HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

GRADUATION RESEARCH 1 (IT5023E)

Final Report

USER MANAGEMENT WEBSITE

Instructor: **M.S. Nguyen Duc Tien**

Class code: **135469**

Student: **Mai Thi Ngoc Anh - 20205143**

~ Semester 20221 ~

CONTENTS

Acknowledgements

Graduation Research 1 specializing in information technology with the topic of building an e-commerce website is the result of my relentless effort and the dedicated guidance of Mr. Nguyen Duc Tien, as well as my colleagues and friends. Through this, I would like to sincerely express my gratitude to those who have helped me complete this project.

I would like to express my deep respect and gratitude to the teachers who directly guided this project. They provided me with the necessary documents for my project.

Once again, I sincerely thank you!

……

……

……

……

……

……

……

……

……

……

……

Introduction

In the era of industrialization and modernization, society demands a quality workforce with complete skills. Especially in the field of information technology, it requires solid knowledge, sufficient experience, teamwork abilities, and creativity in work. Along with the explosive growth of information technology and websites in recent years, creating, designing, and managing a website is a necessary skill for a programmer, in particular, and IT students in general.

At the hottest moment, everything is just a click away, and the whole world can come close instantly. These are the undeniable benefits that websites bring. Starting from building static web pages with only a few black and white lines of text, websites have evolved to become more visual and lively with sound. As programmers create more functions to meet the "supply-demand" cycle, websites have become even more successful today.

User management applications make it easier and more effective to manage employees of businesses, manage students and students of schools. Along with the development of technology, websites have become part of digital transformation and e-government. Technology is difficult to predict, but it is certain that websites will continue to exist and grow in the years to come.

Website building now has a lot of documentation, but the easiest way to access them may be through the student management websites of universities. Although the topic is familiar, it is an opportunity for me to apply the knowledge I have accumulated from previous studies and this semester. From here, I will equip myself with the necessary skills and familiarize myself with the direction of future projects.

1. **Project title:**

   Research and design of user management website.

2. **Objectives:**

- Research on various technologies, programming languages and frameworks.
- Develop a website for effective user management.

3. **Scope of the project:**

   Scope of the project implementation: Individual.

4. **Research subjects:**

   Technology research orientation:

   - Frontend: html, CSS, JavaScript, bootstrap, redux.
   - Backend: Migrations, database management systems MYSQL, PGSQL.
   - Source code editor: Visual Studio Code.
   - Usage model: MVC.

5. **Overview of chapters in the report:**

   - Chapter 1: Introduction to technology.
   - Chapter 2: Design analysis, application.
   - Chapter 3: Results testing.
   - Chapter 4: Conclusion.
   - Chapter 5: References.

CHAPTER I: LEARNING ABOUT TECHNOLOGY

1.1. Web

1.1.1 What is the web and an overview of its development?

The web (World Wide Web) is a system of internet-based documents that are linked together by hyperlinks and URLs (Uniform Resource Locators). It allows users to access and share information, documents, images, audio, and video on a global network platform. Websites are created using web programming languages such as HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript.

The web was developed by Tim Berners-Lee in the 1980s at CERN (European Organization for Nuclear Research), a scientific research organization in Switzerland. Initially, the web was used as a tool to help CERN scientists access internal documents of the organization. However, with the development of the web, it became a global tool for transmitting information and communication between individuals, organizations, and countries.

The web has gone through several stages of development. The first stage of the web is called Web 1.0, in which websites were created to display static information and often had no interaction with users. The next stage is Web 2.0, in which websites have high interactivity with users, allowing them to participate in content creation and sharing information. Currently, the web is developing towards Web 3.0, a new stage with many applications such as artificial intelligence, blockchain, and virtual reality.

The web has become an important part of modern life and has changed the way we communicate, learn, shop, search for information, and work.

1.1.2 Pure code and using framework

Vanilla code (or pure code) is the act of writing HTML, CSS, and JavaScript code without using any frameworks or libraries. This means you have to write all the code yourself to create the user interface, handle events, and interact with the server.

Using a framework is a way to speed up web application development by using libraries and tools provided by the framework. Some popular frameworks include:

- *React: a JavaScript library used to build user interfaces (UI) for web applications.*
- *Angular: a JavaScript framework used to build high-end dynamic web applications.*
- *Vue: a simple yet powerful JavaScript framework used to build single-page applications (SPAs).*
- *Laravel: a PHP framework used to build complex dynamic web applications.*

Frameworks typically provide features and functions optimized for creating complex web applications. Using a framework also helps with faster application development, high scalability, and easier maintenance. However, using a framework may limit the flexibility and customization of the application and sometimes may make the code harder to understand.

1.2 HTML

HTML stands for Hypertext Markup Language. It is a markup language used to create and design web pages, websites and web applications. HTML provides the basic structure and content of a web page, including text, images, audio, video, links, and other elements that can be displayed in a web browser.

HTML uses markup tags to describe and define the content of a web page, such as headings, paragraphs, lists, tables, and forms. These tags are enclosed in angle brackets, like <tag name>, and can have attributes to specify additional information about the content, such as its size, color, or location on the page.

Web browsers use HTML to display web pages, interpreting the markup tags and rendering the content in a way that can be viewed and interacted with by users. While HTML is the foundation of most web pages, it is often used in conjunction with other technologies such as Cascading Style Sheets (CSS) and JavaScript to create more complex and dynamic web applications.

HTML is a powerful language for creating web pages and web applications. Here are some of the key features and benefits of HTML:

- *Universal Accessibility: HTML is supported by all modern web browsers, making it an ideal choice for creating content that can be accessed by a wide range of users on different devices and platforms.*

- *Structure and Organization: HTML provides a clear and structured way to organize content on a web page, making it easier for users to navigate and understand the information presented.*

- *Compatibility with Other Technologies: HTML can be combined with other technologies like CSS and JavaScript to create dynamic and engaging web experiences.*

- *SEO-Friendly: HTML provides a way to add important information to web pages that search engines use to index and rank them. This means that properly structured HTML can improve a website's visibility and search engine ranking.*

- *Cross-Platform Compatibility: HTML is designed to work across different platforms, devices, and operating systems, making it easy to create content that is accessible to a broad audience.*

- *Flexibility and Customizability: HTML allows developers and designers to create custom web pages and applications that meet their specific needs and requirements.*

Overall, HTML is a foundational technology that plays an essential role in the creation of modern websites and web applications. Its simplicity, versatility, and wide compatibility make it an ideal choice for developers and content creators alike.

1.3 CSS

CSS stands for Cascading Style Sheets, which is a language used for describing the presentation and style of web pages. CSS is used to control the layout, formatting, and visual appearance of HTML elements on a web page, such as fonts, colors, spacing, borders, and backgrounds.

CSS works by selecting HTML elements and applying styles to them through a set of rules. These rules can be written directly into an HTML document, but are

typically stored in a separate CSS file that is linked to the HTML document.

CSS provides a number of benefits for web developers and designers, including:

- Separation of Content and Presentation: CSS allows developers to separate the content of a web page from its presentation, making it easier to manage and maintain web pages.
- Consistency: CSS provides a way to apply consistent styles across multiple web pages, ensuring that the look and feel of a website is cohesive and professional.
- Flexibility and Customizability: CSS allows developers to create custom styles and layouts for web pages, making it possible to create unique and visually appealing designs.
- Responsive Design: CSS can be used to create responsive web designs that adapt to different screen sizes and devices, ensuring that websites look and function well on desktops, tablets, and smartphones.

Overall, CSS is a powerful tool for web developers and designers that helps them create visually appealing and functional web pages. By separating content and presentation, CSS makes it easier to maintain and update web pages, while also providing a way to create unique and engaging designs.

1.4 JS

JS stands for JavaScript, which is a high-level programming language used to create dynamic and interactive web pages and web applications. JavaScript is a client-side language, meaning it runs in the browser and can interact with HTML and CSS to manipulate web page elements and provide interactivity.

JavaScript can be used to add a wide range of interactive features to web pages, such as form validation, animations, responsive menus, pop-up windows, and more. JavaScript can also be used to create complex web applications, such as social networks, online games, and productivity tools.

JavaScript works by adding scripts to HTML documents, which are executed in the browser when the web page loads. JavaScript code can also be loaded asynchronously, allowing it to interact with web page elements and user

input in real-time.

Some of the benefits of using JavaScript in web development include:

- Interactivity: JavaScript allows web developers to create interactive and dynamic user interfaces that respond to user input, improving the user experience and engagement.
- Compatibility: JavaScript is supported by all modern web browsers, making it a versatile and reliable tool for creating web pages and web applications.
- Flexibility: JavaScript is a flexible language that can be used for a wide range of applications, from simple form validation to complex web applications.
- Community: JavaScript has a large and active community of developers, which means there are many resources and tools available for learning and using the language effectively.

Overall, JavaScript is a powerful and essential tool for creating dynamic and interactive web pages and web applications. Its versatility, flexibility, and compatibility make it a popular choice among web developers and designers.

1.5 The relationship between HTML, CSS, JS

HTML, CSS, and JavaScript are three essential technologies used in web development to create and design interactive and visually appealing web pages and web applications. Here's a brief explanation of how they work together:

HTML: HTML is used to create the structure and content of a web page. It defines the various elements that make up a web page, such as headings, paragraphs, images, links, and forms. HTML provides a basic framework for a web page, but it does not control the visual appearance or layout of the page.

CSS: CSS is used to add styles and visual formatting to HTML elements. It defines how the content of a web page should look, such as font size, color, spacing, and layout. CSS provides a way to create visually appealing designs for web pages, and it allows developers to apply consistent styles across multiple web pages.

JavaScript: JavaScript is used to add interactivity and dynamic behavior to web pages. It can be used to manipulate HTML and CSS elements in real-time, allowing web pages to respond to user input and update their content dynamically. JavaScript can also be used to create complex web applications that can run entirely within the browser.

1.6 DBMS, Database, Web

1.6.1 DBMS

DBMS stands for Database Management System, which is software that allows users to manage and manipulate data in a database. A DBMS provides an interface for users to interact with a database, allowing them to perform tasks such as creating, updating, and deleting data.

Some popular DBMS used in web development include MySQL, PostgreSQL, MongoDB, and Oracle. Each DBMS has its own unique features and capabilities, and choosing the right one depends on the specific needs of the application.

1.6.2 Relationship between web and database

Web applications often require a database to store and manage large amounts of data that need to be accessed and manipulated by the application.

A database is an organized collection of data that can be accessed, managed, and updated. It can be thought of as a digital filing cabinet, where data is stored in a structured format, allowing it to be easily retrieved and manipulated. Databases are used in a wide range of applications, including web development, finance, healthcare, and more.

In web development, databases are used to store data such as user profiles, product catalogs, transaction records, and more. Web applications use a database management system (DBMS) to access and manipulate the data stored in the database. Popular DBMS used in web development include MySQL, PostgreSQL, MongoDB, and Oracle.

1.6.3 MySQL

MySQL is a popular open-source relational database management system (RDBMS) that is widely used in web development. It is a powerful and reliable database system that is easy to use and offers excellent performance and scalability.

MySQL is a type of RDBMS, which means that it organizes data into tables that are related to one another through common fields or keys. Each table in a MySQL database contains rows of data, with each row representing a unique record or entry.

MySQL supports a wide range of features, including:

- SQL language support: MySQL supports the standard SQL language, allowing developers to write queries and manipulate data in a familiar syntax.
- High-performance: MySQL is designed for high-performance, with features such as caching, indexing, and optimized query execution.
- Scalability: MySQL can handle large amounts of data and can be scaled up or down to meet the needs of the application.
- Security: MySQL offers robust security features, such as user authentication and access controls, to protect data from unauthorized access or tampering.

MySQL is commonly used in web development to store and manage data for web applications, such as user profiles, product catalogs, and transaction records. It is also used in a wide range of other applications, such as finance, healthcare, and more.

1.7 XAMPP

XAMPP is a free, open-source software package that provides a local web server environment for developers to test and develop web applications on their own computers. It includes several components, such as Apache, MySQL, PHP, and Perl, that are commonly used in web development.

The name XAMPP is an acronym for the components included in the package:

- *X: Cross-platform (works on multiple operating systems)*

- *A: Apache HTTP Server*
- *M: MySQL database server*
- *P: PHP programming language*
- *P: Perl programming language*

In addition to these core components, XAMPP also includes other useful tools and utilities, such as phpMyAdmin for managing MySQL databases, FileZilla FTP client, and Mercury Mail Transport System.

By installing XAMPP, developers can create a local web server environment on their own computer, which enables them to test and debug their applications before deploying them to a production server. This allows for faster and more efficient development, as developers can make changes and see the results immediately without having to upload files to a remote server.

XAMPP is available for Windows, Linux, and macOS, and can be downloaded for free from the official website.

1.8 MVC

MVC stands for Model-View-Controller, which is a software design pattern used in web development to separate an application's data (model), user interface (view), and control logic (controller) into distinct components.

The Model(M) represents the application's data and handles all database-related operations, such as querying the database and updating records.

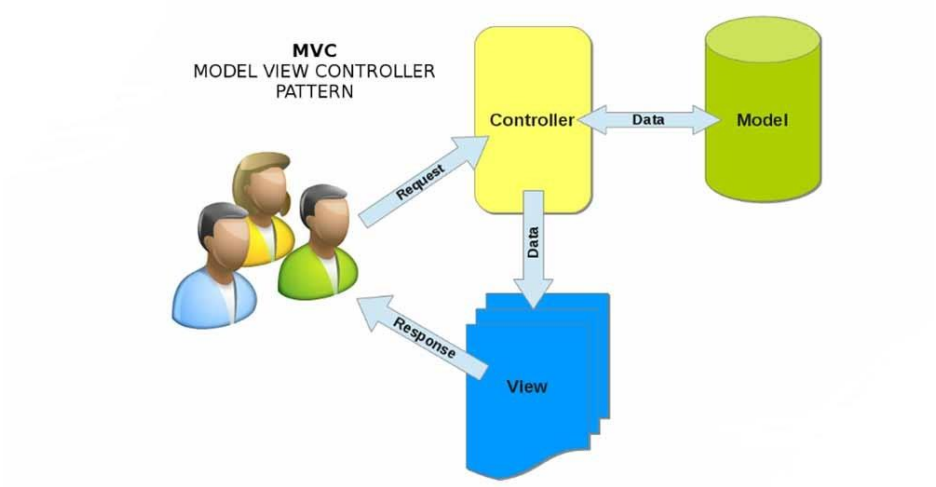
The View(V) is responsible for presenting the data to the user in a clear and organized way. It includes the HTML templates, CSS stylesheets, and JavaScript scripts that make up the user interface.

The Controller (C) handles user requests and serves as an intermediary between the Model and View. It receives input from the user and interacts with the Model to retrieve and update data, and then passes the data to the View for display.

The separation of these components allows for better code organization, easier

maintenance and updates, and improved scalability of the application. It also enables multiple developers to work on different parts of the application simultaneously without interfering with each other's code.

MVC is a widely used pattern in web development, and many popular web frameworks. By following the MVC pattern, developers can create well-structured and maintainable web applications that are flexible and adaptable to changing requirements.



CHAPTER II: DESIGN ANALYSIS, APPLY

2.1 Functional analysis

Account management function:

- *Login*
- *Logout*
- *Display system interface.*
- *Add/remove/edit accounts.*
- *Show avatar.*
- *Change avatar.*
- *Change the display language from English to Vietnamese and vice versa.*

2.2 Database design

CHAPTER III : RESULT VERIFICATION

3.1 Install environment

Installing Xampp

Access the following link:

https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/7.4.15/

Run project by using artisan

Run project by using artisan

Node already provides a way to build the build process with just npm and the package.json file, so basically need to create custom scripts in the script field of the package.json file.

- **Initialization**

  - Create an empty folder and open terminal in that folder.
  - Run the following command:

```
npm init
```

  - Skip all the questions by repeatedly entering. A few important notes from the question in npm init/file package.json A. name: Name of the application (must be unique when pushing to NPMjs.org as an npm package) B. main: starting point of application

```
C:\Users\cuong_000\Desktop\npm>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (npm) test_npm
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\cuong_000\Desktop\npm\package.json:

{
  "name": "test_npm",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this ok? (yes)
```

- Install “npm start” in file package.json:

```
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon --exec npx babel-node src/server.js"
  },
```

IDE, code editor

We need to write code, and of course we need an application that can be used to edit it. The simplest one could be Nodepad, nano... But these applications do not highlight the code, making it difficult to read. Text editors can be categorized as follows:

- *Code editors: Sublime Text, Visual Studio Code, Notepad++, Atom...*

- *IDE (Integrated Development Environment): PHPStorm, Eclipse, NetBean, ZendStudio...*

3.2 Demo interface of some main pages

Login page:

- File Login.js: I use some function to get information and get action to login.
  - Initialize the variables in constructor:

```
constructor(props) {
    super(props);
    this.state = {
        username: '',
        password: '',
        isShowPassword: false,
        errMessage: ''
    }
}
```

  - Get user name:

```
/**get user name */
handleOnChangeUsername = (event) => {
    this.setState({
        username: event.target.value
    })
}
```

  - Get user password:

```
/**get user password */
handleOnChangePassword = (event) => {
    this.setState({
        password: event.target.value
    })
}
```

  - Setting hide or show password:

```
/**setting hind or show password */
handleShowPassword = () => {
    this.setState({
        isShowPassword: !this.state.isShowPassword
    })
}
```

```
<div className='col-12 form-group login-input'>
    <label>Password:</label>
    <div className='eye-show-password'>
        <input
            className='form-control'
            type={this.state.isShowPassword ? 'text' : 'password'}
            placeholder='Enter your password'
            value={this.state.password}
            onChange={(event) => this.handleOnChangePassword(event)}
        />

        <span onClick={() => this.handleShowPassword()}>
            <i class={this.state.isShowPassword ? "fas fa-eye-slash" : "fas fa-eye"}></i></span>
    </div>
</div>
```

- Get action after click on Login button:

```
/**login function */
handleLogin = async () => {
    this.setState({
        errMessage: ''
    })
    try {
        let data = await handleLogin(this.state.username, this.state.password)
        if (data && data.errCode !== 0) {
            this.setState({
                errMessage: data.message
            })
        }
        if (data && data.errCode === 0) {
            this.props.userLoginSuccess(data.users)
            toast.success("Login succeed!")
        }
    } catch (e) {
        if (e.response) {
            if (e.response.data) {
                this.setState({
                    errMessage: e.response.data.message
                })
            }
        }
        console.log(e.response);
    }
}
```

➔ Then will call the handleLogin function in userService.js:

```
const handleLogin = (email, password) => {
    return axios.post('/api/login', { email, password });
}
```

➔ Use API to call the backend:

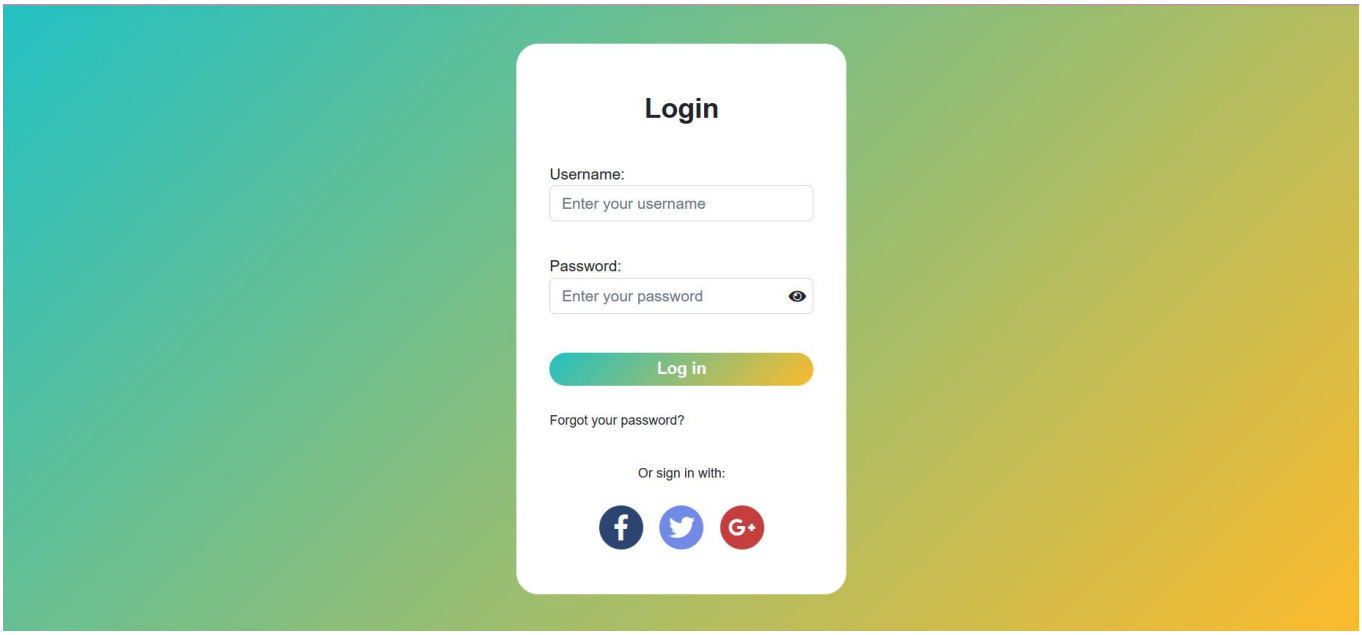
```
router.post('/api/login', userController.handleLogin);
```

➔ Then will call the handleLogin function in userController.js:

```
let handleLogin = async (req, res) => {
    let email = req.body.email;
    let password = req.body.password;

    if (!email || !password) {
        return res.status(500).json({
            errCode: 1,
            message: 'Missing inputs parameter!'
        })
    }

    let userData = await userService.handleUserLogin(email, password);

    return res.status(200).json({
        errCode: userData.errCode,
        message: userData.errMessage,
        users: userData.users ? userData.users : {}
    })
}
```

➔ Then will call the handleUserLoign function in userService.js:

```
/**login function */
let handleUserLogin = (email, password) => {
    return new Promise(async (resolve, reject) => {
        try {
            let userData = {};

            let isExist = await checkUserEmail(email);
            if (isExist) {

                //users already exist
                let users = await db.users.findOne({
                    where: { email: email },
                    attributes: ['id', 'email', 'password', 'fullname'],
                    raw: true
                });
                if (users) {
                    //compare password
                    let check = await bcrypt.compareSync(password, users.password);

                    if (check) {
                        userData.errCode = 0;
                        userData.errMessage = 'Login success';
                        delete users.password;   //An password
                        userData.users = users;
                    } else {
                        userData.errCode = 3;
                        userData.errMessage = 'Password is not true';
                    }
                } else {
                    userData.errCode = 2;
                    userData.errMessage = "users is not found";

                }

            } else {
                //return error
                userData.errCode = 1;
                userData.errMessage = "Your Email isn't exist in the system. Please try again!";

            }
            resolve(userData)
        } catch (e) {
            reject(e);
        }
    })
}
```



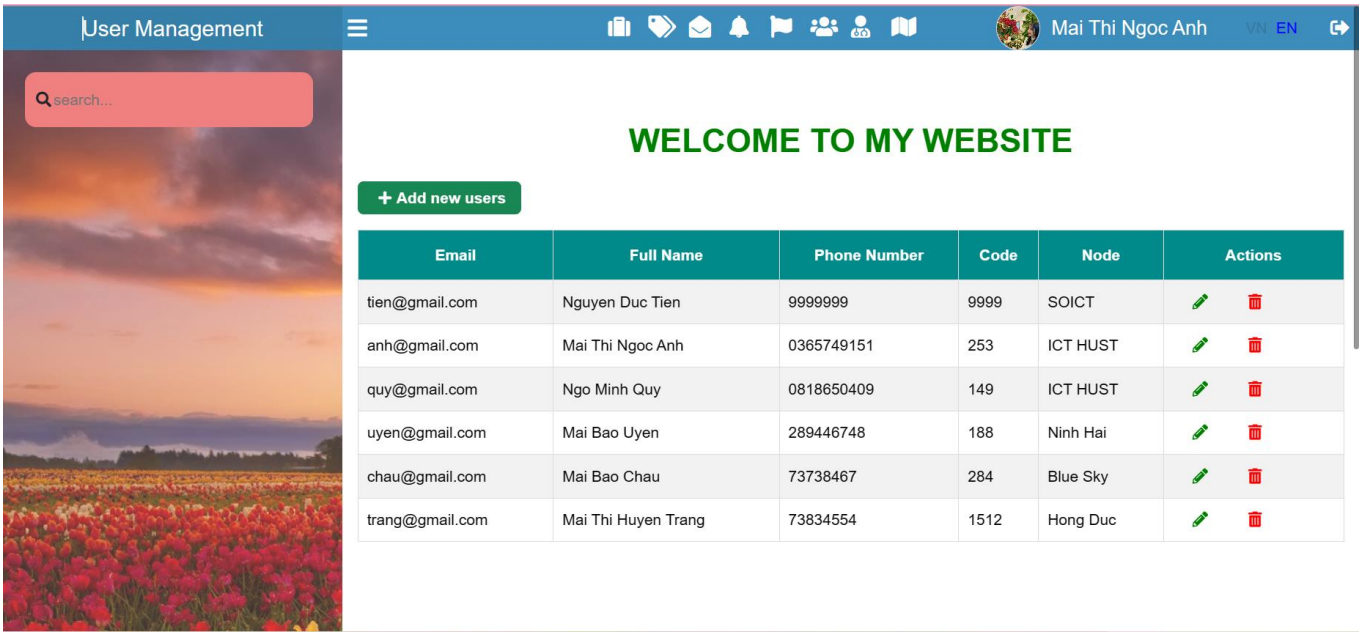
- After click on Login button the website will be redirected to the URL /system/users-manage on the Home.js:

```
render() {
    const { isLoggedIn } = this.props;
    let linkToRedirect = isLoggedIn ? '/system/users-manage' : '/login';

    return (
        <Redirect to={linkToRedirect} />
    );
}
```

- After in isLoggedIn state, go to Homeheader file to display Homepage:

```
render() {

    const { systemMenuPath, isLoggedIn } = this.props;
    return (
        <>
            {/* show HomeHeader after login */}
            {isLoggedIn && <HomeHeader />}

        </>
    );
}
```

Home page:



- Table of user is displayed in UserManage.js file with the getAllUserFromReact function:

```
/**Show list of users  */
getAllUserFromReact = async () => {
    let response = await getAllUsers('All');
    if (response && response.errCode === 0) {
        this.setState({
            arrUsers: response.users
        })
    }
}
```

- The variables in UserManage.js file:

```
/**initialize the variables*/
constructor(props) {
    super(props);
    this.state = {
        arrUsers: [],
        isOpenModalUser: false,     //check open Modal or not
        isOpenModalEditUser: false,
        userEdit: {}
    }
}
```

- After click on Add new users button, go to handleAddNewUser function:

```
<div className='add'>
    <div className='mx-3'>
        <button className='btn btn-success px-3'
            onClick={() => this.handleAddNewUser()}
        ><i className="fas fa-plus"></i><FormattedMessage id="homeheader.button" /> </button>
    </div>
</div>
```

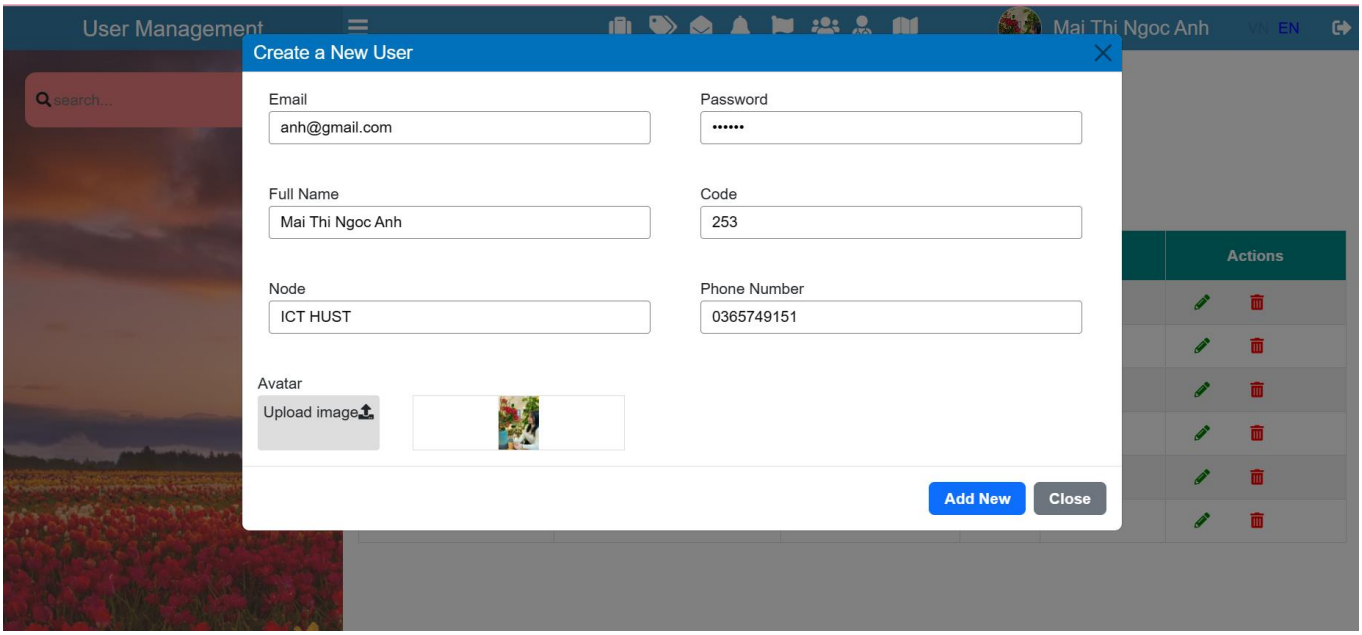
```
/**set up show modal add new users*/
handleAddNewUser = () => {
    this.setState({
        isOpenModalUser: true,
    })
}
```

- Set the toggle of Add new users modal:

```
/**set up toggle of form add new user */
toggleUserModal = (data) => {
    this.setState({
        isOpenModalUser: !this.state.isOpenModalUser
    })
}
```

- Go to ModalUser.js file:

Add new user function:



- The variables in ModalUser.js file:

```
constructor(props) {
    super(props);
    this.state = {
        email: '',
        password: '',
        fullname: '',
        code: '',
        node: '',
        phonenum: '',
        isOpen: false,
        image: '',
        previewImgURL: '',
        isOpenImage: false,
    }

    this.toggleImage = this.toggleImage.bind(this)
    this.listenToEmitter();
}
```

- The input value is stored at function handleOnChangeInput:

```
/**set state value for variables */
hanleOnChangeInput = (event, id) => {
    let copyState = { ...this.state };
    copyState[id] = event.target.value;
    this.setState({
        ...copyState
    })
}
```

- After click on Add New button, go to handleAddNewUser function:

```
/**get values */
handleAddNewUser = () => {
    let isValid = this.checkValidInput();
    if (isValid === true) {
        //call API
        this.props.createNewUser(this.state);
    }
}
```

- checkValidInput function:

```
/**check validity of input */
checkValidInput = () => {
    let isValid = true;
    let arrInput = ['email', 'password', 'fullname', 'code', 'node', 'phonenum'];
    for (let i = 0; i < arrInput.length; i++) {
        if (!this.state[arrInput[i]]) {
            isValid = false;
            alert('Missing parameter:' + arrInput[i]);
            break;
        }
    }
    return isValid;
}
```

- Go to createNewUser function at UserManage.js file:

```
/**save user info*/
createNewUser = async (data) => {
    try {
        let response = await createNewUserService(data)
        if (response && response.errCode !== 0) {
            alert(response.errMessage)
        } else {
            await this.getAllUserFromReact();
            toast.success('Create new user succeed!')
            this.setState({
                isOpenModalUser: false
            })

            //clear data in modal after add new users
            emitter.emit('EVENT_CLEAR_MODAL_DATA')
        }
    } catch (e) {
        console.log(e)
    }
}
```

- Go to createNewUserService function at userService.js file:

```
const createNewUserService = (data) => {
    return axios.post('/api/create-new-user', data)
}
```

- Use API call backend:

```
router.post('/api/create-new-user', userController.handleCreateNewUser);
```

- Go to handleCreateNewUser function at userController.js file:

```
/**create new user function */
let handleCreateNewUser = async (req, res) => {
    let message = await userService.createNewUser(req.body);
    return res.status(200).json(message);
}
```

- Go to createNewUser function at userService.js file:

```
/**create a new user */
let createNewUser = (data) => {
    return new Promise(async (resolve, reject) => {
        try {
            //check email is exist?
            let check = await checkUserEmail(data.email);
            if (check === true) {
                resolve({
                    errCode: 1,
                    errMessage: 'This email is already used. Please try another email!'
                })

            }
            else {
                let hashPasswordFromBcrypt = await hashUserPassword(data.password);
                await db.users.create({
                    email: data.email,
                    password: hashPasswordFromBcrypt,
                    fullname: data.fullname,
                    code: data.code,
                    node: data.node,
                    phonenum: data.phonenum,
                    image: data.image
                })

                resolve({
                    errCode: 0,
                    message: 'Successfully'
                })
            }
        } catch (e) {
            reject(e);
        }
    })
}
```

- When click on Edit button (Pencil symbol) go to the handleEditUser function at UserManage.js file:

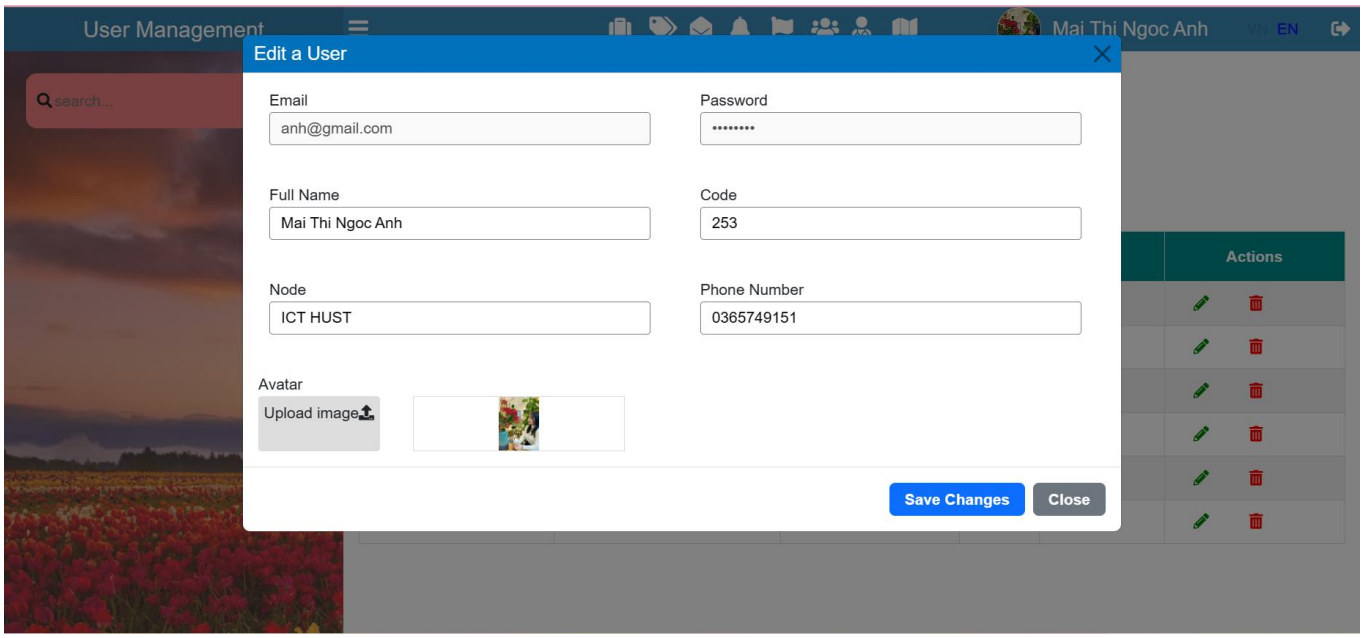
```
<button className='btn-edit' onClick={() => this.handleEditUser(item)}><i className="fas fa-pencil-alt"></i></button>
```

- Set the toggle of Edit user modal:

```
/**set up modal edit users and get user info*/
handleEditUser = (users) => {
    this.setState({
        isOpenModalEditUser: true,
        userEdit: users
    })
}

/**set up toggle form edit user */
toggleEditUserModal = (data) => {
    this.setState({
        isOpenModalEditUser: !this.state.isOpenModalEditUser
    })
}
```

- Go to ModalEditUser.js file.

Edit user function:



- The variables in ModalEditUser.js file:

```
constructor(props) {
    super(props);
    this.state = {
        id: '',
        email: '',
        password: '',
        fullname: '',
        code: '',
        node: '',
        phonenum: '',
        isOpen: false,
        image: '',
        previewImgURL: '',
        isOpenImage: false,
    }
    this.toggleImage = this.toggleImage.bind(this)
}
```

- Get user information:

```
/**get user information */
componentDidMount() {
    let users = this.props.currentUser;
    if (users && !_.isEmpty(users)) {
        let imageBase64 = '';
        if (users.image) {
            imageBase64 = new Buffer(users.image, 'base64').toString('binary');
        }
        this.setState({
            id: users.id,
            email: users.email,
            password: 'hashcode',
            fullname: users.fullname,
            code: users.code,
            node: users.node,
            phonenum: users.phonenum,
            image: '',
            previewImgURL: imageBase64
        })
    }
}
```

- The input value is stored at hanleOnChangeInput function:

```
/**set state value for variables */
hanleOnChangeInput = (event, id) => {
    let copyState = { ...this.state };
    copyState[id] = event.target.value;
    this.setState({
        ...copyState
    })
}
```

- After click on Save Changes button, go to handleSaveUser function to save user information:

```
/**get values */
handleSaveUser = () => {
    let isValid = this.checkValidInput();
    if (isValid === true) {
        //call API
        this.props.editUser(this.state);
    }
}
```

- Go to editUser function at UserManage.js file:

```
/**Edit users function */
editUser = async (users) => {
    try {
        let res = await editUserService(users)
        if (res && res.errCode === 0) {
            this.setState({
                isOpenModalEditUser: false
            })
            await this.getAllUserFromReact();
            toast.success('Edit user succeed!')
        } else {
            alert(res.errCode)
        }
    } catch (e) {
        console.log(e)
    }
}
```

- Go to editUserService function at userService.js file:

```
const editUserService = (userId) => {
    return axios.put('/api/edit-user', userId);
}
```

- Use API to call the backend:

```
router.put('/api/edit-user', userController.handleEditUser);
```

- Go to let handleEditUser function at userController.js file:

```
/**edit user function */
let handleEditUser = async (req, res) => {
    let data = req.body;
    let message = await userService.updateUserData(data);
    return res.status(200).json(message)
}
```

- Go to updateUserData function at userService.js file:

```
/**update user by id after edit */
let updateUserData = (data) => {
    return new Promise(async (resolve, reject) => {
        try {

            if (!data.id) {
                resolve({
                    errCode: 2,
                    errMessage: "Missing requied parameters"
                })
            }
            let users = await db.users.findOne({
                where: { id: data.id },
                raw: false
            })
            if (users) {
                users.email = data.email;
                users.fullname = data.fullname;
                users.phonenum = data.phonenum;
                users.code = data.code;
                users.node = data.node;
                if (data.image) {
                    users.image = data.image;
                }

                await users.save();

                resolve({
                    errCode: 0,
                    message: 'Update the users successful!'
                });
            } else {
                resolve({
                    errCode: 1,
                    errMessage: 'users is not found!'
                });
            }
        } catch (e) {
            reject(e);
        }
    })
}
```

Chapter IV: CONCLUSION

4.1 Conclusion

- **Result:**

Learn about web, pure web code, use framework to do basic CRUD.

- **Accumulated knowledge:**

  - Database design
  - Interface design
  - Learn about MVC framework
  - Ability to study documents to solve problems
  - Debug

4.2 Evaluation

- **Overall:**

The main content of this project is still to learn about technology, and then apply that technology to create interfaces and basic functions. Through this, there is a basic understanding of web and web design.

The output product still has many deficiencies and cannot be considered complete.

- **Desire:**

Complete the functions, control the input data, and have a usable web application in practice.

I would like to express my gratitude once more to M.S. Nguyen Duc Tien for helping me with this project. Although there were some errors in the project, I am hopeful that I can receive additional guidance from you to enhance my future projects.

Thank you with sincerity!

Chapter V: REFERENCES

1. Khoá học Full stack: https://www.youtube.com/watch?v=VvvXhNbFWKY&list=PLncHg6Kn2JT6E38Z3kit9Hnif1xC_9VqI
2. https://www.w3schools.com/
3. https://stackoverflow.com/

~ END ~