Lab – 02

	Course Name: Advanced Web Technology	**EXPERIMENT NO. 2**	
	Course Code: 20CP314P **Faculty: Komal Singh**	**Branch: CSE**	**Semester: VI**
Submitted by: Patel Vraj ChetanKumar **Roll no:** 21BCP362			

Experiment Implement with routing methods (GET and POST) and also implement one built-in middleware, configurable middleware and one third party middleware (Cookie-Parser).

Objective: Building a Web Server with Express.js

In this practical, I have developed an online survey application leveraging Express.js, a versatile web framework for Node.js. my application efficiently handles HTTP GET and POST requests, serving a user-friendly survey form and processing submitted responses.

Key features

- include the implementation of middleware for form data parsing,
- rate-limiting to prevent duplicate submissions,
- and cookie management to track user participation.

I focused on essential web development concepts, including server setup, request routing, middleware integration, and basic front-end design. The application showcases server-side processing and client-server interaction, providing a solid foundation for building more sophisticated web applications. This practical exercise underscores the effectiveness of Express.js in creating dynamic, interactive web projects with a focus on simplicity and functionality.

index.html file

```
<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Survey Form</title> <style> body { background-color: #f3f3f3; font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif; } .container { width: 50%; margin: 0 auto; background-color: #FFFFFF; padding: 2em; box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); margin-top: 5em; border-radius: 8px; } h1 { color: #333333; text-align: center; } form { display: flex; flex-direction: column; } label { margin-top: 1em; color: #555555; } input[type="text"], input[type="submit"] { padding: 0.5em; margin-top: 0.5em; border-radius: 4px; border: 2px solid #E0E0E0; outline: none; } input[type="text"] { background-color: #FAFAFA; } input[type="submit"] { background-color: #3778FF; color: white; font-weight: bold; cursor: pointer; border: none; } input[type="submit"]:hover { background-color: #2C5CC5; } </style> </head> <body> <div class="container"> <h1>Online Survey</h1> <form action="/submit-survey" method="post"> <label for="favoriteColor">What is your favorite color?</label> <input type="text" id="favoriteColor" name="favoriteColor" required> <label for="favoriteFood">What is your favorite food?</label> <input type="text" id="favoriteFood" name="favoriteFood" required> <input type="submit" value="Submit Survey"> </form> </div> </body> </html>
```

server.js

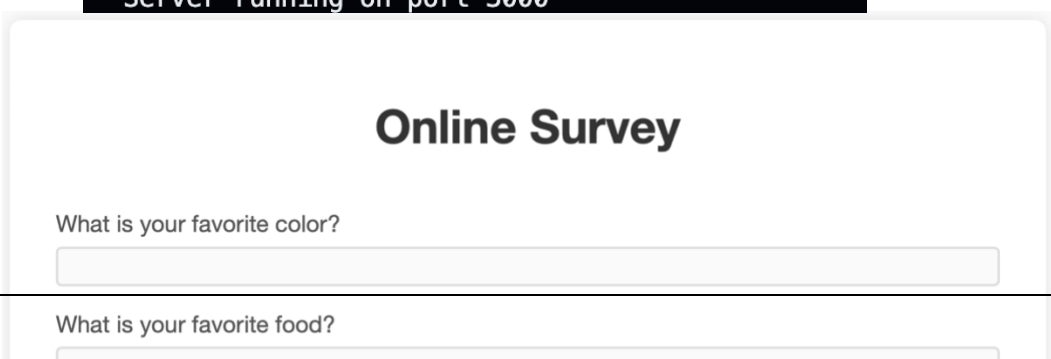
```
const express = require('express'); const cookieParser = require('cookie-parser'); const rateLimit = require('express-rate-limit'); const app = express(); const PORT = 3000; app.use(express.urlencoded({ extended: true })); app.use(cookieParser()); const surveyLimiter = rateLimit({ windowMs: 24 * 60 * 60 * 1000, max: 1, message: 'You have already taken this survey.' }); app.get('/', (req, res) => { res.sendFile(__dirname + '/survey.html'); }); app.post('/submit-survey', surveyLimiter, (req, res) => { if (req.cookies.surveyCompleted) { return res.send('You have already completed this survey.'); } console.log('Survey Response:', req.body); res.cookie('surveyCompleted', 'true', { maxAge: 24 * 60 * 60 * 1000 }); res.send('Thank you for completing the survey!'); }); app.listen(PORT, () => { console.log(`Server running on port ${PORT}`); });
```

GITHUB_RAW_CODE

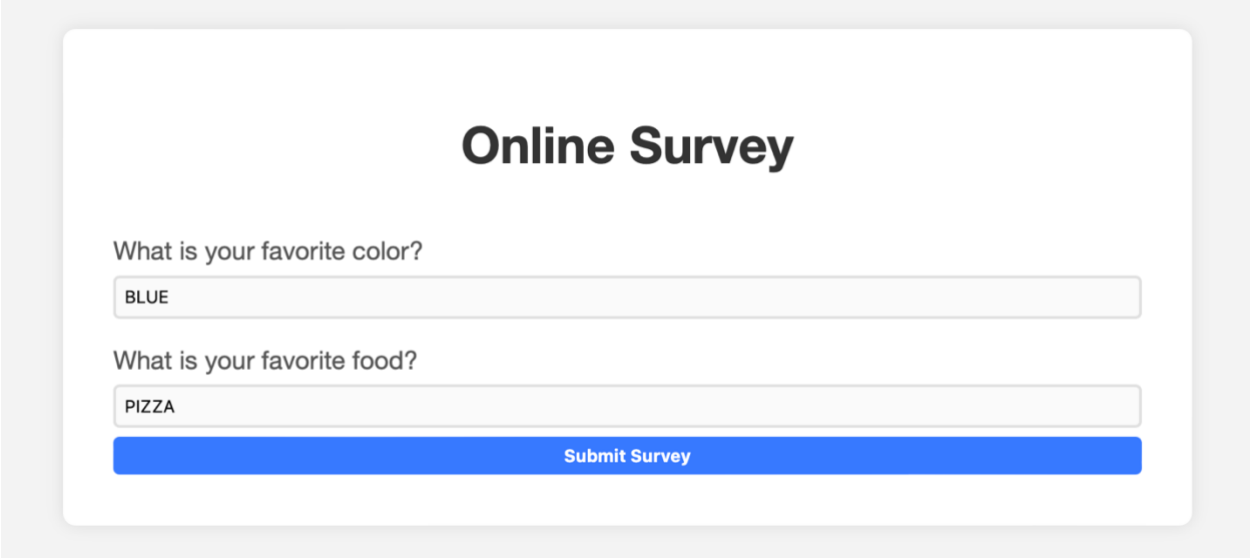
https://github.com/goffycoder/Advance_web_-Technology_Assignments

Starting

```
vrajpatel@192 p2 % nodemon server.js
[nodemon] 3.0.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server running on port 3000
```



Server INPUT Color & Food



OUTPUT

Response In Terminal

```
[nodemon] starting `node server.js`
Server running on port 3000
Survey Response: { favoriteColor: 'BLUE', favoriteFood: 'PIZZA' }
```

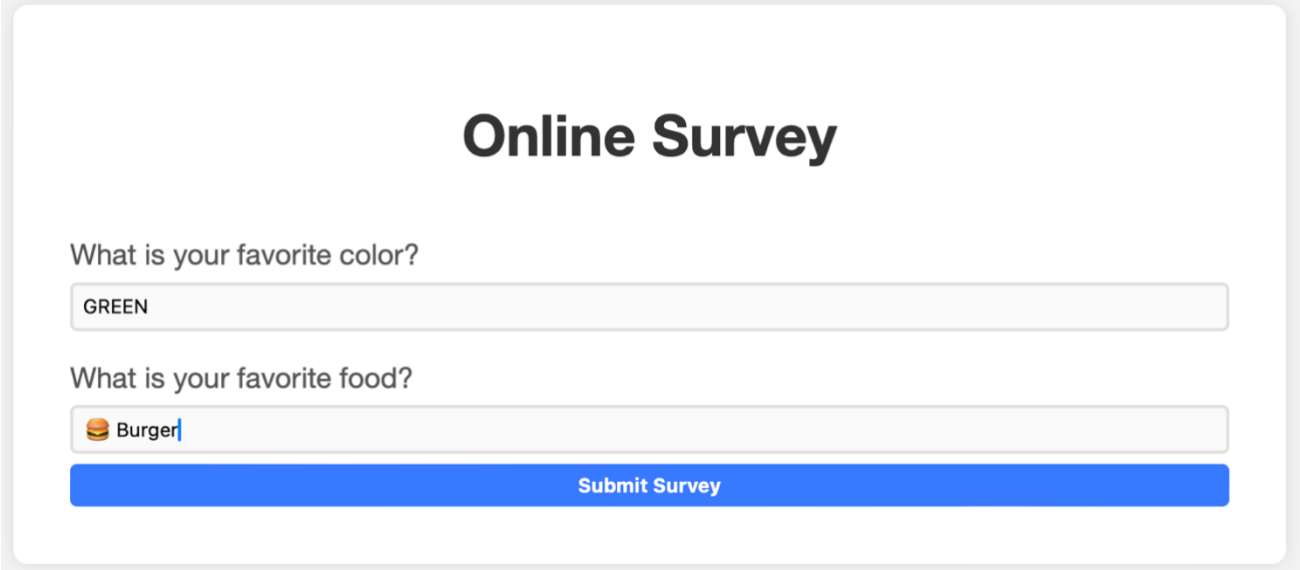
Response on WEB

USE of cookie-parser and express-rate-limit

In our online survey application, I have implemented a mechanism to prevent users from submitting multiple responses by leveraging a combination of middleware and cookies. Specifically, I utilized the **express-rate-limit** middleware to restrict the number of submissions from a single IP address to just one per a 24-hour period. Additionally, I used the **cookie-parser** middleware to set a cookie on the user's browser upon successful survey submission.

This cookie, named **surveyCompleted**, acts as a flag indicating that the user has already participated in the survey. When a user attempts to submit the survey again, the server checks for the presence of this cookie and, if found, prevents another submission by displaying a message that the survey has already been completed. This dual approach ensures that each participant can only submit the survey once, thereby maintaining the integrity of the survey's responses.

Attempting to Resubmit form



Response from our Web app