****

**Project 2.14**

**Electric Power Consume Management**

Supervisor: Vo Thanh Duy

Student ID: 000901372

Full name: Nguyen Tien Khoa

**Introduction**

Vietnamese Electric Power Company has some problem to collect data from consume monitor device. So, they need a program to collect data information from those devices quicker and cheaper. The program should support to get data about electric consume on monitor device though internet. Moreover, It can help customers to be aware about electric bill. Therefore, company can track consume electric of their customer then give good advices.

Because of using internet to collect data, the system must provide sever. And all devices must to have internet connection. But, device works 24/7 to monitor electric consuming, internet connection would be interrupted and device would be down sometimes by some reason. So, server should have a strategy to deal with offline device.

Surely, monitor devices are not perfect. They may be offline, conflict data, dirty data, etc. So, server should be save these problems to write report. Report will be read by administrators, the electric engineers. Then, they can track to solve problems faster and better. Therefore, the new system will satisfy customer and improve productivity and quality.

In short, the new system manage monitor device to collect data from customer then print it to report. It helps Vietnamese electric company manage and support electric consuming customers.

**Open point**

Customers regard to their money, product and service. Example, they need to know when electric bill become expensive, why electric lost, etc. So, system should have SMS, email or something to notify customer about their information.

Customers should able to use the new system to interact and discuss about problems and needs.

**Analysis**

Base on requirement of this project and my experiences in Java, Java programming language is used to develop. Focus on available tools, low price and fast development, this project includes 3 modules to contain web service, device simulator and share project.

Web service, this is the main project.

Device simulator, this is project to simulate real device. Actually, I don't have any real device, this simulator would support to make system be visible.

Share project, this contain model classes, constant is used by both service and device.

**Resources analysis**

**Human resources:**

This project is a personal coursework, therefore I plan to develop by one Java developer.

**Time resources:**

Deadline is November 24, 2016

**Tools resources:**

In the first product, this application will be used for education purpose therefore I will use all the application or other related to freeware.

Java programming language is used to develop, because there are a lot free Java frameworks and libraries.

MySQL Database has community version is free to download.

Linux OS work faster with Java, and it is free. I developed this product on Linux OS environment, Ubuntu 16.04 LTE base on Debian and Fedora 24 base on Red Hat.

Eclipse neon is IDE used to develop this product. Of course, it is totally free.

All frameworks, libraries and software are used to developed this project belong to Apache Software Foundation, Red Hat Community, Software in the Public Interest Inc, Pivotal Software, Oracle Corporation, The Eclipse Foundation and PrimeTek Informatics.

**Requirement analysis**

Program must use java open source technique for keeping low price.

Program must delivery within 3 months with main functions.

Program must use an incremental development model for integrating other requirements.

Program must allow device to register into system automatically.

Program must be limited access, Administrator only on this version.

Program must allow admin to modify customer data.

Program must have a report reviewer, such as dashboard.

Program must have calculator to built electric bill on demand.

Program should validate device information to track problems on devices.

Program would allow admin to modify consume policy data.

Program would allow administrator edit some properties on device.

Program should allow registering new user account.

Program should update device on database when device reconnect.

Program should poll devices every 10 minutes, in this version.

Device Simulator should able to simulate a device error.

Device Simulator should able to be polled by server.

Device Simulator should able to send data to server.

Device Simulator should able to store data in internal memory, such as files.

**Technology analysis**

**Spring Framework:** System must have server to provide service. With the main target is faster and cheaper, using open source library to build the service. Base on those target, Spring framework seem to be the good choice with wide technique support and easy to use. On top, it is a famous Java framework is used by many java projects. Spring framework has a lot of libraries with strong support other technology. In this project, Spring framework support Spring Security, Spring web, integrate with Hibernate, MySQL and JMS ActiveMQ.

**Runnable JAR:** In this article, monitor device is not developed and built so that a virtual device run on JVM is using to simulate real device. I considered to use Spring Shell, but It is a bit heavy with this requirement and I have no experience. So Spring Shell will be expensive. Runnable JAR is the most light weight technique to build device simulator.

**Java Message Service (JMS) and Active MQ:** In this project, deeply knowledge about Network is not require. For connection between device and server, JMS is good technique. ActiveMQ service is build to store, forward and broadcast to device and receive message from devices. Device and server just connect to ActiveMQ for communication through IP Address, port and protocol of ActiveMQ server broker. Therefore, it takes less time in researching new technology. It cost less time, mean it is cheaper. So, ActiveMQ is used to communicate between service and devices. Moreover, Spring framework has good support to JMS and ActiveMQ.

**JavaScript Object Notation (JSON):** There is no important personal information is used. So, JSON is consider to used to convert Java Object to Text for transfer in communication. The reason are easy to use, easy to read, fast speed, light weight. Jackson JSON is used instead of Google GSON because of speed and feature powerful.

**Java Sevlet Facet (JSF) and Primefaces:** It is famous of powerful and less Javascript technology. For user interface, Javascript is difficult to Java developer like me. Primefaces framework is supported to integrate with Spring framework by Spring webflow library. So, with a bit Javascript knowledge and time to research spring webflow and Primefaces, JSF is considered to be best choice to build UI.

By the way, JSF is MVC component-oriented and spring web is MVC event-oriented. In this project JSF is use to make up UI, and their component is spring bean, Spring framework manage all component instead of JSF.

The benefit of using spring bean to manage is the strength of Dependency Injection, @Inject or @Autowired annotation. This support help spring bypass arguments to a component easily. By Using this, I can manage components and services better.

**MySQL:** System have Database server to store device information and customer information. MySQL is used to build a database server. In future, with a huge database, the service should separate into 2 server, one other will use No-SQL data like mongoDB for lighter and faster query device information. Because the security and restrains on device information is not import.

**Hibernate:** It is a famous framework support Java Persistence Access(JPA). There is no reason to avoid Hibernate, this easy to use and powerful. But, I consider to remove it when separate data into SQL Database and No SQL database. In this case, Spring Database is good choice to support JPA because It supported both JPA and MongoDB.

**Spring webflow and JSF:** As I mentioned in JSF, Spring webflow is the solution to integrate JSF with Spring web MVC. Spring web MVC and JSF are 2 difference framework use Model View Controller (MVC) Architecture, so that they need a midleware to work together. Moreover, Spring webflow also support to control life cycle and navigation such as define variable and view, follow step and check state. In this version, I just use Spring webflow to integrate Spring web MVC and JSF, especially Primefaces.

**Spring Security:** Spring security support protect server from unexpected access. Encryption is support by Spring security by many algorithm such as MD5 on 32 characters and Bcrypt on 64 characters. Of course, It supports access with JPA database.

**Apache Maven:** A build tool from Apache, it must work with other project come from Apache Software Foundation. Apache has a repository, which provide many java libraries. Other build is strong and famous as Maven is Gradle, but I am familiar with Maven. Therefore Maven is the final choice to build this project.

**Design**

**­­Use case Diagram**

User can access in to system through authentication system. After that, user with role "ADMIN" can change device information, change customer information, change policy setting, poll device and check consume bill.

Data storage is use to store information of devices, settings and customer.

System process request from users and devices. Then, the result is store in data storage.

System collects and calculates information of device for providing chart report to user.

**Workflow diagrams**

**Data flow:**



- Data is store in data server MySQL, EPCM System access and save data to data server.

- EPCM System sends message to Device through JMS broker. JMS broker is ActiveMQ server, store and forward message from system to device. Actually, System send a broadcast message to broker with MAC address of device contained in message header. Then, message is received by device has MAC address same as MAC Address in message header.

- Agent is software run on device. It helps device send and receive message.

- Monitor device use agent for send request to System. Message is receive by ActiveMQ server then forward to system. In case, device response a request, message is sent to a temporary destination create by requester.

For more information, please read Appendix A session JMS Topic and Queue.

**Authentication flow:**



Workflow of user authentication and authorization. Step to get access to management system.

User login to system by providing login information. System check login information from request, if invalid information, system show error message. When information is corrected, system checks role, if user has not role admin, system show access denied message. When use has admin permission, it go to management page. User can log out, to go to login page.

**Change device data flow:**



Workflow of Changing Device Information by system.

User change device on system and do save new device information. New device information will be saved into database. Then a message is send to device for requesting update new device info. In case device is offline, message is stayed on broker until device get connection and get the request message. For dirty prevent, ActiveMQ messages have "time to live" attribute to detect out of date message.

In case, User remove device from manager list, device just been removed from manager list. There is no request to device after device is remove from the list. When have request contains device info from device, device is registered into system again. Receive data flow diagram will discuss about process message from devices.

**Poll device data flow:**



Workflow of poll device data. Poll device is request device report it status and send it data to EPCM System.

When user do poll devices on management list. Message is sent to broadcast and wait for response from receiver. Each message has time to live. After 10 second, if there is no device receives and responses, system save polled devices is down with status offline and IP address is release, e.g. "0.0.0.0" . If message response in 10 second, system will show polling progress for this devices is success. Devices will send its information device after that. Receive data flow diagram will discuss about process message from devices.

**Receive device data flow:**



Workflow of process when system receive data from devices.

When message is receive on system, system will detect type of data.

In case, message is a device basic information, system tried to find confliction with old data such as, consume number is lower, device's IP address conflict with other one. If there is conflictions, System create notification for those devices have conflictions and save into database. After that device is save into database with as a new device or a old one base on MAC address.

In case, message is a device notification, a new device notification is create and save into database.

Open point, Notification should check existed of device or device will send a basic info message before send device notification for ensure that notification is searchable by device.

**Process device notification data flow:**



Device notification is created when data information of devices meet problems, e.g. IP address confliction, device information is illogical, unexpected device turn off. System will create device notifications to track those events. Other words, this feature look like logging, a kind of device diary.

Open point, System should able to request device send is logging file. The logging file must contain more details of device activities.

By the way, device notification is separated into 4 kinds base on severity of notification. Please see Data design for more information.

**Process customer information data flow:**



Customer is created when has a real customer sign up contact to use electric of company. Customer information will store in database and can only be access by administrators of electric power company.

When customer request or other reason, customer information is changed and saved into data. Before save customer information changes, system will verify that customer information for preventing error in system. If customer information changes are invalid with system standard format, there is no changes is saved into database.

Customer information changes should be create, update and remove customer information in database server. In case customer is remove from database, all other data relate to removed customer will be cleared.

**Process consume policy data flow:**



Administrator is possible to change policy configuration, which used to calculate electric consume bill. Therefore, after policy data is changed successfully, all bill must be re-calculate base on new policy.

Base on commercial of this function, it should be carefully verify before do changes saved into database. For example, confirm message is show before system do save into database and process update all consume bill.

Open point, system should verify by enter administrator password before do continue process.

**Entity Class Diagrams**

Entity classes are define base on database entities. They have all attributes of entities and mapped correctly.



There are 9 entity classes: account, account role, customer, customer device, device, device notification, device configuration, consume group and consume policy.

**Entities**

**Account**

Account entity presents user account, which user use to access system.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| **Account\_id** | INT | Primary key and Auto increment.  Identification of account |
| Username | TEXT | Max 50 characters, Unique.  Username used to enter login form |
| Password | TEXT | 64 characters, Bcrypt hashing encryption.  Password used to enter login form |
| Email | TEXT | Max 50 characters, format email ,e.g. sample@yahoo.com.vn.  Email use to contact and enter login form |
| Name | TEXT | Max 50 characters, display account name, optional.  Name use to display on User Interface. if name is not defined. username is used to display. |
| Status | TEXT | Max 10 characters, status of account, e.g. inactive, active, banned |
| Create At | TIMESTAMP | Time of account creation |
| LastActiveAt | TIMESTAMP | Time of successful last log in |
|  |  |  |

Primary key: Account\_id

Account table has relationship many to many with Account Role table. It means one user can have many roles and one role can be assigned to many user.

**Account Role**

Account entity present role of account, used to authorize and grant permissions to user.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| **Role** | TEXT | Primary key, max 10 characters.  Role name of account, e.g. admin, user |
| **Account\_id** | INT | Primary key, Foreign key references to account\_id of Account table  Identification of account, which is assigned to this role. |
| Expired At | Time | Time when this role is expired |
|  |  |  |

Primary key: Combine key by role and account Id

Account Role table has relationship many to many with Account table. It means one user can have many roles and one role can be assigned to many user.

**Customer**

Customer entity present customer, which store customer information in system.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| **Customer\_id** | INT | Primary key, auto increment.  Identification of customer. |
| Firstname | TEXT | Max 50 characters  First name of customer. |
| Lastname | TEXT | Max 50 characters  Last name of customer. |
| Address | TEXT | Max 250 characters.  Location of customer house. Usually, it is same as device location. |
| Contact | TEXT | Max 250 characters.  Contact of customer. e.g. email, phone number, fax or house address |
| Register date | TIMESTAMP | Time, when customer registers to use electric of company. |
| Note | TEXT | Max 250 characters.  Note of customer. something are special of customer, system need to save for better productivity. |

Primary key: customer\_id

Customer table has one-to-one relation with customer-device table. It means one customer has one Customer-Device. In other word, one customer can used one device at a time.

**Customer Device**

Customer device present the which customer is assigned to use the device in which group. In other word, it stores data of usage device information. For example, customer A is use device TheWatcher380 to consume electric for house daily activities.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| **CustomerDevice\_id** | INT | Primary key, auto increment.  Identification of customer device |
| Customer\_id | INT | Foreign key references to customer\_id in Customer Table |
| Device\_id | INT | Foreign key references to device\_id in Device Table |
| ConsumeGroup\_id | INT | Foreign key references to consumeGroup\_id in Consume Group Table |

Primary key: CustomerDevice\_id

Customer device has one-to-one relationship with Customer Table. It means one customer has no or one customer device.

Customer device has one-to-one relationship with Device Table. It means one device has no or one customer device.

Customer device has many-to-one relationship with Consume group table. It means one consume group has no, one or more customer device.

**Device**

Device entity present device, which store device information.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| **Device\_id** | INT | Primary key, auto increment  Identification of device |
| Model | TEXT | Max 20 characters.  Model of device. e.g. Aru7342, TheWatcher380, E103 |
| Version | TEXT | Max 10 characters.  Version of firmware run device. e.g. 1.0.Beta |
| Mac Address | TEXT | 18 characters.  Physical Address of device. e.g. de:1f:50:9f:e4:30, de:1f:50:9f:e4:31 |
| IP Address | TEXT | Max 16 characters.  Internet Protocol Address of device. e.g. 192.74.22.10, 192.74.22.11 |
| ConsumeNumber | NUMBER | Greater or equal 0.  Current consume number of device. |
| OldNumber | NUMBER | Old consume of the last bill payment. |
| Status | BIT | 0 or 1.  Status of device. 0 means 'ON' and 1 means 'OFF' |
| Location | TEXT | Max 250 characters.  Location of device. e.g. 'Line 1742. Quang Trung Software City' |
| LastUpdate | TIMESTAMP | Last time this device is update information to server. |

Primary key: device\_id

Device table has one-to-one relation with device configuration table. It means one device has no or one device configuration.

Device table has one-to-many relation with device notification table. It means one device has no, one or many device notifications.

Device table has one-to-one relation with customer device. It mean one device has one or no customer device.

**Device Notification**

Device Notification entity present notification of device, which stores information about notification.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| **deviceNotification\_id** | INT | Primary key, auto increment  Identification of device notification |
| Device\_id | INT | Foreign key references device\_id in device table |
| Severity | TEXT | Max 10 characters  Severity of Notification. e.g. ERROR, WARN, INFO, NOTICE |
| Description | TEXT | Max 250 characters  Description of notification used to discuss about notification |
| Create On | TIMESTAMP | Time of Notification creation. |

Primary key: DeviceNotification\_id

Device notification table has many-to-one relation with device table. It means one device has no, one or many device notifications.

**Device Configuration**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| **DeviceConfig\_id** | Int | Primary key, auto increment |
| Account\_id | INT | Primary key, Foreign key references to account\_id of Account table |
| Expired At | Time | Time when this role is expired |
|  |  |  |

**Consume Group**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| **ComsumeGroup\_id** | TEXT | Primary key, max 10 characters, role name of account, e.g. admin, user |
| **Account\_id** | INT | Primary key, Foreign key references to account\_id of Account table |
| Expired At | Time | Time when this role is expired |
|  |  |  |

**Consume Policy**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| **ConsumePolicy\_id** | TEXT | Primary key, max 10 characters, role name of account, e.g. admin, user |
| **Account\_id** | INT | Primary key, Foreign key references to account\_id of Account table |
| Expired At | Time | Time when this role is expired |
|  |  |  |

**Implement**

**Maintenance**