CMPE281 Project Report

Name Yuntian Shen

Research Assistant Fuyao Zhang

Course number CMPE 281

Class time Tu 18:00-20:55

Date Nov 21, 2017

Content

1 Overview

My component is to integrate several ProcessMaker for different departments and modularize the processes in them.

As ProcessMaker is an open source process design and control software, departments in government can use it to design its own process, communicate with different people in the department through Email.

However, ProcessMaker has its limitation. For example, government has many departments and each department has its own processes and data. As these processes and data is private, we need to protect these data's privacy. What is more, the ProcessMaker is defined as a process control application, it is weak in relationship representation.

In order to strength the function of ProcessMaker, we decide to write our own application based on ProcessMaker to solve the shortcomings mentioned above.

We let the ProcessMaker as a component of our project and create different instances for different departments. In this way, each department can only see its own data and process, and we develop encryption module to protect the data when transferring on the Internet. We also develop a modularization part to modularize the process designed in ProcessMaker.

I will introduce all functions in detailed in the next chapters.

2 Basic Functions

The basic functions of our system are to combine different instances of ProcessMaker together and support the modularization function of the processes in the ProcessMaker.

2.1 Combination

In the combination aspect, I create different instances for different departments and develop same gateway for all departments to login and route to their own

ProcesMaker instances. In the meantime, in order to protected the data privacy, I develop the data encryption module and users can configure their own encryption key value.

2.2 Modularization

In the modularization aspect, I get the data from the ProcessMaker's database, and then use Vis.js to modularize the tasks in each process, together with the people in charge of a specific task.

3 Users and use cases

3.1 Users

The users defined in our project are employees belong to different departments in the government, for example Homeless Assistance Department (We will implement two departments Homeless Assistance Department and Illegal Dumping Department for demo).

3.2 Use Cases

The system has five main use cases (except for the function of ProcessMaker): login/logout the central system, route to specific ProcessMaker, configure encryption, check the model of the processes, and create new instance (part of it).

3.2.1 Login/Logout

In login/logout use case, the users can login or logout the central system, using the same gateway by choosing the department they belong to and input their user name and password. If all information is right, the user can login to the central system, if any field is wrong, there will be a notification to let use to input the right information. The user can also logout the system.

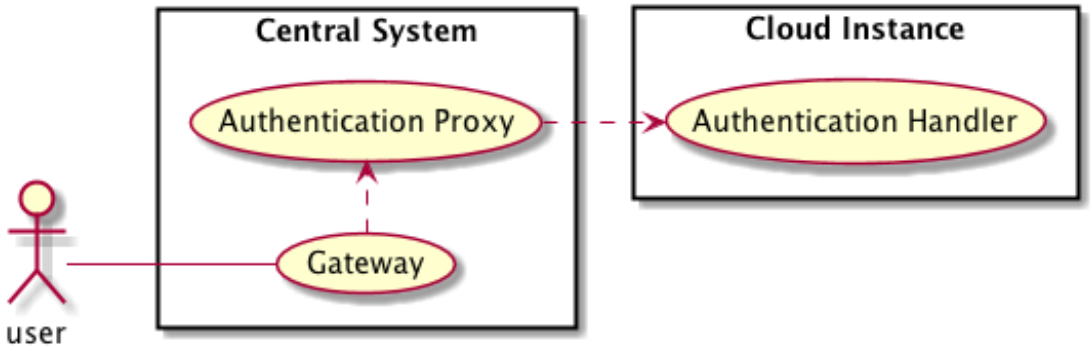

Fig. 1 Login Use Case

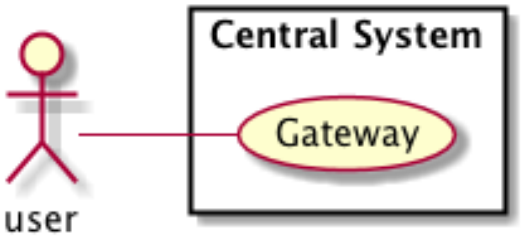

Fig. 2 Login Use Case

3.2.2 Route

In route use case, the users can route to ProcessMaker belongs to their department, and then use the function supported by ProcessMaker.

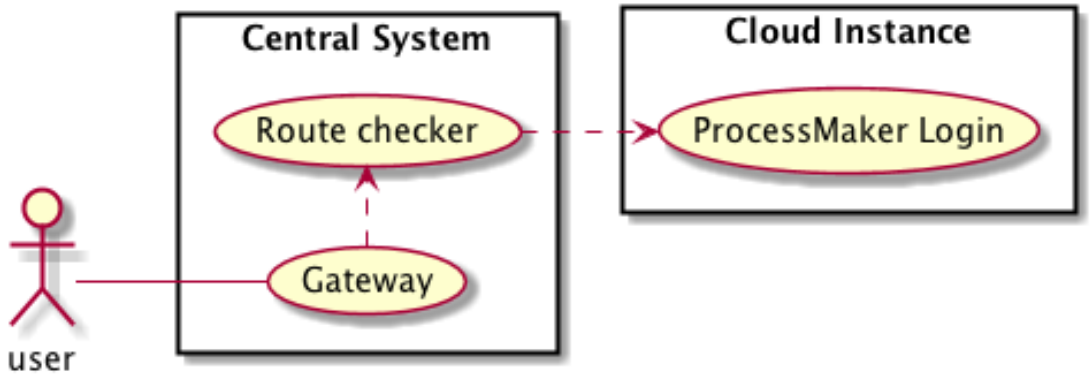

Fig. 3 Route Use Case

3.2.3 Setting Encryption

In setting encryption use case, the users can enable or disable encryption in data transfer. When users enable encryption, they can set their own encryption key value.

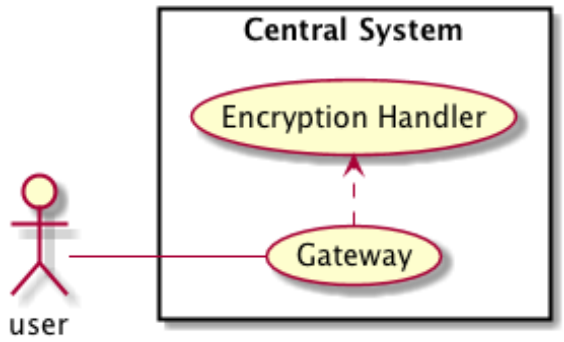

Fig. 4 Setting Encryption Use Case

3.2.4 Checking Model

In checking model use case, the users can check all processes in the department they belong to, for each process, uses can also check the tasks in it and the people in charge of it.

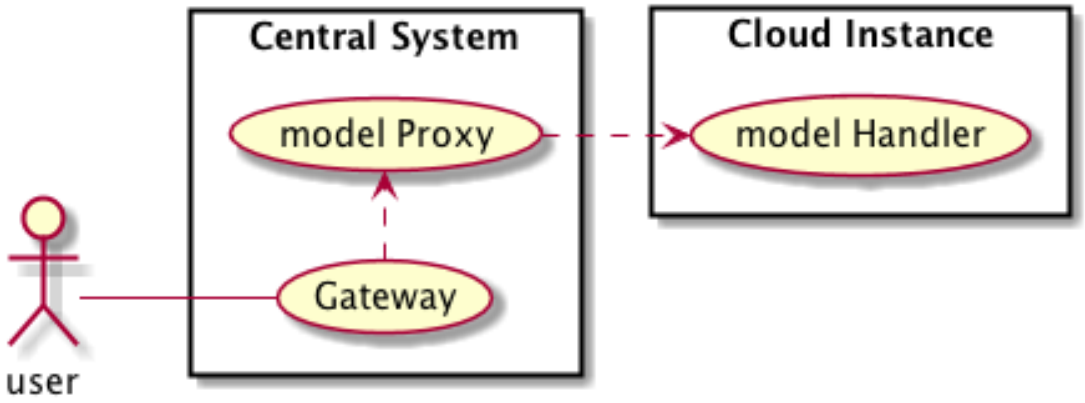

Fig. 5 Checking Model Use Case

3.2.5 Instance Creation

In creation instance use case, administrator can create instance for new departments.

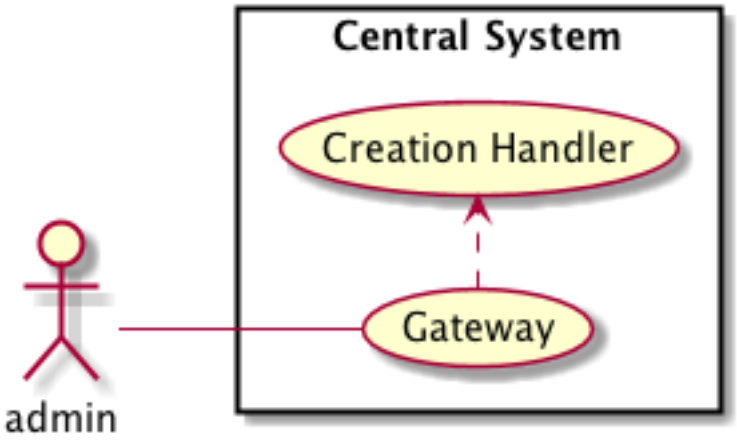

Fig. 6 Instance Creation Use Case

4 Component design

4.1 Component Diagram

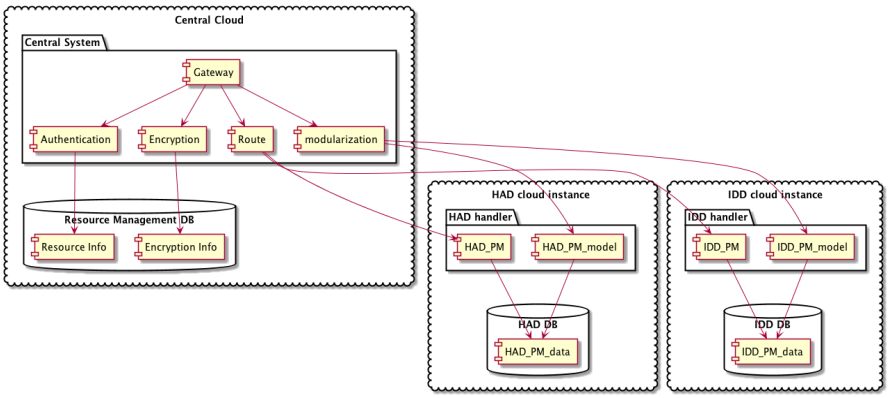

Fig. 7 System Component Diagram

(The HAD is the abbreviation for Homeless Assistance Department, The IDD is the abbreviation for Illegal Dumping Department, The PM is the abbreviation for ProcessMaker)

We can see that the whole system has two type of components: central system and cloud instance (for demo, I create two cloud instance, so in the component diagram will be two different instances, but structures of them are the same). I will

introduce them respectively.

4.2 Central system

The central system is deployed on AWS as an independent instance.

4.2.1 Component Diagram

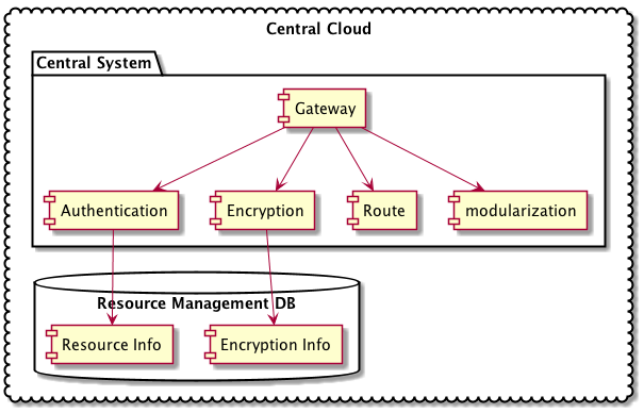

Fig. 8 Central System Component Diagram

4.2.2 Function

The function of central system is to authenticate the users, support the user's encryption, route to right ProcessMaker for specific users, and modularize the processes.

4.2.3 API

The central system has two kinds of API, one kind for browser users to use the system, another kind is restful API for web page to get data.
(1) Web API
"index.html": return Login page for the central system.
"register.html": return register page for creation of new department.
(2) Restful API
"/api/allDepartments": return all department name for login page department selector
"/api/check": validate the input of user for login page
"/api/{department}/processes/{userName}": return processes information of a specific department for user
"/api/{department}/{process}/tasks/{userName}": return tasks information of a specific process for user
"/api/update": use to update encryption key Value
"/api/disable/{department}/{userName}": use to disable encryption function for

a specific user
"/api/create": use to create instance for new department

4.2.4 Data

Central system has its own database and will store its data in it. The database is MySQL.

I design two tables show as below.

```
mysql> show tables;
+------------------------------+
| Tables_in_resourcemanagement |
+------------------------------+
| encryption                   |
| resource                     |
+------------------------------+
2 rows in set (0.00 sec)
```

Fig. 9 Tables in Central Database

The resource table stores the information where the instance is deployed. The schema of the resource table is as shown below.

```
mysql> desc resource;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(40) | NO   | PRI | NULL    |       |
| address | varchar(80) | NO   |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Fig. 10 Resource Table Schema

The encryption table stores all users' encryption configuration. The schema of the encryption table is as shown below

```
mysql> desc encryption;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| department | varchar(40) | NO   | PRI | NULL    |       |
| userName   | varchar(40) | NO   | PRI | NULL    |       |
| keyValue   | varchar(10) | NO   |     | NULL    |       |
| enable     | tinyint(1)  | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Fig. 11 Encryption Table Schema

4.2.5 Algorithms

I will introduce four main algorithms: authentication, encryption, route, and modularization in this part to show how I develop the central system.

(1) Authentication
The authentication algorithm handles the user login. When user login to the central system, the central system will first check which department the user belongs to. If the department exists, the central system will send request to a specific validation API belong to that department to validate the user information, and then return the validation result to the front end.

(2) Encryption
The encryption algorithm handles the user's encryption operations. The user can enable and disable the encryption through then dashboard, when user do these operations, the central system will update the database and change the encryption behavior when sending data to the front end.
If user enable the encryption function, in the back end, the system will use apache's open source library “commons-codec” to encrypt the data, in the meantime, the front end will use Google's open source library “crypto-js” to decrypt the data. In this way, data privacy can be protected when transferred on the internet.

(3) Route
The route algorithm handles the situation in which users belong to different departments. As we hope users have a unified gateway to use our system, the central system has to route users to different cloud instances, and send request to different locations for services for different users (for example: validation users, get process data, and get task data).

(4) Modularization

In the front end, I use open source data visualization library Vis.js to modularize the process, tasks' relationship and users' relationship. In the back end, the central system will not do the actual work to get data from database, it just sends request to cloud instance to get data for specific department or specific process, I will introduce how I implement this in cloud instance in detail in the next cloud instance part.

4.3 Cloud Instance

The cloud instance is deployed on AWS as an independent instance, each department will have its own instance.

4.3.1 Component Diagram

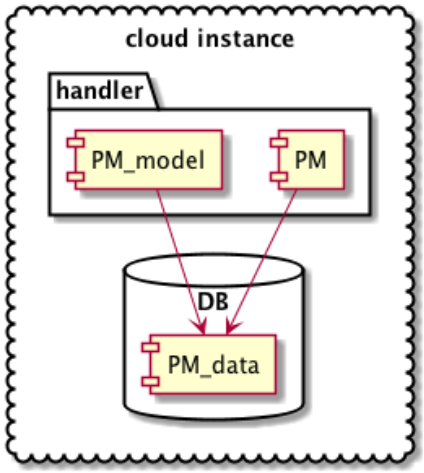

Fig. 12 Cloud Instance Component Diagram

4.3.2 Function

The cloud instance integrates the ProcessMaker and supports our own model part to modularize the process designed in ProcessMaker. The functions of this instance are to let users to design and handle their own process, validate the users's information, and provide process and task data for central system.

4.3.3 API

The cloud instance system has two kinds of API, one kind is supported by ProcessMaker, another kind is restful API for central system to get required data.
(1) ProcessMaker API
Our system routes to ProcessMaker's login API to login to ProcessMaker.
(2) Restful API
"/api/check": called by central system to validate user information
"/api/processes": called by central system to get all processes information from

ProcessMaker's database
"/api/{process}/tasks": called by central system to get a specific process information from ProcessMaker's database

4.3.4 Data

As the instance is developed above the ProcessMaker, we use ProcessMaker's database as our database, the relation schema of the database is as shown below. It can be seen from the figure that the design of tables is very complex, so I won't introduce how all table is organized in this report, but I will explain part of it in the "Algorithms" part to show how I use it.

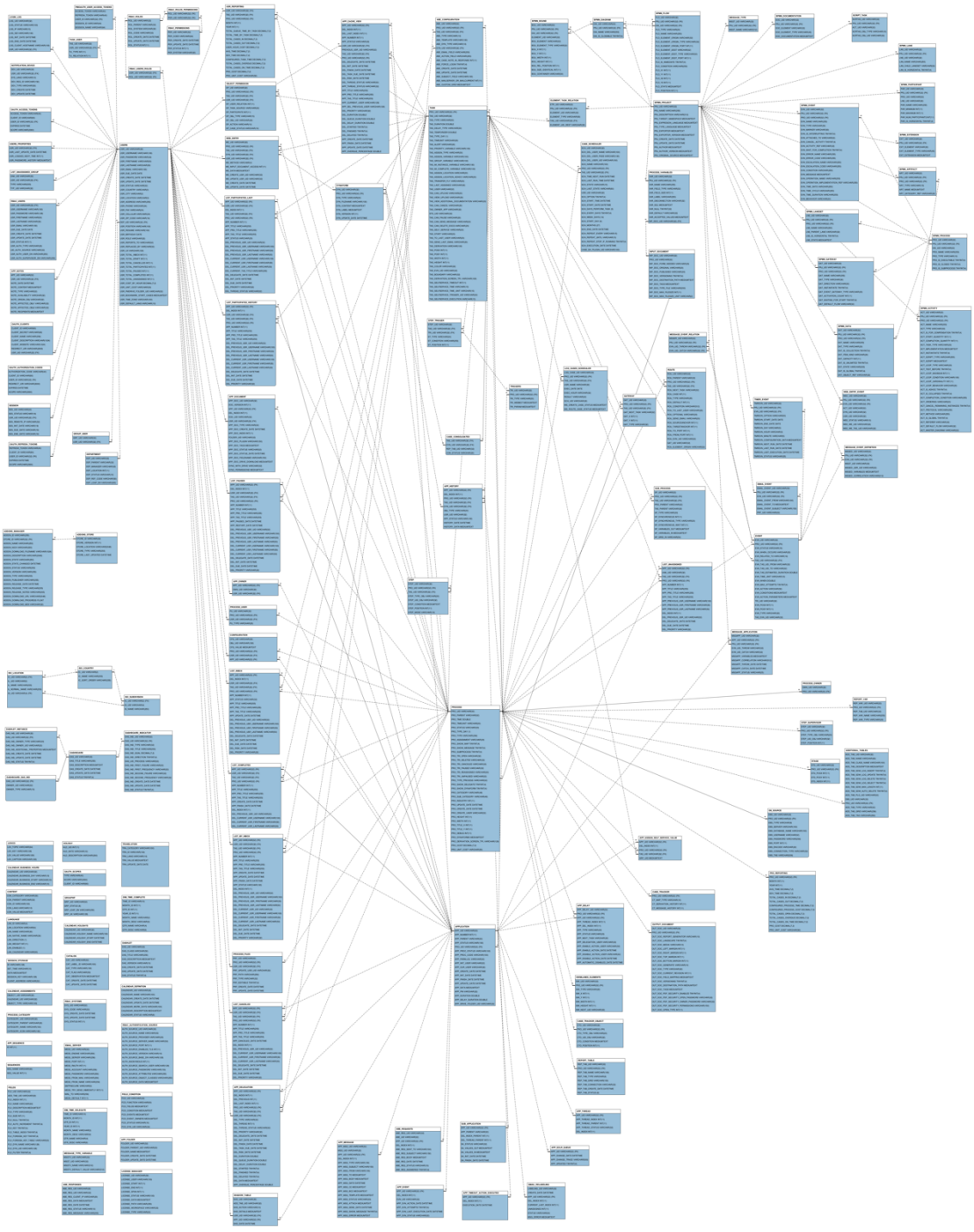

Fig. 13 ProcessMaker's Relation Schema

4.3.5 Algorithms

I will introduce three main algorithms: validation, process data, and task data to show how I develop the cloud instance.

(1) Validation

The validation algorithm handles the validation request from central system. The most important table for this function is "users" table, the schema of this table is

as shown below. The user name and password can be found in this table, however the password is encrypted using SHA256, so we need to encrypt the input password too. After the pre-process, we can compare the user name and password to validate the user.

```
mysql> desc users;
+--------------------------+--------------+------+-----+--------------------+----------------+
| Field                    | Type         | Null | Key | Default            | Extra          |
+--------------------------+--------------+------+-----+--------------------+----------------+
| USR_UID                  | varchar(32)  | NO   | PRI |                    |                |
| USR_ID                   | int(11)      | NO   | UNI | NULL               | auto_increment |
| USR_USERNAME             | varchar(100) | NO   |     |                    |                |
| USR_PASSWORD             | varchar(128) | NO   |     |                    |                |
| USR_FIRSTNAME            | varchar(50)  | NO   |     |                    |                |
| USR_LASTNAME             | varchar(50)  | NO   |     |                    |                |
| USR_EMAIL                | varchar(100) | NO   |     |                    |                |
| USR_DUE_DATE             | date         | NO   |     | NULL               |                |
| USR_CREATE_DATE          | datetime     | NO   |     | NULL               |                |
| USR_UPDATE_DATE          | datetime     | NO   |     | NULL               |                |
| USR_STATUS               | varchar(32)  | NO   |     | ACTIVE             |                |
| USR_COUNTRY              | varchar(3)   | NO   |     |                    |                |
| USR_CITY                 | varchar(3)   | NO   |     |                    |                |
| USR_LOCATION             | varchar(3)   | NO   |     |                    |                |
| USR_ADDRESS              | varchar(255) | NO   |     |                    |                |
| USR_PHONE                | varchar(24)  | NO   |     |                    |                |
| USR_FAX                  | varchar(24)  | NO   |     |                    |                |
| USR_CELLULAR             | varchar(24)  | NO   |     |                    |                |
| USR_ZIP_CODE             | varchar(16)  | NO   |     |                    |                |
| DEP_UID                  | varchar(32)  | NO   |     |                    |                |
| USR_POSITION             | varchar(100) | NO   |     |                    |                |
| USR_RESUME               | varchar(100) | NO   |     |                    |                |
| USR_BIRTHDAY             | date         | YES  |     | NULL               |                |
| USR_ROLE                 | varchar(32)  | YES  |     | PROCESSMAKER_ADMIN |                |
| USR_REPORTS_TO           | varchar(32)  | YES  |     |                    |                |
| USR_REPLACED_BY          | varchar(32)  | YES  |     |                    |                |
| USR_UX                   | varchar(128) | YES  |     | NORMAL             |                |
| USR_COST_BY_HOUR         | decimal(7,2) | YES  |     | 0.00               |                |
| USR_UNIT_COST            | varchar(50)  | YES  |     |                    |                |
| USR_PMDRIVE_FOLDER_UID   | varchar(32)  | YES  |     |                    |                |
| USR_BOOKMARK_START_CASES | mediumtext   | YES  |     | NULL               |                |
| USR_TIME_ZONE            | varchar(100) | YES  |     |                    |                |
| USR_DEFAULT_LANG         | varchar(10)  | YES  |     |                    |                |
| USR_LAST_LOGIN           | datetime     | YES  |     | NULL               |                |
+--------------------------+--------------+------+-----+--------------------+----------------+
34 rows in set (0.00 sec)
```

Fig. 14 The Schema of “users” Table

(2) Process Data

The process data algorithm handles the get all processes request from central system. The most important table for this function is "process" table, the schema of this table is as shown below. We can see that the table store the process name in text as "PRO_TITLE", so we can get all processes' name by get all "PRO_TITLE" columns in the table.

```
mysql> desc process;
+--------------------------+-------------+------+-----+---------+----------------+
| Field                    | Type        | Null | Key | Default | Extra          |
+--------------------------+-------------+------+-----+---------+----------------+
| PRO_UID                  | varchar(32) | NO   | PRI |         |                |
| PRO_ID                   | int(11)     | NO   | UNI | NULL    | auto_increment |
| PRO_TITLE                | mediumtext  | NO   |     | NULL    |                |
| PRO_DESCRIPTION          | mediumtext  | YES  |     | NULL    |                |
| PRO_PARENT               | varchar(32) | NO   |     | 0       |                |
| PRO_TIME                 | double      | NO   |     | 1       |                |
| PRO_TIMEUNIT             | varchar(20) | NO   |     | DAYS    |                |
| PRO_STATUS               | varchar(20) | NO   |     | ACTIVE  |                |
| PRO_TYPE_DAY             | char(1)     | NO   |     | 0       |                |
| PRO_TYPE                 | varchar(256)| NO   |     | NORMAL  |                |
| PRO_ASSIGNMENT           | varchar(20) | NO   |     | FALSE   |                |
| PRO_SHOW_MAP             | tinyint(4)  | NO   |     | 1       |                |
| PRO_SHOW_MESSAGE         | tinyint(4)  | NO   |     | 1       |                |
| PRO_SUBPROCESS           | tinyint(4)  | NO   |     | 0       |                |
| PRO_TRI_CREATE           | varchar(32) | NO   |     |         |                |
| PRO_TRI_OPEN             | varchar(32) | NO   |     |         |                |
| PRO_TRI_DELETED          | varchar(32) | NO   |     |         |                |
| PRO_TRI_CANCELED         | varchar(32) | NO   |     |         |                |
| PRO_TRI_PAUSED           | varchar(32) | NO   |     |         |                |
| PRO_TRI_REASSIGNED       | varchar(32) | NO   |     |         |                |
| PRO_TRI_UNPAUSED         | varchar(32) | NO   |     |         |                |
| PRO_TYPE_PROCESS         | varchar(32) | NO   |     | PUBLIC  |                |
| PRO_SHOW_DELEGATE        | tinyint(4)  | NO   |     | 1       |                |
| PRO_SHOW_DYNAFORM        | tinyint(4)  | NO   |     | 0       |                |
| PRO_CATEGORY             | varchar(48) | NO   |     |         |                |
| PRO_SUB_CATEGORY         | varchar(48) | NO   |     |         |                |
| PRO_INDUSTRY             | int(11)     | NO   |     | 1       |                |
| PRO_UPDATE_DATE          | datetime    | YES  |     | NULL    |                |
| PRO_CREATE_DATE          | datetime    | NO   |     | NULL    |                |
| PRO_CREATE_USER          | varchar(32) | NO   |     |         |                |
| PRO_HEIGHT               | int(11)     | NO   |     | 5000    |                |
| PRO_WIDTH                | int(11)     | NO   |     | 10000   |                |
| PRO_TITLE_X              | int(11)     | NO   |     | 0       |                |
| PRO_TITLE_Y              | int(11)     | NO   |     | 6       |                |
| PRO_DEBUG                | int(11)     | NO   |     | 0       |                |
| PRO_DYNAFORMS            | mediumtext  | YES  |     | NULL    |                |
| PRO_DERIVATION_SCREEN_TPL| varchar(128)| YES  |     |         |                |
| PRO_COST                 | decimal(7,2)| YES  |     | 0.00    |                |
| PRO_UNIT_COST            | varchar(50) | YES  |     |         |                |
| PRO_ITEE                 | int(11)     | NO   |     | 0       |                |
| PRO_ACTION_DONE          | mediumtext  | YES  |     | NULL    |                |
+--------------------------+-------------+------+-----+---------+----------------+
41 rows in set (0.01 sec)
```

Fig. 15 The Schema of “process” Table

(3) Task Data

The task data algorithm handles the get all tasks and people who in charge of each tasks request from central system. The most important tables for this function are process, task, users, group_user, task_user. The schema of process and users and already been shown above, so I will only show the schema of task, group_user, task_user below. We can see that the task table store the process name the task belongs to in “PRO_UID” column, and task_user table store the user who in charge of the task, so we can know the the tasks belong to a specific process and the users in charge of a task. However, these is a trick that the there could be a group of user in charge of a task, so we need to find actual users in both group_user table and

users table. After getting all data from database, the algorithm will process data to form the structure which the Vis.js library need.

```
mysql> desc task;
+-----------------------------------+--------------+------+-----+-----------------------------------+----------------+
| Field                             | Type         | Null | Key | Default                           | Extra          |
+-----------------------------------+--------------+------+-----+-----------------------------------+----------------+
| PRO_UID                           | varchar(32)  | NO   |     |                                   |                |
| TAS_UID                           | varchar(32)  | NO   | PRI |                                   |                |
| TAS_ID                            | int(11)      | NO   | UNI | NULL                              | auto_increment |
| TAS_TITLE                         | mediumtext   | NO   |     | NULL                              |                |
| TAS_DESCRIPTION                   | mediumtext   | YES  |     | NULL                              |                |
| TAS_DEF_TITLE                     | mediumtext   | YES  |     | NULL                              |                |
| TAS_DEF_SUBJECT_MESSAGE           | mediumtext   | YES  |     | NULL                              |                |
| TAS_DEF_PROC_CODE                 | mediumtext   | YES  |     | NULL                              |                |
| TAS_DEF_MESSAGE                   | mediumtext   | YES  |     | NULL                              |                |
| TAS_DEF_DESCRIPTION               | mediumtext   | YES  |     | NULL                              |                |
| TAS_TYPE                          | varchar(50)  | NO   |     | NORMAL                            |                |
| TAS_DURATION                      | double       | NO   |     | 0                                 |                |
| TAS_DELAY_TYPE                    | varchar(30)  | NO   |     |                                   |                |
| TAS_TEMPORIZER                    | double       | NO   |     | 0                                 |                |
| TAS_TYPE_DAY                      | char(1)      | NO   |     | 1                                 |                |
| TAS_TIMEUNIT                      | varchar(20)  | NO   |     | DAYS                              |                |
| TAS_ALERT                         | varchar(20)  | NO   |     | FALSE                             |                |
| TAS_PRIORITY_VARIABLE             | varchar(100) | NO   |     |                                   |                |
| TAS_ASSIGN_TYPE                   | varchar(30)  | NO   |     | BALANCED                          |                |
| TAS_ASSIGN_VARIABLE               | varchar(100) | NO   |     | @@SYS_NEXT_USER_TO_BE_ASSIGNED    |                |
| TAS_GROUP_VARIABLE                | varchar(100) | YES  |     | NULL                              |                |
| TAS_MI_INSTANCE_VARIABLE          | varchar(100) | NO   |     | @@SYS_VAR_TOTAL_INSTANCE          |                |
| TAS_MI_COMPLETE_VARIABLE          | varchar(100) | NO   |     | @@SYS_VAR_TOTAL_INSTANCES_COMPLETE |               |
| TAS_ASSIGN_LOCATION               | varchar(20)  | NO   |     | FALSE                             |                |
| TAS_ASSIGN_LOCATION_ADHOC         | varchar(20)  | NO   |     | FALSE                             |                |
| TAS_TRANSFER_FLY                  | varchar(20)  | NO   |     | FALSE                             |                |
| TAS_LAST_ASSIGNED                 | varchar(32)  | NO   |     | 0                                 |                |
| TAS_USER                          | varchar(32)  | NO   |     | 0                                 |                |
| TAS_CAN_UPLOAD                    | varchar(20)  | NO   |     | FALSE                             |                |
| TAS_VIEW_UPLOAD                   | varchar(20)  | NO   |     | FALSE                             |                |
| TAS_VIEW_ADDITIONAL_DOCUMENTATION | varchar(20)  | NO   |     | FALSE                             |                |
| TAS_CAN_CANCEL                    | varchar(20)  | NO   |     | FALSE                             |                |
| TAS_OWNER_APP                     | varchar(32)  | NO   |     |                                   |                |
| STG_UID                           | varchar(32)  | NO   |     |                                   |                |
| TAS_CAN_PAUSE                     | varchar(20)  | NO   |     | FALSE                             |                |
| TAS_CAN_SEND_MESSAGE              | varchar(20)  | NO   |     | TRUE                              |                |
| TAS_CAN_DELETE_DOCS               | varchar(20)  | NO   |     | FALSE                             |                |
| TAS_SELF_SERVICE                  | varchar(20)  | NO   |     | FALSE                             |                |
| TAS_START                         | varchar(20)  | NO   |     | FALSE                             |                |
| TAS_TO_LAST_USER                  | varchar(20)  | NO   |     | FALSE                             |                |
| TAS_SEND_LAST_EMAIL               | varchar(20)  | NO   |     | TRUE                              |                |
| TAS_DERIVATION                    | varchar(100) | NO   |     | NORMAL                            |                |
| TAS_POSX                          | int(11)      | NO   |     | 0                                 |                |
| TAS_POSY                          | int(11)      | NO   |     | 0                                 |                |
| TAS_WIDTH                         | int(11)      | NO   |     | 110                               |                |
| TAS_HEIGHT                        | int(11)      | NO   |     | 60                                |                |
| TAS_COLOR                         | varchar(32)  | NO   |     |                                   |                |
| TAS_EVN_UID                       | varchar(32)  | NO   |     |                                   |                |
| TAS_BOUNDARY                      | varchar(32)  | NO   |     |                                   |                |
| TAS_DERIVATION_SCREEN_TPL         | varchar(128) | YES  |     |                                   |                |
| TAS_SELFSERVICE_TIMEOUT           | int(11)      | YES  |     | 0                                 |                |
| TAS_SELFSERVICE_TIME              | int(11)      | YES  |     | 0                                 |                |
| TAS_SELFSERVICE_TIME_UNIT         | varchar(15)  | YES  |     |                                   |                |
| TAS_SELFSERVICE_TRIGGER_UID       | varchar(32)  | YES  |     |                                   |                |
| TAS_SELFSERVICE_EXECUTION         | varchar(15)  | YES  |     | EVERY_TIME                        |                |
| TAS_NOT_EMAIL_FROM_FORMAT         | int(11)      | YES  |     | 0                                 |                |
| TAS_OFFLINE                       | varchar(20)  | NO   |     | FALSE                             |                |
| TAS_EMAIL_SERVER_UID              | varchar(32)  | YES  |     |                                   |                |
| TAS_AUTO_ROOT                     | varchar(20)  | NO   |     | FALSE                             |                |
| TAS_RECEIVE_SERVER_UID            | varchar(32)  | YES  |     |                                   |                |
| TAS_RECEIVE_LAST_EMAIL            | varchar(20)  | NO   |     | FALSE                             |                |
| TAS_RECEIVE_EMAIL_FROM_FORMAT     | int(11)      | YES  |     | 0                                 |                |
| TAS_RECEIVE_MESSAGE_TYPE          | varchar(20)  | NO   |     | text                              |                |
| TAS_RECEIVE_MESSAGE_TEMPLATE      | varchar(100) | NO   |     | alert_message.html                |                |
| TAS_RECEIVE_SUBJECT_MESSAGE       | mediumtext   | YES  |     | NULL                              |                |
| TAS_RECEIVE_MESSAGE               | mediumtext   | YES  |     | NULL                              |                |
+-----------------------------------+--------------+------+-----+-----------------------------------+----------------+
66 rows in set (0.00 sec)
```

Fig. 16 The Schema of “task” Table

```
mysql> desc group_user;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| GRP_UID | varchar(32) | NO   | PRI | 0       |       |
| USR_UID | varchar(32) | NO   | PRI | 0       |       |
+---------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

Fig. 17 The Schema of “group_user” Table

```
mysql> desc task_user;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| TAS_UID     | varchar(32) | NO   | PRI |         |       |
| USR_UID     | varchar(32) | NO   | PRI |         |       |
| TU_TYPE     | int(11)     | NO   | PRI | 1       |       |
| TU_RELATION | int(11)     | NO   | PRI | 0       |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

Fig. 18 The Schema of “task_user” Table

5 GUI

As we want users in different departments use our system through a unified interface, we develop GUI for central system, and users can use it to route to their own ProcessMaker by same gateway. I will introduce all these pages.

5.1 Login Page

User use this page to login to the central system

Fig. 19 Login Page

5.2 Main Dashboard

User can use this page to choose the function, see the process model in their department, configure the encryption key value and so on.

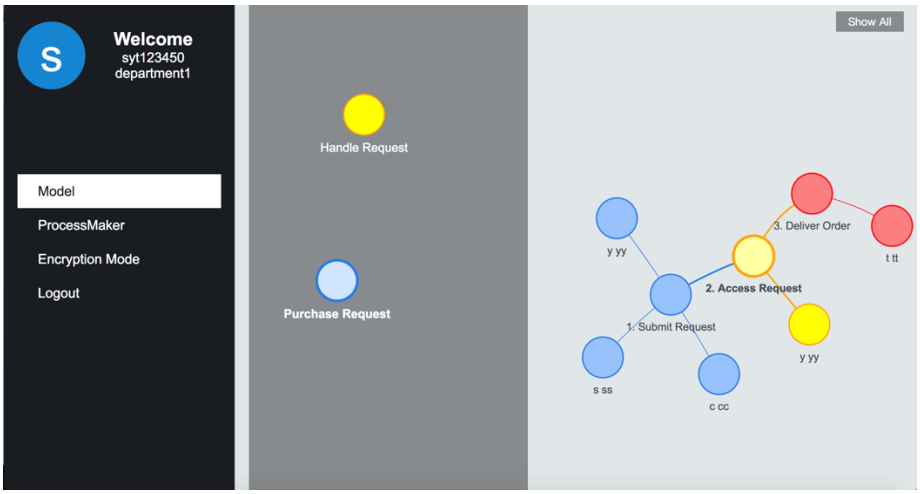

Fig. 20 Main Dashboard

5.3 ProcessMaker Login Page

Through central system, use can route to ProcessMaker's login page

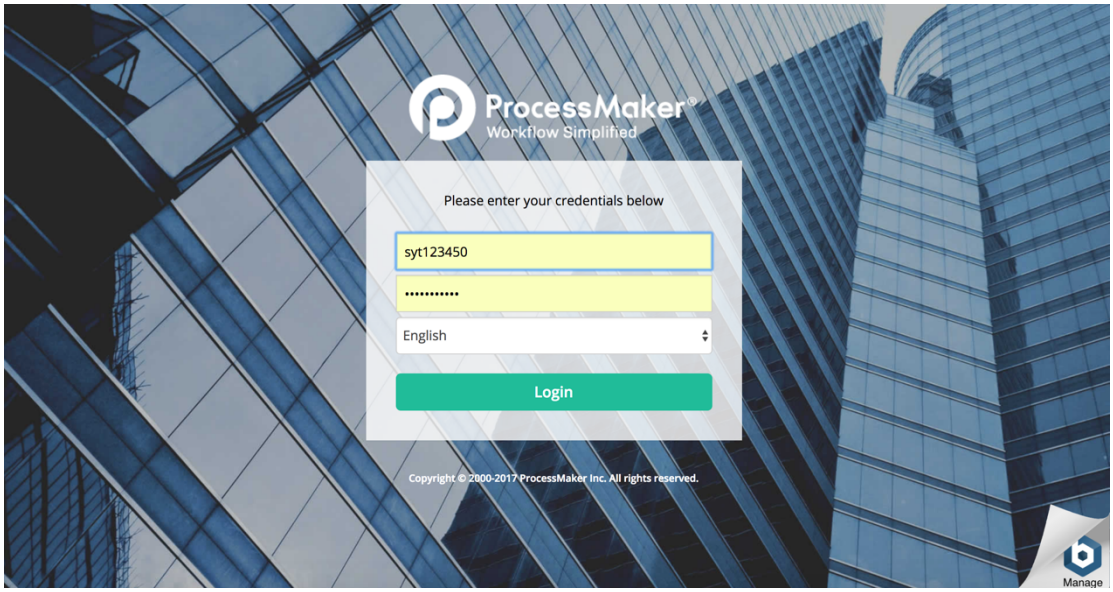

Fig. 21 ProcessMaker Login Page

5.4 ProcessMaker Dashboard

The user can define process or handle process in ProcessMaker, these function are supported by ProcessMaker.

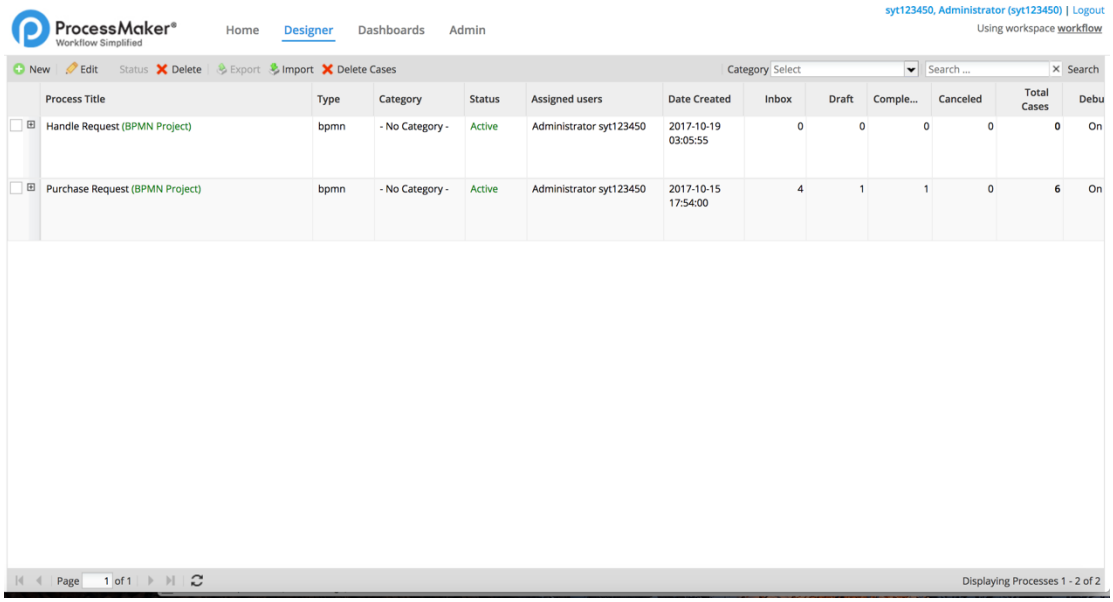

Fig. 22 ProcessMaker Dashboard

5.5 Create Instance Page

Create a new instance for new department in this page

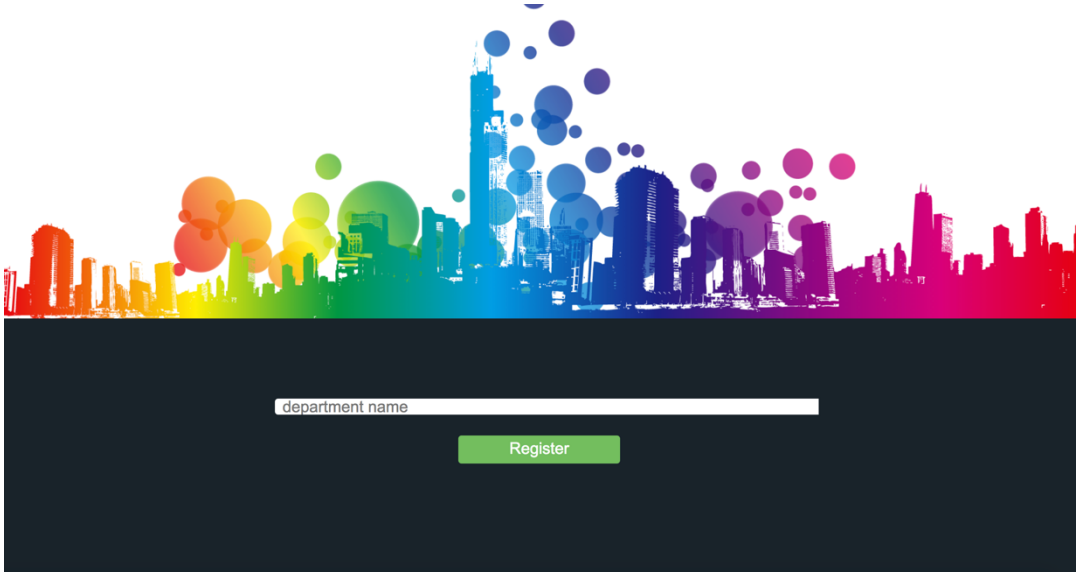

Fig. 23 Create Instance Page

6 Demo-Based Use Cases

6.1 login/logout

user can visit the login page we provide to login to the central system

Fig. 24 Login Page

Users need to choose the department they belong to

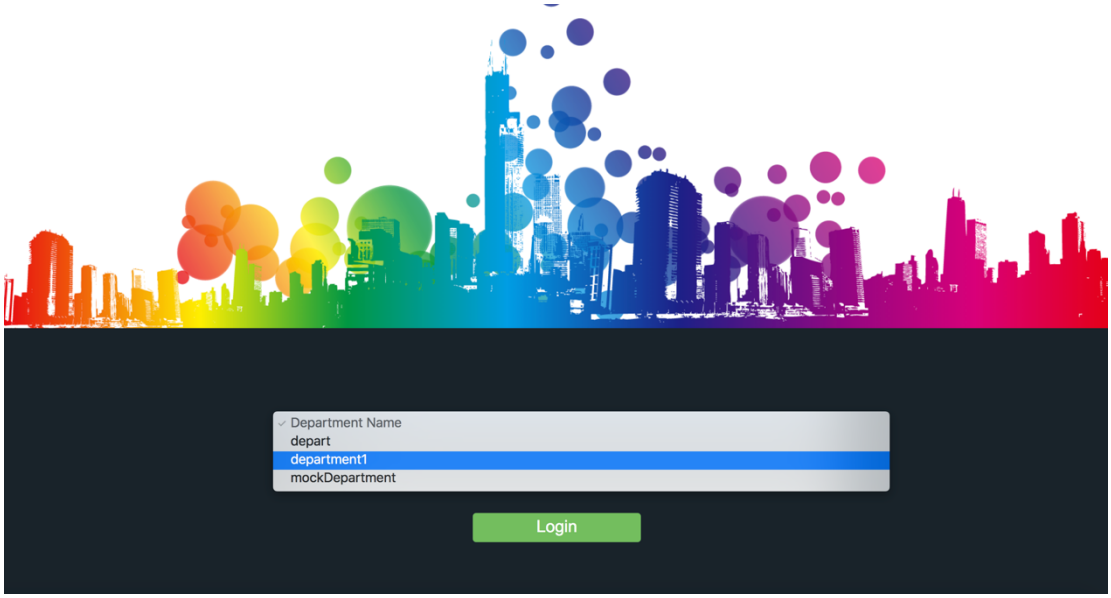

Fig. 25 Choose Department

And then input their user name and password

Fig. 26 Field All Field

If any field is invalid, there will be a hint for invalid input

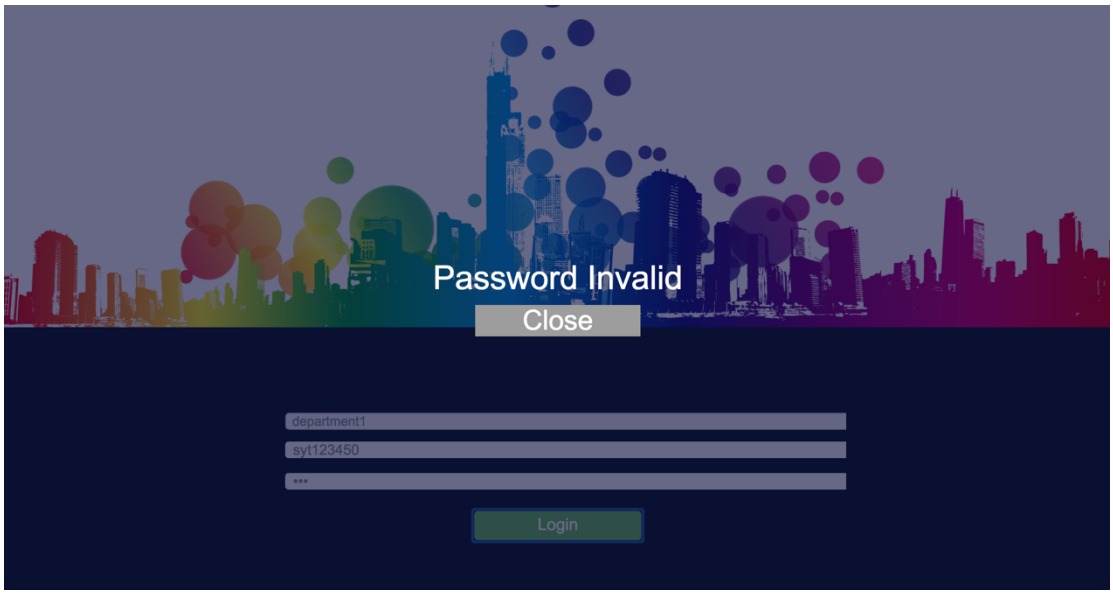

Fig. 27 Invalid Input

If all information is right, users will login to the dashboard

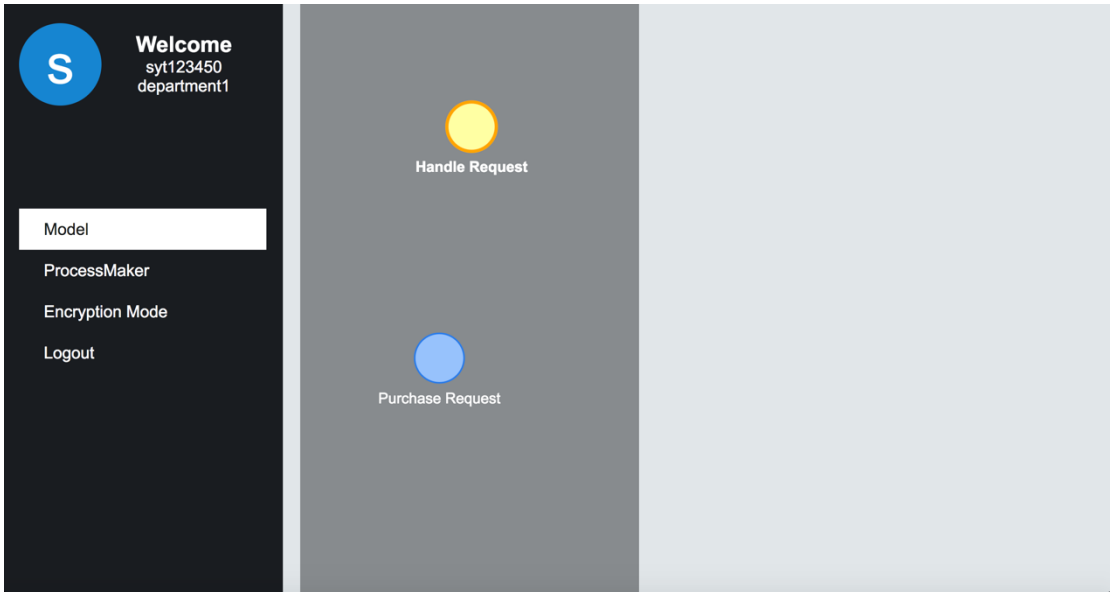

Fig. 28 Success Login

user can click the logout button in the dashboard to logout the system, if users close the dashboard without logout the system, next time the user visit the system, they will automatically login.

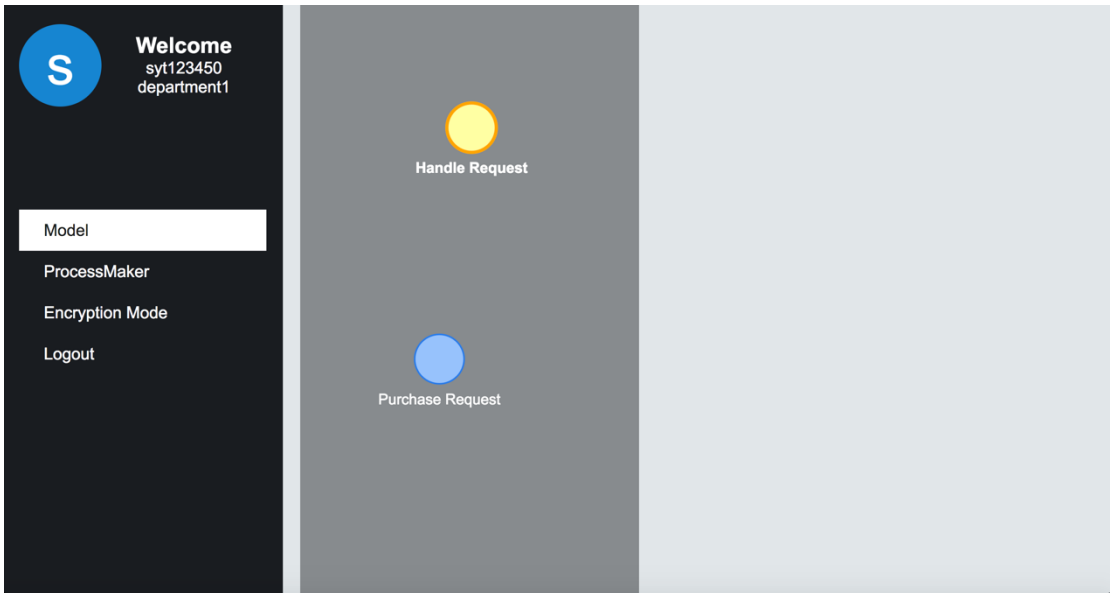

Fig. 29 Logout Button

6.2 Route

The user clicks the "ProcessMaker" button in the left menu bar, can route to the ProcessMaker login page.

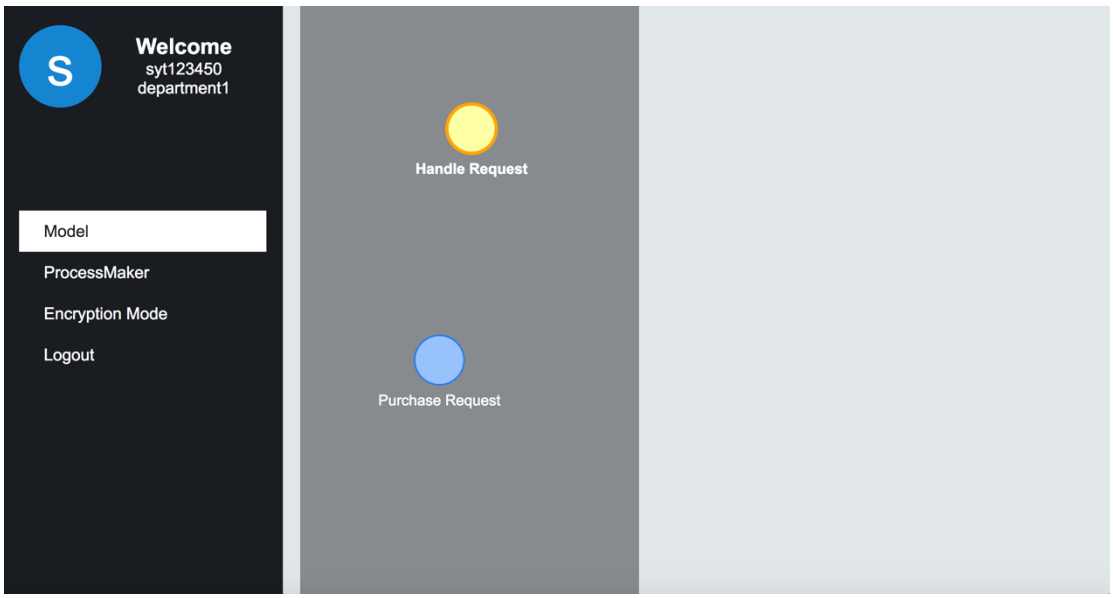

Fig. 30 Route Button

Users can input their user name and password of the ProcessMaker to login to ProcessMaker and use the function provided by ProcessMaker

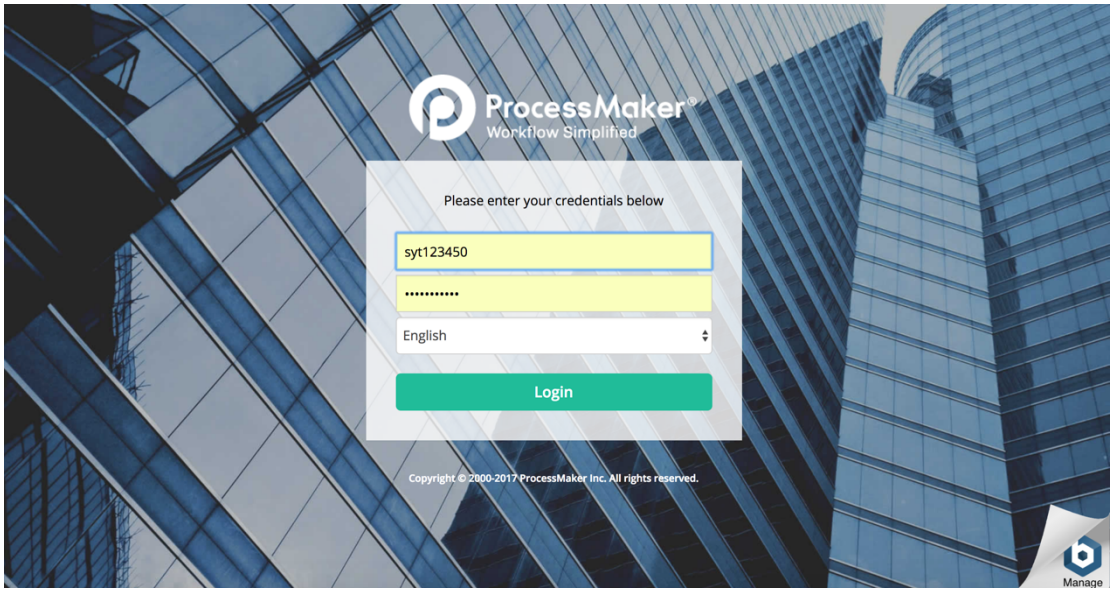

Fig. 31 ProcessMaker Login Page

6.3 Setting Encryption

The users can click the "Encryption Mode" button to slide down encryption field to configure their own encryption.

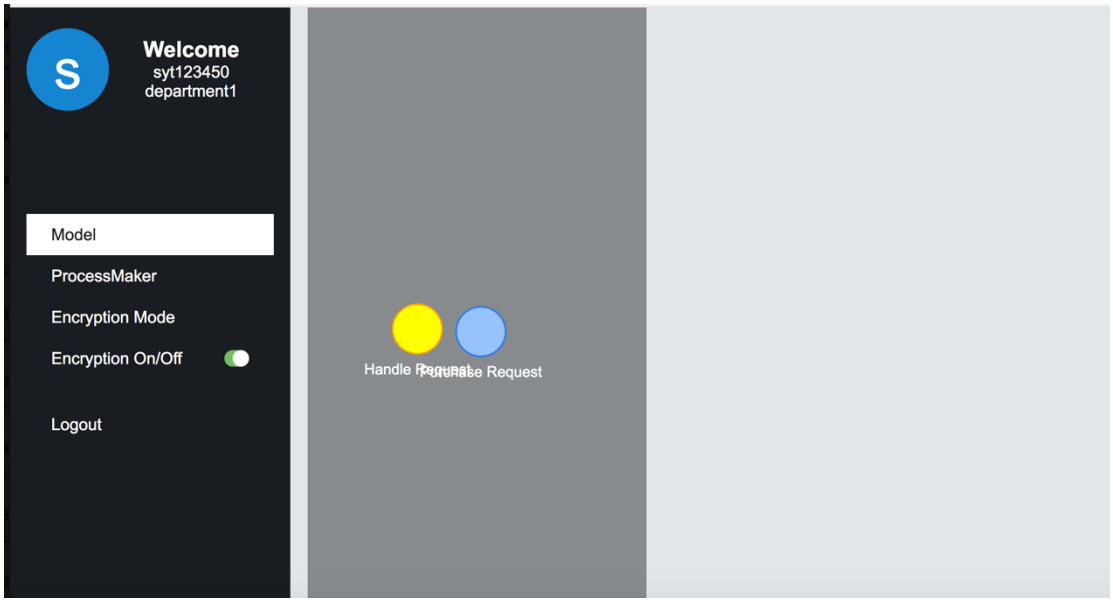

Fig. 32 Encryption Configuration Area

The user can open or close encryption function, when user click the checkbox in the encryption, the operation result message will show to users.

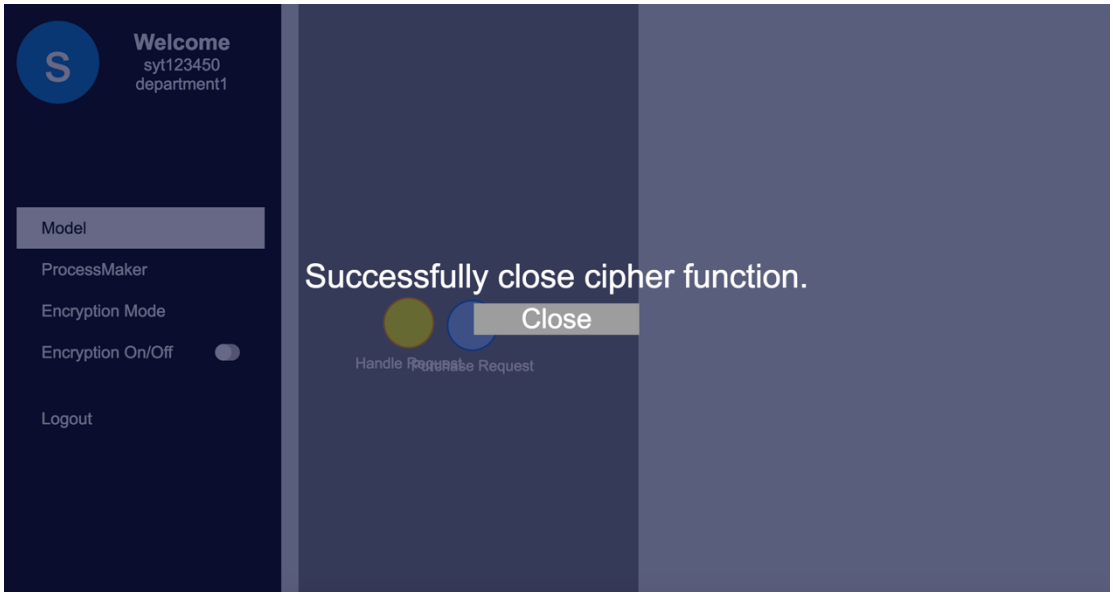

Fig. 33 Success Operation

If the user wants to open the encryption function, users can input their 8 bits encryption key

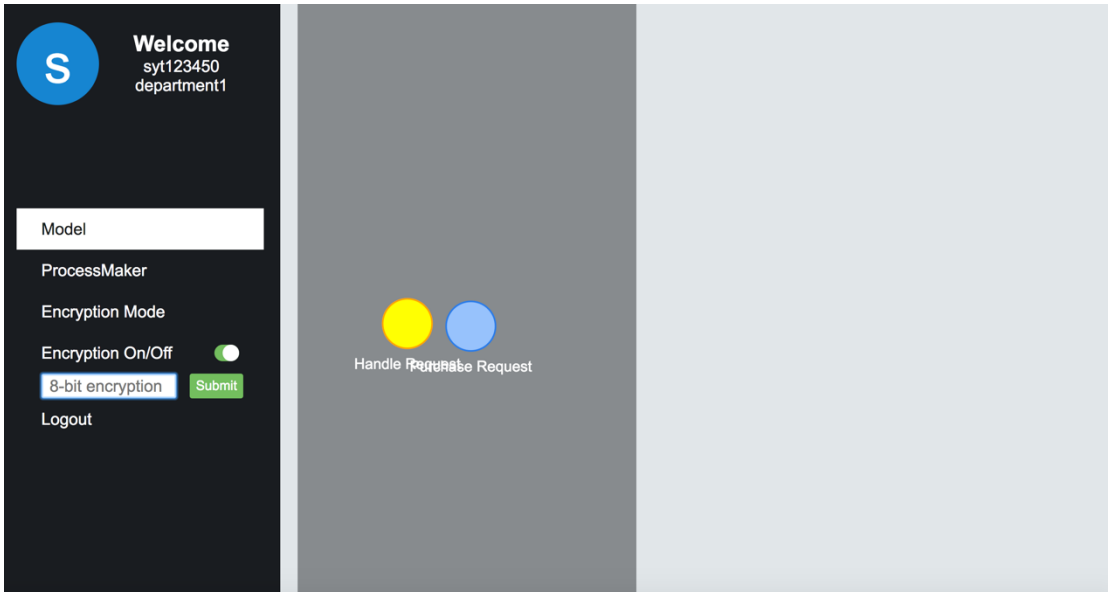

Fig. 34 Update Input Area

If the user input the invalid key value (for example 123), and click the submit button, there will be a hint for user to input the right key value

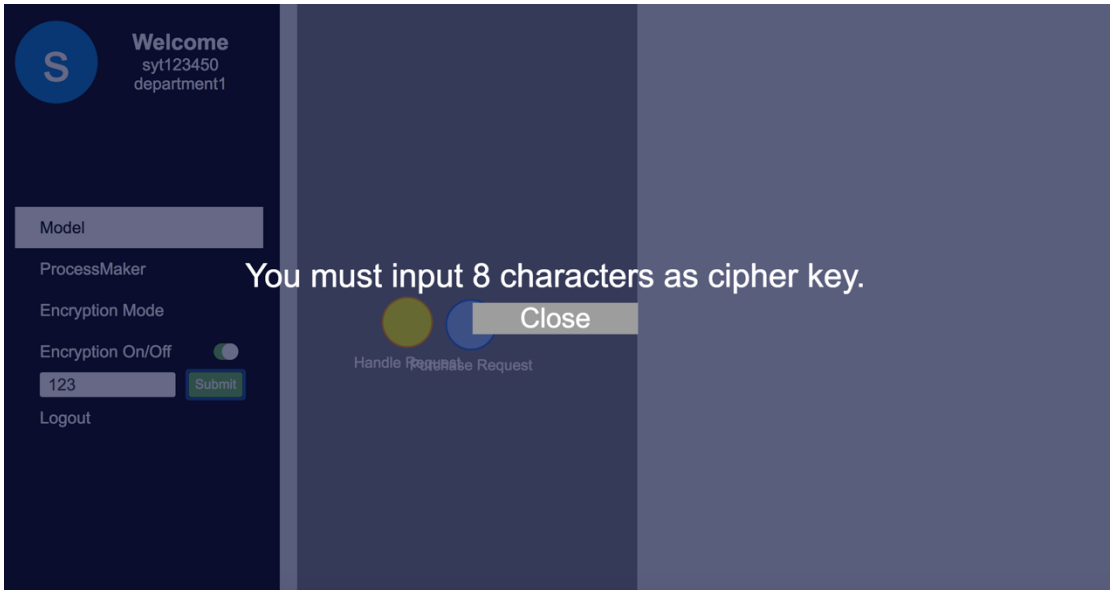

Fig. 35 Invalid Key Value Input

If the user input the right key value and click the submit button, there will be a successful message, and the input area will slide up.

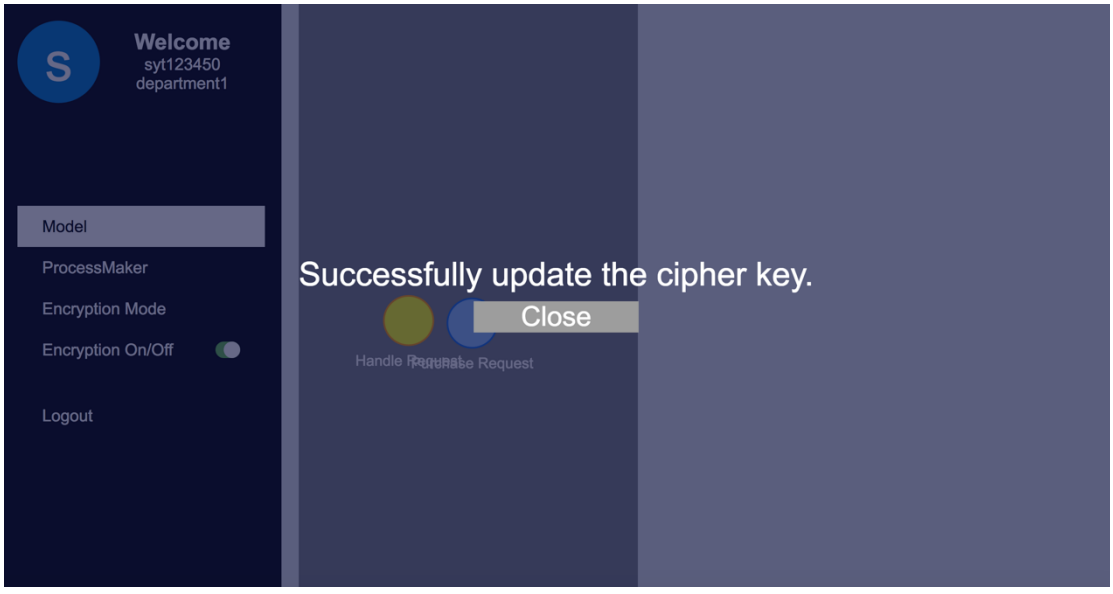

Fig. 36 Success Update

6.4 Checking Model

When user login to the system, all process belong to their department will show in the middle area, each circle represent a specific process designed in ProcessMaker belong to their department.

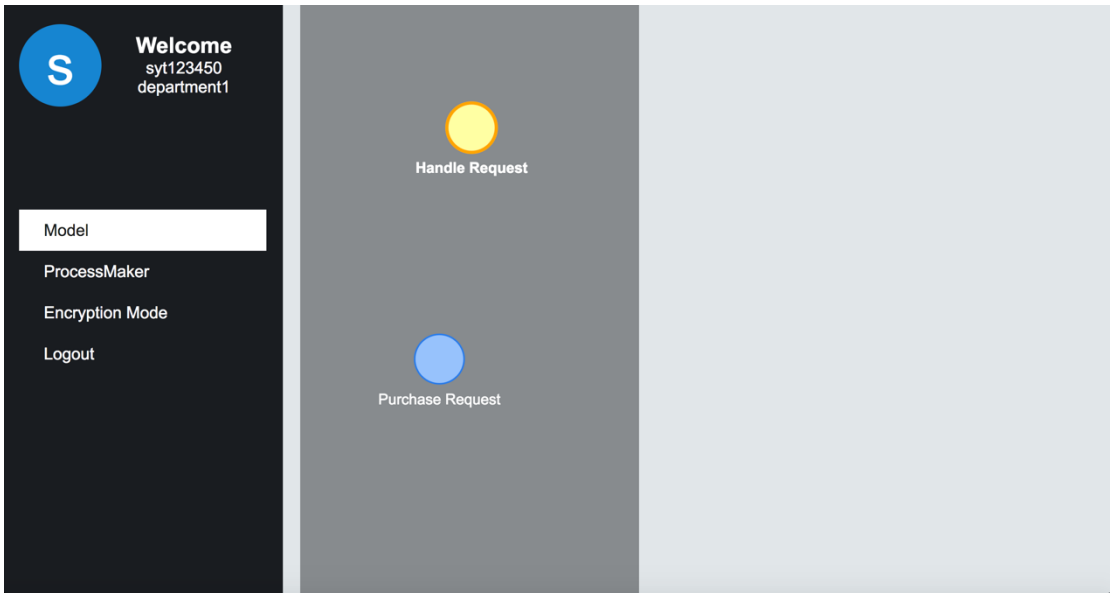

Fig. 37 All Processes

click a specific Process the structure of it will be shown in the right part, each circle represent a task in clicked process, and all circle is sorted and marked.

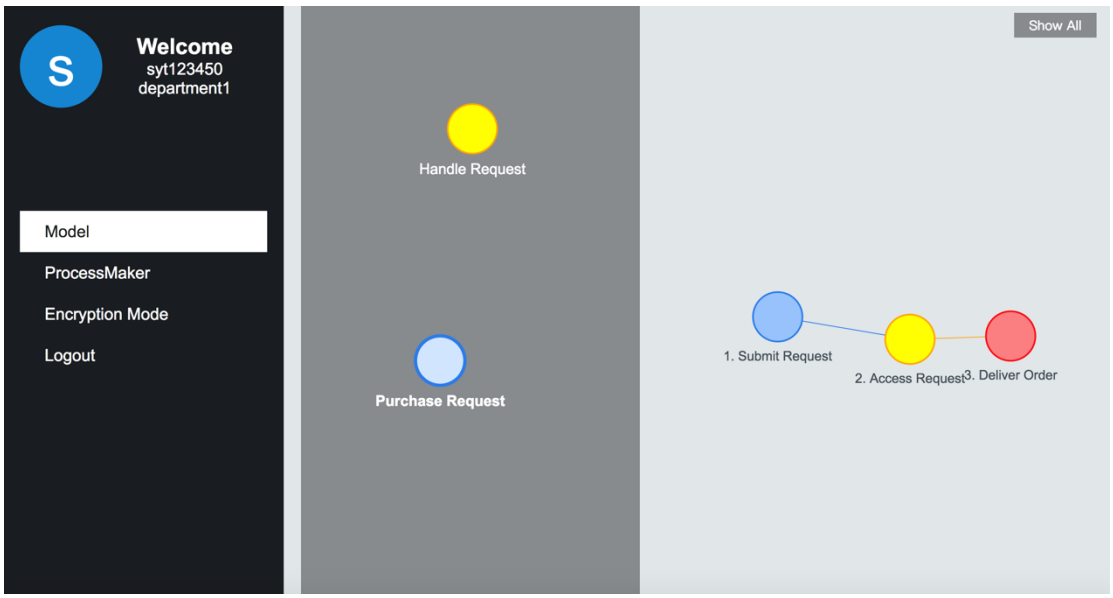

Fig. 38 Tasks for “Purchase Request” Process

click different task in the right area can check all people in charge of this task. (the diagram can be dragged)

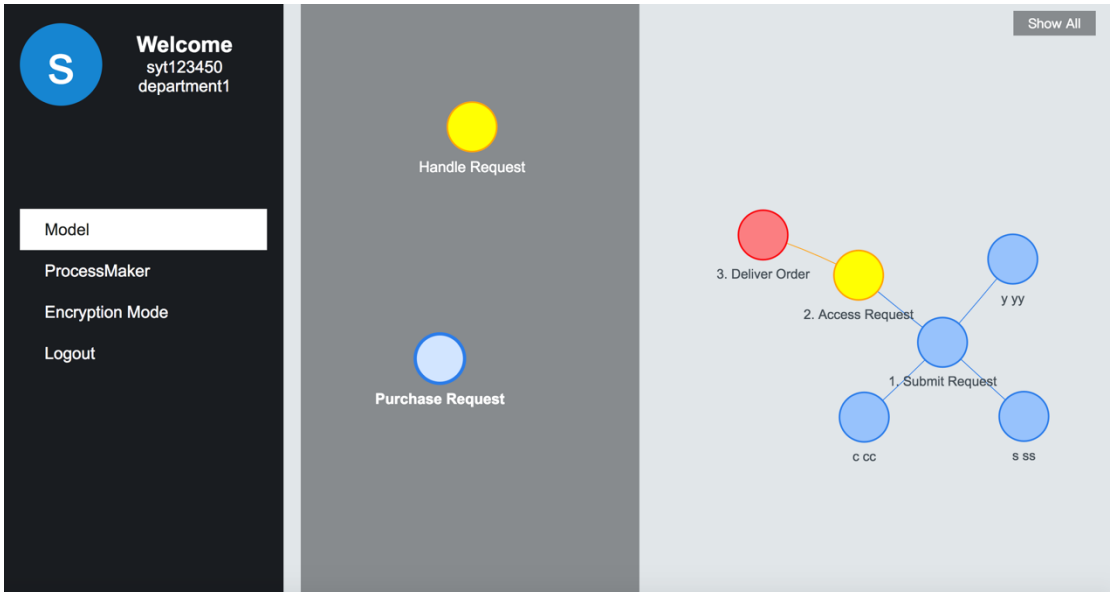

Fig. 39 Handlers for “Submit Request”

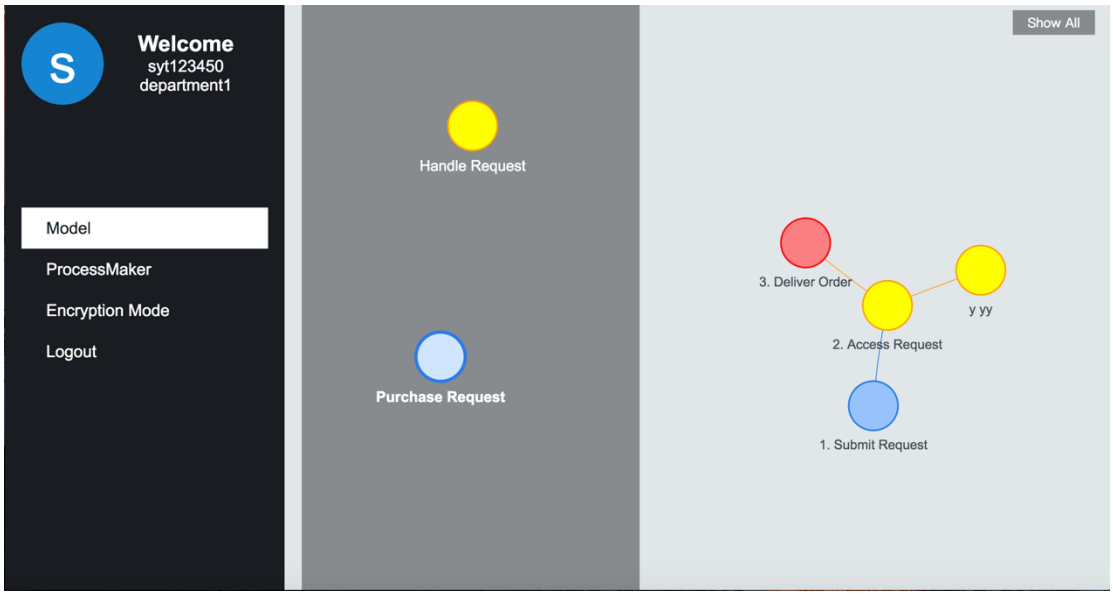

Fig. 40 Handlers for “Access Request”

click "show all" button will show all people in charge of all tasks

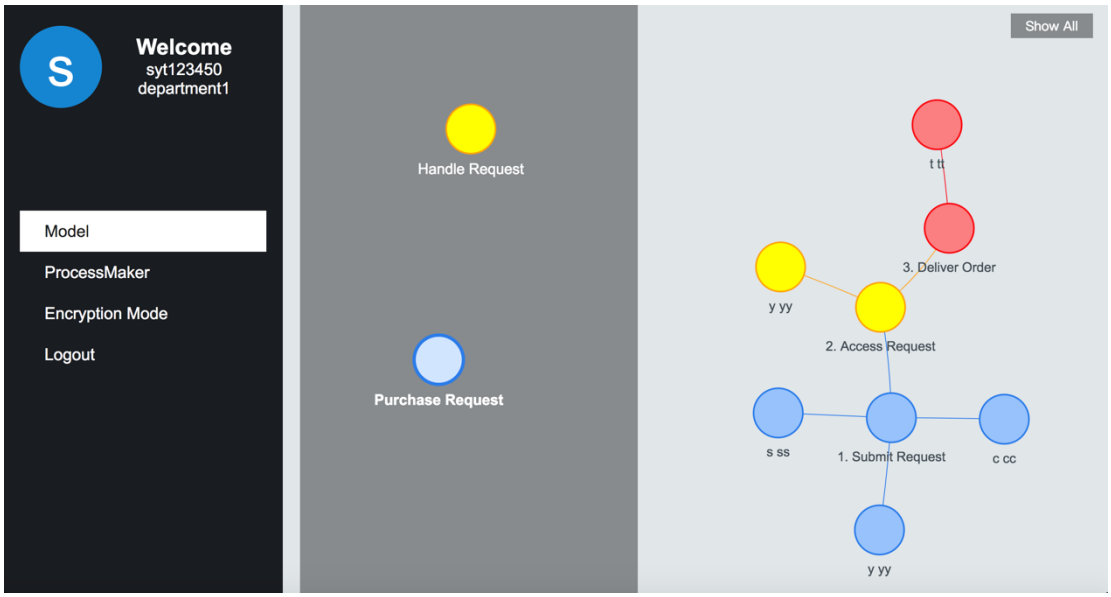

Fig. 41 Handlers for All Tasks

6.5 Instance Creation

There is an instance creation page, user can input the department name to create a new department

Fig. 42 Instance Creation Page

if successfully create a new department (for example "hello department"), it will show a success message

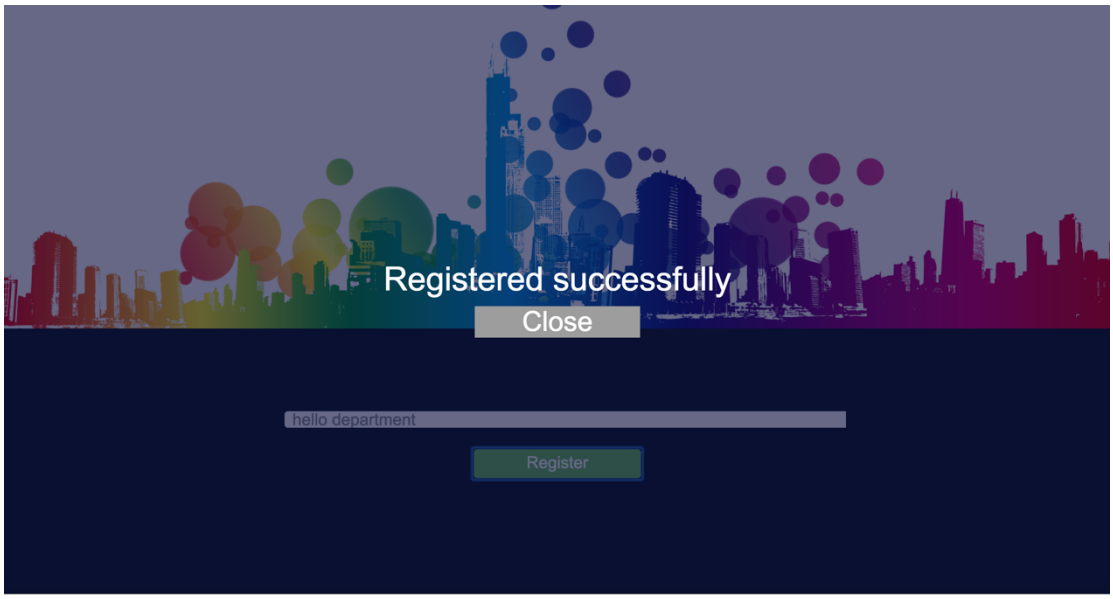

Fig. 43 Success Creation

The created department ("hello department") will be shown in the login page's department select area

Fig. 44 Exist in Select Input

7 Source Code

7.1 Source Code File

The source code is in the SourceCode.zip

7.2 Source Code Link

The source code can also be seen in the github link below:
https://github.com/syt123450/GOV