Міністерство освіти і науки України Національний університет “Львівська політехніка” Кафедра “Системи автоматизованого проектування”

## ЗВІТ

про проходження проектно-технологічної практики студентки ІV курсу групи КН-44

Ситник Мар’яни

**База практики** ФОП “Ситник Р. Н.”, м. Тернопіль

**Термін практики** з «13» серпня 2018 р. по з «01» вересня 2018 р.

**Керівники практики**

від кафедри к.т.н., доцент каф САП Фармага І.В.

(підпис) (наук.ст, вч.звання, прізвище, ім’я, по батькові)

від бази практики ФОП Ситник Р. Н.

(підпис) (посада, прізвище, ім’я, по батькові)

**ОЦІНКА**

**Дата**

Тема

БКР

Керівник БКР д.т.н., с.н.с., доц. каф. САПР Щербовських С. В.

(підпис) (Посада, Прізвище, ініціали)

Львів – 2018

|  |  |
| --- | --- |
| **Зміст** |  |
| **Вступ** | 3 |
| 1. Ознайомлення з базою практики | 5 |
| 1.1 Про базу практики | 5 |
| 1.2 Структура бази практики | 6 |
| 1.3 Послуги: | 7 |
| 1.4 Партнери: | 8 |
| 1.5 Опис бази практики | 8 |
| 2. Теоретична частина | **Error! Bookmark not defined.** |
| 2.1 Предметна область | 9 |
| 2.2 Мова програмування | 10 |
| 2.3 Середовище розробки | 11 |
| 2.4 Аналоги | 11 |
| 2.4 Опис виконаних робіт | 12 |
| 2.6 Знімки екрану програми | 14 |
| 2.7 Висновки про отримані під час практики результати | 16 |
| 3. Список опрацьованих джерел | 17 |
| Додаток А. Фрагмент тексту програми | 18 |

**Вступ**

У кожному вузі України передбачено проходження студентами навчально-технологічної практики, яка є обов’язковим складовим елементом підготовки будь-якого спеціаліста. Під час проходження практики студент закріплює здобуті знання і навички, отримуючи практичний досвід роботи в реальних економічних умовах.

Студент може мати досконалі теоретичні знання, але може розгубитися в ситуації, коли виникне необхідність прийняти рішення в реальних економічних умовах. Тому необхідно забезпечувати студентам змогу пройти навчально-технологічну практику ще під час навчання у вузі, для того, щоб спробувати себе не тільки в ролі теоретика, але й практика.

Переддипломна практика ставить за мету поглиблене ознайомлення зі специфікою майбутньої спеціальності, розширення й доповнення знань, здобутих під час вивчення курсів та виконання практичних і лабораторних завдань з таких дисциплін: “Основи інформаційних технологій”, “Алгоритмізація та програмування”, “Проблемно-орієнтоване програмування”, “Системне програмування”, “Теорія алгоритмів ”, “Схемотехніка та архітектура комп’ютерів”, “Об’єктно-орієнтоване програмування”, “Організація баз даних та знань”, “Операційні системи”, “Системний аналіз”, “Чисельні методи”, “Технології створення програмних продуктів”, “Теоретичні основи САПР”, “Технології комп’ютерного проектування”, “Комп’ютерні мережі”, “Моделювання систем”, “Інтелектуальний аналіз даних”, “Методи синтезу та оптимізації”. А також формування первинних професійних умінь і навичок для прийняття самостійних рішень і розв’язування задач пов’язаних із сучасними комп’ютерними інформаційними технологіями в реальних виробничих умовах.

Під час проходження практики необхідно ознайомитись з нормативно- технологічною документацією підприємства – бази практики, вивчення особливостей адміністративної та інформаційної взаємодії між його окремими підрозділами; ознайомитись з конфігурацією та архітектурою технічних засобів підприємства; ознайомитись з програмним забезпеченням і програмними продуктами, які використовуються або створюються в процесі діяльності даного підприємства; оволодіти сучасними технологіями оброблення інформації та застосування їх для виконання практичних завдань;набути виробничих навичок, а саме: технічної, проектувальної, виконавської;набути навичок роботи в колективі спеціалістів та інше.

1. Ознайомлення з базою практики

# Про базу практики

Високі технології створюють нові товари та послуги — якісні, економні та безпечні для навколишнього середовища.

Ціни на бензин ніколи не знижуються, а постійно ростуть вгору. Тому вже сьогодні автовласники задумуються над тим, як можна **зекономити кошти**, покращити роботу свого автомобіля, і водночас подбати про екологію довкілля. Нині нові **системи газобалонного обладнання** зняли питання про компроміс між економією і збереженням потужності двигуна. Основна перевага ГБО в тому, що газова система дозволяє експлуатувати автомобіль як на газі, так і на бензині. Газове обладнання перестало бути альтернативою бензину, і сьогодні, даючи двократну економію, є безперечним лідером. **Переобладнання автомобіля на систему ГБО** окупиться надзвичайно швидко. При цьому одразу відчувається економічний ефект при експлуатації свого авто, а заощаджені кошти можна витратити на власне задоволення. Більш того, якщо є бажання придбати новий автомобіль, **система ГБО** може бути демонтована та без проблем встановлена на інше авто і продовжить служити ще протягом тривалого часу. Працівники “ГБО на Авто” встановлять газобалонне обладнання в авто, і замість постійних витрат на бензин, знову щоразу можна заправляти повний балон на газових заправках. При цьому деталі двигуна будуть мати набагато більший термін експлуатації, що допоможе зекономити кошти на планових ремонтах. Автомобіль буде екологічно безпечним.

На ринку встановлення та сервісу ГБО підприємство уже понад десять років. Тому сьогодні можна впевнено говорити про надійність та гарантію на обладнання і послуги в “ГБО на Авто”.

* 1. Структура бази практики

Компанія сьогодні є одним з лідерів на ринку встановлення газобалонного обладнання на автомобільний транспорт. І однією з небагатьох фірм, яка має Сертифікат відповідності на встановлення ГБО іноземного виробництва. Професіоналізм і кваліфікація нашого персоналу, укомплектованість обладнання підтвердженні сертифікатами гарантії та якості.

Фірма займається установкою і налаштуванням газобалонного обладнання (ГБО). Фірма використовує найсучасніші системи ГБО визнаних світових виробників: LOVATO, Landi Renzo, BRC, Bigas, Tomasetto, Autogas Italia, Valtec, Rail, AEB (Італія), Atiker (Туреччина ), AC (STAG), Milmet, Lecho, Rybacki, Certools, GMS Europe B.V. (Flash Lube), ELPIGAS SP (Польща). Встановлює газобалонне обладнання на автомобілі будь-якої складності, як на інжекторні, так і на карбюраторні двигуни, на автомобілі як вітчизняного, так і іноземного виробництва.

На ринку сервісу автомобілів та ГБО фірма працює з 1996 року. Спочатку ми надавали послуги з ремонту автомобілів та налагодження ГБО, а у 2006 році був створений сервісний центр з переобладнання автотранспортних засобів для роботи на газі (метан, пропан). Досвід набувається з часом. Тому сьогодні фірма впевнено може говорити про надійність та гарантію роботи встановленого газобалонного обладнання протягом багатьох років експлуатації. Фірма надає гарантію на встановлення ГБО - 1 рік або 20 тис. км пробігу.

Встановлення та регулювання ГБО здійснюють кваліфіковані спеціалісти. Надається увесь необхідний комплект документів для зміни техпаспорта в МРЕВ. Консультації протягом усієї процедури реєстрації та подальшої експлуатації обладнання. Окрім того, здійснюється технічну підтримку при поломці автомобіля за викликом клієнта.

За понад десять років бездоганної роботи клієнтами фірми стали як великі підприємства, так і понад дві тисячі власників приватних авто.

# Послуги:

* Сервіс ГБО

Як і будь-яке обладнання, ГБО також має властивість виходити з ладу. Найчастіше, причинами поломки ГБО є старіння гумовотехнічних виробів, низька якість газового палива чи самого обладнання.

* Монтаж ГБО

Використовуючи багаторічний досвід роботи, ми впораємося з встановленням газового устаткування як на нескладні автомобілі вітчизняного виробництва, так і на більш складні - імпортного виробництва, в тому числі — позашляховики.

* Продаж ГБО

Продаж, діагностика, монтаж, демонтаж, сервіс (ремонт), настройка газового обладнання, планове обслуговування, повна установка ГБО на авто, гарантія на всі види робіт.

Тож, для спрощення роботи був розроблений веб-сайт з каталогами наших партнерів.

# Партнери:

* Milano
* Сервіс Газ
* Авто Газ
* Atiker
* BRC Italy
* Motor Gas
* Коста Газ
* Zavoli
* Auto Trend

# Опис бази практики

Головним завданням практики є одержання певного досвіду в розробці програмних засобів для вирішення певних проблем, а також використання методів та прийомів методологій управління проектами при гнучкій розробці.

В фірмі “ГБО на Авто” 12 працівників, матеріально-технічна база має досить високий рівень. Дружній колектив та домашня обстановка сприяє комфортній праці. Усі кабінети оснащено згідно стандартів, кожен працівник має особисте місце з необхідними засобами праці.

Серед працівників була проблема затрати часу, оскільки бокси для установки обладнання та ремонту авто знаходились на відстані від складу з запчастинами, тож я запропонувала вирішення проблеми шляхом розробки програми, яка б зчитувала номери машин, працівник лише б вибирав марку та модель, а також, за бажанням, додавав свій коментар та відправляв ці дані іншому працівнику в офіс. Ідея була схвалена та затверджена власником фірми.

## Теоретична частина

* 1. **Предметна область**

**Оптичне розпізнавання тексту** (англ. *optical character recognition, OCR*) — це механічне або електронне переведення зображень рукописного, машинописного або друкованого тексту в послідовність кодів, що використовуються для представлення в текстовому редакторі. Розпізнавання широко використовується для конвертації книг і документів в електронний вигляд, для автоматизації систем обліку в бізнесі або для публікації тексту на веб-сторінці. Оптичне розпізнавання тексту дозволяє редагувати текст, здійснювати пошук слова або фрази, зберігати його в компактнішій формі, демонструвати або роздруковувати матеріал, не втрачаючи якості, аналізувати інформацію, а також застосовувати до тексту електронний переклад, форматування або перетворення в мовлення. Оптичне розпізнавання тексту є досліджуваною проблемою в галузях розпізнавання образів, штучного інтелекту і комп'ютерного зору.

Системи оптичного розпізнавання тексту вимагають калібрування для роботи з конкретним шрифтом; у ранніх версіях, для програмування було необхідно зображення кожного символу, програма одночасно могла працювати тільки з одним шрифтом. Зараз найпоширеніші так звані

«інтелектуальні» системи, що розпізнають більшість шрифтів із високим ступенем точності. Деякі системи оптичного розпізнавання тексту здатні відновлювати вихідне форматування тексту, включаючи зображення, колонки й інші нетекстові компоненти.

**Tesseract** - вільна платформа для оптичного розпізнавання тексту, вихідні коди якої Google подарував спільноті в 2006 році. Якщо ви пишете софт для розпізнавання тексту, то вам, напевно, доводилося звертатися до послуг цієї потужної бібліотеки. І якщо вона не впоралася з вашим

текстом, то вихід у вас залишається один - навчити її. Процес цей досить складний і повнен просто-таки магічних дій. Мені знадобилось кілька днів на осягнення всієї його глибини.

## Мова програмування

**Swift** — багатопарадигмова компільована мова програмування, розроблена компанією Apple для того, щоб співіснувати з Objective C і бути стійкішою до помилкового коду. Мова побудована з LLVM компілятором, включеного у Xcode 6 beta.

Мова також пропонує безліч сучасних методів програмування, таких як замикання, узагальнене програмування, лямбда-вирази, кортежі і словникові типи, швидкі операції над колекціями, елементи функційного програмування. Основним застосуванням Swift є розробка користувацьких застосунків для macOS, iOS, tvOS, watchOS з використанням тулкіта Cocoa і Cocoa Touch. При цьому Swift надає об'єктну модель, сумісну з Objective-

1. Код мовою Swift може змішуватися з кодом на С і Objective-C в одному проекті.

Swift щільно інтегровано до власницького середовища розробки Xcode, проте може бути викликано з терміналу, що уможливлює її використання на операційний системах, відмінних від macOS, наприклад, на Linux.

Окремо варто відзначити, що Swift від компанії Apple не варто плутати з досить давно розроблюваною скриптовою мовою Swift, націленої на багатонитеве програмування і поставленого під вільною ліцензією Apache.

# Середовище розробки

**Xcode** — інтегроване середовище розробки(IDE) виробництва Apple. Дозволяє створювати програмне забезпечення з використанням таких технологій як GCC, GDB, Java та ін. На сьогодні є єдиним засобом написання «універсальних»(Universal Binary) прикладних програм для Mac OS X.

Xcode включає в себе більшу частину документації розробника від Apple та Interface Builder — застосунок, який використовується для створення графічних інтерфейсів.

Пакет Xcode містить змінену версію вільного набору компіляторів GNU Compiler Collection і підтримує мови C, C++, Objective-C, Swift, Java, AppleScript, Python і Ruby з різними моделями програмування, включаючи (але не обмежуючись) Cocoa, Carbon і Java. Сторонніми розробниками реалізована підтримка GNU Pascal, Free Pascal, Ada, C #, Perl, Haskell і D. Пакет Xcode використовує GDB як back-end для свого відналагоджувача.

# Аналоги

Оскільки тема дуже популярна в наш час, я вирішила пошукати аналоги даній програмі. На просторах інтернету я знайшла багато різних програм, пов’язаних з OCR та Тesseract. Ось кілька з них. На рис.1. представленна програма зчитування друкованого тексту, написаного як верхнім, так і нижнім регістром. На рис. 2. представлена програма, яка вміє розпізнавати текст з картинки. Ну і на рис. 3. показано як працює програма, яку можна знайти у будь-якому iPhone для зчитування серійного номеру продукту Apple.

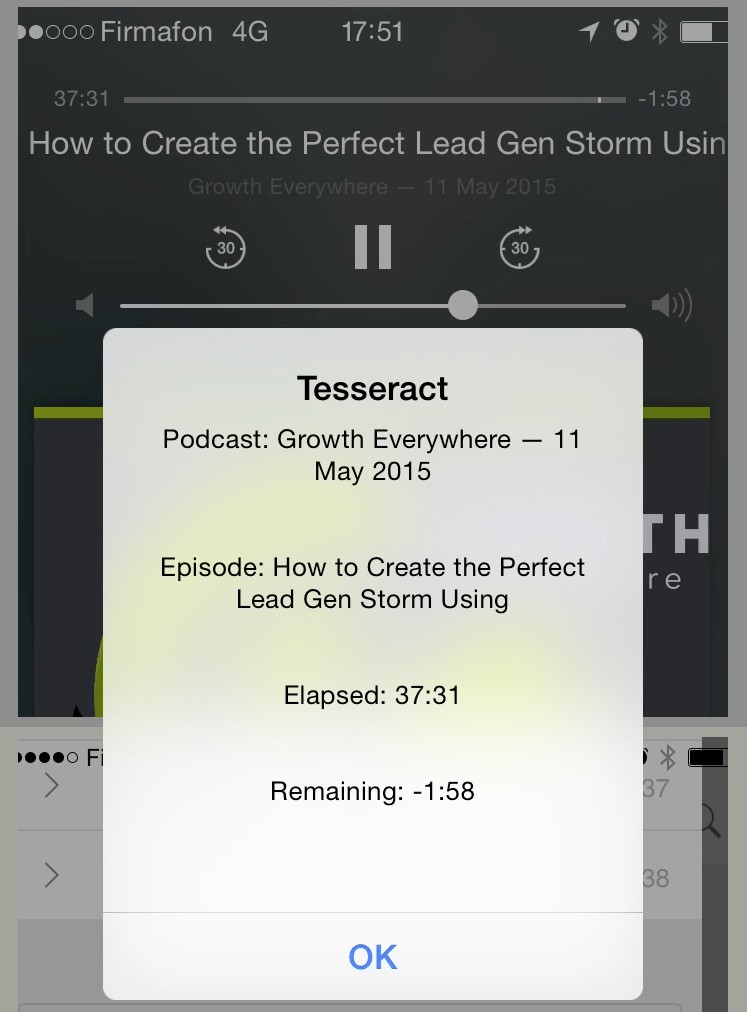


Рис. 1-2. Знімки екранів програм-аналогів

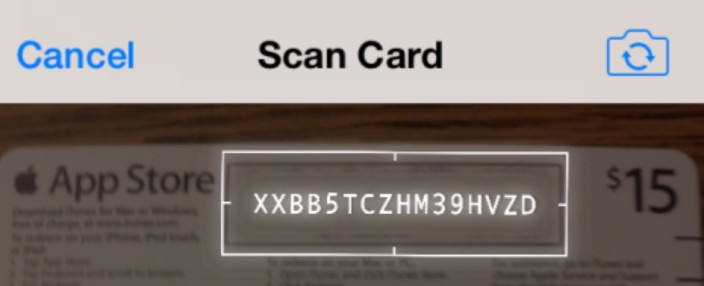


Рис. 3. Знімок екрану програми-аналога

# 2.4 Опис виконаних робіт

Під час проходження навчальної технологічної практики я ознайомився із можливостями бібліотек SwiftOCR та AVFoundation. Ці бібліотеки дуже широко поширені і мають велику базу підтримки по всьому світу. Почала розробляти вирішення поставленої задачі.

Структура проекту представлена на рис. 4., а верстка екранів для додатку зображена на рис. 5.

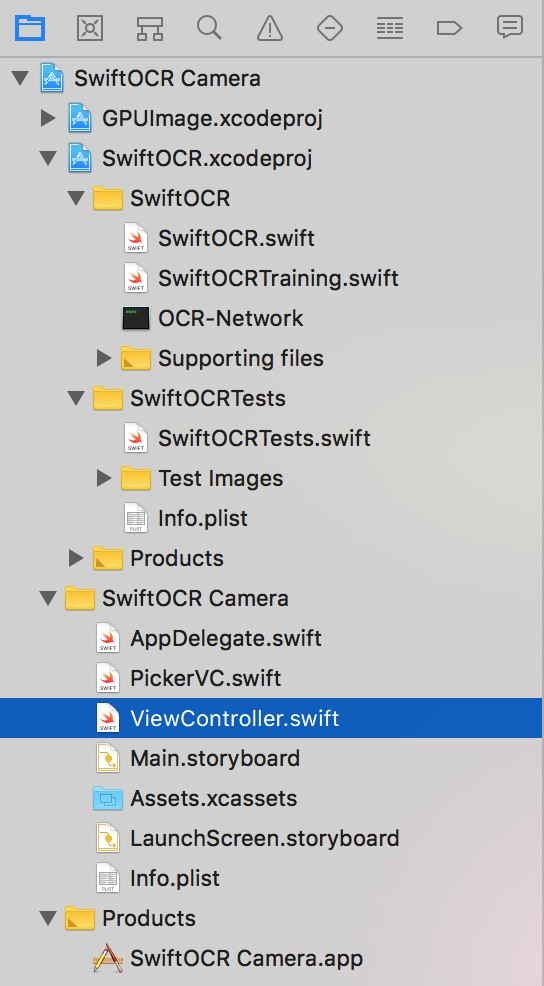


Рис. 4. Структура проекту

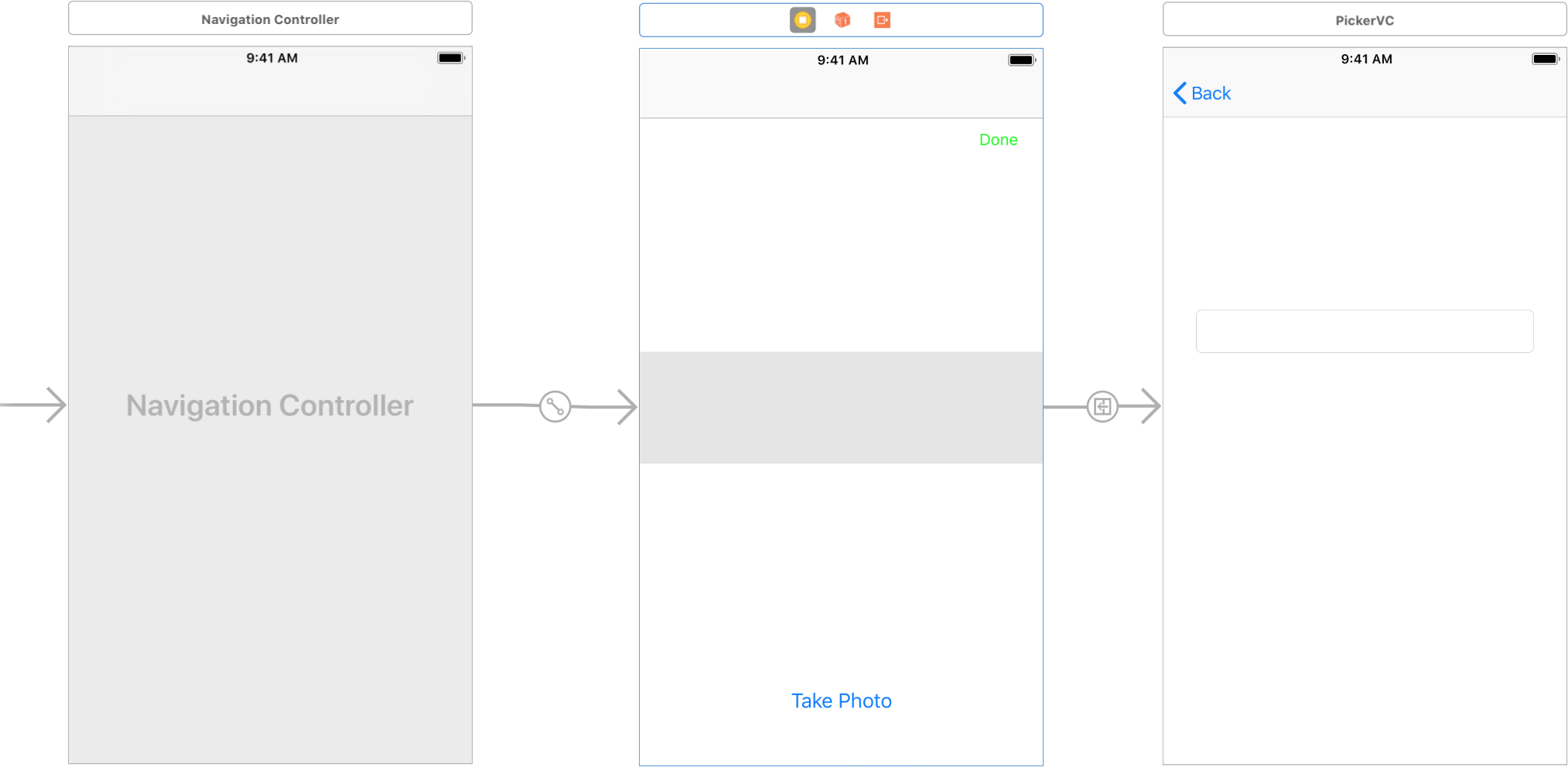


Рис. 5. Зверстані View Controllers

# Знімки екрану програми

Після опрацювання великої кількості матеріалу, я почала розробку додатку під систему iOS. На рис. 6-7. наведені знімки екранів телефону з додатком для розпізнавання номерних знаків.

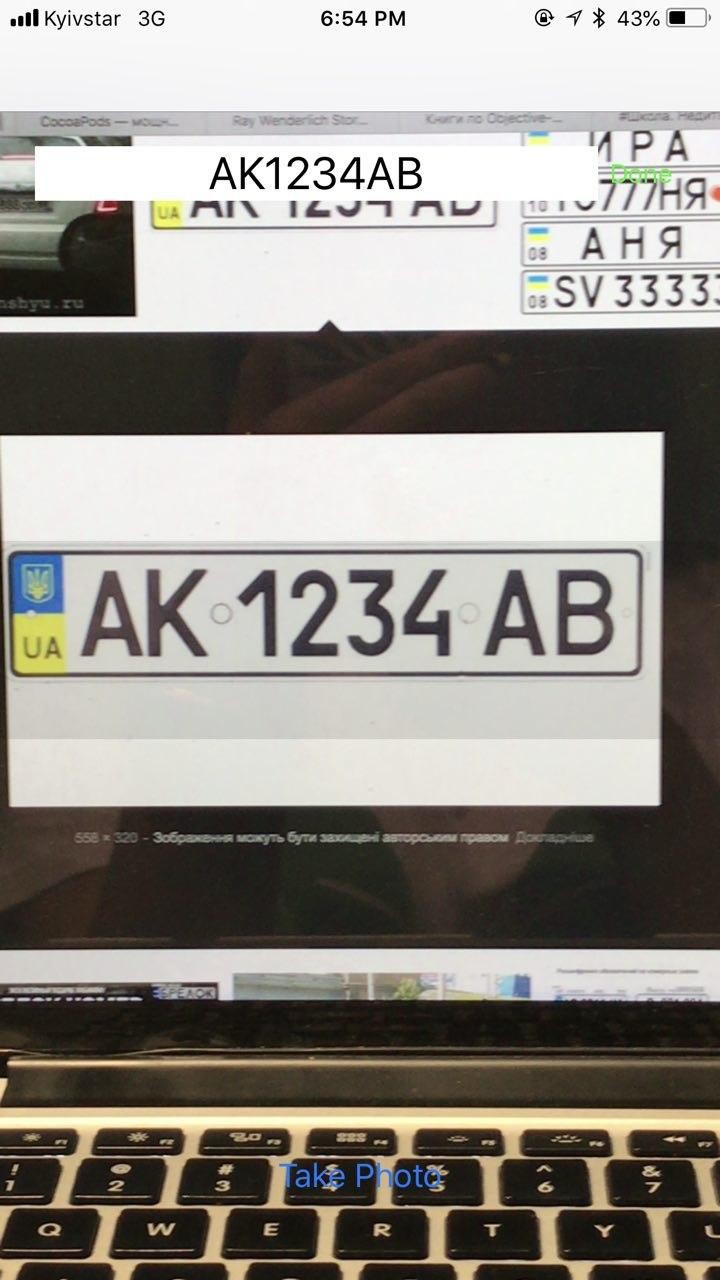
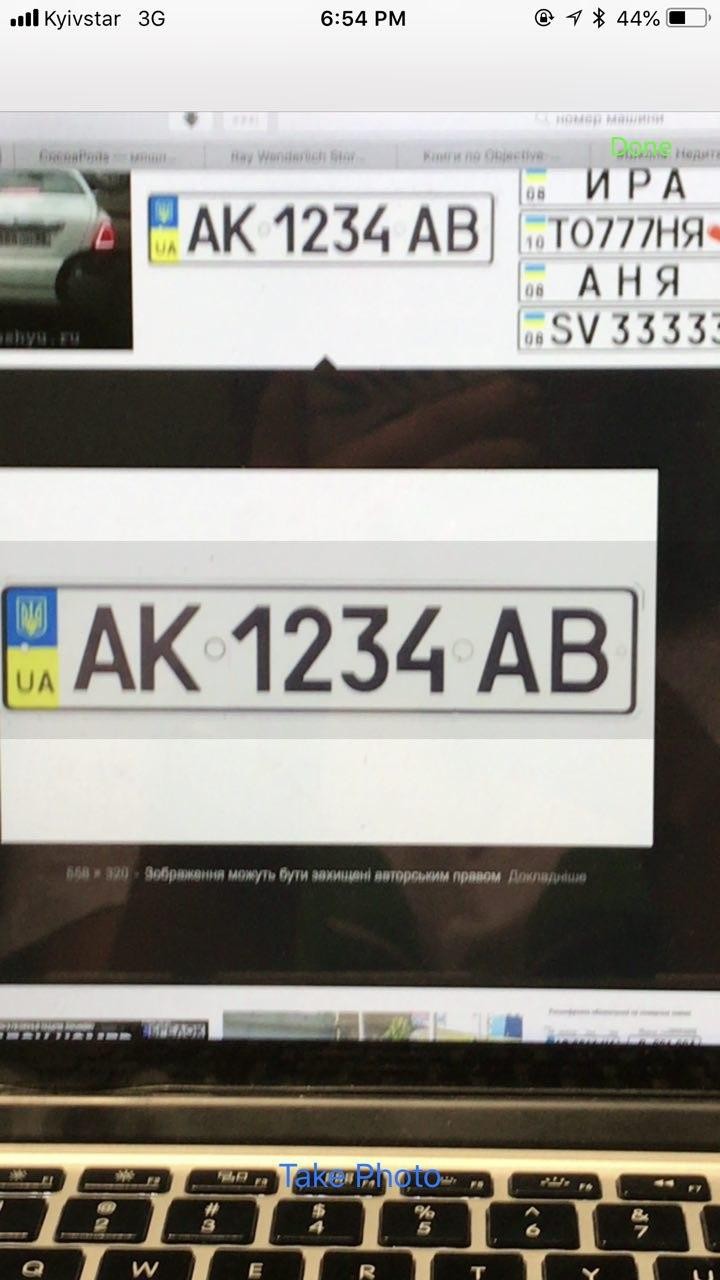


Рис. 6-7. Знімки екранів готової аплікації

На рис. 8. зображено скрін, на якому планується ще додати кілька елементів для заповнення даних про машину.



Рис. 8. Знімок екрану, який потребує допрацювання

# Висновки про отримані під час практики результати

Під час проходження навчальної технологічної практики я ознайомилась із новими можливостями мови Swift та нових Pods SwiftOCR. Посилила свої знання Swift, UIKit.

Для виконання поставленого завдання я склала план робіт та дотримувалась його. Виконуючи роботу я намагалась максимально ефективно використовувати свій час та виконувати її якнайкраще.

# Список опрацьованих джерел

* 1. https://github.com/gali8/Tesseract-OCR-iOS
  2. https://github.com/gali8/Tesseract-OCR-iOS/wiki/Using-Tesseract-OCR- iOS
  3. https://[www.raywenderlich.com/306-tesseract-ocr-tutorial-for-ios](http://www.raywenderlich.com/306-tesseract-ocr-tutorial-for-ios)
  4. https://swift.org/documentation
  5. https://developer.apple.com/videos/play/wwdc2016/503/
  6. https://medium.com/flawless-app-stories/vision-in-ios-text-detection-and- tesseract-recognition-26bbcd735d8f
  7. https://uk.wikipedia.org/wiki/Tesseract

# Додаток А. Фрагмент тексту програми

**import UIKit import SwiftOCR**

**import AVFoundation**

**extension UIImage {**

**func detectOrientationDegree () -> CGFloat { switch imageOrientation {**

**case .right, .rightMirrored: return 90 case .left, .leftMirrored: return -90 case .up, .upMirrored: return 180 case .down, .downMirrored: return 0**

**}**

**}**

**}**

**class ViewController: UIViewController {**

**// MARK: - Outlets**

**@IBOutlet weak var cameraView: UIView! @IBOutlet weak var viewFinder: UIView! @IBOutlet weak var label: UILabel! @IBOutlet weak var doneButton: UIButton!**

**// MARK: - Private Properties**

**fileprivate var stillImageOutput: AVCaptureStillImageOutput! fileprivate let captureSession = AVCaptureSession()**

**fileprivate let device = AVCaptureDevice.default(for: AVMediaType.video) private let ocrInstance = SwiftOCR()**

**// MARK: - View LifeCycle override func viewDidLoad() {**

**super.viewDidLoad()**

**// start camera init**

**DispatchQueue.global(qos: .userInitiated).async { if self.device != nil {**

**self.configureCameraForUse()**

**}**

**}**

**}**

**// MARK: - IBActions**

**@IBAction func doneTouchUpInside(\_ sender: Any) {**

**let storyboard = UIStoryboard(name: "Main", bundle: nil)**

**let pickerVC = storyboard.instantiateViewController(withIdentifier: "PickerVC") as! PickerVC self.navigationController?.pushViewController(pickerVC, animated: true)**

**}**

**@IBAction func takePhotoButtonPressed (\_ sender: UIButton) { DispatchQueue.global(qos: .userInitiated).async {**

**guard let capturedType = self.stillImageOutput.connection(with: AVMediaType.video) else { return**

**}**

**self.stillImageOutput.captureStillImageAsynchronously(from: capturedType) { [weak self] optionalBuffer, error -> Void**

**in**

**guard let buffer = optionalBuffer else { return**

**}**

**let imageData = AVCaptureStillImageOutput.jpegStillImageNSDataRepresentation(buffer) let image = UIImage(data: imageData!)**

**let croppedImage = self?.prepareImageForCrop(using: image!) self?.ocrInstance.recognize(croppedImage!) { [weak self] recognizedString in**

**DispatchQueue.main.async { self?.label.text = recognizedString**

**print(self?.ocrInstance.currentOCRRecognizedBlobs ?? "Recoginzed Blob is empty")**

**}**

**}**

**}**

**}**

**}**

**}**

**extension ViewController {**

**// MARK: AVFoundation**

**fileprivate func configureCameraForUse () { self.stillImageOutput = AVCaptureStillImageOutput()**

**let fullResolution = UIDevice.current.userInterfaceIdiom == .phone && max(UIScreen.main.bounds.size.width,**

**UIScreen.main.bounds.size.height) < 568.0 if fullResolution {**

**self.captureSession.sessionPreset = AVCaptureSession.Preset.photo**

**} else {**

**self.captureSession.sessionPreset = AVCaptureSession.Preset.hd1280x720**

**}**

**self.captureSession.addOutput(self.stillImageOutput) DispatchQueue.global(qos: .userInitiated).async {**

**self.prepareCaptureSession()**

**}**

**}**

**private func prepareCaptureSession () { do {**

**self.captureSession.addInput(try AVCaptureDeviceInput(device: self.device!))**

**} catch {**

**print("AVCaptureDeviceInput Error")**

**}**

**// layer customization**

**let previewLayer = AVCaptureVideoPreviewLayer(session: self.captureSession) previewLayer.frame.size = self.cameraView.frame.size previewLayer.frame.origin = CGPoint.zero**

**previewLayer.videoGravity = AVLayerVideoGravity.resizeAspectFill**

**// device lock is important to grab data correctly from image do {**

**try self.device?.lockForConfiguration() self.device?.focusPointOfInterest = CGPoint(x: 0.5, y: 0.5) self.device?.focusMode = .continuousAutoFocus self.device?.unlockForConfiguration()**

**} catch {**

**print("captureDevice?.lockForConfiguration() denied")**

**}**

**//Set initial Zoom scale do {**

**try self.device?.lockForConfiguration() let zoomScale: CGFloat = 2.5**

**if zoomScale <= (device?.activeFormat.videoMaxZoomFactor)! { device?.videoZoomFactor = zoomScale**

**}**

**device?.unlockForConfiguration()**

**} catch {**

**print("captureDevice?.lockForConfiguration() denied")**

**}**

**DispatchQueue.main.async(execute: { self.cameraView.layer.addSublayer(previewLayer) self.captureSession.startRunning()**

**})**

**}**

**// MARK: Image Processing**

**fileprivate func prepareImageForCrop (using image: UIImage) -> UIImage { let degreesToRadians: (CGFloat) -> CGFloat = {**

**return $0 / 180.0 \* CGFloat(Double.pi)**

**}**

**let imageOrientation = image.imageOrientation let degree = image.detectOrientationDegree() let cropSize = CGSize(width: 400, height: 110)**

**//Downscale**

**let cgImage = image.cgImage! let width = cropSize.width**

**let height = image.size.height / image.size.width \* cropSize.width let bitsPerComponent = cgImage.bitsPerComponent**

**let bytesPerRow = cgImage.bytesPerRow let colorSpace = cgImage.colorSpace**

**let bitmapInfo = cgImage.bitmapInfo let context = CGContext(data: nil,**

**width: Int(width), height: Int(height),**

**bitsPerComponent: bitsPerComponent, bytesPerRow: bytesPerRow,**

**space: colorSpace!,**

**bitmapInfo: bitmapInfo.rawValue) context!.interpolationQuality = CGInterpolationQuality.none**

**// Rotate the image context**

**context?.rotate(by: degreesToRadians(degree));**

**// Now, draw the rotated/scaled image into the context context?.scaleBy(x: -1.0, y: -1.0)**

**//Crop**

**switch imageOrientation { case .right, .rightMirrored:**

**context?.draw(cgImage, in: CGRect(x: -height, y: 0, width: height, height: width)) case .left, .leftMirrored:**

**context?.draw(cgImage, in: CGRect(x: 0, y: -width, width: height, height: width)) case .up, .upMirrored:**

**context?.draw(cgImage, in: CGRect(x: 0, y: 0, width: width, height: height)) case .down, .downMirrored:**

**context?.draw(cgImage, in: CGRect(x: -width, y: -height, width: width, height: height))**

**}**

**let calculatedFrame = CGRect(x: 0, y: CGFloat((height - cropSize.height)/2.0), width: cropSize.width, height: cropSize.height) let scaledCGImage = context?.makeImage()?.cropping(to: calculatedFrame)**

**return UIImage(cgImage: scaledCGImage!)**

**}**

**}**