Power Reader – Devdoc

Atrij Talgery

Started on 15 April, 2025

Contents

Chapter 1

Requirements

1.1 Objective

The aim or objective is to design a power reader for people who want to read fast.

1.2 Use cases

Here are a few possible use cases:

1. **Reading large paragraphs:** Joe wants to power read a large paragraph of text without consuming a lot of time. He does not want to compromise on understanding detail for the sake of saving time.

2. **Studying for exams:** Joe has an exam coming up in a few days. He has a lot to read and does not have the time to go through all the study material and wants to know the key points.

3. **Legal documents:** Joe works as a lawyer and does not have an idea on how to extract key points from large cases and laws. He is short on time and wants to argue on the case fast.

4. **Research:** Joe wants to research on a topic. He has a lot to read and feels fatigued by the overwhelming volume of information. He wants to know a way of shortening text and extracting the main points.

1.3 Use case in detail

Here are a few possible use cases:

1. **Identifying text:** Joe has a lot of text. He neither has the time to read the whole text, nor he is a fast reader. So, he identifies the text and gives it to the power reader.

2. **Submitting text:** Joe submits the text to the power reader. He submits the text with the click of a button.

3. **Retrieving annotated text:** Joe gets annotated text from the power reader. The power reader highlights various parts of the sentences such as entities.

Chapter 2

Design

(TBD)

Chapter 3

Sprint planning

3.1 Product backlog

1. **Upload file:** Joe uploads a file to the Power Reader.
2. **Select model:** Joe selects a model to do the power reading.
3. **List entities:** The Power Reader lists the entities to be read.
4. **Select entities:** Joe can select entities from the list that he wants to highlight in the text.

3.2 Sprint 1: Uploading the file features

This interface has two buttons: one for browsing files and the other for uploading. When you click on the browse button, a list of files and folders are displayed from which you pick the text file that you want to read. You click on the file that you want and press the update button to accept the file for reading.

End Date: 19 April 2025

3.3 Review

Updated uploading file features sprint.

3.4 Sprint 2: Selecting the model and entities

The Power Reader makes use of the Spacy framework. Here, you are able to select the model of your choice by means of radio buttons. This is a separate form where you can select models out of the four Spacy model options. When you select a model you can use it to highlight entities in your text. The checkbox functionality is also implemented in this sprint with the processing to be done in the next sprint.

End Date: 29 April 2025

3.5 Review

Updated selecting the model sprint.

3.6 Sprint 3: Processing using the Spacy model

Start date: 29 April 2025

The actual Spacy models have to be assigned to the radio buttons. The radio buttons have to select a model. The checkboxes show the entities based on the model that you select. Multiple checkboxes are to be made selectable and a "Refresh" button is to be created at the end of the checkboxes to handle input of selected entities.

End date: 2 May 2025

3.7 Review

Updated the Processing using the Spacy model sprint. Dynamically creates entity form based on models chosen from the model form.

3.8 Sprint 4: Implementing the Spacy entities

Start date: 2 May 2025

The checkboxes help selecting the entities, and the selected entities should be highlighted in the text from the chosen file. The highlighting of entities should be based on the entities you select from the list of entities given with checkboxes.

End Date: 3 May 2025

3.9 Review

Updated this sprint with entity checkboxes. Selecting the checkboxes highlights entities in the text based on the selected entities successfully using Spacy's module Displacy.

3.10 Sprint 5: Styling the app

Start date: 4 May 2025

The app is styled using CSS. A two column layout with three forms on one side and an area to hold the text on the other side is to be created so that it first holds the raw text after file upload and then the highlighted text after the entity selection. All other modifications such as changing background color, font color, highlighting widgets after mouse hover are to be implemented. A header and a footer section is also to be added with the necessary color and font modification. The CSS is to be stored in a separate file which is referred to by the HTML file.

End date: 9 May 2025

3.11 Review

Finished implementing the sprint by styling the Flask app successfully. The styling is implemented in a separate CSS file which is referred to by the HTML file. The app now has a responsive layout that fits most screen sizes, and refits widgets to any window resizing.

3.12 Final review

The Power Reader app called Revv is successfully finished within a course of a few sprints. It is now a functional app with capabilities of uploading file, selecting model, selecting entities and then getting highlighted text which can be used for power reading. You also get the feature of highlighting entities that you want by selecting them in the entity list by means of checkboxes.

However, one thing to note is that the window responsiveness does not work well when a file is not loaded, and works well only when a file is loaded and all the entities highlighted. This is an initial release of the webapp with improvements to be done in the coming releases.