CO3102/CO7102
Coursework 2 – Mini Web/App Project

GEVS – An Online Voting Platform

Important Dates:

Handed out: 14th -Nov-2023
Deadline: 15th -Jan-2024 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://le.ac.uk/policies/regulations/senate-regulations/senate-regulation-11/academic
  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

Next year, an election is scheduled to take place in the Valley of Shangri-La. Shangri-La is a parliamentary democracy, a political party that wins an overall majority (party with the most MPs) in a General Election will form the new government. In this system, all eligible voters in a constituency can choose one candidate to serve as their Member of Parliament (MP) to represent them. The candidate with the highest vote in a constituency becomes the MP, regardless of whether they achieve a majority (50% or more) of the votes cast.

In the past, voters in Shangri-La were required to show an accepted form of photo ID at a polling station before they could cast their vote. However, according to reports, many voters lack the necessary ID to participate in the election. In a historic move, the Shangri-La Election Commission decided to conduct its first-ever online General Election through the official app or a website. Your task is to design and implement **GEVS** (General Election Voting System) for the citizens of Shangri-La to help them determine the valley's future.

Requirements

Your task is to develop GEVS, to provide Shangri-La citizens with a platform to register as voters and cast the votes. GEVS may be implemented as a web application, a native Android/iOS app, or a hybrid app.

In addition, GEVS should provide open access to election data and statistics through its Open Data API. The general public, media, and any organisations will be able to access election data using a REST API.

Task 1 – Web/App GUI (80%)

There are two types of accounts in GEVS: (1) **Voter** and (2) **Election Commission Officer** account.

Voter: Every voter must register first to be able to vote in the election. The details they need to provide are:

- Voter ID (email address)
- Full Name
- Date of Birth
- Password
- Constituency *
- An 8-digit Unique Voter Code (UVC) *
  (*See Appendix 3)

Every eligible Shangri-La voter has received a Poll Card via post. Each Poll Card contains a distinct 8-digit Voter Code (UVC), and its corresponding QR code is also printed on the card. To complete the registration process, a voter must either scan the QR code or manually enter a valid UVC number (refer to Appendix 2). Once the voter account is established, they can sign in and access the **Voter Dashboard** where:

- When the election starts, a voter can select exactly one candidate from their constituency as the Member of Parliament (MP) to represent their constituency.
- Every voter can only cast a single vote, and once it is submitted, they are unable to alter or redo their early ballot.

Election Commission Officer: There is only one pre-defined Election Commission Officer account, which has a login name "election@shangrila.gov.sr" and a default password "shangrila2024$". Bear in mind that any passwords must be stored securely in the database. An Election Commission Officer can access the **Election Commission Dashboard** to:

- Start and end the election.
- Monitor real-time election results in each constituency, showing the vote count for each candidate.
- Upon the conclusion of voting:
    - Announce the winner of the election if a political party gains an overall majority (defined as winning over half of the MP seats).
    - Declare a "Hung Parliament" when no single political party secures an overall majority.

Error handling

GEVS should display corresponding messages (error page or ajax message) when:

- UVC code does not match the record in the database during the registration process.
- Another user has already used the given UVC or has already scanned the QR code
- The provided email is already linked to another registered voter.
- Invalid username or password.

Task 2 REST Service interface (20%)

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

2.1 Obtain the electoral district's vote count:

HTTP request:

```
GET /gevs/constituency/northern-kunlun-mountain
```

JSON Response:

```
{
"constituency":"northern-kunlun-mountain",
 "result":
    [
      {
        "name":"candidate 1",
        "party":"Red Party",
        "vote":"4"
      }
      ,
      { "name":"candidate 2",
        "party":"Blue Party",
        "vote":"2"
```

```
      }
     ,
      { "name":"candidate 3",
        "party":"Yellow Party",
        "vote":"1"
      }
   ]
 }
```

2.2 Return the election result by listing all MP seats won across all electoral districts for every political party.

HTTP request:

```
GET /gevs/results
```

JSON Response when the election is still ongoing:

```
{
  "status": "Completed",
  "winner": "Red Party",
  "seats": [
    {
      "party": "Red Party",
      "seat": "3"
    },
    {
      "party": "Blue Party",
      "seat": "1"
    },
    {
      "party": "Yellow Party",
      "seat": "1"
    },
    {
      "party": "Independent",
      "seat": "0"
    }
  ]
}
```

* When the election is ongoing, status and winner should set as “Pending”; Show winner as "Hung Parliament" when no single political party secures an overall majority after the election ended.

Marks breakdown

(1) Voter registration, log-in/sign-out.	[30 marks]
(2) Voter Dashboard	[20 marks]
(3) Election Commission Dashboard	[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 Voter ID (e.g., Cookies / Shared Preferences)

Feel free to use **Election.sql** provided on Blackboard for this coursework, and you are free to may any changes you deem necessary. You do NOT have to use this file 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 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. CW2_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) 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@leicester.ac.uk if you are unsure whether your chosen framework is permitted.

(2) Valid UVC code and QR codes

Below is a list of all valid Shangri-la UVCs. Voters are required to enter one of the provided codes to complete the registration process.

HH64FWPE	ZSRBTK9S	2TEHRTHJ	K96JNSXY	VFBH8W6W
BBMNS9ZJ	B7DMPWCQ	G994LD9T	PFXB8QXM	7983XU4M
KYMK9PUH	YADA47RL	Q452KVQE	8TEXF2HD	2GYDT5D3
WL3K3YPT	9GTZQNKB	75NKUXAH	N6HBFD2X	LVTFN8G5
JA9WCMAS	KSM9NB5L	DHKVCU8T	K3EVS3NM	UNP4A5T7
Z93G7PN9	BQCRWTSG	TH9A6HUB	5492AC6V	UMT3RLVS
WPC5GEHA	ML5NSKKG	2E5BHT5R	U5LGC65X	TZZZCJV8
RXLNLTA6	D5BG6FDH	556JTA32	BKMKJN5S	UVE5M7FR
7XUFD78Y	2LJFM6PM	LUFKZAHW	JF2QD3UF	W44QP7XJ
DBP4GQBQ	38NWLPY3	DBAD57ZR	NW9ETHS7	9FCV9RMT

Example of QR codes:

HH64FWPE

QR codes of all valid UVC are available to download from Blackboard

(3) Constituencies, Parties and Candidates

Constituency list

Shangri-la-Town
Northern-Kunlun-Mountain
Western-Shangri-la
Naboo-Vallery
New-Felucia

Party list

Blue Party
Red Party
Yellow Party
Independent*

*Non-affiliated politician

Candidate List

You may create your own candidate list for testing purpose. Feel free to edit the sample records in **Election.sql** or make any changes you deem necessary.

(4) Miscellaneous

Default Admin credentials

Username: election@shangrila.gov.sr

Password: shangrila2024$

```
    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();

    }
…
```

Marking Scheme

Task 1

N (0%)

- No submission

F (0-40%)

- Static pages (or View/Activity) for sign-up/login exists, but the voters are unable to register or log in.
- No user authentication. Not roles/permissions control.
- Voter and Election 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.
- Voter registration page partially implemented (e.g., email/full name etc ◦) without UVC validation
- Election Commission page exists, but with hard-coded data.

D (50-60%)

- The adoption of responsive web design for Web App; Native or Hybrid App
- Server-side UVC 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)
- Voting page exists and partially functional.
- Election Commission page 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 UVC to complete the registration.
- Voters can view the candidates and cast a ballot.
- Election Commission dashboard fully functional.

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 UVC verification)
- A voter can scan the QR code to autofill EVC.
- The Election Commission 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 Election Commission 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.

Task 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)