

DOCX and PDF Data Extraction Tool

...

Team X
Crichton Institute

Project Motivation and Initial Requirements

- Client needs to extract raw data from PDFs
- Such software exists but it is:
 - a. Expensive
 - b. Potentially difficult to use
 - c. Not customised to the client's needs
- A machine-readable format (CSV) would allow for further data analysis

Evolution of Requirements

- Addressed the issue of PDF extraction accuracy
- Expanded project scope to include DOCX
- Decided on using a single-page front-end
- Established a preferred output format

The Final Product

A web application where you submit a PDF or DOCX document and download a ZIP of CSVs, extracted from the identified tables within the document, as well as all individual images.

Crichton Institute Extraction Tool


A modern way to extract tables from PDF and DOCX files

UPLOAD FILE



SUBMIT



Click on "Upload file" or  Choose from Dropbox

Product Features (1)

- A single-page application front-end
 - Material design
 - Extremely responsive
 - Fast loading times
- Spring-powered back-end
 - Robust and reliable
 - Java - main server language

Product Features (2)

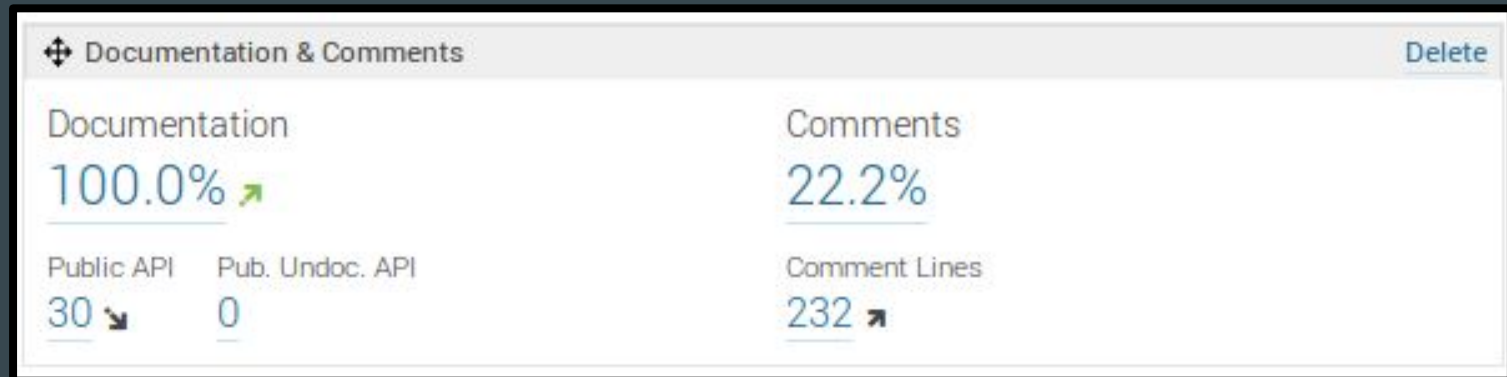
- Data extraction from PDF and DOCX:
 - Table data - CSV files
 - Individual images
- Documents can be uploaded from:
 - Local storage
 - Dropbox cloud storage

Project Achievements (1)

- Technical accomplishments:
 - High accuracy in data extraction from DOCX
 - Industry-tested data extraction from PDF documents,
 - without relying on separators to identify tables
 - Quick document conversion
 - without using up too much resources

Project Achievements (2)

- A high standard of software engineering practices
 - Extensive testing
 - Use of change management systems
 - Good level of documentation



Project Achievements (3)

- Team organisation
 - Allowed us to resolve issues quickly
 - Having specific team roles increased productivity
 - Used additional aids:
 - Wiki pages
 - Tickets
 - Slack
 - Kanban board

Implementation and Technical Challenges

Identifying the Correct Tools

- Tools for reading from PDFs are widespread, however:
 - accuracy in extraction varies wildly
 - quality even more so
- Final software stack we decided on:
 - Tabula for extracting data from the tables
 - PDFBox for opening PDFs and getting all of the images
 - Apache POI for DOCX extraction

Web Application using Spring Boot

- Spring was chosen as the framework for creating our web application for several reasons:
 - Easy to use
 - Compatible with the data extraction tools
 - Scales very well
 - Modular



**spring
boot**

Integration Issues

- Using a wide range of tools and libraries had drawbacks:
 - Adding Tabula to our project was a major hurdle
 - documentation is almost non-existent
 - We had to create the bridge between Spring and the extraction tools
 - Cross-platform compatibility between the different browsers in regards to uploading files

Meeting Non-functional requirements (1)

- Data extraction from documents is quick
- The front-end is appealing and responsive
 - Less than 1 sec loading time for page
- The application is mobile-friendly
 - Adapts well to various screen sizes
- Conforms to client's design guidelines

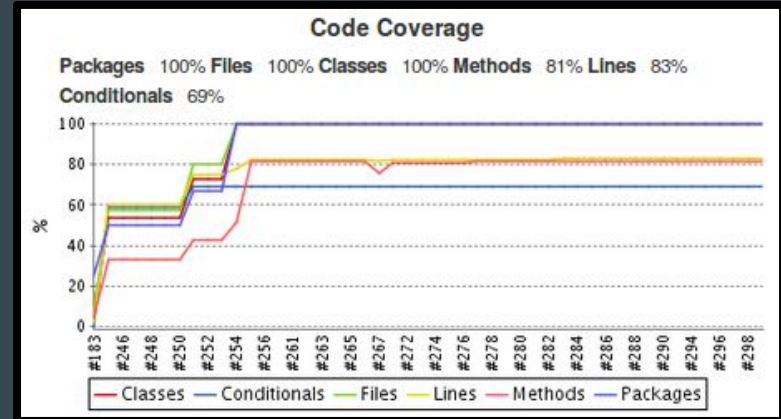


Meeting Non-functional requirements (2)

- It is economical in its resource use by:
 - Deleting uploaded files after conversion
 - Efficient use of external libraries
 - Direct interaction with Tabula
 - No process forking
 - Minimal memory usage
 - No more than 450MB of RAM

Quality Assurance (1)

- Performed a wide array of regression tests using jUnit
- User acceptance testing through client interaction
- Quality Assurance helped discover several major bugs:
 - Extreme memory usage, resulting in a system crash
 - Error with Firefox after two consecutive uploads



Quality Assurance (2)

- Refactored the interaction with Tabula
- Continuous integration on Jenkins
 - allowed us to identify build issues
 - improve code style and quality
- Extensive use of JavaDoc throughout the code base
 - increases readability and maintainability

**Time for a Live
Demonstration**

Future Improvements and Modifications (1)

- Extension to a fully standardised API
 - Currently only some RESTful methods
 - Prevents a monolithic structure
 - Helps to integrate with other products
 - Team A's Economic Dashboard
 - Team V's Area Visualisation Tool

Future Improvements and Modifications (2)

- Processing new formats
 - mostly XML-based
 - XPS is a possibility
- Select which pages to work on
 - would avoid unnecessarily large requests


A modern way to extract tables from PDF and DOCX files



UPLOAD FILE 

PAGES 14-16 

SUBMIT 

Click on "Upload file" or  Choose from Dropbox

Future Improvements and Modifications (3)

- UI for manual accuracy checking

Accuracy Review Table 5 (Page 38)

	FY 2006 Actual	FY 2007 President's Budget	FY 2007 Estimate	FY 2008 Estimate	Change from FY 2007 Estimate
Labor/HHS Discretionary Budget Authority (B.A.)	\$28,287	\$28,190	\$28,389	\$28,631	\$233
Interior B.A.	\$79	\$78	\$79	\$78	-\$1
Total Discretionary B.A.	\$28,366	\$28,268	\$28,468	\$28,700	\$232
Type I Diabetes Initiative	\$150	\$150	\$150	\$150	\$0
Total B. A.	\$28,516	\$28,418	\$28,618	\$28,850	\$232
NIH Program Level	\$28,524	\$28,427	\$28,626	\$28,858	\$232
Number of Competing RPGs	9,129	9,144	9,622	10,188	566
Total Number of RPGs	38,313	37,566	38,089	38,063	-26
FTEs	16,880	17,456	17,216	17,459	+243

Previous
Table

Download
ZIP

Next
Table

Thank You!

Questions?

Project created by:

Ivan Kyosev, Ian Denev,
Richard Pearson, Edvinas Simkus,
Andrei-Mihai Nicolae