**Milestone Update - SDZWA Sensor Team**

Ahmed Hussaini, Akshaya Sundaram, Brandon DalPorto, Ethan Tan, Jiping Lin, Malcolm McSwain

05/19/22

**Previously Defined Minimum Viable Product (MVP):**

Our proposed MVP would provide the Zoo with a functional web app that allows for basic communication between the sensors and a modular web application that may retrieve specified information.

The MVP is as follows:

**Common Requirements:**

* System functional without internet connection
* The system will use HTTPS for communication with a self-signed or provided certificate.

**Front End Requirements**

* Dashboard capable of pulling data from multiple sources
* Dashboard web application can handle various data inputs:
  + Numeric
  + Video
  + Audio
* Displays within the dashboard include the following capabilities:
  + Video/audio player
  + Line graph
  + Bar graph
  + Gauge graph
* Dashboard requires user authentication to access
* User may only access the dashboard if they received an invitation
* The project will use an already in place CRUD API

**Back End Requirements**

* Schema for sensor metadata
* Online Database for admin backend, which interacts with the frontend
* Server functionality for creation, deletion, and updating of admin and sensor database using REST API

**Redefined Minimum Viable Product:**

We have redefined the MVP as follows, with slight modifications in implementation details.

**Common Requirements:**

* System functional without internet connection
* The system will use HTTPS for communication with a self-signed or provided certificate.

**Front End Requirements**

* Dashboard capable of pulling data from multiple sources
* Dashboard web application can handle various data inputs:
  + Numeric
  + Video
  + Audio
* Displays within the dashboard include the following capabilities:
  + Video/audio player
  + Line graph
  + Bar graph
  + Gauge graph
* Dashboard requires user authentication to access
* User may only access the dashboard if they received an invitation
* \*\*NEW/CHANGED\*\* We will not be integrating with the Zoo as part of this project, we will be using sample sensor data from online, web APIs, and sensors available from the Sea Lab.

**Back End Requirements**

The backend team is on track to completing our MVP, with a few minor changes to the architecture that was proposed in the specifications. As we progressed with the project, the requirements became more clear to us, and we were able to redefine our MVP as follows:

* Requirements analysis for understanding what data/fields need to be stored in the database regarding the sensor metadata and administrative user information.
* Online database for admin backend, which interacts with the frontend
* Build, manage and visualize the schemas in the database using an Object Relationship Manager (ORM) tool.
* Write API templates for reading, updating, creating and deleting sensor and user information in the database.

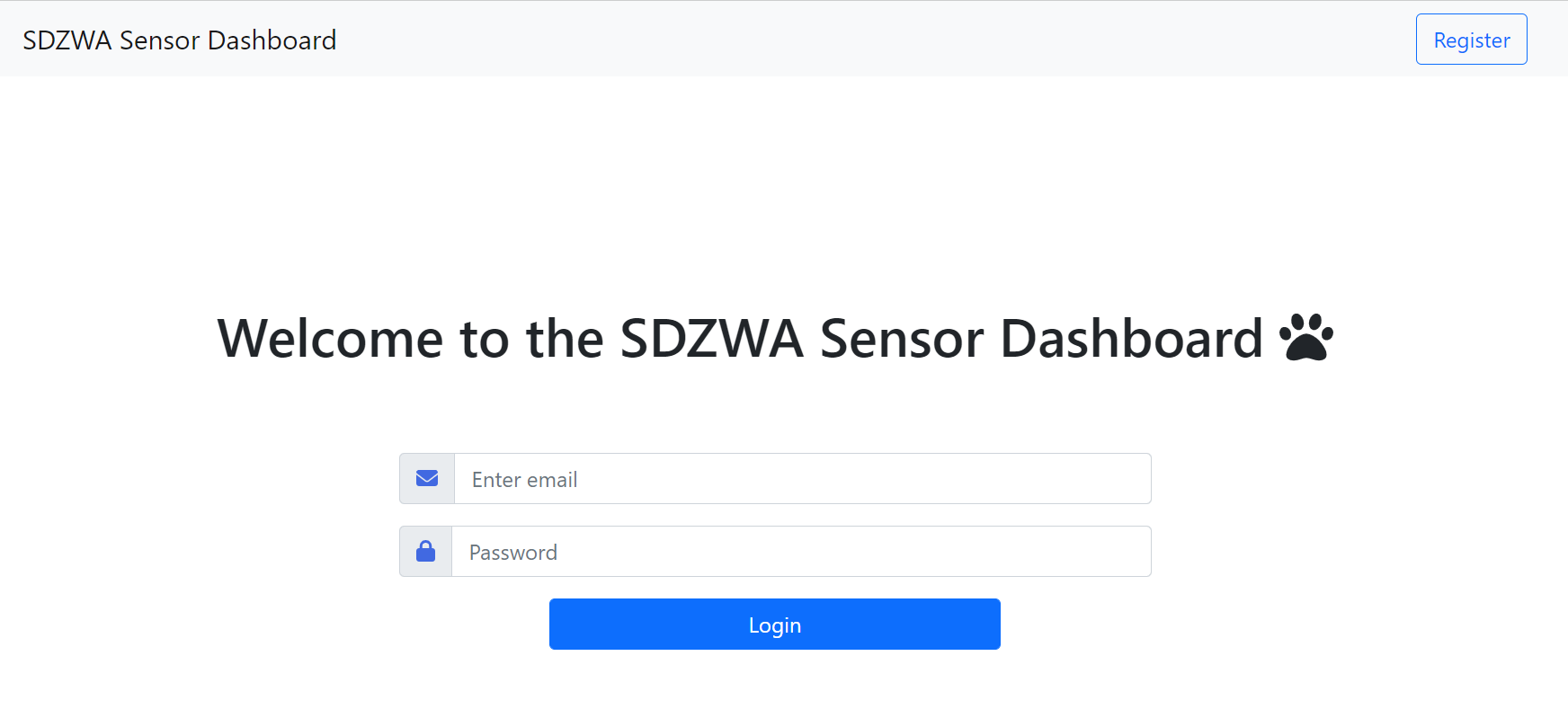
What’s Changed:

* After hearing that we would not be able to access or integrate with the Zoo this quarter, the front end team has changed the CRUD API requirement to using sample data collected ourselves, or from equipment in the Sea Lab.
* The backend team has added the requirement of having a comprehensive understanding of data, fields, and everything that will be stored in the database as part of the MVP. This was added because of the potential portability of this product, a comprehensive overview of database functionality with various interchangeable sensors is required.
* The backend team has also added the requirement of using an ORM tool for managing their schemas, again in the interest of organization and portability of this product.
* With the interest of time, and owing to the lack of availability of actual sensors from the zoo, the back end team will use mock data for testing our MVP.

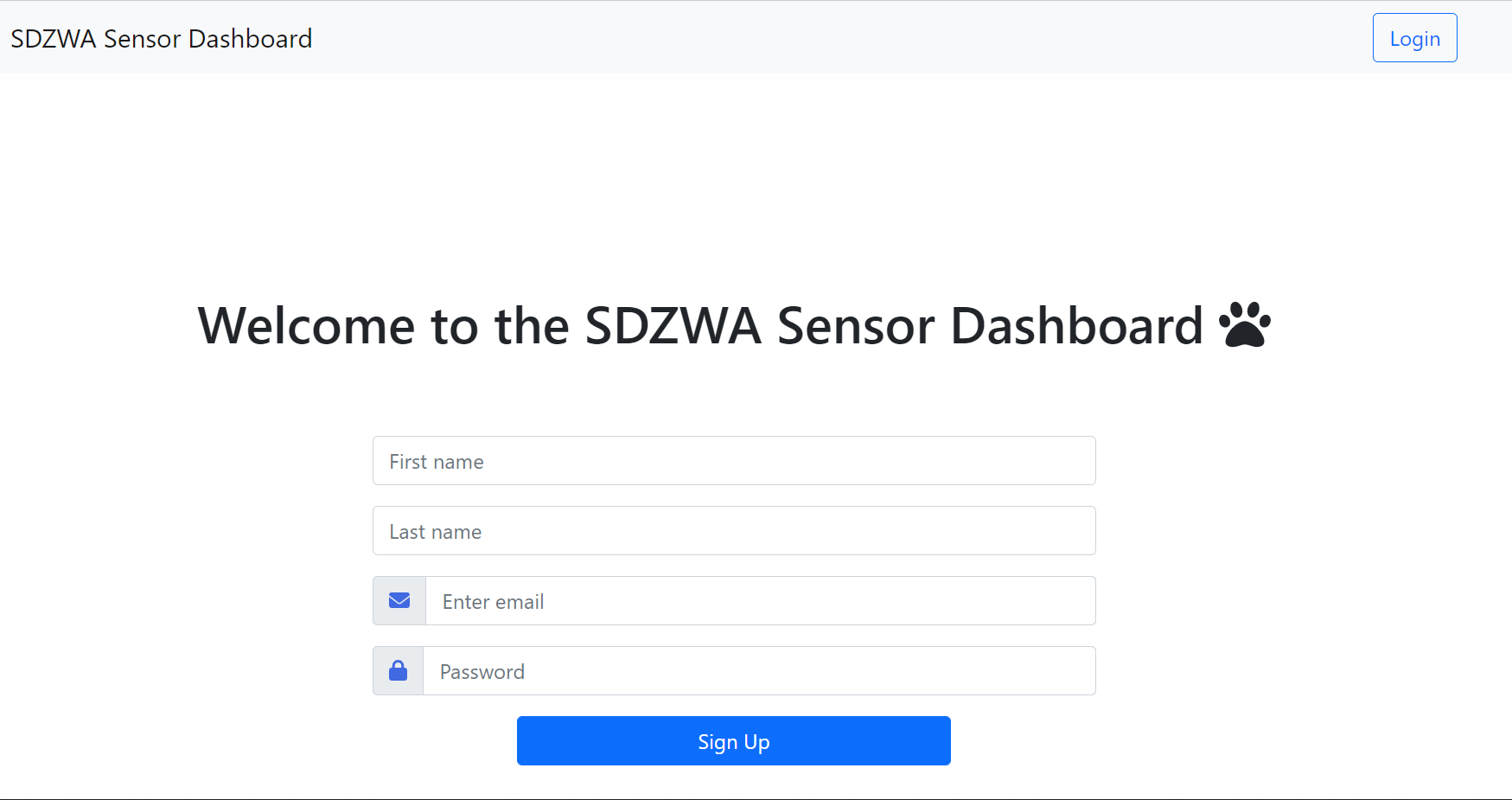
**Current Deliverables (Frontend Team):**

The Front End team has put a large amount of effort into creating a functional React based web application. Nearly all of the functionality as defined in our MVP has already been implemented. Here are the current project deliverables:

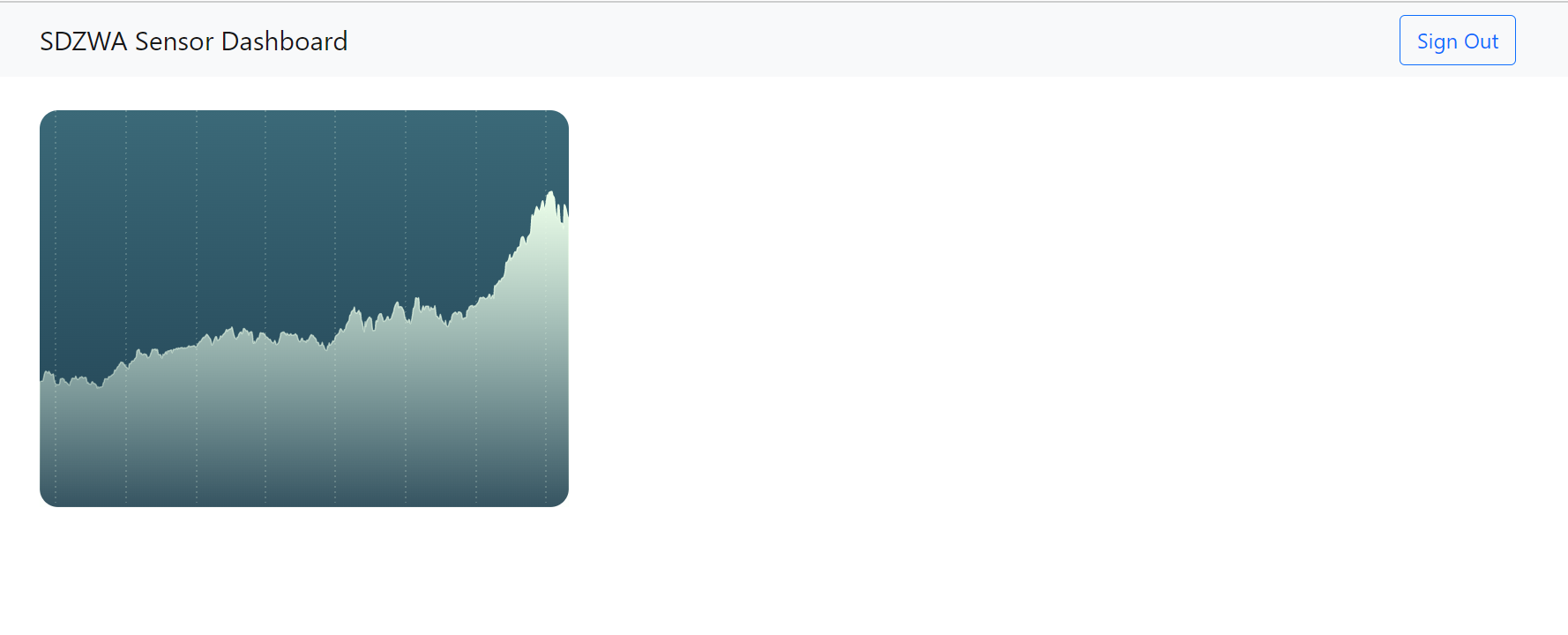
1. **Login Page**



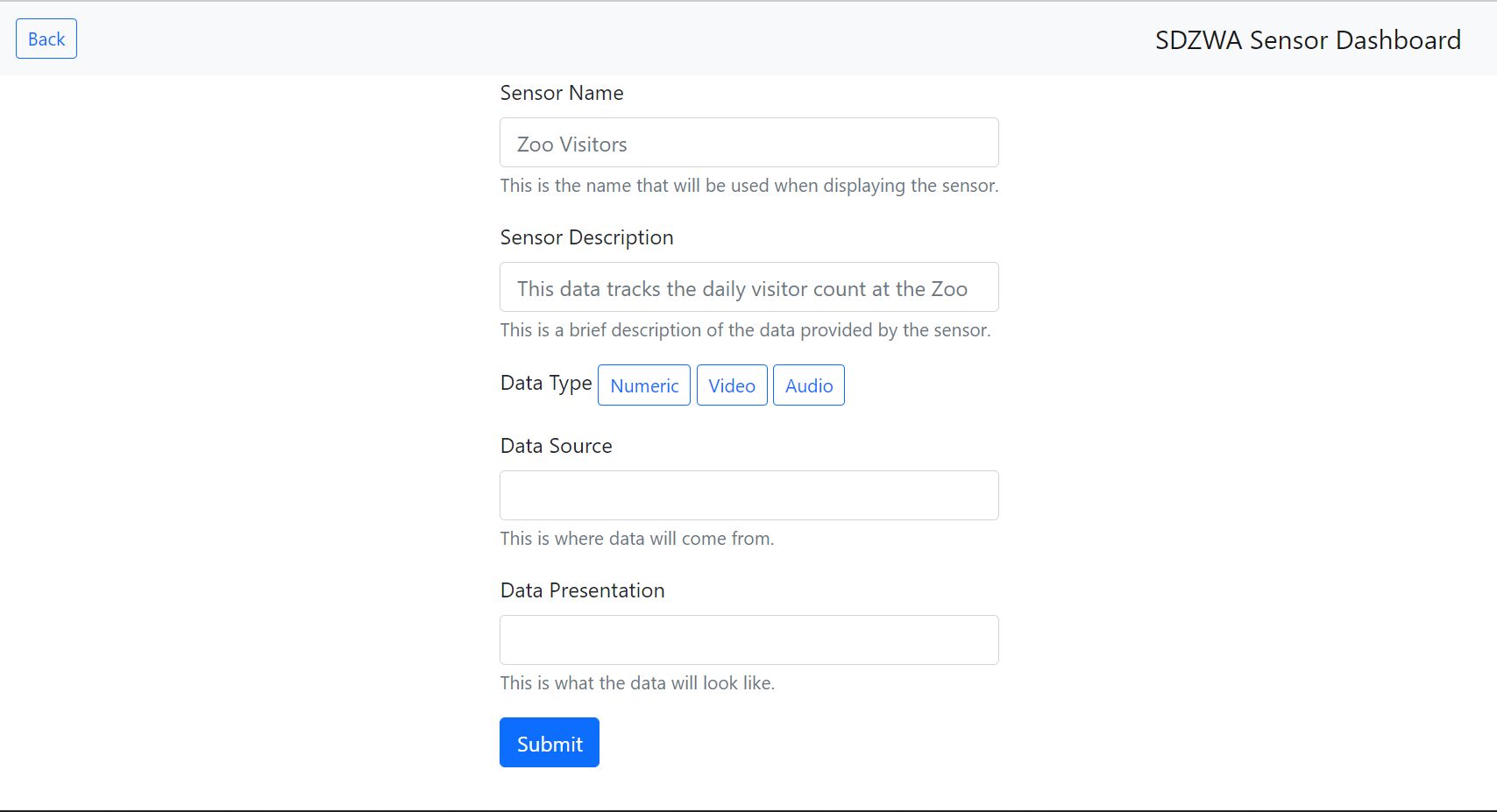
1. **Registration Page**

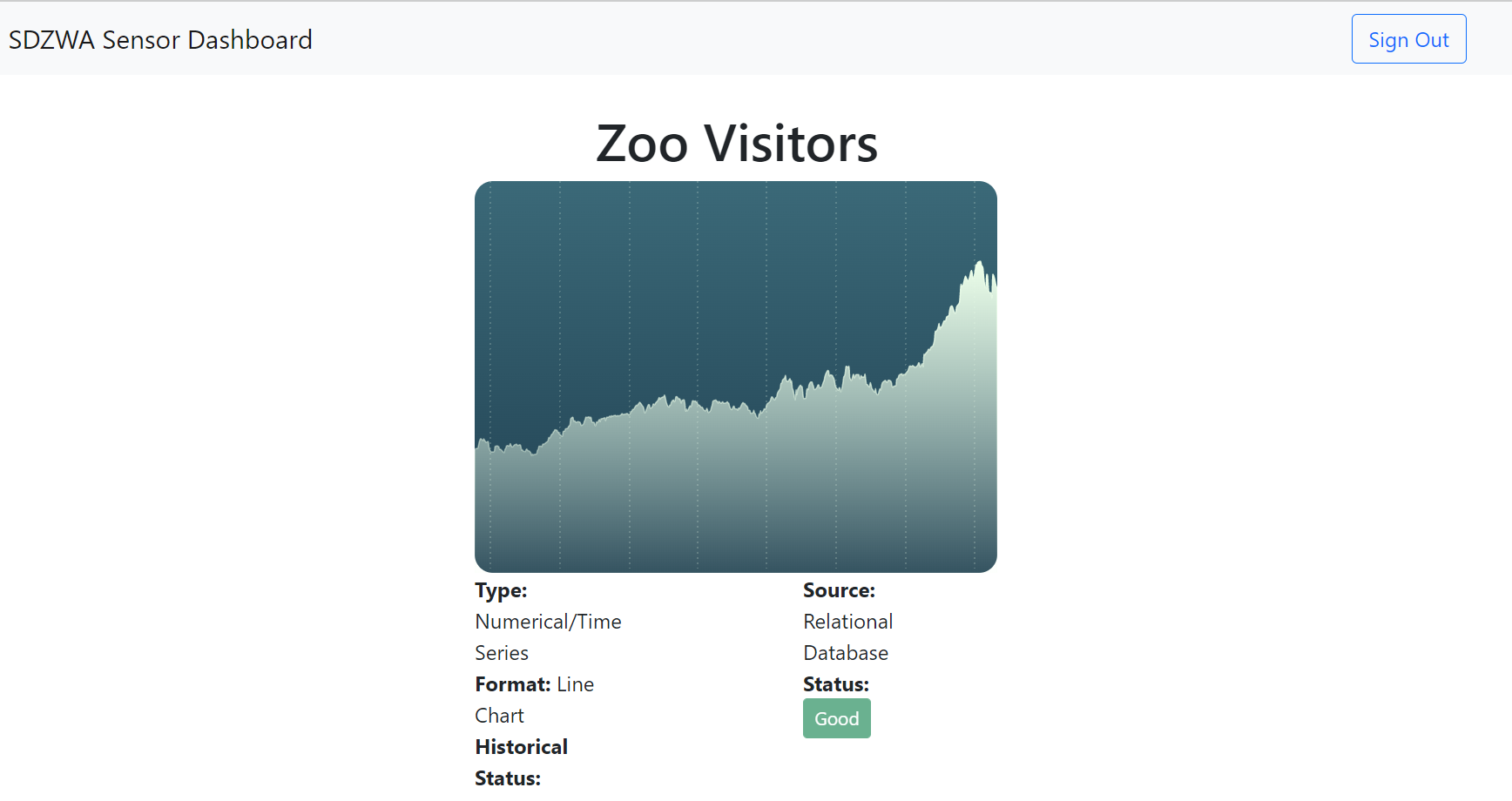


1. **Sensor Dashboard Page (Data Tiles w/ Graphs)**

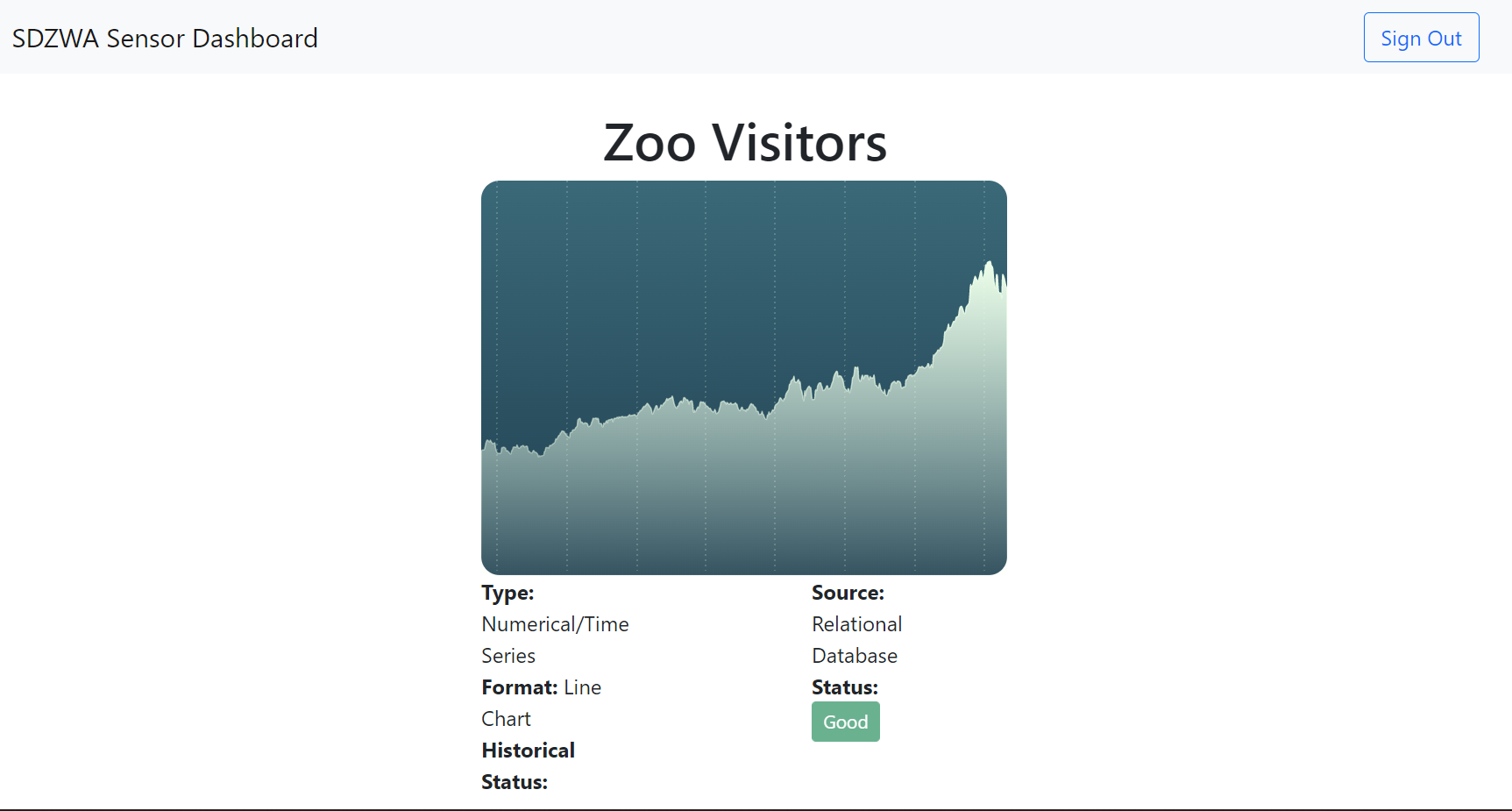


1. **Edit/Add Sensor Page**





1. **Individual Sensor Page**

****

**Future Deliverables (Frontend Team):**

Most of the work for the Frontend team as we continue on this semester will be integrating with everything in place from the Backend team, allowing for data to be retrieved and displayed in our pages.

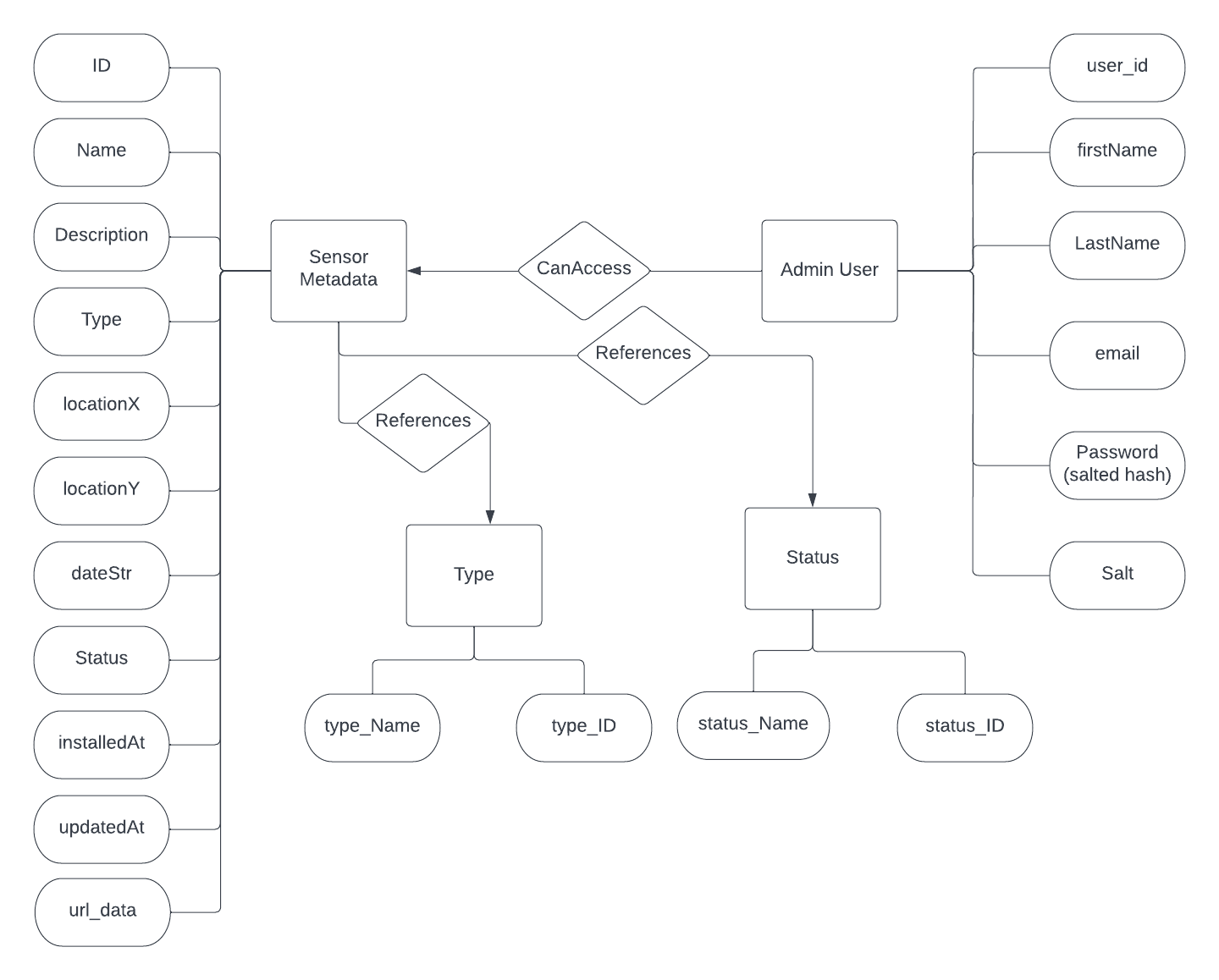
Future deliverables are as follows:

1. Display real/mock sensor data in our web page
2. Integrate with backend APIs to pull data
3. Display data of different types (need to show audio/video)
4. Fully functional web-app

**Current Deliverables (Backend Team):**

The back end team has achieved the following milestones that will lead to our MVP deliverable:

1. Through a process of brainstorming and requirements analysis for database functionalities needed to support the frontend functionalities, we have designed the below schema using PRISMA (an ORM tool) for sensor metadata and administrative user information. The below Entity - Relationship diagram was drawn using Lucidchart.



Each of the column attribute names in each of the tables will be briefly explained in the final report. The aim of our backend MVP is to provide the first and basic iteration of datastore fields and capabilities that are needed to build a sensor dashboard application.

The type and status tables store reference information for the current working condition of sensors, and the kind of sensor that it is as follows:

| type\_id | type\_name |
| --- | --- |
| 0 | Camera |
| 1 | Motion |
| 2 | Therma |

| status\_id | status\_name |
| --- | --- |
| 0 | Idle |
| 1 | Active |
| 2 | Not Active |
| 3 | Needs Attention |

Our design choice for having separate tables for type and status is based on the assumption that they can be generalized, and extended in the future iterative scope of the database.

We also believe that new data fields could be added or removed as face integration challenges and subsequent requirements

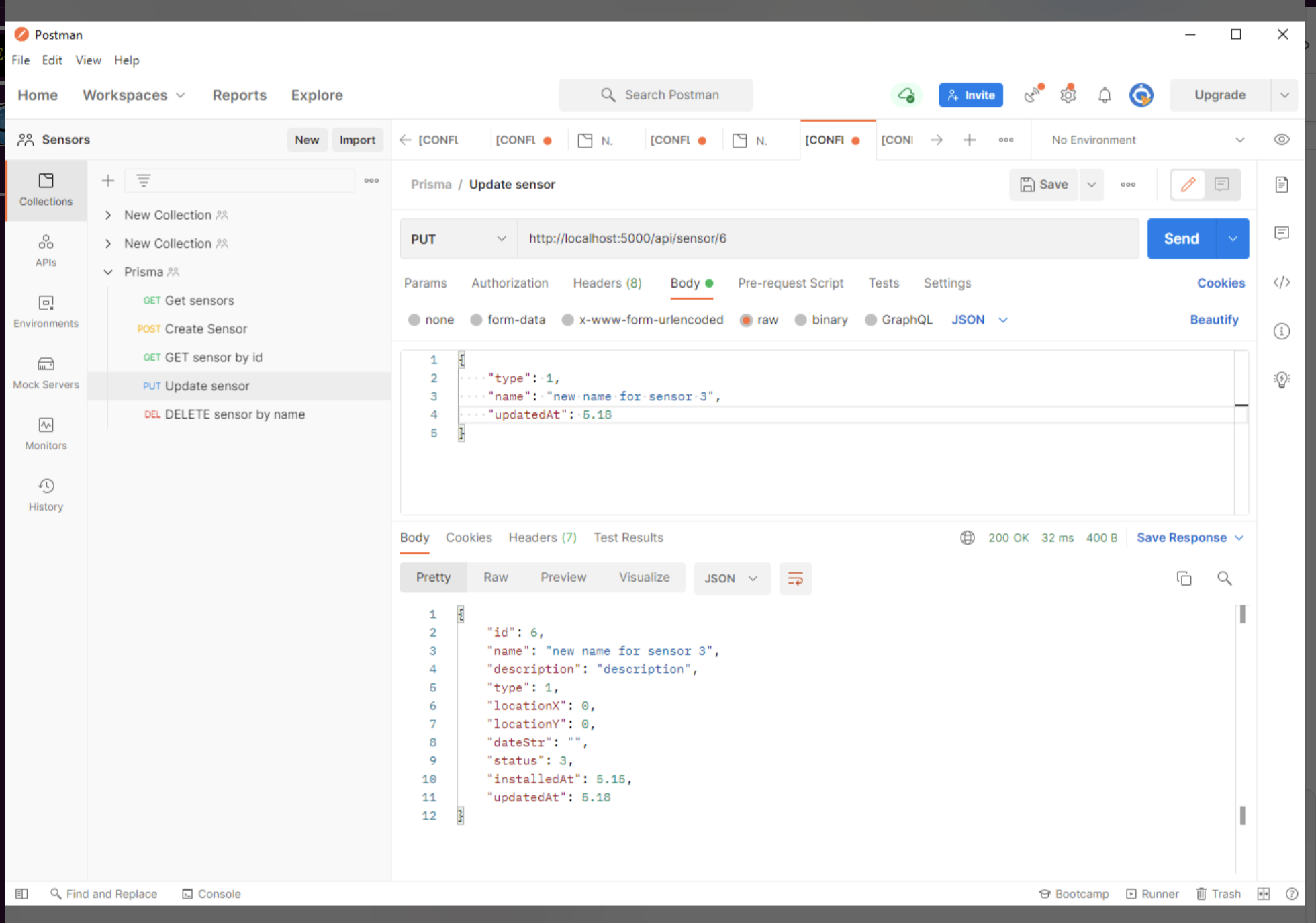
1. Using Prisma studio we were able to visualize our database tables on the localhost.
2. We have also created a docker image for Node.js server for our SQLite local datastore.
3. We have written the following REST API templates for querying our database:

4.1 Find sensor metadata by ID.  
4.2 Find sensor metadata by name.  
4.3 Update sensor metadata by ID.  
4.4 Delete sensor by name.  
4.5 Find an admin user by ID.  
4.6 Find an admin user by email.  
4.7 Find an admin user by name.  
4.8 Update user by email.  
4.9 Delete user by email.  
4.10 Add new user during registration or sign up.

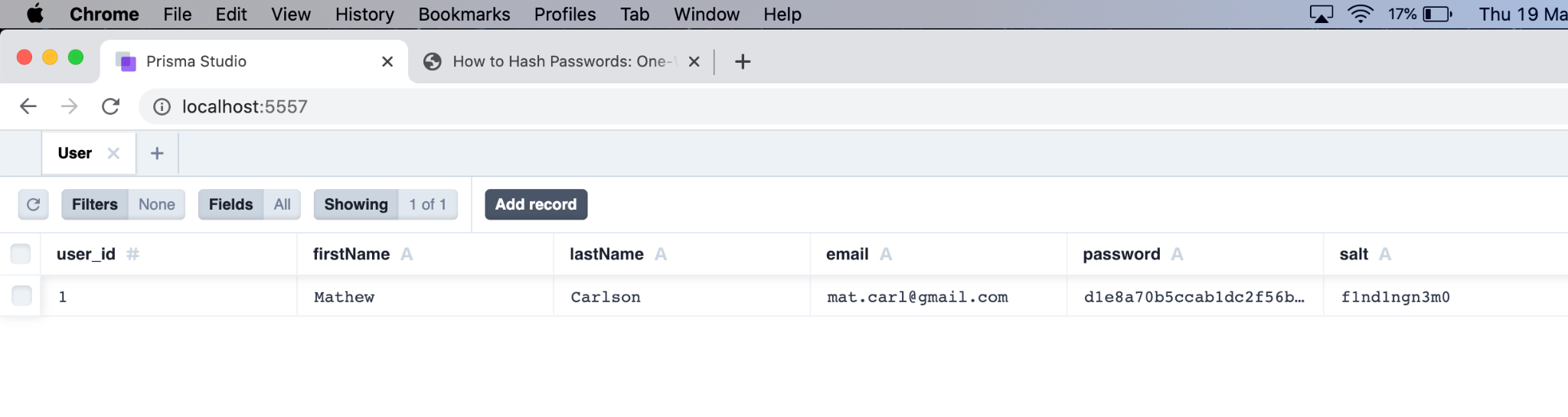
The above templates are not final, and will be refined based on the endpoint specification from the frontend team. As of now we are discussing whether the sesnor\_id and user\_id should be an auto incremented field in the database that can be used for internal reference or if it should be assumed to be assigned by app administrators during creation of that sensor and user record in the database. Based on the design choice with the frontend team, 4.1 and 4.5 will be subject to change.

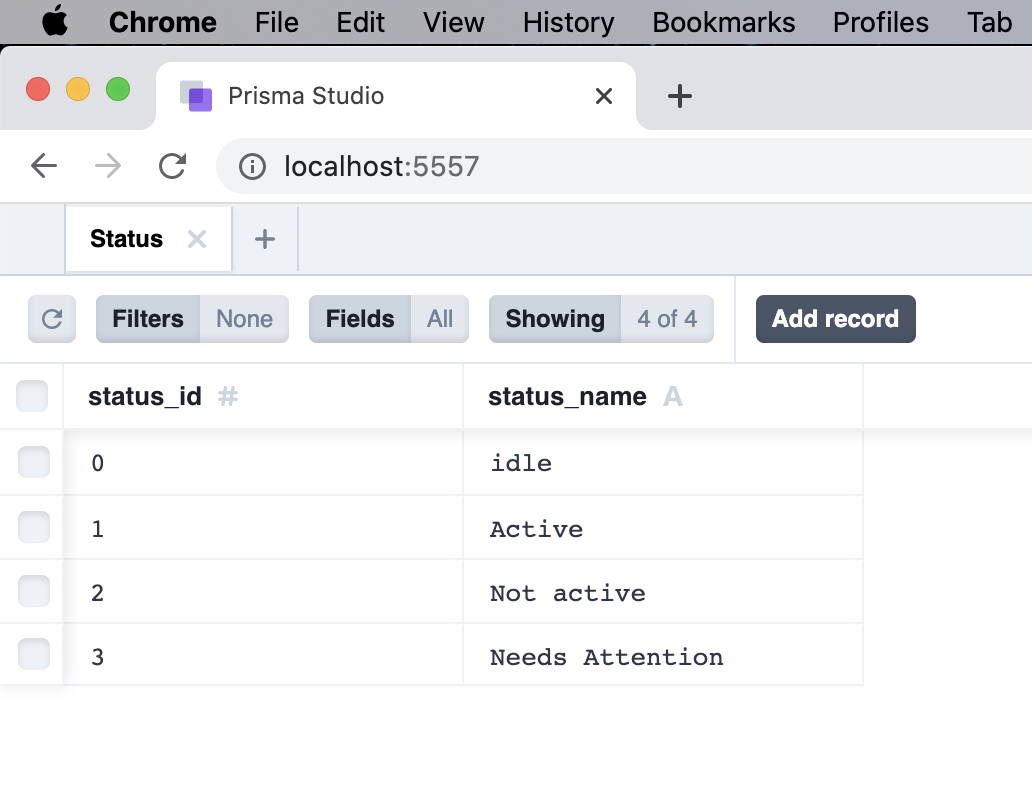
1. We have written the API templates for 4.1 - 4.10 using Express (a Node.js framework), and tested 4.1 - 4.4 using the Postman API.
2. We have also written an Express Node.js API template for handling admin passwords in the database using salted hash encryption. The admin password will not be stored directly in the database. We will be storing only the salted hash of the password, and its salt.

**Sample screenshot for Testing ‘Update Sensor’ API with Postman:**



**Sample screenshots for Visualizing Database tables using Prisma Studio:**





**Future Deliverables (Backend Team):**

The future deliverables of the backend team will be focused on finishing our MVP deliverable, and providing support to the functionalities required by the frontend team.

1. Test the remaining 4.1 - 4.10 using the Postman API.
2. Organize our API functions using routes, and controllers for easy debugging during deployment.
3. Populate the database with mock data that would support the frontend requirements.
4. Work with the frontend team to integrate the database with the React web application.

**Milestones & Schedule (Frontend Team):**

Milestone 1: Deploy initial “hello world” web application **(COMPLETE)**

* + Our first milestone consists of deploying an initial “hello world” basic application which will allow us to familiarize ourselves with the ReactJS framework. Writing a program with simple webpage and button functionality is this first step.

Milestone 2: Registration/authentication flow **(COMPLETE)**

* + Our next milestone regards getting the workflow for the overall application up and working. The flow includes registration screen, authentication/login screen, and dashboard view. This milestone consists of initializing the project using ReactJS, creating the required views, and building out forms to capture user information. We want to create pages for each supported platform, and the dashboard will remain blank for now.

Milestone 3: Dashboard tile view **(COMPLETE)**

* + This milestone will be focused on getting the tile view of the dashboard up and running; this will be the primary front-end display component. We would also want a modular component to these tiles that can be added or removed, and give the user the ability to do this. There will be no graphing or data visualization in this dashboard yet.

Milestone 4: Graphing/data visualization functionality **(In Progress)**

* + In this milestone, we want to have graphing and data visualization functionality on the dashboard. This involves implementing the ability to produce a variety of different graphs and charts, depending on what forms of data need to be tracked. The graphs will be displayed on the “tiles”, and clicking on each tile will bring up a more detailed and interactive form of the graph.

Milestone 5: React Native application (stretch) **(Not Reached)**

* + This final milestone would be to create a mobile port of the web application using React Native. The mobile app would have all the same functionality as the web application, just ported over to a mobile interface.

| **Week** | **Milestone** | **Team members working on Milestone** | **In - charge** | **Maximum Buffer Time - Manage risks** |
| --- | --- | --- | --- | --- |
| 5 | 1 | Frontend Team | - | 1 days |
| 5 | 2 | Frontend Team | Malcolm | 3 days |
| 6 | 3 | Frontend Team | Ahmed | 5 days |
| 7 | 4 | Backend Team + Frontend Team | Brandon | 7 days |
| 8 | 4 + 5 (if time Permts) + System Integration | Frontend Team + Backend Team | Frontend Team as a whole | 10 days |
| 9 | Finish 4 + 5 + System Integration | Backend Team +  Frontend Team | Brandon | 3 days |
| 10 | Report, Video Recording | Backend Team + Frontend Team | Brandon / Akshaya | 1 day |
| 11 | Final Demo, Wrap up | Backend Team + Frontend Team | Brandon / Akshaya | - |

**Frontend Priorities:**

The frontend team is slightly behind schedule, as we have not had access to any real or mock data of sensors similar to those that would be at the Zoo. We have come up with a solution of creating mock data ourselves, as well as pulling from web APIs such as the San Diego Zoo’s live camera feed.

Our current priorities are as follows (urgency measured 1-5)

1. Retrieve test data (5)
2. Integrate test data with current implementation (5)
3. Receive user feedback from Ian Ingram @ SDZWA (3)
4. Adjust based on feedback (3)
5. Integrate with back end team (5)
6. Final class deliverables (2)

**Milestones & Schedule (Backend Team):**

Since integration with the zoo, and access to any kind of sensor is not certain, the backend team has revised our MVP to fit the new requirements of the project. We were also able to define our milestones with more confidence after starting work on the project and understanding the fundamental requirements.

Milestone 1: Set up a database schema for sensor metadata **[COMPLETE]**

* Identify generic metadata sensor information.
* Use ORM to build schemas for storing that metadata in the database.
* Implemented the schemas using SQLite + Prisma, and visualized the tables using Prisma Studio.

Milestone 2: Setup database schema administrative backend **[COMPLETE]**

* Use ORM to create, manage and visualize tables to store user data and hashed passwords for administrative front-end access.
* Host backend online on a reliable Node server
* Create a docker image for the Node.js server.

Milestone 3: REST API templates for sensor queries **[COMPLETE]**

* Create routes and controllers to define sensor functions/queries based on REST API specifications (PUT, POST, GET, DELETE) in Express (a Node.js framework).
* Test API functions using Postman.

Milestone 4: REST API templates for admin user queries **[IN PROGRESS]**

* Create routes and controllers to define admin user functions/queries based on REST API specifications (PUT, POST, GET, DELETE) in Express (a Node.js framework).
* Test API functions using Postman.

Milestone 5: Handle passwords in database using salted hash **[COMPLETE]**

* Identify a secure way on how to store admin passwords in the database.
* Use an encrypted salted hash of the password to store in the database.
* Store the salt also as a field in the database.
* Write API template in Express to implement the above logic.

The completion of the above milestones lead to the development of our MVP.

**Backend Schedule:**

The backend schedule has been updated, to accommodate the revision of our milestones, as below:

| **Week** | **Milestone** | **Team members working on Milestone** | **In - charge** | **Maximum Buffer Time - Manage risks** | **STATUS UPDATE** |
| --- | --- | --- | --- | --- | --- |
| 5 | 1 | Backend Team | Akshaya | 7 days | COMPLETE |
| 5 | 2 | Backend Team | Jiping | 3 days | COMPLETE |
| 6 | 3 | Backend Team + Frontend Team | Jiping | 3 days | COMPLETE |
| 7 | 5 | Backend Team + Frontend Team | Ethan | 5 days | COMPLETE |
| 8 | 4 | Backend Team | Ethan | 3 days | IN - PROGRESS |
| 9 | System Integration | Backend Team +  Frontend Team | Brandon | 5 Days | NOT REACHED |
| 10 | Report, Video Recording | Backend Team + Frontend Team | Brandon / Akshaya | 1 day | NOT REACHED |
| 11 | Final Demo, Wrap up | Backend Team + Frontend Team | Brandon / Akshaya | - | NOT REACHED |

**Backend Priorities:**

The backend team’s highest priority will be to support the requirements and endpoint specifications of the frontend team during integration. We anticipate that changes could be made to our original design choices for the templates and database fields based on the new challenges that arise during the frontend and backend integration. We will modify our previous choices accordingly to create a fully functional web application of the sensor dashboard that will serve as the completed project deliverable for this course.