Requirements Analysis Document
Historic Census Scanning Project

CITS3200 - Professional Computing

Winter 2024

University of Western Australia

Revision History:

Version R0.1 31/07/2024 - Collaboration with all project members

Preface:

This document addresses the requirements of the historic census scanning system. The intended audience for this document are the designers and the clients of the project.

Target Audience:

Client, Developers

Project Members:

- Ciaran David Petrus Engelbrecht <23169641@student.uwa.edu.au>
- William Sydney Lodge <22980141@student.uwa.edu.au>
- William Forrest Stewart van den Wall Bake <23086983@student.uwa.edu.au>
- Shashwat Abrol <23482415@student.uwa.edu.au>
- Connor James Fernie <23443143@student.uwa.edu.au>
- Oliver John Dean <21307131@student.uwa.edu.au>

MILESTONES

9/15 Release of RAD Template (Project Management)
10/7 Team RADs Due (Developers)
10/11 Second Draft of team RADs due (Developers)
10/17 RAD Review Presentation Deadline

1.0 General Goals

1) Create a system able to scan historical census data and turn into machine readable tables at speeds faster and more automated than with human intervention
2) Have this system have accuracy equal to or greater than human equivalents of entry
3) Allow user to examine and see processing of system and be drawn towards any errors or misreading of scans
4) Allow user attention to be highlighted to failed inconsistencies between output of text of different OCR; And allow users to rectify inconsistencies.

2.0 Current System

Currently requires long boring work of manually entering data from tables 1 by 1; And takes long periods of time doing boring work. Automating parts or this entire process would greatly accelerate the availability of this census data to allow for it to be computed for study.

3.0 Proposed System - Connor

The software will comprise of a web-based graphical user interface which will allow a user to upload PDF documents in which they want information extracted from. A flask backend will be implemented, and standard web languages will be used to implement the frontend GUI for the user.

The core functionality of the subsystem software has been divided into the following sections:

1. Conversion of PDF pages into images
2. User input to identify rows and columns of tables
3. Separation of tables into individual cells according to user input
4. Output individual cells to create a cell-by-cell image collection
5. Apply image enhancement techniques and tools to each cell image
6. Apply optical character recognition (OCR) to individual cells
7. Output extracted text in suitable data structure.

The core functionality will be implemented using python and libraries such as pytesseract to implement OCR.

3.1 Overview - Connor

The proposed subsystem will provide a user with a web-based graphical user interface (GUI) to upload a PDF containing data tables and a user-friendly method of selecting the information in the data tables which will be extracted by the software. Information extraction will use optical character recognition (OCR) along with other tools to ensure maximum accuracy. Extracted data will be exported into a CSV format suitable for manipulation and analysis. Tools and techniques will be implemented to ensure image quality is improved to increase the accuracy of the OCR technology to provide the user with confidence in the resulting information contained in the CSV.

3.3.3- 3.3.10 - Ciaran

3.3.3 Hardware Specifications:

Target Hardware: From our current understanding the application/software will be primarily used on a standard desktop or laptop device. 8GB of RAM will be recommended as a minimum for the today's current standards especially during intensive OCR and data processing tasks. Storage of at least 256GB will also be requires for local processing, dependant on the size of total pdf documents required as each image/csv during the process will require storage even temporarily.

3.3.4 Performance Requirements:

Speed and throughput, in such that the system should be capable of processing a standard pdf which may contain 5-20 ages within a minute, and with bulk processing the system should handle multiple PDF's concurrently. The interactive component of the user interface feedback should be immediate ideally. Size and capacity constraints will need to be considered as some PDF's could be hundreds of pages and the extracted table data should be manageable within the csv file software limits (amount of rows/columns)

3.3.5 Error handling:

Input errors will need to be considered, such as validating the file input is a PDF and providing meaningful error prompts for unsupported files or if the file is unreadable.
Then the considerations of accuracy of the text will be another error consideration highly dependent on the document quality.

3.3.6 System Interfacing:

PDF's will be the primary input through user local storage and pathway directory, potential for uploading as an additional component.
Output will be CSV file containing extracted table data that can be managed through the local exported file.
Format restrictions are limited to PDF format as the input and CSV for output, all documents will be PDF as discussed in the client meeting and it was requested all output is CSV.

3.3.7 Quality Issues:

The system should have high reliability, no unnecessary crashing of the software and appropriate error messaging when something does go wrong.
Restarting/opening the software should be quick and be able to continue with processing immediately.
Portability is a further consideration, perhaps being able to use the software on phone OS' in addition to windows/macOS/linux depending on the client's priority.

3.3.8 System Modifications:

The OCR tools and python libraries and table extraction algorithms may need updates as technology improves or are created, the GUI may also need improvements or enhancements

over time.
Potential cloud storage services or AI implementation.

3.3.9 Physical Environment

The operation of the software is dependent on the device being used, such as which operating system and the device specifications, physical location should not matter as everything will be run locally on the device.

3.3.10 Security Issues

Data access control is important to consider, such as sensitive data being digitalised and extracted tables, although after speaking with the client this is not a large concern as long as it isn't being openly shared intentionally, it just needs to be controlled and managed. Potential user authentication for access could be implemented.

3.4 Constraint - Shash

The primary programming languages used for the project are Python for the OCR and data processing tasks and to better improve the performance of some components, however, possibly use C or Java for this part to better the performance as well. Hence, the constraint of the programming language used depends on the library support and ease of use and integration. It heavily depends also on how similar and comfortable the team members are using certain languages. Being rational, it was discussed that Python offers extensive libraries and easy implementation opportunities for OCR (e.g. Tesseract) making it a suitable choice.

Due to the diverse operating systems like Windows, macOS, and Linux. The project should accommodate all members by using universal development environments like Visual Code. Further, development environment constraints to sharing code was eliminated by using Github which allowed code sharing, issue tracking and continuous development.

Constraint on the use of libraries depends on which libraries and framework gets approved by the client since this will ensure that it aligns with the requirements. It is important to use stable and well performing libraries to ensure reliability and minimize risk of vulnerabilities.