

YouthComputing
Inspire. Promote. Educate.
Open Challenges

Challenge Two

YC Store

Submission Deadline: July 26, 2020 @ 11:59PM MST

Difficulty: Medium

Type: Web Application



26-06-2020

Contents

1	Description	1
2	Design Requirements	2
2.1	Login	2
2.2	Shop	2
3	Documentation	5
3.1	GET/ api.youthcomputing.ca/shop/prizes	5
3.2	GET/ api.youthcomputing.ca/shop/prizes/prize-id	5
3.3	GET/ api.youthcomputing.ca/shop/promotions	6
3.4	GET/ api.youthcomputing.ca/shop/promotions/promo-id	6
3.5	GET/ api.youthcomputing.ca/users/uid	6
3.6	PUT/ api.youthcomputing.ca/shop/redeem/prize	7
3.7	PUT/ api.youthcomputing.ca/shop/redeem/event	7
3.8	PUT/ api.youthcomputing.ca/shop/new-user	7
3.9	PUT/ api.youthcomputing.ca/shop/admin/points	8
3.10	PUT/ api.youthcomputing.ca/users/admin/events	8
4	Judging Criteria	9
5	Submission Guideline	11
6	Remarks	12

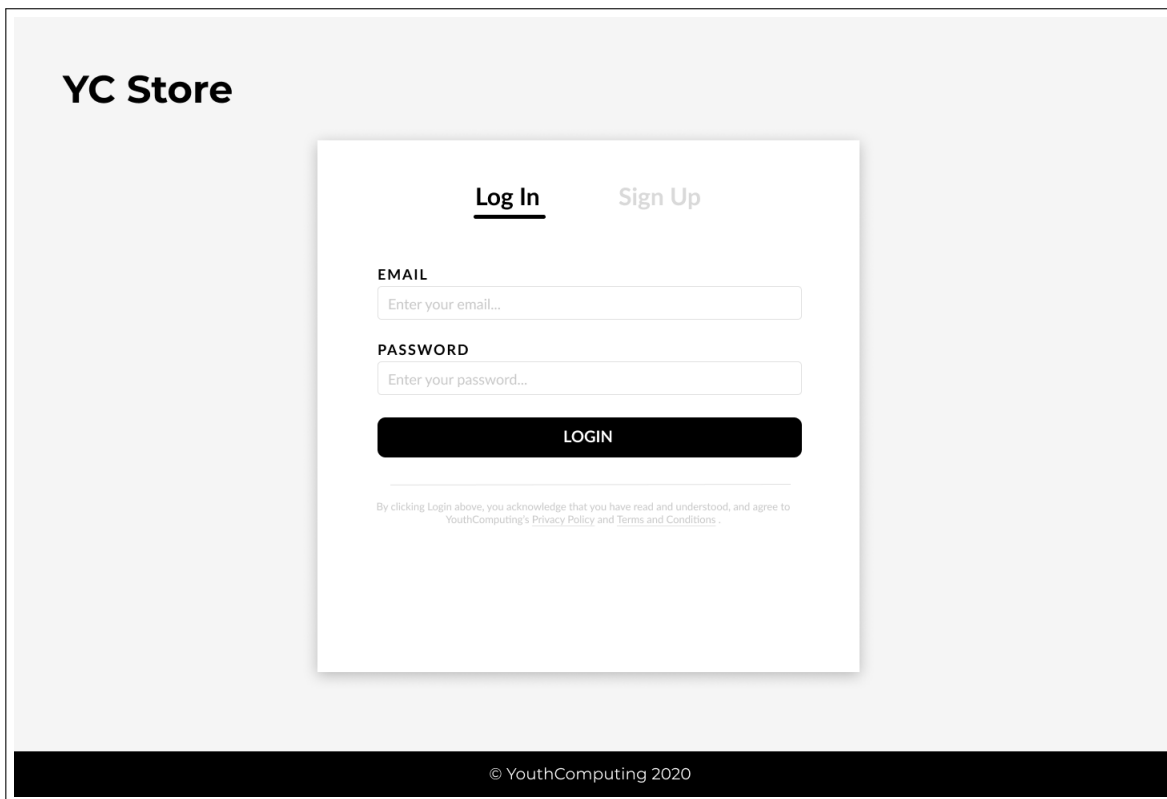
1. Description

The YouthComputing community platform is looking to prototype an online store to redeem virtual currency earned by attending events. When a user opens the online store, they should be able to browse and redeem prizes. Users should also be able to redeem events to points and check their balance. Your challenge is to create the application pages that correspond to the requirements outlined in this document. Make sure to read all of the details in this document prior to writing any code. There is a prize!

2. Design Requirements

These guidelines are not “rules” to follow, however they are recommended paths for the UI/UX design of your application. The majority of the design judging criteria is focused on UX, however some factors will be considered in the UI design. Please check the judging criteria for more details. The web application should have the following pages:

2.1 Login



The image shows a mockup of a login page for 'YC Store'. The page has a light gray background. In the top left corner, the text 'YC Store' is displayed in a bold, black, sans-serif font. Centered on the page is a white rectangular card with a subtle drop shadow. At the top of the card, there are two links: 'Log In' (underlined) and 'Sign Up' (in a lighter gray font). Below these links are two input fields. The first is labeled 'EMAIL' and contains the placeholder text 'Enter your email...'. The second is labeled 'PASSWORD' and contains the placeholder text 'Enter your password...'. Below the password field is a solid black button with the word 'LOGIN' in white, uppercase letters. At the bottom of the card, there is a small line of text: 'By clicking Login above, you acknowledge that you have read and understood, and agree to YouthComputing's Privacy Policy and Terms and Conditions .'. The entire page is framed by a thin black border. At the very bottom, there is a black horizontal bar containing the text '© YouthComputing 2020' in white.

2.2 Shop

Figure 2.1: If user is logged out

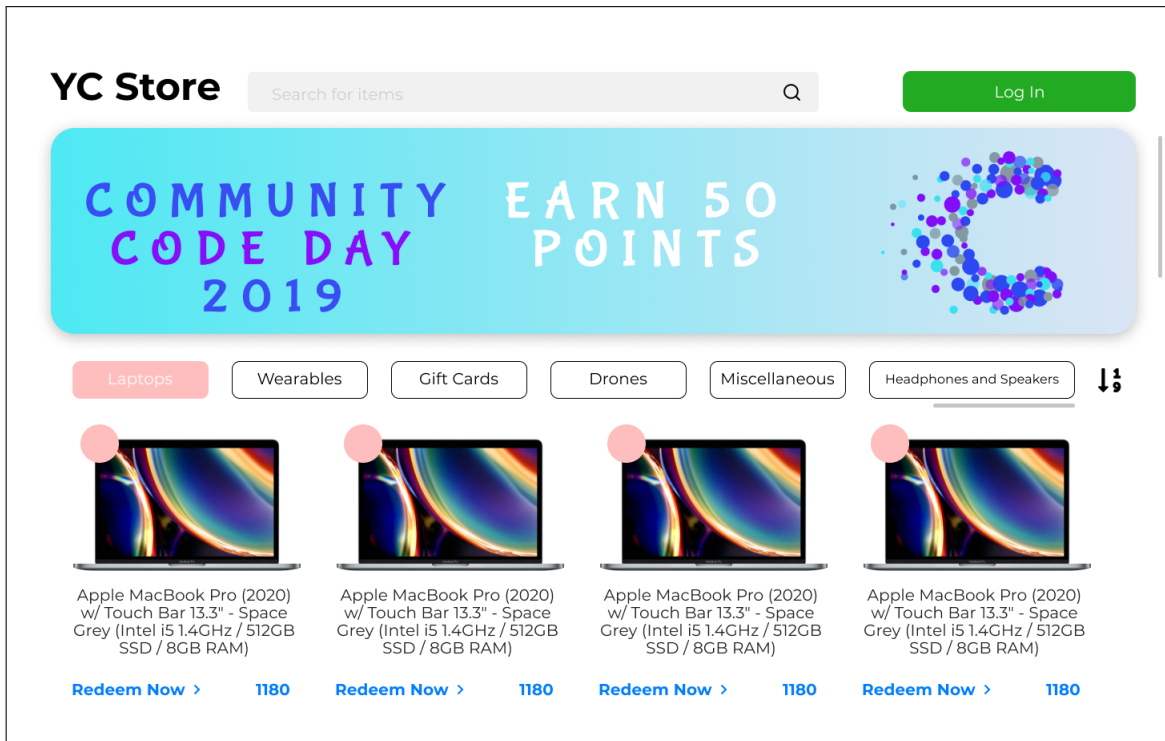


Figure 2.2: If user is logged in

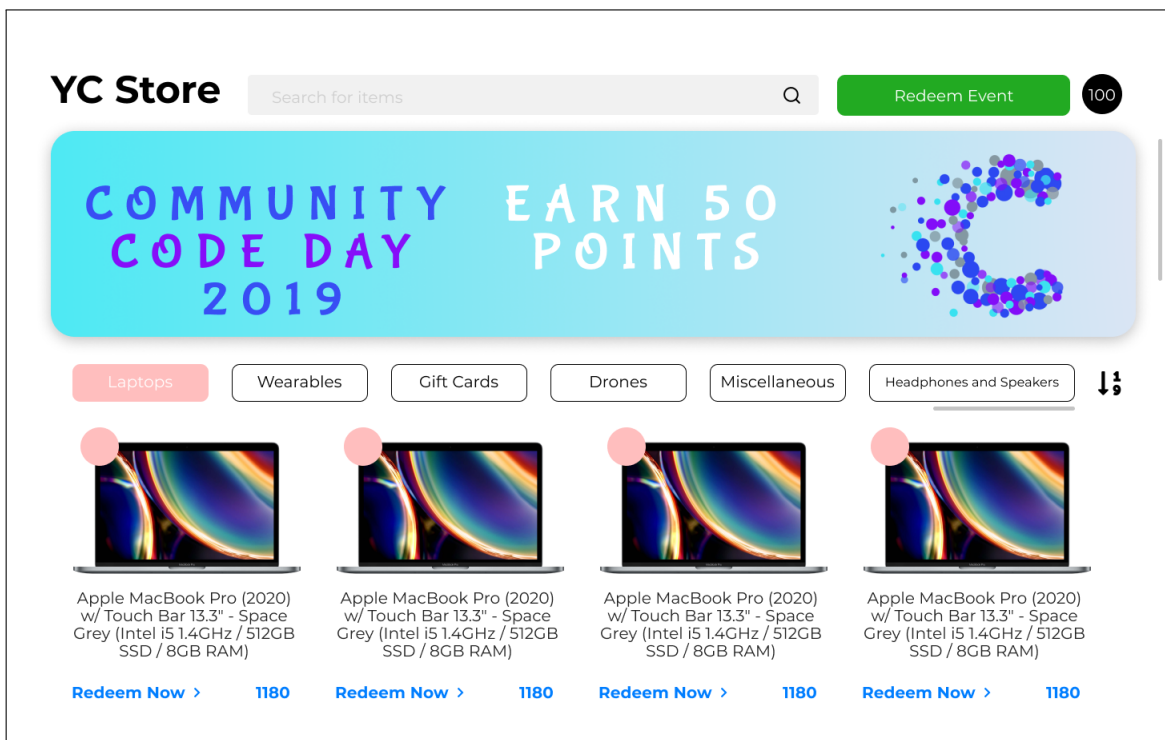
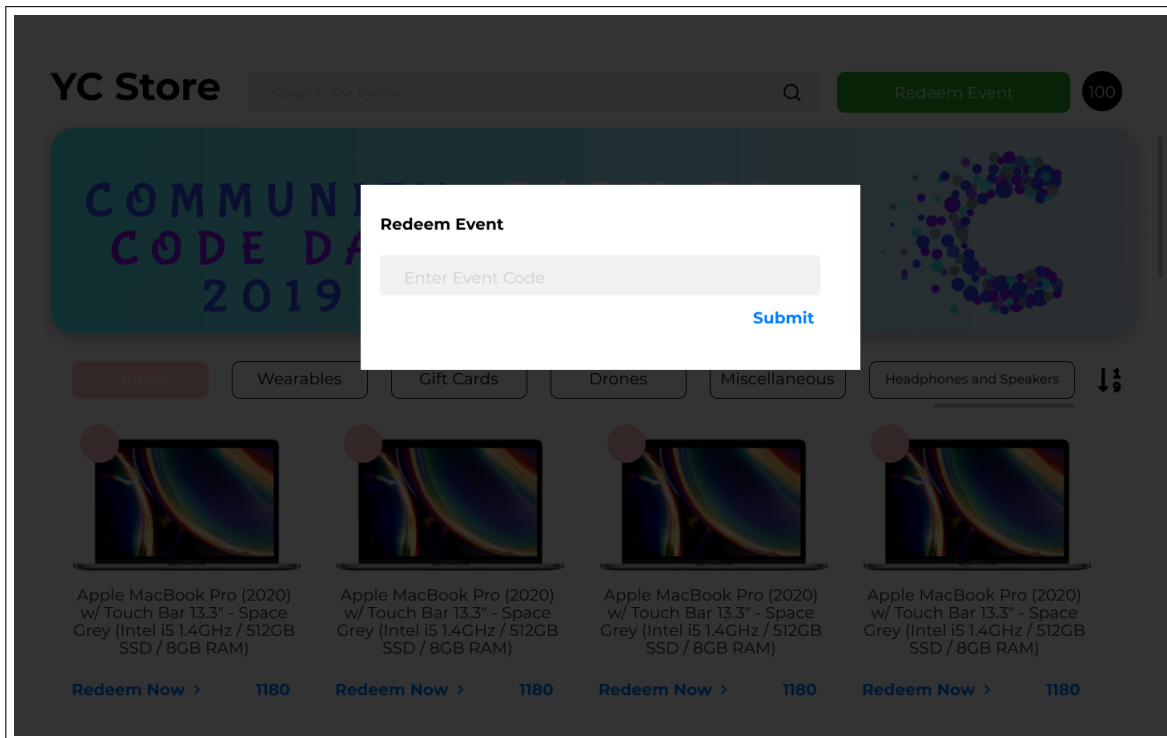


Figure 2.3: Modal for when the user clicks Redeem Event button



3. Documentation

In addition to interacting with the YC database via the YC REST API, you will also be implementing login and sign up for user management. YC uses Firebase for authentication and a Firebase API key will be sent to each group to use. Each group is provided an account to get started working on the app. After login in, Firebase will provide a user object with the user's uid which will be needed for API calls. After sign up, you have to call the new shop user YC API endpoint to setup the user in the database.

The YC Shop API has been made available to you in addition to the Events API that you already have from Challenge 1. Since event codes to get points should be private, instead of an API endpoint, a Google Spreadsheet has been shared with you: **Event Codes**. Below are the following endpoints and corresponding HTTP methods.

3.1 GET/ `api.youthcomputing.ca/shop/prizes`

Returns a list of the prizes in the shop. Specifically, will return the prize id, name, points, number of times redeemed, category, and image url for each prize.

Sample response:

```
{
  "prizes": [
    {
      "name": "Razer Blade 15 Base Model 10th Gen Intel® Core™ i7-10750H 6-Core (2.6GHz / 5.0GHz)",
      "points": 870,
      "redeem_count": 0,
      "category": "laptops",
      "image_url": "https://api.youthcomputing.ca/images/razor-blade.png",
      "id": "3LgPnAHYhi35QtJHYoFm"
    },
    {
      "points": 310,
      "category": "wearable_tech",
      "redeem_count": 0,
      "image_url": "https://api.youthcomputing.ca/images/apple-watch.png",
      "name": "Apple Watch Series 5 (GPS) 44mm Space Grey Aluminum Case with Black Sport Band",
      "id": "7btSTjjZma3C4CYxWm9o"
    },
    {
      "name": "Subway Gift Card",
      "points": 10,
      "category": "gift_cards",
      "redeem_count": 0,
      "image_url": "https://api.youthcomputing.ca/images/subway-25.png",
      "id": "9A7Sm4tlHgoHHJumryAc"
    }
  ]
}
```

3.2 GET/ `api.youthcomputing.ca/shop/prizes/prize-id`

Returns the data for the prize in the shop. Specifically, will return the prize name, points, number of times redeemed, category, and image url.

Sample response: <https://api.youthcomputing.ca/shop/prizes/3LgPnAHYhi35QtJHYoFm>

```

{
  "prizeData": {
    "points": 870,
    "redeem_count": 0,
    "category": "laptops",
    "image_url": "https://api.youthcomputing.ca/images/razor-blade.png",
    "name": "Razer Blade 15 Base Model 10th Gen Intel® Core™ i7-10750H 6-Core (2.6GHz / 5.0GHz)"
  }
}

```

3.3 GET / api.youthcomputing.ca/shop/promotions

Returns a list of the promotions in the shop. Specifically, will return the promotion id, promotion image, and promotion url for each promotion.

Sample response:

```

{
  "promotions": [
    {
      "event_url": "https://youthcomputing.ca/fuse",
      "image_url": "https://api.youthcomputing.ca/images/promo_fuse_2019.png",
      "id": "0T00KjW0nkbVrIlRyyeK"
    },
    {
      "image_url": "https://api.youthcomputing.ca/images/promo_ccd_2019.png",
      "event_url": "https://2019.communitycodeday.com/",
      "id": "UpADj6SvnXpMpjCd10GI"
    },
    {
      "event_url": "https://launchwoodbuffalo.com/",
      "image_url": "https://api.youthcomputing.ca/images/promo_lwb_2019.png",
      "id": "YVWmcGkwZGjPdcuXyA6c"
    }
  ]
}

```

3.4 GET / api.youthcomputing.ca/shop/promotions/promo-id

Returns the data for the promotion in the shop. Specifically, will return the promotion image, and promotion url.

Sample response: <https://api.youthcomputing.ca/shop/promotions/0T00KjW0nkbVrIlRyyeK>

```

{
  "promoData": {
    "image_url": "https://api.youthcomputing.ca/images/promo_fuse_2019.png",
    "event_url": "https://youthcomputing.ca/fuse"
  }
}

```

3.5 GET / api.youthcomputing.ca/users/uid

Returns data about the user. Specifically, will return the user name, points, an array of event codes redeemed, and email.

Sample response:


```

{
  "userData": {
    "name": "Kobe Bryant",
    "points": 33643,
    "event_codes_redeemed": [
      "ball-is-life"
    ],
    "email": "kobe.bryant@youthcomputing.ca",
  }
}

```

3.6 PUT/ api.youthcomputing.ca/shop/redeem/prize

Will subtract the prize points from the user's points and update the redeem count of the prize if successful.
Required body (as JSON):

```

{
  "userId": "uid",
  "prizeId": "pid",
}

```

3.7 PUT/ api.youthcomputing.ca/shop/redeem/event

Will add the event points to the user's points and add the event code to the user's redeemed array if successful.
Required body (as JSON):

```

{
  "userId": "uid",
  "eventCode": "code",
}

```

3.8 PUT/ api.youthcomputing.ca/shop/new-user

Will setup an initial user account in the shop for the user (points = 0).
Required body (as JSON):

```

{
  "userId": "uid",
  "userName": "name",
}

```

3.9 PUT/ api.youthcomputing.ca/shop/admin/points

For testing purposes, this endpoint has been made available for you to modify the user's points. Will set the user's points to the value you provide it.

```
{  
  "userId": "uid",  
  "points": 24,  
}
```

3.10 PUT/ api.youthcomputing.ca/users/admin/events

For testing purposes, this endpoint has been made available for you to remove an event code from the user's redeemed event code list.

```
{  
  "userId": "uid",  
  "eventCode": "code",  
}
```

4. Judging Criteria

Your groups have been posted, and you are responsible for collaborating to solve the challenge. After the submission deadline, the submission form will be closed (no exceptions). A winning team will be chosen and announced by **August 17, 2020 @ 9:00AM MST**.

Before the judges come up with the results, there will be a 15 min call with each team, asking specific team members questions about the implementation. Everyone should understand the the design, code, and any other parts of your process of implementation. The idea of this team call is that everyone comes out with a similar level of understanding of the problem solved.

The judging criteria is as follows. The app should meet the following requirements:

- Users should be able to view all of the prizes with:
 - Item name
 - Points needed (assume prize points in the range $[0, 999999]$)
 - Image
 - Category indicator
- Each time the user loads the page, a new (randomly picked) promotion banner should show up with an image at the top of the page.
 - If the user clicks on the banner, it should direct the user in a new tab to the corresponding event website.
- Users should be able to view the store with or without being signed in, however should be prompted to sign in if they attempt to redeem an item.
- Users should be able to view their points balance if logged in, or be able to login.
 - assume user points in the range $[0, 999999]$
- Logged in users should not be able to access the login/sign up page (i.e. redirect them to the shop page).
- Users should be able to redeem an item.
 - If successful (has enough points):
 - * The user should receive a message telling them they were successful in redeeming the item and their points in the circle should update.
 - If failed (not enough points):
 - * The user should receive a message telling them how many more points they need to redeem that specific item.
- Users should be able to redeem points for going to an event.
 - If successful (correct code and not yet redeemed):
 - * The user should receive a message telling them they were successful in redeeming the event and their points in the circle should update.

- If failed (incorrect code or code already redeemed):
 - * The user should receive a message telling them their code is incorrect or it has already been redeemed.
- Users should be able to search for items by:
 - Name
- Users should be able to filter by:
 - Category
 - * **Note:** the category buttons should be a horizontal scroll if the buttons do not fit on the user's screen.
- Users should be able to sort the items by:
 - Points
 - * High to low
 - * Low to high
- The list of items should be paginated via a vertical scroll.
 - In other words, a certain number of items should be shown when the user first loads the website. Then, as they scroll down, more items should be loaded (think: Instagram app loading more photos as you scroll).
 - As the user scrolls, the header: title label, search bar, redeem event button, and points label should stick to the top of the page.
- There should be a footer with a YC copyright at the bottom (shouldn't be sticky) of all pages.
- A simple login and sign up page should be made available to the user.
 - Users should be able to sign up for a new account with appropriate error handling.
 - Users should be able to login to their account with appropriate error handling.
- Users should be able to log out.
- Users should be able to view their name when logged in.
- The responsiveness of your application will not be assessed, but it should work on a standard Macbook 13" screen. **Bonus points** if your website is responsive!
- The web app must be hosted and accessible from the public web.

Your final score will be a combination of:

$$\text{requirements score (+ bonus)} + \text{team score} + \text{code quality} + \text{vote (UI + UX)} = \text{final score}$$

In the event that multiple teams have the same final score, the submission with the earliest submission date and time will be chosen as the winner.

Note there is a prize: The winning team will get to pie the other team with everyone watching.

5. Submission Guideline

All code **MUST** be submitted through a Github repository. Zip files of the code or any other means will **NOT** be accepted.

To submit your code once you're done, fill out the following Google Form:

- <https://forms.gle/bz4VHmUQ25mqavLT6>

You can submit multiple times, but your latest submission before the submission deadline will be considered as the final one, and the datetime stamp for your final submission will be used.

6. Remarks

If you have any questions about the challenge, reach out to us at info@youthcomputing.ca.

Good luck!