MONASH University

MALAYSIA

FIT3162
Test Report

Project Title: Classification of Retrieved Documents for User Satisfaction

Team MA_A_2

Team Members:
Lim Jun Qing, 30029937
Alfons Fernaldy, 30127831
Chang Yin Cheng, 29792029

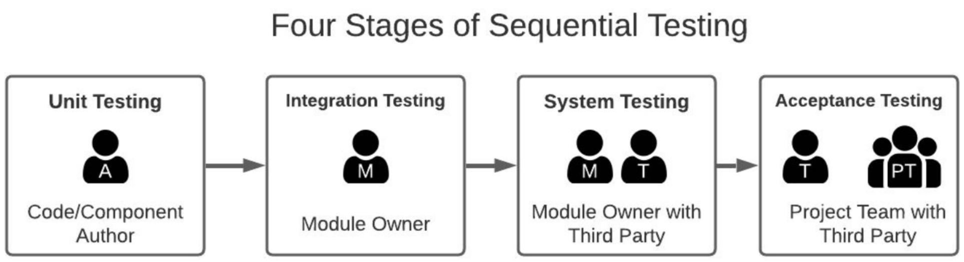
Supervisor: Dr. Prabha Rajagopal

Contents

1 Introduction

This project aims to build a classification model that classifies the readability level of a document provided by the user. Software testing serves as the measurement of software's robustness, scalability and limitations.

The objective of the following tests conducted are to verify the functionality of the program to match the requirements of the project scope. We conduct our testing in the following phases.



Each test is manually performed by the tester, with documentation described in the sections below. Note that Usability testing is equivalent to Acceptance Testing for the purposes of this report.

2 Testing Plan / Setup

The minimum requirements here are applied to any computer performing pre-processing, processing and evaluation. The training of classification models is done separately via Google Collab, thus it is separated from the actual end product testing.

Disclaimer: The following testing setup is tested and verified on Unix-based OS (MacOS/Ubuntu platform).

2.1 Hardware Requirements

Hardware	Specification
Hard Drive / Solid State	500GB of HDD / 128GB of SSD
CPU	Intel i5 5th-Generation and above
RAM	8 GB

2.2 Software Requirements

The software specified here is required to be installed before setting up the testing environment.

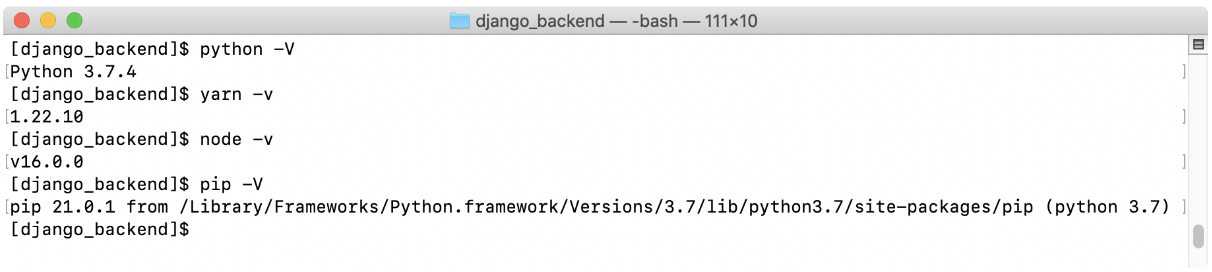
1. Python
   Recommended version: `3.7.*`

2. Yarn
   Recommended version: `1.22.*`

3. Node.JS
   Recommended version: `v16.*.*`

4. Pip
   Recommended version: `21.*.*`

A screenshot is shown below for illustration purposes.



```
[django_backend]$ python -V
Python 3.7.4
[django_backend]$ yarn -v
1.22.10
[django_backend]$ node -v
v16.0.0
[django_backend]$ pip -V
pip 21.0.1 from /Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/pip (python 3.7)
[django_backend]$
```

2.3 Testing Setup

The steps here are undertaken to set up both the frontend, backend and connection to classification model.

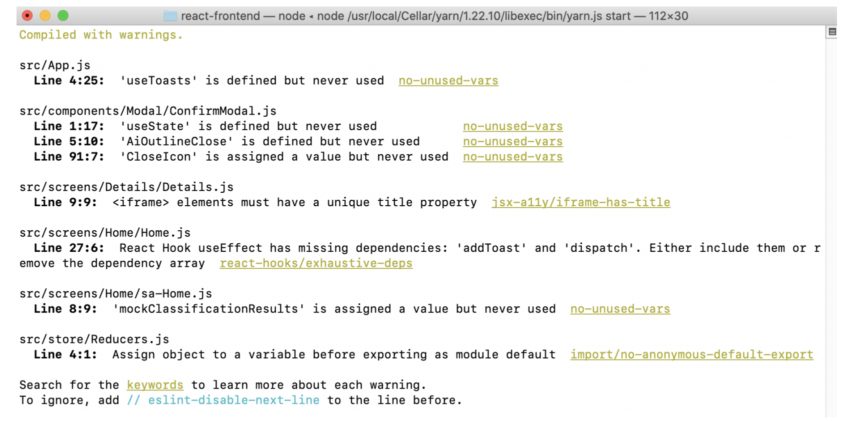
2.3.1 Frontend

1. Pull the frontend source code repository from github.
   `git pull https://github.com/FYP-2020-2021/react-frontend.git`

2. Enter the react_frontend directory.
   `cd react_frontend/`

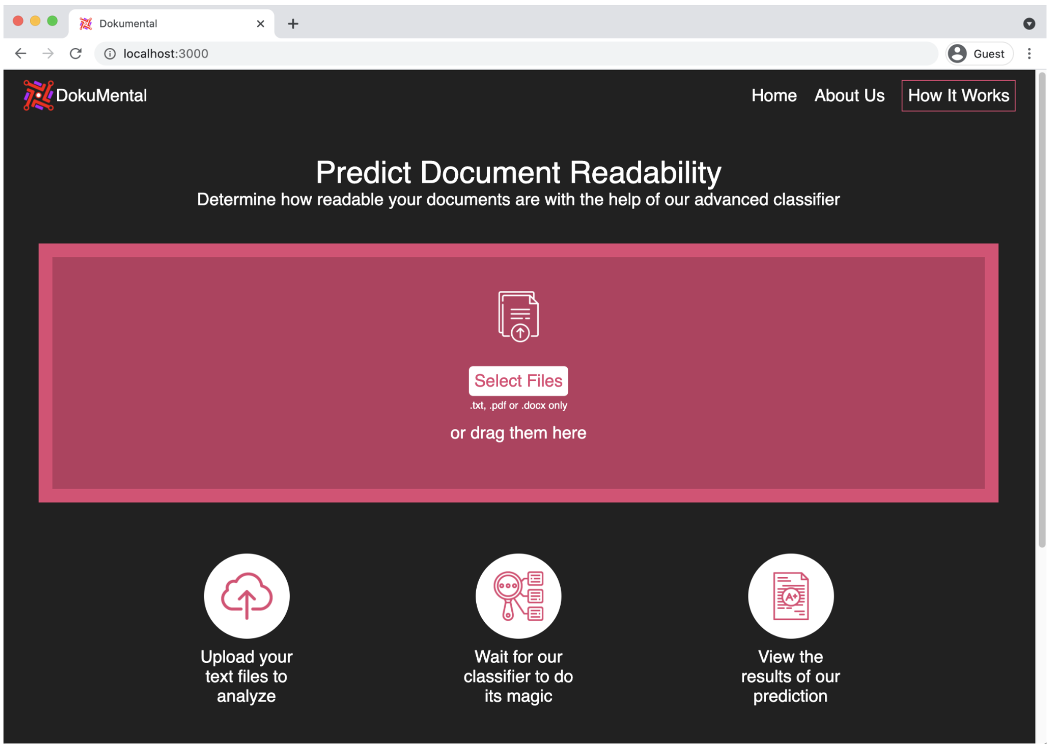
3. Install required dependencies.
   `yarn`

4. Launch the frontend web.
   `yarn start`



```
react-frontend — node ◂ node /usr/local/Cellar/yarn/1.22.10/libexec/bin/yarn.js start — 112×30
Compiled with warnings.

src/App.js
  Line 4:25:  'useToasts' is defined but never used  no-unused-vars

src/components/Modal/ConfirmModal.js
  Line 1:17:  'useState' is defined but never used           no-unused-vars
  Line 5:10:  'AiOutlineClose' is defined but never used      no-unused-vars
  Line 91:7:  'CloseIcon' is assigned a value but never used  no-unused-vars

src/screens/Details/Details.js
  Line 9:9:  <iframe> elements must have a unique title property  jsx-a11y/iframe-has-title

src/screens/Home/Home.js
  Line 27:6:  React Hook useEffect has missing dependencies: 'addToast' and 'dispatch'. Either include them or r
emove the dependency array  react-hooks/exhaustive-deps

src/screens/Home/sa-Home.js
  Line 8:9:  'mockClassificationResults' is assigned a value but never used  no-unused-vars

src/store/Reducers.js
  Line 4:1:  Assign object to a variable before exporting as module default  import/no-anonymous-default-export

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

This launches the web app on localhost:3000



Predict Document Readability

Determine how readable your documents are with the help of our advanced classifier

or drag them here

Upload your text files to analyze

Wait for our classifier to do its magic

View the results of our prediction

2.3.2 Backend

1. Pull the backend source code repository from github.
   `git pull https://github.com/FYP-2020-2021/django_backend.git`

2. Enter the django_backend directory.
   `cd django_backend/`

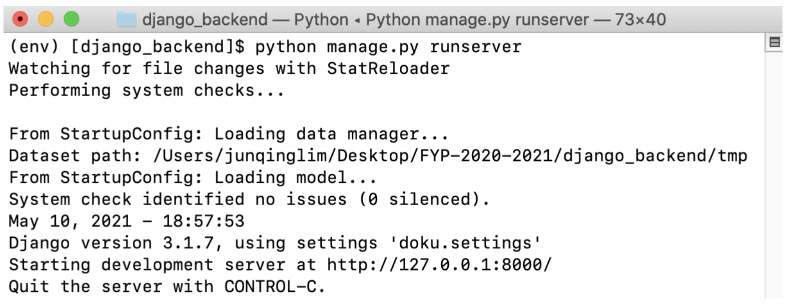
3. Create the virtual environment manually.
   `python3 -m venv venv/`

4. Activate the virtual environment.
   `source venv/bin/activate`

5. Install the required dependencies.
   `pip install -r requirements.txt`

   **Note: If there are dependencies conflicts, users may require to manually resolve them*

6. Launch the server.
   `python manage.py runserver`


django_backend — Python ◂ Python manage.py runserver — 73×40

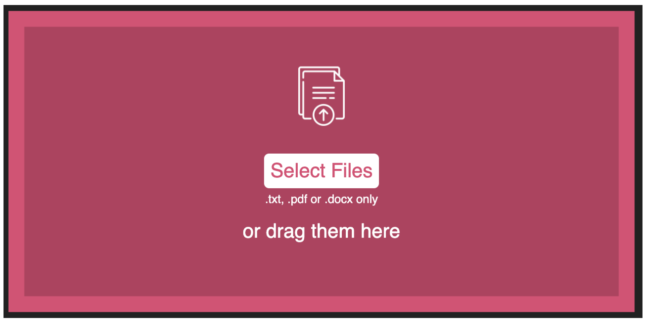
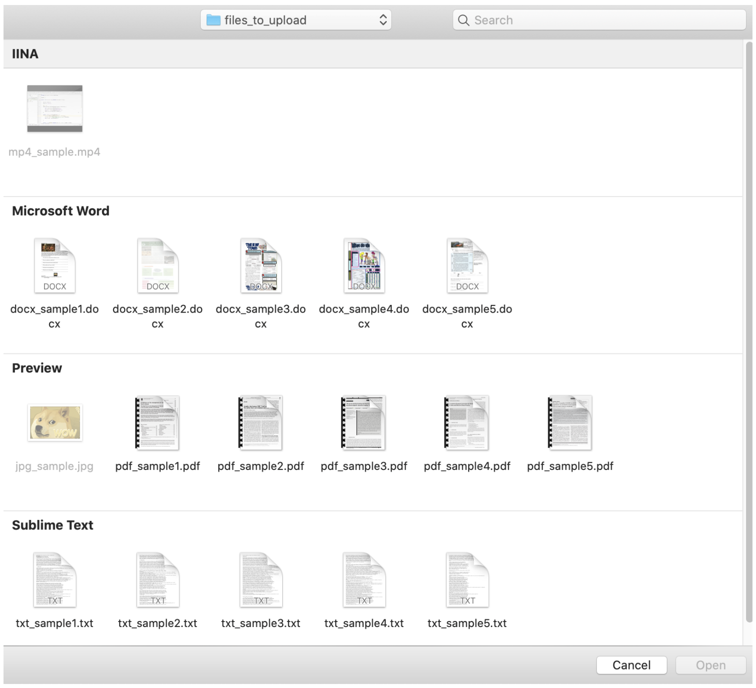
```
(env) [django_backend]$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

From StartupConfig: Loading data manager...
Dataset path: /Users/junqinglim/Desktop/FYP-2020-2021/django_backend/tmp
From StartupConfig: Loading model...
System check identified no issues (0 silenced).
May 10, 2021 - 18:57:53
Django version 3.1.7, using settings 'doku.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

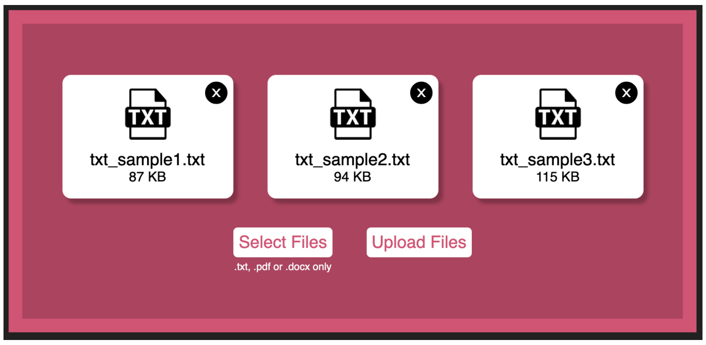
**Note: The server automatically starts up the connection (loading) of the classification model, this is to reduce the time taken required to run the model in obtaining the classification results.*

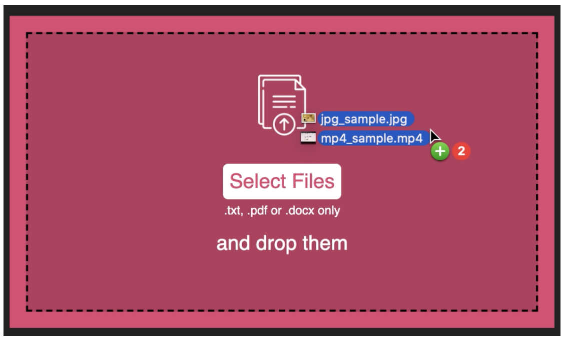
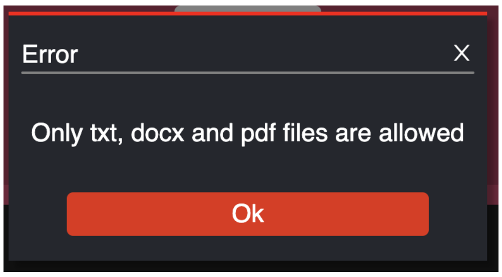
3 Unit Testing (Whitebox Testing)

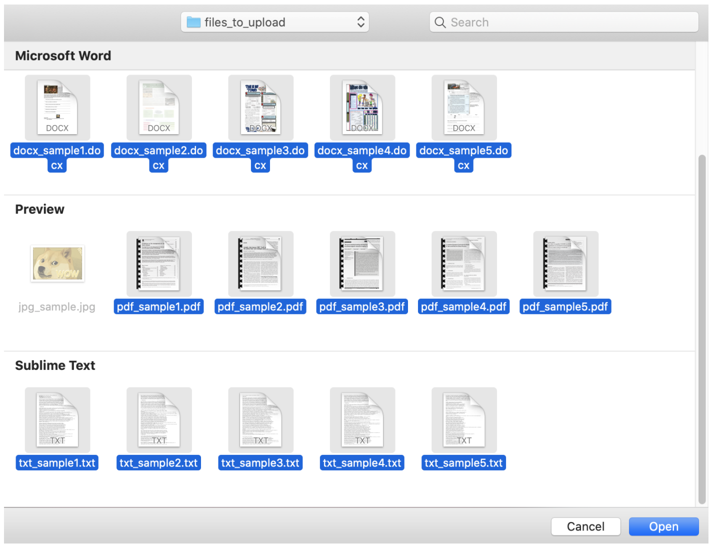
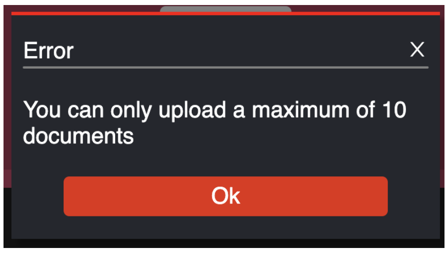
These tests are the first level of testing and are performed by the module owners themselves. In this stage, testing is focused on specific components of the system to ensure it is fully functional and behaves as expected.

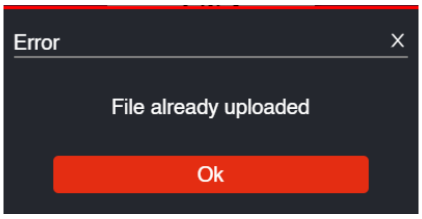
3.1 Frontend

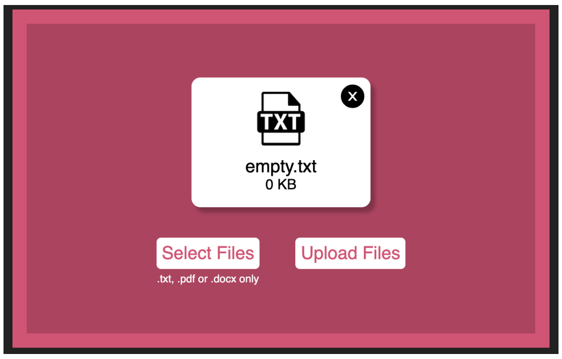
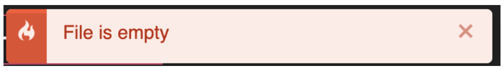
ID	01
Description	Users can select file(s) using the system file picker or drag/drop into the file upload area.
Steps	1. Click on "Select Files"  Select Files .txt, .pdf or .docx only or drag them here 2. Select one or more files.  files_to_upload Search IINA mp4_sample.mp4 Microsoft Word docx_sample1.docx docx_sample2.docx docx_sample3.docx docx_sample4.docx docx_sample5.docx Preview jpg_sample.jpg pdf_sample1.pdf pdf_sample2.pdf pdf_sample3.pdf pdf_sample4.pdf pdf_sample5.pdf Sublime Text txt_sample1.txt txt_sample2.txt txt_sample3.txt txt_sample4.txt txt_sample5.txt Cancel Open
Expected Outcome	1. File(s) are selected 2. Unsupported files (non pdf, docx, txt) cannot be selected

Actual Outcome	File(s) are populated into the file upload area with the name of selected file and its file size, indicating the file(s) were selected.  TXT txt_sample1.txt 87 KB TXT txt_sample2.txt 94 KB TXT txt_sample3.txt 115 KB Select Files .txt, .pdf or .docx only Upload Files
Status	PASS

ID	02
Description	User cannot upload unsupported file formats with drag/drop
Steps	1. Drag unsupported file(s) into file upload area  jpg_sample.jpg mp4_sample.mp4 2 Select Files .txt, .pdf or .docx only and drop them
Expected Outcome	File(s) are restricted to be uploaded.
Actual Outcome	User is notified with an unsupported file message.  Error Only txt, docx and pdf files are allowed
Status	PASS

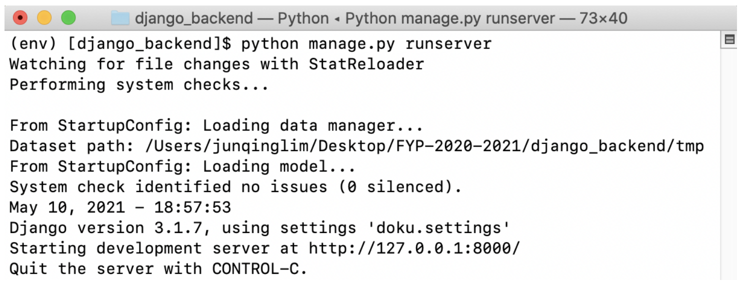
ID	03
Description	User can only upload 1 to 10 documents
Steps	1. Select more than 10 documents to upload  files_to_upload; Search; Microsoft Word: docx_sample1.docx, docx_sample2.docx, docx_sample3.docx, docx_sample4.docx, docx_sample5.docx; Preview: jpg_sample.jpg, pdf_sample1.pdf, pdf_sample2.pdf, pdf_sample3.pdf, pdf_sample4.pdf, pdf_sample5.pdf; Sublime Text: txt_sample1.txt, txt_sample2.txt, txt_sample3.txt, txt_sample4.txt, txt_sample5.txt; Cancel; Open
Expected Outcome	Users cannot upload more than 10 documents.
Actual Outcome	Users are notified with an error message on upload limit.  Error You can only upload a maximum of 10 documents Ok
Status	PASS

ID	04
Description	User cannot upload the same document again
Steps	1. Select and upload any document 2. Select and upload the same document again
Expected Outcome	Users cannot upload the same document again and will see an error message informing them.
Actual Outcome	Users are notified with an error message on the same document uploads.  Error File already uploaded
Status	PASS

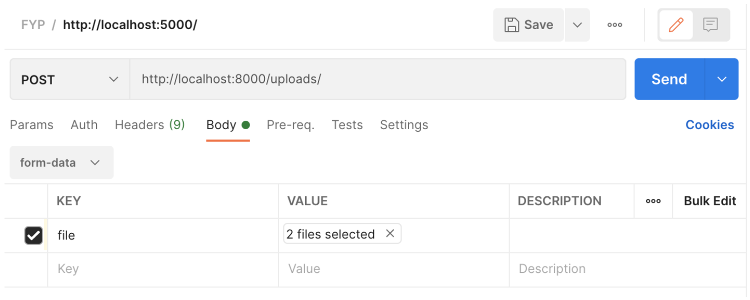
ID	05
Description	User cannot upload empty file
Steps	1. Select and upload empty document  TXT empty.txt 0 KB Select Files .txt, .pdf or .docx only Upload Files
Expected Outcome	Users fail to upload and process the empty file.
Actual Outcome	Users are notified with an empty file message.  File is empty
Status	PASS

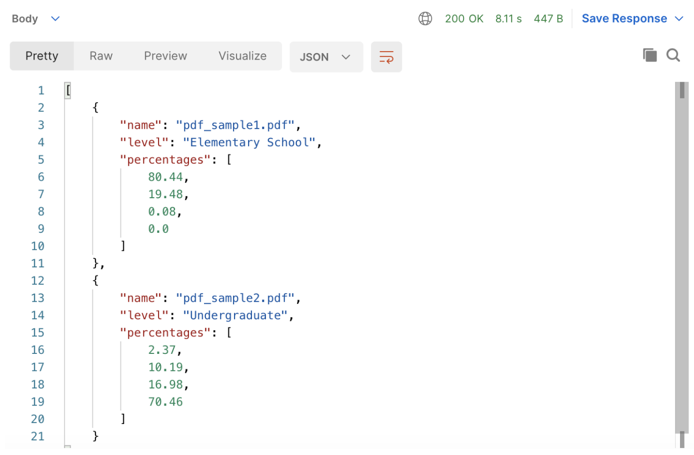
3.2 Backend

ID	06
Description	Server accesses and loads the classification model upon startup

Steps	1. Run `python manage.py runserver` to launch the server
	`django_backend — Python ◂ Python manage.py runserver — 73×40`
	`(env) [django_backend]$ python manage.py runserver`
	`Watching for file changes with StatReloader`
	`Performing system checks...`
	
	`From StartupConfig: Loading data manager...`
	`Dataset path: /Users/junqinglim/Desktop/FYP-2020-2021/django_backend/tmp`
	`From StartupConfig: Loading model...`
	`System check identified no issues (0 silenced).`
	`May 10, 2021 - 18:57:53`
	`Django version 3.1.7, using settings 'doku.settings'`
	`Starting development server at http://127.0.0.1:8000/`
	`Quit the server with CONTROL-C.`

Expected Outcome	Classification model is loaded.
Actual Outcome	Classification model is accessed and loaded into the server.
Status	PASS

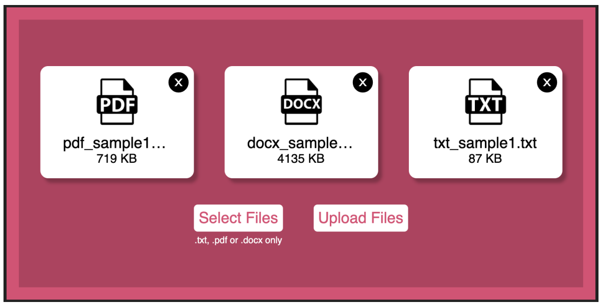
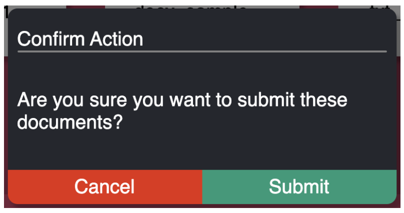
ID	07
Description	Upload files to the server to be classified by the classification model
Steps	1. Launch Postman 2. Upload and select files as the POST body FYP / http://localhost:5000/ POST http://localhost:8000/uploads/ Send Params · Auth · Headers (9) · Body ● · Pre-req. · Tests · Settings · Cookies form-data KEY: file · VALUE: 2 files selected · DESCRIPTION 3. Click "Send"
Expected Outcome	Server returns the classification model results for the file(s) uploaded.
Actual Outcome	Classification model results are returned as JSON.

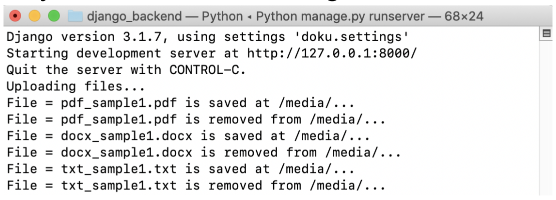
	 Body
	200 OK 8.11 s 447 B
	Save Response
	Pretty
	Raw
	Preview
	Visualize
	JSON
	```
	[
	    {
	        "name": "pdf_sample1.pdf",
	        "level": "Elementary School",
	        "percentages": [
	            80.44,
	            19.48,
	            0.08,
	            0.0
	        ]
	    },
	    {
	        "name": "pdf_sample2.pdf",
	        "level": "Undergraduate",
	        "percentages": [
	            2.37,
	            10.19,
	            16.98,
	            70.46
	        ]
	    }
	```

Status	PASS

4 Integration Testing (Blackbox Testing)

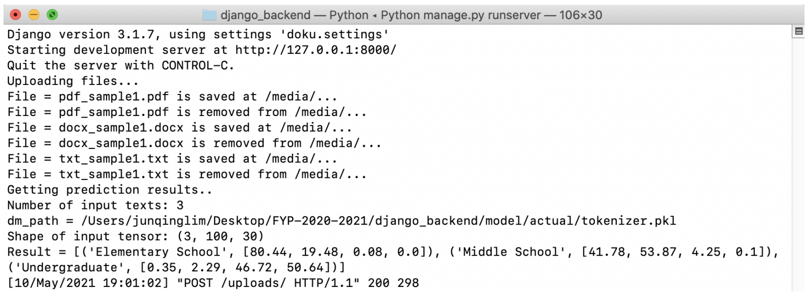
These tests are conducted where smaller modules (components) are combined into a single, larger module to be tested as a whole.

ID	08
Description	Files are uploaded to the server with server receiving and preprocessing it
Steps	*(Continue from <u>ID01</u>)* 1. Select “Upload Files”  PDF pdf_sample1… 719 KB DOCX docx_sample… 4135 KB TXT txt_sample1.txt 87 KB Select Files .txt, .pdf or .docx only Upload Files 2. Select “Submit” Confirm Action Are you sure you want to submit these documents?
Expected Outcome	File(s) uploaded are read by the server.

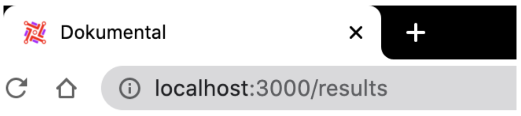
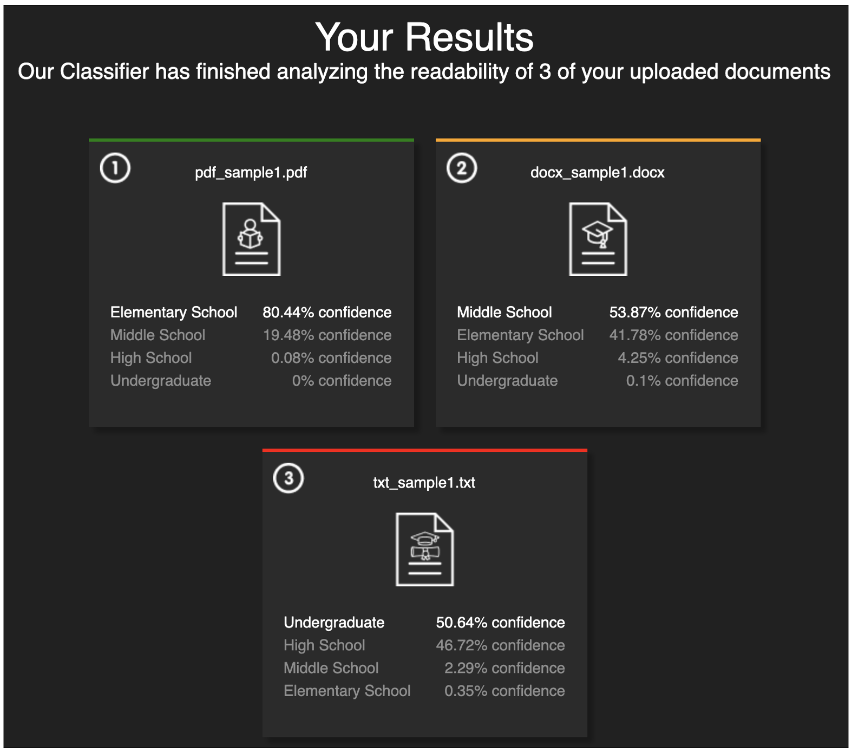
Actual Outcome	File(s) are saved by the server for reading and deleted after reading.
	 django_backend — Python ◂ Python manage.py runserver — 68×24
	`Django version 3.1.7, using settings 'doku.settings'`
	`Starting development server at http://127.0.0.1:8000/`
	`Quit the server with CONTROL-C.`
	`Uploading files...`
	`File = pdf_sample1.pdf is saved at /media/...`
	`File = pdf_sample1.pdf is removed from /media/...`
	`File = docx_sample1.docx is saved at /media/...`
	`File = docx_sample1.docx is removed from /media/...`
	`File = txt_sample1.txt is saved at /media/...`
	`File = txt_sample1.txt is removed from /media/...`

Status	PASS

ID	09
Description	Server obtains the classification results of the documents uploaded
Steps	*(Continue from <u>ID06</u>)*
Expected Outcome	Server inputs the documents uploaded into the classification model to obtain the classification results

Actual Outcome	Classification results are obtained after input into the model. 
Status	PASS

```
django_backend — Python ◂ Python manage.py runserver — 106×30
Django version 3.1.7, using settings 'doku.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
Uploading files...
File = pdf_sample1.pdf is saved at /media/...
File = pdf_sample1.pdf is removed from /media/...
File = docx_sample1.docx is saved at /media/...
File = docx_sample1.docx is removed from /media/...
File = txt_sample1.txt is saved at /media/...
File = txt_sample1.txt is removed from /media/...
Getting prediction results..
Number of input texts: 3
dm_path = /Users/junqinglim/Desktop/FYP-2020-2021/django_backend/model/actual/tokenizer.pkl
Shape of input tensor: (3, 100, 30)
Result = [('Elementary School', [80.44, 19.48, 0.08, 0.0]), ('Middle School', [41.78, 53.87, 4.25, 0.1]),
('Undergraduate', [0.35, 2.29, 46.72, 50.64])]
[10/May/2021 19:01:02] "POST /uploads/ HTTP/1.1" 200 298
```

ID	10
Description	Frontend displays the classification results
Steps	*(Continue from ID07)*
Expected Outcome	Frontend receives the classification results for the documents uploaded and redirects to a results page.
Actual Outcome	Frontend directs the user to a result page, displaying the results of the documents classification.  
Status	PASS

Your Results

Our Classifier has finished analyzing the readability of 3 of your uploaded documents

1 pdf_sample1.pdf

Elementary School	80.44% confidence
Middle School	19.48% confidence
High School	0.08% confidence
Undergraduate	0% confidence

2 docx_sample1.docx

Middle School	53.87% confidence
Elementary School	41.78% confidence
High School	4.25% confidence
Undergraduate	0.1% confidence

3 txt_sample1.txt

Undergraduate	50.64% confidence
High School	46.72% confidence
Middle School	2.29% confidence
Elementary School	0.35% confidence

5 System Testing (Requirements Satisfaction)

In this testing, we test the system as a whole, that is - an application-wide testing where both functional and non-functional requirements are tested for verification. The following requirements are extracted from the initial project proposal to verify if the initial requirements set were satisfied.

5.1 Functional Requirements

ST ID	Description	Expected Outcome	Actual Outcome	Status
01	The web application should be able to handle user document uploads in the form of txt, docx or pdf file types.	Erroneous error message for non-valid file types	From ID02, an error message is used to notify user on unsupported files	PASS
02	The web application is capable of communicating and sending documents to the server.	Files are sent across the server	From ID08, server receives and preprocesses the files	PASS
03	The backend server should be able to host the classification model and communicate with the web application.	Connection to classification model is set whenever the server starts	From ID06, server loads the model upon startup	PASS
04	The classification model is able to preprocess text files containing english words.	A set of english words are used to preprocess to prepare the model	From ID06, data_manager is loaded to handle text files preprocessing	PASS
05	The classification model is able to categorize these files into one of several predetermined readability categories.	Readability level of each document is obtained	From ID09, server obtains classification results of different readability levels	PASS
06	The web application should display assigned readability labels to each document uploaded by the user alongside any other relevant information provided by the model.	Readability level is displayed on UI with extra information	From ID10, web application displays the results to each document, along with its percentages	PASS

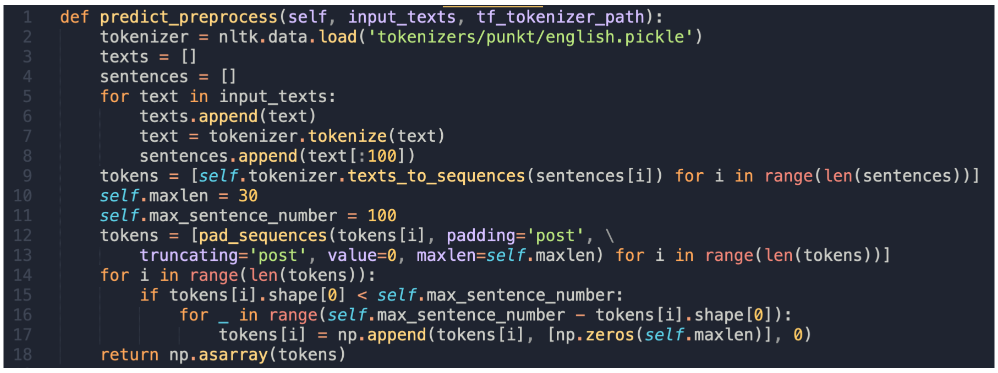
5.2 Non-Functional Requirements

ST ID	Description	Expected Outcome	Actual Outcome	Status
07	The web application should explain in-depth what each label means such that the user not only learns more about the document's readability but also of their own.	Results should indicate each document readability's label	The web application displays the labels of each document submitted on the results card allowing users to understand the document's readability.	PASS
08	The web application should provide a page which explains how the classification model works in depth.	A web page should exist which contains an in depth explanation of the classification model	A web page was created which contains an embedded youtube video explaining how our classification model works.	PASS
09	The UI design should strictly follow well established design principles to ensure the best user experience.	Design follows either Material design, Ant Design, Materialize.	UI Design follows Material design for web applications.	PASS
10	The time it takes to send, process and receive documents should be relatively short.	About 10 seconds to process a file of at most 100MB	Approximately 12 seconds to process 100MB file	PASS

6 Performance, Scalability, Security

6.1 Performance

Model may not be accurate with a mixed level of writing skills.
The model is trained based on the first 100 sentences of every document in our dataset. There may be some words that are unrecognized by the model depending on the variability of the vocabulary used in our dataset. Thus the model may not perform well on some texts which have mixed levels of writing skills. For instance, a document with primary school text followed by a research text in a single document will result in lower accuracy as initial sentences have higher precedence during classification. The code diagram below illustrates the truncation of sentences (line 11).



```python
1   def predict_preprocess(self, input_texts, tf_tokenizer_path):
2       tokenizer = nltk.data.load('tokenizers/punkt/english.pickle')
3       texts = []
4       sentences = []
5       for text in input_texts:
6           texts.append(text)
7           text = tokenizer.tokenize(text)
8           sentences.append(text[:100])
9       tokens = [self.tokenizer.texts_to_sequences(sentences[i]) for i in range(len(sentences))]
10      self.maxlen = 30
11      self.max_sentence_number = 100
12      tokens = [pad_sequences(tokens[i], padding='post', \
13          truncating='post', value=0, maxlen=self.maxlen) for i in range(len(tokens))]
14      for i in range(len(tokens)):
15          if tokens[i].shape[0] < self.max_sentence_number:
16              for _ in range(self.max_sentence_number - tokens[i].shape[0]):
17                  tokens[i] = np.append(tokens[i], [np.zeros(self.maxlen)], 0)
18      return np.asarray(tokens)
```

(Figure 6.1: Code on documents preprocessing)

Even as the model takes in only the first 100 sentences, any sentence that has more than 30 words will have its subsequent sentences truncated (line 10 above). This is to reduce the overall time taken to process the model. Thus the model may not be suitable for the classification of large documents.

Large files require longer processing time

File Size	Time Taken / Result

100KB	`Uploading files...`
	`File = j.1432-2277.2008.00668.x.pdf is saved at /media/...`
	`File = j.1432-2277.2008.00668.x.pdf is removed from /media/...`
	`Getting prediction results..`
	`Number of input texts: 1`
	`dm_path = /Users/junqinglim/Desktop/FYP-2020-2021/django_backend/model/actual/tokenizer.pkl`
	`Shape of input tensor: (1, 100, 30)`
	`Result = [('Undergraduate', [4.85, 20.02, 20.45, 54.67])]`
	`Total time to process = 0.8634`
	`[10/May/2021 20:08:46] "POST /uploads/ HTTP/1.1" 200 112`

1MB	`Uploading files...`
	`File = idh.12452.pdf is saved at /media/...`
	`File = idh.12452.pdf is removed from /media/...`
	`Getting prediction results..`
	`Number of input texts: 1`
	`dm_path = /Users/junqinglim/Desktop/FYP-2020-2021/django_backend/model/actual/tokenizer.pkl`
	`Shape of input tensor: (1, 100, 30)`
	`Result = [('Middle School', [30.99, 67.95, 0.45, 0.61])]`
	`Total time to process = 0.8718`
	`[10/May/2021 20:08:12] "POST /uploads/ HTTP/1.1" 200 96`

10MB

```
Uploading files...
File = pone.0184059.pdf is saved at /media/...
File = pone.0184059.pdf is removed from /media/...
Getting prediction results..
Number of input texts: 1
dm_path = /Users/junqinglim/Desktop/FYP-2020-2021/django_backend/model/actual/tokenizer.pkl
Shape of input tensor: (1, 100, 30)
Result = [('Undergraduate', [1.14, 5.44, 35.31, 58.11])]
Total time to process = 1.2224
[10/May/2021 20:06:07] "POST /uploads/ HTTP/1.1" 200 99
```

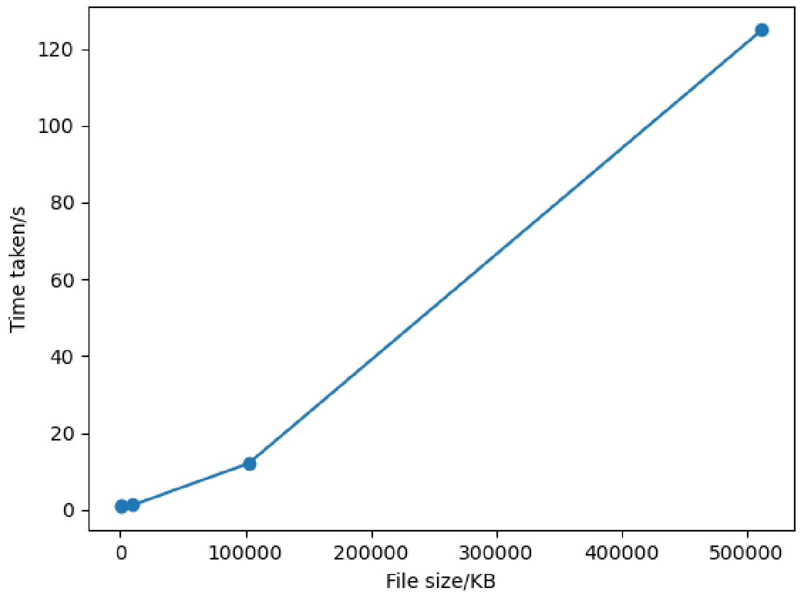
100MB

```
Uploading files...
File = 529-Article Text-2712-1-10-20130506.pdf is saved at /media/...
File = 529-Article Text-2712-1-10-20130506.pdf is removed from /media/...
Getting prediction results..
Number of input texts: 1
dm_path = /Users/junqinglim/Desktop/FYP-2020-2021/django_backend/model/actual/tokenizer.pkl
Shape of input tensor: (1, 100, 30)
Result = [('Undergraduate', [1.49, 7.4, 27.93, 63.18])]
Total time to process = 12.0179
[10/May/2021 20:05:32] "POST /uploads/ HTTP/1.1" 200 121
```

500MB

```
Uploading files...
File = pone.0184059 2021-05-11 at 4.10.44 AM 2 2.pdf is saved at /media/...
File = pone.0184059 2021-05-11 at 4.10.44 AM 2 2.pdf is removed from /media/...
Getting prediction results..
Number of input texts: 1
dm_path = /Users/junqinglim/Desktop/FYP-2020-2021/django_backend/model/actual/tokenizer.pkl
Shape of input tensor: (1, 100, 30)
Result = [('Undergraduate', [1.15, 5.45, 35.32, 58.09])]
Total time to process = 124.8335
[10/May/2021 20:15:59] "POST /uploads/ HTTP/1.1" 200 128
```

(Table 6.1: Processing time for each different file size)

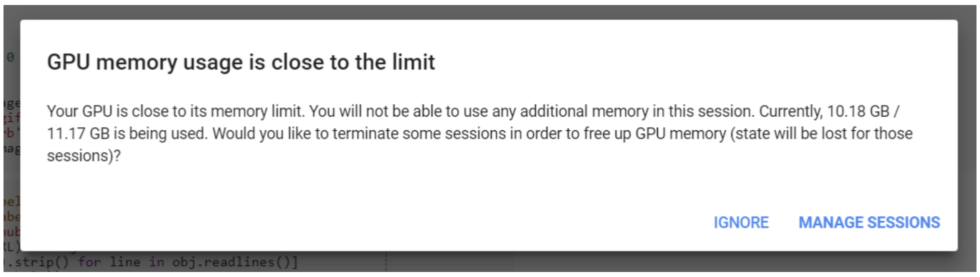


(Figure 6.2: A graph of time taken against file size)

A graph was plotted to illustrate the speed of processing inputs vs the input file sizes. It can be observed that the processing time is linearly proportional to the input file size. A linear slow-down in time is generally acceptable especially when processing text documents. Given the project scope, we are only concerned about the system's scalability when there is a significant increase in processing time following an increase in file size.

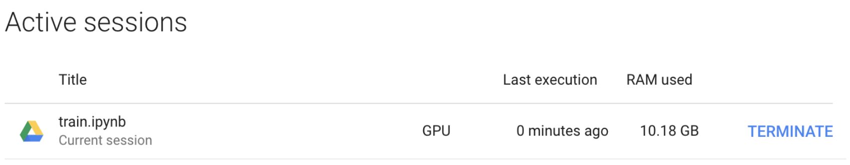
6.2 Scalability

Number of documents to be processed at once is restricted
The web interface restricts the user to only uploading 10 documents at once (as illustrated in ID03 test above). Such a restriction was developed upon the training of classification models on Google Collab where we encountered insufficient RAM issues shown below.


GPU memory usage is close to the limit

Your GPU is close to its memory limit. You will not be able to use any additional memory in this session. Currently, 10.18 GB / 11.17 GB is being used. Would you like to terminate some sessions in order to free up GPU memory (state will be lost for those sessions)?

(With GPU memory reached close to the limit provided)


Active sessions

Title		Last execution	RAM used	
train.ipynb Current session	GPU	0 minutes ago	10.18 GB	TERMINATE

(With a single active session of running that uses most resources)

As such, we have mitigated this issue by training the model with a limited total size of input text. This is why we have restricted the number of files that can be uploaded to prevent the overloading of the classification model. The model itself was trained using the default amount of resources available in Google Collab so it is not built to process large amounts of documents at once (100+ documents).

Documents to be processed by batch has lower turnaround time
To process the user documents that were uploaded, users can choose to upload a single file or multiple files at the same time. The classification model that is developed allows documents to be processed by "batches" (i.e., 1 batch that contains many files), hence resulting in shorter turnaround time for uploading multiple documents at once than uploading "document-by-document". The chart below shows the time taken measured for both approaches.

Upload Type	Filename	Time Taken to Classify (seconds)	Total Time Taken (seconds)
A. Uploading single file for 5 times	university_50.txt	0.3776	2.0684
	university_33.txt	0.3141	
	university_34.txt	0.3389	
	university_71.txt	0.3942	
	university_80.txt	0.6436	
B. Uploading 5 files all at once	university_50.txt university_33.txt university_34.txt university_71.txt university_80.txt	1.2661	1.2661

As can be observed above, the total time taken for type B is faster than type A. Hence, processing multiple files at the same time has a shorter and better turnaround time than processing individual and separate files. This in turn provides a better overall scalability performance as the number of files grows.

6.3 Security

Security risks do exist as our project's front end web application and backend server doesn't implement any advanced security measures. This is because the focus of our project is on developing a readability classifier and less so on developing a secure system architecture. If our project was hypothetically deployed for public use then the following scenarios could happen:

1. Attackers can upload malware injected into the documents
2. Attackers can flood the server with requests which would effectively shut down the service
3. Attackers can intercept packages sent from the front end client bypassing all the checks we have implemented on the front end

At the moment, only a basic security measurement is implemented such as limiting documents sent to 10 documents and only allowing documents as large as 500MB to be uploaded. This is to prevent regular users from overloading the backend server by uploading hundreds of documents or large documents.

Any future work or plans to release this project for widespread use will require additional security measures to be implemented particularly on the server side. This can range from user authentication, tracking and blacklisting suspicious IP addresses and captchas.

7 Usability Testing

Usability testing involves testing with a third party. Multiple users (colleagues from Monash) are invited to participate in the testing of our web application. The primary purpose of this test is to record feedback from the users upon interacting with the application. The team will then implement further changes to the system based on the feedback provided.

Participants were invited to a Zoom meeting where they are instructed to connect to our local machine via SSH protocol. Both server and the web application are started locally allowing the participants to interact with the system. Participants were not provided with any briefing to ensure a blind testing process where users begin the test without any prior knowledge about the application.

The time taken to successfully upload a document for each participant is recorded, alongside any errors that were encountered during the process. Feedback on the system and user interface were also documented.

Participant ID	**Time**	**Remarks**	**Errors**
01	1 mins 22 seconds	User tried to upload more documents than 10 and subsequently got the document limit error.	None
02	4 mins 48 seconds	User took a long time to find documents and was generally trying to break the system by uploading non supported file types and really large text files. Suffice to say these are handled by our system successfully.	None
03	1 mins 08 seconds	None	None
04	2 mins 04 seconds	User uploads different levels of documents ranging from primary level up to university level.	None

(the exact participants identities were kept anonymous for confidentiality)

Below are excerpts written or quoted from the participant's feedback:

Participant ID	Feedback
01	Good user interface, however the result screen can still be improved as it is pretty bland at the current stage.
02	System handled edge cases well when trying to upload variations of file types or oversized files.
03	*User did not write any feedback but did bring up concerns about the lack of authentication and security of the web application during testing*
04	Some documents were not accurately classified, such as university level documents being classified as high school level.

8 Possible Improvements

Improvements on model training

The existing model was trained using a dataset of English documents with a minor skew on readability distribution. One possible improvement that can be made is by feeding a dataset with better quality - such as a dataset with little to no skew with a larger variety of languages. This would let TensorFlow tokenize unicode characters such as emojis which may further improve the performance of the model for documents that contain a significant amount of non ASCII characters.

We aren't hoping to construct a model that can 100% accurately predict readability. In fact, during our proposal's literature review, we found out that the researchers who published the paper only managed to attain 77% accuracy. Categorizing readability itself is a very subjective task so the results of the model may not be very precise. This is why we had opted to only use 4 readability categories instead of more as doing so would mean categorizing documents into more precise readability levels.

Official deployment on a cloud platform

The backend repository takes up to 2.8GB, greatly exceeding our initial expectation of 500MB as illustrated in our project proposal. This is due to the use of large-scale machine learning tools such as tensorflow (up to 1GB). As such, we were unable to officially deploy on a cloud platform (Heroku) as according to our initial plan, due to repository size constraints.

9 Testing Limitations

Localhost deployment restricts large-scale testing
As our application is hosted locally (localhost), network testing that involves streaming large amounts of files onto the server is not possible.

Furthermore local testing prevents us from doing basic network tests such as measuring server response times or testing the bandwidth consumed by an average request. All of this is highly important if we want to deploy our application as users expect fast results.

Lack of participants for proper usability testing
Usability testing normally involves multiple third parties, in particular a subset of the target users of the software. Unfortunately, we are unable to secure this and can only perform limited testing on a few individuals, in particular our colleagues from Monash.

Our usability testing is also biased as all participants are technically skilled individuals who can instinctively know how to use most software. For better test results, a diversification on the participants background is required, such as getting non technical participants to review the overall result, performance, and any errors encountered.

10 Conclusion

In this report we have observed and analyzed the results of each testing phase, discussed the performance, scalability and security of the system as well as collected early user data and their feedback when using our system.

We had also addressed several shortcomings in our testing, notably the lack of online deployment which barred us from performing network and security testing. Since our project scope does not focus on cloud deployment, we believe that these kinds of tests are not required.

In a nutshell, the project has met all quality standards that were outlined during the project proposal phase. We conclude that the requirements of the classification model and end product of web application has been satisfactory based on the test results and user feedback we had obtained.