

---

# Deep-Sea Developers



## Project Documentation and Final Report

### Team Members

Name	Email
Sunny Ali	hassanli@uw.edu
Makai Martinez	mlm1738@uw.edu
Miguel Ramos	ramosmig@uw.edu
Veasna Bun	veasnab@uw.edu

### Prepared for

Course	TCSS 445 Database System Design
Date	11/21/2023

## Introduction

Our project stems from a desire to find a meaningful application of our computer science training towards the betterment of the natural world. Oceanographers of the 21st century use computer technology to monitor hundreds of thousands of research sites across the world's oceans and lakes. These electronic instruments are called CTD's (*conductivity, temperature, depth*), but accessing their datasets reveals a number of issues. One is that the frontend interfaces are notably unreliable. There is a course at the University of Washington Tacoma that relies on these frontend applications for student labs, however they are so inaccessible that accessing the public data is usually considered optional. Another issue is that, although efforts have been made in this regard, the data is scattered across countless government and non-government agencies. The final problem identified was the sheer number of data points that needed to be handled.

As students of data and information, we felt well suited to tackle these problems on behalf of our colleagues working to maintain the health of our planet's oceans.

## Objectives and Scope of the Project

Our resources are slim. We have limited time to complete the project and only a working knowledge of web application programming. What we do have is an entire quarter to study databases in both implementation and theory. So, our objective is to present a highly scalable relation schema that may serve as a solution to two of the three major problems facing oceanographers trying to access CTD data. These problems are:

1. Unreliable frontend applications for accessing CTD data.
2. Scattered data sets across countless organizations.
3. Storing millions of diverse data points in a format that makes them easy to access.

Because we are not web programmers, we will not be able to make a meaningful improvement to the first mentioned problem. Our final deliverable will be a relation schema filled with bogus data that can be accessed through a web application. This web application will store predefined sql queries that can be used to demo the types of datasets that can be composed using our relation schema. Our hope is that a researcher may find our schema useful as a format for compiling the datasets of various organizations.

## Related or Existing Solutions

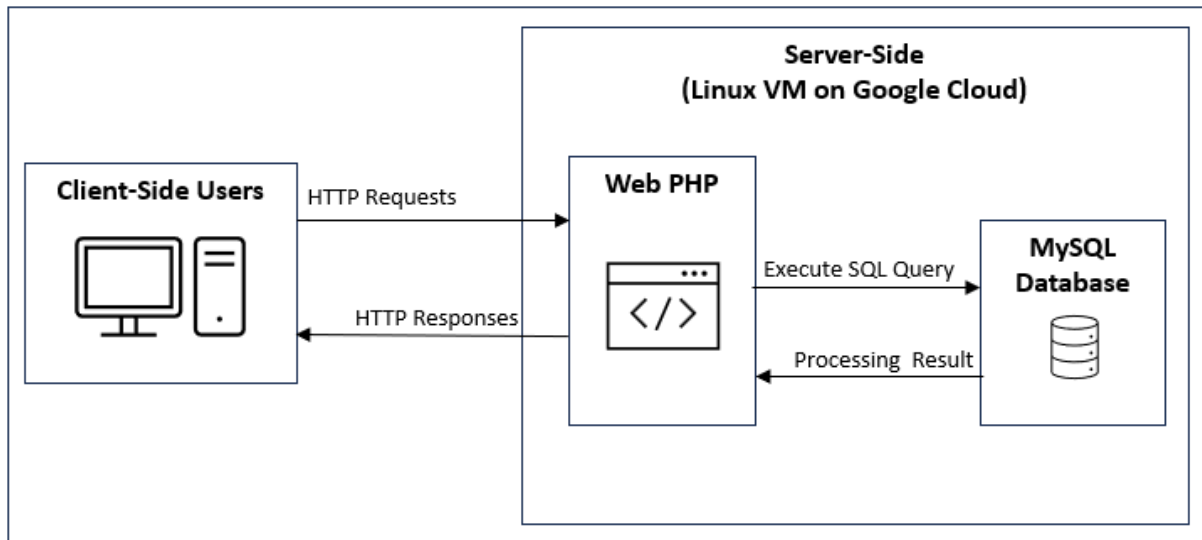
The Ocean Observatories Initiative is a oceanographic project to build and maintain the infrastructure necessary to make available realtime CTD data on research sites across the continent. They do incredible work. However, their data portal is consistently malfunctioning. It can be viewed using this link:

<https://app.interactiveoceans.washington.edu/>

This data portal includes many different features, including an interactive map and data visualizers, but we were hoping to create a more simple, text based interface for fetching data for a desired location. Our aim is not to interpret data, but to create a tool that will allow users to independently upload data from their own organization and download data from other organizations, leading to a collaboration amongst disjoint oceanographers.

## Architecture Overview

In our project, we utilize the Google Cloud Platform to deploy a web interface and a MySQL database. We leverage a Linux Virtual Machine (VM) to host our setup. The web interface is constructed using PHP and interacts with the MySQL database.



**Figure 1:** Architecture diagram

### Communication Flow:

1. **Client-Side Users:** Initiate an HTTP request to the server.
2. **Server-Side Processing (Linux VM on Google Cloud):** Processes relevant PHP scripts responsible for both data processing and server-side logic.
3. **Web PHP:** Establishes a connection within the PHP scripts with the MySQL database residing on the same Google Cloud Linux VM.
4. **Execute SQL Queries:** Depending on client-user interaction, queries are executed within the PHP scripts, primarily focusing on data retrieval operations.
5. **Process Result:** The PHP script processes the results retrieved from the database, generating content for the server response.
6. **Server-Side Response (Linux VM on Google Cloud):** The web server promptly sends the generated HTTP response back to the client.

### Key Technologies:

- Google Cloud Platform
- Linux Virtual Machine
- PHP for server-side scripting
- MySQL for database management

## Conclusions with Contribution

In summary, our project represents a concerted effort to address critical challenges faced by oceanographers in accessing and managing CTD data. Despite our limited resources and expertise, we have crafted a solution centered around a scalable relation schema deployed on the Google Cloud Platform. Our web interface, hosted on a Linux VM, interacts seamlessly with a MySQL database to offer a tool for collaborative data sharing among oceanographers.

As we reflect on the journey, we would like to express our sincere gratitude to our professor E. Al-Masri, and the dedicated members of our team whose collective efforts have made this project possible. Each team member has played a crucial role, contributing their unique skills and perspectives. The following is a glimpse into the distinctive contributions of each team member:

1. **Makai Martinez:** Demonstrated creativity in contributing to the project. In addition to documenting the cover page, Makai brought an artistic flair to our project by creating our logo and selecting our team name. Additionally, Makai contributes to the web interface and its documentation.
2. **Miguel Ramos:** Took charge of implementing the web interface and successfully deploying it on the Google Cloud Platform. Miguel also played a key role in the project's final documentation, covering aspects such as the introduction, objectives, and related solutions.
3. **Sunny Ali:** Played a role in the project by revising the Relation Schema Diagram, ensuring the cloud infrastructure operated efficiently and met the project's needs. Worked on comprehensive and clear documentation for users and technical staff. Established the functional and aesthetic specifications for the web interface.
4. **Veasna Bun:** Played a role in designing the ER diagram, ensuring its normalization up to BCNF, and crafting the SQL script. Veasna participated in the project's final documentation, covering aspects such as the architecture overview, conclusions, and future work. Additionally, Veasna contributes to the web interface and its documentation.

## Future Works and References

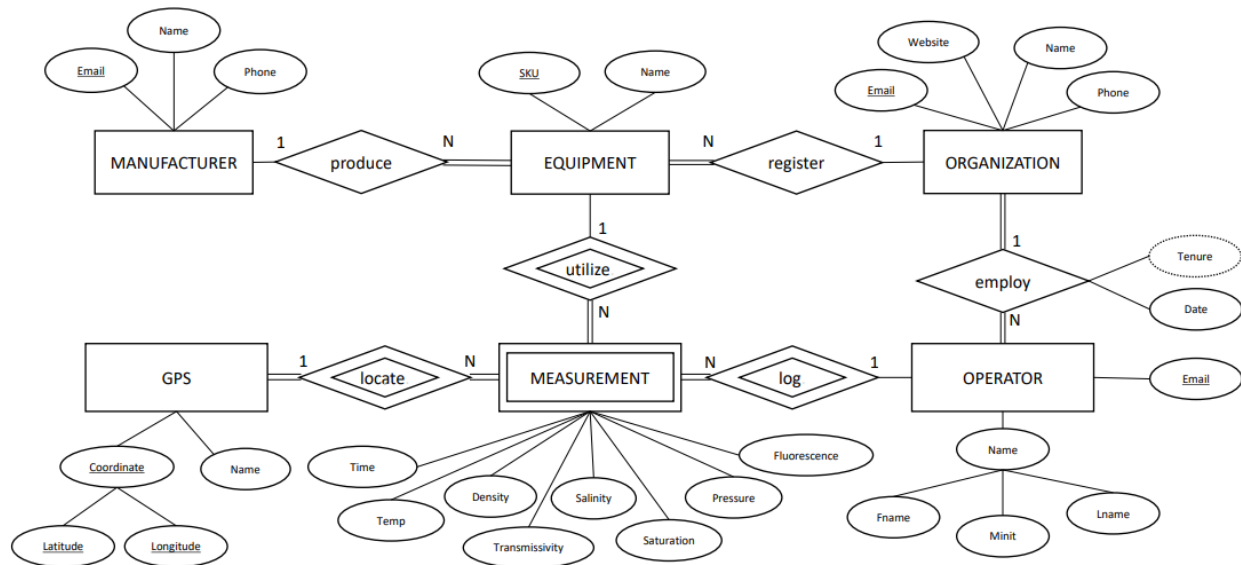
Looking ahead, our focus shifts towards shaping the future of our project. In the following list are the key ideas that we believe will make our web page more interactive and user-friendly:

1. **Integration of Google Maps API:** Explore the integration of the Google Maps API into the web interface. This addition would enhance the user experience by providing an interactive map component, allowing oceanographers to visualize geographical locations associated with the CTD data.
2. **Extension of Relational Schema:** Extend the existing relational schema to incorporate an "ADDRESS" relation. This addition would facilitate the mapping of geographical information, aligning with the potential integration of the Google Maps API.

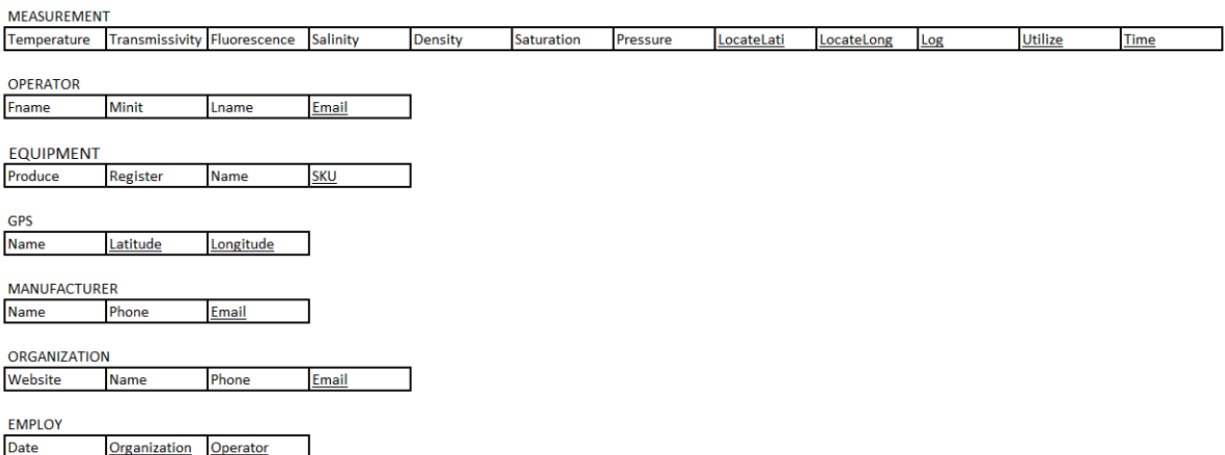
In references of the envisioning these future works, we recognize the potential for significant enhancement to our project. Although time constraints prevented the immediate implementation of certain

features, the inclusion of the Google Maps API and the extension of the relational schema to accommodate address-related data stand out as valuable directions for future development.

## Appendix A: ER Diagram



**Figure 2:** ER Diagram for CTD



**Figure 3:** ER Diagram to Relational Mapping for CTD

## Appendix B: Normalization Proof to BCNF

### BCNF Normalization (1 of 2)

**Prove:** CTD database is in BCNF. A database schema is in BCNF if each relation schema is in BCNF.

#### MANUFACTURER

Name	Phone	<u>Email</u>
------	-------	--------------

**FD1:**  $\{Email\} \rightarrow \{Name, Phone\}$

$\therefore$  Email is a candidate key for MANUFACTURER as it is needed to identify Name, Phone.

**Proof:** Since FD1 has the determinant as a candidate key, this relation is in BCNF.

#### ORGANIZATION

Website	Name	Phone	<u>Email</u>
---------	------	-------	--------------

**FD2:**  $\{Email\} \rightarrow \{Website, Name, Phone\}$

$\therefore$  Email is a candidate key for ORGANIZATION as it is needed to identify Website, Name, Phone.

**Proof:** Since FD2 has the determinant as a candidate key, this relation is in BCNF.

#### OPERATOR

Fname	Minit	Lname	<u>Email</u>
-------	-------	-------	--------------

**FD3:**  $\{Email\} \rightarrow \{Fname, Minit, Lname\}$

$\therefore$  Email is a candidate key for OPERATOR as it is needed to identify Fname, Minit, Lname.

**Proof:** Since FD3 has the determinant as a candidate key, this relation is in BCNF.

#### GPS

Name	<u>Latitude</u>	<u>Longitude</u>
------	-----------------	------------------

**FD4:**  $\{Latitude, Longitude\} \rightarrow \{Name\}$

$\therefore$  Latitude and Longitude is a candidate key for GPS as it is needed to identify Name.

**Proof:** Since FD4 has the determinant as a candidate key, this relation is in BCNF.

## BCNF Normalization (2 of 2)

### EQUIPMENT

Produce	Register	Name	<u>SKU</u>
---------	----------	------	------------

**FD5:**  $\{SKU\} \rightarrow \{Produce, Register, Name\}$

$\therefore$  SKU is a candidate key for EQUIPMENT as it is needed to identify Produce, Register, Name.

**Proof:** Since FD5 has the determinant as a candidate key, this relation is in BCNF.

### EMPLOY

Date	<u>Organization</u>	<u>Operator</u>
------	---------------------	-----------------

**FD6:**  $\{Organization, Operator\} \rightarrow \{Date\}$

$\therefore$  Organization, Operator is a candidate key for EMPLOY as it is needed to identify Date.

**Proof:** Since FD6 has the determinant as a candidate key, this relation is in BCNF.

### MEASUREMENT

Temperature	Transmissivity	Fluorescence	Salinity	Density	Saturation
Pressure	<u>LocateLati</u>	<u>LocateLong</u>	<u>Log</u>	<u>Utilize</u>	<u>Time</u>

**FD7:**  $\{LocateLati, LocateLong, Log, Utilize, Time\}$

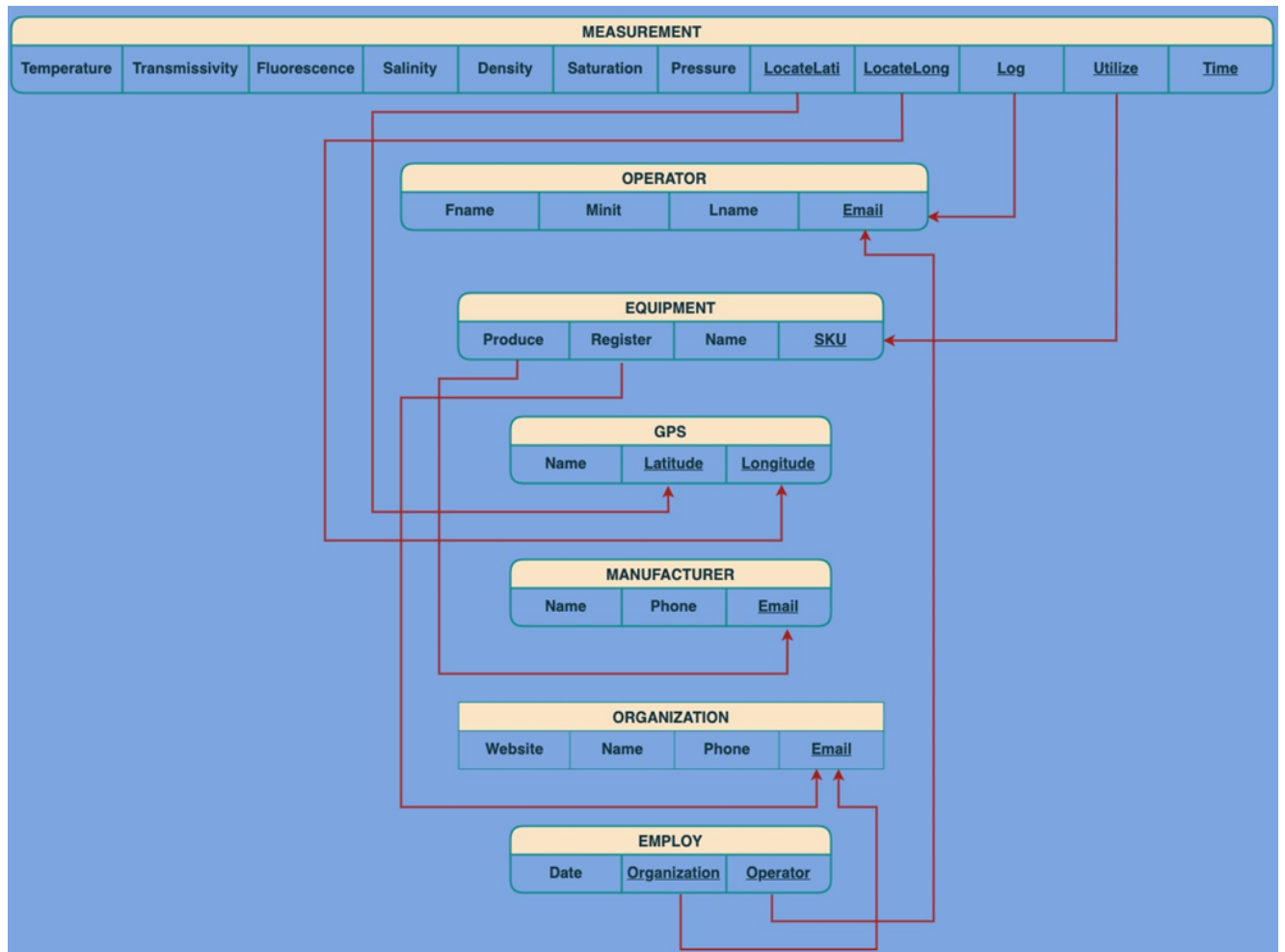
$\rightarrow \{Temperature, Transmissivity, Fluorescence, Salinity, Density, Saturation, Pressure\}$

$\therefore$  LocateLati, LocateLong, Log, Utilize, Time is a candidate key for MEASUREMENT as it is needed to identify Temperature, Transmissivity, Fluorescence, Salinity, Density, Saturation, and Pressure.

**Proof:** Since FD7 has the determinant as a candidate key, this relation is in BCNF.

**Conclusion:** Since all relations are in BCNF, the database CTD is in BCNF.

## Appendix C: Relational Schema Diagram



**Figure 4:** Boy-Codd Normalized Relational Schema



## Appendix D: Screenshots of functional SQL Queries

```
// This SQL query retrieves information about an operator and their employment details,
// including the organization name, operator's full name, employment start date, and
// the number of years they have been employed.
// It JOIN three relation: ORGANIZATION, EMPLOY, AND OPERATOR.
// Using the foreign key from the EMPLOY relation to their respect PRIMARY KEY in the ORGANIZATION and OPERATOR relation.
// The number of year an operator of employment is calculated by YEAR(CURDATE()) - YEAR(E.Edate).
// The recieve data is base on the select operator name,
// where the user have the option to select: WHERE OP.Email = '{$_GET['op']}'
$sql = "SELECT O.Oname AS 'Organization',
        CONCAT(OP.Fname, ' ', COALESCE(OP.Minit, ''), ' ', OP.Lname) AS 'Operator',
        E.Edate AS 'Date',
        YEAR(CURDATE()) - YEAR(E.Edate) AS 'Year'
        FROM EMPLOY E
        JOIN ORGANIZATION O ON O.Email = E.Organization
        JOIN OPERATOR OP ON E.Operator = OP.Email
        WHERE OP.Email = '{$_GET['op']}'";
```

Figure 5: SQL Query | operator.php

```
// This SQL query retrieve measurement data and associated GPS information based on a specified location name.
// It join two relation: MEASUREMENT and GPS. Using the foreign key in MEASUREMENT relation {LocateLati, LocateLong}
// which map to the primary key in the GPS relation {Latitude, Longitude}.
// The recieve data is base on the selected location name,
// where the user have the option to select: WHERE G.Gname = '{$_GET['loc']}'
$sql = "SELECT M.LocateLati, M.LocateLong, M.Mtime,
        M.temperature, M.transmissivity, M.salinity, M.saturation,
        M.florescence, M.density, M.pressure, G.Gname
        FROM MEASUREMENT M
        JOIN GPS G ON M.LocateLati = G.Latitude AND M.LocateLong = G.Longitude
        WHERE G.Gname = '{$_GET['loc']}'";
```

Figure 6: SQL Query | location.php

```
// This SQL query retrieves information about equipment and their associated measurement details.
// It calculates the average values of various measurements for a specific equipment based on the equipment SKU.
// The measurements include temperature, transmissivity, salinity, saturation, florescence, density, and pressure.
// The query uses a LEFT JOIN between the EQUIPMENT and MEASUREMENT tables, linking them via the Utilize and SKU columns, respectively.
// The COALESCE function is used to handle potential NULL values in the average calculations and replaces them with the string 'NULL'.
// The result is grouped by the equipment SKU.
// Users can specify the equipment SKU through the GET parameter {$_GET['eq']}.
$sql = "SELECT E.Ename,
        COALESCE(AVG(M.temperature), 'NULL') AS 'avgtemp',
        COALESCE(AVG(M.transmissivity), 'NULL') AS 'avgtran',
        COALESCE(AVG(M.salinity), 'NULL') AS 'avgsal',
        COALESCE(AVG(M.saturation), 'NULL') AS 'avgsat',
        COALESCE(AVG(M.florescence), 'NULL') AS 'avgflo',
        COALESCE(AVG(M.density), 'NULL') AS 'avgden',
        COALESCE(AVG(M.pressure), 'NULL') AS 'avgpre'
        FROM EQUIPMENT E
        LEFT JOIN MEASUREMENT M ON M.Utilize = E.SKU
        WHERE E.SKU = '{$_GET['eq']}'
        GROUP BY E.SKU";
```

Figure 7: SQL Query | equipment.php

```
// This SQL query retrieves information about an organization and the number of operators employed by the organization.
// It includes the organization name, website, phone, email, and the count of operators.
// The query utilizes a LEFT JOIN between the ORGANIZATION and EMPLOY tables, linking them via the Email and Organization columns, respectively.
// The COALESCE function is employed to handle potential NULL values in the organization's website, phone, and email, replacing them with the string 'NULL'.
// The result is grouped by the organization's email.
// Users can specify the organization name through the GET parameter {$_GET['org']}.
$sql = "SELECT O.Oname, COALESCE(O.Website, 'NULL') AS Website, COALESCE(O.Phone, 'NULL') AS Phone,
        COALESCE(O.Email, 'NULL') AS Email, COUNT(E.Operator) AS 'Count'
        FROM ORGANIZATION O
        LEFT JOIN EMPLOY E ON O.Email = E.Organization
        WHERE O.Oname = '{$_GET['org']}'
        GROUP BY O.Email";
```

Figure 8: SQL Query | organization.php

```
// This SQL query retrieves details about manufacturer and their number of active CTDs.
// It calculates the count of SKU's linked to a manufacturer based on the manufacturer Email.
// The details of the manufacturer include name, phone, and email.
// The query uses a LEFT JOIN between the MANUFACTURER and EQUIPMENT tables, linking them via the 'Email' and 'produce' columns, respectively.
// The result is grouped by the manufacturer Email. Used a left join because some manufacturers dont have active equipment.
// Users can specify the manufacturer Email through the GET parameter {$_GET['mu']}.
$sql = "SELECT Mname, Email, COALESCE(Phone, 'NULL') AS 'phonenumber', COUNT(SKU) AS ActiveCTD
        FROM MANUFACTURER Left JOIN EQUIPMENT ON MANUFACTURER.Email = EQUIPMENT.Produce
        WHERE Email = '{$_GET['mu']}'
        GROUP BY Email";
```

Figure 9: SQL Query | manufacturer.php

## Appendix E: Screenshots of functional Web Interface

Select an operator

Select a name ▼

Operator Name	Employer	Start Date	Year of Employment
Emily Johnson	WHOI	2021-08-22	2

Figure 10: Web Interface | operator.php

Select a location

location name ▼

Latitude	Longitude	Timestamp	Temperature	Transmissivity	Salinity	O. Saturation	Fluorescence	Density	Pressure
33.943200	-118.543900	2022-05-12 16:30:00	16.70	82.10	34.90	91.20	5.90	1020.10	148.60
33.945000	-118.471900	2022-03-10 14:15:00	18.50	75.60	34.20	91.80	5.50	1021.00	155.30
33.951400	-118.518400	2022-04-05 10:00:00	14.30	88.00	36.20	89.70	6.80	1018.50	140.80

Figure 11: Web Interface | location.php

Select an equipment sku#

Select a name ▼

Equipment Name	Average Temperature	Average Transmissivity	Average Salinity	Average Saturation	Average Fluorescence	Average Density	Average Pressure
DeepBlue Voyager	16.500000	82.800000	34.650000	91.350000	6.100000	1020.100000	155.450000

Figure 12: Web Interface | equipment.php

Select a Organization

Select a name ▼

Organization Name	Website	Contact Phone Number	Contact Email	Number of Employee
NOAA	www.noaa.gov	301-713-1208	outreach@noaa.gov	3

Figure 13: Web Interface | organization.php

Select a manufacturer's Email


Select an Email ▼

Manufacturer Name	Email	Phone	Active CTD's
Sea Explorers	sea@example.com	777-888-9999	2

Figure 14: Web Interface | manufacturer.php

[Deep Sea Developers](#) [Home](#) [Operator](#) [Location](#) [Equipment](#) [Organization](#) [Manufacturer](#) Miguel Vearna Makai Sunny

## Deep Sea Developers



"A CTD Database. We're going ocean!"

### Overview

Welcome to the Deep Sea Developer webpage! This project was created by students enrolled in the TCSS445 Database and System class of Fall 2023 at the University of Washington Tacoma. We were motivated by our passion for oceanography and the desire to contribute to the field. Our goal is to address the challenges faced by oceanographers in accessing and managing Conductivity, Temperature, Depth (CTD) data. With our knowledge of database implementation and theory, we aim to provide a scalable and user-friendly solution.

### Description

In this project, we deployed and hosted the webpage on the Google Cloud Platform using a Linux VM. We also connected it to a CTD (MySQL) database. The web interface, crafted using PHP, interacts with the CTD database. You can explore the different tabs on our website - Operator, Location, Equipment, and Organization. Each tab provides a unique way for users to engage with our CTD database.

### Team Members' Contact Information

Sunny Ali	hassanli@uw.edu
Makai Martinez	mim1738@uw.edu
Miguel Ramos	ramosmig@uw.edu
Vearna Bun	vearnab@uw.edu

Figure 14: Web Interface | index.php