

# **Software Lab Report on Proof Reader**

Team8bit

Priyansh Kimtee:193050009  
Mehul Jain:193050046  
Satyam Agarwal:193050032

27st November, 2019

# Contents

1	Introduction . . . . .	2
2	Problem Statement . . . . .	2
3	Literature Survey . . . . .	3
4	Design and Implementation . . . . .	4
5	User Documentation . . . . .	9
6	Conclusion . . . . .	9
7	Bibliography . . . . .	10

## 1 Introduction

One of the things clouding people understanding of what proofreading entails is the fact that the word is used differently in different fields. Asking "What is proofreading?" to someone in the publishing profession, for example, will likely garner a very different reply than asking someone at a university.

Someone in the publishing industry would view proofreading as the last possible opportunity to revise a manuscript before it is printed and published. The proofreader compares the proofs—printed versions of the manuscript, which include all the formatting, page numbers, headers, etc. that will be included in the final edition—with the edited copy to make sure that no errors have been introduced by the formatting or printing.

By the time a text is ready to be proofread, it should have been edited already. This means its content should already be well organized, well written, and easy to understand. Editing also involves removing errors, but it focuses more on making sure the text makes sense as a whole. This approach has theoretical limits compared to parsing checkers, but still reaches acceptable results on real life.

## 2 Problem Statement

Proofreading is about finding errors both small and large that were either missed or introduced during editing. Proofreaders ensure that the document's final draft is completely free of grammatical errors (e.g., subject-verb agreement problems, incorrect word choices, improper punctuation usage, and incorrect spelling) as well as formatting and typographical errors. They also make sure the document adheres to the chosen style guide. Unlike traditional proofreaders in the publishing industry, document proofreaders are not limited in the number of revisions they can make to a document, as there is generally no elevated proofreading cost associated with making more changes. However, if proofreaders find that most of the document still requires extensive changes, they may recommend that it undergo another round of editing.

The aim of our project is to develop an Open Source style and grammar checker for the English language. Although all major Open Source word processors offer spell checking, none of them offer a style and grammar checker feature. Such a feature is not available as a separate free program either. Thus the result of our project will be a free program which can be used both as a stand-alone style and grammar checker and as an integrated part of a word processor. The style and grammar checker described in our project takes a text and returns a list of possible errors. Unlike traditional proofreaders in the publishing industry, document proofreaders are not limited in the number of revisions they can make to a document, as there is generally no elevated proofreading cost associated with making more changes. However, if proofreaders find that most of the document still requires extensive changes, they may recommend that it undergo another round of editing.

### 3 Literature Servey

There are online APIs and tools like Grammarly for proofreading and Spinbot.com for rewriting. But this project makes use of standard NLTK library and APIs like Language tool for grammer correction.

There are 3 Approaches possible:

- First is data driven which makes use of learning grammatically valid sentence form using machine learning though lots of Documents.
- Second is Rule based, in which Grammer rules are defined for possible instances.
- Third is Hybrid approach which combines both approach and lots of research is going on to effectively combine merits of both approach like Ability to handle multiple rules qualification By first approach and Detailed error message by second.

## 4 Design and Implementation

Our project uses Django Framework, and Language tool API which takes text as request and returns a string in JSON format.

**Example Response from API:**

```
"matches": [  
  {  
    "message": "This sentence does not start with an uppercase letter",  
    "shortMessage": "",  
    "replacements": [  
      {  
        "value": "This"  
      }  
    ],  
    "offset": 0,  
    "length": 4,  
    "context": {  
      "text": "this is a eaxmple",  
      "offset": 0,  
      "length": 4  
    },  
    "sentence": "this is a eaxmple",  
    "type": {  
      "typeName": "Other"  
    },  
    "rule": {  
      "id": "UPPERCASE_SENTENCE_START",  
      "description": "Checks that a sentence starts with an uppercase letter",  
      "issueType": "typographical",  
      "category": {  
        "id": "CASING",  
        "name": "Capitalization"  
      }  
    },  
    "ignoreForIncompleteSentence": true,  
    "contextForSureMatch": -1  
  },  
  {  
    "message": "Use \"an\" instead of 'a' if the following word starts with a vowel",  
    "shortMessage": "Wrong article",  
    "replacements": [  
      {  
        "value": "an"  
      }  
    ]  
  }  
]
```

```

    }
  ],
  "offset": 8,
  "length": 1,
  "context": {
    "text": "this is a eaxmple",
    "offset": 8,
    "length": 1
  },
  "sentence": "this is a eaxmple",
  "type": {
    "typeName": "Other"
  },
  "rule": {
    "id": "EN_A_VS_AN",
    "description": "Use of 'a' vs. 'an'",
    "issueType": "misspelling",
    "category": {
      "id": "MISC",
      "name": "Miscellaneous"
    }
  },
  "ignoreForIncompleteSentence": false,
  "contextForSureMatch": 1
},
{
  "message": "Possible spelling mistake found",
  "shortMessage": "Spelling mistake",
  "replacements": [
    {
      "value": "example"
    }
  ],
  "offset": 10,
  "length": 7,
  "context": {
    "text": "this is a eaxmple",
    "offset": 10,
    "length": 7
  },
  "sentence": "this is a eaxmple",
  "type": {
    "typeName": "Other"
  },
},

```

```

    "rule": {
        "id": "MORFOLOGIK_RULE_EN_US",
        "description": "Possible spelling mistake",
        "issueType": "misspelling",
        "category": {
            "id": "TYPOS",
            "name": "Possible Typo"
        }
    },
    "ignoreForIncompleteSentence": false,
    "contextForSureMatch": 0
}
]
}

```

**We have creates 4 Views:**

- **Register:**For registration purpose of new user
- **Landing:**Which shows the home screen
- **Logout:**When user logs out and session is destroyed
- **Index:**Deals with the main page to take text and return corrected results

**For Front end we used:**

Jquery,Bootstrap,Javascript,CSS3,Html5

**For Back end we used:**

Sqlite3,Python

**Server**

Django Built-in Server

Team8bit's Proofreader

Sign Up for Free

First Name \*

Last Name \*

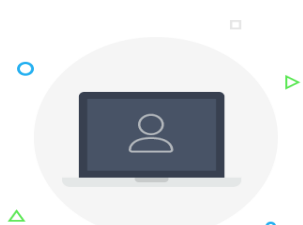
Email Address \*

Set A Password \*

GET STARTED

Figure 1.1: Signup

Team8bit's Proofreader



Member Login

Email

Password

LOGIN

[Forgot Username / Password?](#)

Figure 1.2: Login



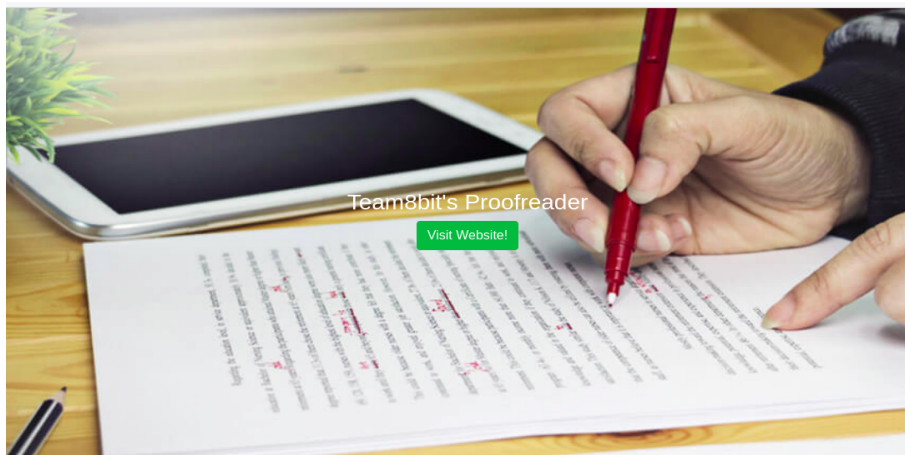


Figure 1.3: Main page

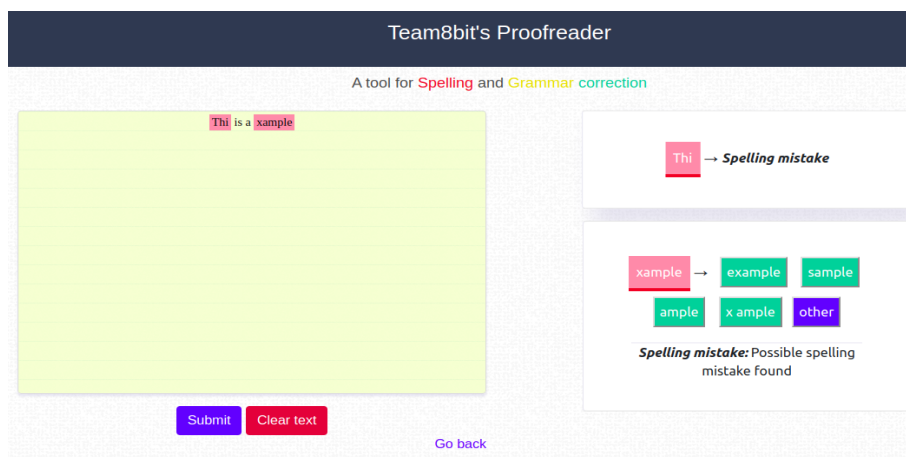


Figure 1.4: Text being corrected

## 5 User Documentation

Our project is very easy to use just follow this steps:

- Install Django on your system command for Linux:

```
sudo apt-get update  
sudo apt-get install python-django
```

- Download the project from the given link of Github.
- Go to the main directory ProofReader(which contains the file manage.py).
- Open the terminal and run the command:

```
python3 manage.py runserver
```

-This will start the server and host our website on the local host.

- Signup if newuser or Login if existing.

## 6 Conclusion

In our project spelling and grammar checker has been developed and has been tested on real-world errors. We used the online API-language tool. Though a proofread is less extensive than an edit, it is an important step when preparing a piece of writing to be read by other people, as errors can cause confusion or be seen as unprofessional. Our proofreading services will help you polish your writing and ensure it is ready for your readers. Thanks to a clear separation between backend and frontend, it is easy to develop graphical user interfaces which interact with the backend. A simple web interface with a limited number of options has been written and the checker has been integrated using Django. We required just Django which had its own built-in server on which we hosted our project as a local host.

## 7 Bibliography

- [1] <https://docs.djangoproject.com/en/2.2/>
- [2] <https://stackoverflow.com/>
- [3] <youtube.com/djangotutorial/telusko>
- [4] <https://languagetool.org/>