Milestone 3 Report

Bethe OPS Project
March 29, 2019

1 Team

Jaewon Sim - js2689@cornell.edu
Albert Tsao - awt46@cornell.edu
Yun Ping Tseng - yt376@cornell.edu
Yuxin Xu - yx67@cornell.edu
Angela Zhang - az337@cornell.edu
Amy Zhong - az287@cornell.edu
Jennifer Zhou - jz547@cornell.edu

2 Client

Erica Ostermann; Assistant Dean; Hans Bethe House; eo93@cornell.edu, hadbethe@gmail.com

3 Objective

The task for this project is to implement a responsive web interface for the Hans Bethe House that serves as an automated tool for house event sign-ups to replace the current mechanism of signing up for events on paper at the front desk.

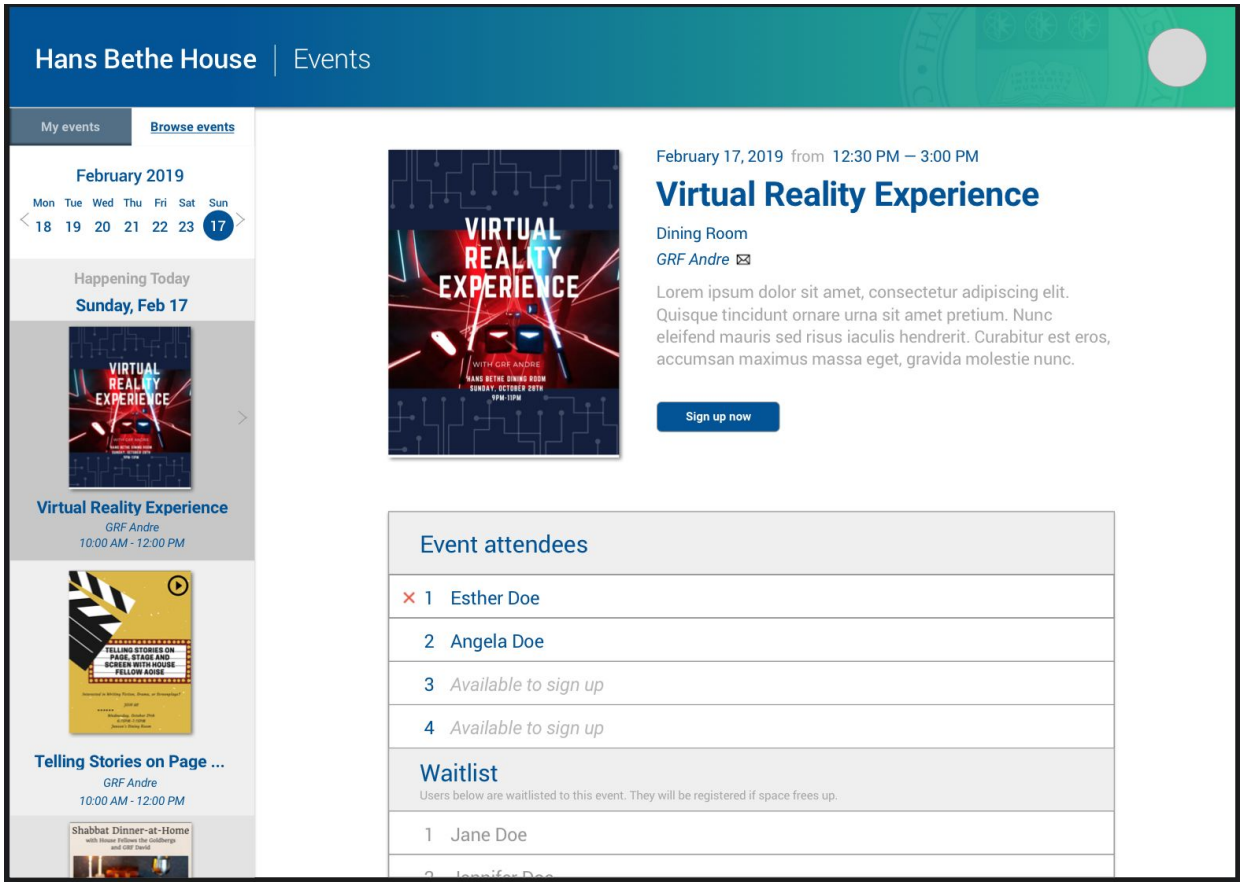
The main objective of this report is to review and receive feedback on the provisional design mockups, and document the progress made and the next steps to be made by both the user interface team and the back-end team. This report will give the breakdown of tasks and timeline in the form of a Gantt chart for the final milestone.

4 User Interface Progress

4.1 Updated Event Landing Page Design Mockups

Major Changes to Browse Event Pages

Below is an example of the “Browse events” page.



Some of the major changes introduced in this iteration include:

- Refinement of the event browser sidebar in regards to coloring, tabbed navigation between “My events” and “Browse events”
- Removal of the event availability indicator and its replacement with the numbered event attendees list
- Sign-up availability indicator
- Ability for privileged users to see the details of each registrant’s net ID and housing information

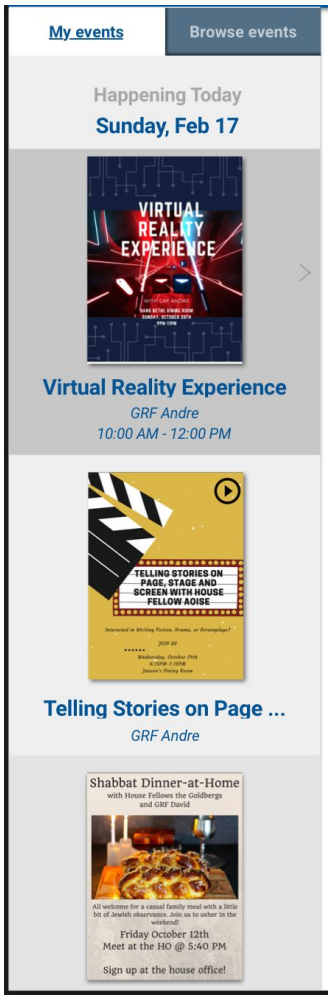
The details regarding these changes are explained below, along with the mockups for what each type of user — student, event leader, or admin — would see when logged into his/her account.

4.2 User Dashboard Design Mockups

Student Dashboard

For this iteration, the design team mainly focused on designing a feature that would allow users to view the upcoming events that they had signed up for. As before, the designers wanted to focus on integrating the feature into the interface in a way that was simple,

intuitive, and didn't require many additional actions from the user. The team considered a few different options, such as using a navigation bar, or displaying a new page when switching between viewing registered events and browsing events. After some discussion, the team decided to simply add a tab menu to the sidebar (see image below).

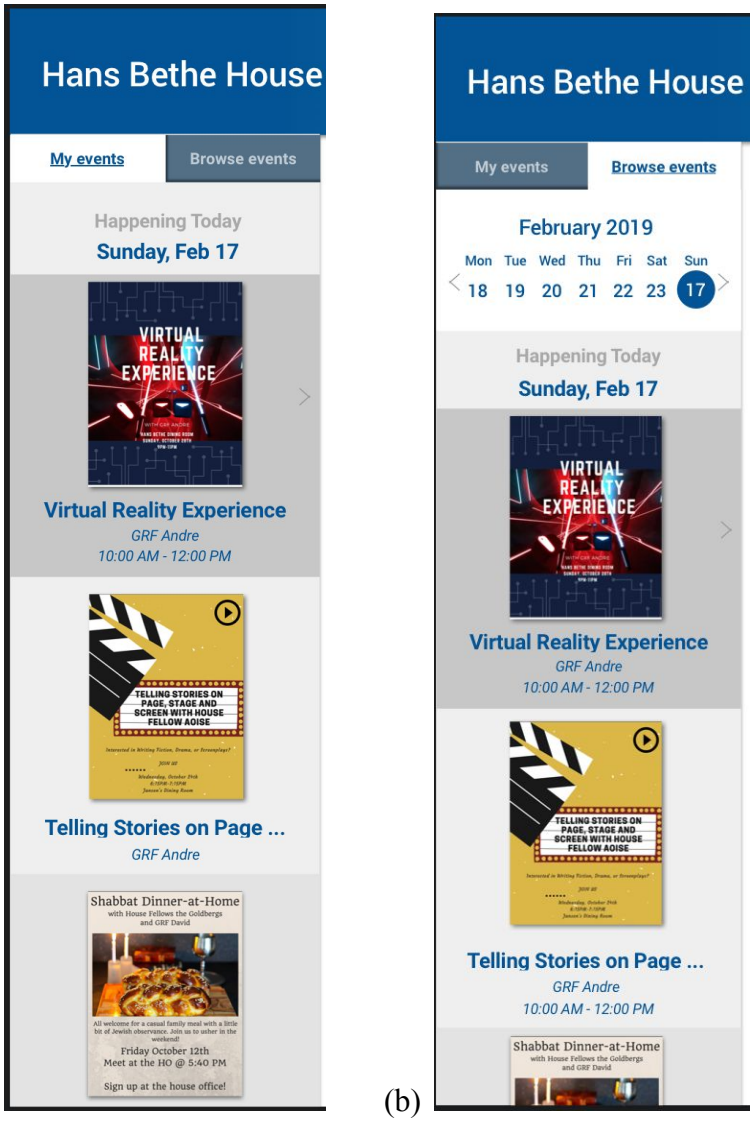


Modified Sidebar with Tab Menu

With this design, users can easily and quickly switch between seeing events they've signed up for and browsing upcoming events. While viewing the "My Events" tab, the calendar displayed in the header is removed to prevent any confusion relating to the dates of the events that have been signed up for (as users are likely to have signed up for less events than is offered during a week). When users first log in to the site, the sidebar will show the "My Events" tab to users by default.

Overall, the client was very satisfied with this design and provided some minor feedback. The main requests were to make the indication of the selected tab more clear (e.g. make it

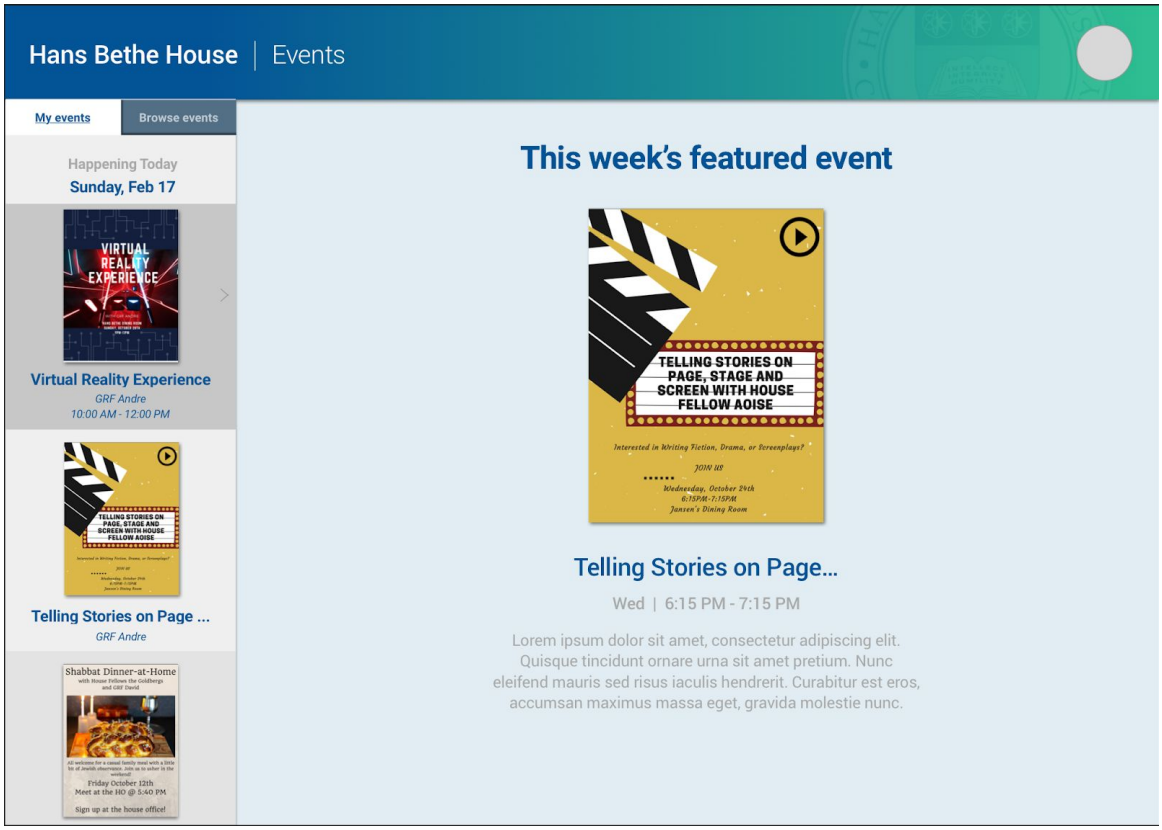
more obvious which tab you were on) and to add the start times to the events shown in the sidebar. Furthermore, the client also requested that the “My Events” feature also be made available for admins and event leaders as many of them are often users who will also be signing up for events. Below are images reflecting the requested changes.



(a) (b)

Updated Sidebar - (a) “My events” Tab and (b) “Browse events” Tab

The team also designed a preliminary design to display a featured event that is selected by an admin. If no events are selected as a featured event, the default is to display the next upcoming event as the featured event. Only one featured event is shown as a time and will be replaced after the event passes. To bring up the event profile for the featured event, the user should click on the poster displayed on the right-hand side. This sketch is shown below and will be implemented during the next iteration of development.



Preliminary sketch of featured event page

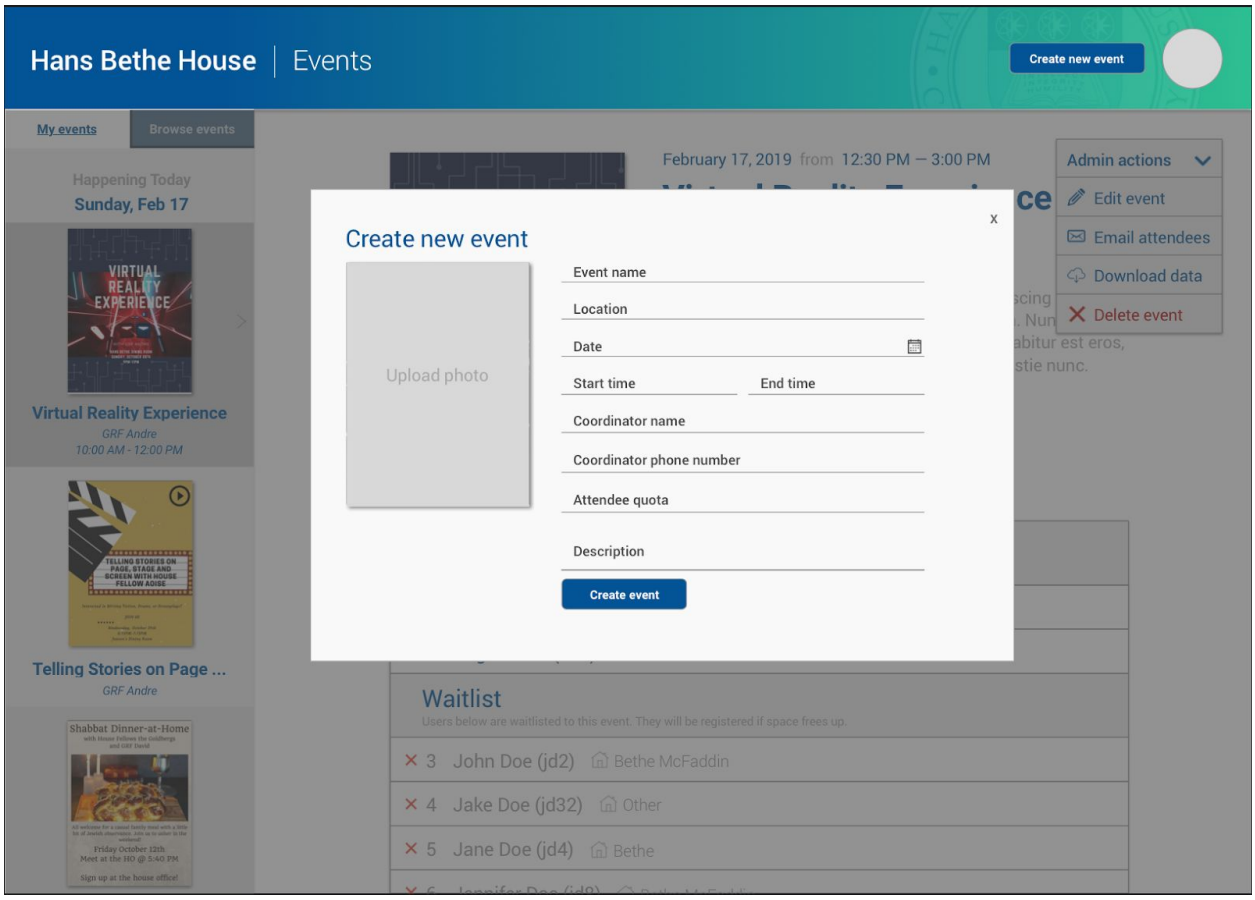
Admin Dashboard

During this iteration, the design team also began mocking up the dashboard view for admins. Admins are users who have the highest authority and thus have extra privileges that are not available to other types of users. As with the rest of the interface design, the team focused on integrating the required features in a way that was both seamless and intuitive. According to the client, admins have the following privileges:

- Create events
- Edit events
- Email attendees (time permitting)
- Download event data
- Delete events
- Add/Remove attendees from sign-ups

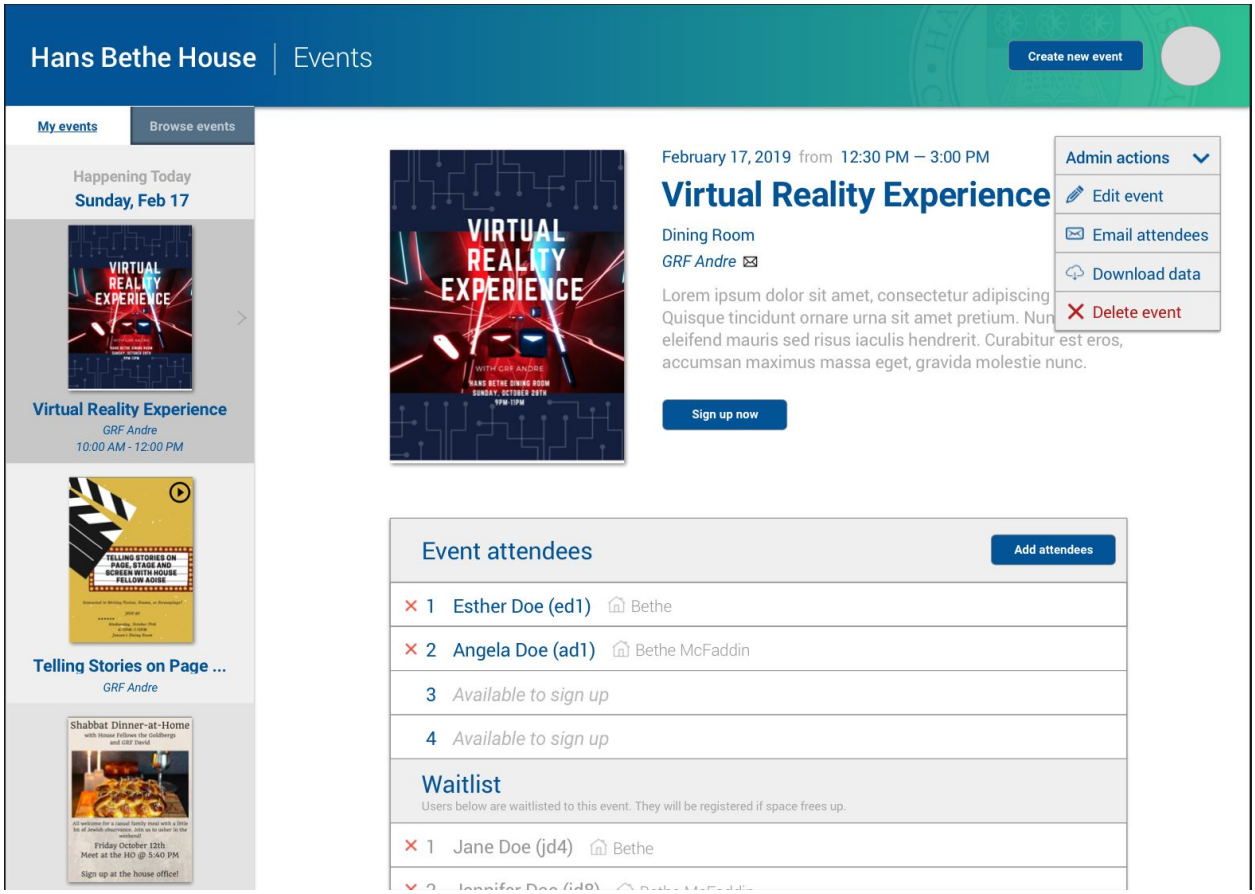
To create a new event, admins simply click the “Create new event” button located in the header at the top right part of the screen. Clicking this event brings up a pop-up form that contains the fields required to create an event (event name, date, location, etc.). These fields are mostly text-based, though the client requested that dates be selected using a calendar feature. The design team felt that this might lead to users incorrectly selecting the date and thus proposed the use of user-inputted dates. The eventual compromise was

to create a text-based input (e.g. mm/dd/yyyy) with an additional calendar option. This is not shown in the mock-ups but has already been implemented by the front-end.



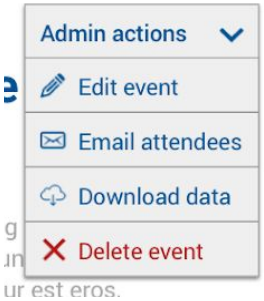
Form to create events

To edit an event, email attendees, download event data, delete an event, and add/remove attendees, users are required to first open the event profile of the specified event. This design choice was made intentionally as a safeguard to ensure that the admin could not accidentally perform any of these actions without going through a series of several steps.



Admin dashboard view

As seen in the image above, the edit event, email attendees, download data, and delete event actions are found in a drop-down menu labeled “Admin actions” located at the top right side of the event profile. The drop-down menu is initially hidden with only the “Admin actions” rectangle visible. Clicking on the box brings up the menu, allowing admin to access these actions. The team decided to use a clicking gesture as opposed to hovering to make this menu usable in the mobile version.

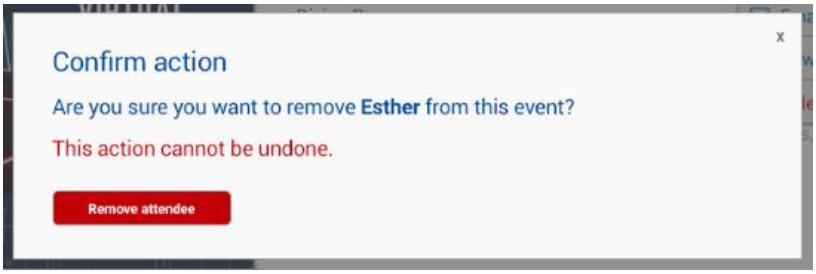


Admin actions drop-down menu

To add attendees to a sign-up list, admin simply need to click on the “Add attendee” button located on the right side of the list. On the other hand, to remove attendees from the sign-up list, admin simply click the red “X” located next to a student’s name.

Admin view of sign-up list and waitlist

To prevent users from accidentally deleting students from the sign-up list, a confirmation pop-up appears (see image below), requesting users to confirm the deletion. This action is consistent with the actions that regular students would take to remove his/herself from a sign-up list with the main difference being that admin can add/remove any student from the list.



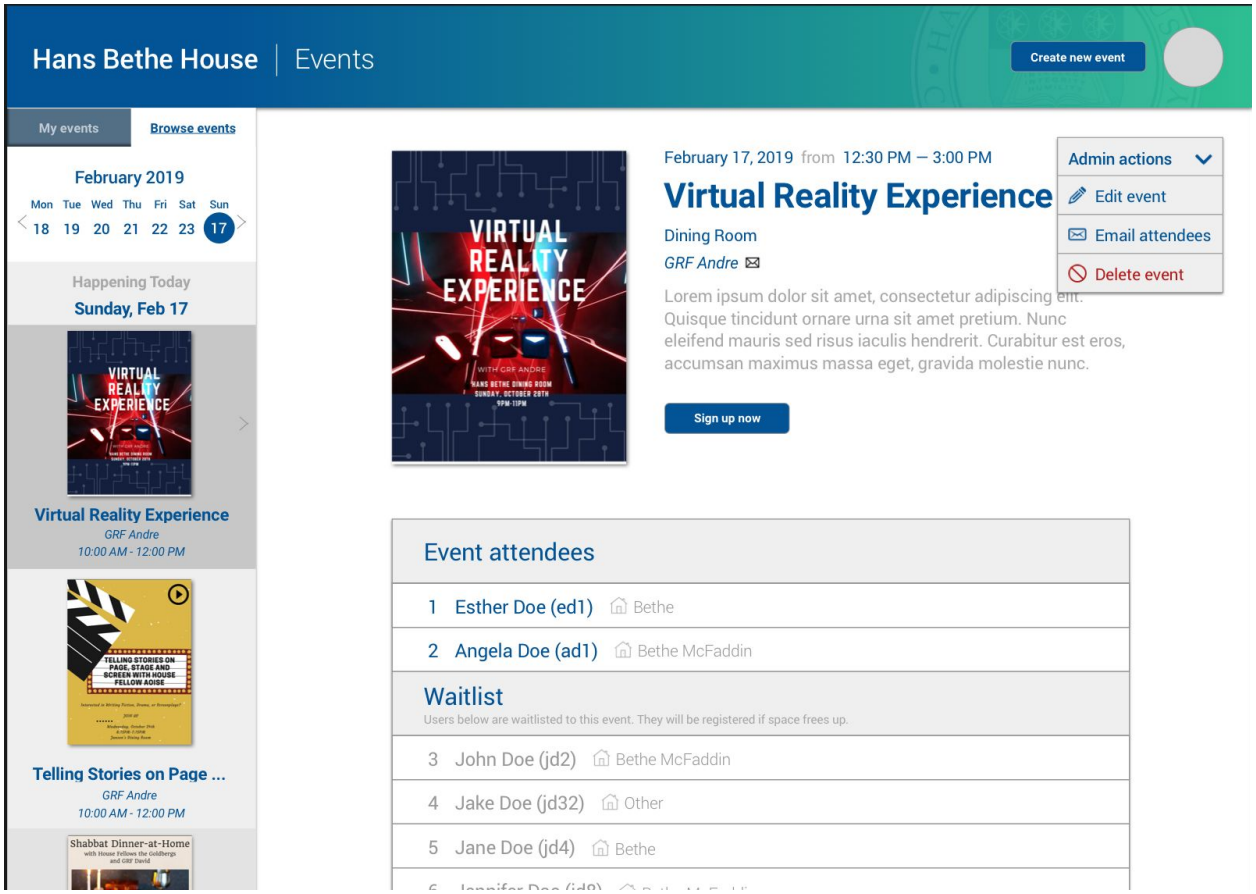
Pop-up confirming the deletion of an attendee

In providing feedback on the initial admin dashboard designs, the client requested that the team modify the sign-up list and waitlist to show the number of spots that each list had. As seen above, entries that weren’t populated by a student were still shown with a placeholder text saying “Available to sign up.” Because the sign-up list and waitlist now implicitly conveys the number of available spots to users, the client felt that the icons used previously were redundant in conveying the availability of the event and thus requested that the team remove that feature. These changes can be seen in the admin dashboard view shown previously.

Furthermore, another difference between the student and admin view of the sign-up and waitlist is the visibility of the attendees' private information. Per the client's request, admin and event leaders (described below) are able to view student's personal info including their netID and building location whereas regular students are only able to view the first and last name of the people signed up for the event.

A quick note about the email attendees action:
During the initial requirements phase, the client requested that the team implements an automatic emailing feature that would remind attendees of an upcoming event and inform waitlisted users of when they had been taken off the waitlist. Though the system itself would automatically send these emails, a potential extension to this feature would allow admin or event leaders to create custom emails to send to attendees. After discussing this feature with the client, the team and the client agreed that this extension was a "nice but not required" feature and thus would only be implemented if there was additional time before the final delivery. For now, the team will focus on implementing functionality of the automatic email reminders/notifiers and, should time prove insufficient, will remove the "Email attendees" option from the drop-down menu.

<u>Event Leader Dashboard</u>
Based on the client's requirements, the design team created a separate dashboard for event leaders and admins. Event leaders are users who lead an event but may have lower authority/privileges than admins. Thus, the dashboard and user flow for event leaders is very similar to that of the admin but with a few restrictions/differences. Event leaders are able to see the same information and perform all actions available to admin except for downloading data from an event and adding/removing users from the event sign-ups. These actions are unavailable to event leaders and thus cannot be seen in the event leader dashboard. An example of this is shown below.



Event leader dashboard view (notice the missing actions)

4.4 Second Iteration Front-End Implementation

The front-end team worked on finishing the implementation of the event landing page layout given the client's feedback from the last iteration, beginning progress of the implementation of the administrator event landing page, implementing prototypes of the user (e.g., student, event leader, admin) dashboards, and linking the front-end and back-end, that is, implementing and testing a means for data to pass from back-end to front-end and vice versa.

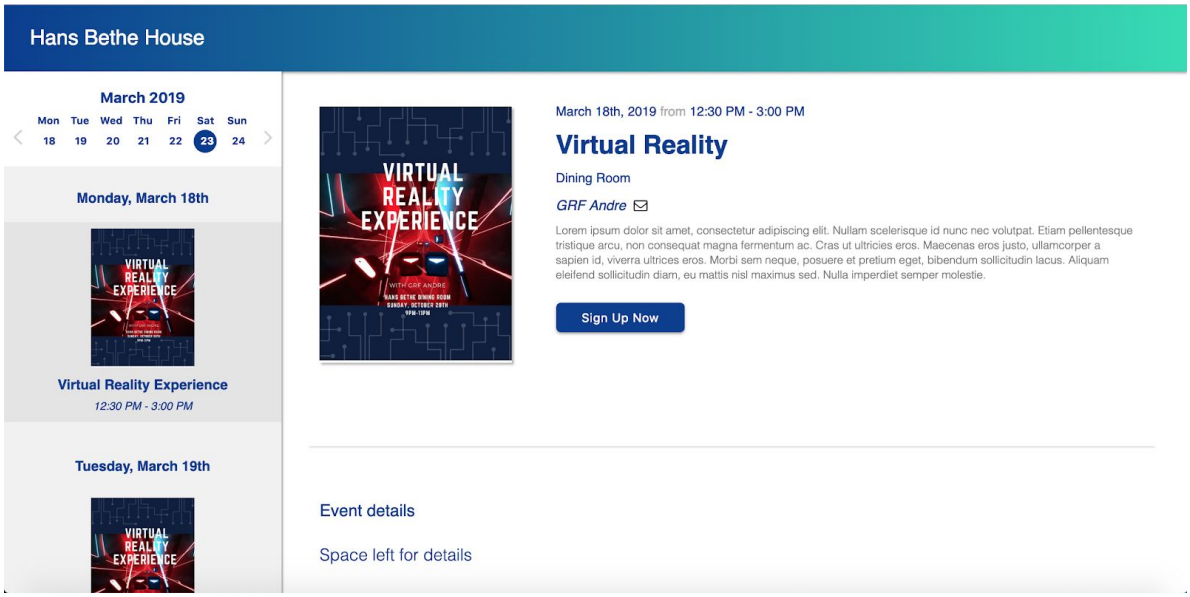
Finishing Event Landing Page

Finishing the event landing page mainly involved finishing the layout of the sidebar. This encompassed the section of the sidebar below the weekly calendar header in which the days and dates of the current week displayed are inserted along with events (poster graphic, event title, event start and end times) for each day of the displayed week. Dummy data was used to test the insertion of the event posters; the focus of the database component for this iteration was the creation of the database and preliminary testing (discussed in more detail in section 6.3).

One of the main goals for this iteration was to completely implement the user flow on event landing page. Hence, the functionality of the user click on the sidebar was implemented. Clicking between different events on the sidebar will dynamically change the event profile and display the details of the corresponding event that was clicked. The front-end team was able to establish a connection between the application and the database, and with this connection, details of events from the database have also been successfully loaded onto the application.

A few refinements were made to the event landing page based on feedback from the client. For example, the sidebar was made slightly wider, and after discussion about creating student and admin dashboards in the event profile, the display of how many spots are left in an event was deemed redundant, so this display was removed. A user will be able to see how many spots left by looking at the dashboard, which will have a list of taken spots as well as empty spots.

Student Event Landing Page Desktop View



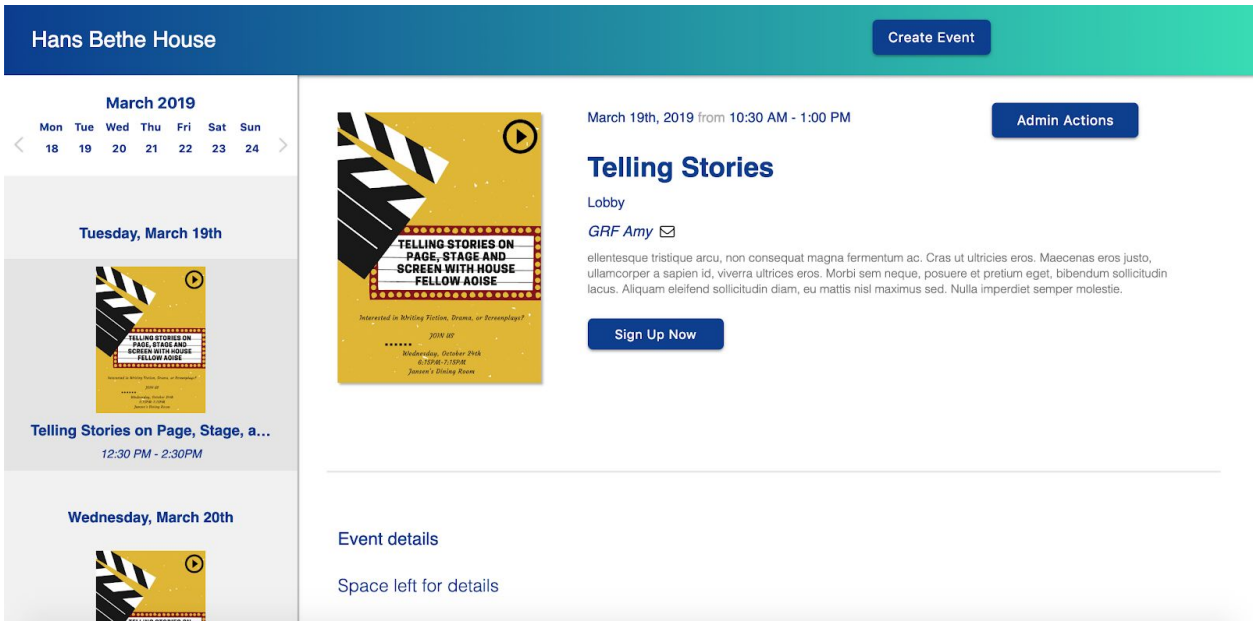
Student Event Landing Page Desktop View

Administrative Landing Page

After receiving feedback from the client on the administrative interface design mockups, the front-end team implemented most of the functionality for the administrative interface. The administrative interface is identical to the student event landing page, except with the added functionalities of administrator privileges. This includes the dropdown menu with administrator actions such as “Edit Event”, “Delete Event”, and “Download Data”, a

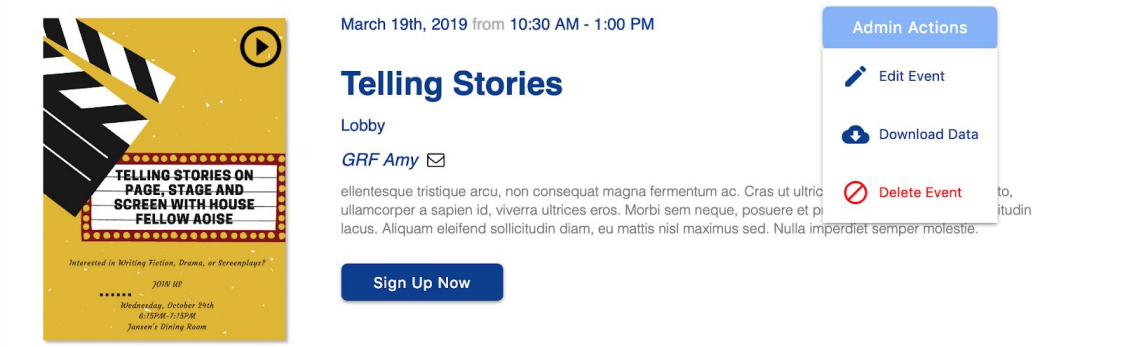
“Create Event” button, and a form for creating events upon clicking the “Create Event” button.

Administrative Landing Page Desktop View



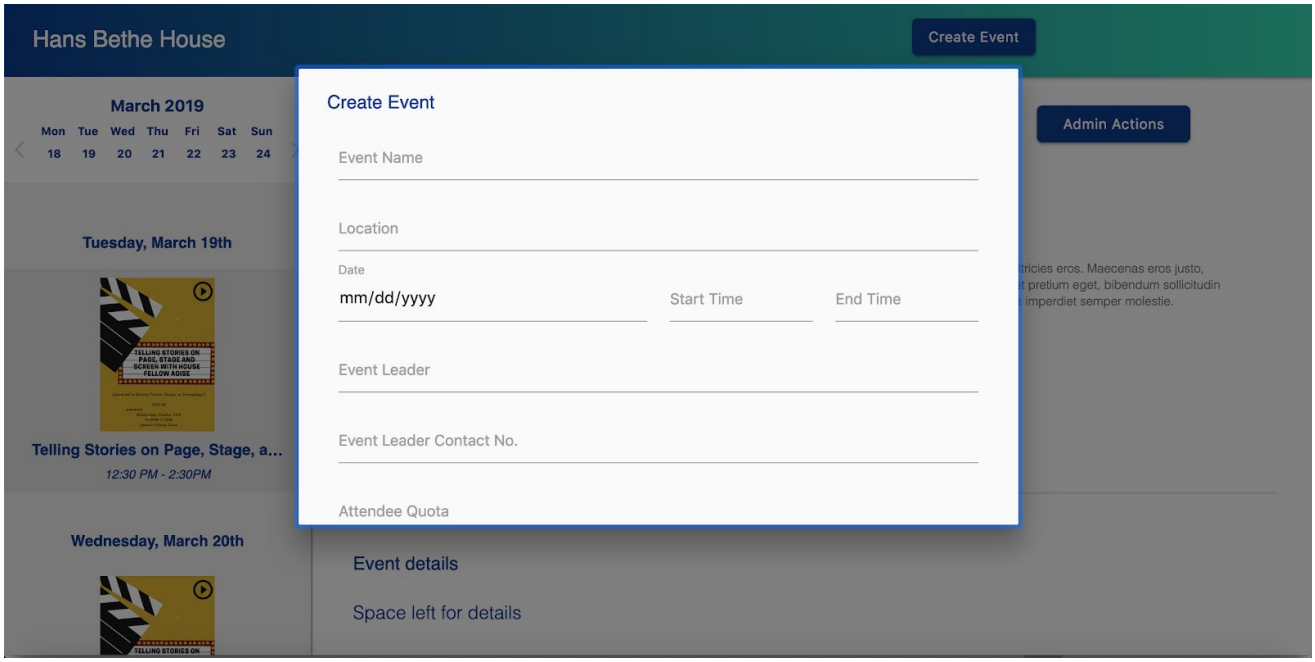
Admin Landing Page Desktop View

Dropdown

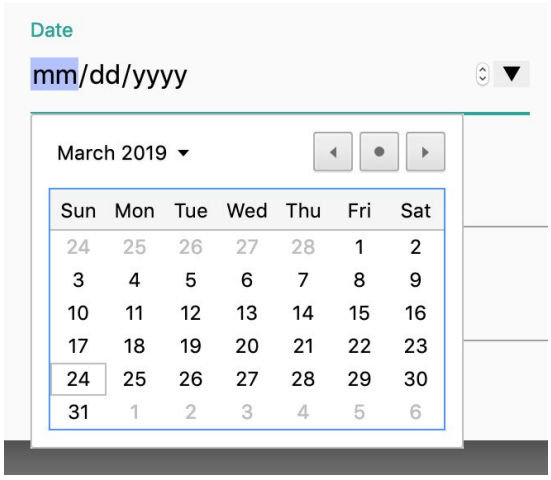


Dropdown in Administrative Landing Page

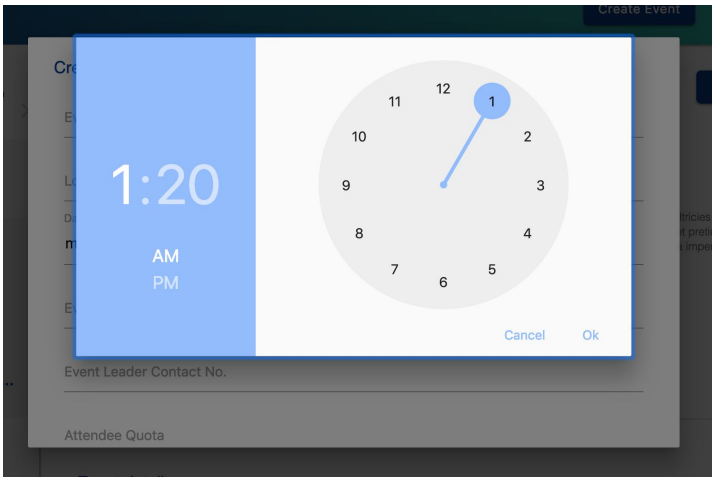
Create Event Form



Create Event Modal View



Date Picker



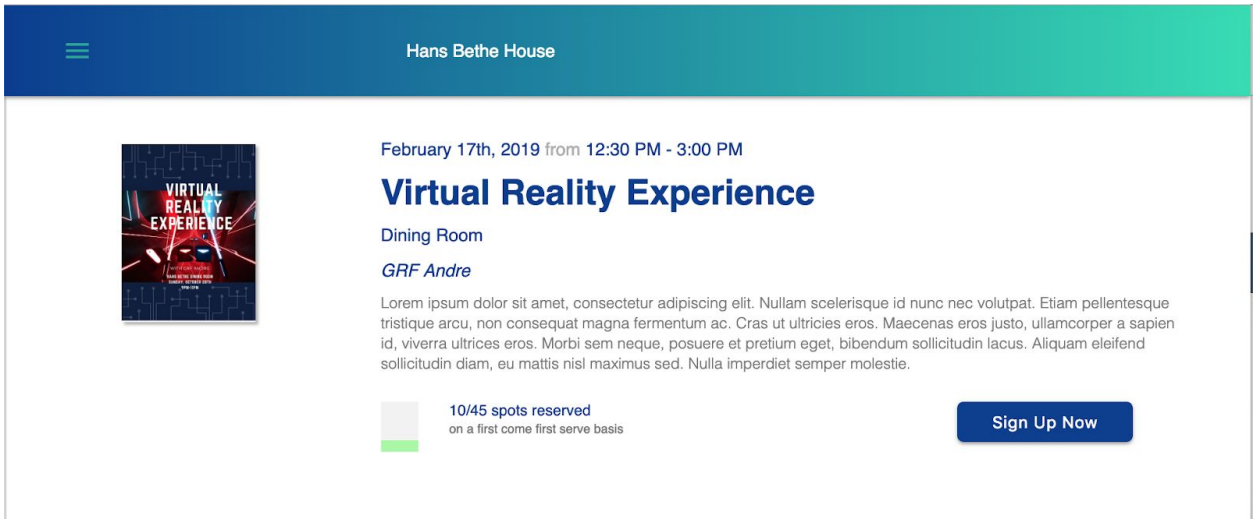
Time Picker

4.5 Front-End Mobile Implementation

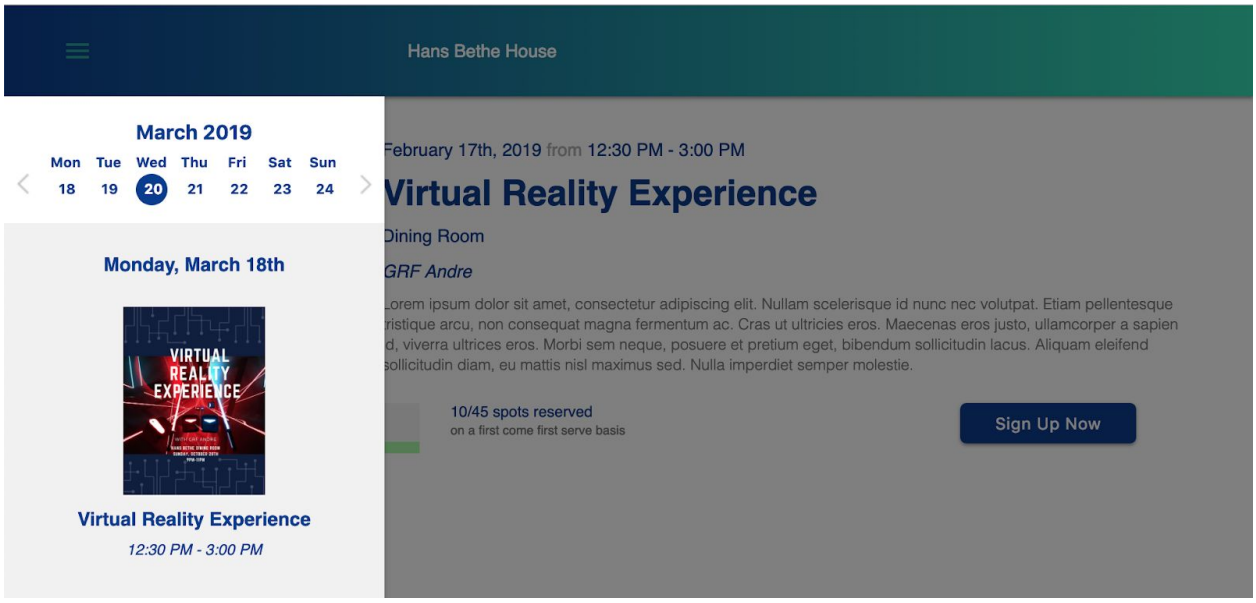
Since mobile responsiveness is an important requirement for the client, in this iteration, the front-end team implemented the event landing page so that the view would not look strange on smaller screens. For example, the sidebar collapses if the screen is smaller than a certain size, and the event details appear under the event poster when the poster becomes too small to see. When the sidebar is collapsed, there is an icon that appears in the navbar, and when this is clicked, the sidebar will slide out from the left. Adjustments

were made to the administrative “Create Event” form in order for it to be responsive on mobile. The time picker also appears slightly differently to fit a mobile screen. The date picker becomes a scrolling mechanism on the phone.

Tablet Size

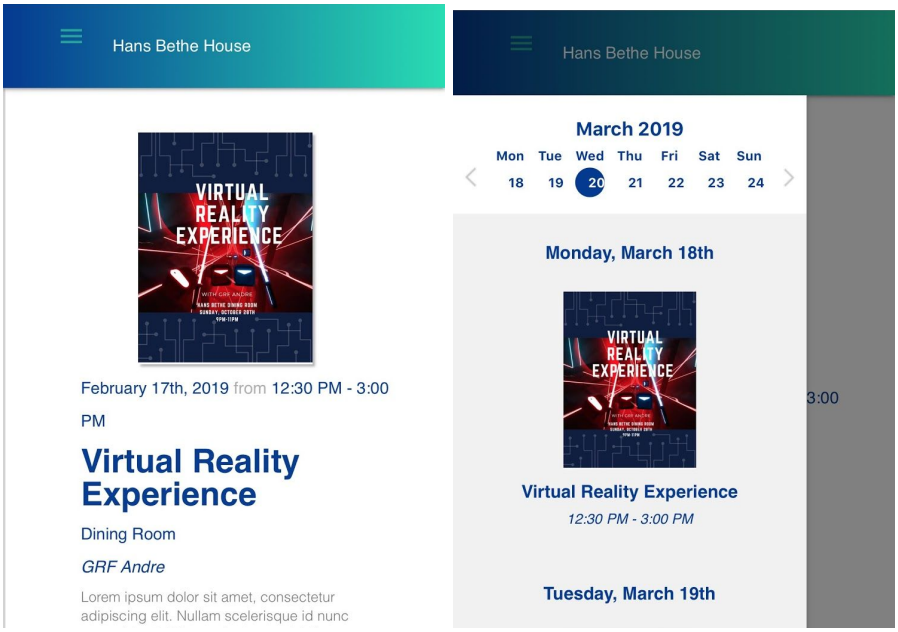


Student Event Landing Page Tablet Size, Sidebar Hidden



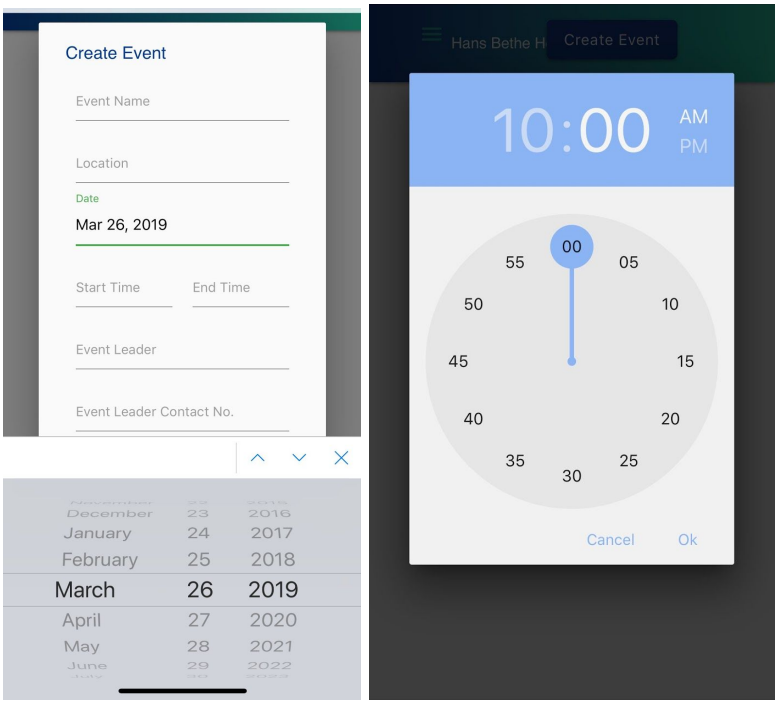
Student Event Landing Page Tablet Size, Sidebar Shown

Phone Size (Pictures below are on iPhone XS Screen)



Student Event Landing Page Mobile Size

Date Picker and Time Picker on Mobile



Administrative interface Date Picker and Time Picker on Mobile

5 User Interface Next Steps

5.1 Design Next Steps

In the next iteration, the design team will revise the featured events design according the feedback from the client. In addition, the design team will create a design for the login page, which will also be implemented by the front-end and back-end teams within the next iteration.

5.2 Front-End Next Steps

In the next iteration, the front-end team will work on completing the functionality of all pages. The front-end team will implement the design for the homepage that gives the user the ability to sign in.

In regards to the existing pages, the front-end team will implement the ability for users to toggle between “My events” and “Browse events” in the sidebar, implement the display of student and admin dashboards which will include the names of students who are going and how many spots are left, and refine the mobile responsiveness of the pages. The front-end team will also implement a “Featured Events” profile as the first event profile a user sees when they log in. Once the user clicks on another event, the “Featured Events” profile will disappear, and in order to see this profile again, the user can click on the “Bethe House” logo on the navbar. The front-end team will also work on the display of editing events on the admin interface.

The front-end team will also focus on loading event data from the database and updating the database based on user actions. For example, currently the sidebar has one event for each day. In the next iteration, the front-end team will work on loading events in the sidebar based on the events that exist in the database, so if there is no event for a certain day, this will not show up in the sidebar. When a student signs in, the database should insert a new row representing a student that has registered for an event. When an admin creates, deletes, or edits events, the corresponding updates will be made to the database. This will be done using templating.

6 Back-End Progress

6.1 Routing

Routing between the student event landing page, admin landing page, and the login page have been implemented. When the interface is launched locally, the first page that appears is now the login page instead of the event landing page. The login page consists of two buttons, one labeled “Student” and one labeled “Admin”. Clicking on either of

these buttons will redirect the user to the respective event landing pages, where the admin landing page will have the added privileges of admins. The routes for the event landing pages are connected to the database, so that all the events displayed on the page are taken from the database.

6.2 Email Notification Setup

The focus of the email notification setup implementation for this iteration involved three main objectives: (i) send an email when a student signs up for an event, (ii) send an email when a student removes him/herself from an event roster, and (iii) send an email when a student is removed the waitlist and added to the event roster.

To send each of these emails, the back-end team implemented a client-side script that utilizes the Fetch API in which the appropriate endpoint or URL for sending the desired email notification is fetched and the notification is sent to the student. Since the focus of the database component of the system focused on creation and preliminary testing (discussed in more detail in section 6.3) and login authentication was not set up during this iteration (discussed in more detail in section 6.5), the back-end team was unable to use real-time data to test the functionality of the mentioned email notifications implemented. Instead, the back-end team manually entered dummy data to test and ensure the implementation of these email notifications executed as expected.

6.3 Database

Creating the database was one of the main priorities of this iteration as the database would store all the necessary information needed to make the system work. After confirming the design of the EER diagram for the database, the database was initialized using MySQL Workbench. This software includes a very convenient tool called “forward engineering” that allows users to create the code for initializing a database from an EER diagram. After using this functionality and inspecting the code, the database was initialized.

After initializing the database and running the server locally to confirm that the correct tables were created, the next task was to fill the database was “dummy” data so the rest of the team could do testing. To create data for the entity tables “students”,” admins”, and “events”, the software from generatedata.com was used. This software is a free, GNU-licensed, random, custom, data generator that allowed us to quickly populate the database with data. Once the data for the entity tables was passed into the database, the intermediary tables, “Students_has_Events” and “Admin_has_Events”, also needed to be filled in with data by randomly creating relationships between students, events, and

admins. Two “stored procedures” were created in MySQL Workbench to do this, which essentially queried the data randomly from the three entity tables and created relationships to be put into the intermediary tables. Once this was done, the entire database was exported and uploaded to the project repository for the rest of the team to test and use.

As the team continues to test the database, future modifications may be made to the design of the database.

6.4 Login Authentication

During this past iteration, the back-end team followed up with SSIT on using CUWebLogin in regards how to apply to use the login authentication system and whether the Bethe OPS system would need to be reviewed to use the authentication software. Unfortunately, SSIT was not the most responsive and, therefore, there was much waiting on the team’s end and clarification on how the back-end team was to proceed with using Cornell’s authentication software in Bethe OPS was not immediately given.

In the end, SSIT recommended the team use Shibboleth — a standards-based, open source software from Internet2 which provides federated authentication/authorization for web-enabled services — for login authentication, and pointed the back-end team to resources and documentation concerning how to utilize Shibboleth. After looking through the provided and other resources, the back-end team agreed to proceed with Shibboleth, especially given that other Cornell applications and web services use the authentication software which means there is support the back-end team can turn to if need be when integrating Shibboleth.

In addition, in researching how to use Shibboleth with Node.js applications, the back-end team found and plans to use Passport, an authentication middleware for Node.js that was recommended for integrating Shibboleth into Node.js applications in regards to authentication.

6.5 Hosting

An instance for the website is launched using AWS EC2. All the current files on the project repository have been transferred to the virtual server. To edit the files, the team SSHs into the server and edits or creates files and directories using nano. As the project continues, the team will update the files on virtual server each time when it commits to the project repository. Right now, there are still some bugs in the virtual server, so it will not launch the website successfully when running the server.

7 Back-End Next Steps

Connecting to the Database

The next steps for the back-end side of the system will mainly focus on connecting appropriate components of the system to the database. That is, connecting the server to the database such that the following can be done:

- Query the database for event and user information to dynamically populate the user interface (e.g., getting an event's information and the names of students who signed up for the event to generate the event profile section on the event landing page)
- Query event information to insert into email notifications
- Update the database with data inputted on or received from the client side (e.g., storing new user information, students submitting event sign-up forms, updating the waitlist for an event)

Login Authentication

For login authentication, the back-end team will utilize the technologies mentioned in section 6.4 to implement the login authentication component of the system. The back-end team will reach out to the Identity Management Team at Cornell to inquire about using Cornell as the Identity Provider (IDP) needed for Shibboleth. The back-end team will also ask about using CUWebLogin as part of the authentication process, such is done for the Cornell Interlibrary Loan site that uses Shibboleth.

Hosting

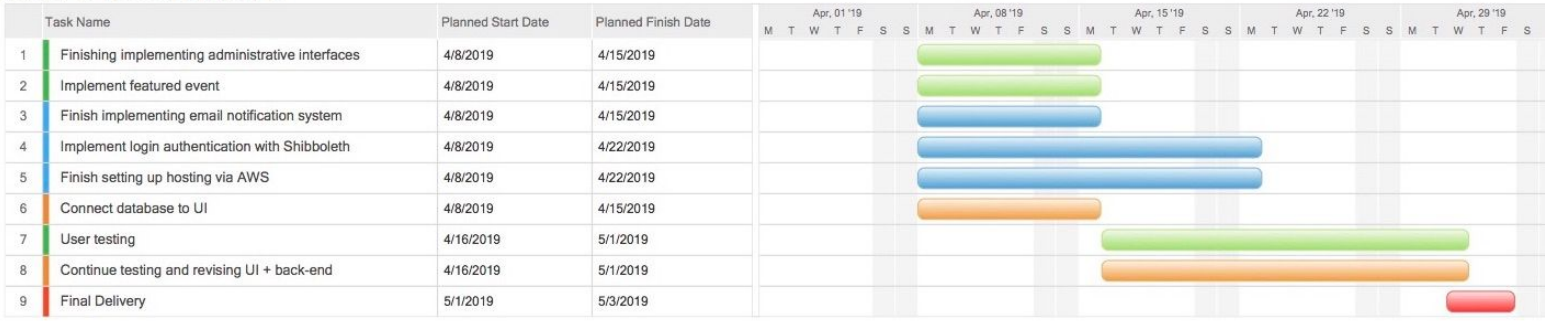
For hosting, the back-end team will continue testing and debugging in regards to transferring local files for the system to the server provided by AWS EC2. Right now, the file transfer process is mostly manual copy and paste. The back-end team tried various ways to upload local file systems as a whole using scp (secure copy), but no methods have worked out yet. The back-end team will continue to look for ways to update the virtual server more automatically. The back-end team will also need to launch another instance to host the database for the system as well.

8 Timeline and Tasks

The Gantt chart below shows the team's revised schedule for the remainder of the project. By the last milestone, the team hopes to connect the database to the server and thus the user interface; fully implement the user dashboards; implement the featured event page; complete the email notification system; implement login authentication; finish setting up hosting via AWS and deploy the system; formally start user testing with the system as a whole; and conduct program testing.

The green bars represent tasks to be done by the user interface team, the blue bars represent tasks to be done by the back-end team, the orange bars represent the tasks to be done by both the user interface and the back-end team, and the red bar represents the final delivery by the team to the client.

After Presentation 3



	Task Name	Planned Start Date	Planned Finish Date
1	Finishing implementing administrative interfaces	4/8/2019	4/15/2019
2	Implement featured event	4/8/2019	4/15/2019
3	Finish implementing email notification system	4/8/2019	4/15/2019
4	Implement login authentication with Shibboleth	4/8/2019	4/22/2019
5	Finish setting up hosting via AWS	4/8/2019	4/22/2019
6	Connect database to UI	4/8/2019	4/15/2019
7	User testing	4/16/2019	5/1/2019
8	Continue testing and revising UI + back-end	4/16/2019	5/1/2019
9	Final Delivery	5/1/2019	5/3/2019

Updated Gantt Chart