CSCI 4661/5661 Homework #3
CRUD Adventures

This assignment is going to be reminiscent of the blog app that we developed together in class, though with a twist in theming!

Summary:

For this assignment, you'll be making another simple game. Arguably even simpler from a gameplay standpoint than the last one, although we'll be leveraging plenty of new stuff from the class that we've learned since then.

In the previous game, you controlled a brave hero dueling with a foul villain. In this game, you'll be taking a much more bureaucratic role. Here, you'll be managing a guild of daring adventurers. You'll hire new heroes, send them out on missions, and tally the spoils of their journeys. You can then send them out on even more dangerous missions, enabling them to bring back ever increasing amounts of gold.

Your task is to create such a game as this!

Objectives:

This assignment is meant to be a way for you to practice several concepts of the class, and hopefully be a means for you to express some creativity and have some fun while doing it! The major part of this assignment though is to give you a chance to experiment making an app that relies on a Context object and a Provider (to make your app's data available to all aspects of your application) and to create a suite of CRUD operations to be able to change your app's data easily. CRUD, you'll recall, stands for Create, Read, Update and Delete.

As mentioned above – these are the concepts we first introduced when we started our blog post app. Except instead of creating, reading, updating, and deleting blog posts, now we'll be doing the same with brave heroes as we send them out for fame and fortune!

Breakdown

Perhaps the most important aspect of this assignment is the use of a context object to store your data. You'll recall that we did this in class with the blog app. The data in that app was a single array that was full of blog post objects. Here, our data will also be an array of objects, but each object will represent a hero that we have hired.

Although you are *welcome* to make your heroes *more* complicated than this, at minimum, each hero object should have the following properties:

id

Identifier so that you know which hero to update/delete/show data about, etc.

name

Just a fun name! A string

level

an int – an indication of how strong they are. You can spend gold to gain more levels. Gaining levels increases your health and power.

power

Used against the "challenge rating" of adventures. The higher the power, the greater the chance of success. Goes up when leveled up.

maxHealth

The maximum amount of health the hero can have. Increases on level up.

currentHealth

The current amount of health they have. Decreases when sent on adventures.

gold

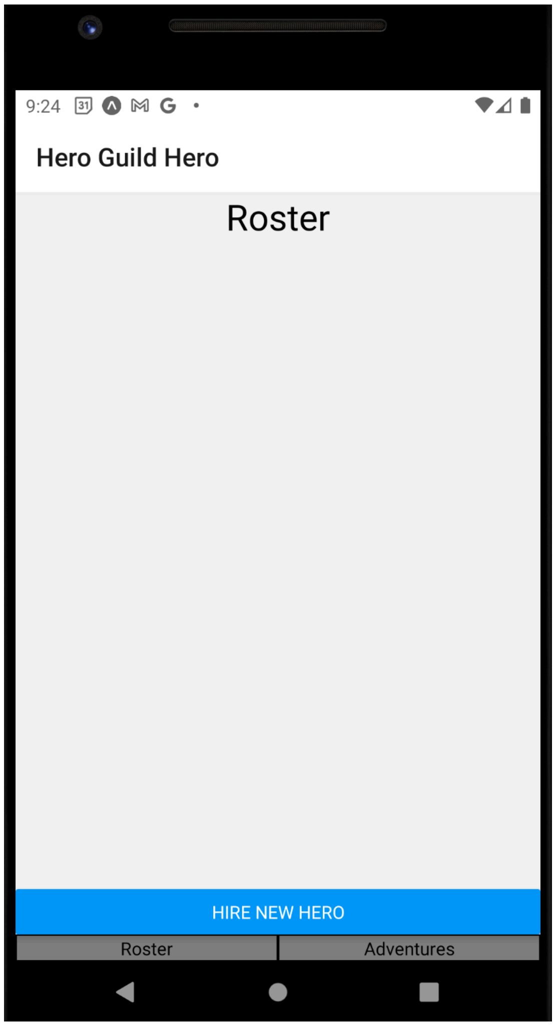
How much money they have. Can be spent to level up. Gained from going on adventures.

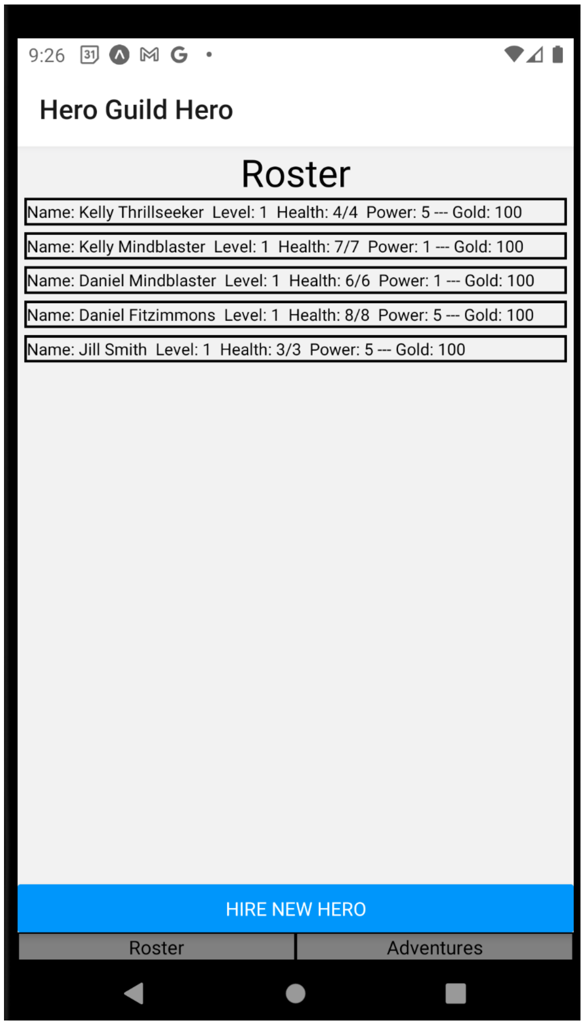
Although I encourage you to get creative with this assignment and make your own artistic choices, I'm looking for work that consists of at least three screens: The "**roster screen**" the "**hero detail screen**", and the "**adventure screen**".

Screen 1: The Roster Screen

This is a screen that primary serves two functions:

1. It displays the current roster of heroes that you have hired. Clicking on a hero takes you to a separate screen for more details about them.
2. It has a button to hire a hero.





```
Name: Kelly Thrillseeker  Level: 1  Health: 4/4  Power: 5 --- Gold: 100
Name: Kelly Mindblaster  Level: 1  Health: 7/7  Power: 1 --- Gold: 100
Name: Daniel Mindblaster  Level: 1  Health: 6/6  Power: 1 --- Gold: 100
Name: Daniel Fitzimmons  Level: 1  Health: 8/8  Power: 5 --- Gold: 100
Name: Jill Smith  Level: 1  Health: 3/3  Power: 5 --- Gold: 100
```

In the above images, on the left you'll see when you first load up the app – an empty roster. After clicking on the "Hire New Hero" button along the bottom, you should see the heroes start lining up.

To really break it down, here is the minimum that this screen should have:

1.) A title like "Roster" that lets the player know what screen it is.
2.) A flat list that gets all of the heroes from your Context Object and displays them.
3.) A button that, when pressed, generates a new hero. This should make use of a "Create" action function, which adds a new hero to the data stored in your Context object.
4.) A "Nav Bar" along the bottom. This should be a reusable component that has two Touchable Opacity components (which I'm going to call buttons since that's what

function they serve). One button takes you to the Roster screen (i.e., this one you are currently on). The other should take you to the Adventures screen (more on that in a moment).

5.) Clicking on a hero in the list should take you to the “Hero Detail” screen.

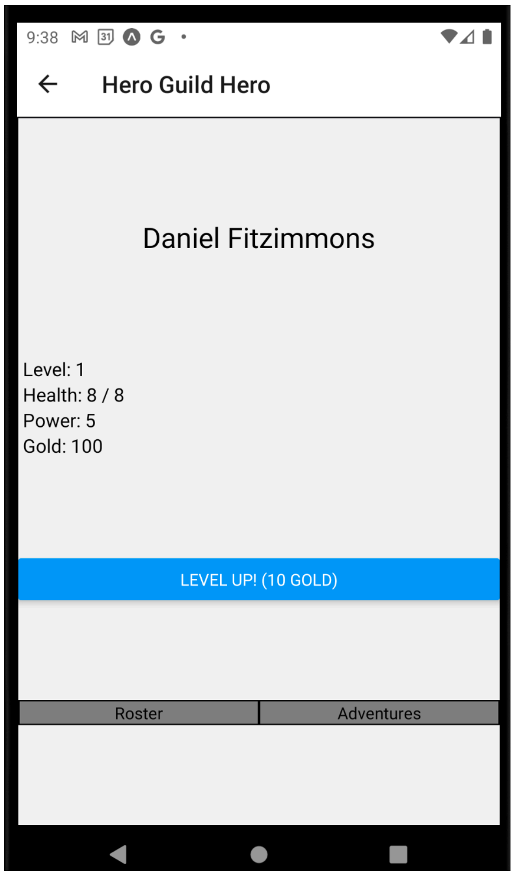
Screen 2: The Hero Detail Screen

Clicking on a hero from the roster screen should take you to the Hero Detail screen. This is a page that is dedicated solely to that hero, much like how when we made the blog post app, clicking on a blog post on the index screen let us see all of the relevant details about that post on the “show” screen. Just like how in the blog app we had to pass over the id of the post that we cared about, we’ll have to do the same thing here, too.

This screen also primarily serves two functions:

1.) Displays all information there is to display about the hero you selected.
2.) Allows you to level the hero up.

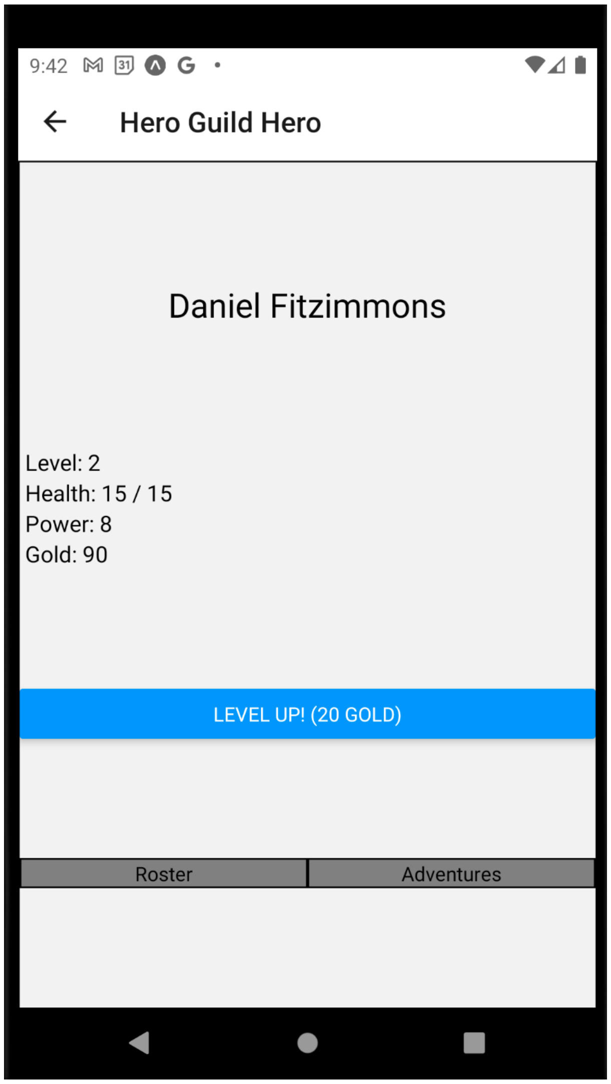
You can see my sample detail screen below:



Daniel Fitzimmons is Level 1, has eight health and five power, and holds one hundred gold pieces. For the low price of 10 gold, you could level Daniel here up to level two

You can make your heroes stronger by having them spend their hard won gold on levelling up. I simply made the cost of a level up equal to 10 * currentLevel. So to go from level one to level two, it would cost ten gold, but then to go from level two to level three, it would cost twenty. There should be *some* cost to pay for levelling up, but feel free to make it whatever you want if my simply formula isn't exciting enough for you.

Whatever you end up going with – this should ultimately call an "Update" action function. That is, it should actually update the context object with the new and improved version of this hero.



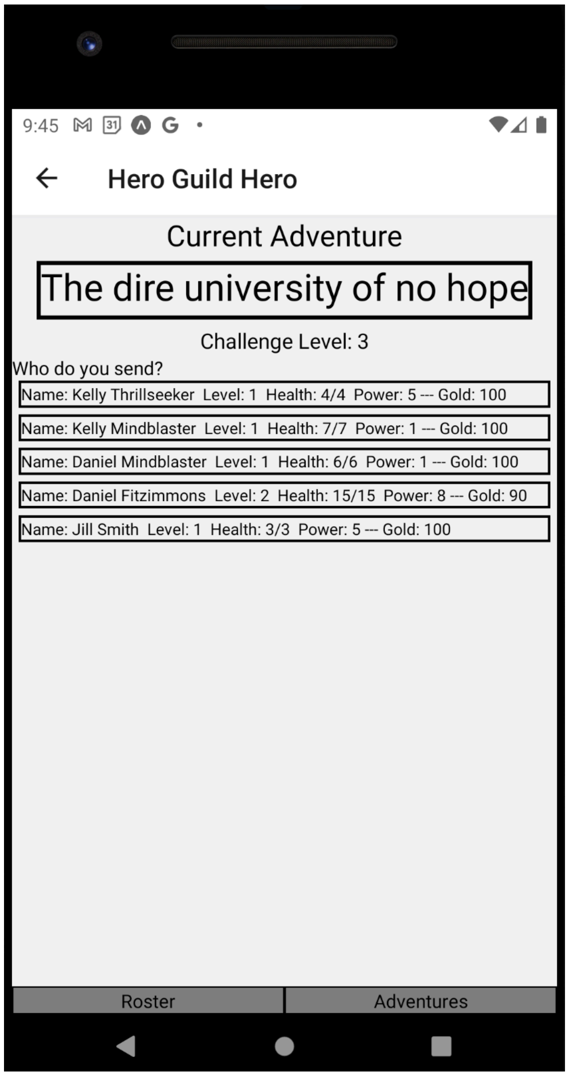
This screenshot was taken after I pushed the level up button. You can see that Daniel lost some gold (10 pieces to be precise), but advanced to level two, and gained some health and power. I have the health and power games basically just being completely random, but again, this is something you can play with!

Note that the NavBar component is on this screen too. Good thing you made it reusable!

Screen 3: The Adventure Screen

This screen is accessibly by clicking on the “Adventure” button on the nav bar in the lower right corner. When you go to this screen, you’ll see a few things.

1.) Along the top, information about the current adventure can be seen. In my game, an adventure only really has two pieces of information: a name and a challenge level. These can/should be randomly generated, so leaving the screen and coming back should yield a new adventure.
2.) Another flat list of the hero roster. Might be another good candidate for a reusable component, eh?



Current Adventure

The dire university of no hope

Challenge Level: 3

Who do you send?

Name: Kelly Thrillseeker Level: 1 Health: 4/4 Power: 5 --- Gold: 100
Name: Kelly Mindblaster Level: 1 Health: 7/7 Power: 1 --- Gold: 100
Name: Daniel Mindblaster Level: 1 Health: 6/6 Power: 1 --- Gold: 100
Name: Daniel Fitzimmons Level: 2 Health: 15/15 Power: 8 --- Gold: 90
Name: Jill Smith Level: 1 Health: 3/3 Power: 5 --- Gold: 100

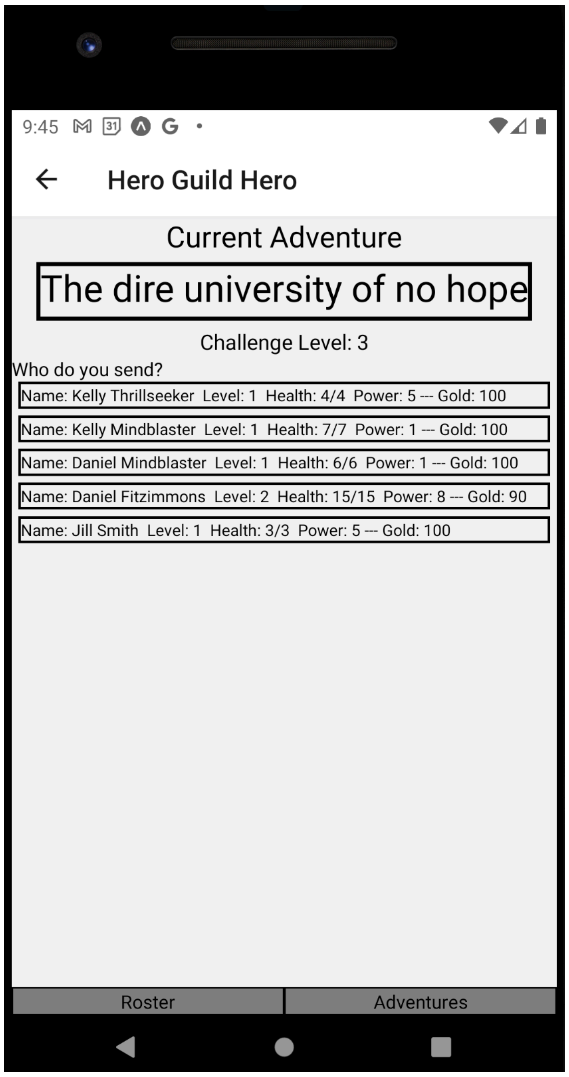
On this screen, clicking on a hero in the roster should *NOT* take you to the hero detail screen. At least not right away.

Instead, it first should send the hero you select ON the adventure. When you do that, a few things should happen.

1.) Figure out if the mission is a success or failure. You can get as complicated as you want for this.

    a. For my game, it is *VERY* simple. It simply compares the “power” stat of the selected hero to the “challenge level” of the generated adventure. If the hero’s power is greater, then it is a big success; they earn lots of gold and only lose a little health! Otherwise, they lose a lot of health and gain little gold.

2.) Figure out what the revised stats of the hero should be based on how the adventure went. Probably most of the stats should stay the same, but you can update currentHealth and gold based on how much gold you have them earn and how much damange they took.
    a. Again, my game has very simple calculations for this. They simply lose “a little” health and gain “a lot” of gold on a success, and the inverse on a failure.
3.) Once you’ve figured out the revised stats, THEN use the context object to use the “Update” action function to update the data of the hero in question (i.e., they lose health and gain gold).
4.) THEN navigate to the hero detail screen.
    a. When you do, pass along a “message” with feedback about how the adventure went. This message should include how much gold they gained and how much health they lost.
    b. Note this message should only display when they navigate to the hero detail screen after going on an adventure. There should be no message when going to the hero detail screen from the Roster screen.
        i. We’ve seen how we can use the ternary operator in JSX to only display components under certain circumstances.



9:45

Hero Guild Hero

Current Adventure

The dire university of no hope

Challenge Level: 3

Who do you send?

Name: Kelly Thrillseeker Level: 1 Health: 4/4 Power: 5 --- Gold: 100

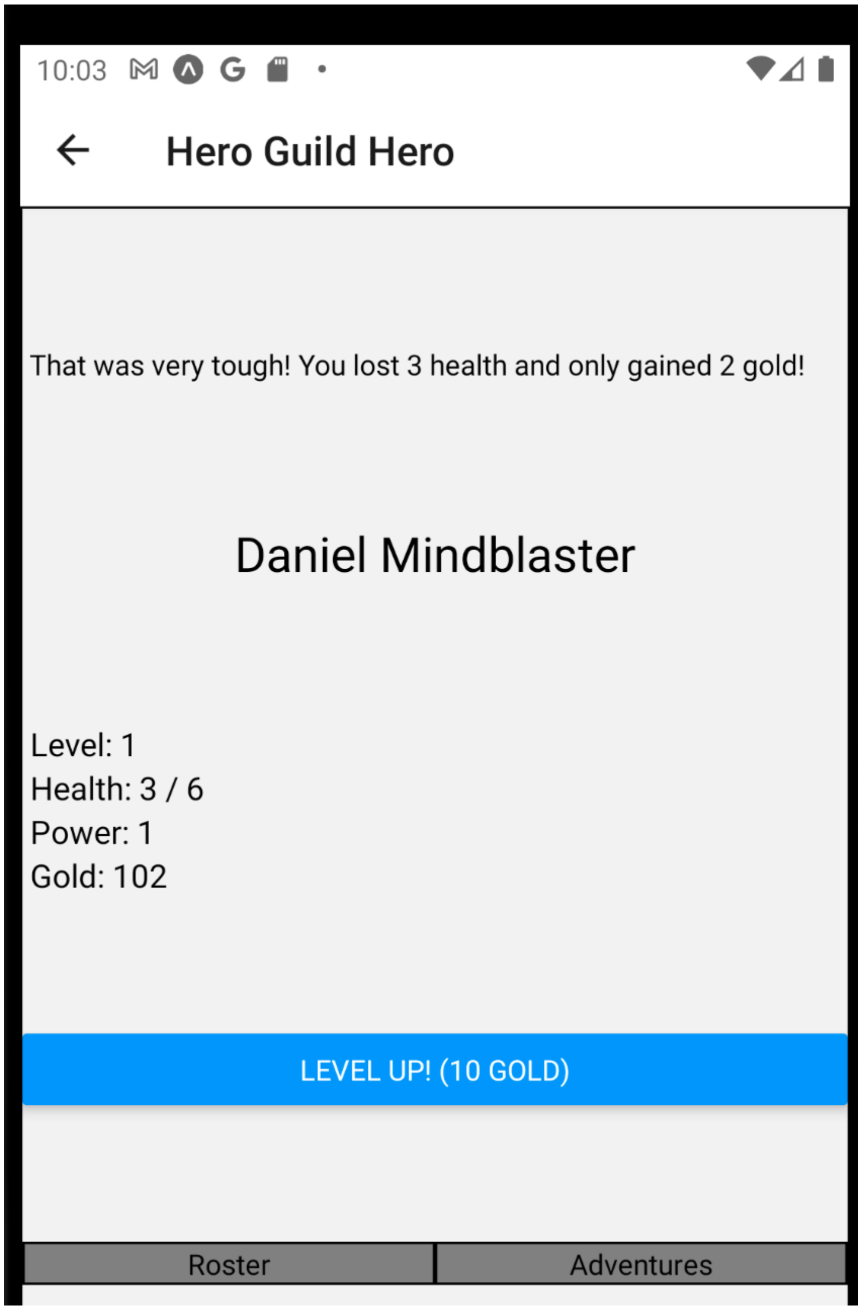
Name: Kelly Mindblaster Level: 1 Health: 7/7 Power: 1 --- Gold: 100

Name: Daniel Mindblaster Level: 1 Health: 6/6 Power: 1 --- Gold: 100

Name: Daniel Fitzimmons Level: 2 Health: 15/15 Power: 8 --- Gold: 90

Name: Jill Smith Level: 1 Health: 3/3 Power: 5 --- Gold: 100

I’m going to send “Daniel Mindblaster” to The dire university of no hope. I note that the challenge level of this adventure is 3, but Daniel here only has a power level of 1…


10:03

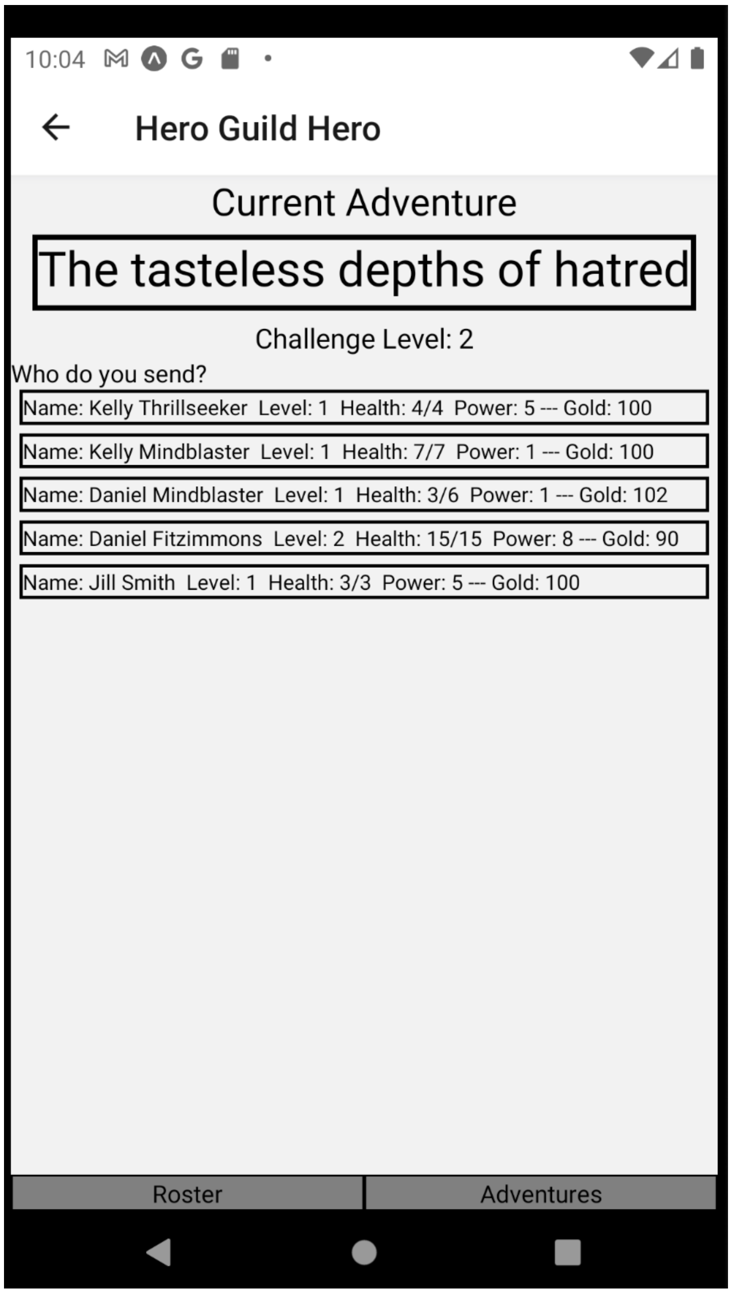
Hero Guild Hero

That was very tough! You lost 3 health and only gained 2 gold!

Daniel Mindblaster

Level: 1
Health: 3 / 6
Power: 1
Gold: 102

Ooph! Because Daniel’s power was less than the challenge rating, he “failed” the adventure. He lost half his life (three health is a lot when you’re a newbie Level 1 hero like Daniel!), and only gained a couple of gold pieces! Bum deal!


10:04

Hero Guild Hero

Current Adventure

The tasteless depths of hatred

Challenge Level: 2

Who do you send?

Name: Kelly Thrillseeker Level: 1 Health: 4/4 Power: 5 --- Gold: 100

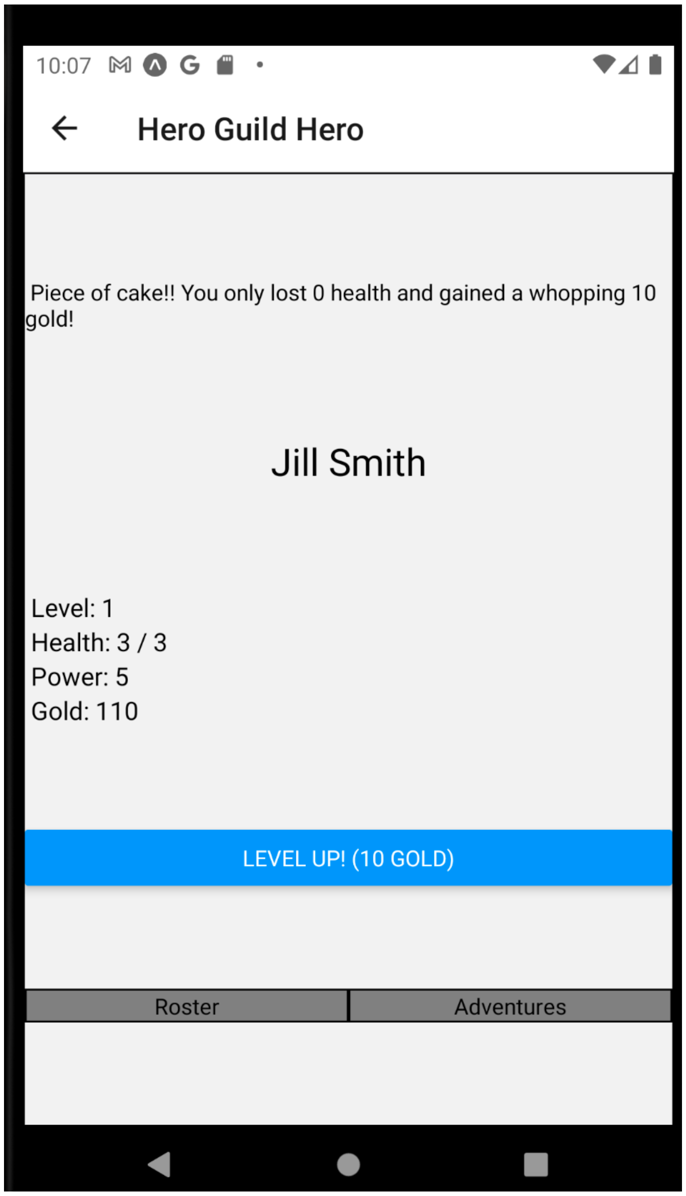
Name: Kelly Mindblaster Level: 1 Health: 7/7 Power: 1 --- Gold: 100

Name: Daniel Mindblaster Level: 1 Health: 3/6 Power: 1 --- Gold: 102

Name: Daniel Fitzimmons Level: 2 Health: 15/15 Power: 8 --- Gold: 90

Name: Jill Smith Level: 1 Health: 3/3 Power: 5 --- Gold: 100

If we navigate back to the adventure screen, we see we have a new adventure lined up. “The tasteless depths of hatred.” This one only has a challenge level of 2, so it isn’t so bad! Let’s send “Jill Smith” on this mission – she has a Power of 5 so she should be able to handle it!


10:07

Hero Guild Hero

Piece of cake!! You only lost 0 health and gained a whopping 10 gold!

Jill Smith

Level: 1
Health: 3 / 3
Power: 5
Gold: 110

Look at that, she did great! Didn't get a scratch on her, and earned enough gold for her first level up, right then and there!

And there you have it! For the "base version" of the game, that's all you need! But there's clearly lots of room for improvement…

Suggestions for Bells and Whistles (AKA Extra Credit)
Graduate Students are expected to include at least TWO of these:

***Visual Appeal**: I intentionally had this be as bare bones as possible, but clearly there is a lot of room for improvement! Have nice styling so that the buttons look good and the screens are artfully arranged! Have there be actual images / icons for the heroes and the adventures! Again, the bar was set really low here; please raise it a little! Please!

*__Better Feedback:__ Right now, sending the hero on an adventure gives the tiniest amount of feedback possible, kind of shoe-horned in to the hero detail screen. Make it so that the adventure feels more MEATY. Create a dedicated “Results.js” screen, and fill it up with both the WHAT and the WHY of how the adventure went down the way it did

*__Guild Coffers__: Right now each hero has their own supply of money, but there isn’t really a sense of YOU, the PLAYER having money. Make it so that when adventurers succeed, YOU get a little something on the side that you can spend in a variety of ways. Perhaps make it so that the guild as a whole can level up, and when It levels up it affects the quality / quantity of heroes you can hire. Make it so that hiring a hero costs YOU money, and if you run out of heroes and can’t afford to buy a new one, then you lose! Those are just a couple ideas to spend guild money, but the possibilities are truly endless!

*__A Second Context__: If you do “Guild Coffers” you really should do it in this way – have there be a SECOND Context object representing a different data resource (e.g., the guild).

***Can’t Spell CRUD Without a D…**: You may have noticed from the above description that at present, there’s nothing that lets you actually DELETE a hero (in my game, if a hero ever “died” they still stay in the roster). Find ways to incorporate a Delete action function! Maybe you can dismiss a hero and fire them. Maybe if a hero dies by their health falling below zero they can be removed!

*__A Brave Band__ – instead of sending SINGLE heroes on an adventure, instead have there be adventures that require MULTIPLE heroes. You select multiple characters before launching the adventure.

*__Complex Character Classes__ – right now the heroes are very simple. Feel free to spice them up! Give them more stats! Maybe give them classes like “wizard” or “fighter” and have certain adventures depend on there being heroes of particular classes to go on them.

*__Scaled Progression__ – Make it so that instead of challenge ratings on adventures being random, have them be somehow based on the make-up of your guild. That way as the heroes in your guild get stronger and stronger, they will always be guaranteed to have interesting challenges to overcome.

*__Something of your own design__: Let me know what you’re thinking! I’ll probably love it!

Submission:

Some folks last time had a hard time making a .zip file of your project (or at least one that was small enough to fit on Moodle). If you CAN make a .zip file of your entire project less than 100MB, please do so. If not, here are the most important things that you just to make sure get in there:

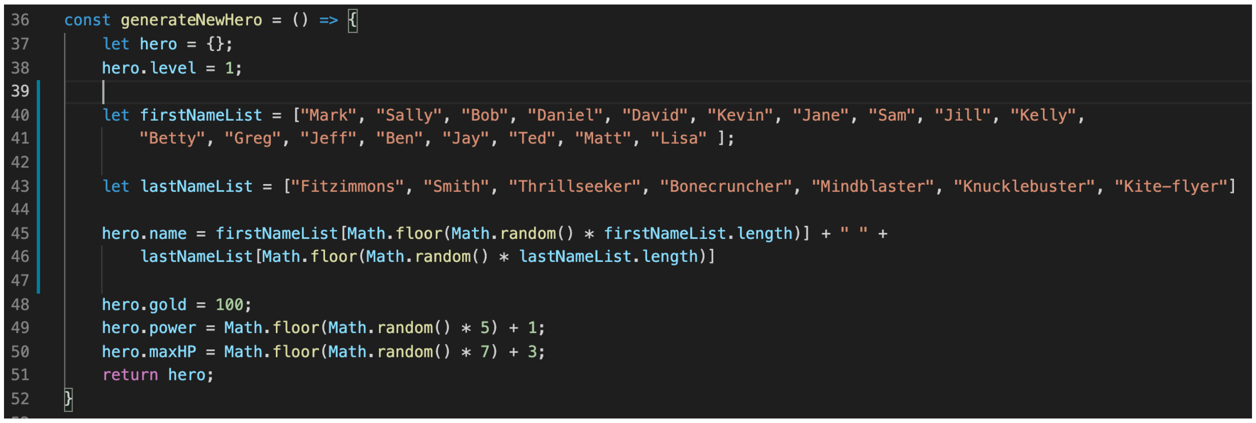
1.) All of your source code that you wrote (ideally it all lives in a src directory in your project).
2.) Any assets that you used (e.g., images) – ideally they all live in the assets directory
3.) Your App.js file
4.) Your package.js file
5.) A README file where you talk about what bells and whistles you went for, and anything else you need me to know (credits for artwork you used, bugs you know about, etc.).

Once you have your .zip file, please upload it to the appropriate place on Moodle.

Some Hints / Suggestions:

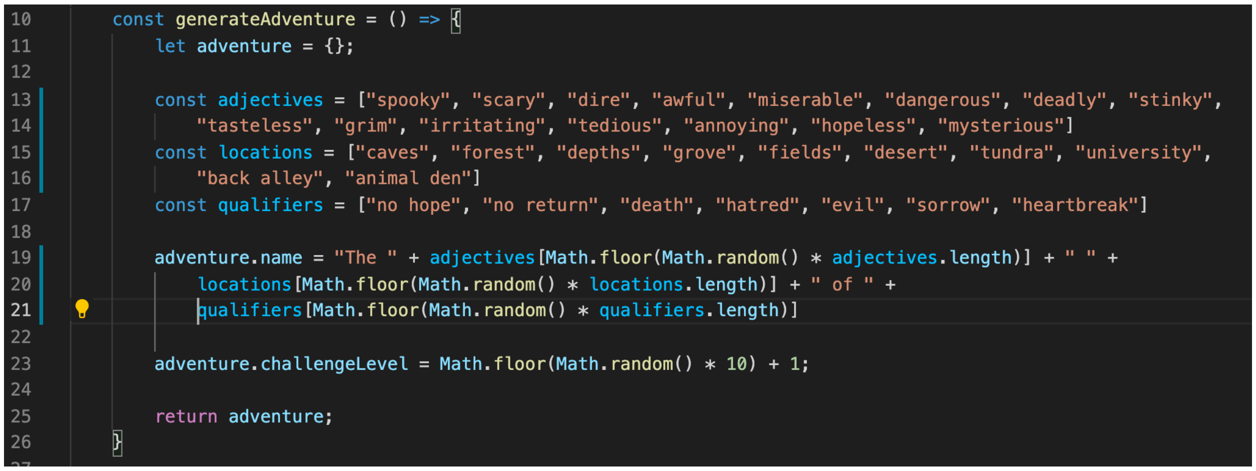
How did you make those awesome / hilarious character and adventure names!?!

Oh shucks. They aren't that fancy. But it's just a bit of procedural content generation. I basically just create a bunch of arrays and then pick random words from them to form names and sentences. Here, you can have them!



```
const generateNewHero = () => {
    let hero = {};
    hero.level = 1;

    let firstNameList = ["Mark", "Sally", "Bob", "Daniel", "David", "Kevin", "Jane", "Sam", "Jill", "Kelly",
        "Betty", "Greg", "Jeff", "Ben", "Jay", "Ted", "Matt", "Lisa" ];

    let lastNameList = ["Fitzimmons", "Smith", "Thrillseeker", "Bonecruncher", "Mindblaster", "Knucklebuster", "Kite-flyer"]

    hero.name = firstNameList[Math.floor(Math.random() * firstNameList.length)] + " " +
        lastNameList[Math.floor(Math.random() * lastNameList.length)]

    hero.gold = 100;
    hero.power = Math.floor(Math.random() * 5) + 1;
    hero.maxHP = Math.floor(Math.random() * 7) + 3;
    return hero;
}
```

Here is my hero generator.



```
const generateAdventure = () => {
    let adventure = {};

    const adjectives = ["spooky", "scary", "dire", "awful", "miserable", "dangerous", "deadly", "stinky",
        "tasteless", "grim", "irritating", "tedious", "annoying", "hopeless", "mysterious"]
    const locations = ["caves", "forest", "depths", "grove", "fields", "desert", "tundra", "university",
        "back alley", "animal den"]
    const qualifiers = ["no hope", "no return", "death", "hatred", "evil", "sorrow", "heartbreak"]

    adventure.name = "The " + adjectives[Math.floor(Math.random() * adjectives.length)] + " " +
        locations[Math.floor(Math.random() * locations.length)] + " of " +
        qualifiers[Math.floor(Math.random() * qualifiers.length)]

    adventure.challengeLevel = Math.floor(Math.random() * 10) + 1;

    return adventure;
}
```

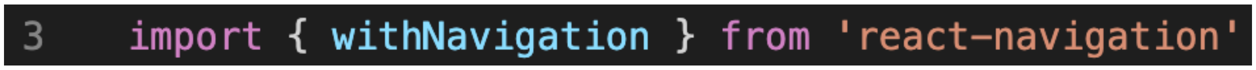
And there is my adventure generator.

Why do we only need a HeroContext – why don't we need one for adventures, too?

I mean, you are welcome to make one for them if you want! But right now, Heroes are the only things that appear on multiple screens, so it makes sense to use a Context for them. Also, Heroes are the only things that change with any permanence. If the adventures had data to them that you wanted to keep track of (e.g., number of times completed, number of monsters remaining, amount of gold left to be pillaged, etc.), then it would make a lot more sense to have a separate Context for them as well.

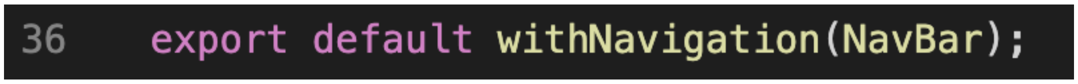
My reusable component needs to navigate, but it doesn't have access to the navigation.navigate function! What gives!

We saw this in class once too. The solution was to make use of the "withNavigation" hook. In your reusable component, you'll want to import withNavigation from the 'react-navigation' library.



```
import { withNavigation } from 'react-navigation'
```

And then use it when you export your component like so:



```
export default withNavigation(NavBar);
```

Do I need to make a brand new project? Can I just make it inside of rn-starter again?

Ah… I mean… I would *love* for you to make a brand new project. It's an important skill to have! But I'd rather you spend more time developing the actual app than fighting with creating the project. So I leave it to you. I will say that if you follow in the instructions on Lecture 16

("Advanced State Management Part 1"), from slides 10 – 16, I think it might go pretty smoothly for you! It went smoothly for me!

Good luck!! Have fun!