//First step

//import RSKImageCropper

//And variable in your controller

//let imagePickerController = UIImagePickerController()

//Then add extension to your view controller

extension YourViewController: UIImagePickerControllerDelegate, UINavigationControllerDelegate,RSKImageCropViewControllerDelegate,RSKImageCropViewControllerDataSource{

//delegate function of cancel for RSKimage Cropper

func imageCropViewControllerDidCancelCrop(\_ controller: RSKImageCropViewController) {

self.navigationController?.popViewController(animated: true)

}

//functions for making cropper rectangular for custom cropper you can change value as you want

internal func imageCropViewControllerCustomMaskRect(\_ controller: RSKImageCropViewController) -> CGRect {

let aspectRatio = CGSize(width: 16.0, height: 9.0)

let viewWidth = controller.view.frame.width

let viewHeight = controller.view.frame.height

var maskWidth: CGFloat

if controller.isPortraitInterfaceOrientation() {

maskWidth = viewWidth

} else {

maskWidth = viewHeight

}

var maskHeight: CGFloat

repeat {

maskHeight = maskWidth \* aspectRatio.height / aspectRatio.width

maskWidth -= 1.0

} while maskHeight != floor(maskHeight)

maskWidth += 1.0

let maskSize = CGSize(width: maskWidth, height: maskHeight)

let maskRect = CGRect(x: (viewWidth - maskSize.width) \* 0.5, y: (viewHeight - maskSize.height) \* 0.5, width: maskSize.width, height: maskSize.height)

return maskRect

}

internal func imageCropViewControllerCustomMaskPath(\_ controller: RSKImageCropViewController) -> UIBezierPath {

let rect = controller.maskRect

let point1 = CGPoint(x: rect.minX, y: rect.maxY)

let point2 = CGPoint(x: rect.maxX, y: rect.maxY)

let point3 = CGPoint(x: rect.maxX, y: rect.minY)

let point4 = CGPoint(x: rect.minX, y: rect.minY)

let rectangle = UIBezierPath()

rectangle.move(to: point1)

rectangle.addLine(to: point2)

rectangle.addLine(to: point3)

rectangle.addLine(to: point4)

rectangle.close()

return rectangle

}

func imageCropViewControllerCustomMovementRect(\_ controller: RSKImageCropViewController) -> CGRect {

if controller.rotationAngle == 0 {

return controller.maskRect

} else {

let maskRect = controller.maskRect

let rotationAngle = controller.rotationAngle

var movementRect = CGRect.zero

movementRect.size.width = CGFloat(maskRect.width \* fabs(cos(rotationAngle)) + maskRect.height \* fabs(sin(rotationAngle)))

movementRect.size.height = CGFloat(maskRect.height \* fabs(cos(rotationAngle)) + maskRect.width \* fabs(sin(rotationAngle)))

movementRect.origin.x = maskRect.minX + (maskRect.width - movementRect.width) \* 0.5

movementRect.origin.y = maskRect.minY + (maskRect.height - movementRect.height) \* 0.5

movementRect.origin.x = floor(movementRect.minX)

movementRect.origin.y = floor(movementRect.minY)

movementRect = movementRect.integral

return movementRect

}

}

//delegate function of RskimageCropper for cropping image

func imageCropViewController(\_ controller: RSKImageCropViewController, didCropImage croppedImage: UIImage, usingCropRect cropRect: CGRect, rotationAngle: CGFloat) {

//isImageViewChange is variable as for check between circular cropped image usage or rectangular cropped image usage.

//outletImageProfile is outlet of profile image which needs to be circular

//backgroundImageForProfileEdit is outlet for cover which needs to be rectangular cropped

if isImageViewChange{

self.outletImageProfile.image = croppedImage

}

else

{

self.backgroundImageForProfileEdit.image = croppedImage

}

self.navigationController?.popViewController(animated: true)

}

//delegate function of image picker view

func imagePickerController(\_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey : Any]) {

var selectedImage: UIImage?

if isImageViewChange == false

{

if let editedImage = info[.editedImage] as? UIImage {

selectedImage = editedImage

backgroundImageForProfileEdit.image = selectedImage!

imagePickerController.dismiss(animated: true, completion: {

var imageCropVC = RSKImageCropViewController()

// you can change cropMode to circle, square, custom

//—————————————————————————————————————————

//this is the main part where your photo is cropped

imageCropVC = RSKImageCropViewController(image: self.backgroundImageForProfileEdit.image!, cropMode: RSKImageCropMode.custom)

imageCropVC.delegate = self

imageCropVC.dataSource = self

self.navigationController?.pushViewController(imageCropVC, animated: true)

//—————————————————————————————————————————

})

} else if let originalImage = info[.originalImage] as? UIImage {

selectedImage = originalImage

backgroundImageForProfileEdit.image = selectedImage!

imagePickerController.dismiss(animated: true, completion:

var imageCropVC : RSKImageCropViewController!

imageCropVC = RSKImageCropViewController(image: self.backgroundImageForProfileEdit.image!, cropMode: RSKImageCropMode.custom)

imageCropVC.dataSource = self

imageCropVC.delegate = self

self.navigationController?.pushViewController(imageCropVC, animated: true)

})

}

}

else{

if let editedImage = info[.editedImage] as? UIImage {

selectedImage = editedImage

outletImageProfile.image = selectedImage!

imagePickerController.dismiss(animated: true, completion: {

var imageCropVC = RSKImageCropViewController()

imageCropVC = RSKImageCropViewController(image: self.outletImageProfile.image!, cropMode: RSKImageCropMode.circle)

imageCropVC.delegate = self

self.navigationController?.pushViewController(imageCropVC, animated: true)

})

} else if let originalImage = info[.originalImage] as? UIImage {

selectedImage = originalImage

outletImageProfile.image = selectedImage!

imagePickerController.dismiss(animated: true, completion: {

var imageCropVC = RSKImageCropViewController()

imageCropVC = RSKImageCropViewController(image: self.outletImageProfile.image!, cropMode: RSKImageCropMode.circle)

imageCropVC.delegate = self

self.navigationController?.pushViewController(imageCropVC, animated: true)

})

}

}

}

//delegate function for cancel button in image picker view

func imagePickerControllerDidCancel(\_ picker: UIImagePickerController) {

dismiss(animated: true, completion: nil)

}

}

v