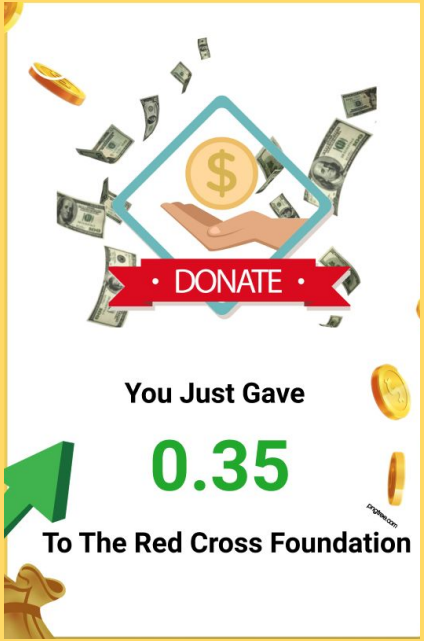
DonateUp

Team: Cyber Success

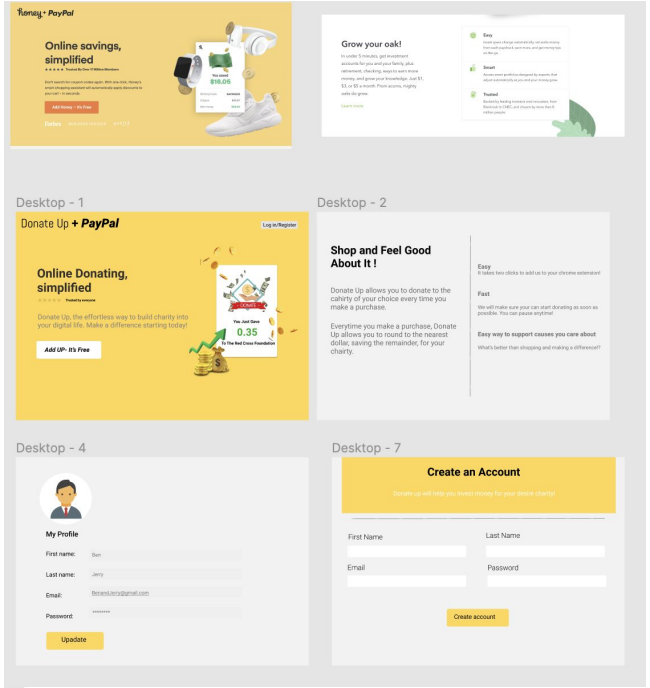


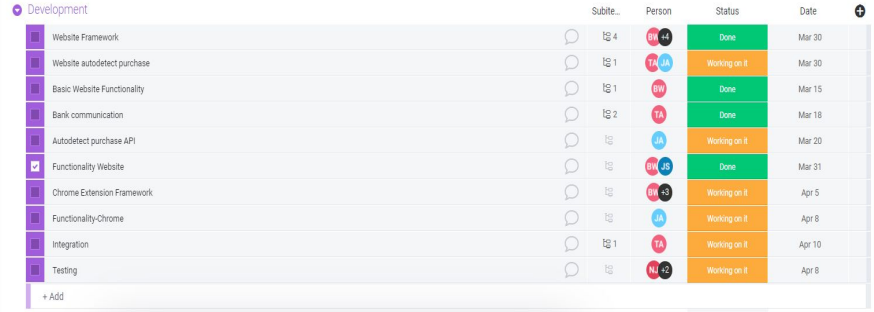
Members: Eliza Mimi To , Jesse Andringa, Timothy Ashour, Nathan Jager, Jeremy Schlagel, Benjamin Weiler

https://docs.google.com/document/d/1V4OG9k7Ve2v0Wtc1XXs0n_J3ebou67djJj9o_CeSE8c/edit SPEECH :)

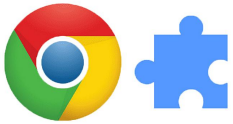
Tools we used

1. Figma: figma let us design and mess around with the UI of the chrome extension and the website. We give this a 5. Because it was easy to use and very handy.
2. Monday: monday was where we planned our user stories and our workflow. We give it a 3 because it was only somewhat easy to follow and had more confusing aspects than needed.
3. Github: Github held our repository, and was extremely helpful when collaborating on our code. We give it a 5.
4. Discord: Discord is where we discussed and bounced ideas off each other and collaborated over voice and video calls. We give discord a 5 for its convenience.





Tools



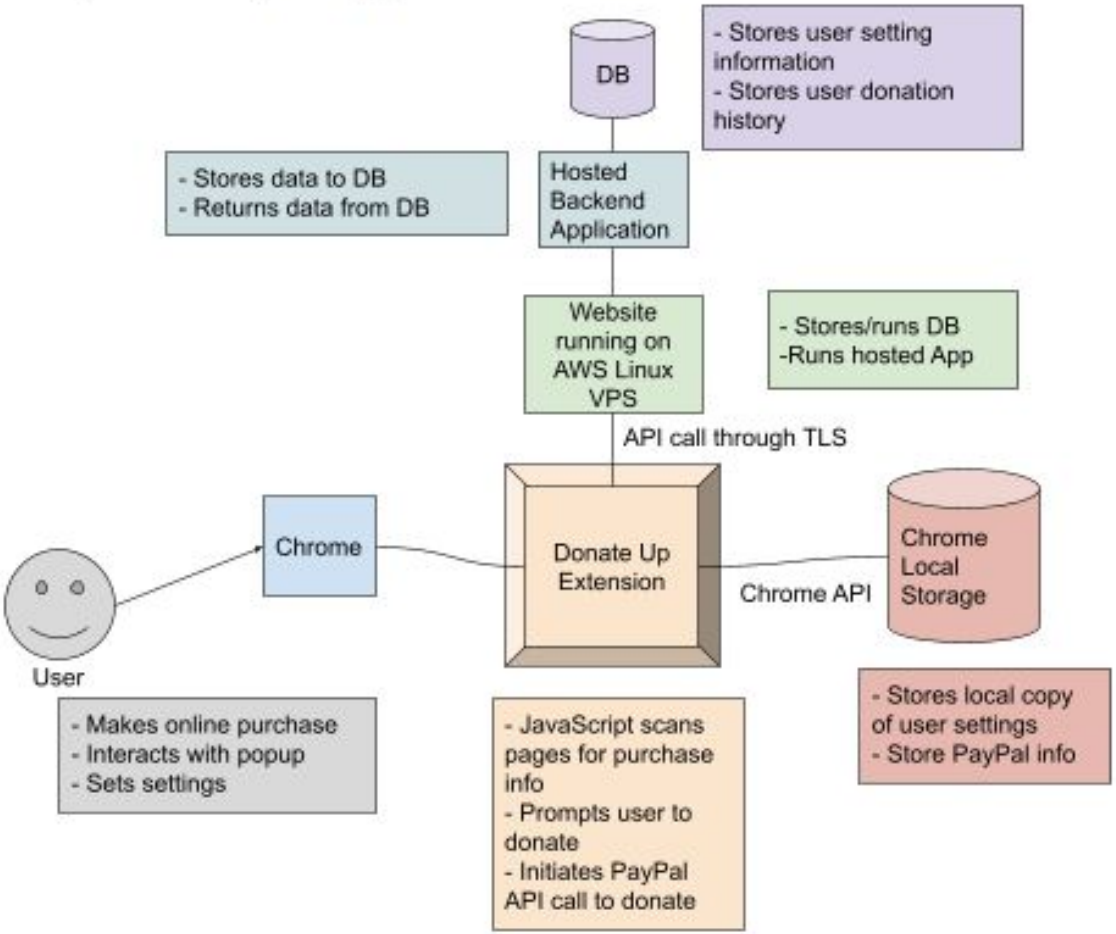
o Database: PostgreSQL. Used to store donation and user information. It was difficult to figure out how to integrate at first, but works well with our simple code. Rated 5.

o Deployment environment: AWS. Used to host the website through a free EC2 instance. Difficult to provision good access and hard to add our needed packages. Most likely it would have worked better if we used containerization and AWS services other than EC2. Rated 2.

o Framework: Flask. Used for the web app to integrate our python functions, HTML pages and our database. Very easy to pick up and write working code. Rated 5.

o Extension Platform: Chrome: Used to pop up and ask user if they'd like to donate. Chrome is the host of the extension package. Chrome's documentation is weak, but basic extension is easy to figure out, and can be very powerful. Rated 4.

Architecture Diagram



Challenges we encountered

1. Handling paypal transactions: we did not want to deal with the high level of security needed in handling money. We then made a prototype where we track all of the data of each transaction, but never handle real money.
2. Skimming the price of the purchase of a website. Cart pages are high level security and make it much more difficult to skim through html. We made a API that skims certain pages, but for our general chrome extension we changed and offered set donations: $.25, $.50 etc. instead of scraping data from the webpage.
3. Integration with database. It took a bit of testing and documentation reading to figure out how to integrate flask with the postgresql database.