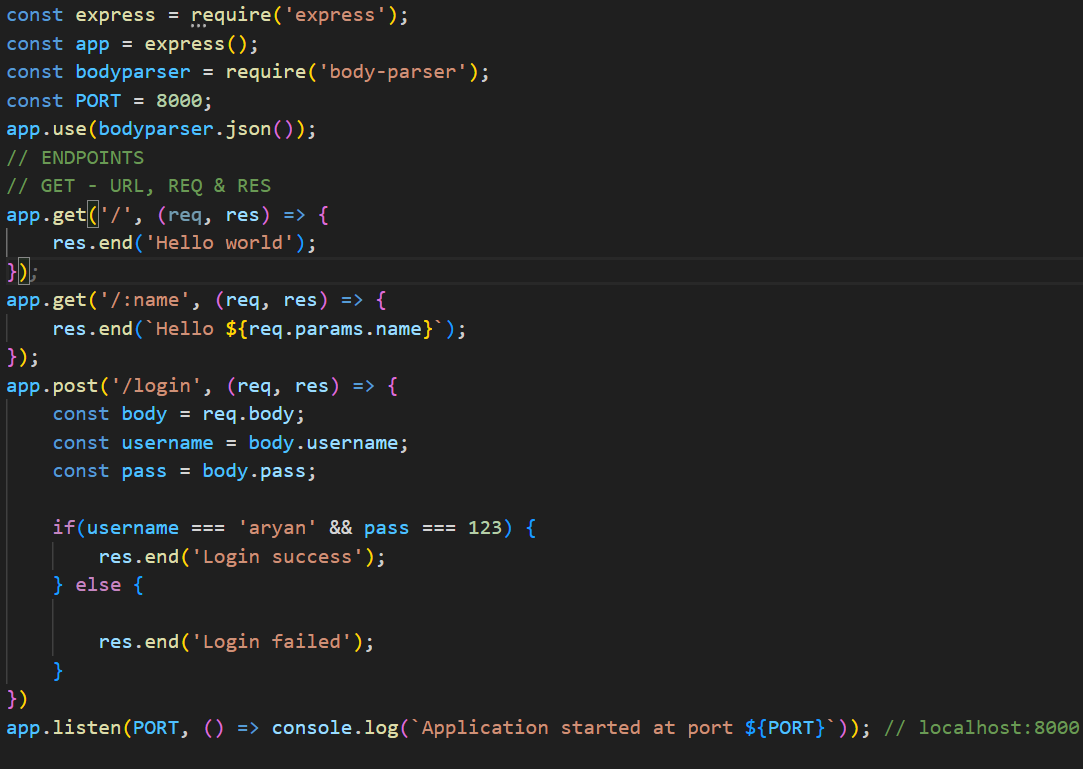
***DAY 9 – API Testing***

API testing for a simple Express.js server using Playwright's testing capabilities.

**1. Express Server (Backend)**

The **backend** is a simple Express.js application with the following components:

* **Server Configuration**: Uses Express.js with body-parser middleware on port 8000



* **API Endpoints**:
  + GET /: Returns "Hello world"
  + GET /:name: Returns a personalized greeting with the provided name parameter
  + POST /login: Accepts username and password credentials and returns success/failure message

**2. Playwright Tests**

****

* **Root Endpoint Test**: Verifies that the root endpoint returns "Hello world"
* **Parameterized Route Test**: Tests the dynamic name endpoint with multiple values (currently skipped)
* **Login API Test**: Validates the authentication logic of the login endpoint

**Technical Implementation Details**

**Express Server Features**

* **Port Configuration**: Runs on port 8000
* **Middleware**: Uses **body-parser** for JSON parsing
* **Dynamic Routes**: Demonstrates parameterized routes with :name
* **POST Handler**: Processes **JSON body data** for authentication
* **Simple Authentication**: Validates credentials against hardcoded values

**Playwright Testing Approach**

* **HTTP Request Testing**: Direct API calls without browser launching
* **Response Validation**: Checking status codes and response bodies
* **Data Parameterization**: Testing multiple inputs for the same endpoint
* **Test Control**: Demonstrates test skipping functionality
* **Assertion Patterns**: Uses Playwright's expect API for validation
* Install dependencies: **npm install express body-parser @playwright/test typescript**
* Start the server: node index.js
* Run tests: npx playwright test

This API testing implementation showcases how Playwright can be used beyond browser testing to validate API functionality directly, making it a versatile tool for full-stack testing.