**Pipelining Information Channels for Future Expansion**

# Introduction

90% of all blockchain projects never get past the experimental stages. [1] With such a bad track record of success, it often seems risky to venture into any investment regarding blockchain. In this project, we work towards minimizing these failures with a company that is built around the blockchain ecosystem.

# What is Blockchain

The blockchain industry is one that is still in its growing stages. In 2017, the global market value for Blockchain sat at $708 million. This value is predicted to increase to around $60.7 billion in 2024. [2] It is around this growing market that The Blockchain Challenge (TBC) hopes to build its platform around. We will now explore how they intend to achieve this.

# The Company

The Blockchain Challenge is a company built around the blockchain ecosystem. They strive to build a platform where users can interact, share knowledge and make the most out of blockchain technologies. They have identified 3 major user groups within their ecosystem:

1.       Users – The programmers within the current network

2.       Platforms – The technology providers within the current network

3.       Companies – The providers of the problems that need to be solved

TBC creates these interaction channels using multiple third-party tools like Slack, GitHub, Devpots and MailChimp. This creates the issue where information is unstructured and very hard to reorganize into valuable insight for the platform.

The company is also currently in the initial stages of its development and lack the capacity for data discovery. Thus, they have tasked us with the following problem: With an unstructured platform, how can TBC gather valuable information that could help them curate high-value interactions amongst their user groups?

# Problem Analysis

This problem is one of data discovery, but as we have mentioned TBC is a new company with very little structured or unstructured data to analyses. Thus, to answer the questions set out for us by TBC, we will have to draw from external sources to better understand the existing patterns within the blockchain ecosystem. Following which will develop a solution around this reorganization.

To ensure that we will be able to deliver a viable solution within the project timeframe, we have also opted to focus our project on deriving benefits for Companies and Technology Providers within the platform. We opted for this focus as TBC’s forecasted revenue mainly derive from subscription fees paid by companies and platforms that are in their network. Thus, it would be wise to establish a system that first derives value for companies and platforms. Also, in making this decision, we noted that although our focus was on companies, many of the outcomes will benefit the other user groups as well. Thus, this focus in not contrary to TBC’s overall business strategy.

# Our Process

We have settled with a 3-stage process for this project:

1.       Research

2.       Reorganize

3.       Retool

In our research phase, we focused on finding existing evaluation tools, case studies, and patterns within the blockchain industry. We then took the lessons learned from this phase to the next phase, reorganize, to reorganize the current channels that TBC uses in order to simplify the analysis process. Finally, in stage 3, retool, we created a tool for TBC to access these reorganized channels and pull out valuable insights.

# Stage 1: Research

In this phase, our 2 key sources for information were case studies of past successful blockchain projects and industry reports by companies like Gartner. We shall first cover the findings of our case studies.

## Case Study 1: Walmart

Walmart’s venture into the blockchain industry started with a simple question. How do we track the source of our food projects to ensure that they are safe for consumption? [3] A system that could answer this question could save Walmart a lot of resources when food-borne disease outbreaks occur. They can easily identify affected products from affected areas and discard them, ensure customer safety while still not incurring heavy losses from indiscriminate disposal of products that might not have been affected.

As simple as this sounds, creating a viable solution that can answer such a question would require information gathered from multiple suppliers within their supply chain who each had their own suppliers. As such, it is hard to guarantee the authenticity of information when there are a lot of participants in the whole system.

It turned out blockchain is a great technology to solve this problem. Its verification mechanism and irreversible data record can provide both transparency and traceability. Through the cooperation with IBM and GS1, Walmart launched two Proof of Concept project in 2016. One year later, they reduced the tracking time of the mango in the US market from 7 days to 2.2 seconds. [3] [4]

However, Walmart realized that the true value of such a system can only be achieved with its widespread adoption. Thus, after the success of the POC projects, Walmart continued cooperation with IBM to launch IBM food trust. Now, IBM food trust has also attracted companies like Nestle and Unilever to join the ecosystem and promote cooperation in the whole industry. [4]

From this case study, we see the value of the interactions within companies and how it is the backbone for a successful application of blockchain to solve a problem. We also see the potential that TBC could have on the acceleration of such interactions within the blockchain industry. Having the right tools to connect the right companies and platforms can help accelerate the development of future projects, allowing them to achieve the success of IBM food trust through the interactions curated by TBC.

## Case Study 2: FedEx

FedEx is also using a blockchain ledger to record and store all information pertaining to a transaction and acts as a single source of truth for customers, sellers, customs officials, delivery services and so forth.

In May 2019, FedEx CIO announced that they are looking to Industry Collaboration to Scale Blockchain. Now, FedEx has banded together with competitors – DHL Express and UPS -- to hammer out blockchain standards that could be deployed industry-wide. All three shipping giants are part of the Blockchain in Transport Alliance (BiTA), an industry organization with more than 500 members. [5]

Again, we see that for many successful blockchain proof-of-concept projects, the next big step is industrial cooperation. Blockchain’s nature as a decentralized system requires the participation of multiple parties before the full value of its implementation can be achieved. Thus, at a specific stage of success, companies will try to scale the blockchain by cooperation with other companies to encourage an industrial wide adoption of the technology. Thus, TBC should pay close attention to curating such interactions as it might be the key to blockchain’s success as a technology.

## Case Study 3: Health Care

We also find two other cases targeted in the health care field. One is about Medicalchain. [6] This project has a quiet logic way to structure what they want to do, the values and targeted clients that aligned with the methodology of key questions to ask. The other case that shared a similar use case is a solution from one of the hackathons launched by TBC, named “Hippocrate”. [7] Both projects focus on leveraging Blockchain techniques to solve health care problems.

From this case, we realize that it would be valuable for companies to know what has been tried before. It will be valuable, for companies especially, to know what technologies have been applied to what problems to better plan the implementation of their solution. We also see the value of categorizing TBC’s hackathon solutions into a single repository of knowledge. Having such a repository can help TBC delivery quick and valuable suggestions to companies on what past problems have been solved with what technologies. Helping them to plan their implementation better though the lessons learned by users during hackathons.

However, to achieve this, we need to have a way to categories projects. Thus, we investigated existing patterns within the blockchain industry to see if we can learn anything from it.

## Other Evaluation Tools

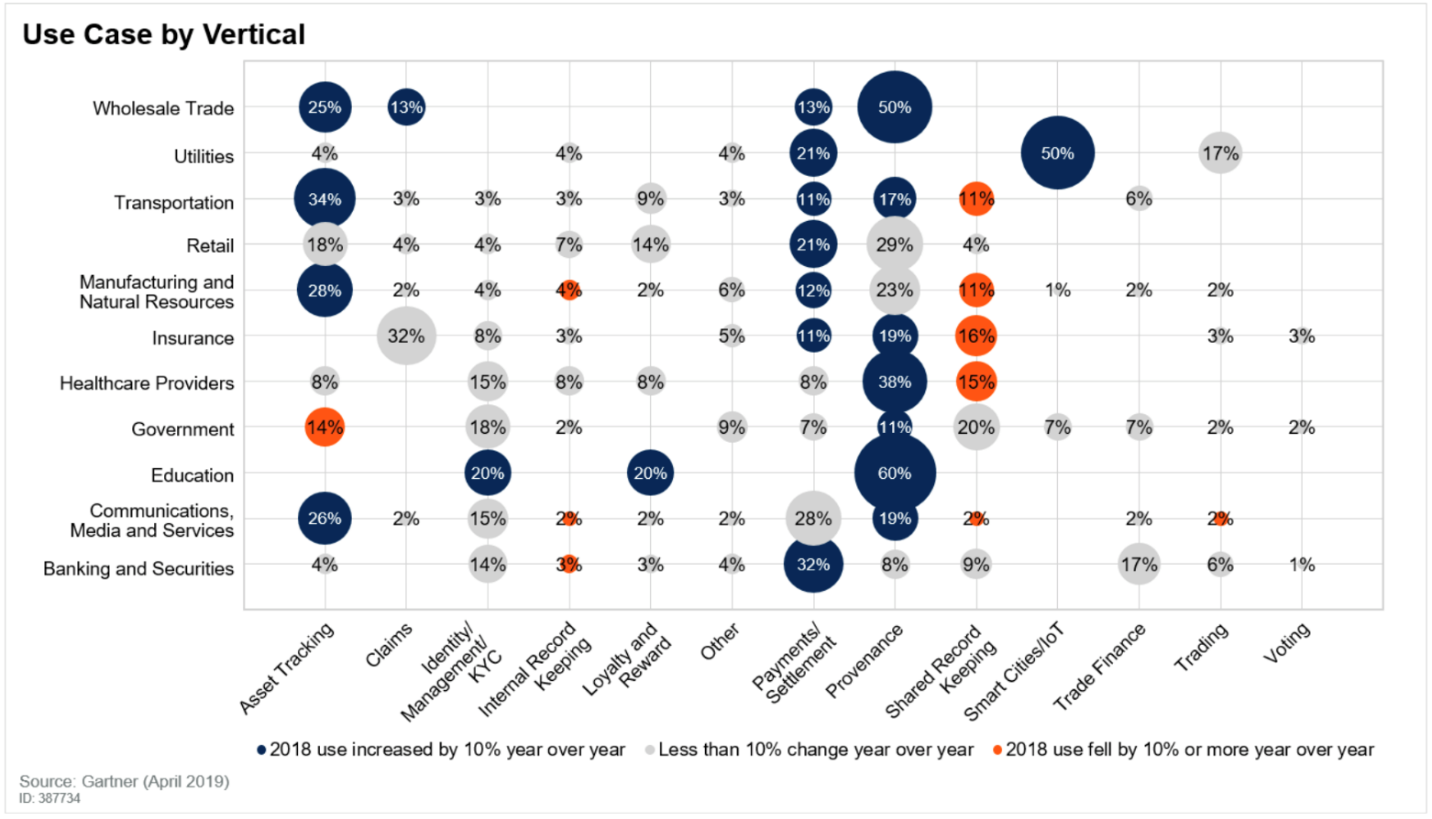


Figure 1 Use Case by Industry and Problem Type [8]

From our research, what we found, which is summarized by this graph produced by Gartner, was that a pattern did exist in the Blockchain industry. It seems that the growth of the adoption of blockchain solutions to solve problems varies in different industries by problem type.

Referring to Figure 1, we see industries like Education seeing strong growth in the adoption of blockchain to solve provenance problems but no use cases in the payments and settlements problem space. Conversely, when we look at the Retail industry, we see little change in the problem space of provenance, but we see growth in the payments and settlements problem space.

This is a simple pattern to understand as payments and settlements are a core aspect of retail while proof of certification is a core aspect of the education industry. However, there are many more patterns that are not as clear as this. Thus, having a system to capture these patterns within TBC can really help provide value for the users of their platform.

## Learning outcomes

We distilled 2 key learning outcomes from these case studies and tools:

1. Company to company interactions seem to be highly valuable in the overall success of blockchain projects
2. Categorizing projects by industry and problem can be a strong basis to reorganize TBC’s platform around

# Stage 2: Reorganize

With the lessons learned, we moved on to looking at the communication channels available in TBC and how we can improve their organization to gain the most information out of them. In our project, we chose to focus on Slack and GitHub as the 2 channels that we will reorganize and retool for.

## Slack

For now, each hackathon owns a workspace with two or more channels. The “hackathon” channel and the “general” channel are the default channels that are created upon the initialization of each hackathon workspace. However, these channels are usually filled with administrative information and do not hold much value for TBC.

To streamline searches, it will be beneficial to create separate channels for different topics during each hackathon for specific purposes. Our suggestion for slack reorganization is to create three main channels:

1. General hackathon channel for important notifications.
2. Technology channel for participants to ask or answer technical questions.
3. Business channel for participants to discuss business-related questions

Having specific channels would help to increase the value of searches done on Slack. For example, having a keyword search for the word “Ethereum” within a message within the Technology channel will help to gain more insight into technical questions around Ethereum. If the same is done without the channel reorganization, the information retrieved might contain administrative updates on the next Ethereum based seminar or other nontechnical questions related to Ethereum. The same applies to searches on other parameters like certain timeframes or within certain hackathon workspaces. Thus, this reorganization is key to having more specific search results.

## GitHub

GitHub is an important channel for storing and reviewing Hackathon solutions. We suggest TBC govern the submission format of every solution for hackathons by following these instructions:

1. Ensure that all projects have a GitHub repository
2. Add structured headers to README.md files

* Project name
* Project problem type
* Project industry
* Technologies used
* Participants

An example would be like:

# Project name

AlgoMed

# Project problem type

Provenance

# Project industry

Health Care

# Technologies used

Progressive Web App, Algorand SDK, Tezos Smart contract

# Participants

John Smith: jhons@andrew.cmu.edu<br/>

Ellen Walkar: [ellenw@andrew.cmu.edu](mailto:ellenw@andrew.cmu.edu)

# Stage 3: Retool

This section of the report goes over the basic structures of the tools we developed. Full documentation of the tools can be found [here](https://github.com/CMU-Capstone/TBC-Final-Deliverable/tree/master/Documents/API%20Manuals).

## Slack Scanner

Currently, TBC creates one workspace for each hackathon they hold, and the usage for each channel is not defined clearly. Our recommendation is to use the technical channel for asking technical questions and the business channel for questions related to business problems.

When a new workspace is created, TBC needs to install a Slack App into the workspace and upload its tokens to the Slack Scanner server. Then the server can use tokens to hit Slack endpoint and get the information of participants and messages.

When searching for messages in Slack workspaces, it is better to provide both participants data and message data, so for every message, the server will combine the participants’ information and message information into documents and store them to MongoDB. One document in MongoDB's collection contains the data of one message in slack, it has the hackathon name, channel name, timestamp, text and the participant’s email.  The search function provides multiple filters, and the server will use MongoDB’s ad-hoc query to filter messages that fulfill user’s search requirements and send them back to the user.

## GitHub Scanner

Currently, TBC records data for hackathons and solutions using excel and GitHub which are not an efficient way for them to organize and analyze the data that have generated in TBC’s platform.

So, we create a GitHub Scanner to help TBC capture data from every hackathons’ solutions in the GitHub repositories. As mentioned in the reorganize section, for every hackathon solution, we suggest attaching a structured README file which contains certain headers like “project name”, “problem type”, “project industry”, “technologies used”, “participants”. These 5 sections are the required sections that every README file should have and provide a basic overview of every solution. And we also provide spaces for some customized fields for contributors to fill in other information about the project such as “Descriptions” and “Achievements”. All these headers would be later served as the key parameters in the search query to look up for certain projects. These add and search functions would help provide insights about the successful patterns for blockchain projects based on the industry and problem type.

In addition to the information retrieved from the README file, the tool would also scan every script submitted in the repository and check whether it is in Java, Python or other programming languages. And then we would count the number of files for each programming language and save the data into the file\_extention category. This extra information could provide TBC with a lookout for what is the most used language for specific problems.

## Combine Services (DEMO)

We also created a sample implementation that combines the 2 services into a single endpoint that can help TBC search for Technology used and questions asked along with different problem types and industry slices.

This demonstration service also includes a front end that shows off the main services that we have created and, most importantly, contains a dashboard that we believe TBC can use moving forward to monitor high-level trends within their network.

## Frontend

Our frontend implementation was created to show off the capabilities of our services. In this report, we will cover the basic views of our frontend.

### Dashboard

## 

Figure 2 Dashboard Screenshot

Figure 2 shows the main dashboard of our application. This dashboard has seven cards:

1. Number of Technologies
2. Number of Projects
3. Number of Users
4. Pass Hackathons
5. Top 5 Industries
6. Top 5 Technologies
7. Top 5 Problems

This page shows a high-level overview of the current state of TBC and the trends within the platform

### GitHub Scanner

### 

Figure 3 GitHub Scanner Frontend

This is the frontend that calls the search function of the GitHub Scanner mentioned in the retool section. From Figure 3, we can see the search values accepted by the service. The two interesting points to take note of is the Hackathon Name drop list and the Header and Data field.

#### Hackathon Name

This drop list will restrict users to select from only hackathons that exists within the system. This makes the service more user friendly.

#### Header and Data Fields

These two fields come as a pair. As mentioned in the retooling section, the GitHub scanner can take in customer headers from the README file provided to it. Thus, in order to allow for searches on those fields, the Header and Data field was provided to allow users to search base on these custom headers.

If a README had a custom header of References, and TBC can input the search value “References” into the header text field and enter the value that they intend to search for in the Data field.

#### Results

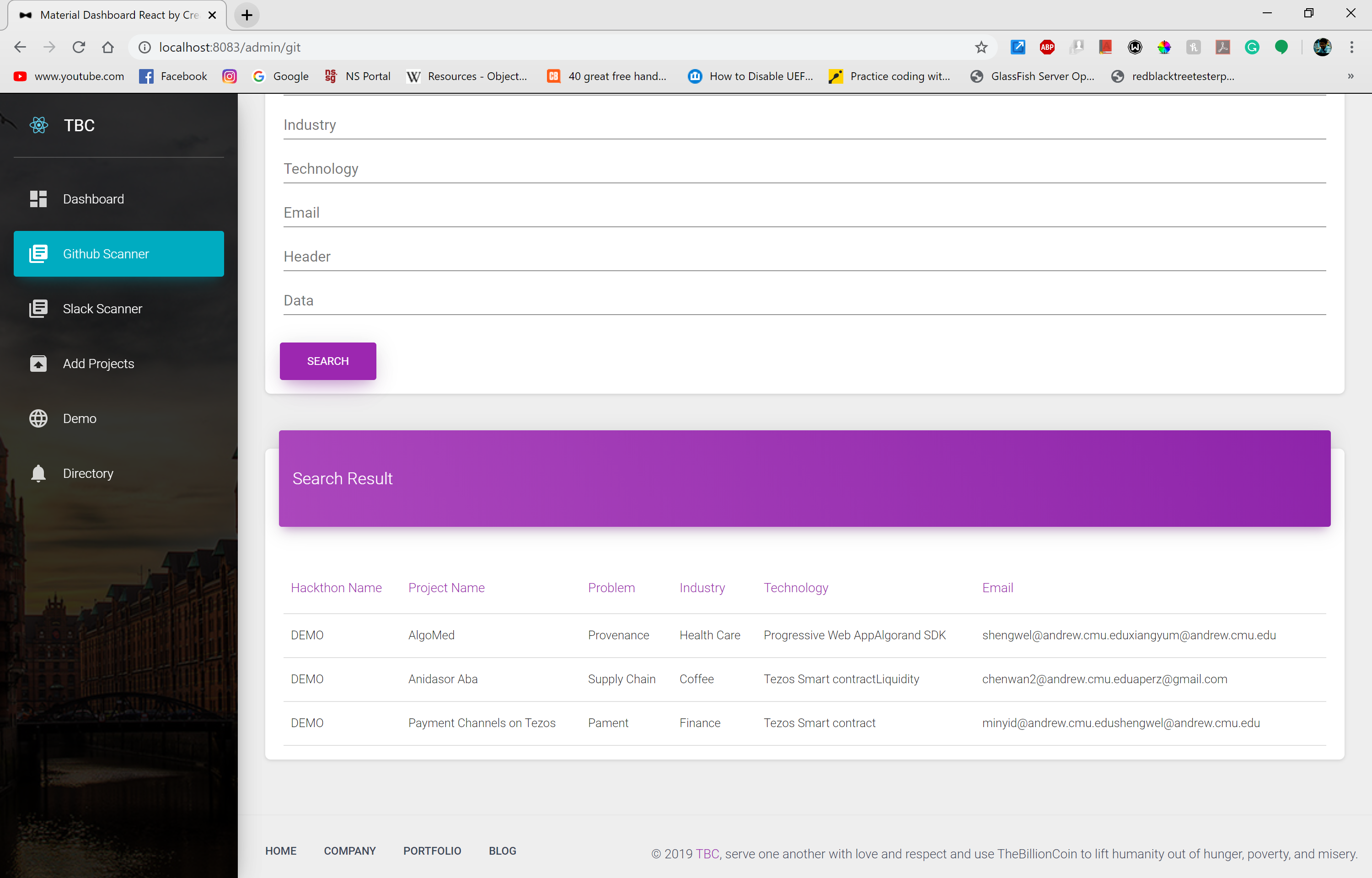


Figure 4 Search results for GitHub

Once a search is submitted, the results will be displayed in a card below the search card.

### Slack Scanner

### 

Figure 5 Slack Scanner frontend

The next service is the frontend that calls the Slack Scanner service mentioned in the retooling section. The search fields map directly to the slack scanner’s input fields. Take note of the date picker for the Start and End Date input. This allows users to filter results between a selected time period.

#### Results

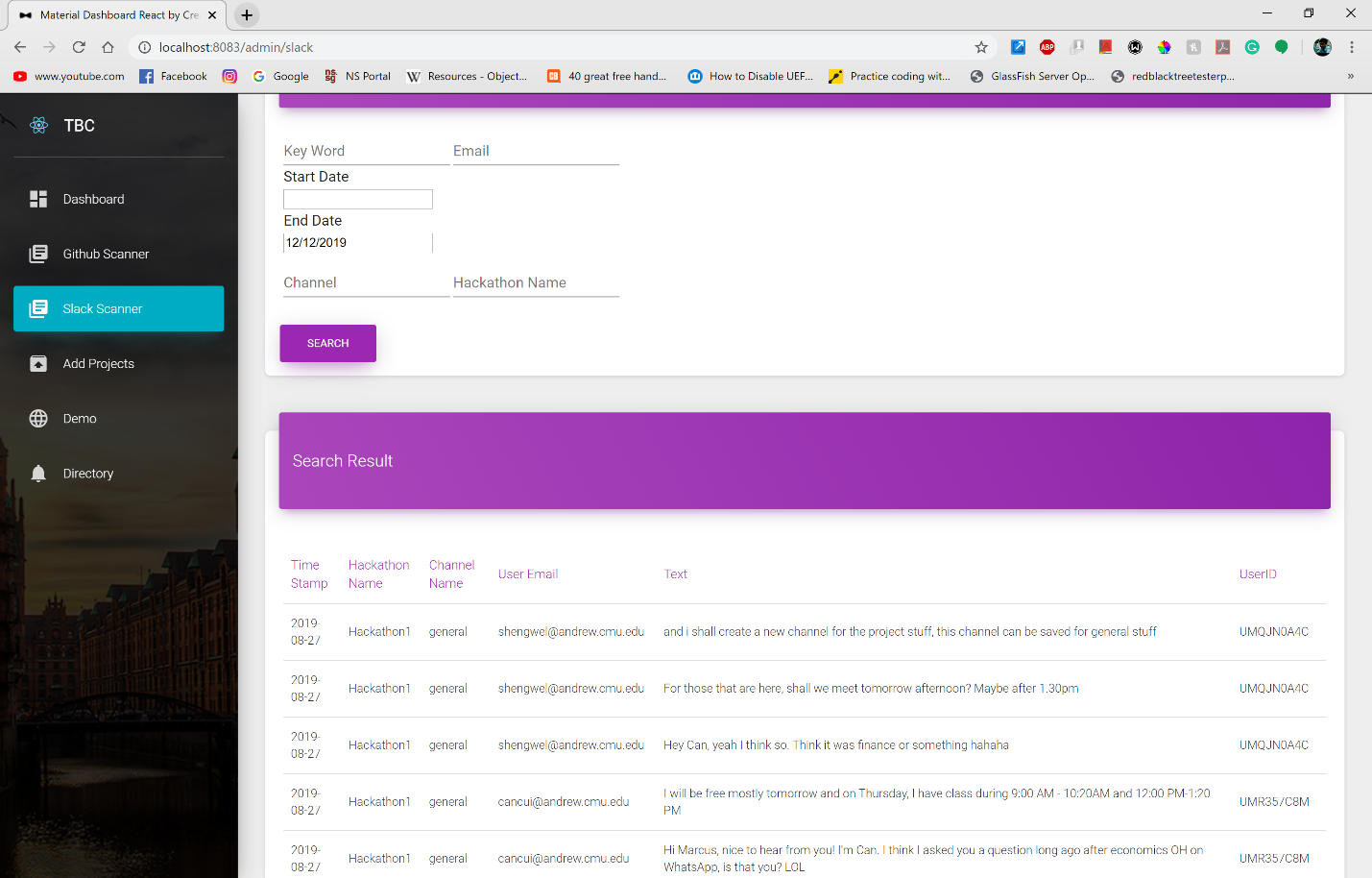


Figure 6 Search results for Slack

Like the GitHub frontend, the results will be displayed below the search fields.

### Add Project

### 

Figure 7 Add Project

This allows TBC to upload hackathon solutions to the GitHub scanner service using a csv file.

#### csv format

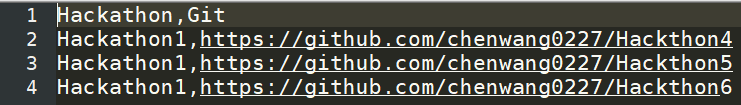


Figure 8 Project csv format

From Figure 8, we see the simple csv that is used to add solution repositories to the GitHub scanner. This file just consists of the Hackathon that a project was done in and its GitHub the repository link.

### DEMO

### 

Figure 9 Demo service frontend

This is the frontend implementation of the Demo service. This service combines the results of both the GitHub and Slack Scanner services to generate insights on projects done within TBC. The search returns results base on a given industry and problem type. For the results below, we searched for projects within the **Health care** industry solving **Provenance** problems.

#### Results

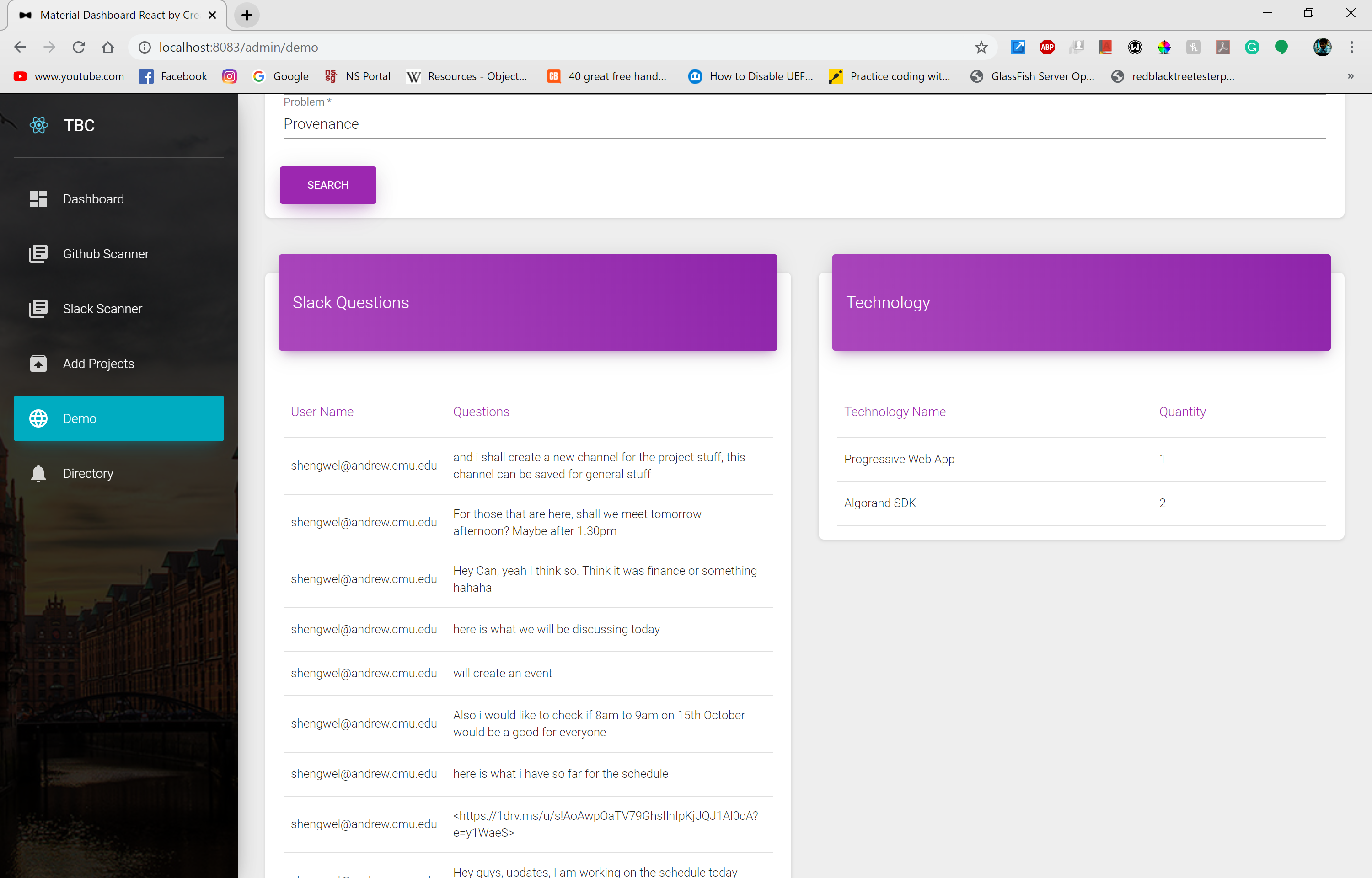


Figure 10 Demo search results

#### Slack Questions (Left Panel)

This panel contains information on users who have attempted a given project and the questions they asked while attempting the project. This will help TBC summarize the possible questions that one might ask when attempting to solve provenance problems in the healthcare industry.

Also, with the email of users who attempted these problems, TBC can direct companies who are trying to solve similar problems towards users who have attempted to solve such problems.

#### Technology (Right Panel)

The right panel shows the technologies used to try and solve such problems during TBC’s hackathons. What this brings to TBC is the ability to summarize the technologies popularly used to solve such problems.

On top of this, TBC can direct companies coming to them for advice on such projects to the right technology platforms for assistance.

### Directory

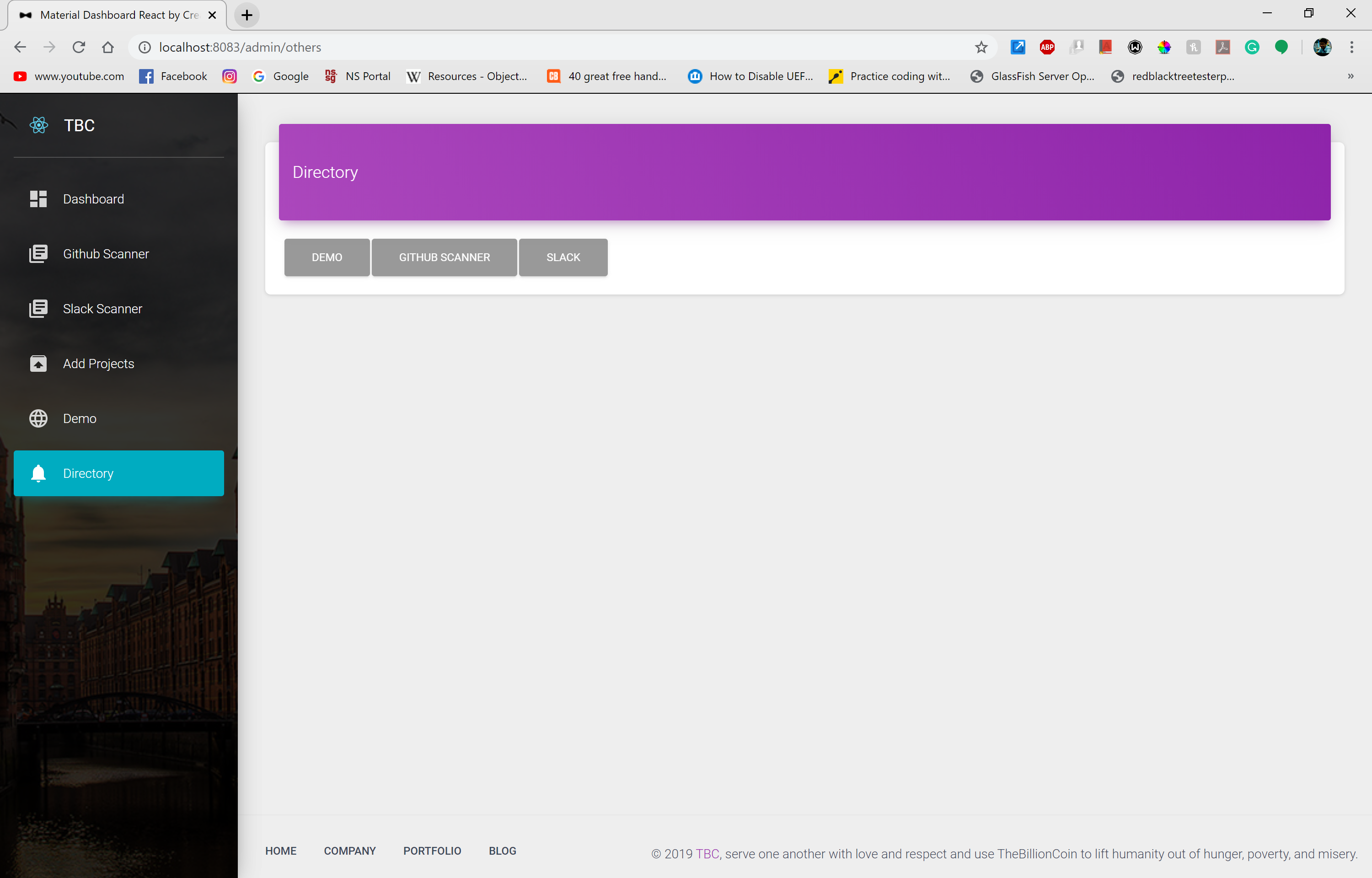


Figure 11 Directory page

This page links users back to the service endpoints.

## Deployment

These 3 services are deployed using Docker Containers with and added container for the frontend used for our project demonstration. At the request of TBC, we have chosen this technology as they were most familiar with Docker and they wanted to be able to redeploy the services in a platform-agnostic manner. Thus, Docker was the best tool for this deployment requirements.

# Impacts

We set out to answer the question: With an unstructured platform, how can TBC gather valuable information that could help them curate high-value interactions amongst their user groups?

The solution that we have provided in response to this problem statement fall under 2 main technical categories:

### Data Ingestion

We provided TBC a way to structure their data such that they can more easily access it when needed for analysis.

### Data Exploration

The services combined with the frontend representation of these services provide an easy way for TBC to explore the data within their system. We would like to bring special attention to the dashboard for this case. The dashboard will be useful moving forward as TBC can use it to find specific areas in which to further expand its data ingestion process.

As we mentioned in the retooling section, under the DEMO frontend, we see the value that such a service can bring to TBC. With a simple combined service that we created; we see that with a single search we can answer questions like:

* Given that my company is in the healthcare industry, what technology is commonly used to solve provenance problems and what issues should I expect to face in the process of executing this project?
* Who can I approach for advice on advice when solving this type of problem?
* Who has the technological expertise to advise me on what is possible and what is not?

It is our opinion that such a pipeline will only grow in value as we collect more information from future hackathons. Thus, let us explore the areas where this project can grow.

# Future Improvements

This project is a proof of concept that shows the value that having a well-structured data ingestion pipeline can benefit TBC in its plans to move towards curating higher value interactions between their user groups. Moving forward, we hope TBC can build upon these foundations to create more robust information extraction services. To help guide these future improvements, here are some of the steps we recommend as next steps from this project.

## Fully applying the Reorganization phase

This phase is part of our project but has yet been fully implemented. Within the limited four-month time span of this project, TBC has yet to have the time or manpower to reorganize all their data to match the new standards.

### Suggestions

Too easy in this transition, we recommend that TBC focus on future hackathons rather than trying to reorganize their current datasets. Future hackathons should follow these new data structures so that TBC can use the available services we provide as a good testing ground for the viability of our product in their workflow.

What TBC should look out for is the information on the dashboards. Focus on any interesting data points that seem to stand out or differ from what is expected.

## Adding in more channels of communication

Currently, the services provided are focused on GitHub and Slack as sources for their information. However, we understand that TBC has other channels like GitLab and Discord that they should also capture within this pipeline.

### Suggestion

We suggest that TBC does this after reorganizing the data from the current target channels. They can then use the lessons learned from these channels to better implement their reorganization of the news channels.

## Questions identification

Another area to work on moving forward is question identification. Our team concluded that with the data set available currently, question labeling would not be a very valuable tool for TBC’s current needs. However, moving forward, as more data is gathered, TBC might want to implement some form of question tagging within their Slack technology and business channels.

This will help to filter out questions from the answers and also could lead to a future implementation for Artificial Intelligence tools that could streamline the data processing process on Slack. Helping them to extract out questions more quickly and consolidate them for other parties to access.

### Suggestions

A simple implementation of this idea would be to get users to tag all questions with a #question within the message body. This can then be retrieved through our service under a keyword search.

## Education

The previous improvement will build the foundations of this next improvement step. Education is another aspect that was discussed during the course of this project. TBC is interested in creating a more inclusive platform for blockchain people of not only technical background but also non-technical backgrounds.

Thus, non-technical education around blockchain is another aspect that TBC has shown interest in. We believe that if TBC wants to grow this area of its platform, they should invest time in developing this question labeling improvement mentioned above.

# Conclusion

In sum, our project’s objective was to create an overall pipeline for TBC, from data ingestion to exploration with the specific target of helping TBC find the information that can help them curate higher-value interactions. What we have delivered is a Proof of Concept project that has well-defined data reorganization steps to help TBC achieve its goals and added services to start the data exploration process for TBC. Moving forward, we hope that TBC can build off the future suggestions that we have recommended to fully utilize the potential that its data can provide.

# References

|  |  |
| --- | --- |
| [1] | O. Kharif, "Blockchain, Once Seen as a Corporate Cure-All, Suffers Slowdown," Bloomberg, 31 July 2018. [Online]. Available: https://www.bloomberg.com/news/articles/2018-07-31/blockchain-once-seen-as-a-corporate-cure-all-suffers-slowdown. |
| [2] | Market Study Report LLC, "Blockchain Market Size Analytical Overview, Demand, Trends and Forecast to 2024," Market Watch, 5 April 2019. [Online]. Available: https://www.marketwatch.com/press-release/blockchain-market-size-analytical-overview-demand-trends-and-forecast-to-2024-2019-04-05. |
| [3] | M. Smith, "In Wake of Romaine E. coli Scare, Walmart Deploys Blockchain to Track Leafy Greens," Walmart, [Online]. Available: https://corporate.walmart.com/newsroom/2018/09/24/in-wake-of-romaine-e-coli-scare-walmart-deploys-blockchain-to-track-leafy-greens. |
| [4] | Hyperledger, "How Walmart brought unprecedented transparency to the food supply chain with Hyperledger Fabric," Hyperledger, [Online]. Available: https://www.hyperledger.org/resources/publications/walmart-case-study. |
| [5] | S. Castellanos, "FedEx CIO Looks to Industry Collaboration to Scale Blockchain," The Wall Street Journal, 29 April 2019. [Online]. Available: https://www.wsj.com/articles/fedex-cio-looks-to-industry-collaboration-to-scale-blockchain-11556572820. |
| [6] | Medicalchain, "Medicalchain Whitepaper 2.1," Medicalchain, [Online]. Available: https://medicalchain.com/Medicalchain-Whitepaper-EN.pdf. |
| [7] | C. Roldan, "Hippocrates," Devpost, [Online]. Available: https://devpost.com/software/hippocrates-ldwftv. |
| [8] | A. Litan, D. Groombridge, C. Healey and A. Leow, "Blockchain Unraveled: Determining Its Suitability for Your Organization," Gartner, Stamford, 2019. |

**Appendix: User Manual**

**Slack Scanner**:

**API Provided:**

* **Update Token (Post)**

Method Description:

User can use update token function to add new tokens or replace the old one. Server uses hackathonName as the primary key, when token pair are uploaded, server will check whether this hackathonName exist in the system, if it is a new name, just add it to the token set. If the hackathonName exist in the system, server will replace the old pair.

URL: <http://18.220.249.85:8081/slack_analyser/api/updateToken>

Input format:

* + Headers: "Content-Type": "application/json"
  + Body:

{

"hackathonName":String,

"oAuthToken":String,

"botToken":String

}

Response: “OK”

* **Get All Token (Get)**

Method Description:

User can use get all tokens function to view all tokens exist in the system. Server will reply a json array to show all the token pairs and also the validity of them. If the token is not valid, it may be expired or wrong, user can use update token method to update new token pair.

URL: <http://18.220.249.85:8081/slack_analyser/api/getAllTokens>

Response:

[

{

"hackathonName":String,

"Valid":"true",

"OAuthToken":String,

"botToken":String

},

{

"hackathonName":String,

"Valid":"false",

"OAuthToken":String,

"botToken":String

},

...

]

* **Update new messages to MongoDB (Get)**

Method Description:

When user send a request to update new messages, server will first use tokens in the system to update all slack workspace one by one. For each workspace, first it gets the newest timestamp in mongoDB collection and use it to call Slack API and get the new messages. Server will also call another Slack API to get user information and combined it with the message, then write it into one document in the mongo collection.

URL: [http://18.220.249.85:8081](http://18.220.249.85:8081/slack_analyser/api/getAllTokens)/[slack\_analyser/api/update](http://18.223.102.2:8080/slack_analyser/api/update)

Input: null

Response：200“OK”

* **Search messages (Get)**

Method Description:

User can search messages by using this API call, and we provide the search by hackathon name, channel name, user’s email, keyword, time period. When this api is called, the server will connect to the mongoDB and use mongoDB’s ad-hoc query to filter the data and send back a json array contains all the messages which fulfil the requirement to the user.

URL: [http://18.220.249.85:8081/slack\_analyser/api/search](http://18.220.249.85:8081/slack_analyser/api/getAllTokens)

Input format:

Parameters in the URL (these parameters can be null):

{

         "keyword": String, // keyword the message contains

         "email": String,      // email for the user

         "start": String,   // a long number of the oldest message’s timestamp

         "end": String  // a long number of the newest message’s timestamp

         "channel": String, // channel name

"hackathonName": String, // hackathon name

}

Example:

URL: [http://18.220.249.85:8081/slack\_analyser/api/search](http://18.220.249.85:8081/slack_analyser/api/getAllTokens)?keyword=today&email=xiangyum@andrew.cmu.edu

Output:

[

    {

        "\_id": "d76d0998-dbdf-4c56-9f22-aba374ed0249",

        "timeStamp": "1570024872.000100",

        "hackathonName": "ISM Project",

        "channelName": "capstone",

        "userEmail": "xiangyum@andrew.cmu.edu",

        "text": "Room for today's meeting",

        "userID": "UMR3577EZ"

    },

    {

        "\_id": "0d9aa9a3-df4f-4c2e-bb7a-6ef98bb8196d",

        "timeStamp": "1568120700.000700",

        "hackathonName": "ISM Project",

        "channelName": "capstone",

        "userEmail": "xiangyum@andrew.cmu.edu",

        "text": "Booked interview room A007C for today's meeting",

        "userID": "UMR3577EZ"

    }

]

GitHub Scanner:

Based on the structured README file, we developed a GitHub scanner service that provides three main RESTful APIs for TBC to manage the hackathon solutions’ data into a cloud-based MongoDB.

* **Add hackathons and projects data into MongoDB**

Method Description:

This function would take a list of all solutions’ GitHub repository links for every launched

hackathon, and save the data into MongoDB.

URL:<http://18.217.114.101:8080/github_scanner/api/add>

Type: POST

Input:

* Headers: "Content-Type": "application/json"
* Body: A hackthon\_name, with an array of String which contains all solutions’ GitHub repositories links. In JSON format.

 Example:

 {

   "hackthon\_name": "Hackthon1",

   "gitRepos": ["<https://github.com/chenwang0227/Hackthon1>"]

    }

Response: 200“OK”

* **Search specific hackathon projects**

Method Description:

This function would take a list of key parameters based on certain criteria

to look up for specific projects from MongoDB.

URL:<http://18.217.114.101:8080/github_scanner/api/search>

Type: POST

Input:

* Headers: "Content-Type": "application/json"
* Body:

Example: these parameters can be null

{

         "hackthon\_name": List<String>,

         "project\_name": String,

         "problem": String,

         "industry": String,

         "technology": String,

         "email": String,

         "header": String,

         "data": String

}

Response: a Json array of projects which fulfill the requirements of the search criteria

Example:

[

{

     "\_id": {

            "timestamp": 1573434205,

            "machineIdentifier": 5113427,

            "processIdentifier": 5502,

            "counter": 3491831,

         "time": 1573434205000,

         "date": "2019-11-11T01:03:25.000+0000",

            "timeSecond": 1573434205

     },

        "hackthon\_name": "Hackthon1",

        "project\_name": "AlgoMed",

     "problem": [

            "Provenance"

     ],

     "industry": [

         "Health Care"

     ],

     "technology": [

         "Progressive Web App",

         "Algorand SDK"

     ],

     "email": [

            "jhons@andrew.cmu.edu",

            "ellenw@andrew.cmu.edu"

     ],

        "Descriptions": "The AlgoMed – Computerized Pressure Algometer is the first software-based computerized Algometer to offer real-time visual & auditory feedback to control & monitor applied pressure rates.<br/>Algometers are designed to quantify and document levels of tenderness via pressure threshold measurement and pain sensitivity via pain tolerance measurement. Pressure algometry is a reliable measure of pain in muscle, joints, tendons, and ligaments.<br/>The challenge in performing algometry tests is to apply continuous pressure at a constant rate on the patient selected body site without having the possibility to monitor it in real time.",

        "Achievements": "Finally build a solution to solve.."

}

]

* **Get all launched hackathons’ names**

Method Description:

This function would return all launched hackathons’ names that are already stored in the database.

URL:<http://18.217.114.101:8080/github_scanner/api/all>

Type: GET

Response: An array of all hackathons’ names in JSON format.

Example:

[

"Hackthon1",

"Hackthon2",

"Hackthon3",

"Hackthon4"

]

* **Reset data for a specific hackathon**

Method Description:

This function would delete all information for a specific hackathon from database.

URL:<http://localhost:8080/github_scanner/api/reset>Type: POST

Input:

* Headers: "Content-Type": "application/json"
* Body: the name of the hackathon that needed to be reset, In JSON format.

{

"hackthon\_name" : "Hackthon1"

}

Response: 200“OK”