const request = require('supertest');

const { app } = require('./app'); // Adjust path if necessary

const { Pool } = require('pg');

describe('API Routes', function () {

  let server;

  let pool;

  beforeAll(async function () {

    pool = new Pool({

      user: process.env.PG\_TEST\_USER,

      host: process.env.PG\_HOST,

      database: process.env.PG\_TEST\_DATABASE,

      password: process.env.PG\_TEST\_PASSWORD,

      port: process.env.PG\_PORT, // Make sure this matches your test database port

    });

    // Create tables if they don't exist

    await pool.query(`

      CREATE TABLE IF NOT EXISTS users (

        user\_id SERIAL PRIMARY KEY,

        email VARCHAR(255) UNIQUE NOT NULL,

        password\_hash TEXT NOT NULL

      );

      CREATE TABLE IF NOT EXISTS passwords (

        password\_id SERIAL PRIMARY KEY,

        user\_id INTEGER REFERENCES users(user\_id),

        service\_name VARCHAR(255) NOT NULL,

        password\_encrypted TEXT NOT NULL,

        encrypted\_data\_key TEXT

      );

      CREATE TABLE IF NOT EXISTS change\_logs (

        log\_id SERIAL PRIMARY KEY,

        user\_id INTEGER REFERENCES users(user\_id),

        password\_id INTEGER REFERENCES passwords(password\_id),

        description TEXT,

        timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

      );

    `);

    // Start the server and store the instance

    server = app.listen(3001);

  });

  afterAll(async function () {

    // Properly close database connections

    await pool.query('DROP TABLE IF EXISTS change\_logs CASCADE');

    await pool.query('DROP TABLE IF EXISTS passwords CASCADE');

    await pool.query('DROP TABLE IF EXISTS users CASCADE');

    // Close the pg pool

    await pool.end();

    // Close the server properly

    await new Promise((resolve, reject) => {

      server.close((err) => {

        if (err) return reject(err);

        resolve();

      });

    });

  });

  beforeEach(async function () {

    // Clean the users table before each test to avoid conflicts

    await pool.query('DELETE FROM users');

  });

  it('should register a user successfully', async function () {

    // Use a unique email in this test

    const uniqueEmail = `unique${Date.now()}@example.com`;

    const res = await request(server)

      .post('/api/registration')

      .send({ email: uniqueEmail, password: 'newpassword' });

    console.log(res.body); // Log response body for debugging

    expect(res.status).toBe(200);

    expect(res.body).toHaveProperty('message', 'User registered successfully');

  });

  it('should return error when email or password is missing during registration', async function () {

    const res = await request(server)

      .post('/api/registration')

      .send({ email: 'missingpassword@example.com' });

    expect(res.status).toBe(400);

    expect(res.body).toHaveProperty('error', 'Email and password are required');

  });

  it('should log in a user successfully', async function () {

    const loginEmail = `testlogin${Date.now()}@example.com`;

    // Register the user before login

    await request(server)

      .post('/api/registration')

      .send({ email: loginEmail, password: 'password123' });

    const res = await request(server)

      .post('/api/login')

      .send({ email: loginEmail, password: 'password123' });

    console.log(res.body); // Log response body for debugging

    expect(res.status).toBe(200);

    expect(res.body).toHaveProperty('message', 'User logged in successfully');

  });

});

Server Unit Tests:

> server@1.0.0 test

> jest

console.log

Registration request received

at log (app.js:80:11)

console.log

SUCCESS: User registered successfully.

at log (app.js:98:13)

console.log

{ message: 'User registered successfully', user\_id: 27 }

at Object.log (app.test.js:76:13)

console.log

Registration request received

at log (app.js:80:11)

console.log

Error: Email and password are required

at log (app.js:84:13)

console.log

Registration request received

at log (app.js:80:11)

console.log

SUCCESS: User registered successfully.

at log (app.js:98:13)

console.log

Login request received

at log (app.js:110:11)

console.log

Database query result: [

{

user\_id: 28,

password\_hash: '$2a$10$fyj5WjZIMr.qyXouRm0nbufZw0iJvcUC2KdlWGX7mcAMjcPmpjYqy'

}

]

at log (app.js:119:13)

console.log

Existance of user with inputted email is confirmed.

at log (app.js:124:13)

console.log

SUCCESS: User logged in successfully

at log (app.js:126:15)

console.log

SUCCESS: User logged in successfully

at log (app.js:147:15)

console.log

{ message: 'User logged in successfully' }

at Object.log (app.test.js:102:13)

PASS ./app.test.js

API Routes

√ should register a user successfully (303 ms)

√ should return error when email or password is missing during registration (21 ms)

√ should log in a user successfully (266 ms)

Test Suites: 1 passed, 1 total

Tests: 3 passed, 3 total

Snapshots: 0 total

Time: 3.322 s

Ran all test suites.

Jest did not exit one second after the test run has completed.

'This usually means that there are asynchronous operations that weren't stopped in your tests. Consider running Jest with `--detectOpenHandles` to troubleshoot this issue.