Funrazor

JIF Team 4345

Keyang Lu, Kia Narayani, Tessa Orozco, Santiago Tovar, Paolo Morote

Client: Jay Elliot

Repository: https://github.com/JDF-4345-NasTech/JDF-4345-NasTech

Table of Contents

List of Figures

Terminology

API- Functions and protocols that allow interaction between otherwise separate applications, used for software development

Auth0- Verification platform that can be integrated into web applications, used for login

Backend- The part of the application not directly accessible to the user, database and business logic

Database- Structured digital data system, used to store app data

Frontend- The portion of the application that the user interacts with directly, the user interface

Layered Architecture- A software design system that separates the application’s functions into distinct layers, built to interact with each other.

Node.js- Highly useable test platform for JavaScript, used for development

Prisma- Simplifies communication with databases, used for development

PostgreSQL- Database management system often used for web development, used for development

React.js- Frontend library for building UI modularly, used in development

User Authentication- Verifying user credentials on login to ensure security of the application, handled by Auth0 in the application

Introduction

Background

The FunRazor project is centered around creating a web application tailored specifically for non-profit organizations. The goal is to streamline their fundraising activities by providing end-to-end capabilities and workflows. Current fundraising platforms are often either too costly or lack the necessary features to accommodate the unique needs of smaller non-profits. These organizations frequently face challenges such as managing donor data, executing marketing campaigns, organizing events, and handling payments, often relying on multiple fragmented tools that do not work in sync.

Our solution consolidates all the necessary tools into one system, providing an intuitive and affordable product designed specifically for non-profits. The platform will feature a central database for managing contacts, event planning, email campaigns, e-commerce functionalities for donations, and integrated tools for proposal writing and basic analytics.

System Architecture

Introduction

The overall goal of this system is to support non-profit organizations. More specifically, our system achieves this by aiding non-profit organization managers in achieving their tasks, as well as providing a platform for general users to discover and donate to organizations. We established this system as a web application that leverages React.js for the frontend, Node.js with Express in the backend, and a PostgreSQL database. The system relies on Auth0 for user authentication. This section provides diagrams outlining our static system architecture (Figure 1.1) and dynamic system architecture (Figure 1.2) to demonstrate the dependencies and interactions between our components.

Rationale

The system architecture was selected to provide a scalable, secure, and user-friendly platform using technologies familiar to our team to maximize implementation efficiency. In our frontend, React enables a dynamic and responsive user interface for managing campaigns and engaging donors. In the backend, Express handles API requests, routing, and core business logic. We also integrate Prisma to simplify database interactions, ensuring type-safe queries and schema management. Our database, PostgreSQL, serves as a reliable and efficient relational database to store user, organization, and event data, ensuring data integrity and scalability. To secure the system, Auth0 is used for authentication, allowing users to log in seamlessly without managing passwords and encryption directly. Our authorization logic, based on user roles (e.g., general users vs. organization admins), is implemented internally in the backend to ensure cost efficiency, avoiding the need for a more expensive Auth0 plan with role management. All of these technologies are widely used and industry standard, providing our team with extensive documentation.

Static Architecture

The FUNRAZOR system follows a layered architecture, with three main layers: Presentation, Business, and Database. The separation of layers and the dependencies between components is shown in our static architecture dependency diagram, seen in Figure 1.1 below. The presentation layer consists of React.js, which builds the user interface. React interacts with Auth0 to handle user authentication securely. The React front end communicates with the business layer, implemented using Node.js with Express, to process requests that involve accessing or updating stored information. Within the business layer, Prisma serves as the data access layer, facilitating interactions with the database layer, which uses PostgreSQL to store and manage data, such as user accounts, organizations, events, and donations.

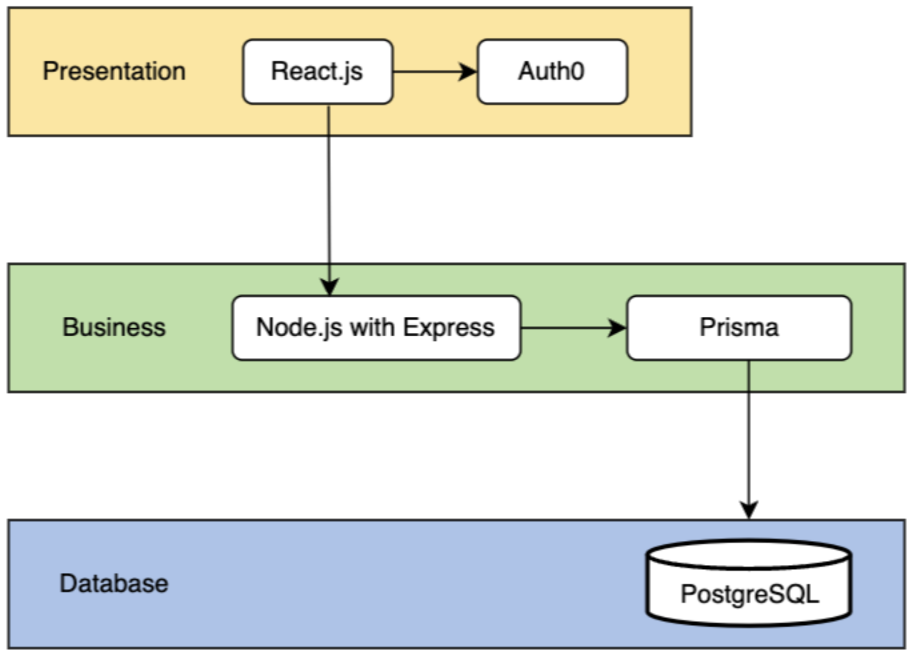


Figure 1.1 - Static System Diagram

Dynamic Architecture

Our dynamic architecture is best demonstrated by the System Sequence Diagram (SSD) in Figure 1.2. This diagram represents necessary interactions between system components and the user, allowing us to outline how our architecture allows a user to complete a task. The chosen task is a general user RSVPing to an organization's event. We selected this task, as it requires the activation of all components and demonstrates their interactions.

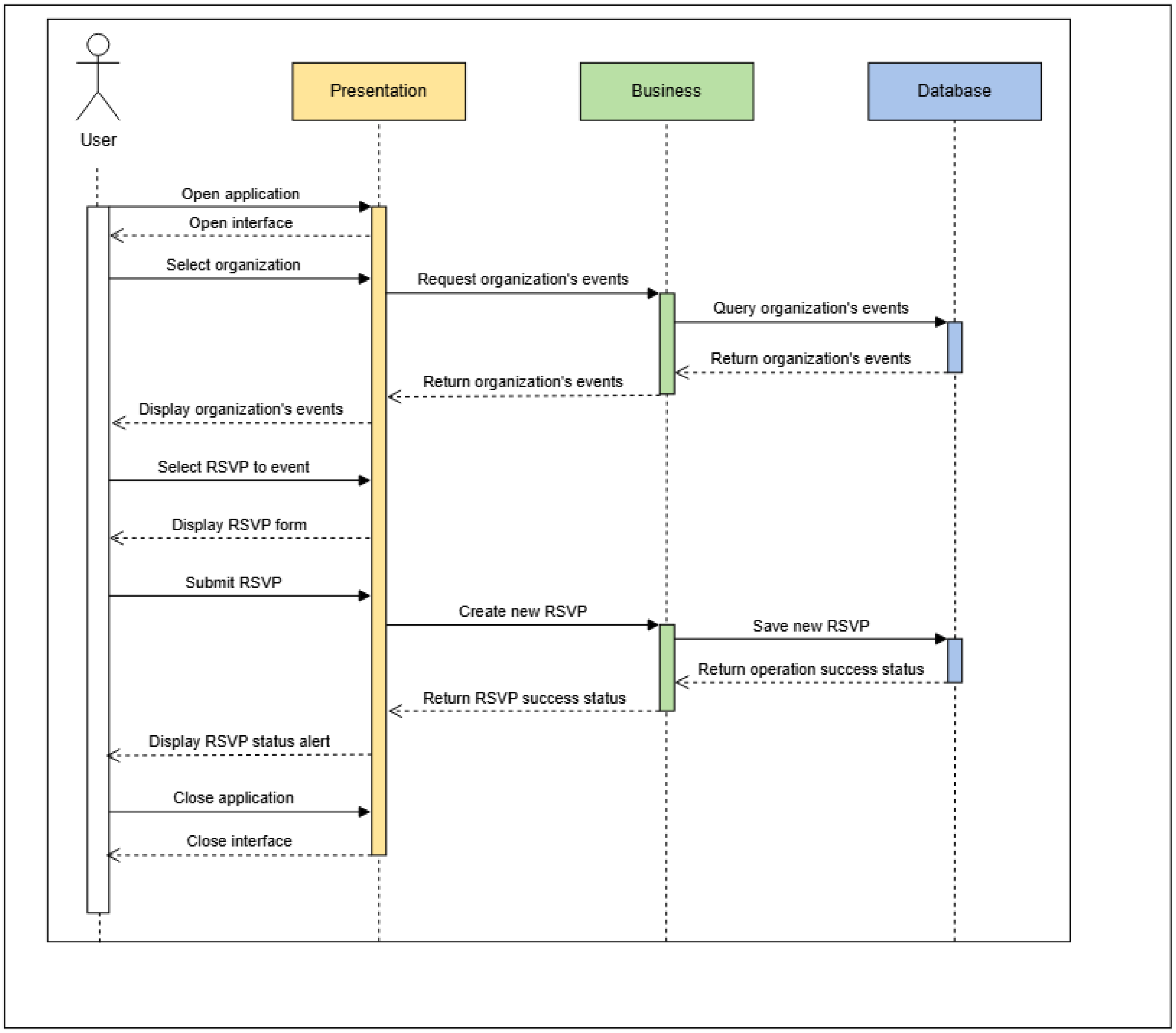

User
Presentation
Business
Database
Open application
Open interface
Select organization
Request organization's events
Query organization's events
Return organization's events
Return organization's events
Display organization's events
Select RSVP to event
Display RSVP form
Submit RSVP
Create new RSVP
Save new RSVP
Return operation success status
Return RSVP success status
Display RSVP status alert
Close application
Close interface

Figure 1.2 - System Sequence Diagram

Component Design

Introduction

The diagrams below demonstrate the structure of our web application and how the user can interface with it. For both diagrams, we used the use case of a user logging in to find events and rsvp to them. The static components of the backend database are shown in a class diagram (Figure 2.1), while dynamic runtime interactions with the system architecture are shown with an interaction overview (Figure 2.2).

Static Elements

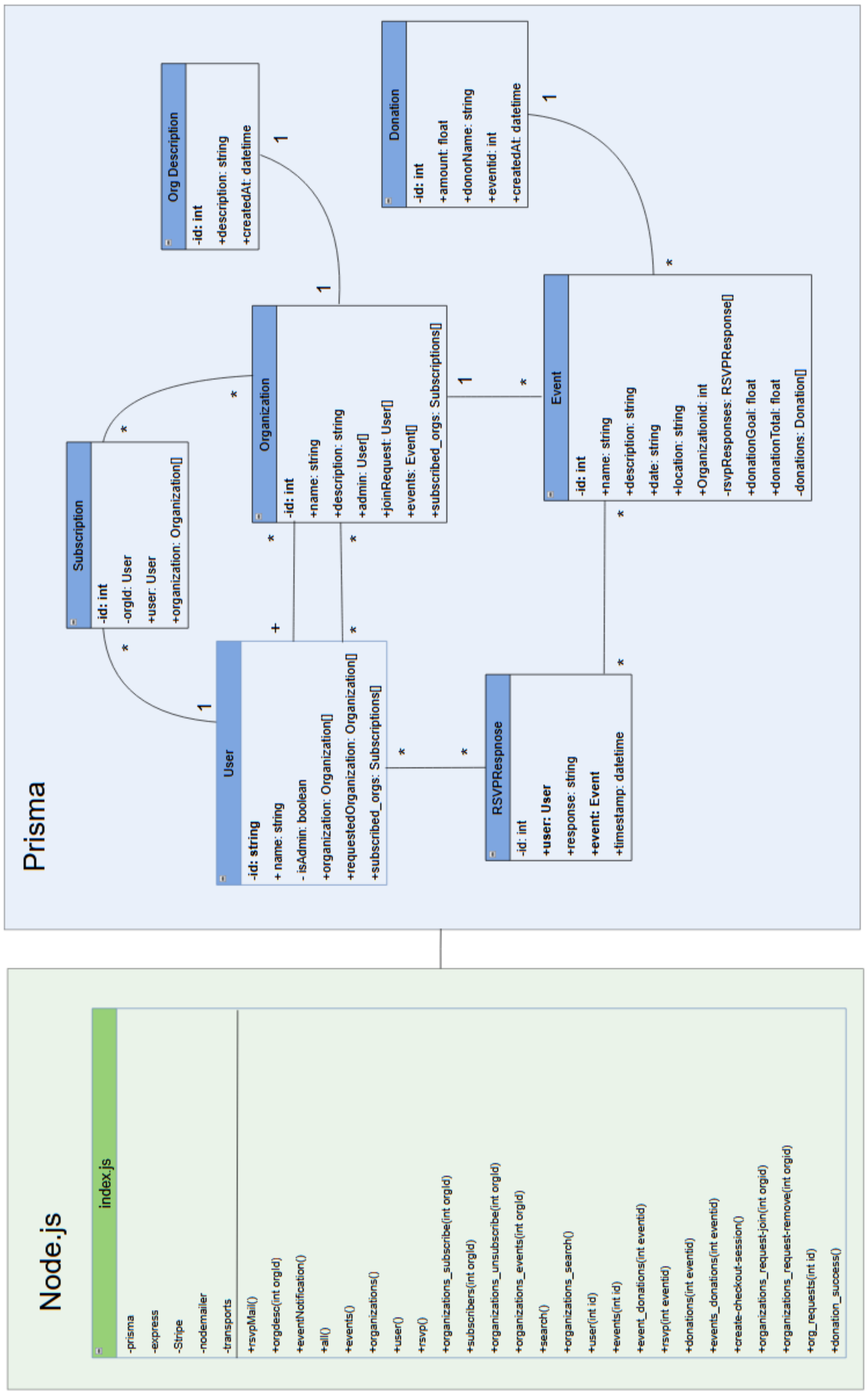


Figure 2.1 - Class Diagram

Dynamic Elements

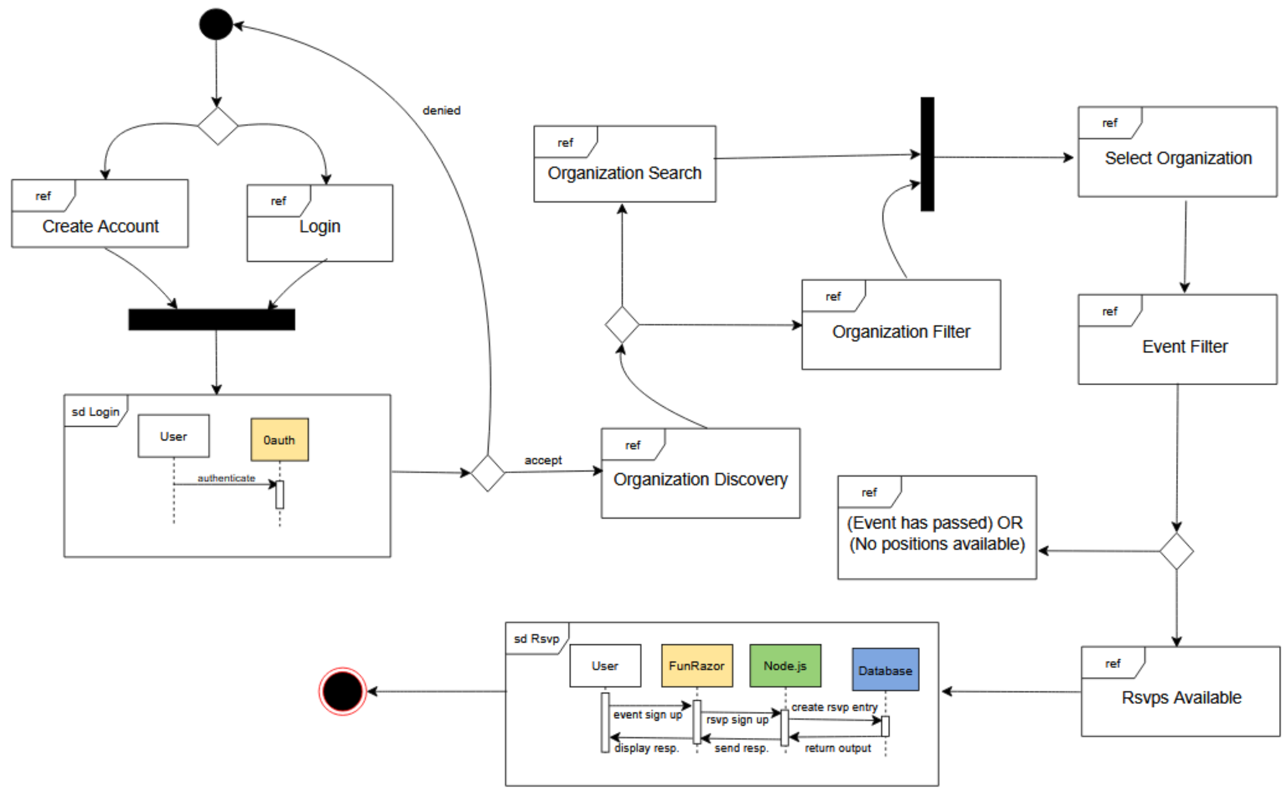


Figure 2.2 - Interaction Overview

Data Design

Introduction

This section will show how Funrazor uses PostgresSQL to save and load user, nonprofit, and event information for users. The Entity Relationship digram below (3.1) illustrates the relational interactions within the system. The data is stored on the Node.js backend and is accessed using API calls. Security of the backend is maintained using Auth0. Primary keys, used to denote unique entries in each table, are bolded.

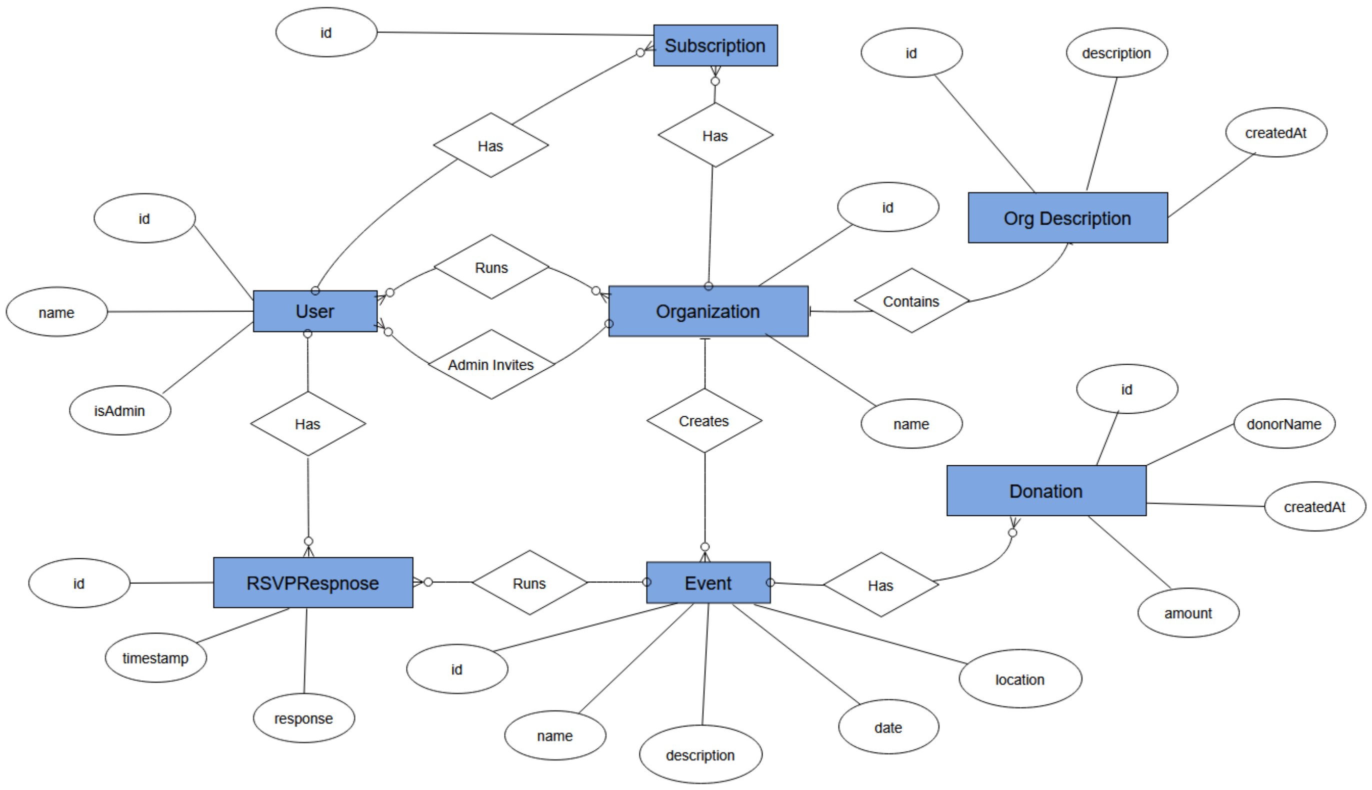


Figure 3.1 - Entity Relationship Diagram

File Use

Since this is a web application, we do not interact with any files for data storage or processing. There are several reasons, from maintaining the integrity and security of other users' data, to ensuring the scalability of the design

Data Exchange

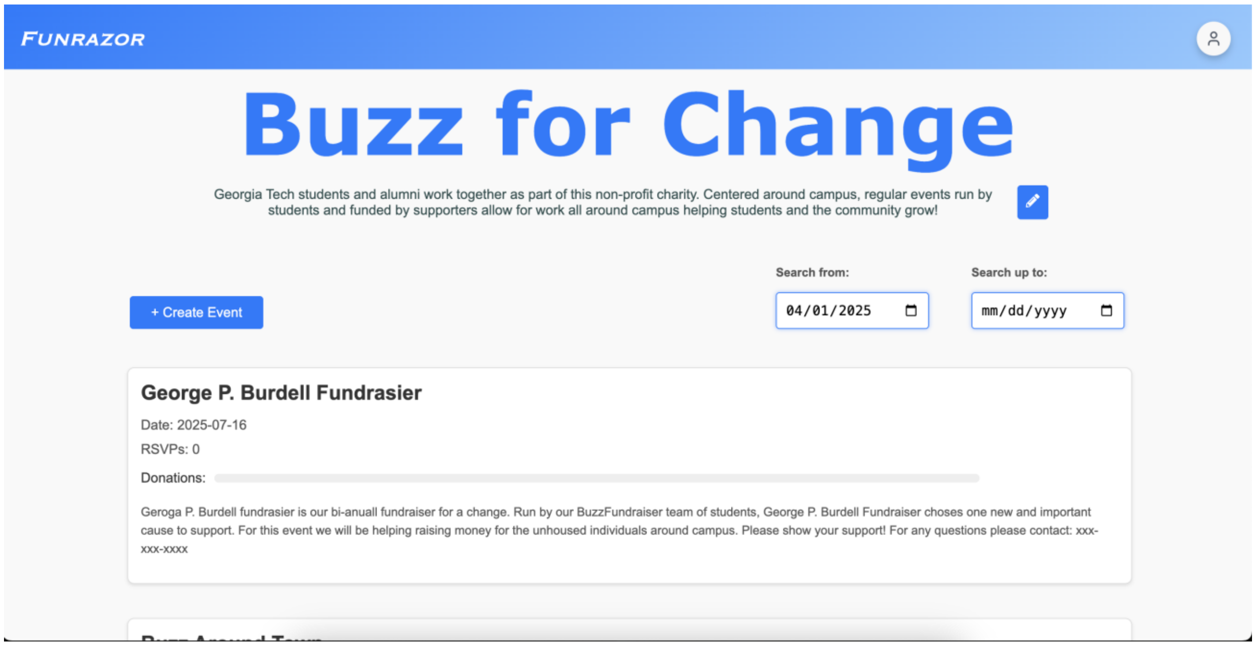
Data is sent and retrieved using Express. React sends Get and Post calls, and Node.js sends back a message depending on the outcome. Data from Prisma is sent and received in the JSON format, with the headings corresponding to the attributes to one or more tables. Security and user access is regulated with Auth0, allowing to user ease of access once they login. If users attempt to access any page without being logged in, they will be prompted to log in. We have also made sure that user-made data is validated on the frontend, prevent malformed or malicious data from interfering with the database

UI Design

Introduction

This section will show the user interface starting with the administrator side followed by the client side. It will go over how a general user would use the application page by page followed by a description highlighting heuristics used during the design process that are displayed in the completed UI.

Administrator User Interface



Buzz for Change

Georgia Tech students and alumni work together as part of this non-profit charity. Centered around campus, regular events run by students and funded by supporters allow for work all around campus helping students and the community grow!

George P. Burdell Fundrasier

Date: 2025-07-16

RSVPs: 0

Donations:

Geroga P. Burdell fundrasier is our bi-anuall fundraiser for a change. Run by our BuzzFundraiser team of students, George P. Burdell Fundraiser choses one new and important cause to support. For this event we will be helping raising money for the unhoused individuals around campus. Please show your support! For any questions please contact: xxx-xxx-xxxx

Figure 4.1 - Administrator Home Page - When signed in the administrator is sent directly to their organization's home page. Here the user can view, change their organization's description and create events as well as filter and browse event details.

In comparison to other less user-friendly designs of industry applications, this page uses a minimalistic design, putting all the user's available functions clearly on the page without including any irrelevant information. Similarly, this page makes use of common real-world symbols to increase readability such as the calendar icon for the filters, the pencil for description editing, and the "+" sign for event creation to provide clear affordances to the user. This page also remains consistent with general industry verbiage using "event", "donations" and "RSVPs" to clearly show to non-profit organizers where the information they need may be found without needing to learn specifics about Funrazor.

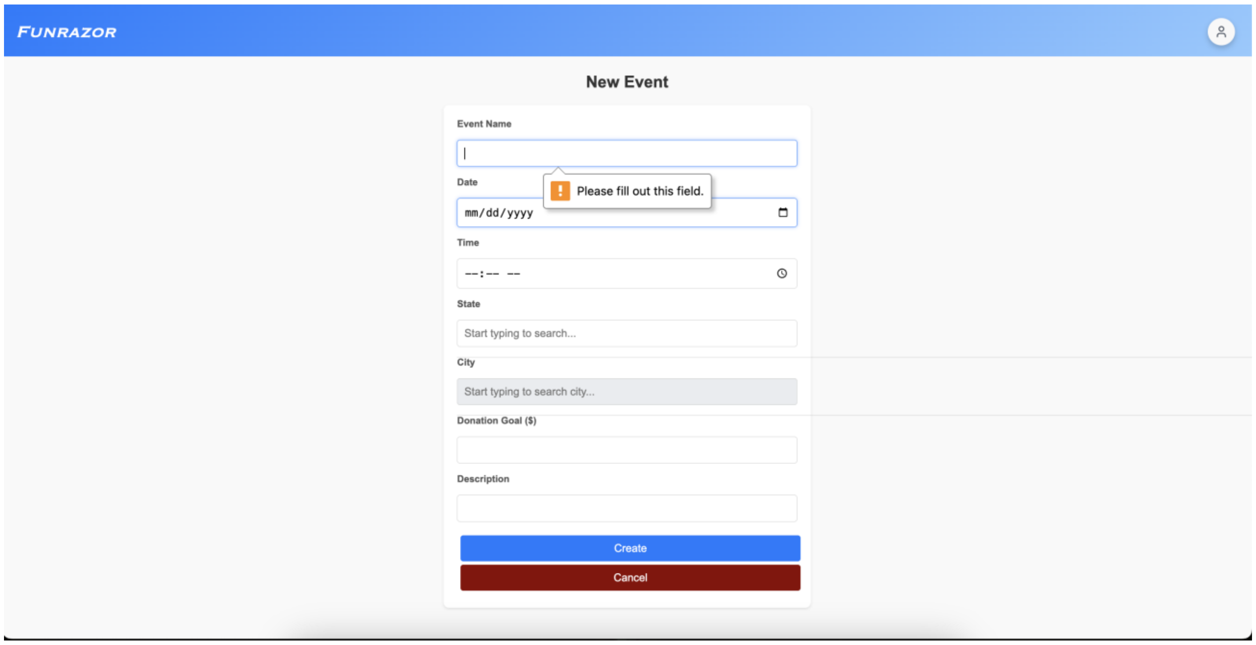
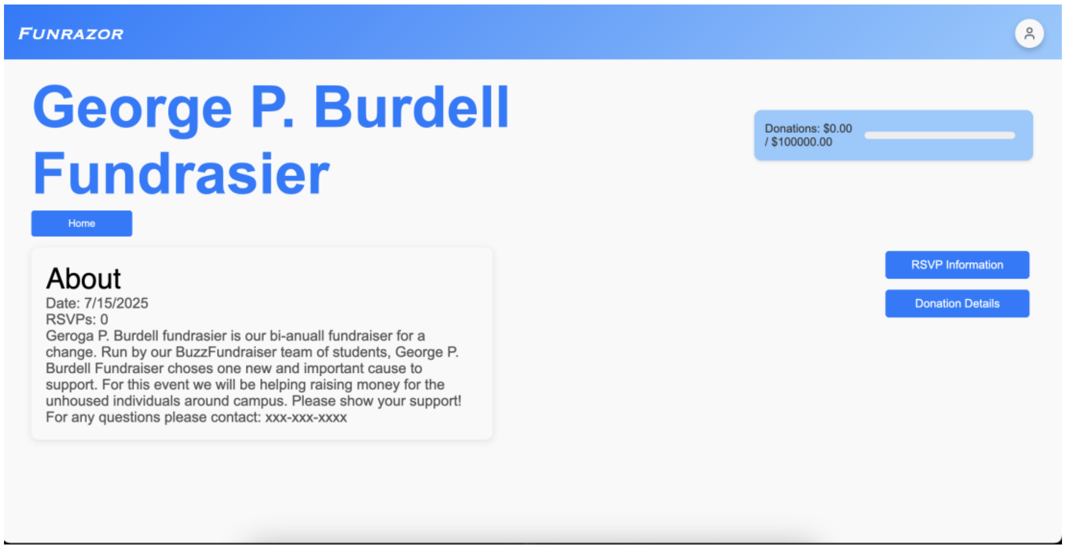


Figure 4.2 - Event Creation - An administrator may create an event to send out information to subscribers or simply to have a landing page to share and collect RSVPs and donations. Here the user is given the option to add all necessary information to create an event and is prompted to either create or cancel event creation.

This page aligns with user control heuristics with the important inclusion of the cancel button. Should the user realize creating an event was an unwanted action, having a clear and immediate way to return to the home page is vital and is clearly shown with the cancel button. This page also fulfills important user error understanding heuristics with field tips. The "! Please fill out this field" is displayed if the "Create" button is pressed without filling out any of the necessary fields on the empty field in question. This allows the user to understand the error minimizing confusion and allowing the user to act on these errors minimizing friction. Finally, this page also makes use of minimalistic design consistent with Funrazor's idea of a highly useable interface, built for simple untrained use without cutting out industry vital features. The information on the page is only that which is needed for event creation and nothing more giving clarity of function to the user.

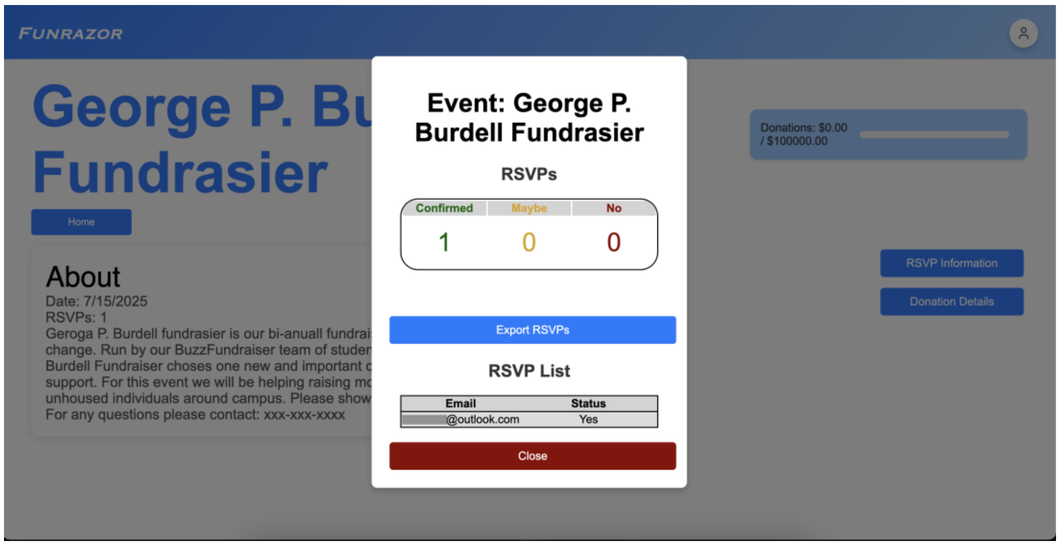


George P. Burdell Fundrasier

Donations: $0.00 / $100000.00

About
Date: 7/15/2025
RSVPs: 0
Geroga P. Burdell fundrasier is our bi-anuall fundraiser for a change. Run by our BuzzFundraiser team of students, George P. Burdell Fundraiser choses one new and important cause to support. For this event we will be helping raising money for the unhoused individuals around campus. Please show your support! For any questions please contact: xxx-xxx-xxxx

Figure 4.3 - Administrator Event Details Page – This page allows the administrator to see event title and information as well as RSVP information and donation progress.



Event: George P. Burdell Fundrasier

RSVPs

Confirmed	Maybe	No
1	0	0

RSVP List

Email	Status
[redacted]@outlook.com	Yes

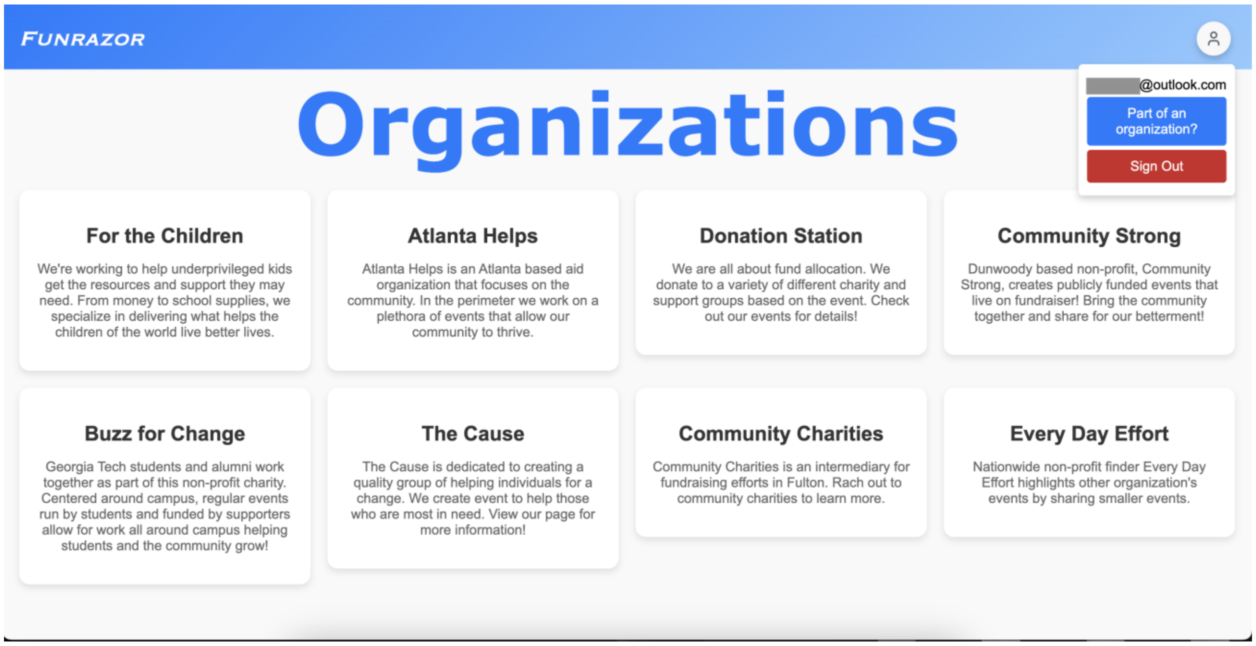
Donations: $0.00 / $100000.00

About
Date: 7/15/2025
RSVPs: 1

Figure 4.4 - Administrator Event Details RSVP Popup– This popup upon selecting RSVP information allows the user to see RSVP count, lists, and export RSVP data for external use.

This page aims to follow minimal heuristic design while also communicating useful information before clicking any buttons on screen. Donation progress is shown in the progress bar at the top right with all the other information about the event along with confirmed RSVPs is listed in the detail's container on the left under the event title. For more information the user might press the RSVP or donation button. This popup also allows users the ability to close the popup allowing the user to recover from an unwanted page click. This popup also follows industry verbiage for consistency with “Export RSVPs” referring clearly to exporting RSVP data as a spreadsheet as is available in other applications that handle RSVPs used in the industry.

Client User Interface



Organizations

For the Children
We're working to help underprivileged kids get the resources and support they may need. From money to school supplies, we specialize in delivering what helps the children of the world live better lives.

Atlanta Helps
Atlanta Helps is an Atlanta based aid organization that focuses on the community. In the perimeter we work on a plethora of events that allow our community to thrive.

Donation Station
We are all about fund allocation. We donate to a variety of different charity and support groups based on the event. Check out our events for details!

Community Strong
Dunwoody based non-profit, Community Strong, creates publicly funded events that live on fundraiser! Bring the community together and share for our betterment!

Buzz for Change
Georgia Tech students and alumni work together as part of this non-profit charity. Centered around campus, regular events run by students and funded by supporters allow for work all around campus helping students and the community grow!

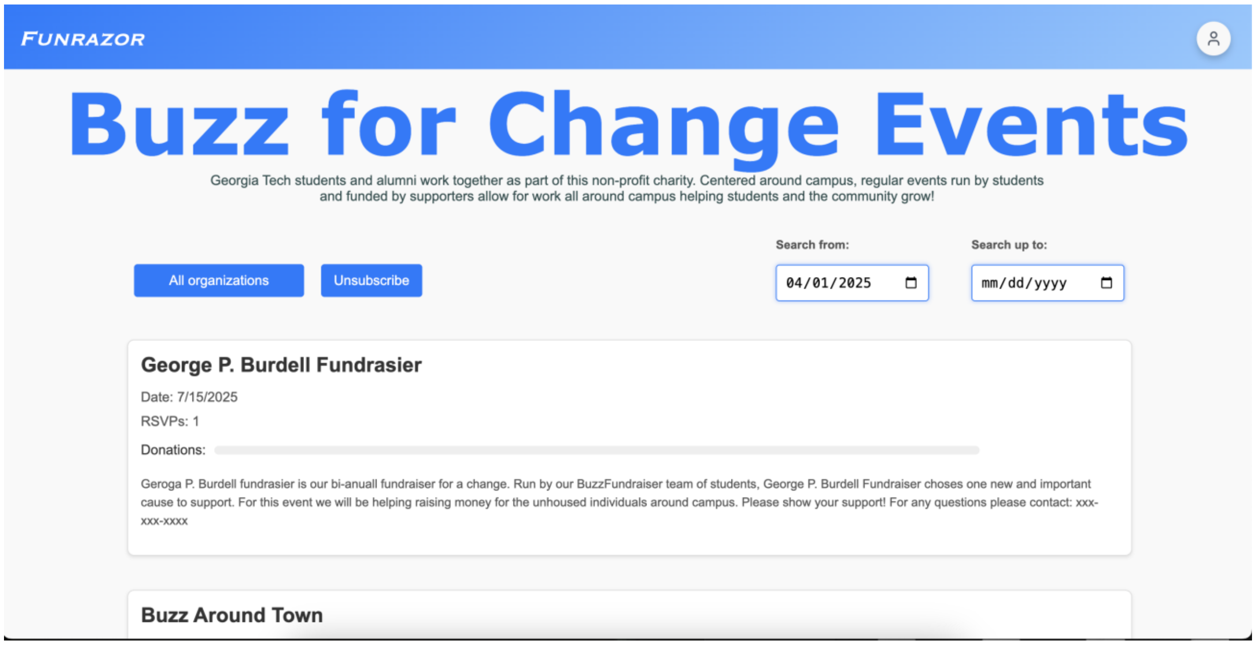
The Cause
The Cause is dedicated to creating a quality group of helping individuals for a change. We create event to help those who are most in need. View our page for more information!

Community Charities
Community Charities is an intermediary for fundraising efforts in Fulton. Rach out to community charities to learn more.

Every Day Effort
Nationwide non-profit finder Every Day Effort highlights other organization's events by sharing smaller events.

Figure 4.5 - Client Organizations Page – Upon sign-in a general user account will be presented with the organization page which populates with organization cards with the organization name and description.

This page uses a design that attempts to match the real world through its card system. Hovering over any of the organization cards makes them lift off the page slightly with a drop shadowing, making it clear to the user that these cards are clickable and that they are meant to look through them as though they are business cards. Visible on all other pages as well as this one is the account drop down. We use real world simples here again using a person icon to suggest that this is where the user might handle their account status such as being part of an organization and signing out. Again, this page follows Funrazor's mission to create a highly usable minimalistic design. A client-side user who is not explicitly looking to join an organization is not meant to do anything else on this page but look at organizations, as such this page clearly states it is a list of organizations and only shows the user what they might need to judge whether they are interested in learning more about the any of the listed organizations.



Buzz for Change Events

Georgia Tech students and alumni work together as part of this non-profit charity. Centered around campus, regular events run by students and funded by supporters allow for work all around campus helping students and the community grow!

George P. Burdell Fundrasier

Date: 7/15/2025

RSVPs: 1

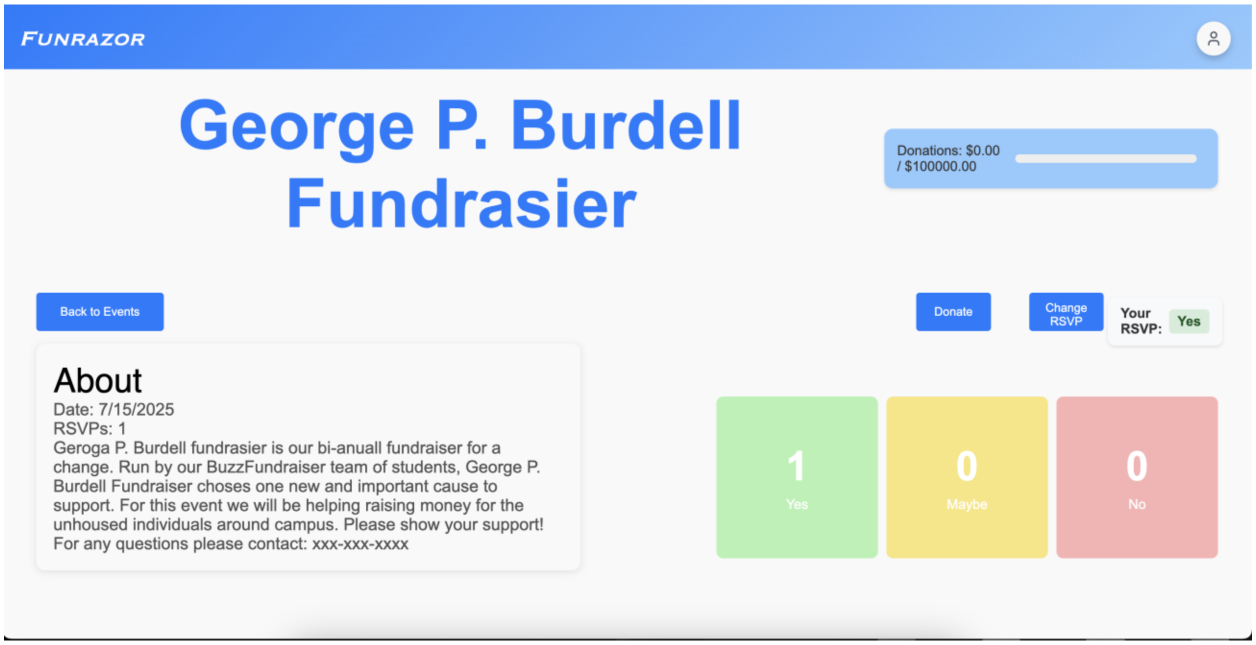
Donations:

Geroga P. Burdell fundrasier is our bi-anuall fundraiser for a change. Run by our BuzzFundraiser team of students, George P. Burdell Fundraiser choses one new and important cause to support. For this event we will be helping raising money for the unhoused individuals around campus. Please show your support! For any questions please contact: xxx-xxx-xxxx

Buzz Around Town

Figure 4.6 - Client Organization Details Page – This page shows the client an organization's events and description as well as allowing the user to subscribe to the organization.

This page follows similar design principles that the app allows throughout. The user is given a minimalistic design without extra clutter. The user's options are clearly to return to the organization page with the "All organization buttons", subscribe or in this case unsubscribe as the user is already subscribed to organization and filter and select events. These event items also ground themselves in the real-world with card-like motion when hovered affording the user's ability to select them as well as calendar icons showing the user that they can click on these filters to select the date. Finally, the use of unsubscribe/subscribe is important here as a form of consistency with other industry applications as it is common to "subscribe" to an organization to receive updates and notifications about events. Using verbiage consistent to what a user would expect from an application such as Funrazor increases usability.



About
Date: 7/15/2025
RSVPs: 1
Geroga P. Burdell fundrasier is our bi-anuall fundraiser for a change. Run by our BuzzFundraiser team of students, George P. Burdell Fundraiser choses one new and important cause to support. For this event we will be helping raising money for the unhoused individuals around campus. Please show your support! For any questions please contact: xxx-xxx-xxxx

Figure 4.7 - Client Event Page – This page shows the user event information, RSVP information including their own RSVP status if applicable, as well as allowing the user to decide to donate.

This page also follows Funrazor's minimal design while also communicating useful information before clicking any buttons on screen. Donation progress is shown in the progress bar at the top right with all the other information about the event along with confirmed RSVPs listed in the on the right side of the page. This page also allows the user to see system status with the "Your RSVP" widget showing up upon RSVP. This ensures that any system information relevant to the user is immediately visible along with the confirmation email sent to the user on RSVP. The user is also given a "Back to Events" button should they have miss clicked or feel they do not like the event allowing the user to recover from their error should they need to.

Appendix

Stripe API

Stripe is a secure payment platform that allows businesses to accept various payment methods like credit cards, debit cards, and bank transfers. Stripe is used to accept and process donations for Funrazor as well as to disperse receipts for users' reference or use for tax credit if applicable.

Team Members

Team Member	**Contribution**	**Contact Information**
Keyang Lu		2156781226, klu303@gatech.edu
Kia Narayani	Coding, Documentation	3414659057, anarayani3@gatech.edu
Tessa Orozco	Tech stack set up and guidance, Coding, API research and implementation, Task assignment, Testing	5713100204, torozco6@gatech.edu
Santiago Tovar	Code, Design, Client Liaison	6785546712, stovar6@gatech.edu
Paolo Morote	Coding, Debugging, Documentation	9253516998, pmorote6@gatech.edu