Bachelor's thesis

Information Technology

Internet Technology

2016

Mikias Berhanu Gebre

DEVELOPING AN IOS APPLICATION FOR VALUE STREAM MAPPING WITH SWIFT

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Internet Technology

2016| 24

Instructor: Tiina Ferm

Mikias Berhanu Gebre

DEVELOPING AN IOS APPLICATION FOR VALUE STREAM MAPPING WITH SWIFT

Value Stream Mapping (VSM) is a tool that helps to visualize and understand streamline work processes using lean manufacturing‘s techniques and tools. Ovikone OY commissioned this thesis whose purpose was to create an iOS tablet application that implements the VSM process. In this thesis an iOS tablet application that implements the VSM process is built for Ovikone Oy.

The theoretical part of the thesis handles the different aspects of the Apple operating system (iOS). Scrum was used as a project management methodology, Trello was used as the Scrum tool and the different features of the app were implemented during each sprint. This thesis follows the development process from the beginning of the application development through the end.

This thesis does not only focus on these specific application features but also broadens and explores different aspects of the Swift programming language which can be used for other applications.

KEYWORDS:

iOS, Swift, VSM (Value Stream Mapping), Apple, Scrum.

CONTENTS

APPENDICES

PICTURES

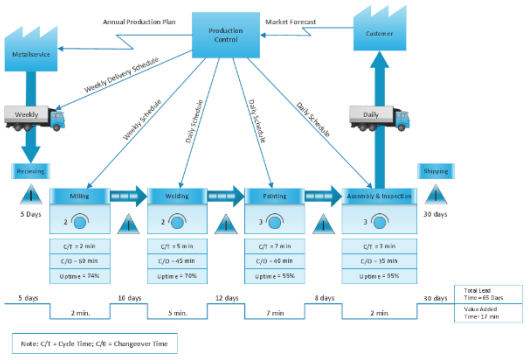
LIST OF ABBREVIATIONS (OR) SYMBOLS

iOS	IPhone Operating System
VSM	Value Stream Mapping
TPS	Toyota Production System
MAC	Macintosh Computer
APP	Application software for mobile devices
Apple	Apple Inc. is a multinational corporation that designs, manufactures, develops, and sells consumer electronics.
LLVM	Low Level Virtual Machine is a compiler infrastructure designed to be a set of reusable library with well-defined libraries.
IDE	Integrated Development Environment is a software application for computer programmers that facilitate software development.
SDK	Software Development Kit
UI	User Interface

1 INTRODUCTION

The aim of this thesis is to develop an iOS tablet application that implements a VSM (Value Stream Mapping) process with Swift. This thesis explains different techniques and methods used to program and build the application.

In the modern age of productivity, cutting losses and maximizing productivity has become a key issue in the work environment. Many methods have been implemented along the years to achieve maximum productivity with minimum waste. From Frederick Winslow Taylor's standardization and best practice deployment in 1921 to Sakichi Toyoda's [1] TPS (Toyota Production System) have all contributed of today's VSM (Value Stream Mapping). VSM is a tool that helps to visualize and understand streamline work processes using lean manufacturing techniques and tools. Lean manufacturing as the name suggests is a method that helps to eliminate waste in production. VSM's main purpose is to provide optimal value to the customer through a complete value creation process with minimum waste in design and build. Using VSM allows seeing waste and plan to eliminate it faster. Picture 1 is an example of a VSM graphic document.



Picture 1. VSM example [2].

Ovikone Group OY has commissioned an IPad application to help them with productivity and reduce waste in their production. The VSM app should map the current service and production process chains with different sections of the process streams and produce value stream maps in graphic documents. This thesis focuses on the programming side of this application.

The thesis begins with the introduction chapter which states the purpose of the thesis. The Swift programming language and terminology are discussed in detail in the Chapter 2. Chapter 3 discusses the technical part of the thesis with the method required and the application design. Finally, the thesis is concluded in Chapter 4.

2 IOS AND SWIFT

iOS (originally iPhone OS) is a mobile operating system created and developed by Apple Inc. and distributed exclusively for Apple hardware and products. It is not licensed to other manufacturers. [3]

2.1 IOS operating system

IOS was unveiled to the public in 2007 [3] for the iPhone. Since then, it has been extended to support other Apple devices such as the iPod Touch and the IPad. IOS is a closed source operating system and runs only on Apple device. However, these restrictions have not prevented it from succeeding. In 2015 there were more than 1.4 million iOS applications and apps running on the iOS have been downloaded more than 100 billion times.[4] iOS major versions are released annually and the current release, iOS 9 was released in September 2015.

Most of the devices that run iOS are touch devices apart for the old devices such as the IPod. Touch gestures help the user to interact with the touch devices. Touch gestures are used for scrolling, zooming, rotating and many more functions depending on the application. iOS is written with C, C++, Objective-C, and Swift. IOS apps are either written with Objective–C or using Swift.

2.2 Swift programming language

Swift is a multi-paradigm, compiled programming language created by Apple. A multi-paradigm language is a programming language that supports more than one programming paradigm. A paradigm is a fundamental style of building the structure and elements of a computer program. The main programming paradigms are imperative, declarative, functional, object-oriented, procedural, logic, and symbolic programming. Chris Lattner began developing Swift in 2010. It was four year later that Swift was publicly released at the World Wide Developer's Conference. Swift is a powerful and intuitive programing language for devices running on iOS, OS X and WatchOS. In June 2015, Apple announced Swift 2.0 at the Apple Worldwide Developer Conference. [5]

Apple's main objective when building Swift was to make it fast. Using the high-performance Low Level Virtual Machine (LLVM) compiler, Swift code is transformed into optimized native code that achieves the maximum out of the modern hardware. The syntax and library have also been tuned to make programming code easier to read. Swift has been derived from both the C and Objective-C languages. It is a low-level primitive language that contains types, operators, and flow control and provides object-oriented features such as protocols, classes and generics. [6]

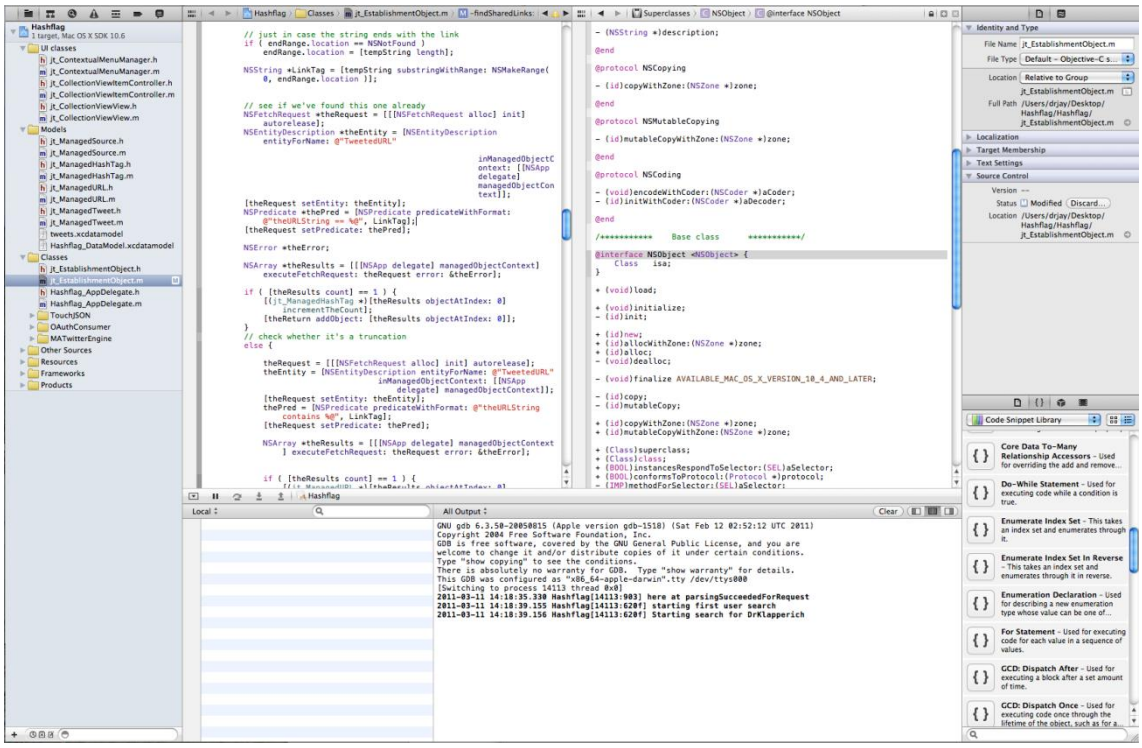
2.1.1 Swift Vs Objective–C

The @ symbol in Objective–C is removed in Swift. In addition, semi-colons and brackets are not necessary anymore. Swift is easier to maintain because it combines the Objective-C's header (.h) and implementation files (.m) into a single code file (.swift). Objective-C on the other hand requires programmers to maintain two code files in order to improve the build time and create efficiently the executable app. Swift is safer because it triggers a runtime crash if a nil variable is used whereas in Objective–C, a call with an uninitialized pointer variable triggers nothing which can be beneficial but is error prone. Swift is the language of the future because Apple allows developers to influence directly the language. [6]

2.3 Development Environment

Xcode is the only official IDE for developing software for Macintosh computers running OS X and iOS mobile devices. Xcode was first released in 2003 by Apple for developers. Apple released Xcode 7.1 on October 2015. It is available via the Mac App store free of charge for only OS X, Yosemite, and OS X El Captain users.

Downloading and using Xcode is free of charge but releasing an application to the market is not free. Developers have to pay 99 dollars to Apple so that they can test their application on a real device. Xcode is built with a simulator so that each users can test their program in an IPad environment. Xcode is helpful because it has a Professional Editor with advanced code completion, code folding, and message bubbles that display warning and errors. In addition it can build, install, run and then debug any Cocoa Touch apps in Mac-based iOS Simulator. [7]



Picture 2. Xcode split screen [7].

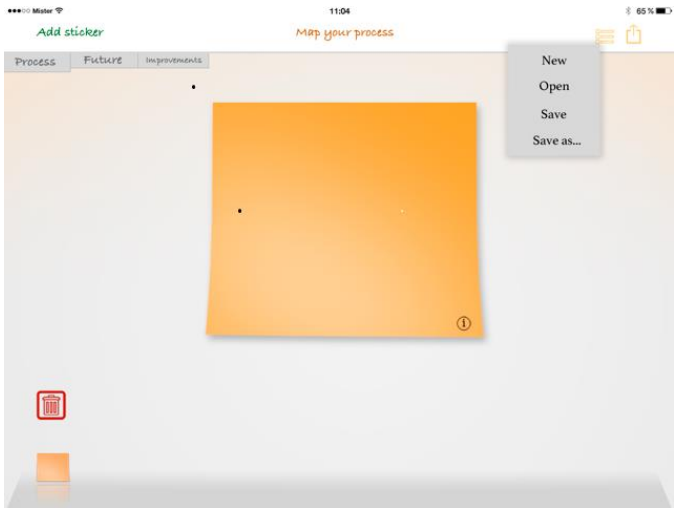
2.4 Programming Environment

Cocoa and Cocoa Touch are Apple native Object-Oriented Application Programming Interface (API) for OS X and iOS respectively.

Cocoa Touch is a User Interface (UI) framework for building software programs to run Apple iOS. It is written in Objective-C and it is based on Mac OX. Cocoa Touch was based on Model View Controller (MVC) architecture. MVC is a software pattern to help organize and implement user interfaces. The Cocoa Touch framework implements the root class and NSObject which defines basic object behavior. It implements classes that represent primitive types such as string and numbers; it also represents collections such as arrays and dictionaries. The framework also provides object persistence, file management and XML processing. The VSM app user interface uses the AppKit and UIKit frameworks. These two frameworks include everything needed for developing the app, such as event handling, drawing, animation, image-handling, text processing, typography and inter-application data transfer. These two frameworks also include UI elements like buttons, table views, text fields, alert dialogs, slider, and many more elements.

3 METHOD AND APPLICATION DESIGN

The design is based on the commissioning company's short UI presentation. (Picture 3) Scrum and Trello [8] [9] are used as project management tools that help prioritize tasks and make collaboration easy. Scrum is an agile framework which is consistent with the agile manifesto [10]. Scrum consists of a product backlog that contains user stories (short, simple description of a feature told for the perspective of the client), sprint planning (the team prioritize one or two user stories), sprint time (usually two weeks to four weeks) and at the end of the sprint time, there is a sprint review and retrospective so that the team can meet and discuss the effectiveness of the sprint time. There are four sprint in this thesis.



Picture 3. UI presentation for the app.

3.1 First sprint

The first task on Trello is to create a simple button that creates a UIView as a placeholder.

UIViews are very important and essential in iOS development. The UIView class is defined by a rectangular area on the screen on which it shows its content. This class handles the rendering of any content in this area where gesture, touch recognizer and many more features can be added. [11]

The objective in this sprint is to map (model, document) an existing process and to classify the tasks of that process timeline as value-adding work, auxiliary work or pure waste by using post-it note like symbols so that user can use the VSM app in a way that is intuitive and customary, and so that the waste in the processes can be identified.

After seeing another VSM app (LucidChart), the practical approach is to create a timeline similar to that iPhone photo viewer. On button press, a new sticker is generated positioned right next to the previous one just like when scrolling photos on iPhone. For proper implementation of this behavior, there are three choices: the first one is to create a UITableView so that each sticker would fit in each column. The second one is to create a UICollectionView so that each sticker would go inside a UICollectionViewCell. The last one is to create a UIStackView which is quite simple because the class itself provides a streamlined interface for laying out a collection of views in either column or a row. At the end the UIStackView is convenient for this thesis and also this class provides much more functionality compared to the other two. Furthermore, the UIStackView provides a user interface that can dynamically adapt to the device orientation which cannot be found in the other two classes.

When the user creates a sticker, the UIStackView adds it to an array called StickerArray. This array come in handy because UIStackView has a property called arrangedSubviews that manages the layout of all views inside the array. These Views are arranged along the stackView's axis based on their order in the arrangedSubviews array. It can be seen from picture 4, StickerArray is the array's name and stackView is the UIStackView.

```
StackView = UIStackView(arrangedSubviews: StickerArray)
```

Picture 4: Initializing the UIStackview

Most of the time in Value Stream Mapping different stages of work or production are represented by a symbol or a color. This app uses color to represent different stages of the work. There are three traditional colors of VSM (green, orange and red). The value adding work will be represented by green, orange will represent the auxiliary work and red will represent pure waste. To implement this, three buttons are created on the UIView. Buttons in Swift are a component of UIView and they can be added as subviews in a parent View. These three buttons can change the color of their respective UIView. A target and a function are added on each button so that each time the button is clicked, the respective UIView changes color. ButtonGreenColor is the button name and buttonGreenColorPressed is the function name. (Picture 5)

```
buttonGreenColor.addTarget(self,action:#selector(ViewController.buttonGreenColorPressed(_:)),f
orControlEvents: UIControlEvents.TouchUpInside)
```

Picture 5: buttonGreenColorPressed function

3.2 Second sprint

The main objective in the second sprint is to create one infinite scroll of stickers which the user can scroll to bring the sticker to edit mode. The focused sticker should be zoomed in the center of the screen while others subside.

The UIScrollView class provides support for displaying content that is larger than the size of the superView and also enables scrolling within that content by making swiping gestures. [12]

After creating the UIScrollView, it is added to the superView as a subview. After that, the stackView is added into the UIScrollView as a subview so the user can scroll through the stickers that he created.

CGAffineTransformMakeScale(X, Y) is a function that scales a given View. X and Y represent the factor by which to scale the x-axis and y-axis respectively of the coordinate system. The default values of X and Y is 1. To make a larger View 1.5 is used for X and Y.

The zooming part works now, but a method is needed in order to focus a sticker in the center of the screen. To make this happen, the UIScrollViewDelegate protocol is used. This protocol allows the adopting delegate to respond from the UIScrollView class and respond to some event such as scrolling, zooming or deceleration of scrolled content., the scrollViewDidEndDragging function is used to decelerate scroll and to call the CGAffineTransformMakeScale(X, Y) inside it.(Picture 6)

```
func scrollViewDidEndDragging(_ scrollView: UIScrollView,
                              willDecelerate decelerate: Bool).
```

Picture 6: scrollView Function

3.3 Third Sprint

The main objective of the third sprint is to create the settings of each sticker by flipping it. The desired setting includes change of background color, font, and font size. There is a data type called UIViewAnimationOptions that animates two views using block objects. The goal is to create a transition animation with transitionwithView and when the user touches the flip button the first view in this case the stickerView (it is the UIView that the user creates with the create button at the beginning) becomes hidden, and then the second View (the view that contains the settings with the background color, font, and font size option) becomes visible. (Picture 7)

```
Func flip()

{

let transitionOptions: UIViewAnimationOptions = [.TransitionFlipFromRight,
.ShowHideTransitionViews]


        UIView.transitionWithView(secondView, duration: 1.0, options:
transitionOptions, animations: {

            self.secondView.hidden = false

            }, completion: nil)


        UIView.transitionWithView(sentView!, duration: 1.0, options:
transitionOptions, animations: {

            self.sentView.backgroundColor = UIColor.clearColor()

            self.sentView.opaque = false


            }, completion: nil)
```

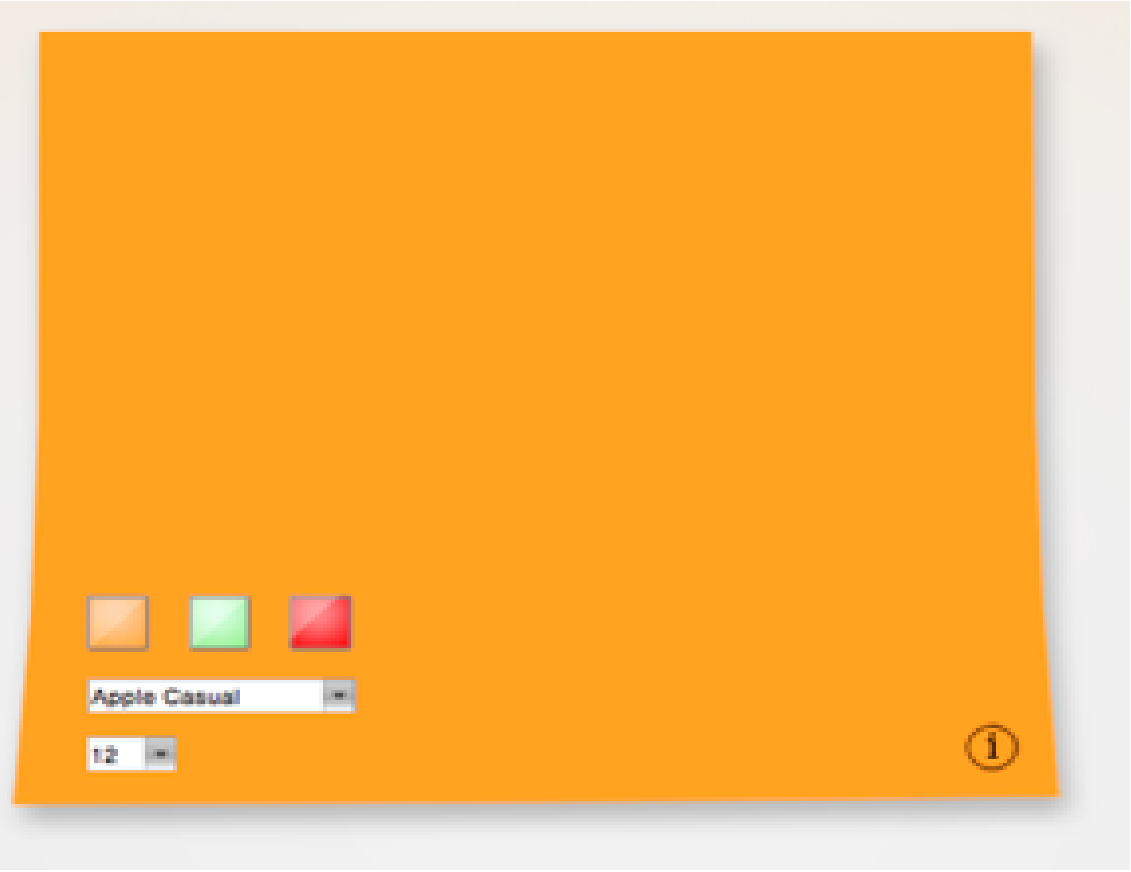
Picture 7: Function flip

Picture 8 is the buttonFlip function.

```
{
performSelector(#selector(ViewController.flip), withObject: nil, afterDelay: 0)
}
```

Picture 8: performSelector function.

To represent options to choose a font size and font family, two dropdown lists are created for the font family and the font size and three separate buttons for the color. Picture 9 is the UI presentation after some small changes implemented by the client.



Picture 9. UI presentation.

In other programming languages, for example in HTML or Java, creating a dropdown list is very simple because these languages already have a given library that supports this feature. UIPickerView uses a spinning-wheel or a slot machine to show one or more sets of values. Using UIPickerView by itself will not look or appear as a dropdown list. In order to that, the best approach is to create a label with a picture as a background image of an arrow pointing down. When the user presses the label, the UIPickerView becomes visible just below the label. This gives the user, the same option as a dropdown list. The same method was also applied for both the font size and font family. The next objective is to create two scrollable timelines. The main timeline should contain stickers of a normal size. The second timeline should be an icon sticker. To implement this a new smaller version of the normal size sticker from the sticker array is created and put as a subview in other UIScrollView. (Picture 10)

```
for sticker in stickerDictionary{
            let iconView = UIView()
            iconView.frame = CGRectMake(0,0,200, 130)
            iconView.layer.cornerRadius = 5
            iconView.tag = sticker.tag
            //iconView.backgroundColor = UIColor.redColor()
            for case let textField as UITextField in sticker.subviews{
                if (textField.tag == 7){
                    switch(textField.text!){
                    case "orange":
                        iconView.backgroundColor = UIColor.orangeColor()
                    case "red":
                        iconView.backgroundColor = UIColor.redColor()
                    case "green":
                        iconView.backgroundColor = UIColor.greenColor()
                    default:
                        iconView.backgroundColor = UIColor.greenColor()
                    }
                    }
                }
            for case let textField as UITextField in sticker.subviews{
                if (textField.tag == 8){
                    let  textCurrentName  =  UITextView(frame:  CGRectMake(2.0,
1.0, 50.0, 30.0))
                    textCurrentName.textAlignment = NSTextAlignment.Center
                    textCurrentName.font = UIFont.italicSystemFontOfSize(8)
                    textCurrentName.textColor = UIColor.blackColor()
                    textCurrentName.backgroundColor = UIColor.clearColor()
                    textCurrentName.tag = 8
                    textCurrentName.text = textField.text!
                    textCurrentName.editable = false
                    iconView.addSubview(textCurrentName)
                }
            }
            iconView.heightAnchor.constraintEqualToConstant(35).active = true
            iconView.widthAnchor.constraintEqualToConstant(55).active = true
            stickerDictionaryIcon.append(iconView)
        }
```

Picture 10: Icon sticker function

3.4 Fourth sprint

The target in the last sprint is to save the stickers as pdf files and to be able to search a file for a particular sticker.

Loading and saving data in apps development is very common. In iOS, there are many ways to accomplish this: NSUserDefaults, Plist, CoreData, NSCoding Protocol and many more. NSUserDefaults is a simple key value dictionary that stores basic data type, such as YES/NO values, for specific user preferences. It does not support querying and it is restricted to a limited set of objects so it is not convenient for larger amount of data. For this app CoreData is used because it is fast and provides tools for complex data models, such as querying and automatic migrations.

The disadvantage of working with CoreData is that it is so large and complex that every developer who worked on this faced some kind of problems in the earlier days. In 2014, Realm was introduced and integrated to the core data. Realm is the first database built from scratch for mobile devices [13]. It is fast and simple to use, and querrable by code. The Realm database was used since it is very convenient for this thesis [14].

To implement this, a class object was created.

```
class StickerFile: Object
 {
    **appendix 1**
}
```

After creating this class, (in the appendix 1) it just need to call this object whenever it is necessary to write to the file, save to the file or delete the file. Picture 11 is the code used for writing to the file and saving it. (Picture 11)

```
let realm = try! Realm()
            try! realm.write {
                realm.add(stickerFile)
            }
```

Picture 11: write to a file

To delete a file, the following code was used.(Picture 12)

```
try! realm.write {
                realm.delete(retrievedFileNames.first!)
            }
```

Picture: 12 delete a file

In order to search for a saved file, UITableView is created and each saved file name is put inside the table rows. The file name will be arranged in order of the saved date in the table and will be much easier for the user to search for a specific saved file. To put the saved file in each table rows, this code below is used.(Picture 13)

```
func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath:
NSIndexPath)->UITableViewCell{

        let cell:UITableViewCell =
allFilesTableView.dequeueReusableCellWithIdentifier("cell", forIndexPath:
indexPath) as UITableViewCell

        cell.textLabel?.text = self.retrievedFileNames[indexPath.row]

        return cell

    }
```

Picture 13: TableView Function

4 CONCLUSION

The main goal of this thesis is to develop an iOS tablet application that implements a VSM process. This thesis focuses on different methods and functions used to develop this application. Scrum and Trello are also used as a project management tool to help prioritize task and make collaboration easy. To make this thesis clear, the third chapter is organized by the order of the sprints carried out in the project. Most of the app developments processes mentioned in this thesis are reusable over and over again in so many different ways.

My first obstacle when writing this thesis was learning a new programming language and at the same time participating in the project. When working with scrum, the success of the project depend on the technical skill of the developer. And that is why all four sprint lasted longer than intended.

The first sprint was hard to implement because it is hard to visualize and program a process timeline. When writing this thesis, the main obstacle was communicating with the client and using Trello. Communication is a key issue in a successful project work.

REFERENCES

[1] “Lean manufacturer” [online]http://www.leandeployment.com/lean-manufacturing.html [Accessed 01 October 2015].

[2] “Best Value Stream mapping mac software” [online] http://www.conceptdraw.com/examples/free-value-stream-mapping-for-mac [Accessed 10 October 2015].

[3] “Swift” [online] https://developer.apple.com/swift/blog/?id=29. [Accessed 15 October 2015].

[4] “Google play VS app store” [online] http://www.androidauthority.com/google-play-store-vs-the-apple-app-store-601836/ [Accessed 15 October 2015].

[5] “WWDC” [online] http://www.macrumors.com/roundup/wwdc/ [Accessed 15 October 2015].

[6] “Swift vs. Objective-C” [online] http://www.infoworld.com/article/2920333/mobile-development/swift-vs-objective-c-10-reasons-the-future-favors-swift.html [Accessed 15 October 2015].

[7] “Xcode” [online] https://developer.apple.com/xcode/features/. [Accessed 15 January 2015].

[8] “Scrum” [online] https://www.scrum.org/ [Accessed 15 March 2016].

[9] “Trello” [online] https://trello.com/ [Accessed 15 March 2016].

[10] “Why Scrum” [online] https://www.scrumalliance.org/why-scrum [Accessed 15 March 2016].

[11] “UIView” [online] https://developer.apple.com/library/iOS/documentation/UIKit/Reference/UIView_Class/ [Accessed 15 March 2016].

[12] “UIScrollView” [online] https://developer.apple.com/library/iOS/documentation/UIKit/Reference/UIScrollViewDelegate_Protocol/. [Accessed 10 April 2016].

[13] “Realm” [online] https://realm.io/docs/swift/latest/api/ [Accessed 25 March 2016].

[14] “Realm” [online] http://bsktapp.com/blog/why-is-realm-great-and-why-are-we-not-using-it/ [Accessed 25 March 2016].

Appendix 1. File Object

```
class StickerFile: Object {
    dynamic var fileName = ""
    dynamic var dateCreated: NSDate? = nil
    dynamic var dateUpdated: NSDate? = nil
    let stickers = List<Sticker>()
    override static func indexedProperties() -> [String] {
        return ["fileName"]
    }
}
```