Team 5

NGRY HUB virtual testbed

November, 2019.

1. Purpose

This design document is produced to capture all the important design decisions that we have decided for this project. The purpose of this document is that the reader should provide a good basis for understanding the implementation that will be done. The document is written in a way that a reader with no knowledge about the project, can easily understand in order to be able to join the team and start to maintain the system further.

For now, this document is a first version and can come to be evolved later when the implementation is done.

2. Client and project

Our client is Future Energy Center, MDH. The European Regional Development Fund project, NRGY HUB, aims at creating a HUB for innovative energy solutions, and one important part of the project is a virtual testbed that will digitally describe a city with open data on its various flows with regards to energy and water. The city that will be digitally describe is Västerås, and in order to accomplish that a map will be used. Due to GDPR regulations the data on energy and water usage can only be displayed by clustering at least five close buildings together, and then calculate the average water and energy usage within each cluster. The displayed data will be averaged water and energy usage within each cluster.

3. System Design

3.1. System main parts and communication

Our system consists of two databases and one webpage. The first database consists of the raw data that has been provided by the client. The data consists of one Excel file (at the moment) but our database is ready to get a connection with the client's database in the future, as required by the client. The data from the Excel file will be exported in JSON format in order to fit the collection requirements of the database.

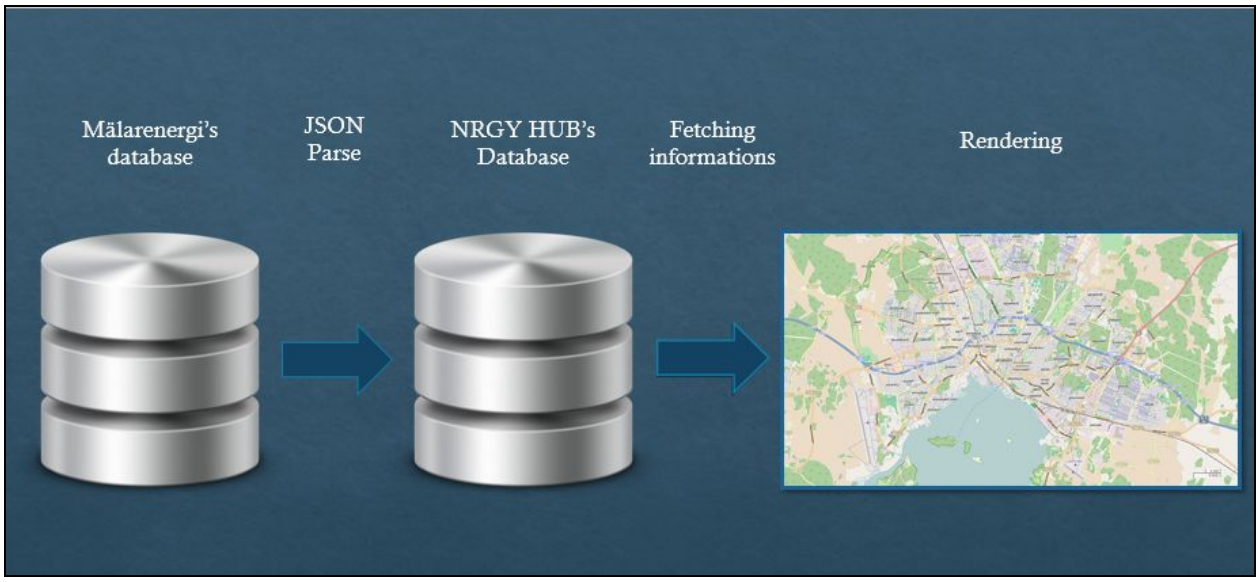
The second database we have created contains all the information on the different users that have created an account for the system. When a user tries to login into the webpage, the credentials will be sent to the database. The database will check the entries to find that user. If no user is found, the page will redirect him to the login page where an error message will tell him that the account does not exist and that he needs to create one.

After the user is logged in, the user will be able to see a map with different clusters. When one cluster is clicked, the webpage will send a request to the NRGY data collection in order to find information about that cluster. The information will be sent back as JSON string to the webpage that will display those beside the map. This will be repeated as long as the user wants to use the webpage.

3.2. Communication with external systems

Our system does not have to interact with external system at the moment. However, the request of the client is to make this possible for the future. The data that the user will include in the database will be provided from an energetic company located in Västerås (Sweden) called Mälarenergi. The data from their database will be sent to ours. As we use mLab as database system, we know that the collection of data that is present in the database will be in JSON. It means that in order to get good communication between Mälarenergis database and ours the data will need to be parsed to JSON before they are sent into our program's database.

The data collected will be displayed on our own webpage that will present a map and different clusters with information of energy consumption and district heating. The map that will be imported in our project will be an offline copy of a "free to use"-component called Open Street Map. In order to use their map without compromising other maps we will download an offline copy of the map and use that.



Picture 1. Communication in the system

3.3. 3rd party frameworks

For the system we are using the following technologies, IDE and third party framework:

Technologies	IDE	Third party
React JS	VS Code	Open Street Map
Java	IntelliJ	mLab (Hosting)
MongoDB		
JSON		

When editing Front-End we're using VS code meanwhile Back-End is used in IntelliJ. To populate the database we use a JSON parse tool and for hosting the database we use mlab. For displaying clustered data we use Open Street Map.

3.4. Structured Implementation

The implementation of our application is structured in Front-End and Back-End. In the future implementation we are planning to connect these two modules.

The Front-End part is conformed by the following folders and files:

- **Public** - folder where all the static files reside.
    - **favicon.ico**
    - **Index.html** - is the page template with main div (id = “root”)
    - **Manifest.json**
- **Src** - Folder where all the dynamic files reside. All .css and .js code have to be in this folder.
    - **App.css** - The style of the main page is defined in this file.
    - **App.js** - The layout of main page is defined in this file. So everything that will be rendered in the div with id=”root”.
    - **App.test.js** - In this file tests for the app will be defined.
    - **Index.css** - The style of application should be in this file.
    - **Index.js** - This file represents the JavaScript entry point.
    - **serviceWorker.js** - This lets the app load faster on subsequent visits in production and gives it offline capabilities. However, it also means that developers will only see deployed updates on subsequent visit to a page.
- **.gitignore** - file that we use for defining module that we want to be ignored when pushing repository on Github.
- **README.md** - contents the information related to the authors and also the modules that are required in order to deploy the project and execute it.
- **package-lock.json** - package-lock.json is automatically generated for any operations where npm modifies either the node_modules tree.
- **package.json** - package-lock.json is automatically generated for any operations where npm modifies either the node_modules tree.
- **yarn.lock** - Yarn caches every package it downloads so it never needs to download it again. It has more features than npm so if somebody is used to yarn start for running project, they can do that.

The Back-End part is conformed by the following main folders and files:

- **Src**
    - **Main**
        - **Java** - This folder contains the main class for running the application where is specified the way of how the app is going to work and things that are initialized at the beginning. All collections, repositories, controllers, services are written in this folder, separated into four different folders:
            - **Collections** - In this folder structure of each collection we are going to have in the database is made in separate classes.
            - **Controllers** - In this folder controllers per collection are defined using services per collection.

        - **Repositories** - In this folder repositories per collection are defined as interfaces and extended with MongoDBRepository because we are working with MongoDB so it was easier to use their Repository as foundation for creating ours.
        - **Services** - In this folder services per collection are defined with basic CRUD operation that we need.
    - **Resources** - This folder contains one crucial file for defining properties of application as well as json files that are imported in the beginning to database as collections.
        - **application.properties** - definition of properties of application and database connection string.
        - **users.json** - json file with some users to fill collection in database.
  - **Test** - Folder where we can specify test and run them in order to test our services.
- **README.md** - contents the information related to the authors and also the modules that are required in order to deploy the project and execute it.
- **mvnw** - This file is imported from Maven Wrapper. It allows user to run the Maven project without having Maven installed and present on the path.
- **mvnw.cmd** - This file is imported from Maven Wrapper. It allows user to run the Maven project without having Maven installed and present on the path.
- **nrgy.iml** - This file stores information about a development module, which may be a Java, Plugin, Android, or Maven component; saves the module paths, dependencies, and other settings.
- **pom.xml** - This file contains all dependencies of our project.

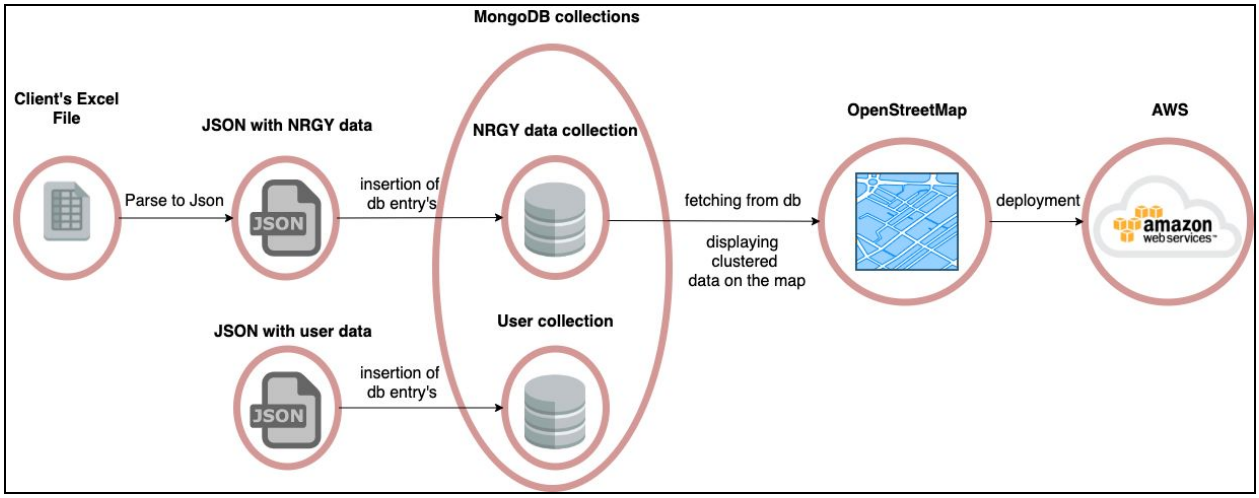
Links for the repositories https://github.com/nejrabahtic/NRGY-HUB-Virtual-TestBed-FrontEnd and https://github.com/nejrabahtic/NRGY-HUB-Virtual-TestBed-BackEnd for the frontEnd and BackEnd folders respectively.

3.5. How do the parts collaborate to provide the functionality (data and control flow)?

The systems is formed by several parts that collaborate together by transmitting data from one part to the next in order to provide the required functionalities.

In the beginning a script to parse data from Excel document to JSON document needs to be created. That script is going to be useful later to fill the database. When the project starts it will initialize the transfer from the JSON documents to the database. It actually creates two collections user and NRGY data and fill them with the data provided in those JSON documents. This part is hardcoded for now, because the registration page is not functional and therefore new entries are not possible through the app for now. When the registration page is finished, it will be able to register on our system and only that data will be saved into user collection in the

database. With the data imported into NRGY data collection, clusters needed to be defined and based on the clustering process, that data will be presented on map on the page that the user can see after finishing with registration and login processes. After the map is filled with the data, users will be able to click on the pins on the map and see the clustered data. In the end, this application should be deployed in the Amazon Web Services (AWS) and running there.

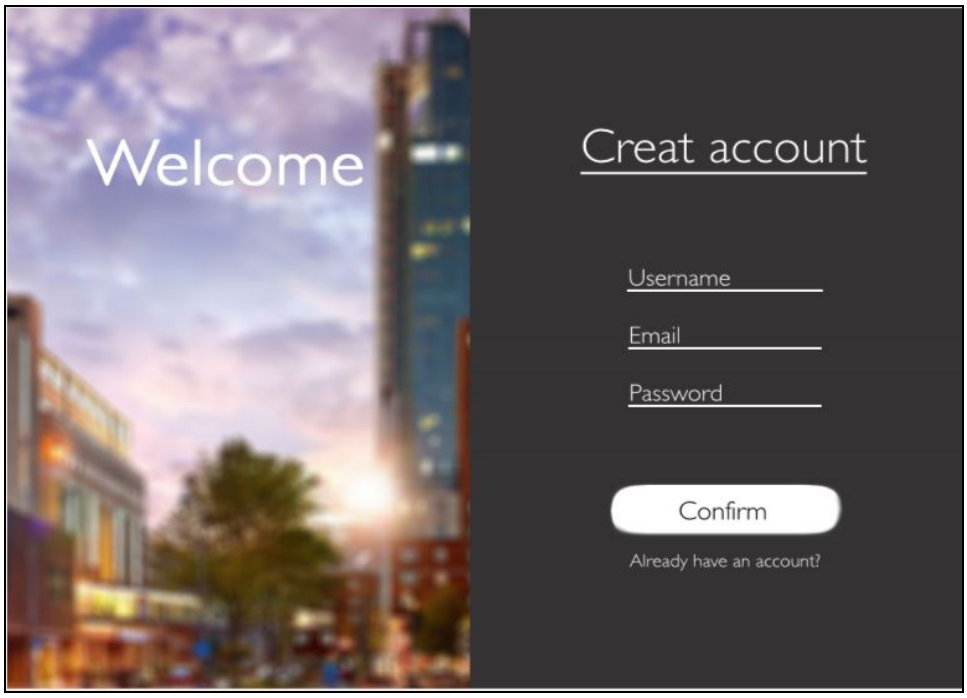


Picture 2: Diagram that shows the flow of the data between parts of the application

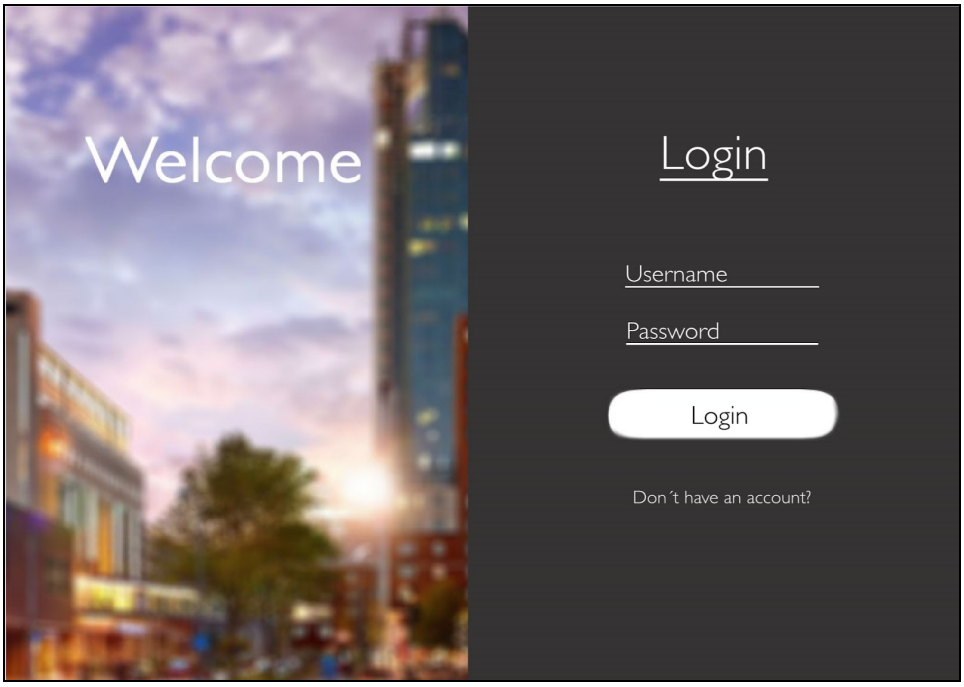
4. How will your system be used?

4.1. GUI structure

The website consists of a total of three different pages. When first enter the website, a login page will be displayed. There the user is required to fill in username and password. If the user do not already have an account there will be a label with a question about if the user is not registered. When pressing that label, the user will be navigated to the registration page instead. On the registration page there are some required forms to fill in in order to create an account. It also exists a label here to press if you already have an account and only need to login. Then the user will be navigated back to the login page. When logged in, the main page will be reached. The main page consists of a big map that covers most of the screen. In the main page there will be an icon for sign out. When pressing on that, the user will be logged out and navigated to the login page. On the main page, there will also be a side navbar where the information about the area is displayed.



Picture 3. Mockup for the registration page



Picture 4. Mockup for the login page

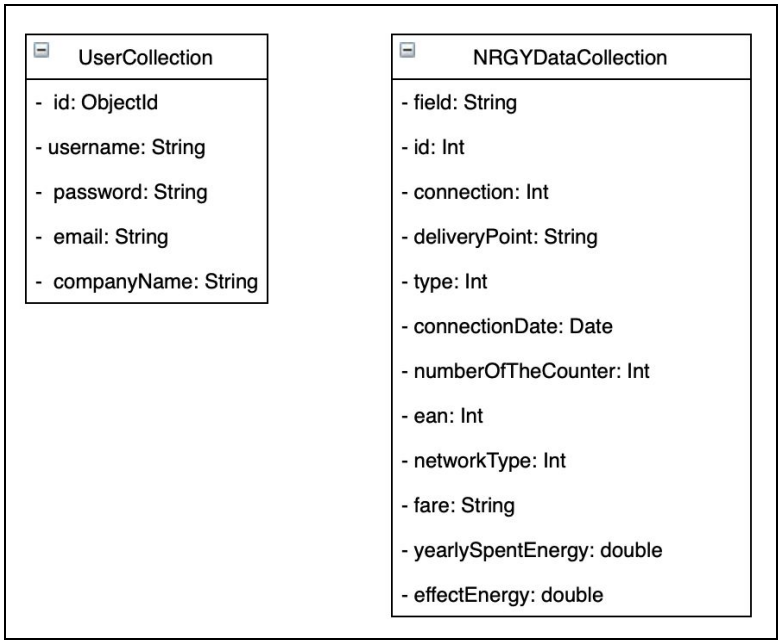
4.2. Interaction between user and system

The NRGY HUB system will be used by registered users. If the user tries to log into the website, the website will ask the NRGY user collection if the account is present. If the user is not registered, the user will need to create an account in order to use the service. If the user is already registered the website will show the map over Västerås and the different clusters that have been created. When the user clicks on one of those clusters, the website will send a request to NRGY data collection and fetch the data of all the buildings in that cluster, create an average of the requested data and will show them to the user. In order to be in line with the GDPR regulations, the system will only show information of at least 5 buildings.

5. Design Description

5.1. Class Diagram

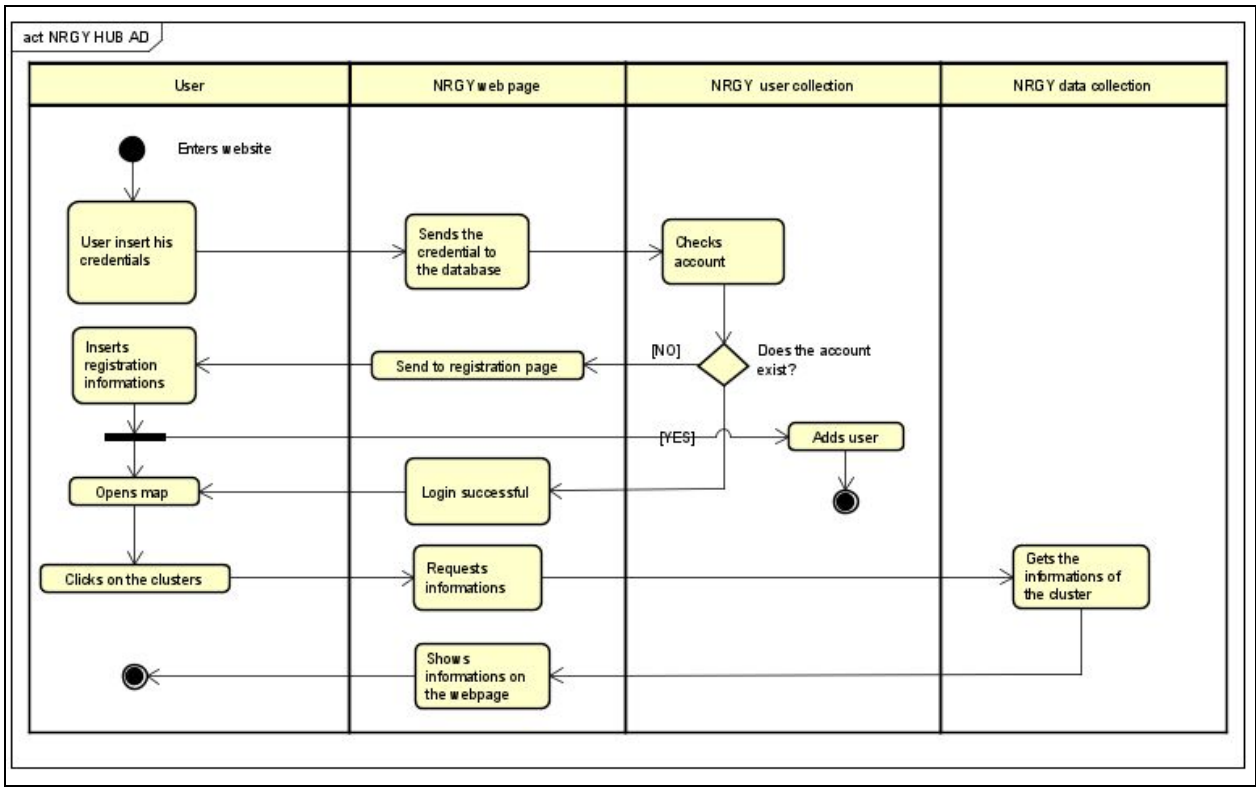
For the project, the modeling of the main class diagram shows the attributes that are going to be used for UserCollection and NGRYDataCollection. Since it is a Non-relational database we do not have a relation between these two collections. Here there can only be seen the attributes of each one.



Picture 5: Class diagram for NRGY HUB system

5.2. Activity Diagram

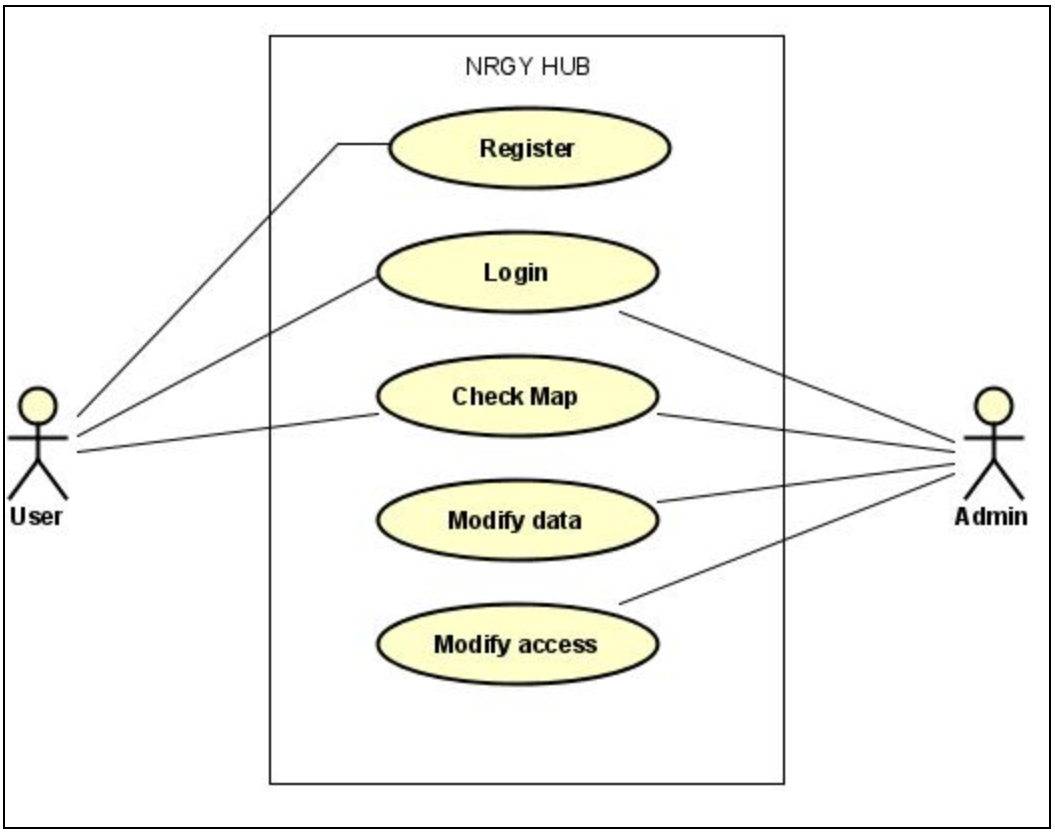
The activity diagram shows the control flow from a starting point in the website login, and the paths that will take through the web page and the database collections while its been executed.



Picture 6: Activity diagram of NRGY HUB system

5.3. Use Case

Use Case shows the interaction that takes part the user on the system, the same as the ones for the administrator. They both have different and share one activity.



Picture : Use Case diagram for NRGY HUB system