***NguyenLien Shop***

Backend – NodeJS

Frontend – React/TS

Database – MySQL

***IDE: VSCODE, XAMPP***

\*Lưu ý: Project chưa hoàn thiện, vẫn còn đang phát triển và chỉnh sửa rất nhiều thứ!

# **Backend – NodeJS**

## **/src**

### **/src/config/**

#### **config.json**

1. {
2. "development": {
3. "username": "root",
4. "password": null,
5. "database": "nguyenlien",
6. "host": "127.0.0.1",
7. "dialect": "mysql",
8. "logging": false,
9. "timezone": "+07:00"
10. },
11. "test": {
12. "username": "root",
13. "password": null,
14. "database": "database\_test",
15. "host": "127.0.0.1",
16. "dialect": "mysql"
17. },
18. "production": {
19. "username": "root",
20. "password": null,
21. "database": "database\_production",
22. "host": "127.0.0.1",
23. "dialect": "mysql"
24. }
25. }

#### **connectDB.js**

1. import { Sequelize } from 'sequelize';
2. import dotenv from 'dotenv';
3. // Load biến môi trường từ .env
4. dotenv.config();
5. // Tạo instance Sequelize dùng biến từ .env để tiện tái sử dụng
6. const sequelize = new Sequelize(
7. process.env.DB\_NAME || 'nguyenlien',
8. process.env.DB\_USER || 'root',
9. process.env.DB\_PASSWORD || null,
10. {
11. host: process.env.DB\_HOST || 'localhost',
12. port: process.env.DB\_PORT || 3306,
13. dialect: 'mysql',
14. logging: false, // Có thể bật thành true khi debug query
15. }
16. );
17. // Hàm kết nối DB
18. const connectDB = async () => {
19. try {
20. await sequelize.authenticate();
21. console.log('✅ Đã kết nối MySQL thành công!');
22. } catch (error) {
23. console.error('❌ Kết nối MySQL thất bại:', error.message);
24. process.exit(1); // Thoát chương trình nếu kết nối DB thất bại
25. }
26. };
27. export default connectDB;
28. export { sequelize };

#### **viewEngine.js**

1. import express from "express";
2. let configViewEngine = (app) => {
3. app.use(express.static("./src/public"))
4. app.set("view engine", "ejs")
5. app.set("views", "./src/views")
6. }
7. module.exports = configViewEngine;

### **/src/controllers/**

#### **adminController.js**

1. import adminService from "../services/adminService.js";
2. import dotenv from 'dotenv';
3. dotenv.config();
4. // ==================== USER MANAGER ====================
5. // Get All Users or Single User
6. let handleGetAllUsers = async (req, res) => {
7. const id = req.params.id;
8. if (!id) {
9. return res.status(400).json({
10. errCode: 1,
11. errMessage: "Missing required parameter",
12. users: []
13. });
14. }
15. try {
16. const users = await adminService.getAllUsers(id);
17. return res.status(200).json({
18. errCode: 0,
19. errMessage: "Ok",
20. users
21. });
22. } catch (error) {
23. console.error("Get Users Error:", error);
24. return res.status(500).json({
25. errCode: -1,
26. errMessage: "Error from server"
27. });
28. }
29. };
30. // Delete User
31. let handleDeleteUser = async (req, res) => {
32. const userId = req.params.id;
33. if (!userId) {
34. return res.status(400).json({
35. errCode: 1,
36. errMessage: "Missing user ID"
37. });
38. }
39. try {
40. const message = await adminService.deleteUser(userId);
41. return res.status(200).json(message);
42. } catch (error) {
43. console.error("Delete User Error:", error);
44. return res.status(500).json({
45. errCode: -1,
46. errMessage: "Error from server"
47. });
48. }
49. };
50. // Edit User
51. let handleEditUser = async (req, res) => {
52. const data = req.body;
53. if (!data || !data.id) {
54. return res.status(400).json({
55. errCode: 1,
56. errMessage: 'Missing user ID'
57. });
58. }
59. try {
60. const message = await adminService.updateUserData(data);
61. return res.status(200).json(message);
62. } catch (error) {
63. console.error("Edit User Error:", error);
64. return res.status(500).json({
65. errCode: -1,
66. errMessage: "Server error while updating user"
67. });
68. }
69. };
70. // ==================== EXPORT ====================
71. export default {
72. handleGetAllUsers,
73. handleEditUser,
74. handleDeleteUser
75. };

#### **authController.js**

1. import authService from "../services/authService.js";
2. import jwt from 'jsonwebtoken';
3. import dotenv from 'dotenv';
4. import sendResponse from '../utils/sendResponse.js';
5. dotenv.config();
6. const handleLogin = async (req, res) => {
7. const { identifier, password } = req.body;
8. if (!identifier || !password) {
9. return sendResponse(res, {
10. status: 400,
11. errCode: 1,
12. message: 'Missing inputs',
13. });
14. }
15. const userData = await authService.loginUser(identifier, password);
16. if (userData.errCode !== 0) {
17. return sendResponse(res, {
18. status: 401,
19. errCode: userData.errCode,
20. message: userData.errMessage || 'Sai thông tin đăng nhập!',
21. });
22. }
23. const user = userData.user;
24. const token = jwt.sign(
25. { id: user.id, roleId: user.roleId },
26. process.env.JWT\_SECRET,
27. { expiresIn: '1h' }
28. );
29. return sendResponse(res, {
30. message: 'Login successful',
31. token,
32. data: user
33. });
34. };
35. const handleRegister = async (req, res) => {
36. const result = await authService.registerUser(req.body);
37. return sendResponse(res, {
38. status: result.errCode === 0 ? 200 : 400,
39. errCode: result.errCode,
40. message: result.errMessage || result.message,
41. });
42. };
43. export default {
44. handleLogin,
45. handleRegister,
46. };

#### **backendController.js**

1. import backendService from '../services/backendService';
2. import { checkEmailExists, checkPhoneNumberExists, checkUserNameExists } from '../services/validators';
3. let homePage = (req, res) => {
4. return res.render('homePage.ejs');
5. };
6. let userManager = async (req, res) => {
7. try {
8. const data = await backendService.getAllUser();
9. const message = req.query.message || null;
10. return res.render('userManager.ejs', { dataTable: data, message });
11. } catch (error) {
12. console.error(error);
13. return res.status(500).send('Đã xảy ra lỗi khi tải danh sách người dùng.');
14. }
15. };
16. let createNewUser = (req, res) => {
17. return res.render('createNewUser.ejs', {
18. errorMessage: null,
19. oldInput: null,
20. emailError: null,
21. phoneError: null,
22. passwordError: null
23. });
24. };
25. let postCreateNewUser = async (req, res) => {
26. try {
27. const { userName, password, phoneNumber, roleId, email } = req.body;
28. // Trim dữ liệu
29. req.body.userName = userName?.trim();
30. req.body.password = password?.trim();
31. req.body.phoneNumber = phoneNumber?.trim();
32. req.body.email = email?.trim();
33. const oldInput = req.body;
34. // Validate bắt buộc
35. if (!userName || !password || !roleId) {
36. return res.render('createNewUser.ejs', {
37. errorMessage: 'Please fill in all required fields!',
38. oldInput,
39. emailError: null,
40. phoneError: null,
41. passwordError: null
42. });
43. }
44. // Validate password length`
45. if (password.length < 6) {
46. return res.render('createNewUser.ejs', {
47. errorMessage: null,
48. oldInput,
49. passwordError: 'Password must be at least 6 characters!',
50. emailError: null,
51. phoneError: null
52. });
53. }
54. // Check username exists
55. const loginNameExists = await checkUserNameExists(userName);
56. if (loginNameExists) {
57. return res.render('createNewUser.ejs', {
58. errorMessage: 'Username already exists!',
59. oldInput,
60. emailError: null,
61. phoneError: null,
62. passwordError: null
63. });
64. }
65. // Validate phone format
66. const phonePattern = /^0\d{9}$/;
67. if (phoneNumber && !phonePattern.test(phoneNumber)) {
68. return res.render('createNewUser.ejs', {
69. errorMessage: null,
70. oldInput,
71. phoneError: 'Invalid phone number format (must start with 0 and be 10 digits)',
72. emailError: null,
73. passwordError: null
74. });
75. }
76. // Check phone exists
77. if (phoneNumber) {
78. const phoneExists = await checkPhoneNumberExists(phoneNumber);
79. if (phoneExists) {
80. return res.render('createNewUser.ejs', {
81. errorMessage: null,
82. oldInput,
83. phoneError: 'Phone number already exists!',
84. emailError: null,
85. passwordError: null
86. });
87. }
88. }
89. // Validate email format
90. const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
91. if (email && !emailPattern.test(email)) {
92. return res.render('createNewUser.ejs', {
93. errorMessage: null,
94. oldInput,
95. emailError: 'Invalid email format!',
96. phoneError: null,
97. passwordError: null
98. });
99. }
100. // Check email exists
101. if (email) {
102. const emailExists = await checkEmailExists(email);
103. if (emailExists) {
104. return res.render('createNewUser.ejs', {
105. errorMessage: null,
106. oldInput,
107. emailError: 'Email already exists!',
108. phoneError: null,
109. passwordError: null
110. });
111. }
112. }
113. // Tạo người dùng
114. await backendService.createNewUser(req.body);
115. return res.redirect('/user-manager?message=User created successfully');
116. } catch (error) {
117. console.error('Error creating account: ', error);
118. return res.status(500).send("An error occurred while creating the user!");
119. }
120. };
121. let editUser = async (req, res) => {
122. const userId = req.query.id;
123. if (!userId) return res.status(404).send("Không tìm thấy người dùng.");
124. try {
125. const userData = await backendService.getUserInfoById(userId);
126. return res.render("updateUser.ejs", {
127. user: userData,
128. error: null
129. });
130. } catch (error) {
131. console.error(error);
132. return res.status(500).send("Lỗi khi tải dữ liệu người dùng.");
133. }
134. };
135. let updateUser = async (req, res) => {
136. try {
137. const data = req.body;
138. const result = await backendService.updateUserData(data);
139. if (result.errCode === 0) {
140. return res.redirect('/user-manager');
141. }
142. else {
143. return res.render("updateUser.ejs", {
144. user: data,
145. error: result.errMessage || 'Update user failed.',
146. emailError: result.errCode === 3 ? result.errMessage : null,
147. phoneError: result.errCode === 5 ? result.errMessage : null
148. });
149. }
150. } catch (error) {
151. console.error("User update error:", error);
152. return res.status(500).send("An error occurred while updating the user.");
153. }
154. };
155. let deleteUser = async (req, res) => {
156. const id = req.query.id;
157. if (!id) return res.status(404).send("Missing user!");
158. try {
159. const result = await backendService.deleteUserById(id);
160. if (result.errCode === 0) {
161. return res.redirect('/user-manager');
162. } else {
163. return res.status(404).send("User does not exist.");
164. }
165. } catch (error) {
166. console.error("Error deleting user::", error);
167. return res.status(500).send("An error occurred while deleting the user.");
168. }
169. };
170. export default {
171. homePage,
172. userManager,
173. createNewUser,
174. postCreateNewUser,
175. editUser,
176. updateUser,
177. deleteUser,
178. };

#### **userController.js**

1. import userService from "../services/userService";
2. import dotenv from 'dotenv';
3. dotenv.config();
4. let handleGetUserProfile = async (req, res) => {
5. try {
6. const userId = req.user?.id;
7. if (!userId) {
8. return res.status(401).json({
9. errCode: 1,
10. message: 'Unauthorized',
11. });
12. }
13. const user = await userService.getUserInfoById(userId);
14. if (!user) {
15. return res.status(404).json({
16. errCode: 2,
17. message: 'User not found',
18. });
19. }
20. return res.status(200).json({
21. errCode: 0,
22. message: 'Success',
23. data: user,
24. });
25. } catch (error) {
26. console.error(error);
27. return res.status(500).json({
28. errCode: -1,
29. message: 'Server error',
30. });
31. }
32. };
33. let handleEditUserProfile = async (req, res) => {
34. };
35. let updateAddress = async (req, res) => {
36. };
37. let updateEmail = async (req, res) => {
38. };
39. let updatePhoneNumber = async (req, res) => {
40. };
41. let changePassword = async (req, res) => {
42. };
43. export default {
44. handleGetUserProfile: handleGetUserProfile,
45. handleEditUserProfile: handleEditUserProfile,
46. updateAddress: updateAddress,
47. updateEmail: updateEmail,
48. updatePhoneNumber: updatePhoneNumber,
49. changePassword: changePassword
50. };

### **/src/middlewares/**

#### **authMiddleware.js**

1. import jwt from 'jsonwebtoken';
2. import dotenv from 'dotenv';
3. dotenv.config(); // để đọc biến .env
4. const secret = process.env.JWT\_SECRET;
5. export const verifyToken = (req, res, next) => {
6. const token = req.headers.authorization?.split(' ')[1]; // Bearer <token>
7. if (!token) {
8. return res.status(401).json({ message: 'Token not provided!' });
9. }
10. try {
11. const decoded = jwt.verify(token, secret);
12. req.user = decoded; // thêm info user vào req
13. next();
14. } catch (err) {
15. return res.status(403).json({ message: 'Invalid or expired token' });
16. }
17. };
18. export const checkOwner = (req, res, next) => {
19. const loggedInUserId = req.user?.id;
20. const targetUserId = parseInt(req.params.id);
21. if (loggedInUserId !== targetUserId) {
22. return res.status(403).json({ message: 'Access denied: Not owner' });
23. }
24. next();
25. };
26. export const isRole = (requiredRoleId) => {
27. return (req, res, next) => {
28. if (req.user?.roleId !== requiredRoleId) {
29. return res.status(403).json({ message: 'Access denied' });
30. }
31. next();
32. };
33. };

#### **validateBodyFieds.js**

1. const validateBodyFields = (requiredFields = []) => {
2. return (req, res, next) => {
3. const missing = requiredFields.filter(field => !req.body[field]);
4. if (missing.length > 0) {
5. return res.status(400).json({
6. errCode: 1,
7. message: `Missing fields: ${missing.join(', ')}`,
8. });
9. }
10. next();
11. };
12. };
13. export default validateBodyFields;

### **/src/migration/**

#### **migration-allcode.js**

1. 'use strict';
2. module.exports = {
3. up: async (queryInterface, Sequelize) => {
4. await queryInterface.createTable('AllCodes', {
5. id: { //k can khai bao trong user
6. allowNull: false,
7. autoIncrement: true,
8. primaryKey: true,
9. type: Sequelize.INTEGER
10. },
11. key: {
12. type: Sequelize.STRING
13. },
14. type: {
15. type: Sequelize.STRING
16. },
17. valueEn: {
18. type: Sequelize.STRING
19. },
20. valueVi: {
21. type: Sequelize.STRING
22. },
23. createdAt: {
24. allowNull: false,
25. type: Sequelize.DATE
26. },
27. updatedAt: {
28. allowNull: false,
29. type: Sequelize.DATE
30. }
31. });
32. },
33. down: async (queryInterface, Sequelize) => {
34. await queryInterface.dropTable('AllCodes');
35. }
36. };

#### **migration-role.js**

1. 'use strict';
2. module.exports = {
3. up: async (queryInterface, Sequelize) => {
4. await queryInterface.createTable('Roles', {
5. id: {
6. allowNull: false,
7. autoIncrement: true,
8. primaryKey: true,
9. type: Sequelize.INTEGER
10. },
11. name: { type: Sequelize.STRING, allowNull: false },
12. description: Sequelize.STRING,
13. createdAt: Sequelize.DATE,
14. updatedAt: Sequelize.DATE,
15. });
16. },
17. down: async (queryInterface, Sequelize) => {
18. await queryInterface.dropTable('Roles');
19. }
20. };

#### **migration-user.js**

1. 'use strict';
2. /\*\* @type {import('sequelize-cli').Migration} \*/
3. module.exports = {
4. async up(queryInterface, Sequelize) {
5. await queryInterface.createTable('Users', {
6. id: {
7. allowNull: false,
8. autoIncrement: true,
9. primaryKey: true,
10. type: Sequelize.INTEGER
11. },
12. userName: {
13. type: Sequelize.STRING
14. },
15. password: {
16. type: Sequelize.STRING
17. },
18. email: {
19. type: Sequelize.STRING
20. },
21. phoneNumber: {
22. type: Sequelize.STRING
23. },
24. fullName: {
25. type: Sequelize.STRING
26. },
27. gender: {
28. type: Sequelize.STRING
29. },
30. roleId: {
31. type: Sequelize.INTEGER,
32. references: {
33. model: 'Roles',
34. key: 'id'
35. }
36. },
37. slug: {
38. type: Sequelize.STRING
39. },
40. createdAt: {
41. allowNull: false,
42. type: Sequelize.DATE
43. },
44. updatedAt: {
45. allowNull: false,
46. type: Sequelize.DATE
47. }
48. });
49. },
50. async down(queryInterface, Sequelize) {
51. await queryInterface.dropTable('Users');
52. }
53. };

### **/src/models/**

#### **allcode.js**

1. 'use strict';
2. const {
3. Model
4. } = require('sequelize');
5. module.exports = (sequelize, DataTypes) => {
6. class Allcode extends Model {
7. /\*\*
8. \* Helper method for defining associations.
9. \* This method is not a part of Sequelize lifecycle.
10. \* The `models/index` file will call this method automatically.
11. \*/
12. static associate(models) { // cac moi quan he o day (nhieu - nhieu, nhieu - it,...)
13. // define association here
14. }
15. };
16. Allcode.init({
17. key: DataTypes.STRING,
18. type: DataTypes.STRING,
19. valueEn: DataTypes.STRING,
20. valueVi: DataTypes.STRING
21. }, {
22. sequelize,
23. modelName: 'Allcode',
24. });
25. return Allcode;
26. };

#### **index.js**

1. 'use strict';
2. require('dotenv').config();
3. const fs = require('fs');
4. const path = require('path');
5. const Sequelize = require('sequelize');
6. const process = require('process');
7. const basename = path.basename(\_\_filename);
8. const env = process.env.NODE\_ENV || 'development';
9. const config = require(\_\_dirname + '/../config/config.json')[env];
10. const db = {};
11. let sequelize;
12. if (config.use\_env\_variable) {
13. sequelize = new Sequelize(process.env[config.use\_env\_variable], config);
14. } else {
15. sequelize = new Sequelize(config.database, config.username, config.password, config);
16. }
17. fs
18. .readdirSync(\_\_dirname)
19. .filter(file => {
20. return (
21. file.indexOf('.') !== 0 &&
22. file !== basename &&
23. file.slice(-3) === '.js' &&
24. file.indexOf('.test.js') === -1
25. );
26. })
27. .forEach(file => {
28. const model = require(path.join(\_\_dirname, file))(sequelize, Sequelize.DataTypes);
29. db[model.name] = model;
30. });
31. Object.keys(db).forEach(modelName => {
32. if (db[modelName].associate) {
33. db[modelName].associate(db);
34. }
35. });
36. db.sequelize = sequelize;
37. db.Sequelize = Sequelize;
38. module.exports = db;

#### **role.js**

1. 'use strict';
2. const {
3. Model
4. } = require('sequelize');
5. module.exports = (sequelize, DataTypes) => {
6. class Role extends Model {
7. /\*\*
8. \* Helper method for defining associations.
9. \* This method is not a part of Sequelize lifecycle.
10. \* The `models/index` file will call this method automatically.
11. \*/
12. static associate(models) { // cac moi quan he o day (nhieu - nhieu, nhieu - it,...)
13. Role.hasMany(models.User, { foreignKey: 'roleId' });
14. }
15. };
16. Role.init({
17. name: DataTypes.STRING,
18. description: DataTypes.STRING,
19. }, {
20. sequelize,
21. modelName: 'Role',
22. });
23. return Role;
24. };

#### **user.js**

1. 'use strict';
2. const {
3. Model
4. } = require('sequelize');
5. module.exports = (sequelize, DataTypes) => {
6. class User extends Model {
7. /\*\*
8. \* Helper method for defining associations.
9. \* This method is not a part of Sequelize lifecycle.
10. \* The `models/index` file will call this method automatically.
11. \*/
12. static associate(models) {
13. User.belongsTo(models.Role, { foreignKey: 'roleId', targetKey: 'id' });
14. }
15. }
16. User.init({
17. userName: DataTypes.STRING,
18. password: DataTypes.STRING,
19. email: DataTypes.STRING,
20. phoneNumber: DataTypes.STRING,
21. fullName: DataTypes.STRING,
22. gender: DataTypes.STRING,
23. roleId: DataTypes.INTEGER,
24. slug: DataTypes.STRING
25. }, {
26. sequelize,
27. modelName: 'User',
28. });
29. return User;
30. };

### **/src/public/image/**

### **/src/routes/**

#### **apiRoutes.js**

1. import express from 'express';
2. import authController from '../controllers/authController.js';
3. import userController from '../controllers/userController.js';
4. import adminController from '../controllers/adminController.js';
5. import { verifyToken, isRole } from '../middlewares/authMiddleware.js';
6. import validateBodyFields from '../middlewares/validateBodyFields.js';
7. const router = express.Router();
8. //public APIs
9. router.post(
10. '/login',
11. validateBodyFields(['identifier', 'password']),
12. authController.handleLogin
13. );
14. router.post('/register', authController.handleRegister);
15. // Admin APIs
16. router.get('/admin/users-manager/:id', verifyToken, isRole(1), adminController.handleGetAllUsers);
17. router.put('/admin/user/:id', verifyToken, isRole(1), adminController.handleEditUser);
18. router.delete('/admin/user/:id', verifyToken, isRole(1), adminController.handleDeleteUser);
19. // // User APIs
20. router
21. .route('/profile/me')
22. .get(verifyToken, isRole(2), userController.handleGetUserProfile)
23. .put(verifyToken, isRole(2), userController.handleEditUserProfile);
24. router.patch('/user/me/address', verifyToken, isRole(2), userController.updateAddress);
25. router.patch('/user/me/email', verifyToken, isRole(2), userController.updateEmail);
26. router.patch('/user/me/phone', verifyToken, isRole(2), userController.updatePhoneNumber);
27. router.patch('/user/me/password', verifyToken, isRole(2), userController.changePassword);
28. export default router;

#### **web.js**

1. import webRoutes from './webRoutes.js';
2. import apiRoutes from './apiRoutes.js';
3. const initRoutes = (app) => {
4. app.use('/', webRoutes);              // View-based routing
5. app.use('/api', apiRoutes);           // All API under /api/\*
6. };
7. export default initRoutes;

#### **webRoutes.js**

1. import express from 'express';
2. import backendController from '../controllers/backendController.js';
3. const router = express.Router();
4. router.get('/', backendController.homePage);
5. router.get('/user-manager', backendController.userManager);
6. router.get('/create-new-user', backendController.createNewUser);
7. router.post('/post-create-new-user', backendController.postCreateNewUser);
8. router.get('/edit-user', backendController.editUser);
9. router.post('/update-user', backendController.updateUser);
10. router.get('/delete-user', backendController.deleteUser);
11. export default router;

### **/src/services/**

#### **adminService.js**

1. import db from "../models/index"
2. import validator from "validator";
3. // User Manager
4. let getAllUsers = (userId) => {
5. return new Promise(async (resolve, reject) => {
6. try {
7. let users = '';
8. if (userId === 'ALL') {
9. users = await db.User.findAll({
10. attributes: { exclude: ['password'] },
11. include: [{ model: db.Role, attributes: ['name'] }]
12. })
13. }
14. if (userId && userId !== 'ALL') {
15. users = await db.User.findOne({
16. where: { id: userId },
17. attributes: {
18. exclude: ['password'] // remove password
19. }
20. })
21. }
22. resolve(users)
23. } catch (e) {
24. reject(e);
25. }
26. })
27. }
28. let deleteUser = (userId) => {
29. return new Promise(async (resolve, reject) => {
30. try {
31. let user = await db.User.findOne({
32. where: { id: userId }
33. });
34. if (!user) {
35. return resolve({
36. errCode: 2,
37. errMessage: `The user isn't exist`
38. });
39. }
40. await user.destroy();
41. resolve({
42. errCode: 0,
43. errMessage: `The user is deleted`
44. });
45. } catch (error) {
46. reject({
47. errCode: -1,
48. errMessage: "An error occurred while deleting the user"
49. });
50. }
51. });
52. }
53. const updateUserData = (data) => {
54. return new Promise(async (resolve, reject) => {
55. try {
56. if (!data.id) {
57. return resolve({
58. errCode: 1,
59. errMessage: 'Missing user ID'
60. });
61. }
62. const user = await db.User.findOne({
63. where: { id: data.id },
64. raw: false
65. });
66. if (!user) {
67. return resolve({
68. errCode: 3,
69. errMessage: 'User not found'
70. });
71. }
72. // Kiểm tra email
73. if (data.email && data.email !== user.email) {
74. if (!validator.isEmail(data.email)) {
75. return resolve({
76. errCode: 5,
77. errMessage: 'Invalid email format'
78. });
79. }
80. const emailExists = await db.User.findOne({
81. where: {
82. email: data.email,
83. id: { [db.Sequelize.Op.ne]: data.id }
84. }
85. });
86. if (emailExists) {
87. return resolve({
88. errCode: 4,
89. errMessage: 'Email already in use by another user'
90. });
91. }
92. user.email = data.email;
93. }
94. // Các trường tùy chọn
95. user.firstName = data.firstName?.trim() || null;
96. user.lastName = data.lastName?.trim() || null;
97. user.address = data.address?.trim() || null;
98. user.phoneNumber = data.phoneNumber?.trim() || null;
99. if (['M', 'F', 'O'].includes(data.gender)) {
100. user.gender = data.gender;
101. } else {
102. user.gender = null;
103. }
104. // Vai trò người dùng (chỉ admin nên được cập nhật)
105. if (data.roleId && typeof data.roleId === "number") {
106. user.roleId = data.roleId;
107. }
108. await user.save();
109. return resolve({
110. errCode: 0,
111. errMessage: 'Update user successfully'
112. });
113. } catch (error) {
114. console.error("Update User Error:", error.message || error);
115. return reject({
116. errCode: -1,
117. errMessage: 'Server error while updating user'
118. });
119. }
120. });
121. };
122. export default {
123. getAllUsers,
124. deleteUser,
125. updateUserData,
126. };

#### **authService.js**

1. import { resolveInclude } from "ejs";
2. import db from "../models/index"
3. import bcrypt from 'bcryptjs';
4. import { where } from "sequelize";
5. import { checkEmailExists, checkPhoneNumberExists, checkUserNameExists } from "./validators";
6. import { Op } from 'sequelize';
7. const loginUser = async (identifier, password) => {
8. try {
9. const userData = {};
10. const user = await db.User.findOne({
11. where: {
12. [Op.or]: [
13. { userName: identifier },
14. { email: identifier }
15. ]
16. },
17. attributes: ['id', 'userName', 'email', 'roleId', 'password'],
18. include: [
19. {
20. model: db.Role,
21. attributes: ['name']
22. }
23. ]
24. });
25. if (!user) {
26. return {
27. errCode: 1,
28. errMessage: "Wrong account or password!"
29. };
30. }
31. if (!password || !user.password) {
32. return {
33. errCode: 2,
34. errMessage: "Missing password!"
35. };
36. }
37. const match = bcrypt.compareSync(password, user.password);
38. if (!match) {
39. return {
40. errCode: 3,
41. errMessage: "Wrong password!"
42. };
43. }
44. const userPlain = user.get({ plain: true });
45. delete userPlain.password;
46. delete userPlain.Role;
47. return {
48. errCode: 0,
49. errMessage: "OK",
50. user: userPlain
51. };
52. } catch (err) {
53. console.error("Login error:", err);
54. return {
55. errCode: -1,
56. errMessage: "Internal server error"
57. };
58. }
59. };
60. let registerUser = (data) => {
61. return new Promise(async (resolve, reject) => {
62. try {
63. // validate roleId tồn tại thật sự
64. const validRole = await db.Role.findOne({ where: { id: parseInt(data.roleId) } });
65. if (!validRole) {
66. return resolve({ errCode: 5, errMessage: 'Invalid roleId' });
67. }
68. if (!validator.isEmail(data.email)) {
69. return resolve({ errCode: 6, errMessage: 'Invalid email format' });
70. }
71. const errors = [];
72. if (await checkEmailExists(data.email)) errors.push("Email already exists");
73. if (await checkUserNameExists(data.userName)) errors.push("Username already exists");
74. if (await checkPhoneNumberExists(data.phoneNumber)) errors.push("Phone number already exists");
75. if (errors.length > 0) {
76. return resolve({
77. errCode: 1,
78. errMessage: errors.join(", "),
79. });
80. }
81. // Nếu không trùng thì tạo user
82. let hashPassword = await hashUserPassword(data.password);
83. await db.User.create({
84. userName: data.userName,
85. email: data.email,
86. phoneNumber: data.phoneNumber,
87. password: hashPassword,
88. firstName: data.firstName,
89. lastName: data.lastName,
90. address: data.address,
91. gender: data.gender || null,
92. roleId: parseInt(data.roleId) || 2
93. });
94. resolve({
95. errCode: 0,
96. message: 'User created successfully!'
97. });
98. } catch (error) {
99. reject(error);
100. }
101. });
102. };
103. const hashUserPassword = async (password) => {
104. try {
105. const salt = await bcrypt.genSalt(10);
106. const hashed = await bcrypt.hash(password, salt);
107. return hashed;
108. } catch (e) {
109. throw e;
110. }
111. };
112. export default {
113. loginUser,
114. registerUser,
115. }

#### **backendService.js**

1. import bcrypt from 'bcrypt';
2. import db from '../models/index';
3. import { where } from 'sequelize';
4. const salt = bcrypt.genSaltSync(10);
5. let getAllUser = async () => {
6. try {
7. let users = await db.User.findAll({
8. attributes: { exclude: ['password'] },
9. include: [
10. {
11. model: db.Role,
12. attributes: ['name']
13. }
14. ],
15. raw: true,
16. nest: true
17. });
18. return users;
19. } catch (error) {
20. throw error;
21. }
22. };
23. let createNewUser = async (data) => {
24. try {
25. data.userName = data.userName?.trim();
26. data.email = data.email?.trim();
27. data.phoneNumber = data.phoneNumber?.trim();
28. data.fullName = data.fullName?.trim();
29. if (!data.userName || !data.password || !data.roleId) {
30. return {
31. errCode: 1,
32. errMessage: 'Missing required fields'
33. };
34. }
35. const loginExists = await db.User.findOne({ where: { userName: data.userName } });
36. if (loginExists) return { errCode: 2, errMessage: 'Username already exists' };
37. if (data.phoneNumber) {
38. const phoneExists = await db.User.findOne({ where: { phoneNumber: data.phoneNumber } });
39. if (phoneExists) return { errCode: 3, errMessage: 'Phone number already exists' };
40. }
41. if (data.email) {
42. const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
43. if (!emailRegex.test(data.email)) {
44. return { errCode: 6, errMessage: 'Invalid email format' };
45. }
46. }
47. const emailExists = data.email ? await db.User.findOne({ where: { email: data.email } }) : null;
48. if (emailExists) return { errCode: 4, errMessage: 'Email already exists' };
49. const hashedPassword = await bcrypt.hash(data.password, salt);
50. await db.User.create({
51. userName: data.userName,
52. password: hashedPassword,
53. email: data.email || null,
54. phoneNumber: data.phoneNumber || null,
55. fullName: data.fullName || null,
56. gender: data.gender || null,
57. roleId: parseInt(data.roleId),
58. });
59. return { errCode: 0, errMessage: 'Create new user succeed' };
60. } catch (error) {
61. throw error;
62. }
63. };
64. let getUserInfoById = async (userId) => {
65. return new Promise(async (resolve, reject) => {
66. try {
67. let user = await db.User.findOne({
68. where: { id: userId },
69. raw: true
70. })
71. if (user) {
72. resolve(user)
73. } else {
74. resolve({});
75. }
76. } catch (error) {
77. reject(error);
78. }
79. })
80. }
81. let updateUserData = async (data) => {
82. try {
83. if (!data.id) return { errCode: 1, errMessage: 'Missing user ID' };
84. const user = await db.User.findOne({ where: { id: data.id } });
85. if (!user) return { errCode: 2, errMessage: 'User not found' };
86. // Email
87. if (data.email !== undefined) {
88. const emailValue = data.email.trim();
89. if (emailValue === "") {
90. user.email = null;
91. } else if (emailValue !== user.email) {
92. const emailExists = await db.User.findOne({
93. where: {
94. email: emailValue,
95. id: { [db.Sequelize.Op.ne]: data.id }
96. }
97. });
98. if (emailExists) return { errCode: 3, errMessage: 'Email already in use' };
99. user.email = emailValue;
100. }
101. }
102. // Phone number
103. if (data.phoneNumber !== undefined) {
104. const phoneValue = data.phoneNumber.trim();
105. if (phoneValue === "") {
106. user.phoneNumber = null;
107. } else {
108. const phonePattern = /^0\d{9}$/;
109. if (!phonePattern.test(phoneValue)) {
110. return { errCode: 5, errMessage: 'Invalid phone number. Must be 10 digits and start with 0.' };
111. }
112. user.phoneNumber = phoneValue;
113. }
114. }
115. // Full name
116. user.fullName = data.fullName?.trim() || null;
117. // Gender (cẩn thận vì 0 là falsy)
118. if (data.gender !== undefined) {
119. user.gender = data.gender;
120. }
121. // Role ID (nếu không muốn cho update, có thể bỏ)
122. if (data.roleId !== undefined) {
123. user.roleId = data.roleId;
124. }
125. await user.save();
126. return { errCode: 0, errMessage: 'Update successful' };
127. } catch (error) {
128. throw error;
129. }
130. };
131. let deleteUserById = async (userId) => {
132. try {
133. const user = await db.User.findOne({ where: { id: userId } });
134. if (!user) return { errCode: 1, errMessage: 'User not found' };
135. await user.destroy();
136. return { errCode: 0, errMessage: 'User deleted successfully' };
137. } catch (error) {
138. throw error;
139. }
140. };
141. module.exports = {
142. getAllUser: getAllUser,
143. createNewUser: createNewUser,
144. getUserInfoById: getUserInfoById,
145. updateUserData: updateUserData,
146. deleteUserById: deleteUserById,
147. };

#### **userService.js**

1. import db from '../models';
2. let getUserInfoById = async (userId) => {
3. try {
4. const user = await db.User.findOne({
5. where: { id: userId },
6. attributes: { exclude: ['password'] },
7. include: [
8. {
9. model: db.Role,
10. attributes: ['name']
11. }
12. ]
13. });
14. return user || null;
15. } catch (error) {
16. throw new Error('Database error when fetching user info');
17. }
18. };
19. export default {
20. getUserInfoById: getUserInfoById,
21. };

#### **validators.js**

1. import db from "../models/index"
2. export const checkEmailExists = async (userEmail) => {
3. return new Promise(async (resolve, reject) => {
4. try {
5. let user = await db.User.findOne({
6. where: { email: userEmail }
7. });
8. if (user) {
9. resolve(true);
10. } else {
11. resolve(false);
12. };
13. } catch (e) {
14. reject(e);
15. }
16. });
17. };
18. export const checkPhoneNumberExists = async (phoneNumber) => {
19. try {
20. let user = await db.User.findOne({
21. where: { phoneNumber: phoneNumber }
22. });
23. return user ? true : false;
24. } catch (error) {
25. console.log("Error checking phoneNumber:", error);
26. return false;
27. }
28. };
29. export const checkUserNameExists = async (userName) => {
30. try {
31. let user = await db.User.findOne({
32. where: { userName: userName }
33. });
34. return user ? true : false;
35. } catch (error) {
36. console.log("Error checking userName:", error);
37. return false;
38. }
39. };

### **/src/utils/**

#### **sendResponse.js**

1. const sendResponse = (res, {
2. status = 200,
3. errCode = 0,
4. message = 'Success',
5. data = null,
6. token = null
7. }) => {
8. const response = { errCode, message };
9. if (data !== null) response.data = data;
10. if (token) response.token = token;
11. return res.status(status).json(response);
12. };
13. export default sendResponse;

### **/src/views/**

#### **createNewUser.ejs**

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport" content="width=device-width, initial-scale=1.0">
6. <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet">
7. <title>Create User</title>
8. <style>
9. body {
10. background-color: #f8f9fa;
11. font-family: Arial, sans-serif;
12. padding: 30px 0;
13. }
14. .container {
15. background-color: #ffffff;
16. border-radius: 10px;
17. box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
18. padding: 40px;
19. max-width: 600px;
20. margin: auto;
21. }
22. h2 {
23. text-align: center;
24. color: #495057;
25. margin-bottom: 30px;
26. }
27. .form-control,
28. .form-select {
29. border-radius: 10px;
30. font-size: 16px;
31. box-shadow: 0 1px 2px rgba(0, 0, 0, 0.1);
32. }
33. .btn-primary {
34. background-color: #007bff;
35. border: none;
36. border-radius: 10px;
37. padding: 10px 20px;
38. font-size: 16px;
39. width: 100%;
40. margin-top: 20px;
41. transition: background-color 0.3s ease;
42. }
43. .btn-primary:hover {
44. background-color: #0056b3;
45. cursor: pointer;
46. }
47. label {
48. font-weight: bold;
49. color: #495057;
50. }
51. .form-group input[type="file"] {
52. padding: 0;
53. }
54. .btn-group {
55. margin-bottom: 20px;
56. display: flex;
57. justify-content: space-between;
58. }
59. .btn-secondary {
60. width: 48%;
61. }
62. </style>
63. </head>
64. <body>
65. <div class="btn-group">
66. <button onclick="location.href='/'" class="btn btn-secondary">Home</button>
67. <button onclick="location.href='/user-manager'" class="btn btn-secondary">User Manager</button>
68. </div>
69. <div class="container">
70. <h2>Create New User</h2>
71. <% if (typeof errorMessage !=='undefined' && errorMessage) { %>
72. <div class="alert alert-danger">
73. <%= errorMessage %>
74. </div>
75. <% } %>
76. <form action="/post-create-new-user" method="POST" onsubmit="return">
77. <p><small style="color: red;">\* Required</small></p>
78. <div class="mb-3">
79. <label for="userName">Username <span style="color: red;">\*</span></label>
80. <input type="text" class="form-control" name="userName" required
81. value="<%= oldInput?.userName || '' %>">
82. </div>
83. <div class="mb-3">
84. <label for="password">Password <span style="color: red;">\*</span></label>
85. <input type="password" class="form-control" name="password" required>
86. <% if (passwordError) { %>
87. <div class="text-danger mt-1">
88. <%= passwordError %>
89. </div>
90. <% } %>
91. </div>
92. <div class="mb-3">
93. <label for="email">Email</label>
94. <input type="email" class="form-control" name="email" value="<%= oldInput?.email || '' %>">
95. <% if (emailError) { %>
96. <div class="text-danger mt-1">
97. <%= emailError %>
98. </div>
99. <% } %>
100. </div>
101. <div class="mb-3">
102. <label for="phoneNumber">Phone number</label>
103. <input type="text" class="form-control" name="phoneNumber" pattern="^0\d{9}$"
104. value="<%= oldInput?.phoneNumber || '' %>">
105. <% if (phoneError) { %>
106. <div class="text-danger mt-1">
107. <%= phoneError %>
108. </div>
109. <% } %>
110. </div>
111. <div class="mb-3">
112. <label for="fullName">Full name</label>
113. <input type="text" class="form-control" name="fullName" value="<%= oldInput?.fullName || '' %>">
114. </div>
115. <div class="mb-3">
116. <label for="gender">Gender</label>
117. <select class="form-select" name="gender">
118. <option value="" disabled <%=!oldInput?.gender ? 'selected' : '' %>>Choose</option>
119. <option value="M" <%=oldInput?.gender==='M' ? 'selected' : '' %>>Male</option>
120. <option value="F" <%=oldInput?.gender==='F' ? 'selected' : '' %>>Female</option>
121. </select>
122. </div>
123. <div class="mb-3">
124. <label for="roleId">Role <span style="color: red;">\*</span></label>
125. <select class="form-select" name="roleId" id="roleId" required>
126. <option value="" disabled <%=!oldInput?.roleId ? 'selected' : '' %>>Choose</option>
127. <option value="1" <%=oldInput?.roleId==1 ? 'selected' : '' %>>Admin</option>
128. <option value="2" <%=oldInput?.roleId==2 ? 'selected' : '' %>>User</option>
129. </select>
130. </div>
131. <button type="submit" class="btn btn-primary">Create</button>
132. </form>
133. </div>
134. </body>
135. </html>

#### **homePage.ejs**

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport" content="width=device-width, initial-scale=1.0">
6. <title>Trang quản trị - Backend</title>
7. <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet">
8. </head>
9. <style>
10. /\* Reset mặc định \*/
11. body,
12. h1,
13. h4,
14. p {
15. margin: 0;
16. padding: 0;
17. box-sizing: border-box;
18. }
19. /\* Toàn trang \*/
20. body {
21. font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
22. background-color: darkgray;
23. padding: 40px 20px;
24. color: #333;
25. }
26. /\* Tiêu đề \*/
27. h1.title {
28. text-align: center;
29. color: #2c3e50;
30. margin-bottom: 10px;
31. font-size: 4rem;
32. }
33. h6.description {
34. text-align: center;
35. color: #2c3e50;
36. margin-bottom: 60px;
37. font-size: 2rem;
38. }
39. /\* Grid layout cho menu \*/
40. .menu-grid {
41. display: grid;
42. grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
43. gap: 20px;
44. max-width: 1000px;
45. margin: 0 auto;
46. }
47. .menu-item {
48. background-color: #fff;
49. border-radius: 12px;
50. padding: 24px;
51. box-shadow: 0 4px 12px rgba(0, 0, 0, 0.06);
52. transition: transform 0.2s ease, box-shadow 0.3s ease;
53. cursor: pointer;
54. border-left: 5px solid #0d6efd;
55. }
56. .menu-item:hover {
57. transform: translateY(-5px);
58. box-shadow: 0 8px 20px rgba(0, 0, 0, 0.12);
59. border-left-color: #0a58ca;
60. }
61. .menu-item h4 {
62. font-size: 1.2rem;
63. margin-bottom: 8px;
64. color: #0d6efd;
65. }
66. .menu-item p {
67. font-size: 0.95rem;
68. color: #555;
69. }
70. </style>
71. <body>
72. <h1 class="title">ADMIN</h1>
73. <h6 class="description">Website Management</h6>
74. <div class="menu-grid">
75. <div class="menu-item" onclick="location.href='/user-manager'">
76. <h4>User Manager</h4>
77. <p>View, edit, delete user accounts</p>
78. </div>
79. <div class="menu-item" onclick="location.href='/create-new-user'">
80. <h4>Create New User</h4>
81. <p>Add new user with role</p>
82. </div>
83. <!-- 🔽 Có thể thêm nhiều mục khác ở đây -->
84. <div class="menu-item" onclick="alert('Function under construction...')">
85. <h4>Product Manager</h4>
86. <p>View, create, edit, delete products</p>
87. </div>
88. <div class="menu-item" onclick="alert('Function under construction...')">
89. <h4>System statistics</h4>
90. <p>Report by role or time</p>
91. </div>
92. </div>
93. </body>
94. </html>

#### **updateUser.ejs**

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport" content="width=device-width, initial-scale=1.0">
6. <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet">
7. <title>Update User</title>
8. <style>
9. body {
10. background-color: #f8f9fa;
11. font-family: Arial, sans-serif;
12. padding: 30px 0;
13. }
14. .container {
15. background-color: #ffffff;
16. border-radius: 10px;
17. box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
18. padding: 40px;
19. max-width: 600px;
20. margin: auto;
21. }
22. h2 {
23. text-align: center;
24. color: #495057;
25. margin-bottom: 30px;
26. }
27. .form-control,
28. .form-select {
29. border-radius: 10px;
30. font-size: 16px;
31. box-shadow: 0 1px 2px rgba(0, 0, 0, 0.1);
32. }
33. .btn-primary {
34. background-color: #007bff;
35. border: none;
36. border-radius: 10px;
37. padding: 10px 20px;
38. font-size: 16px;
39. width: 100%;
40. margin-top: 20px;
41. transition: background-color 0.3s ease;
42. }
43. .btn-primary:hover {
44. background-color: #0056b3;
45. cursor: pointer;
46. }
47. label {
48. font-weight: bold;
49. color: #495057;
50. }
51. .form-group input[type="file"] {
52. padding: 0;
53. }
54. .btn-group {
55. margin-bottom: 20px;
56. display: flex;
57. justify-content: space-between;
58. }
59. .btn-secondary {
60. width: 48%;
61. }
62. </style>
63. </head>
64. <body>
65. <div class="btn-group">
66. <button onclick="location.href='/'" class="btn btn-secondary">Home</button>
67. <button onclick="location.href='/manager-user'" class="btn btn-secondary">User Manager</button>
68. </div>
69. <div class="container">
70. <h2>Update User</h2>
71. <% if (typeof error !=='undefined' && error) { %>
72. <div class="alert alert-danger">
73. <%= error %>
74. </div>
75. <% } %>
76. <form action="/update-user" method="POST" onsubmit="return validateForm()">
77. <input type="hidden" name="id" value="<%= user.id %>">
78. <p><small style="color: red;">\* Bắt buộc</small></p>
79. <div class="mb-3">
80. <label for="userName" class="form-label">User Name</label>
81. <input type="text" class="form-control" name="userName" value="<%= user.userName %>" readonly>
82. </div>
83. <div class="mb-3">
84. <label for="email" class="form-label">Email</label>
85. <input type="email" class="form-control" name="email" id="email" value="<%= user.email %>">
86. <% if (typeof emailError !=='undefined' && emailError) { %>
87. <div class="text-danger mt-1">
88. <%= emailError %>
89. </div>
90. <% } %>
91. </div>
92. <div class="mb-3">
93. <label for="phoneNumber" class="form-label">Phone number</label>
94. <input type="text" class="form-control" name="phoneNumber" id="phoneNumber"
95. value="<%= user.phoneNumber %>">
96. <% if (typeof phoneError !=='undefined' && phoneError) { %>
97. <div class="text-danger mt-1">
98. <%= phoneError %>
99. </div>
100. <% } %>
101. </div>
102. <div class="mb-3">
103. <label for="fullName" class="form-label">Full name</label>
104. <input type="text" class="form-control" name="fullName" value="<%= user.fullName %>">
105. </div>
106. <div class="mb-3">
107. <label for="gender" class="form-label">Gender</label>
108. <select class="form-select" name="gender">
109. <option value="1" <%=user.gender===true || user.gender===1 ? 'selected' : '' %>>Nam</option>
110. <option value="0" <%=user.gender===false || user.gender===0 ? 'selected' : '' %>>Nữ</option>
111. </select>
112. </div>
113. <div class="mb-3">
114. <label for="roleId" class="form-label">Role</label>
115. <select class="form-select" name="roleId" disabled>
116. <option value="1" <%=user.roleId==1 ? 'selected' : '' %>>Admin</option>
117. <option value="0" <%=user.roleId==0 ? 'selected' : '' %>>User</option>
118. </select>
119. <input type="hidden" name="roleId" value="<%= user.roleId %>">
120. </div>
121. <button type="submit" class="btn btn-primary">Update</button>
122. </form>
123. </div>
124. <script>
125. function validateForm() {
126. document.querySelectorAll('.text-danger').forEach(el => el.innerText = '');
127. const email = document.getElementById("email").value.trim();
128. const phone = document.getElementById("phoneNumber").value.trim();
129. let isValid = true;
130. const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
131. if (email && !emailPattern.test(email)) {
132. const emailError = document.createElement('div');
133. emailError.className = 'text-danger mt-1';
134. emailError.innerText = "Email không hợp lệ.";
135. document.getElementById("email").parentNode.appendChild(emailError);
136. isValid = false;
137. }
138. const phonePattern = /^0\d{9}$/;
139. if (phone && !phonePattern.test(phone)) {
140. const phoneError = document.createElement('div');
141. phoneError.className = 'text-danger mt-1';
142. phoneError.innerText = "Số điện thoại phải gồm 10 số và bắt đầu bằng số 0.";
143. document.getElementById("phoneNumber").parentNode.appendChild(phoneError);
144. isValid = false;
145. }
146. return isValid;
147. }
148. </script>
149. </body>
150. </html>

#### **userManager.ejs**

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport" content="width=device-width, initial-scale=1.0">
6. <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet">
7. <title>Manager User</title>
8. <style>
9. .table th,
10. .table td {
11. vertical-align: middle;
12. padding: 10px 15px;
13. text-align: center;
14. }
15. .table-hover tbody tr:hover {
16. background-color: #f1f1f1;
17. transition: background-color 0.3s ease;
18. }
19. .btn-outline-warning,
20. .btn-outline-danger {
21. font-weight: bold;
22. padding: 10px 15px;
23. border-radius: 5px;
24. transition: all 0.3s ease;
25. }
26. .btn-outline-warning:hover {
27. background-color: #ffc107;
28. border-color: #ffc107;
29. color: white;
30. }
31. .btn-outline-danger:hover {
32. background-color: #dc3545;
33. border-color: #dc3545;
34. color: white;
35. }
36. .table-container {
37. background-color: #f8f9fa;
38. padding: 20px;
39. border-radius: 10px;
40. box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
41. margin-top: 30px;
42. }
43. .table-sm th,
44. .table-sm td {
45. padding: 5px 10px;
46. }
47. </style>
48. </head>
49. <body>
50. <div class="container table-container">
51. <h2 class="my-4 text-center text-dark">Manager User</h2>
52. <div class="d-flex justify-content-between mt-4">
53. <button onclick="location.href='/'" class="btn btn-secondary">HomePage</button>
54. <button onclick="location.href='/create-new-user'" class="btn btn-success">Tạo tài khoản</button>
55. </div>
56. <table class="table table-dark table-hover table-sm">
57. <thead>
58. <tr>
59. <th>Username</th>
60. <th>Email</th>
61. <th>Phone</th>
62. <th>Full name</th>
63. <th>Gender</th>
64. <th>Role</th>
65. <th>Action</th>
66. </tr>
67. </thead>
68. <tbody>
69. <% for (let user of dataTable) { %>
70. <tr>
71. <td>
72. <%= user.userName || 'Null' %>
73. </td>
74. <td>
75. <%= user.email || 'Null' %>
76. </td>
77. <td>
78. <%= user.phoneNumber || 'Null' %>
79. </td>
80. <td>
81. <%= user.fullName || 'Null' %>
82. </td>
83. <td>
84. <% if (user.gender==='M' ) { %>Nam
85. <% } else if (user.gender==='F' ) { %>Nữ
86. <% } else { %>Null<% } %>
87. </td>
88. <td>
89. <%= user.Role?.name || 'Null' %>
90. </td>
91. <td>
92. <a href="/edit-user?id=<%= user.id %>" class="btn btn-outline-warning">Edit</a>
93. <a href="javascript:void(0)" onclick="confirmDelete(<%= user.id %>, '<%= user.loginName %>')"
94. class="btn btn-outline-danger">Delete</a>
95. </td>
96. </tr>
97. <% } %>
98. </tbody>
99. </table>
100. </div>
101. <script>
102. function confirmDelete(userId, userName) {
103. if (confirm("Bạn có chắc muốn xóa tài khoản '" + userName + "' không?")) {
104. window.location.href = '/delete-user?id=' + userId;
105. }
106. }
107. </script>
108. </body>
109. </html>

### **/src/.gitignore/**

#### **.gitignore**

1. /node\_modules
2. /vendor
3. /.idea
4. .idea/
5. .env

### **/src/server**

#### **server.js**

1. import path from 'path';
2. import { config } from 'dotenv';
3. import express from "express";
4. import bodyParser from "body-parser";
5. import cors from "cors";
6. import viewEngine from "./config/viewEngine";
7. import initRoutes from './routes/web';
8. import connectDB from './config/connectDB';
9. // Load biến môi trường từ .env
10. config({ path: path.resolve(\_\_dirname, '../.env') });
11. const app = express();
12. // ✅ Cấu hình CORS cho phép frontend truy cập backend
13. app.use(cors({
14. origin: 'http://localhost:3000',
15. methods: ['GET', 'POST', 'PUT', 'DELETE'],
16. credentials: true
17. }));
18. // ✅ Bổ sung header để cho phép Authorization
19. app.use((req, res, next) => {
20. res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept, Authorization');
21. next();
22. });
23. // ✅ Parse body của request
24. app.use(bodyParser.json());
25. app.use(bodyParser.urlencoded({ extended: true }));
26. // ✅ Khởi tạo view engine nếu bạn đang dùng EJS cho các route test
27. viewEngine(app);
28. // ✅ Khai báo các route API/web
29. initRoutes(app);
30. // ✅ Kết nối cơ sở dữ liệu
31. connectDB();
32. // ✅ Chạy server trên cổng .env hoặc 8080 mặc định
33. const port = process.env.PORT || 5050;
34. app.listen(port, () => {
35. console.log("Dự án NguyenLien đã chạy THÀNH CÔNG trên CỔNG " + port + " !!!");
36. });

## **/.babelrc**

### **.babelrc**

1. {
2. "presets": [
3. "@babel/preset-env"
4. ]
5. }

## **/.env**

### **.env**

1. PORT=8080
2. NODE\_ENV=development
3. JWT\_SECRET=nguyenlien-secret-key
4. JWT\_EXPIRES=1d

## **/pakage.json**

### **package.json**

1. {
2. "name": "backend-nodejs-nguyenlienproject",
3. "version": "1.0.0",
4. "main": "server.js",
5. "scripts": {
6. "start": "nodemon --exec ./node\_modules/.bin/babel-node src/server.js",
7. "dev": "nodemon src/server.js",
8. "test": "echo \"No test defined\"",
9. "migrate": "sequelize-cli db:migrate",
10. "seed": "sequelize-cli db:seed:all"
11. },
12. "author": "PhamAnhKhoi",
13. "license": "ISC",
14. "description": "manager",
15. "dependencies": {
16. "bcrypt": "^6.0.0",
17. "bcryptjs": "^3.0.2",
18. "body-parser": "^1.19.0",
19. "cors": "^2.8.5",
20. "dotenv": "^8.6.0",
21. "ejs": "^3.1.5",
22. "express": "^4.17.1",
23. "jsonwebtoken": "^9.0.2",
24. "mysql2": "^3.14.1",
25. "sequelize": "^6.37.7",
26. "slugify": "^1.6.6",
27. "validator": "^13.15.0"
28. },
29. "devDependencies": {
30. "@babel/core": "^7.12.10",
31. "@babel/node": "^7.12.10",
32. "@babel/preset-env": "^7.12.10",
33. "nodemon": "^2.0.7",
34. "sequelize-cli": "^6.6.3"
35. }
36. }

# **Frontend – React / JS**

## **public/**

## **server/**

## **src/**

### **src/assets/**

### **src/components/**

#### **src/components/footer/**

##### **Footer.js/**

import React, { Component } from 'react';

import { connect } from 'react-redux';

import \* as actions from "../../store/actions";

import Navigator from '../../components/Navigator';

import './Footer.scss';

class Footer extends Component {

   render() {

      const { processLogout } = this.props;

      return (

         <div className="footer-container">

            Hello, this is the footer section.

         </div>

      );

   }

}

const mapStateToProps = state => {

   return {

      isLoggedIn: state.admin.isLoggedIn

   };

};

const mapDispatchToProps = dispatch => {

   return {

      processLogout: () => dispatch(actions.processLogout()),

   };

};

export default connect(mapStateToProps, mapDispatchToProps)(Footer);

##### **Footer.scss**

#### **src/components/formating/**

##### **FormattedDate.js/**

import React, { Component } from 'react';

import moment from 'moment';

/\*\* For valid format please see <a href="https://momentjs.com/docs/#/displaying/">Moment format options</a> \*/

const dateFormat = 'DD/MM/YYYY';

class FormattedDate extends Component {

    render() {

        const { format, value, ...otherProps } = this.props;

        var dFormat = format ? format : dateFormat;

        const formattedValue = value ? moment.utc(value).format(dFormat) : null;

        return (

            <span {...otherProps}>{formattedValue}</span>

        );

    }

}

export default FormattedDate;

##### **index.js**

export { default as FormattedDate } from './FormattedDate';

#### **src/components/feader/**

##### **Header.js**

import React, { Component } from 'react';

import { connect } from 'react-redux';

import \* as actions from "../../store/actions";

import Navigator from '../../components/Navigator';

import { adminMenu } from './menuApp';

import './Header.scss';

class Header extends Component {

    render() {

        const { processLogout } = this.props;

        return (

            <div className="header-container">

                {/\* thanh navigator \*/}

                <div className="header-tabs-container">

                    <Navigator menus={adminMenu} />

                </div>

                {/\* nút logout \*/}

                <div className="btn btn-logout" onClick={processLogout}>

                    <i className="fas fa-sign-out-alt"></i>

                </div>

            </div>

        );

    }

}

const mapStateToProps = state => {

    return {

        isLoggedIn: state.admin.isLoggedIn

    };

};

const mapDispatchToProps = dispatch => {

    return {

        processLogout: () => dispatch(actions.processLogout()),

    };

};

export default connect(mapStateToProps, mapDispatchToProps)(Header);

##### **Header.scss**

@import "../../styles/common.scss";

.header-container {

    z-index: 99;

    display: flex;

    justify-content: space-between;

    background-color: $colormain;

    color: $common-text-color;

    height: 40px;

    position: relative;

    .btn-logout {

        color: $common-text-color;

        line-height: 40px;

        height: 40px;

        padding: 0 10px;

        &:hover {

            background-color: darken($colormain, 5);

        }

        i {

            font-size: $base-size + 1px;

        }

    }

}

##### **menuApp.js**

export const adminMenu = [

    { //hệ thống

        name: 'menu.system.header', menus: [

            {

                name: 'menu.system.system-administrator.header',

                subMenus: [

                    { name: 'menu.system.system-administrator.user-manage', link: '/system/user-manage' },

                    { name: 'menu.system.system-administrator.product-manage', link: '/system/product-manage' },

                    { name: 'menu.system.system-administrator.register-package-group-or-account', link: '/system/register-package-group-or-account' },

                ]

            },

            // { name: 'menu.system.system-parameter.header', link: '/system/system-parameter' },

        ]

    },

];

#### **src/components/input/**

##### **DatePicker.js**

import React, { Component } from 'react';

import Flatpickr from 'react-flatpickr';

import moment from 'moment';

import { KeyCodeUtils } from "../../utils";

import './DatePicker.scss';

// const CustomInput = ({ value, defaultValue, inputRef, onInputChange, onInputBlur, ...props }) => {

//     return <input {...props} className='custom-form-control custom-date-input' defaultValue={defaultValue} ref={inputRef}

//         onChange={onInputChange}

//         onBlur={onInputBlur}

//     />;

// };

class DatePicker extends Component {

    flatpickrNode = null;

    nodeRef = element => {

        this.flatpickr = element && element.flatpickr;

        this.flatpickrNode = element && element.node;

        if (this.flatpickrNode) {

            this.flatpickrNode.addEventListener('blur', this.handleBlur);

            this.flatpickrNode.addEventListener('keydown', this.handlerKeyDown);

        }

    };

    handlerKeyDown = (event) => {

        const keyCode = event.which || event.keyCode;

        if (keyCode === KeyCodeUtils.ENTER) {

            event.preventDefault();

            const { onChange } = this.props;

            const value = event.target.value;

            // Take the blur event and process the string value

            const valueMoment = moment(value, 'DD/MM/YYYY');

            onChange([valueMoment.toDate(), valueMoment.toDate()]);

        }

    }

    componentWillUnmount() {

        if (this.flatpickrNode) {

            this.flatpickrNode.removeEventListener('blur', this.handleBlur);

            this.flatpickrNode.removeEventListener('keydown', this.handlerKeyDown);

        }

    }

    handleBlur = (event) => {

        const { onChange } = this.props;

        const value = event.target.value;

        // Take the blur event and process the string value

        event.preventDefault();

        const valueMoment = moment(value, 'DD/MM/YYYY');

        onChange([valueMoment.toDate(), valueMoment.toDate()]);

    };

    onOpen = () => {

        if (this.flatpickrNode) {

            this.flatpickrNode.blur();

        }

    }

    close() {

        this.flatpickr.close();

    }

    checkDateValue = (str, max) => {

        if (str.charAt(0) !== '0' || str === '00') {

            var num = parseInt(str);

            if (isNaN(num) || num <= 0 || num > max) num = 1;

            str = num > parseInt(max.toString().charAt(0)) && num.toString().length === 1 ? '0' + num : num.toString();

        };

        return str;

    }

    // autoFormatonBlur = (value) => {

    //     var input = value;

    //     var values = input.split('/').map(function (v, i) {

    //         return v.replace(/\D/g, '')

    //     });

    //     var output = '';

    //     if (values.length == 3) {

    //         var year = values[2].length !== 4 ? parseInt(values[2]) + 2000 : parseInt(values[2]);

    //         var month = parseInt(values[0]) - 1;

    //         var day = parseInt(values[1]);

    //         var d = new Date(year, month, day);

    //         if (!isNaN(d)) {

    //             //document.getElementById('result').innerText = d.toString();

    //             var dates = [d.getMonth() + 1, d.getDate(), d.getFullYear()];

    //             output = dates.map(function (v) {

    //                 v = v.toString();

    //                 return v.length == 1 ? '0' + v : v;

    //             }).join(' / ');

    //         };

    //     };

    //     // this.value = output;

    //     return output;

    // }

    autoFormatOnChange = (value, seperator) => {

        var input = value;

        let regexForDeleting = new RegExp(`\\D\\${seperator}$`);

        //if (/\D\/$/.test(input)) input = input.substr(0, input.length - 3); // dat.nt: Xóa thêm 1 ký tự nếu xóa dấu cách sau / (VD: 12 / 12 /=> 12 / 1)

        if (regexForDeleting.test(input)) input = input.substr(0, input.length - 3);

        var values = input.split(seperator).map(function (v) {

            return v.replace(/\D/g, '')

        });

        if (values[0]) values[0] = this.checkDateValue(values[0], 31);

        if (values[1]) values[1] = this.checkDateValue(values[1], 12);

        var output = values.map(function (v, i) {

            return v.length === 2 && i < 2 ? v + ' ' + seperator + ' ' : v;

        });

        return output.join('').substr(0, 14);

    }

    onInputChange = (e) => {

        if (this.DISPLAY\_FORMAT === this.DATE\_FORMAT\_AUTO\_FILL) {

            let converted = this.autoFormatOnChange(e.target.value, this.SEPERATOR);

            e.target.value = converted;

        }

    }

    onInputBlur = (e) => {

    }

    //dat.nt : Auto Fill cho dạng ngăn cách và format cụ thể (seperator có thể dc thay thế)

    SEPERATOR = "/";

    DATE\_FORMAT\_AUTO\_FILL = "d/m/Y"; // Format không thay đổi

    // dat.nt : Format ngày hiển thị

    DISPLAY\_FORMAT = "d/m/Y";

    render() {

        const { value, onChange, minDate, onClose, ...otherProps } = this.props;

        const options = {

            dateFormat: this.DISPLAY\_FORMAT,

            allowInput: true,

            disableMobile: true,

            onClose: onClose,

            onOpen: this.onOpen

        };

        if (minDate) {

            options.minDate = minDate;

        }

        return (

            <Flatpickr

                ref={this.nodeRef}

                value={value}

                onChange={onChange}

                options={options}

                // render={

                //     ({ defaultValue, value, ...props }, ref) => {

                //         return <CustomInput defaultValue={defaultValue} inputRef={ref} onInputChange={this.onInputChange} onInputBlur={this.onInputBlur} />

                //     }

                // }

                {...otherProps}

            />

        );

    }

}

export default DatePicker;

##### **DatePicker.scss**

@import "../../styles/common";

$bezier: cubic-bezier(0.23, 1, 0.32, 1);

$slideTime: 400ms;

$daySize: 39px;

$padding: $daySize / 16;

$dayMargin: 2px;

$daysWidth: $daySize \* 7 + $dayMargin \* 14 + $padding \* 2 + 2;

$calendarWidth: $daysWidth;

$monthNavHeight: 28px !default;

$weekdaysHeight: 28px !default;

$timeHeight: 40px;

$disabledBorderColor: transparent;

$selectedDayForeground: #fff;

$selectedDayBackground: $colormain;

input {

  &.custom-date-input {

    word-spacing:-3px

  }

}

@-webkit-keyframes fpFadeInDown {

  from {

    opacity: 0;

    transform: translate3d(0, -20px, 0);

  }

  to {

    opacity: 1;

    transform: translate3d(0, 0, 0);

  }

}

@-moz-keyframes fpFadeInDown {

  from {

    opacity: 0;

    transform: translate3d(0, -20px, 0);

  }

  to {

    opacity: 1;

    transform: translate3d(0, 0, 0);

  }

}

@-ms-keyframes fpFadeInDown {

  from {

    opacity: 0;

    transform: translate3d(0, -20px, 0);

  }

  to {

    opacity: 1;

    transform: translate3d(0, 0, 0);

  }

}

@-o-keyframes fpFadeInDown {

  from {

    opacity: 0;

    transform: translate3d(0, -20px, 0);

  }

  to {

    opacity: 1;

    transform: translate3d(0, 0, 0);

  }

}

@keyframes fpFadeInDown {

  from {

    opacity: 0;

    transform: translate3d(0, -20px, 0);

  }

  to {

    opacity: 1;

    transform: translate3d(0, 0, 0);

  }

}

.flatpickr-calendar {

  background: transparent;

  opacity: 0;

  display: none;

  text-align: center;

  visibility: hidden;

  padding: 0;

  animation: none;

  direction: ltr;

  border: 0;

  font-size: 14px;

  line-height: 24px;

  border-radius: 5px;

  position: absolute;

  width: $calendarWidth;

  box-sizing: border-box;

  touch-action: manipulation;

  &.open, &.inline {

    opacity: 1;

    max-height: 640px;

    visibility: visible;

  }

  &.open {

    display: inline-block;

    z-index: 99999;

  }

  &.animate.open {

    animation: fpFadeInDown 300ms $bezier;

  }

  &.inline {

    display: block;

    position: relative;

    top: 2px;

  }

  &.static {

    position: absolute;

    top: calc(100% + 2px);

    &.open {

      z-index: 999;

      display: block;

    }

  }

  &.multiMonth {

    .flatpickr-days .dayContainer:nth-child(n+1) {

      & .flatpickr-day.inRange:nth-child(7n+7) {

        box-shadow: none !important;

      }

    }

    .flatpickr-days .dayContainer:nth-child(n+2) {

      & .flatpickr-day.inRange:nth-child(7n+1) {

        box-shadow: -2px 0 0 #e6e6e6, 5px 0 0 #e6e6e6;

      }

    }

  }

  .hasWeeks, .hasTime {

    .dayContainer {

      border-bottom: 0;

      border-bottom-right-radius: 0;

      border-bottom-left-radius: 0;

    }

  }

  &.showTimeInput.hasTime {

    .flatpickr-time {

      height: $timeHeight;

    }

  }

  &.noCalendar.hasTime {

    .flatpickr-time {

      height: auto;

    }

  }

  &:before, &:after {

    position: absolute;

    display: block;

    pointer-events: none;

    border: solid transparent;

    content: '';

    height: 0;

    width: 0;

    left: 22px;

  }

  &.rightMost {

    &:before, &:after {

      left: auto;

      right: 22px;

    }

  }

  &:before {

    border-width: 5px;

    margin: 0 -5px;

  }

  &:after {

    border-width: 4px;

    margin: 0 -4px;

  }

  &.arrowTop {

    &:before, &:after {

      bottom: 100%;

    }

  }

  &.arrowBottom {

    &:before, &:after {

      top: 100%;

    }

  }

  &:focus {

    outline: 0;

  }

  .flatpickr-months {

    display: flex;

    .flatpickr-month {

      height: $monthNavHeight;

      line-height: 1;

      text-align: center;

      position: relative;

      user-select: none;

      overflow: hidden;

      flex: 1;

    }

    .flatpickr-prev-month, .flatpickr-next-month {

      text-decoration: none;

      cursor: pointer;

      position: absolute;

      top: 0;

      line-height: 16px;

      height: $monthNavHeight;

      padding: 10px;

      z-index: 3;

      &.disabled {

        display: none;

      }

      i {

        position: relative;

      }

      &.flatpickr-prev-month {

        left: 0;

      }

      &.flatpickr-next-month {

        right: 0;

      }

      svg {

        width: 14px;

        height: 14px;

        path {

          transition: fill 0.1s;

        }

      }

    }

  }

  background: $date-picker-bg;

  box-shadow: 1px 0 0 $date-picker-border, -1px 0 0 $date-picker-border, 0 1px 0 $date-picker-border, 0 -1px 0 $date-picker-border, 0 3px 13px rgba(black, 0.08);

  &.arrowTop {

    &:before {

      border-bottom-color: $date-picker-border;

    }

    &:after {

      border-bottom-color: $date-picker-bg;

    }

  }

  &.arrowBottom {

    &:before {

      border-top-color: $date-picker-border;

    }

    &:after {

      border-top-color: $date-picker-bg;

    }

  }

  &.showTimeInput.hasTime {

    .flatpickr-time {

      border-top: 1px solid $date-picker-border;

    }

  }

  .flatpickr-months {

    .flatpickr-month {

      background: $date-picker-month-bg;

      color: $date-picker-month-fg;

      fill: $date-picker-month-fg;

    }

    .flatpickr-prev-month, .flatpickr-next-month {

      color: $date-picker-month-fg;

      fill: $date-picker-month-fg;

      &:hover {

        color: $date-picker-today-bg;

        svg {

          fill: $date-picker-arrow-hover;

        }

      }

    }

  }

}

.flatpickr-wrapper {

  position: relative;

  display: inline-block;

}

.numInputWrapper {

  position: relative;

  height: auto;

  input, span {

    display: inline-block;

  }

  input {

    width: 100%;

    &::-ms-clear {

      display: none;

    }

    &::-webkit-outer-spin-button, &::-webkit-inner-spin-button {

      margin: 0;

      -webkit-appearance: none;

    }

  }

  span {

    position: absolute;

    right: 0;

    width: 14px;

    padding: 0 4px 0 2px;

    height: 50%;

    line-height: 50%;

    opacity: 0;

    cursor: pointer;

    box-sizing: border-box;

    &:after {

      display: block;

      content: "";

      position: absolute;

    }

    &.arrowUp {

      top: 0;

      border-bottom: 0;

      &:after {

        top: 26%;

      }

    }

    &.arrowDown {

      top: 50%;

      &:after {

        top: 40%;

      }

    }

    svg {

      width: inherit;

      height: auto;

    }

  }

  &:hover {

    span {

      opacity: 1;

    }

  }

    span {

      border: 1px solid rgba($date-picker-day-fg, 0.15);

      &:hover {

        background: rgba(invert($date-picker-bg), 0.1);

      }

      &:active {

        background: rgba(invert($date-picker-bg), 0.2);

      }

      &.arrowUp {

        &:after {

          border-left: 4px solid transparent;

          border-right: 4px solid transparent;

          border-bottom: 4px solid rgba($date-picker-day-fg, 0.6);

        }

      }

      &.arrowDown {

        &:after {

          border-left: 4px solid transparent;

          border-right: 4px solid transparent;

          border-top: 4px solid rgba($date-picker-day-fg, 0.6);

        }

      }

      svg {

        path {

          fill: rgba($date-picker-month-fg, 0.5);

        }

      }

    }

    &:hover {

      background: rgba(invert($date-picker-bg), 0.05);

    }

}

.flatpickr-current-month {

  font-size: 135%;

  font-weight: 300;

  color: inherit;

  position: absolute;

  width: 75%;

  left: 12.5%;

  padding: 0.22 \* $monthNavHeight 0 0 0;

  line-height: 1;

  height: $monthNavHeight;

  display: inline-block;

  text-align: center;

  transform: translate3d(0px, 0px, 0px);

  span.cur-month {

    font-family: inherit;

    font-weight: 700;

    color: inherit;

    display: inline-block;

    margin-left: 0.5ch;

    padding: 0;

  }

  .numInputWrapper {

    width: 6ch;

    display: inline-block;

  }

  input.cur-year {

    background: transparent;

    box-sizing: border-box;

    color: inherit;

    cursor: text;

    padding: 0 0 0 0.5ch;

    margin: 0;

    display: inline-block;

    font-size: inherit;

    font-family: inherit;

    font-weight: 300;

    line-height: inherit;

    height: auto;

    border: 0;

    border-radius: 0;

    vertical-align: initial;

    -webkit-appearance: textfield;

    -moz-appearance: textfield;

    appearance: textfield;

    &:focus {

      outline: 0;

    }

    &[disabled], &[disabled]:hover {

      font-size: 100%;

      pointer-events: none;

    }

  }

    span.cur-month {

      &:hover {

        background: rgba(invert($date-picker-bg), 0.05);

      }

    }

    .numInputWrapper {

      span.arrowUp:after {

        border-bottom-color: $date-picker-month-fg;

      }

      span.arrowDown:after {

        border-top-color: $date-picker-month-fg;

      }

    }

    input.cur-year {

      &[disabled], &[disabled]:hover {

        color: rgba($date-picker-month-fg, 0.5);

        background: transparent;

      }

    }

}

.flatpickr-monthDropdown-months {

  border: none;

  border-radius: 0;

  box-sizing: border-box;

  color: inherit;

  cursor: pointer;

  font-size: inherit;

  font-family: inherit;

  font-weight: 300;

  height: $monthNavHeight - 6px;

  line-height: inherit;

  margin: -1px 0 0 0;

  outline: none;

  padding: 0 0 0 0.5ch;

  position: relative;

  vertical-align: initial;

  -webkit-box-sizing: border-box;

  -webkit-appearance: none;

  -moz-appearance: none;

  appearance: none;

  width: auto;

  &:focus, &:active {

    outline: none;

  }

  .flatpickr-monthDropdown-month {

    outline: none;

    padding: 0;

  }

    background: $date-picker-month-bg;

    &:hover {

      background: rgba(invert($date-picker-bg), 0.05);

    }

    .flatpickr-monthDropdown-month {

      background-color: $date-picker-month-bg;

    }

}

.flatpickr-weekdays {

  text-align: center;

  overflow: hidden;

  width: 100%;

  display: flex;

  align-items: center;

  height: $weekdaysHeight;

  .flatpickr-weekdaycontainer {

    display: flex;

    flex: 1;

  }

  background: $date-picker-weekdays-bg;

}

span.flatpickr-weekday {

  cursor: default;

  font-size: 90%;

  line-height: 1;

  margin: 0;

  text-align: center;

  display: block;

  flex: 1;

  font-weight: bolder;

    background: $date-picker-month-bg;

    color: $date-picker-weekdays-fg;

}

.dayContainer, .flatpickr-weeks {

  padding: 1px 0 0 0;

}

.flatpickr-days {

  position: relative;

  overflow: hidden;

  display: flex;

  align-items: flex-start;

  width: $daysWidth;

  &:focus {

    outline: 0;

  }

}

.dayContainer {

  padding: 0;

  outline: 0;

  text-align: left;

  width: $daysWidth;

  min-width: $daysWidth;

  max-width: $daysWidth;

  box-sizing: border-box;

  display: inline-block;

  display: -ms-flexbox;

  display: flex;

  flex-wrap: wrap;

  -ms-flex-wrap: wrap;

  -ms-flex-pack: justify;

  justify-content: space-around;

  transform: translate3d(0px, 0px, 0px);

  opacity: 1;

    & + .dayContainer {

      box-shadow: -1px 0 0 $date-picker-border;

    }

}

.flatpickr-day {

  background: none;

  border: 1px solid transparent;

  border-radius: 150px;

  box-sizing: border-box;

  cursor: pointer;

  font-weight: 400;

  width: 14.2857143%;

  flex-basis: 14.2857143%;

  max-width: $daySize;

  height: $daySize;

  line-height: $daySize;

  margin: 0;

  display: inline-block;

  position: relative;

  justify-content: center;

  text-align: center;

  &, &.prevMonthDay, &.nextMonthDay {

    &.inRange, &.today.inRange, &:hover, &:focus {

      cursor: pointer;

      outline: 0;

    }

  }

  &.selected, &.startRange, &.endRange {

    &.startRange {

      border-radius: 50px 0 0 50px;

    }

    &.endRange {

      border-radius: 0 50px 50px 0;

    }

    &.startRange.endRange {

      border-radius: 50px;

    }

  }

  &.inRange {

    border-radius: 0;

  }

  &.flatpickr-disabled, &.flatpickr-disabled:hover,

  &.prevMonthDay, &.nextMonthDay,

  &.notAllowed, &.notAllowed.prevMonthDay, &.notAllowed.nextMonthDay {

    cursor: default;

  }

  &.flatpickr-disabled, &.flatpickr-disabled:hover {

    cursor: not-allowed;

  }

  &.week.selected {

    border-radius: 0;

  }

  &.hidden {

    visibility: hidden;

  }

    background: none;

    border: 1px solid transparent;

    color: $date-picker-day-fg;

    &, &.prevMonthDay, &.nextMonthDay {

      &.inRange, &.today.inRange, &:hover, &:focus {

        background: $date-picker-day-hover-bg;

        border-color: $date-picker-day-hover-bg;

      }

    }

    &.today {

      border-color: $date-picker-today-border;

      &:hover, &:focus {

        border-color: $date-picker-today-border;

        background: $date-picker-today-bg;

        color: $date-picker-today-fg;

      }

    }

    &.selected, &.startRange, &.endRange {

      &, &.inRange, &:focus, &:hover, &.prevMonthDay, &.nextMonthDay {

        background: $selectedDayBackground;

        box-shadow: none;

        color: $selectedDayForeground;

        border-color: $selectedDayBackground;

      }

      &.startRange + .endRange:not(:nth-child(7n+1)) {

        box-shadow: -5 \* $dayMargin 0 0 $selectedDayBackground;

      }

    }

    &.inRange {

      border-radius: 0;

      box-shadow: -2.5 \* $dayMargin 0 0 $date-picker-day-hover-bg, 2.5 \* $dayMargin 0 0 $date-picker-day-hover-bg;

    }

    &.flatpickr-disabled, &.flatpickr-disabled:hover,

    &.prevMonthDay, &.nextMonthDay,

    &.notAllowed, &.notAllowed.prevMonthDay, &.notAllowed.nextMonthDay {

      color: rgba($date-picker-day-fg, 0.3);

      background: transparent;

      border-color: $disabledBorderColor;

    }

    &.flatpickr-disabled, &.flatpickr-disabled:hover {

      color: rgba($date-picker-day-fg, 0.1);

    }

}

.rangeMode .flatpickr-day {

  margin-top: 1px;

}

.flatpickr-weekwrapper {

  float: left;

  .flatpickr-weeks {

    padding: 0 12px;

  }

  .flatpickr-weekday {

    float: none;

    width: 100%;

    line-height: $weekdaysHeight;

  }

  span.flatpickr-day {

    &, &:hover {

      display: block;

      width: 100%;

      max-width: none;

      cursor: default;

    }

  }

    .flatpickr-weeks {

      box-shadow: 1px 0 0 $date-picker-border;

    }

    span.flatpickr-day {

      &, &:hover {

        color: rgba($date-picker-day-fg, 0.3);

        background: transparent;

        border: none;

      }

    }

}

.flatpickr-innerContainer {

  display: flex;

  box-sizing: border-box;

  overflow: hidden;

}

.flatpickr-rContainer {

  display: inline-block;

  padding: 0;

  box-sizing: border-box;

}

.flatpickr-time {

  text-align: center;

  outline: 0;

  height: 0;

  line-height: $timeHeight;

  max-height: $timeHeight;

  box-sizing: border-box;

  overflow: hidden;

  display: flex;

  &:after {

    content: "";

    display: table;

    clear: both;

  }

  .numInputWrapper {

    flex: 1;

    width: 40%;

    height: $timeHeight;

    float: left;

  }

  &.hasSeconds .numInputWrapper {

    width: 26%;

  }

  &.time24hr .numInputWrapper {

    width: 49%;

  }

  input {

    background: transparent;

    box-shadow: none;

    border: 0;

    border-radius: 0;

    text-align: center;

    margin: 0;

    padding: 0;

    height: inherit;

    line-height: inherit;

    font-size: 14px;

    position: relative;

    box-sizing: border-box;

    -webkit-appearance: textfield;

    -moz-appearance: textfield;

    appearance: textfield;

    &.flatpickr-hour {

      font-weight: bold;

    }

    &.flatpickr-minute, &.flatpickr-second {

      font-weight: 400;

    }

    &:focus {

      outline: 0;

      border: 0;

    }

  }

  .flatpickr-time-separator, .flatpickr-am-pm {

    height: inherit;

    float: left;

    line-height: inherit;

    font-weight: bold;

    width: 2%;

    user-select: none;

    align-self: center;

  }

  .flatpickr-am-pm {

    outline: 0;

    width: 18%;

    cursor: pointer;

    text-align: center;

    font-weight: 400;

  }

    .numInputWrapper {

      span.arrowUp:after {

        border-bottom-color: $date-picker-day-fg;

      }

      span.arrowDown:after {

        border-top-color: $date-picker-day-fg;

      }

    }

    input {

      color: $date-picker-day-fg;

    }

    .flatpickr-time-separator, .flatpickr-am-pm {

      color: $date-picker-day-fg;

    }

    input, .flatpickr-am-pm {

      &:hover, &:focus {

        background: lighten($date-picker-day-hover-bg, 3);

      }

    }

}

.flatpickr-input[readonly] {

  cursor: pointer;

}

##### **Index.js**

export { default as DatePicker } from './DatePicker';

export { default as InputSuggest } from './InputSuggest';

##### **InputSuggest.js**

import React from 'react';

import Autosuggest from 'react-autosuggest';

import { connect } from 'react-redux';

import \_ from 'lodash';

import "./InputSuggest.scss";

const isAlphaNumericChar = keycode => {

    return (keycode >= 48 && keycode <= 57) || (keycode >= 65 && keycode <= 90);

};

class InputSuggestion extends React.Component {

    constructor() {

        super();

        this.state = {

            textInput: '',

            suggestions: []

        };

        document.addEventListener('keydown', this.freeTyping.bind(this), false);

    }

    freeTyping(e) {

        if (!isAlphaNumericChar(e.keyCode)) {

            return;

        }

        if (e.target.value === undefined) {

            this.inputSearch.focus();

        }

    }

    getSuggestions = textInput => {

        // if (!textInput || textInput.length === 0) return []; //chưa nhập -> chưa gợi ý

        const inputValue = textInput.trim().toLowerCase();

        if (!this.props.inputsWithIndex) {

            return [{ textInput: textInput }];

        };

        let inputsWithIndex = this.props.inputsWithIndex;

        let keyArr = Object.keys(inputsWithIndex).filter(

            textInput => {

                return textInput.toLowerCase().indexOf(inputValue) >= 0

            }

        );

        var suggestArr = keyArr.map(function (key) {

            return inputsWithIndex[key];

        });

        return suggestArr;

    };

    storeInputReference = autosuggest => {

        if (autosuggest !== null) {

            this.inputSearch = autosuggest.input;

        }

    };

    shouldRenderSuggestions = value => {

        return true;

    };

    getSuggestionValue = suggestion => {

        this.props.onSelected(suggestion);

        return suggestion.displayName;

    }

    renderSuggestion = suggestion => {

        return (

            <div className="suggest-item">

                {suggestion.displayName}

            </div>

        );

    };

    onSuggestionsFetchRequested = ({ value }) => {

        this.setState({

            suggestions: this.sortSuggestions(this.getSuggestions(value), value)

        });

    };

    sortSuggestions(suggestions, value) {

        var results =  \_.sortBy(suggestions, (element) => {

            return element.displayName

        })

        return results;

    }

    onSuggestionsClearRequested = () => {

        this.setState({

            suggestions: []

        });

    };

    onSuggestionSelected = (event, selected) => {

        this.props.onSelected(selected.suggestion)

        this.setState({

            textInput: selected.suggestion && selected.suggestion.displayName

        });

    };

    handleChangeInput = (event, { newValue }) => {

        this.setState({

            textInput: newValue || ''

        });

    };

    reset() {

        this.setState({

            textInput: ''

        });

        this.onSuggestionsFetchRequested({ value: '' });

    }

    handleBlurInput() {

        // this.setState({

        //     textInput: ''

        // });

    }

    render() {

        const { textInput, suggestions } = this.state;

        const inputProps = {

            value: textInput,

            className: "custom-form-control",

            onChange: this.handleChangeInput,

            onClick: () => {

                this.reset();

            },

            onBlur: () => {

                this.handleBlurInput();

            }

        };

        return (

            <Autosuggest

                suggestions={suggestions}

                onSuggestionsFetchRequested={this.onSuggestionsFetchRequested}

                onSuggestionsClearRequested={this.onSuggestionsClearRequested}

                getSuggestionValue={this.getSuggestionValue}

                renderSuggestion={this.renderSuggestion}

                onSuggestionSelected={this.onSuggestionSelected}

                shouldRenderSuggestions={this.shouldRenderSuggestions}

                highlightFirstSuggestion={true}

                inputProps={inputProps}

                ref={this.storeInputReference}

            />

        );

    }

}

const mapStateToProps = state => {

    return {

    };

};

export default connect(mapStateToProps, null)(InputSuggestion);

##### **InputSuggest.scss**

@import "../../styles/common";

.react-autosuggest\_\_suggestions-container {

    min-width: 90%;

    position: absolute;

    top: 50px;

    z-index: 4;

    max-height: 140px;

    overflow: auto;

    background: $footer-container;

    box-shadow: 2px 2px 4px 0 $box-shadow-color;

    -webkit-box-shadow: 2px 2px 4px 0 $box-shadow-color;

}

.react-autosuggest\_\_suggestions-list {

    list-style-type: none;

    margin-bottom: 0;

    padding: 0px;

}

.react-autosuggest\_\_suggestion {

    padding: 0 5px;

    height: 28px;

    line-height: 28px;

    border-bottom: 1px solid $border;

    &:hover {

        cursor: pointer;

        background-color: darken($footer-container, 5);

    }

}

.react-autosuggest\_\_suggestion--highlighted {

    background-color: darken($footer-container, 5);

}

#### **src/components/Breadcrumb.js**

import React from 'react';

import { useLocation, Link } from 'react-router-dom';

const Breadcrumb = () => {

   const location = useLocation();

   const pathnames = location.pathname.split('/').filter(x => x);

   return (

      <nav className="breadcrumb">

         <Link to="/">Trang chủ</Link>

         {pathnames.map((name, index) => {

            const routeTo = `/${pathnames.slice(0, index + 1).join('/')}`;

            return <span key={index}> / <Link to={routeTo}>{name}</Link></span>;

         })}

      </nav>

   );

};

export default Breadcrumb;

#### **src/components/ConfirmModal.js**

import React, { Component } from 'react';

import { FormattedMessage } from 'react-intl';

import { connect } from 'react-redux';

import { Modal } from 'reactstrap';

import './ConfirmModal.scss';

import \* as actions from "../store/actions";

import { KeyCodeUtils } from "../utils";

class ConfirmModal extends Component {

    constructor(props) {

        super(props);

        this.acceptBtnRef = React.createRef();

    }

    initialState = {

    };

    state = {

        ...this.initialState

    };

    componentDidMount() {

        document.addEventListener('keydown', this.handlerKeyDown);

    }

    componentWillUnmount() {

        document.removeEventListener('keydown', this.handlerKeyDown);

    }

    handlerKeyDown = (event) => {

        const keyCode = event.which || event.keyCode;

        if (keyCode === KeyCodeUtils.ENTER) {

            if (!this.acceptBtnRef.current || this.acceptBtnRef.current.disabled) return;

            this.acceptBtnRef.current.click();

        }

    }

    onAcceptBtnClick = () => {

        const { contentOfConfirmModal } = this.props;

        if (contentOfConfirmModal.handleFunc) {

            contentOfConfirmModal.handleFunc(contentOfConfirmModal.dataFunc);

        }

        this.onClose();

    }

    onClose = () => {

        this.props.setContentOfConfirmModal({

            isOpen: false,

            messageId: "",

            handleFunc: null,

            dataFunc: null

        });

    }

    render() {

        const { contentOfConfirmModal } = this.props;

        return (

            <Modal isOpen={contentOfConfirmModal.isOpen} className='confirm-modal' centered={true}>

                <div className="modal-header">

                    <div className="modal-title">

                        <FormattedMessage id={"common.confirm"} />

                    </div>

                    <div className="col-auto">

                        <button className="btn btn-close" onClick={this.onClose}>

                            <i className="fal fa-times" />

                        </button>

                    </div>

                </div>

                <div className="modal-body">

                    <div className="confirm-modal-content">

                        <div className="row">

                            <div className="col-12">

                                <FormattedMessage id={contentOfConfirmModal.messageId ? contentOfConfirmModal.messageId : "common.confirm-this-task"} />

                            </div>

                            <hr />

                            <div className="col-12">

                                <div className="btn-container text-center">

                                    <button className="btn btn-add" onClick={this.onClose} >

                                        <FormattedMessage id="common.close" />

                                    </button>

                                    <button ref={this.acceptBtnRef} className="btn btn-add" onClick={this.onAcceptBtnClick}>

                                        <FormattedMessage id={"common.accept"} />

                                    </button>

                                </div>

                            </div>

                        </div>

                    </div>

                </div>

            </Modal >

        );

    }

}

const mapStateToProps = state => {

    return {

        lang: state.app.language,

        contentOfConfirmModal: state.app.contentOfConfirmModal

    };

};

const mapDispatchToProps = dispatch => {

    return {

        setContentOfConfirmModal: (contentOfConfirmModal) => dispatch(actions.setContentOfConfirmModal(contentOfConfirmModal))

    };

};

export default connect(mapStateToProps, mapDispatchToProps)(ConfirmModal);

#### **src/components/ConfirmModal.scss**

@import "../styles/common";

.confirm-modal {

  &.modal-dialog {

    @media (min-width: 766px) {

      min-width: 650px;

    }

  }

  hr {

    border: none;

    border-bottom: 1px solid $border;;

    width: 100%;

    margin: 10px 15px 5px 15px;

  }

  .btn-container {

    margin-top: 10px;

    .btn {

      width: 100px;

      &:first-child {

        margin-right: 10px;

      }

    }

  }

  .custom-form-group {

    margin: 5px 0;

    .custom-form-control.readonly {

      min-height: 28px;

      height: auto;

    }

  }

}

#### **src/components/CustomScrollbars.js**

import React, { Component } from 'react';

import { Scrollbars } from 'react-custom-scrollbars-2';

import './CustomScrollbars.scss';

class CustomScrollbars extends Component {

    ref = React.createRef();

    getScrollLeft = () => {

        const scrollbars = this.ref.current;

        return scrollbars.getScrollLeft();

    }

    getScrollTop = () => {

        const scrollbars = this.ref.current;

        return scrollbars.getScrollTop();

    }

    scrollToBottom = () => {

        if (!this.ref || !this.ref.current) {

            return;

        }

        const scrollbars = this.ref.current;

        const targetScrollTop = scrollbars.getScrollHeight();

        this.scrollTo(targetScrollTop);

    };

    scrollTo = (targetTop) => {

        const { quickScroll } = this.props;

        if (!this.ref || !this.ref.current) {

            return;

        }

        const scrollbars = this.ref.current;

        const originalTop = scrollbars.getScrollTop();

        let iteration = 0;

        const scroll = () => {

            iteration++;

            if (iteration > 30) {

                return;

            }

            scrollbars.scrollTop(originalTop + (targetTop - originalTop) / 30 \* iteration);

            if (quickScroll && quickScroll === true) {

                scroll();

            } else {

                setTimeout(() => {

                    scroll();

                }, 20);

            }

        };

        scroll();

    };

    renderTrackHorizontal = (props) => {

        return (

            <div {...props} className="track-horizontal" />

        );

    };

    renderTrackVertical = (props) => {

        return (

            <div {...props} className="track-vertical" />

        );

    };

    renderThumbHorizontal = (props) => {

        return (

            <div {...props} className="thumb-horizontal" />

        );

    };

    renderThumbVertical = (props) => {

        return (

            <div {...props} className="thumb-vertical" />

        );

    };

    renderNone = (props) => {

        return (

            <div />

        );

    };

    render() {

        const { className, disableVerticalScroll, disableHorizontalScroll, children, ...otherProps } = this.props;

        return (

            <Scrollbars

                ref={this.ref}

                autoHide={true}

                autoHideTimeout={200}

                hideTracksWhenNotNeeded={true}

                className={className ? className + ' custom-scrollbar' : 'custom-scrollbar'}

                {...otherProps}

                renderTrackHorizontal={disableHorizontalScroll ? this.renderNone : this.renderTrackHorizontal}

                renderTrackVertical={disableVerticalScroll ? this.renderNone : this.renderTrackVertical}

                renderThumbHorizontal={disableHorizontalScroll ? this.renderNone : this.renderThumbHorizontal}

                renderThumbVertical={disableVerticalScroll ? this.renderNone : this.renderThumbVertical}

            >

                {children}

            </Scrollbars>

        );

    }

}

export default CustomScrollbars;

#### **src/components/CustomScrollbars.scss**

@import "../styles/common.scss";

.custom-scrollbar {

  .thumb-horizontal {

    background-color: rgba($bg-scrollbar, 0.4);

    box-shadow: inset 1px 1px 0 rgba($bg-scrollbar, 0.10), inset 0 -1px 0 rgba($bg-scrollbar, 0.07);

    &:hover {

      background-color: rgba($bg-scrollbar, 0.8);

      box-shadow: inset 1px 1px 1px rgba($bg-scrollbar, 0.50);

    }

    &:active {

      background-color: rgba($bg-scrollbar, 0.9);

      box-shadow: inset 1px 1px 3px rgba($bg-scrollbar, 0.6);

    }

  }

  .thumb-vertical {

    background-color: rgba($bg-scrollbar, 0.4);

    box-shadow: inset 1px 1px 0 rgba($bg-scrollbar, 0.10), inset 0 -1px 0 rgba($bg-scrollbar, 0.07);

    &:hover {

      background-color: rgba($bg-scrollbar, 0.8);

      box-shadow: inset 1px 1px 1px rgba($bg-scrollbar, 0.50);

    }

    &:active {

      background-color: rgba($bg-scrollbar, 0.9);

      box-shadow: inset 1px 1px 3px rgba($bg-scrollbar, 0.6);

    }

  }

  &:hover {

    > .track-horizontal, > .track-vertical {

      opacity: 1 !important;

      > .thumb-horizontal, > .thumb-vertical {

        opacity: 1;

      }

    }

  }

  .track-horizontal {

    right: 0;

    bottom: 0;

    left: 0;

    height: 5px !important;

    &:hover, &:active {

      height: 10px !important;

    }

  }

  .track-vertical {

    right: 0;

    bottom: 0;

    top: 0;

    width: 5px !important;

    &:hover, &:active {

      width: 10px !important;

    }

  }

  .thumb-horizontal {

    cursor: pointer;

    border-radius: 10px;

    background-clip: padding-box;

    min-width: 0px;

  }

  .thumb-vertical {

    cursor: pointer;

    border-radius: 10px;

    background-clip: padding-box;

    min-height: 0px;

  }

}

#### **src/components/CustomToast.js**

import React, { Component, Fragment } from 'react';

import { FormattedMessage, FormattedTime } from 'react-intl';

import CustomScrollBar from '../components/CustomScrollbars';

import './CustomToast.scss';

class CustomToast extends Component {

    render() {

        const { titleId, message, messageId, time } = this.props;

        return (

            <Fragment>

                <div className="custom-toast">

                    <div className="toast-title">

                        {time && (

                            <span className="date">

                                <FormattedTime hour='numeric' minute='numeric' second='numeric' hour12={true} value={time} />

                            </span>

                        )}

                        <i className="fa fa-fw fa-exclamation-triangle" />

                        <FormattedMessage id={titleId} />

                    </div>

                    {

                        (message && typeof message === 'object') ?

                            <CustomScrollBar autoHeight={true} autoHeightMin={50} autoHeightMax={100}>

                                {

                                    message.map((msg, index) => {

                                        return (

                                            <Fragment key={index}>

                                                <div className="toast-content">{msg}</div>

                                            </Fragment>

                                        )

                                    })

                                }

                            </CustomScrollBar> :

                            <div className="toast-content">

                                {message ? message : (messageId ? (<FormattedMessage id={messageId} />) : null)}

                            </div>

                    }

                </div>

            </Fragment>

        );

    }

}

export class CustomToastCloseButton extends Component {

    render() {

        return (

            <button type="button" className="toast-close" onClick={this.props.closeToast}>

                <i className="fa fa-fw fa-times-circle" />

            </button>

        );

    }

}

export default CustomToast;

#### **src/components/CustomToast.scss**

@import "../styles/common.scss";

.toast-container {

  .toast-item {

    white-space: pre-wrap;

    box-shadow: 0 0 5px 5px $box-shadow-color;

    -webkit-box-shadow: 0 0 5px 5px $box-shadow-color;

    &.Toastify\_\_toast--success {

      background: $common-green;

    }

    &.Toastify\_\_toast--error {

      background: $common-red;

    }

    &.Toastify\_\_toast--warn {

      background: $colormain;

    }

    &.Toastify\_\_toast--info {

      background: $colormain;

    }

    .toast-close {

      position: absolute;

      right: 10px;

      top:10px;

      color: $common-text-color;

      font-size: $base-size;

      padding: 0;

      cursor: pointer;

      background: transparent;

      border: 0;

      -webkit-appearance: none;

      transition: opacity 0.2s ease-out;

    }

    .toast-item-body {

      color: $common-text-color;

      display: block;

      flex: none;

      width: 100%;

      .toast-title {

        font-weight: bold;

        font-size: $base-size;

        margin: 0 20px 5px 0;

        .fixed-scroll-bar {

          height: 50px;

        }

        .date {

          float: right;

          font-size: $base-size - 2px;

          vertical-align: middle;

          margin-right: 5px;

          margin-bottom: 0;

          padding: 2px 5px;

        }

        i {

          position: relative;

          margin-right: 3px;

        }

      }

      .toast-content {

        font-size: $base-size - 2px;

      }

    }

  }

}

#### **src/components/Navigator.js**

import React, { Component, Fragment } from 'react';

import { Link, withRouter } from 'react-router-dom';

import { FormattedMessage } from 'react-intl';

import { connect } from 'react-redux';

import './Navigator.scss';

class MenuGroup extends Component {

    render() {

        const { name, children } = this.props;

        return (

            <li className="menu-group">

                <div className="menu-group-name">

                    <FormattedMessage id={name} />

                </div>

                <ul className="menu-list list-unstyled">

                    {children}

                </ul>

            </li>

        );

    }

}

class Menu extends Component {

    render() {

        const { name, active, link, children, onClick, hasSubMenu, onLinkClick } = this.props;

        return (

            <li className={"menu" + (hasSubMenu ? " has-sub-menu" : "") + ("") + (active ? " active" : "")}>

                {hasSubMenu ? (

                    <Fragment>

                        <span

                            data-toggle="collapse"

                            className={"menu-link collapsed"}

                            onClick={onClick}

                            aria-expanded={"false"}

                        >

                            <FormattedMessage id={name} />

                            <div className="icon-right">

                                <i className={"far fa-angle-right"} />

                            </div>

                        </span>

                        <div>

                            <ul className="sub-menu-list list-unstyled">

                                {children}

                            </ul>

                        </div>

                    </Fragment>

                ) : (

                        <Link to={link} className="menu-link" onClick={onLinkClick}>

                            <FormattedMessage id={name} />

                        </Link>

                    )}

            </li>

        );

    }

}

class SubMenu extends Component {

    getItemClass = path => {

        return this.props.location.pathname === path ? "active" : "";

    };

    render() {

        const { name, link, onLinkClick } = this.props;

        return (

            <li className={"sub-menu " + this.getItemClass(link)}>

                <Link to={link} className="sub-menu-link" onClick={onLinkClick}>

                    <FormattedMessage id={name} />

                </Link>

            </li>

        );

    }

}

const MenuGroupWithRouter = withRouter(MenuGroup);

const MenuWithRouter = withRouter(Menu);

const SubMenuWithRouter = withRouter(SubMenu);

const withRouterInnerRef = (WrappedComponent) => {

    class InnerComponentWithRef extends React.Component {

        render() {

            const { forwardRef, ...rest } = this.props;

            return <WrappedComponent {...rest} ref={forwardRef} />;

        }

    }

    const ComponentWithRef = withRouter(InnerComponentWithRef, { withRef: true });

    return React.forwardRef((props, ref) => {

        return <ComponentWithRef {...props} forwardRef={ref} />;

    });

};

class Navigator extends Component {

    state = {

        expandedMenu: {}

    };

    toggle = (groupIndex, menuIndex) => {

        const expandedMenu = {};

        const needExpand = !(this.state.expandedMenu[groupIndex + '\_' + menuIndex] === true);

        if (needExpand) {

            expandedMenu[groupIndex + '\_' + menuIndex] = true;

        }

        this.setState({

            expandedMenu: expandedMenu

        });

    };

    isMenuHasSubMenuActive = (location, subMenus, link) => {

        if (subMenus) {

            if (subMenus.length === 0) {

                return false;

            }

            const currentPath = location.pathname;

            for (let i = 0; i < subMenus.length; i++) {

                const subMenu = subMenus[i];

                if (subMenu.link === currentPath) {

                    return true;

                }

            }

        }

        if (link) {

            return this.props.location.pathname === link;

        }

        return false;

    };

    checkActiveMenu = () => {

        const { menus, location } = this.props;

        outerLoop:

        for (let i = 0; i < menus.length; i++) {

            const group = menus[i];

            if (group.menus && group.menus.length > 0) {

                for (let j = 0; j < group.menus.length; j++) {

                    const menu = group.menus[j];

                    if (menu.subMenus && menu.subMenus.length > 0) {

                        if (this.isMenuHasSubMenuActive(location, menu.subMenus, null)) {

                            const key = i + '\_' + j;

                            this.setState({

                                expandedMenu: {

                                    [key]: true

                                }

                            });

                            break outerLoop;

                        }

                    }

                }

            }

        }

    };

    componentDidMount() {

        this.checkActiveMenu();

    };

    // componentWillReceiveProps(nextProps, prevState) {

    //     const { location, setAccountMenuPath, setSettingMenuPath } = this.props;

    //     const { location: nextLocation } = nextProps;

    //     if (location !== nextLocation) {

    //         let pathname = nextLocation && nextLocation.pathname;

    //         if ((pathname.startsWith('/account/') || pathname.startsWith('/fds/account/'))) {

    //             setAccountMenuPath(pathname);

    //         }

    //         if (pathname.startsWith('/settings/')) {

    //             setSettingMenuPath(pathname);

    //         };

    //     };

    // };

    componentDidUpdate(prevProps, prevState) {

        const { location } = this.props;

        const { location: prevLocation } = prevProps;

        if (location !== prevLocation) {

            this.checkActiveMenu();

        };

    };

    render() {

        const { menus, location, onLinkClick } = this.props;

        return (

            <Fragment>

                <ul className="navigator-menu list-unstyled">

                    {

                        menus.map((group, groupIndex) => {

                            return (

                                <Fragment key={groupIndex}>

                                    <MenuGroupWithRouter name={group.name}>

                                        {group.menus ? (

                                            group.menus.map((menu, menuIndex) => {

                                                const isMenuHasSubMenuActive = this.isMenuHasSubMenuActive(location, menu.subMenus, menu.link);

                                                const isSubMenuOpen = this.state.expandedMenu[groupIndex + '\_' + menuIndex] === true;

                                                return (

                                                    <MenuWithRouter

                                                        key={menuIndex}

                                                        active={isMenuHasSubMenuActive}

                                                        name={menu.name}

                                                        link={menu.link}

                                                        hasSubMenu={menu.subMenus}

                                                        isOpen={isSubMenuOpen}

                                                        onClick={() => this.toggle(groupIndex, menuIndex)}

                                                        onLinkClick={onLinkClick}

                                                    >

                                                        {menu.subMenus && menu.subMenus.map((subMenu, subMenuIndex) => (

                                                            <SubMenuWithRouter

                                                                key={subMenuIndex}

                                                                name={subMenu.name}

                                                                link={subMenu.link}

                                                                onClick={this.closeOtherExpand}

                                                                onLinkClick={onLinkClick}

                                                            />

                                                        ))}

                                                    </MenuWithRouter>

                                                );

                                            })

                                        ) : null}

                                    </MenuGroupWithRouter>

                                </Fragment>

                            );

                        })

                    }

                </ul>

            </Fragment>

        );

    }

}

const mapStateToProps = state => {

    return {

    };

};

const mapDispatchToProps = dispatch => {

    return {

    }

}

export default withRouterInnerRef(connect(mapStateToProps, mapDispatchToProps)(Navigator));

#### **src/components/Navigator.scss**

@import "../styles/common.scss";

.navigator-menu {

    display: flex;

    margin: 0;

    padding: 0;

    &.list-unstyled,

    .list-unstyled {

        list-style-type: none;

    }

    .menu-group {

        &:hover {

            cursor: pointer;

            background-color: darken($colormain, 5);

        }

        .menu-group-name {

            line-height: 40px;

            padding: 0 15px;

            border-right: 1px solid $border;

            &:first-child {

                border-left: 1px solid $border;

            }

        }

        .menu-list {

            display: none;

            background-color: $bg-menu-color;

            box-shadow: 2px 2px 4px 0 $box-shadow-color;

            -webkit-box-shadow: 2px 2px 4px 0 $box-shadow-color;

            color: $text-in-light;

            position: absolute;

            padding: 0;

            .menu {

                width: 205px;

                padding: 0 15px;

                height: 35px;

                line-height: 35px;

                text-transform: none;

                .menu-link {

                    text-decoration: none;

                    color: $text-in-light;

                }

                .sub-menu-list {

                    display: none;

                    background-color: $bg-menu-color;

                    box-shadow: 2px 2px 4px 0 $box-shadow-color;

                    -webkit-box-shadow: 2px 2px 4px 0 $box-shadow-color;

                    position: absolute;

                    top: 0;

                    left: 205px;

                    padding: 0;

                    .sub-menu {

                        padding: 0 15px;

                        height: 35px;

                        line-height: 35px;

                        white-space: nowrap;

                        &:hover {

                            background-color: darken($bg-menu-color, 5);

                        }

                        .sub-menu-link {

                            text-decoration: none;

                            color: $text-in-light;

                        }

                        a {

                            display: block;

                        }

                        &.active a {

                            font-weight: 500;

                            color: $colormain;

                        }

                    }

                }

                &.active span {

                    font-weight: 500;

                    color: $colormain;

                }

                &:hover {

                    background-color: darken($bg-menu-color, 3);

                    .sub-menu-list {

                        display: block;

                    }

                }

                .icon-right {

                    display: block;

                    position: absolute;

                    top: 0;

                    right: 10px;

                }

            }

        }

        &:hover {

            .menu-list {

                display: block;

            }

        }

    }

}

### **src/hoc/**

##### **Authentication.js**

import locationHelperBuilder from "redux-auth-wrapper/history4/locationHelper";

import { connectedRouterRedirect } from "redux-auth-wrapper/history4/redirect";

const locationHelper = locationHelperBuilder({});

export const userIsAuthenticated = connectedRouterRedirect({

    authenticatedSelector: state => state.admin.isLoggedIn,

    wrapperDisplayName: 'UserIsAuthenticated',

    redirectPath: '/login'

});

export const userIsNotAuthenticated = connectedRouterRedirect({

    // Want to redirect the user when they are authenticated

    authenticatedSelector: state => !state.admin.isLoggedIn,

    wrapperDisplayName: 'UserIsNotAuthenticated',

    redirectPath: (state, ownProps) => locationHelper.getRedirectQueryParam(ownProps) || '/',

    allowRedirectBack: false

});

##### **IntProviderWrapper.js**

import React, { Component } from "react";

import { connect } from 'react-redux';

import { IntlProvider } from "react-intl";

import '@formatjs/intl-pluralrules/polyfill';

import '@formatjs/intl-pluralrules/locale-data/en';

import '@formatjs/intl-pluralrules/locale-data/vi';

import '@formatjs/intl-relativetimeformat/polyfill';

import '@formatjs/intl-relativetimeformat/locale-data/en';

import '@formatjs/intl-relativetimeformat/locale-data/vi';

import { LanguageUtils } from '../utils'

const messages = LanguageUtils.getFlattenedMessages();

class IntlProviderWrapper extends Component {

    render() {

        const { children, language } = this.props;

        return (

            <IntlProvider

                locale={language}

                messages={messages[language]}

                defaultLocale="vi">

                {children}

            </IntlProvider>

        );

    }

}

const mapStateToProps = state => {

    return {

        language: state.app.language

    };

};

export default connect(mapStateToProps, null)(IntlProviderWrapper);

##### **PrivateRoute.js**

import React from 'react';

import { Route, Redirect } from 'react-router-dom';

const PrivateRoute = ({ component: Component, role, render, ...rest }) => {

   const token = localStorage.getItem('token');

   const roleId = localStorage.getItem('roleId');

   return (

      <Route

         {...rest}

         render={(props) => {

            if (!token) return <Redirect to="/login" />;

            if (role && roleId !== role) return <Redirect to="/" />;

            return render ? render(props) : <Component {...props} />;

         }}

      />

   );

};

export default PrivateRoute;

### **src/layouts/**

##### **AdminLayout.js**

Đang phát triển…

##### **PublicLayout.js**

import React from 'react';

import Header from '../components/Header/Header';

import Footer from '../components/Footer/Footer';

const PublicLayout = (props) => {

   return (

      <>

         <Header />

         <main className="main-content">

            {props.children}

         </main>

         <Footer />

      </>

   );

};

export default PublicLayout;

### **src/pages/**

#### **src/admin/**

##### **ProductManager.js**

import React, { Component } from 'react';

import { FormattedMessage } from 'react-intl';

import { connect } from 'react-redux';

class ProductManage extends Component {

    state = {

    }

    componentDidMount() {

    }

    render() {

        return (

            <div className="text-center" >Manage products</div>

        )

    }

}

const mapStateToProps = state => {

    return {

    };

};

const mapDispatchToProps = dispatch => {

    return {

    };

};

export default connect(mapStateToProps, mapDispatchToProps)(ProductManage);

##### **RegisterPackageGroupOrAcc.js**

import React, { Component } from 'react';

import { FormattedMessage } from 'react-intl';

import { connect } from 'react-redux';

class RegisterPackageGroupOrAcc extends Component {

    constructor(props) {

        super(props);

    }

    render() {

        return (

            <div className="text-center">

                register package group or account

            </div>)

    }

}

const mapStateToProps = state => {

    return {

    };

};

const mapDispatchToProps = dispatch => {

    return {

    };

};

export default connect(mapStateToProps, mapDispatchToProps)(RegisterPackageGroupOrAcc);

##### **UserManager.js**

import React, { Component } from 'react';

import { FormattedMessage } from 'react-intl';

import { connect } from 'react-redux';

class UserManage extends Component {

    state = {

    }

    componentDidMount() {

    }

    render() {

        return (

            <div className="text-center">Manage users</div>

        );

    }

}

const mapStateToProps = state => {

    return {

    };

};

const mapDispatchToProps = dispatch => {

    return {

    };

};

export default connect(mapStateToProps, mapDispatchToProps)(UserManage);

#### **src/auth/**

##### **Login.js**

import React, { Component } from 'react';

import { connect } from 'react-redux';

import { push } from "connected-react-router";

import \* as actions from "../../store/actions";

import './Login.scss';

import { FormattedMessage } from 'react-intl';

class Login extends Component {

   constructor(props) {

      super(props);

   }

   render() {

      return (

         <div>login</div>

      )

   }

}

const mapStateToProps = state => {

   return {

      lang: state.app.language

   };

};

const mapDispatchToProps = dispatch => {

   return {

      navigate: (path) => dispatch(push(path)),

      adminLoginSuccess: (adminInfo) => dispatch(actions.adminLoginSuccess(adminInfo)),

      adminLoginFail: () => dispatch(actions.adminLoginFail()),

   };

};

export default connect(mapStateToProps, mapDispatchToProps)(Login);

##### **Login.scss**

#### **src/home/**

##### **Home.js**

import React, { Component } from 'react';

import { connect } from 'react-redux';

import { push } from "connected-react-router";

import \* as actions from "../../store/actions";

import './Home.scss';

import { FormattedMessage } from 'react-intl';

class Home extends Component {

   constructor(props) {

      super(props);

   }

   render() {

      return (

         <div>Home</div>

      )

   }

}

const mapStateToProps = state => {

   return {

      lang: state.app.language

   };

};

const mapDispatchToProps = dispatch => {

   return {

      navigate: (path) => dispatch(push(path)),

      adminLoginSuccess: (adminInfo) => dispatch(actions.adminLoginSuccess(adminInfo)),

      adminLoginFail: () => dispatch(actions.adminLoginFail()),

   };

};

export default connect(mapStateToProps, mapDispatchToProps)(Home);

##### **Home.scss**

#### **src/user/**

##### **Profile.js**

import React, { Component } from 'react';

import { connect } from 'react-redux';

import { push } from "connected-react-router";

import \* as actions from "../../store/actions";

import './Profile.scss';

import { FormattedMessage } from 'react-intl';

class Profile extends Component {

   constructor(props) {

      super(props);

   }

   render() {

      return (

         <div>Profile</div>

      )

   }

}

const mapStateToProps = state => {

   return {

      lang: state.app.language

   };

};

const mapDispatchToProps = dispatch => {

   return {

      navigate: (path) => dispatch(push(path)),

      adminLoginSuccess: (adminInfo) => dispatch(actions.adminLoginSuccess(adminInfo)),

      adminLoginFail: () => dispatch(actions.adminLoginFail()),

   };

};

export default connect(mapStateToProps, mapDispatchToProps)(Profile);

##### **Profile.scss**

### **src/routes/**

#### **AppRoutes.js**

import React from 'react';

import { Switch, Route, Redirect } from 'react-router-dom';

import PublicLayout from '../layouts/PublicLayout';

import AdminLayout from '../layouts/AdminLayout';

import PrivateRoute from '../hoc/PrivateRoute';

import Home from '../pages/home/Home';

import Login from '../pages/auth/Login';

import Profile from '../pages/user/Profile';

const AppRoutes = () => {

   return (

      <Switch>

         {/\* Trang HOME - không cần đăng nhập \*/}

         <Route

            path="/"

            exact

            render={(props) => (

               <PublicLayout>

                  <Home {...props} />

               </PublicLayout>

            )}

         />

         {/\* Login \*/}

         <Route

            path="/login"

            render={(props) => (

               <PublicLayout>

                  <Login {...props} />

               </PublicLayout>

            )}

         />

         {/\* Profile - cần login với role = 2 (user) \*/}

         <PrivateRoute

            path="/profile"

            role="2"

            render={(props) => (

               <PublicLayout>

                  <Profile {...props} />

               </PublicLayout>

            )}

         />

         {/\* Admin - cần login với role = 1 (admin) \*/}

         <PrivateRoute

            path="/admin"

            role="1"

            render={(props) => (

               <AdminLayout>

                  <div>Trang admin</div>

               </AdminLayout>

            )}

         />

         {/\* Redirect fallback \*/}

         <Redirect to="/" />

      </Switch>

   );

};

export default AppRoutes;

### **src/services/**

#### **adminService.js**

import axios from '../axios';

import \* as queryString from 'query-string';

const adminService = {

    /\*\*

     \* Đăng nhập hệ thống

     \* {

     \*  "username": "string",

     \*  "password": "string"

     \* }

     \*/

    login(loginBody) {

        return axios.post(`/admin/login`, loginBody)

    },

};

export default adminService;

#### **index.js**

export { default as adminService } from './adminService';

### **src/store/**

#### **src/store/actions/**

##### **actionType.js**

const actionTypes = Object.freeze({

    //app

    APP\_START\_UP\_COMPLETE: 'APP\_START\_UP\_COMPLETE',

    SET\_CONTENT\_OF\_CONFIRM\_MODAL: 'SET\_CONTENT\_OF\_CONFIRM\_MODAL',

    //admin

    ADMIN\_LOGIN\_SUCCESS: 'ADMIN\_LOGIN\_SUCCESS',

    ADMIN\_LOGIN\_FAIL: 'ADMIN\_LOGIN\_FAIL',

    PROCESS\_LOGOUT: 'PROCESS\_LOGOUT',

    //user

    ADD\_USER\_SUCCESS: 'ADD\_USER\_SUCCESS',

})

export default actionTypes;

##### **adminActions.js**

import actionTypes from './actionTypes';

export const adminLoginSuccess = (adminInfo) => ({

    type: actionTypes.ADMIN\_LOGIN\_SUCCESS,

    adminInfo: adminInfo

})

export const adminLoginFail = () => ({

    type: actionTypes.ADMIN\_LOGIN\_FAIL

})

export const processLogout = () => ({

    type: actionTypes.PROCESS\_LOGOUT

})

##### **appActions.js**

import actionTypes from './actionTypes';

export const appStartUpComplete = () => ({

    type: actionTypes.APP\_START\_UP\_COMPLETE

});

export const setContentOfConfirmModal = (contentOfConfirmModal) => ({

    type: actionTypes.SET\_CONTENT\_OF\_CONFIRM\_MODAL,

    contentOfConfirmModal: contentOfConfirmModal

});

##### **index.js**

export \* from './appActions'

export \* from './adminActions'

export \* from './userActions'

##### **userActions.js**

import actionTypes from './actionTypes';

export const addUserSuccess = () => ({

    type: actionTypes.ADD\_USER\_SUCCESS

})

#### **src/store/reducers/**

##### **adminReducer.js**

import actionTypes from '../actions/actionTypes';

const initialState = {

    isLoggedIn: false,

    adminInfo: null

}

const appReducer = (state = initialState, action) => {

    switch (action.type) {

        case actionTypes.ADMIN\_LOGIN\_SUCCESS:

            return {

                ...state,

                isLoggedIn: true,

                adminInfo: action.adminInfo

            }

        case actionTypes.ADMIN\_LOGIN\_FAIL:

            return {

                ...state,

                isLoggedIn: false,

                adminInfo: null

            }

        case actionTypes.PROCESS\_LOGOUT:

            return {

                ...state,

                isLoggedIn: false,

                adminInfo: null

            }

        default:

            return state;

    }

}

export default appReducer;

##### **appReducer.js**

import actionTypes from '../actions/actionTypes';

const initContentOfConfirmModal = {

    isOpen: false,

    messageId: "",

    handleFunc: null,

    dataFunc: null

}

const initialState = {

    started: true,

    language: 'vi',

    systemMenuPath: '/system/user-manage',

    contentOfConfirmModal: {

        ...initContentOfConfirmModal

    }

}

const appReducer = (state = initialState, action) => {

    switch (action.type) {

        case actionTypes.APP\_START\_UP\_COMPLETE:

            return {

                ...state,

                started: true

            }

        case actionTypes.SET\_CONTENT\_OF\_CONFIRM\_MODAL:

            return {

                ...state,

                contentOfConfirmModal: {

                    ...state.contentOfConfirmModal,

                    ...action.contentOfConfirmModal

                }

            }

        default:

            return state;

    }

}

export default appReducer;

##### **rootReducer.js**

import {combineReducers} from 'redux';

import { connectRouter } from 'connected-react-router';

import appReducer from "./appReducer";

import adminReducer from "./adminReducer";

import userReducer from "./userReducer";

import autoMergeLevel2 from 'redux-persist/lib/stateReconciler/autoMergeLevel2';

import storage from 'redux-persist/lib/storage';

import { persistReducer } from 'redux-persist';

const persistCommonConfig = {

    storage: storage,

    stateReconciler: autoMergeLevel2,

};

const adminPersistConfig = {

    ...persistCommonConfig,

    key: 'admin',

    whitelist: ['isLoggedIn', 'adminInfo']

};

export default (history) => combineReducers({

    router: connectRouter(history),

    admin: persistReducer(adminPersistConfig, adminReducer),

    user: userReducer,

    app: appReducer

})

##### **userReducer.js**

Đang phát triển…

### **src/styles/**

#### **\_base.scss**

/\*

 \* General styles

 \*/

 body, html {

    background-image: "#fff";

    background-repeat: no-repeat;

    background-size: cover;

    font-family: $fontmain;

    font-size: $base-size - 2px;

    height: 100vh;

    overflow: hidden;

}

.btn {

    padding: 0;

    border: none;

    cursor: pointer;

    font-weight: 700;

    height: 30px;

    line-height: 30px;

    -moz-user-select: none;

    -webkit-user-select: none;

    user-select: none;

    i {

      font-size: $base-size - 3px;

    }

}

.btn-edit,

.btn-add,

.btn-delete,

.btn-assign {

  background-color: $colormain;

  color: $common-text-color;

  &:hover {

    color: $common-text-color;

    background-color: darken($colormain, 5);

  }

}

#password:invalid {

  font-family: $fontmain; // Fixed TH hien thi placeholder cho EDGE

}

#password:-webkit-input-placeholder {

  font-family: $fontmain; // CHROME 1

}

#password::-webkit-input-placeholder {

  font-family: $fontmain; //CHROME 2   EDGE 1

}

#password:-moz-placeholder {

  font-family: $fontmain; // Firefox 4-18

}

#password::-moz-placeholder {

  font-family: $fontmain; // Firefox 19+

}

#password:-ms-input-placeholder {

  font-family: $fontmain; // IE 10-11

}

#password::-ms-input-placeholder {

  font-family: $fontmain; // EDGE 2

}

#password::placeholder {

  font-family: $fontmain;

}

.content-container {

  .title {

    text-align: center;

    text-transform: uppercase;

    font-weight: bold;

    font-size: $base-size + 4px;

    margin-top: 15px;

    color: $colormain;

  }

  .content {

    margin: 15px 30px;

    box-shadow: 2px 2px 4px 0 $box-shadow-color;

    -webkit-box-shadow: 2px 2px 4px 0 $box-shadow-color;

  }

}

.modal {

  .modal-dialog {

    .modal-content {

      .modal-header {

        padding: 0 10px;

        height: 35px;

        background-color: $colormain;

        color: $common-text-color;

        .modal-title {

          line-height: 35px;

          font-weight: 500;

          font-size: $base-size + 1px;

        }

        .btn-close {

          position: absolute;

          color: $common-text-color;

          i {

            line-height: 35px;

            font-size: $base-size + 2px;

          }

        }

      }

    }

  }

}

#### **\_form.scss**

@import "common";

.custom-form-group {

  .custom-form-control {

    background-color: $common-white;

    border: 1px solid $border;

    &.readonly {

      cursor: not-allowed;

      background-color: darken($common-white, 5);

    }

    &:focus {

      outline: none;

      border: 1px solid darken($border, 50);

    }

  }

  select.custom-form-control {

    background-image: $dropdown-image;

  }

  margin-bottom: 10px;

  label {

    margin-bottom: 1px;

  }

  .custom-form-control {

    display: block;

    width: 100%;

    height: 28px;

    padding: 0 5px;

    line-height: 26px;

    border-radius: 5px;

  }

  textarea.custom-form-control {

    min-height: 28px;

    height: auto;

  }

  select.custom-form-control {

    -webkit-appearance: none;

    -moz-appearance: none;

    padding-right: 15px;

    background-position: right top;

    background-repeat: no-repeat;

    background-size: 15px 100%;

    cursor: pointer;

    &:disabled {

      cursor: not-allowed;

    }

  }

  .custom-checkbox-control {

    line-height: 28px;

    input {

      cursor: pointer;

    }

    label {

      margin-left: 5px;

    }

  }

}

#### **\_variables.scss**

$base-size: 16px;

$lh: 1.313;

$fontmain:"Helvetica", -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Arial, "Noto Sans", sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji";

$colormain:#0071ba;

$colorsub:#bb8d09;

$common-text-color:white;

$common-btn-confirm: #185ba8;

$common-btn-deny: #bb8d09;

$common-error: #ff9600;

$common-notify: greenyellow;

$common-green: #00C087;

$common-red: #d01a1d;

$common-white: #fff;

$common-orange: #FFAC42;

$border: #ccc;

$text-in-light: #181818;

$input-focus: #FFAC42;

$footer-container: #f5f5f5;

$footer-text: #262626;

$footer-next-disable: #ccc;

$bg-menu-color: #f5f5f5;

$bg-scrollbar: #000;

$bg-table-color: #f5f5f5;

$box-shadow-color: #0000004d;

$dropdown-image: url("../assets/images/dropdown.svg");

//date picker

$date-picker-bg: $common-white;

$date-picker-border: darken(#3f4458, 50%);

$date-picker-arrow-hover: $common-orange;

$date-picker-month-fg: #000;

$date-picker-month-bg: $common-white;

$date-picker-weekdays-fg: #000;

$date-picker-weekdays-bg: transparent;

$date-picker-day-fg: #000;

$date-picker-day-hover-bg: darken($common-white, 25%);

$date-picker-today-border: $common-orange;

$date-picker-today-bg: $common-orange;

$date-picker-today-fg: #000;

$date-picker-day-selected-fg: $common-white;

#### **common.scss**

@import "variables";

.text-center {

    text-align: center !important;

}

.text-left {

    text-align: left !important;

}

.text-right {

    text-align: right !important;

}

.text-red {

    color: $common-red !important;

}

#### **styles.scss**

@import "variables";

@import "../../node\_modules/bootstrap/scss/bootstrap";

$fa-font-path : "~@fortawesome/fontawesome-free-webfonts/webfonts";

@import "~@fortawesome/fontawesome-free-webfonts/scss/fontawesome.scss";

@import "~@fortawesome/fontawesome-free-webfonts/scss/fa-solid.scss";

@import "~@fortawesome/fontawesome-free-webfonts/scss/fa-regular.scss";

@import "~@fortawesome/fontawesome-free-webfonts/scss/fa-brands.scss";

@import "base";

@import "form";

### **src/translations/**

#### **en.json**

{

    "common": {

        "add": "Add",

        "edit": "Edit",

        "delete": "Delete"

    },

    "login": {

        "login": "Login",

        "username": "Username",

        "password": "Password"

    }

}

#### **vi.json**

{

    "common": {

        "add": "Thêm",

        "edit": "Sửa",

        "delete": "Xóa"

    },

    "login": {

        "login": "Đăng nhập",

        "username": "Tên đăng nhập",

        "password": "Mật khẩu"

    }

}

### **src/utils/**

#### **CommonUtils.js**

class CommonUtils {

    static isNumber1 (number) {

        if (number === 1) return true;

        return false;

    }

}

export default CommonUtils;

#### **Constant.js**

export const path = {

    HOME: '/',

    LOGIN: '/login',

    LOG\_OUT: '/logout',

    SYSTEM: '/system'

};

export const languages = {

    VI: 'vi',

    EN: 'en'

};

export const manageActions = {

    ADD: "ADD",

    EDIT: "EDIT",

    DELETE: "DELETE"

};

export const dateFormat = {

    SEND\_TO\_SERVER: 'DD/MM/YYYY'

};

export const YesNoObj = {

    YES: 'Y',

    NO: 'N'

}

#### **index.js**

export \* from './constant';

export {default as CommonUtils} from './CommonUtils';

export {default as KeyCodeUtils} from './KeyCodeUtils';

export {default as LanguageUtils} from './LanguageUtils';

export {default as ToastUtil} from './ToastUtil';

#### **KeyCodeUtils.js**

class KeyCodeUtils {

    static UP = 38;

    static DOWN = 40;

    static TAB = 9;

    static ENTER = 13;

    static E = 69;

    static ESCAPE = 27;

    static isNavigation(e) {

        return (e >= 33 && e <= 40) || e === 9 || e === 8 || e === 46 || e === 14 || e === 13;

    }

    static isNumeric(e) {

        return (e >= 48 && e <= 57) || (e >= 96 && e <= 105);

    }

    static isAlphabetic(e) {

        return (e >= 65 && e <= 90);

    }

    static isDecimal(e) {

        return e === 190 || e === 188 || e === 108 || e === 110;

    }

    static isDash(e) {

        return e === 109 || e === 189;

    }

}

export default KeyCodeUtils;

#### **LanguageUtils.js**

import messages\_vi from '../translations/vi.json';

import messages\_en from '../translations/en.json';

const flattenMessages = ((nestedMessages, prefix = '') => {

    if (nestedMessages == null) {

        return {}

    }

    return Object.keys(nestedMessages).reduce((messages, key) => {

        const value = nestedMessages[key];

        const prefixedKey = prefix ? `${prefix}.${key}` : key;

        if (typeof value === 'string') {

            Object.assign(messages, {[prefixedKey]: value})

        } else {

            Object.assign(messages, flattenMessages(value, prefixedKey))

        }

        return messages

    }, {})

});

const messages = {

    'vi': flattenMessages(messages\_vi),

    'en': flattenMessages(messages\_en),

};

export default class LanguageUtils {

    static getMessageByKey(key, lang) {

        return messages[lang][key]

    }

    static getFlattenedMessages() {

        return messages;

    }

}

#### **ToastUtil.js**

import React from 'react';

import { toast } from 'react-toastify';

import axios from 'axios';

import CustomToast from "../components/CustomToast";

const TYPE\_SUCCESS = 'SUCCESS';

const TYPE\_INFO = 'INFO';

const TYPE\_WARN = 'WARN';

const TYPE\_ERROR = 'ERROR';

class ToastUtil {

    static success(title, message) {

        this.show(TYPE\_SUCCESS, title, message);

    }

    static info(title, message) {

        this.show(TYPE\_INFO, title, message);

    }

    static warn(title, message) {

        this.show(TYPE\_WARN, title, message);

    }

    static error(title, message) {

        this.show(TYPE\_ERROR, title, message);

    }

    static successRaw(title, message) {

        this.show(TYPE\_SUCCESS, title, message, true);

    }

    static errorRaw(title, message, autoCloseDelay = 3000) {

        this.show(TYPE\_ERROR, title, message, true, autoCloseDelay);

    }

    static errorApi(error, title = 'common.fail-to-load-data', autoCloseDelay = 3000) {

        if (axios.isCancel(error)) {

            // Do nothing if request was cancelled

            return;

        }

        let message = null;

        let messageId = 'common.unknown-error';

        if (error.httpStatusCode >= 500) {

            messageId = 'common.internal-server-error';

        } else if (error.httpStatusCode < 500 && error.httpStatusCode >= 400) {

            if (error.httpStatusCode === 400) {

                messageId = 'common.bad-request';

            } else if (error.httpStatusCode === 403) {

                messageId = 'common.forbiden-request';

            }

        } else {

            // Request fail even server was returned a success response

            if (error.errorMessage) {

                message = error.errorMessage

            }

        }

        toast.error(<CustomToast titleId={title} message={message} messageId={messageId} time={new Date()} />, {

            position: toast.POSITION.BOTTOM\_RIGHT,

            pauseOnHover: true,

            autoClose: autoCloseDelay

        });

    }

    static show(type, title, message, rawMessage = false, autoCloseDelay = 3000) {

        const content = <CustomToast titleId={title} messageId={rawMessage ? null : message} message={rawMessage ? message : null} time={new Date()} />;

        const options = {

            position: toast.POSITION.BOTTOM\_RIGHT,

            pauseOnHover: true,

            autoClose: autoCloseDelay

        };

        switch (type) {

            case TYPE\_SUCCESS:

                toast.success(content, options);

                break;

            case TYPE\_INFO:

                toast.info(content, options);

                break;

            case TYPE\_WARN:

                toast.warn(content, options);

                break;

            case TYPE\_ERROR:

                toast.error(content, options);

                break;

            default:

                break;

        }

    }

}

export default ToastUtil;

### **src/App.js/**

import React from 'react';

import { BrowserRouter } from 'react-router-dom';

import AppRoutes from './routes/AppRoutes';

import { Provider } from 'react-redux';

import store from './redux';

function App() {

    return (

        <Provider store={store}>

            <BrowserRouter>

                <AppRoutes />

            </BrowserRouter>

        </Provider>

    );

}

export default App;

### **src/App.scss/**

@import "../styles/common.scss";

### **src/axios.js/**

import axios from 'axios';

import \_ from 'lodash';

const instance = axios.create({

    baseURL: process.env.REACT\_APP\_BACKEND\_URL,

    withCredentials: true,

});

const createError = (httpStatusCode, statusCode, errorMessage, problems, errorCode = '') => {

    const error = new Error();

    error.httpStatusCode = httpStatusCode;

    error.statusCode = statusCode;

    error.errorMessage = errorMessage;

    error.problems = problems;

    error.errorCode = String(errorCode);

    return error;

};

export const isSuccessStatusCode = (s) => {

    return (typeof s === 'number' && s === 0) || (typeof s === 'string' && s.toUpperCase() === 'OK');

};

instance.interceptors.response.use(

    (response) => {

        const { data } = response;

        if (data.hasOwnProperty('s') && !isSuccessStatusCode(data.s) && data.hasOwnProperty('errmsg')) {

            return Promise.reject(createError(response.status, data.s, data.errmsg, null, data.errcode || ""));

        }

        if (data.hasOwnProperty('s') && data.hasOwnProperty('d')) return data.d;

        if (data.hasOwnProperty('s') && \_.keys(data).length === 1) return null;

        return data;

    },

    (error) => {

        const { response } = error;

        if (!response) return Promise.reject(error);

        const { data } = response;

        if (data?.s && data?.errmsg) {

            return Promise.reject(createError(response.status, data.s, data.errmsg));

        }

        if (data?.code && data?.message) {

            return Promise.reject(createError(response.status, data.code, data.message, data.problems));

        }

        return Promise.reject(createError(response.status));

    }

);

export default instance;

### **src/index.js/**

import React from 'react';

import ReactDOM from 'react-dom';

import 'react-toastify/dist/ReactToastify.css';

import './styles/styles.scss';

import App from './App';

import \* as serviceWorker from './serviceWorker';

import IntlProviderWrapper from "./hoc/IntlProviderWrapper";

import { Provider } from 'react-redux';

import reduxStore, { persistor } from './redux';

const renderApp = () => {

    ReactDOM.render(

        <Provider store={reduxStore}>

            <IntlProviderWrapper>

                <App persistor={persistor} />

            </IntlProviderWrapper>

        </Provider>,

        document.getElementById('root')

    );

};

renderApp();

// If you want your app to work offline and load faster, you can change

// unregister() to register() below. Note this comes with some pitfalls.

// Learn more about service workers: https://bit.ly/CRA-PWA

serviceWorker.unregister();

### **src/redux.js/**

import { logger } from "redux-logger";

import thunkMiddleware from "redux-thunk";

import { routerMiddleware } from 'connected-react-router';

import { createBrowserHistory } from 'history';

import { createStore, applyMiddleware, compose } from 'redux';

import { createStateSyncMiddleware } from 'redux-state-sync';

import { persistStore } from 'redux-persist';

import createRootReducer from './store/reducers/rootReducer';

import actionTypes from './store/actions/actionTypes';

const environment = process.env.NODE\_ENV || "development";

let isDevelopment = environment === "development";

//hide redux logs

isDevelopment = false;

export const history = createBrowserHistory({ basename: process.env.REACT\_APP\_ROUTER\_BASE\_NAME });

const reduxStateSyncConfig = {

    whitelist: [

        actionTypes.APP\_START\_UP\_COMPLETE,

    ]

}

const rootReducer = createRootReducer(history);

const middleware = [

    routerMiddleware(history),

    thunkMiddleware,

    createStateSyncMiddleware(reduxStateSyncConfig),

]

if (isDevelopment) middleware.push(logger);

const composeEnhancers = (isDevelopment && window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_COMPOSE\_\_) ? window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_COMPOSE\_\_ : compose;

const reduxStore = createStore(

    rootReducer,

    composeEnhancers(applyMiddleware(...middleware)),

)

export const dispatch = reduxStore.dispatch;

export const persistor = persistStore(reduxStore);

export default reduxStore;

### **src/serviceWorker.js/**

// This optional code is used to register a service worker.

// register() is not called by default.

// This lets the app load faster on subsequent visits in production, and gives

// it offline capabilities. However, it also means that developers (and users)

// will only see deployed updates on subsequent visits to a page, after all the

// existing tabs open on the page have been closed, since previously cached

// resources are updated in the background.

// To learn more about the benefits of this model and instructions on how to

// opt-in, read https://bit.ly/CRA-PWA

const isLocalhost = Boolean(

  window.location.hostname === 'localhost' ||

    // [::1] is the IPv6 localhost address.

    window.location.hostname === '[::1]' ||

    // 127.0.0.1/8 is considered localhost for IPv4.

    window.location.hostname.match(

      /^127(?:\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)){3}$/

    )

);

export function register(config) {

  if (process.env.NODE\_ENV === 'production' && 'serviceWorker' in navigator) {

    // The URL constructor is available in all browsers that support SW.

    const publicUrl = new URL(process.env.PUBLIC\_URL, window.location.href);

    if (publicUrl.origin !== window.location.origin) {

      // Our service worker won't work if PUBLIC\_URL is on a different origin

      // from what our page is served on. This might happen if a CDN is used to

      // serve assets; see https://github.com/facebook/create-react-app/issues/2374

      return;

    }

    window.addEventListener('load', () => {

      const swUrl = `${process.env.PUBLIC\_URL}/service-worker.js`;

      if (isLocalhost) {

        // This is running on localhost. Let's check if a service worker still exists or not.

        checkValidServiceWorker(swUrl, config);

        // Add some additional logging to localhost, pointing developers to the

        // service worker/PWA documentation.

        navigator.serviceWorker.ready.then(() => {

          console.log(

            'This web app is being served cache-first by a service ' +

              'worker. To learn more, visit https://bit.ly/CRA-PWA'

          );

        });

      } else {

        // Is not localhost. Just register service worker

        registerValidSW(swUrl, config);

      }

    });

  }

}

function registerValidSW(swUrl, config) {

  navigator.serviceWorker

    .register(swUrl)

    .then(registration => {

      registration.onupdatefound = () => {

        const installingWorker = registration.installing;

        if (installingWorker == null) {

          return;

        }

        installingWorker.onstatechange = () => {

          if (installingWorker.state === 'installed') {

            if (navigator.serviceWorker.controller) {

              // At this point, the updated precached content has been fetched,

              // but the previous service worker will still serve the older

              // content until all client tabs are closed.

              console.log(

                'New content is available and will be used when all ' +

                  'tabs for this page are closed. See https://bit.ly/CRA-PWA.'

              );

              // Execute callback

              if (config && config.onUpdate) {

                config.onUpdate(registration);

              }

            } else {

              // At this point, everything has been precached.

              // It's the perfect time to display a

              // "Content is cached for offline use." message.

              console.log('Content is cached for offline use.');

              // Execute callback

              if (config && config.onSuccess) {

                config.onSuccess(registration);

              }

            }

          }

        };

      };

    })

    .catch(error => {

      console.error('Error during service worker registration:', error);

    });

}

function checkValidServiceWorker(swUrl, config) {

  // Check if the service worker can be found. If it can't reload the page.

  fetch(swUrl)

    .then(response => {

      // Ensure service worker exists, and that we really are getting a JS file.

      const contentType = response.headers.get('content-type');

      if (

        response.status === 404 ||

        (contentType != null && contentType.indexOf('javascript') === -1)

      ) {

        // No service worker found. Probably a different app. Reload the page.

        navigator.serviceWorker.ready.then(registration => {

          registration.unregister().then(() => {

            window.location.reload();

          });

        });

      } else {

        // Service worker found. Proceed as normal.

        registerValidSW(swUrl, config);

      }

    })

    .catch(() => {

      console.log(

        'No internet connection found. App is running in offline mode.'

      );

    });

}

export function unregister() {

  if ('serviceWorker' in navigator) {

    navigator.serviceWorker.ready.then(registration => {

      registration.unregister();

    });

  }

}

## **.env/**

PORT=3000

NODE\_ENV=development

REACT\_APP\_BACKEND\_URL=http://localhost:8080

#The base URL for all locations. If your app is served from a sub-directory on your server, you'll want to set

#this to the sub-directory. A properly formatted basename should have a leading slash, but no trailing slash.

REACT\_APP\_ROUTER\_BASE\_NAME=

## **.gitignore/**

# See https://help.github.com/articles/ignoring-files/ for more about ignoring files.

# dependencies

/node\_modules

/.pnp

.pnp.js

# testing

/coverage

# production

/build

# misc

.DS\_Store

.env.local

.env.development.local

.env.test.local

.env.production.local

npm-debug.log\*

yarn-debug.log\*

yarn-error.log\*

## **package.json/**

{

  "name": "nguyenlien",

  "version": "0.1.0",

  "private": true,

  "dependencies": {

    "@formatjs/intl-pluralrules": "^3.5.6",

    "@formatjs/intl-relativetimeformat": "^7.3.6",

    "@fortawesome/fontawesome-free-webfonts": "^1.0.9",

    "axios": "^0.21.1",

    "bootstrap": "^5.0.1",

    "connected-react-router": "^6.9.1",

    "lodash": "^4.17.21",

    "moment": "^2.29.0",

    "react": "^17.0.2",

    "react-auth-wrapper": "^1.0.0",

    "react-autosuggest": "^10.1.0",

    "react-custom-scrollbars-2": "^4.5.0",

    "react-dom": "^17.0.2",

    "react-flatpickr": "^3.10.7",

    "react-intl": "^5.20.2",

    "react-redux": "^7.2.4",

    "react-router": "^5.2.0",

    "react-router-dom": "^5.2.0",

    "react-scripts": "^4.0.3",

    "react-toastify": "^5.5.0",

    "reactstrap": "^8.9.0",

    "redux": "^4.1.0",

    "redux-auth-wrapper": "^2.1.0",

    "redux-logger": "^3.0.6",

    "redux-persist": "^5.10.0",

    "redux-state-sync": "^2.1.0",

    "redux-thunk": "^2.3.0",

    "typescript": "^4.3.2"

  },

  "scripts": {

    "start": "set NODE\_OPTIONS=--openssl-legacy-provider && react-scripts start",

    "build": "react-scripts build",

    "test": "react-scripts test",

    "eject": "react-scripts eject"

  },

  "eslintConfig": {

    "extends": "react-app"

  },

  "browserslist": {

    "production": [

      ">0.2%",

      "not dead",

      "not op\_mini all"

    ],

    "development": [

      "last 1 chrome version",

      "last 1 firefox version",

      "last 1 safari version"

    ]

  },

  "devDependencies": {

    "sass": "^1.89.2"

  }

}

# **Database – MySQL**

Ảnh có chứa văn bản, Phông chữ, số, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.