


Beijing-Dublin International College

Databases and Information Systems

Database Design Report — Smart Home System

Name: Zihao Wang **UCD Student No: 16206563**
Name: Jingxiang Sun **UCD Student No: 16206869**
Name: Yuchen Wang **UCD Student No: 16206561**

We have read and fully understand the consequences of plagiarism as discussed in the documents on Moodle. I also fully understand the definition of plagiarism.

Date: 2019.05.28

Abstract

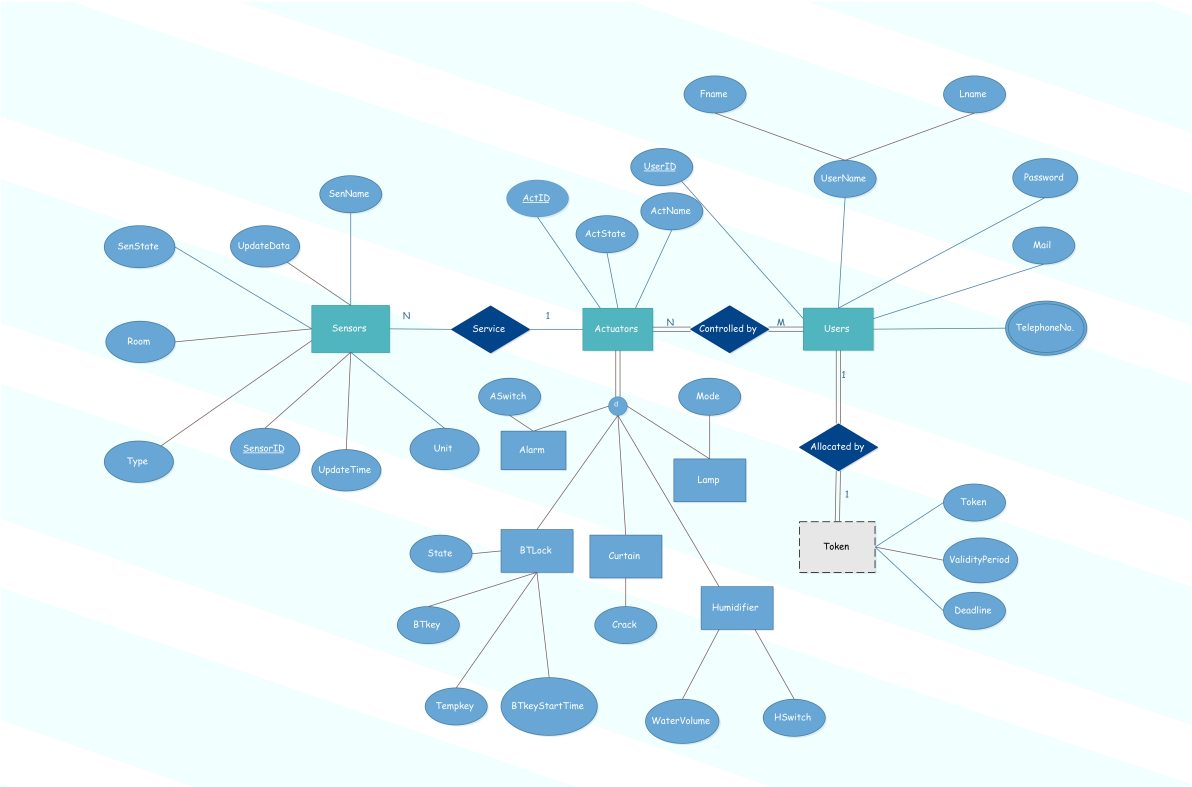
Homes have undergone rapid transformations since the Industrial revolution. Over the last hundred and fifty years, various technologies from ovens to ranges to refrigerators, lawn mowers to sprinklers along with televisions and automobiles have become an integral part of home. **Each wave of technology has remade what the home is, what it means to live in a home and what each member of the family does in it.**

These smart home devices belong to a larger phenomenon called the Internet of Things (IoT). As BDIC students who are major in IoT, we are learning how to capture data from the physical world and transmit it virtually and similarly gather instructions virtually and perform physical actions, and we have ability to integrate the knowledge acquired from subjects like COMP2004J and others applying to practical issues.

In this project, we combine the properties of Database Information Systems and physical devices in to our Intelligent Home Control System design, where one can interact with terminals in house. To be more precise, we deploy the project on personal server (VPS), so at any time each of us can access the system. We believe this can show exactly what we learned in class and after class. The host ip of this project is: **http://39.108.231.244:8080/DBLab/**. You can login through our public Username: “sun” and password: “123456”. Be free to raise questions!

Content

Section 1. ER Diagram



Assumptions:

- Both users and sensors can control actuators.
- One user can control many actuators, but one sensor can control only one actuator.
- The id is unique, but the name is not.
- The actuator and sensor have only two state: on and off.
- The initial "UpdateTime" of a sensor is the time it is added to the database.
- An actuator having a id smaller than 2000 is an alarm, between 2000 and 3000 is a bluetooth lock, between 3000 and 4000 is a curtain, between 4000 and 5000 is a humidifier, beyond 5000 is a lamp.

Section 2. Mapping

The whole process converting ER diagram into a logical model is divided into 8 detached steps:

1 Mapping Regular Entity Types
2 Mapping Weak Entity Types
3 Mapping 1:1 Relationships
4 Mapping 1:N Relationships
5 Mapping M:N Relationships
6 Mapping Multivalued Attributes
7 Mapping N-ary Relationships (skipped)
8 Mapping supertypes/subtypes

1. Mapping Regular Entity Types:

Select all strong entities: **Actuators**, **Users** and **Sensors**.
Consider just simple relations and map to the following models:

Actuators	<u>ActID</u>	ActState	ActName		
Users	<u>UserID</u>	Fname	Lname	Password	Mails
Sensors	<u>SenID</u>	SenName	SenState	UpdateTime	UpdateData

2. Mapping Weak Entity Types:

Select all weak entities: **Token**

There are several rules here:

a. Include all simple attributes.

b. The primary key is a combination of the foreign key and weak entity's partial key.

Actuators	ActID	ActState	ActName		
Users	UserID	Fname	Lname	Password	Mails
Sensors	SenID	SenName	SenState	UpdateData	UpdateTime
Token	Token	*UserID*	Deadline	ValidityPeriod	

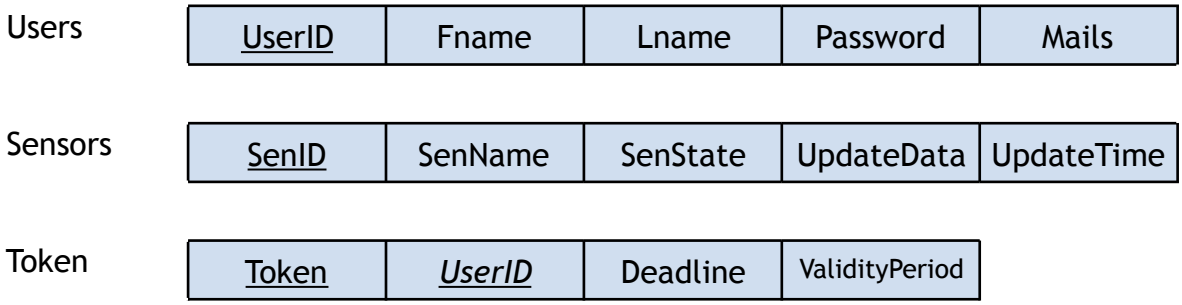
3. Mapping 1:1 Relationships:

Select all 1:1 relationships in ER diagram:

One unique token is allocated to every user.

To map a 1:1 relationship, we use Foreign Key Approach. However, in this case, as we have already added the primary key of Users as the foreign key of Token, the tables keep maintained:

Actuators	ActID	ActState	ActName

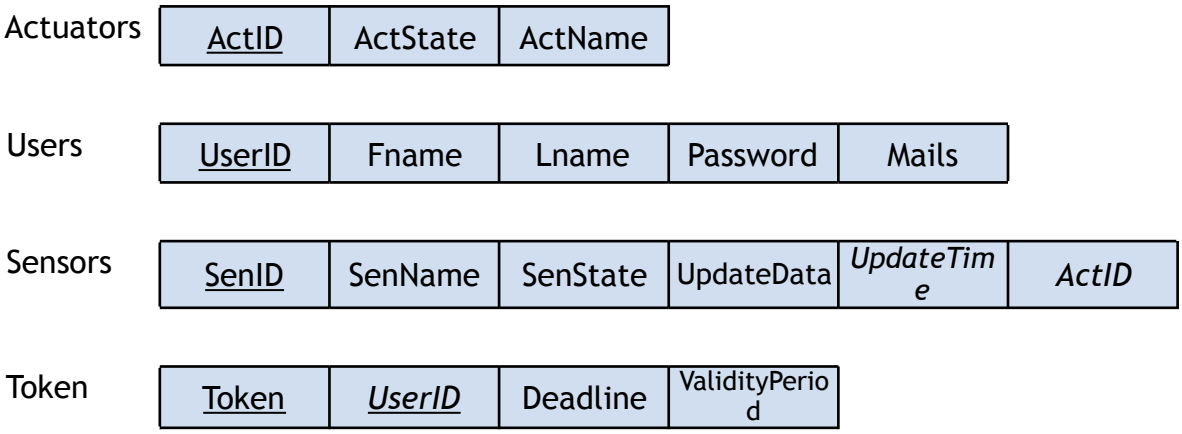


4. Mapping 1:N Relationships:

Select all the 1:N relationships:

N sensors can simultaneously serve to one actuator.

In the entity type on the N side, include a foreign key that refers to the primary key of the attribute on the 1 side, and the resulting model is depicted as below:

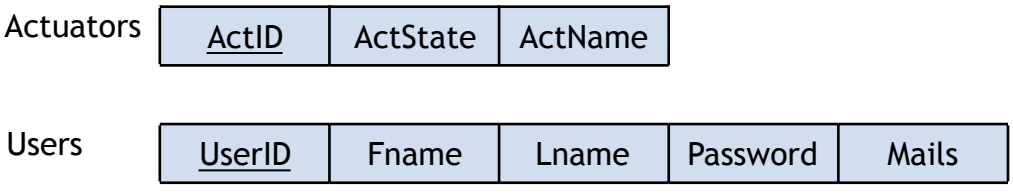


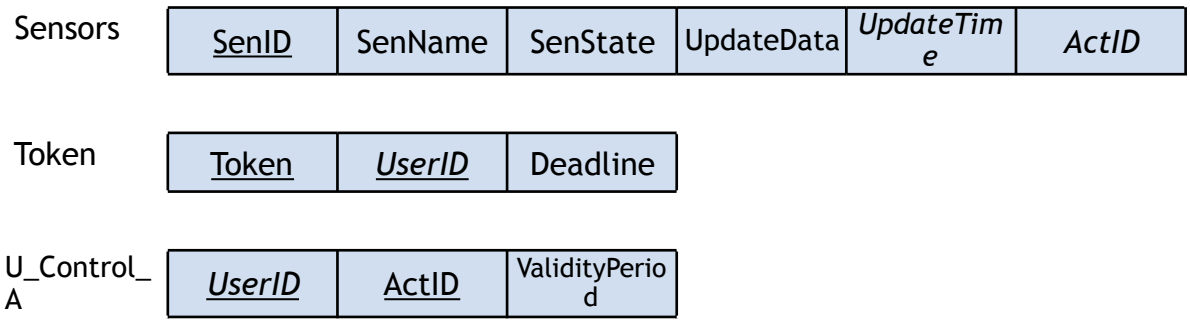
5. Mapping N:M Relationships:

Select all the N:M relationships:

N actuator s are controlled by M users simultaneously.

Create a new relation that represents the relationship. Include foreign keys to refer to both of the entity types involved. What's more, the primary key of this table will be the combination of both of these foreign keys. So the resulting model is depicted as below:

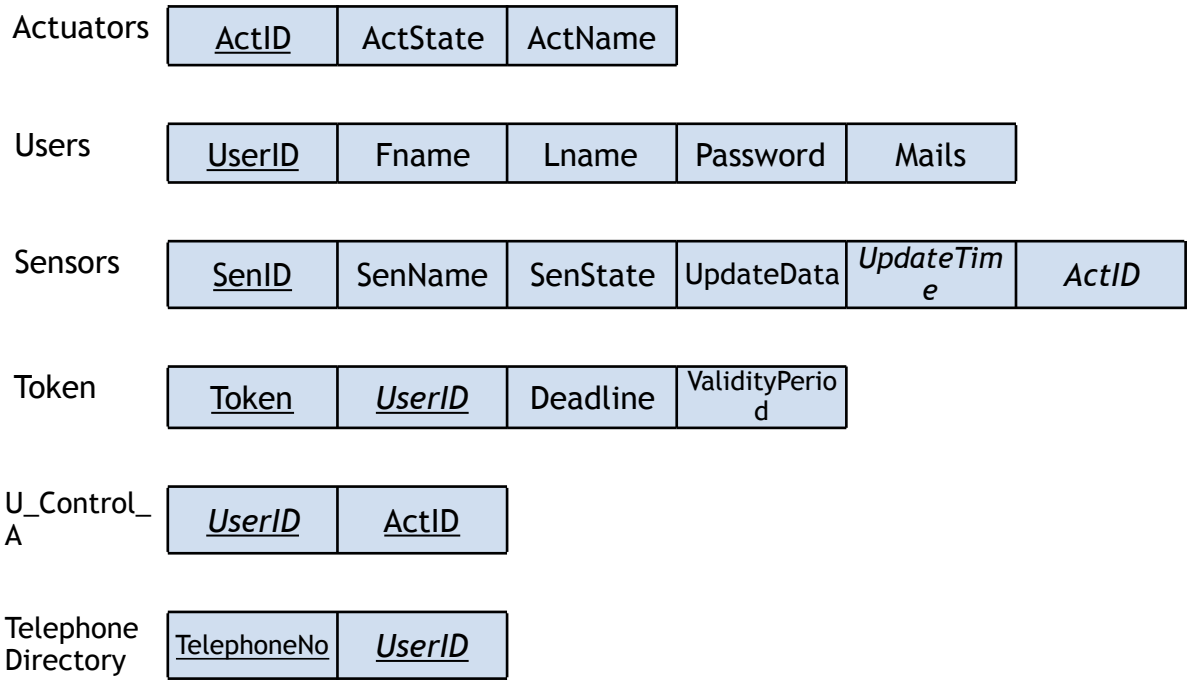




6. Mapping Multivalued Attributes:

Select all the multivalued attributes: TelephoneNo from **Actuators**

For each multi-valued attribute, create a new relation to represent it, which contains an attribute to store the multi-valued attribute itself and a foreign key that refers to the entity type that the attribute belongs to.



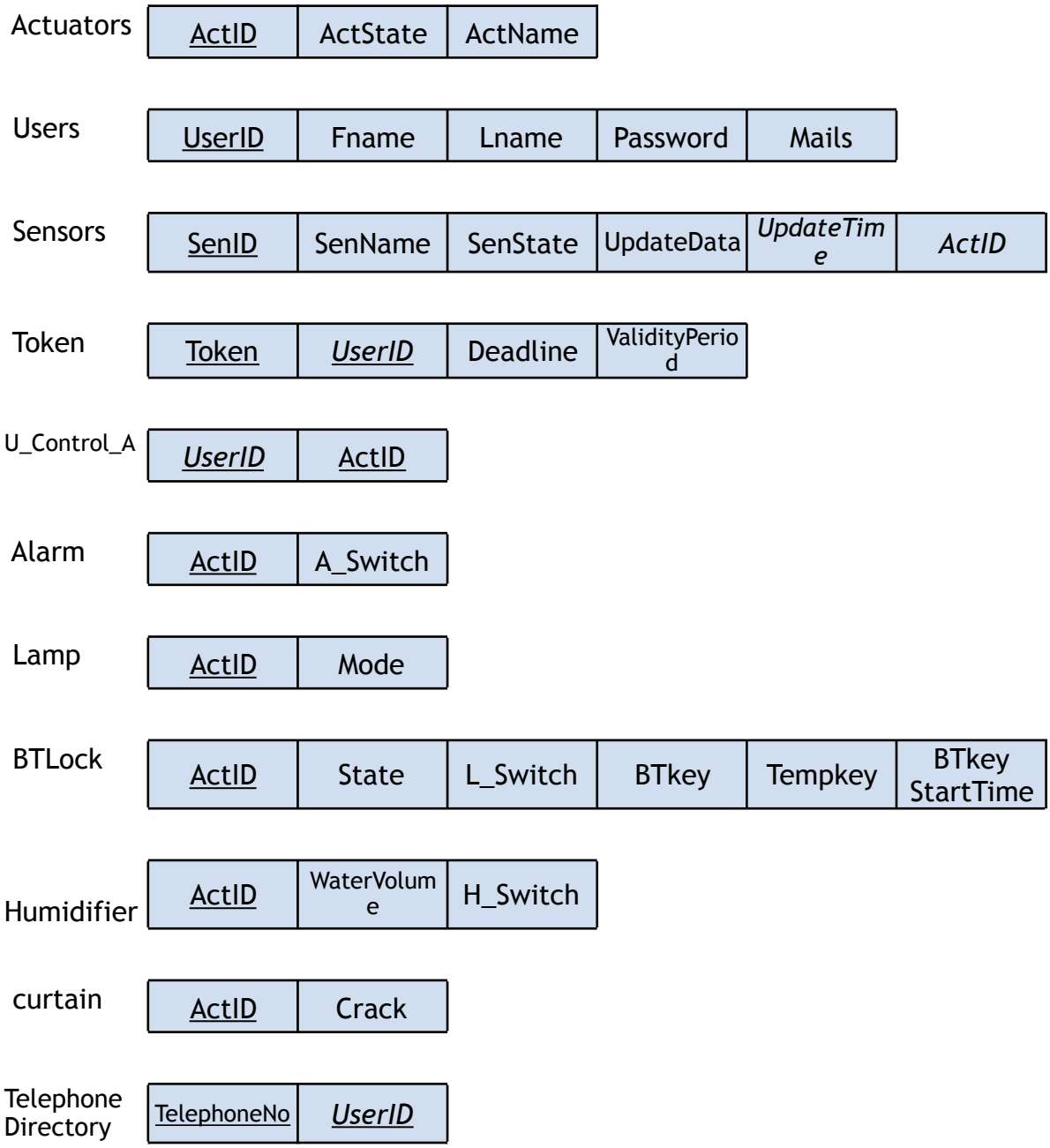
8. Mapping Supertypes/Subtypes:

Select all supertypes and subtypes:

Supertype: **Actuators**

Subtypes: **Alarm Lamp BTLock Curtain Humidifier**

For each subtype, create a relation and include a foreign key that refers to the supertype's primary key which is the primary key of the subtype.



Section 3. Normalization

To balance the data integrity and the data redundancy, we adapt our database to Boyce-Codd Normal Form.

1. First Normal Form

The database consists of ten relations, none of them contain repeating attributes or groups of attributes.
It means, these relations are in 1NF

2. Second Normal Form

There are two relations having compound primary key: Token and U_Control_A.
Token is a weak entity, the two non-key attributes depend on the whole key.
U_Control_A doesn't contain any non-key attribute.
Thus, our relations are in 2NF.

3. Third Normal Form

Now we check the transitive functional dependencies.
There are six relations having more than one non-key attribute: Actuators, Users, Sensors, Token, BTLock and Humidifier.
The non-key attributes in these relations are all independent.
Therefore, our database satisfies 3NF.

3.5 Boyce-Codd Normal Form

In our relations, all dependencies are fully functionally dependent on the primary keys, which means every determinant is a candidate key.
Hence, BCNF is verified.

Section 4. Create DataBase & Structure of Tables

We create our database following the above definition of tables.

1. Actuators

- Actuator's ID is set to be the primary key in table 'Actuators', it is set to be an "INT" value as common.
- Actuator's State is set to be "SMALLINT" to save space.
- Actuator's Name is set to be "NOT NULL" which means it must have a name. The attribute is recorded with the type of "VARCHAR" and length is 20.

```
CREATE TABLE Actuators (
ActID INT PRIMARY KEY,
ActState SMALLINT,
ActName VARCHAR(20) NOT NULL
);
```

```
mysql> describe Actuators;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| ActID    | int(11)     | NO   | PRI | NULL    |       |
| ActState | smallint(6) | YES  |     | NULL    |       |
| ActName  | varchar(20) | NO   |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

2. Users

- User's ID is set to be PRIMARY KEY and is defined as "INT" .
- It has to be noticed that in a family, the first name is unique. That is the reason we set FName as an "VARCHAR" and to be UNIQUE.
- Apart from that, we set Password as required and Mail as an optional. Both of them are set to be "VARCHAR" type.

```
CREATE TABLE Users (
UserID INT PRIMARY KEY,
FName VARCHAR(20) UNIQUE NOT NULL,
LName VARCHAR(20) NOT NULL,
Password VARCHAR(30) NOT NULL,
Mail VARCHAR(30)
);
```

```
mysql> describe Users;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| UserID   | int(11)     | NO   | PRI | NULL    |       |
| FName    | varchar(20) | NO   | UNI | NULL    |       |
| LName    | varchar(20) | NO   |     | NULL    |       |
| Password | varchar(30) | NO   |     | NULL    |       |
| Mail     | varchar(30) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

3. Token

- This table stores the keys for the user. For each user, he/she should maintain an token for him to unlock the devices. Thus in this table “Token” attribute is set to be “VARCHAR” and our primary key consist of Token and UserID.
- For each token, it should be changed after a period of time. This can improve the security of user’s account. To go a step further, we can use TOTP-HMAC-MD5 to encrypt token.
- As it shown in the ER diagram and step 2 of mapping, Token is a weak entity of Users, such that UserID should be set as FOREIGN KEY.

```
CREATE TABLE Token (
Token VARCHAR(30),
DeadLine DATETIME,
UserID INT,
PRIMARY KEY(Token, UserID),
FOREIGN KEY (UserID) REFERENCES Users(UserID) ON UPDATE CASCADE
ON DELETE CASCADE
);
```

```
mysql> describe Token;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| Token    | varchar(30) | NO   | PRI | NULL    |       |
| DeadLine | datetime    | YES  |     | NULL    |       |
| UserID   | int(11)     | NO   | PRI | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

4. Sensors

- The information of a sensor should be recorded completely. Thus, SenName should be constraint as “NOT NULL” and SensorID should be set as PRIMARY KEY.
- Another noticeable thing is we should also record sensor’s update time and data respectively. Thus we use DATETIME to record last update time and VARCHAR to record last update data.
- The type of sensors should be one of “TEMPERATURE SENSORS”, “ILLUMINATION SENSORS” and “HUMIDITY SENSORS”. Therefore, we set Type as ENUM.
- As it shown in ER diagram, Sensors service for Actuators and it is an N-to-1 relationship, it means that an actuator are possible be serviced by multiple sensors. By the 4th step of mapping, ActID is set to be FOREIGN KEY and set to be Cascade when update and delete.

```
CREATE TABLE Sensors (
SensorID INT PRIMARY KEY,
SenName VARCHAR(20) NOT NULL,
SenState SMALLINT,
UpdateTime DATETIME,
UpdateData VARCHAR(20),
ActID INT,
Unit VARCHAR(10),
Type ENUM('TEMPERATURE SENSORS','ILLUMINATION
SENSORS','HUMIDITY SENSORS') NOT NULL,
FOREIGN KEY (ActID) REFERENCES Actuators(ActID) ON UPDATE
CASCADE ON DELETE SET NULL
);
```

```
mysql> describe Sensors;
+------------+-----------------------------------------------------------------------+------+-----+---------+-------+
| Field      | Type                                                                  | Null | Key | Default | Extra |
+------------+-----------------------------------------------------------------------+------+-----+---------+-------+
| SensorID   | int(11)                                                               | NO   | PRI | NULL    |       |
| SenName    | varchar(20)                                                           | NO   |     | NULL    |       |
| SenState   | smallint(6)                                                           | YES  |     | NULL    |       |
| UpdateTime | datetime                                                              | YES  |     | NULL    |       |
| UpdateData | varchar(20)                                                           | YES  |     | NULL    |       |
| ActID      | int(11)                                                               | YES  | MUL | NULL    |       |
| Unit       | varchar(10)                                                           | YES  |     | NULL    |       |
| Type       | enum('TEMPERATURE SENSORS','ILLUMINATION SENSORS','HUMIDITY SENSORS') | NO   |     | NULL    |       |
| Room       | enum('Bedroom','Living Room','Kitchen')                               | YES  |     | NULL    |       |
+------------+-----------------------------------------------------------------------+------+-----+---------+-------+
9 rows in set (0.00 sec)
```

5. U_Control_A

- This is a table produced in mapping processing. It can be observed from ER diagram that the figure for relationship between Sensors and Actuators shows that they are M-to-N relationships.
- By the mapping rules, we create a new table to represent this relationship. ActID and UserID are together set to be the primary key.

```
CREATE TABLE U_Control_A (
ActID INT,
UserID INT,
LastManageTime DATETIME,
PRIMARY KEY (ActID, UserID),
);
```

```
mysql> describe U_Control_A;
+----------------+----------+------+-----+---------+-------+
| Field          | Type     | Null | Key | Default | Extra |
+----------------+----------+------+-----+---------+-------+
| ActID          | int(11)  | NO   | PRI | NULL    |       |
| UserID         | int(11)  | NO   | PRI | NULL    |       |
| LastManageTime | datetime | YES  |     | NULL    |       |
+----------------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

6. TelephoneDirectory

- Since Telephone is a multi-value attribute in table named “Users”, according to the sixth mapping rule, we create a separate table to record the Telephone Book for users.
- The table contains the multi-value attribute and the primary key of its original table. These two attribute combined as PRIMARY KEY of the table. Apart from that, UserID should also be a FOREIGN KEY of Telephone Directory.

```
CREATE TABLE TelephoneDirectory (
TelephoneNo VARCHAR(30),
UserID INT,
PRIMARY KEY (TelephoneNo, UserID),
FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE
);
```

```
mysql> describe TelephoneDirectory;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| TelephoneNo | varchar(30) | NO   | PRI | NULL    |       |
| UserID      | int(11)     | NO   | PRI | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

7. subtype table of Actuators

7.1 BTLock

- Bluetooth Lock is a subtype of Actuators. Comparing with Actuator, BTLock has some features cannot be included in “common actuators”. For instance, it maintain an attribute named “BTKey” which is a foreign key of “Token” in the table Token.
- Apart from that, due to this table is also a subtype of Actuator, we separate it from its supertype.
- The cascade of ActID should be set to ON DELETE CASCADE. The reason is since we delete Actuator from supertype, the corresponding subtype should no longer exist.

```
CREATE TABLE BTLock (
ActID INT PRIMARY KEY,
State SMALLINT,
BTKey VARCHAR(30),
TempKey VARCHAR(30),
TempKeyStartTime DATETIME,
FOREIGN KEY (BTKey) REFERENCES Token(Token) ON UPDATE CASCADE
ON DELETE CASCADE,
FOREIGN KEY (ActID) REFERENCES Actuators(ActID) ON DELETE
CASCADE
);
```

```
mysql> describe BTLock;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| ActID          | int(11)     | NO   | PRI | NULL    |       |
| State          | smallint(6) | YES  |     | NULL    |       |
| BTKey          | varchar(30) | YES  | MUL | NULL    |       |
| TempKey        | varchar(30) | YES  |     | NULL    |       |
| BTKeyStartTime | datetime    | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

7.2 Alarm

- Alarm is a also subtype of Actuators. Comparing with Actuator, it has a characteristic that cannot be represented by common “actuators”, that is the reason we build subtype to represent it.
- By the rules of mapping subtypes, we create this table to represent subtype relationships.

```
CREATE TABLE Alarm (
ActID INT PRIMARY KEY,
ASwitch SMALLINT,
FOREIGN KEY (ActID) REFERENCES Actuators(ActID) ON DELETE
CASCADE
);
```

```
mysql> describe Alarm;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| ActID   | int(11)     | NO   | PRI | NULL    |       |
| ASwitch | smallint(6) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

7.3 Lamp

- Similarly as BTLock and Alarm, we use Mode to represent the light strength of the lamp, which is defined as SMALLINT.
- Similarly as before, ActID in this table is set to be FOREIGN KEY and ON DELETE CASCADE.

```
CREATE TABLE Lamp (
ActID INT PRIMARY KEY,
Mode SMALLINT,
FOREIGN KEY (ActID) REFERENCES Actuators(ActID) ON DELETE
CASCADE
);
```

```
mysql> describe Lamp;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| ActID | int(11)     | NO   | PRI | NULL    |       |
| Mode  | smallint(6) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

7.4 Humidifier

- This table maintains water volume remained in the humidifier and switch level of humidifier. We use INT to represent the remaining water and we use SMALLINT to represent switch level.

```
CREATE TABLE Humidifier (
ActID INT PRIMARY KEY,
WaterVolume INT,
HSwitch SMALLINT,
FOREIGN KEY (ActID) REFERENCES Actuators(ActID) ON DELETE
CASCADE
);
```

```
mysql> describe Humidifier;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| ActID       | int(11)     | NO   | PRI | NULL    |       |
| WaterVolume | int(11)     | YES  |     | NULL    |       |
| HSwitch     | smallint(6) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

7.5 Curtain

- Curtain is simple table comparing with previous subtypes. It has an attribute named “Crack”, representing the degree of opening and drawing.

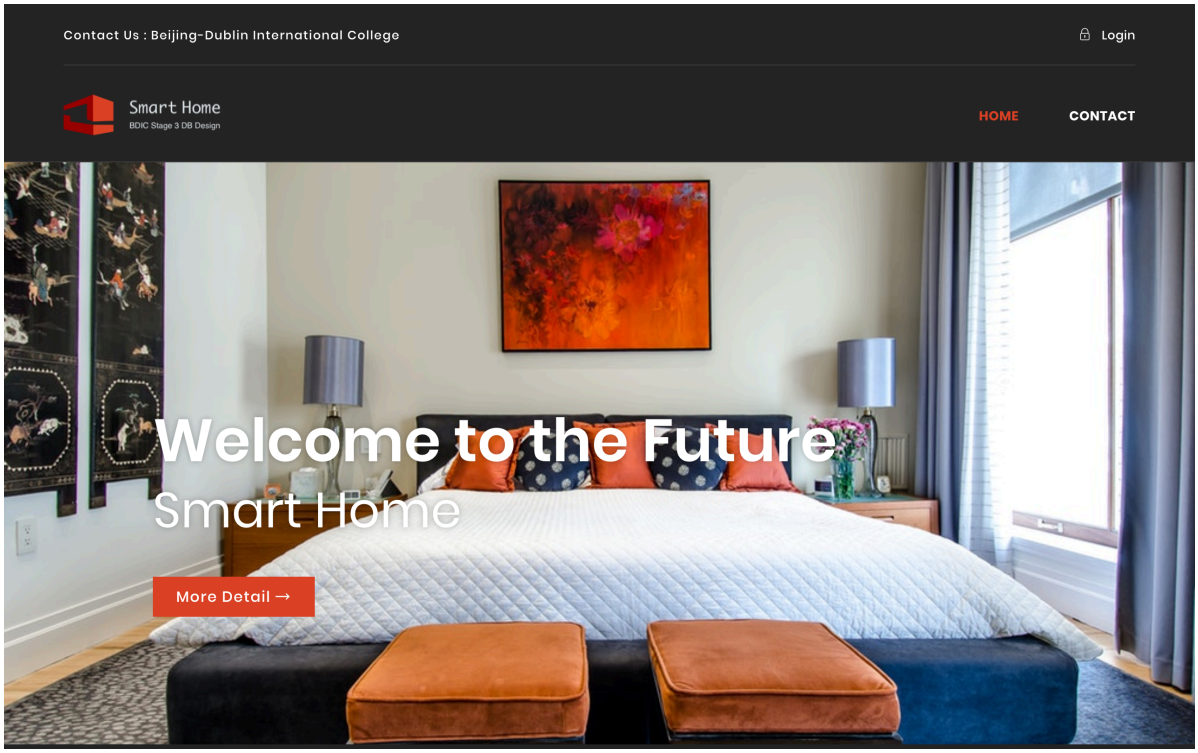
```
CREATE TABLE Curtain (
ActID INT PRIMARY KEY,
Crack INT
);
```

```
mysql> describe Curtain;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| ActID | int(11) | NO   | PRI | NULL    |       |
| Crack | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Section 5. JAVA Project

1. index.jsp

- This page is aimed to serve the user to login the system. To make the page looks beautiful, we add some images into the page.



2. login.jsp

- This is a service page, the duty of this page is to call the login method in UserDAO.java, after that the method will return the sql query result, indicating

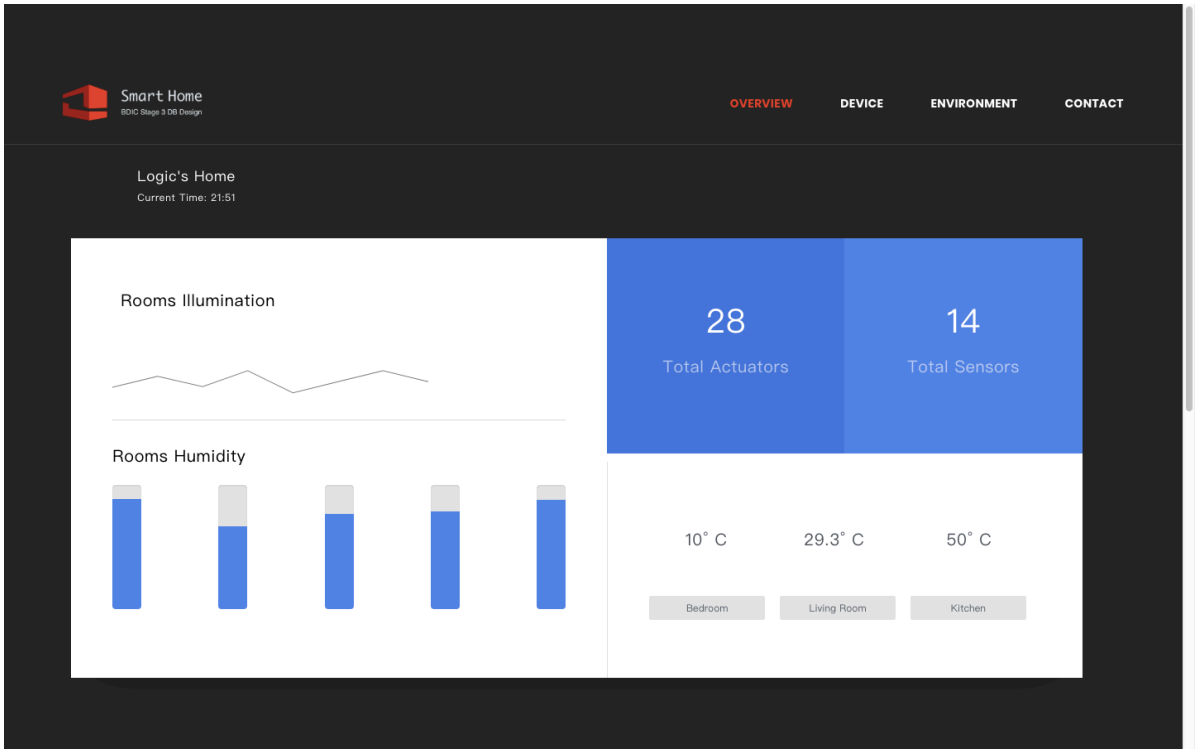
whether the login information is valid. If it is not, the page will redirect to index.jsp, vice the user can enter the next screen called “Overview.jsp”.

3. logout.jsp

- This page is write to withdraw the session information. We use session.invalidate() method to delete the current session.

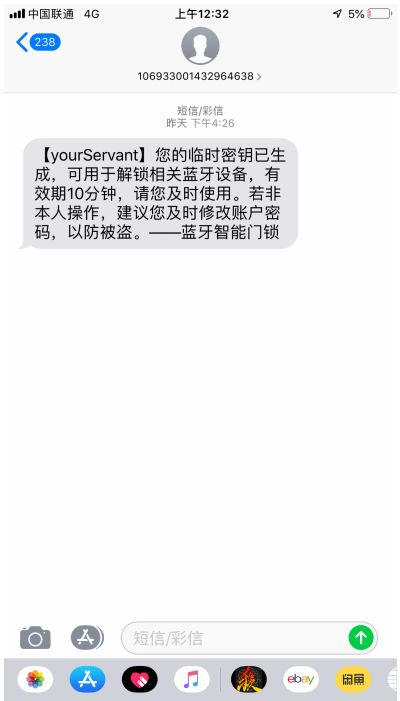
4. Overview.jsp

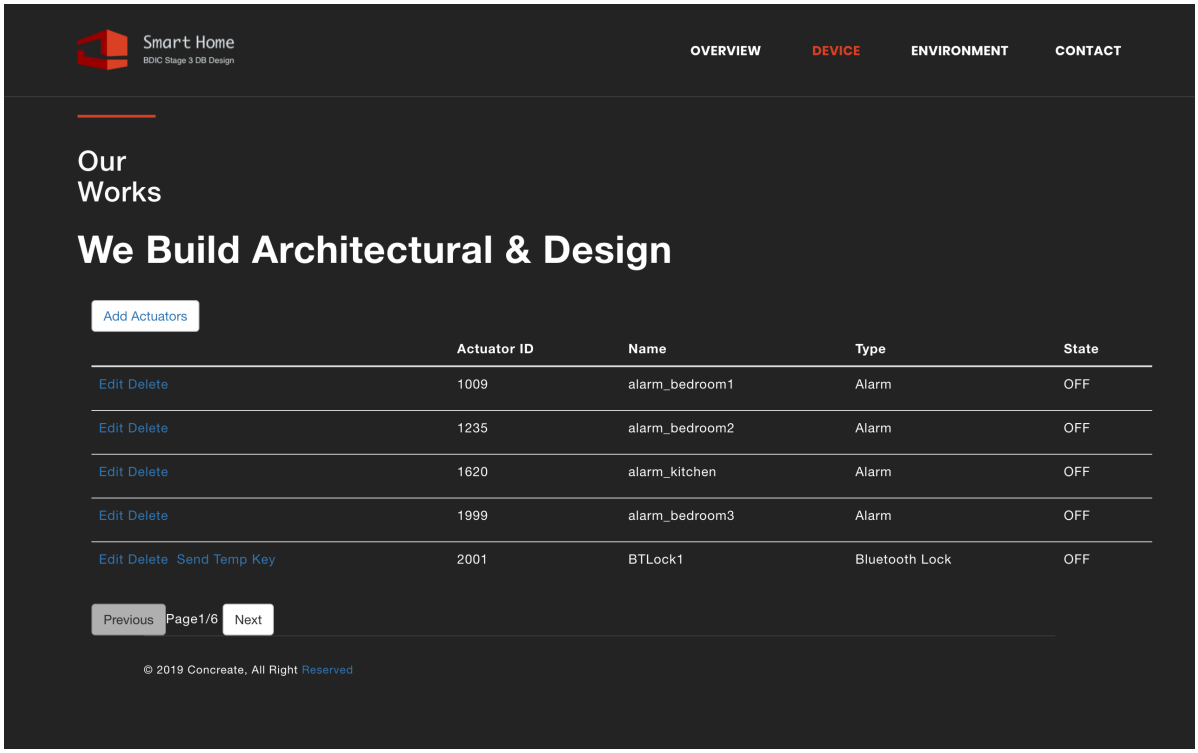
- Overview page give us a glance of the situation in different rooms. This page is a navigation page as soon as the user log into the system.
- From this page, users can check how much actuators and sensors in the rooms (mentioned in detail in Section 7).
- Users can click on the navigation bar to see the devices and sensors in their home.



5. actuator-control.jsp

- This page list all type of actuators in a table, different type of actuators have different characteristics.
- User can **ADD**, **EDIT** and **DELETE** every single actuator, including the name of the actuator and it's state.
- We also use aliyun api to sent message to our phone, indicating that we generate a temp key of the Bluetooth Lock.





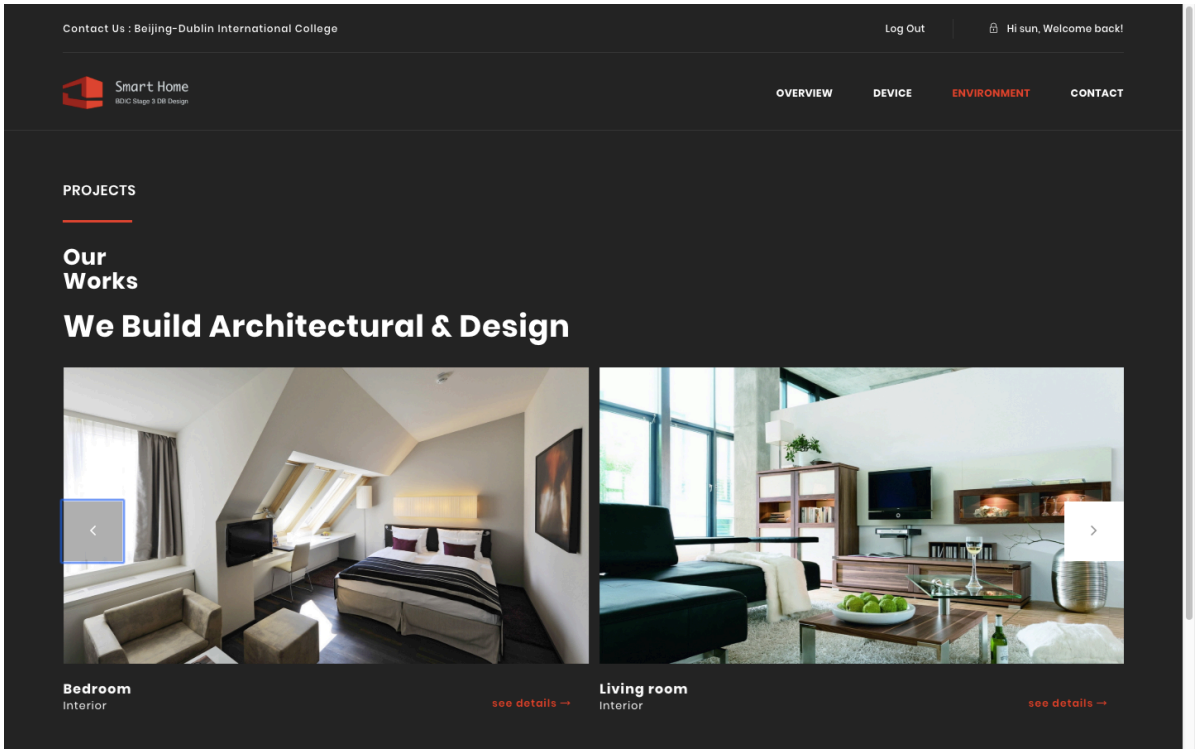
confirmUpdatingActuator.jsp

- These three pages are served for the ADD, EDIT and DELETE actuator and its subtype tables' information.
- The three pages call the methods in java files and return the result of the above three operations.
- Because they are service pages, they have no necessary to design interface, you can check them in the source code.

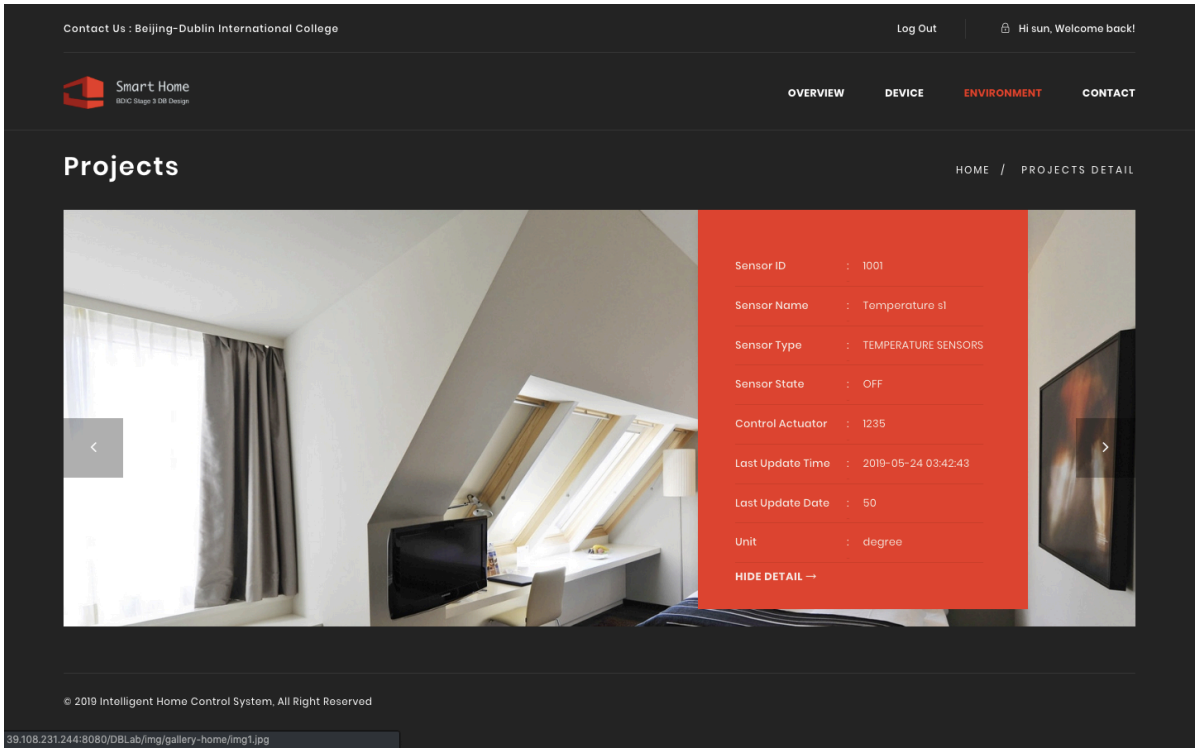
7. environment.jsp

- User can choose specific room's situation they want to look from this page.
- We set three common rooms in our database: **Bedroom**, **Living Room**, **Kitchen.** User can click either link in the page to observe details.

8. bedroom-detail.jsp living-room.jsp kitchen.jsp



- This three page use the method named “getSensorsByRoom(String room)” which we defined in “SensorDAO.java”, this method search sensors in specific rooms. So for the above three jsp pages, we can get sensors in three rooms respectively.
- The three pages are look like the following figure.



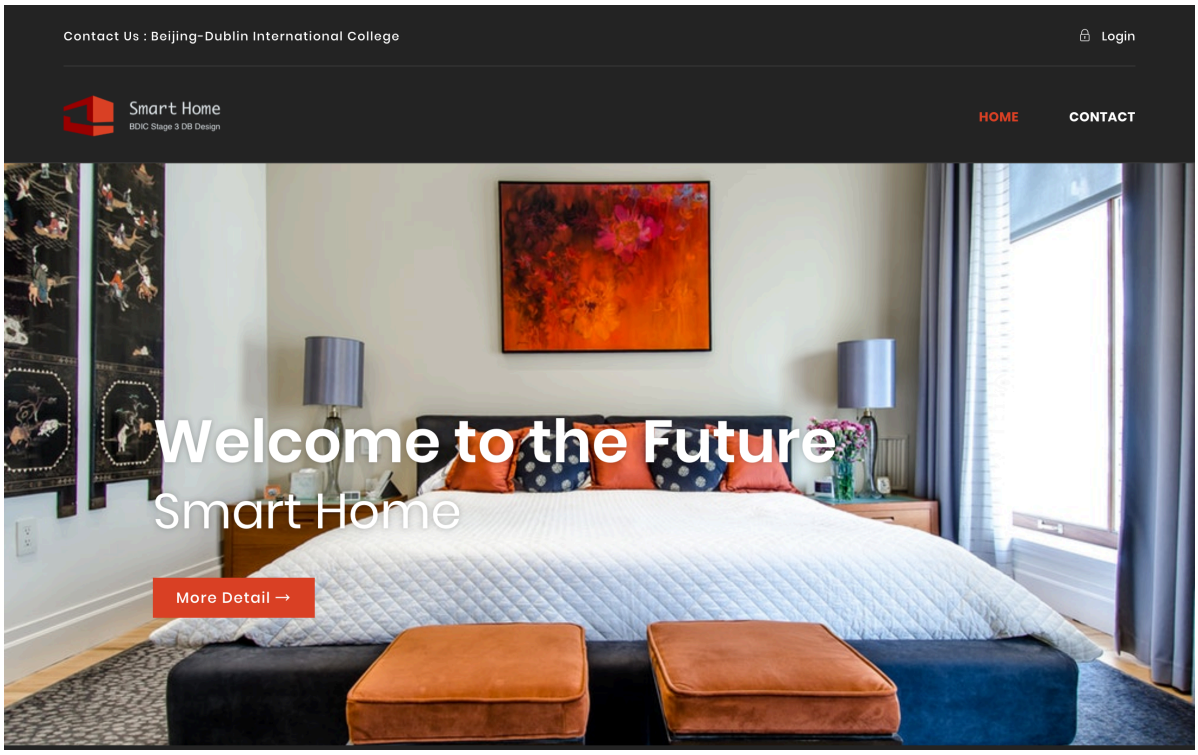
Section 7. Working Showcase

1. Welcome Page

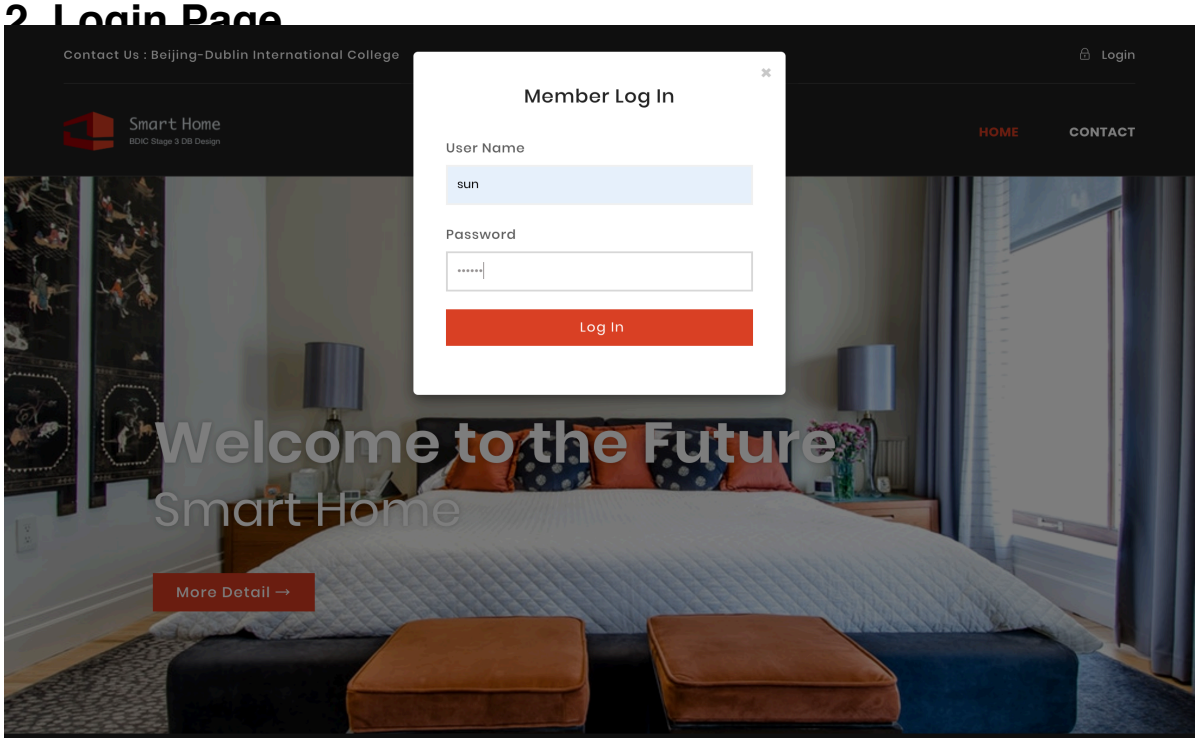
· Click "More Detail" or the "login" button at the top right-hand corner to log in.

· Click "CONTACT" buttons beneath it to get the team information.

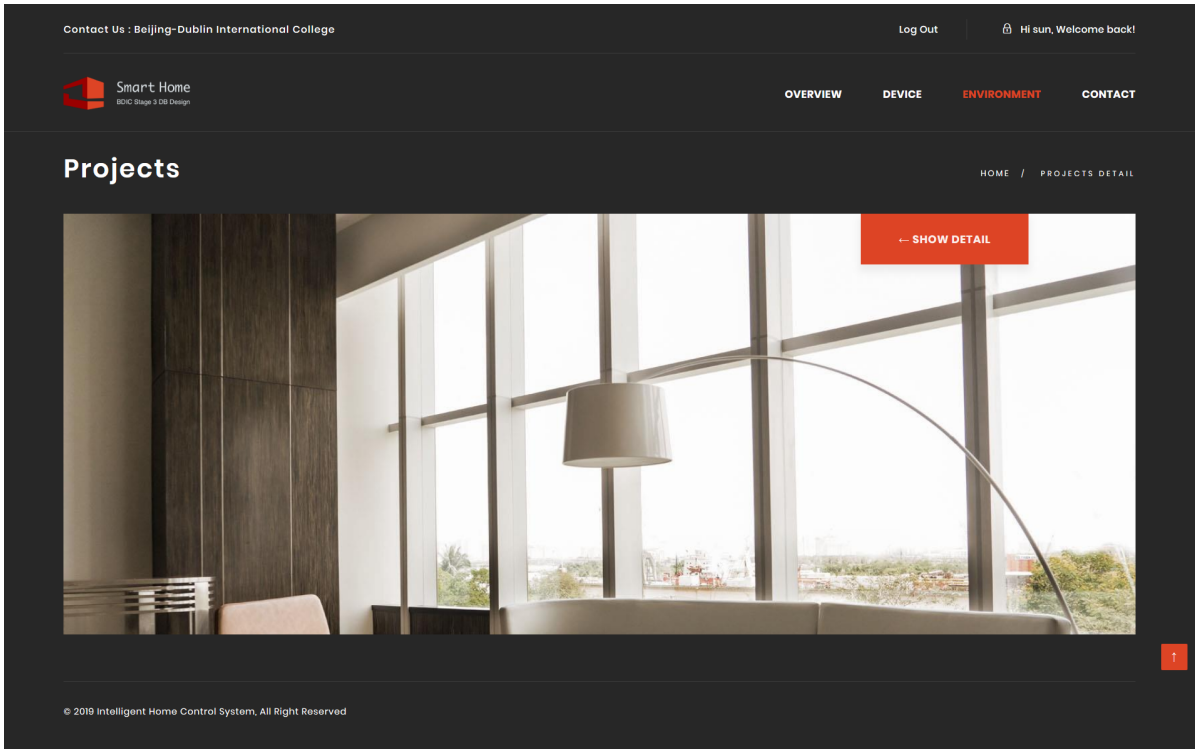
· Click "HOME" button to get back.



2. Login Page

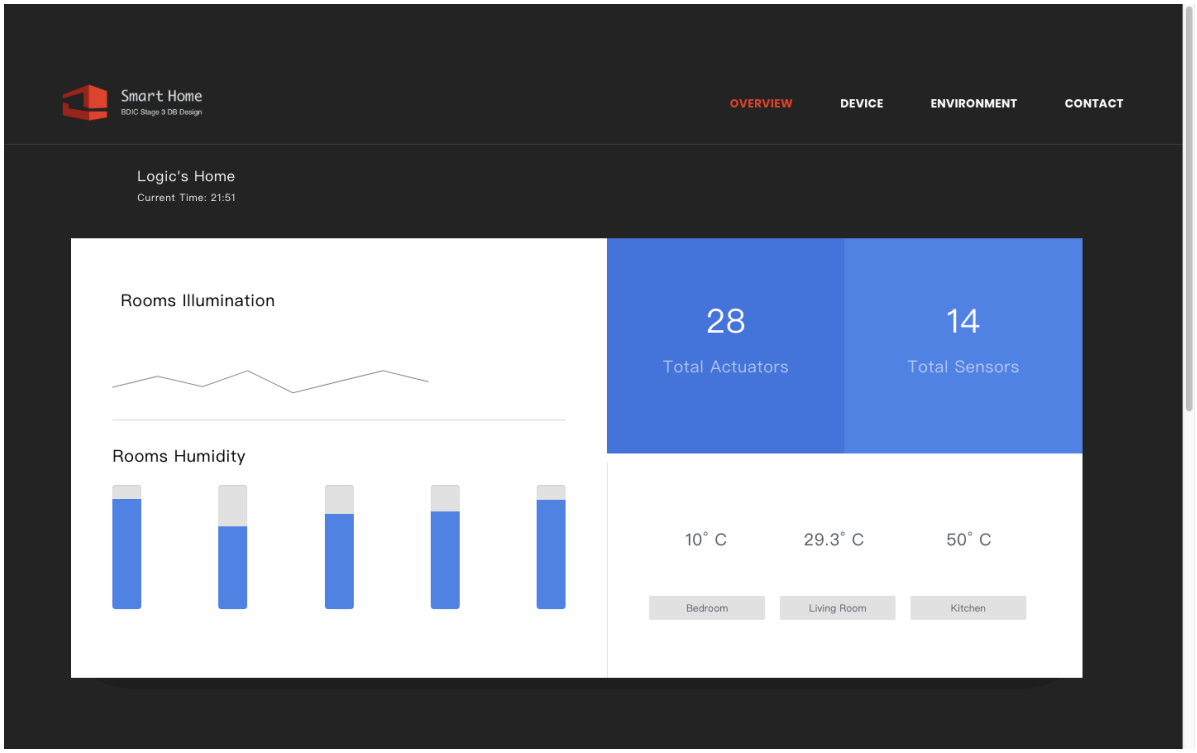


· Input your own user name and password, click "Log In" button and jump to the Overview Page.



3. Overview Page

After logging in, you will enter the main page, which is also the overview page.



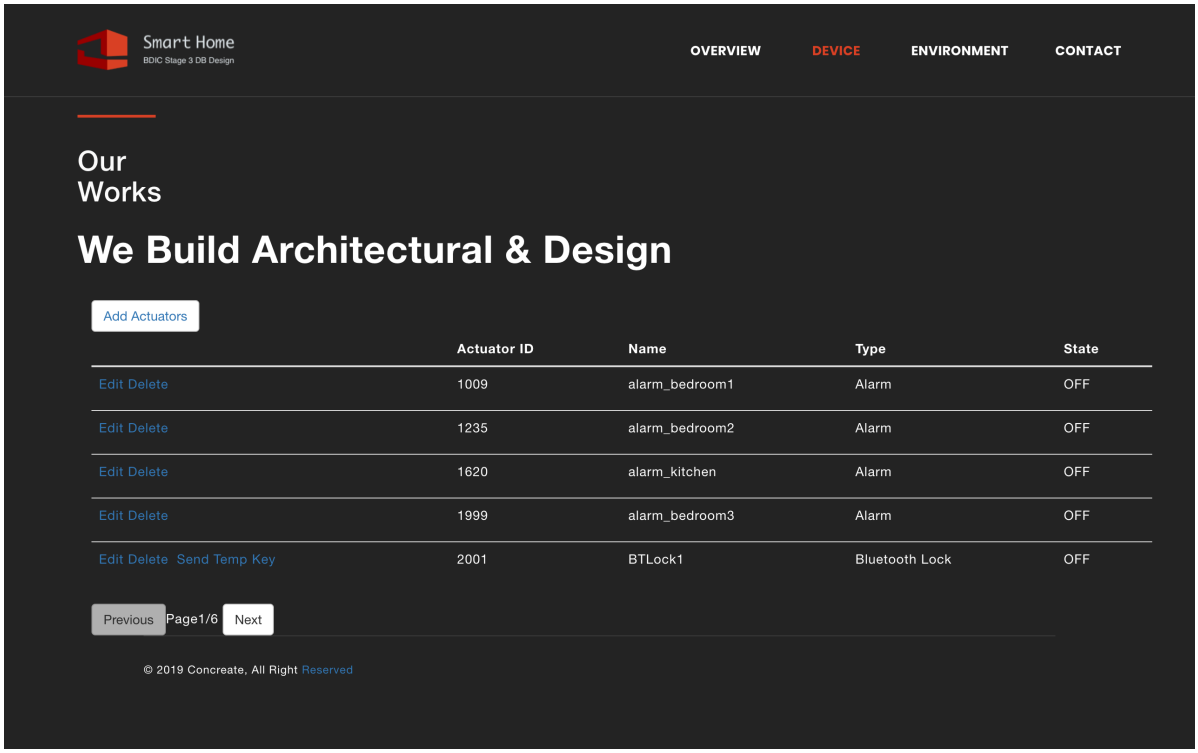
Our team's logo is at the top left corner, combined with the team name. On the opposite side, there are four buttons indicating "OVERVIEW", "DEVICE", "ENVIRONMENT" and "CONTACT".

The dominating part of the overview page is the most common and desirable indexes of home, like temperature, state of working actuators and sensors, real time, humidity and illumination degree of different rooms. Our aim is to illustrate these important indexes as intuitively as possible.

What' more, you can scroll mouse to go through the page. There are huge interactively round button used for setting the working humidity of our humidifier. Besides, we are planning to add real time monitoring function in the future.

4. Actuator Page

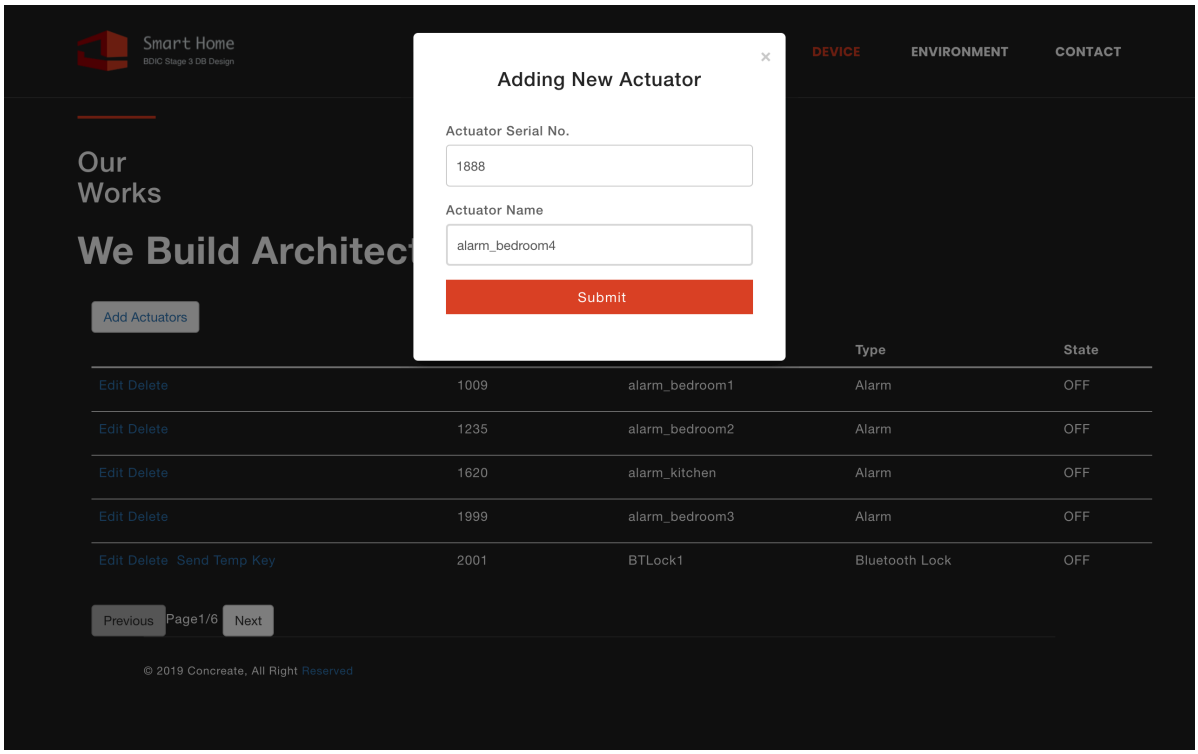
On the next stage, please click "DEVICE" if you want to check the information about actuators in your home, including working state, device type, actuator's ID or even add a new device or delete an old one. Here we will illustrate every operation with aid of screenshoots.



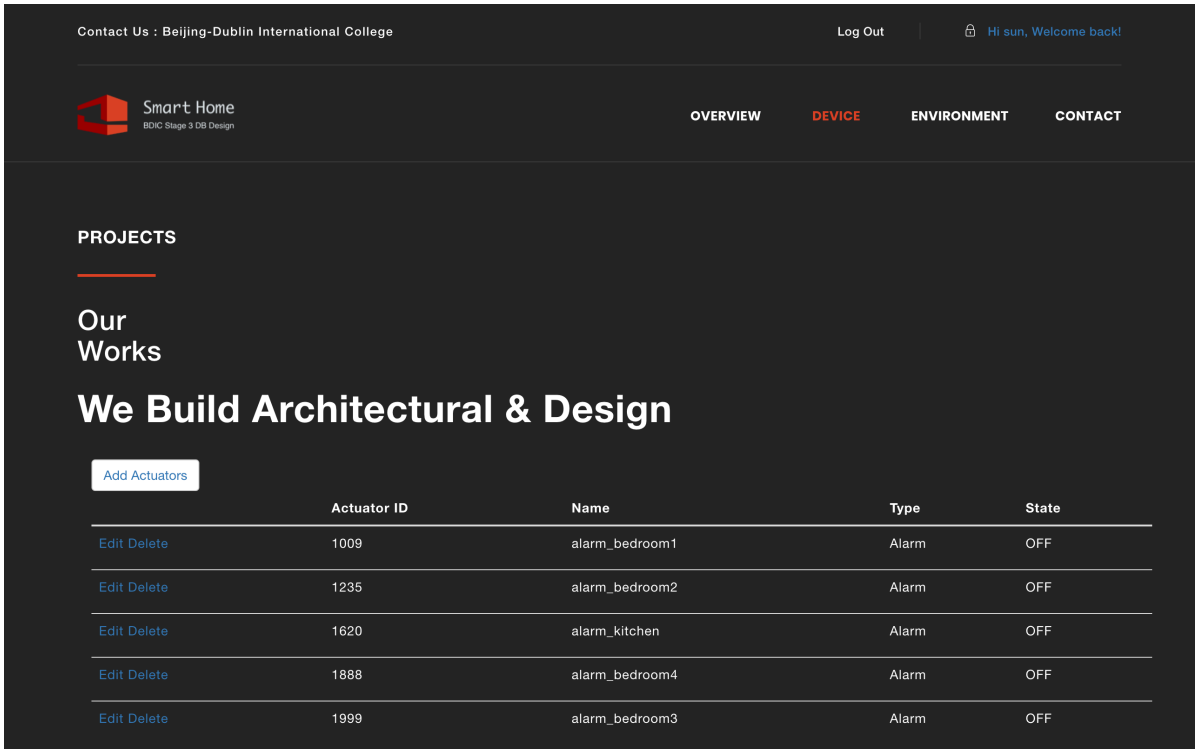
• Addition

· Click "Add Actuators" to add a new device.

For example, add an alarm with ActID 1888 and name “alarm_bedromm4":



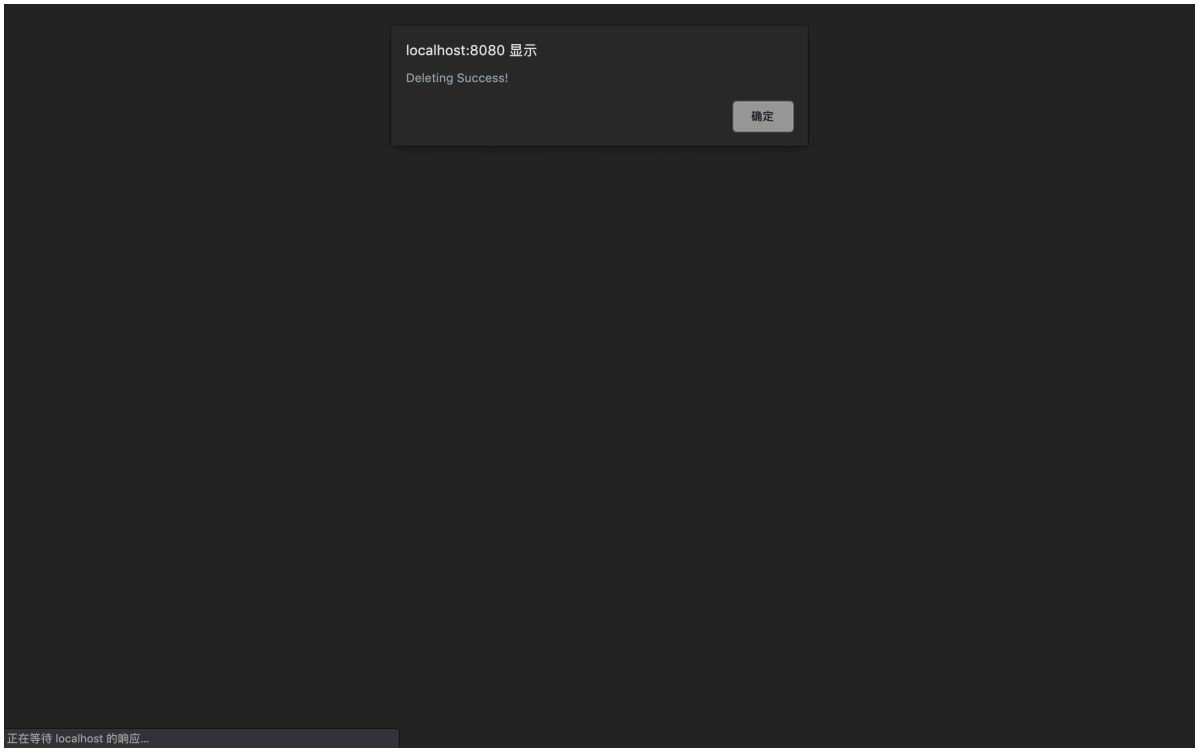
The page after inserting successfully:



•Deletion

· Click "Delete" to delete a present device. In this case, you can delete the alarm that you just inserted:

Indication of deleting successfully:

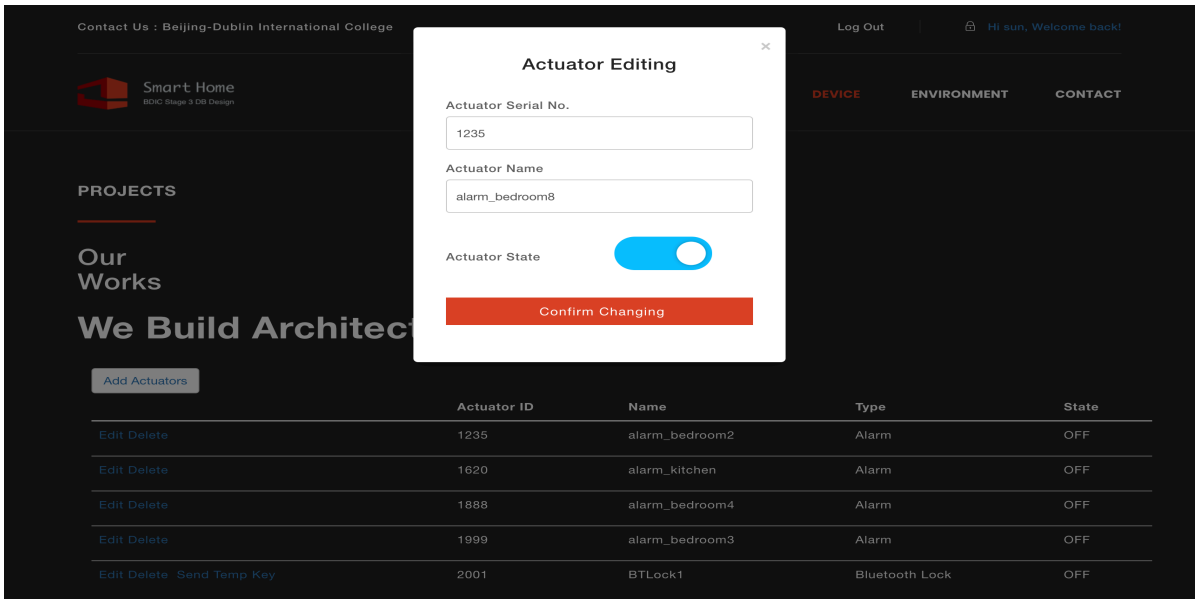


The page after deleting successfully:

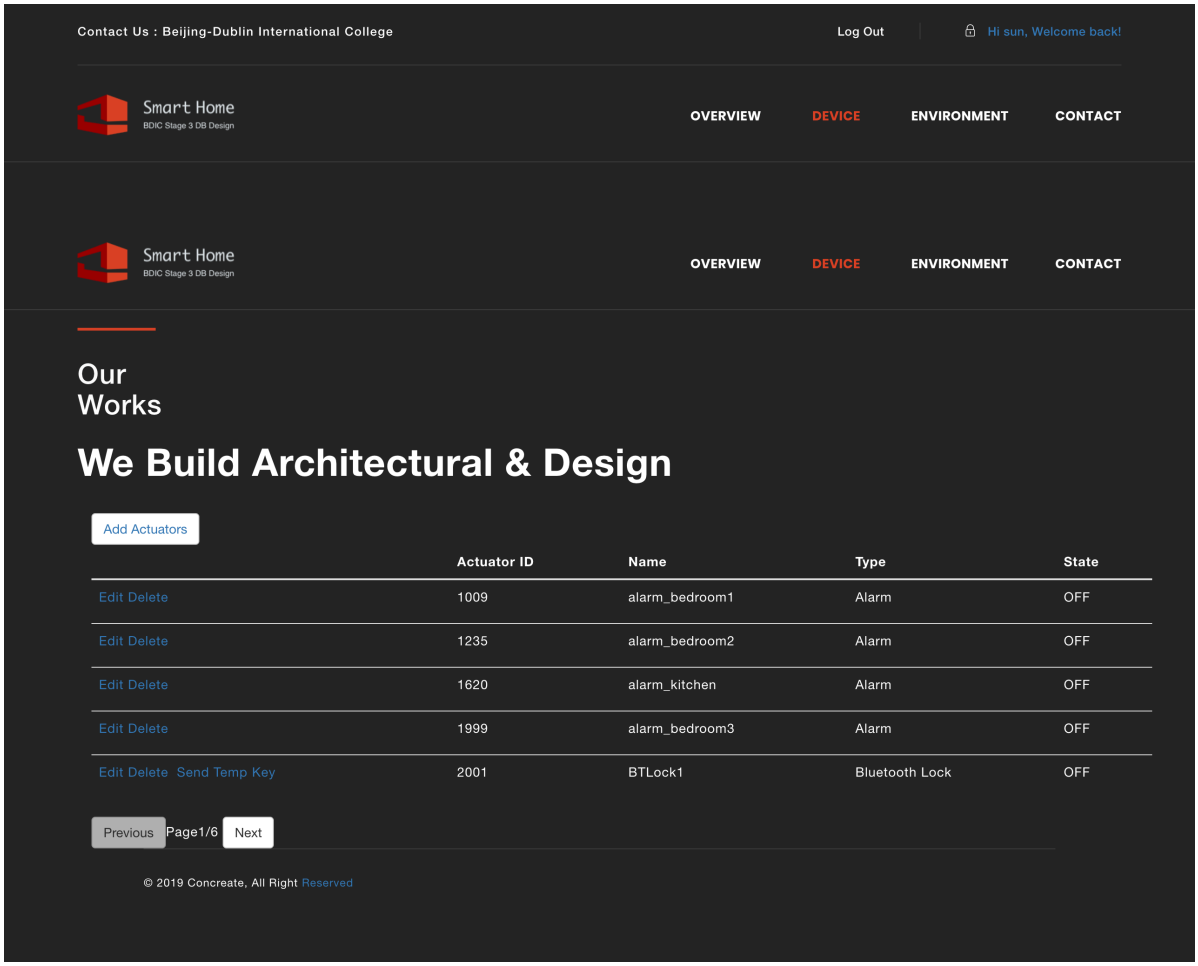
•Edition

· Click "Edit" to edit Actuators' names and relative working state.

For example, the present working state of alarm_bedroom2 is OFF, and we want to turn it on and change its name as alarm_bedroom8:



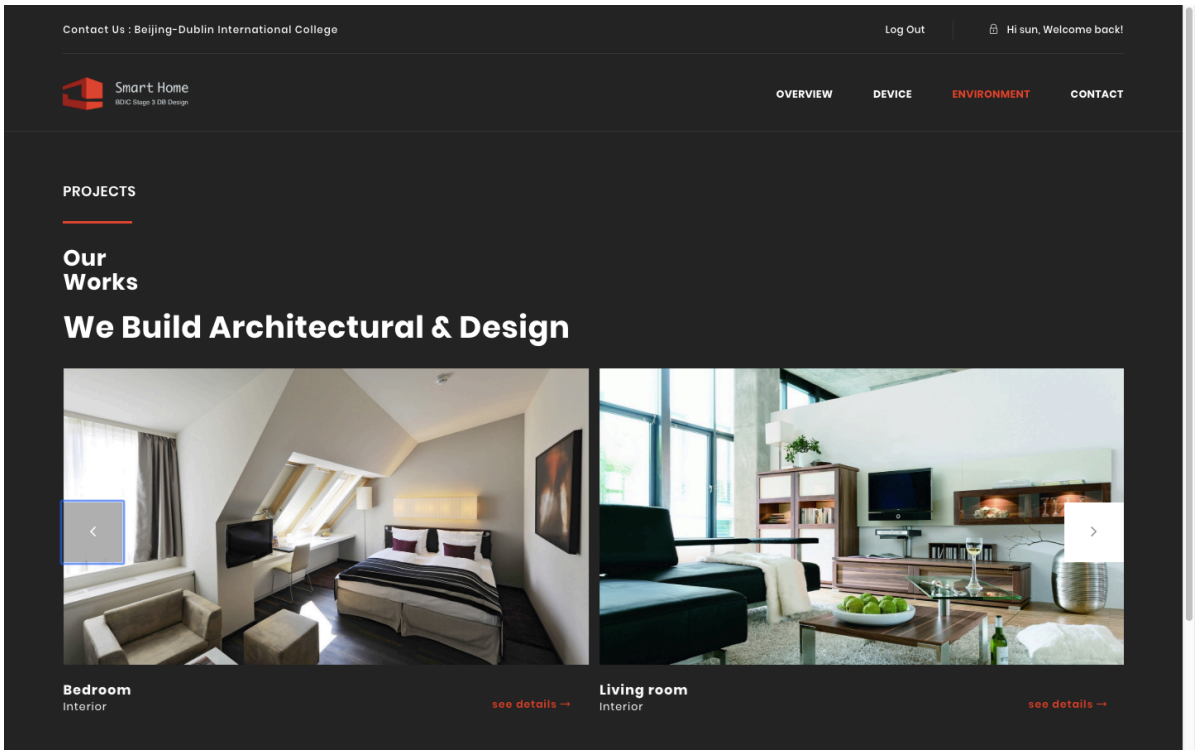
The page after changing successfully:



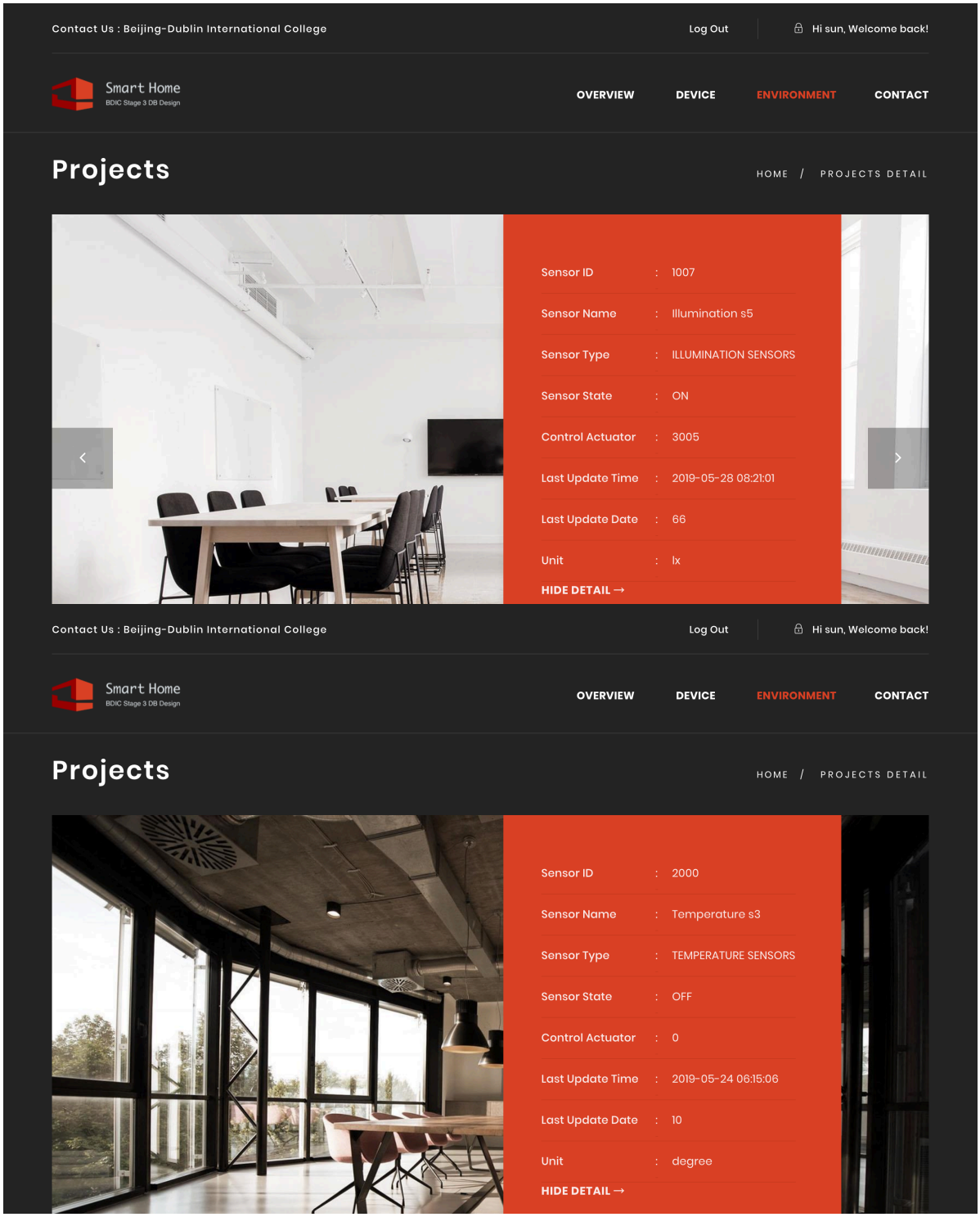
5. Sensor Page

5. Sensor Page

· Click "Environment" to check relative information about sensors within different rooms.

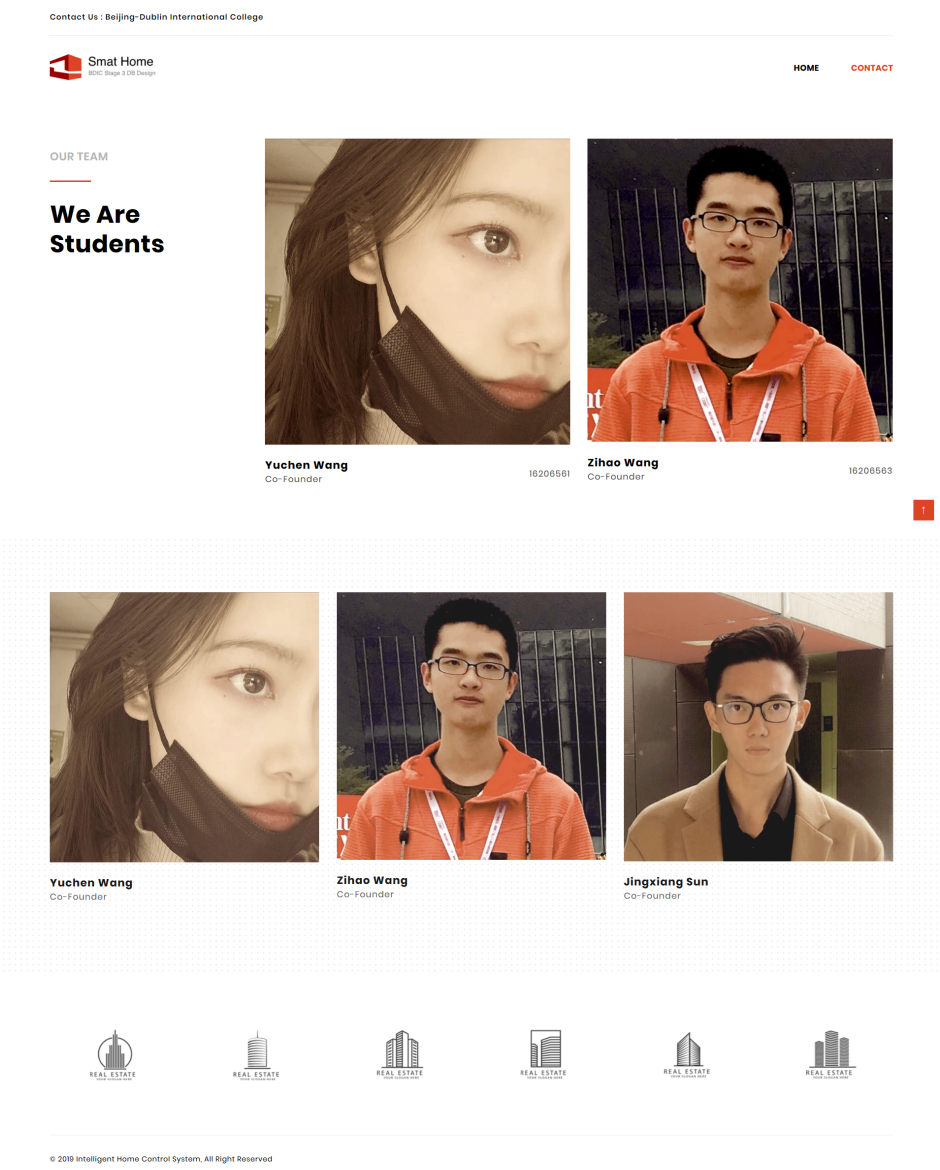


· Click "see details" to specify each room. For instance, in bedroom, here are precise information which are supported by our database shown on the page.



6. Contact Page

· Click "CONTACT" to know more information about us.



Section 8. Team Member Distribution

Zihao Wang : Java Project, Report, Table Creating, Website Design.

Jingxiang Sun : ER Diagram, Report.

Yuchen Wang : Table Creating.

Issues can be raised through the following link:

https://github.com/wzh1998/DBLab

Server IP: 39.108.231.244

Port: 8080

Project Address: http://39.108.231.244:8080/DBLab/index.jsp