ST10092354
ICE Task 3

Question 1
Creating a self-signed SSL certificate involves generating a private key, creating a Certificate Signing Request (CSR), and then creating the certificate. While useful for testing and internal applications, self-signed certificates lack the trust level of those issued by recognized Certificate Authorities (CAs). Here's a simplified process to create one:

1. Generate a private key.
2. Create a CSR using the private key.
3. Create the self-signed certificate using the CSR.

Question 2
A self-signed SSL certificate can be used in a web application to secure server-client communication via HTTPS by encrypting data over the network. Here's how:

1. Establishing a Secure Connection

When the client accesses your app over HTTPS, the server presents its self-signed certificate during the SSL handshake. The client may warn about trust since the certificate isn't from a recognized CA.

2. Encryption of Data

Despite the trust warning, the SSL handshake establishes secure, encrypted communication between the client and server using session keys.

3. Data Integrity

SSL ensures data integrity, verifying that data hasn't been altered during transmission.

4. Authentication

A self-signed certificate allows server authentication, with users explicitly acknowledging and accepting the certificate.

5. Development and Testing

Self-signed certificates are cost-effective and quick to set up, making them ideal for development and testing environments.

6. Educational Purposes

Self-signed certificates are useful for teaching SSL/TLS configurations and security practices without a CA.

7. Implementation in Web Applications

- **Server Configuration:** Configure the web server to use the certificate.
- **Testing:** Access your app using HTTPS to test.
- **User Acknowledgment:** Inform users how to bypass trust warnings if needed.