Start Time _____

End Time _____

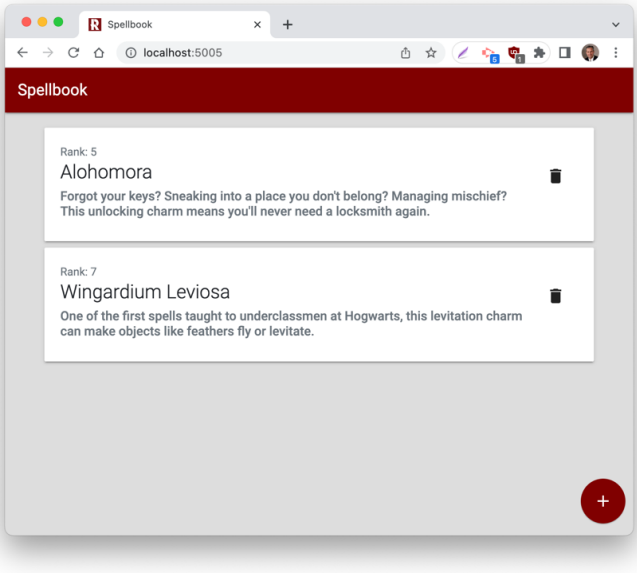
CSSE280 Summer 2022 Exam 2

You may use your computer, any prior notes or programs, the Moodle page, and the internet for general searching on web development. However, ***you must not communicate with anyone*** except your instructors and their assistants, if any. In particular:

- You must not talk with anyone else or exchange information with them during this exam.

	Points
Create	___ / 20
Read	___ / 50
Update	___ / 0
Delete	___ / 20
Other	___ / 10
Total	___ / 100

Continuing our Summer 2022 exam them, this exam is a **Harry Potter** Spellbook:



As you can see the Firestore object is a Spell. A Spell is somewhat similar to a MovieQuote and many of the concepts are similar. There are some differences in this exam versus the MovieQuotes No Auth web app, but let's look at similarities first:

- This exam has no auth, just like MovieQuotes **No Auth**
- A MovieQuote had a **movie** string, **quote** string, and a **lastTouched** date. The objects were in order by the **lastTouched** date (in reverse). A Spell has a **name** string, **description** string, and a **rank** number. The objects are in order by the **rank** number (notice a Spell doesn't have a date).
- This exam has a list page of all Spells. It uses a FAB + a dialog to create a Spell (see later images). All similar to the Movie Quotes No Auth web app.

A few differences:

- This Exam has no detail page. Clicking on a spell does nothing at all. There is only a list page.
- There is no way to edit a Spell. If you want to change a Spell, you must delete it and make a new one. This is done, just to make the exam shorter, but it is impractical (oh well).
- You can delete a Spell by clicking on the trash can icon (this is a new concept, to have a button on the card).
- Spells have a **rank** field. The rank field displays on the card and is used to order the Spells. A lower number is higher on the list. If two or more Spells have the same **rank**, the Spells with the same **rank** can be displayed in any order.

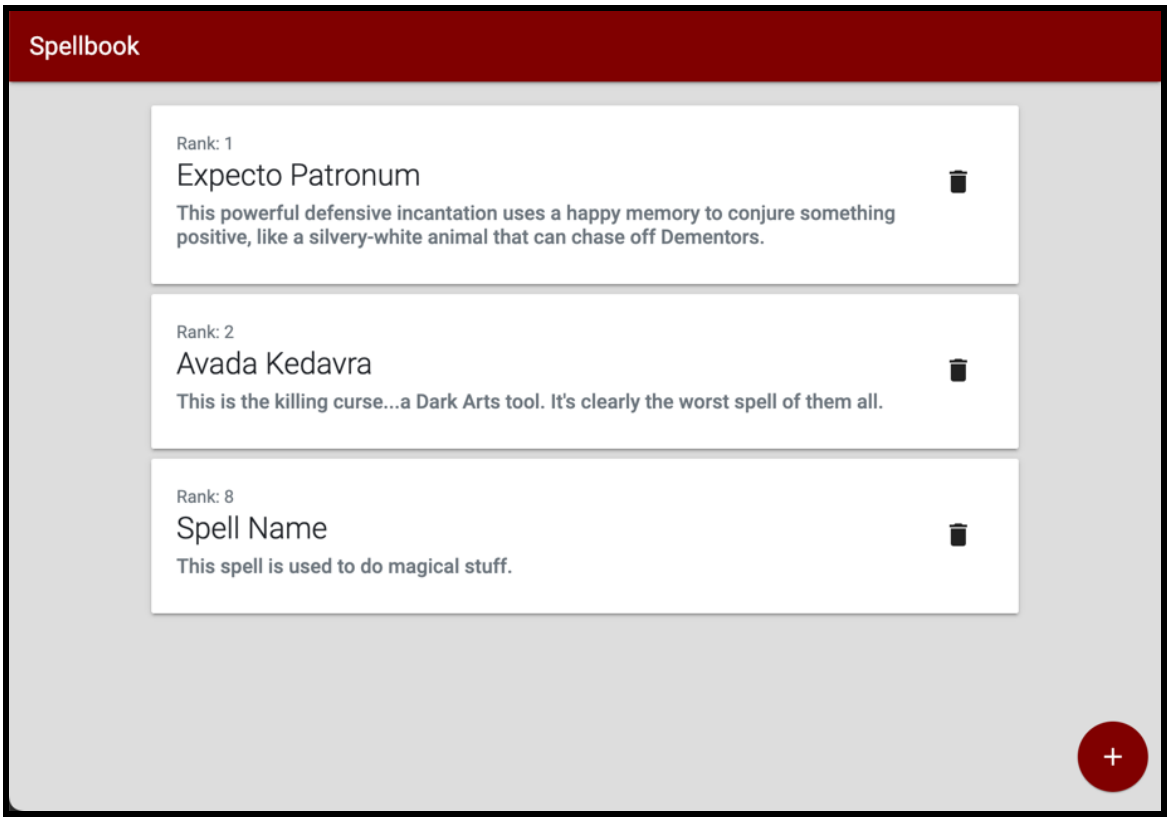
Those are the big picture items to help you understand the exam. There are images below with the details.

Given files

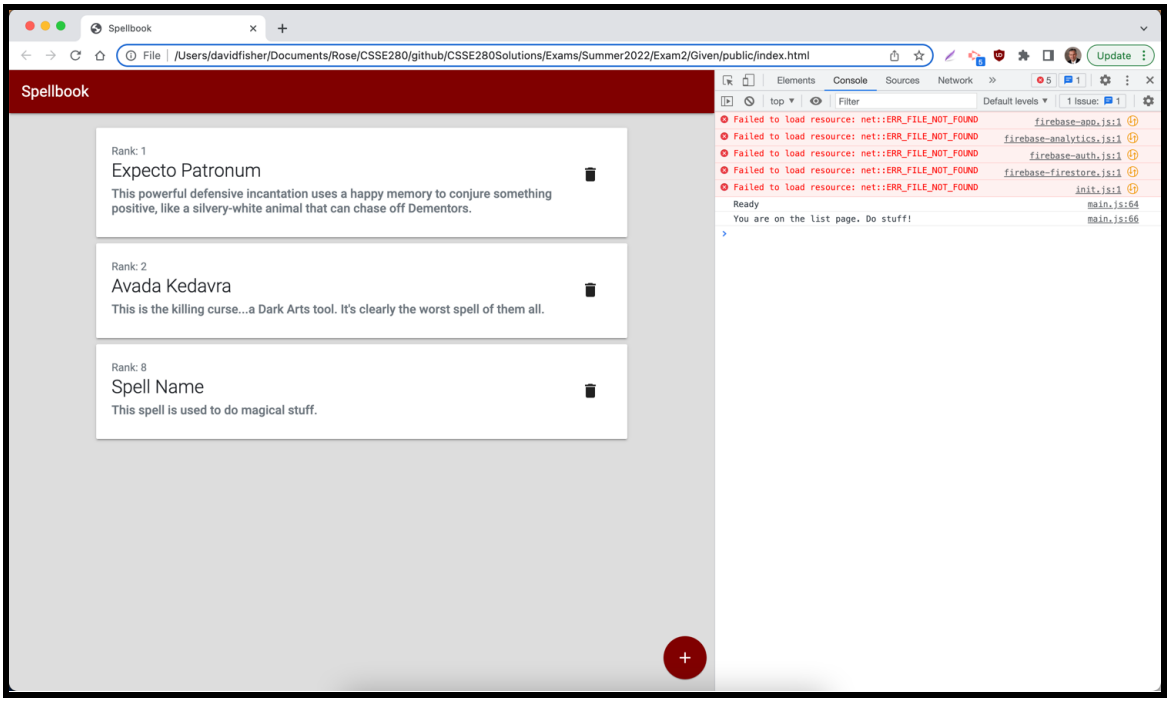
Instead of starting this project from the template. We will provide you with a folder to use as your starting point. You can download the starter code here:

Exam 2 Starter Code

You can open the index.html file to view it in Chrome. It should look like this:



If you open the Chrome dev tools, you'll notice the file has errors with the firebase files.

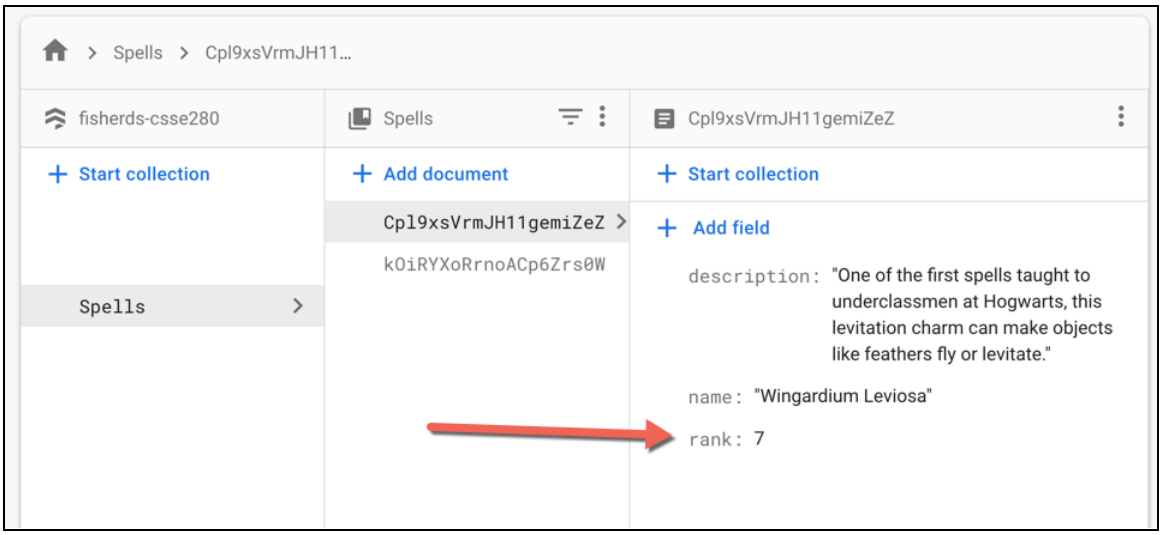


To fix that your first step is to do the `firebase init` process. You can make a new project in the Firebase Console if you want, but I recommend you just link it to any existing project you have, for example, username-moviequotes. You won't be deploying an exam anyway. Just make sure you use a unique collection name for this web app. I recommend the collection name: `Spells`

Once you have done firebase init, do a firebase serve and visit the localhost:#### url. Now it should load without errors. Try clicking the FAB to see the dialog that is given. Take a few minutes to review the index.html and main.js to see what you have been given. In short, you have examples of what the dynamic cards will look like (notice that they are all hardcoded HTML at the moment) and a few stubs in the main.js + names for Firebase keys. Otherwise, it's pretty empty.

Functionality

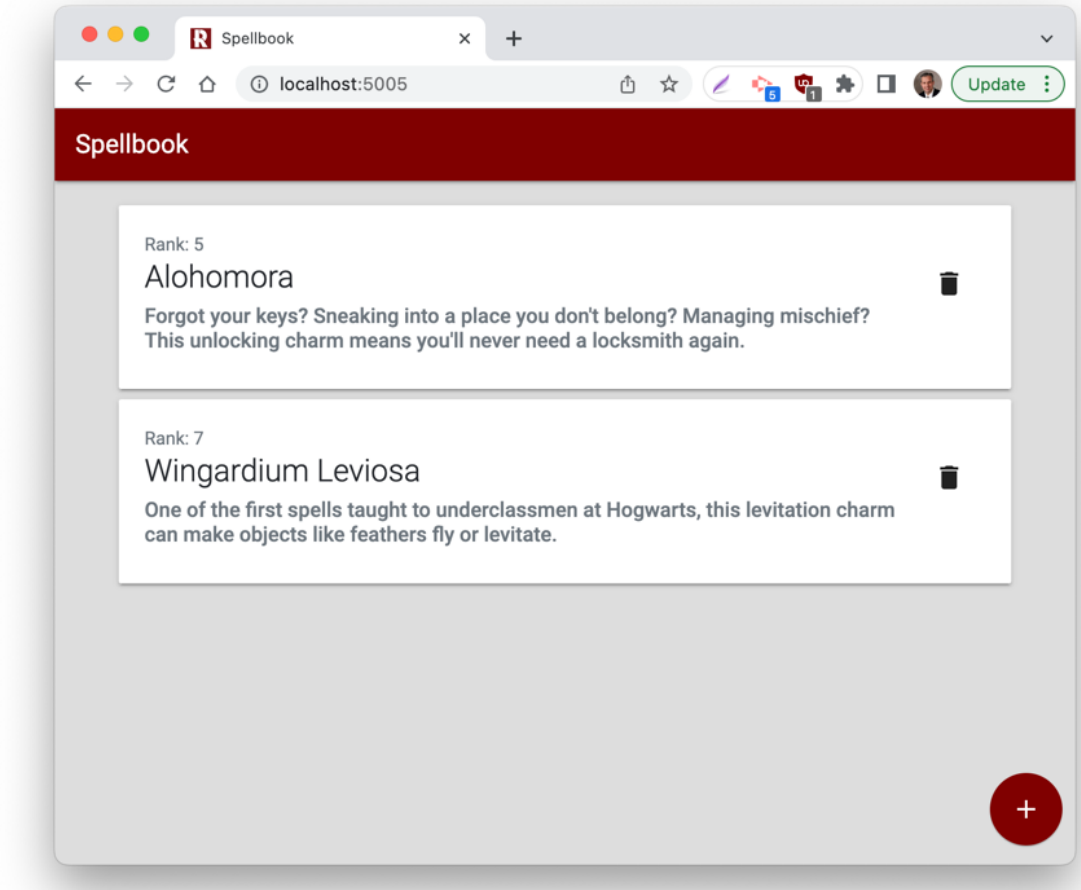
You can start the exam however you would like, but I choose to start by putting in fake date to my console and then worked on READ first. I used this website to add two spells: https://www.cnet.com/pictures/harry-potter-spells-ranked/9/ Here is what a Spell should look like in the console (note, capitalization, spelling etc all matter in the collection name and keys):



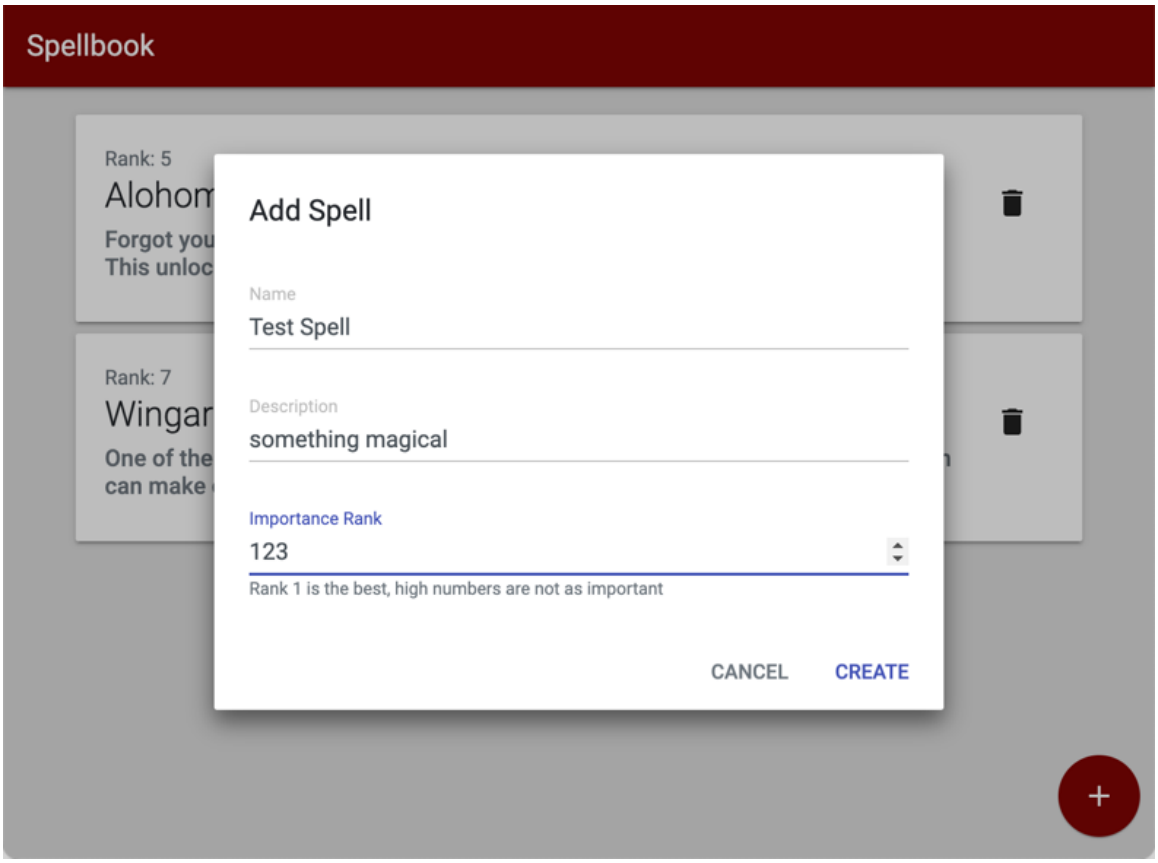
Note, that the rank is a NUMBER not a STRING (no double quotes on it). Make sure your Collection name (Spells), and keys (description, name, rank) match exactly because we'll test your code with our backend when grading your work.

Then I used the example card HTML, my knowledge from MovieQuotes about dynamically creating html and put together a read and update view. BTW make sure the code you submit for this exam does not include the words: movie, quote, or mq.

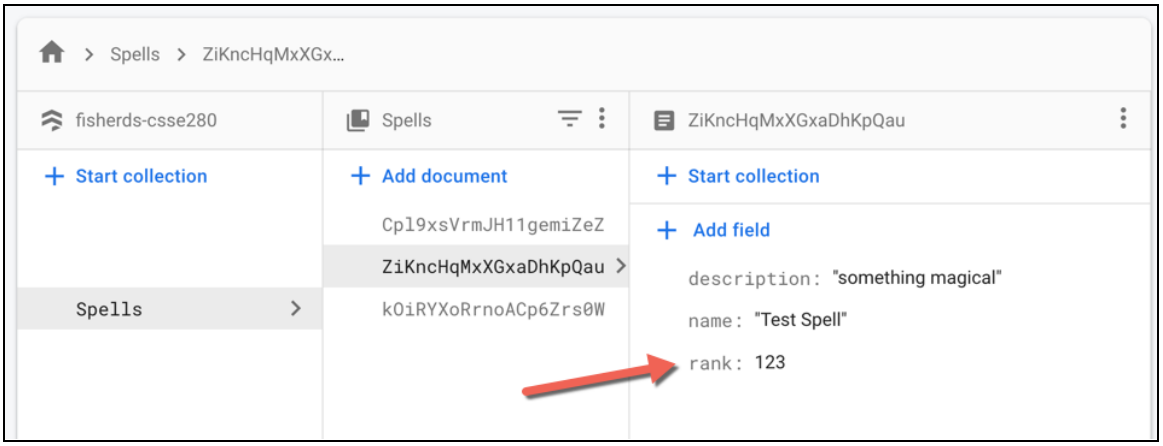
I was able to view those two quotes once I got the model object R and the controller's updateView methods done:



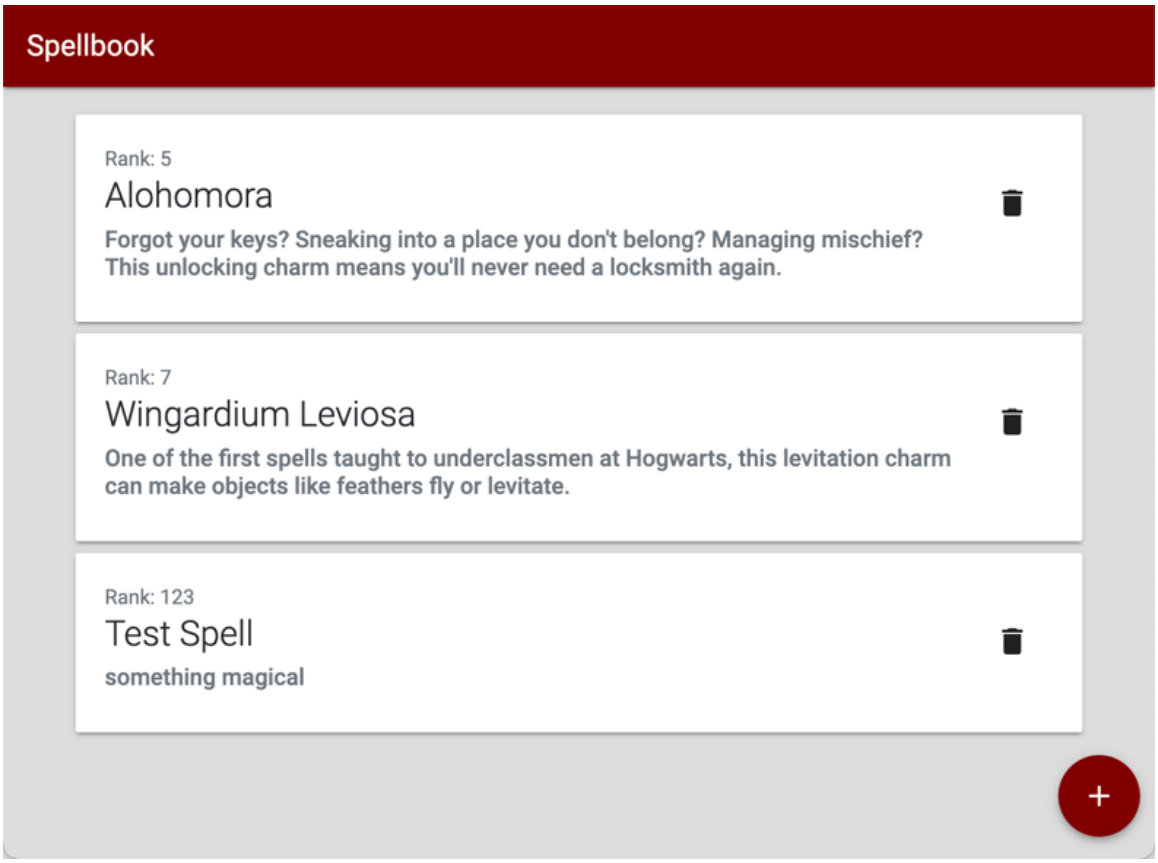
Next I did create. When you click the FAB you can type in values:



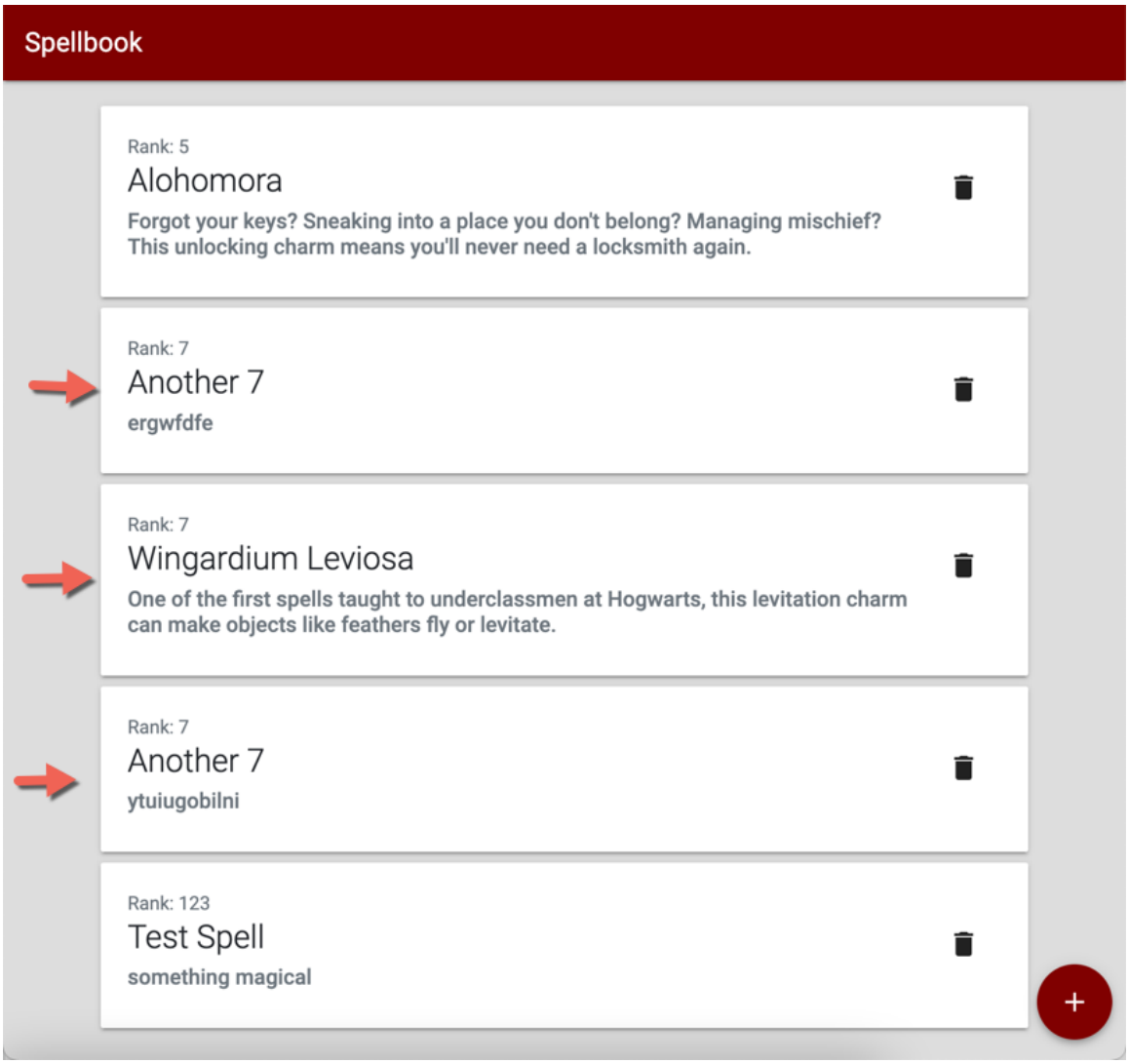
then click create. Make the CREATE button in the dialog get the input values and call the model object **add** method. A big warning, the Rank value needs to be a NUMBER, not a String in the Firestore (otherwise it'll sort alphabetically instead of numerically). The input value will be a string by default, so make sure you convert it to a number before you put it into the Firestore (parseInt works well). Check your Firestore to make sure it's a number (no quotes around the 123):



When you get things working, it will show your new data:

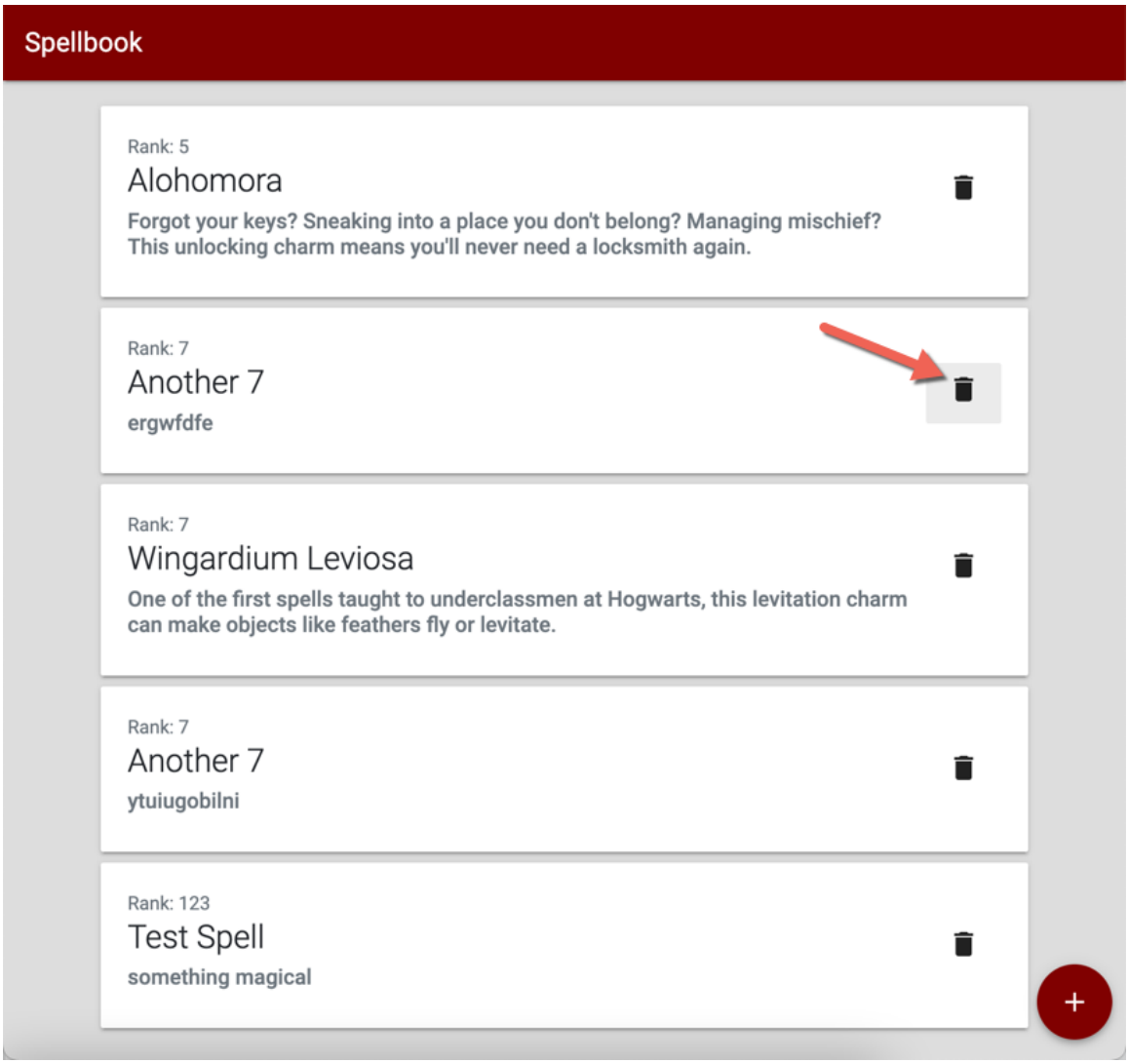


Add a few different ranks to be sure they are in numeric order. Also you can have duplicate ranks and that's ok:

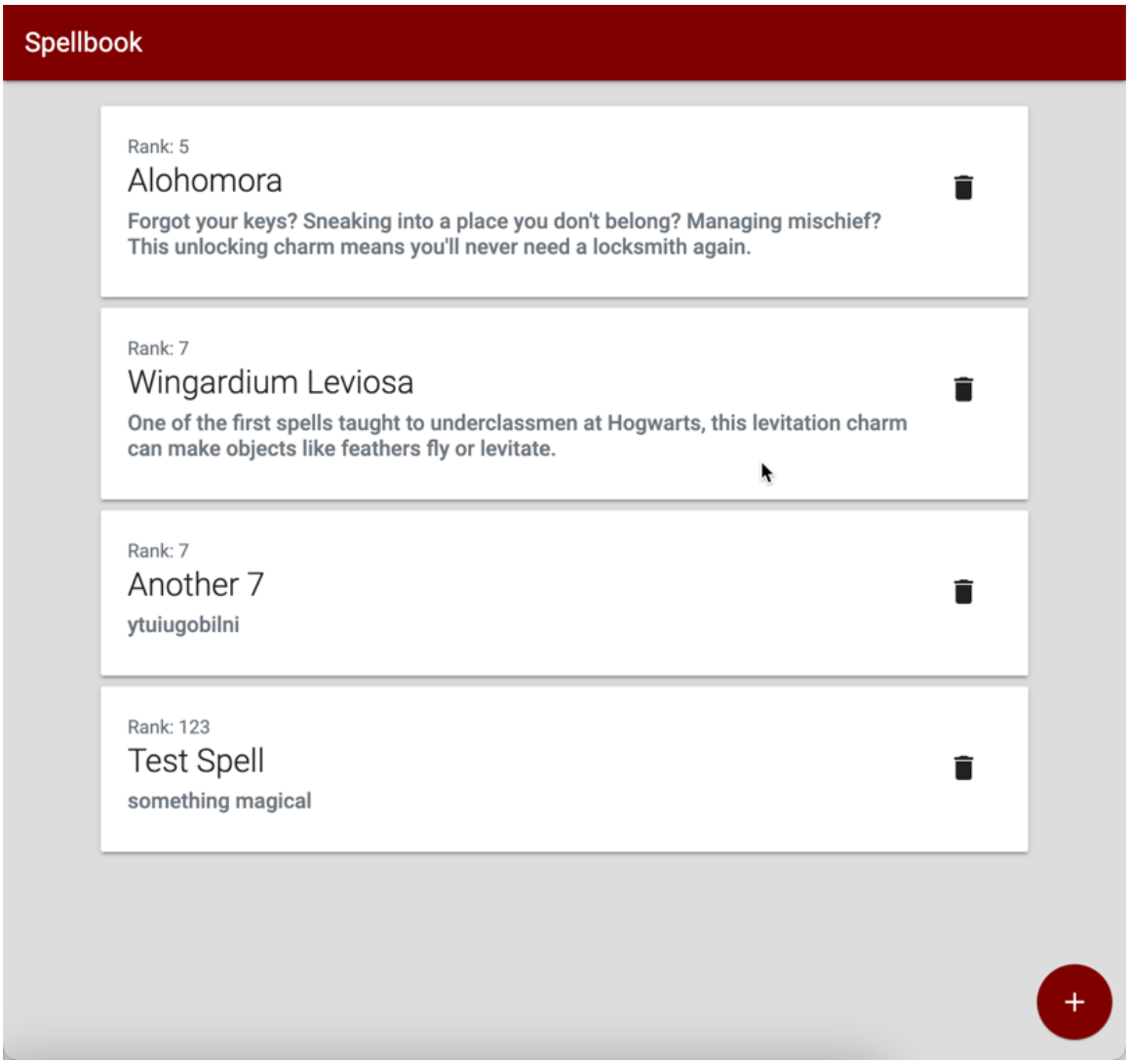


Delete

Next (finally), I'd implement Delete. Clicking on the card must do nothing (there is no detail view). So clicking on text has no effect. However, if you click on the trash can icon on a card…



it will delete **that** spell.



Here is a hint from my code about making a click target within the card instead of being on the whole card:

```
const newCard = this._createCard(spell);
newCard.querySelector("button").onclick = (event) => {
    console.log("You clicked delete");
    rhit.fbSpellsCollectionManager.delete(spell.id);
};
```

If you understand that code you can do something similar. I think it's cool to use a closure on the spell.id so you know which one to delete. IIf you solve the problem in a different way that's fine too. Mainly I just wanted to show you how to do .**querySelector** on something that wan't the whole document. You'll have to figure out how to do delete in the model object class.

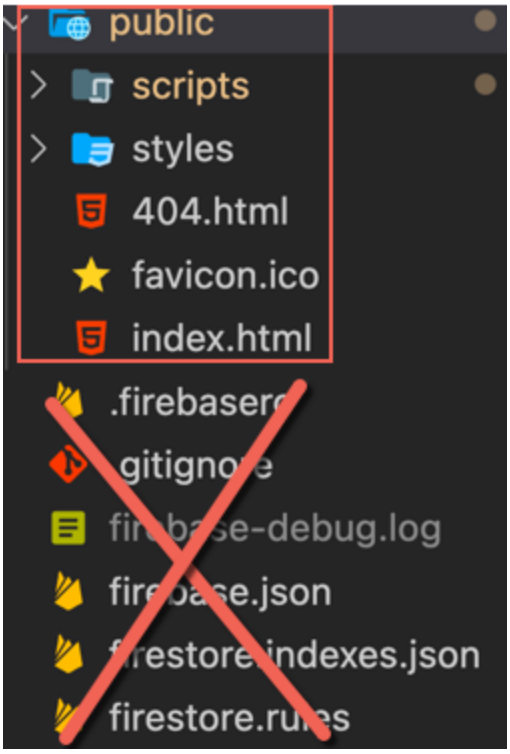
That's it. Once you get it all working or run out of time you will submit it. Details below.

Firebase

You can firebase init and firebase serve, but you'll **NEVER** firebase deploy this exam. That would make your exam code public.

Submitting your work

You will submit a .zip of your project folder to Moodle. Note: we don't want or need your firebase files.



We'll do a `firebase init` on a project when we run your code. That way we can see the firebase console and use our own test data (that is why your Collection name and key names had to match our plan).

You will be answering some questions on Gradescope as well.