# Final Report

Team 5 – Reporting

12/16/2023

## Team

- Dhananjay Jagdish Dubey – Data Architect
- Mugunthan Krishnan – Microservices Engineer
- Alexander Morozov – Product Manager
- Saravana Prabhu Ramasamy – Quality and DevOps Engineer
- Ankush Ranapure – UI/UX Engineer

## Objective

Our team provides Agency Managers with a set of report-generating tools to help them aggregate the necessary information for the legally required reports and make informed decisions based on the most current data from the system.

Our web service generates a series of reports compliant with the requirements of SECTION 704 ANNUAL PERFORMANCE REPORT FOR STATE INDEPENDENT LIVING SERVICES PROGRAM (later referred to as the Federal Report), which serves as a performance activity measuring instrument of independent living programs, including both quantitative and qualitative information.

Our web service also provides Agency Managers with statistics and other information essential for running the agency. The goal is to give the managers an overview of the state of their agencies so that they can fulfill the needs of both the organization and its patients.

## Requirements Analysis and Specification

The Federal Report requires comprehensive information about the performance of the agency expressed in a number of tables. For better user experience and performance, we decided not to generate a single complete report, but to provide a set of web pages featuring a specific portion of the Federal Report. The combination of these subreports can be used to produce the final document. Even though an automatic compilation can be achieved as a part of the far vision of the product, in the current implementation, we provide an export feature for all reports, so that they can be kept for agencies' records or be easily compiled into the Federal Report.

We split the Federal Report into several topics and addressed the following ones as part of the initial implementation and near vision of the product:

- Patients' Demographic Information Report
- Funded Services and Achievements Report
- Community Activities and Coordinatiom Report

After analyzing the common-sense needs of Agency Managers, we identified a set of reports that should help them stay informed about the agency's state:

- Services Provided Report
- Case Managers's Utilization Report
- Case Manager's Performance Report
- Types of Individuals with Significant Disabilities Report

## System and Program Design

Our web service consists of four main components: Frontend, Reports API module, Community Events API module, and database.

The most significant constraint of our project was the dependence on the other teams' APIs, which were also going to be unavailable until the end of the entire product development process. Therefore, one of the goals we tried to achieve while designing the system was to decrease coupling with the other teams' projects in order to be able to work on our portion of the product in parallel with other teams. Thus, we introduced an additional communication layer between our web service and other team's APIs – the Reports API module, which was supposed to be an interface between our web service and the outer world that would deal with all external communication and produce a clean, formatted, and ready-to-use data for our frontend. This way, the front end could be developed independently and rely on the data format our team had agreed upon, which also removes the need for frequent updates of the frontend portion once there is a change to any of the external APIs. In addition to that, the backend would also remove the burden of processing and aggregating potentially large amounts of data from the frontend, and thus preserve the resources of the web server hosting the common application.

Since not all data is captured by the other teams, we had to provide Agency Managers with a way to create, read, update, and delete information about their agency's community activity events, so that we could generate the related portion of the Federal Report. For this purpose, we developed the Community Events API module and set up a database.

### Database Design

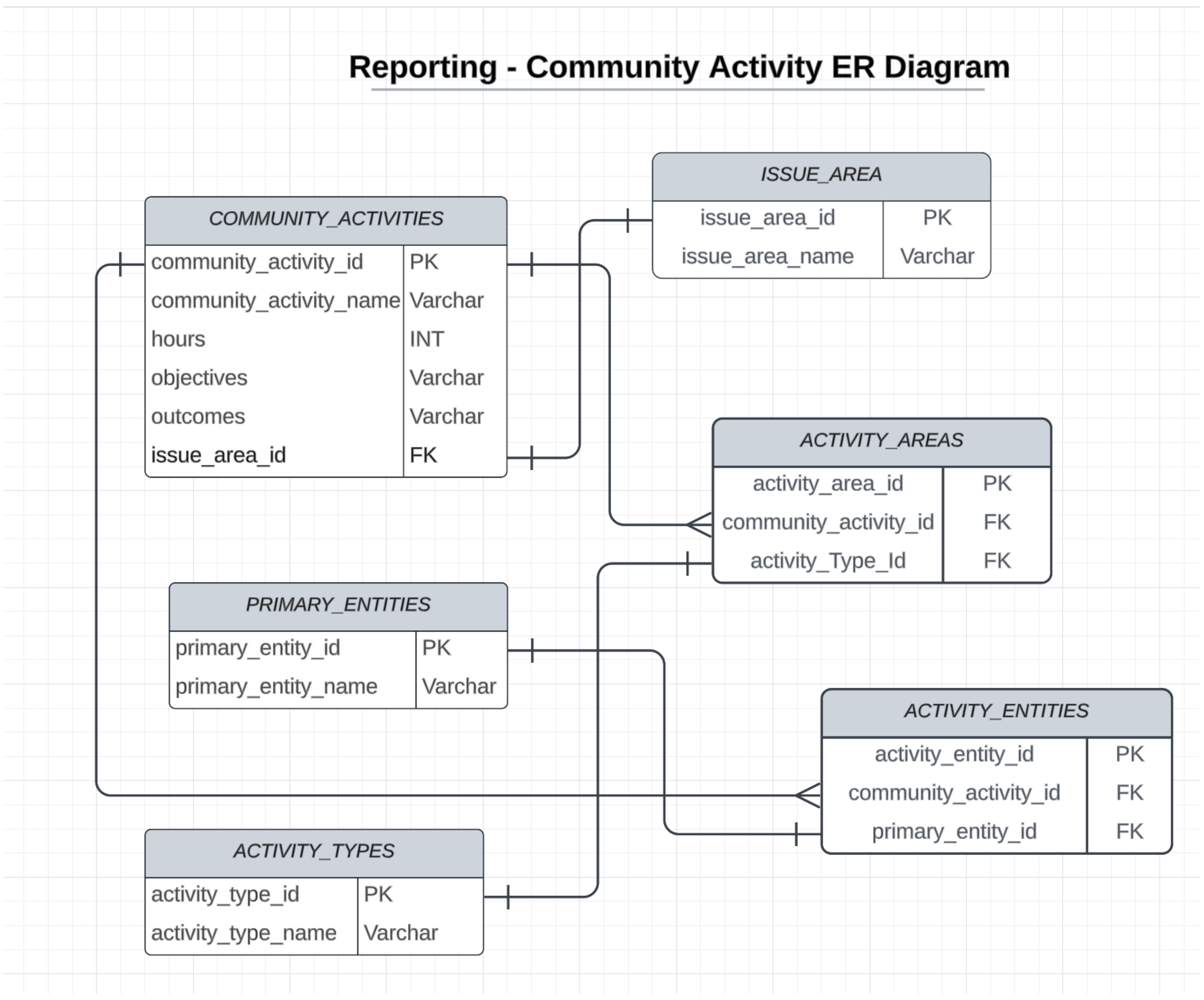
https://g8666ffd13ccfb8-wecarecommunitydatabase.adb.us-ashburn-1.oraclecloudapps.com/ords/admin/_sdw/

Database Technology: Oracle Database

Host: Our database is hosted in Oracle Cloud Infrastructure and we are using a relational database and our APIs connect to the OCI database.

DB Schema:
The reporting team has data for Community Activity and it has six tables: `ISSUE_AREA`, `COMMUNITY_ACTIVITIES`, `PRIMARY_ENTITIES`, `ACTIVITY_TYPES`, `ACTIVITY_AREAS`, and `ACTIVITY_ENTITIES`. These tables are related as follows:

**Issue_Area and Community_Activities**: The `ISSUE_AREA` table is related to the `COMMUNITY_ACTIVITIES` table through a foreign key `issue_area_id` in the `COMMUNITY_ACTIVITIES` table. This relationship signifies that each community activity is associated with an issue area. For example, an issue area like "Healthcare" might have multiple community activities associated with it.
**Community_Activities and Activity_Types**: The `COMMUNITY_ACTIVITIES` table has a relationship with the `ACTIVITY_TYPES` table through the `ACTIVITY_AREAS` table. `ACTIVITY_AREAS` is an associative table connecting `COMMUNITY_ACTIVITIES` and `ACTIVITY_TYPES`. Each activity in the `COMMUNITY_ACTIVITIES` table can be associated with one or more activity types, like "Collaborating and Networking" or "Staff Education/Training".
**Community_Activities and Activity_Entities**: The `COMMUNITY_ACTIVITIES` table is related to the `ACTIVITY_ENTITIES` table through the `ACTIVITY_ENTITIES` table's foreign key `community_activity_id`. This relationship indicates that each community activity may involve one or more primary entities. For instance, a community activity might involve multiple primary entities such as "CILs" (Center for Independent Living) or "SILC" (Statewide Independent Living Council).



## API Design

GitHub API Repo: https://github.com/LeoXander/cs673-project/tree/prod

### Reports API

GitHub: https://github.com/LeoXander/cs673-project/tree/prod/report_api

**Endpoints:**
**GET /demographicchart -** Retrieve aggregated patient information to display as a chart.
**GET /serviceproviderreport -** Retrieve service providers' information.
**GET /casemanagerutilizationreport -** Retrieves case manager info and the patients assigned to them.
**GET /casemanagerperformancereport -** Retrieves case manager info and count of patients assigned.

**GET /servicesachievementsreport -** Retrieves all services & count of completed, scheduled services

## Community Events API

GitHub: [https://github.com/LeoXander/cs673-project/tree/prod/community_activity](https://github.com/LeoXander/cs673-project/tree/prod/community_activity)

**Endpoints:**
**GET /activitytypes -** Retrieves the activity types associated with the community event.
**GET /primaryentities -** Retrieves the primary entities associated with the community event.
**GET /issueareas -** Retrieves the issue areas associated with the community event.
**GET /communityactivity -** Retrieves all the community activity events in the system.
**POST /communityactivity -** Add a new community activity event to the system. Body parameters - Community Name, Issue Area ID, Primary Entities as a list, Activity Types as a list, Objectives, Outcomes, Number of Hours.
**PUT /communityactivity/{communityactivityid} -** Update an existing community activity event. Parameters - communtyactivityid. Body parameters - Community Name, Issue Area ID, Primary Entities as a list, Activity Types as a list, Objectives, Outcomes, and Number of Hours.
**DELETE /communityactivity/{communityactivityid} -** Delete an existing community activity event. Parameters - communityactivityid
**GET /communityactivityreport -** Retrieve the data needed for the community activity report. Parameters - startDate and endDate

## API Implementation

The APIs are implemented in Python and FastAPI web frameworks. FastAPI uses asynchronous programming features like async and await. FastAPI also generates interactive documentation with tools like Swagger UI and Redoc supported. These provide a user-friendly interface for development and testing.

The community activity API module is dependent on the Oracle database hosted in Oracle Cloud Infrastructure. The API application connects to the database using the Python library **OracleDB**.

**API Features:**

1. **CORS Policy:** The APIs have CORS policy enabled to allow access from the UI React application.
2. **Error Handling:** The APIs send different status codes such as 200 for success, 201 for creating a record, and 404 for errors. Since our APIs are dependent on other modules' APIs, if external APIs fail for some reason, the APIs send a 404 error message to the application. If there is any invalid data in the external APIs, the data are handled with proper replacement and sent in the response. For example, if a booking remark is received from the Bookings API as NULL, then it is set as 'No remarks'.
3. **Parameters:** The APIs accept query parameters and body parameters to accept specific requests for data from the database and to add filters for the data.

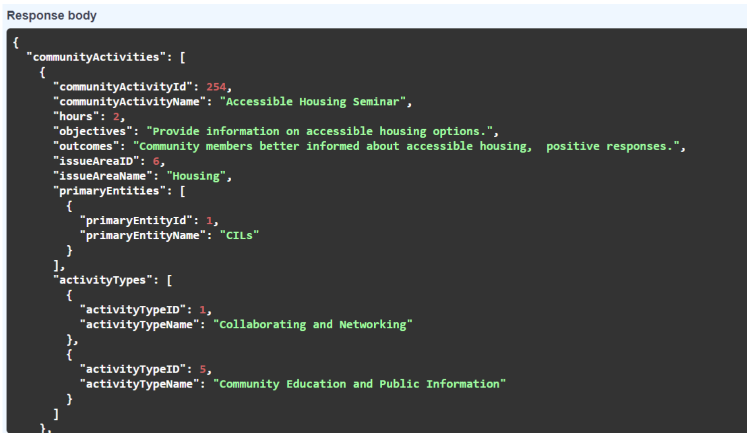
**API Schema**
There are two API modules - Reports and Community Activity Events Module.

Reports API - The reports APIs coalesces the data from the apis from different modules of the product and creates a single point of data for the different types of reports that are displayed in the UI. There are 5 types of reports and each report has a separate API endpoint to it.

Community Activity Events API - The community activity events APIs interact with the Oracle database to retrieve, update, add, and delete community activity events that are being added from the UI. The community activity event report API is the API that supplies the data to the community activity event report that is being used in the FEDERAL 704 report.

The below are the request and response bodies of the community activity events apis:

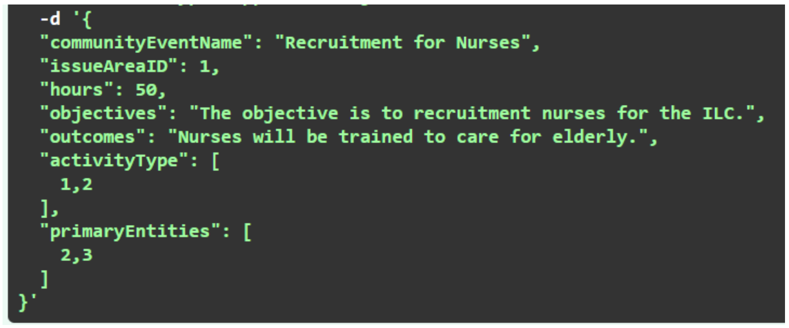
1. GET /communityactivity



Response body

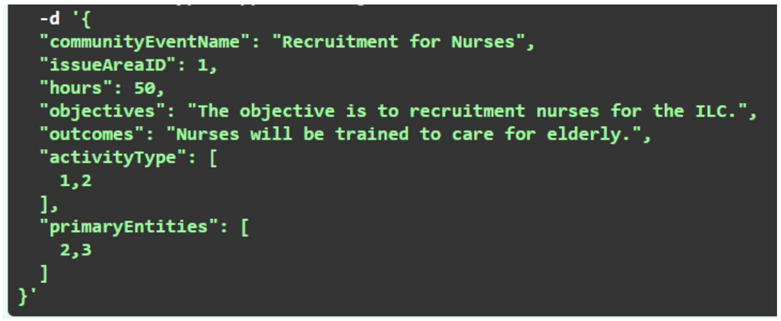
```
{
  "communityActivities": [
    {
      "communityActivityId": 254,
      "communityActivityName": "Accessible Housing Seminar",
      "hours": 2,
      "objectives": "Provide information on accessible housing options.",
      "outcomes": "Community members better informed about accessible housing,  positive responses.",
      "issueAreaID": 6,
      "issueAreaName": "Housing",
      "primaryEntities": [
        {
          "primaryEntityId": 1,
          "primaryEntityName": "CILs"
        }
      ],
      "activityTypes": [
        {
          "activityTypeID": 1,
          "activityTypeName": "Collaborating and Networking"
        },
        {
          "activityTypeID": 5,
          "activityTypeName": "Community Education and Public Information"
        }
      ]
    },
```

2. POST /communityactivity



```
  -d '{
  "communityEventName": "Recruitment for Nurses",
  "issueAreaID": 1,
  "hours": 50,
  "objectives": "The objective is to recruitment nurses for the ILC.",
  "outcomes": "Nurses will be trained to care for elderly.",
  "activityType": [
    1,2
  ],
  "primaryEntities": [
    2,3
  ]
}'
```

3. PUT /communityactivity/{communityactivityid}



```
  -d '{
  "communityEventName": "Recruitment for Nurses",
  "issueAreaID": 1,
  "hours": 50,
  "objectives": "The objective is to recruitment nurses for the ILC.",
  "outcomes": "Nurses will be trained to care for elderly.",
  "activityType": [
    1,2
  ],
  "primaryEntities": [
    2,3
  ]
}'
```

## Frontend Design

GitHub Team 5 module:
https://github.com/himanshu98/digital-health-frontend/tree/team5/src/modules/reporting

Front-End Framework: React JS https://react.dev/
UI Library: Ant Design https://ant.design/
Charts: Chart.js https://www.chartjs.org/docs/latest/
Routing: React Router https://reactrouter.com/en/main
API calls: Axios https://axios-http.com/docs/intro
Deployment: Netlify https://www.netlify.com/
Front End Deployed URL: https://sdpm-wecare.netlify.app/

## DevOps

Heroku Community Events API: https://dashboard.heroku.com/apps/communityactivity
Heroku Reports API: https://dashboard.heroku.com/apps/sdpm-reports

**CI/CD - Github Actions**

- Source code versioning and management are done via GitHub.
- Created **GitHub workflows** in the project for the dev and prod branches.
  - **dev workflow** - Used for CI which includes development integration and testing.
  - **prod workflow** - Used for CD and testing.

**Automated CI/CD Pipeline - Actions**

1. **Triggers** when the code is pushed to the dev or prod branch.
2. A job is created in the action to **build the project**. Project build has the following steps
   a. Get required **environment variables** from GitHub secrets like database authentication information and source code directory to build.
   b. Checkout the code in GitHub actions: https://github.com/actions/checkout
   c. Run tests before deployment
      i. Install the necessary Python package required for the API.
      ii. Testing is done using Pytest
   d. Build, Push, and Release a Docker container to Heroku.

3. Build action is successful when all test cases pass and when deployment is complete. Any errors found are identified and fixed before it gets deployed.

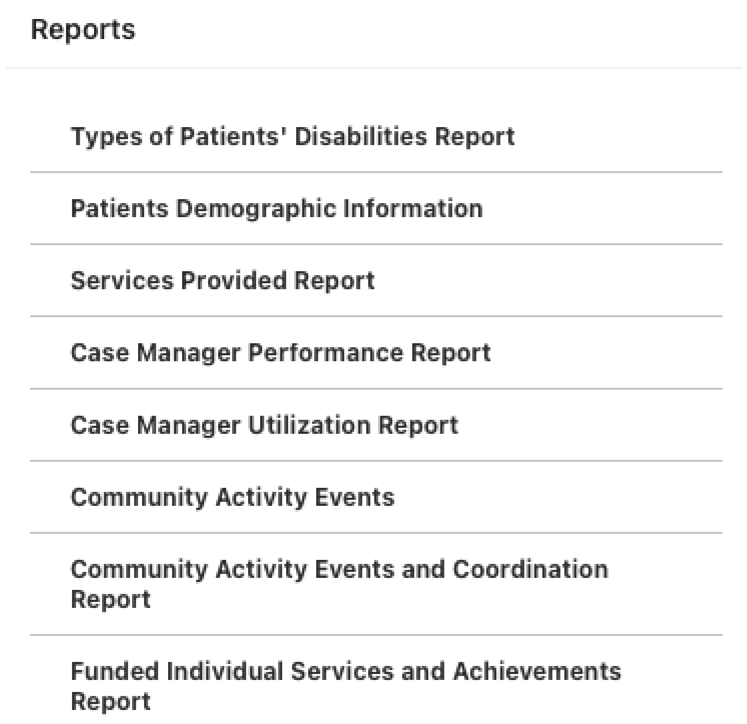
**Container Deployment**

- Developed a docker file using the python:3.10-slim image.
- Docker image holds all the required source code along with its dependencies installed.
- A container is deployed in the Heroku app that starts the API service.

**CI Apps**

Apps used for integration and testing
Community Events API: https://dashboard.heroku.com/apps/dev-communityactivity
Reports API: https://dashboard.heroku.com/apps/dev-sdpm-reports

## User Interface and Features

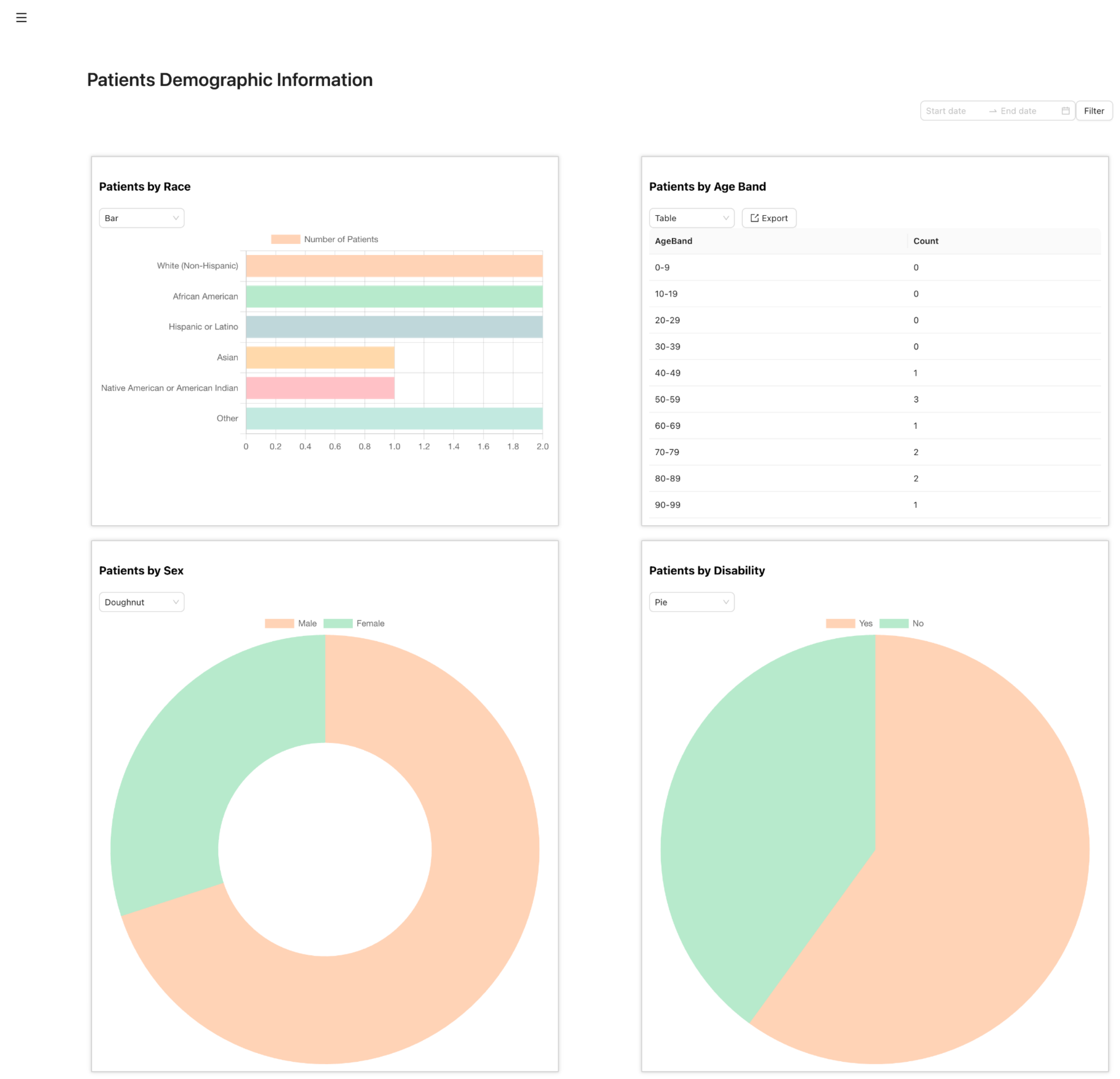

Reports
Types of Patients' Disabilities Report
Patients Demographic Information
Services Provided Report
Case Manager Performance Report
Case Manager Utilization Report
Community Activity Events
Community Activity Events and Coordination Report
Funded Individual Services and Achievements Report

This is a list of all functions and reports available for Agency Managers.

# Patients’ Demographic Information Report

The Federal Report explicitly requires a numerical distribution of patients by age groups, sex, race, ethnicity, and disability types. Since the report only considers information for the last calendar year and the date may change from year to year, we provide the users with the ability to specify an exact date range for the report. As an additional use case, we made the displayed information more digestible by allowing Agency Managers to visualize the data using different types of charts.

GitHub:
https://github.com/LeoXander/cs673-project/blob/prod/report_api/routers/demographic_chart.py



Patients Demographic Information

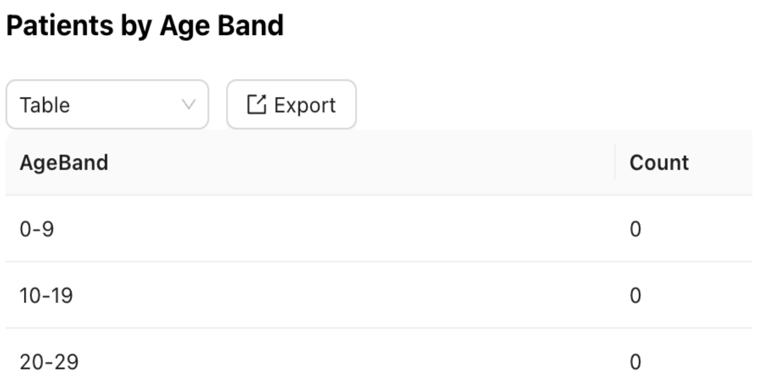
Patients by Race

Patients by Age Band

| AgeBand | Count |
|---|---|
| 0-9 | 0 |
| 10-19 | 0 |
| 20-29 | 0 |
| 30-39 | 0 |
| 40-49 | 1 |
| 50-59 | 3 |
| 60-69 | 1 |
| 70-79 | 2 |
| 80-89 | 2 |
| 90-99 | 1 |

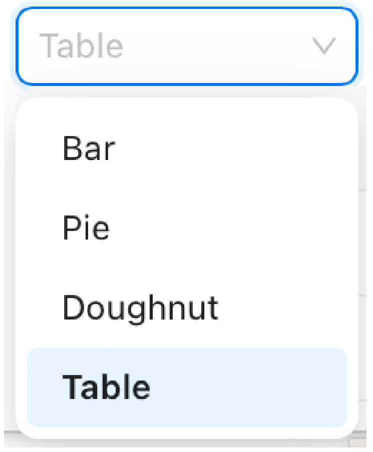
Patients by Sex

Patients by Disability


**Patients by Age Band**

| AgeBand | Count |
|---|---|
| 0-9 | 0 |
| 10-19 | 0 |
| 20-29 | 0 |

Each individual report can be exported to Excel.
This user-digestible view of the statistics. In the top right corner, a user can specify an exact date range.

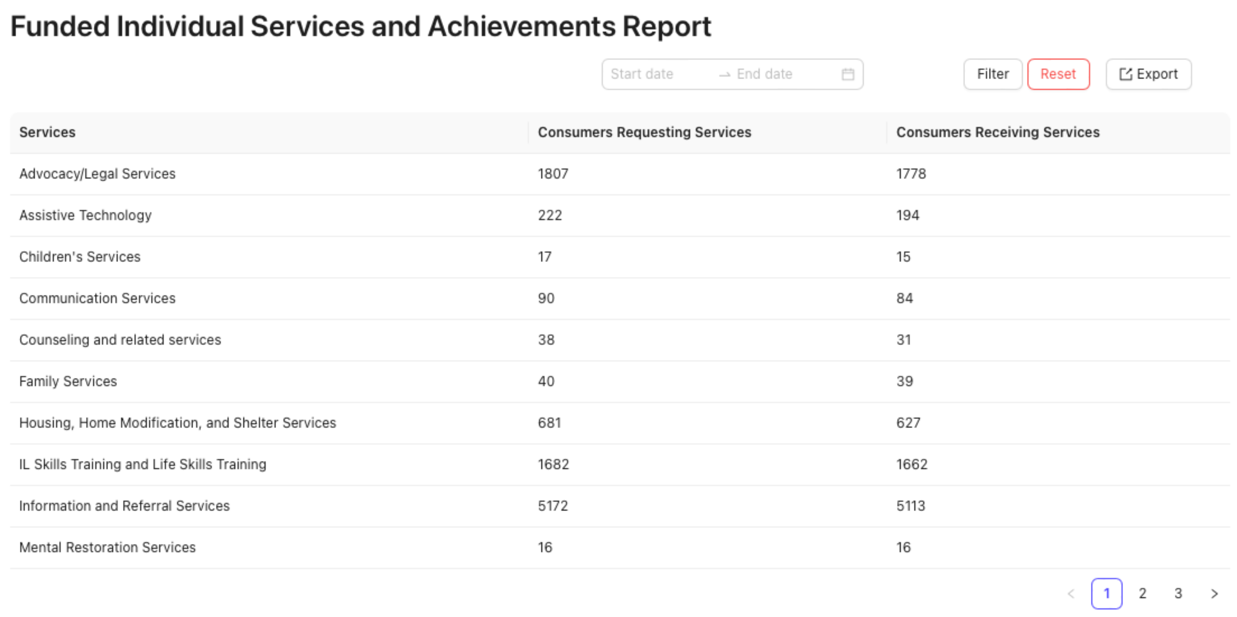


Supported views: Bar, Pie, Doughnut, Table

## Funded Services and Achievements Report

The Federal Report needs information about how many consumers requested and received each type of independent living services provided by the agency. The date range filter and the export feature are also essential here.

GitHub:
[https://github.com/LeoXander/cs673-project/blob/prod/report_api/routers/individual_services_report.py](https://github.com/LeoXander/cs673-project/blob/prod/report_api/routers/individual_services_report.py)


Funded Individual Services and Achievements Report

| Services | Consumers Requesting Services | Consumers Receiving Services |
|---|---|---|
| Advocacy/Legal Services | 1807 | 1778 |
| Assistive Technology | 222 | 194 |
| Children's Services | 17 | 15 |
| Communication Services | 90 | 84 |
| Counseling and related services | 38 | 31 |
| Family Services | 40 | 39 |
| Housing, Home Modification, and Shelter Services | 681 | 627 |
| IL Skills Training and Life Skills Training | 1682 | 1662 |
| Information and Referral Services | 5172 | 5113 |
| Mental Restoration Services | 16 | 16 |

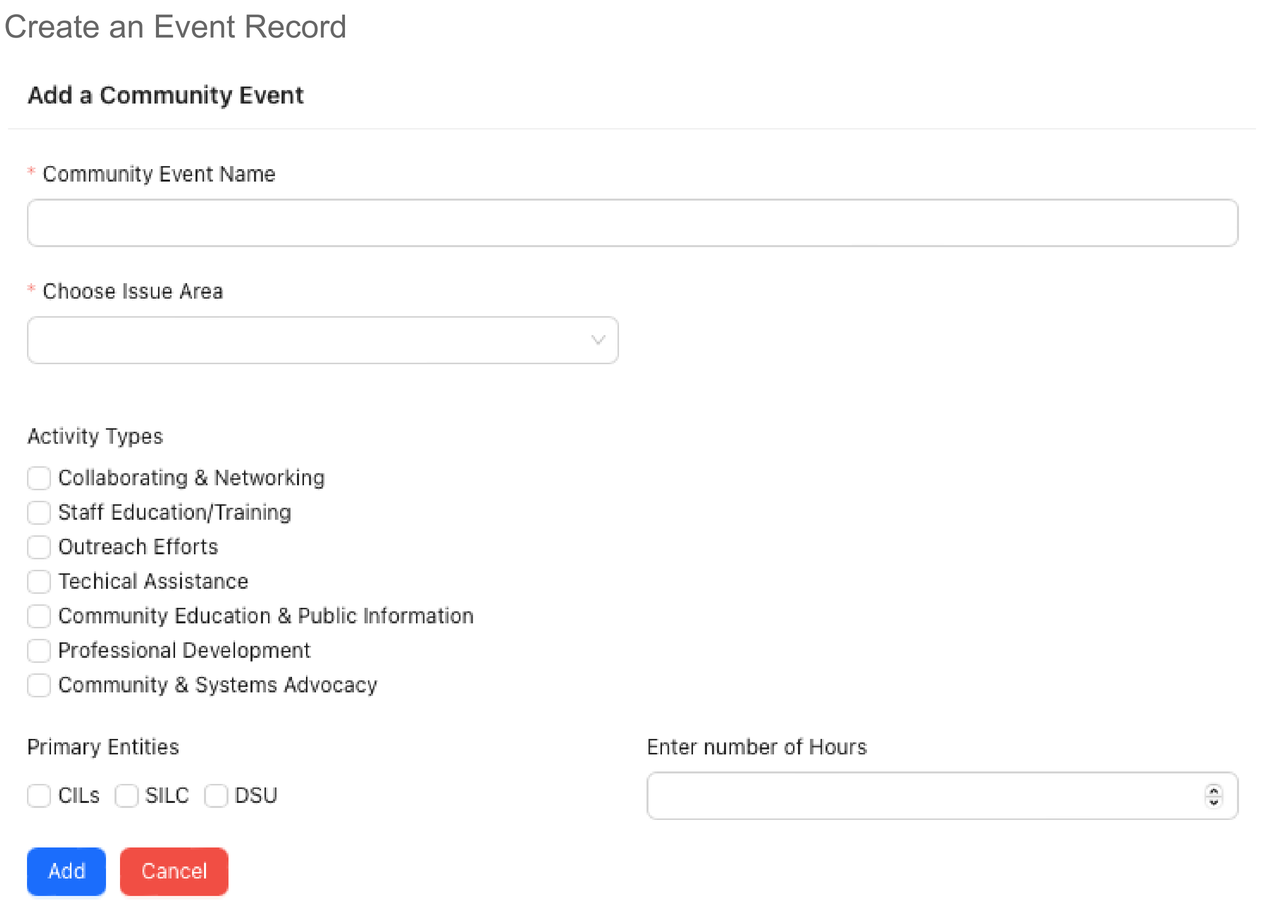
## Community Activities and Coordination Report

The Federal Report requests a summary of all community activity events the agency organized during the reporting year. Since no team collects such information, we've implemented ways to add, view, update, and delete records of the events. Based on this data, we generate the actual Community Activities and Coordination Report.

### Data Collection

The data is collected and managed by our team. For the DB schema and management, please, refer to the Database Design section.

GitHub:
https://github.com/LeoXander/cs673-project/blob/prod/community_activity/routers/community_activity.py



Create an Event Record

**Add a Community Event**

* Community Event Name

* Choose Issue Area

Activity Types

- [ ] Collaborating & Networking
- [ ] Staff Education/Training
- [ ] Outreach Efforts
- [ ] Techical Assistance
- [ ] Community Education & Public Information
- [ ] Professional Development
- [ ] Community & Systems Advocacy

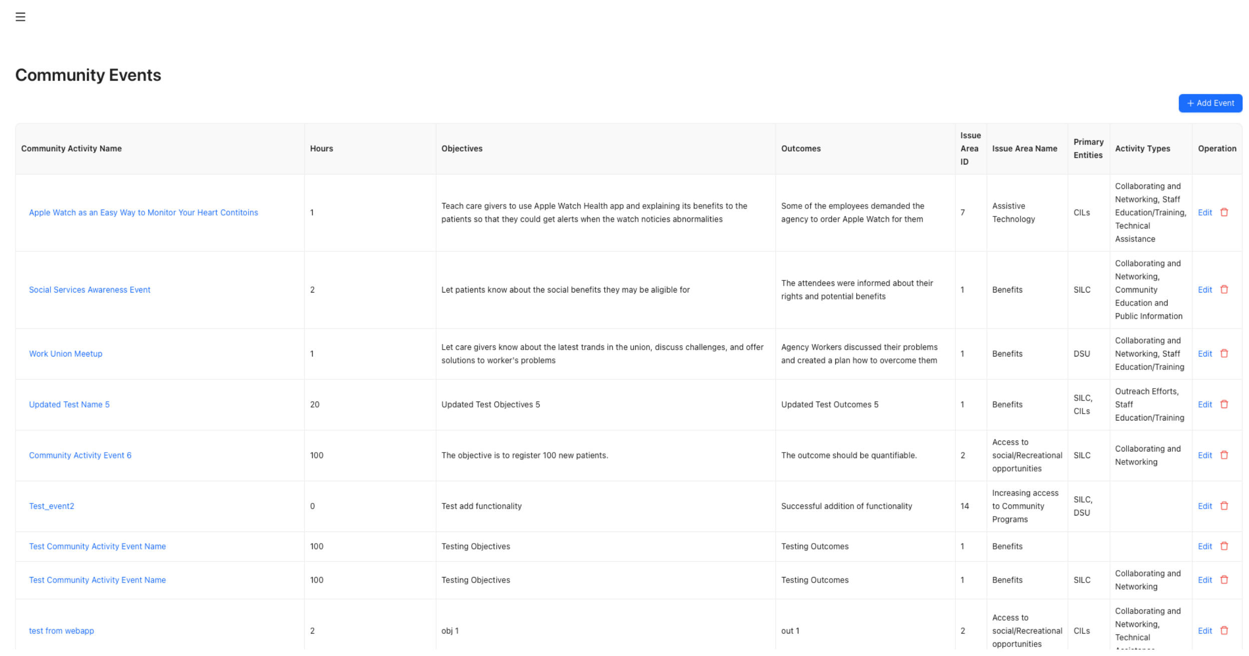
Primary Entities

- [ ] CILs
- [ ] SILC
- [ ] DSU

Enter number of Hours

The user has to specify the name of the event, select or enter a new Issue Area, check all Activity Types and Primary Entities that apply, and enter the number of hours it took.

## View/Update/Delete an Event Record



≡

**Community Events**

+ Add Event

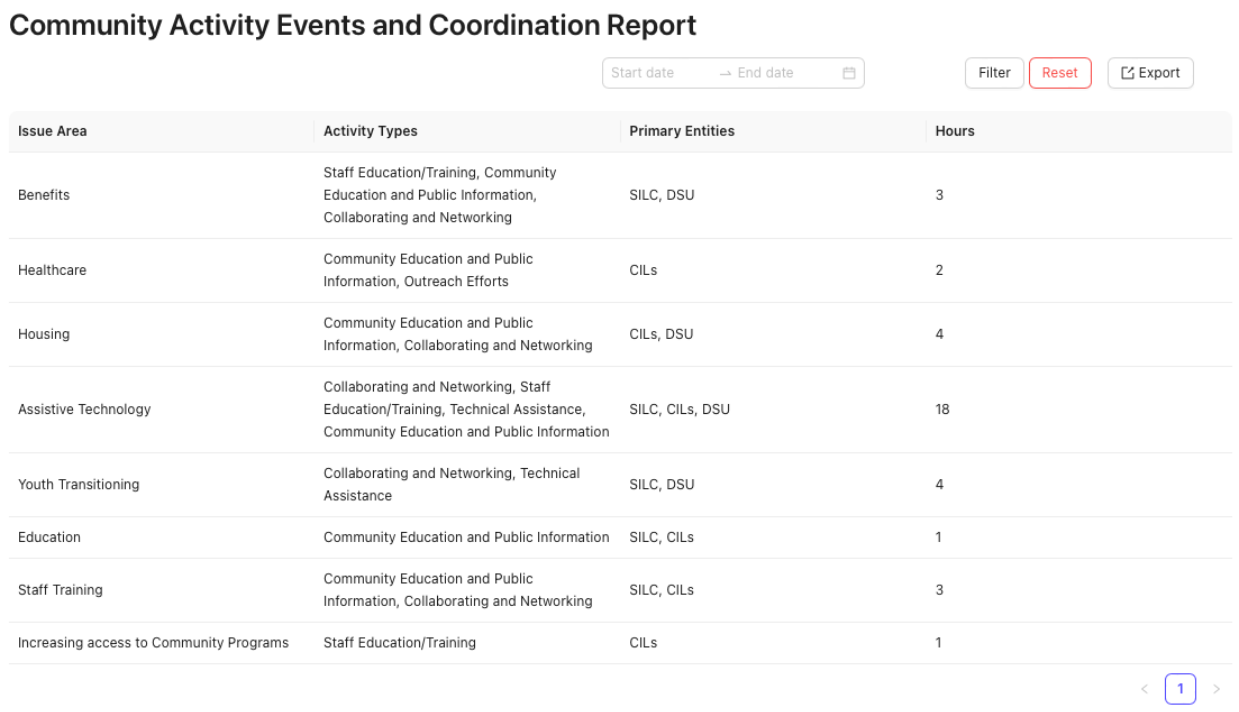
| Community Activity Name | Hours | Objectives | Outcomes | Issue Area ID | Issue Area Name | Primary Entities | Activity Types | Operation |
|---|---|---|---|---|---|---|---|---|
| Apple Watch as an Easy Way to Monitor Your Heart Contitoins | 1 | Teach care givers to use Apple Watch Health app and explaining its benefits to the patients so that they could get alerts when the watch noticies abnormalities | Some of the employees demanded the agency to order Apple Watch for them | 7 | Assistive Technology | CILs | Collaborating and Networking, Staff Education/Training, Technical Assistance | Edit 🗑 |
| Social Services Awareness Event | 2 | Let patients know about the social benefits they may be aligible for | The attendees were informed about their rights and potential benefits | 1 | Benefits | SILC | Collaborating and Networking, Community Education and Public Information | Edit 🗑 |
| Work Union Meetup | 1 | Let care givers know about the latest trands in the union, discuss challenges, and offer solutions to worker's problems | Agency Workers discussed their problems and created a plan how to overcome them | 1 | Benefits | DSU | Collaborating and Networking, Staff Education/Training | Edit 🗑 |
| Updated Test Name 5 | 20 | Updated Test Objectives 5 | Updated Test Outcomes 5 | 1 | Benefits | SILC, CILs | Outreach Efforts, Staff Education/Training | Edit 🗑 |
| Community Activity Event 6 | 100 | The objective is to register 100 new patients. | The outcome should be quantifiable. | 2 | Access to social/Recreational opportunities | SILC | Collaborating and Networking | Edit 🗑 |
| Test_event2 | 0 | Test add functionality | Successful addition of functionality | 14 | Increasing access to Community Programs | SILC, DSU | | Edit 🗑 |
| Test Community Activity Event Name | 100 | Testing Objectives | Testing Outcomes | 1 | Benefits | | | Edit 🗑 |
| Test Community Activity Event Name | 100 | Testing Objectives | Testing Outcomes | 1 | Benefits | SILC | Collaborating and Networking | Edit 🗑 |
| test from webapp | 2 | obj 1 | out 1 | 2 | Access to social/Recreational opportunities | CILs | Collaborating and Networking, Technical | Edit 🗑 |

This page lists all community activity event records in the system. It also allows you to edit or delete a record.

## Report

The report aggregates available information about the community activity events and generates a table per the requirements of the Federal Report.

GitHub:
https://github.com/LeoXander/cs673-project/blob/prod/community_activity/routers/community_activity_report.py



Community Activity Events and Coordination Report

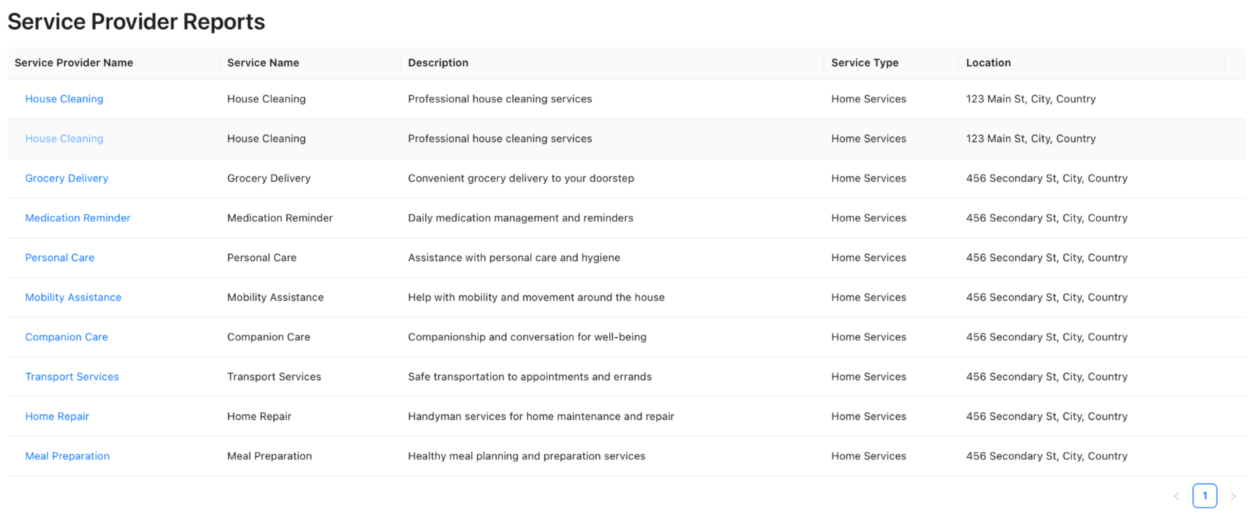
| Issue Area | Activity Types | Primary Entities | Hours |
|---|---|---|---|
| Benefits | Staff Education/Training, Community Education and Public Information, Collaborating and Networking | SILC, DSU | 3 |
| Healthcare | Community Education and Public Information, Outreach Efforts | CILs | 2 |
| Housing | Community Education and Public Information, Collaborating and Networking | CILs, DSU | 4 |
| Assistive Technology | Collaborating and Networking, Staff Education/Training, Technical Assistance, Community Education and Public Information | SILC, CILs, DSU | 18 |
| Youth Transitioning | Collaborating and Networking, Technical Assistance | SILC, DSU | 4 |
| Education | Community Education and Public Information | SILC, CILs | 1 |
| Staff Training | Community Education and Public Information, Collaborating and Networking | SILC, CILs | 3 |
| Increasing access to Community Programs | Staff Education/Training | CILs | 1 |

Users can filter it by date range. The export feature is also available.

## Services Provided Report

This report provides an overview of all services provided by the agency and their performance expressed by patients' feedback. It lists the basic info about all services available. By clicking on a service, the Agency Manager can see the history of visits and review its rating which is based on the patients' reviews.

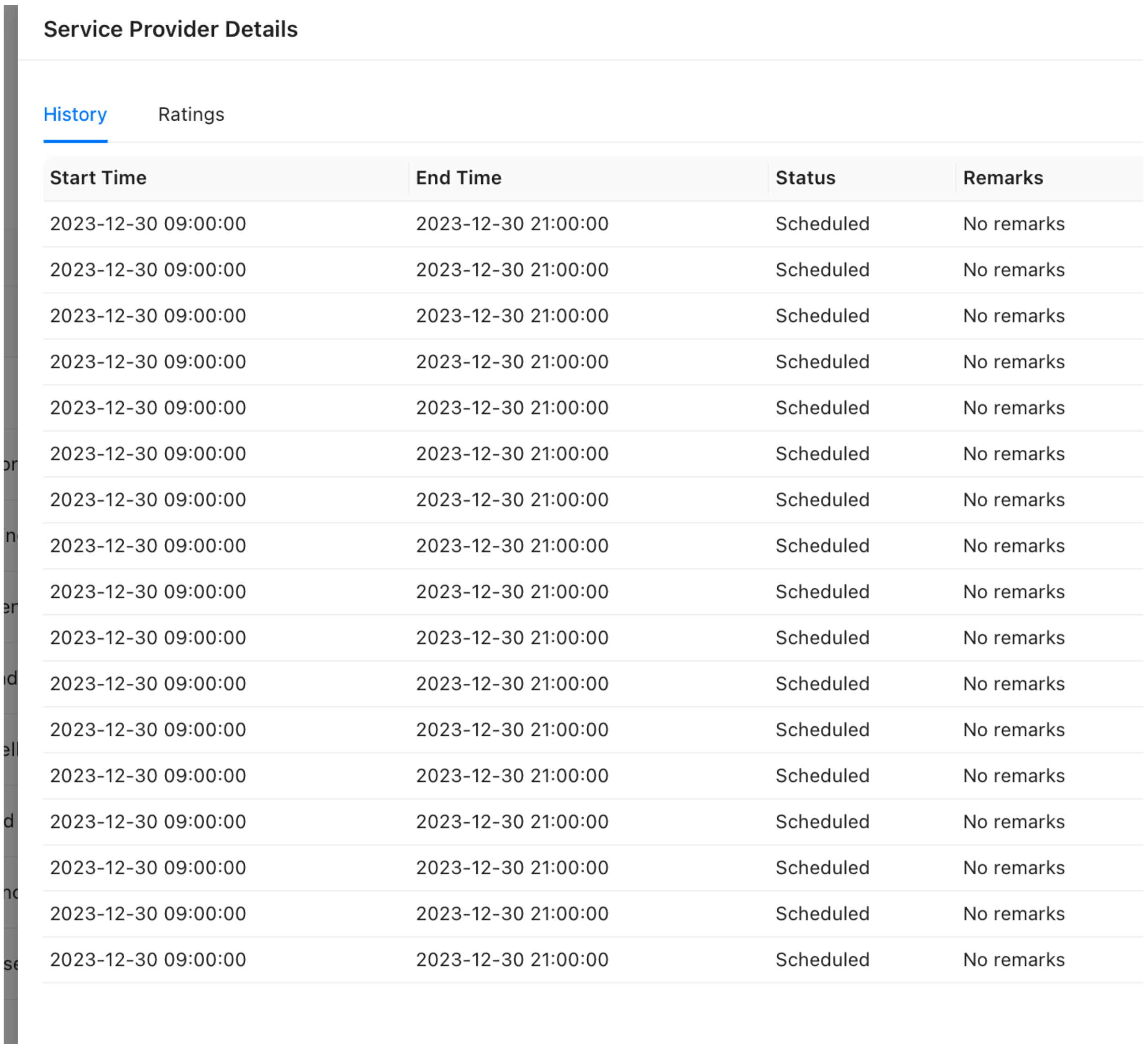
GitHub:
https://github.com/LeoXander/cs673-project/blob/prod/report_api/routers/servicesOffered.py



## Service Provider Reports

| Service Provider Name | Service Name | Description | Service Type | Location |
|---|---|---|---|---|
| House Cleaning | House Cleaning | Professional house cleaning services | Home Services | 123 Main St, City, Country |
| House Cleaning | House Cleaning | Professional house cleaning services | Home Services | 123 Main St, City, Country |
| Grocery Delivery | Grocery Delivery | Convenient grocery delivery to your doorstep | Home Services | 456 Secondary St, City, Country |
| Medication Reminder | Medication Reminder | Daily medication management and reminders | Home Services | 456 Secondary St, City, Country |
| Personal Care | Personal Care | Assistance with personal care and hygiene | Home Services | 456 Secondary St, City, Country |
| Mobility Assistance | Mobility Assistance | Help with mobility and movement around the house | Home Services | 456 Secondary St, City, Country |
| Companion Care | Companion Care | Companionship and conversation for well-being | Home Services | 456 Secondary St, City, Country |
| Transport Services | Transport Services | Safe transportation to appointments and errands | Home Services | 456 Secondary St, City, Country |
| Home Repair | Home Repair | Handyman services for home maintenance and repair | Home Services | 456 Secondary St, City, Country |
| Meal Preparation | Meal Preparation | Healthy meal planning and preparation services | Home Services | 456 Secondary St, City, Country |

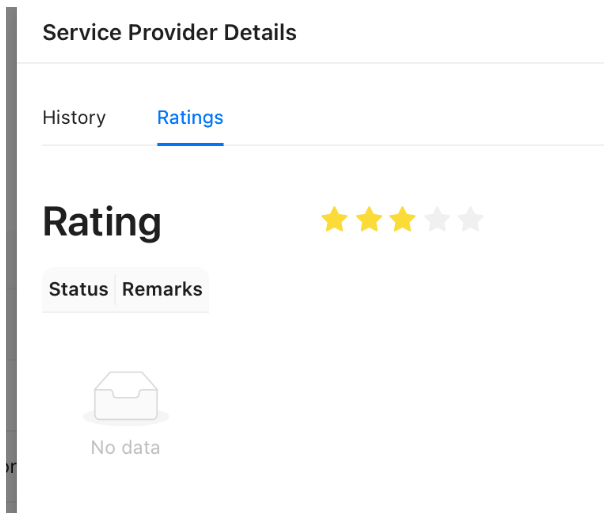
< 1 >



## Service Provider Details

History Ratings

| Start Time | End Time | Status | Remarks |
|---|---|---|---|
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |
| 2023-12-30 09:00:00 | 2023-12-30 21:00:00 | Scheduled | No remarks |



Service Provider Details

History Ratings

Rating ★★★☆☆

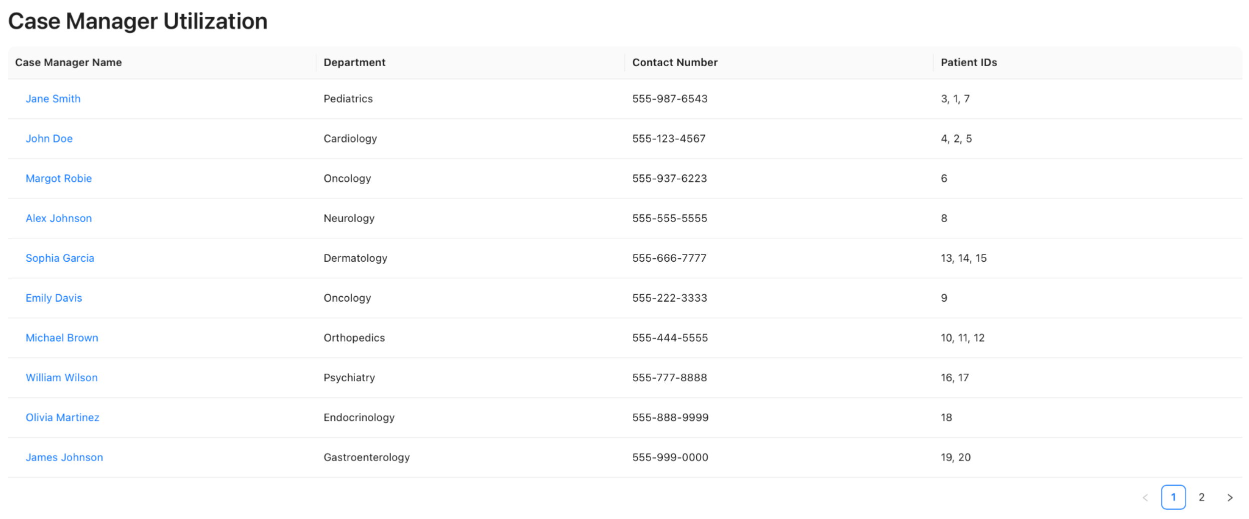
| Status | Remarks |
|---|---|

No data

## Case Manager’s Utilization Report

This report is meant to help Agency Managers control the utilization of Case Managers. The Agency Managers can view how busy the Case Managers currently are by reviewing the number of their current patients. Also by clicking on a Case Manager’s name, a details panel with information about the manager’s patients is displayed.

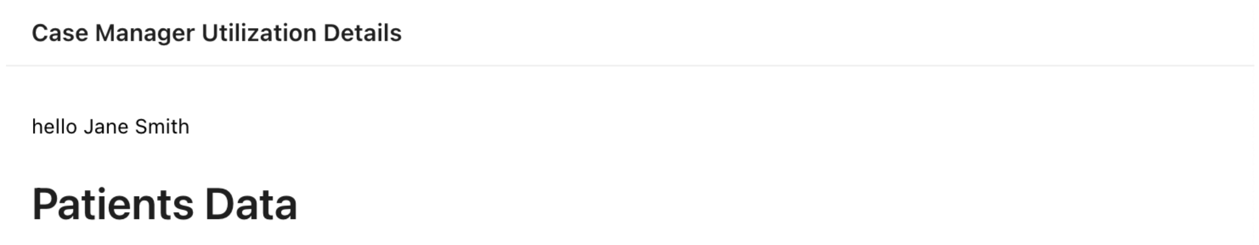
GitHub:
https://github.com/LeoXander/cs673-project/blob/prod/report_api/routers/casemanager.py



Case Manager Utilization

| Case Manager Name | Department | Contact Number | Patient IDs |
|---|---|---|---|
| Jane Smith | Pediatrics | 555-987-6543 | 3, 1, 7 |
| John Doe | Cardiology | 555-123-4567 | 4, 2, 5 |
| Margot Robie | Oncology | 555-937-6223 | 6 |
| Alex Johnson | Neurology | 555-555-5555 | 8 |
| Sophia Garcia | Dermatology | 555-666-7777 | 13, 14, 15 |
| Emily Davis | Oncology | 555-222-3333 | 9 |
| Michael Brown | Orthopedics | 555-444-5555 | 10, 11, 12 |
| William Wilson | Psychiatry | 555-777-8888 | 16, 17 |
| Olivia Martinez | Endocrinology | 555-888-9999 | 18 |
| James Johnson | Gastroenterology | 555-999-0000 | 19, 20 |

< 1 2 >



Case Manager Utilization Details

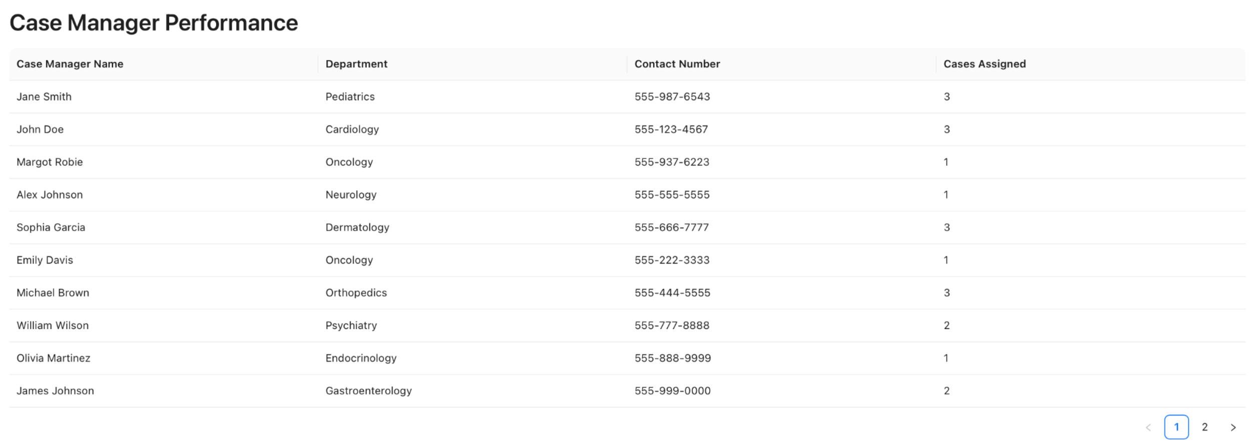
hello Jane Smith

Patients Data

## Case Manager’s Performance Report

Case Manager’s Performance Report gives Agency Managers an idea of the performance of their Case Managers by providing statistical information about their cases and achievements throughout the employment of the Case Managers.

GitHub:
https://github.com/LeoXander/cs673-project/blob/prod/report_api/routers/casemanager.py

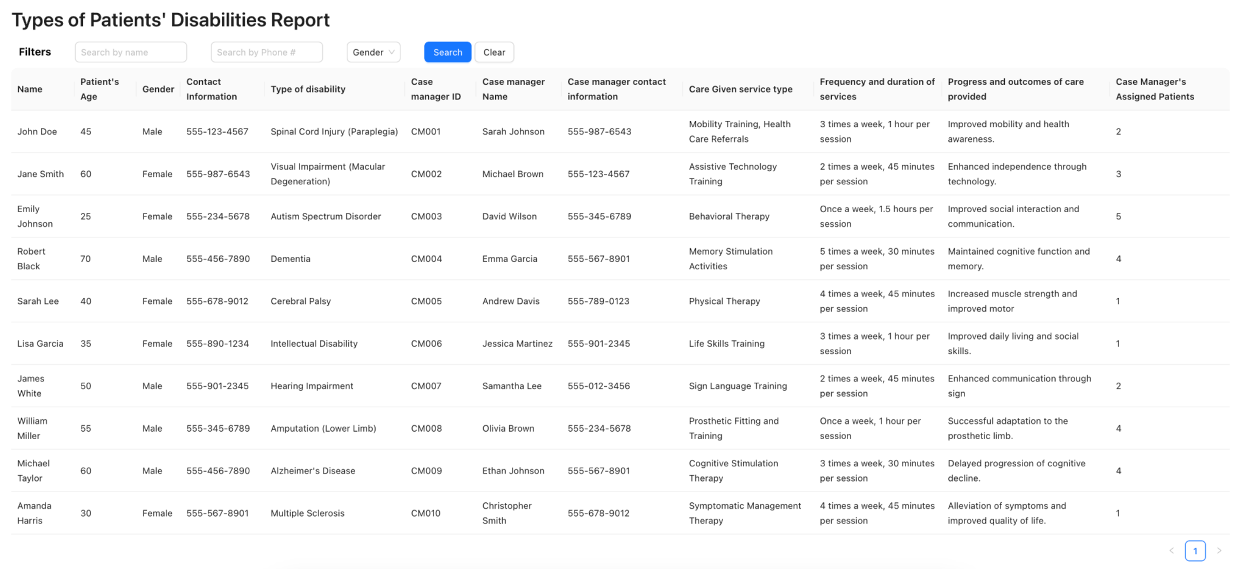


Case Manager Performance

| Case Manager Name | Department | Contact Number | Cases Assigned |
|---|---|---|---|
| Jane Smith | Pediatrics | 555-987-6543 | 3 |
| John Doe | Cardiology | 555-123-4567 | 3 |
| Margot Robie | Oncology | 555-937-6223 | 1 |
| Alex Johnson | Neurology | 555-555-5555 | 1 |
| Sophia Garcia | Dermatology | 555-666-7777 | 3 |
| Emily Davis | Oncology | 555-222-3333 | 1 |
| Michael Brown | Orthopedics | 555-444-5555 | 3 |
| William Wilson | Psychiatry | 555-777-8888 | 2 |
| Olivia Martinez | Endocrinology | 555-888-9999 | 1 |
| James Johnson | Gastroenterology | 555-999-0000 | 2 |

## Types of Individuals with Significant Disabilities Report

In order to help Agency Managers understand the number of patients with significant disabilities, the types of disabilities, and the rate of demand for related services, we provide this report. Given the service type information, frequency, and duration, and analyzing the number of patients requesting it, the Agency Managers can prioritize the development and expansion of the busiest and most demanded services.

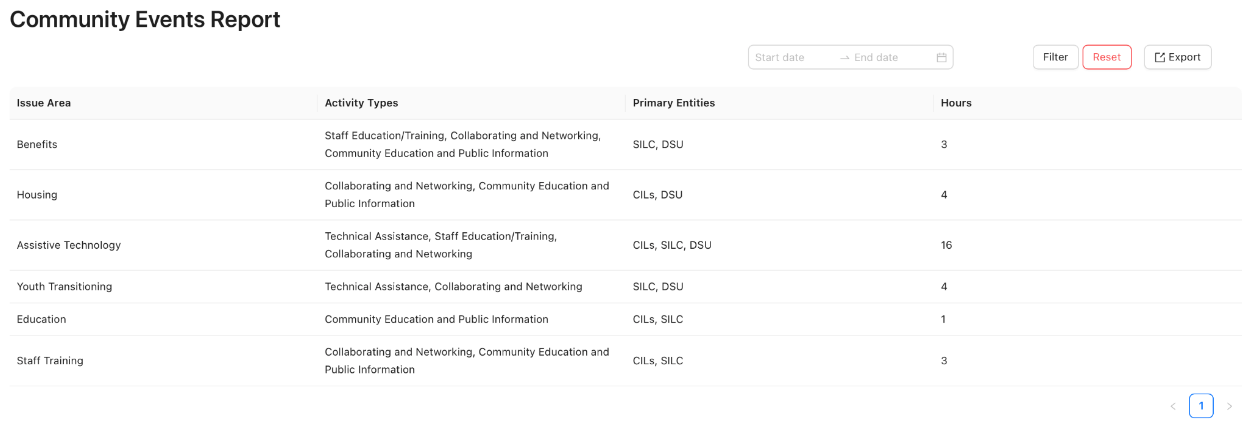


| Name | Patient's Age | Gender | Contact Information | Type of disability | Case manager ID | Case manager Name | Case manager contact information | Care Given service type | Frequency and duration of services | Progress and outcomes of care provided | Case Manager's Assigned Patients |
|---|---|---|---|---|---|---|---|---|---|---|---|
| John Doe | 45 | Male | 555-123-4567 | Spinal Cord Injury (Paraplegia) | CM001 | Sarah Johnson | 555-987-6543 | Mobility Training, Health Care Referrals | 3 times a week, 1 hour per session | Improved mobility and health awareness. | 2 |
| Jane Smith | 60 | Female | 555-987-6543 | Visual Impairment (Macular Degeneration) | CM002 | Michael Brown | 555-123-4567 | Assistive Technology Training | 2 times a week, 45 minutes per session | Enhanced independence through technology. | 3 |
| Emily Johnson | 25 | Female | 555-234-5678 | Autism Spectrum Disorder | CM003 | David Wilson | 555-345-6789 | Behavioral Therapy | Once a week, 1.5 hours per session | Improved social interaction and communication. | 5 |
| Robert Black | 70 | Male | 555-456-7890 | Dementia | CM004 | Emma Garcia | 555-567-8901 | Memory Stimulation Activities | 5 times a week, 30 minutes per session | Maintained cognitive function and memory. | 4 |
| Sarah Lee | 40 | Female | 555-678-9012 | Cerebral Palsy | CM005 | Andrew Davis | 555-789-0123 | Physical Therapy | 4 times a week, 45 minutes per session | Increased muscle strength and improved motor | 1 |
| Lisa Garcia | 35 | Female | 555-890-1234 | Intellectual Disability | CM006 | Jessica Martinez | 555-901-2345 | Life Skills Training | 3 times a week, 1 hour per session | Improved daily living and social skills. | 1 |
| James White | 50 | Male | 555-901-2345 | Hearing Impairment | CM007 | Samantha Lee | 555-012-3456 | Sign Language Training | 2 times a week, 45 minutes per session | Enhanced communication through sign | 2 |
| William Miller | 55 | Male | 555-345-6789 | Amputation (Lower Limb) | CM008 | Olivia Brown | 555-234-5678 | Prosthetic Fitting and Training | Once a week, 1 hour per session | Successful adaptation to the prosthetic limb. | 4 |
| Michael Taylor | 60 | Male | 555-456-7890 | Alzheimer's Disease | CM009 | Ethan Johnson | 555-567-8901 | Cognitive Stimulation Therapy | 3 times a week, 30 minutes per session | Delayed progression of cognitive decline. | 4 |
| Amanda Harris | 30 | Female | 555-567-8901 | Multiple Sclerosis | CM010 | Christopher Smith | 555-678-9012 | Symptomatic Management Therapy | 4 times a week, 45 minutes per session | Alleviation of symptoms and improved quality of life. | 1 |

https://github.com/himanshu98/digital-health-frontend/blob/master/src/modules/reporting/Patient/Patient.js

## Exporting Data

All reports used in the Federal Report, support the export feature. By clicking the button in the top right corner, the report data is saved as a .csv file. This way Agency Managers can not only effortlessly compile the Federal Report, but also keep a data snapshot for their records.



| Issue Area | Activity Types | Primary Entities | Hours |
|---|---|---|---|
| Benefits | Staff Education/Training, Collaborating and Networking, Community Education and Public Information | SILC, DSU | 3 |
| Housing | Collaborating and Networking, Community Education and Public Information | CILs, DSU | 4 |
| Assistive Technology | Technical Assistance, Staff Education/Training, Collaborating and Networking | CILs, SILC, DSU | 16 |
| Youth Transitioning | Technical Assistance, Collaborating and Networking | SILC, DSU | 4 |
| Education | Community Education and Public Information | CILs, SILC | 1 |
| Staff Training | Collaborating and Networking, Community Education and Public Information | CILs, SILC | 3 |

# Testing

## Test Objectives

- Functional Testing and Integration Testing are carried out on the Report APIs and Community Activity APIs. This focuses on validating the results of API implementation. identified based on requirements.
- Acceptance Testing is carried out on the Report and Community Activity module to validate the application's behaviors in UI based on scenarios identified from smart stories.

## Test Environments

**Functional and Integration testing**
Requires a Python environment. Integrated into deployment workflow. Requires the following installations:

- Pytest
  - REST APIs are developed using FastAPI which is Python based. Pytest is a Python framework that helps to create readable test cases to include simple and complex scenarios.
- Httpx
  - HTTP client for Python that can be effectively used with Pytest to perform functional and integration testing

**Acceptance testing**
Requires a react environment. Testing is performed individually in the dev/QA environment before a git commit. Requires the following installation:

- Cypress
  - Interactive tool to create end-to-end (e2e) workflow test cases with the web application based on React.
  - Create spec files to test workflows identified in the stories.

## Identified Features for Testing

**Build Community Events**

- CRUD Operations

**Build Reports**

- Service Provider report
- Community Events report
- Demographic report
- Case Manager report
- Funded services and Achievements report

## Test Procedure

### Test Execution

**Test Cases:**

For each requirement, API, or feature to be tested, the tester will identify the test cases. Once manual testing is completed. Test cases will be automated. Each test case will have the following:

- Setup procedure
- Action event
- Validating the results

The automation would give a PASS value for all accepted results in the test case. If the observed results are not equal to the expected results, a FAIL status is returned.

A bug should be created and attached to the project manager and involved developers.

**Test Implementation Order:**

- Functional & Integration Testing
  - Identify the API and data requirements for a feature. Create test cases for business logic testing and API data validation.
  - Integrate any test case developed to be part of the deployment build.
- Acceptance testing
  - Once the UI implementation is available. Identify test scenarios for the feature. Scenarios should include happy and sad paths.
  - Manually test the scenarios. Once tested without bugs, Create a spec file to automate the test scenario.
  - Include the spec file in the project. Before any changes are pushed, the local execution of all spec files will validate any breakage of functionality and can be integrated as part of unit testing for the developers.

## Test Examples

### Functional & Integration Testing

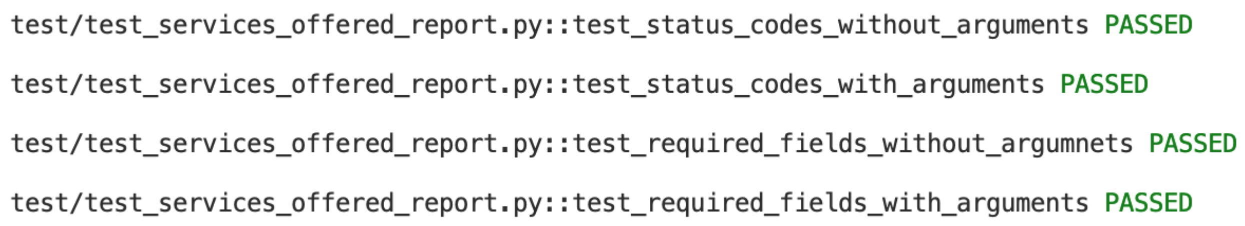
#### Feature - Service Provider Report

**Test Cases**

- Test status codes without arguments:
  - Test valid response codes and error messages that are acceptable for the API to deploy
  - Test mandatory fields exist in the responses of a get request without any arguments.

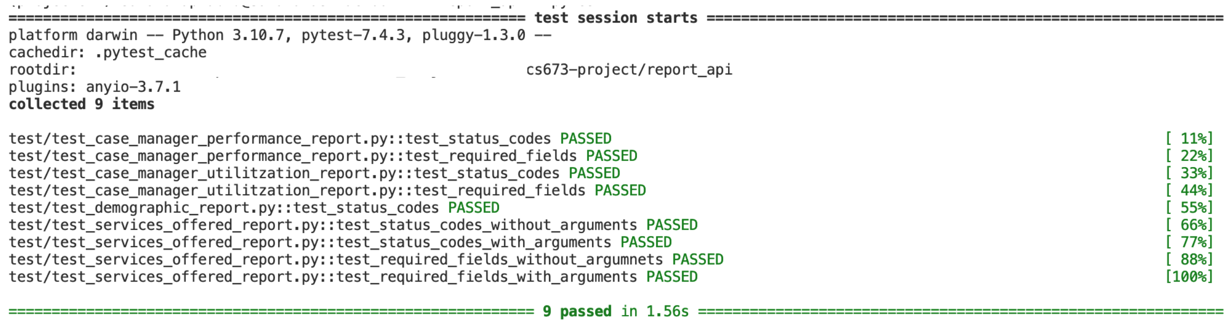
- Test status code with arguments:
    - Test valid response codes and error messages that are acceptable for the API to deploy
    - Test mandatory fields exist in the responses of a get request with arguments.

**Test Result**

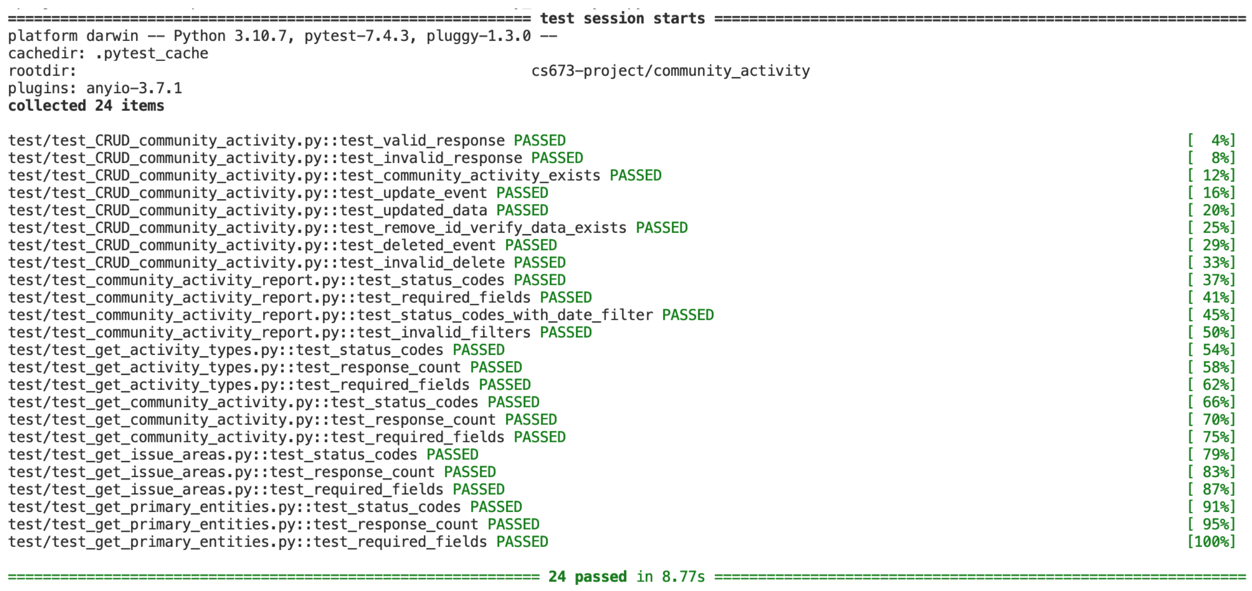


```
test/test_services_offered_report.py::test_status_codes_without_arguments PASSED

test/test_services_offered_report.py::test_status_codes_with_arguments PASSED

test/test_services_offered_report.py::test_required_fields_without_argumnets PASSED

test/test_services_offered_report.py::test_required_fields_with_arguments PASSED
```

**Existing Test Results**



```
============================================================ test session starts =============================================================
platform darwin -- Python 3.10.7, pytest-7.4.3, pluggy-1.3.0 --
cachedir: .pytest_cache
rootdir:                                                    cs673-project/report_api
plugins: anyio-3.7.1
collected 9 items

test/test_case_manager_performance_report.py::test_status_codes PASSED                                                                  [ 11%]
test/test_case_manager_performance_report.py::test_required_fields PASSED                                                               [ 22%]
test/test_case_manager_utilitzation_report.py::test_status_codes PASSED                                                                 [ 33%]
test/test_case_manager_utilitzation_report.py::test_required_fields PASSED                                                              [ 44%]
test/test_demographic_report.py::test_status_codes PASSED                                                                               [ 55%]
test/test_services_offered_report.py::test_status_codes_without_arguments PASSED                                                        [ 66%]
test/test_services_offered_report.py::test_status_codes_with_arguments PASSED                                                           [ 77%]
test/test_services_offered_report.py::test_required_fields_without_argumnets PASSED                                                     [ 88%]
test/test_services_offered_report.py::test_required_fields_with_arguments PASSED                                                        [100%]

============================================================= 9 passed in 1.56s ==============================================================
```



```
============================================================ test session starts =============================================================
platform darwin -- Python 3.10.7, pytest-7.4.3, pluggy-1.3.0 --
cachedir: .pytest_cache
rootdir:                                                    cs673-project/community_activity
plugins: anyio-3.7.1
collected 24 items

test/test_CRUD_community_activity.py::test_valid_response PASSED                                                                        [  4%]
test/test_CRUD_community_activity.py::test_invalid_response PASSED                                                                      [  8%]
test/test_CRUD_community_activity.py::test_community_activity_exists PASSED                                                             [ 12%]
test/test_CRUD_community_activity.py::test_update_event PASSED                                                                          [ 16%]
test/test_CRUD_community_activity.py::test_updated_data PASSED                                                                          [ 20%]
test/test_CRUD_community_activity.py::test_remove_id_verify_data_exists PASSED                                                          [ 25%]
test/test_CRUD_community_activity.py::test_deleted_event PASSED                                                                         [ 29%]
test/test_CRUD_community_activity.py::test_invalid_delete PASSED                                                                        [ 33%]
test/test_community_activity_report.py::test_status_codes PASSED                                                                        [ 37%]
test/test_community_activity_report.py::test_required_fields PASSED                                                                     [ 41%]
test/test_community_activity_report.py::test_status_codes_with_date_filter PASSED                                                       [ 45%]
test/test_community_activity_report.py::test_invalid_filters PASSED                                                                     [ 50%]
test/test_get_activity_types.py::test_status_codes PASSED                                                                               [ 54%]
test/test_get_activity_types.py::test_response_count PASSED                                                                             [ 58%]
test/test_get_activity_types.py::test_required_fields PASSED                                                                            [ 62%]
test/test_get_community_activity.py::test_status_codes PASSED                                                                           [ 66%]
test/test_get_community_activity.py::test_response_count PASSED                                                                         [ 70%]
test/test_get_community_activity.py::test_required_fields PASSED                                                                        [ 75%]
test/test_get_issue_areas.py::test_status_codes PASSED                                                                                  [ 79%]
test/test_get_issue_areas.py::test_response_count PASSED                                                                                [ 83%]
test/test_get_issue_areas.py::test_required_fields PASSED                                                                               [ 87%]
test/test_get_primary_entities.py::test_status_codes PASSED                                                                             [ 91%]
test/test_get_primary_entities.py::test_response_count PASSED                                                                           [ 95%]
test/test_get_primary_entities.py::test_required_fields PASSED                                                                          [100%]

============================================================= 24 passed in 8.77s =============================================================
```

## Acceptance Testing

### Community Events

**Test Case 1**: **Test required fields in the community event**
**Given**: The user is on the reporting page.
**When**: The user navigates to the Community Events section.

**Then**: The Community Events page should display the required table columns, including "Community Activity Name", "Hours", "Objectives", "Outcomes", "Issue Area Name", "Primary Entities", "Activity Types", and "Operation."

**Test Case 2: Test adding community event**
**Given**: The user is on the reporting page.
**When**: The user navigates to the Community Events section and clicks the "Add" button.
**Then**:

- The form for adding a new community event should be visible.
- The user fills in the required fields such as "UI_Testing" for "Event Name", selects an issue area, checks relevant checkboxes, and provides values for "Hours", "Objectives", and "Outcomes."
- The user clicks the "Save" button.
- A success notification should appear with the message "Community Event added successfully."
- The newly added event with the name "UI_Testing" should be visible in the table.

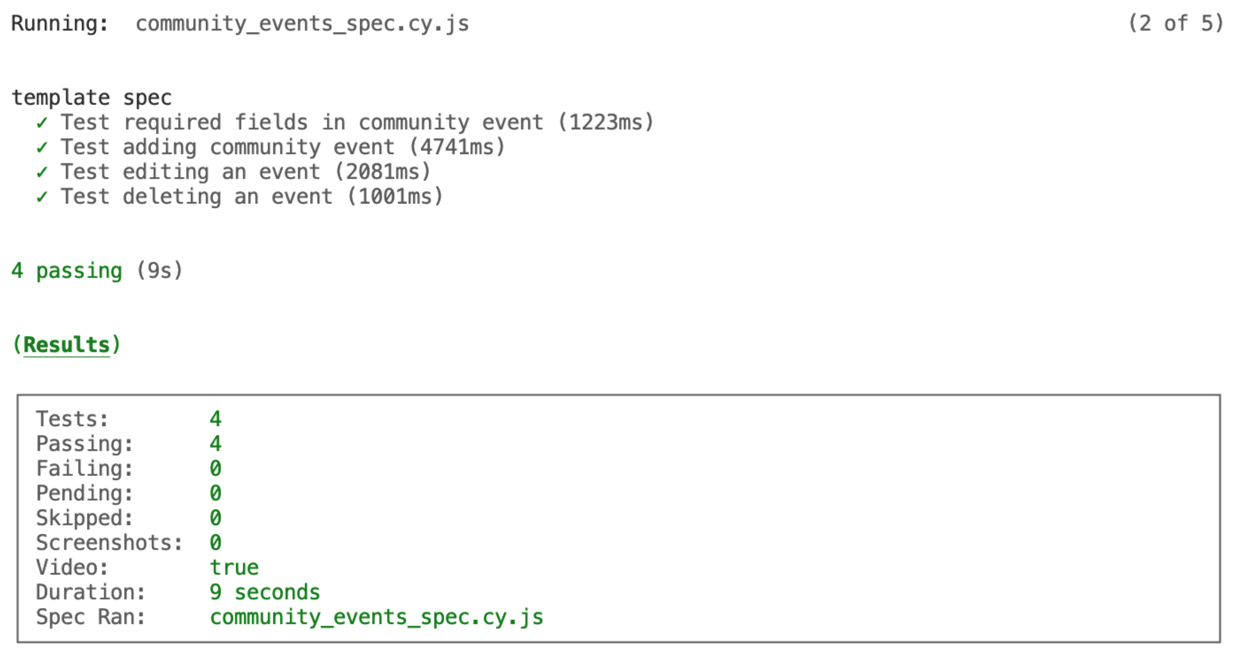
**Test Case 3: Test editing an event**
**Given**: The user is on the reporting page.
**When**: The user navigates to the Community Events section, finds the "UI_Testing" event, and clicks the "Edit" button.
**Then**:

- The edit form should be visible with the existing values filled in.
- The user modifies the values, such as changing the event name to "UI_Test", updating the hours to 7, and changing the objectives.
- The user clicks the "Save" button.
- A success notification should appear with the message "Record updated successfully".
- The edited event with the name "UI_Test" and updated details should be visible in the table.

**Test Case 4: Test deleting an event**
**Given**: The user is on the reporting page.
**When**: The user navigates to the Community Events section, finds the "UI_Test" event, and clicks the delete button.
**Then**:

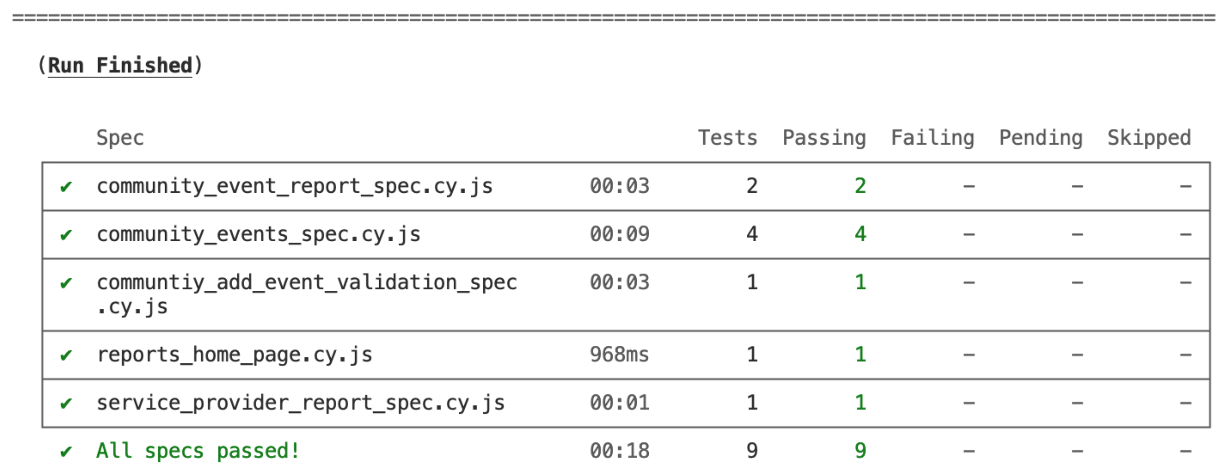
- A confirmation dialog should appear.
- The user confirms the deletion by clicking the confirmation button.
- A success notification should appear with the message "Record deleted successfully."
- The deleted event with the name "UI_Test" should no longer be visible in the table.

**Test Result**



```
Running:  community_events_spec.cy.js                                               (2 of 5)


template spec
  ✓ Test required fields in community event (1223ms)
  ✓ Test adding community event (4741ms)
  ✓ Test editing an event (2081ms)
  ✓ Test deleting an event (1001ms)


4 passing (9s)


(Results)

┌────────────────────────────────────────────────────────────────────────────────────────┐
│ Tests:        4                                                                        │
│ Passing:      4                                                                        │
│ Failing:      0                                                                        │
│ Pending:      0                                                                        │
│ Skipped:      0                                                                        │
│ Screenshots:  0                                                                        │
│ Video:        true                                                                     │
│ Duration:     9 seconds                                                                │
│ Spec Ran:     community_events_spec.cy.js                                              │
└────────────────────────────────────────────────────────────────────────────────────────┘
```

### Existing Test Results



```
====================================================================================================

  (Run Finished)


       Spec                                              Tests  Passing  Failing  Pending  Skipped
  ┌────────────────────────────────────────────────────────────────────────────────────────────────┐
  │ ✔  community_event_report_spec.cy.js        00:03        2        2        -        -        - │
  ├────────────────────────────────────────────────────────────────────────────────────────────────┤
  │ ✔  community_events_spec.cy.js              00:09        4        4        -        -        - │
  ├────────────────────────────────────────────────────────────────────────────────────────────────┤
  │ ✔  communtiy_add_event_validation_spec      00:03        1        1        -        -        - │
  │    .cy.js                                                                                      │
  ├────────────────────────────────────────────────────────────────────────────────────────────────┤
  │ ✔  reports_home_page.cy.js                  968ms        1        1        -        -        - │
  ├────────────────────────────────────────────────────────────────────────────────────────────────┤
  │ ✔  service_provider_report_spec.cy.js       00:01        1        1        -        -        - │
  └────────────────────────────────────────────────────────────────────────────────────────────────┘
    ✔  All specs passed!                        00:18        9        9        -        -        -
```

## Next steps in Testing

- Include more sad paths in acceptance testing to make the product more robust.
- Include acceptance testing in the CI pipeline to make the process automated.

# Work Breakdown

All aspects and main design decisions of the project, including requirements collection, definition of our persona, development of user stories and the product vision, and sharing implementation ideas, were a team effort. Each of the team members equally participated in and contributed to the project design. Nevertheless, each of us also had unique responsibilities that are described in the following sections.

## Dhananjay Jagdish Dubey – Data Architect

Dhananjay was primarily responsible for the database-related tasks such as:

- Database design
- Setting up and providing connection to the database instance
- Preparing queries necessary for our use cases.

## Mugunthan Krishnan – Microservices Engineer

Mugunthan was responsible for developing the backend portion of our web service. Mugunthan's tasks included but were not limited to:

- Analyzing APIs exposed by the other teams
- Requesting API changes, if needed
- Pulling, processing, and adapting for our needs the data from external APIs

## Alexander Morozov – Product Manager

Alexander's duties included the management of the project:

- Development and delivery of project-related assignments, presentations, and reports
- Setting up effective communication within our team and being the main point of communication with other teams during the entire product development process
- Providing resources for the project, distributing the work, and organizing the workflow

## Saravana Prabhu Ramasamy – Quality and DevOps Engineer

Saravana worked on delivery and ensuring the quality of our project:

- Setting up automatic building and deployment of the backend portion of the project
- Researching, preparing, and executing the test plan
- Ensuring the overall quality and usability of the project

## Ankush Ranapure – UI/UX Engineer

Ankush took care of the entire frontend portion of the project:

- Designing and developing UI
- Ensuring high quality of user experience and overall usability of the webpages

- Collaborating with other teams to ensure consistency across the product
- Deployment of the react application to Netlify