1. Differences Between HTTP and HTTPS

- **Security**: The main difference between HTTP and HTTPS is security. HTTP sends data over the network in plain text, which can be intercepted by attackers (eavesdropping). HTTPS uses SSL/TLS encryption to secure the data.
- **Port**: HTTP operates on port 80, while HTTPS operates on port 443.
- **Certificate**: HTTPS requires a **SSL/TLS certificate**, which ensures that the server is authentic and establishes a secure connection.
- **Data Integrity**: HTTPS ensures that data is not altered during transfer by using message integrity checks.
- **Trust**: Browsers show a padlock symbol or a secure message in the address bar when a site uses HTTPS, signaling users that their connection is secure.

2. Structure of an HTTP Request and Response

HTTP Request Structure:

- **Request Line**: Contains the HTTP method, the path of the resource, and the protocol version (e.g., `GET /index.html HTTP/1.1`).
- **Headers**: Additional information sent with the request, such as:
  - `Host`: Specifies the domain name of the server (e.g., `Host: www.example.com`).
  - `User-Agent`: Information about the client making the request (e.g., browser or app).
  - `Accept`: Specifies the types of media the client can accept.
  - `Content-Type`: For requests like `POST`, specifies the data format being sent.
- **Body**: Contains data being sent to the server (used in `POST`, `PUT`, etc.).

HTTP Response Structure:

- **Status Line**: Indicates the HTTP version, the status code, and a brief reason phrase (e.g., `HTTP/1.1 200 OK`).
- **Headers**: Provide meta-information about the response, such as:
  - `Content-Type`: The type of data returned (e.g., `Content-Type: text/html`).
  - `Content-Length`: The length of the response body in bytes.
  - `Set-Cookie`: Used to store cookies on the client.
- **Body**: Contains the actual data returned by the server (e.g., HTML content).

3. Common HTTP Methods

- **GET**:

    - **Description**: Requests data from a server (read-only).
    - **Use Case**: Fetching a web page or reading data from an API.
- **POST**:
    - **Description**: Submits data to be processed by the server.
    - **Use Case**: Submitting form data or uploading a file.
- **PUT**:
    - **Description**: Updates an existing resource or creates a new one.
    - **Use Case**: Updating user data or replacing a file on a server.
- **DELETE**:
    - **Description**: Deletes a specified resource.
    - **Use Case**: Removing a record from a database or deleting a file.

4. Common HTTP Status Codes

- **200 OK**:
    - **Description**: The request was successful, and the server returned the requested resource.
    - **Scenario**: A webpage successfully loads in the browser.
- **201 Created**:
    - **Description**: The request was successful, and a new resource was created.
    - **Scenario**: When a user registers successfully and a new user account is created.
- **301 Moved Permanently**:
    - **Description**: The requested resource has been permanently moved to a new URL.
    - **Scenario**: A website URL has changed, and the user is redirected to the new location.
- **404 Not Found**:
    - **Description**: The server cannot find the requested resource.
    - **Scenario**: A user tries to access a non-existent page on a website.
- **500 Internal Server Error**:
    - **Description**: The server encountered an error and couldn’t complete the request.
    - **Scenario**: A bug in the server-side code causes the server to crash when handling a request.