

CSCE 606 Team: SoftwareSetSails
Event Guest List Automation
Final Report

Team Members

Asees (Product Owner)
Sunayna Ray (Scrum Master)
Dharmendra Baruah
Jiyoon Hwang
Leon Lin
Pavankumar Prakash Kakkanavar
Suma Katabattuni
Sukanya Sravasti

TABLE OF CONTENTS

- 1. Summary**
- 2. Broad Description of User Stories**
 - 1. Setup page: Creation of an event**
 - 2. First step of event creation**
 - 3. Event Preview and Edit**
 - 4. Box Office Data User Interface**
 - 5. Guest List Page**
 - 6. Referral Rewards Form**
 - 7. Email Invitation**
 - 8. Add New Seat Category**
 - 9. Edit Existing Seat Category**
 - 10. Show Seat Category**
 - 11. Reconcile Seat Category**
- 3. Stories not implemented**
- 4. Legacy Project Overview**
- 5. Team Roles**
- 6. Iteration Wise Summary**
- 7. Customer Meeting Summary**
- 8. BDD/TDD Process**
- 9. Configuration Management**
- 10. Initial Setup Issues**
- 11. Steps to launch the app**
- 12. External tools**
- 13. Relevant Links: Repository, Pivotal tracker, etc.**

1. Summary:

This was a legacy project from last semester. Our main customer was the founder of FashionNXT, Tito Chowdhury. Regarding software as a service, FashionNXT focuses on increasing efficiency in various processes in the fashion industry through automation. For this project, the main customer need was the improvement of the legacy application from last semester. This legacy application automated managing guest communications and ticket commitments for FashionNXT events. The main functionality of the application is to allow an event owner to track ticket sales and manage communication with guests, which allows the owner to strategize event promotion and ensure that he/she is not over-committing. Event ticket commits have two different sources: (1) box-office ticket sales which come from a spreadsheet (2) Guest list that require RSVP tracking with guests and email communication.

Last semester, the software engineering group focused on providing a simple access control to the event owner. It managed RSVP communication with the Guest list and merged those commitments with Box Office sales. The main customer need this semester was improving two main features, as well as implementing four new features.

The two improvements from the past year's project are as follows.

1. Last year, the guest submitted the RSVP response form through the email link and received RSVP communication that was automatically generated. The event owner was not able to edit the contents of the RSVP email and add graphics from the user interface. In our improvement, we allowed the event owner to edit the email that was sent to guests, so that he could insert graphics and customize the email through the application.
2. Last year the confirmations of RSVPs from guests were not separated by seat category. This year we separated the confirmation of RSVPs through seat category, so that the event owner can see how many VIP guests of a particular seat category provided feedback and email them manually.
3. Additionally, we faced a huge number of errors as it was a legacy project that worked on a very old rails version. The update of this version made the code extremely fragile and prone to errors. To improve this, we solved all the bugs that resulted from the update. These bugs were very difficult to isolate and remove and often did not have a stack trace. The action would simply not be triggered.

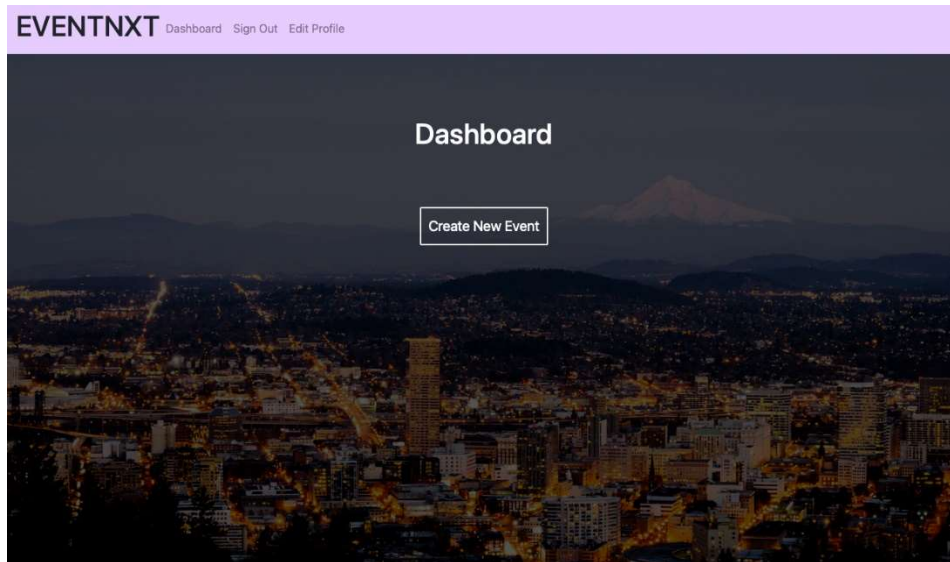
The new user features that we implemented focused on the following:

1. We restructured the application so that event owners could create an event and set up different features of the event such as RSVP email content.
2. We implemented a feature to provide counts of available seats based on seat category so that the event owner can send appropriate emails to that specific seat category.
3. We implemented a feature that allows the event owner to see the allotted seats in the box office category and guest list category and track the balance left in each category.
4. We added an extra column of seat category to the event management framework
5. We implemented the design and front-end of the referral reward system which would allow event managers to draft customized emails to guests with custom rewards.

2. Broad description of user stories

We implemented multiple features and bug resolve stories in the project. The story points ranged from 1-3 points. There were also stories for iteration reports etc. We have broadly coupled the stories and listed them here. For a more detailed view, the pivotal tracker and GitHub links are presented in the report.

I. Setup page: Creation of an event (12 Points)



Status: Fully Implemented

As an administrator, I want to see a setup/dashboard page with a “Create New Event” button when I log in so I can create an event. This Setup Page no longer contains adding seat category functionality as it was moved to another page.

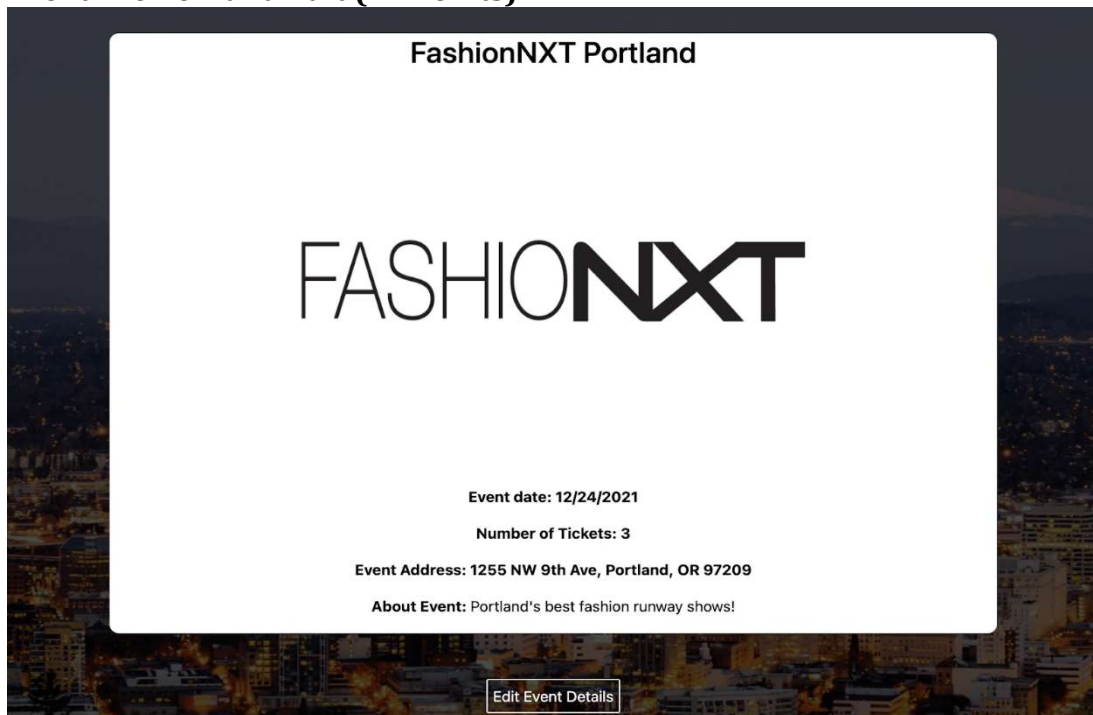
II. First step of event creation (13 Points)

A screenshot of the event creation form, which is overlaid on the same cityscape background as the dashboard. The form is enclosed in a white border and contains the following fields and buttons: 'Event Name' (text input), 'MM/DD/YYYY' (date input), 'Number of Tickets' (text input), 'Image address or path' (text input), 'Height: e.g. 300px' (text input), 'Width: e.g. 300px' (text input), 'Event description' (text input), and a 'Save and Continue' button at the bottom.

Status: Partially Implemented

As an administrator, I want to see a form when I click on “Create New Event” so I can enter event details and save the details to an invitation template to be sent to my guests. This menu is refined to be more user friendly and less cluttered. Details still need to be saved for each user account and crop function needs to be added.

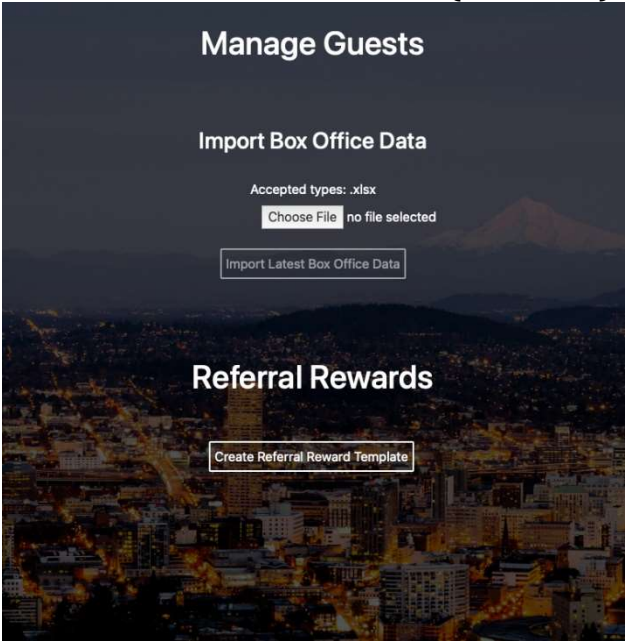
III. Event Preview and Edit (14 Points)



Status: Fully Implemented

As an administrator, I want to see a preview of the event invitation generated from the details I entered on the “Dashboard” after I click on “Save and Continue”, and I want to be able to edit the invitation when I click the back button. This way, I can notice whether or not I made a mistake and can then proceed to correct the mistake. This preview is now shown on a separate page after the admin enters event details.

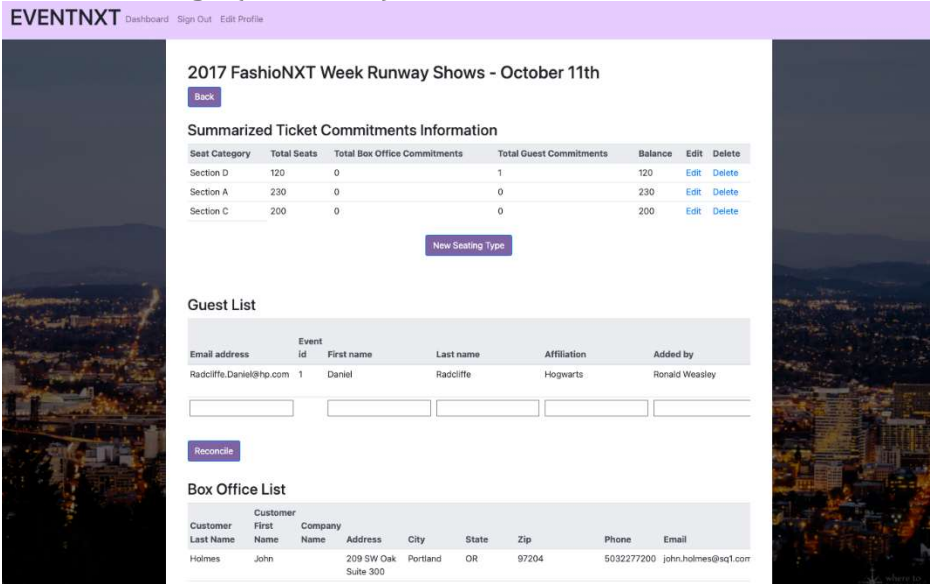
IV. Box Office Data User Interface (16 Points)



Status: Partially Implemented

As an administrator, I want to see an option to import a spreadsheet so I can add guests and enter seat level information. I also want to see an option to create referral rewards. This page still needs a table showing guest commitments.

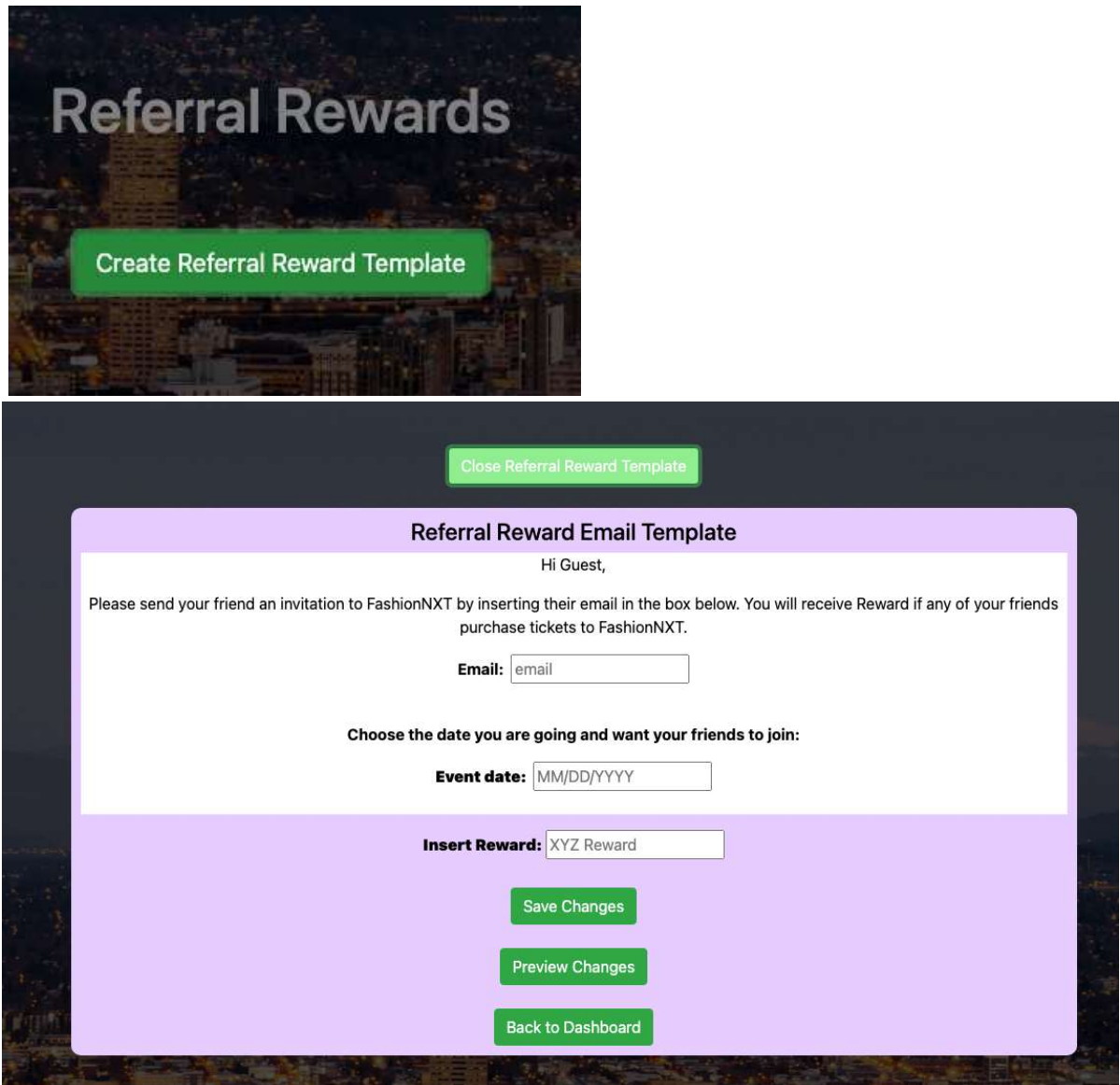
V. Guest List Page (11 Points)



Status: Fully Implemented

As an administrator, when I click on “Import Latest Box Office Data”, I want to see a table of summarized commitments and a list of invited guests.

VI. Referral Rewards front end (12 points):



The image displays the 'Referral Rewards' front end user interface. At the top, a dark banner with a city skyline at night features the text 'Referral Rewards' in large white letters. Below this, a green button labeled 'Create Referral Reward Template' is visible. The main content area is a light purple box titled 'Referral Reward Email Template'. Inside this box, there is a 'Close Referral Reward Template' button at the top. The template content includes a greeting 'Hi Guest,', a paragraph explaining the referral reward system, an 'Email:' label with a text input field containing 'email', a prompt to 'Choose the date you are going and want your friends to join:', an 'Event date:' label with a date input field showing 'MM/DD/YYYY', an 'Insert Reward:' label with a text input field containing 'XYZ Reward', and three green buttons at the bottom: 'Save Changes', 'Preview Changes', and 'Back to Dashboard'.

Status: Fully implemented front end user interface

As an event owner, after I set up an event, create an event and visit the guest list page, I want to create an email template of a referral reward system which would allow me to send emails to guests, where I can enter custom rewards. The email should have options for the guest to enter the date of the fashion show they are going to and add their friend's emails.

VII. Email Invitation (20 Points)

Hi Hank,

Congratulations! You're invited by NXT to an event.



Status: Fully Implemented

As a guest, I want to see an email invitation with the event logo/picture so I know which event I am invited to. I also want to see the event details so I know where and when the event is.

VIII. Add New Seat Category (12 Points)

EVENTNXT [Dashboard](#) [Sign Out](#) [Edit Profile](#)

New Seating Type

Seat category

Total seat count

Create Seating type

[Back](#)

Status: Fully Implemented

As an event owner, I want to add a new seating type to the seat category. I want to type a seat category name and total number of seats for the seating type. I also want to see the create button and back button.

IX. Edit Existing Seat Category (10 Points)

EVENTNXT [Dashboard](#) [Sign Out](#) [Edit Profile](#)

Editing Seating Type

Seat category

Total seat count

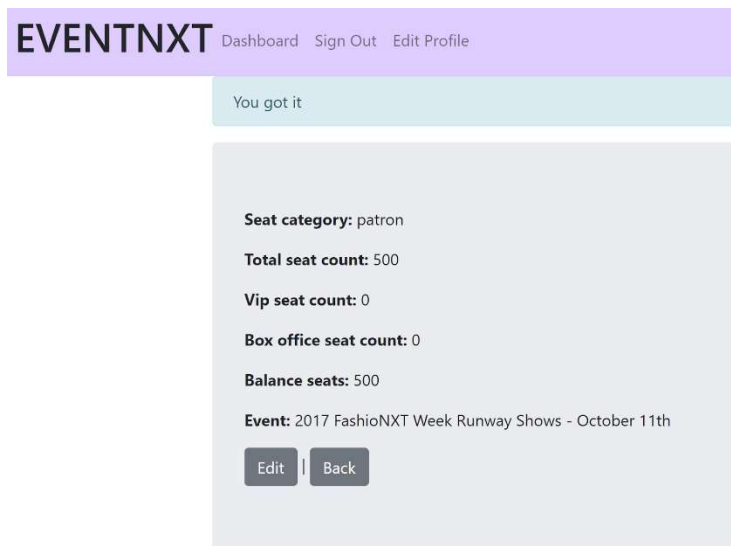
[Update Seating type](#)

[Show](#) | [Back](#)

Status: Fully Implemented

As an event owner, I want to edit an existing seating type and change the total seat count. I also want to see the update button and show button to show the preview of the updated seating type and back button to go back to the main board.

X. Show Seat Category Features (7 Points)



Status: Fully Implemented

As an event owner, I want to see seat category features including total seat count, VIP seat count, box office seat count, balance seats, and event name. I also want to see edit buttons and the back button to go back to the main board.

XI. Reconcile Seat Category (7 Points)

Original Table:

Summarized Ticket Commitments Information						
Seat Category	Total Seats	Total Guest Commitments	Total Box Office Commitments	Balance	Edit	Delete
PATRON	500	40	10	450	Edit	Delete
GA STAND	300	0	0	300	Edit	Delete

Reconcile Button:

Reconcile

Box Office List

Updated values:

Summarized Ticket Commitments Information

Seat Category	Total Seats	Total Guest Commitments	Total Box Office Commitments	Balance	Edit	Delete
PATRON	500	40	10	450	Edit	Delete
GA STAND	300	118	0	182	Edit	Delete

New Seating Type

Status: Fully Implemented

Count the seat categories allocated from the box office and guest list table.
Reconcile the summation on the Summarized Ticket Commitment Information.
Update balance and commitments made according to reconciled values.

3. **Stories not implemented**

- After Guests accept the RSVP, the count should be updated in Summarized Ticket Commitments Information table
- After VIP accepts the RSVP request, the status should get reflected
- Seat Category Drop Down list to be added for VIP guest list
- Train the App to recognize the entrees in Box Office XLX header in sync with with Guest List column headers. (Get to only do it once when App reads the
- Box Office spread-sheet once)
- Make Box Office List Editable
- Display rejected and accepted box office rows
- Push rejected box office entries in reject db
- Box office Excel seat category check if in admin list or not
- Logged in email should be sender of invite emails

4. **Legacy Project Overview**

We ran into challenges while building the application from the previous one. The main challenges were as follows:

- We faced issues in deploying the application because of database management systems and missing libraries. The previous code was not compatible with Postgresql
- The default branch of the previous application was set to main but the working branch was master due to which we lost a week of work since it took time to get in touch with the other team member
- After Deploying successfully to Heroku, we were not able to import the box office excel sheet because of bugs in the existing code. The email functionality was not working as the previous team had not captured the add-on requirement for Heroku.

5. **Team Roles**

- Product Owner:
 - Iteration 0: Asees Sidhu
 - Iteration 1: Asees Sidhu
 - Iteration 2: Asees Sidhu
 - iteration 3: Leon Lin
 - Final Iteration: Asees Sidhu
- Scrum Master: Sunayna Ray (All Iterations)

6. Iteration Wise Summary

- **Iteration 0:** For this iteration, after our first customer meeting, we decided the main stories and features of the application for this semester. We summarized the main customer needs for this semester based on our first meeting. We came up with new features that the customer wants, as well as improvements of old features from the previous application that the customer wanted.
- **Iteration 1:** For this iteration, we solved the bugs in the previous group's project that lead to deployment issues. We documented this extensively in iteration 1.
- **Iteration 2:** For this iteration, we completed three user stories and fixed two additional bugs from the previous project. The three user stories completed were implementing the customizable RSVP feature, editing the seat category feature and merging these former two features into an event setup panel in the application. After showing these features to the customer, he gave us more edits for the structure of the application. We fixed two bugs: one related to migration and database incompatibility, and the other related to duplicated entries in the ticket summary.
- **Iteration 3:** We completed five main user stories in this iteration. 1) We made the seat category editable in the application by the admin. 2) We added a new feature so that admin could add seat categories 3) We changed the view of the seating table 4) We added a feature such that the admin can see the summary of a particular seat category (e.g. the total seat count and how many seats are remaining) 5) We created a setup page where the admin can create events, add seating levels, create invitation card and go to event creation page using buttons. 6) We added a feature where the event owner can create an invitation card for events by typing event title, event date and number of tickets. The admin can also insert images in the card. 7) We added a new feature where the admin can preview the invitation card after creating the invitation card .
- **Final report iteration:** We showed the front end of the referral reward feature to the customer, and the working application after incorporating his edits from the past meeting that week.

7. Customer Meeting Summary

- **October 21, 2021:** In this meeting, the customer gave us an introduction to the project and discussed his expectations for the main deliverables for this semester.
- **October 27, 2021:** In this meeting, we communicated the bugs and problems in deployment we were facing with the previous code with the customer. He asked us to reach out to the previous team and solve the bug issues.
- **Nov 3, 2021:** In this meeting, we showed the customizable RSVP feature of the application to the customer, as well as communicated how we solved the deployment issues we were having.
- **Nov 10, 2021:** In this meeting, we showed the seat category feature of the application to the customer. This feature maintains ticket information for an event and summarizes it by seat category based on the excel sheet imported and the admin input.
- **Nov 17, 2021:** In this meeting, we showed the structure of the event setup panel, where we merged the branches of the past features we demoed into the overall application, such that when an admin logs in, he/she has the option to select his/her options for setting up an event.
- **November 24, 2021:** In this meeting showed the customer the added features of making the seating type editable, option to add a new seating type and the view of the seat category table and summarized information.
- **December 1, 2021:** In this meeting, we showed the changes implemented in the main event setup page that the customer had asked for in the previous meeting, as well as changes to the feature of adding the invitation card contents.
- **Decembre 8, 2021:** In this meeting, we showed the front end design of the referral reward module to the customer, as well as the overall application. We communicated the further remaining changes that we will be able to implement that he wanted and set up a final date to deliver the application changes we implemented this semester.
- **December 13, 2021:** In this meeting, we showed the final product to the customer including fancy invite update, referral update and seat category updates. We came to a conclusion about work that needs to be done by the next team, which includes, removing the login feature from event response email and removing ticket number printed in the email invite. We also finalized that the seat category drop down will be done by the next team.

8. BDD/TDD Process

We found that this process was of great help. Since this was a legacy project we found a lot of bugs in it's implementation. The existing features were constantly failing on every update. This was even worse due to lack of sufficient test cases. Hence, we wanted to avoid this scenario in our code.

All Files (74.7% covered at 0.79 hits/line)

16 files in total.

384 relevant lines, 314 lines covered and 70 lines missed. (74.7%)

Search: <input type="text"/>							
File		% covered ▲	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/models/seating_type.rb		10.00 %	6	6	2	4	0.30
app/mailers/application_mailer.rb		44.20 %	4	4	3	1	0.47
app/models/event.rb		89.89 %	46	28	21	7	0.45
app/models/application_record.rb		60.08 %	3	2	1	1	1.00
app/controllers/events_controller.rb		60.59 %	459	180	147	33	1.00
app/helpers/events_helper.rb		77.77 %	2	2	1	1	1.00
app/controllers/guests_controller.rb		78.80 %	236	76	57	19	1.10
app/models/seat_category_details.rb		90.00 %	8	8	6	2	0.85
app/controllers/seating_types_controller.rb		98.01 %	71	52	50	2	0.80
app/controllers/application_controller.rb		100.00 %	6	4	4	0	0.40
app/helpers/application_helper.rb		100.00 %	2	2	2	0	0.80
app/helpers/guests_helper.rb		100.00 %	2	2	2	0	1.00
app/helpers/seating_types_helper.rb		100.00 %	2	2	2	0	0.76
app/mailers/guest_mailer.rb		100.00 %	16	12	12	0	1.00
app/models/guest.rb		100.00 %	2	2	2	0	1.00
app/models/user.rb		100.00 %	6	2	2	0	0.80

Showing 1 to 16 of 16 entries

Controllers (56.9% covered at 0.8 hits/line)

4 files in total.

312 relevant lines, 181 lines covered and 131 lines missed. (56.9%)

Search: <input type="text"/>							
File		% covered ▲	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/controllers/application_controller.rb		50.00 %	6	4	2	2	0.80
app/controllers/events_controller.rb		53.80 %	459	180	98	82	0.54
app/controllers/seating_types_controller.rb		57.86 %	71	52	32	20	1.00
app/controllers/guests_controller.rb		61.00 %	236	76	49	27	0.92

Showing 1 to 4 of 4 entries

Channels (100.0% covered at 0.0 hits/line)

0 files in total.

0 relevant lines, 0 lines covered and 0 lines missed. (100.0%)

Search: <input type="text"/>							
File		% covered ▲	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
No data available in table							

Showing 0 to 0 of 0 entries

Models (38.33% covered at 0.72 hits/line)

6 files in total.
48 relevant lines, 19 lines covered and 29 lines missed. (38.33%)

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/models/event.rb	23.70 %	46	28	10	18	0.76
app/models/seat_category_details.rb	27.00 %	8	8	3	5	0.54
app/models/seating_type.rb	37.98 %	6	6	2	4	0.19
app/models/guest.rb	50.00 %	2	2	1	1	0.36
app/models/user.rb	50.00 %	6	2	1	1	1.00
app/models/application_record.rb	100.00 %	3	2	2	0	1.00

Showing 1 to 6 of 6 entries

Mailers (100.0% covered at 0.91 hits/line)

2 files in total.
16 relevant lines, 16 lines covered and 0 lines missed. (100.0%)

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/mailers/application_mailer.rb	100.00 %	4	4	4	0	0.80
app/mailers/guest_mailer.rb	100.00 %	16	12	12	0	1.00

Showing 1 to 2 of 2 entries

Helpers (100.0% covered at 0.68 hits/line)

4 files in total.
8 relevant lines, 8 lines covered and 0 lines missed. (100.0%)

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/helpers/application_helper.rb	100.00 %	2	2	2	0	0.75
app/helpers/events_helper.rb	100.00 %	2	2	2	0	0.71
app/helpers/guests_helper.rb	100.00 %	2	2	2	0	0.68
app/helpers/seating_types_helper.rb	100.00 %	2	2	2	0	1.00

Showing 1 to 4 of 4 entries

Jobs (100.0% covered at 0.0 hits/line)

0 files in total.
0 relevant lines, 0 lines covered and 0 lines missed. (100.0%)

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
No data available in table						

Showing 0 to 0 of 0 entries

Libraries (100.0% covered at 0.0 hits/line)

0 files in total.
0 relevant lines, 0 lines covered and 0 lines missed. (100.0%)

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
No data available in table						

Showing 0 to 0 of 0 entries

9. Configuration Management

One team member forked the project from the old project. Then, the rest of the team members forked from this branch and implemented their respective features in their fork. Then after each feature was delivered, the members got together to merge their code to the forked master branch and push the changes there. There were also multiple branches that were created to work on different user stories. The branches were named using 'feature/user-story-name' convention. The branches were merged and deleted to keep the repository clean.

10. Challenges faced during initial setup:

Since this is a legacy project, there were a few challenges we faced during deployment due to lack of proper documentation

1. The GitHub repository which was shared to us as part of the legacy project was not the correct repository due to which deployment of the project wasn't happening locally as well as to Heroku.
2. Another issue with the deployment of the application was that the default branch was set to main, but the working branch was master due to which all forms of the application weren't loading. Below are the steps followed to fix this challenge:
*git checkout master"
3. One error message with the deployment of the application to Heroku was:
Git push error '[remote rejected] master -> master (branch is currently checked out)'
Below are the steps followed to fix the challenge:

(a) Add a following line into the Gemfile
gem 'bigdecimal', '1.4.2'
(b) bundle install
4. After the latest repository of the legacy project was sent we were also facing issues with SQL and encountered the following error: could not connect to server: No such file or directory Is the server running locally and accepting connections on Unix domain socket
"/var/run/postgresql/.s.PGSQL.5432"?
Below are the steps followed to fix the challenge:

(a) add line "gem 'pg', '~> 0.21'" in the Gemfile
group :development, :test do
gem 'pg', '~> 0.21'

end

(b) `sudo yum install postgresql postgresql-server postgresql-devel
postgresql-contrib postgresql-docs`

(c) `bundle install --with production`

5. After Deploying successfully to Heroku, we were not able to import the box office excel sheet which was rectified by commenting the following line in `db/migrate/20210426211246_remove_type_from_guests.rb`

```
# remove_column :guests, :type, :string
```

11. Steps to launch the App

```
sudo yum install postgresql-devel  
bundle install --with production  
sudo yum update -y  
sudo yum install postgresql-server postgresql-contrib  
sudo postgresql-setup initdb  
sudo systemctl start postgresql  
netstat -an | grep 5432 (Port of Postgres)  
sudo -u postgres createdb myapp_development  
sudo -u postgres createuser -s ec2-user  
sudo -u postgres createdb ec2-user  
rake db:migrate  
rails s -p $PORT -b $IP
```

12. External Tools

Since Heroku does not include a mailing service, we used a service called Sendgrid to handle email delivery to our guests. We chose Sendgrid because it is free and easy to integrate with Heroku. SendGrid also comes with great analytic and security features.

13. Relevant Links

The API key for the SendGrid tool has to be kept in a private repository to be able to use it, otherwise the key can be misused and the account gets banned. Hence we have created 2 git repositories with the same code. The only difference is that one of them is private and has the API key. To include the email functionality, one needs to clone the public repository, add the API keys and launch the app. User can also clone the private repository directly and launch the app.

- **Public Github Repository:**
[sunayna-ray/Event-Guest-List-Automation \(github.com\)](https://github.com/sunayna-ray/Event-Guest-List-Automation)
- **Private Github Repository:**
[Pavankumar-code/EventNxt \(github.com\)](https://github.com/Pavankumar-code/EventNxt)
- **Public Repository Heroku App:** (Without SendGrid API Link)
App: event-guest-list-automation.herokuapp.com
Dashboard: [event-guest-list-automation | Heroku](https://event-guest-list-automation.herokuapp.com)
- **Private Repository Heroku App:**
eventnxtapp.herokuapp.com
- **Pivotal Tracker:**
<https://www.pivotaltracker.com/n/projects/2543623>
- **Old tracker:**
<https://www.pivotaltracker.com/n/projects/2536172>
- **Link to Poster:**
<https://docs.google.com/presentation/d/1oN042bFvOoj9KnCerT0OdysuQflYVwlN/edit?usp=sharing&ouid=115031403820389208733&rtpof=true&sd=true>
- **Demo and Poster Presentation video link:**
<https://www.youtube.com/watch?v=orityZjlSek>