is currently being updated, there are two forms of storage (sheet, database), so in the case of data loss on one end, there will be backups to replace the data from the other end. In addition to this, periodic backups can be made of the sheet and JSON format of the realtime database.

We will do our best to keep the Rkhive's editing access within the Mass Academy Community. This can be accomplished through correspondence through the WPI email system for current students, as well as potential passwords or email lists of Alumni. To ensure that Rkhive is not housing incorrect or malicious information, manual audits can be quickly performed (annually, monthly, etc.) to check that the students in the yearbook and alumni page are actual students of the Academy.

While we do not expect to have malicious events or attacks, the security of this information is very important to us. The realtime Firebase database will have secure rules only making it accessible to the founders and appointed managers. The Google Sheet will be locked for editing and viewing on all pages, save for the data administration page, which can be edited only by founders and managers. A secure Google account for the manager (more can be created for additional managers) has been created, with a strong password and potential 2-factor authentication. It is also possible that the link for the Rkhive website will only be shared within the MAMS community, and will not be posted online, but that is to be determined.

In the case of a malicious event, the entire website can be promptly taken down and the damage can be assessed. An attack on the HTML, CSS, or Javascript pages will not harm Rkhive, as they can be replaced with backup files. An attack that modifies database information will be taken more seriously, in which case we plan to relocate data to new databases, or restore backups.

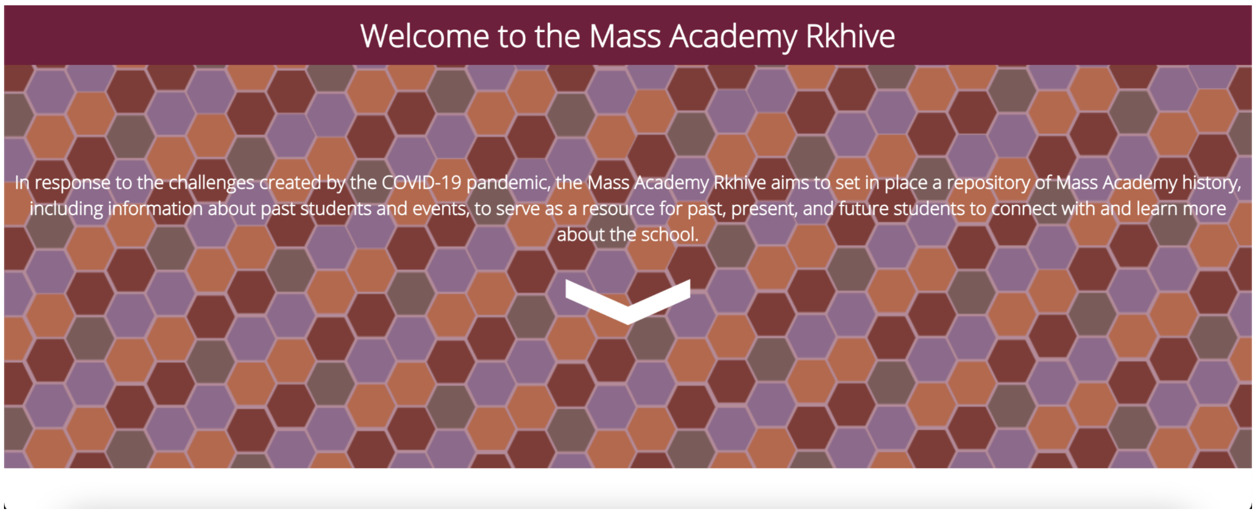
**Data Management and Updates**

To keep Rkhive running, it is paramount that at least one manager is appointed yearly. Our prime candidate for the main management position is the guidance counselor of the school (currently Ms. Post), but we also wish to have a student representative as a manager. As of now, we plan for that student to be the class secretary of Student Government. These managers will be tasked with periodically scanning the data for inconsistencies, problems, or malicious behavior. They will also be tasked with updating the data (this is not a large task, just a simple click of a button), which would help keep the site more up-to-date. However, if for any reason this is too much of a commitment for a manager, a script can be created to periodically run the updates automatically (daily, monthly, hourly, etc.), though this would then forego the monitoring step of data processing.

The founders plan to keep in touch with committed managers, hoping to create a line of experienced and trusted people to pass on the responsibilities of keeping the app afloat. If no managers are interested in maintaining the site, for any reason, the founders are willing to maintain the responsibility.

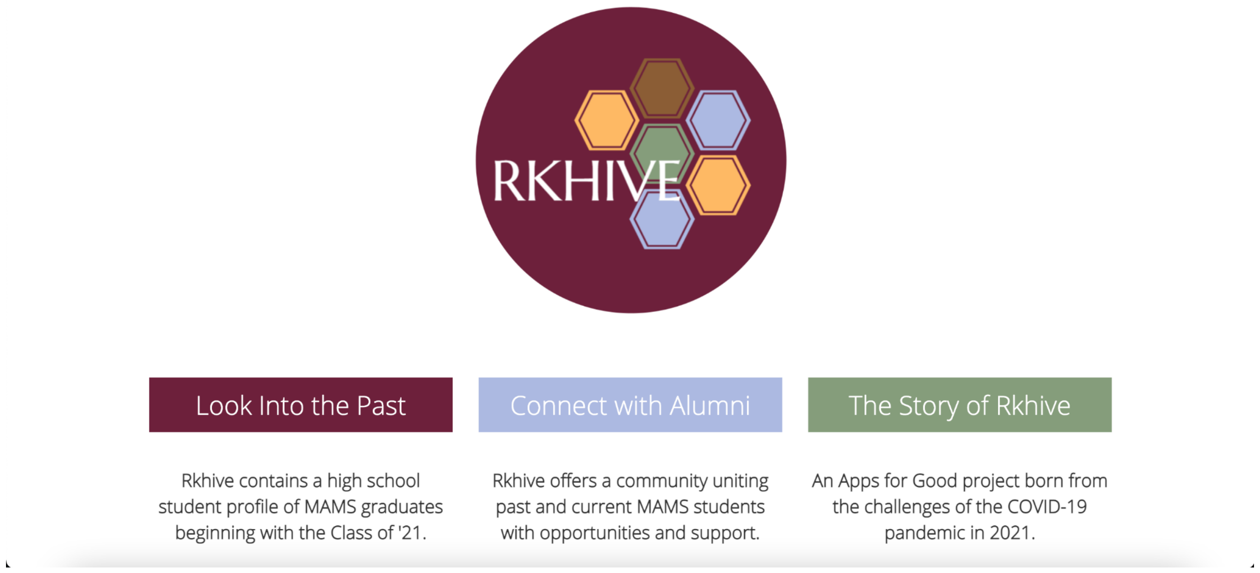
Of course, more features will be added to Rkhive past the minimum viable product. We are excited to create a cohesive and up-to-date center for the school, hopefully enhancing the experience of all students, faculty, and staff.

**User Interface**


Welcome to the Mass Academy Rkhive

In response to the challenges created by the COVID-19 pandemic, the Mass Academy Rkhive aims to set in place a repository of Mass Academy history, including information about past students and events, to serve as a resource for past, present, and future students to connect with and learn more about the school.

Rkhive contains a main index page with the purpose of introducing visitors to the app and its functionality. The initial view of the index page directs to a separate section listing key components of the website. Upon clicking, these direct the user to the given location on the website.



RKHIVE

Look Into the Past

Rkhive contains a high school student profile of MAMS graduates beginning with the Class of '21.

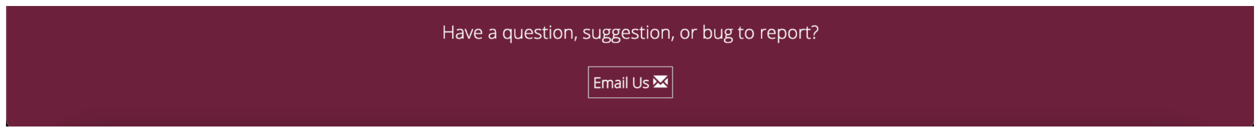
Connect with Alumni

Rkhive offers a community uniting past and current MAMS students with opportunities and support.

The Story of Rkhive

An Apps for Good project born from the challenges of the COVID-19 pandemic in 2021.

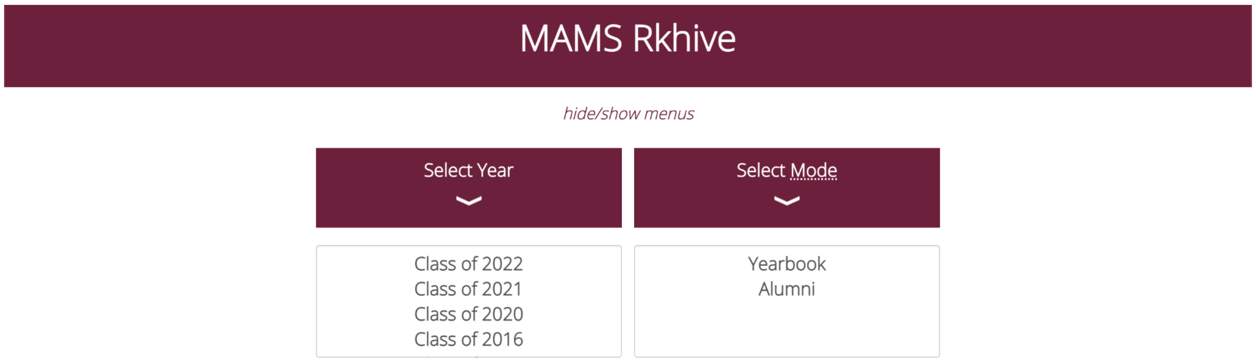
The page also includes a footer which incorporates a system of app feedback. A button, directing to an email to the Rkhive management account, aims to provide an outlet for users to submit suggestions or bug reports to be addressed by the current managers or the creators of the app.



Have a question, suggestion, or bug to report?

Email Us

The main Rkhive page, upon first visitation, will display two menus prompting the user to choose a MAMS class and one of two modes: yearbook (for high school profiles) and alumni (continuously updated profiles).


MAMS Rkhive

*hide/show menus*

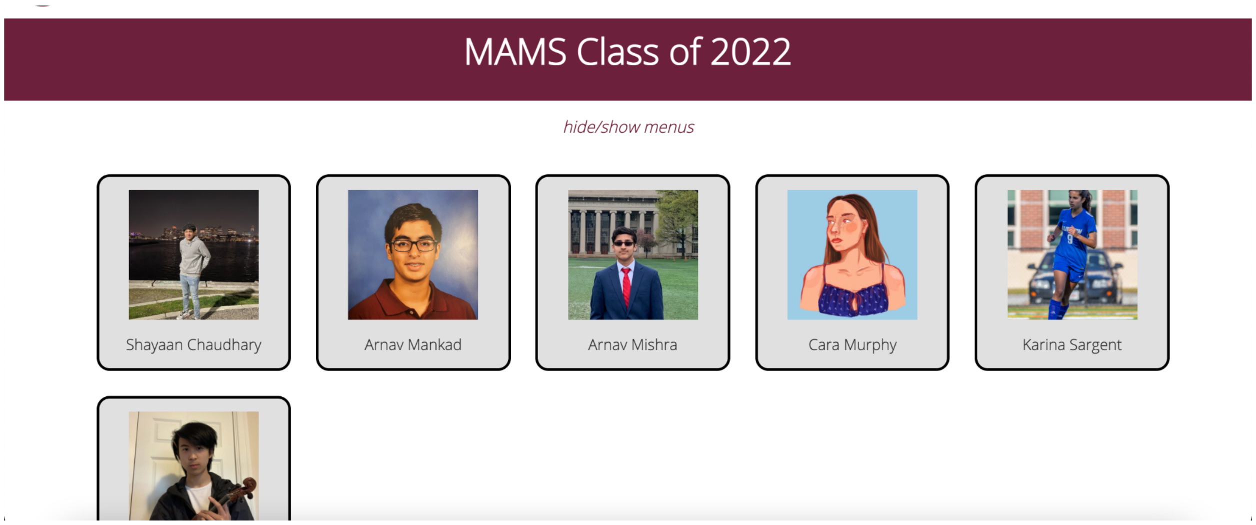
Select Year

Class of 2022
Class of 2021
Class of 2020
Class of 2016

Select Mode

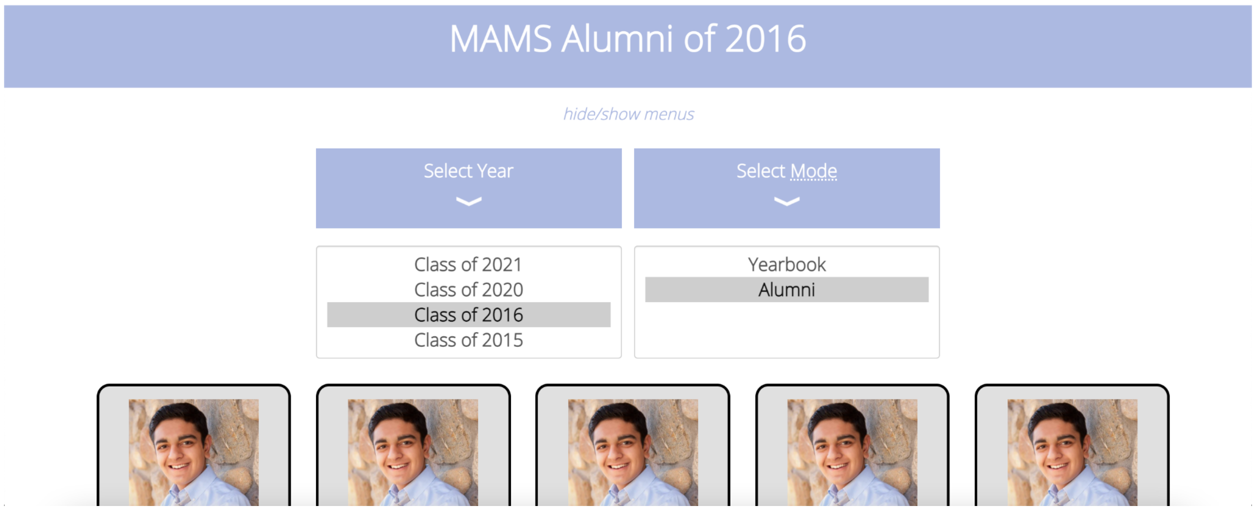
Yearbook
Alumni

Upon selection of these two criteria, the page will populate with the corresponding profiles. The selection menus are able to be hidden and shown at all times, to eliminate the overconsumption of space on the website.


MAMS Class of 2022

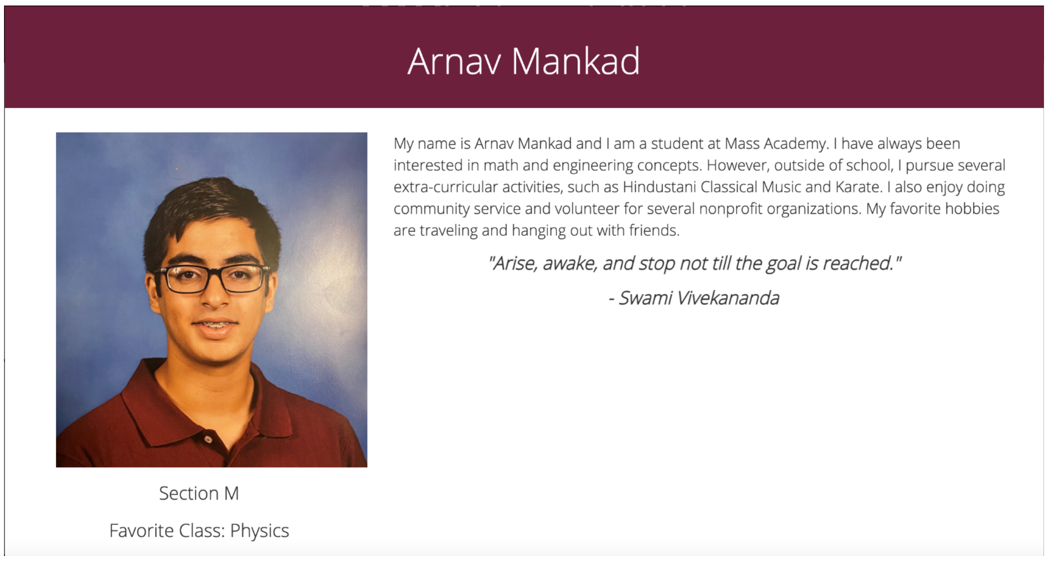
*hide/show menus*

Shayaan Chaudhary
Arnav Mankad
Arnav Mishra
Cara Murphy
Karina Sargent


MAMS Alumni of 2016

hide/show menus

Clicking a profile on the page will open a modal containing more extensive information, arranged in a standardized format, as demonstrated above. The modal is able to be closed with another click in any location.


Arnav Mankad

My name is Arnav Mankad and I am a student at Mass Academy. I have always been interested in math and engineering concepts. However, outside of school, I pursue several extra-curricular activities, such as Hindustani Classical Music and Karate. I also enjoy doing community service and volunteer for several nonprofit organizations. My favorite hobbies are traveling and hanging out with friends.

*"Arise, awake, and stop not till the goal is reached."*

*- Swami Vivekananda*

Section M

Favorite Class: Physics