# Demand Response Portal

Elijah C Monroe | Enora Boscher | Evyn Price | Mykola (Nick) Omelchenko | Serena L LaBelle | Shelby Smith | William Goodson | Won (Brian) Lee
Eric L. Brown
Dr. Michael Rogers
Department of Computer Science, Tennessee Technological University

## INTRODUCTION & ELEVATOR STATEMENT

We are developing an innovative demand response and smart grid system as part of a larger ARC project focused on modeling renewables and empowering energy providers and consumers. Our solution leverages cutting-edge communication protocols such as OpenADR and advanced hardware like Typhoon HIL to create a seamless data exchange between providers and customers. This user-friendly web portal offers real-time data visualization, incentives load shedding during peak demand, and fosters transparency for both parties. By integrating advanced ARIMA models and analytics, our system ensures fair rewards for energy-saving behaviors while offering valuable insights for providers. Together, we are shaping the future of energy management and building a smarter, more sustainable grid.

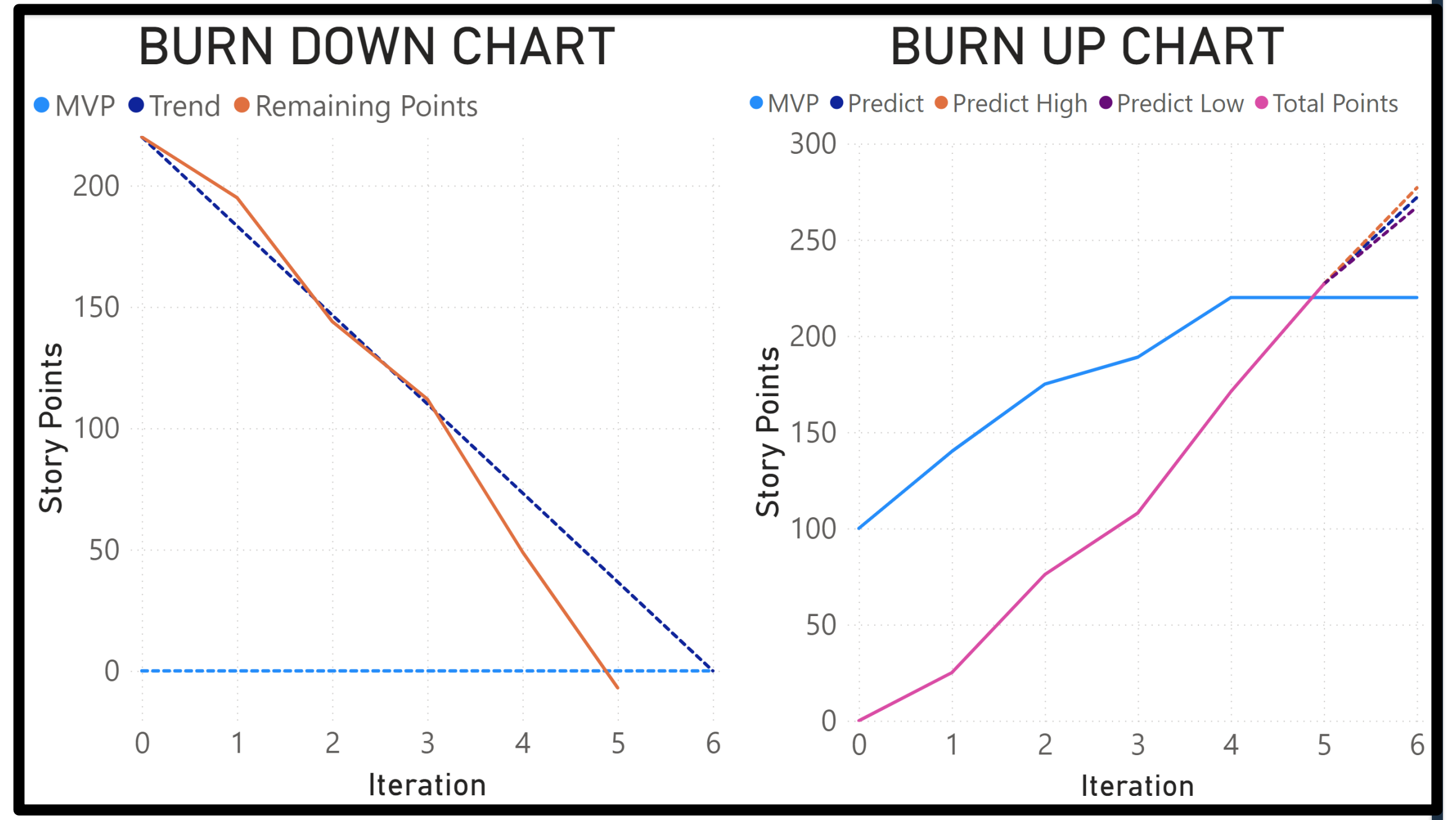
We are developing a responsive and accessible web portal that allows power customers and utilities to interact with demand response data in real time to ensure fair rewards for energy-saving behaviors while offering valuable insight for providers.

## PROJECT OVERVIEW

To organize this project, we used Agile techniques more specifically Scrum techniques. We started off by communicating with the client (Dr. Michael Rogers) about the requirements for the project. Then we used the requirements to make user stories and assigned each story points to help prioritize and manage the team's goals. We had a total of 270 story points and an MVP (Minimal Viable Product) of 220. We as a team split the story points to each member based on their specialty.

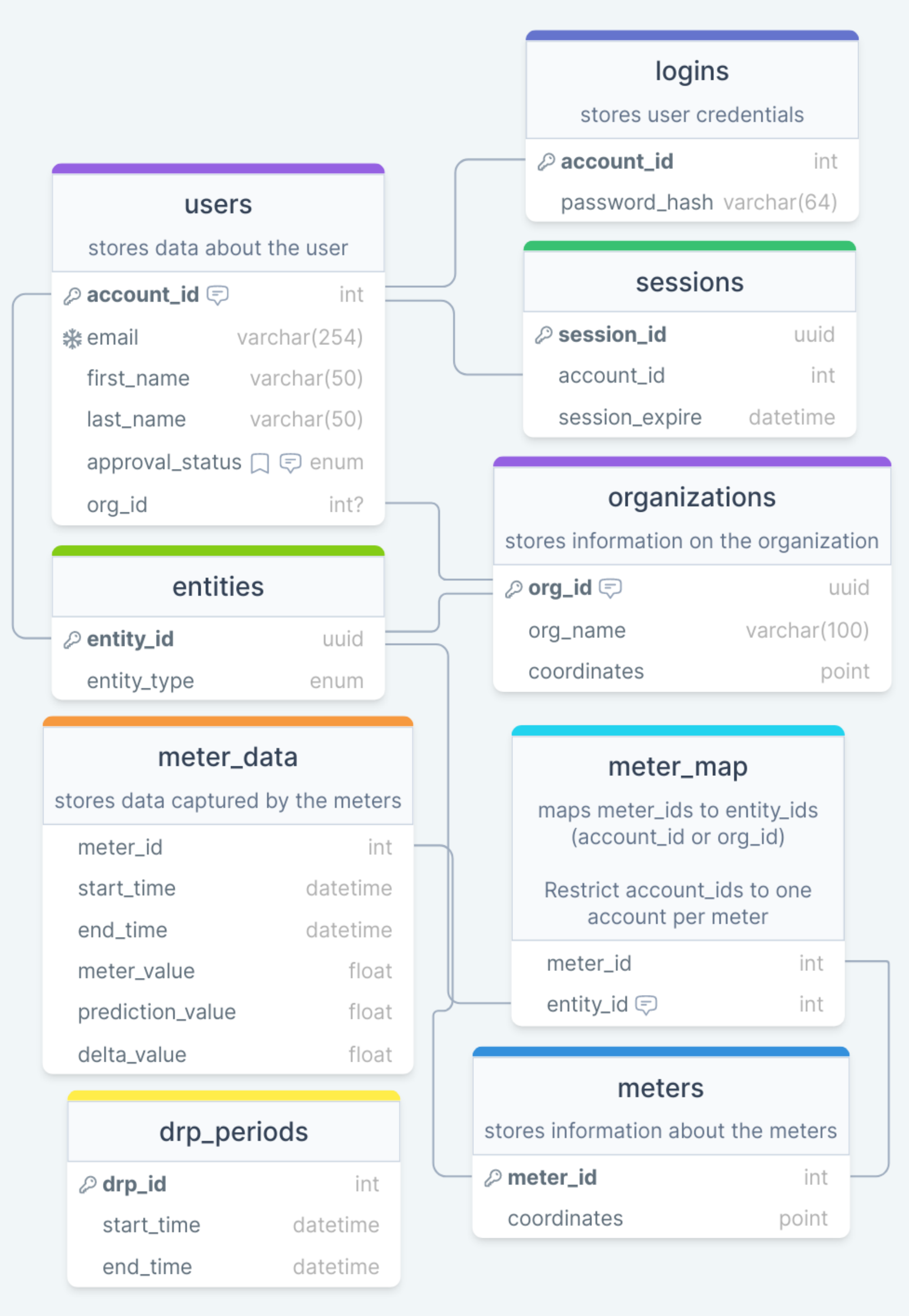
| NAME | TITLE | STORY POINTS |
|---|---|---|
| Enora Boscher | Data Scientist | 5 |
| William Goodson | Lead Back-End Developer / DBA | 52 |
| Serena L LaBelle | Front-End Developer | 20 |
| Won (Brian) Lee | Back-End Developer | 22 |
| Elijah C Monroe | Lead Front–End Developer | 56 |
| Mykola (Nick) Omelchenko | Lead Security Specialist | 25 |
| Evyn Price | Scrum Master | 41 |
| Shelby Smith | Lead Data Scientist | 55 |

We had 6 iterations to work on the project, each was 3 weeks long for a total of 18 weeks. To meet the MVP we scheduled 40 iteration points for each iteration, a few more then the necessary 37 required. This was to ensure we would be able to complete the project on time. We used Github to track these story points. Below is a Burn Up and Burn Down Chart for how many story points we completed each iteration.



From our client we received data and some previous work on the project. Most of it we did not use but we did optimize the ARIMA they provided to be more accurate. The rest of the project was designed by the team.
In the following sections we will cover in detail the Database Schema, API Calls, UX/UI design, and Server Flow chart. In each section we will discuss the different types of technology used for each aspect of the project.

## DATABASE INFORMATION



For our project we had to design a database that would store the information required by the client. To the right you will see our database schema, you can read this to see how we have organized the data. Below is the legend used to identify primary keys, if the entries are Unique, and the foreign keys contained in the database. We chose MariaDB as our project's primary database. We chose MariaDB because it's a cost-effective, open-source database that guarantees compatibility with MySQL while offering improved features and performance. It has ACID compliance, support for different storage engines, and solid performance, because of this MariaDB ensures our system's reliability and scalability. It plays a crucial role in our project's success by maintaining data efficiently and securely.

### SCHEMA LEGEND

| | |
|---|---|
| 🔑 | Primary Key |
| ❄ | Unique |
| — | Foreign Key |

### API CALLS

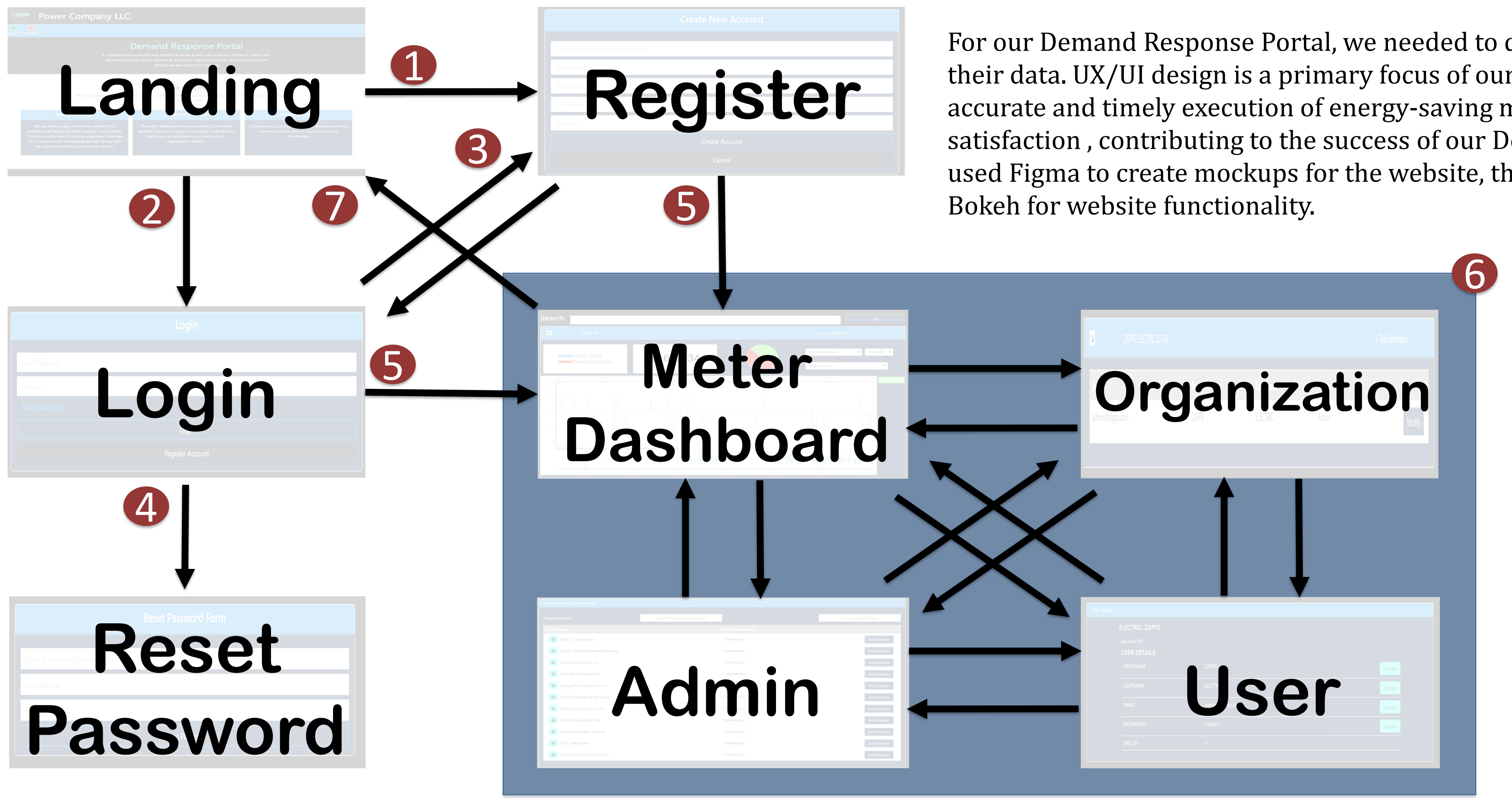
To be able to quickly and securely access the database we created an API using Flask. This is how the user retrieves and adds information to the database. It also allows us to update the charts periodically for real-time data visualizations

### DATABASE TRIGGERS

To ensure data integrity, automating certain data management tasks, and controlling the associations and permissions within the database; we created the following triggers.

1. **add_user_id**: Generates user IDs for new user records.
2. **add_org_id**: Creates organization IDs for new organization records.
3. **limit_residential_associations**: Ensures one user per meter association to maintain data integrity.
4. **link_user_entity_delete**: Deletes user-related entries from the "entities" table upon user deletion.
5. **link_org_entity_delete**: Removes organization-related entries in the "entities" table when organizations are deleted.
6. **populate_deltas**: Calculates and updates meter data delta values.
7. **org_prime_admin**: Automatically assigns admin status to the first user in an organization.
8. **approve_residential**: Automatically approves users not associated with organizations.
9. **disallow_org_user_map**: Prevents users in organizations from mapping to meters.
10. **assign_distributor_meters**: Assigns newly added meters to a distributor entity for tracking.

## UX/UI DESIGN



For our Demand Response Portal, we needed to design a front end for users to access their data. UX/UI design is a primary focus of our project, because it facilitates the accurate and timely execution of energy-saving measures and improves client satisfaction , contributing to the success of our Demand Response Portal project. We used Figma to create mockups for the website, then used HTML, JavaScript, CSS and Bokeh for website functionality.

### PAGE DESCRIPTION

- **Landing Page** : Shows the basic information about the project.
- **Register Page** : Lets users register an account
- **Login Page** : Lets users with an account login
- **Password Reset Page** : Lets users with an account reset their passwords
- **Meter Dashboard** : Displays the bokeh plot of the meters the user has access too
- **Organization Page** : Allows Org Admins to alter users within their organization
- **Admin Page** : Lets Distributor Admins manage organization groups
- **User Page** : Lets users manage their own accounts.

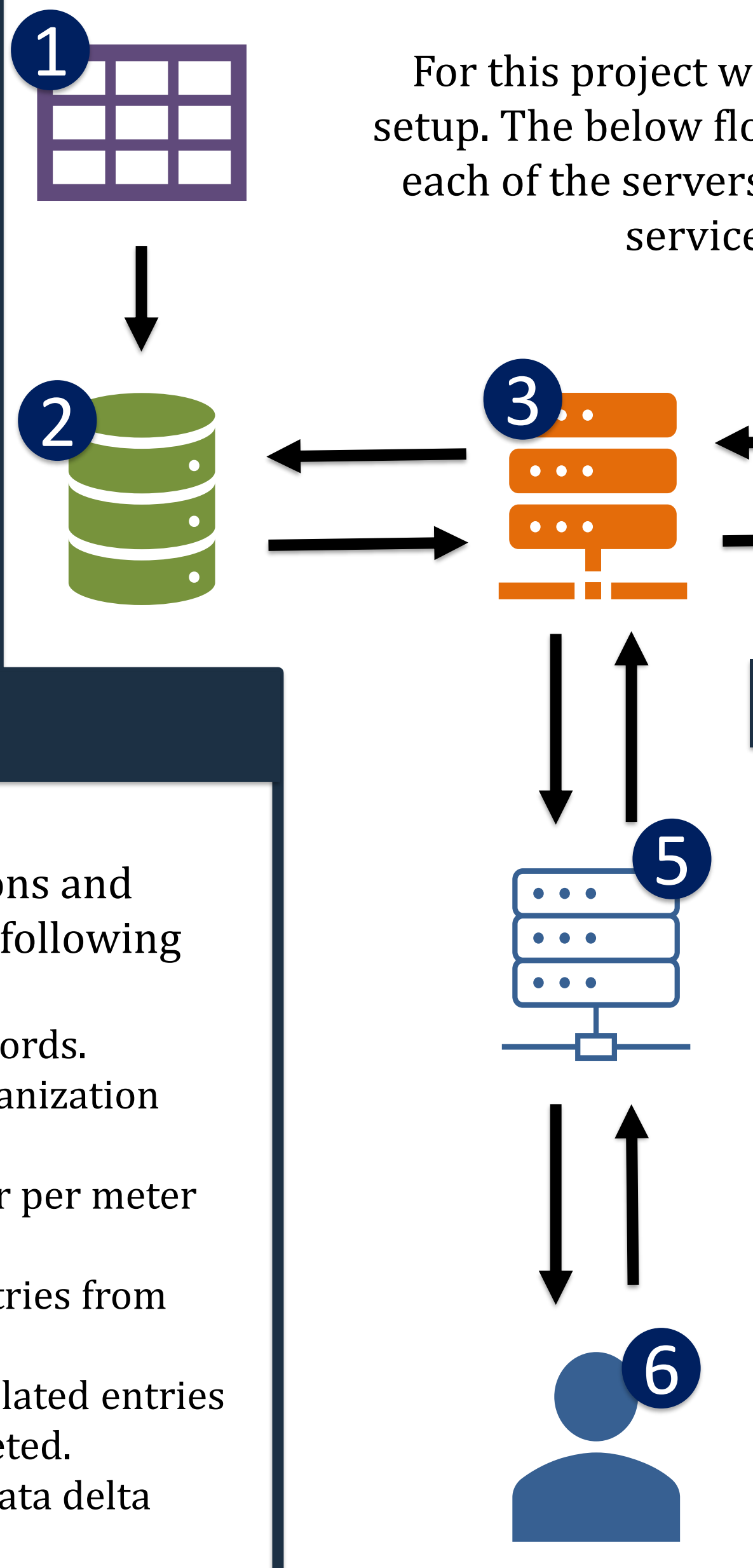
### TYPICAL USER JOURNEY

The typical user journey starts when users access the website where they are brought to the landing page.

1. If the users do not have an account, they can register.
2. If the users have an account, they can login.
3. The users can switch between the two if they accidently chose the wrong option.
4. If the user forgot their password, they can navigate to the reset password page to reset it.
5. After the user registers the account or logs in they are brought to the meter dashboard.
6. using the navigation bar at the top of the page they can move between the meter dashboard, the admin, the organization page, and the user page however they want.
7. When the user logs out, they are brought back to the landing page.

## SERVER FLOWCHART

For this project we used docker to automate our infrastructure setup. The below flow chart show how data is transferred between each of the servers. Each plays an important part in keeping our service reliable, efficient and responsive.



### SERVER DESCRIPTION

Here is a description of each of the servers:

1. **DB IMPORTER**: Fills the Database with the data received from client
2. **DB (MariaDB)**: Holds all the data.
3. **FLASK**: This is our API. Allows us to interact with the database using http methods (GET, POST, DELETE, etc.)
4. **DB CACHE (REDIS)**: Caches specific data in memory for fast delivery (sessions, meter data, meter-user connections)
5. **NGINX**: Pulling double duty as a web server for thing single page app and also a reverse proxy for the flask server
6. **USER**: Person accessing the website

## PROJECT OUTCOME

As mentioned in the project overview we measured our project using story points, we set our MVP at 220 points. To calculate the amount of points required to consider the project complete (Definition of Done) we counted the number of points needed to have:

1. A Complete Dashboard
2. A Functional Website
3. An Efficient Database
4. A Reliable API
5. Security Tested Software

We managed to reach this goal at during Iteration 5, with the client also satisfied with the quality of the project.

As a team we did not encounter any significant roadblocks. Any roadblocks encountered were by individuals and were resolved quickly without impacting the rest of the team.

To aid this project in the future, it would be good to receive more data from the electric company and create a new prediction algorithm based on that data. It would also be good to directly change the VTN to call the database API instead of it creating a CSV file that is then imported into the database.

Overall the progress of the project was great, we easily completed the project near the beginning of iteration 5. This left us with plenty of time to add extra improvements to the project, leaving the client with a great product.

## REFERENCES & RESOURCES

| | |
|---|---|
| Flask | TTU CyberRange |
| Nginx | Bokeh |
| MariaDB | Redis |
| Github | Docker |

## ACKNOWLEDGEMENTS

Special Thanks for help on this project to:
Benjamin Burchfield
Travis Lee
Rajesh Manicavasagam