# Final Report
# Cracked Pepper

# API Requirement

## APIs used to build Virchu:

### 1) Twitter API:

- Users can use the Twitter feed which uses the Twitter API to make informed decisions about who needs the most help. For example people may be donating to a cause and sharing about it in social media, users would donate to them as opposed to other countries. We have used the Twitter embeds API to embed Tweets and display relevant data.

### 2) Stripe API:

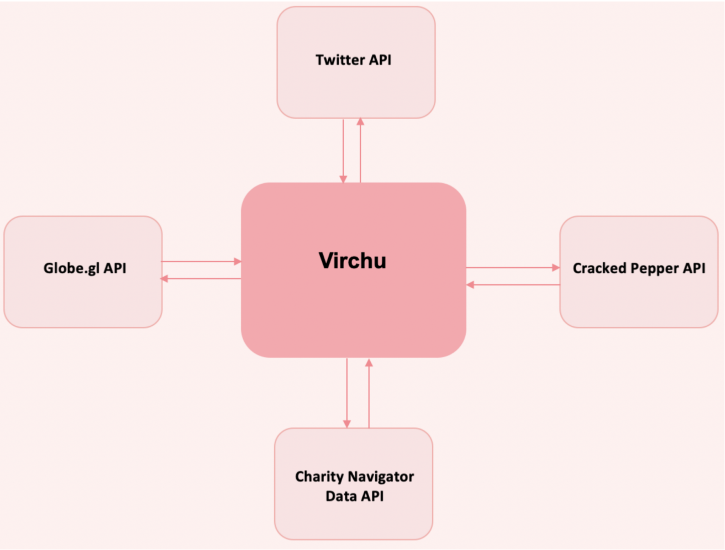
- Used the stripe API to streamline donations so that users do not need to visit the charity site when they want to donate.

### 3) Globe.gl:

- Users can move around the 3D interactive globe API which shows the countries affected and click on the country to view the major cases and find a list of charities to which they can donate.

### 4) Cracked Pepper API:

- The Cracked Pepper API is used to retrieve articles in order to keep the users up to date with the latest news about the pandemic.



# Use Case and Requirements of Virchu

## Virchu's Use Cases:

| No. | Use Case |
|---|---|
| 1 | As a user I want to see the latest tweets containing information about pandemics, or epidemics that are currently happening in the world so that I have all the tweets in one place. |
| 2 | As a user I want to view the charities from all around the world on a map in order to easily locate the charity. |
| 3 | As a user I want to view the outbreak on the globe in countries so I can easily compare the outbreak in different countries. |
| 4 | As a user I want to read the latest articles based on the disease and country so I am up to date. |
| 5 | As a user I want to donate to the charity directly instead of being redirected to the site. |

## Virchu's Functional Requirements:

| No. | Functional Requirements |
|---|---|
| 1 | Twitter feed that retrieves the latest medical information based on the country chosen. |
| 2 | Interactive globe that displays the list of charities in its relevant country. |
| 3 | Colour code the countries in the globe based on the outbreak information retrieved. |
| 4 | Articles are retrieved from the database and presented to the users. |
| 5 | Donations are made through the website. |

## System Design and Implementation

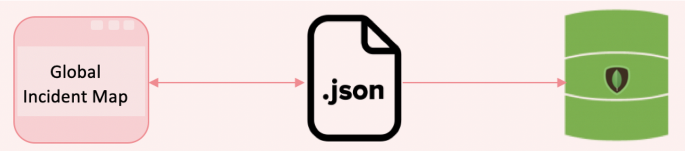
- The design has not changed from the previous iteration. However we decided to prioritize the use of the Outbreaks resource in our displayed globe. Hence the carousal and latest new articles currently show static data that will be update by us to encourage and bring greater attention to obscure/unheard of pandemics/outbreaks. We intend to make them dynamic in the future.

- As a result, we decided to use a different team's API (Akuna Capital API), and therefore we no longer need to use our data which is stored in MongoDB Atlas. Hence, our stack is last dependant on MongoDB, however new data such as user details will be stored there.

- Being a react project, we structured our frontend according to react conventions. This meant that our code structure made use of componentization to abstract different functionalities and write them in separate files to increase code readability. Another convention we followed was to use index.js as a medium for exporting the different components so they can be rendered on the front end view.

### Software Architecture:

1) Scrapper:
   - Used threading to increase requests from 1 request/second to approximately 150 requests/second.
   - We stored the data scrapped in mongoDb cloud storage.
   - Reason behind choosing MongoDb over MySQL is that MongoDb stores information as a series of JSON-like documents while MySQL uses relational systems. This was a huge advantage as the main source of Virtchu's data are stored in JSON files.

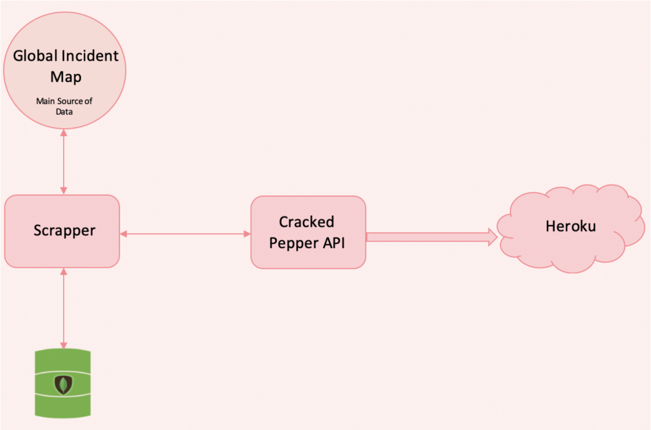


   - Used Mongoose to verify data with predefined schemas ensuring clean data enters the database.

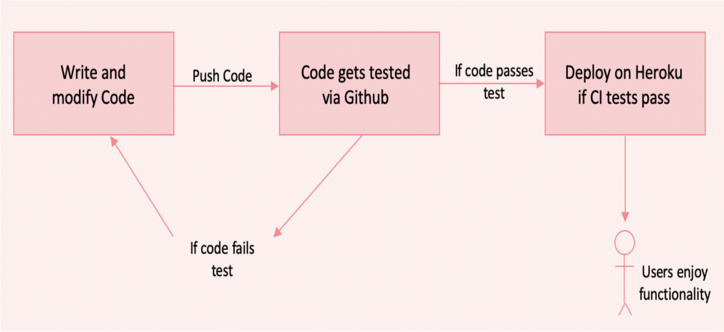
2) Cracked Pepper API:
   - We then made this data accessible through a well documented API.
   - We used nodeJS and Express framework since:
     - To add authentication and authorisation preventing abuse and ddos.
     - To have routes to decouple code making it submit to best coding practices.

    - To have a middleware to easily add logs to monitor performance of different endpoints.

3) Deployment:
    - We hosted the API on Heroku:
        - As it contains premade buildpacks that easily prepare a nodeJS for server deployment.
        - Integrates with GitHub for build versioning
        - Wide range of add ons including log analysis
        - Heroku is way simpler to use than Amazon web service as it equips us with a ready runtime environment and application servers, while Amazon doesn't.



4) Continuous Integration and Continuous Deployment:
    - Used precommit hooks with eslint to lint committed code, and weed out bad code.
    - We have integrated Continuous integration and Continues deployment practices by utilizing GitHub actions and Heroku configurations. Hence every time we push to the repo automatic tests written with Mocka and chai are run, this ensures the quality of code and minimises bugs present when it reaches users through Heroku.


Write and modify Code
Push Code
Code gets tested via Github
If code passes test
Deploy on Heroku if CI tests pass
If code fails test
Users enjoy functionality

## External APIs used:

- The external APIs used in the development of our webapp were the twitter standard API and the charity navigator API.

- We used these APIs for the development of the twitter feed and the charity-per-country features respectively. For the twitter standard API specifically, in order to fetch tweets relating to a specific type of content, we use a content filtering stream which made use of 'rules' to fetch tweets and posts containing certain words and media.

- We also used another Twitter API called Twitter embeds, such that after querying relevant ids of Twitter Tweets, we are able to fetch the rest of Tweet values such as profile picture, message, likes and shares. And easily parse them into tweets.

- We used a globe API to render a 3D globe in react which is interactive and we linked the data pulled from the Akuna Capital API to each individual country in the globe. We also used the Charity Navigator API to list the charities for the respective for which the users may want to donate.

## Team Organisation

1) Responsibilities:

- All team members have worked equally.

| Responsibility | Team Member |
|---|---|
| Cracked Pepper API | Mohammad, Mustafa, Sarah |
| Testing API | Mohammad, Mustafa |
| Researching third party APIs | Veeraj, Mohammad, Pramith |
| Generating the UI | Mustafa, Pramith, Mohammad |
| Handling the database | Sarah |
| Scrapping | Veeraj, Pramith |
| API Documentation | Mohammad |
| Generating reports | All team members |
| Presentation | Veeraj, Sarah |

2) Major achievements:

- Having some dynamically functioning features on our webapp.
- Being able to use theTwitter standard API.
- Getting acquainted with ReactJS and learning how to use proper react conventions.
- Having a balanced frontend UI and making use of some design principles.
- Conducting some good research on the market concerned with our app and getting results we could work with for our business pitch.
- Learn how to use github actions alongside mocha and chai.
- Learn how to deploy a server to Heroku, and create a CI/CD pipeline.

3) Issues encountered:

- Accessing APIs especially Twitter API was time consuming.
- We were short on time to implement more features.
- Most of the team members did not have experience in reactJs.

4) Skills gained:

- Greater familiarity with reacts class components rather than functional components as many of the libraries we used have used class components
- Better management skills and greater documentation of workflow, such that all members actively use github issues, milestones and boards to highlight problems and indicate that they have been fixed.
- Better understanding of how to use third party APIs.