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

Version R0.2 12/07/2024 – Editing and proofreading

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

14/08/24	Sprint 1
18/09/24	Sprint 2
14/10/24	Unit deadline
16/10/24	Sprint 3

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 to manually review PDF documents, potentially scan them and enter data into a desired format. Automating parts of, 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

The software will comprise of a graphical user interface which will allow a user to upload PDF documents in which they want information extracted from

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

The proposed subsystem will provide a user with a 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.

3.2 Functional Requirements

The functional requirements for the software are as follows:

1. PDF upload: the system must allow users to upload PDF documents.
2. Image conversion: the system must convert uploaded PDF pages into images.
3. Table identification: users must be able to interactively identify rows and columns within the PDF images, specifying the areas from which data should be extracted.
4. Cell separation: the system must separate identified tables into individual cells based on user input, creating a collection of cell images.
5. Image preprocessing: images are required to be pre-processed by the software to improve the quality of the text in the images. OCR accuracy will improve as a result.
6. OCR text extraction: OCR must be used to accurately extract text from the PDF documents uploaded to the system.
7. Error detection and correction: the software requires functionality to detect and flag errors in the OCR process and notify the user of errors. General errors such as logic errors in the program must be handled appropriately.
8. Output format: the system must format the text that is extracted from the PDFs so that it can be restructured into the original table and be manipulated using languages like python and R for data analysis.
9. Language translation: the system may be required to handle multiple languages and translate language to English if required.
10. User interface: the system requires a basic user interface to allow users to upload documents for OCR extraction processing.

3.3 Nonfunctional Requirements

3.3.1 User Interface and Human Factors

Types of users and training requirements:

the software is expected to be used mainly by non-technical user. Therefore, it is important that the user interface is usable without requiring understanding of OCR technology. In this case, an easy to use, simple user interface will be built so that non-technical users will not require understanding of how OCR technology works and will not require extensive training prior to using the software. Potential non-technical user training could be as simple as a presentation/guide on how to navigate the UI. Technical users will be provided with documentation that explains the software architecture and the processes used by the system.

Error handling:

The software must be intuitive and clearly explain errors when they occur. The software must not be error prone and errors in the results of the OCR process must be indicated to the user in a clear way.

User devices:

The software is going to be built for computers/laptop devices. Windows devices will be supported by this software. The software will not account for mobile devices or other operating systems.

3.3.2 Documentation

There are multiple documentation requirements for the software: user interface guide, technical documentation and deployment documentation.

1. User manuals: documentation will be provided so that users of the system will have instructions on how to navigate the user interface, upload PDFs to the system and retrieve results.
2. Technical documentation: technical users require documentation to maintain and edit the code for the system. Architecture of the/OCR tools and procedures will be documented. This will include in-line documentation to describe the processes of the software.
3. Deployment documentation: technical users and non-technical users may require a guide to downloading the software and setting it up within their system.

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. Meeting these hardware requirements is important to ensure a throughput of 10 pages per minute is achieved with a mid-end device.

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 pages within a minute, and with bulk processing the system should handle multiple PDFs concurrently. The subsystem must process documents at a rate of 10 PDF pages per minute to achieve required performance expectations.

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 (number 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. These meaningful error prompts must be simple enough to be understood by a non-technical user and should not require intervention from a technical user.

Text accuracy and errors in the final extracted text output by the OCR software must be considered. Therefore, the system should make a best attempt to flag errors to the user so that the user can intervene and manually check flagged cells.

3.3.6 System Interfacing

PDFs will be the primary input through user local storage and pathway directory, potential for uploading as an additional component.
Output will be a 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, the initial system will be supported on Windows devices, specifically being tested on Windows 11 devices.

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 may improve the sub system in the future as technology develops.

Libraries used by OCR tools may be upgraded or become deprecated in the future and will require modification to ensure that new libraries are being used, potentially for security or performance purposes.

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. The locality of the device will not affect the functionality of the software.

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 if it isn't being openly shared intentionally, it just needs to be controlled and managed. Potential user authentication for access could be implemented.

Out of date libraries and functions used by OCR tools represent security issues and libraries may or functions may become deprecated as security flaws are discovered. These shouldn't affect this system as it will be run locally and an attacker would need physical access to a machine, however, must be considered.

3.3.11 Resource Issues

This section discusses data management for the subsystem.

1. System backups: the subsystem is not required to be backed up as it is expected to be used as a tool for text extraction for PDF documents. It is a responsibility of the user to ensure that PDF documents and extracted text output is backed up.
2. Data management responsibilities: it is the responsibility of the end-user to ensure that the PDFs uploaded and extracted text outputs are backed up in their system. The sub-system will not provide extensive data management functionality.
3. System installation: the end user is responsible for installing the software on their device. However, it is expected that the software will include an installer and installation guide for Windows.
4. System maintenance: it is expected that technical end users will be responsible for maintaining the system. Technical documentation will be available for technical end users so that they can maintain the software.

3.4 Constraint

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.

3.5 System Model

3.5.1 Scenarios

1. Uploading a PDF for text extraction: a user will start the software and upload a PDF to be processed for OCR text extraction. The PDF will be split into images that are processed by OCR and reassembled into a structured format that the user can manipulate.
2. OCR accuracy errors: a user will start the software and upload a low-quality PDF to be processed for OCR text extraction. The PDF will be split into images that are processed by OCR and reassembled into a structured format that the user can manipulate. However, errors detected in the text extracted will be clearly flagged for manual verification by the user. The user will verify and correct the error, and the text will be output in a structured format.
3. Batch uploading and processing: a user will start the software and upload a batch of PDF documents. The system will prompt the user to verify the tables automatically identified in the PDFs and will process the PDFs after doing so. The extracted text will be provided to the user in a csv format.

3.5.2 Use Case Models

3.5.2.1 Actors

Actors of this system include non-technical users who will be using the system to process PDF documents to extract text from tables and retrieve the output in a structured format, and technical users who will use the system for the same purpose as above but may be required to maintain the sub system and integrate OCR tools.

3.5.2.2 Use Cases

Uploading PDFs for OCR processing:

- The user requires PDFs containing census data in a tabular format to be processed so that the text is extracted and formatted in a csv.
- The user will start the software, upload the PDF documents, verify that the automatic table detection has worked correctly and confirm for processing.
- The software will use OCR to process the documents and output the extracted text in a csv for the user to manipulate.
- Errors will be flagged to the user in a clear method.

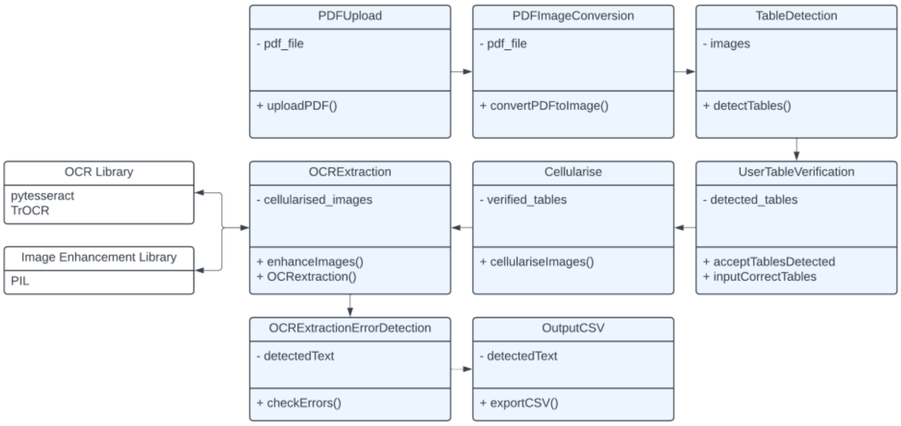
3.5.3 Object Models

3.5.3.1 Data Dictionary

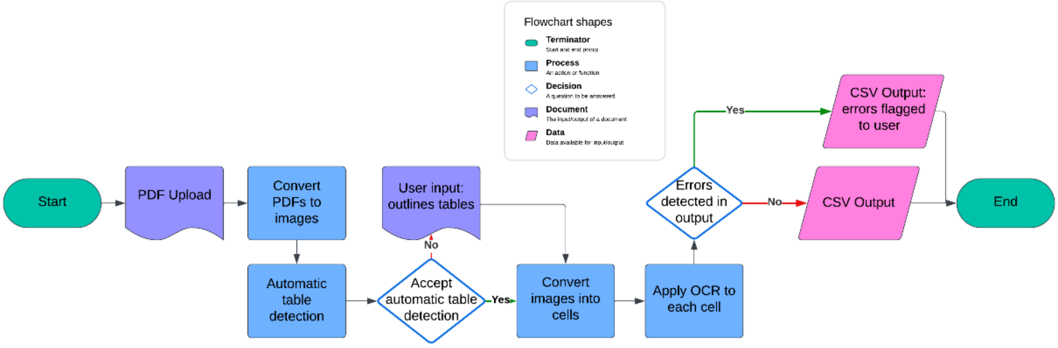
The system is not expected to have a database storing the inputs and outputs of the data, rather it is the user's responsibility to securely store their PDF documents and the corresponding output CSV files after the PDFs have been processed. This is beneficial for security purposes as the software will not be storing important files and storage purposes as files are not stored multiple times on the user's computer.

1. PDF documents: user uploaded PDF documents
2. Image files: temporary files stored for OCR processing because of converting the PDF documents to images and cellularisation for OCR accuracy purposes
3. CSV output: the software will output data in a CSV format for the user.

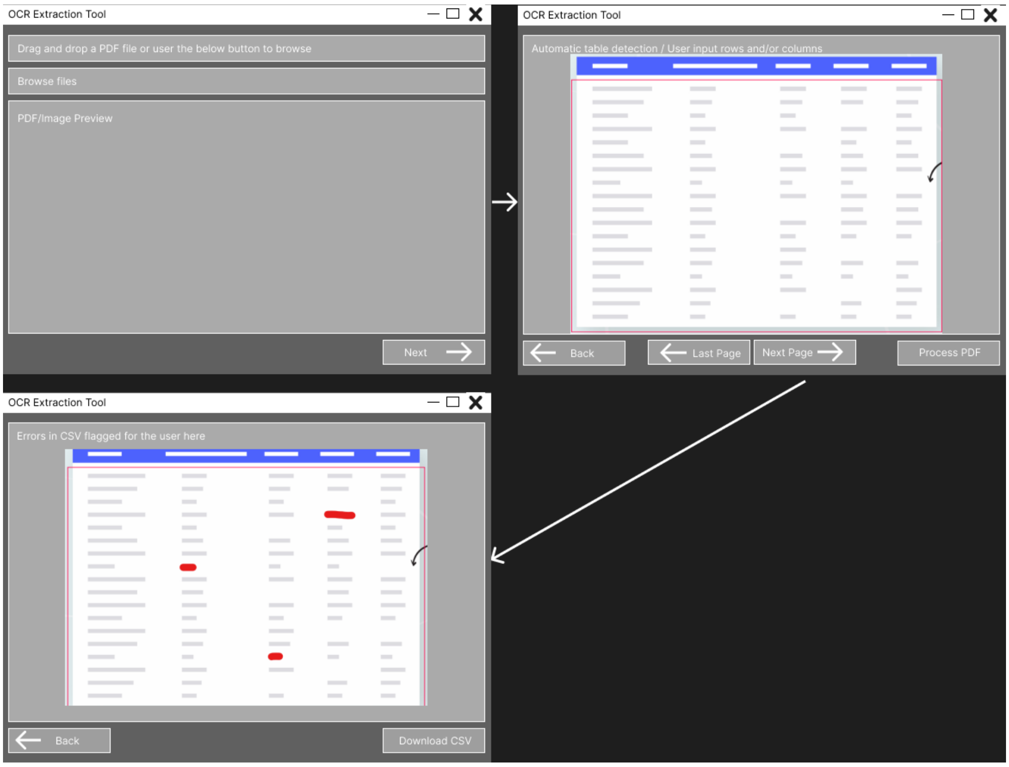
3.5.3.2 Class Diagrams



3.5.4 Dynamic Models


Flowchart shapes
Terminator
Start and end points
Process
An action or function
Decision
A question to be answered
Document
The input/output of a document
Data
Data available for input/output
Start
PDF Upload
Convert PDFs to images
Automatic table detection
Accept automatic table detection
No
Yes
User input: outlines tables
Convert images into cells
Apply OCR to each cell
Errors detected in output
Yes
No
CSV Output: errors flagged to user
CSV Output
End

3.5.5 User Interface – Navigational Paths and Screen Mock-ups


OCR Extraction Tool
Drag and drop a PDF file or user the below button to browse
Browse files
PDF/Image Preview
Next
OCR Extraction Tool
Automatic table detection / User input rows and/or columns
Back
Last Page
Next Page
Process PDF
OCR Extraction Tool
Errors in CSV flagged for the user here
Back
Download CSV