Final Report

CSCE 606 Software Engineering

EventNXT

Team Members

Amin Isazadeh

Jakob Kirby

Casey Quinn

Sushmita Pattanaik

Vaibhav Bajaj

Table of Contents

1. Introduction

EventNXT is a comprehensive Event and RSVP management app that is specifically designed for event organizers. An event organizer can create events, upload box office data, manage seating categories, manage VIP guests, and send emails related to RSVP, Referrals, or Purchase to these guests. They will also be able track the purchase information available in box office spreadsheets. Normally a user will get the box office spreadsheet from any ticketing website like Ticketmaster / Ticket Tomato / Event brite, but we have also added a feature to directly connect to these websites from inside the EventNXT app and get the latest data. A user can send referral emails to guests, where a guest can enter the email of a person who they want to refer. By doing so, they can get some rewards, and the amount or percentage of rewards can be managed under Referral Management in the app. We are also able to send QR codes to users for guest identification and ticket verification.

The current application is operational, with the exception of some issues related to the authentication system. The client has requested that the application be integrated with Event 360 and be able to connect seamlessly with their CRM app to enable login and other data sharing functionalities. However, the current application's authentication system is embedded deeply within its functionalities, making it extremely complex to modify or integrate with external systems. Moreover, the application employs complex JavaScript, which poses challenges for future maintenance and scalability. To address these issues, it was decided that a new application would be developed following SaaS standards to ensure ease of maintenance and scalability. The new application is a bare-bones version with basic functionalities that can be further developed by the next team. While it is possible for the next team to continue working on the old application and resolve its limitations, it is recommended that they treat it as a prototype to identify the desired functionalities and implement them in the new application. This approach will ensure that the new application meets all necessary requirements and is better equipped to handle future changes and enhancements. The stakeholders for this application are the client (Mr. Tito Chowdhury, CRM team, and Event Organizers)

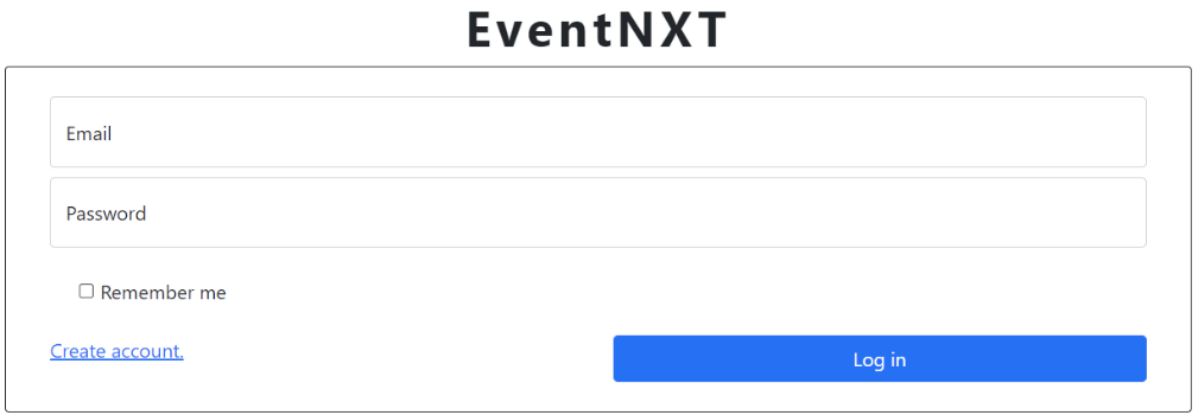
2. User Stories

Note: All the user stories mentioned below are completed and deployed in the latest app, if not explicitly stated as incomplete or in progress.

Front-End Improvements

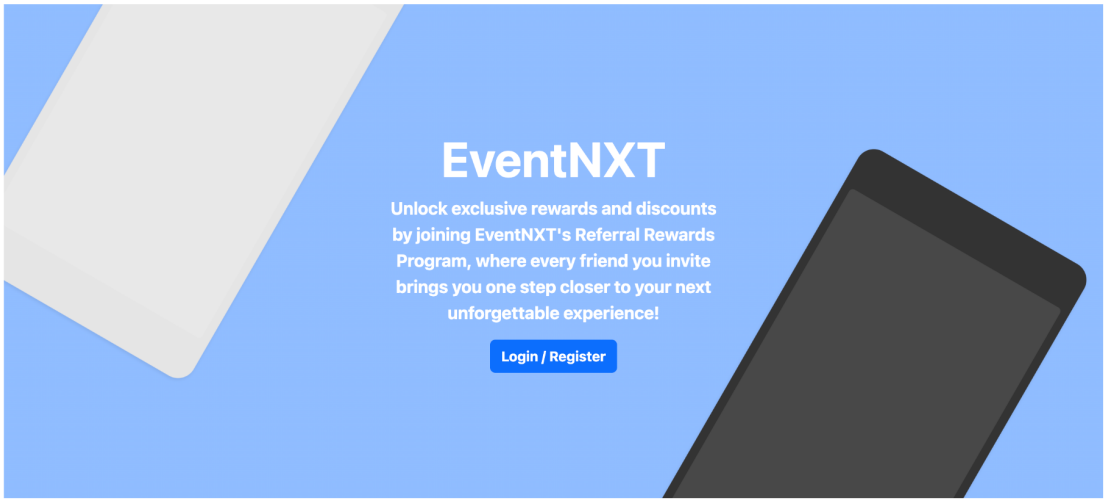
The client desired a complete overhaul of the existing front-end, articulated through several user stories. Each story was assigned a single point, reflective of the time-intensive nature of front-end work. While this work is less complex than implementing functional features, it typically doesn't involve substantial back-end modifications. The original views were not uniformly mobile-friendly, and their design was minimalistic, utilizing basic HTML and Bootstrap5 CSS styling. The implemented user stories relating to the front-end revamp include:

Landing Page Revamp and Mobile Optimization (1 point):
The client sought a mobile-friendly landing page with promotional content and a visual design that could compete with industry peers like EventBrite. In response, a promotional banner highlighting the RSVP program was introduced under a fresh, color-themed navigation bar. A hero section was created to explain the application's purpose, paired with a timed image carousel showcasing diverse events. The page now features an icon-accompanied section outlining the app's features, and includes multiple buttons for user-friendly navigation to the login section, where users can log in or create an account.



Original Landing Page

EventNXT Login



At **EventNXT**, we understand that planning an event is a complex task. Managing guest lists, ticket sales, and RSVPs can be overwhelming and stressful. That's why we have created a comprehensive tool that automates the guest list management process and streamlines ticket sales, so you can focus on planning the perfect event.

Login / Register

Features

Guest List Management

Automates the management of guest lists, making it easier to keep track of RSVPs and ticket sales.

Easy RSVP Tracking

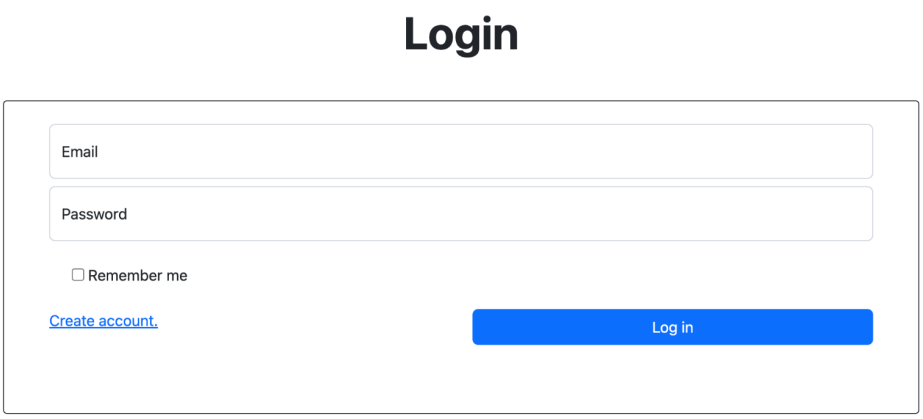
Automates the tracking of RSVPs with guests, reducing the need for manual communication and streamlining the process.

Box Office Integration

Integrates with box office software to automatically accumulate ticket sales data in a spreadsheet, reducing manual data entry and streamlining the process.

Referral Rewards

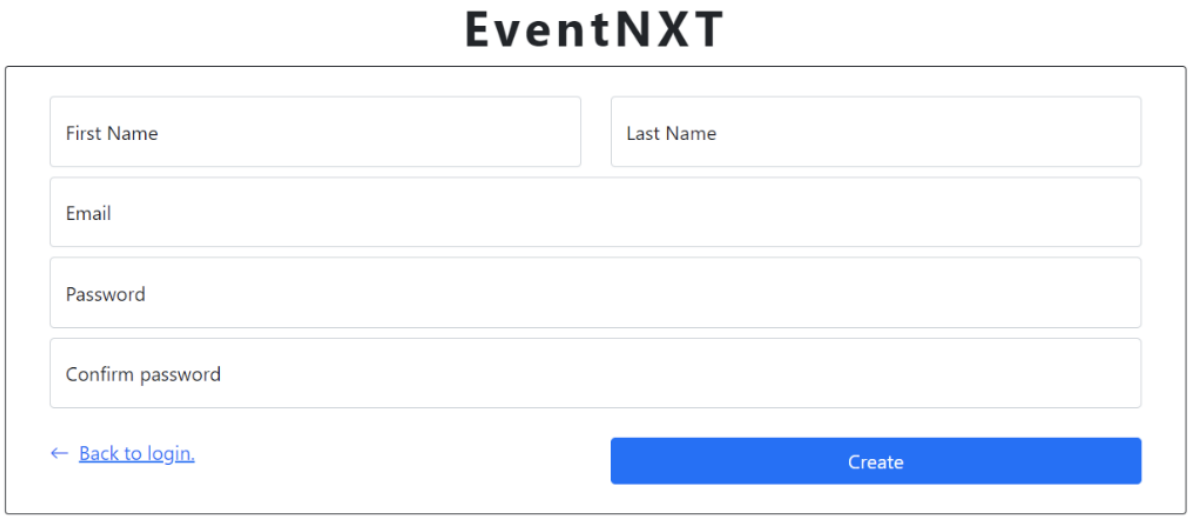
Incorporates a unique way to manage referral reward programs, encouraging guests to spread the word about events and increase ticket sales.



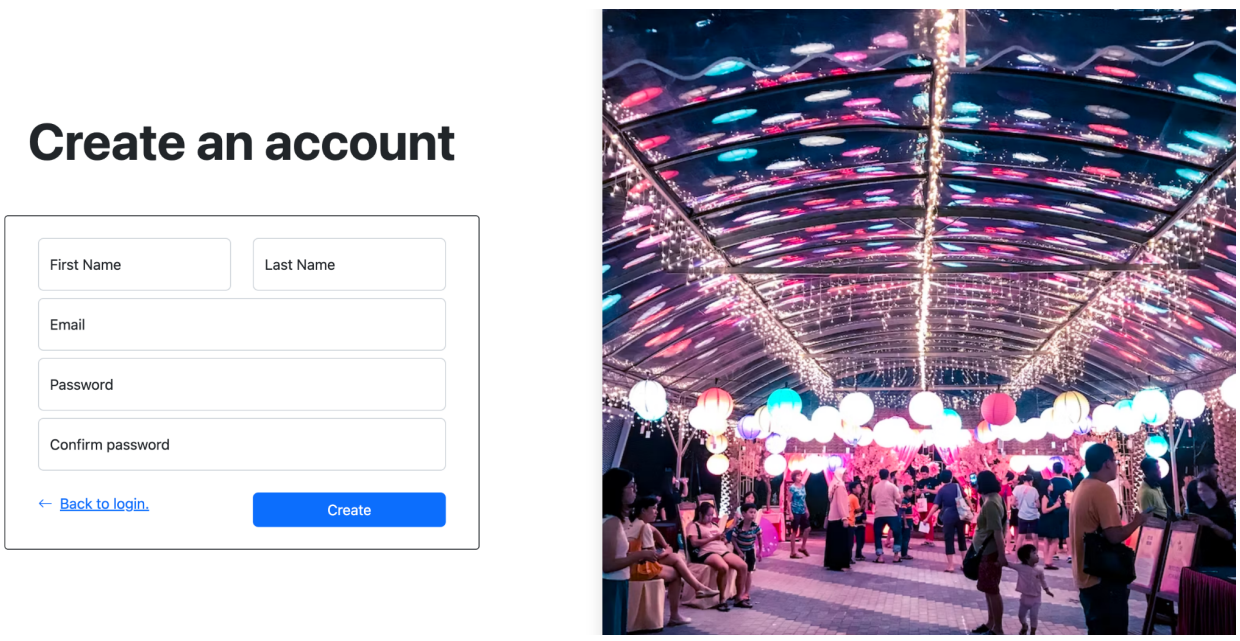
New Landing Page

Registration Page Enhancement (1 point):

The registration page was enriched with a clear title indicating account creation capability. It now splits the screen between the registration form and an event image. In the mobile view, the form appears at the top with the image below, ensuring an enjoyable mobile experience.



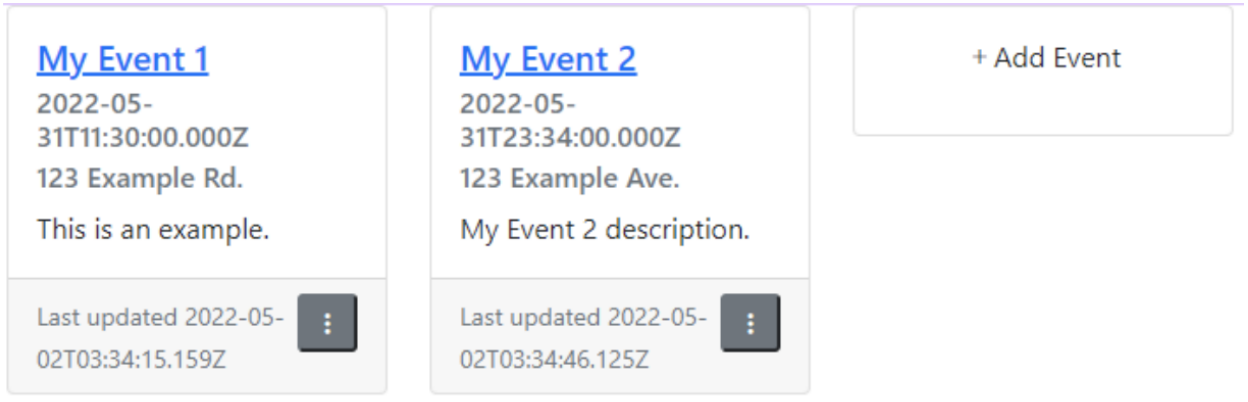
Original Registration Page



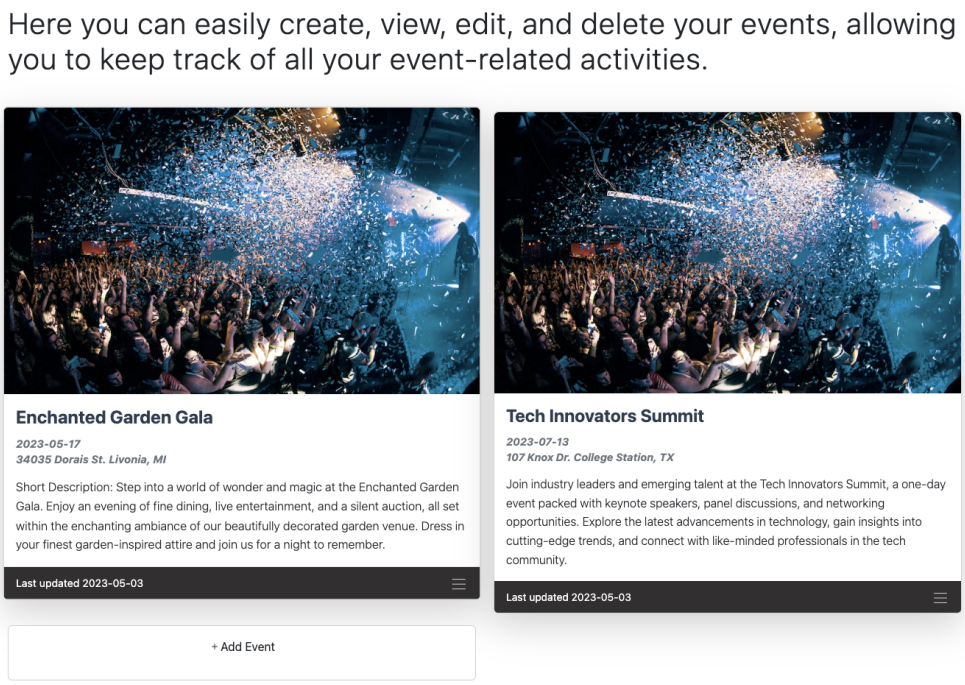
New Registration Page

Events Page Revamp and Mobile Optimization (1 point):

The original events page displayed a user's events in three-column rows, which, while suitable for larger screens, was challenging for mobile users. The revamped page displays one event per row on mobile devices with two events per row on larger devices. Bootstrap5 cards were introduced to add visual appeal, with each card functioning as a navigational button to the event detail page. A fallback image was incorporated to address instances of user-uploaded images failing to load correctly, a feature which could be further developed. The card enlarges on hover, indicating to users that they can click it to proceed. Finally, an animated header with a page description was added, outlining the available actions: adding, editing, and deleting events.



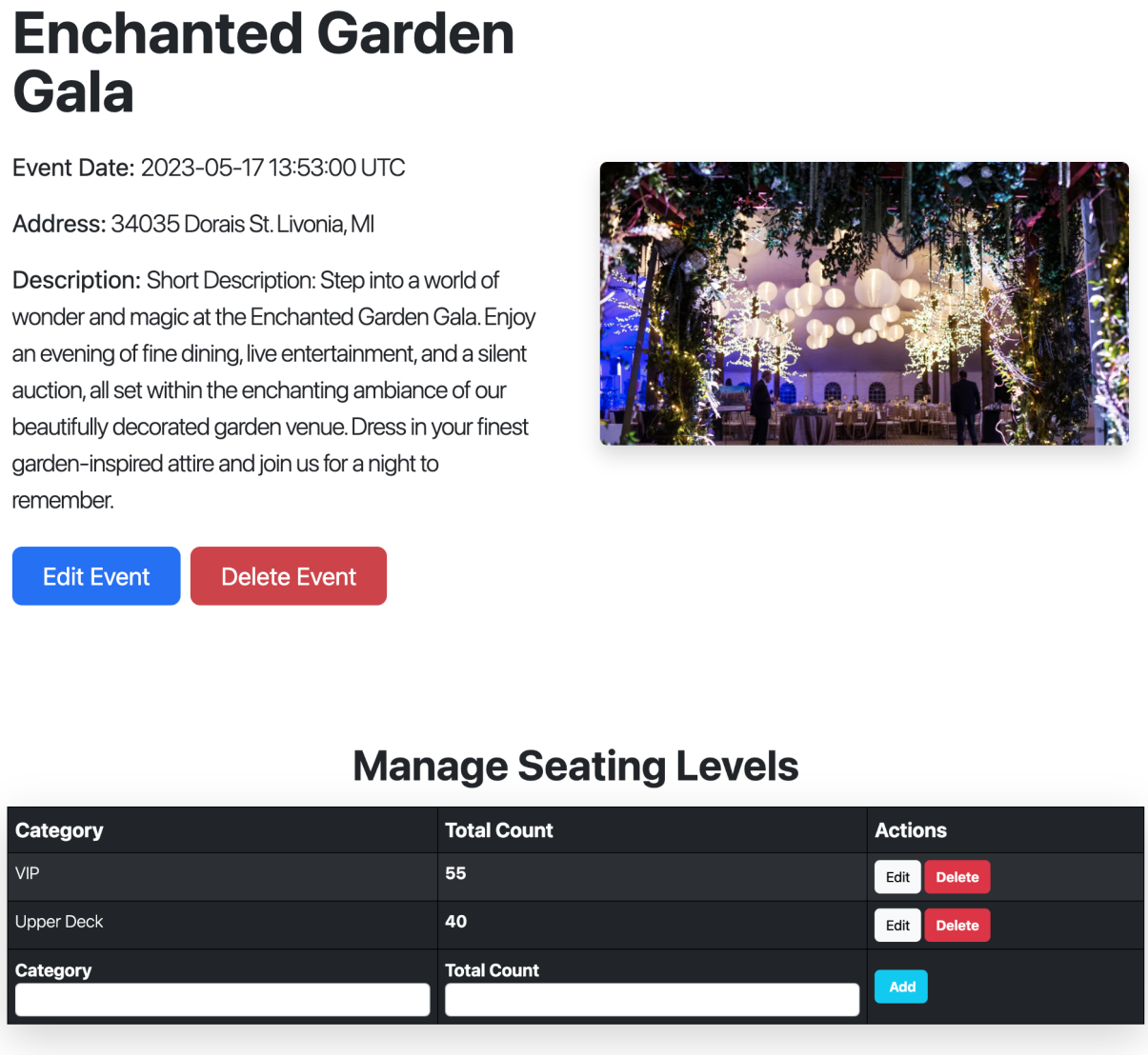
Old Events Page



New Events Page

Event Detail Page Overhaul (1 point):

The event details page now features a hero section that displays the event image on the right and its description on the left. Enhanced buttons have been added for event editing or deletion, designed responsively for mobile viewing. The 'Manage Seating Levels' table was entirely redesigned for better visual appeal. Although the plan was to redo all tables, functionality improvements took precedence, pushing the front-end changes for the remaining tables to future updates.



New Events Detail Page Content

Backend-End Improvements

Import Box Office Data (2 points):

As a user, I want to be able to import box-office spreadsheets from the main ticket sellers including Ticketmaster, Tickettomato, and Eventbrite.

The import feature is implemented and will be imported from Ticketmaster and Eventbrite. Tickettomato needs to be added. Feature was first written to work only with Ticketmaster. However, when other ticketing websites needed to be added the code was rewritten and

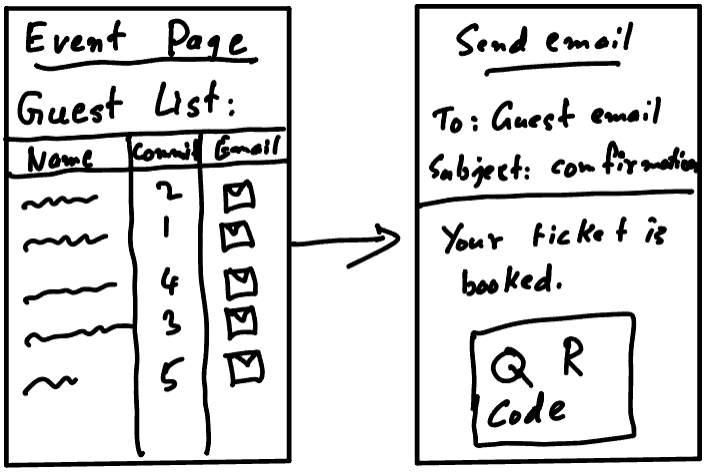
organized in a way that would allow us the programmers to update the feature by adding and removing ticketing websites with ease.
This is how the feature works in our application.

- app/views/guests/newimport.html.erb Allow users to input API key, EventID, and Ticketing Website. Input is sent to the guest controller to the import method.
- app/controllers/guest_controller.rb Uses input to get a list of guests from the website. The list of guests is sent to guests_import.
- app/models/guests_import.rb The add_guests method goes through the guest list and creates a guest calling the guest model to create a guest object. Guests are saved to the database using an automatic database save method that would be guest.save.
- App/model/guest.rb Creates new guest objects from API import.

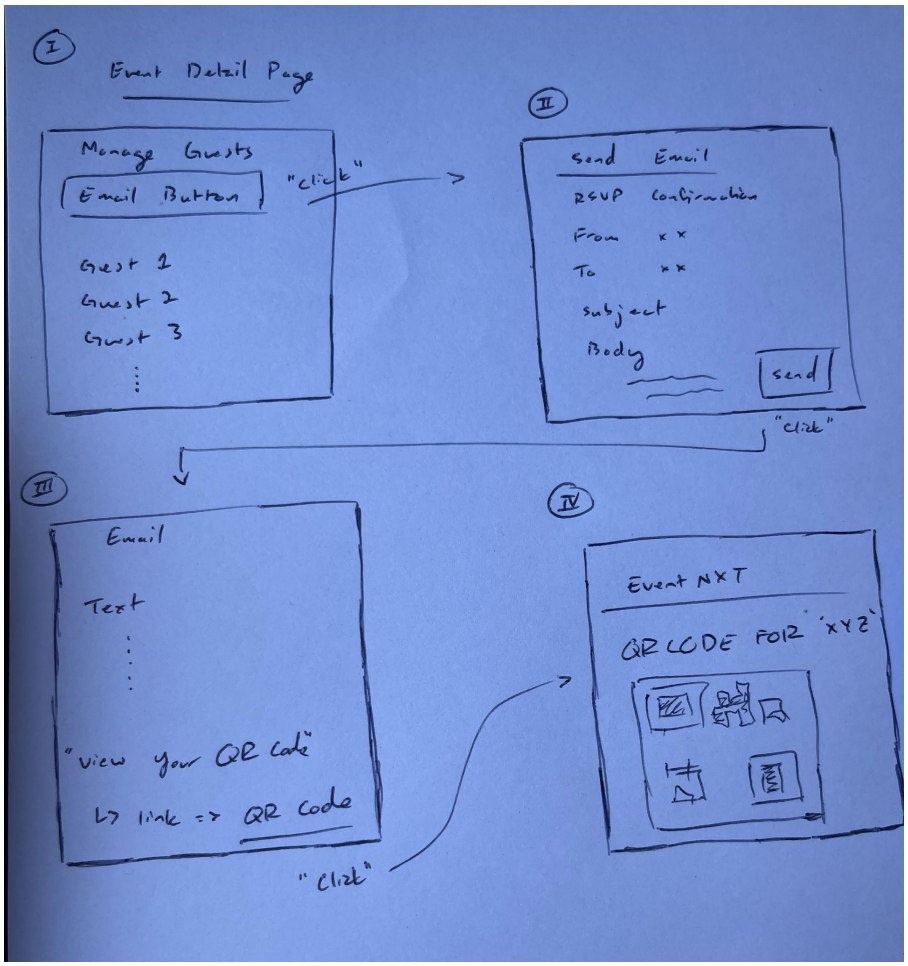
QR Code (2 points):

As a user I would like to be able to send QR codes to guests via RSVP Confirmation emails

This feature introduces a streamlined process for event entry and ticket confirmation through the use of QR codes. When a user adds a guest to an event's guest list, a unique QR code is automatically generated and stored in the database for that guest. This facilitates a simple yet effective method of guest identification and ticket verification.
The RSVP confirmation email template has been updated to include a unique URL specific to each guest. This URL directs the guest to a webpage that displays their personal QR code. When the guest clicks on this link, a request is sent to the server to load the webpage containing their QR code.
This QR code serves dual purposes - it facilitates easy access to the event and also acts as a confirmation of the guest's ticket purchase. The ease of displaying the QR code on a mobile device streamlines the event entry process and reduces the need for physical ticket checks.
This feature enhances the guest experience by providing a seamless, digital-first approach to event attendance and ticket confirmation, reinforcing the efficiency and user-friendly nature of the event management system.
This user story was altered during the development process. Originally the email was going to include the actual QR code image itself. However, with the email service we are currently using, there was difficulty in displaying certain images. The process was then altered to include a link to the QR code which can then let the customer view the code in a web browser. This ensured that no matter what email software is being used the QR code can be sent properly.



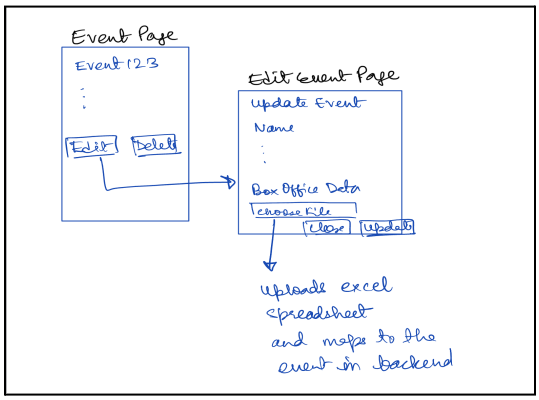
1st Lo-fi Mockup



Revised Lo-fi Mockup

Edit Event - Upload Box Office Data (1 point):

A user can edit an event and upload the box office spreadsheet from that page. In the previous app users could only upload box office data while creating a new event.

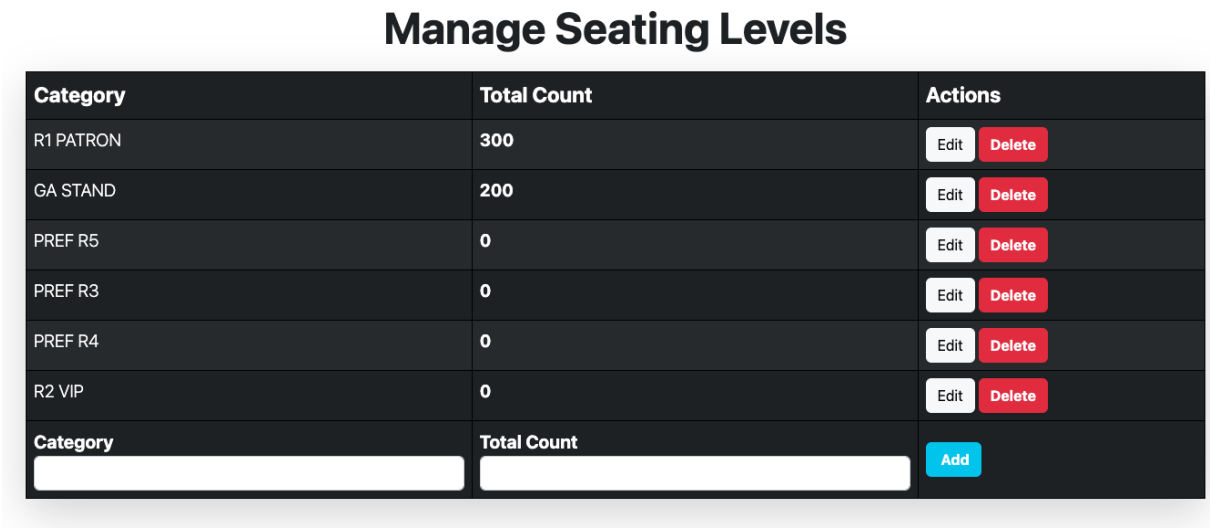


Lo-fi mockup

Manage Seating Levels - Auto populate from Box office data (2 points):

The user can edit the category and enter the total number of seats in that category, by default it is 0.

Previously, users had to manually add seating categories in this section, and ensure that the category name is the same as seating categories in the box office data.
Now, when a user uploads box office data in any event, the system will automatically parse all the seating categories in it and display them under Manage Seating Levels section. The user can add/edit/delete the seat categories from the Manage Seating Levels section. Then the user can edit the category and enter the total number of seats in that category, by default it is 0.
When event organizers re-uploads the box office data, the system will create only new seating categories in this section which were not present previously. So, the user doesn't have to manually do the same task each time he/she uploads the box office data.



<u>Seating Summary Information (2 point)</u>:
Tentative available seats are calculated as (total seats - seats sold - seats allotted to all guests). - (1 point)

Also, when a user creates a new seating category through Manage Seating it will also reflect in the Seating Summary section. - (1 point)

This section provides an overview of the seating categories and the number of tickets allotted to guests in each category. It displays the total number of seats in each category, as well as the number of seats that have been committed to guests and the number of seats that are still available.
Additionally, it shows the number of tentative free seats based on the allotted count, which gives event organizers an idea of how many more tickets can be sold in each category. Tentative available seats are calculated as (total seats - seats sold - seats allotted to all guests).
This information helps organizers manage their seating inventory effectively and make informed decisions about ticket sales.

Seating Summary Information

Category	Total Seats	VIP Guests	Total Allotted	Total Committed	Total Sold	Free Seats	Tentative Free Seats (based on seats allotted to guests)
R1 PATRON	300	2	7	0	252	48	41
GA STAND	200	0	0	0	126	74	74
PREF R5	0	0	0	0	73	-73	-73
PREF R3	0	0	0	0	118	-118	-118
PREF R4	0	0	0	0	85	-85	-85
R2 VIP	0	0	0	0	80	-80	-80

Looking at the above screenshot, there are some categories for which free seats and tentative free seats are negative values, that is because the user has not entered the total number of seats in Manage Seating Levels for those categories.

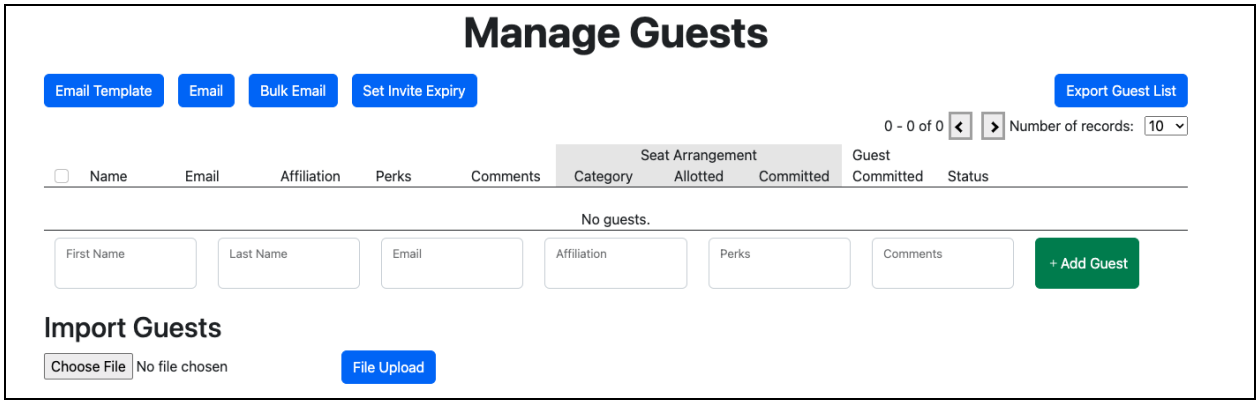
Manage Guests - Export Guest List (1 point):

In the Manage Guests section, users can allocate and commit seat counts for each seating category, and if a guest is assigned to multiple seat categories, the export file will contain rows for each category.

Before, when users exported the guest information, the seating information was not included in the exported file. However, we have now added this functionality. In the Manage Guests section, users can allocate and commit seat counts for each seating category, and if a guest is assigned to multiple seat categories, the export file will contain rows for each category.
This allows users to have a more comprehensive and organized view of the seating information when exporting guest data.

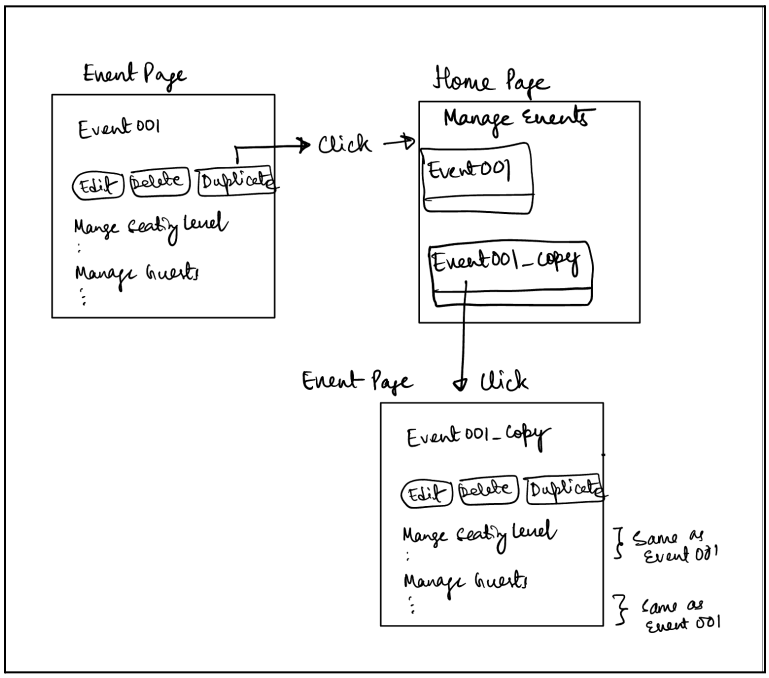
Manage Guests - Import Guest using file upload (1 point):

A user can import a csv file to add guests to an event. The application will check if any of the email addresses in the imported file already exist in the current event's guest list. If an email address already exists, the system will skip creating a new guest record for that email and move on to the next record in the csv file. This ensures that there are no duplicate guest entries in the event's guest list.
The csv file should have three columns: 'first_name', 'last_name', and 'email'.
Users can upload a guest list exported from other events, or any other csv file with these 3 columns in it.



Duplicate Event (1 point):

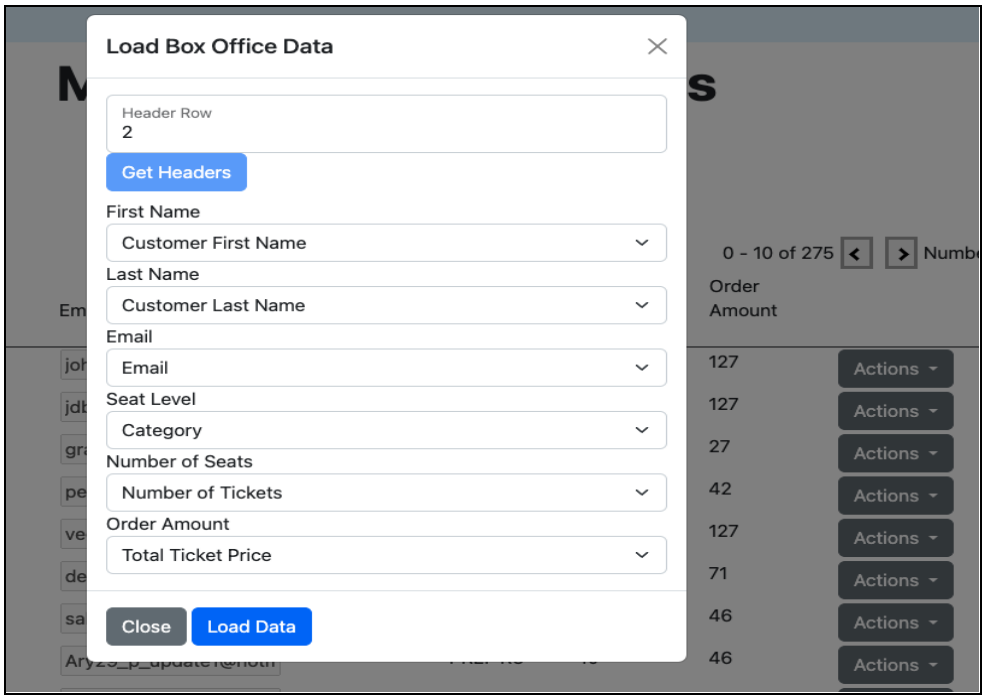
A user can duplicate an event and a new event will be created with the same event details - date, address, description, seating categories, and guests.



Lo fi mockup

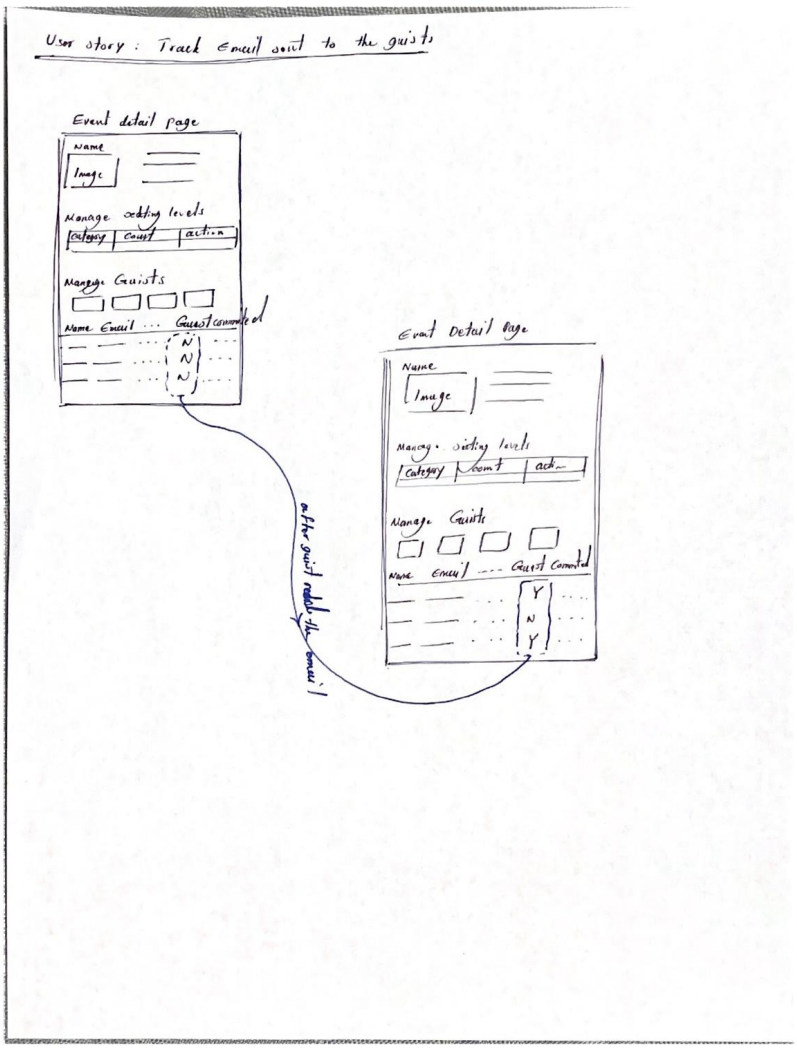
Manage Box Office Sales (1 point):

We added a new column in this section to read ticket price/amount from the box office data. When users click on Load Box Office Data, they will see a field named Order Amount, here they can select the appropriate column name from the box office spreadsheet.



<u>Sending Email Invites (1 point)</u>:

As an event organizer, I want to be able to send invitations to an event to guests through email and provide purchase confirmations. This functionality had been previously implemented, but it was not functioning when we deployed the app. The email functionality currently implemented is configured to send emails through a specific G-mail account. This requires the email ID and a special key from G-mail that allows third party apps to send emails through this account. These details are provided in the config/environments directory.

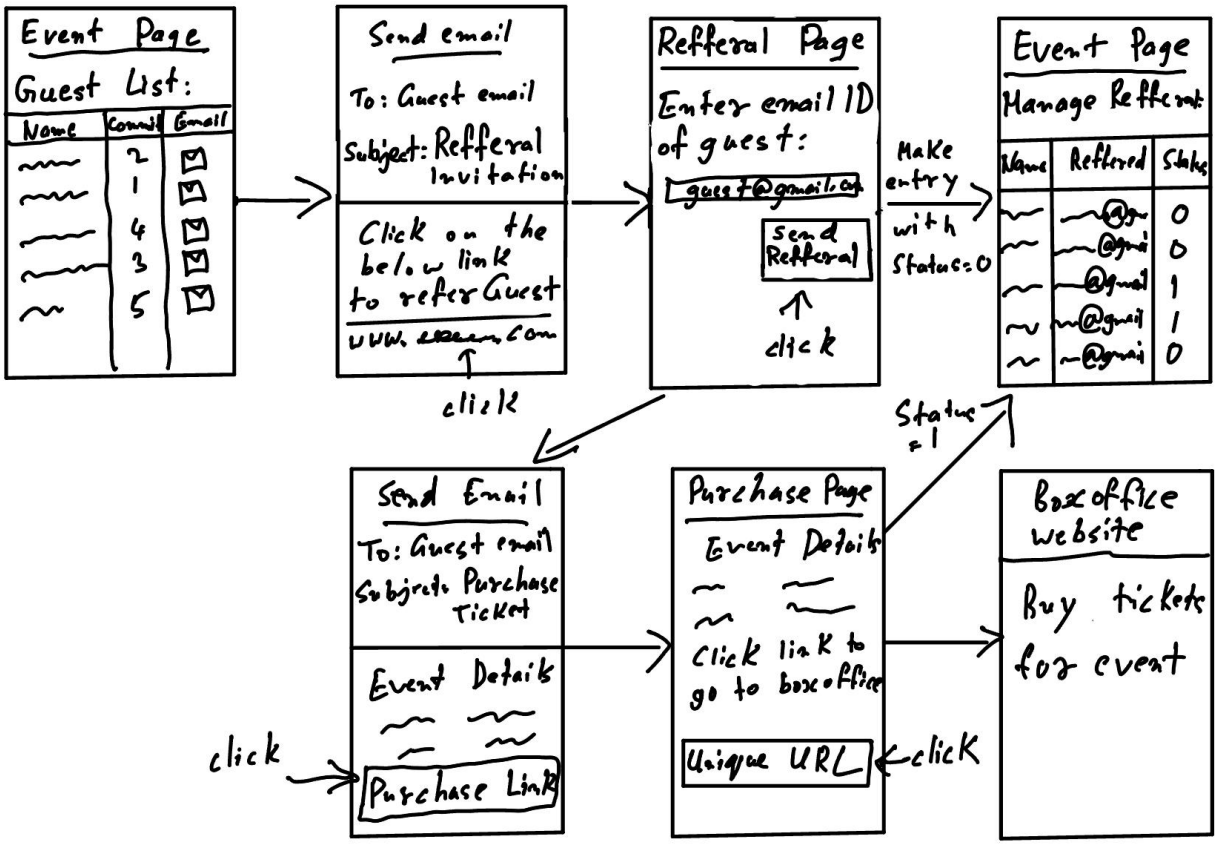


Referral Invitation System (3 points) :

As an event organizer, I want to make it possible for guests to refer other guests who would be interested in purchasing tickets to the event. The event organizer should also be able to track the referrals being sent out and if the referral links have been used to make a purchase. Previously, the email template to send the referral invitation and the view to enter the referred guest email had been implemented. But the functionality to send that user a purchase link and track the referrals had not been implemented. This user story is given 3 points as it is slightly tricky to propagate the necessary event and guest parameters over 2 consecutive emails.

The referral invitation system functions as follows:

- The event organizer can send a “Referral Invitation” email to any of the invited guests. This email consists of a link to a page in our app where the guest can enter the email ID of someone they wish to refer.
- Upon entering an email ID, a “Purchase ticket” email is sent to that guest consisting of a link to purchase tickets and an entry is made on the events “Manage Referrals” table that a referral invitation has been sent by an invited guest to the entered email ID.
- The link in the “Purchase ticket” email redirects the user to a page in our app that consists of a unique referral URL. Upon clicking this link, the “Manage Referrals” table is updated to show that the referred guest has clicked the link.
- In future iterations, clicking on the unique link should also redirect the user to the boxoffice website where they can purchase the tickets.

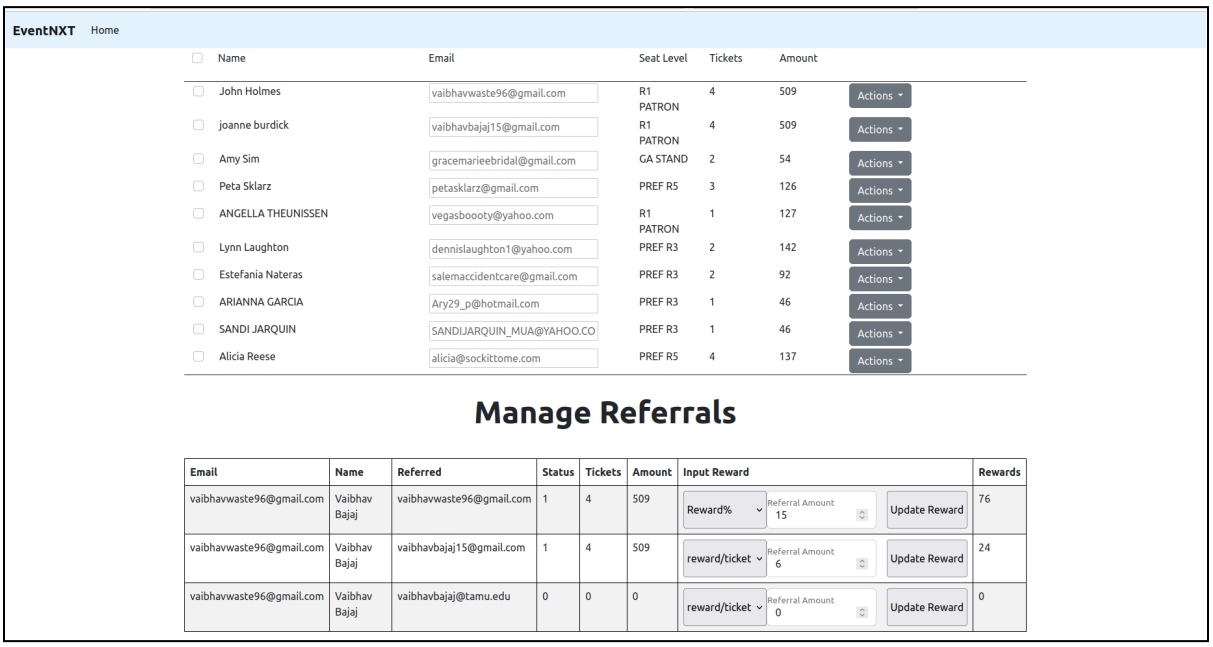


Referral Rewards (3 points):

As an event organizer, I want to incentivize guests to refer more people by providing them with referral rewards. I want to be able to provide rewards to a guest either based on the number of tickets bought by the referred guest or the total amount they spent. I would also like the freedom to decide and modify the amount that should be given to each guest as a reward. If someone receives multiple referral links, only the guest who’s link was clicked first gets the reward.

The referral rewards function as follows:

- When the status of a referral is 1 and the box-office data is updated, the controller parses through the box-office data to find the number of tickets purchased by the referred guest and their total amount.
- We update the “Manage Referrals table with this information.
- The event organizer can then decide to reward the guest by either providing a reward per ticket or the reward can be a percentage of the amount spent.
- In future iterations, the event manager should be restricted from sending the referral reward to a guest

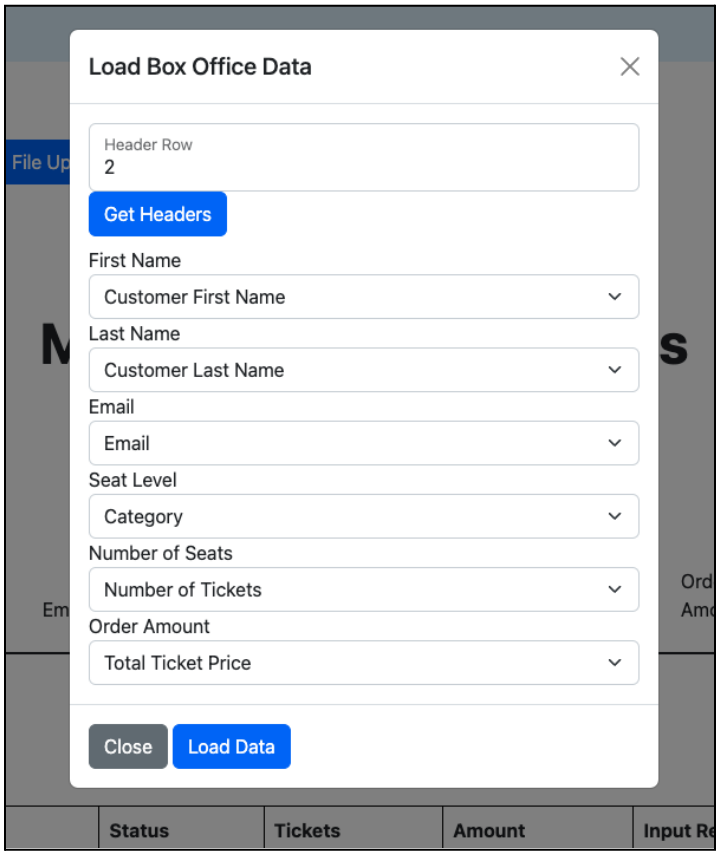


3. Project Details

3.1. Legacy App

The current codebase is developed using the Ruby on Rails framework and utilizes a considerable amount of JavaScript logic for its core functionalities. Some tags present in the View pages are sourced from JavaScript controllers, hence it would be beneficial to review them. Additionally, some files in the controllers/api/v1 directory are also used in specific scenarios, so they should be inspected when necessary.

During our development, we found that when updating text or other elements in email templates, the changes were not immediately reflected in the application. To see the changes, new events needed to be created since the templates were set during the event creation process.



To load box office data into the EventNXT application, the user must first upload the spreadsheet provided by the client. This can be done either during event creation or by clicking on "edit

event" and uploading the box office data. Once the spreadsheet is uploaded, the user can go to "Manage Box Office Sales" and click on "load box office data." The user will then need to enter the header row value, which corresponds to the row in the spreadsheet where the header is located (e.g., 2 for the second row). After clicking on "get headers," the application will read the header data from the spreadsheet, and the user can proceed to map the column names to the appropriate fields using the provided screenshot. Finally, clicking on "Load Data" will load the box office data into the application. To view the updated information, the user should refresh the page, and the box office data will be displayed under "Manage Box Office Sales," with seating categories and tickets sold reflected under "Seating Summary Information."

Apart from the newly added features or bug fixes, we have not made any major changes in the structure/implementation of this application. We have provided the steps to deploy the app in the local machine as well as heroku and they should work without any changes.

3.2. New App

The major fault of this legacy app has been faulty authentication, which allowed every user to visit the event dashboard regardless of being logged in, i.e., lack of security. In addition, it was not possible to connect the app to the central authentication developed by the CRM team. After several attempts and meetings with the instructor, we decided to redesign and build a new Ruby-on-Rails app from scratch, which not only imitates all the existing features of the old app, but all also provides the following benefits:

- Fully functional authentication
- Secure app, i.e., no access to any page unless the user is signed into the app.
- Multiple registration methods, including EventNXT-based registration and Google registration, which allows users to register via their gmail account.
- Easy integration to the CRM's central authentication system
- Faster operation
- Using only Ruby language, i.e., no hacky implementation
- No Javascript implementation
- Replicable features, i.e., similar features have similar implementations
- Comprehensive explanation of changes made to each line of the application.

On top of all these benefits, we only focused on implementing features instead of spending time on styling with CSS or Bootstrap. We believe that front-end changes can be done when all features are implemented. Since there are still some features remaining to be completed, we did not add any front-end styling to the new app to make it easier for the next team to understand the codes.

How to Load the new Application

The following simple steps should be taken to deploy the app into your local machine. Same steps can be used for Amazon Web Service (AWS) Cloud-9 as well.

1. Check if Ruby, Rails, and Bundle are installed

```
$ ruby -v
$ rails -v
$ bundle -v
```

2. If you are using AWS-Cloud9, you can install all these by running the following commands. Please make sure you're not entered the app folder yet. Run the following command in your created environment, before entering your app's local folder.

```
# For Amazon Linux:
$ sudo yum -y update
$ sudo yum -y install ruby

# Update RVM by running:
$ rvm get stable

# Install ruby version:
$ rvm install 3.1.2

# Set default version in RVM
$ rvm --default use 3.1.2

# Update/install bundler
$ gem install bundler

# Update/install rails
$ gem install rails
```

3. Now, you can clone the app's GitHub repo and run the app on your local server.

```
$ git clone <github repo>
$ cd <app name>
$ bundle install
$ rails db:migrate
$ rails s
```

4. If you are using AWS-Cloud9, you will get access to the web app via "Preview". You will probably be asked to add the host address to your app's environment. For example, in the case of development, you may put the following code line in your "development.rb" located at "config/environments/developement.rb". Change the URL with your own AWS cloud-9 url. Now, you will be able to rerun the server and preview the app.

```
config.hosts << "da8311—-.vfs.cloud9.us-east-2.amazonaws.com"
```

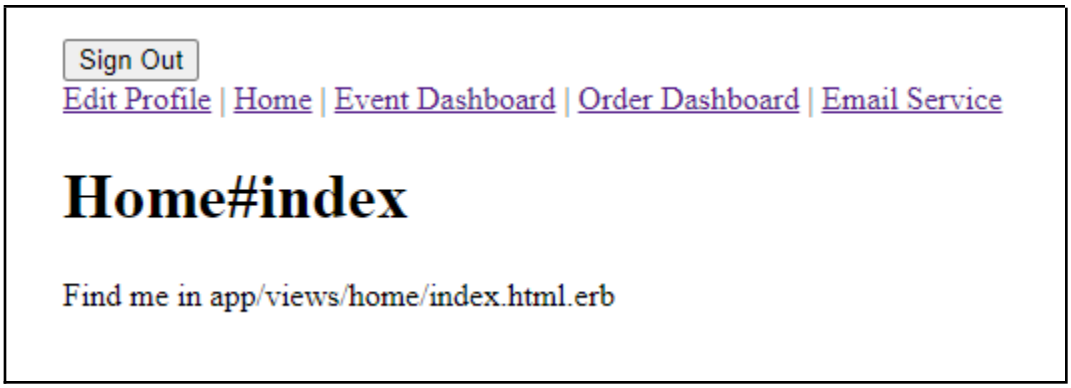
Implemented Features in the New Application

1. **Navigation Bar (2 points):** the navigation bar shows only "Sign-In" and "Sign-Up" options to the users when the user is not signed-in yet. The nav bar changes to "Edit-Profile", "Sign-Out", and extra links to other pages of the app once the user is

logged-in. A partial, "_user_detail.html.erb" is used to handle that. The partial can be found in "app/views/shared".

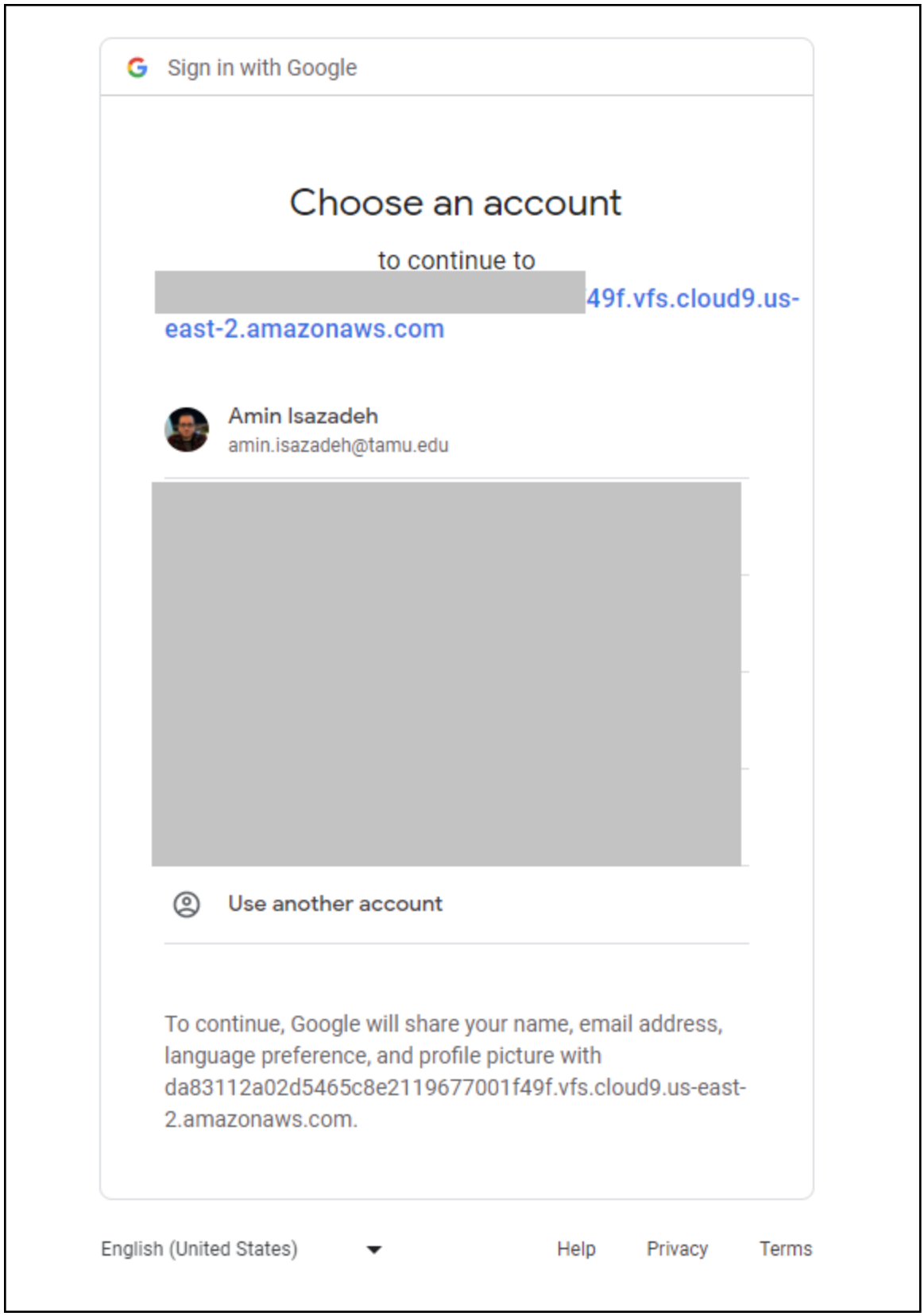
a. Before Sign-In

b. After Sign-In



2. **Devise Authentication (3 point):** famous "Devise" gem is used to implement a solid authentication system. You can find more information about how to use it in their GitHub repository. The authentication is deployed in a way that does not allow any user to enter the app before being authenticated.

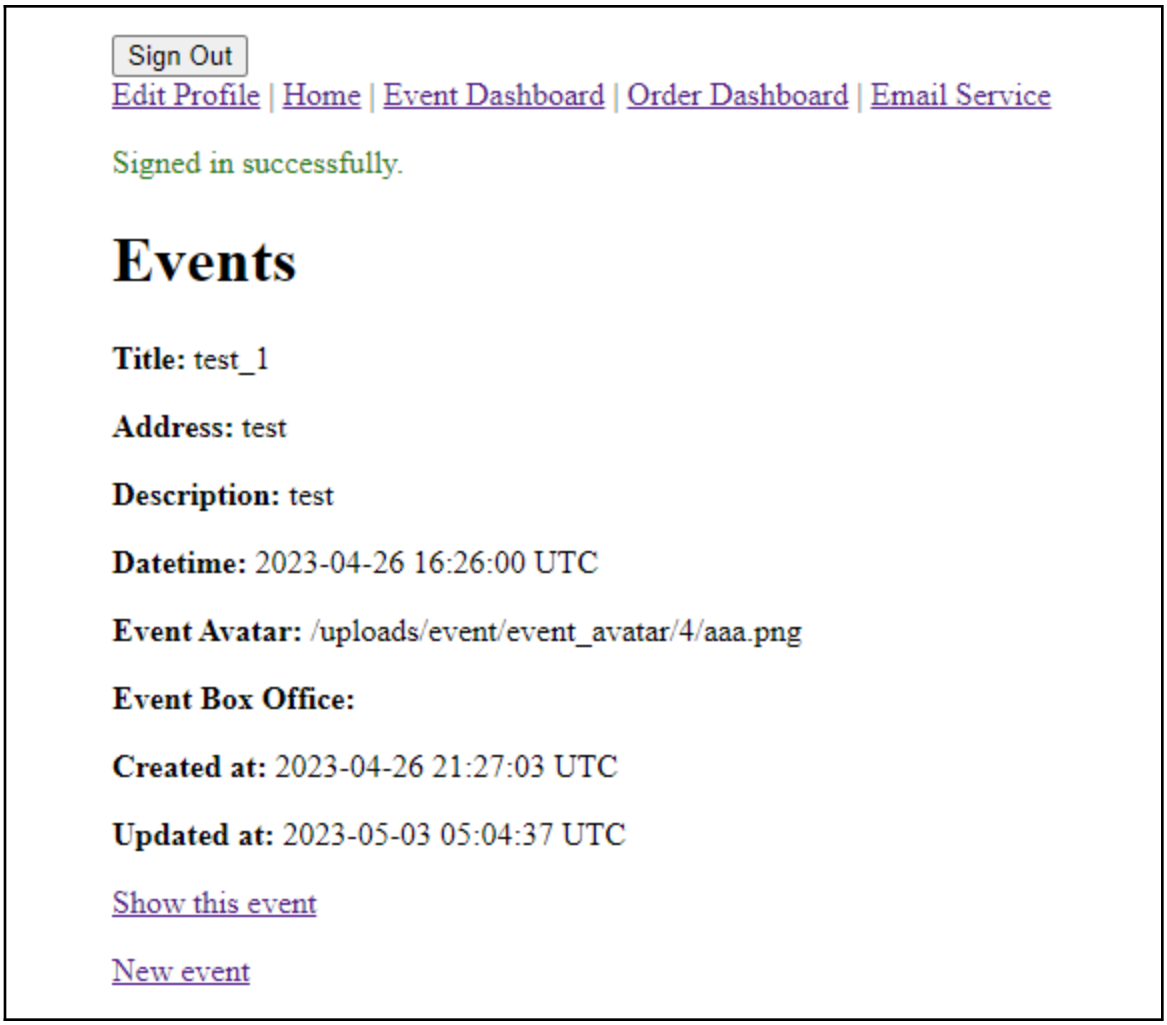
3. **Google Oauth-2 Authentication (2 points):** to give more flexibility to our users, we implemented Google authentication, which is the most common type of authentication nowadays. The user will be redirected to the google page to choose any of their google accounts and then they will be redirected back to the home page. Regardless of what method of authentication will be taken, the user information such as email will be stored in the User model and can be accessed. Already, the user model does not include "first_name" and "last_name". Once these fields are added to the user model, more information from the user authentication can be retrieved from their Google account and can be stored in the database.


Sign in with Google
Choose an account
to continue to
[redacted]49f.vfs.cloud9.us-east-2.amazonaws.com
Amin Isazadeh
amin.isazadeh@tamu.edu
[redacted]
Use another account
To continue, Google will share your name, email address, language preference, and profile picture with da83112a02d5465c8e2119677001f49f.vfs.cloud9.us-east-2.amazonaws.com.
English (United States)
Help
Privacy
Terms

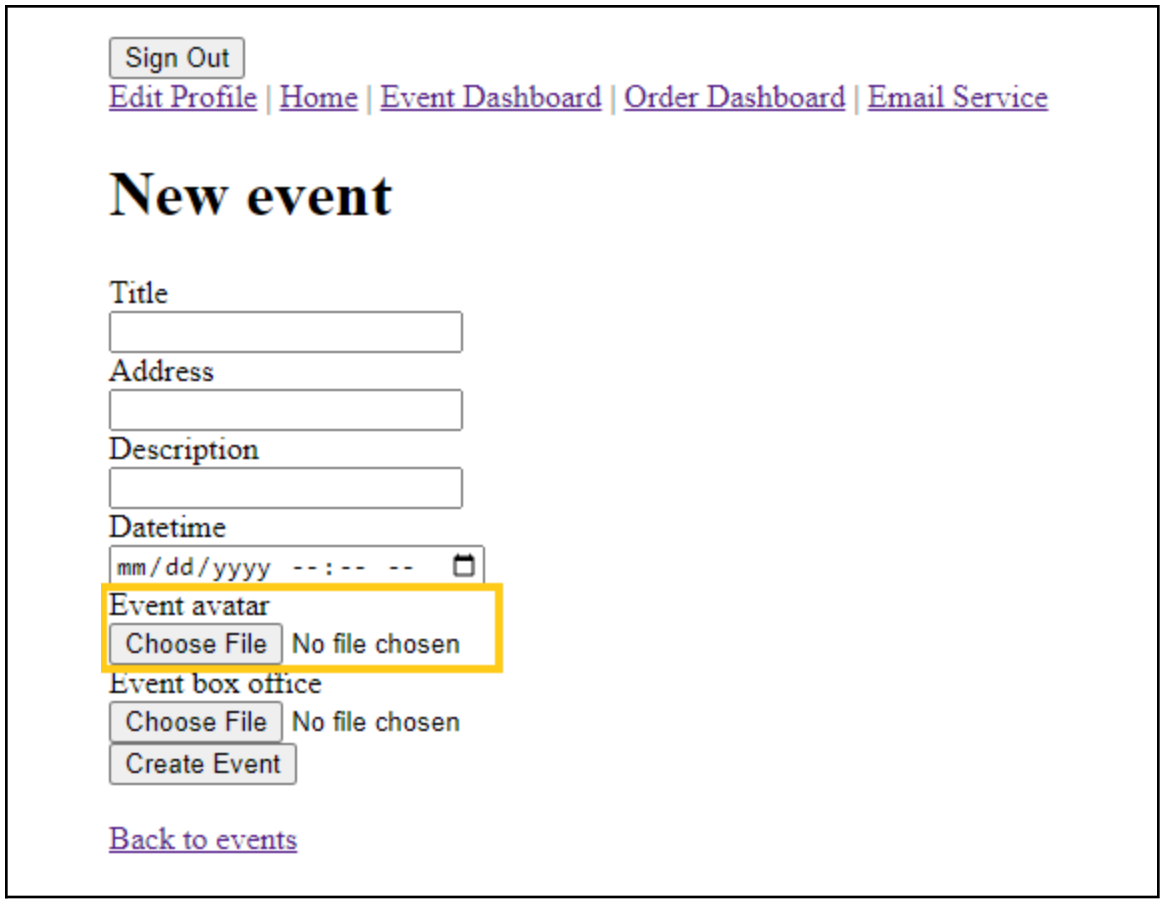
4. **Home Page (1 point):** the landing page of the new app is called “Home” (which was called “Welcome” in the old app). The page was simply created by the following command. No further change is made to this page.

```
$ rails g controller Home index
```

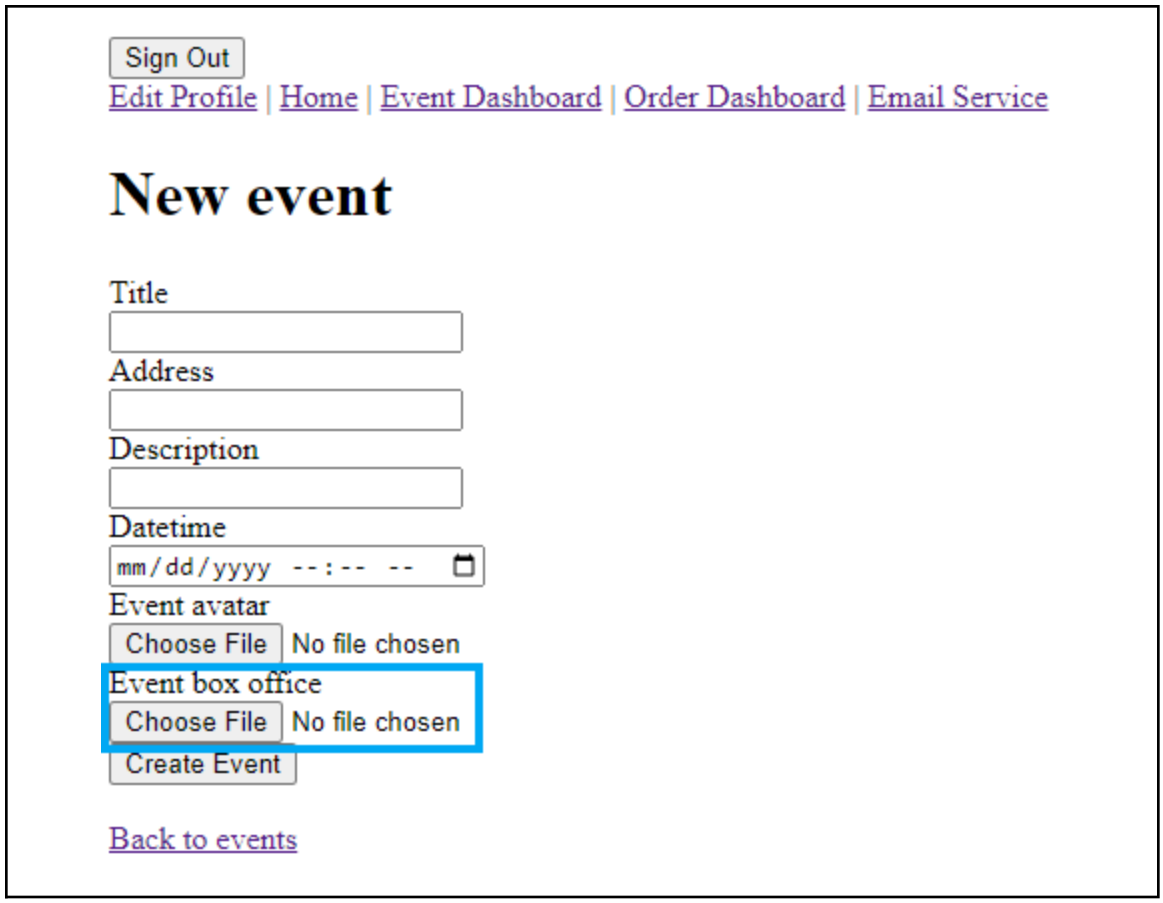
5. **Event Dashboard (2 points):** Event page is created to allow users to create event specific details. The page is a CRUD page and created by the “$ rails scaffold” command.



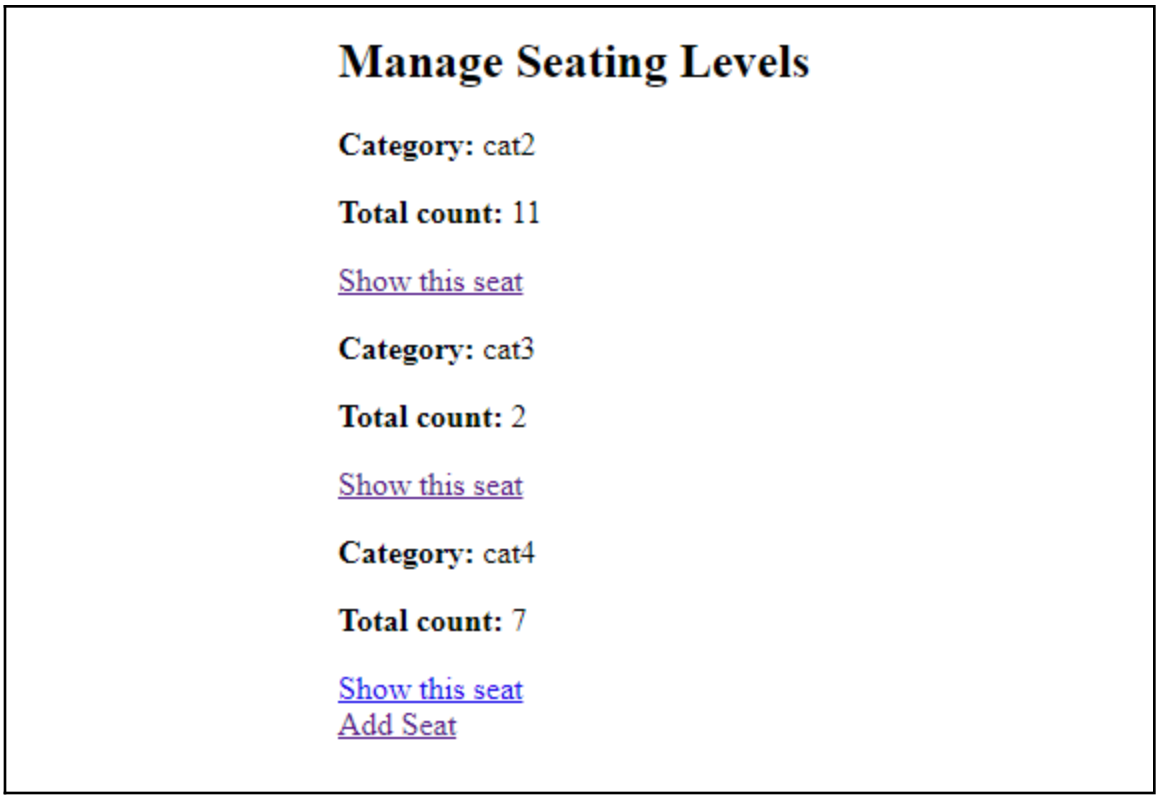
6. **Image Uploader (2 points):** a powerful gem, “carrierwave”, is used to build a helper method for uploading any image. The image uploader is named “avatar_uploader.rb” located at “app/uploaders”. As an example, users are able to add images to each event when they are creating them. Same uploader can be used in any other part of the app. Just follow the same implementation process used for adding avatar to each event. “carrierwave” supports the “mini_magick” gem, which can be used to resize the rendered image. You will find tons of tutorials online on how to use carrierwave for image loading and modifying..



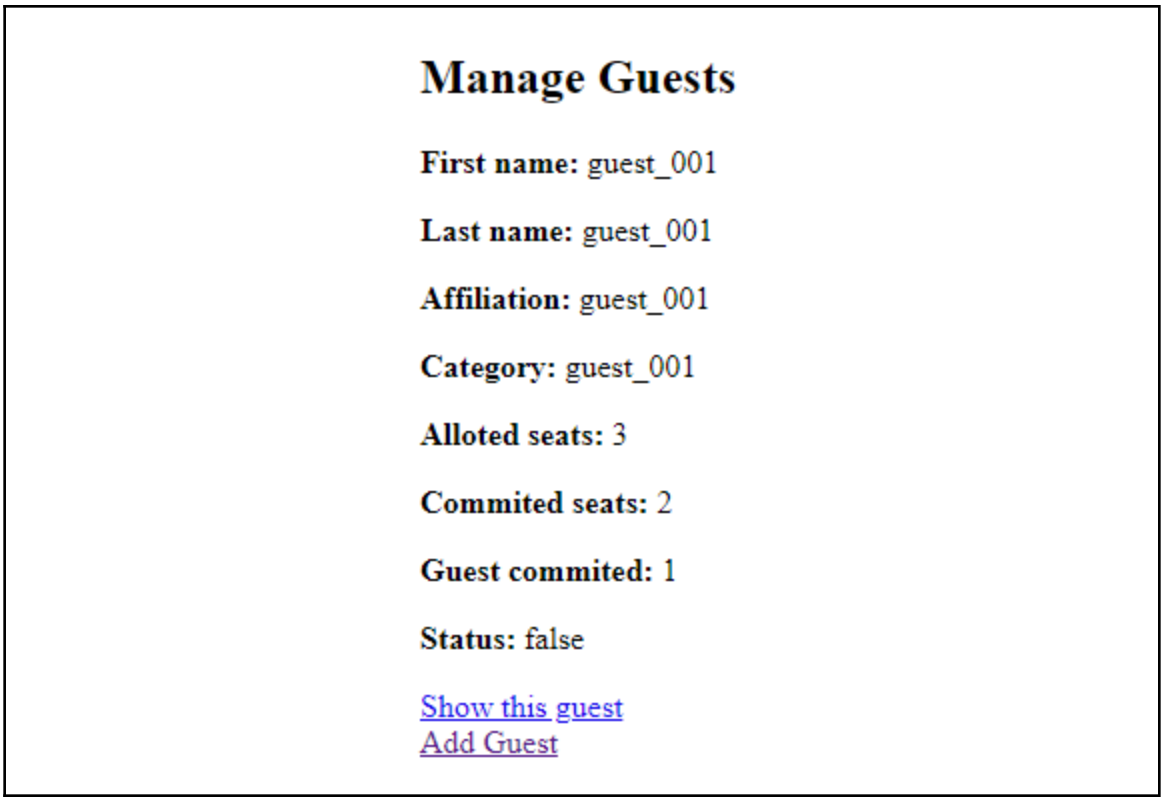
7. **Spreadsheet Uploader (2 points):** same gem, "carrierwave", is used to build a helper method for uploading any spreadsheet. The spreadsheet uploader is named "spreadsheet_uploader.rb" located at "app/uploaders". As an example, users are able to add box office data spreadsheets to each event when they are creating them. Same uploader can be used in any other part of the app. Just follow the same implementation process used for adding spreadsheets to each event. Once you enter each event, you will see the uploaded spreadsheet is parsed into the event. You can modify how many rows of the data sheet you want to display. Follow the tutorial of "carrierwave" on how to implement spreadsheet uploading and parsing.



8. **Manage Seating Levels (2 points):** a nested scaffold CRUD table is added to each event so that every user can create event-specific tables for seating categories. The Event model and the Seat Model share a foreign key. You can find the Seating table in the show page of each event. You will be able to do all the CRUD operations for this nested table.



9. **Manage Guests (2 points):** same pattern as the previous one (seating table) was followed to create another nested scaffold for managing guests. The guest table can be found inside the show page of each event. The Event and the Guest models share a foreign key.



10. **Mailer System (6 points):** the old app allowed the event organizer to send individual or bulk email to the guests. A simple mailer system is activated in the app that uses the Rails built-in Action Mailer. To test its functionality, we created a CRUD page, called "Email Service", where users are allowed to send emails with a certain subject and email body. Then the index page filters the sent email from unsent emails. The index page shows which emails are not sent yet. In addition, extra information about what time the email was drafted, last updated, and sent will be shown for each created email. An extra attribute "commited_at" needs to be implemented to show what time the email recipient opened the email. More functionalities can be added based on the client's requirements.

Sign Out

Edit Profile | Home | Event Dashboard | Email Service

Email Service

Compose New Email

****

Unsent Emails

To: amin.isazadeh@tamu.edu

Subject: Get Ready to Party: You're Invited to the Ultimate Celebration!

drafted at: 2023-05-04 18:44:06 UTC | **updated at:** 2023-05-04 18:44:06 UTC | **sent at:** | **commited at:**

Body: Dear [Name], We're thrilled to invite you to the most amazing party of the year! It's going to be a night to remember, with great r
fun and laughter. So mark your calendar and get ready to join us for the ultimate celebration! We can't wait to see you there. Please RSV

Show this email

****

Sent Emails

To: amin.isazadeh@tamu.edu

Subject: Invitation to our Company's Annual Fundraising Gala

drafted at: 2023-05-04 07:02:47 UTC | **updated at:** 2023-05-04 18:44:52 UTC | **sent at:** 2023-05-04 18:01:22 UTC | **commited at:**

Body: Dear [Recipient's Name], We are delighted to invite you to our company's Annual Fundraising Gala, which will be held on Satur
all proceeds from the event will be donated to support education programs for underprivileged children in our community. The evening
esteemed speakers who will share their inspiring stories of how education has transformed their lives. We would be honored if you coul
them online through the link provided below. [Insert Ticket Purchase Link] We appreciate your support and look forward to your presen

Show this email

To: amin.isazadeh@tamu.edu

Sign Out

Edit Profile | Home | Event Dashboard | Email Service

To: amin.isazadeh@tamu.edu

Subject: Get Ready to Party: You're Invited to the Ultimate Celebration!

drafted at: 2023-05-04 18:44:06 UTC | **updated at:** 2023-05-04 18:44:06 UTC | **sent at:** | **commited at:**

Body: Dear [Name], We're thrilled to invite you to the most amazing party of the year! It's going to be a nig
and laughter. So mark your calendar and get ready to join us for the ultimate celebration! We can't wait to se

Edit this email service | Back to email services | Send the email

Destroy this email service

Test Cases for the New Application

1. **SimpleCov:** is installed and added to the "test" and "spec" folder of the app to provide test coverage. In the case of writing Rspec tests, don't forget to add the following line to your "_spec.rb" files. You will have access to the code coverage report by opening the "index.html" file in the "coverage" folder of the app.

```
require 'rails_helper'
```

2. **Rspec-Rails:** is used as a Behaviour-Driven Development (BDD) tool to test various functionalities of the application. All Rspec test cases can be found under the "spec" folder. We already wrote Rspec tests for every implemented functionality of the app. However, some of them require modification to pass the tests; The test files with fully commented lines require modification. You will find more information about the test coverage by opening the SimpleCov report, explained above. Rspec tests are written for,
    a. Controllers (need modification)
    b. Channels
    c. Models
    d. Mailers (need modification)
    e. Helpers
    f. Jobs (need modification)
    g. Libraries
    h. Ungrouped such as image and spreadsheet uploaders (need improvement)
3. **Shoulda-Matchers:** is installed to provide Rspec-compatible matchers.
4. **Cucumber:** you need to install Cucumber to add more test cases. You can find some written Cucumber tests in the old app. They might be helpful.
5. **Built-in Tests:** the test folder of the app has several test cases provided by Rails itself during app development. You can run them by the following command. You may need to modify those files to pass the test.

```
$ rails test
$ rails test test/controllers/users_controller_test.rb
$ rails test --verbose --fail-fast
```

To-Do List for the Next Team for the New App

1. **Test Cases (Rspec and Cucumber):** no Cucumber/Rspec tests are written for the new app. The new team should develop these tests for every single feature. You can get some ideas on how to write them by looking at the tests written for the old app.

2. **Add CRM Authentication:** the new app already has two authentication methods, you need to add another option to allow users to use the central authentication developed by the CRM team.
3. **Associate Event to each User:** the app currently allows all users to see all the created events. You need to associate each event to each user that limits the signed-in users to only see their own created events, not others. You can find numerous online tutorials on how to do it.
4. **Mailer System:** the event organizers were able to send individual or bulk emails to the guests in the old app. You can find all the mailing templates in the old app. You need to transfer those into the new app. Furthermore, you should allow every event organizer to be able to send emails from their own email address. The app is currently configured in a way that only allows certain gmail addresses to send email from. You can learn more about it online by searching how to send email from gmail in the third-party app. As a hint, there are some other mailer service providers, such as “mailgun” and “sendgrid” that may be the way to allow sending emails from any email address.
5. **Referral System:** the referral system header is only placed in the show page of each event, but nothing about referral is implemented in the new app. You need to fully understand what exactly the client wants for this system and learn from the implemented referral system in the old app. Then you can implement a fully functional referral system.
6. **Inline CRUD Scaffold:** rails recently introduced a very powerful tool called “Hotwire” to create dynamic tables. From which, you can create inline editable tables inside any page of the app. It is highly suggested to learn and implement it for this app, instead of using Javascript.
7. **CSS Styling:** it is highly suggested to do styling after all features are implemented in ruby language. Styling or front-end changes can be done as the final step of building the app.
8. **Test Cases (Rspec and Cucumber):** to remind you, don’t forget to write test cases!

4. Team Roles

We had a team of 5 members: Amin Isazadeh, Jakob Kirby, Casey Quinn, Vaibhav Bajaj, Sushmita Pattanaik.

The roles for each iteration:

	Scrum Master	Product Owner
Iteration 0	Vaibhav Bajaj	Sushmita Pattanaik
Iteration 1	Vaibhav Bajaj	Sushmita Pattanaik
Iteration 2	Jakob Kirby	Amin Isazadeh
Iteration 3	Jakob Kirby	Amin Isazadeh
Iteration 4	Jakob Kirby	Amin Isazadeh
Iteration 5	Jakob Kirby	Amin Isazadeh

5. Scrum Iterations

Iteration 0: (0 points)

In the initial iteration, our team convened with the client to gain a comprehensive understanding of the project's current status and overarching objectives. This encompassed a detailed walkthrough of the entire project ecosystem, which includes multiple interrelated applications. Additionally, we delved into the existing functionality and features of the application under our purview.

At this early stage, we had yet to get the application up and running. This was primarily due to a paucity of documentation from the prior teams regarding the setup process. Furthermore, we were not fully cognizant of the numerous bugs and issues embedded in the codebase that would later require our attention.

The client elucidated several key features during our discussions. Of particular significance was the requirement for the application to integrate ticket sales data from a variety of sources, including Box Office transactions and RSVP confirmations from guests. This would facilitate an up-to-the-minute tally of remaining tickets or inventory. The client emphasized the strategic importance of this feature, explaining that when combined with analytical insights on event earnings, it could significantly enhance the effectiveness of their event promotion strategies.

Iteration 1: (4 points)

In this iteration, the team faced some challenges due to the lack of documentation which hindered the delivery of user stories. However, the team remained focused on ensuring the application was set up on local machines and deployed on Heroku, a cloud-based platform for web application deployment. This effort required significant troubleshooting and configuration to ensure the application was functioning properly.

To further enhance the application's quality, the team worked diligently to increase the test coverage by creating RSpec test cases for the existing functionalities.

Iteration 2: (2 points)

We restored the application's capacity to dispatch event invitations via email to guests. With the successful reconfiguration, we were able to dispatch a variety of email notifications to the client, encompassing RSVP invitations, confirmations, expirations, referral invitations, and ticket

purchase confirmations. We fortified this feature by implementing comprehensive test cases using RSpec to ensure its robustness.

The client had underscored the importance of enhancing the application's front-end interface. So the team started to make the interface look modern and mobile friendly. The client's vision included the integration of promotional content, images, and copyright text throughout the website. Furthermore, he expressed the need for the application to be fully responsive, highlighting that the majority of web traffic is now mobile. This underscored the imperative of ensuring an optimal user experience across all mobile devices.

Iteration 3: (7 points)

We worked with the CRM team to enable GET and POST requests to fetch user data and alter user status. Front-end enhancements were made to the landing page, registration page, and event detail page to improve user experience and functionality. We also updated the Edit Events page to allow for importing box office data. The emailing system was fixed to ensure that guests receive their invitations, and work was started on making the referral system fully functional.

One of our major focuses was on the authentication system, which we discovered had several issues. We found that the sign-out and edit-profile links were not working, and that authentication was too deeply embedded in the application's functionality.

Iteration 4: (8 points)

A referral system was implemented, allowing guests to generate and track unique referral links for potential referrals. The system checks if referred guests purchase a ticket and updates the referral status accordingly. The app can now connect to Ticketmaster API to fetch box office data for events. The Seating Summary section received a new column to calculate tentative available seats. The Manage Seating Levels section now populates seating categories from box office data, but still allows event organizers to add/edit/delete categories. Lastly, the Events Page underwent a layout revamp to display one card per row on mobile and two cards per row on tablet screens, introduced a scale effect upon hover, and implemented a default image fallback feature for a consistent visual experience.

Iteration 5: (7 points)

We introduced a new feature that generates unique QR codes for each guest upon their addition to an event's guest list. A dedicated view was created to display the QR code of a specific guest, accessible via a unique URL, based on the guest's ID. Guests receive a direct link to their

personal QR code in the RSVP confirmation email. Another new feature was the Guest Referral Rewards, which allows event organizers to set a reward for each ticket bought by a guest that they referred. We updated the guest export and added seat categories, allocated, and committed counts. Furthermore, the team developed a guest import feature that allows users to import the guest list for one event on Ticketmaster and Eventbrite websites.

The team also started development on a new application from scratch aiming to provide a clean slate for future development teams.

6. Customer Meetings

We had weekly meetings with the customer where we discussed the current development status and expectations for upcoming releases.

SNo	Date	Current Meeting Discussions	Action items for next meeting
1	7-Feb-2023	referral code sharing system; integration of EventNXT into Event360 umbrella	deploy the application and get familiarized with the existing functionalities
2	16-Feb-2023	discussion about existing functionalities; local app deployment issues; possibilities of making the app mobile friendly, and add social media	refer Dio box mobile app and see if that can be implemented in current app; import guest list from prev events; need a common login via Event360
3	23-Feb-2023	edge cases to consider in referral system; possibility of tracking if guests read any email, or clicked on links	test rsvp, bulk email, and other existing functionalities and discover bugs; add sort and search in manage guests section
4	2-Mar-2023	Transfer the ownership of the heroku app to FashioNXT; provide different importing options to the users; import from major ticketing websites such as ticketmaster, eventbrite, and tickettomato	Ownership transfer; improving data importing options; allowing box office data load to the app from the major ticketing websites
5	9-Mar-2023	Front-end modification without changing the app codes; connect the app to the CRM team's app; load data from major ticketing websites; provide editable tables for the users	Using OAuth service provider (CRM team) to allow them capture our user analytics; modifying the tables to make them editable
6	23-Mar-2023	Fix the authentication system; idea of central authentication system; transfer ownership of both Heroku app and GitHub account	Ownership transfer; fixing authentication
7	30-Mar-2023	Central authentication	Communicating with other team to fix the faulty authentication
8	6-Apr-2023	Discussion on allotted, committed and guest committed in guest management;	Changes required in guest export file; add tentative free seats;
9	13-Apr-2023	Move the GitHub repo to the Github repo of FashioNXT; load box-office data from major ticketing websites; Dio-Box mobile app; QR code implementation	remove box office seating; change table layout in seating summary; test 4 edge cases in manage guests when uploading box office data, and sending emails; add guest import, duplicate event features;
10	20-Apr-2023	Talk about authentication with the CRM team; sending promo code; dropdown to select the major ticketing website to load data from; manage guest list; duplicate	Fixing the faulty authentication; give reward to who refers; create a dropdown to allow users to select which ticketing website they want to import data from; improving the

		event; import guest list; navigation bar	guest table; working on importing guest list and duplicating an event
11	27-Apr-2023	Sending user information to the CRM team	Adding an API callback controller and modify the routes.rb to allow the CRM team get access to our user data
12	4-May-2023	We showed the referral rewards, qr code features, and also the new app	Improving the new app; adding clear explanation; adding test cases; fixing minor issues of the old app

7. BDD/TDD Process

In our project, we made use of Behavior Driven Development (BDD) methodology to ensure that our team and the client were always on the same page regarding the requirements and expectations for the project. We held weekly client meetings and internal team meetings to discuss the requirements and progress of the project. During these meetings, we focused on defining and refining the user stories and acceptance criteria, which were documented in Pivotal Tracker. We also wrote cucumber tests to ensure that the code we were developing fulfilled the acceptance criteria, and that the behavior of the application aligned with the user stories. By using BDD, we were able to maintain a clear understanding of the project's goals and requirements, and we were able to quickly catch and address any issues or changes in requirements that arose during development.

While we were not able to follow a strict TDD approach due to the complexity of our application and the embedded JavaScript, we did recognize the importance of testing and ensured that we added test cases after development to increase our test coverage. We took a pragmatic approach to testing, focusing on adding tests for the critical and high-risk areas of the application first.
Overall, while we weren't able to follow a pure TDD approach, we recognized the value of testing and made sure to incorporate it into our development process in a way that made sense for our complex application.

8. Configuration Management Approach

Each team member worked on new features and changes for the project within their own branch, which was separate from the master branch. This approach ensured that development work was independent, organized, and not disruptive to the main codebase.

Once fully developed, changes and features were merged every two weeks, coinciding with the end of the iteration. This allowed the team to consolidate their work and review progress, ensuring that the new code integrated seamlessly with the existing codebase. At the end of each iteration, the team tagged the current version, providing a clear and traceable reference point for future development work. This helped to maintain version control and streamline the development process, enabling the team to work efficiently and effectively.

We created 14 branches to work on different tasks and issues and merged them before releases. We had 4 releases for each iteration except Iteration 0 and 1. For the new application, we created 18 branches and a single master branch to work on different features of the application, from implementing a simple home page, to implementing mailer service and writing test cases.

9. Production Release Process to Heroku

At the end of each iteration, all team members' work is merged into the master branch, ensuring that the latest changes and features are incorporated into the main codebase. Once the master branch is updated, it is pushed to our Heroku app, providing a seamless deployment process for our software system. This allows us to quickly and easily deploy new code changes to our production environment, ensuring that our app is always up-to-date and running smoothly.

To deploy the latest code to heroku, the team would need to merge all the changes in master branch, then connect to heroku from the command line, and push the code to heroku. We will list the commands and required credentials in the documentation folder in Github repo.

Link: Heroku Deployment

10. Issues with AWS, Cloud9, Github etc

During our use of AWS Cloud9, the team encountered a minor connectivity issue related to connecting our development environment to our GitHub repository. Despite this, the issue was quickly resolved, and we were able to continue working on our project without further interruption. Overall, we found AWS Cloud9 to be a reliable and effective tool for supporting our software development process.

11. Other tools or Gems used

No additional gems or tools are used for the legacy application.

12. **Public Github repository**

We have pushed all code including all test cases for cucumber and rspec into the public github repository.

Link to the repo:
https://github.com/FashioNXT/EventNXT-606-Spring2023

13. Deployment steps

The steps to deploy the app in local are listed in file named: EventNXT code setup

To deploy the EventNXT app locally, the necessary steps are outlined in a file called "EventNXT code setup". The team can initiate the process by forking the current GitHub repository and creating a new repository in their own account. Once the new repository is cloned to a local machine or cloud 9 instance, the instructions in the "EventNXT code setup" file can be followed to complete the deployment process. By following these steps, the team can ensure a smooth and successful deployment of the application.

14. Links

- GitHub: https://github.com/FashioNXT/EventNXT-606-Spring2023
- Pivotal Tracker: https://www.pivotaltracker.com/n/projects/2630477
- Heroku: https://eventnxt-606.herokuapp.com/
- Presentation ppt: https://docs.google.com/presentation/d/1NzgOyRXVHw1_QpbdO-9XAr5GGesIFyvTSQRlc8dyyeg/edit#slide=id.p
- Presentation Video: https://vimeo.com/824284492/07924a924a
- Demo Video: https://drive.google.com/drive/folders/1dUpCsod2lxgNqooWxkGQ8RzkrfrU-32z?usp=sharing
- New Application Github Repository: https://github.com/Aminisazadeh/AI_FashionNXT_002