Cord Cutting Calculator - Spring 2021 Final Report

Team Name: Coughing Cats

Team roles:

Scrum Master: Chris Kornosky

Product Owner: Shreyas Rameshkumar

Team Members: Arbin Bhuiyan, Elizabeth Hokaj, Manav Gurumoorthy, Srishti Madan, Tanay Kulkarni

Customer meeting date/time/place:

Wednesdays 4:15 pm, Zoom

Links:

GitHub Repo: https://github.com/Sugarcandy16/Cord-Cutting-Calculator/

Pivotal Tracker: https://www.pivotaltracker.com/n/projects/2495463

App (Heroku) Link: https://cord-cutting-calculator-sp2021.herokuapp.com/

Demo Video:

https://youtu.be/oom5kA98EF4

Poster:
https://github.com/Sugarcandy16/Cord-Cutting-Calculator/blob/master/documentation/Spring2021/Cord%20Cutting%20Calculator%202021%20-%20Poster.pptx

Summary

There exist many streaming services which provide the ability to stream live tv channels. This has given rise to an age where viewers have multiple options from which to consume entertainment (e.g. FuboTV, Sling, YouTube TV, Hulu, Philo).

Each service provider bundles channels differently from each other and the channels offered by one are not necessarily offered by others. Thus, there is not one best service provider for all. It differs for each individual user based on the content they want to consume and the available budget. The individual's preferences change over time which might result in reconsidering their current service provider. This app recommends users the best streaming packages based on their list of provided channels which fall under the categories of: "must have", "would like to have", and "OK to have".

There are two distinct interfaces: Administrator and User. Both of which require sign up and logging in. On the admin interface, the administrator can add and remove database content, namely channels, streaming packages, and other information related to each package. In the user interface, the user can put in his preferences and existing devices through the user profile, enter a list of channels (by priority), and include a budget, then our app will suggest recommendations for streaming package subscriptions.

User Stories

Added a reset button to all selection views:

As a channel watcher…

So that I don't need to click selected channels again to remove them from my list.

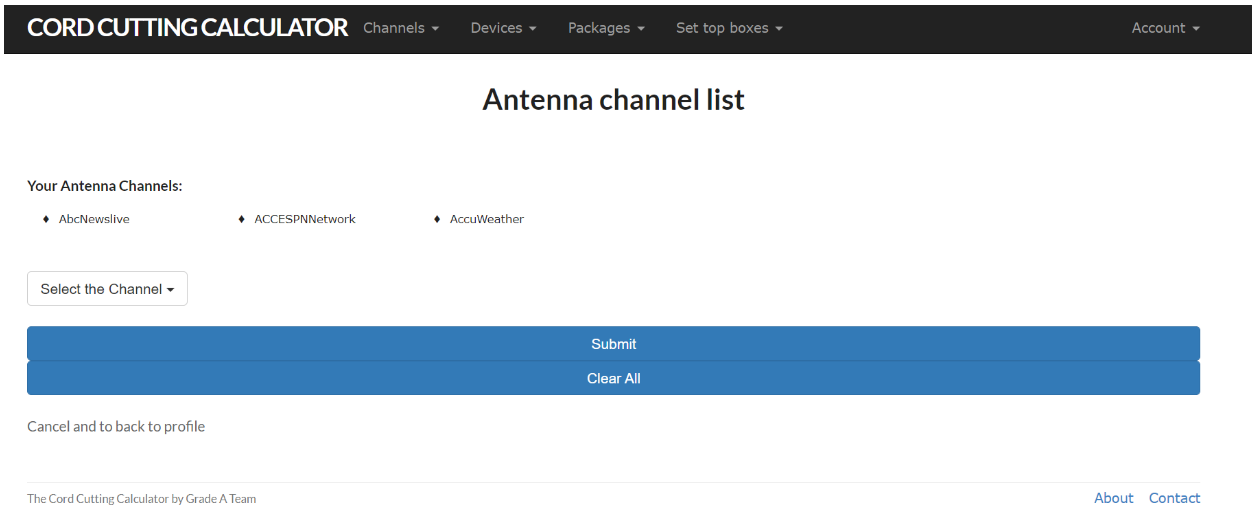
I want to add a button to the "Select the Channels" dropdown.

After much thought, we ultimately decided to break this user story down into several, more manageable user stories with a smaller scope….

Sub-story 1: Add "Reset" button to User's "Edit Antenna Channels" page

This feature is complete and has been deployed to the Heroku app. This was also a scratch feature which we added to the app. In iteration 1, when we added the "Reset" button and selected it the user was redirected to the home page; however after some changes, the reset button now keeps the user on the same page and resets any previously selected options successfully. To avoid any further confusion, we also re-labeled the button to read "Clear All" in iteration 2.

New "Clear All" button:



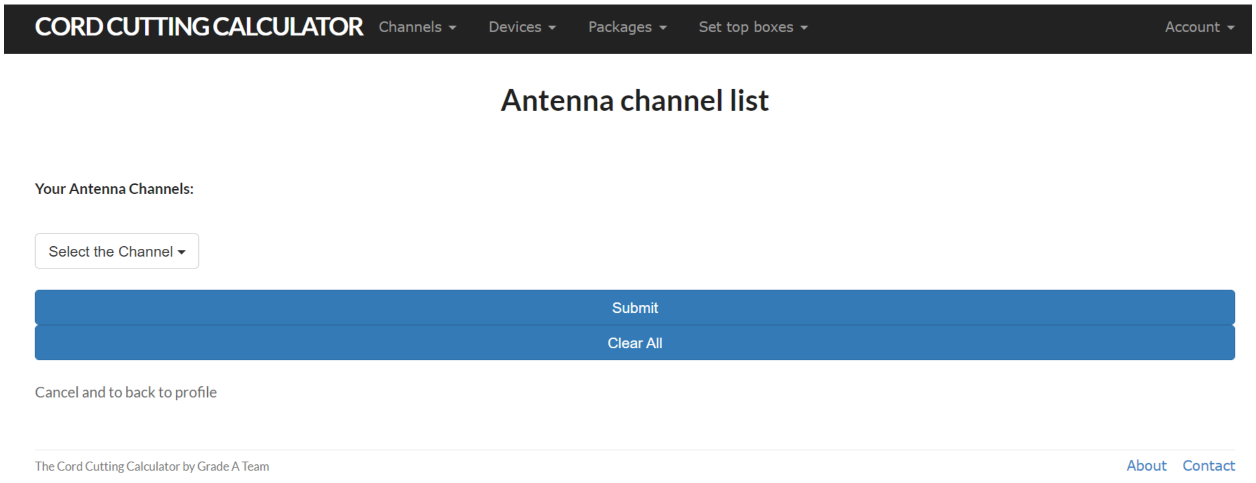
Antenna channel list

Your Antenna Channels:

- AbcNewslive
- ACCESPNNetwork
- AccuWeather

The Cord Cutting Calculator by Grade A Team

Resulting page when selecting "Clear All":



Antenna channel list

Your Antenna Channels:

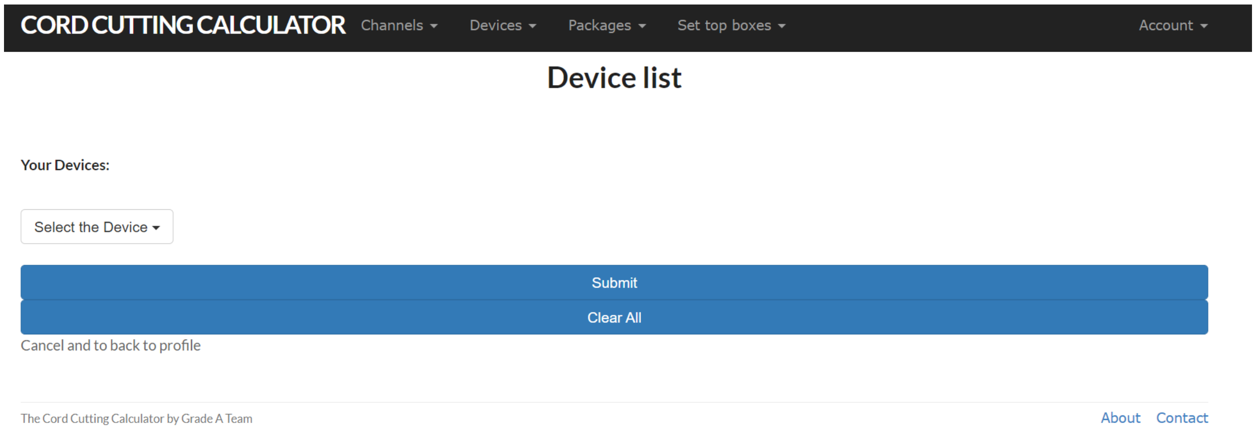
Cancel and to back to profile

The Cord Cutting Calculator by Grade A Team

Sub-story 2: Add "Reset" button to the User's "Edit Devices Page"

This feature is also complete and, like the previous sub-story, we also relabeled this button to read "Clear All" and the same requirement of refreshing the current page was also met.

Deployed Feature:



Device list

Your Devices:

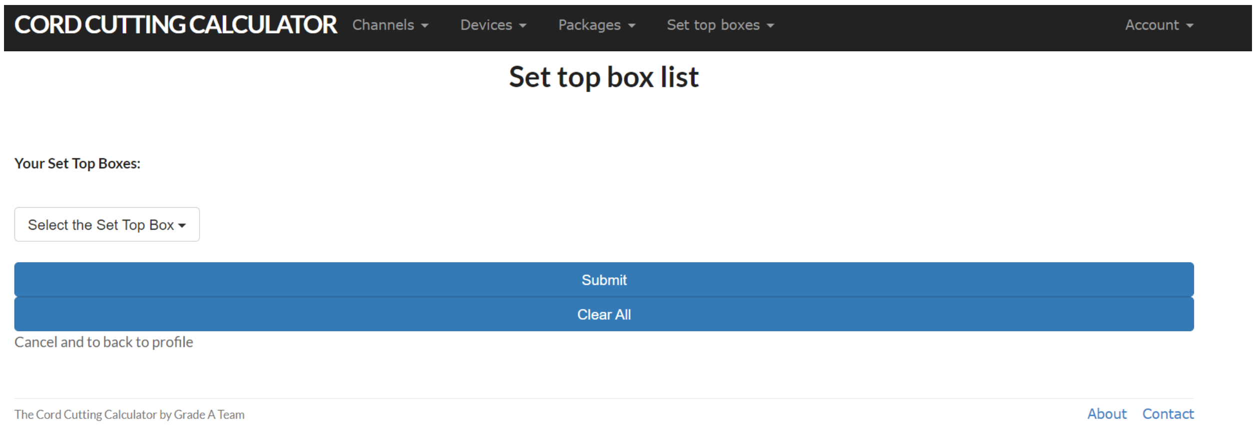
Cancel and to back to profile

The Cord Cutting Calculator by Grade A Team

Sub-story 3: Add “Reset” button to User’s “Edit Set Top Box” page

This feature has been deployed to Heroku and the same changes as the past two sub-stories were implemented.

Deployed Feature:



Set top box list

Your Set Top Boxes:

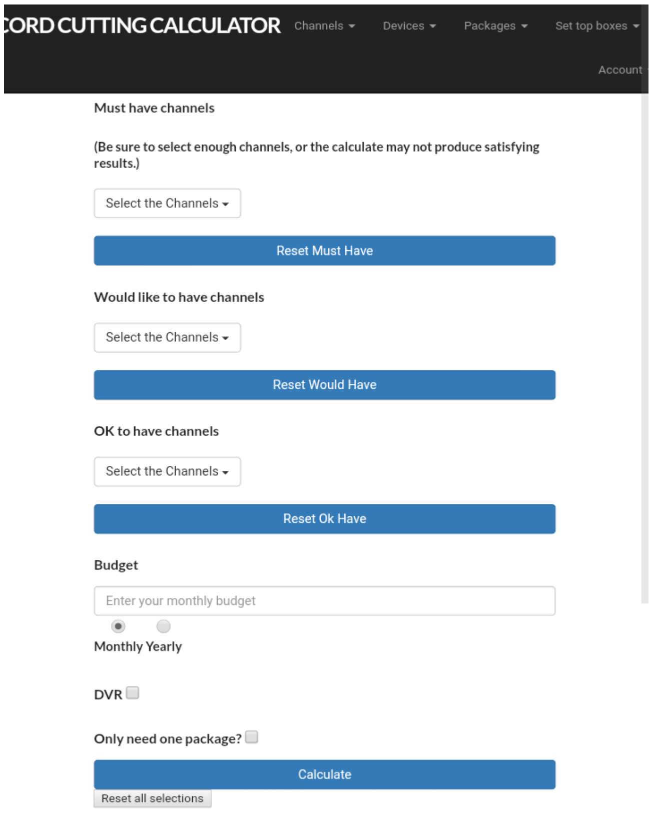
Cancel and to back to profile

The Cord Cutting Calculator by Grade A Team

Sub-story 4: Add “Reset” button to Calculator page

For this implementation, we actually added four reset buttons, three of which are used to clear out specific sections rather than the whole page. There is also a “Reset All Selections!” button if the user wants to refresh the entire page.

Deployed Feature:



Must have channels

(Be sure to select enough channels, or the calculate may not produce satisfying results.)

Select the Channels

Reset Must Have

Would like to have channels

Select the Channels

Reset Would Have

OK to have channels

Select the Channels

Reset Ok Have

Budget

Enter your monthly budget

[✓ Monthly] Yearly

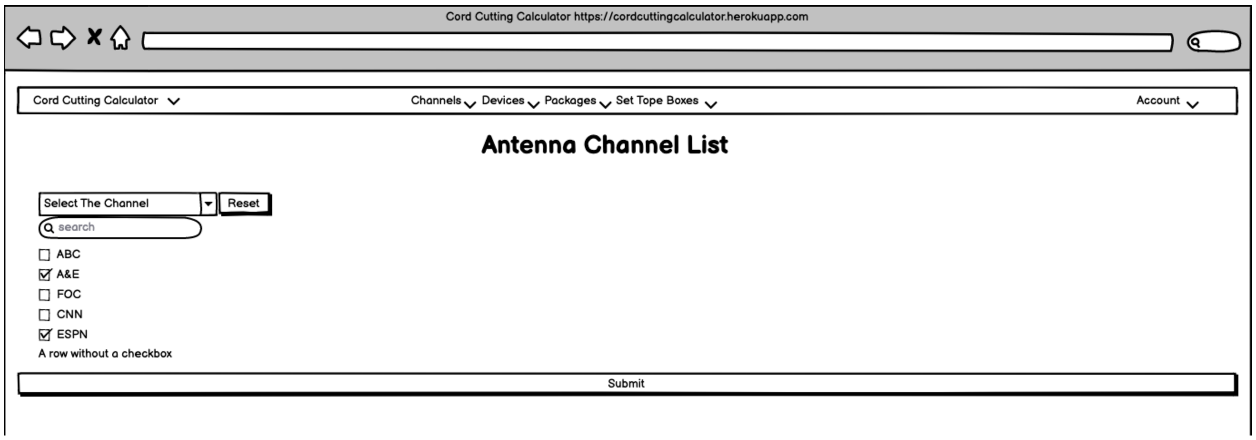
DVR ☐

Only need one package? ☐

Calculate

Reset all selections

Lo-Fi UI Mockup:


Cord Cutting Calculator https://cordcuttingcalculator.herokuapp.com
Cord Cutting Calculator
Channels
Devices
Packages
Set Tope Boxes
Account

Antenna Channel List

Select The Channel | Reset

search

☐ ABC
☑ A&E
☐ FOC
☐ CNN
☑ ESPN

A row without a checkbox

Submit

User Profile UI Improvements:

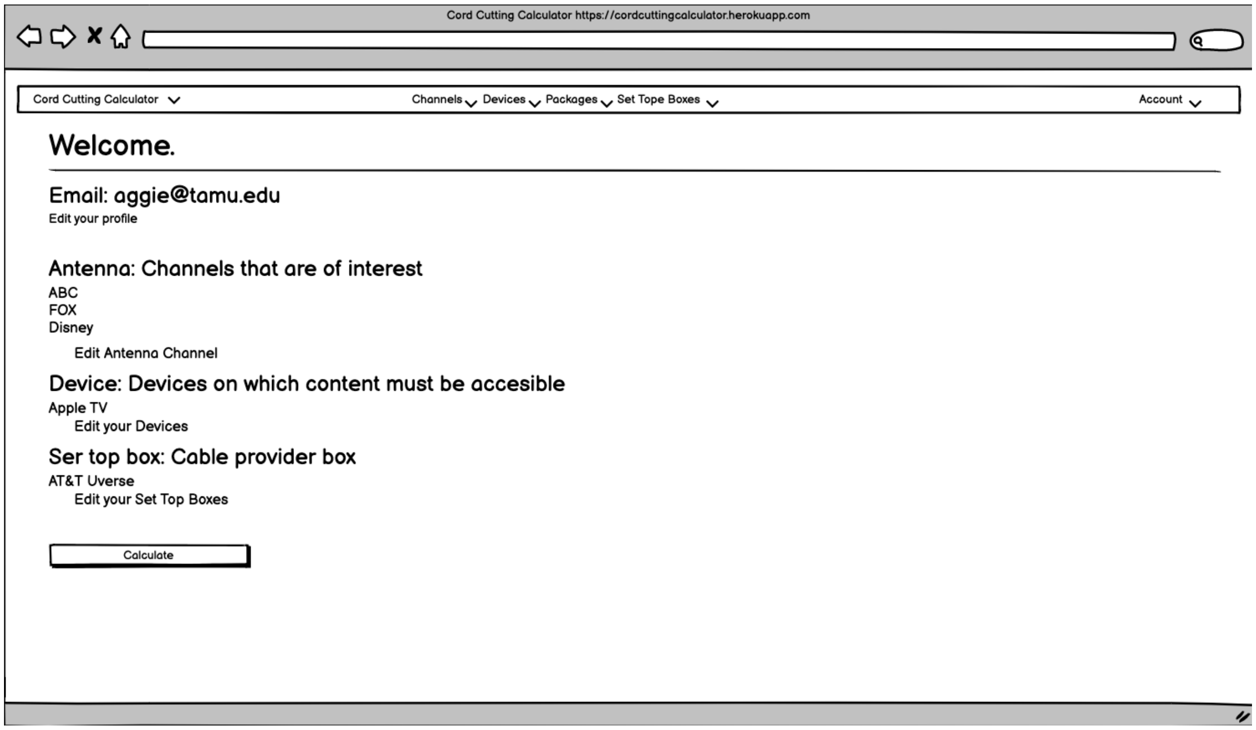
As a new user...

Current descriptions are sparse and confusing.

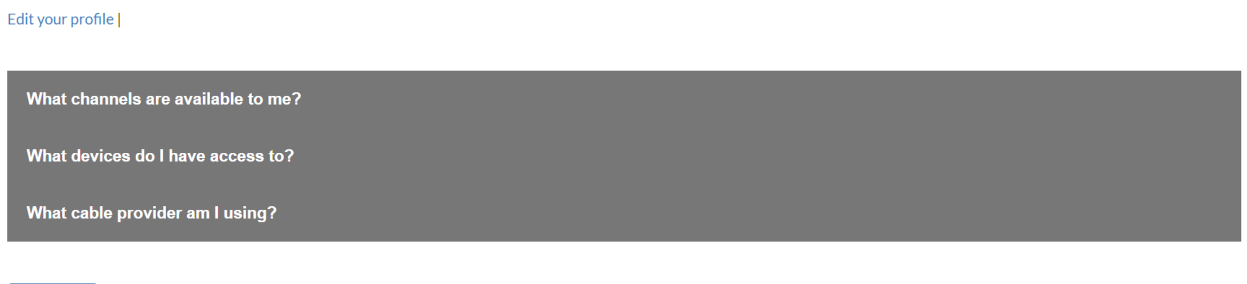
Add more descriptive labels.

As a team, we found the pre-existing labels a bit vague and difficult to understand, so we implemented a simple fix by rewording the options. Since it was such a quick fix, the feature has been fully implemented and there was no need for Cucumber or RSpec testing.

Low-Fi UI mockup :



Deployed Feature:

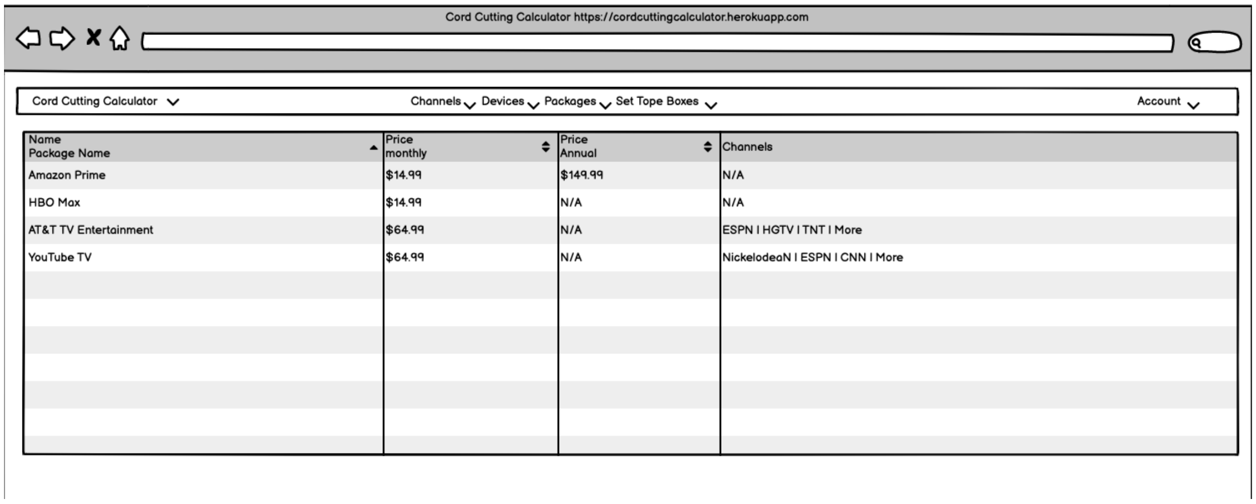

Edit your profile |
What channels are available to me?
What devices do I have access to?
What cable provider am I using?

Improved selection drop downs

> As a channel watcher and admin…
>
> Current experience to view different items is tedious and unfriendly.
>
> Change how items are listed and interacted with.

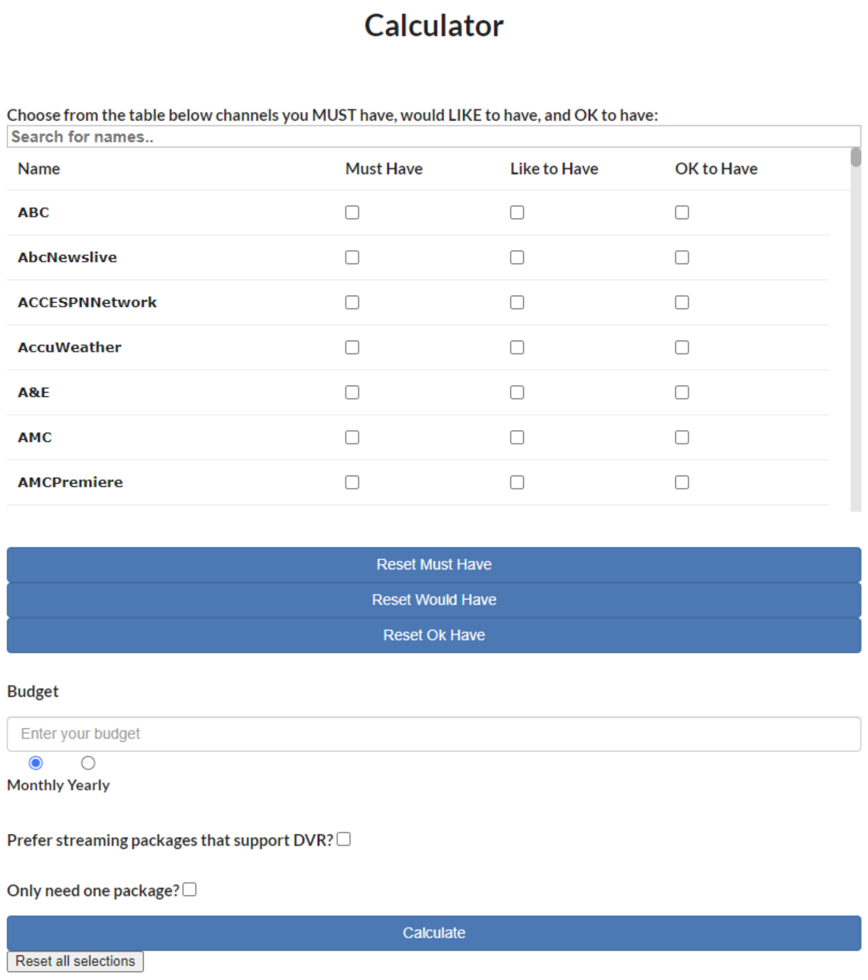
The team decided that this feature was necessary, but a huge undertaking. The current app uses long, tedious, unfriendly dropdown menus that forces the user to comb through all options alphabetically unless they can type in what they're searching for. Our approach is to use a table, similar to one featured on Rotten Potatoes website. That way users won't have a visual disconnect when selecting their "must have", "would like to have", and "OK to have".

Low-Fi UI Mockup:


Cord Cutting Calculator https://cordcuttingcalculator.herokuapp.com
Cord Cutting Calculator
Channels
Devices
Packages
Set Tope Boxes
Account

Name Package Name	Price monthly	Price Annual	Channels
Amazon Prime	$14.99	$149.99	N/A
HBO Max	$14.99	N/A	N/A
AT&T TV Entertainment	$64.99	N/A	ESPN I HGTV I TNT I More
YouTube TV	$64.99	N/A	NickelodeaN I ESPN I CNN I More

Deployed Feature:



Calculator

Choose from the table below channels you MUST have, would LIKE to have, and OK to have:

Search for names..

Name	Must Have	Like to Have	OK to Have
ABC	☐	☐	☐
AbcNewslive	☐	☐	☐
ACCESPNNetwork	☐	☐	☐
AccuWeather	☐	☐	☐
A&E	☐	☐	☐
AMC	☐	☐	☐
AMCPremiere	☐	☐	☐

Reset Must Have

Reset Would Have

Reset Ok Have

Budget

Enter your budget

● ○

Monthly Yearly

Prefer streaming packages that support DVR? ☐

Only need one package? ☐

Calculate

Reset all selections

Allowed for specification of monthly vs. yearly budget

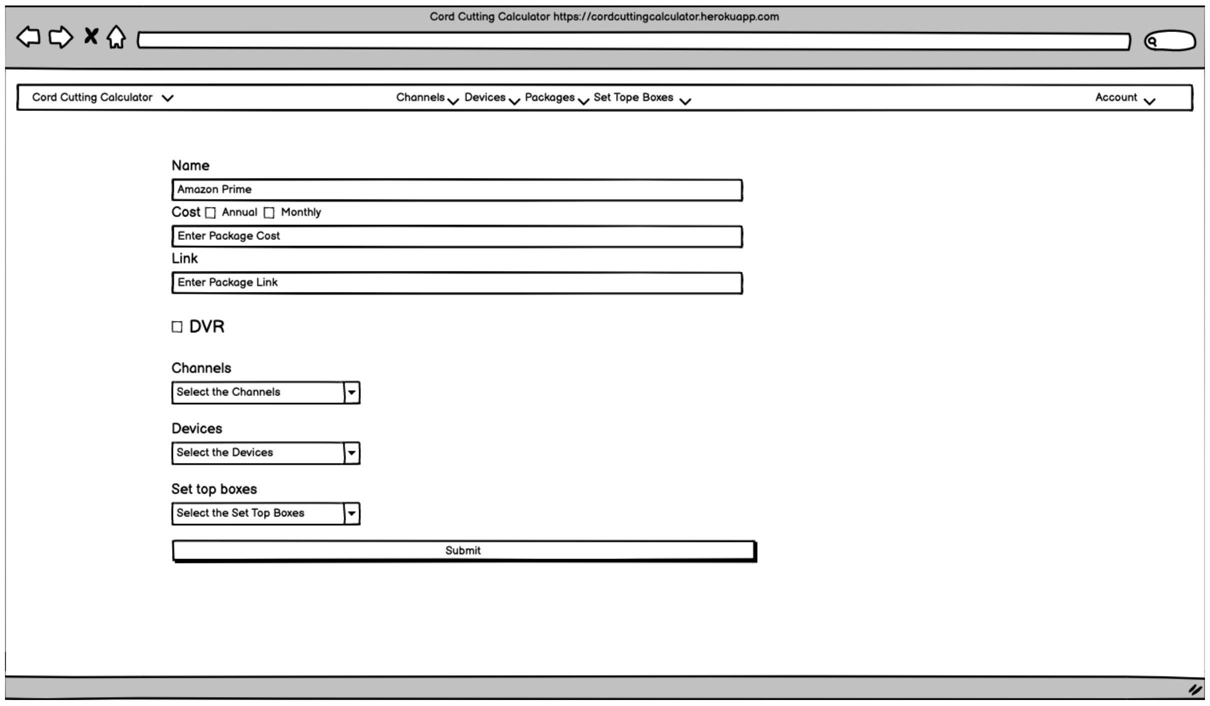
As a channel watcher

Currently the budget does not specify whether it is monthly or yearly

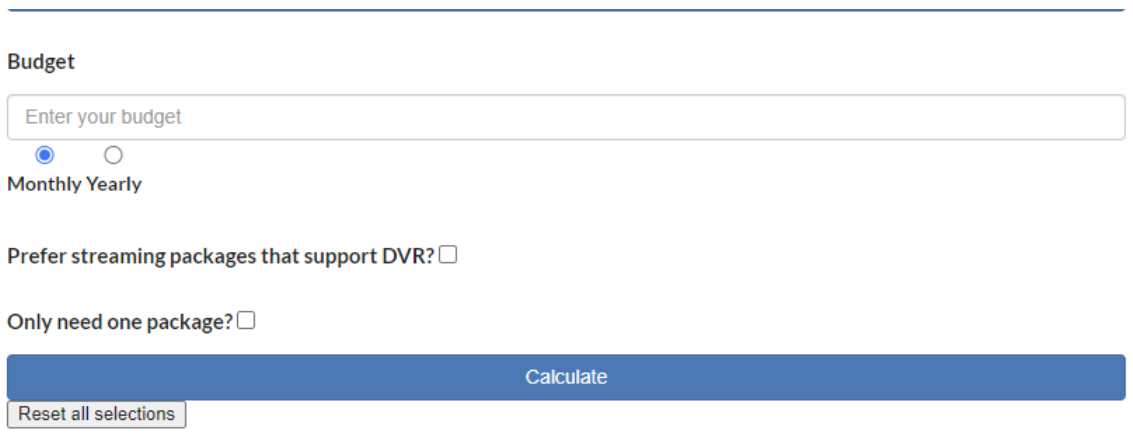
Specify this near the budget field

Before adding this feature, the team had to decide what this feature would look like and ultimately decided on radio buttons which *only* allow the user to select one or the other, rather than checkboxes which would allow the user to select *both* at the same time. The team felt as though radio buttons were more intuitive since most users will have either a monthly or yearly budget, not both. This feature is now fully complete and has been deployed to the Heroku app.

Low-Fi UI Mockup:



Deployed Feature:



Understanding the Codebase

Due to shortage of time we refrained from making any significant changes to the existing code base. However, we did have a discussion about what changes can be made in the legacy code such as fixing the name of perference.rb file without breaking anything—We have highlighted such modifications in the future work section of our poster and later in the document. To better understand the codebase we ran RSpec and Cucumber tests. These tests gave us a fair idea of the passing/failing scenarios within the application. Furthermore, we also learned about the existing functionality that was left untested.

Summary of Iterations

Iteration 0:

In this iteration, we looked through the legacy codebase and the app to better understand what it was doing currently and how this was implemented. This iteration focussed more on figuring out the current state and functionality of the legacy codebase, refactoring if required, and possible additional features according to customer requirements. This iteration was also more focussed in meeting with the customer and understanding his needs and requirements from us regarding changes to the legacy code. After understanding the customer's needs and taking into consideration our time limit, we created four unique user stories to add features that seemed to be missing but essential for improving the user's experience and ease of use of the application.

Iteration 1:

In this iteration, we started implementing some of our user stories. We implemented "Monthly" and "Annual" radio buttons for the user to choose between in the calculator view, and the underlying functionality. This pertains to user story 2 from Iteration 0. No prior code had been refactored in this iteration. We also broke down the "Add Reset button" story from Iteration 0 into sub-stories to add "Reset" selections buttons for User's Edit Antenna Channel/Set Top Box/Devices page and the Calculator Page, and implemented the "Reset" button to the User's Edit Antenna Channel and Set Top Box and Devices Page. The consequent blockers pertaining to proper implementation of these features were realized and noted in this iteration (mentioned in the iteration report)

Iteration 2:

Documented the steps required to successfully run the behavioral + technical tests for this project using a AWS Cloud9 environment. Additionally, we have now figured out how to run the app locally, which proved more difficult than our homework assignments. Both of these blockers are well documented to speed up future teams' progress. We also completed the Sub-story #4 for Adding Reset buttons to each type of channel selections in the Calculator Page, and applied some modifications to the implementation of the previously designed Reset buttons.. Also, there were some Cucumber tests which were failing from the legacy code for some of the signup tests, while the app was demonstrating appropriate signup implementation. These were identified and updated to verify that the implementation wasn't breaking at any point, and was in fact performing the appropriate role. This further improved the coverage.

Summary of Meetings

Meeting 4/7:

Learned that the biggest challenge of the app is keeping the channels and services updated. This challenge is best overcome by scraping or improved admin controls. If you see a lame story then propose a fix for it (then upon approval, fix it!). Database security is a concern, but not yet a priority for the app. Email verification is a possible user story to be implemented in the future, although it requires more effort than you would expect.

Meeting 4/14:

Current labels on features are confusing, we hope to fix these accessibility issues in future iterations.

For the drop-down list user story:

1) Avoid one big sublist of channels.
2) One channel should only be included in one category i.e. one genre/channel.
3) If the legacy code is broken, create a new user story for that.

Customer was satisfied with the given user stories and provided some suggestions as well.

Meeting 4/21:

Presented user-story progress to the customer. The customer was satisfied with progress.

Meeting 4/28:

Last minute discussion about our implemented features and what direction the app should take moving forward. The talking points are **also** found on our poster under "Future Considerations":

Possible Future Userstories (descriptions to help the next team):

- Increase visual density of interface (compact), but be careful not go overboard. Especially in the calculator menu where it features “must have”, “would like to have”, “ok to have”.
- Make the application's interface actually formatted for mobile view.
- Rank list of packages: Additional checkbox that is something like "show all packages ranked by <number of channels / price>". The app can give you all the ones that meet the requirements but also provide ways to sort the calculated results.
- Add more user email functionality such as confirming email account and add security to protect the emails in the database.
- Allow the customer to submit requests for channels, devices, and packages.
- Change perference.rb to be more configurable, currently uses embedded constants for its weighted values.
- Also fix the name of perference.rb without breaking anything! (Change it to preference.rb).
- Modify the channel table and database to allow for sorting and grouping by user-chosen criteria.
- “Only need one package?” checkbox in the calculator view does not function correctly, when used it will occasionally show a second package.

Testing Process

With a severe lack of time for this project, we focused on implementing simple, completable stories that wouldn't require extensive testing. Thus we were able to complete the features, but lacked the time and understanding necessary to write technical tests of the code. Behavioral tests of the code were written that will help future developers understand the intended behavior of our changes:

`Budget.feature`

`ResetButton.feature`

Also, there were some legacy code features which had been failing the Cucumber tests in the signup functionality. This was leading to as low coverage as 14.97% when we ran the tests initially. These were inspected to find out that while the app was demonstrating appropriate signup implementation, tests seemed to be checking the redirect to user profile using the wrong path. These were identified and updated to verify that the implementation wasn't breaking at any point, and was in fact performing the appropriate role. This further improved the coverage.

Code Management

We didn't make any changes significant enough to warrant creating additional branches. We forked the legacy code repo and shared our repository to get our team ready to make any changes to the codebase. All changes were pushed directly to `master` for convenience. We made sure that we frequently pushed our changes to our main branch, in order to avoid any stale branches. All merge conflicts were also resolved appropriately without any overwriting of working features from our own changes or legacy codebase. Also note that code should be *quality over quantity* and *functionality over appearance*.

Issues + Solutions

Heroku Issues

Don’t run `db:migrate` on the heroku application without knowing what you’re doing! You’ll inadvertently delete the pre-existing data.

Repository Issues

Fork the existing repository and add your team as collaborators, besides that just use best git practices.

Cloud9 Development Environment Issues

Running Tests

We faced **many** issues while trying to run RSpec and Cucumber tests in the Cloud9 environment. The list of steps we ran to set up the environment for testing:

1) `sudo yum install postgresql postgresql-server postgresql-devel postgresql-contrib postgresql-docs`
2) `bundle exec install`
3) `sudo postgresql-setup initdb`
4) To get rid of the postgres 5432 error run: sudo service postgresql start
5) To fix "role “ec2-user” does not exist":

   ```
   sudo -u postgres createuser -s ec2-user

   sudo -u postgres createdb ec2-user
   ```

6) Run:

   ```
   bundle exec rake db:create

   bundle exec rake db:migrate
   ```

7) Now running `RSpec` and `Cucumber` should work!

Running App In Cloud9 IDE

We also had issues getting the app to run locally on the EC2 instance, to fix these issues we needed to add “environment” entries to the secrets.yml and database.yml files. Although note that they are updated so you will simply need to run step 3.

1) Add "environment" to secrets.yml:

```
environment:
     secret_key_base: <insert secret key here>
```

Note: You can use rails secret to develop a unique hash, or you can just mash the keyboard.

2) Add "environment" to database.yml:

```
environment:

  <<: *default

  database: cordcut_test
```

3) `rails server -p $PORT -e $RAILS_ENV`

Additional Relevant Links:

Spring 2019:

Previous Cord Cutter: https://cord-cutting-calculator.herokuapp.com/

Previous Documentation:
https://github.com/Ziyi1215/Cord-Cutting-Calculator/tree/master/documentation/Spring2019

Fall 2019 (1):

Cord Cutter: https://revised-cord-cutter.herokuapp.com/

Revised Documentation:

https://github.com/dileep-g/Cord-Cutting-Calculator

Fall 2019 (2):

Previous Cord Cutter: https://cord-cutting-calculator-fall19.herokuapp.com/

Previous Documentation:

https://github.com/sbuvaneish/Cord-Cutting-Calculator