Employee Profile & Edit Profile Implementation Guide

This document contains all the code changes needed to implement employee profile viewing and editing functionality.

Backend Changes

1. EmployeeController.java - Add New Endpoints

File: `backend/src/main/java/com/corehive/backend/controller/EmployeeController.java`

Add these two methods to the controller:

```java
/**
 * Get current logged-in employee's profile
 */
@GetMapping("/me")
@PreAuthorize("hasRole('EMPLOYEE')")
public ResponseEntity<ApiResponse<Employee>> getCurrentEmployeeProfile(HttpServletRequest reque
    String userEmail = (String) request.getAttribute("userEmail");
    String organizationUuid = (String) request.getAttribute("organizationUuid");

    if (userEmail == null || organizationUuid == null) {
        log.warn("User email or organization UUID not found in request");
        return ResponseEntity.badRequest()
                .body(ApiResponse.error("Authentication information not found"));
    }

    log.info("Fetching profile for employee: {}", userEmail);
    ApiResponse<Employee> response = employeeService.getEmployeeByEmail(organizationUuid, userEr
    return ResponseEntity.ok(response);
}

/**
 * Update current logged-in employee's profile
 */
@PutMapping("/me")
@PreAuthorize("hasRole('EMPLOYEE')")
public ResponseEntity<ApiResponse<Employee>> updateCurrentEmployeeProfile(
        HttpServletRequest request,
        @Valid @RequestBody EmployeeRequestDTO employeeRequest) {

    String userEmail = (String) request.getAttribute("userEmail");
    String organizationUuid = (String) request.getAttribute("organizationUuid");

    if (userEmail == null || organizationUuid == null) {
        log.warn("User email or organization UUID not found in request");
        return ResponseEntity.badRequest()
                .body(ApiResponse.error("Authentication information not found"));
    }

    log.info("Updating profile for employee: {}", userEmail);
    ApiResponse<Employee> response = employeeService.updateEmployeeByEmail(organizationUuid, use
    return ResponseEntity.ok(response);
}
```

2. EmployeeService.java - Add Service Methods

File: `backend/src/main/java/com/corehive/backend/service/EmployeeService.java`

Add these two methods to the service:

```java
/**
 * Get employee by email (for current user profile)
 */
public ApiResponse<Employee> getEmployeeByEmail(String organizationUuid, String email) {
    try {
        log.info("Fetching employee with email: {} for organization: {}", email, organizationUu

        Optional<Employee> employeeOpt = employeeRepository.findByEmailAndOrganizationUuid(emai

        if (employeeOpt.isEmpty()) {
            log.warn("Employee not found with email: {}", email);
            return ApiResponse.error("Employee profile not found");
        }

        Employee employee = employeeOpt.get();
        log.info("Employee profile retrieved successfully for: {}", email);
        return ApiResponse.success("Employee profile retrieved successfully", employee);

    } catch (Exception e) {
        log.error("Error fetching employee with email: {}", email, e);
        return ApiResponse.error("Failed to retrieve employee profile");
    }
}

/**
 * Update employee by email (for current user profile)
 */
@Transactional
public ApiResponse<Employee> updateEmployeeByEmail(String organizationUuid, String email, Emplo
    try {
        log.info("Updating employee with email: {} for organization: {}", email, organizationUu

        Optional<Employee> employeeOpt = employeeRepository.findByEmailAndOrganizationUuid(emai

        if (employeeOpt.isEmpty()) {
            log.warn("Employee not found with email: {}", email);
            return ApiResponse.error("Employee profile not found");
        }

        Employee employee = employeeOpt.get();

        // Employees can only update limited fields (not email, salary, department, etc.)
        employee.setFirstName(request.getFirstName());
```

```java
            employee.setLastName(request.getLastName());
            employee.setPhone(request.getPhone());
            employee.setUpdatedAt(LocalDateTime.now());

            Employee savedEmployee = employeeRepository.save(employee);

            log.info("Employee profile updated successfully for: {}", email);
            return ApiResponse.success("Profile updated successfully", savedEmployee);

        } catch (Exception e) {
            log.error("Error updating employee with email: {}", email, e);
            return ApiResponse.error("Failed to update profile: " + e.getMessage());
        }
    }
```

3. EmployeeRepository.java - Add Query Method

File: `backend/src/main/java/com/corehive/backend/repository/EmployeeRepository.java`

Add this method to the repository interface:

```java
Optional<Employee> findByEmailAndOrganizationUuid(String email, String organizationUuid);
```

Frontend Changes

4. employeeApi.js - Add API Functions

File: `frontend/src/api/employeeApi.js`

Add these functions to the file:

```javascript
/**
 * Get current logged-in employee's profile
 */
export const getCurrentEmployeeProfile = async () => {
  try {
    const response = await apiClient.get('/employees/me');
    return response.data;
  } catch (error) {
    console.error('Error fetching current employee profile:', error);
    throw error;
  }
};

/**
 * Update current logged-in employee's profile
 */
export const updateCurrentEmployeeProfile = async (employeeData) => {
  try {
    const response = await apiClient.put('/employees/me', employeeData);
    return response.data;
  } catch (error) {
    console.error('Error updating current employee profile:', error);
    throw error;
  }
};
```

Update the default export to include new functions:

```javascript
export default {
  getAllEmployees,
  getEmployeeById,
  createEmployee,
  updateEmployee,
  deleteEmployee,
  getCurrentEmployeeProfile,
  updateCurrentEmployeeProfile
};
```

5. EmployeeProfile.jsx - Complete Component

File: `frontend/src/pages/employee/EmployeeProfile.jsx`

Replace the entire file content with:

```
import React, { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import DashboardLayout from '../../components/layout/DashboardLayout';
import LoadingSpinner from '../../components/common/LoadingSpinner';
import Alert from '../../components/common/Alert';
import { getCurrentEmployeeProfile } from '../../api/employeeApi';

export default function EmployeeProfile() {
  const navigate = useNavigate();
  const [employee, setEmployee] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    fetchEmployeeProfile();
  }, []);

  const fetchEmployeeProfile = async () => {
    try {
      setLoading(true);
      setError(null);
      const response = await getCurrentEmployeeProfile();

      if (response.success) {
        setEmployee(response.data);
      } else {
        setError(response.message || 'Failed to load profile');
      }
    } catch (err) {
      console.error('Error fetching employee profile:', err);
      setError('Failed to load profile. Please try again.');
    } finally {
      setLoading(false);
    }
  };

  const getInitials = () => {
    if (!employee) return '';
    return `${employee.firstName?.charAt(0) || ''}${employee.lastName?.charAt(0) || ''}`.toUppe
  };

  const formatDate = (date) => {
    if (!date) return 'N/A';
    return new Date(date).toLocaleDateString('en-US', {
      year: 'numeric',
      month: 'long',
      day: 'numeric'
    });
  };

  if (loading) {
    return (
      <DashboardLayout>
        <div className="flex justify-center items-center min-h-screen">
          <LoadingSpinner />
        </div>
      </DashboardLayout>
    );
  }

  if (error) {
    return (
      <DashboardLayout>
        <div className="px-10 mt-10">
          <Alert type="error" message={error} />
        </div>
      </DashboardLayout>
    );
  }

  if (!employee) {
    return (
      <DashboardLayout>
        <div className="px-10 mt-10">
          <Alert type="error" message="Employee profile not found" />
        </div>
      </DashboardLayout>
    );
  }

  return (
    <DashboardLayout>
      <div className="bg-gray-100 min-h-screen">
        {/* MAIN PROFILE CARD */}
        <div className="px-10 mt-10">
          <div className="bg-white p-10 rounded-xl shadow w-full">
```

```
{/* Header with Edit Button */}
<div className="flex justify-between items-center mb-8">
  <h2 className="text-xl font-semibold">My Profile</h2>
  <button
    onClick={() => navigate('/employee/edit-profile')}
    className="px-6 py-2 bg-blue-600 text-white rounded-lg hover:bg-blue-700
             transition-colors duration-200 flex items-center gap-2"
  >
    <svg className="w-5 h-5" fill="none" stroke="currentColor" viewBox="0 0 24 24">
      <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2}
            d="M11 5H6a2 2 0 00-2 2v11a2 2 0 002 2h11a2 2 0 002-2v-5m-1.414-9.414a2
    </svg>
    Edit Profile
  </button>
</div>

<div className="flex gap-16">
  {/* LEFT SIDE */}
  <div className="w-1/4 text-center">
    <div className="bg-gradient-to-br from-blue-700 to-blue-400 text-white
                    w-40 h-40 rounded-full flex items-center justify-center
                    text-5xl font-semibold mx-auto">
      {getInitials()}
    </div>

    <h3 className="text-xl font-semibold mt-6">
      {employee.firstName} {employee.lastName}
    </h3>
    <p className="text-gray-500">{employee.designation || 'N/A'}</p>
    <p className="text-gray-600 mt-2">ID: {employee.employeeCode}</p>

    {/* Status Badge */}
    <div className="mt-4">
      <span className={`px-4 py-1 rounded-full text-sm font-medium ${
        employee.isActive
          ? 'bg-green-100 text-green-800'
          : 'bg-red-100 text-red-800'
      }`}>
        {employee.isActive ? 'Active' : 'Inactive'}
      </span>
    </div>
  </div>
```

```
              {/* RIGHT SIDE */}
              <div className="w-3/4">
                {/* Contact Info */}
                <div className="mb-10">
                  <h3 className="text-lg font-semibold mb-3 text-gray-800">Contact Information<
                  <div className="space-y-3 text-gray-700">
                    <p><strong>Email:</strong> {employee.email}</p>
                    <p><strong>Phone:</strong> {employee.phone || 'N/A'}</p>
                    <p><strong>National ID:</strong> {employee.nationalId || 'N/A'}</p>
                  </div>
                </div>

                <hr className="my-6" />

                {/* Employment Info */}
                <div className="mb-10">
                  <h3 className="text-lg font-semibold mb-3 text-gray-800">Employment Informati
                  <div className="space-y-3 text-gray-700">
                    <p><strong>Department:</strong> {employee.department?.name || 'N/A'}</p>
                    <p><strong>Position:</strong> {employee.designation || 'N/A'}</p>
                    <p><strong>Date of Joining:</strong> {formatDate(employee.dateOfJoining)}</
                    <p><strong>Salary Type:</strong> {employee.salaryType}</p>
                    <p><strong>Leave Balance:</strong> {employee.leaveCount} days</p>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </DashboardLayout>
  );
}
```

6. EditProfile.jsx - Complete Component

File: `frontend/src/pages/employee/EditProfile.jsx`

Replace the entire file content with:

```
import React, { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import DashboardLayout from '../../components/layout/DashboardLayout';
import LoadingSpinner from '../../components/common/LoadingSpinner';
import Alert from '../../components/common/Alert';
import Input from '../../components/common/Input';
import Button from '../../components/common/Button';
import { getCurrentEmployeeProfile, updateCurrentEmployeeProfile } from '../../api/employeeApi'

const EditProfile = () => {
  const navigate = useNavigate();
  const [loading, setLoading] = useState(true);
  const [saving, setSaving] = useState(false);
  const [error, setError] = useState(null);
  const [success, setSuccess] = useState(null);

  const [formData, setFormData] = useState({
    firstName: '',
    lastName: '',
    phone: '',
    email: '',
    designation: '',
    nationalId: '',
    department: '',
    dateOfJoining: '',
    salaryType: '',
    leaveCount: 0,
    basicSalary: 0,
    status: 'Active'
  });

  useEffect(() => {
    fetchEmployeeProfile();
  }, []);

  const fetchEmployeeProfile = async () => {
    try {
      setLoading(true);
      setError(null);
      const response = await getCurrentEmployeeProfile();

      if (response.success) {
        const employee = response.data;
        setFormData({
          firstName: employee.firstName || '',
          lastName: employee.lastName || '',
          phone: employee.phone || '',
          email: employee.email || '',
          designation: employee.designation || '',
          nationalId: employee.nationalId || '',
          department: employee.departmentId || '',
          dateOfJoining: employee.dateOfJoining || '',
          salaryType: employee.salaryType || 'MONTHLY',
          leaveCount: employee.leaveCount || 0,
          basicSalary: employee.basicSalary || 0,
          status: employee.isActive ? 'Active' : 'Inactive'
        });
      } else {
        setError(response.message || 'Failed to load profile');
      }
    } catch (err) {
      console.error('Error fetching employee profile:', err);
      setError('Failed to load profile. Please try again.');
    } finally {
      setLoading(false);
    }
  };

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData(prev => ({
      ...prev,
      [name]: value
    }));
  };

  const handleSubmit = async (e) => {
    e.preventDefault();

    // Validate required fields
    if (!formData.firstName.trim() || !formData.lastName.trim()) {
      setError('First name and last name are required');
      return;
    }

    try {
```

```
    setSaving(true);
    setError(null);
    setSuccess(null);

    const response = await updateCurrentEmployeeProfile(formData);

    if (response.success) {
      setSuccess('Profile updated successfully!');
      setTimeout(() => {
        navigate('/employee/profile');
      }, 1500);
    } else {
      setError(response.message || 'Failed to update profile');
    }
  } catch (err) {
    console.error('Error updating profile:', err);
    setError(err.response?.data?.message || 'Failed to update profile. Please try again.');
  } finally {
    setSaving(false);
  }
};

if (loading) {
  return (
    <DashboardLayout>
      <div className="flex justify-center items-center min-h-screen">
        <LoadingSpinner />
      </div>
    </DashboardLayout>
  );
}

return (
  <DashboardLayout>
    <div className="bg-gray-100 min-h-screen">
      <div className="px-10 mt-10">
        <div className="bg-white p-10 rounded-xl shadow w-full max-w-4xl mx-auto">

          {/* Header */}
          <div className="flex justify-between items-center mb-8">
            <h2 className="text-2xl font-semibold">Edit Profile</h2>
            <button
              onClick={() => navigate('/employee/profile')}
              className="px-4 py-2 text-gray-600 hover:text-gray-800 flex items-center gap-2"
            >
              <svg className="w-5 h-5" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M6 18L1
              </svg>
              Cancel
            </button>
          </div>

          {/* Alerts */}
          {error && <Alert type="error" message={error} onClose={() => setError(null)} />}
          {success && <Alert type="success" message={success} onClose={() => setSuccess(null)

          {/* Form */}
          <form onSubmit={handleSubmit} className="space-y-6">

            {/* Personal Information */}
            <div>
              <h3 className="text-lg font-semibold mb-4 text-gray-800">Personal Information</
              <div className="grid grid-cols-2 gap-6">
                <Input
                  label="First Name"
                  name="firstName"
                  value={formData.firstName}
                  onChange={handleChange}
                  required
                  placeholder="Enter first name"
                />
                <Input
                  label="Last Name"
                  name="lastName"
                  value={formData.lastName}
                  onChange={handleChange}
                  required
                  placeholder="Enter last name"
                />
                <Input
                  label="Phone"
                  name="phone"
                  type="tel"
                  value={formData.phone}
                  onChange={handleChange}
                  placeholder="Enter phone number"
              />
              <Input
                label="National ID"
                name="nationalId"
                value={formData.nationalId}
                onChange={handleChange}
                disabled
                className="bg-gray-100"
                helperText="National ID cannot be changed"
              />
            </div>
          </div>

          <hr className="my-6" />

          {/* Employment Information (Read-only) */}
          <div>
            <h3 className="text-lg font-semibold mb-4 text-gray-800">Employment Information
            <p className="text-sm text-gray-500 mb-4">
              The following information is managed by your HR department and cannot be edit
            </p>
            <div className="grid grid-cols-2 gap-6">
              <Input
                label="Email"
                name="email"
                type="email"
                value={formData.email}
                disabled
                className="bg-gray-100"
              />
              <Input
                label="Designation"
                name="designation"
                value={formData.designation}
                disabled
                className="bg-gray-100"
              />
              <Input
                label="Date of Joining"
                name="dateOfJoining"
                type="date"
                value={formData.dateOfJoining}
                disabled
```

```
                className="bg-gray-100"
              />
              <Input
                label="Salary Type"
                name="salaryType"
                value={formData.salaryType}
                disabled
                className="bg-gray-100"
              />
            </div>
          </div>

          {/* Action Buttons */}
          <div className="flex justify-end gap-4 pt-6">
            <Button
              type="button"
              variant="secondary"
              onClick={() => navigate('/employee/profile')}
              disabled={saving}
            >
              Cancel
            </Button>
            <Button
              type="submit"
              variant="primary"
              disabled={saving}
            >
              {saving ? 'Saving...' : 'Save Changes'}
            </Button>
          </div>
        </form>
      </div>
    </div>
  </div>
</DashboardLayout>
);
};

export default EditProfile;
```

Implementation Summary

Backend Endpoints

1. **GET** `/api/employees/me` - Get current employee profile (EMPLOYEE role required)
2. **PUT** `/api/employees/me` - Update current employee profile (EMPLOYEE role required)

Editable Fields for Employees

Employees can only edit:

- First Name
- Last Name
- Phone

Read-only Fields

The following fields are managed by HR and cannot be edited by employees:

- Email
- National ID
- Designation
- Department
- Date of Joining
- Salary Type
- Leave Count
- Basic Salary

Navigation Flow

1. Employee Profile Page (`/employee/profile`) - View profile details
2. Click "Edit Profile" button
3. Edit Profile Page (`/employee/edit-profile`) - Edit allowed fields
4. Save changes and redirect back to profile page

Testing Checklist

- [ ] Backend endpoints return correct data
- [ ] Employee can view their profile
- [ ] Edit button navigates to edit page
- [ ] Only allowed fields are editable
- [ ] Read-only fields are disabled
- [ ] Form validation works
- [ ] Success message shows after update
- [ ] Redirects to profile after successful update
- [ ] Error handling works correctly
- [ ] Loading states display properly