**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP.HCM**



**BÁO CÁO MÔN HỌC**

**CÔNG CỤ VÀ MÔI TRƯỜNG PHÁT TRIỂN PHẦN MỀM**



**HOTPOST- MẠNG XÃ HỘI**

Ngành**:** **CÔNG NGHỆ THÔNG TIN**

Chuyên ngành: **CÔNG NGHỆ PHẦN MỀM**

Giảng viên hướng dẫn: **NGUYỄN THANH TÙNG**

Lớp: **19DTHC3**

Sinh viên thực hiện:

|  |  |
| --- | --- |
| **Trần Chấn Hưng** | **1911065247** |
| **Lê Hoàng Minh Tuấn** | **1911065880** |
| **Ngô Nguyễn Ngọc Thành** | **1911160680** |

Thành phố Hồ Chí Minh, năm 2023

# LỜI CẢM ƠN

Trước tiên với tình cảm sâu sắc và chân thành nhất, cho phép chúng em được bày tỏ lòng biết ơn đến tất cả thầy cô và nhà trường đã tạo điều kiện hỗ trợ, giúp đỡ chúng em trong suốt quá trình học tập và nghiên cứu đề tài này. Trong suốt thời gian từ khi bắt đầu học tập tại trường đến nay, chúng em đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý Thầy Cô và bạn bè.

Với lòng biết ơn sâu sắc nhất, chúng em xin gửi đến quý Thầy Cô ở khoa Công Nghệ Thông Tin đã truyền đạt vốn kiến thức quý báu cho chúng em trong suốt thời gian học tập tại trường. Nhờ có những lời hướng dẫn, dạy bảo của các thầy cô nên đề tài nghiên cứu của chúng em mới có thể hoàn thiện tốt đẹp.

Một lần nữa, chúng em xin chân thành cảm ơn thầy **Nguyễn Thanh Tùng** người đã trực tiếp giúp đỡ, quan tâm, hướng dẫn chúng em hoàn thành tốt bài báo cáo này trong thời gian qua.

Bài báo cáo thực hiện trong khoảng thời gian không nhiều bước đầu đi vào thực tế của chúng em còn hạn chế và còn nhiều bỡ ngỡ nên không tránh khỏi những thiếu sót. Chúng em rất mong nhận được những ý kiến đóng góp quý báu của quý Thầy Cô để kiến thức của chúng em trong lĩnh vực này được hoàn thiện hơn đồng thời có điều kiện bổ sung, nâng cao kiến thức của mình.

Chúng em xin chân thành cảm ơn!

Sinh viên thực hiện

**Trần Chấn Hưng**

**Ngô Nguyễn Ngọc Thành**

**Lê Hoàng Minh Tuấn**

# LỜI MỞ ĐẦU

Trong thời đại hiện nay, mạng xã hội đã trở thành một phần không thể thiếu trong cuộc sống của chúng ta. Những nền tảng mạng xã hội như Facebook, Instagram, Twitter, LinkedIn... đã thay đổi cách chúng ta kết nối, giao tiếp và chia sẻ thông tin. Tuy nhiên, với sự phát triển của công nghệ thông tin, chúng ta cần những nền tảng mạng xã hội mới, đáp ứng được nhu cầu và mong muốn của người dùng.

Với sự phát triển của công nghệ web, việc xây dựng một mạng xã hội trên nền tảng web đã trở nên dễ dàng hơn bao giờ hết. Một mạng xã hội trên web không chỉ mang lại cho người dùng sự kết nối với bạn bè, mà còn mang lại cho họ nhiều tính năng khác như tìm kiếm việc làm, mua sắm trực tuyến, kết nối với doanh nghiệp và cộng đồng.

Với đề tài xây dựng một mạng xã hội trên web, chúng ta có cơ hội thử thách bản thân trong việc tạo ra một sản phẩm đáp ứng được nhu cầu của người dùng, đồng thời cũng học hỏi được nhiều kỹ năng và kiến thức mới trong lĩnh vực công nghệ thông tin. Vì vậy, việc xây dựng một mạng xã hội trên web là một thử thách đầy hứa hẹn và hứa hẹn mang lại nhiều giá trị cho người dùng.

Xuất phát từ những lợi ích trên cùng với sự cho phép và tận tình giúp đỡ của Thầy **Nguyễn Thanh Tùng** nên em xin chọn đề tài “HotPost- Mạng xã hội”. Với đề tài này, chúng em mong muốn áp dụng các kiến thức đã được học trong trường cùng với việc tìm hiểu nghiên cứu nhiều ngôn ngữ giúp ích cho đề tài trở nên hoàn thiện.

Sinh viên thực hiện

**Trần Chấn Hưng**

**Ngô Nguyễn Ngọc Thành**

**Lê Hoàng Minh Tuấn**

# MỤC LỤC

[LỜI CẢM ƠN i](#_Toc131552850)

[LỜI MỞ ĐẦU ii](#_Toc131552851)

[MỤC LỤC iii](#_Toc131552852)

[CHƯƠNG 1: Đặc tả User Story và Activity 1](#_Toc131552853)

[1. Đặc tả user story 1](#_Toc131552854)

[2. Activity 2](#_Toc131552855)

[CHƯƠNG 2: Unit test 5](#_Toc131552856)

[1. Thư viện sử dụng 5](#_Toc131552857)

[2. Test case cho phần auth Login và Register 5](#_Toc131552858)

[Phân Công 11](#_Toc131552859)

# 

# CHƯƠNG 1: Đặc tả User Story và Activity

## Đặc tả user story

User Story 1 : Là một user tôi muốn đăng nhập bằng gmail và password

* What: Đăng nhập bằng email và password đã đăng ký
* Who: Người dùng
* Why: Đơn giản và có được tính bảo mật

User Story 2 : Là một user tôi có thể đăng ký tài khoản

* What: Đăng ký bằng email và password và các thông tin cần thiết như số điện thoại ngày tháng năm sinh, hình ảnh avatar
* Who: Người dùng
* Why: Đơn giản và có thể lưu trữ user một cách toàn vẹn vê mặt dữ liệu

User Story 3 : Là một user tôi có thể thấy được các bài đăng của các người khác

* What: Thấy được nội dung thông tin và các bài đăng của người khác
* Who: Người dùng
* Why: Đơn giản và có sự thích thú khi được cập nhật các bài đăng mới

User Story 4 : Là một user tôi có thể nhấn vào nút like và nút comment và thực hiện thao tác đó trên bài đăng của người khác

* What: Có thể nhấn vào nút like và comment để thể hiện sự tuong tác
* Who: Người dùng
* Why: Đơn giản và tạo thích thú và tạo tương tác với các người chơi khác

User Story 5 : Là một user tôi có thể xem trang cá nhân của tôi bào gồm thông tin và các bài đăng của tôi

* What: Xem được thông tin của mình ở trong trang cá nhân của mình
* Who: Người dùng
* Why: Để hiển thị những gì minh đã làm và sử dụng trên mạng xã hội

User Story 6 : Là một user tôi cũng có thể truy cập tới trang cá nhân của người khác và xem các bài đăng của họ

* What: Có thể truy cập tới một cá nhân nào đó và có thể thấy được các hoạt động của họ
* Who: Người dùng
* Why: Đơn giản và có thể kết nối tới mọi người

User Story 7 : Là một user tôi có thể kết bạn được với những người tôi muốn và hiển thị thông tin bạn bè đã kết bạn

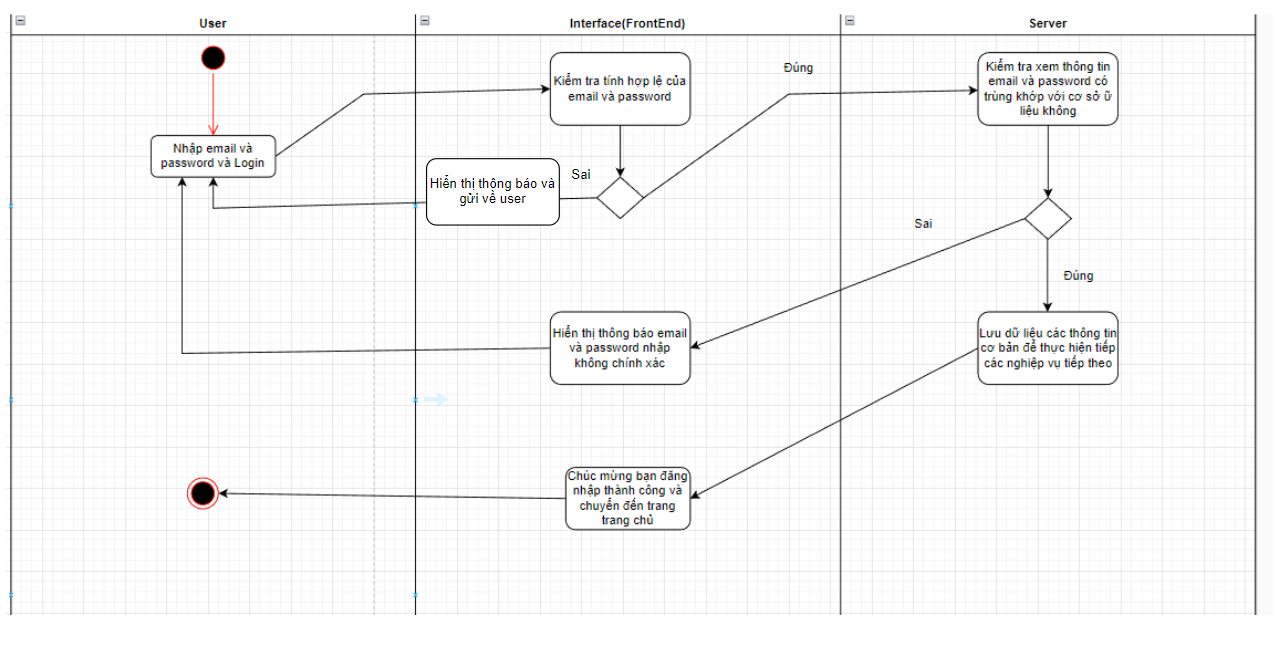
* What: Khi nhấn vào kết bạn sẽ hiển thị ai là người mình đã két bạn
* Who: Người dùng
* Why: Đơn giản và biết chính xác mình có quen biết họ hay không

User Story 8 : Là một người dùng tôi có thể đăng bài với nội dung và kèm hình ảnh

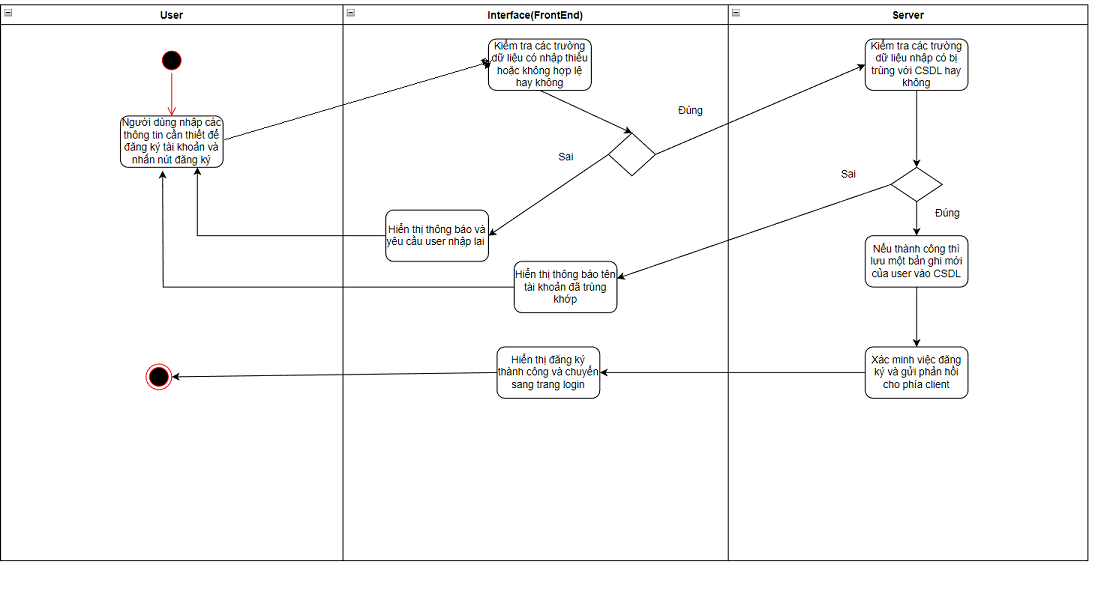
* What: Mỗi người dùng có thể đăng nội dung của mình lên mạng xã hội
* Who: Người dùng
* Why: Đơn giản và có thể tạo sự chú ý cho nhiều người khác

## Activity

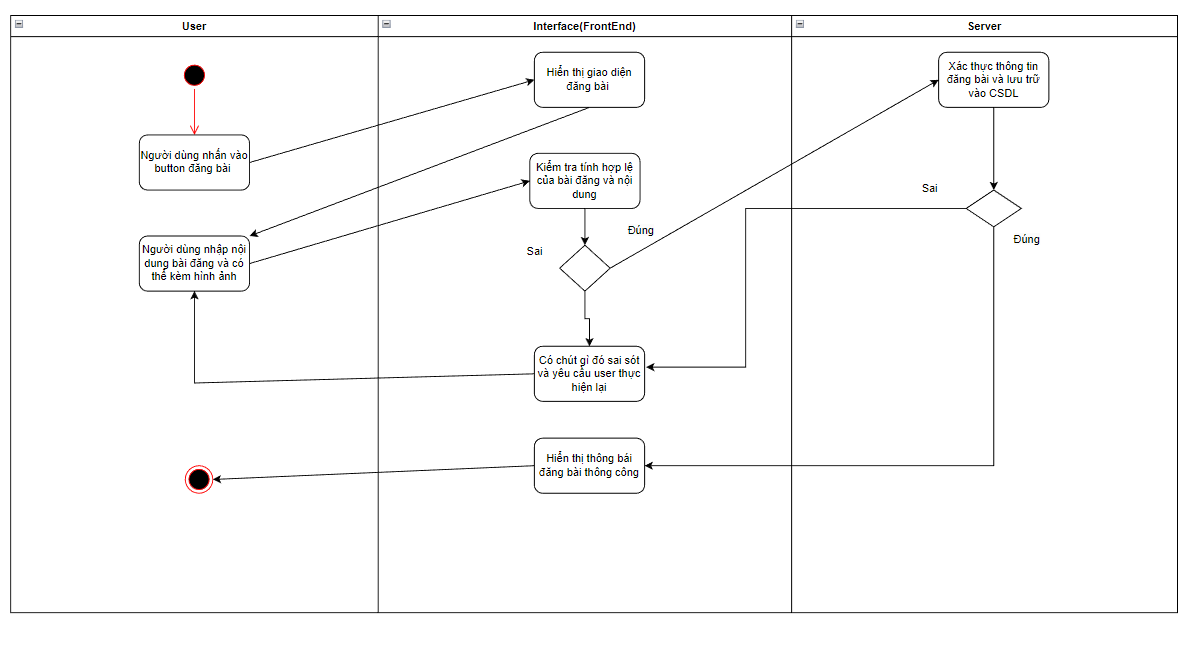
User Story 1:



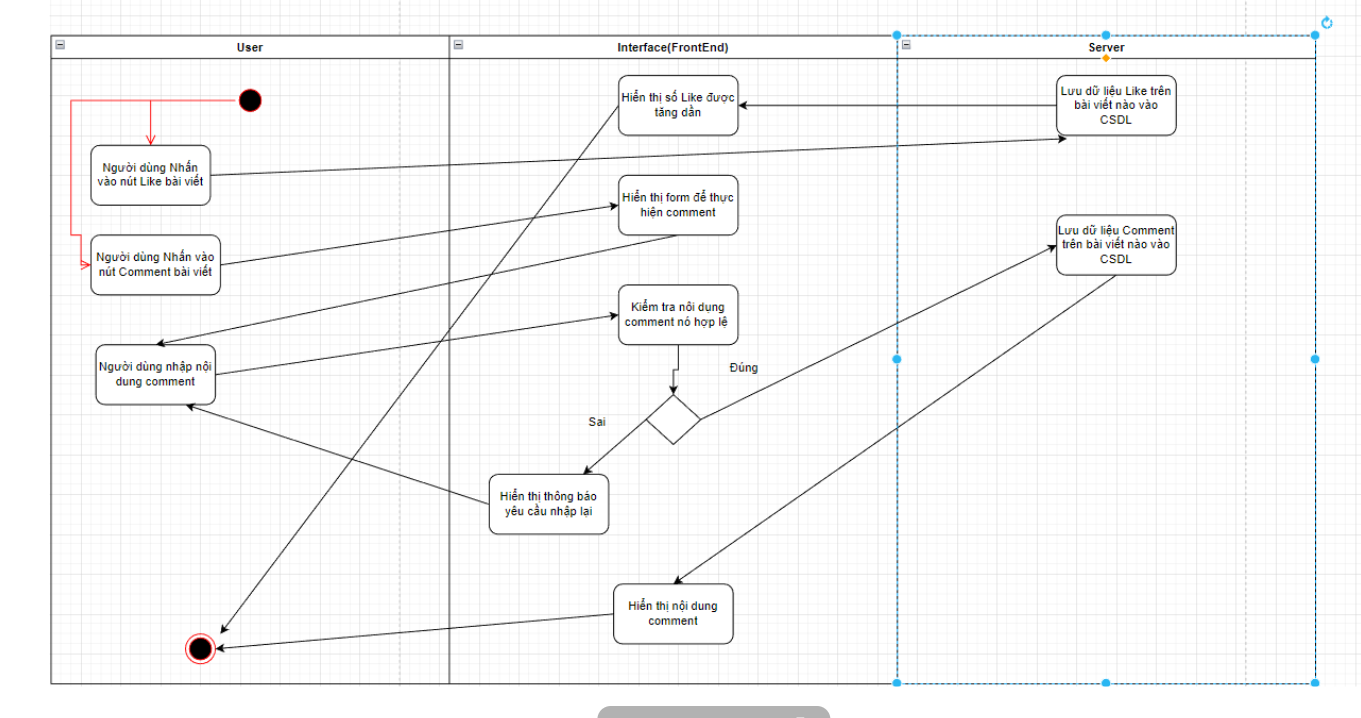
User Story 2:



User Story 3,4:

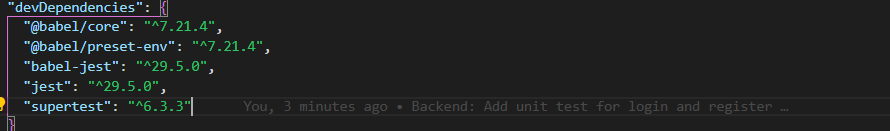


User Story 4,7:



# CHƯƠNG 2: Unit test

## Thư viện sử dụng



## Test case cho phần auth Login và Register

AuthControllers:

import bcrypt from "bcrypt";

import jwt from "jsonwebtoken";

import User from "../models/User.js";

/\* REGISTER USER \*/

export const register = async (req, res) => {

  try {

    const {

      firstName,

      lastName,

      email,

      password,

      picturePath,

      friends,

      location,

      occupation,

    } = req.body;

    const salt = await bcrypt.genSalt();

    const passwordHash = await bcrypt.hash(password, salt);

    const newUser = new User({

      firstName,

      lastName,

      email,

      password: passwordHash,

      picturePath,

      friends,

      location,

      occupation,

      viewedProfile: Math.floor(Math.random() \* 10000),

      impressions: Math.floor(Math.random() \* 10000),

    });

    const savedUser = await newUser.save();

    res.status(201).json(savedUser);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

/\* LOGGING IN \*/

export const login = async (req, res) => {

  try {

    const { email, password } = req.body;

    const user = await User.findOne({ email: email });

    if (!user) return res.status(400).json({ msg: "User does not exist. " });

    const isMatch = await bcrypt.compare(password, user.password);

    if (!isMatch) return res.status(400).json({ msg: "Invalid credentials. " });

    const token = jwt.sign({ id: user.\_id }, process.env.JWT\_SECRET);

    delete user.password;

    res.status(200).json({ token, user });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

Unit Test:

import bcrypt from 'bcrypt'

import jwt from 'jsonwebtoken'

import User from '../models/User.js'

import { register, login } from '../controllers/auth.js'

describe('register function', () => {

  afterEach(() => {

    jest.clearAllMocks()

  })

  it('should create new user if given valid input and return status 201 with savedUser', async () => {

    const req = {

      body: {

        firstName: 'John',

        lastName: 'Doe',

        email: 'johndoe@gmail.com',

        password: 'password123',

        picturePath: 'example.jpg',

        friends: ['friend1'],

        location: 'New York',

        occupation: 'Software Engineer',

      },

    }

    const mockSalt = jest.spyOn(bcrypt, 'genSalt').mockResolvedValue(10)

    const mockHash = jest

      .spyOn(bcrypt, 'hash')

      .mockResolvedValue('hashedPassword')

    const mockSave = jest.spyOn(User.prototype, 'save').mockResolvedValue({

      \_id: 1,

      ...req.body,

      password: 'hashedPassword',

      viewedProfile: expect.any(Number),

      impressions: expect.any(Number),

    })

    const res = {

      status: jest.fn().mockReturnThis(),

      json: jest.fn(),

    }

    await register(req, res)

    expect(mockSalt).toHaveBeenCalled()

    expect(mockHash).toHaveBeenCalledWith(req.body.password, 10)

    expect(mockSave).toHaveBeenCalled()

    expect(res.status).toHaveBeenCalledWith(201)

    expect(res.json).toHaveBeenCalledWith({

      \_id: 1,

      ...req.body,

      password: 'hashedPassword',

      viewedProfile: expect.any(Number),

      impressions: expect.any(Number),

    })

  })

  it('should return status 500 with error message', async () => {

    const req = {

      body: {

        firstName: 'John',

        lastName: 'Doe',

        email: 'johndoe@gmail.com',

        password: 'password123',

      },

    }

    const mockError = new Error('failed to save user')

    const mockSalt = jest.spyOn(bcrypt, 'genSalt').mockRejectedValue(mockError)

    const res = {

      status: jest.fn().mockReturnThis(),

      json: jest.fn(),

    }

    await register(req, res)

    expect(mockSalt).toHaveBeenCalled()

    expect(res.status).toHaveBeenCalledWith(500)

    expect(res.json).toHaveBeenCalledWith({ error: mockError.message })

  })

})

describe('login function', () => {

  afterEach(() => {

    jest.clearAllMocks()

  })

  it('should return status 400 if user does not exist', async () => {

    const req = {

      body: {

        email: 'nonexistentuser@gmail.com',

        password: 'password123',

      },

    }

    const mockFindOne = jest.spyOn(User, 'findOne').mockResolvedValue(null)

    const res = {

      status: jest.fn().mockReturnThis(),

      json: jest.fn(),

    }

    await login(req, res)

    expect(mockFindOne).toHaveBeenCalledWith({ email: req.body.email })

    expect(res.status).toHaveBeenCalledWith(400)

    expect(res.json).toHaveBeenCalledWith({ msg: 'User does not exist. ' })

  })

  it('should return status 400 if given invalid credentials', async () => {

    const req = {

      body: {

        email: 'johndoe@gmail.com',

        password: 'invalidPassword',

      },

    }

    const mockUser = {

      \_id: 1,

      firstName: 'John',

      lastName: 'Doe',

      email: 'johndoe@gmail.com',

      password: 'hashedPassword',

    }

    const mockFindOne = jest.spyOn(User, 'findOne').mockResolvedValue(mockUser)

    const mockCompare = jest.spyOn(bcrypt, 'compare').mockResolvedValue(false)

    const res = {

      status: jest.fn().mockReturnThis(),

      json: jest.fn(),

    }

    await login(req, res)

    expect(mockFindOne).toHaveBeenCalledWith({ email: req.body.email })

    expect(mockCompare).toHaveBeenCalledWith(

      req.body.password,

      mockUser.password

    )

    expect(res.status).toHaveBeenCalledWith(400)

    expect(res.json).toHaveBeenCalledWith({ msg: 'Invalid credentials. ' })

  })

  it('should return status 500 with error message', async () => {

    const req = {

      body: {

        email: 'johndoe@gmail.com',

        password: 'password123',

      },

    }

    const mockError = new Error('failed to find user')

    const mockFindOne = jest.spyOn(User, 'findOne').mockRejectedValue(mockError)

    const res = {

      status: jest.fn().mockReturnThis(),

      json: jest.fn(),

    }

    await login(req, res)

    expect(mockFindOne).toHaveBeenCalledWith({ email: req.body.email })

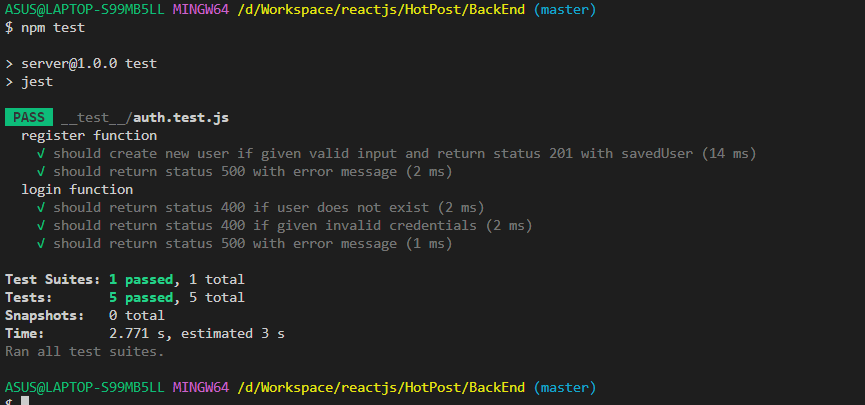
    expect(res.status).toHaveBeenCalledWith(500)

    expect(res.json).toHaveBeenCalledWith({ error: mockError.message })

  })

})

Result:



# Phân Công

|  |  |  |  |
| --- | --- | --- | --- |
| **Bảng phân công** | **User Story** | **Activity** | **Test** |
| Ngọc Thành | **US : 1,2,5** | **1,2** | **Test: register** |
| Chấn Hưng | **US : 4,3,6** | **3,4** | **Test: login** |
| Minh Tuấn | **US : 7,8** | **4,7** | **Test: login** |
|  |  |  |  |