****

**Student Number:** 12300216

**Student Name:** - John Brennan

**Programme:** CASE4 - B.Sc. in Computer Applications (Software Engineering)

**Faculty:** School of Computing

**Academic Year:** 2015/2016

**Project:** Sonrasc - Technical Specification

**Supervisor:** Alexander O’Connor

**Module code:** CA4004

**Project Due Date:** 23rd May 2016

**Declaration**

I the undersigned declare that the project material, which I now submit, is my own work. Any assistance received by way of borrowing from the work of others has been cited and acknowledged within the work. I make this declaration in the knowledge that a breach of the rules pertaining to project submission may carry serious consequences.

I am aware that the project will not be accepted unless this form has been handed in along with the project.

**Signed:    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. Introduction

1.1 Purpose

1.2 Scope

1.3 System Overview

1.4 Solution Requirements

1.5 Solution Deliverables

1.5.1 Using and configuring

2. Design Considerations

2.1 Assumptions and Dependencies

2.2 General Constraints

2.3 Goals and Guidelines

2.4 Development Methods

2.5 Architectural Strategy

3. System Architecture

4. Policies and Tactics

5. Detailed Subsystem Design

5.1.1 <<Component>>

6. Test Suite

7. Current planning and Issues

8. Acceptance Sign-off

**Motivation**

the problem at home.

Lots of paper. No single source to view everything. Messy.

Want a way to help track the finances, where money is going and what kind of items is the farm purchasing, means of giving a way to plan expenditure with the past in mind.

**Research**

Trying to do with machine learning.

The problem

Invoices are like snowflakes.

Not enough data

But means of future work.

A rule based approach.

Architecture of a system

**Design**

**Implementation**

**sample code**

**problems solved**

**results**

**future work**

**microservices**

scalability http://microservices.io/patterns/microservices.html

Netflix, amazon.

Communication of processes

Functionality to consumer.

Separation of concerns, independently evolve, deploy and scale.

Rest based, lightweight based.

https://www.youtube.com/watch?v=X0\_d0ci8UcY

Benefits over monolithic

* Continuous development hindered where a single component is slow and risky.
* Sub optimal impact on scalability, can’t scale individual parts.
* Adverse impact on development,
  + Larger codebase, harder to maintain.
  + Long-term commitment to stack

Disadvantages of microservices:

* Deployment of monolithic is simpler
* Commitment to automation for testing and deployment
* Operational complexity of hosting/starting/managing/monitoring larger number of separate processes
* Need to consider messaging patterns

The Scale cube

Can help for sharding in databases, but a lot of complexity

Functional decomposition.

Why node is good for it

Good package management system

Minimal ceremony in publishing packages with npm publish

Lightweight, narrlowly focused packages, designed from beginning to end to end async i/o to prmote highly scalable services

Good performance

End to end javascript stack can accelerate development.

**1.0 Introduction**

Sonrasc is the Irish translation of the word ‘Invoice’. This document will discuss the system Sonrasc, an invoice processing and visualisation tool. The system is a web application which users can log on to and upload invoices, where it will learn the layout of an invoice based on the pictures and regions of interest the user indicates. It is to facilitate a single source of

relevant information that is stored and can be visualised in graphs and search for automatic retrieval, which businesses can use for expenditure planning.

**1.1 Purpose**

The purpose of Sonrasc is to aid business owners and employees (‘the users’) with a method of processing and storing invoices that they receive from other companies. The user indicates regions of importance in an invoice so that the system can automatically extract information from new invoices of the same type in the future. Automatic extraction of information is for the cutting down the manual input of data into a method of tracking invoices in a work environment. The system provides a means of viewing invoices in a single application, where they can visualise an aggregated view of all invoices that are on the system. With purchases present on the system, it is to facilitate expenditure planning/tracking in the business with the help from historic data.

**1.2 Motivation**

Sonrasc is derived from the need of a more efficient solution in Pepperstown Holsteins, based in Ardee, Co. Louth. New invoices regularly come into the paper workflow of the management of the farm. Before the development of the system there has been no automatic way to process the data. When the need for budgetary planning arises, invoices had to be handled manually. It is a time-consuming and difficult process when the only method of storage is manual filing with no way to automatically retrieve documents. With no single-source to view an overview of invoice information, the current method was cumbersome to work with. A solution of a way to track finances, where money was being spent and a method to determine quickly what kind of items the farm was spending was desired.

With the rise of popularity, availability and use of smartphones and the capabilities they provide, it is desired to be able to leverage the power of a smartphone camera. Specifications of phone cameras are improving every year, which can be used to take clear pictures of text which is legible in the output. The motivation is to have a means of using a user's phone to capture information of an invoice.

**1.3 Scope**

The system is concerned with the extraction of information from invoices that are relevant to an individual business. The scope has also applies to the storage, means of searching and viewing of invoice information. Sonrasc is concerned with making the information extraction process quicker for the user than of manual input by providing means to automatically extract information based on previous invoices. From the motivation the system was derived from, it is in scope to improve the invoice handling process of Pepperstown Holsteins to introduce a means of automatic information retrieval and lessen the need for manual handling invoices.

For this project to demonstrate its viability and the contextual usage with a real life example, permission was given by the owners of Pepperstown Holsteins to use their invoices as testing data. The scope of the project is used against over 500 invoice images sent to the business from 2009-2011. Invoices were captured using a Nexus 5 smartphone camera.

**1.4 System Overview**

**1.5 Solution Requirements**

Safe storage of invoices.

**1.6 Solution Deliverables**

Website which users can log into.

**2.0 Research**

**2.1 Automatic Invoice Information Extraction**

In order to meet the core requirement of having a method for a system to extract information out of an image (“The invoice”), the system must have a way to “read” the image and output text information. This is an application of Optical Character Recognition (OCR). Upon investigate, an open source OCR engine called ‘Tesseract’, a package developed by Hewlett-Packard and then maintained by Google [1] was recommended as the system to try and fulfill the requirements. On initial project research, it was believed that this package alone would be sufficient to meet the project's requirements.

**2.1.1 Tesseract OCR**

In order to achieve an output of text, Tesseract uses a step-by-step pipeline [2]. It does region detection of text and stores them, the regions stored are then organised and broken into words depending on character spacing. Recognition then takes place where words are identified by an adaptive classifier. An adaptive classifier outputs a sequence of binary classifications employing both supervised and unsupervised learning techniques [3]. In this case, outputting a series of classification for each text character identifying if parts of the image correspond to a certain character. An image may go through several stages of preprocessing to ensure optimal accuracy of text extraction.

Deskewing takes places as part of the pipeline, the process of straightening an image. This is highly relevant for the system where input can be a photograph. Pictures can be taken from an angle and trying to mitigate the effect it has on the OCR engines performance.

**2.1.2 The Complexity of the Requirement**

On initial testing, Tesseract performed well on individual invoices. Text was being output in a single text file. Problems lied in the issue of attempting to scale the solution. Invoices have different formats for different businesses, with possibly different fields and layouts. It proved quickly that there is no standardised way to collect and extract information from Tesseract outputs for a range of different invoices. For this method to work, there would need to be individual training to develop a model for a specific invoice type, which could potentially require over a thousand samples to gain an accurate result. This scale was too large for the current scope of this project and proved infeasible.

In order to overcome this, the solution employed in the system is to get the user to define regions of interest in an invoice. Most of the invoice we are not concerned about, more so getting the information that applies to the majority of invoices, such as items purchased, their cost, quantity, who it came from and on what date was the invoice was received. So it was a design change to have the user, in the application to crop the spaces where the information, where it would remember for future invoices of the same type. There is drawbacks to this, where the system will suffer to the Cold Start problem, where systems are unable to infer solutions to information it hasn’t collected information about yet. Having user defined regions of interest has its own challenges too with user-uploaded photographs. Pictures taken at an angle or at a different position to the previously taken invoices photographs will have an effect on the system accuracy.

**2.2  Research of System Architecture Design**

For the design of the system, a trend of the ‘microservice architecture’ arose. This is the idea of design ‘independently deployable services’ [4]. The architecture has been implemented and made popular by companies such as Netflix, HubSpot and Amazon. Each component in the system is designed as a small service, components are accessible by representational state transfer (REST) interfaces. There is a separation of concerns, services and independently evolve, deploy and scale.

Compared to a traditional Monolithic architecture there is several benefits, where continuous development is hindered by the slowness and risk involved in replacing a module. Scalability of individual components isn’t possible in a monolithic architecture, this slows down development. There is a larger codebase to maintain which could pose more difficult. There is also a long-term commitment to a particular stack.  With microservices and continuous deployment , there is an emphasis and encouragement for test-driven development to ensure all functionality works and nothing breaks when a new change is pushed into a production environment. Scaling of individual services is much more flexible than a monolithic environment. For these reasons, it has been a design decision that the product is implemented using Microservices.

**3.0 Design**

1.5.1 Using and configuring

2. Design Considerations

2.1 Assumptions and Dependencies

2.2 General Constraints

2.3 Goals and Guidelines

2.4 Development Methods

2.5 Architectural Strategy

3. System Architecture

4. Policies and Tactics

5. Detailed Subsystem Design

5.1.1 Invoice Uploading

5.1.2 Optical Character Recognition System

5.1.3 Text parsing

5.2 Invoice View

5.3 Businesses View

5.4

6. Test Suite

7. Current planning and Issues

8. Acceptance Sign-off

based on previous invoices from the same business is used to try and extract information from a new invoice automatically, A user can intervene when needed to ensure the system is capturing the correct information.

**Design**

**Implementation**

**sample code**

**problems solved**

**results**

**future work**

Improvement of ocr

Domain specific training data.

**microservices**

        scalability http://microservices.io/patterns/microservices.html

        Netflix, amazon.

https://www.youtube.com/watch?v=X0\_d0ci8UcY

Disadvantages of microservices:

-       Deployment of monolithic is simpler

-       Commitment to automation for testing and deployment

-       Operational complexity of hosting/starting/managing/monitoring larger number of separate processes

-       Need to consider messaging patterns

The Scale cube

Can help for sharding in databases, but a lot of complexity

Functional decomposition.

Why node is good for it

Good package management system

Minimal ceremony in publishing packages with npm publish

Lightweight, narrlowly focused packages, designed from beginning to end to end async i/o to prmote highly scalable services

Good performance

End to end javascript stack can accelerate development.

**References**

[1] Tesseract, Tesseract OCR, (2016), GitHub repository, https://github.com/tesseract-ocr/tesseract

[2] Smith, R. (2005). An Overview of the Tesseract OCR Engine. *Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR)*, 629–633.

[3] IEEE Xplore Abstract - Adaptive classification . 2016. IEEE Xplore Abstract - Adaptive classification . [ONLINE] Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=882446.

[4] Lewis, J. and Fowler, M. (2014) Microservices. Available at: http://martinfowler.com/articles/microservices.html