CSE 331: Software Design & Implementation

Homework 9 (due Friday, May 31st at 11:00 PM)

In this assignment, you will implement the client and server portions of an application described below. This assignment consists entirely of coding work. Submit your final version to "HW9 Coding" on Gradescope. Turning in your work for this assignment is a little trickier than usual, so follow these steps carefully!:

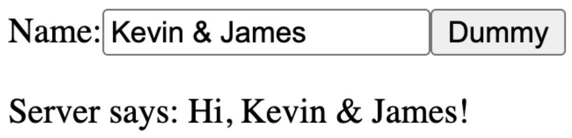
- `cd` into the directory that contains the `/client` and `/server` directories.
- Delete the `node_modules` directories from each directory (you can do this manually or use the command: `rm -r client/node_modules && rm -r server/node_modules`). (The `rm` command will remove and the `-r` makes it recursive, to remove directories and all their contents.)
- Generate a zipped file containing both of these folders
  - **On Mac** you can run: `zip -r submission.zip client/ server/` (The `zip` command compresses the specified arguments (`client/ server/`) into a `zip` file under the name `submission.zip` and again is, `-r`, recursive to get the full contents of the folders.)
  - **On Windows** you can select both folders, right click, and select 'Send to' > 'Compressed (zipped) folder'. You'll want to rename the zip that is created to 'submission.zip' so you can find it.
- Go to Gradescope and select the `submission.zip` file that was created in the last step.
- Make sure you get all the autograder points! It *does not check correctness* (i.e. no staff tests), but you should make sure it passes because it verifies that we can run your app manually. If you decide to work on your app some more after turning it in, you'll need to run `npm install --no-audit` in both of the directories again to get your `node_modules` back.

You can start by checking out the starter code using the command:

```
git clone https://gitlab.cs.washington.edu/cse331-24sp/cse331-24sp-materials/hw-wedding.git
```

Install the node modules, with `npm install --no-audit`, in both the `client` and `server` directories.

If you start the client and server and open `http://localhost:8080` in Chrome, you will see a dummy application that simply sends your name to the server when the button is pressed and shows the response message that comes back (which just says hi).



Server says: Hi, Kevin & James!

We included that code to give you a basic working example of client/server communication, but you can (and should) remove it when you start writing the actual application, which is described on the next page.

Read the entire spec before getting started. Understanding the requirements and making a comprehensive design is essential to being successful on this assignment.

James is engaged!! So now he and his fiancée, Molly, are wedding planning. There are lots of details to organize, one big task being the guest list. They need some way to easily input information and keep track all the guests they are inviting, their guests' plus ones, everyone's dietary restrictions, and so on. What better tool for this task than a web application!

Conveniently, James has a whole class of students who have been practicing building apps all quarter long! So he has delegated the task to you to design and build an app to manage his guest list.

(And rumor has it, the student with the best app gets an invite to the wedding.)

1. Surely You Guest (60 points)

Write a client side wedding RSVP web application, meeting the "Required Functionality" detailed in the following pages.

When implementing your application, it is, as usual, a good idea to crash if the program has a bug, but you are not allowed to crash on invalid inputs from the user. (Those are not bugs!) You must either show an error message or prevent the user from submitting invalid inputs.

Thoroughly test your application. Like in homework 8, the UI portions of this assignment are best tested with manual testing (which we will, of course, perform on your app), but any complex utility functions in your client code (think similar to problem 1 of hw-squares), should be tested with unit tests.

When grading we will be looking for high quality code (including organization, documentation, testing) and app functionality (including error handling).

Note that `let` variable declarations are not allowed on the client side.

2. Paint the Town Wed (40 points)

Our client side app is great for adding all the guest information, but the app data is not persistent. If the web page reloads, the whole guest list will be lost! We also want to allow guests access the app from a different tab/window, so they can set preferences like their dietary restrictions and if they're bringing a plus one. To enable this, we need back up the guest data in a server.

Implement a server with routes that allow you to store and load the necessary data. Change your client side app to store and fetch guest information from the server.

Write unit tests and document your server. Again, we will be looking for high quality code while grading.

Note that, unlike homework 8, you need to write all the client-server communication code (fetch requests) yourself. Since you are writing them yourself, you can integrate these requests into your components like we've shown in lecture examples instead of placing them in a separate file as in homework 8.

3. Extra Credit: Feeding Time at the New (0 points)

Add any new features that seem useful! You will get points for any feature that works correctly and seems like it would be valuable to the user.

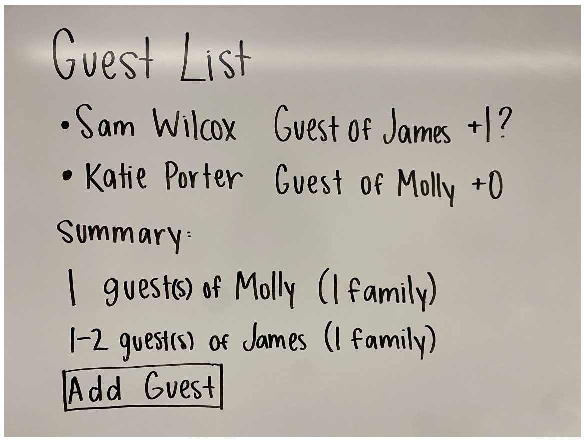
Note: adding CSS alone is not eligible for extra credit. You're definitely encouraged to add CSS, but extra credit will only be awarded to submissions that have additional *functionality* beyond the requirements.

Required Functionality

James and Kevin sketched out a design for the look of the wedding RSVP app that they'd like you to implement. This section describes the minimum, required functionality they'd like in the app and includes sketches of one way to provide that functionality to the user. This video **includes a walk-through of a staff attempt at the app**. You do not need to make your application look exactly like this, nor must the functionality be identical. It is fine to make changes that make the application *more* functional.

View current Guest List

Users must be able to view the list of all guests that have been added so far. Each guests name must be listed along with an indication of whether they are a guest of James or Molly and if they are bringing at most 1 additional guest, if known.



Guest List
• Sam Wilcox Guest of James +1?
• Katie Porter Guest of Molly +0
Summary:
1 guest(s) of Molly (1 family)
1-2 guest(s) of James (1 family)
Add Guest

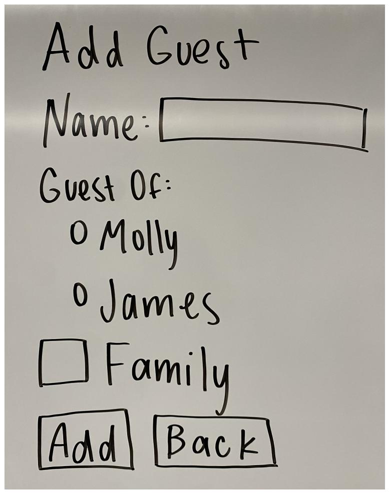
Along with the list of individual guests, there should be a summary of the number of guests of Molly and of James that have been added to the list and, for each, the number of those invited guests that are family.

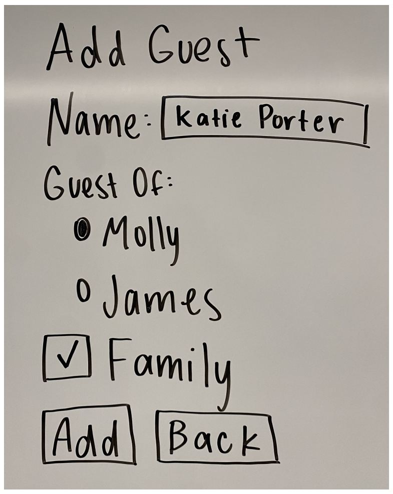
Because guests are allowed to bring an additional guest but might not, if the attendance of any additional guests is unconfirmed, the number of guests should be shown as a range from the minimum to maximum number of guests that could attend, given the number that are unconfirmed.

(Continued on the next page...)

Add a guest

James and Molly must be able to add guests and specify their name, whether they are a guest of either James or Molly, and if they are family. If they decide not to add the guest, they should be able to go "back" to the guest list page and the guest will not be added.





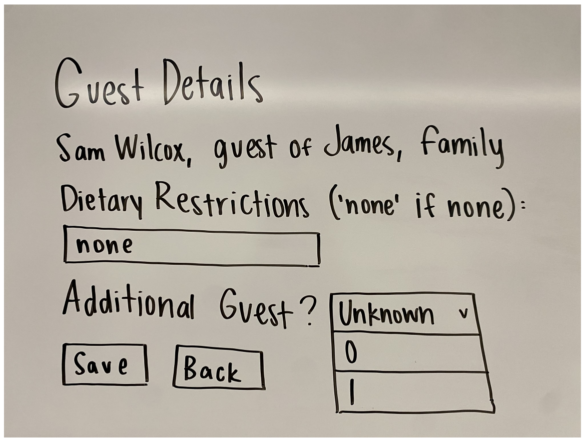
All of these details must be specified in order for the guest to be added, otherwise an error message should appear indicating what's missing.

Ideally we would prevent duplicate guests, but handling this case gets tricky because, as we'll see next, some information about guests needs to be editable, so we will not require you to handle it.

(Continued on the next page...)

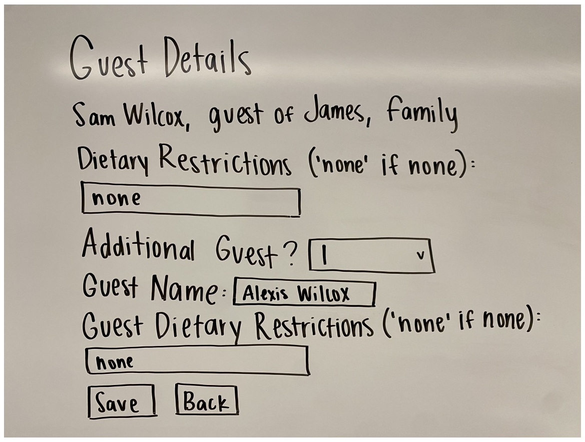
Add guest details

Once a guest has been added, they must be able to set additional details about themselves and their additional guest, if they're bringing one. Clicking on the names of guests in the Guest List should open a page with fields to enter their dietary restrictions and if they're bringing an additional guest.



Guest Details
Sam Wilcox, guest of James, family
Dietary Restrictions ('none' if none):
none
Additional Guest?
Unknown v
0
1
Save
Back

If they have selected that they are bringing 1 additional guest, additional fields to set their additional guest's name and dietary restrictions should be available.



Guest Details
Sam Wilcox, guest of James, family
Dietary Restrictions ('none' if none):
none
Additional Guest? 1 v
Guest Name: Alexis Wilcox
Guest Dietary Restrictions ('none' if none):
none
Save
Back

The guest's dietary restrictions and the name and dietary restrictions of their additional guest (if they are bringing 1 additional guest) must be specified in order for the guest details to be updated, otherwise an error message should appear indicating what's missing.

If they decide not to update these details, they should be able to go "back" to the guest list page and the guest will not be changed.

(Continued on the next page...)

Tips & Notes

- Sketch out your design on paper first to divide the app into components and map how the data will be stored and flow between them before you code anything.
- Utilize your resources! The section and lecture code (especially `Auction`) and prior assignments (especially hw8) give you amazing examples to borrow from.
- Create types and helpers to parse those types! Notice how in the lecture example, we defined a custom type for `Auctions` and created a `parseAuction` function to parse out each field of the record and verify it was the correct type, formatting the data into an `Auction` object.
- The calculation to determine the range of number of possible guests is tricky! We recommend creating a utility function to do this outside of your components, designing your loop/recursion carefully, and using your reasoning skills from this course to get it right!
- Don't get stuck on the HTML, we care more about testing your skills in other areas, not your knowledge of HTML. If you're unfamiliar with which elements you need to render what you see in our app outline, **please ask**.
- If you are running short on time, you can exclude the dietary restrictions for the guest, the dietary restrictions for the additional guest, and/or whether the guest is family or not, and only receive a small deduction (per attribute skipped) compared to skipping other features.
- If you are *really* short on time, just focus on the client-only version of the app from task 1. The client-only version is worth 60% of the points!
- **Have fun and start early!** Students often enjoy the freedom in this assignment and the opportunity to use your skills, but there are also many potential bugs, so give yourself time to deal with those bugs as well as enjoy it.

Congratulations on completing your last app of 331!! You have designed and implemented a functional client-server application starting from only sketches, way to go!