COLLEGE OF ENGINEERING

Electrical Engineering and Computer Science

INDABA SCHEDULER

Live app: https://indaba-scheduler.herokuapp.com/
GitHub repository: https://github.com/pauladams12345/indaba

Team members: Everett Williams – williaev@oregonstate.edu
Paul Adams – adamspa@oregonstate.edu

USER PERSPECTIVE

Indaba is a meeting scheduling application for OSU students, faculty, and staff. Using the app, users can:

- Log in using your ONID credentials: No need to create a new account! Only university-affiliated users can access the app.
- Organize new events: Enter event details like name and description. Select time slots on a visual calendar. Specify details like meeting location and maximum attendees.
- Manage your events: Change event details, modify or create time slots, and view reservations from the event management page.
- Send invitations: Enter email addresses to send invitations from either the event creation page or event management page.
- Register for time slots: Follow the link in your invitation email to make a reservation (see Fig. 2). Select 1+ available slots or respond that you cannot attend.
- View/manage reservations. See reservation details on your personal homepage (see Fig. 1). Make additional reservations or cancel existing reservations by clicking the buttons

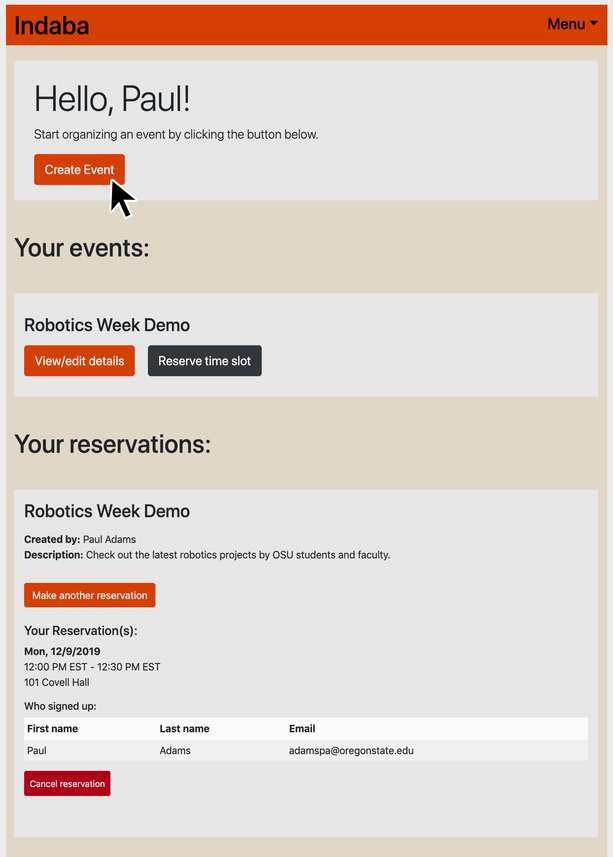

Fig. 1 Personal homepage

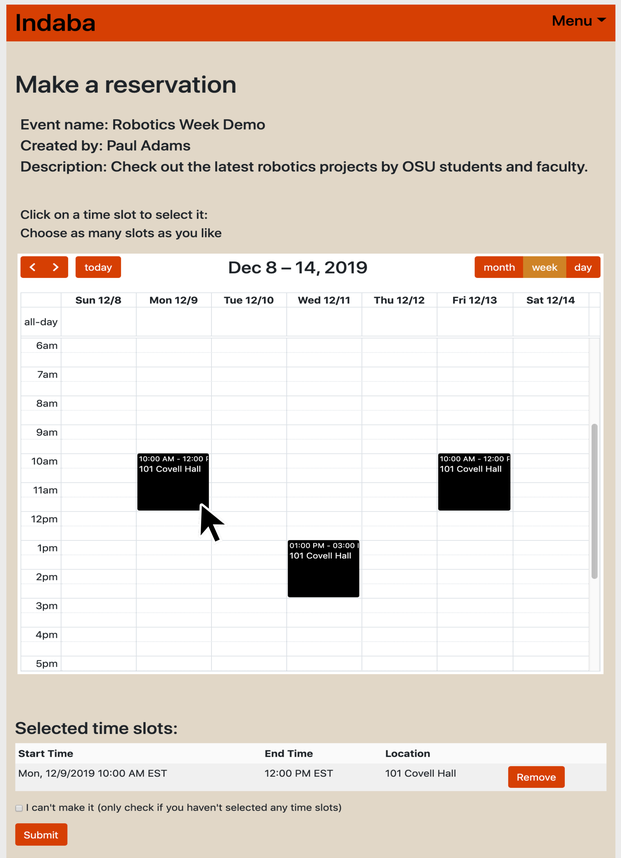

Fig. 2 "Make a reservation" page

IMPLEMENTATION DETAILS

Indaba is powered by a Node.js web server running an Express application. Pages are constructed server-side using handlebars templates before being delivered to the client via the Heroku cloud hosting platform.

The app was designed using an MVC architecture, abstracting the functions into:

- Models: database interactions (see Fig. 3)
- Views: user interface (see Fig. 4)
- Controllers: user request handling and business logic implementation (see Fig. 5).

KEY FEATURES

- CAS authentication: ONID credentials used for log-in, allowing FERPA compliance
- Local time zones: Automatic date/time conversion to a user's local time zone
- Mobile-friendly user interface: Bootstrap 4 framework allows the same code to serve both desktop and mobile users
- Graphical calendar: Simple UI for organizers to create/modify slots and for attendees to reserve slots

TECHNOLOGIES

Back end

- Node.js: server runtime environment (Javascript)
- CAS: Central Authentication Service via OSU
- Notable Node.js packages:
  - express: web framework
  - express-handlebars: templating engine
  - mysql2: database driver

Front end

- 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

```
// Create row in Reserve_Slot with the given information
module.exports.createReservation = async function(onid, slotId) {
    try {
        const connection = await sql.createConnection(dbcon);
        await connection.query(
        "INSERT INTO `Reserve_Slot` " +
        "(`fk_onid`,`fk_slot_id`) VALUES (?,?)",
          [onid, slotId]);
        connection.end();
    }
    catch (err) {
        console.log(err);
    }
};

// Delete reservation for the corresponding onid and slot Id
module.exports.deleteReservation = async function(onid, slotId){
    try{
        const connection = await sql.createConnection(dbcon);
        await connection.query(
        "DELETE FROM `Reserve_Slot` " +
        "WHERE `fk_onid` = ? and `fk_slot_id` = ?;",
         [onid, slotId]);
        connection.end();
    }
    catch (err) {
        console.log(err);
    }
};
```

Fig. 3 Model: Manipulating reservation in database

```
<h6>{{#each this.reservations}}</h6>
<div class="row d-flex justify-content-center">
  <div class="col-12 col-lg-10">
    <p><p class="mb-0 date">{{this.slot_date}}</p></p>
    <p class="mb-0"><span class="time">{{this.start_time}}</span> - <span class="time">
    {{this.end_time}}</span></p>
    <p class="mb-0">{{this.slot_location}}</p>
    <h6 class="mt-3">Who signed up:</h6>
  </div>
</div>
```

Fig. 4 View: Display reservations to user

```
// Display a user's Past Reservations page
router.get('/past-reservations', async function (req, res, next) {
    // If there is no session established, redirect to the landing page
    if (!req.session.onid) {
        res.redirect('/login');
    }

    // If there is a session, render user's past reservations
    else {
        let context = {};
        context.eventsManaging = await createsEvent.getPastUserEvents(req.session.onid);
        context.eventsAttending = await helpers.processPastReservationsForDisplay(req.session.onid);
        context.firstName = req.session.firstName;
        context.stylesheets = ['main.css'];
        context.scripts = ['convertISOToLocal.js'];
        res.render('past-reservations', context);
    }
});
```

Fig. 5 Controller: Serve the "Past reservations" page

Oregon State University