EZONE Building Energy Management

Prototype Project Plan

CITS5551 Software Design Project

Jason Chu	21300674@student.uwa.edu.au
Lifsania Francis Xavier Robin	22873367@student.uwa.edu.au
Willem Meyer	21496322@student.uwa.edu.au
Yiwen Li	22061312@student.uwa.edu.au

Team Contributions

1.1. Jason

1.1.1. Project Management

Requirements Gathering and Analysis

1. *Project Requirements gathering/ Analysis*

   Gathering questions for sponsor and project for requirements – write up requirements gathered for proposal, requirements regarding database and accessibility

2. *Feasibility Study*

   Gathering information on feasibility on both an API and existing technologies for specifically the UWA Future Farm including; devices used, existing data format, access to data, architecture

System Design

1. *Functional and Technical Specification Document*

   Responsible for Technical Specification – Research and report available and possible technologies to use including web development, data storage and collection, development and performance requirements

Prototyping

1. *Sponsor Meeting*

   Attend for update on access to databases and existing PostgreSQL database hosted by UWA and REV website. Currently used for prototype.

2. Prototype Development

   Research and develop prototype Data Connector between required data sources (UWA Human Movement Building, UWA Future Farm) and database– developed a python script to be run that will, at request:

   1. Take the latest csv in a directory (currently local for prototype, changeable to ftp server)
   2. Clean the csv into a new dataframe and output into a suitable format for the PostgreSQL database using pandas
   3. Connect to *therevpr_ocpp* database through SSHTunnel and psycopg2
   4. Upload the requested information from a pandas dataframe into the suitable *ems_transaction_ext* table in *therevpr_ocpp* database

1.2. Yiwen

1.2.1. Database Design

Overview

The Energy Management System is built to present energy devices' information on to web pages, therefore a database will be required for the system to store the devices' data. The database is designed to maintain energy devices' master information and receive the transactional data collected from devices which are connected to the system. The received raw devices' data will be transformed and stored in the database for analyzation and comparison. And then the web pages can fetch the treated information from the database and present to the website according to the business requirement. As a result, the database of the Energy management system has been divided to six layers: input layer, transformation layer, master data layer, configuration layer, transaction layer and output layer.

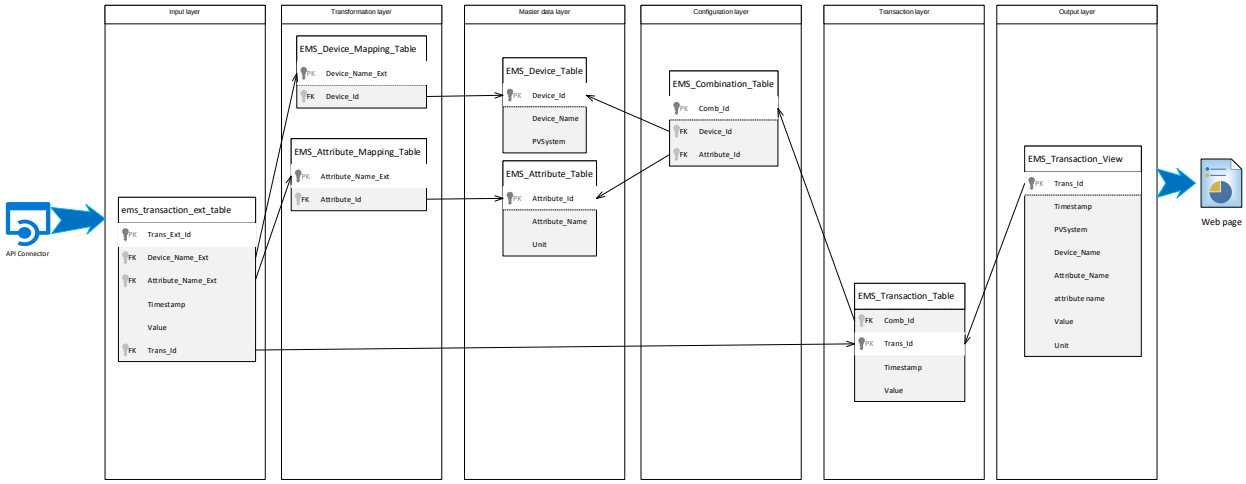
Design decisions

The key design principle of the database to make the data structure of Energy Management system to be flexible and extensible, so that the system can handle the information from different kinds of energy devices and meet the business requirement of presentations. To achieve the design principle, there are a few key decisions made as below:

- The input layer is designed as an API to receive the raw device data from API connectors. Because the database is at version 8.4.20, the input layer is built with a table rather than a complicated updatable view.
- The combination layer is introduced to manage the dimensions of the energy data. The current known data indicates that Energy management system will only handle two dimensions: devices (such as Battery) and attributes (such as Power kWh). And the combination layer allows the system to extend to more dimensions.
- The transaction layer includes a table to store all the transactional data of the Energy management system. The transactional table has minimal number of columns to reduce the storage size. And the transaction table will join the configuration tables and master data tables to provide the required data for the output layer.
- The output layer is designed to be output interface with a view. As a result, the web pages will be able to fetch treated data, without worrying the data structures in the database.

Detailed database design

The Energy management system database is designed to receive the raw data from the device API connectors and provided treated data to the web pages. To visualize how the data flows from input to out, the database has been structured into six layers in below diagram:



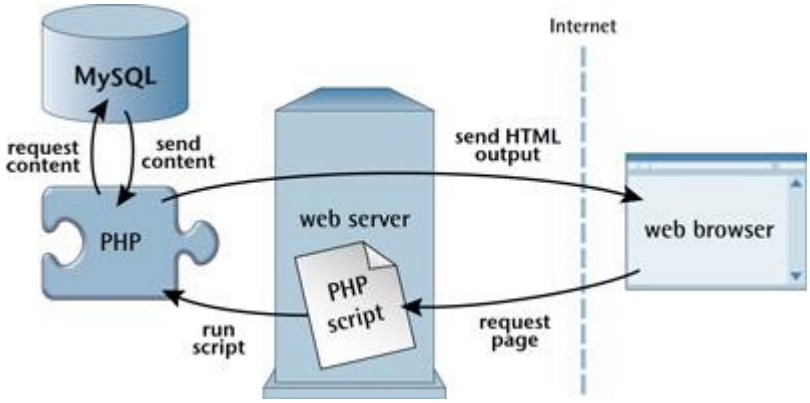
- Input layer – the ems_transaction_ext_table is built to receive and store the raw data from the API connects. When the record is inserted at the input layer, the inserted record will be transformed and passed to transaction layer.
- Transformation layer – the mapping tables (EMS_Device_Mapping_Table and EMS_Attribute_Mapping_Table) are built to translate the external ids to internal ids which can be easily understood by the end users.
- Master data layer – the master data (EMS_Device_Table and EMS_Attribute_Table) are built to store the information speficied by the end users
- Configuration layer – the EMS_Combination_Table allows end users to configure the Energy Management system to analyzation and comparation requirements
- Transaction layer – the EMS_Transaction_Table all the transformed historical data from the energy devices.
- Output layer – the EMS_Transaction_View is built as the API for the web pages. In the view, the output layer will join the tables from master data layer, configuration layer and transaction layer.

1.3. Lifsania

1.3.1. Server – PHP

Overview

The main task of this section is to provide the bridge between the database and the front end. In other words, fetching the data from the database into the front end so that it can be better visualized. The technology used to implement this task is PHP which is well known due to its general-purpose scripting language.



Design

The energy management system is required to fetch data from the database so that the user can see the energy created and used. There are two ways to tackle this. The first would be to fetch the data every 24 hours or the second option, which is that the data would be fetched every time the page is refreshed. After further understanding of the requirements, it was clear that the second option would be best so that the user can have an instant update of the energy levels. This would mean that the user will have view over the data and be able to visualize how the data has changed throughout the day in real time rather than just seeing the past 24-hour data.

Implementation

A new file created separately for the connection so we can refer it later inside the code. Variables are created to hold the database server name, username, password, and the database name after which we use another variable called conn to perform the connection with help of the inbuilt function mysqli_connect. Next, in the code file, we can include the path of the connection file. Many web applications including this application, would expect external data in JSON format. We convert the data from MySQL to JSON format using PHP. The result of the query is saved in a variable called result. An array called json_array is created which will be holding the data from the database. The data is first converted into associate array which will be later converted into json format with the help of the inbuilt function json_encode. This data can now be used to visualize in the front end.

1.4. Willem

1.4.1. Front end

Overview

The user-facing portion of the Energy Management System is a single page web app which dynamically produces visualizations of the energy data being passed to it. This interface has two main visualization types, a “real-time” hub-and-spoke network view and line graph view. Ultimately these visualizations are required to have a simulation feature wherein they rapidly replay values over the last 24 hours. Additionally, the user will be able to toggle the line graph view to access various timescales such as weekly and monthly data. The interface must be deployable as an interactive and non-interactive display in public spaces and the visualizations should update automatically as data comes in. Uncertainty surrounding the availability of data sources such as those in EZONE, as well as the possibility of an expanding scope for this project make flexibility and robustness core requirements of the UI design.

Design

The front-end prototype has been designed as a proof-of-concept for the use of d3.js, SVG and HTML to fulfil UI requirements and allow for rapid development. The app receives curated data from the EMS database via server-side PHP and renders it dynamically using D3.js as SVG vector objects. D3, much like react.js or vue.js, servers to bind dynamic data to HTML DOM elements but unlike those other libraries, D3 specializes in visualizations using SVG. Give this team’s lack of web development experience, a key component of this prototyping phase was the development of a bespoke development environment using Node Package Manager, Webpack and Express.js. This allows for local web server hosting and rapid prototyping. As data flexibility is key, the front-end prototype has been built to parse data from arbitrary devices as long as the incoming JSON objects are valid (Which is ensured at various stages of the data pipeline). Given that most data sources are unavailable during this prototyping stage, mock data is generated to simulate various device configurations.

1.4.2. Project Management

Timesheet Summary

Organization

Team Meetings/ Management.....16

Client Meetings/Management.....4

External Stakeholder Management.....1

Total.....21

Deliverables

Project Specification.....15

Prototype.....24

Essay.....35

Total.....74

Professional Development

Project-specific research.....8

Client-side learning and research.....20

Server-side learning and research.....2

Total.....30

Total: 125 hours

Project Plan: Overview

Stage	Tasks	Deliverables	Planned Start Date	Planned End Date
Project initialization	Project team building	Team members confirmation	**27/07/2020**	2/08/2020
	Project Selection with the project sponsor	Project confirmation	3/08/2020	9/08/2020
Requirements gathering and analysis	Project requirements gathering	Project requirement confirmation	10/08/2020	16/08/2020
	Requirement Analysis		17/08/2020	23/08/2020
	Feasibility study	Feasibility Analysis document	24/08/2020	30/08/2020
System Design	Solution Design overview documenting	Solution diagram	31/08/2020	9/09/2020
	Functional specification documenting	Functional specification document	31/08/2020	9/09/2020
	Technical specification documenting	Technical specification document	31/08/2020	9/09/2020
	Project plan documenting	Project plan	31/08/2020	9/09/2020
	Spec and project plan confirm with the project sponsor	Confirmed specification and project plan	9/09/2020	9/09/2020
	Functional and technical specification Documentation	**Functional and technical specification Project plan**	9/09/2020	**13/09/2020**

Prototyping	Development tools selection	Development Environment	14/09/2020	20/09/2020
	Database Structure Design	Database Table structure document	21/09/2020	27/09/2020
	User interface design	User interface Design document	28/09/2020	4/10/2020
	Prototype development		5/10/2020	11/10/2020
	Prototype review with the project sponsor	Feedback from the project sponsor	12/10/2020	18/10/2020
	Prototype completion	**Prototype presentation and prototype**	19/10/2020	**25/10/2020**
Sprint 1 Implementation	Database and web application establishment	Database and website	**22/02/2021**	28/02/2021
	System framework development	System framework	1/03/2021	7/03/2021
	Testing and bug fixing	Test Evidence	8/03/2021	14/03/2021
	Sprint deployment and System framework Review	Sprint 1 Deliverable	15/03/2021	21/03/2021
Sprint 2 Implementation	Connect existing devices to the system		22/03/2021	28/03/2021

	Testing and bug fixing		29/03/2021	4/04/2021
	Sprint deployment and existing devices connection review	Sprint 2 Deliverable	5/04/2021	11/04/2021
Sprint 3 Implementation	Connect to Ezone or build industry standard API		12/04/2021	18/04/2021
	Testing and bug fixing		19/04/2021	25/04/2021
	Sprint deployment and Sprint 3 Review	Sprint 3 Deliverable	26/04/2021	2/05/2021
System Integration Testing	Test cases documenting and SIT	Test cases	3/05/2021	9/05/2021
	SIT completion	SIT Evidence	10/05/2021	16/05/2021
Go-Live	Deployment to REV Portal	**The working system** **Source code hosted on a shared GitHub repository** **User manual for the system** **Video presentation**	17/05/2021	**23/05/2021**

Project Plan: Contributions

Stage	Willem Meyer		Jason Chu		Lifsania Francis Xavier Robin		Yiwen Li	
Project initialization	Team kick off meeting	2 hrs	Team kick off meeting	2 hrs	Team kick off meeting	2 hrs	Team kick off meeting	2 hrs
	Project analysis and discussion	6 hrs	Project analysis and discussion	6 hrs	Project analysis and discussion	6 hrs	Project analysis and discussion	6 hrs
Requirements gathering and analysis	Question preparation and interview meeting to gather requirements	6 hrs	Question preparation and interview meeting to gather requirements	6 hrs	Question preparation and interview meeting to gather requirements	6 hrs	Question preparation and interview meeting to gather requirements	6 hrs
	Requirement analysis and discussion	4 hrs	Requirement analysis and discussion	4 hrs	Requirement analysis and discussion	4 hrs	Requirement analysis and discussion	4 hrs
	JavaScript research for presentation	10 hrs	Python research for API connector	10 hrs	PHP research for Web application backend	10 hrs	Database research for data storage	10 hrs
System Design	Solution Design overview documenting	12 hrs	Solution Design overview documenting Peer review	2 hrs	Solution Design overview documenting Peer review	2 hrs	Solution Design overview documenting Peer review	2 hrs
	Functional specification documenting Peer review	2 hrs	Functional specification documenting Peer review	2 hrs	Functional specification documenting	12 hrs	Functional specification documenting Peer review	2 hrs
	Technical specification documenting Peer review	2 hrs	Technical specification documenting	12 hrs	Technical specification documenting Peer review	2 hrs	Technical specification documenting Peer review	2 hrs
	Project plan and scopes documenting	12 hrs	Project plan and scopes documenting Peer review	12 hrs	Project plan and scopes documenting Peer review	12 hrs	Project plan and scopes documenting Peer review	12 hrs
	Spec and project plan confirmation and discussion with project sponsor	2 hrs	Spec and project plan confirmation and discussion with project sponsor	2 hrs	Spec and project plan confirmation and discussion with project sponsor	2 hrs	Spec and project plan confirmation and discussion with project sponsor	2 hrs
	Functional and technical specification and project plan finalization	2 hrs	**Functional and technical specification and project plan finalization**	2 hrs	**Functional and technical specification and project plan finalization**	2 hrs	**Functional and technical specification and project plan finalization**	2 hrs

Prototyping	Communicate with project sponsor to setup the development environment	8 hrs	Setup data connector development environment	8 hrs	Setup PHP development environment	8 hrs	Setup database development environment	8 hrs
	Data structure design peer review	2 hrs	Data structure design peer review	2 hrs	Data structure design peer review	2 hrs	Create table structure and import sample data for system prototype	12 hrs
	Develop JavaScript to display the demo data into web pages	12 hrs	User interface design peer review	2 hrs	Develop PHP to read the data from the database and pass to the JavaScript	12 hrs	User interface design peer review	2 hrs
	Data connector peer review	2 hrs	Develop data connector to fetch the devices' data and insert into database	12 hrs	Data connector peer review	2 hrs	Data connector peer review	2 hrs
	Prototype review with the project sponsor	2 hr	Prototype review with the project sponsor	2 hr	Prototype review with the project sponsor	2 hr	Prototype review with the project sponsor	2 hr
	Prototype system overview documenting and presentation	**12 hrs**	**Data connector design documenting and presentation**	12 hrs	**User interface design documenting and presentation**	12 hrs	**Prototype system database design documenting and presentation**	12 hrs
Sprint 1 Implementation	Manage the team and gather the access into UWA future farm	10 hrs	Create a data connect to fetch data from UWA future farm	10 hrs	Create a PHP and JavaScript web page to display information from the database	10 hrs	Extend the database to store required business data	10 hrs
	Documenting UWA future farm end to end testing cases	10 hrs	UWA future farm Data connector unit testing	10 hrs	Create a PHP backend pages for admin users	10 hrs	Create tables to store backend information	10 hrs
	Perform UWA future farm end to end testing	10 hrs	Bug fixing of UWA future farm data connector	10 hrs	Bug fixing of UWA future farm web pages	10 hrs	Bug fixing of database structure	10 hrs
	Documenting test evidence and generate the release note of sprint 1	10 hrs	Finalize the UWA future farm data connector and deploy into REV project	10 hrs	Finalize the web pages and deploy into REV project	10 hrs	Finalize the database structure and deploy into REV project	10 hrs
Sprint 2 Implementation	Manage the team and gather the access into UWA Human Movement Solar and UWA REV DC Charging	10 hrs	Create data connects to fetch data from UWA Human Movement Solar and UWA REV DC Charging	10 hrs	Update the web pages to allow switch the displaying board of different energy sources	10 hrs	Update the database to store the information from UWA Human Movement Solar and UWA REV DC Charging	10 hrs

	Perform end to end testing of UWA Human Movement Solar and UWA REV DC Charging	10 hrs	Bug fixing of UWA Human Movement Solar data connector and UWA REV DC Charging data connector	10 hrs	Bug fixing of web pages of UWA Human Movement Solar and UWA REV DC Charging	10 hrs	Bug fixing of database basing on the UWA Human Movement Solar and UWA REV DC Charging	10 hrs
	Documenting test evidence and generate the release note of sprint 2	10 hrs	Finalize the UWA Human Movement Solar connector and UWA REV DC Charging connector and deploy into REV project	10 hrs	Finalize the web pages and deploy into REV project	10 hrs	Finalize the database structure and deploy into REV project	10 hrs
Sprint 3 Implementation	Manage the team and gather the access into Ezone	10 hrs	Create data connects to fetch data from Ezone	10 hrs	Update the web pages to allow display Ezone information	10 hrs	Update the database to store the information from Ezone	10 hrs
	Perform end to end testing of Ezone	10 hrs	Bug fixing of Ezone data connector	10 hrs	Bug fixing of Ezone web pages	10 hrs	Bug fixing of database basing on the Ezone data	10 hrs
	Documenting test evidence and generate the release note of sprint 3	10 hrs	Finalize the Ezone data connector and deploy into REV project	10 hrs	Finalize the Ezone web pages and deploy into REV project	10 hrs	Finalize the database structure and deploy into REV project	10 hrs
System Integration Testing	Prepare system integration test cases for the whole system	10 hrs	Prepare system integration test cases focusing on data connectors	10 hrs	Prepare system integration test cases focusing on web pages	10 hrs	Prepare system integration test cases focusing on database structure	10 hrs
	Perform system integration test	10 hrs	Perform system integration test and bug fixing for data connectors	10 hrs	Perform system integration test and budget fixing for web pages	10 hrs	Perform system integration test and bug fixing for database	10 hrs
Go-Live	**Project closure documenting and presentation**	**10 hrs**	**User manual of data connector and final deployment into REV project**	**10 hrs**	**User manual of web pages and final deployment into REV project**	**10 hrs**	**User manual of database and final deployment into REV project**	**10 hrs**
Total		**228 hrs**		**228 hrs**		**228 hrs**		**228 hrs**