```
using Meditech.App.CommonService.Services;
using Meditech.Lib.Share.CommonClass.Paging;
using Microsoft.AspNetCore.Mvc;
using Meditech.Core.Auth.Entities;
using Meditech.Core.Auth.Authorization;
using Meditech.App.CommonService.ViewModels;
using Meditech.Lib.Share;
using Newtonsoft.Json;
using Meditech.App.CommonService.Dtos;
using Meditech.Lib.CommonShare;
using Meditech.App.CommonService.Contanst;

namespace Meditech.App.CommonService.Controllers
{
    [MediAuthorize]
    [Route("api/[controller]")]
    [ApiController]
    public class SignatureProcessController : ControllerBase
    {
        private readonly ILogger<SignatureProcessController> _logger;
        private readonly ISignatureProcessService _signatureProcessService;

        public SignatureProcessController(ILogger<SignatureProcessController> logger,
            ISignatureProcessService signatureProcessService)
        {
            _logger = logger;
            _signatureProcessService = signatureProcessService;
        }

        [HttpPost]
        [Route("search-process")]
        //[MediAuthorize(new[] { Lib.Share.PermitValue.ReadAll }, Lib.Share.ActionPermit.Common_SignatureProcess)]
        public IActionResult GetAllSignatureProcess(SignatureProcessSearchQuery search)
        {
            try
            {
                var requestUser = (BaseUser)Request.HttpContext.Items["User"];
                if (requestUser == null)
                    return Forbid();

                var data = _signatureProcessService.Search(search);
                return Ok(new { Data = data });
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, $"GetAll SignatureProcess Error. ");
                return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
            }
        }

        [HttpGet]
        [Route("get-by-id")]
        //[MediAuthorize(new[] { Lib.Share.PermitValue.Read }, Lib.Share.ActionPermit.Common_SignatureProcess)]
        public IActionResult GetSignatureProcessById(Guid id)
        {
            try
            {
                var requestUser = (BaseUser)Request.HttpContext.Items["User"];
                if (requestUser == null)
                    return Forbid();

                var data = _signatureProcessService.getById(id);

                if (data.IsOk)
                    return Ok(new { Data = data.Data });
                return StatusCode(600, data);
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, $"Get SignatureProcessById Error. ");
                return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
            }
        }
```

```
[HttpPost]
[Route("create")]
[MediAuthorize(new[] { Lib.Share.PermitValue.ChoPhep }, Lib.Share.ActionPermit.Common_SignatureProcess)]
public IActionResult CreateSignatureProcess(SignatureProcessViewModel model)
{
    try
    {
        var requestUser = (BaseUser)Request.HttpContext.Items["User"];
        if (requestUser == null)
            return Forbid();

        var data = _signatureProcessService.Create(model, requestUser.Id);
        if (data.IsOk)
            return Ok(new { Data = data });
        return StatusCode(600, data);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"Create SignatureProcess Error. ");
        return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
    }
}

[HttpPut]
[Route("update")]
[MediAuthorize(new[] { Lib.Share.PermitValue.ChoPhep }, Lib.Share.ActionPermit.Common_SignatureProcess)]
public IActionResult UpdateSignatureProcess(UpdateSignatureProcessViewModel model)
{
    try
    {
        var requestUser = (BaseUser)Request.HttpContext.Items["User"];
        if (requestUser == null)
            return Forbid();
        var data = _signatureProcessService.Update(model);
        if (data.IsOk)
            return Ok(new { Data = data });
        return StatusCode(600, data);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"Update SignatureProcess Error. ");
        return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
    }
}

[HttpDelete]
[Route("delete")]
[MediAuthorize(new[] { Lib.Share.PermitValue.ChoPhep }, Lib.Share.ActionPermit.Common_SignatureProcess)]
public IActionResult DeleteSignatureProcess(Guid id)
{
    try
    {
        var requestUser = (BaseUser)Request.HttpContext.Items["User"];
        if (requestUser == null)
            return Forbid();

        var data = _signatureProcessService.Delete(id);
        if (data.IsOk)
            return Ok(new { Data = data });
        return StatusCode(600, data);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"Delete SignatureProcess Error. ");
        return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
    }
}

[HttpPost]
[Route("init-sign-positions")]
[MediAuthorize(new[] { Lib.Share.PermitValue.ChoPhep }, Lib.Share.ActionPermit.Common_SignatureProcess)]
public IActionResult InitSignPositions()
{
    var rt = _signatureProcessService.InitSignPositions();
    return Ok();
}
```

```
[HttpPost]
[Route("init-document-types")]
[MediAuthorize(new[] { Lib.Share.PermitValue.ChoPhep }, Lib.Share.ActionPermit.Common_SignatureProcess)]
public IActionResult InitDocumentType()
{
    var requestUser = (BaseUser)Request.HttpContext.Items["User"];
    if (requestUser == null)
        return Forbid();

    var rt = _signatureProcessService.InitDocumentTypes(requestUser.Id);
    return Ok();
}

[HttpGet]
[Route("getall-document-types")]
public IActionResult GetAllDocumentTypes()
{
    try
    {
        var requestUser = (BaseUser)Request.HttpContext.Items["User"];
        if (requestUser == null)
            return Forbid();

        var data = _signatureProcessService.GetAllDocumentType();
        return Ok(new { Data = data });
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"GetAll Document Type Error. ");
        return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
    }
}

[HttpGet]
[Route("getall-sign-positions")]
public IActionResult GetAllSignPositions()
{
    try
    {
        var requestUser = (BaseUser)Request.HttpContext.Items["User"];
        if (requestUser == null)
            return Forbid();

        var data = _signatureProcessService.GetAllSignPositions();
        return Ok(new { Data = data });
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"GetAll Sign Position Error. ");
        return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
    }
}

[HttpGet]
[Route("search-grid")]
public IActionResult SearchSignPosition()
{
    try
    {
        var requestUser = (BaseUser)Request.HttpContext.Items["User"];
        if (requestUser == null)
            return Forbid();

        var data = _signatureProcessService.GetAllSignPositions(showHide: true);
        return Ok(new { Data = data });
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"Search Sign Position Error. ");
        return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
    }
}

[HttpPost]
[Route("get-signed-document")]
public IActionResult GetSignedDocument(SignedDocumentSearchQuery search)
```

```
{
    try
    {
        var requestUser = (BaseUser)Request.HttpContext.Items["User"];
        if (requestUser == null)
            return Forbid();

        var data = _signatureProcessService.GetSignedDocument(search, requestUser.Id);
        return Ok(new
        {
            TotalSign = data.TotalSign,
            SignaturePositionGroupByOrder = data.SignaturePositionGroupByOrder,
            SignNameCurrent = data.SignNameCurrent,
            SignedNumber = data.SignedNumber,
            SignedStatus = data.SignedStatus,
            SignedStatusName = data.SignedStatusName,
        });
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"Get Signed Document Error. ");
        return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
    }
}

[HttpPost]
[Route("create-signed-document")]
public IActionResult CreateSignedDocument(SignedDocumentCreateViewModel model)
{
    try
    {
        var requestUser = (BaseUser)Request.HttpContext.Items["User"];
        if (requestUser == null)
            return Forbid();

        var data = _signatureProcessService.CreateSignedDocument(model, requestUser.Id);
        if (data.IsOk)
            return Ok(new { Data = data });
        return StatusCode(600, data);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"Create SignatureProcess Error. ");
        return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
    }
}

[HttpPut]
[Route("update-signed-document")]
public IActionResult UpdateSignedDocument(SignedDocumentUpdateViewModel model)
{
    try
    {
        var requestUser = (BaseUser)Request.HttpContext.Items["User"];
        if (requestUser == null)
            return Forbid();

        var data = _signatureProcessService.UpdateSignedDocument(model, requestUser.Id);
        if (data.IsOk)
            return Ok(new { Data = data });

        return StatusCode(600, data);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"Update SignatureProcess Error. ");
        return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
    }
}

[HttpPut]
[Route("delete-signed-document")]
public IActionResult DeleteSignedDocument(SignedDocumentSearchQuery model)
{
    try
    {
        var requestUser = (BaseUser)Request.HttpContext.Items["User"];
```

```
        if (requestUser == null)
            return Forbid();

        var data = _signatureProcessService.DeleteSignedDocument(model, requestUser.Id);
        if (data.IsOk)
            return Ok(new { Data = data });

        return StatusCode(600, data);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"Update SignatureProcess Error. ");
        return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
    }
}

[HttpPost]
[Route("sign-remind")]
public IActionResult SignRemind(SignedDocumentSearchQuery model)
{
    try
    {
        var requestUser = (BaseUser)Request.HttpContext.Items["User"];
        if (requestUser == null)
            return Forbid();

        var data = _signatureProcessService.SignRemind(model, requestUser.Id);
        if (data.IsOk)
            return Ok(new { Data = data });

        return StatusCode(600, data);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"Create SignatureProcess Error. ");
        return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
    }
}

[HttpPost]
[Route("sign-reject")]
public IActionResult SignReject(SignRejectViewModel model)
{
    try
    {
        var requestUser = (BaseUser)Request.HttpContext.Items["User"];
        if (requestUser == null)
            return Forbid();

        var data = _signatureProcessService.SignReject(model, requestUser.Id);
        if (data.IsOk)
            return Ok(new { Data = data });

        return StatusCode(600, data);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"Create SignatureProcess Error. ");
        return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
    }
}

[HttpPost]
[Route("get-sign-report")]
[MediAuthorize(new[] { Lib.Share.PermitValue.Read }, Lib.Share.ActionPermit.Common_SignatureProcess)]
public IActionResult GetSignByUserIdAndFilter(FilterSignedDocumentViewModel filter)
{
    try
    {
        var requestUser = (BaseUser)Request.HttpContext.Items["User"];
        if (requestUser == null) return Forbid();
        var model = new List<SignReportViewModel>();
        model = _signatureProcessService.GetSignByUserId(filter, requestUser.Id);
        return Ok(new { Data = model });
    }
    catch (Exception ex)
    {
```

```
        _logger.LogError(ex, $"DocumentManage error.");
        return StatusCode(600, new ReturnCode
        {
            Code = ErrorCode.HisSystemError,
            Description = $"Lỗi hệ thống. Error={ex.Message}"
        });
    }
}

[HttpPost]
[Route("mobile/search")]
public IActionResult SearchSignDocument(PaginationQueryModel<SearchSignDocumentViewModel> model)
{
    try
    {
        var requestUser = (BaseUser)Request.HttpContext.Items["User"];
        if (requestUser == null) return Forbid();
        var data = _signatureProcessService.SearchSignDocument(model, requestUser.Id);
        return Ok(new { Data = data });
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"SearchSignDocument error.");
        return StatusCode(600, new ReturnCode
        {
            Code = ErrorCode.HisSystemError,
            Description = $"Lỗi hệ thống. Error={ex.Message}"
        });
    }
}

[HttpPost]
[Route("create-sign-position")]
[MediAuthorize(new[] { Lib.Share.PermitValue.ChoPhep }, Lib.Share.ActionPermit.Common_SignatureProcess)]
public IActionResult CreateSignPosition(SignPositionViewModel model)
{
    try
    {
        var requestUser = (BaseUser)Request.HttpContext.Items["User"];
        if (requestUser == null) return Forbid();

        var data = _signatureProcessService.CreateSignPosition(model, requestUser.Id);
        if (data.IsOk)
            return Ok(new { Data = data });
        return StatusCode(600, data);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"CreateSignPosition error.");
        return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
    }
}

[HttpPost]
[Route("update-sign-position")]
[MediAuthorize(new[] { Lib.Share.PermitValue.ChoPhep }, Lib.Share.ActionPermit.Common_SignatureProcess)]
public IActionResult UpdateSignPosition(SignPositionViewModel model)
{
    try
    {
        var requestUser = (BaseUser)Request.HttpContext.Items["User"];
        if (requestUser == null) return Forbid();

        var data = _signatureProcessService.UpdateSignPosition(model, requestUser.Id);
        if (data.IsOk)
            return Ok(new { Data = data });
        return StatusCode(600, data);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"UpdateSignPosition error.");
        return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
    }
}

[HttpDelete]
[Route("delete-sign-position")]
```

```
        [MediAuthorize(new[] { Lib.Share.PermitValue.ChoPhep }, Lib.Share.ActionPermit.Common_SignatureProcess)]
        public IActionResult DeleteSignPosition(int id)
        {
            try
            {
                var requestUser = (BaseUser)Request.HttpContext.Items["User"];
                if (requestUser == null) return Forbid();

                var data = _signatureProcessService.DeleteSignPosition(id);
                if (data.IsOk)
                    return Ok(new { Data = data });
                return StatusCode(600, data);
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, $"DeleteSignPosition error.");
                return StatusCode(600, $"Lỗi hệ thống. Error={ex.Message}");
            }
        }
    }
}
```