CO3102/CO7102
Coursework 2 – Mini Web/App Project

iGSE – An Energy Tool

Important Dates:

Handed out: 10th -Nov-2022
Deadline: 9th -Jan-2023 at 16:59 GMT

Please ensure that you submit your work on time.

- This coursework counts as 25% of your final mark.
- Please read guidelines on plagiarism on https://www2.le.ac.uk/offices/sas2/assessments/plagiarism/penalties
- Learning outcome:
  - Use appropriate server-side and client-side scripting languages to create a web application
  - Solve security and session handling issues and use supporting techniques

Coursework Description

The Valley of Shangri-La is experiencing an unprecedented energy crisis due to the recent disruption to gas supplies. As a result, the government launched a public consultation on how to help its residents manage their fuel bills and keep warm as it gets closer to its winter season. The residents of Shangri-La voted overwhelmingly to support the Energy Bills Support Scheme, and the creation of GSE (Great Shangri-La Energy), a publicly-own energy company committed to providing affordable and sustainable energy supplies to Shangri-La.

Requirements

Your task is to develop a web interface for an energy tool, **iGSE**, to provide Shangri-La residents with a platform to submit regular meter readings and view their bills online, and pay the bills using energy vouchers provided by the government. iGSE may be implemented as a Web Application, a Native Android/iOS app, or a Hybrid app.

In addition, GSE plans to provide open access to energy consumption data and statistics via its platform. Your second task is to develop a REST API to allow the general public to search energy consumption data.

Task 1 – Web/App GUI (80%)

There are two types of accounts in iGSE: (1) **Customer** account and (2) GSE **Admin** account.

Customer account**:** a customer must register first to be able to use the customer dashboard. A new customer will need to provide the detail as follows to complete the registration.

- Customer ID (email address)
- Password
- Address
- Property type (detached, semi-detached, terraced, flat, cottage, bungalow and mansion)
- Number of bedrooms (integer)
- One valid 8-digit Energy voucher code (EVC)

As part of Shangri-La's Energy Bills Support Scheme, the government gives away free energy vouchers worth £200 each to all eligible households. Every energy voucher has a unique 8-digit EVC (Energy Voucher Code). Also, a QR code is printed on the voucher [See Appendix (3) for examples of all valid vouchers]. A customer must either scan the QR code or manually enter a valid EVC to complete the registration. A newly created customer account is pre-loaded with £200 energy credit.

Functional requirements (Customer):

1. A customer can submit new meter readings, which consists of four parts:
    a. Submission date (e.g. 2022-11-05, default value: today)
    b. Electricity meter reading - Day (e.g. 100 kWh)
    c. Electricity meter reading - Night (e.g. 250 kWh)
    d. Gas meter reading (e.g. 800 kWh [1])
2. A customer can view and pay the latest unpaid bill with energy credit*.
3. A customer can top up the credit with a valid EVC*.

* See Appendix (2) for more information on how to calculate energy bills; See Appendix (3) for a list of valid EVC and its QR codes.

iGSE admin account**:** there is only one pre-defined GSE admin account, which has a login name "gse@shangrila.gov.un" and a default password "gse@energy". Bear in mind that any passwords must be stored securely in a database.

Functional requirements (Admin):

1. Admin can set the price per kWh (or unit cost) for the electricity (day/night) and gas.
2. Admin can access meter readings submitted by all customers.
3. Admin can view the energy statistics– show the average gas and electricity consumption (in kWh) per day for all customers based on their latest billing period.

Error handling

The system should display meaningful error messages (using error pages or ajax message). For example:

- Invalid EVC code during the registration or top-up (a voucher code is unique and can only be used once per customer.)
- Invalid customer ID or password.
- Another customer has already used the provided EVC or already scanned the QR code.
- The provided email is already associated with an existing customer.
- The new meter reading is less than a previous reading.

Task 2 REST Service interface (20%)

Your second task is to implement "iGES Open Data REST API" according to the specification below:

2.1 Get the number of properties by property type:

HTTP request:

```
GET /igse/propertycount
```

JSON Response:

```
[
    {
```

[1] Assume measurement unit for gas is kWh not M3 (cubic meter)

```
        "detached":"10"
     },
     {
        "semi-detached":"2"
     },
     {
        "terraced":"2"
     },
     {
        "flat":"2"
     },
…
]
```

2.2 Get energy usage statistics for a specific property type and the number of bedroom.

e.g. Get energy usage statistics for all 3 bedroom semi-detached houses

HTTP request:

```
GET ./igse/semi-detached/3
```

JSON Response:

```
{
   "type":"semi-detached",
   "bedroom":"3",
   "average_electricity_gas_cost_per_day":"9",
   "unit":"pound"
}
```

Marks breakdown

(1) Customer registration (for resident users), log-in/sign-out.	[30 marks]
(2) Customer Dashboard: submit readings, pay bills, credit top-up.	[20 marks]
(3) iGSE admin Dashboard: unit price setting, view bills and statistics.	[30 marks]
(4) REST API	[20 marks]

Note that

- Your solutions to (1)(2)(3) can either be a web application or a native mobile app or hybrid app (Android or iOS); for (4), you are allowed to use any languages or frameworks. See Appendix 1 for more detail.
- Use appropriate techniques to remember the last Customer ID (e.g., Cookies / Shared Preferences)

Feel free to use **Shangri-La.sql** provided on Blackboard for this coursework, and you are free to may any changes you deem necessary. You do NOT have to use it if you intend to use **NoSQL** (e.g. Mongodb, Firebase etc.) **or other data persistence frameworks** (e.g. Spring JPA). If you intended to use departmental MySQL server (mysql.mcscw3.le.ac.uk) for this coursework, please make sure you tested the connection string before the submission.

Submission

- Compress all files in a single zip file for submission via Blackboard:
  - Your Project folder
  - READ.ME (explaining how to deploy and run you application)
  - Your database schema and all data, if applicable (Your_email_ID.sql)
- The archive should be named **CW2_abc123.zip**. Where abc123 is your email id (e.g. CW3_yh37.zip)

You are allowed to re-submit as many times as you like **before** the deadline.

Anonymous marking

We operate an anonymous marking scheme. All submitted submission (or other project files) will be deployed anonymous using the fingerprinting generated by SHA256.

Appendix

(1) Choice of languages & frameworks

You may use **ANY** programming languages for this coursework, including but not limited to:

- Java - Android Studio
- JavaScript -React Native, Ionic etc
- PHP
- Python
- .NET
- Swift/Objective-C – Xcode - iOS
- Any other languages/web technologies for developing Progressive Web Apps

You may use **ANY** framework for this part, including but not limited to:

- Spring MVC
- ASP .NET MVC
- Ruby On Rails
- Node.js/React.js
- Laravel
- Flask
- Angular
- Django
- Ember.js

You are allowed to use any Third-party CSS/JS/HTML libraries e.g., Bootstrap. The architecture and good coding practice will also be considered when allocating marks; For Native app, you are allowed to build you application based on a template. Please email yh37@le.ac.uk if you are unsure whether your chosen framework is permitted.

(2) How to calculate your energy bill

Once the meter readings are submitted, the system should be able to calculate the bill by (1) subtracting a customer's previous reading from their latest reading to get the number of units used over the billing period (2) multiply this figure by the price per kWh (or unit cost) for the electricity (day/night) and gas respectively (3) Count the number of days in billing period and add standing charge. A standing charge is a fixed daily charge you have to pay for energy, no matter how much you use (exclude the current date). You pay a fixed charge per day.

For example:

Given the tariff below:

Electricity (Day) per kWh	Electricity (Night) per kWh	Gas per kWh	Standing charge per day
£0.34	£0.2	£0.1	£0.74

Initial readings date: 2022-11-20

Date	Electricity usage in kWh		Gas (Reading) in kWh
	Day reading	Night reading	
2022-11-20	100	250	800

No payment required

Meter readings date: 2022-12-20

Date	Electricity usage in kWh		Gas (Reading) in kWh
	Day reading	Night reading	
2022-12-20	200	500	1600

When a customer submitted the reading on 2022-12-20, the latest bill (period 2022-11-20 to 2022-12-20) will be generated:

Example: Bill = (200 -100) x 0.34 + (500-250) x 0.2 + (1600-800) x 0.1 + 0.74 x 30 days = £186.2 [2]

Meter readings date: 2023-01-20

Date	Electricity usage in kWh		Gas (Reading) in kWh
	Day reading	Night reading	
2023-01-20	300	800	2500

When a customer submitted the reading on 2023-01-20, if the previous bill was paid in full, the latest bill (period 2022-12-20 to 2023-01-20) will be generated:

Example: Bill = (300 -200) x 0.34 + (800-500) x 0.2 + (2500-1600) x 0.1 + 0.74 x 31 days = £206.94

(3) Valid EVC (Energy Voucher Codes)

Below are a list of valid EVC (Energy Voucher Code) which can be used for testing purpose. A customer must enter one of the following EVC (or scan its QR code) to complete the registration or top up the account. (You might create additional test cases yourself https://www.qr-code-generator.com/)

Energy Voucher	Energy Voucher	Energy Voucher	Energy Voucher
£200	£200	£200	£200
XTX2GZAD	NDA7SY2V	RVA7DZ2D	DM8LEESR

(4) Miscellaneous

Default Admin credentials

```
Username: gse@shangrila.gov.un

Password: gse@energy
 [SHA256-Hash= 7D03BDC673FE753D95EF4E1FE70CEC3D794E26273A75F0F93002E9AB3F6EE5EC]
```

[2]iGSE always use the current tariff to calculate the bills.

```
Username: test@gmail.com
Password: 12345
[SHA256-Hash= 5994471ABB01112AFCC18159F6CC74B4F511B99806DA59B3CAF5A9C173CACFC5]
```

* Feel free to edit **Shangri-La.sql** as you wish

Generating SHA256 Password hash in Java

(See HashGenerator.java)

```
    public static String getSHA256(String data) {
        String result = null;
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] hash = digest.digest(data.getBytes("UTF-8"));
            return bytesToHex(hash); // make it printable
        }catch(Exception ex) {
            ex.printStackTrace();
        }
        return result;
    }
    private static String bytesToHex2(byte[] hash) {
       // return DatatypeConverter.printHexBinary(hash);
       final StringBuilder builder=new StringBuilder();
       for(byte b:hash) {
             builder.append(String.format("%02x", b));
       }
       return builder.toString();

    }
…
```

Calculating difference between two days

```
DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd MM yyyy");
String inputString1 = "23 01 1997";
String inputString2 = "27 04 1997";

try {
    LocalDateTime date1 = LocalDate.parse(inputString1, dtf);
    LocalDateTime date2 = LocalDate.parse(inputString2, dtf);
    long daysBetween = Duration.between(date1, date2).toDays();
    System.out.println ("Days: " + daysBetween);
} catch (ParseException e) {
    e.printStackTrace();
}
```

Marking Scheme

Task 1

N (0%)

- No submission

F (0-40%)

- Static pages (or View/Activity) for sign-up/login exists, but the resident users are not able to submit the form or log in.
- No user authentication. Not roles/permissions control.
- Customer and GSE admin dashboard not implemented.

E (40-50%)

- Web app/Hybrid app: Basic HTML pages/view without any front-end framework or CSS; do not adopt a responsive design; no form validation; major issue with sign-up and login page.
- Native or Hybrid App: basic Page/Activity layouts exist but without any backend; backend issues with the; no form validation; major issue with sign-up and login page.
- Registration page partially implemented (e.g., email/full name etc。) without EVC validation
- Admin page exists, but with hard-coded statistical data

D (50-60%)

- The adoption of responsive web design for Web App; Native or Hybrid App
- Server-side EVC validation implemented.
- Sign-up page and login page works to a certain extent. User authentication and DB backend partially implemented, though there may still be major issues. (e.g., failed to connect, 500 internal server error)
- Customer dashboard partially implemented.
- GSE admin exists and password-protected, though there are issues with the results.

C (60-70%) Meeting the criteria above in D, and

- Sign-up and Login pages work reasonably well; necessary form validation implemented.
- A user can enter the personal detail and manually enter EVC to complete the registration.
- Role/permission control in place despite some minor issues.
- Customer dashboard fully functional.
- Admin dashboard: setting rates and browsing bills partially implemented.

B (70-85%) Meeting the criteria above in C, and

- Sign-up and Login pages work very well
- User-friendly sign-up form validation with detailed error messages (e.g. Use of Ajax for form validations and EVC verification)
- A customer can scan the QR code to autofill EVC.
- The Admin dashboard fully functional; results being correctly displayed.
- Use Cookies or Shared Preference to remember last login name.
- Use appropriate chart to visualise the result in the Admin dashboard (e.g. Google Chart, C3 JS or Chart JS etc)
-

A (>85%) Meeting the criteria above in B, and

- Secure RESTful Service with OAuth2 Tokens.
- Data mashup e.g., Google Map integration for heatmap.

Bonus:

- Extra bonus points will be awarded for additional features, included but not limited to:. Please complete "CW2 Self-assessment form" attached.

Part 2:

REST Service API testing process will be automated. A series of test cases will be used for evaluating each REST service endpoint to see if they behave as expected. Marks will be distributed according to test results.

How your score will be calculated:

$$S_{overall} = \frac{\sum_{i=1}^{p} w_i}{\sum_{i=1}^{n} w_i} \times 80\,\% + S_{part2} \times \ 20\%$$

(Where *p* is the number of passed test cases, *n* is the total number of test cases, *wi* refers to the weight for each test case)