Software documentation for

“Legal Proofreading helper”

**Name: Wei Tan and Reuben kirkham**

**Date: 28/04/2022**

[**Product documentation**](#_prw9m0rw4286) **3**

[System documentation](#_ysvnymmr2ft7) 3

[Requirements document](#_6x015xey9lw6) 3

[Software architecture document](#_b7gwk1ppowg6) 4

[Source code](#_8evklqqdpp8f) 4

[Maintenance and help guide](#_zb1g7qtuzet) 4

[User documentation](#_ld6jbsf0gr0l) 5

[End User](#_n6kxoqfciaad) 5

[System Administrator](#_3vo3yhmxl0ip) 6

[**Process documentation**](#_u594wcsejs3c) **7**

[Plans and estimates](#_f2brye6azjfk) 7

[Stage 1: Simple annotations](#_a1hrqzjns58o) 7

[Stage 2: Somewhat more challenging cases](#_xam2uv1mhogl) 7

[Stage 3: More complex annotations](#_jb05v6g0fvrw) 8

[Timeline schedules](#_9951mcvv5ja8) 8

[Standards](#_tg2zty4rfz1d) 8

[Interactions](#_y03ogisgtjp3) 8

[Experiment](#_y2864w1hwqov) 9

[**APPENDIX**](#_w6csxdr2wi8a) **10**

[References](#_k9jnef7xii4r) 10

[Legal document links](#_m0zfcpucpiz0) 10

[Annotations github links](#_q49i83jc3frw) 10

# 

# Product documentation

This software is designed to help the user proofread legal documents.It outlines the product being developed and includes instructions on how to use it to accomplish various tasks.

## System documentation

System documentation refers to documents that describe the annotation system and its components. It comprises requirements specifications, design decisions, architectural descriptions, programme source code, and help documentation.

### Requirements document

A requirements document details the software's capabilities. In general, requirements are statements about the functions that a system should do. The details are explained in the Appendix.

* + - * Set up a experiment for the user to annotate on the legal document
        + Configure the pdfList.json to predefine legal documents

Match the names and paths of the local PDF files with the keys and values in json file

* + - * + Configure the taskData.json to predefine the tasks for the experiment

The question (e.g. “What is Judges Name”)

Help information (“Free text of some kind with further hints”)

The type of response we are expecting (e.g. Free text, multiple values, choose from list)

The search function - which takes the document as a string, returns locations to highlight.

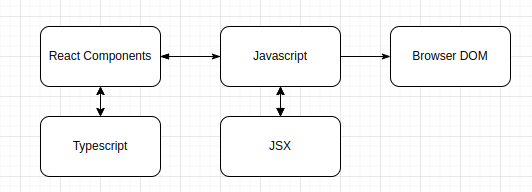
Validation rules - function that checks answer to the response and see if it complies, if not warns the user

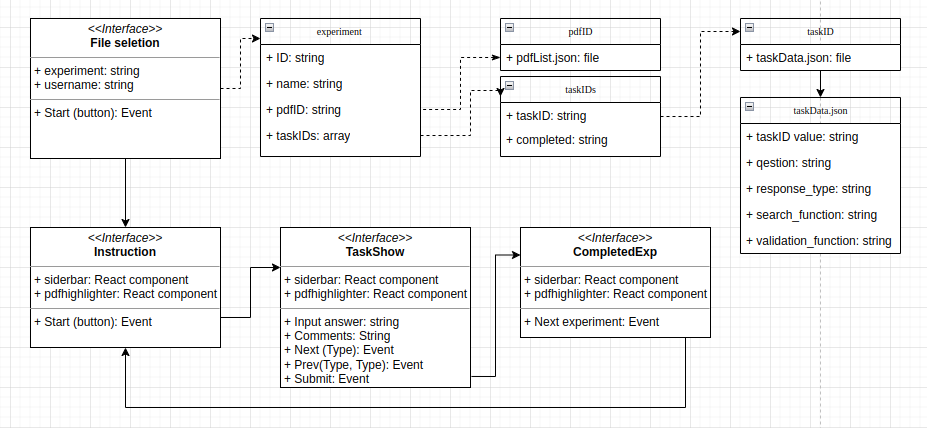
* + - * + Configure the experiment.json to preload the experiment for the user

Experiment’s IDs, usernames, PDF’s IDs and task IDs

* + - * + Pregenerate highlighted keywords into a transformArrayAll.json file
      * Run experiment
        + Start the application by entering npm start
        + Select the experiment file and enter the user name
        + Load the instruction once the experiment starts
        + Start the tasks once the user is ready
        + Repeat the experiment with the different PDF files stored in experiment.json files once the user completes the previous experiment.
        + All the log files are generated during the experiment

### Software architecture document

* + - * Architecture Design
      * Conceptual Design



### Source code

* + - * <https://github.com/davidtw999/LegalProofReadHelper>

### Maintenance and help guide

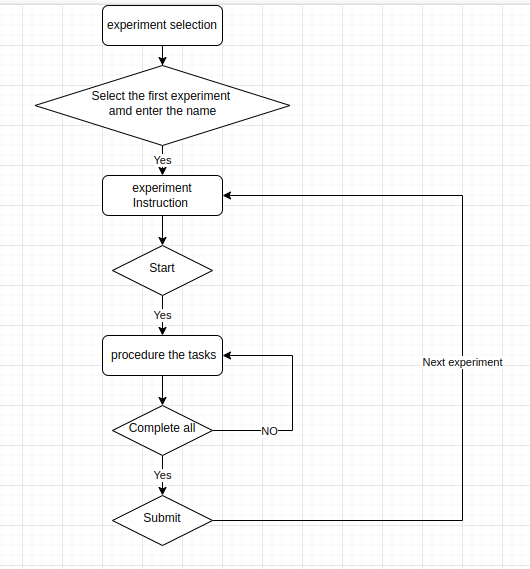
* + - * Dependencies
        + "dependencies": {
        + "@emotion/react": "^11.9.0",
        + "@emotion/styled": "^11.8.1",
        + "@material-ui/core": "^4.12.3",
        + "@mui/icons-material": "^5.6.0",
        + "@mui/material": "^5.6.0",
        + "@types/lodash.debounce": "^4.0.6",
        + "@types/pdfjs-dist": "^2.7.4",
        + "@types/react": "^16.4.0",
        + "@types/react-dom": "^16.4.0",
        + "dom-to-json": "^0.0.4",
        + "domjson": "^0.1.2",
        + "eslint-import-resolver-typescript": "^2.5.0",
        + "express-promise-router": "^4.1.1",
        + "lodash.debounce": "^4.0.8",
        + "pdf-lib": "^1.17.0",
        + "pdfjs-dist": "2.8.335",
        + "react-csv": "^2.2.2",
        + "react-rnd": "^10.1.10",
        + "react-router-dom": "^6.2.1",
        + "state-pool": "^0.7.1",
        + "uuidv4": "^6.2.12"
        + "devDependencies": {
        + "@types/react-csv": "^1.1.2",
        + "@vitejs/plugin-react-refresh": "^1.3.1",
        + "jest": "^27.1.0",
        + "jest-puppeteer": "^5.0.4",
        + "prettier": "^2.3.2",
        + "puppeteer": "^9.1.1",
        + "typescript": "4.4.2",
        + "vite": "^2.5.1"
        + }

## User documentation

User documentation refers to manuals that are generally written for product end users and system administrators. Tutorials, user guides, troubleshooting manuals, installation instructions, and reference manuals are all examples of user documentation.

### End User

* + - * The flowchart below shows how the application is used for the end user in the experiment.



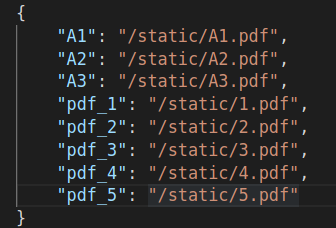
### System Administrator

* + - * Download application
        + git clone -b interface\_v3ser --single-branch https://github.com/davidtw999/LegalProofReadHelper
      * Install application
        + npm install
      * Run application
        + Configure json files for data preparation

Edit pdfList.json

Download the legal PDF files and saved under example/static path

Edit the key and values for the file name and correct path



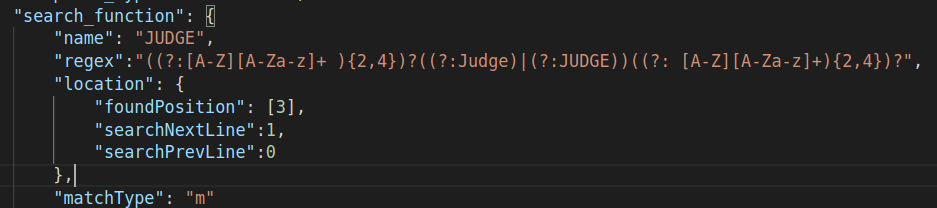
Edit taskData.json

Define the task question

Define the hint message

Define the response type

Define the search function



name is used to show the title of the input box for the task 1

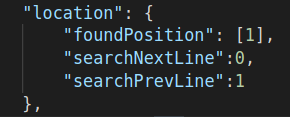


regex is used to search for keywords. It is combined with three parts: beginning, middle and end shown below. Admin can change the specified keywords in the middle part for the specified task, and keep the same regex of begin and end parts to capture the names. 

((?:[A-Z][A-Za-z]+ ){2,4})

Indicates the keywords captured can only be two to four words of the uppercase or lowercase letters. Admin can modify any regex pattern inside the brackets for a particular task.  
More information of regex can be found in <https://www.regular-expressions.info/>

location is used to search for the specific position of the matched keywords.



The value of founcPosition can only be 1,2,3. 1 indicates the beginning part of regex, 2 indicates the middle parts, and 3 indicates the end part.

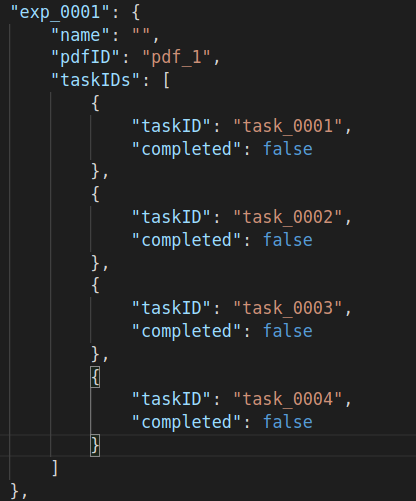
The value of searchNextLine and searchPrevLine can be 0,1,2,3. 0 indicates only searching for the current line. 1 indicates searching for one line before or after the current line, 2 indicates searching for two lines before or after the current line. 3 indicates searching for the three lines before or after the current line.

matchType is used for the regex function. The letter “m” indicates Multi-line search. The letter “i” indicates Case-insensitive search.  
More information can be found <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions>

Define validation\_function

Warning is used to help the user when the answer is blank.

Edit experiment.json



Experiment contains the experiment ID and its user and task information. Name indicates the user’s name. The default value is empty. pdfID indicates the specific PDF file. taskIDS contains all the tasks related to the tasks from taskData.json files. All the experiments can be set up inside this array.

* + - * + Highlighted hint json files <expTaskArray.json, transformArrayAll.json> by using dataPreproces.tsx

enable the line   
import "./dataPreproces.tsx";   
in app.tsx.

run the application by entering “npm start” to generate two json files as mentioned above.

move the json files into example/static/data path

* + - * + Run the application

Disable the line   
import "./dataPreproces.tsx";   
in app.tsx.

Enable the line  
import { testHighlights, userRecords } from "./dataDetect";

in app.tsx

Run the command “npm start”

Open the localhost line showing in the terminal



# Process documentation

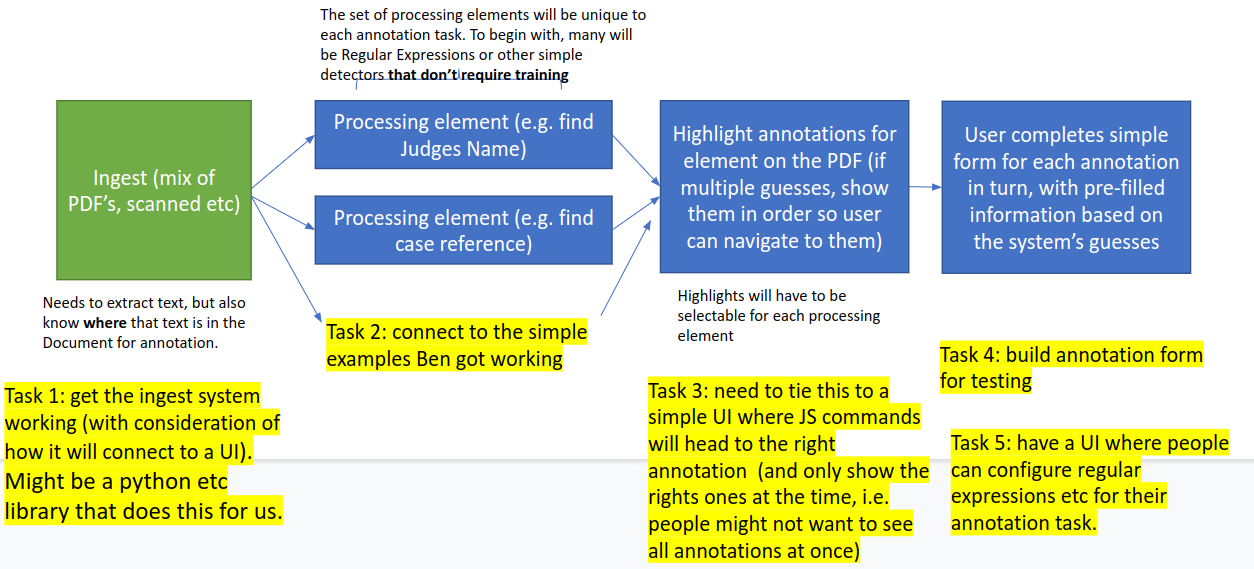
The process documentation covers all activities related to the product development process. Documentation takes considerable preparation and paperwork both before and during the development phase of the project.

## Plans and estimates

For simple annotations, where we have a system that can identify and highlight relevant information. The development takes three stages in the following order.

### Stage 1: Simple annotations

* + - * Find in document (e.g. Judges Name)
      * Highlight suggestions
      * User Yes / Edit’s them (if they can be found)



### Stage 2: Somewhat more challenging cases

* + - * Find all sections of the Freedom of Information Act (2000) cited?
        + Likewise with CaseLaw / Authorities.
      * More challenging because we don’t know how many we expect to find (goal is to find all of them).

### Stage 3: More complex annotations

* + - * Subjective decision.
      * Find relevant parts and highlight them, to save reading time.
        + Perhaps also connected to the ‘somewhat more challenging cases’, as automation for this could be the basis of the decision (e.g. find all legal discussion / analysis).

## Timeline schedules

| Stage 1: Simple annotations | | | |
| --- | --- | --- | --- |
| Task title | Start Date | Duration | Task complete |
| Task 1: Application structure design on react and javascript | 25/10/2021 | 10 days | 100% |
| Task 2:  Application files implementation with legal statements of pdf files | 05/11/2021 | 30 days | 100% |
| Task 3:  Application functionality implementation such as read and highlight on the pdf files | 05/12/2021 | 45 days | 100% |
| Task 4  Application algorithm implementation such as annotation on the keywords generated by the tasks, search | 20/01/2021 | 45 days | 100% |
| Task 5  Finalise the prototype of the annotation system | 05/03/2021 | 60 days | 100% |

## Standards

### Interactions

* + - * Some tasks will be open-ended:
        + E.g. Case reference.
      * Others might use closed sets:
        + E.g. Judges name – so will alert if they are using a similar one.
        + Can use initial input to enhance performance.
      * Cases where we are looking for all instances of section numbers will still require reading the document:
        + But an autofill will really help, because its just a matter of highlighting any that were missed / deleting any accidentally added ones!
      * UI should also indicate/warn if a TextLayer is OCR based:
        + Expect lower performance in such cases, due to OCR inaccuracy.

### Experiment

* + - * Quicker + more accurate? That’s the goal of the process.
      * Get someone to do a manual ground truth annotation and time it.
        + As well as work out mistakes (get 3 people to do this and compare!)
      * Then test using the system, different system, same people, different data.
      * Need to also artificially modify the Precision and Recall etc of annotations:
        + Do people do worse when there is missing data?
        + What about a positively wrong recommendation, how often will it be added by mistake?
        + How does this vary with different performances?

# 

# 

# APPENDIX

## References

### Legal document links

* <https://informationrights.decisions.tribunals.gov.uk/>
* <https://github.com/benphua/AdministrativeTribunal-SemiAutomation-Tools/tree/main/Datasets/Decision%20Report%20PDFs>

### Annotations github links

* <https://github.com/klassif-ai/react-pdf-ner-annotator>
* <https://react-annotator-demo.netlify.app/>
* <https://github.com/benphua/AdministrativeTribunal-SemiAutomation-Tools/tree/main/Datasets/Decision%20Report%20PDFs>
* <https://github.com/agentcooper/react-pdf-highlighter>
* <https://github.com/AboutGoods/Banksy-annotation-tool>
* <https://github.com/deepwel/Chinese-Annotator>
* <https://github.com/paperai/pdfanno>
* <https://github.com/tecoholic/ner-annotator>
* <https://github.com/doccano/doccano>
* <https://github.com/heartexlabs/label-studio#try-out-label-studio>
* <https://github.com/taivop/awesome-data-annotation>