Cold Chain Mapper

Yin Yin Low
University of Washington
Seattle, Washington
yinlow63@cs.washington.edu

Aeron Langford
University of Washington
Seattle, Washington
aeronl2@cs.washington.edu

Andrew Siew
University of Washington
Seattle, Washington
aks42@cs.washington.edu

Alexandra Ash
University of Washington
Seattle, Washington
asha3@cs.washington.edu

Shourya Jain
University of Washington
Seattle, Washington
shourj@cs.washington.edu

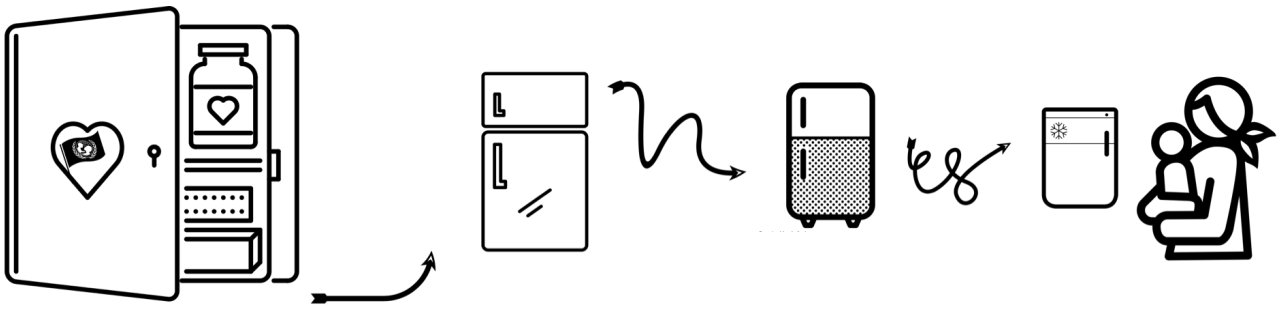


Figure 1: The vaccine cold chain.

ABSTRACT

The global distribution of vaccines is a complex supply chain and resource allocation problem that can hinder disease eradication efforts if handled incorrectly. Stakeholders such as the WHO, UNICEF and various non-governmental organizations manage these cold chains which differ dramatically from country to country. With the CCEI data model, a new standard in defining cold chain inventory, we present an online web-based dashboard that allows for the visualization of cold chain data. Specifically, we focus on the problem of equipment upgrades. Our solution, developed in collaboration with PATH, provides users an interactive map in which they can select different countries and compare cold chain data. It also allows users to compare different administrative regions within a country and makes it straightforward to add new countries. Our dashboard is a proof of concept for what would be possible with a global database of standardized cold chain data. We hope to use it in the effort to push the global cold chain community to invest in collecting standardized and useful data, which can then be used to drive informed resource allocation decisions.

Team YAAAS, CSE 482 Capstone, Spring 2019

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

CCS CONCEPTS

• **Visualizations → Visualization theory, concepts and paradigms**.

KEYWORDS

Vaccine Cold Chain, CCEI, CCEM, Vaccination Data Mapper

ACM Reference Format:
Yin Yin Low, Aeron Langford, Andrew Siew, Alexandra Ash, and Shourya Jain. 2019. Cold Chain Mapper. In *Proceedings of Team YAAAS*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

1.1 Problem Overview

1.1.1 Vaccine Impact. Vaccines are crucial for the improvement of global health. Studies have shown that childhood mortality rates have decreased significantly due to the distribution of vaccines. The high impact of vaccines can be demonstrated through the World Health Assembly's commitment to eradicate polio. The World Health Assembly is the decision making body of the World Health Organization (WHO). Since the commitment was made, there has been a 99.9% drop in the number of polio cases worldwide [1]. Through the use of vaccinations and other preventative measures these levels have been attained, and are strategies which can be widely adopted throughout the world for many different diseases. According to the Global Polio Eradication Initiative, as of June 2019 there are only three polio endemic countries remaining: Afghanistan, Nigeria, and Pakistan. [2].

The above exhibits how the widespread distribution of vaccines can help to protect populations from polio, and can be generalized to apply to many other diseases that would otherwise be disrupting many nations. For example, pneumonia, caused by pneumococcus, and severe diarrhoea, often caused by rotavirus, are leading factors in child mortality for developing countries. Therefore the development and distribution of effective vaccines against pneumococcus and rotavirus would likely result in a major reduction in the global child mortality rate [1]. It is of high importance to invest in the vaccines in developing nations in order to improve the welfare of the world population and help eradicate diseases that would otherwise prove to be deadly.

1.1.2 Cold Chain Structure. As vaccines are such a high-impact commodity, there are many stakeholders which fund efforts to increase efficiency of vaccine distribution. They can be divided into the following groups:

(1) Global Stakeholders: United Nations International Children's Emergency Fund (UNICEF), World Health Organization (WHO) and Gavi, a vaccine alliance
(2) Donors: Bill & Melinda Gates Foundation, United States Agency for International Development (USAID) and Department for International Development (DFID)
(3) Non-governmental Organizations: Program for Appropriate Technology in Health (PATH), Cooperative for Assistance and Relief Everywhere (CARE) and John Snow Inc. (JSI)

With the assistance of the above stakeholders, developing countries must first attain the vaccines from the manufacturers or other suppliers, distribute the vaccines effectively and also make sure to maintain the storage equipment that is required to keep the vaccines from spoiling so they can be distributed to their population. Once the vaccines have reached the national vaccine stores from the global suppliers, they are shipped to regional stores, district stores, health centers and health posts throughout the country. These different facilities serve to both store and administer the vaccines, though the majority of the vaccines occur at the end of the chain at the health centers and health posts.

Throughout the whole transportation and storage process it is essential that the vaccines are kept at a specific range of temperatures, which is why the process of getting the vaccines from the manufacturers to the health facilities that administer them and detailed above is called the cold chain.

1.1.3 Cold Chain Challenges. There are a wide range of challenges in transporting and storing vaccines. Vaccines are sensitive to heat, so they must be stored at a constant temperature, generally around 2° to 8° Celsius [3]. A vaccine's ability to inoculate a user decreases on exposure to temperatures outside of the specified range. Once this ability has been lost, it cannot be regained and the vaccines subsequently must be disposed of since they are no longer viable. Thus, to ensure a high quality of vaccines, they must be stored in reliable refrigerators or coldrooms. These refrigerators must be specifically built for vaccines; domestic refrigerators allow temperatures to vary widely, unsuitable for the vaccine's requirements. Depending on the most reliable energy source in the location, these refrigerators are powered mainly by electricity, gas, kerosene, or solar energy.

Health facilities generally carry a few months of vaccines at a single time, so if a refrigerator were to break or run out of energy, a significant amount of vaccines could spoil. Refrigerator maintenance can take also take an unpredictably long time which can cause the vaccines to spoil before the necessary maintenance can be completed. This can result in a stockout, which occurs when a health facility runs out of a vaccine they usually carry and are no longer able to administer the vaccine until a resupply occurs. This is especially detrimental in the case when populations travel large distances for vaccination.

Stakeholders would like prevent stockouts and ensure health centers and other medical facilities always have sufficient amounts of vaccines for the population they serve. This is a challenge because data about vaccine stores become obsolete quickly and is not collected frequently. Sending someone to count vaccine stores and then disseminate the data to both the leadership of the facility and more global stakeholders is both time consuming and costly. Furthermore when this data is collected, it is not in a standardized format, which is fine for the local facilities but makes it difficult for global stakeholders to understand the state of the facilities when they must parse a wide range of data formats.

These problems all force funding decisions to be less data-driven then one would hope, and leaves much room for improvement in terms of resource allocation and understanding the condition of the global cold chain.

1.2 Solution

After learning about the vaccine cold chain, we saw a clear need for a global standardized data format that would allow stakeholders to back their decisions with concrete data. We decided to create a proof of concept web dashboard to explore the possible uses of such a standardized data model, and how it could potentially improve decision making to increase the efficiency of resource allocation. This dashboard is based on the Cold Chain Equipment Inventory (CCEI) data model for cold chain inventory, since this model creates a format that ensures that data can be queried, regularly updated and be used for consistent and informative visualizations. CCEI was created in collaboration with the WHO, UNICEF, and other non-governmental cold chain experts, and is the new, accepted data format for cold chain inventory data.

1.2.1 Interactive Map. Since the stakeholders interact with and manage the cold chains of many different countries, we wanted to provide them with an interactive map in which a user can click on different countries and compare cold chain data outside of a spreadsheet. The map also allows a user to explore different administrative levels in a country and view data specific to each level. This can provide the user the ability to compare the number of working refrigerators and coldrooms in different regions along a cold chain path.

1.2.2 Data Visualizations. The data visualizations are the crux of our dashboard. Since we had a limited amount of time to develop our solution, we wanted to focus on high-impact data visualizations. In consultation with PATH, we decided to tackle the challenge of visualizing equipment status in order to see where equipment upgrades and maintenance is needed. Cold chain facilities may have

a large number of refrigerators, but often only a percentage of those refrigerators are fully functional. The rest may require maintenance or simply need to be retired. Another important aspect of upgrades is evaluating the equipment's power source. Older refrigerators might still run on kerosene or gasoline and need to be switched in favor of more consistent of efficient fuels. On the other hand, facilities that function in areas with low or poor access to electricity might need equipment that runs on alternative power sources that are optimized to their region and the available resources.

1.2.3 Adding new countries. Since we want our stakeholders to have access to a global dashboard, it was important for us to add a feature that made it easy to add new country data. Our import feature allows for quick input of new countries and all of its corresponding cold chain data given that it is in the CCEI data model. If the data is unclean, or otherwise doesn't strictly follow the CCEI model, we provide users with clear error messages so that they can fix the data to work with our dashboard.

1.2.4 Compatibility with existing data. We recognize that many countries do not currently collect cold chain data in the CCEI format. The prevalent tool currently used to manage this data is the Cold Chain Equipment Manager (CCEM) application which uses Microsoft Access to store the data and provide generic reports. Although this application allows for arbitrary columns and is far from being standardized, we were able to create a converter that manipulates CCEM data to the CCEI data model, which can then be used in our dashboard for visualizations.

2 TOOLS AND INTERFACES

Technology choices were made with the goal of completing our proposed solution within a ten week time frame. This means we chose tools based on our needs as well as the existing skill-sets of the team members.

2.1 Front End

With the goal to improve data-driven decision making, we created our web application for a global cold chain dashboard with Node.js as the framework. We used EJS to manage our HTML/CSS code since we utilized many common components and EJS reduces redundancy. Also, in order to have better analysis on important statistics, we developed our interactive map with the Google Maps API and used D3.js, a Javascript visualization library. Lastly, we used Bootstrap to style the dashboard and facilitate the dynamic responses of our application.

Map View To visualize the boundaries for each country on our interactive map, we use user-provided shapefiles. We made this design decision since GIS coordinate data for cold chain facilities is not a required component of the CCEI data model and is therefore frequently absent, or if it exists, incorrect or malformed. Additionally, shapefile data is not consistent across different sources and this gives the user power over how they want to visualize different country boundaries. Often different shapefiles also significantly differ at lower administrative regions.

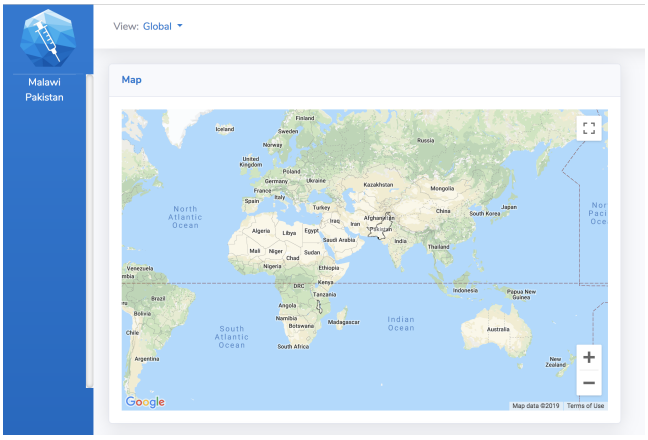


Figure 2: Global View.

In the global view, we display a world map consisting of countries available in our database. With the help of the Google Maps API and the Google Places API, we implemented highly accurate base maps which have standard zooming and scrolling capabilities. In addition, the Google Maps API allows for the layering of additional spatial data and the Google Places API allows for the geolocating of particular regions.

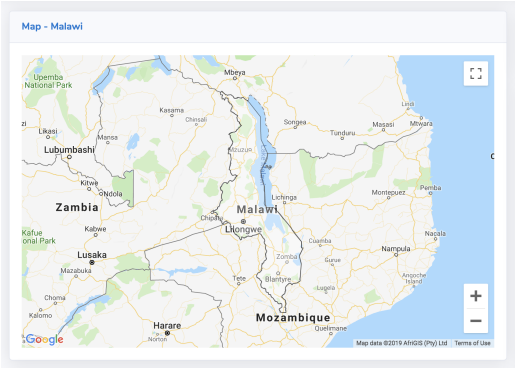


Figure 3: Country View.

We have also provided our users with the ability to select different administrative regions within a country. This allows them to see high level views of a country's cold chain data as well as highly specific low level views.

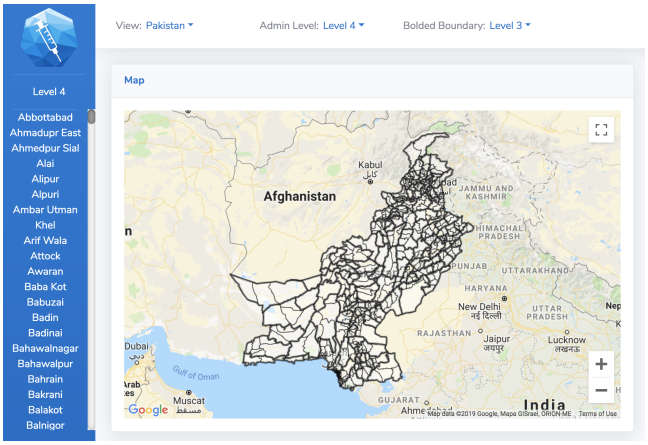


Figure 4: Bolded Boundary View.

After communicating with a PATH representative, we learned that most store managers and officers are only responsible for certain regions. With the implementation of a bolded boundary feature, those interested in particular sub-regions can easily distinguish districts within a higher level region easily. This will also ease of experience for users interested in visualizing large amounts of low level data in the context of higher administrative levels.

Utility Count As we discussed earlier, the most important factor to consider when making decisions regarding equipment upgrades is to know the utility count of the regions or countries selected. Thus, our dashboard displays counts of the total number of refrigerators and cold-rooms in the selected bounded region.

Figure 5: Utility Count.

Power Source and Storage Utilization

With the help of the D3 Javascript Data Visualization library, we developed two interactive donut charts to present available statistics of Power Source as well as Storage Utilization. These are both highly important statistics when considering decisions regarding equipment upgrades, and viewing this data in an understandable "Donut" format greatly increases ease of analysis. Note that if we do not have the data needed to visualize a statistic, we display it as "undefined", which gives the user a chance to edit their data and try again.

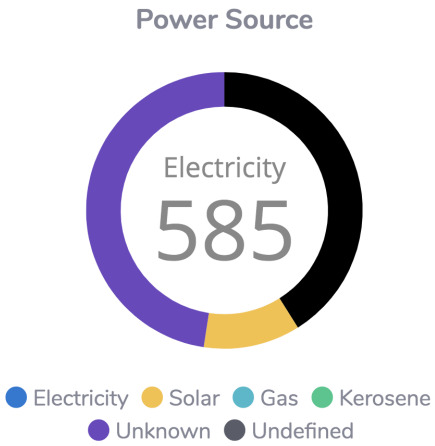


Figure 6: Power Source View.

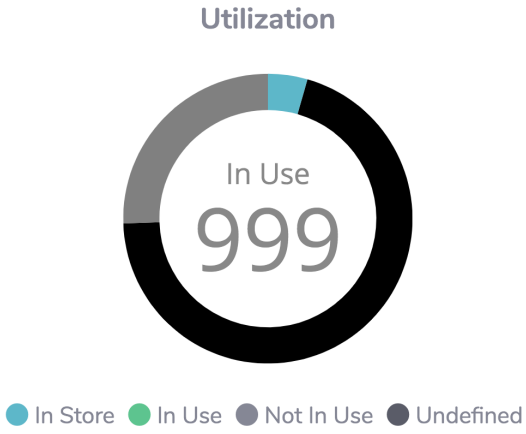


Figure 7: Storage Utilization View.

By utilizing D3.js's bar chart, our dashboard shows the combination of the power source and storage utilization distribution of the corresponding region. For instance, in the following figure, out of the 678 undefined type refrigerators, 201 are not in use. Stakeholders can use this chart to draw conclusions regarding the efficiency of different types of refrigerators in a region. Perhaps solar-powered refrigerators do not work well in some regions, or most of the kerosene refrigerators are old and not in use. This detailed analysis can inform specific decisions on equipment upgrades and equipment purchasing.

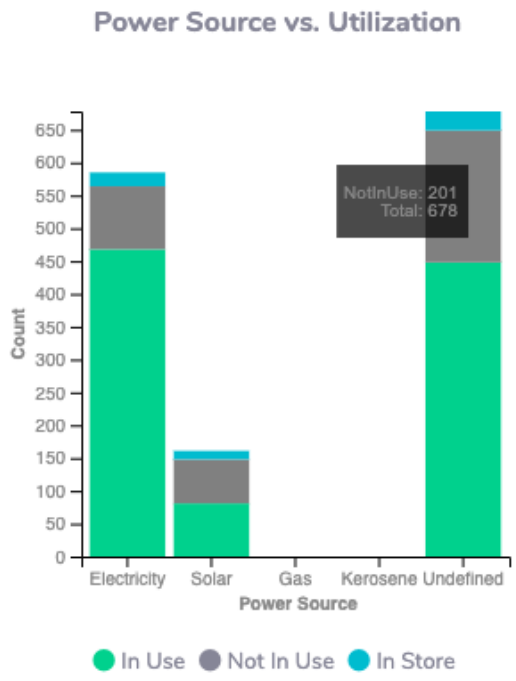


Figure 8: Storage Utilization of corresponding Power Source View.

Facility View

Our dashboard can also display the list of facilities available in the selected region. This was a feature that was explicitly asked for by our PATH representative since its provides district and facility level managers the option to monitor the status of the facilities under their charge. This includes facility identification information as well as refrigerator data.

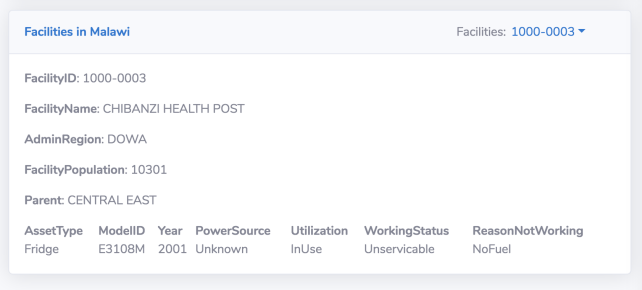


Figure 9: Facility View.

2.2 Back End

The back end consists of parsing the data, error checking the data, uploading data and retrieving data.

Parsing + Uploading Data

We use the "panda" and "csv" Python packages to parse through the data. As we parse through the data, we conduct heavy error checking to ensure that the data is clean and ready to upload to the Azure database. If any errors are found, they are simplified for the user and propagated back to the front end. The user can then use the error prompts to fix the data and try to re-upload the data. For the data to successfully pass through the error checker, it must strictly adhere to the CCEI data model. We then use the "pyodbc" python library to upload the data into the Azure database. This process of uploading is currently slow due to our use of multiple insert statements instead of a bulk insert. We hope to fix this feature in the future.

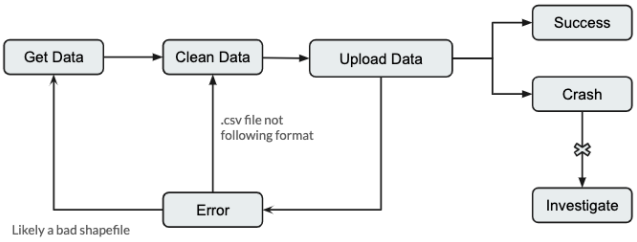


Figure 10: Importing Data Work Flow

The figure above displays the work flow of a user attempting to upload data into the database. The user would get data, clean the data and attempt to upload the data. The application may reject the data, so the user would need to go back and either get better data or clean the existing data. Once the data passes all tests, the application will attempt to upload it to the database. The upload may be successful or will fail unexpectedly. If it fails there was likely a case that was not considered in the checking process. Preventing these kinds of crashes can be be fixed through more tests to identify the issues, then expanding the checking code.

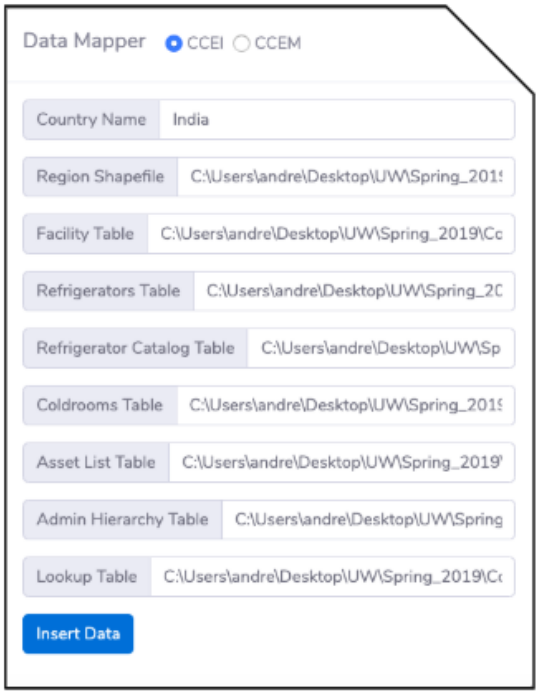


Figure 11: Inserting data

The user will have the option to use either input the data in a CCEI or CCEM format, by selecting through the radio button. Importing both types of data is a similar process involving the user inputting the direct paths to the necessary files along with the country name. The shape files are to be put in a folder and sorted with the top most level at the top.

Figure 12: Checking data

The application will start to check if the data is valid. Some of the cases that are checked are the uniqueness of primary keys and the existence of required data fields. This process takes some time, so a loading signal is displayed to indicate to the user that the process is still checking. In the future, more exhaustive tests will be implemented as the files are dependent on each other.

Figure 13: Error on upload

The application may reject data files and then return errors for the user to fix. The two errors above are for missing files or incorrect file path and for missing column name. There is a chance that the checking process passes invalid data as the tests are not exhaustive yet.

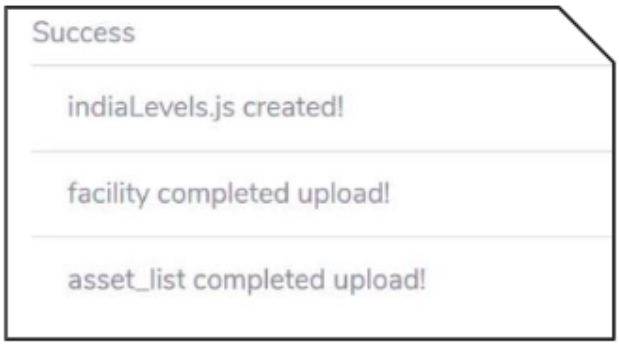


Figure 14: Successful upload

When an upload is successful, the feedback will indicate that each table got updated. In the event there is an unexpected crash, not all the table names will be reported back. Do note that the upload is an all or nothing process. This means that if there is ever a crash due to invalid data in any of the files, none of the changes to other tables will be saved.

Retrieving data

There is a web API that was coded with Node.js to make requests to the Azure database. The results are then sent to the front end to be visualized. To allow for easy retrieval of data from the Azure database, we wrote our web API in Node.js. The front end communicates with our API to get specific information from the database. Depending on the request, the API makes SQL queries to retrieve information about countries, facilities and calculate visualization statistics.

2.3 Packages

Here are some Python and JavaScript packages that we have been using to develop our cold chain dashboard:

- pyodbc
- csv
- panda
- numpy
- python-shell
- node.js
- D3.js

3 APPLICATION

The usage of this application is heavily dependant on the user providing clean data that is up to CCEI standards and fixing any errors reported back by the program. With clean data, the user will be able to see the visualizations.

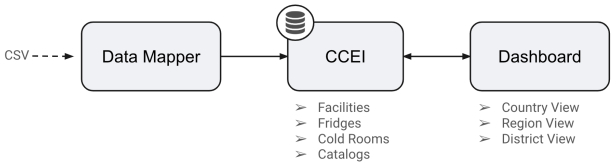


Figure 15: Architecture Outline.

3.1 Work Flow

A typical user would first get the necessary CSV files needed for the application to work. Most of the details of this can be referred to in Section 2.2. Although the CCEI format currently is relaxed on certain fields such as Power Source, it is important to still try to populate those sections. Without some of these 'non-required' data, the application's visualizations will be limited. Once data has been collected, the user will try to upload the data and probably notice there is something wrong with the data. The application will keep sending error messages to the user to fix certain areas in files. This keeps repeating until the data is clean. Common errors include duplicate primary keys and incorrect column names. Once this data cleaning cycle is complete, the data can be uploaded to the database. If the application fails to upload the database or does not show the data accurately, there must have been an error in one of the files. When this happens the it is up to the user to report the errors to the maintainers of the application. Continuous use of this application will improve the checking process. If everything is successful, the change will appear in the dashboard and the user will be able to interact with the application as described in section 2.1.

3.2 Scalability

While there are many aspects of the application that have been designed for scale, we predict that there might be some limiting factors that may hinder the ease of its scalability.

The visualizations scale well as users are able to compare countries and regions within countries. This process is easy and can be done with a few clicks. However, if many people will be utilizing the dashboard, the map may need to be reevaluated. As stated in section 2.1, we utilized Google Maps Platform with Google Maps API and Google Places Geocoding API. According to the number of page loads and requests, a small fee is placed on a Google Maps Cloud Platform Account. This was a trivial expense for our team as users receive a certain amount of free credits upon creating an account and our tests did not surpass the amount of free credits. If the number of users were to scale dramatically, we may want to reconsider the price for Google Maps, or utilize an open source mapping framework.

Another scalable feature is uploading data. As long as the data is clean, uploading the data is simple and works with as many countries. If the data isn't clean, the application will communicate the errors accurately so that user will know exactly what is wrong with the provided data.

Despite these features, there are some issues with the scalability of this application. Because this application is heavily dependent on having data follow the CCEI or CCEM format, it may be difficult to have 'good' data. Having clean data of all the facilities of a country can cause inconsistencies as many regions may input data that is either incorrect or not up to the CCEI/CCEM standard. Another issue is the ability to update an existing country's data, as the update feature has not been implemented yet. A crucial feature that we have also yet to address is security. The data that is inputted is very sensitive and there has yet to be a security system in place to limit the data that can be seen by unqualified users.

4 CODE STRUCTURE

The structure of the code is complicated and spans multiple files. Depending on whether the user is importing data or trying to view visualization the code works differently.

Most of the HTML code is stored in `views\index.ejs`. The JavaScript code which deals with events is mostly stored in `public\map.js`. The `db.js` file stores the credentials for any connection to the database via JavaScript. The `routes\index.js` file is the web API which most event handlers will call upon. Calling this will either use a python shell to run some python scripts when importing data or will immediately send queries to the database. Once the API returns data, the front end will present the data appropriately.

5 CHALLENGES

A large part of completing a proof of concept is identifying specific pain points within the problem domain. In the future, other designers and developers can be cognizant of these issues.

5.1 Unformatted Data

Because our application deals with many files with many different requirements, it is likely that the checking process is not comprehensive enough to catch all possible mistakes. Due to this, the application may crash trying to upload data and the user may not understand the cause. If it does not crash in the uploading process, the visualization process may fail as the data in the database might be 'dirty'. If any of these crashes were to occur, it will be extremely difficult to track and identify the source of the error. The application heavily relies on clean data and as the data set increases in size, it will be more challenging to have clean data. Being able to consider all possible errors will be a challenge and can be addressed best after extensive use of the application with various data sources.

5.2 Database Constraints

Uploading data can take a long time due to the way the uploading process is implemented. We are essentially adding many 'INSERT' statements for a query to upload data. A good alternative to fix this is to use a 'BULK INSERT'; however, this would require us to create a storage system within Azure, which was outside of the scope of the project. Due to the lack of that storage system in Azure we had to also use paths to locate the files instead of using an uploaded version of the file.

5.3 CCEM to CCEI

As mentioned earlier, the primary solution to collecting cold chain data is an application through Microsoft Access called the Cold

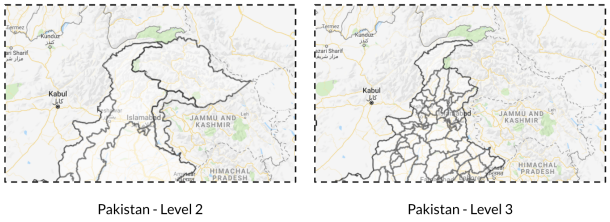
Chain Equipment Manager (CCEM) [4]. This application served as a means to gather information about the cold chain in a region and provide general reports off the inputted information. The user has the flexibility to add data rows as needed and generally customize the application as they see fit.

As this solution offers too much flexibility for our use, CCEI was developed to be a standardized data model. The changes that CCEI made from CCEM is requiring certain fields, providing a rigid data structure for the data and generally improving the uniformity of the data. While CCEM provided the user the ability to customize the application, this is problematic when scoping the application out to a more global level because it leaves the data in a wide range of formats. CCEI chose the essential data components of the cold chain to include and regularizes the format so utilizing the information in a global manner is more reasonable.

We were able to create a CCEM to CCEI conversion script that can take the existing CCEM data and parse it into CCEI format. This is not a perfect conversion because the data collected for CCEM versus CCEI is different, it provides many of the essential fields which gives us a means of visualizing more countries given the higher prevalence of CCEM data compared to CCEI.

5.4 Inconsistent Shape Files

The inconsistency of shape files has caused many complications. There are many sources for shapefiles, each of which being vastly different with regards to the same data. Depending on the source, the shapefile may not be up to date as some countries have different geopolitical regions depending on the year. Another possible reason for differing shapefiles is disputed regions. Some shapefiles include a disputed area as part of one country, while another will not. For example, in Pakistan, we have Northern Areas, where is a disputed area, in level 2 but it is unavailable in level 3.



Pakistan - Level 2 Pakistan - Level 3

Figure 16: Shapefile Inconsistencies.

Another problem with the shape files is being inconsistent with the provided data. The data file that deals with the region is the `admin_hierarchy` file in CCEI. Sometimes there are some regions that are not accounted for in either the data file or shape files. Due to this might we might receive inconsistent results from the database that show the number of facilities not matching up.

CCEI Database	Shapefile	Wikipedia
Balochistan	Baluchistan	Balochistan
GBaltistan	Northern Areas	Gilgit-Baltistan
Islamabad	Federal Capital Territory (FCT)	Islamabad/FCT

Figure 17: Naming Inconsistencies.

Sometimes the problem is more subtle as a single character difference such as the 'Baluchistan' not matching with 'Balochistan'. Other naming inconsistencies are significantly larger: 'GBalistan', 'Northern Areas', and 'Gilgit-Balistan' are all the same location despite having very different names. This is a difficult problem to solve with just additional script to check because each region may want certain sub-regions to be in certain levels but editing a shape file may be very difficult and time-intensive. We would need a centralized and credible database that provided up-to-date shape files for countries to base their data files of.

6 DESIGN DECISIONS AND COMPROMISES

6.1 The Lookup Table

When inserting a new country, users are required to provide the application with a `lookup.csv` file. This file is typically in a unique format with the top row having the column type and every other cell below is it's corresponding value. An example of this is having the top cell called PowerSource and the cells below it could be called Electricity, Gas, Kerosene, Solar and Unknown.

	P
1	PowerSource
2	Electricty
3	Gas
4	Kerosene
5	Solar
6	Unknown
7	

Figure 18: Lookup Table Example.

We are expecting that many countries will have different formats as some may include different values such as "Hybrid". Due to this particular issue, the application creates a lookup table for every country in the database. This creates lot of redundancy. Another issue with the format of the `lookup.csv` is that the top cells are linked to the values in the 2nd row of some of the csv files. An example of this is in `facility.csv` (Figure 18) where the second row has multiple cells with the values YesOrNo and the first row would be ClimateSuitable. Due to this unique format the back end code needed to be compromised because the lookup table could not be joined with a query with other tables as the tables would need to have its 2nd row removed. The work-around solution for this is to have a centralized translation table in the database hold

the top row values linked to its second row value. The lookup table will be queried and stored in a HashMap when the application ever needs to retrieve data. With the values in the HashMap, linking the number of the columns with the lookup table value is possible. Although this is a solution, its not ideal as the application should only have single centralized lookup table that all countries share.

1	KeroseneAvai	SolarSuitable	SolarSuitable	Climate Zone
2	FuelAvailabilit	YesOrNo	YesOrNo	TempartureZo
3	4	1	1	2

Figure 19: Facility.csv.

6.2 Extra columns not in CCEI format

Although we would require the data to follow the CCEI format, the data files are not stored in the database in full CCEI format. The first addition to the format is a Country column for all the tables. This is to prevent the need to created multiple tables for countries. As some countries may have similar primary keys the best solution was to use the country as a another primary key in addition to the table's primary key. Another column that might be added in the future is a time stamp since it would be necessary for future features involving time specific data.

7 DISCUSSION

7.1 Lessons Learned

For much of the team, this was our first introduction to the vaccine cold chain. We learned how complicated and political vaccinations are. This means that a cold chain data mapper will need to be as clean and accessible as possible, while respecting various political climates.

We also learned the extent of how messy data can be. All of our major problems were related to data in some form or another; whether from the shapefile or country collected CCEI level. Problems from small naming typos to non-unique primary keys made for unexpected surprises that slowed our progress down significantly.

Ultimately, we learned how vital a standardized data source is to make information accessible on a global scale. The frequency of updating is also critical, since stakeholders cannot make informed decisions if the data is stale.

8 FUTURE PLANS

8.1 Server

Currently, the application runs on localhost:8000. As the project expands, it would better to run on a server instead. Hosting this application on a server would allow available data to be consistent with everyone's usage of the application. Making it consistent is crucial for accurate data analysis especially for large organizations which need to oversee many different countries' data. This would also make it easier for more stakeholders to access the project.

Another benefit of hosting it on a server is the added safety for other people's data. Right now a user can easily take the credentials from `db.js` and maliciously take other countries' data. To prevent this, we would need to hide `db.js` and any other files which contain the database credentials. This can be done with a server as these files will not be saved locally as it is now.

8.2 More Visualizations

As more stakeholders gain access to the project, they will likely have suggestions for visualizations that would help better inform their process. We chose to focus on power source and refrigerator type as our main factors, but other stakeholders may have other desired visualizations. In the future, our stakeholders may want to consider more factors, such as population vs vaccination storage etc.

8.3 Customizable visualizations

It may be the case that certain countries care more about certain features. It could be beneficial for each country to be able to modify their personal dashboard to meet their requirements and priorities by displaying the data that may be more important to them. For example, if one country is experiencing an outbreak of a particular disease, it could be nice to customize their dashboard around the storage and distribution of vaccines for that specific disease. This may bring more flexibility to our user in modifying the dashboard interface to fit their needs.

8.4 Heat Maps

As the cold chain is an inherently spatial problem, it would be beneficial to overlay additional aspects of available data directly on the map. This could be accomplished through a heat map. A heat map is a representation of data in the form of a map in which the density of data values are represented as colors. One application would be to display refrigerators in a heat map based on the density of refrigerators in a certain area. This could be expanded to discover areas full of old or faulty equipment, or lacking functional refrigerators in general.

8.5 Upload Files

The current manner that data is uploaded is to enter the full path of each file into different text boxes. This is a bit awkward for the user to do, as it requires the user to manually locate each file. It would be better to follow the existing design pattern of an upload mechanic. This upload feature may work well with a future implementation of using a 'BULK INSERT' as it seems that the files would need to be uploaded into the cloud storage of the database before that type of query will work.

8.6 Admin Level Accounts

Cold chain data is sensitive material for many countries, so various people working within the cold chain have different levels of access to the data. We would like to implement this into the project by creating admin level accounts in which a particular user only has access limited to their specific region or level. A rough idea of how these accounts could be handled is having a Global account, Country account and Regional account. A global stakeholder such as WHO or Gavi would have a Global account so they could analyze data between countries and make informed decisions on where to send funding. This account will not be able to edit data unless given authorization. A Country account will belong to a cold chain leader

within a country and would allow the user to control the regional accounts, view, and update data of its own country. A regional account would work similarly on a smaller scale. This will ensure that the sensitive data is better protected, yet accessible.

8.7 Time Stamps

Once a stable version of the application is developed we would like to upgrade the database to hold more data with time stamps. The time stamp could be a number representing the year because every facility gets updated at different times so a yearly time stamp would make sense to enforce consistency. This time stamp will be very helpful for seeing a country's progress in upgrading their technology. This would be especially useful for large stakeholder to see the impact of their contributions.

If it were feasible to collect more data frequently, it would be interesting to display a chart with the number of vaccines each month or week. This way we could readily track and perhaps predict and prevent stock-outs.

8.8 Updating Data

Unfortunately, the only way to update/delete data from the database is to manually write and run queries. There is no feature in the application that supports the updating of data through the user interface. This feature should be one of the first features to be implemented due to the importance of the dynamic changes of data, as mentioned in the previous section. This feature should be available to the authorized users only. There are two types of ways we would like to implement this feature: big updates and small updates. We understand that not all countries have the same work flow so we would like to give the option to certain users to either edit the data with a large .csv that would be provided by the a high level account or editing a small amount of data of a region/facility, an example of this can be if a user would like to report a one less coldroom available in the region.

8.9 Exhaustive Error Checking

The `check_ccei` method checks the data for very basic errors which is not sufficient for a scalable product as the larger the data set, the more specialized the errors can be. At the moment, the checker checks for a valid file path, correct column names, correct cells with unique inputs and correct cell inputs types. Although this seems like it may be enough, the files provided are very dependent on each other so an inconsistency in one of the files may return inaccurate results. Some of the possible checks that have yet to be implemented are checking that the shape files and admin region data are consistent, checking that fridge model exists in the catalog and facility regions exist as regions. Many errors can really only be found and checked for the more usage the application gets as the data is very complex.

8.10 Country/Region Comparison

This feature would be wonderful for stakeholders to make funding decisions. The users can directly compare visualizations of regions to decide which area objectively could use the assistance. This would help increase the efficiency of resource allocation.

8.11 Coding Cleanup

As we only had 10 weeks to research the problem, design and implement a solution, the code is not scoped for a large scale project. One area which could use improvement is the general architecture of the files and dividing the code into more files. Much of the code base is put in just a few files making it very difficult navigate methods and regions.

9 CONCLUSION

As we discussed in this paper, vaccines are an incredibly important commodity and are our only hope for eradicating high-effect diseases that have plagued the human race for far too long. The vaccine cold chain is an essential supply chain management system that allows stakeholders like the WHO, UNICEF and NGO organizations to effectively distribute vaccines. With the new CCEI data model, there now exists a formal definition for cold chain inventory and collection of data using this model allows for querying of data, regular updates and predictive modeling. Our web dashboard, built upon the CCEI data model visualizes important statistics regarding equipment upgrades, and demonstrates but a small fraction of what is possible for the cold chain if there were to exist a global database of standardized data.

ACKNOWLEDGMENTS

Thank you Richard Anderson and Fahad Pervaiz for assisting us through this project from start to finish!

Thank you Matt Morio from PATH for taking the time to meet with us and provide us with real-world refrigerator insights!

REFERENCES

[1] Brian Greenword. 2013. The contribution of vaccination to global health: past, present and future. Retrieved June 1, 2019 from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4024226/

[2] Polio Global Eradication Initiative. [n. d.]. Endemic Countries. Retrieved June 1, 2019 from http://polioeradication.org/where-we-work/polio-endemic-countries/

[3] World Health Organization. [n. d.]. Immunization in Practice - A practical guide for health staff: The vaccine cold chain. Retrieved June 1, 2019 from https://www.who.int/immunization/documents/IIP2015_Module2.pdf

[4] Fahad Pervaiz Richard Anderson. 2014. Data Migration for Cold Chain Inventory Systems. Retrieved June 2, 2019 from https://pdfs.semanticscholar.org/a88b/7e6a22a902988d57d6fd8655b0e84fb83a35.pdf