Question 1

Question 1a

Based on the URL in the use case:

i. Explain the structure of the URL,

- **Structure**:
    - **Protocol**: `http://` indicates the Hypertext Transfer Protocol is used.
    - **Subdomain**: `covid-19` specifies a subdomain dedicated to COVID-19 information.
    - **Domain**: `govmu.org` is the main domain, indicating a government website in Mauritius.

ii. Highlight its related issue,

- **Issue**: The URL uses `http` instead of `https`, which means the data transmitted between the client and server is not encrypted, posing a security risk.

iii. and propose a solution to address it.

- **Solution**: Switch to `https` by obtaining an SSL/TLS certificate for the domain to ensure that all data transmitted is encrypted and secure.

Question 1b

Describe how you will structure this project?

1. **Project Structure**:
    - **Frontend**:
        - **HTML/CSS**: Structure and style of web pages.
        - **JavaScript/jQuery**: For dynamic content and interaction, including the date picker and AJAX requests.
    - **Backend**:
        - **Server-side Language**: PHP or Python for handling logic and database interactions.

    - **Database**: MySQL or PostgreSQL for storing COVID-19 stats and patient records.
- **Security**:
    - **HTTPS**: Secure data transmission.
    - **Authentication**: Login system for authorized health personnel.
    - **Data Validation**: Ensure the data entered is correct and secure.
- **API**:
    - **Endpoints**: For retrieving and updating statistics.
- **Deployment**:
    - **Web Server**: Apache or Nginx for hosting the web application.
    - **Version Control**: Git for tracking changes and collaboration.

Question 1c

The following terms are usually used in web development:

i. Describe each term.

- **$_POST**:
    - **Description**: A global PHP variable used to collect form data after submitting an HTML form with method="post".
- **$_GET**:
    - **Description**: A global PHP variable used to collect form data after submitting an HTML form with method="get".
- **$_SESSION**:
    - **Description**: A way to store information (in variables) to be used across multiple pages in PHP.
- **$_COOKIE**:
    - **Description**: A small file stored on the user's computer used to remember information about the user.
- **AJAX**:
    - **Description**: A technique for creating fast and dynamic web pages by

exchanging data with a web server asynchronously.

ii. and in what specific scenarios will you use them in the above use case?

- $_POST: Used to handle form submissions for updating COVID-19 stats.
- $_GET: Used to retrieve and display statistics for a specific date based on the date parameter in the URL.
- $_SESSION: Used to manage user login sessions to ensure only authorized personnel can update stats.
- $_COOKIE: Could be used to remember user preferences or settings.
- AJAX: Used to dynamically update the stats displayed on the homepage without reloading the page.

Question 1d

Describe the structure of the patient table that you will use for storing the above information.

- Patient Table Structure:
    - Columns:
        - `id`: INT, Primary Key, Auto Increment
        - `nid`: VARCHAR(12), Not Null
        - `name`: VARCHAR(100), Not Null
        - `age`: INT, Not Null
        - `gender`: VARCHAR(6), Not Null
        - `status`: ENUM('New Case', 'Recovered', 'Death'), Not Null
        - `date_reported`: DATE, Not Null

Question 1e

Write the SQL create command to create the above table, paying attention to the field type and default values.

```
CREATE TABLE patient (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nid VARCHAR(12) NOT NULL,
    name VARCHAR(100) NOT NULL,
    age INT NOT NULL,
    gender VARCHAR(6) NOT NULL,
    status ENUM('New Case', 'Recovered', 'Death') NOT NULL,
    date_reported DATE NOT NULL
);
```

Question 1f

Write the SQL query to get the count of death for a particular date (e.g., 02 NOV 2021).

```
SELECT COUNT(*) AS death_count
FROM patient
WHERE status = 'Death' AND date_reported = '2021-11-02';
```

Question 2

Question 2a

Complete the following Perl code that will print the reverse of a string:

```
my $string = "\n?sihtdaeruoynac";
print scalar reverse $string;
```

Question 2b

Provide four (4) examples where cookies can be used.

1. **User Authentication**: Storing session tokens or user login information to keep the user authenticated across different pages.

    - Example: Remembering a user's login session on a website so they don't have to log in again each time they visit a new page.
2. **User Preferences**: Saving user preferences such as theme, language, or layout settings.
    - Example: Keeping a website in dark mode if the user has selected that preference.
3. **Shopping Cart**: Maintaining the state of a shopping cart in an e-commerce website.
    - Example: Storing items that a user has added to their shopping cart so they can continue shopping without losing their selections.
4. **Analytics and Tracking**: Tracking user behavior for analytics purposes.
    - Example: Recording page views and user interactions on a website to gather data on user behavior and preferences.

Question 2c

List down three (3) AJAX security concerns.

1. **Cross-Site Scripting (XSS)**:
    - Attackers can inject malicious scripts into web pages, which can then be executed in the context of another user's session. This can lead to data theft, session hijacking, and other malicious activities.
2. **Cross-Site Request Forgery (CSRF)**:
    - An attacker tricks a user into performing actions on a web application in which they are authenticated, without the user's consent. This can result in unauthorized actions such as changing account details or making transactions.
3. **Data Exposure**:
    - Sensitive data transmitted via AJAX requests can be intercepted if not properly encrypted. Using plain HTTP instead of HTTPS can expose data to man-in-the-middle attacks.

Question 2d

Give five (5) examples of how you can target an HTML element using jQuery.

1. **By ID**:
    - **Example**: `$("#elementID")`

```
$("#elementID").text("Hello, World!");
```

2. **By Class**:
   - **Example**: `$(".elementClass")`

     ```
     $(".elementClass").css("color", "red");
     ```

3. **By Tag Name**:
   - **Example**: `$("p")`

     ```
     $("p").hide();
     ```

4. **By Attribute**:
   - **Example**: `$("[name='elementName']")`

     ```
     $("[name='elementName']").val("New Value");
     ```

5. **By Descendant**:
   - **Example**: `$("div p")`

     ```
     $("div p").addClass("highlight");
     ```

These examples demonstrate various ways to select and manipulate HTML elements using jQuery.

Question 3

Question 3a

Describe five (5) categories of attacks and the solution to protect against each one of them.

1. **SQL Injection**:
   - **Description**: Attackers inject malicious SQL code into input fields to manipulate the database.

    - **Solution**: Use prepared statements and parameterized queries to sanitize inputs and prevent SQL injection.
2. **Cross-Site Scripting (XSS)**:
    - **Description**: Attackers inject malicious scripts into web pages viewed by other users.
    - **Solution**: Validate and sanitize user input, and use Content Security Policy (CSP) to prevent execution of malicious scripts.
3. **Cross-Site Request Forgery (CSRF)**:
    - **Description**: Attackers trick users into performing actions they did not intend by exploiting authenticated sessions.
    - **Solution**: Implement CSRF tokens in forms and AJAX requests to validate the legitimacy of requests.
4. **Denial of Service (DoS)**:
    - **Description**: Attackers overwhelm the server with requests, making it unavailable to legitimate users.
    - **Solution**: Use rate limiting, web application firewalls, and scalable infrastructure to mitigate the impact of DoS attacks.
5. **Man-in-the-Middle (MITM)**:
    - **Description**: Attackers intercept and alter communication between two parties without their knowledge.
    - **Solution**: Use HTTPS to encrypt data transmission and ensure secure communication channels.

Question 3b

Describe how you will ensure that these are met when developing a website:

Confidentiality

- **Description**: Ensure that sensitive data is accessible only to authorized users.
- **Measures**:
    - Implement strong encryption for data at rest and in transit.
    - Use secure authentication and authorization mechanisms.
    - Implement access controls and role-based access permissions.

Access Control

- **Description**: Restrict access to resources based on user roles and permissions.
- **Measures**:
    - Implement role-based access control (RBAC) to assign permissions based on user roles.
    - Use authentication mechanisms such as OAuth, JWT, or LDAP.
    - Enforce least privilege principle, granting only necessary permissions to users.

Integrity

- **Description**: Ensure that data is accurate and has not been tampered with.
- **Measures**:
    - Use hashing and digital signatures to verify the integrity of data.
    - Implement input validation and sanitization to prevent data corruption.
    - Use version control and backups to maintain data integrity.

Availability

- **Description**: Ensure that the website and its services are available to users when needed.
- **Measures**:
    - Use load balancing and redundancy to distribute traffic and prevent single points of failure.
    - Implement regular backups and disaster recovery plans.
    - Monitor system performance and use scalable infrastructure to handle high traffic.

Question 3c

State two (2) advantages and two (2) disadvantages of CGI scripts.

Advantages:

1. **Language Flexibility**: CGI scripts can be written in various programming languages, such as Perl, Python, and C, providing flexibility for developers.
2. **Simplicity**: CGI scripts are easy to understand and implement, making them suitable for small-scale web applications.

Disadvantages:

1. **Performance Overhead**: Each CGI request spawns a new process on the server, leading to high resource consumption and slower performance.
2. **Scalability Issues**: CGI scripts are not suitable for handling high traffic due to the overhead of creating new processes for each request.

Question 3d

Describe the following CGI variables:

i. SERVER_NAME

- **Description**: Contains the server's hostname, DNS alias, or IP address as it appears in the URL.

ii. REMOTE_HOST

- **Description**: Contains the hostname of the client making the request, if available. Otherwise, it may be empty or contain the IP address.

iii. REQUEST_METHOD

- **Description**: Contains the method used to make the request, such as GET, POST, HEAD, etc.
