PaperCoin

Testconcept

Background

The management and storage of the portfolios at PaperCoin is based on MongoDB. After registration, users can log in and manage their portfolio. This part is the most sensitive part of the whole project, because it is about user data and the portfolios. For this reason, we decided to test the most security relevant part for full functionality.

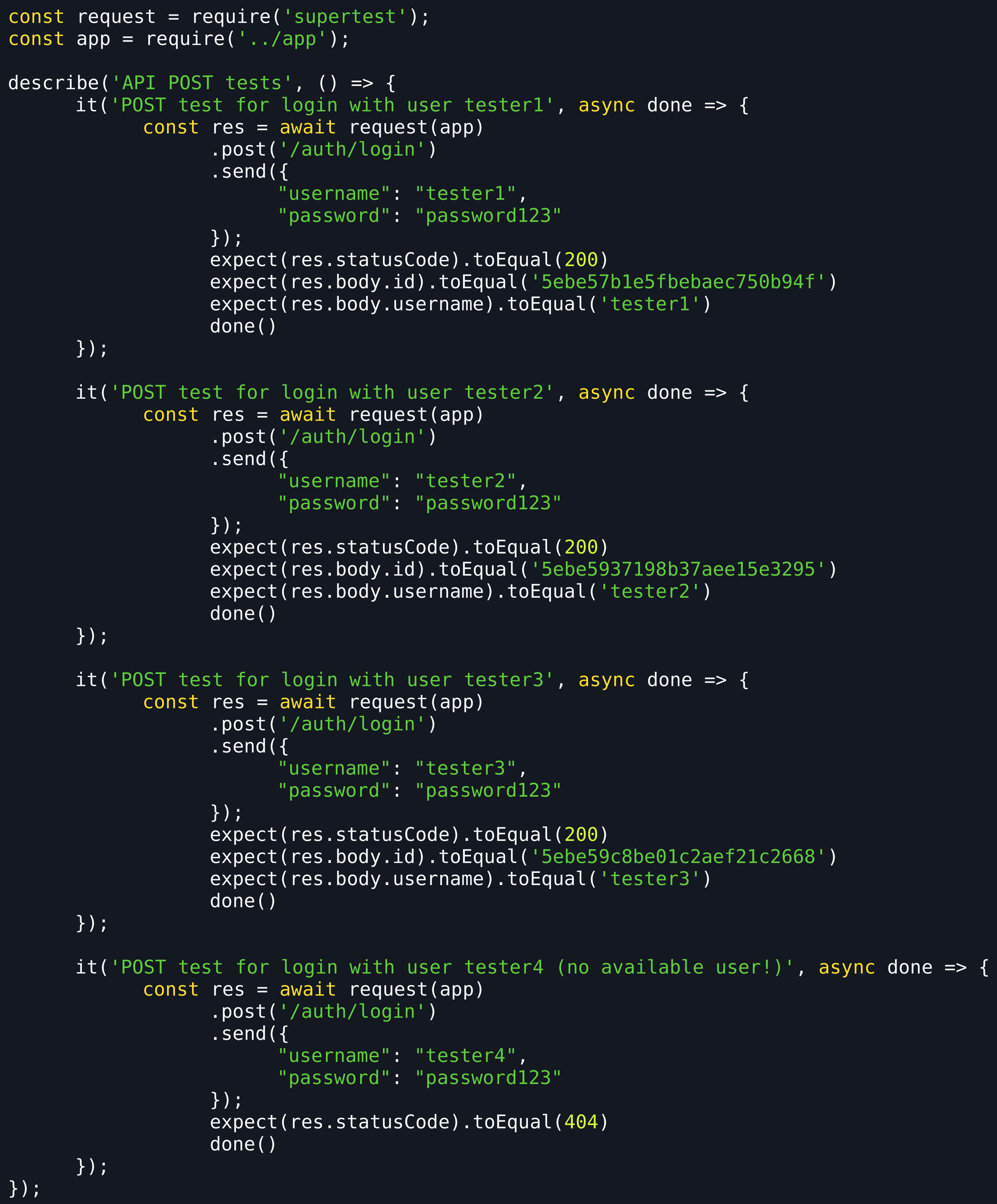
What is tested?

The entire registration process as well as the login of an existing user account will be tested by Jest.

On the following pages the tests written for this purpose are attached.

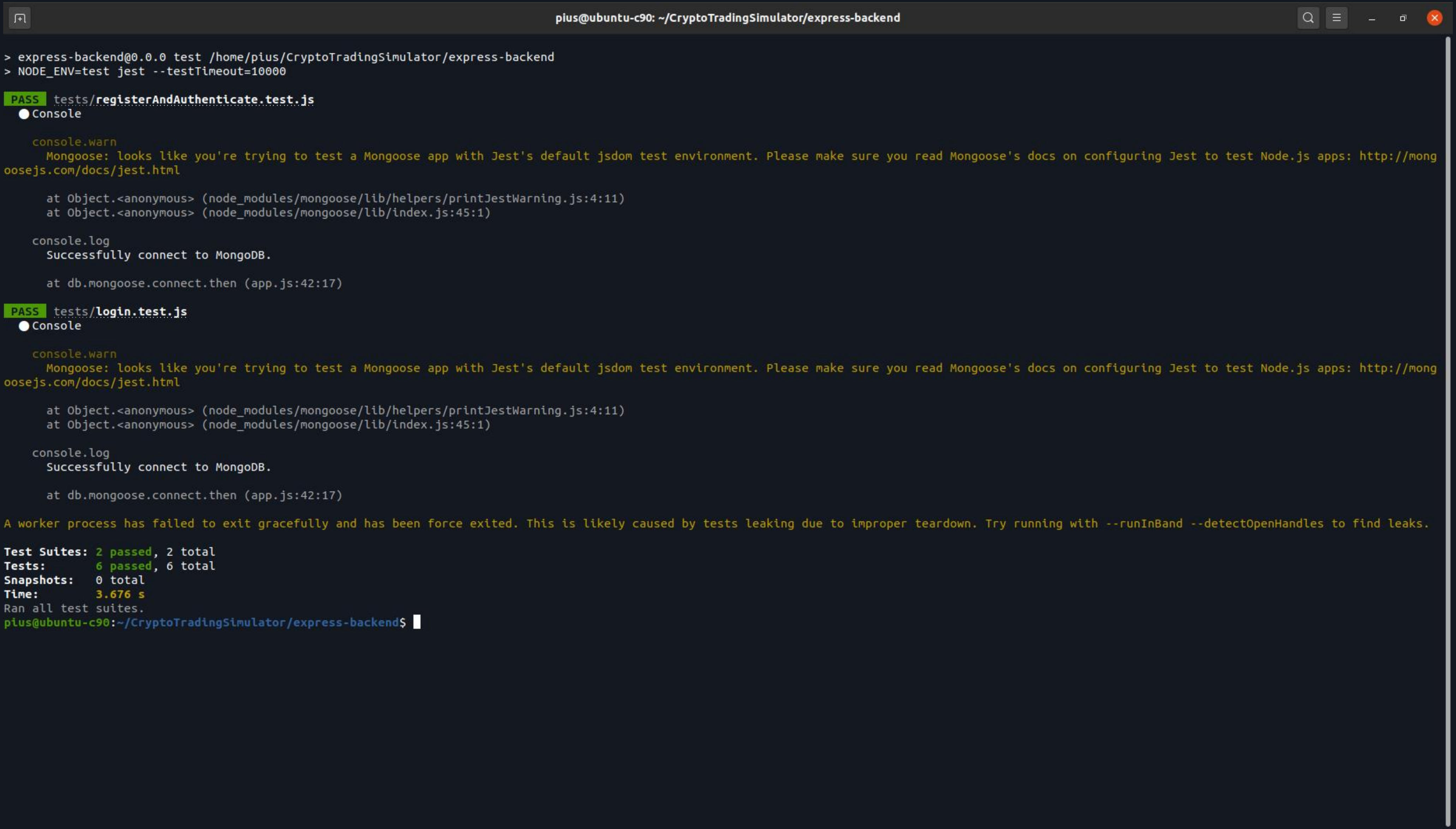
1. registerAndAuthenticate.test.js
2. login.test.js

Finally, the test run was attached as a screenshot.



```
const request = require('supertest');
const app = require('../app');

describe('API POST tests', () => {
      it('POST test for login with user tester1', async done => {
            const res = await request(app)
                  .post('/auth/login')
                  .send({
                        "username": "tester1",
                        "password": "password123"
                  });
                  expect(res.statusCode).toEqual(200)
                  expect(res.body.id).toEqual('5ebe57b1e5fbebaec750b94f')
                  expect(res.body.username).toEqual('tester1')
                  done()
      });

      it('POST test for login with user tester2', async done => {
            const res = await request(app)
                  .post('/auth/login')
                  .send({
                        "username": "tester2",
                        "password": "password123"
                  });
                  expect(res.statusCode).toEqual(200)
                  expect(res.body.id).toEqual('5ebe5937198b37aee15e3295')
                  expect(res.body.username).toEqual('tester2')
                  done()
      });

      it('POST test for login with user tester3', async done => {
            const res = await request(app)
                  .post('/auth/login')
                  .send({
                        "username": "tester3",
                        "password": "password123"
                  });
                  expect(res.statusCode).toEqual(200)
                  expect(res.body.id).toEqual('5ebe59c8be01c2aef21c2668')
                  expect(res.body.username).toEqual('tester3')
                  done()
      });

      it('POST test for login with user tester4 (no available user!)', async done => {
            const res = await request(app)
                  .post('/auth/login')
                  .send({
                        "username": "tester4",
                        "password": "password123"
                  });
                  expect(res.statusCode).toEqual(404)
                  done()
      });
});
```

```
const request = require('supertest');
const app = require('../app');
const controller = require("../app/controllers/auth.controller");

describe('API POST tests', () => {
      it('POST test for registering registrar1', async done => {
            const res = await request(app)
                  .post('/auth/register')
                  .send({
                        "username": "registrar1",
                        "email": "registrar1@gmail.com",
                        "password": "passwordregister"
                  });
            expect(res.statusCode).toEqual(200);
            expect(res.body.message).toEqual('User was registered successfully!');
            done()
      });

      it('POST test for login with user registrar1', async done => {
            const res = await request(app)
                  .post('/auth/login')
                  .send({
                        "username": "registrar1",
                        "password": "passwordregister"
                  });
            accessToken = res.body.accessToken;
            expect(res.statusCode).toEqual(200);
            expect(res.body.username).toEqual('registrar1');
            expect(res.body.email).toEqual('registrar1@gmail.com');
            expect(res.body.accessToken).toBeDefined();
            done()
      });
});

afterAll(() => {
      controller.deleteTestUser({"username": "registrar1"}, {"message": "User
content."})
});
```



```
> express-backend@0.0.0 test /home/pius/CryptoTradingSimulator/express-backend
> NODE_ENV=test jest --testTimeout=10000

 PASS  tests/registerAndAuthenticate.test.js
  ● Console

    console.warn
      Mongoose: looks like you're trying to test a Mongoose app with Jest's default jsdom test environment. Please make sure you read Mongoose's docs on configuring Jest to test Node.js apps: http://mong
oosejs.com/docs/jest.html

      at Object.<anonymous> (node_modules/mongoose/lib/helpers/printJestWarning.js:4:11)
      at Object.<anonymous> (node_modules/mongoose/lib/index.js:45:1)

    console.log
      Successfully connect to MongoDB.

      at db.mongoose.connect.then (app.js:42:17)

 PASS  tests/login.test.js
  ● Console

    console.warn
      Mongoose: looks like you're trying to test a Mongoose app with Jest's default jsdom test environment. Please make sure you read Mongoose's docs on configuring Jest to test Node.js apps: http://mong
oosejs.com/docs/jest.html

      at Object.<anonymous> (node_modules/mongoose/lib/helpers/printJestWarning.js:4:11)
      at Object.<anonymous> (node_modules/mongoose/lib/index.js:45:1)

    console.log
      Successfully connect to MongoDB.

      at db.mongoose.connect.then (app.js:42:17)

A worker process has failed to exit gracefully and has been force exited. This is likely caused by tests leaking due to improper teardown. Try running with --runInBand --detectOpenHandles to find leaks.

Test Suites: 2 passed, 2 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        3.676 s
Ran all test suites.
pius@ubuntu-c90:~/CryptoTradingSimulator/express-backend$ 
```