**Bahria University,**

**Karachi Campus**



**LAB EXPERIMENT NO.**

**13**

**LIST OF TASKS**

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 1 | **Create a simple Express app that sets and retrieves a cookie from the client's browser. You can use the cookie-parser middleware to make it easy to work with cookies.** |
| 2 | **Experiment with different options for setting cookies, such as setting the expiration time and the path for which the cookie is valid.** |
| 3 | **Create an Express app that uses the express-session middleware to store session data on the server. You can create a simple route that sets and retrieves session data and observe how the session data persists across multiple requests.** |
| 4 | **Explore the different options available for storing session data, such as using a memory store, a file store, or a database store.** |
| 5 | **Implement User login/logout feature and handling session and cookies properly with redirecting to login page when no session exist.** |
| 6 | **Try and implement ‘remember me’ feature which allows user to stay logged in by storing a long-lived cookie and validating it at each request** |

**Task # 01: Create a simple Express app that sets and retrieves a cookie from the client's browser. You can use the cookie-parser middleware to make it easy to work with cookies.**

**Solution:**

const express = require('express');

const cookieParser = require('cookie-parser');

const app = express();

app.use(cookieParser());

app.get('/', (req, res) => {

    // Check if the "myCookie" cookie exists

    if (req.cookies.myCookie) {

        res.send(`Cookie value: ${req.cookies.myCookie}`);

    } else {

        res.send('No cookie found');

    }

});

app.get('/setCookie', (req, res) => {

    // Set a "myCookie" cookie with a value of "Hello World"

    res.cookie('myCookie', 'Hello World', { maxAge: 900000, httpOnly: true });

    res.send('Cookie set');

});

app.listen(3000, () => {

    console.log('Server running on port 3000');

});

const express = require('express');

const cookieParser = require('cookie-parser');

const app = express();

app.use(cookieParser());

app.get('/', (req, res) => {

    // Check if the "myCookie" cookie exists

    if (req.cookies.myCookie) {

        res.send(`Cookie value: ${req.cookies.myCookie}`);

    } else {

        res.send('No cookie found');

    }

});

app.get('/setCookie', (req, res) => {

    // Set a "myCookie" cookie with a value of "Hello World"

    res.cookie('myCookie', 'Hello World', { maxAge: 900000, httpOnly: true });

    res.send('Cookie set');

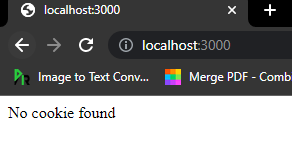
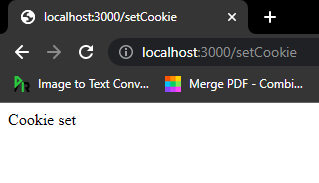
});

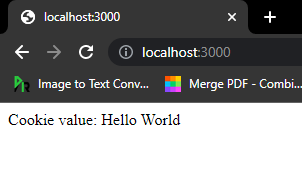
app.listen(3000, () => {

    console.log('Server running on port 3000');

});

**Output:**



**Task # 02: Experiment with different options for setting cookies, such as setting the expiration time and the path for which the cookie is valid.**

**Solution:**

var express = require('express')

var cookieParser = require('cookie-parser')

var app = express()

app.use(cookieParser())

app.get('/', function (req, res) {

  var expires = new Date();

  expires.setSeconds(expires.getSeconds() + 5);

  res.cookie("user", "John Doe", { expires: expires, path: '/members' });

  res.send('Cookie is set');

});

app.listen(3000,()=>{

    console.log(`Listening to port 3000`);

})

var express = require('express')

var cookieParser = require('cookie-parser')

var app = express()

app.use(cookieParser())

app.get('/', function (req, res) {

  var expires = new Date();

  expires.setSeconds(expires.getSeconds() + 5);

  res.cookie("user", "John Doe", { expires: expires, path: '/members' });

  res.send('Cookie is set');

});

app.listen(3000,()=>{

    console.log(`Listening to port 3000`);

})

**Output:**

Graphical user interface, application, chat or text message, website

Description automatically generated

Graphical user interface, application

Description automatically generated

**Task # 03: Create an Express app that uses the express-session middleware to store session data on the server. You can create a simple route that sets and retrieves session data, and observe how the session data persists across multiple requests.**

**Solution:**

var express = require('express');

var session = require('express-session');

var app = express();

app.use(session({

  secret: 'secret-key',

  resave: false,

  saveUninitialized: true

}));

app.get('/set-data', function (req, res) {

  req.session.data = { name: 'Amjad' };

  res.send('Session data set');

});

app.get('/get-data', function (req, res) {

  if (req.session.data) {

    res.json(req.session.data);

  } else {

    res.send('No session data found');

  }

});

app.listen(3000, function () {

  console.log('Example app listening on port 3000!');

});

**Output:**

Graphical user interface, text, application, chat or text message

Description automatically generated

**Task # 04: Explore the different options available for storing session data, such as using a memory store, a file store, or a database store.**

**Solution:**

const express = require('express');

const session = require('express-session');

const app = express();

app.use(session({

  secret: 'secret-key',

  resave: false,

  saveUninitialized: true

}));

app.get('/', (req, res) => {

  if (req.session.views) {

    req.session.views++;

  } else {

    req.session.views = 1;

  }

  res.send(`Viewed ${req.session.views} times`);

});

app.listen(3000, () => {

  console.log('Server started on port 3000');

});

**Output:**

Graphical user interface, text, application, chat or text message

Description automatically generated

**Task # 05: Implement User login/logout feature and handling session and cookies properly with redirecting to login page when no session exist.**

**Solution:**

const express = require('express');

const session = require('express-session');

const cookieParser = require('cookie-parser');

const hbs=require("hbs")

const path=require("path")

const app = express();

const template\_path=path.join(\_\_dirname,"./templates/views")

const partial\_path=path.join(\_\_dirname,"./templates/partials")

app.set("view engine","hbs")

app.set("views",template\_path)

hbs.registerPartials(partial\_path)

app.use(cookieParser());

app.use(session({

  secret: 'your secret key',

  resave: false,

  saveUninitialized: true,

  cookie: { secure: true }

}));

const sessionChecker = (req, res, next) => {

  if (req.session.user) {

    res.redirect('/dashboard');

  } else {

    next();

  }

};

app.get('/login', sessionChecker, (req, res) => {

  res.render('signin');

});

app.post('/login', (req, res) => {

  req.session.user = {username: "Ali"};

  res.redirect('profile');

});

app.get('/logout', (req, res) => {

  req.session.destroy(err => {

    if (err) {

      console.log(err);

    } else {

      res.redirect('/login');

    }

  });

});

app.get('/dashboard', (req, res) => {

  if (req.session.user) {

    res.render('dashboard', {username: req.session.user.username});

  } else {

    res.redirect('/login');

  }

});

app.listen(3000,()=>{

    console.log("Listening to port");

})

**Output:**

Graphical user interface, application, Teams

Description automatically generated

**Task # 06: Try and implement ‘remember me’ feature which allows user to stay logged in by storing a long-lived cookie and validating it at each request**

**Solution:**

const express = require('express');

const session = require('express-session');

const cookieParser = require('cookie-parser');

const moment = require('moment');

const app = express();

app.use(cookieParser());

app.use(session({

  secret: 'your secret key',

  resave: false,

  saveUninitialized: true,

  cookie: { secure: true }

}));

// Middleware to check for an active session

const sessionChecker = (req, res, next) => {

  if (req.session.user) {

    res.redirect('/dashboard');

  } else {

    next();

  }

};

// Login route

app.get('/login', sessionChecker, (req, res) => {

  res.render('login');

});

app.post('/login', (req, res) => {

  // Validate user credentials

  // ...

  // Start a new session

  req.session.user = {username: req.body.username};

  // Check for "remember me" checkbox

  if (req.body.remember) {

    // Set a long-lived cookie with the user's ID

    const expiresAt = moment().add(7, 'days');

    res.cookie('user\_id', req.body.username, { expires: expiresAt.toDate() });

  }

  res.redirect('/dashboard');

});

// Middleware to check for a valid "remember me" cookie

const rememberMeChecker = (req, res, next) => {

  // Check for a "remember me" cookie

  if (!req.session.user && req.cookies.user\_id) {

    // Look up the user by their ID in the cookie

    // ...

    // Start a new session for the user

    req.session.user = {username: req.cookies.user\_id};

  }

  next();

};

app.use(rememberMeChecker);

// Logout route

app.get('/logout', (req, res) => {

  // Clear the "remember me" cookie

  res.clearCookie('user\_id');

  req.session.destroy(err => {

    if (err) {

      console.log(err);

    } else {

      res.redirect('/login');

    }

  });

});

// Dashboard route

app.get('/dashboard', (req, res) => {

  if (req.session.user) {

    res.render('dashboard', {username: req.session.user.username});

  } else {

    res.redirect('/login');

  }

});