INDABA
FINAL REPORT

CS 467 Capstone Project

December 2019

Team Members

Paul Adams adamspa@oregonstate.edu

Everett Williams williaev@oregonstate.edu

I. Introduction

Over the past three months Team Indaba developed the Indaba Scheduling application in response to the complexity created by using numerous third party applications to make appointments. Indaba means gathering or conference in the Zulu language and is used widely throughout South Africa. The Indaba scheduling application simplifies scheduling activities at OSU by consolidating scheduling activities within one application. The use of various third party applications created the potential for missed meetings for attendees who did not have the required accounts. It also forced our OSU teammates to sign up for and monitor numerous redundant scheduling accounts, some of which may only be used once during a student's or faculty member's association with OSU. The use of third party services also created the potential for Family Educational and Privacy (FERPA) violations, possibly violating of a students' privacy rights. Faculty, students, and administrative personnel find the current state of affairs frustrating, unnecessary, and possibly in violation of privacy regulations.

Our client for this project was Donald Heer, an OSU ECE professor working towards his Ph.d. His motivation for the project is to simplify scheduling which is a common task performed throughout the institution and to ensure OSU is compliant with FERPA regulations. Mr. Heer specified the following requirements:

- Create a web tool on OSU web servers that uses OSU ONID credentials for authentication via Central Authentication Service (CAS)
- Webtool has a 'reserver' portal and an admin portal for seeing reservations.
- Webtool support Chrome, Firefox, Opera, and Edge browsers with identical functionality.
- If requested, reservation can have files attached viewable by the admins and by other users who have reserved the same time slot.
- Users can message all other users in a slot via a web form.
- Time slots can have 1-N reservation slots.
- Administrator can override user selections
- Front end web design uses Bootstrap 3 or 4
- Assume 1200 or 1280px widescreen

II. User Perspective

The Indaba Scheduling Application is designed to only be accessible to users who have valid ONID credentials. There are two user roles: event creators and event attendees.

Event creators' experience will begin by logging into Indaba via CAS. Then, they may navigate to a page to create a new event. Here, they can fill out a form with their event's details, and select time slots on a calendar-based interface. They will enter email addresses to which they would like to send invitations, and Indaba will generate and send these invitations. After an event is created, its creator can manage it in the web app via the event management portal. Here they can change event details, add or edit

time slots, send new invitations, and delete their event. They will also be able to view, edit, and delete existing reservations. Event creators may register for time slots in their own events if they choose.

The role of an event attendee begins when they receive an emailed invitation. To RSVP, they must follow a link and log into the web app using their ONID credentials. Once logged in, they can view the event details and choose to reserve one or more time slots or decline the invitation. Once an attendee has made a reservation, it will be visible on their personal homepage under the heading "Your Schedule". Adjacent to each reservation, there will be an area where users can send a message and upload a file to other attendees of their slot. If a user wishes to edit or delete their reservation, they may click on the event's name in their schedule which will redirect them to the event's reservation portal. A page listing past reservations will be accessible from the homepage and the menu

A users who have created events, the personal homepage will also display any events they are currently managing. Clicking on the name of an event they are managing will redirect them to the event management portal described above. A separate page listing past events will be accessible from the homepage and the menu.

III. Demonstration Instructions

Create and manage an event (Event Creator)

1. Log into https://indaba-scheduler.herokuapp.com/login. If the user presents valid ONID credentials the user will be redirected to the users homepage. Click the Create Event button to create a new event.
2. Clicking the create event button sends the user to Create New Event page. Fill out the following (see Figure 1 and 2):
    a. Event Name: Add a name of your choice.
    b. Description: Add an event description.
    c. Location: Add a location of your choice.
    d. Max attendee per time slot: 2
    e. Max reservations per attendee: 2
    f. Time Durations: Enter 1 hr and 15 minutes.
    g. Yes make names visible: Select this option.
    h. Calendar: Select two time slots by clicking the calendar. Make sure they are *after* today's date.
    i. Email Address: Enter your ONID email address and select Add emails.
    j. Finally, select Create event.

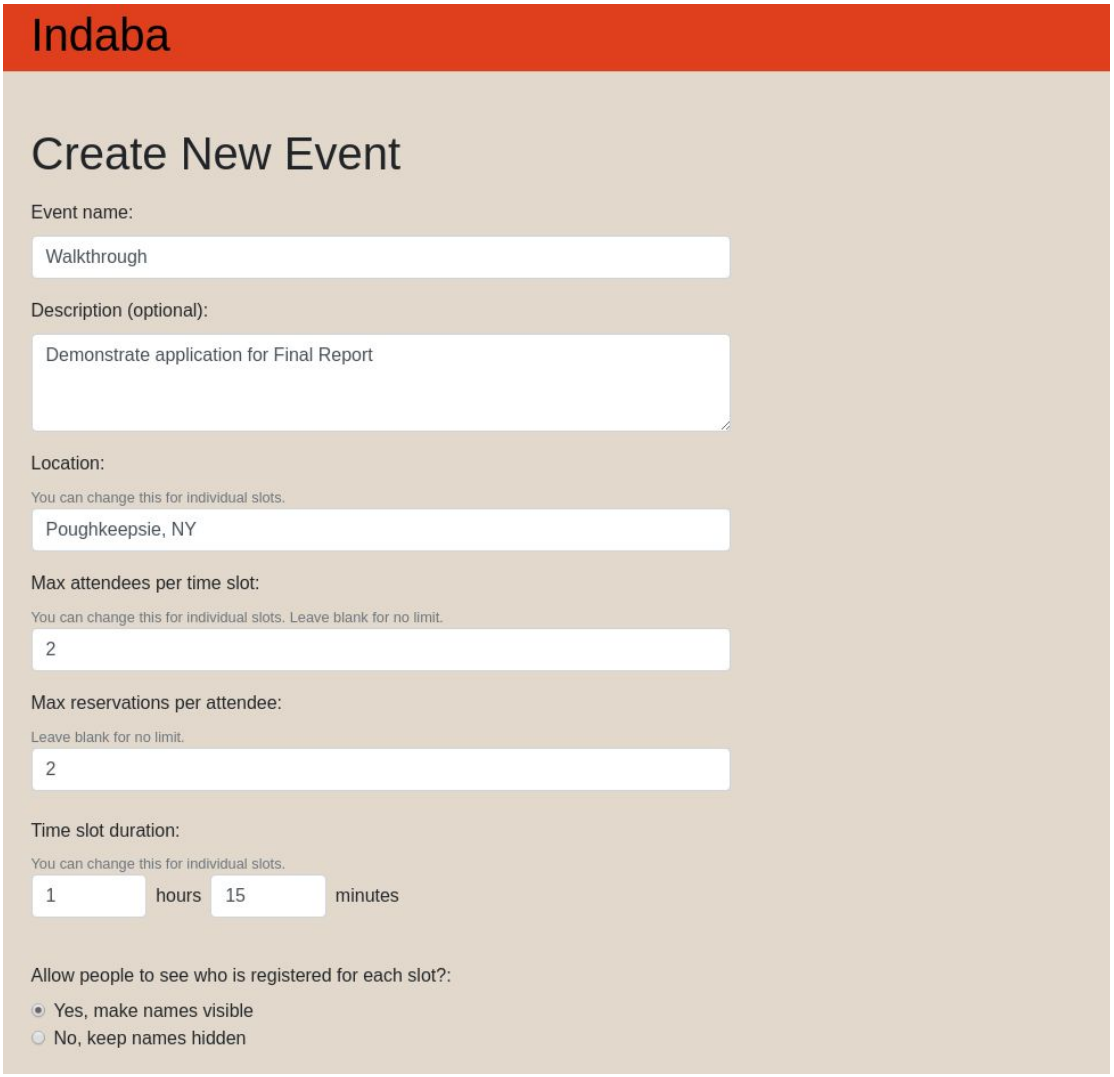
Indaba
Create New Event
Event name:
Walkthrough
Description (optional):
Demonstrate application for Final Report
Location:
You can change this for individual slots.
Poughkeepsie, NY
Max attendees per time slot:
You can change this for individual slots. Leave blank for no limit.
2
Max reservations per attendee:
Leave blank for no limit.
2
Time slot duration:
You can change this for individual slots.
1
hours
15
minutes
Allow people to see who is registered for each slot?:
Yes, make names visible
No, keep names hidden

Figure 1.

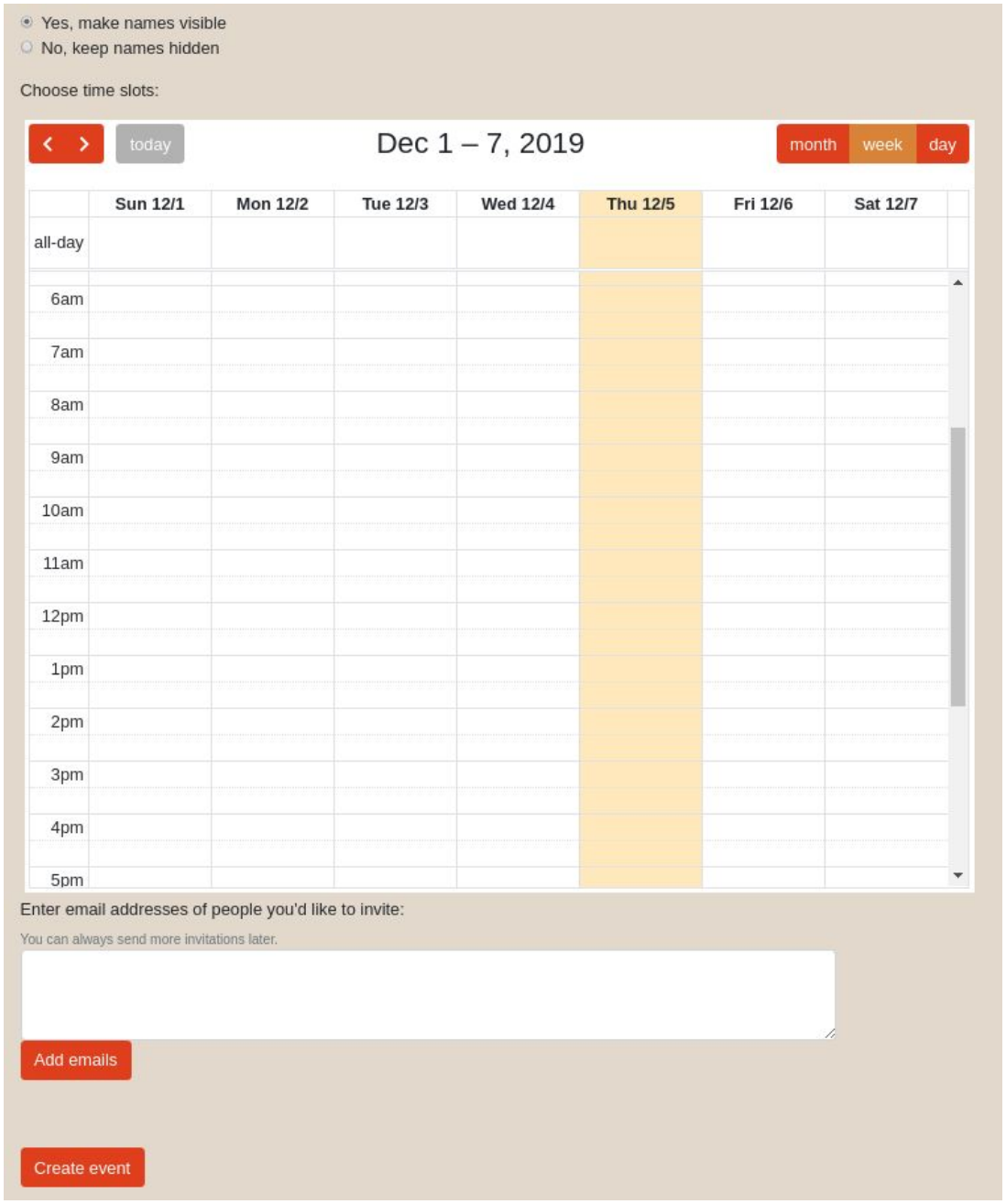

Figure 2

3. Selecting create event redirects to the Organizer Portal which allows the user to make updates to the event. Select a third time slot for the event then select Save time slot changes.

4. On the Organizer Portal select the Menu in the top right corner and then select Home. You'll be redirected to the home page where the new event will be displayed.
5. For the event in step 4 select Reserve time slot. You'll find the three slots you created on the calendar. Select all three slots to create a reservation in each. You'll only be able to select two of the slots because of 1e. Select Submit to register for the two slots. You'll be redirected back to the home page where the reservations will appear.
6. Select Cancel reservation to remove a reservation.
7. Select Make another reservation and follow step 5 to create a new reservation.
8. Return to the Menu in the upper right and select Create New Event. Follow step two to create a new event, but for this event select time slots prior to the current day. Use Menu to return to the home page. Notice the newly created event is not displayed.
9. Select Past events from the Menu. The past event from step 8 is displayed on this page.
10. From the Menu drop down select Home. On the home page select View/Edit details for the event you created in step 2. At the bottom of the Organizer Portal select Delete Event. You'll be redirected back to the home page and the event will no longer be present.

Make a reservation from email invite (Event Attendee)

11. To Demonstrate Event Attendee interactions, follow step 2 to create a new event and using your ONID email address in the email field. Click Add emails then check your email for the invitation.
12. Click the link in the email (you may need to check spam/junk) which takes you to the Make a reservation page. Follow step 4 to create a reservation and select Submit. You will be redirected to the home page where you will see new reservations.

IV. Software and Systems Overview

At a basic level, the Indaba Scheduler is client-server web application utilizing a relational database for persistent storage. The server uses the Node.js runtime environment to power a web app created using an Express framework. Indaba's architecture is based on the Model View Controller (MVC) pattern where the code is organized into roughly 3 categories.

The models are the database related functions, allowing for insertion, deletion, modification, and querying of values in the database. Although NoSQL (non-relational database) is a more common choice for Node.js applications, we used the MySQL RDBMS because the information we needed to store was highly relational and our user base is not expected to exceed several thousand users. Our server used the mysql2 driver package to interface with the database. Figures 3 and 4 represent the updated ER Diagram and Schema respectively.

INDABA ER DIAGRAM

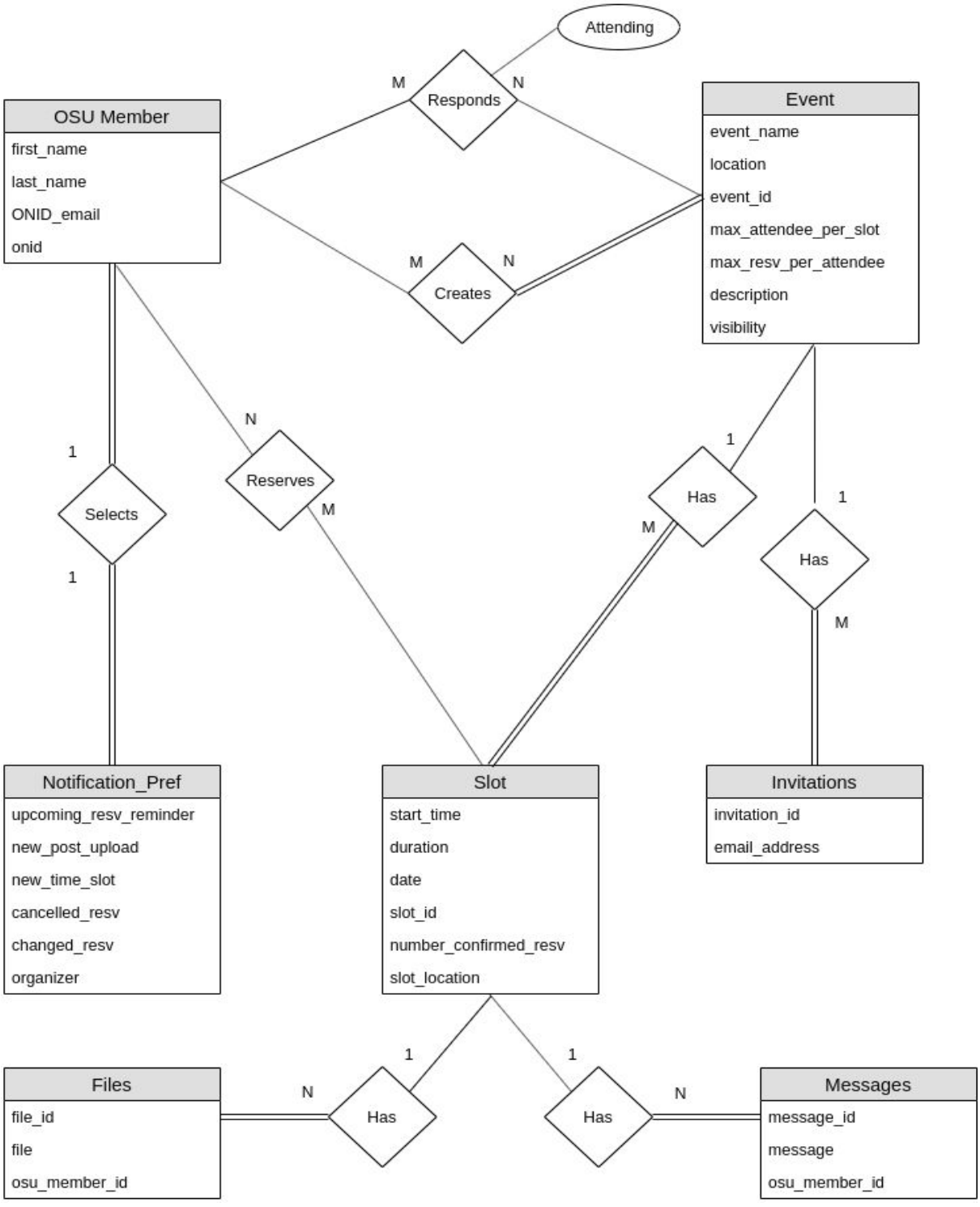

Figure 3

Figure 4

The views are the user interface. We created our views using handlebars templates. Before rendering a page, a controller on the server (described below) collects any necessary information by using models functions, processes that information into a

template-friendly format, and then plugs that information into the template. The client is delivered a fully-constructed HTML web page. One exception to this is the calendar, which is constructed on the client side with FullCalendar to reduce server load. Styling and layout for the views is provided by Bootstrap 4. We used both jQuery and vanilla JavaScript to facilitate dynamic behavior on web pages. Among this dynamic behavior is the conversion of dates and times into the user's local timezone.

The controllers, also called routes, are the glue that binds the application together by handling user requests. When a user navigates to a page on the site or initiates a post request by submitting a form, Express directs the request to the appropriate controller. Inside the controller, the details of a users request are parsed and the necessary actions are taken. This might include initiating authentication, validating input, interacting with the database via the models functions, constructing a view, managing the session, or redirecting to another page. All of the application's business logic is located in the routes.

V. Software Tools, Languages, APIs, and Libraries

Backend

Node.js:	server runtime environment (Javascript)
CAS:	Central Authentication Service via OSU

Node.js packages:

body-parser:	parse body of post requests
express:	web framework
express-handlebars:	templating engine
express-mysql-session:	create/manage session store in MySQL
express-promise-router:	add asynchronous functionality to express router
express-session:	session middleware
helmet:	set security-related HTTP headers
mysql2:	database driver
request:	HTTP request client
request-promise-native:	add asynchronous functionality to request client
sendgrid:	connect to SendGrid communications platform API
xml2json:	parse XML to JSON

Frontend

Bootstrap 4:	mobile-first front-end framework (HTML and CSS)
FullCalendar:	full-sized event calendar (JavaScript)
JavaScript + jQuery:	facilitate page interactions

Database

MySQL	relational database management system

Hosting

Heroku:	Node.js server
AWS:	database
SendGrid:	email

VI. Team Mate Contributions

The creation of our project plan was a collaboration between both team members. We began by meeting frequently to brainstorm and come up with a list of use cases. Paul directed front-end design of the project by creating wireframes for each anticipated page of the application. Everett took charge of the database design and created an ER diagram and schema. We worked together to come up with a build schedule that divided up the work.

The first phase of implementation involved setting up our development environment. Everett designed the database tables and set up hosting on AWS. Paul created a skeleton Express app hosted on Heroku and coordinated with OSU IT staff to set up a CAS service to handle authentication for the app.

Having set up a skeleton application, we started building core features. Paul created a basic template for the login and home pages and set up routes to serve them. Everett helped to connect authenticated users to the database. Both team members contributed to the creation of a display of a user's reservations on their personal homepage.

Next, we implemented a "Create Event" page. Paul designed the template, set up client-side input validation, and created the route on the server to store event details on the database. Everett added the FullCalendar package to the rendered page to allow users to click on a visual calendar to add time slots.

The "Manage Event" page was a true team effort, with both team members working on the requisite parts. Everett took full responsibility for the creation of the "Make a Reservation" page. This included template design, rendering of existing data, routing, and data storage. Paul Created the "Past Events" and "Past Reservations" pages, and set up emailed invitations. Both members contributed to testing and creation of final documentation.

VII. Deviations From Original Plan

There are several features that our team was unable to complete due to time constraints. As the quarter drew to a close and testing became a bigger focus we unfortunately were not able to implement the following pieces of our original plan:

Notifications system. We intended for users to be able to set notification preferences for a range of situations, such as choosing to be notified by SMS or email if the location of one of their reservations was changed. In our completed project we were only able to implement emailed event invitations.

Comment/file upload. Our original plan specified a feature for users to be able to post messages and upload files that would be visible to other users registered for the same time slot. We created tables in the database to support this, but ran out of time to implement the frontend/backend portions towards the end of our development cycle while we focused on debugging the core app.

Calendar integration. We planned for users to be able to download their upcoming reservations in .ics format, or to otherwise integrate with cloud-based calendars.

Multiple organizers. Our database was designed to allow for multiple users to have administrative control over a single event. However, the user-facing portion of the app does not expose this capability, instead only allowing the original event creator to have managerial control.

Display "not attending" responses. When users respond to an invitation, they have the option to respond that they are unable to attend. Our app records this information in the database, but we did not have time to incorporate these responses into the "Reservations" table on the "Manage Event" page.

VIII. Project Continuance

Unfortunately, the first step to project continuance is to refactor all server-side code into PHP. OSU's IT infrastructure does not currently support scalable Node.js web apps so continuation of this project will require porting the server code. Fortunately, much of the code is straightforward to convert. All of the SQL queries in the models functions can be repurposed with minimal changes. In addition, the handlebars templates should be able to be reused via the lightncandy package, an open-source PHP implementation of handlebars. Alternatively, the templates can be refactored by finding another solution for code that is contained within double curly braces: {{variable_name}}, or is placed between handlebars helper functions such as:

{{#each}}
...
{{/each}}

and

{{#if}}
...
{{else}}
...

{{/if}}

The heavy lifting of a refactor will be rewriting the routes. To this end, detailed comments have been provided in the source code. Additionally, the function for emailing invitations (/helpers/email.js) will need to be rewritten completely to use a university SMTP server instead of the current SendGrid service. We had intended to make this change ourselves, but were unable to because we couldn't port our Node.js app to university servers.

Another detail which will need to be managed is setting up a new CAS service for the app. We set up a student-based service which had limited functionality (though sufficient for our needs) and will expire at the end of the Fall 2019 term. A new service can be set up by sending a request to iamteam@oregonstate.edu. Our request included the following details:

Service Name: Indaba Scheduler

Service URL: https://indaba-scheduler.herokuapp.com

Description: Indaba is a meeting scheduling application for Oregon State University students, faculty, and staff. The app allows users to organize events with 1 or more time slots and send invitation emails directing people to RSVP. Invitees can register for the time slots of their choice or respond that they are unable to attend. An intuitive interface allows users to manage reservations they have made and events they have organized.

Attributes required: ONID, full name, email address

A new request should update the URL to http://eecs.oregonstate.edu/education/indaba, the webspace set up to host the app once it is ported. In order for the new CAS service to be permanent, the request must be sent by the person or department that will take long-term ownership of the project.

We will not be providing the credentials to our AWS-hosted database or to the Heroku hosting service, as these will not be used in the future. Instead, users can access the database hosted on university servers with the below details:

Server: engr-db.engr.oregonstate.edu
Port: 3307
Database: indaba

Admin User: indaba
Admin Password: ojYzRaSv
Read-Only User: indaba_ro
Read-Only Password: cgN6wpdw
phpMyAdmin Interface: https://tools.engr.oregonstate.edu/phpMyAdmin/?server=2

This database matches the one hosted on AWS in structure, though it is not populated with any data. A fresh database can also be constructed using the following file in our project directory: /docs/indaba.sql.

The ported website will be hosted in the following filespace: /nfs/ca/info/eecs_www/education/indaba A new group should contact Don Heer to be added to the unix group required to access this filespace.

All of our project's code can be accessed on our GitHub repository: https://github.com/pauladams12345/indaba. Simply fork the project to begin work.

After the existing site has been ported, a group should look through the features described in section "Deviations From Original Plan" to get ideas for what to work on next. In particular, they would be encouraged to add the comment/file upload to registered slot feature, as this was a direct client request and the most important feature we were unable to complete.

IX. Conclusion

Building the Indaba Scheduling Application allowed the members of Team Indaba to develop skills crucial to a software developer. The Indaba project allowed the team to gain expertise in integrating various software technologies, conducting stakeholder analysis, executing User Interface and User Experience (UI/UX) design decisions, and following common software engineering practices.

As noted throughout the report, we worked with several technologies. A critical decision at the beginning of the project was choosing which technologies would best meet our requirements, with easy collaboration being a top priority. AWS, Heroku, and Github were chosen because they allowed a level of collaboration that would not have been attainable on OSU servers. Other technologies were considered such as Mongo DB and postgreSQL, but were not chosen because they did not suit the needs of the project.

In terms of stakeholder analysis, we met several times with our client, Mr. Heer. After each meeting the team would discuss the best way to incorporate the clients requirements and then develop or adjust our plan to create the functionality. This involved discussions about progress and use cases that were critical to our development efforts.

As neither team member had extensive experience with UI/UX design, an interesting part of the project was developing everything from the color schemes on each web page to the name of the project itself. Another excellent experience that allowed for creativity and interesting discussions was how to create interactions and visualizations that entice the user to continue using the application.

From the software engineering perspective the team used Git for version control. This allowed both team members to gain a clear understanding of the strengths and pitfalls inherent in version control. In general, using git worked well and was critical to project development.

In conclusion, this proved to be an exciting and challenging project. It forced both team members to interact with technology they don't normally use like FullCalendar, AWS, Heroku and SendGrid. It also stressed the importance of teamwork and being supportive and positive in moving the project forward. We hope our efforts move forward for further development and ultimately lead to a fully endorsed OSU scheduling application.