## Part-1: Short Answer Questions

1. What is client-side and server-side in web development, and what is the main difference between the two?

**Answer:** The main difference between client-side and server-side is that client-side code is executed on the user's computer, while server-side code is executed on the web server. This means that client-side code can be used to create interactive and dynamic web pages, while server-side code is typically used to store and process data.

***Main Difference between client-side and server-side:***

| Client-side | Client-side |
|---|---|
| Source code is visible to the user | Source code is not visible to the user because its output of server-side is an HTML page. |
| Its main function is to provide the requested output to the end user | Its main function is to provide the requested output to the end user |
| It runs on the user’s computer | It runs on the webserver |
| It does not provide security for data | It provides more security for data |
| HTML, CSS, and JavaScript are used | PHP, Python, Java, Ruby are used |

2. What is an HTTP request and what are the different types of HTTP requests?

**Answer:** An HTTP request is a message that is sent from a client to a server. It is used to request a resource, such as a web page, from the server. HTTP requests are used to communicate between clients and servers.

- **GET:** It is used to retrieve a resource from the server.
- **POST:** This request is used to send data to the server. It is often used to submit forms or to upload files.
- **PUT:** This request is used to replace an existing resource on the server.
- **PATCH:** The PATCH request is similar to the PUT request, but it is used to make a partial update to an existing resource on the server. It sends data to the server to modify specific parts of the resource, rather than replacing the entire resource.
- **DELETE:** This request is used to delete a resource from the server.
- **OPTIONS:** This request is used to get information about the server's capabilities.
- **HEAD:** This request is similar to GET, but it does not return the body of the resource.

## 3. What is JSON and what is it commonly used for in web development?

**Answer:** JSON stands for JavaScript Object Notation. It is a lightweight data-interchange format. It is easy for humans to read and write. It is also easy for machines to parse and generate. JSON is commonly used in web development for transmitting data between a server and a web application. JSON represents data as key-value pairs, similar to how objects are structured in many programming languages. It supports various data types, including strings, numbers, Boolean, arrays, and objects. JSON data is often stored in plain text files with a .json extension.

Here are some of the common uses of JSON in web development:

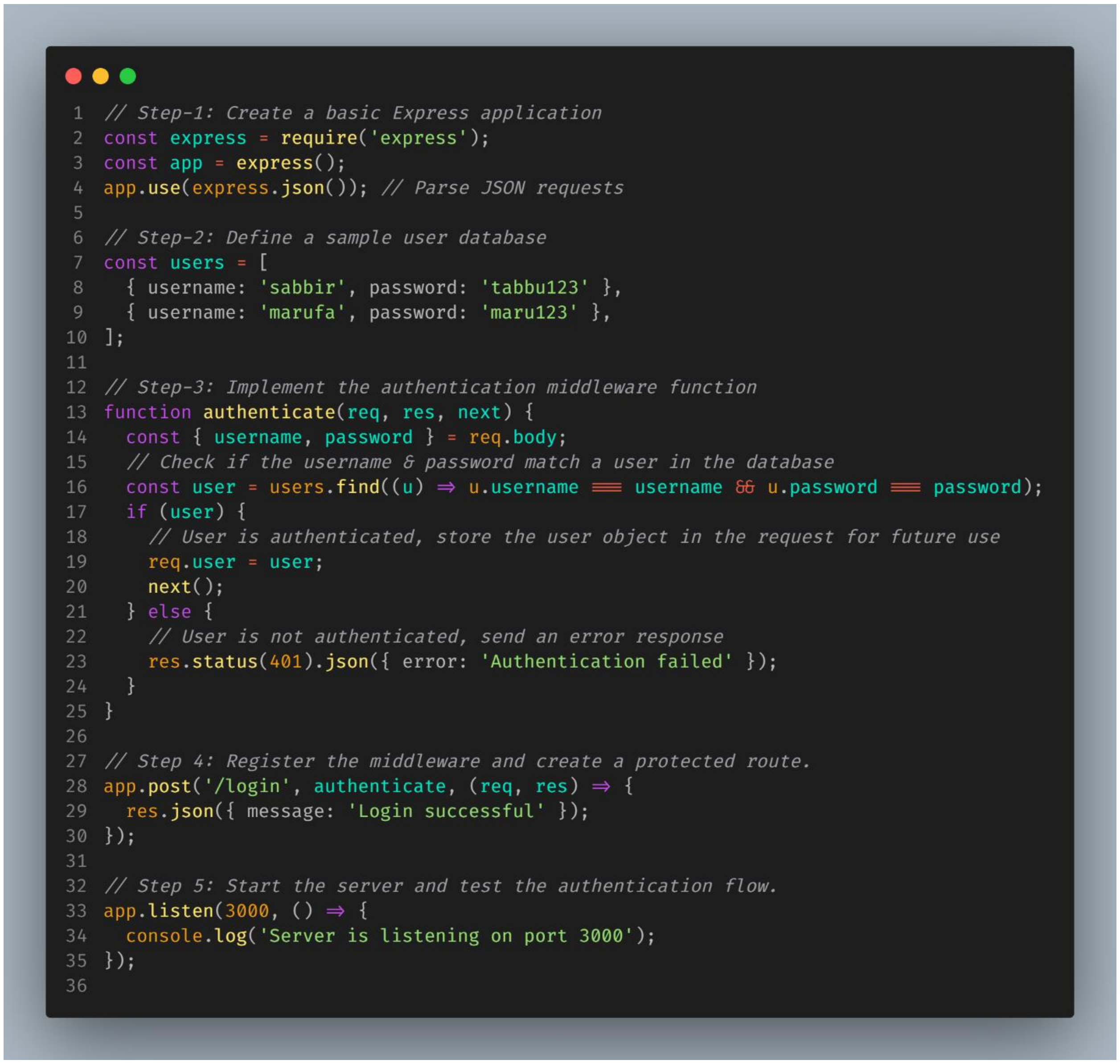
- Data Exchange
- API Responses
- Configuration File Management
- AJAX Requests

## 4. What is a middleware in web development, and give an example of how it can be used.

**Answer:** Middleware in web development is software that sits between the client and the server. It is used to handle tasks such as authentication, authorization, and routing. Middleware can be used to improve the security, performance, and scalability of web applications.

- **Step-1:** We create a basic Express application.
- **Step-2:** We define a sample user database or authentication mechanism. In this example, we use a hardcoded array of user objects for simplicity.
- **Step-3:** We implement the authentication middleware function authenticate. It takes three arguments: request object, response object, and next to pass control to the next middleware or route handler. Inside the authenticate function, we extract the username and password from the request body (req.body). We check if the provided username and password match a user in the local database. If a match is found, we store the user object in the request (req.user) for future use and call next() to proceed to the next middleware or route handler. If the provided credentials don't match, we send an error response with a status code of 401 (Unauthorized).
- **Step-4:** We register the authentication middleware using app.use() to apply it to all routes. We create a /login route with the authenticate middleware before the route handler. This route handles the login request and checks the user's credentials

- **Step-5:** We start the server and test the authentication flow. When the server is running, you can make requests to the /login route with valid credentials to receive a JSON response saying "Login successful". If authentication fails, you will receive a 401 error response.



```javascript
// Step-1: Create a basic Express application
const express = require('express');
const app = express();
app.use(express.json()); // Parse JSON requests

// Step-2: Define a sample user database
const users = [
  { username: 'sabbir', password: 'tabbu123' },
  { username: 'marufa', password: 'maru123' },
];

// Step-3: Implement the authentication middleware function
function authenticate(req, res, next) {
  const { username, password } = req.body;
  // Check if the username & password match a user in the database
  const user = users.find((u) => u.username === username && u.password === password);
  if (user) {
    // User is authenticated, store the user object in the request for future use
    req.user = user;
    next();
  } else {
    // User is not authenticated, send an error response
    res.status(401).json({ error: 'Authentication failed' });
  }
}

// Step 4: Register the middleware and create a protected route.
app.post('/login', authenticate, (req, res) => {
  res.json({ message: 'Login successful' });
});

// Step 5: Start the server and test the authentication flow.
app.listen(3000, () => {
  console.log('Server is listening on port 3000');
});

```

5. What is a controller in web development, and what is its role in the MVC architecture?

**Answer:** In web development, a controller is a part of the Model-View-Controller (MVC) architectural pattern. The controller receives user input from the view and passes it to the model. The model then processes the input and returns data to the controller. The controller then selects the view to render based on the data returned by the model.

***Here are some of the role in MVC architecture:***

- **Separation of concerns:** The MVC architecture separates the concerns of the application into three distinct parts: model, view, and controller. This makes the application easier to understand, maintain, and extend.
- **Scalability:** The MVC architecture is scalable. As the application grows, the model, view, and controller can be easily scaled to meet the increased demand.
- **Security:** The MVC architecture is secure. The model, view, and controller are isolated from each other, which makes it more difficult for attackers to exploit security vulnerabilities.