CDC Data Reconciliation Project: Setup Documentation

Source Code:
https://github.com/waffy1901/JID-3314-CDC-Data-Reconciliation

Tutorial Video (demonstrating this documentation):
https://youtu.be/VMjU3pbrHXg

Technologies Used:
- Python - FastAPI
- JavaScript - ReactJS
- SQLite

Prerequisite Downloads:
- Python version 3.10+ (Download)
- Install PIP packages after downloading Python
    - Run inside terminal:
        - **For Windows:**
            - `pip install uvicorn fastapi pyodbc python-multipart`
        - **For Linux/MacOS:**
            - `pip3 install uvicorn fastapi pyodbc python-multipart`
- NodeJS version 20+ (Download)
- ODBC (Open Database Connectivity) Driver
    - You will need to have installed an ODBC driver that is specific to the database that your state uses for storing cases. For NBS, this would be Microsoft SQL Server - (Download).
    - After installing the ODBC driver, restart your computer.

Build / Run Frontend:
1. Navigate to frontend folder - run `cd CDC-Data-Reconciliation-Frontend`
2. Run `npm i` to install necessary npm packages such as ReactJS
3. Specify the backend server URL inside config.json
    a. config.json is located inside the “src” folder
    b. The API_URL field should include the local IPv4 address of where the backend will be hosted as well as the port number that is used.
4. Run `npm run build` to build the frontend UI.

If you would like to work on the frontend UI while having automatic reloading when you change files you can run `npm run dev` to start up a development server for the frontend.

Run Backend:

1. Navigate to backend folder - run `cd CDC-Data-Reconciliation-Backend`
2. Edit config.json
    a. driver - this is where you will specify the ODBC driver you have installed. To see a list of all ODBC drivers installed on your system, you can run the following commands while in the backend folder.
        i. **For Windows:**
            1. `python odbc_list.py`
        ii. **For Linux/MacOS:**
            1. `python3 odbc_list.py`
    b. server - this field should be the URL to your States SQL Database.
    c. database - this field specifies the name of the database for the backend server to use. For NBS, this should be “NBS_ODSE”.
    d. database_username - this field should be set to the username of the login for the state SQL database if you do not want to set an environment variable.
    e. database_password - this field should be set to the password of the login for the state SQL database if you do not want to set an environment variable.
    f. config_password - this field specifies the password that users will have to enter in the UI in order to update settings for the application.
    g. port - this field specifies the port number the server should use. Make sure this port is the same as the port used in the frontend API_URL.
    h. If you are using backslashes in any of the aforementioned fields, ensure that you use 2 backslashes. If you merely use one backslash, it will result in a JSON error, indicating an invalid escape. For instance, in lieu of setting “database\name” for the database field, you would set the database field to **“**`database\\name`**”**.
    i. If the `database_username` and `database_password` fields are set, there is no need to set environment variables for the database login and step 3 can be safely skipped.
3. Set environment variables for database login
    a. DB_USERNAME - this environment variable should be set to the username of the login for the state SQL database.

b. DB_PASSWORD - this environment variable should be set to the password of the login for the state SQL database.
c. The following instructions are for temporary setup
d. For Linux/MacOS:
    i. In your terminal, type the following 2 commands to set the environment variables for the database login
    ii. `export DB_USERNAME='your_database_username'`
    iii. `export DB_PASSWORD='your_database_password'`
e. For Windows:
    i. In the command prompt, type the following 2 commands to set the environment variables for the database login
    ii. `set DB_USERNAME=your_database_username`
    iii. `set DB_PASSWORD=your_database_password`
    iv. Alternatively, if you are using Powershell, the commands are:
    v. `$Env:DB_USERNAME = "your_database_username"`
    vi. `$Env:DB_PASSWORD = "your_database_password"`

4. Ensure that the frontend is built, if not follow the steps under **Build / Run Frontend**
5. Finally, you can start the server.
    a. For Windows:
        i. Run `python server.py`
    b. **For Linux/MacOS:**
        i. Run `python3 server.py`

If you experience the application becoming slow and unresponsive after loading large amounts of discrepancies for reports or many users using the application at once, you might want to consider increasing the number of workers that uvicorn uses when running the backend server. This can be done by editing the last line of code in the server.py file. You can edit the parameter named, “workers”, and set the number of workers you would like. Note that you should not specify a number of workers that is larger than the amount of cores that your CPU has as this may cause issues. Alternatively, you can run uvicorn or gunicorn from the command line in order to specify the number of workers. Here is a link to the documentation going over backend deployment using uvicorn or gunicorn, https://www.uvicorn.org/deployment/. This link also specifies how to add SSL certification to the server so that data is encrypted in transmission while using the application.

Query Configuration:

The query used to get the state case data from the state SQL database is specified in a file called, query.sql, located inside the CDC-Data-Reconciliation-Backend folder. The default query.sql file is for states that utilize NBS.

If your state does not use NBS, you can write a SQL query that gets case data and returns the columns below. These columns except for *add_time* should have the same values as the CDC dataset for a specific CaseID. An example of some data that is returned by the default query.sql file is also shown below.

The query needs to have a “?” where we are able to pass in the year for which the query should be run for. In the query.sql file provided, the “?” appears at the very end of the file.

- add_time: The time at which the case was added to the state database (Not present within CDC dataset)
- CaseID
- CountyReporting
- EventCode
- EventName
- MMWRYear
- MMWRWeek
- CaseClassStatus
- Sex
- BirthDate
- Age
- AgeType
- Race
- Ethnicity

add_time	CaseID	CountyReporting	EventCode	EventName	MMWRYear	MMWRWeek	CaseClassStatus	Sex	BirthDate	Age	AgeType	Race	Ethnicty
2023-02-10 15:18:30.933	CAS10001004GA01	004	11065	2019 Novel Coronavirus	2023	06	Not a Case/Deleted	1	1990-01-01	33	0	2	2
2023-02-22 08:54:11.103	CAS10001023GA01	004	11065	2019 Novel Coronavirus	2023	08	Suspect	2	2023-02-06	16	3	1	9
2023-02-22 08:54:11.103	CAS10001023GA01	004	11065	2019 Novel Coronavirus	2023	08	Suspect	2	2023-02-06	16	3	2	9
2023-02-22 08:54:11.103	CAS10001023GA01	004	11065	2019 Novel Coronavirus	2023	08	Suspect	2	2023-02-06	16	3	3	9
2023-02-22 08:54:11.103	CAS10001023GA01	004	11065	2019 Novel Coronavirus	2023	08	Suspect	2	2023-02-06	16	3	2	9
2023-02-22 08:54:11.103	CAS10001023GA01	004	11065	2019 Novel Coronavirus	2023	08	Suspect	2	2023-02-06	16	3	5	9
2023-02-22 13:26:57.417	CAS10001025GA01	004	10245	African Tick Bite Fever	2023	08	Suspect	2	2023-02-06	16	3	1	2
2023-02-22 13:26:57.417	CAS10001025GA01	004	10245	African Tick Bite Fever	2023	08	Suspect	2	2023-02-06	16	3	2	2
2023-02-22 13:26:57.417	CAS10001025GA01	004	10245	African Tick Bite Fever	2023	08	Suspect	2	2023-02-06	16	3	3	2
2023-02-22 13:26:57.417	CAS10001025GA01	004	10245	African Tick Bite Fever	2023	08	Suspect	2	2023-02-06	16	3	2	2
2023-02-22 13:26:57.417	CAS10001025GA01	004	10245	African Tick Bite Fever	2023	08	Suspect	2	2023-02-06	16	3	5	2
2023-02-23 10:09:19.437	CAS10001027GA01	005	50221	Zika Virus Infection, Non-Congenital	2023	08	Suspect	1	1970-03-02	52	0	1	1
2023-02-23 10:09:19.437	CAS10001027GA01	005	50221	Zika Virus Infection, Non-Congenital	2023	08	Suspect	1	1970-03-02	52	0	2	1
2023-02-23 10:09:19.437	CAS10001027GA01	005	50221	Zika Virus Infection, Non-Congenital	2023	08	Suspect	1	1970-03-02	52	0	3	1
2023-02-23 10:09:19.437	CAS10001027GA01	005	50221	Zika Virus Infection, Non-Congenital	2023	08	Suspect	1	1970-03-02	52	0	2	1
2023-02-23 10:09:19.437	CAS10001027GA01	005	50221	Zika Virus Infection, Non-Congenital	2023	08	Suspect	1	1970-03-02	52	0	5	1
2023-02-23 10:09:19.437	CAS10001027GA01	005	50221	Zika Virus Infection, Non-Congenital	2023	08	Suspect	1	1970-03-02	52	0	9	1
2023-02-21 11:34:15.637	CAS10001015GA01	001	10100	Hepatitis B, acute	2023	08	Confirmed	1	1990-01-01	33	0	5	2
2023-03-01 07:51:44.993	CAS10001030GA01	001	10660	Yellow Fever	2023	09	Confirmed	2	2000-01-29	23	0	1	2
2023-03-01 07:51:44.993	CAS10001030GA01	001	10660	Yellow Fever	2023	09	Confirmed	2	2000-01-29	23	0	2	2
2023-03-01 07:51:44.993	CAS10001030GA01	001	10660	Yellow Fever	2023	09	Confirmed	2	2000-01-29	23	0	3	2
2023-03-01 07:51:44.993	CAS10001030GA01	001	10660	Yellow Fever	2023	09	Confirmed	2	2000-01-29	23	0	2	2

CLI Comparison:

For states or users that do not want to have to set up the frontend application, we have created a command line interface that can be run. This command line will take command line arguments for the local path of the CDC csv file you want to compare against the state database and is denoted by -c, as well as a local path to an output folder which will contain results.csv and stats.csv files after the comparison is made and is denoted by -o. Additionally, it takes in a year argument, denoted by -y to specify which year you will be querying. Moreover, it takes in a -a argument that allows you to filter comparisons by attributes. If the -a argument is not specified, it will compare all attributes. For instance, we can filter comparisons to only compare event codes and case class statuses. We can run our command-line interface via the following commands:

- **For Windows:**
  - `python cli.py -c example-data/cdc.csv -o output -y 2023 -a EventCode CaseClassStatus`
- **For Linux/MacOS:**
  - `python3 cli.py -c example-data/cdc.csv -o output -y 2023 -a EventCode CaseClassStatus`