International Journal of Aquatic Science
ISSN: 2008-8019
Vol 12, Issue 02, 2021

An Implementation Approach Towards The Automation Of Document Formatting Using Python

Dr. Venkatesan V[1], Hariharan G[2]

[1,2]*Department of Computer Application, PSG College of Technology Coimbatore– 641 004*
[1]*vvn.mca@psgtech.ac.in,* [2]*h.hariran@gmail.com*

Abstract: Documentation is an essential component for reporting / documenting the work done. Manual document formatting might lead to some minor human errors which would remain unnoticed for most of the time. Documents like project report, research papers, thesis etc. often require a certain set of rules and standards which needs to be followed strictly. If the formatting is not done according to those rules, it might even result in rejection of the document. Automation of document formatting will help in solving these problems.
Automation helps in speeding up the documentation process by automatically making corrections in the user's document by comparing it against a template document. The template document will have the formatting rules to be followed and using which the user's document can be validated and verified. Thus this process would not just eliminate minor human errors but also reduce a lot of human effort and time.
The users can access the automation tool through a web application interface which would allow them to upload and format the document based on a particular template and also allows the user to create new custom templates.

Keywords – Documentation, formatting, automation, word processors, documents.

1. INTRODUCTION

Document formatting refers to the way a document is laid out on the page, the way it looks and is visually organized and it addresses things like font selection, font size and style like bold or italics, spacing, margins, alignment, columns, indentation, and lists.

Basically, it refers to the mechanics of how the words appear on the page. A well formatted document is consistent, correct and easy to read.

The visual appeal of a document has an effect on the reader and how they perceive the information, so it's important in any piece of writing or documentation to be concerned with its formatting. Formatting also makes information more accessible to the reader by creating and labelling sections with headings, highlighting key words or ideas with bold, italics, or lists styles and making a good impression with professional look and feel with appropriate font choice for the document type.

There are many word processor tools available with many rich formatting options. It is up to the users to format the document in the required style. Popular word processors like MS Word have an option called Style Templates which can be used to format the document

matching a particular style. These templates are usually available as pre-set options and the users can select the option which matches the requirements of the document. There are options to create own custom templates, but it is quite tedious and time consuming task. The formatting of documents is usually done manually and hence there is a huge scope for automation of this task.

I. EXISTING SYSTEM

In the existing system, formatting is usually done manually. There are tools like Typeset and Latex but they also have their limitations.

Drawbacks of existing system:

- Time consuming and tedious manual work.
- Chances of errors and mistakes is high.
- Existing tools have limited templates.
- No option to add customised templates in the existing tools.
- Inconsistency in the formatting styles.
- Limited support for document file formats.
- Templates limited to certain universities and organizations.

PROPOSED SYSTEM

The proposed system is an automation tool which can be used for document formatting and allows users to add customized templates which can be used to validate and check the formatting of documents to comply with the required formatting rules and specifications.

Benefits of the application:

- Reduces manual effort and time.
- Reduces the mistakes in formatting.
- Options to add customised templates.
- Support for multiple document file formats.

MODULES FOR DOCUMENT FORMATTING

A. TEMPLATES SECTION

Template section will contain a list of template files that are already made available by the institution or organization. The user can select any of the required template and it will have an option to download the template files either in pdf or docx file format. The template files will be stored in the database and will be retrieved as per the user requirements.

B. CUSTOM TEMPLATES SECTION

The Custom template section allows the users the create custom templates based on the requirements. This section allows the user to set custom formatting properties like font type, size, color and style. It also has options to set values for margins, header and footer. It also has option to set dimension values for images that can be added in the document. With all these as input, the application will generate a custom template which will be stored in the database and the users can format their document based on these custom templates also.

C. AUTO FORMAT SECTION

The Auto format section allows the users to upload their existing documents and the application will auto format the document according to the template that the user has

selected. The existing documents can be in any file format. Once the auto format is complete, the formatted document can be downloaded either in pdf or docx file format.

D. AUTOMATION ENGINE

The Automation Engine module acts as the heart of the application. This contains the script files that will be used to automate the formatting process. Whenever a request is sent from the user interface, this module will receive the request and invoke the respective function in the script to handle the automation operation. The script would split the document as runs and make the formatting changes to each of them.

E. FORMAT CONVERTOR

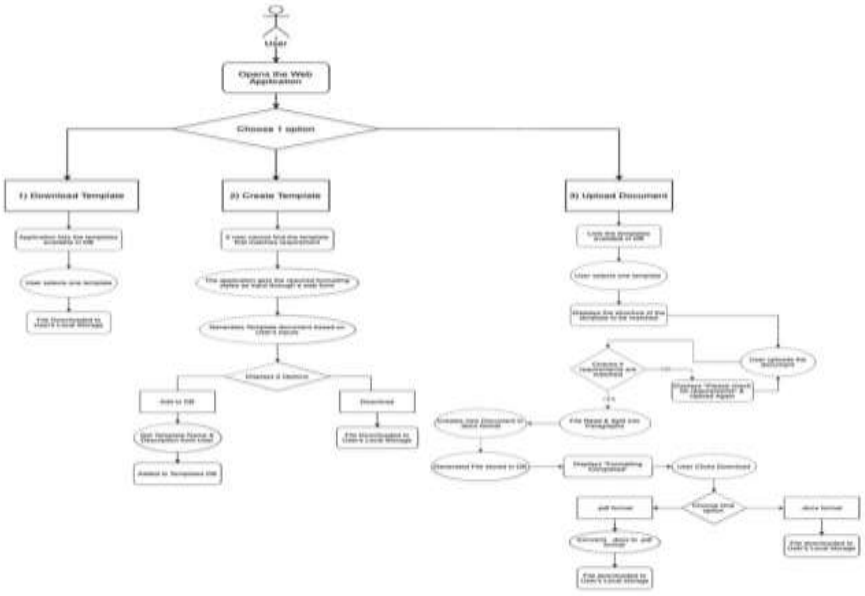
The Format Converter module handles all the file conversion operations. The application can convert input files in pdf or txt file format into docx file format and vice-versa. This module will be called whenever there is a need for format conversion during the process of document formatting. This module will run as a background process at OS level separately. This will not affect the normal execution of the application hence not compromising the performance.

F. DATABASE MODULE

The Database module handles the data storage and retrieval operations. The templates are organized and stored in the database. Each template has a name and a short description about the template which will help in easy reading and retrieval from the database.

The database module will be accessed by all other remaining modules whenever needed respectively for reading and writing files into the database. The template files and document files are stored separately. The files are stored in BLOB data format. This ensures the file data is safe and the database cannot read or modify the data stored in the files.

PROCESS FLOW DIAGRAM



The process flow diagram for the Automation of document formatting is shown in fig.1

Fig. 1 Process Flow Diagram

International Journal of Aquatic Science
ISSN: 2008-8019
Vol 12, Issue 02, 2021

SEQUENCE DIAGRAM

The sequence diagram for the document template module, Custom template module and Auto format module are discussed in this section.

A. DOWNLOAD TEMPLATES MODULE

The sequence flow diagram of Download Templates module is shown in fig. 2.

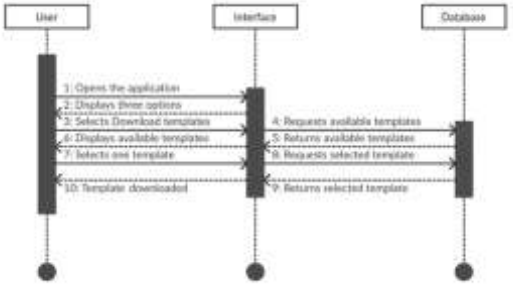


Fig 2 Sequence diagram of Download Templates module

B. CUSTOM TEMPLATES MODULE

The sequence flow diagram of Download Templates module is shown in fig 3.

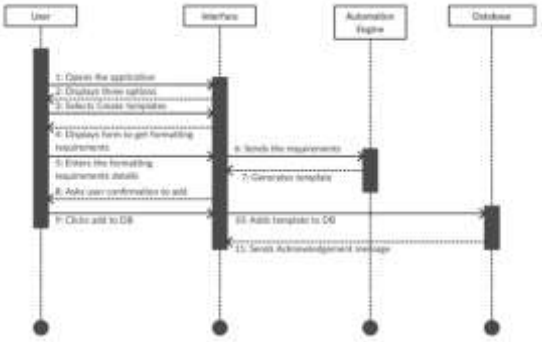


Fig 3 Sequence Diagram Of Custom Templates Module

C. AUTO FORMAT MODULE

The sequence flow diagram of Download Templates module is shown in fig 4.

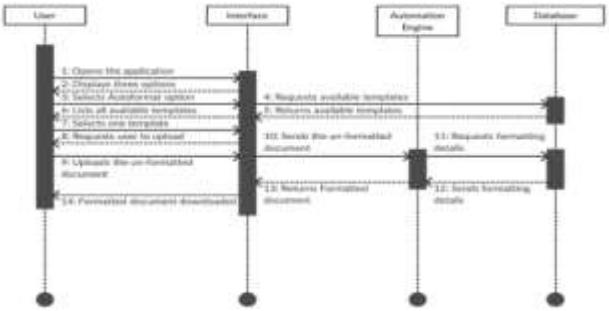


Fig. 4 Sequence diagram of Auto format module

USE CASE DIAGRAM

The use case diagram is an excellent tool for simulating potential users to talk about a system from their own perspective. The use case diagram is used to capture the actors and the communication between them. The fig. 5 depicts the functionality offered and the actors that are involved in this system.

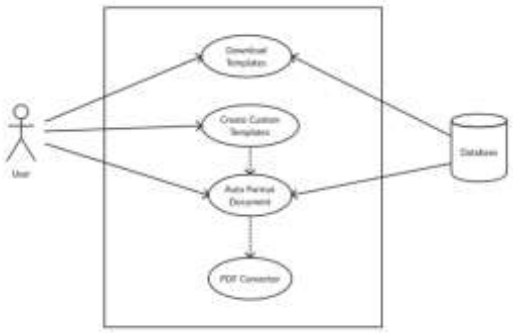


Fig. 5 Use case diagram of the Automation of document formatting system

DATABASE DESIGN

The data modal contains all the logical and physical design. The database is created with the help of a data model. The tables that are designed for the automation tool are given below.

- Templates table
- Custom templates table
- Auto format table

A. TEMPLATES TABLE

The templates table is used to store the predefined template files. The templates table as shown in Table 1 contains id, name, desc, img, template.

Table 1 Templates table

COLUMN NAME	**DATA TYPE**	**NULLABLE**
ID	INTEGER (primary key)	No
Name	VARCHAR	No
Desc	VARCHAR	No
img	BLOB	No
template	BLOB	No

B. CUSTOM TEMPLATES TABLE

The assignment table is used to store the custom user created templates files. The Custom templates table as shown in Table 2 contains id, tname, tdesc, ticon, tformat, customtemplate.

Table 2 Custom templates table

COLUMN NAME	DATA TYPE	NULLABLE
ID	INTEGER (primary key)	No
tname	VARCHAR	No
tdesc	VARCHAR	No
ticon	BLOB	No
formatrequirements	VARCHAR	No
customtemplate	BLOB	No

C. AUTO FORMAT TABLE

The auto format table is used to store the upcoming assignments. The auto format table as shown in Table 3 contains id, Sourcedoc, Formatteddoc.

Table 3 Auto format table

COLUMN NAME	DATA TYPE	NULLABLE
ID	INTEGER (primary key)	No
Sourcedoc	BLOB	No
Formatteddoc	BLOB	No

SYSTEM IMPLEMENTATION

A. HOMEPAGE MODULE

The Homepage module is the main landing page of the application. It welcomes the user and gives a short introduction about the application. It acts as a reception to the entire application. It helps the user by guiding through the various features of the application and also helps them to access the various modules of the application.

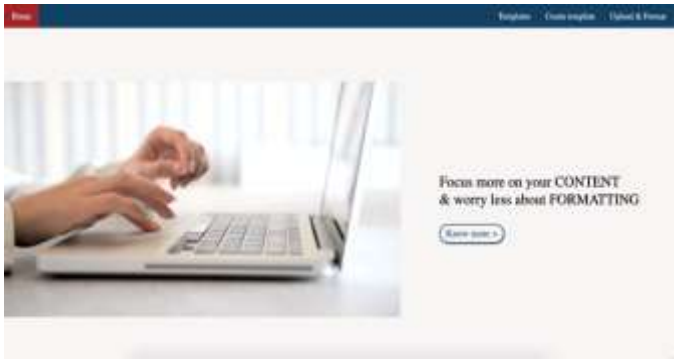


Fig. 6 Homepage

On clicking the know more button in Homepage as shown in Fig. 6 the users will be listed with the features of the application.

Fig. 7 Features of application

The three major modules of the application are listed in this section as shown in fig. 7 above. Each section has a separate card which gives a short description about the modules. All cards also have a know more button which would re-direct to the respective modules of the application. The users can directly access the module of their choice from this features section.

B. DOWNLOAD TEMPLATES MODULE

The Download template module allows the users to download templates directly as .docx files from the application. There will be two templates available as default for the user to download from the application.

Fig. 8 Available templates section

This section lists the templates after fetching details about the templates stored in the database. If the user creates their own custom templates, then those templates will also be listed in this section as shown in Fig. 8. The user has to click on the required template and the application re-directs to the next page of this download templates module.

When the user selects the required template, the name of the template and a short description about the template will be displayed as shown in Fig. 9.

Fig. 9 Template download page

This section gives a glimpse about the template so that the user can know if their requirement is met correctly. Once the user clicks on the Download button, the template file will be download as template.docx file and will be stored in the user's local system storage. Once downloaded, the user can continue to manually format the document using the template that was downloaded from the application.

C. CREATE TEMPLATES MODULE

This module allows the user to create own custom templates if the default do not match the exact requirements of the user.

Fig. 10 Create template homepage

The homepage of the create template module as shown in Fig. 10 gets the requirement details from the user. This entire module is comprised of input forms which is used to get the formatting requirements for the document. Starting with the homepage of the module, the user has to enter the template name which will be used to identify the template uniquely by the automation engine. The user can also enter a short description about the template so that other users can understand much better about the new custom template which will be available in the templates database.

The next page of the module is used to get font requirements from the user.

Fig. 11 Font requirements page

The fonts requirements page as shown in Fig. 11 requests the user to enter details about the font formatting requirements. The user has to enter the Font name or can select one font from the available list. If the required font is not available in the list, they can add their own font into the text field. After entering the font name, the user has to select the formatting requirement for each level of document. The user has to select one formatting style such as

Bold, Italic, Underline for the Headings and Sub-headings texts in the document. Once selected the user has to specify the font size for Heading, Sub-heading and paragraph texts respectively.

The next page of the module is used to get margin requirements from the user.

Fig. 12 Margin requirements page

The margin requirements page as shown in Fig. 12 requests the user to enter details regarding the margin of the document. Margins refer to the padding spacing between the border of the page and the contents of the document. The user has to enter the margin size in pixels for Top, Bottom, Left and Right directions. These margin requirements will apply for all the pages in the entire document. After entering the margin requirements, the user clicks on the Next button.

The next page of the module is to get the document structure from the user.

Fig. 13 Document structure page

The document structure page as shown in Fig. 13 gives an introduction about what is document structure and the importance of it. The document structure plays a vital role for the automation engine to easily identify and format each section of the document correctly. The user has to then select the numbering style which would be followed for all the headings and sub-headings in the document. After selecting the numbering style, the user has to click the Next button.

The next page of the module intimates the end of create template process.

Fig. 14 Margin requirements page

The template ready page as shown in Fig. 14 marks the final page of the create template module. Once all the formatting requirements are collected from the user, the application will generate the template file for the user. The user will be provided with two option at this stage. Firstly, the user can download the generated custom template as .docx file and continue the formatting of document manually. Secondly, the user can upload the generated custom template to the database so that, the custom template along with the details stored in requirements.json file can be used to auto format the user document in the final module of the application. There are two buttons for each option and based on the option selected by the user, the template will either be downloaded to the user's local system storage or will be uploaded to the custom templates table in the database.

D. AUTO FORMAT MODULE

This module allows the user to upload their documents and the automation engine will auto format the document.

Fig. 15 Auto format homepage

The auto format homepage as shown in Fig. 15 lists the templates available in templates table and also custom templates table from the database of the application. The users can select one template from the list. If the user cannot find any template from the list that does not match exact requirements of the user, they can click on the link at the end which would re-direct to the create templates module. If the user has selected any template from the list, the respective template will be opened in the next page.

Fig. 16 Template structure page

The template structure page as shown in Fig. 16 displays the name of the selected template at the top of the page. It also lists the structure of the selected template. The user has to upload the unformatted document with only the contents part matching the structure of the template as shown at the left section of the page. The users can select the file by clicking choose file button.

On clicking choose file button, the browser will open the file explorer window as shown in Fig. 17.

Fig. 17 Select file from file explorer

After the document file is selected, the user should click on the Upload button. After which, the automation engine will compare if the uploaded document matches with the structure of the template. If the document does not match the exact structure, an error message will be displayed. If the document matches the exact structure, then a success message will be displayed.

After the document is formatted, the completion page of the module will be displayed as shown in Fig. 18

Fig. 18 Format complete page

The format complete page will have two option for the user. Firstly, the formatted document can be downloaded in .docx file format. Secondly, the formatted document can be converted into .pdf file format using the application's built-in PDF converter feature.

Based on the user's selection, the formatted document will be downloaded as .docx file or .pdf file and stored in the user's local system storage.

After the file is downloaded, the thank you page will be displayed. The thank you page will also have a link below which would redirect to the About me module so that the users can provide feedback or clear their queries with the developer directly through the contact me section.

Fig. 19 About me section

About me section as shown in Fig. 19, gives a short description about the developer of the application. The about me section also allows the users to send their feedback and queries regarding the application to the developer directly through the contact details provided. This provides a better means for the developer to connect with the users of the application.

TOOLS AND TECHNOLOGIES

A. PyCharm IDE

PyCharm is an integrated development environment (IDE) used specifically for the Python language. It was developed by JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester and also supports multiple web development frameworks including Flask. It has inbuilt support for terminal and so hosting and running the application on internal host server is very convenient and simple.

B. Flask Framework

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. Flask is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. Flask's framework is more explicit than Django's framework and is also easier to learn because it has less base code to implement a simple web-Application. A Web-Application Framework or Web Framework is the collection of modules and libraries that helps the developer to write applications without writing the low-level codes such as protocols, thread management, etc. Flask is based on WSGI (Web Server Gateway Interface) toolkit and Jinja2 template engine.

C. Jinja2 Template Engine

Jinja2 is a full-featured template engine for Python. It is one of the most used template engines for Python. It is inspired by Django's templating system. A template contains

variables which are replaced by the values which are passed in when the template is rendered. Variables are helpful with the dynamic data. Using templates, you are able to set a basic layout for your pages and mention which element will change. This way you can define your header once and keep it consistent over all the pages of your website, and if you need to change your header, you will only have to update it in one place. Using a template engine will save you a lot of time when creating your application but also when updating and maintaining it. Jinja2 is not exclusive to web frameworks. It can be used anytime you want to have text generated from a template. In this post we will not be using a web framework. This will reduce the complexity of the example code. Jinja2 template engine runs on a wide range of Python versions from 2.5 to current versions including Python 3.

D. Python-Docx Library

python-docx is a Python library for creating and updating Microsoft Word (.docx) files. Word documents contain formatted text wrapped within three object levels. Lowest level- Run objects, Middle level- Paragraph objects and Highest level- Document object. So, we cannot work with these documents using normal text editors. But, we can manipulate these word documents in python using the python-docx module. It is a mature package that can parse the MS Word docx file, find the merge fields and populate them with whatever values you need. The package also supports some helper functions for populating tables and generating single files with multiple page breaks. This library can be used to add new text, pictures to document and can also apply formatting styles like bold, italic, underline, etc. It can set margins and padding and change text sizes too. It can also extract text from any given .docx file.

E. WTForms Module

WTForms is a flexible forms validation and rendering library for Python web development. It can work with whatever web framework and template engine you choose. It supports data validation, CSRF protection, internationalization (I18N), and more. There are various community libraries that provide closer integration with popular frameworks. WTForms is designed to work with any web framework and template engine. There are a number of community-provided libraries that make integrating with frameworks even better. Flask-WTF integrates with the Flask framework. It can automatically load data from the request, uses Flask-Babel to translate based on user-selected locale, provides full-application CSRF, and more. WTForms-Alchemy provides rich support for generating forms from SQLAlchemy models, including an expanded set of fields and validators. WTForms-SQLAlchemy provides ORM-backed fields and form generation from SQLAlchemy models. WTForms-AppEngine provides ORM-backed fields and form generation from AppEnding db/ndb schema. WTForms-Django provides ORM-backed fields and form generation from Django models, as well as integration with Django's I18N support.

F. SQLite3 Module

SQLite3 can be integrated with Python using sqlite3 module, which was written by Gerhard Haring. It provides an SQL interface compliant with the DB-API 2.0 specification described by PEP 249. You do not need to install this module separately because it is shipped by default along with Python version 2.5.x onwards. To use sqlite3 module, you must first create a connection object that represents the database and then optionally you can create a cursor object, which will help you in executing all the SQL statements. To execute SQLite statements in Python, you need a cursor object. You can create it using the cursor() method.

The SQLite3 cursor is a method of the connection object. To execute the SQLite3 statements, a connection is established at first and then an object of the cursor is created using the connection object. We can use the cursor object to call the execute() method to execute any SQL queries. First, the sqlite3 module is imported, then a function named sql_connection is defined. Inside the function, we have a try block where the connect() function is returning a connection object after establishing the connection. Then we have except block which in case of any exceptions, prints the error message. If there are no errors, the connection will be established. When you create a connection with SQLite, a database file is automatically created if it doesn't already exist. This database file is created on disk, we can also create a database in RAM. This database is called in-memory database. The great flexibility and mobility of the SQLite3 database make it the first choice for any developer.

2. CONCLUSION

The developed application helps in the automation of document formatting with the help of Python scripts. It allows the users to upload documents of any file format and can download the completely formatted document as output in docx or pdf file formats. It also gives the choice to add custom templates according to the user requirements. The future enhancement can include smart text prediction based formatting and inbuilt option to check grammar of the text in the document.

3. REFERENCES

[1] J. B. S. de Oliveira. Two algorithms for automatic document page layout. In DocEng '08: Proceeding of the eighth ACM symposium on Document engineering, pages 141–149, New York, NY, USA, 2008. ACM.
[2] N. Hurst and K. Marriott. Satisficing scrolls: a shortcut to satisfactory layout. In DocEng '08: Proceeding of the eighth ACM symposium on Document engineering, pages 131–140, New York, NY, USA, 2008. ACM.
[3] J. Lumley, R. Gimson, and O. Rees. Extensible layout in functional documents. In Proceedings of SPIE, volume 6076, pages 177–188, 2006.
[4] Pahwa, P., Taneja, S. and Jain, S. "Design of a Multidimensional Model Using Object Oriented Features in UML", IARS' International Research Journal. Vic. Australia, 1(1) 2011. doi: 10.51611/iars.irj.v1i1.2011.6.
[5] E. Schrier, M. Dontcheva, C. Jacobs, G. Wade, and D. Salesin. Adaptive layout for dynamically aggregated documents. In Proc. of the 13th Intl. Conf. on Intelligent User Interfaces, pages 99–108. ACM, 2008.
[6] Pothula, J.; Prasad, C. D.; Veerrraju, M. S. Dynamic Stability and Analysis of SMIB system with FLC Based PSS including Load Damping Parameter Sensitivity. IARS' International Research Journal, v. 4, n. 2, 2014. DOI: 10.51611/iars.irj.v4i2.2014.37.
[7] https://pypi.org/project/python-docx/
[8] https://palletsprojects.com/p/flask/
[9] https://www.jetbrains.com/pycharm/
[10] https://automatetheboringstuff.com/