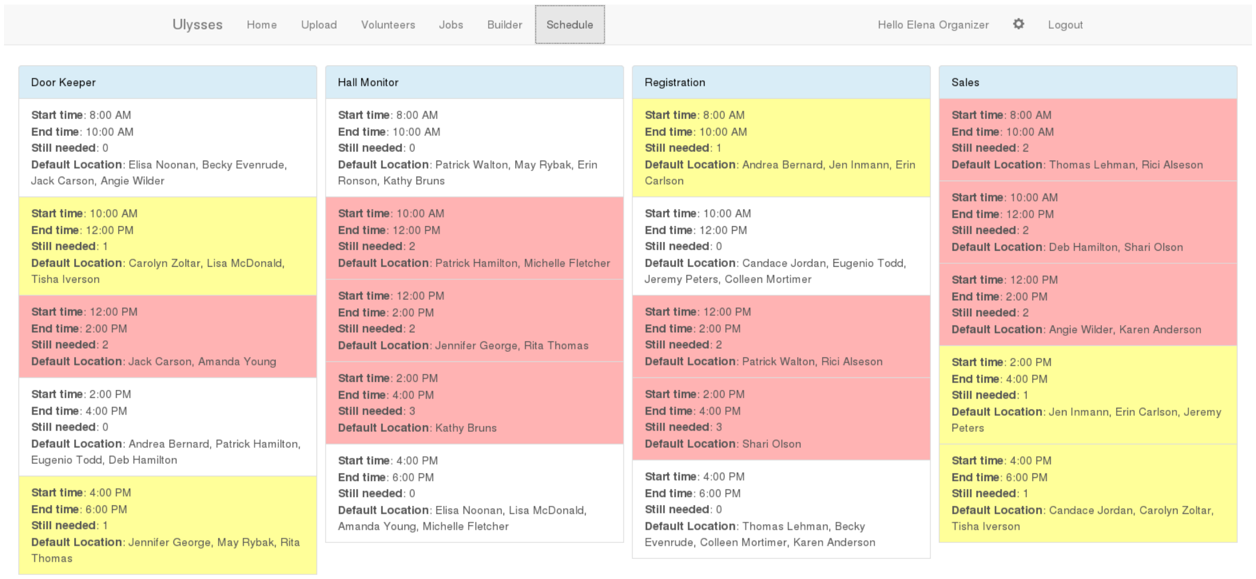
Ulysses: Scheduler of the Mind

Shawn Seymour

1 The Project

- A web-based tool created to schedule volunteers for an event
- Created for use by the Odyssey of the Mind competition
- Organizers can easily create jobs, time slots, and manually schedule volunteers
- Scheduler algorithm schedules volunteers to jobs automatically



Door Keeper

Start time: 8:00 AM
End time: 10:00 AM
Still needed: 0
Default Location: Elisa Noonan, Becky Evenrude, Jack Carson, Angie Wilder

Start time: 10:00 AM
End time: 12:00 PM
Still needed: 1
Default Location: Carolyn Zoltar, Lisa McDonald, Tisha Iverson

Start time: 12:00 PM
End time: 2:00 PM
Still needed: 2
Default Location: Jack Carson, Amanda Young

Start time: 2:00 PM
End time: 4:00 PM
Still needed: 0
Default Location: Andrea Bernard, Patrick Hamilton, Eugenio Todd, Deb Hamilton

Start time: 4:00 PM
End time: 6:00 PM
Still needed: 1
Default Location: Jennifer George, May Rybak, Rita Thomas

Hall Monitor

Start time: 8:00 AM
End time: 10:00 AM
Still needed: 0
Default Location: Patrick Walton, May Rybak, Erin Ronson, Kathy Bruns

Start time: 10:00 AM
End time: 12:00 PM
Still needed: 2
Default Location: Patrick Hamilton, Michelle Fletcher

Start time: 12:00 PM
End time: 2:00 PM
Still needed: 2
Default Location: Jennifer George, Rita Thomas

Start time: 2:00 PM
End time: 4:00 PM
Still needed: 3
Default Location: Kathy Bruns

Start time: 4:00 PM
End time: 6:00 PM
Still needed: 0
Default Location: Elisa Noonan, Lisa McDonald, Amanda Young, Michelle Fletcher

Registration

Start time: 8:00 AM
End time: 10:00 AM
Still needed: 1
Default Location: Andrea Bernard, Jen Inmann, Erin Carlson

Start time: 10:00 AM
End time: 12:00 PM
Still needed: 0
Default Location: Candace Jordan, Eugenio Todd, Jeremy Peters, Colleen Mortimer

Start time: 12:00 PM
End time: 2:00 PM
Still needed: 2
Default Location: Patrick Walton, Rici Alseson

Start time: 2:00 PM
End time: 4:00 PM
Still needed: 3
Default Location: Shari Olson

Start time: 4:00 PM
End time: 6:00 PM
Still needed: 0
Default Location: Thomas Lehman, Becky Evenrude, Colleen Mortimer, Karen Anderson

Sales

Start time: 8:00 AM
End time: 10:00 AM
Still needed: 2
Default Location: Thomas Lehman, Rici Alseson

Start time: 10:00 AM
End time: 12:00 PM
Still needed: 2
Default Location: Deb Hamilton, Shari Olson

Start time: 12:00 PM
End time: 2:00 PM
Still needed: 2
Default Location: Angie Wilder, Karen Anderson

Start time: 2:00 PM
End time: 4:00 PM
Still needed: 1
Default Location: Jen Inmann, Erin Carlson, Jeremy Peters

Start time: 4:00 PM
End time: 6:00 PM
Still needed: 1
Default Location: Candace Jordan, Carolyn Zoltar, Tisha Iverson

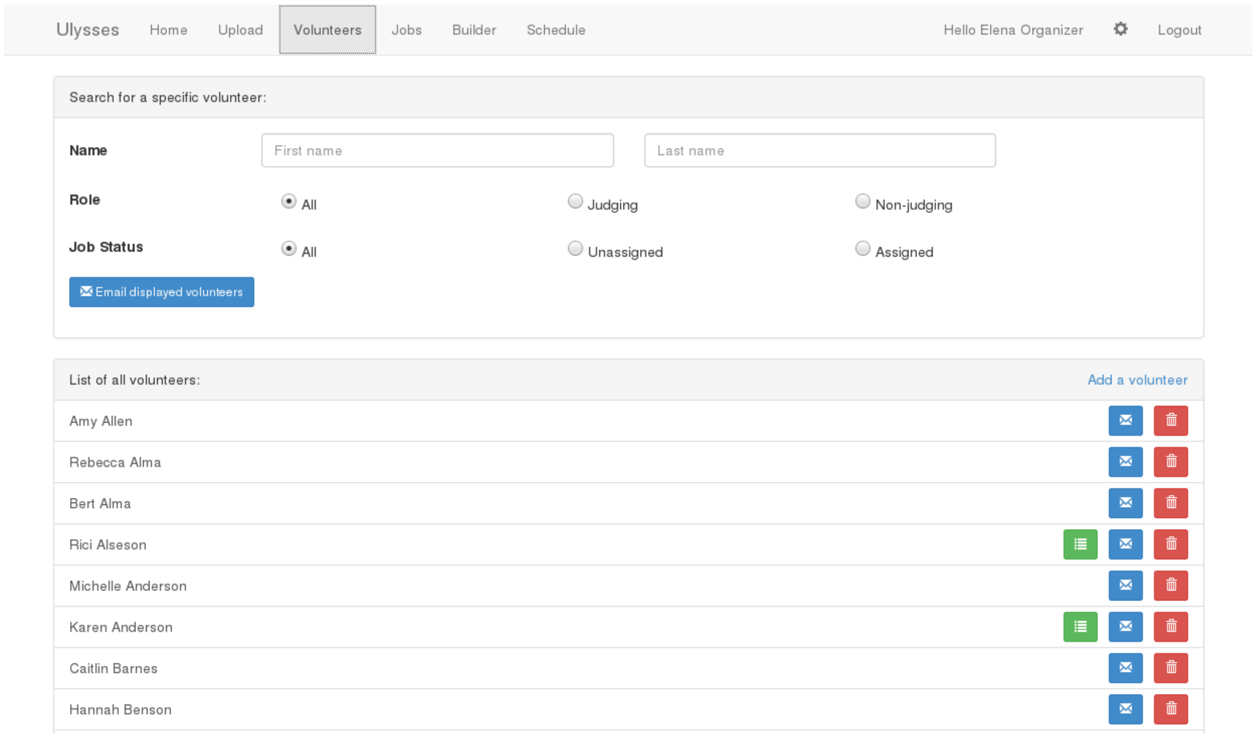
Figure 1: A generated schedule with some slots full, partially full, and mostly empty

2 The Tools

- JavaScript with MongoDB, Express, Angular JS, and Node.js (MEAN)
- Grunt build system, Bootstrap, Yeoman AngularJS-Fullstack Generator
- GitHub and Git for version control, Atom and WebStorm for writing code
- Protractor, Karma, and Jasmine for end-to-end and unit testing
- Pair programming and agile software development practices

3 Project Statistics

- Four main iterations over a period of eight weeks
- Worked in teams of four through seven
- Over 15,000 lines of code and more than 250 commits



Search for a specific volunteer:

Name: First name / Last name

Role: [● All] ○ Judging ○ Non-judging

Job Status: [● All] ○ Unassigned ○ Assigned

Email displayed volunteers

List of all volunteers: Add a volunteer

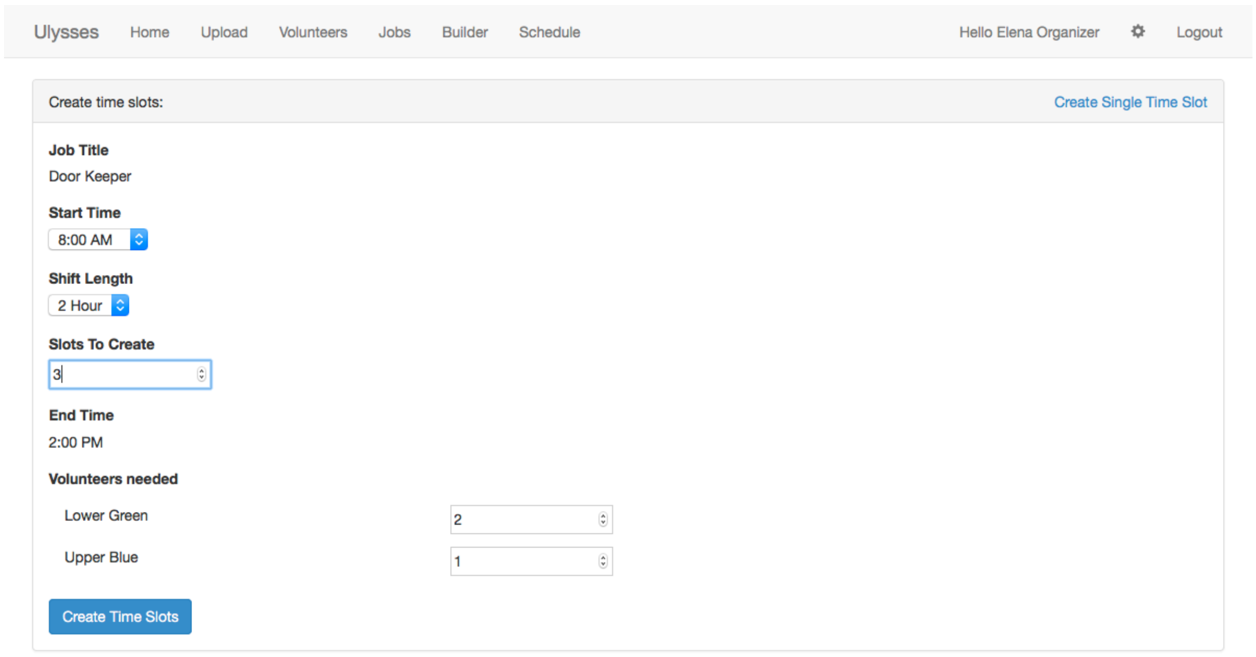
Amy Allen
Rebecca Alma
Bert Alma
Rici Alseson
Michelle Anderson
Karen Anderson
Caitlin Barnes
Hannah Benson

Figure 2: The list of volunteers with schedule viewing, emailing and sorting abilities

4 My Contributions

- Generated initial project, planned the layout and nature of time slots and jobs
- Designed and programmed the master schedule view
- Developed editing abilities and design of the volunteer detail page
- Implemented the ability to assign locations to jobs and volunteers to certain locations
- Helped implement the scheduling algorithm to schedule volunteers to the organizer-defined time slots while checking for conflicts
- Wrote initial end-to-end testing of the website along with a guide on how to add more end-to-end testing

5 Implementations



Create time slots:

Job Title
Door Keeper

Start Time
8:00 AM

Shift Length
2 Hour

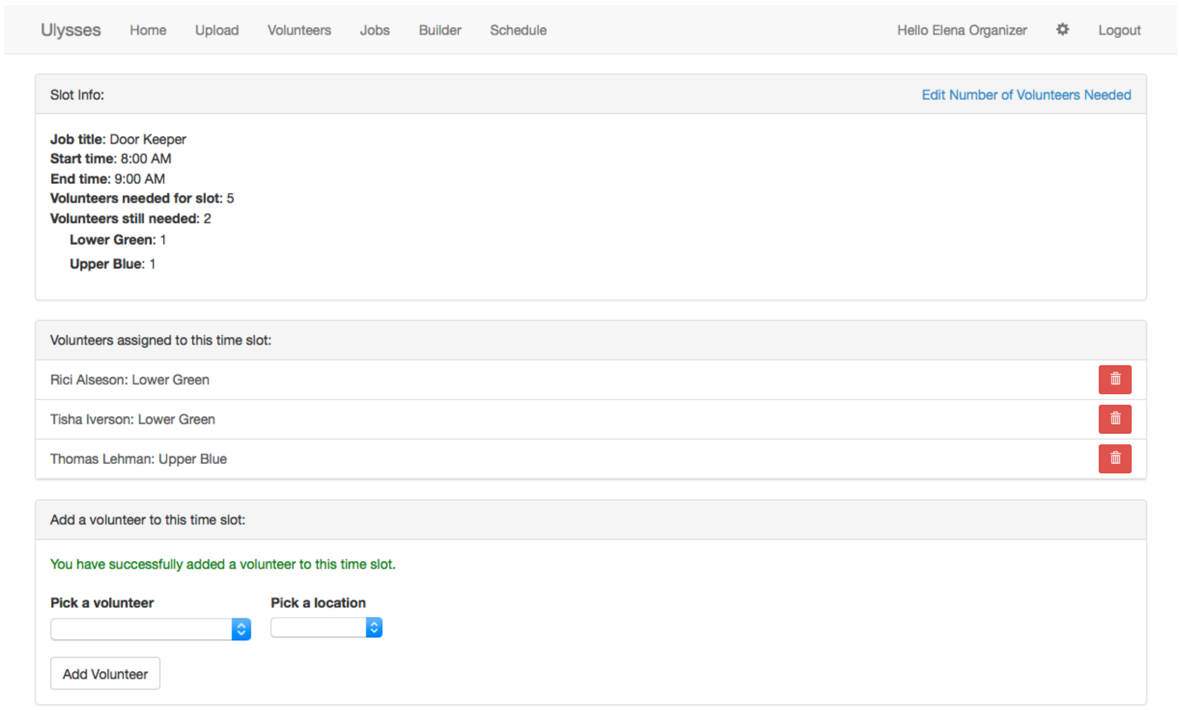
Slots To Create
3

End Time
2:00 PM

Volunteers needed

Lower Green	2
Upper Blue	1

Figure 3: Adding multiple time slots to a job and defining volunteers needed



Slot Info:

Job title: Door Keeper
Start time: 8:00 AM
End time: 9:00 AM
Volunteers needed for slot: 5
Volunteers still needed: 2
Lower Green: 1
Upper Blue: 1

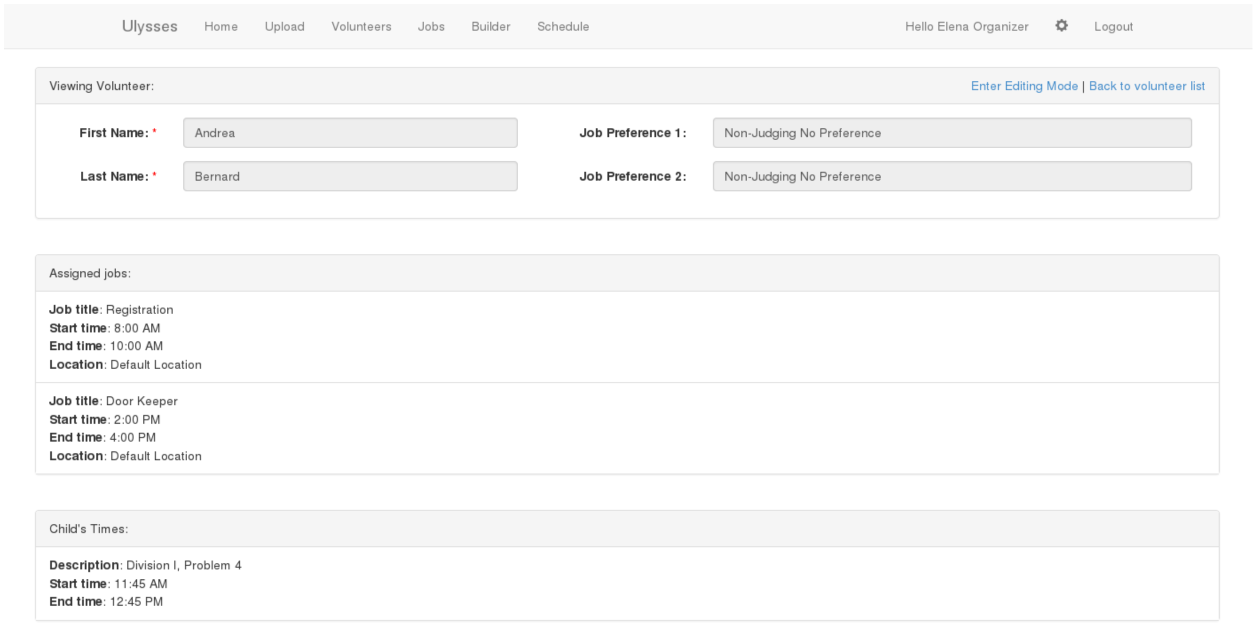
Volunteers assigned to this time slot:

Rici Alseson: Lower Green
Tisha Iverson: Lower Green
Thomas Lehman: Upper Blue

Add a volunteer to this time slot:

You have successfully added a volunteer to this time slot.

Figure 4: Adding volunteers to locations in a certain time slot



Viewing Volunteer:

First Name: *	Andrea	Job Preference 1:	Non-Judging No Preference
Last Name: *	Bernard	Job Preference 2:	Non-Judging No Preference

Assigned jobs:

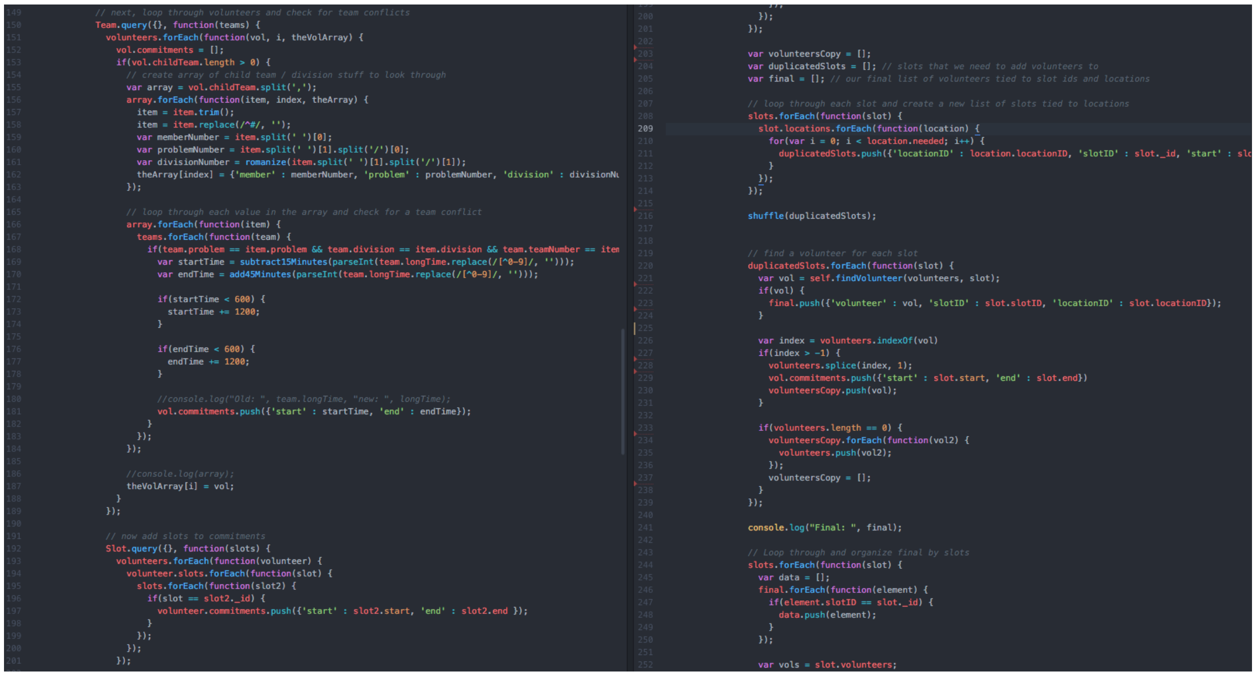
Job title: Registration
Start time: 8:00 AM
End time: 10:00 AM
Location: Default Location

Job title: Door Keeper
Start time: 2:00 PM
End time: 4:00 PM
Location: Default Location

Child's Times:

Description: Division I, Problem 4
Start time: 11:45 AM
End time: 12:45 PM

Figure 5: The layout and editing abilities for volunteer details with list of assigned jobs



```
        // next, loop through volunteers and check for team conflicts
        Team.query({}, function(teams) {
          volunteers.forEach(function(vol, i, theVolArray) {
            vol.commitments = [];
            if(vol.childTeam.length > 0) {
              // create array of child team / division stuff to look through
              var array = vol.childTeam.split(',');
              array.forEach(function(item, index, theArray) {
                item = item.trim();
                item = item.replace(/^#/, '');
                var memberNumber = item.split(' ')[0];
                var problemNumber = item.split(' ')[1].split('/')[0];
                var divisionNumber = romanize(item.split(' ')[1].split('/')[1]);
                theArray[index] = {'member' : memberNumber, 'problem' : problemNumber, 'division' : divisionNu
              });

              // loop through each value in the array and check for a team conflict
              array.forEach(function(item) {
                teams.forEach(function(team) {
                  if(team.problem == item.problem && team.division == item.division && team.teamNumber == iten
                    var startTime = subtract15Minutes(parseInt(team.longTime.replace(/[^0-9]/, '')));
                    var endTime = add45Minutes(parseInt(team.longTime.replace(/[^0-9]/, '')));

                    if(startTime < 600) {
                      startTime += 1200;
                    }

                    if(endTime < 600) {
                      endTime += 1200;
                    }

                    //console.log("Old: ", team.longTime, "new: ", longTime);
                    vol.commitments.push({'start' : startTime, 'end' : endTime});
                  }
                });
              });

              //console.log(array);
              theVolArray[i] = vol;
            }
          });

          // now add slots to commitments
          Slot.query({}, function(slots) {
            volunteers.forEach(function(volunteer) {
              volunteer.slots.forEach(function(slot) {
                slots.forEach(function(slot2) {
                  if(slot == slot2._id) {
                    volunteer.commitments.push({'start' : slot2.start, 'end' : slot2.end });
                  }
                });
              });
            });
```

```
              });
            });

            var volunteersCopy = [];
            var duplicatedSlots = []; // slots that we need to add volunteers to
            var final = []; // our final list of volunteers tied to slot ids and locations

            // loop through each slot and create a new list of slots tied to locations
            slots.forEach(function(slot) {
              slot.locations.forEach(function(location) {
                for(var i = 0; i < location.needed; i++) {
                  duplicatedSlots.push({'locationID' : location.locationID, 'slotID' : slot._id, 'start' : slc
                }
              });
            });

            shuffle(duplicatedSlots);


            // find a volunteer for each slot
            duplicatedSlots.forEach(function(slot) {
              var vol = self.findVolunteer(volunteers, slot);
              if(vol) {
                final.push({'volunteer' : vol, 'slotID' : slot.slotID, 'locationID' : slot.locationID});
              }

              var index = volunteers.indexOf(vol)
              if(index > -1) {
                volunteers.splice(index, 1);
                vol.commitments.push({'start' : slot.start, 'end' : slot.end})
                volunteersCopy.push(vol);
              }

              if(volunteers.length == 0) {
                volunteersCopy.forEach(function(vol2) {
                  volunteers.push(vol2);
                });
                volunteersCopy = [];
              }
            });

            console.log("Final: ", final);

            // Loop through and organize final by slots
            slots.forEach(function(slot) {
              var data = [];
              final.forEach(function(element) {
                if(element.slotID == slot._id) {
                  data.push(element);
                }
              });

              var vols = slot.volunteers;
```

Figure 6: Code for the scheduling algorithm written via pair programming