Project Overview

The **Agent Call Tracker** is a web-based application and **API** that allows users to track the number of connected and not connected calls handled by agents over a given time period. Users can input specific timestamps, and the application fetches relevant data from a backend API to display in a tabular format.

The project is built with a simple HTML/CSS frontend and uses JavaScript to call a backend API that provides the call data for agents. This project is a good fit for scenarios like call centers or customer service departments where managers need to monitor agent performance.

Key Features

- **Timestamp-based Query**: Users can query call data by providing a start and end timestamp.
- **Display Agent Data**: Displays the agent's name, email, and call details (connected and not connected calls).

Technology Stack

- **Frontend**:
  - HTML5
  - CSS3 (for styling)
  - JavaScript (for dynamic data fetch)
- **Backend**:
  - Node.js with Express (for the API)
- **Development Tools**:
  - VS Code or any other IDE for development.
  - npm (for package management).
- **Mongo bd atlas for database**

Setup Instructions

Open terminal in vs code and Install the following packages:

- npm install express
- npm install node
- npm Install mongodb
- npm install express mongodb cors body-parser

Database structure

Agent collections

```
{
"_id": "varchar", [primary key]
"name": "string",
"email": "varchar",
"password": "varchar"
}
```

call _logs collections

```
{
"_id": "varchar",
"agent_id": "varchar", [ reference key]
"timestamp": number,
"call_status": "string"
}
```

Detailed Workflow

1. **User Interaction**:
    - The user enters two Unix timestamps (start and end).
    - Upon clicking the "**Get Call Data**" button, the JavaScript fetches data from the backend API using these timestamps.
2. **Backend API**:

The frontend makes a GET request to the backend API

The backend processes this request, queries the database (or other data sources), and returns a JSON array containing agent information like:

- `name`: Agent's name.
- `connected_calls`: Number of connected calls.
- `not_connected_calls`: Number of not connected calls.
- `mail`: Agent's email.

3. **Rendering Data**:
    - Once the data is fetched successfully, JavaScript dynamically updates the table in the frontend with the fetched information.

Example Input

- **Start Timestamp**: `1630454400`
- **End Timestamp**: `1630540800`.

Set up

1 Open the program file in vs code

2 Install all the packages mentioned above:
- npm install express
- npm install node
- npm Install mongodb
- npm install express mongodb cors body-parser

3 In order to run the program use:

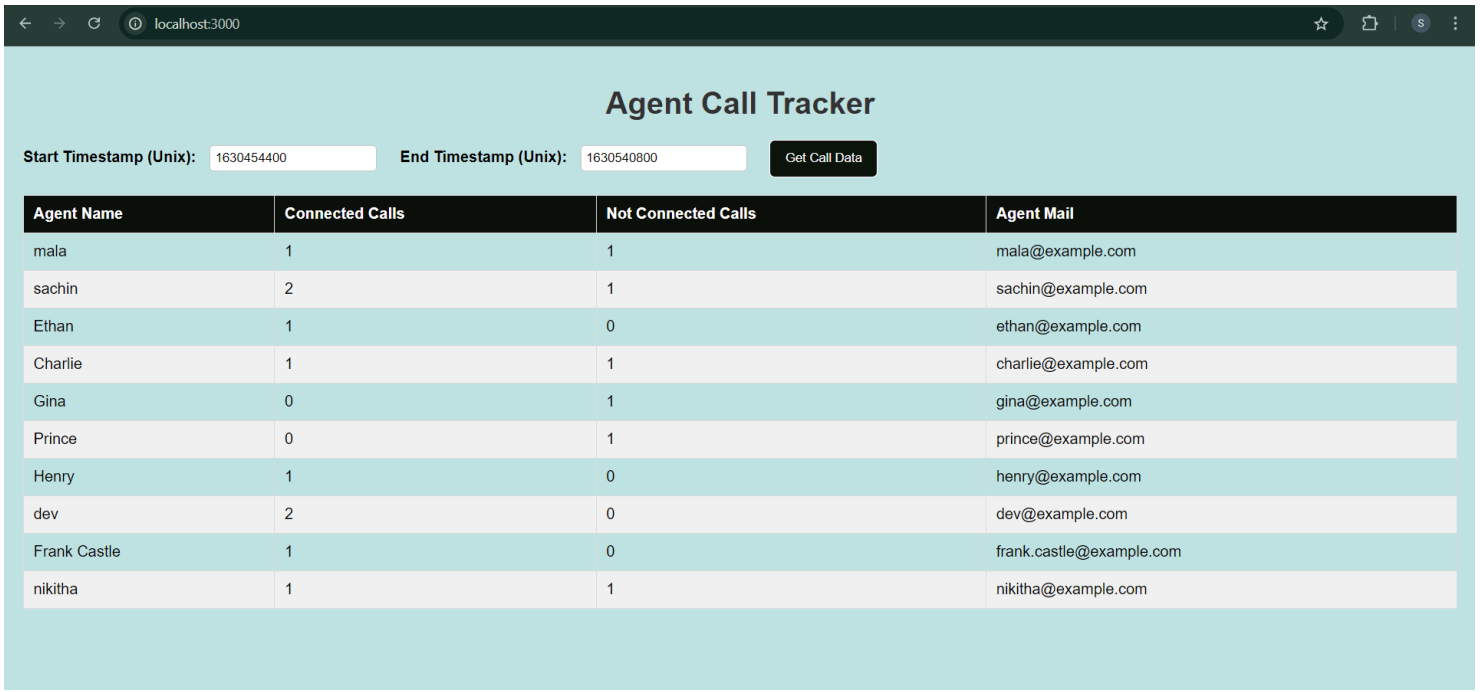
- [node app.js] in terminal

4 Lastly open any browser and type this url for the api to work :

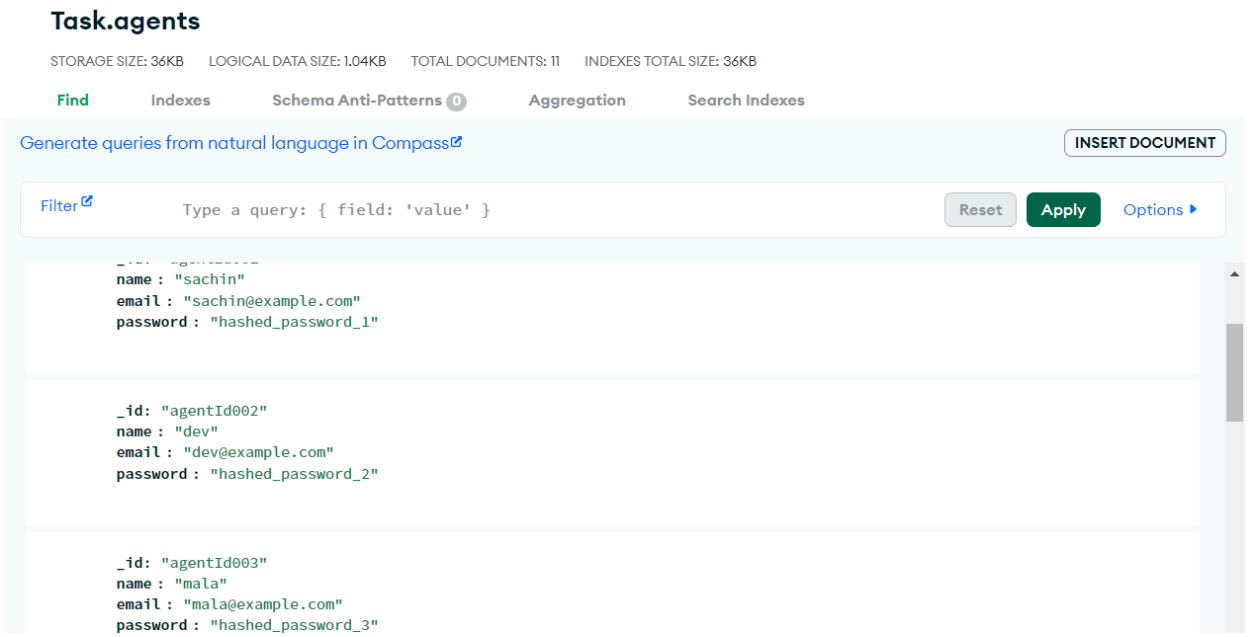
- http://localhost:3000/

5 Finally click the button **“get call data”** to see the detailed output.

Output


localhost:3000
Agent Call Tracker
Start Timestamp (Unix):
1630454400
End Timestamp (Unix):
1630540800
Get Call Data
Agent Name
Connected Calls
Not Connected Calls
Agent Mail
mala
1
1
mala@example.com
sachin
2
1
sachin@example.com
Ethan
1
0
ethan@example.com
Charlie
1
1
charlie@example.com
Gina
0
1
gina@example.com
Prince
0
1
prince@example.com
Henry
1
0
henry@example.com
dev
2
0
dev@example.com
Frank Castle
1
0
frank.castle@example.com
nikitha
1
1
nikitha@example.com

Mongo db atlas (database):



Task.call_logs

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 1.47KB TOTAL DOCUMENTS: 17 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns 0 Aggregation Search Indexes

Generate queries from natural language in Compass INSERT DOCUMENT

Filter Type a query: { field: 'value' } Reset Apply Options

QUERY RESULTS: 1-17 OF 17

```
_id: ObjectId('66faa4e1d55fd6936e6c8d2f')
```

```
_id: "callId001"
agent_id : "agentId001"
timestamp : 1630454400
call_status : "connected"
```

```
_id: "callId002"
agent_id : "agentId001"
timestamp : 1630458000
call_status : "not connected"
```