@LIPPY

Smart PDF Reader for better Paper Reading Experience and Knowledge Mining

Summary Report

Md. Siam

Abu Jafar Saifullah Mahin

Kazi Muktadir Ahmed

Mustahid Hasan

Jitesh Sureka

Tasmia Zerin

March 2, 2023

CONTENTS

List of Figures

.

1. Introduction

SCORE 2023 is a worldwide competition organized by the 45th International Conference on Software Engineering (ICSE) for promoting Software Engineering practice. Student teams are required to design and implement one among the several projects given, proving their skills in software engineering.
Our team consists of six undergraduate students from the Institute of Information Technology, University of Dhaka, Bangladesh..
We have selected "**Clippy: Smart PDF Reader for better Paper Reading Experience and Knowledge Mining**", which is a project for a smart pdf reader. We chose it as our project as it can benefit us in achieving better understanding of research. At our institute, we have a lot of students and faculties dedicated to research work. We believe that our work can help a lot of beginners and advanced students to facilitate their efforts and bring their best works.
This report contains requirement analysis, architectural design, trade offs and choices, a management plan, source code management, time management, implementation and platform choices, challenges faced, and lessons learned.

2. Requirement Analysis

2.1. Overview

A smart pdf reader called Clippy will be created expressly to assist scholars by offering a number of useful functions for reading, identifying similar documents, and also comprehending the relationship between them.

2.1.1. Authentication

There are two different types of user modes in our system - Guest and authenticated user. A user needs to provide a full name, email, password, institution and designation to register to the system. A 6 digit OTP will be sent to email for verification. Users can log in with email and password. After a successful login users can access tools that allow him or her to maintain track of previously read research publications.

2.1.2. Searching a Paper

A user can search paper in two ways: By uploading the PDF manually or by searching by the Paper Title. If a user searches using the paper title, a list of related papers will be displayed. Among the displayed papers, users can select a paper. After selection of the paper, they will get three options:

1. Cross Referencing of Objects
2. Summarization of PDF
3. Generating a knowledge Graph

2.1.3. Cross Referencing of Objects

A PDF usually contains a number of figures, tables, charts etc. It references other papers as well. For efficient and easy PDF reading, Clippy supports cross referencing of the objects. Clippy works in two modes: cross referencing enabled mode and cross referencing disabled mode. When cross referencing is enabled, the referenced objects will be shown in a popup overlay whenever the reader hovers over the reference keyword.

2.1.4. Building a knowledge graph

A research paper usually references other papers. The referenced papers may have citations among themselves too. Clippy will generate a knowledge graph showing the citations within the pape.

2.1.5. Summary of Important Elements

A research paper's essential components include an abstract, background information, references, methodology, experiment baselines, and result analysis. Summarizing the important elements may help a reader in grabbing the quicker understanding of the paper. Clippy comes with two types of summarization: extractive and abstractive. Clippy will highlight the important sentences from the paper if an extractive summary is selected. A summary from each block of the paper ie. abstract, background study etc. and TLDR will be shown as a distinct PDF if the abstractive summary is selected.
All the summary generated by Clippy can be exported as PDF.

2.2. Assumption

- Metadata of papers will be available (title, author, journal, pages, year of the article)
- Generally, the paper will follow the IEEE citation style or another accepted format.
- Number of requests to be made to Semantic Scholar will not exceed the request limit of Semantic Scholar
- Number of requests to be made to Meaning Cloud will not exceed the request limit of Meaning Cloud (Extractive Summarization)
- Number of requests to be made to OneAI will not exceed the request limit of OneAI (Abstractive Summarization)

2.3. Scope

- We will use information from the search results provided by Google Scholar
- We support Citation styles: IEEE only.
- We will use the TLDR summary supplied by the Semantic Scholar API.
- We will use meaningCloud API for extractive summary generation.
- We will integrate OneAI API for abstractive summary generation.
- We will use the services provided by Semantic Scholar API.

2.4. Requirements Specification

2.4.1. Use Case Diagram

The following use cases show the list of events that take place between the users and the system to accomplish the goal.

Level 0: Clippy

Primary Actors: User
Goal in context: The diagram represents the whole Clippy application at a glance

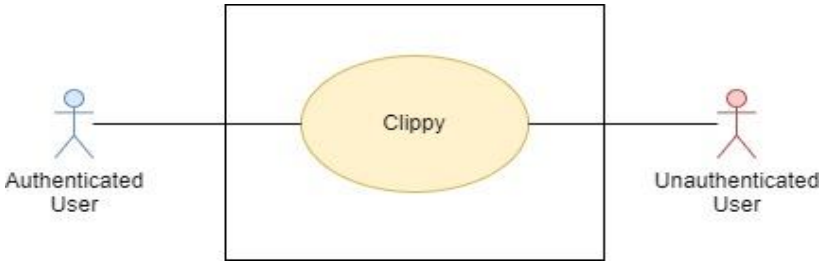

Figure 1: Use Case of Clippy

Level 1: Modules of Clippy

Primary Actors: Authenticated User, Unauthenticated User
Goal in context: The diagram shows all the modules of Clippy application

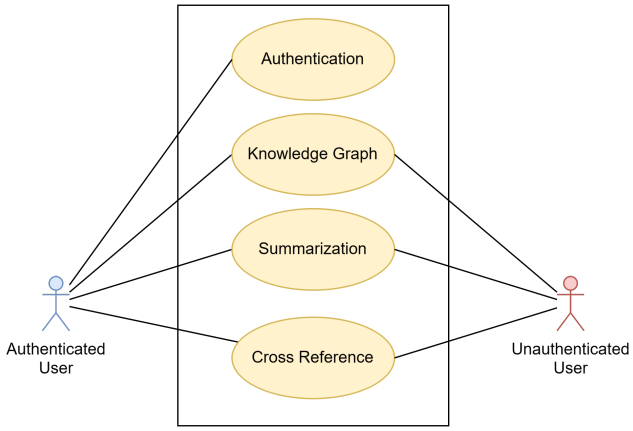

Figure 2: Modules of Clippy

There are 5 modules in Clippy Application
Level 1.1 : Authentication
Level 1.2 : Cross Reference
Level 1.3 : Knowledge Graph
Level 1.4 : Summarization

Level 1.1: Authentication

Primary Actors : Authenticated User, Unauthenticated User

Goal in context : The diagram refers to the details of the Authentication module of level 1.

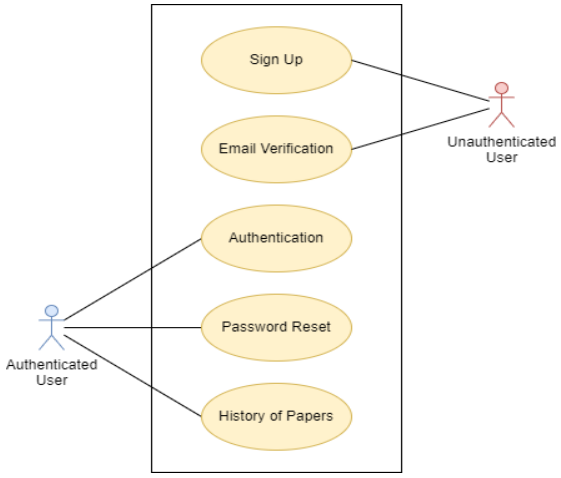

Figure 3: Authentication Module

Action and Replies

A1: Unauthenticated user enters email address and password to register.

R1: System checks whether any personal account exists under the same email or not. If the request is valid, the applicant will receive a confirmation email.

A2: Authenticated user enters email address and password to authenticate.

R2: He/she is allowed to enter into the system upon entering correct credentials.

A3: Authenticated user wants to reset password.

R3: System allows to reset the password through email.

Level 1.2: Knowledge Graph

Primary Actors : User

Goal in context : The diagram represents the Knowledge Graph module of level 1

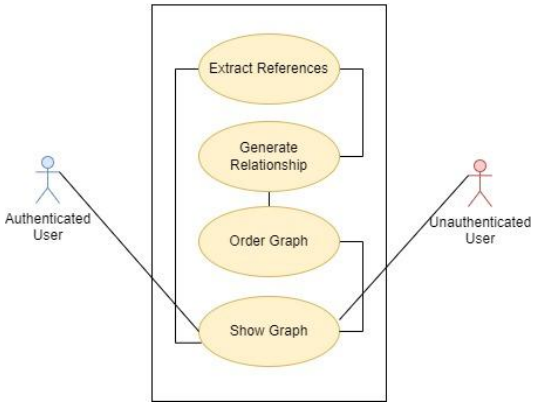

Figure 4 : Knowledge Graph Module

Actions and Replies
A1: User clicks to generate knowledge graph
R1: The knowledge graph is shown

Level 1.3: Summarization

Primary Actors : User
Goal in context : The diagram represents the Summarization module of level 1

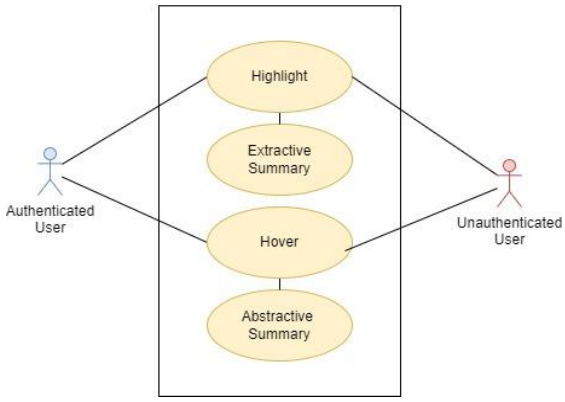

Figure 5 : Summarization Module

Actions and Replies
A1: User selects Highlights option
R1: System takes in the data and highlights the paper using extractive summarization
A2: User hovers over the article
R2: System takes in the data and shows an abstractive summary on a modal.

Level 1.4: Cross-reference

Primary Actors : User
Goal in context : The diagram represents the Cross-reference module of level 1

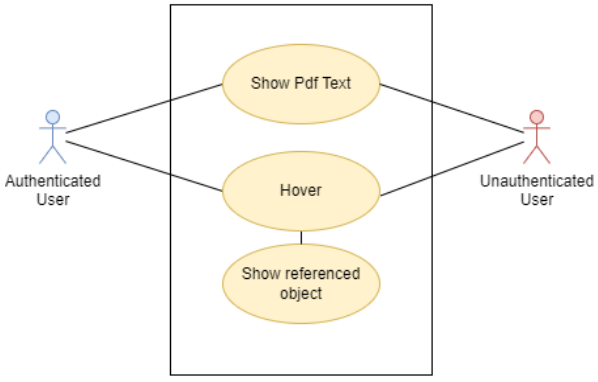

Figure 6: Cross Reference Module

Action and Replies
A1: Users hover on keyword
R1: System shows the referenced object in a popup window.

2.4.2. Activity Diagram

An activity diagram represents the complete flow of a particular use case.

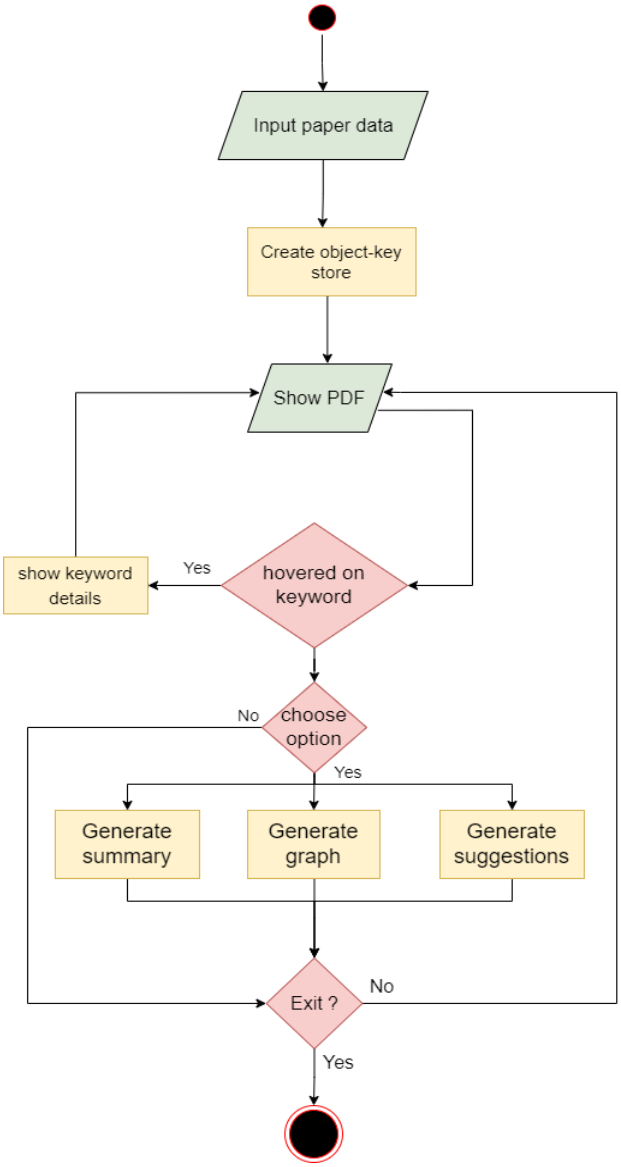

Figure 7: A.D. of Clippy Application

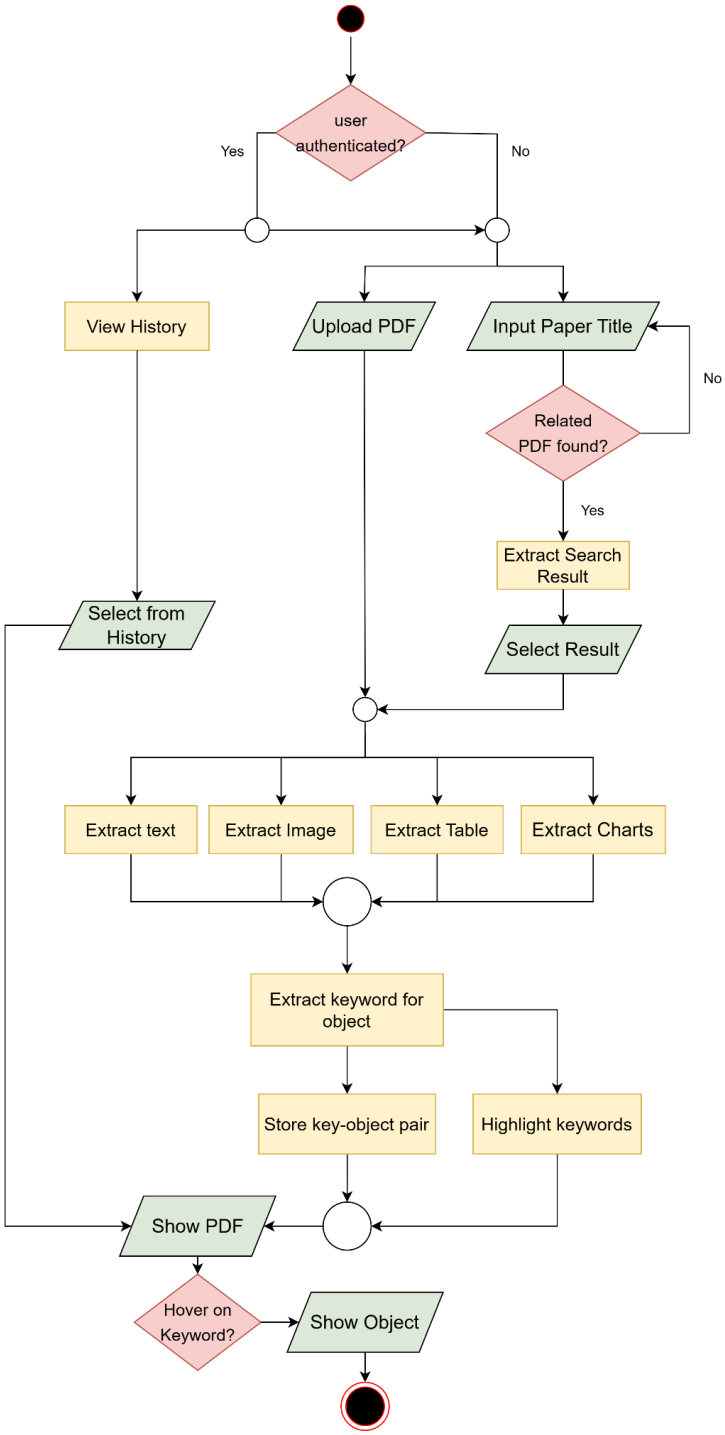

Figure 8: A.D. of Cross Reference module

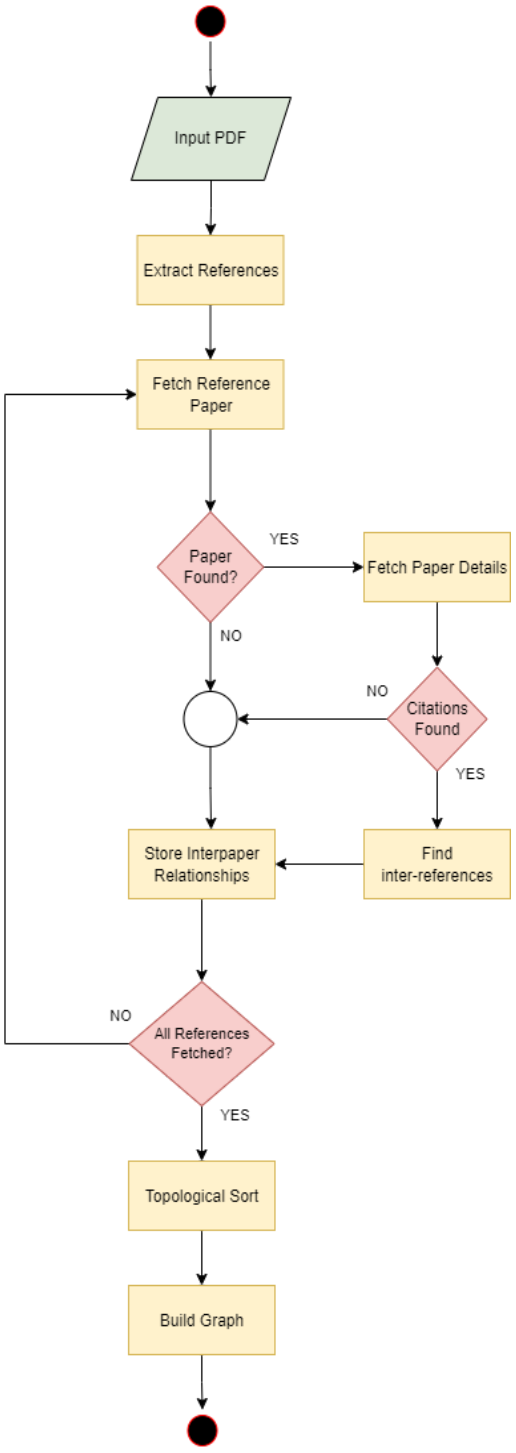

Figure 9: A.D. of Knowledge Graph module

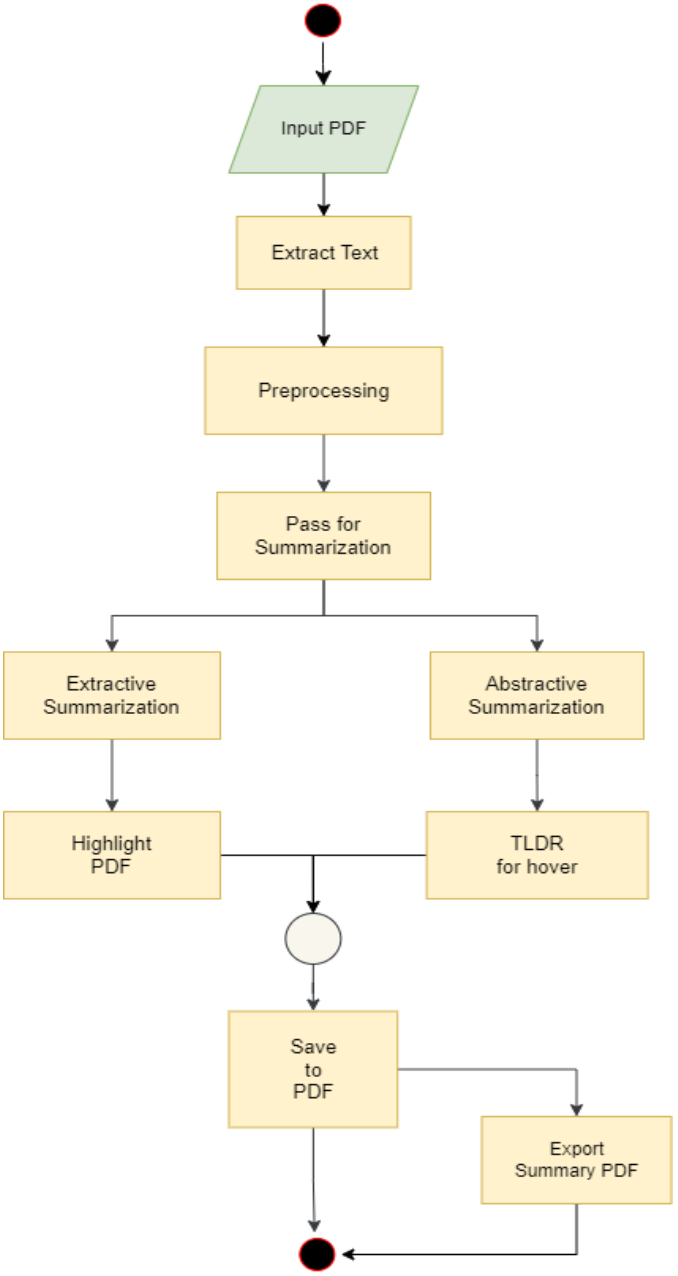

Figure 10: A.D. of Summarization module

2.4.3. Entity Relationship Diagram

Figure 11 shows Entity Relationship diagram of Clippy application. For simplicity only primary keys are shown in the ER diagram.

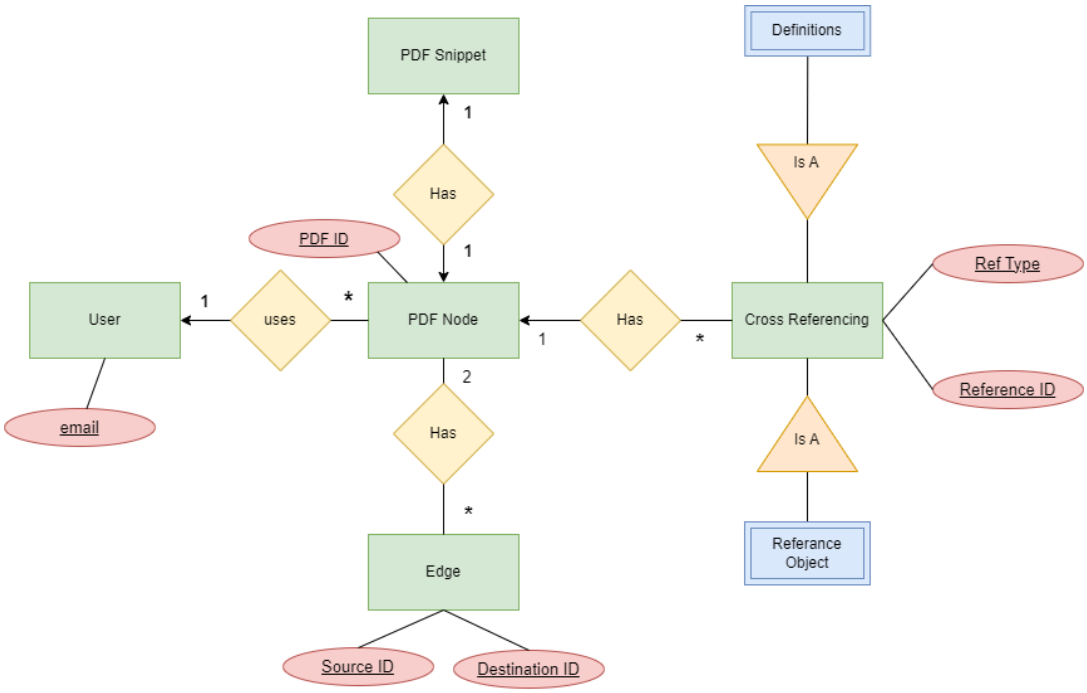

Figure 11: ER Diagram

2.4.4. Schema Tables

We have derived the following tables from the ER diagram (Figure 11).

User

Attribute	Type	Size
email	varchar	50
password	varchar	100
name	varchar	50
reg_Date	Date	

Reference Object

Attribute	Type	Size
reference_ID	varchar	50
reference_type	varchar	50
content	varchar	50

PDF Node

Attribute	Type	Size
PDF_ID	varchar	50
journal_name	varchar	50
PDF_name	varchar	50
Author_name	varchar	50

PDF Snippet

Attribute	Type	Size
PDF_ID	varchar	50
abstractive_summary	varchar	300
extractive_summary	varchar	300
tldr	varchar	100

Edge

Attribute	Type	Size
source_ID	varchar	50
destination_ID	varchar	50

Cross-Referencing

Attribute	Type	Size
reference_ID	varchar	50
reference_type	varchar	50

Definitions

Attribute	Type	Size
title	varchar	50
content	varchar	50

Table 01: Schema chart of Clippy

3. Architectural Design

Our project's architecture is built using a three-tier architecture pattern, which helps us to achieve a high level of modularity, maintainability, and scalability. The three tiers in our architecture are the presentation layer, the business logic layer and the data storage layer.

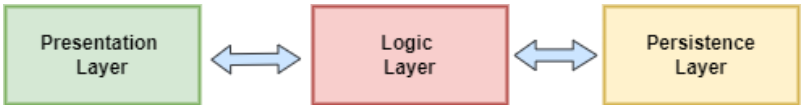

Figure 12: 3-tier architecture of Clippy

The presentation layer is responsible for displaying information to the users and handling user inputs.
The logic layer takes care of implementing the business rules and logic of the project by interacting with both the presentation layer for retrieving user inputs and data storage layer for retrieving and updating data. Our logic layer depends on REST API that provides URL endpoints for the presentation layer to communicate.
The data storage layer is responsible for storing and retrieving data and providing data to the business logic layer.

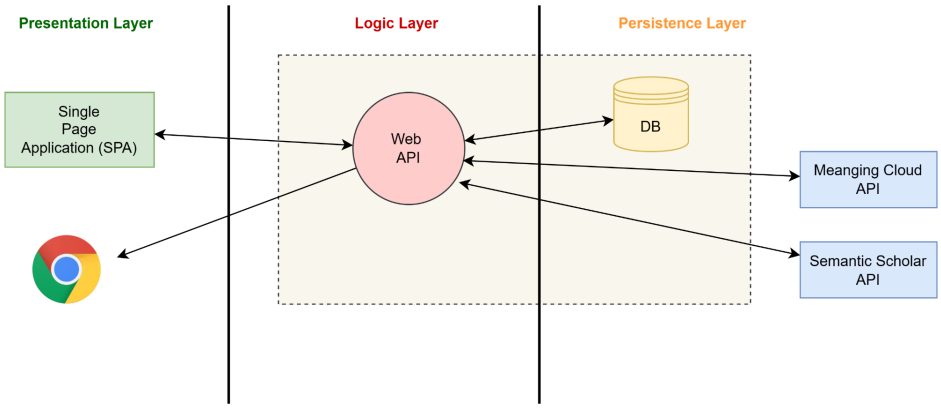

Figure 13: Architectural design of Clippy

3.1. Choices and Tradeoffs

Initially, we thought a user would request and for this we had to make multiple requests from different APIs. However, this task requires a lot of resources. As a result, for an authenticated user, we decide to save summary, cross-referencing, and other information in the database for efficiency and quicker access.

4. Management Plan

4.1. Team Members & Assignments

SI No.	Member Name	Role	Area of Expertise
1.	Md. Siam	Scrum Master	Software Architecture, Networking and Self-adaptive Systems
2.	Abu Jafar Saifullah	Developer	Backend Development and Documentation
3.	Kazi Muktadir Ahmed	Developer	Front-end Development and Documentation
4.	Mustahid Hasan	Developer	Software Requirement Specification, API Integration
5.	Jitesh Sureka	Developer	Backend Development, API Integration
6.	Tasmia Zerin	Designer, Developer	Designing the application and Front-end Development

Table 02: Short statistics of 'Manual Cross Referencing' (MCR)

4.2. Team Coordination

For team management, we followed Scrum. For agile software development, Scrum is a popular methodology. It is chosen because it is lightweight and simple to understand.
The roles are chosen according to the knowledge of the members. Our Scrum Master is Md. Siam. In our development team, there are five members: Abu Jafar Saifullah, Kazi Muktadir, Mustahid Hasan, Jitesh Sureka, Tasmia Zerin.
We have used Trello project management application to assign tasks to members. In every scrum meeting, progress of the tasks were discussed. When tasks were completed, they were marked as complete and new tasks were assigned.
For working together on a document we used Google docs. This is a collaborative platform to share and edit our document among team members. Our diagrams are constructed with draw.io. In draw.io diagrams can be made collaboratively.
We also have used Excel sheet to report and keep track of the bugs in testing.

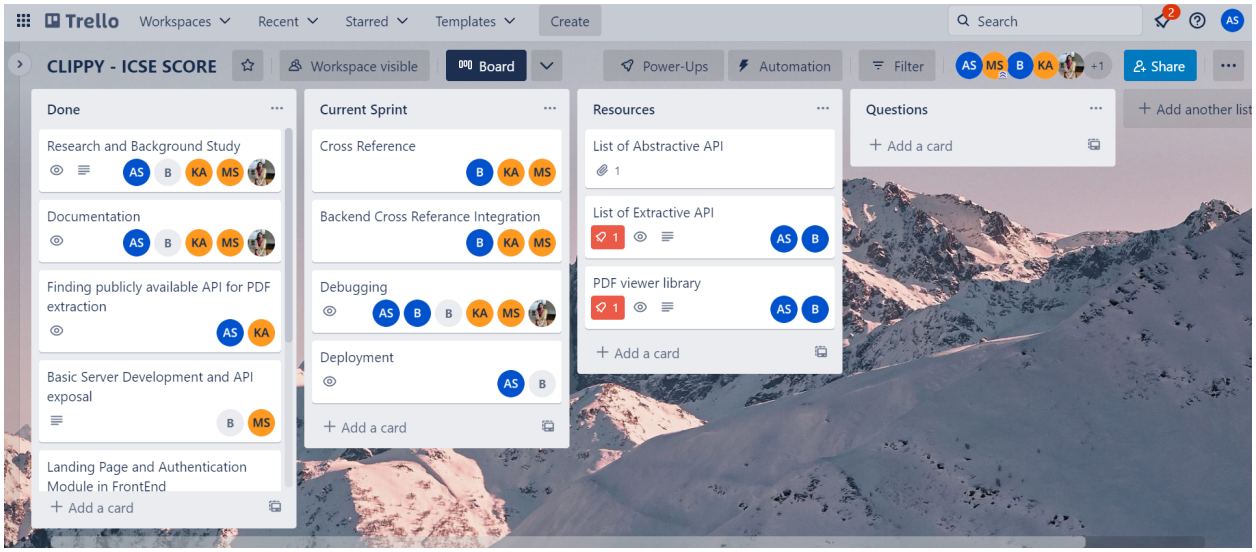

Figure 14: Trello project management of Clippy

Bug reports of Clippy

File Edit View Insert Format Data Tools Extensions Help Last edit was 13 minutes ago

knowledge graph

B	C	D	E	F
Issue	Detail	Issue area	Assigned to	STATUS
Wrong navigation	When user clicks any keyword of the pdf, it ridirects to the home Page	Frontend	Siam	done
Page number updation	When user scroll up/down the page, the page number did not update	Frontend	Tasmia	done
Unnecessary logout button	There shows logout button in the guest mode	Frontend	Tasmia	done
Css of popover div	Popover div goes under the main pdf div	Frontend	Tasmia	done
Array buffer/ formdata error	Does not pass the pdf through the array buffer.	Frontend/backend	Tasmia, Jitesh	done from backend
Image position, backend	When user hovers over the keyword images does not show when the image is at top of the page	frontend/ backend	Tasmia Mustahid	done
Text chunk sanitization	Need to sanitize text chunk	Backend	Jafar, Kazi	done from backend

Figure 15: Excel Sheet of bug reports

5. Source Code Management

To work collaboratively we used Git to manage our project. Our project is hosted at GitHub. We worked in a distributed manner and each of us pushed our code to the remote to keep everyone updated. Git also helped manage versions and different branches of our project.
Github Link: https://github.com/jaf107/Clippy.git

6. Time Management

The time distribution of our project is shown in figure 14.

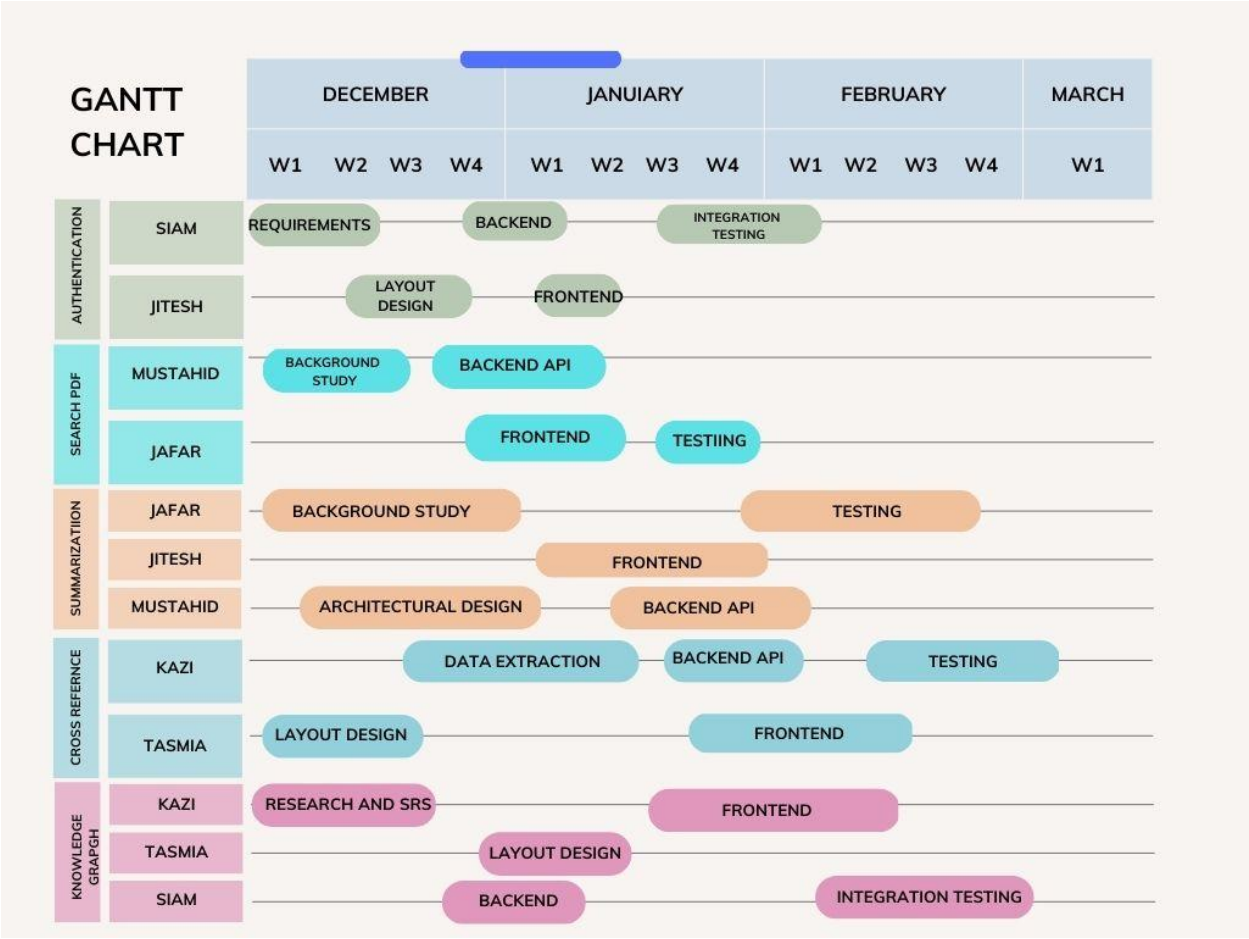

Figure 16: Time distribution of Clippy

7. Implementation

The whole implementation has three parts - the database for storing data, the web api for data communication (Backend) and the UI for presentation (Frontend).

We are following the technologies in order to develop our web application:

Frontend - Angular
Backend - Node Express JS
Database - Mongodb

With the help of the PDF.js library, we can extract the data from the PDF. The extracted data are offered in chunks that are utilized for summarizing and cross-referencing.
A list of all the libraries and packages used by us are attached here [1]

The implementation of core features are given below:
Citation Graph:
In order to generate the citation graph, we are using the paperId obtained from the paper. We then pass it to backend and then request the Semantic Scholar API to get the list of citations for that paper. Following that the frontend receives the result and generates the Citation Graph.

Cross Referencing:
Cross Referencing is done for 3 types of objects: Hyperlink, Table and Image.
We're doing 2 types of cross referencing. Automatic Cross Referencing (ACR) for handling PDFs with annotations. And Manual Cross Referencing (MCR) is used to manage PDF files that lack any pre-installed annotations.

In ACR, we parse the PDF, and take the embedded annotations. Then for each annotation, we render a Pop Over which shows the referred object.

In MCR, we manually parse the PDF and analyze the text chunks to detect images or tables in a page. We cross check the distance between two subsequent text chunks to see if there is any image or tables in between those chunks. Then we extract the keyword for that image/table and the estimated coordinate for the reference on that page. After that a reference object is generated and passed to the frontend for preview.

Hyperlink Extraction:
Using PDFJS, we were able to extract the embedded annotations from the PDF and generate a preview PDF when the user hovered over the annotations. The refDestination data types of PDFJS were used to determine the displayed PDF's dimensions.

Image/ Figure & Table Extraction
From initial retrieval of chunks, chunks with garbage values are removed. Then through analysis, we classify the texts as general text, title based on height and frequency. We use 'Caption Start Identification'

and ‘Region Identification’ approaches in order to extract images from the PDF. The keywords for captions were ‘Table’ & ‘Figure’. We took a threshold value to determine the presence of an object and the value was set through trial & error.
For image extraction, we checked if continuous chunks had a height difference more than the threshold in order to identify the image.
After the identification of the image, it is stored as a JSON object which contains the coordinates & size of the image.
We manually handled special corner cases in which an image can be at the start of a page.

In the case of Table extraction, we took a fixed height in order to retrieve the table. Extracting out Table was very challenging as it was difficult to separate the chunks of table rows with general texts.

Moreover, we had to reverse engineer the parsed information to consumable information for PDF.js. We extended a library named ‘ng2 pdf viewer’ and added new features for our project

Summarization:
Our web app provides 2 types of summary: abstractive and extractive summaries respectively. Initially, the PDF is parsed into chunks that contain the height, width, coordinates, and string. The chunks are then preprocessed and are made ready for passing onto summarization apis.

Preprocessing part was challenging as we needed to extract out title and the general text. The title and general text for each paragraph is extracted out using the frequency of chunk heights and then the paragraphs are structured as JSON objects containing title and general-text. General-text of each paragraph are passed onto summarization apis to generate summaries.
OneAI is used to generate an abstract summary.
MeaningCloud API is used to generate extractive summary. We are considering the summary to have 30% sentence size of the main paragraphs number of sentences. If the paragraph has more than 50 sentences, we consider the extractive summary to have 1/10th number of sentences.

The extractive summary is then further used to highlight the PDF. The highlighting part was very challenging as we obtained chunks of text which are marked if needed to be highlighted.

Backend:
The backend server is coded using ExpressJs and NodeJs. We created REST endpoints for the frontend to hit. Research Paper once entered are stored Database for faster and efficient access and also reducing loading time..

Frontend:

7.1. Platform choices

We chose to develop the system over as a web application. Our reasoning behind the choice was -

1. A web application is globally accessible and the user can use the system without any installation phase.
2. We wanted to build a cross platform system. Thus a web application would enable us to implement a suitable cross platform system.
3. We decided to implement a searching platform for research papers. A web framework would enable us to implement this feature

8. Description and User Manual

Our project supports the key requirements given by the respected ‘Sponsor’. Description & User Manual of the project is described below:

Welcome Page

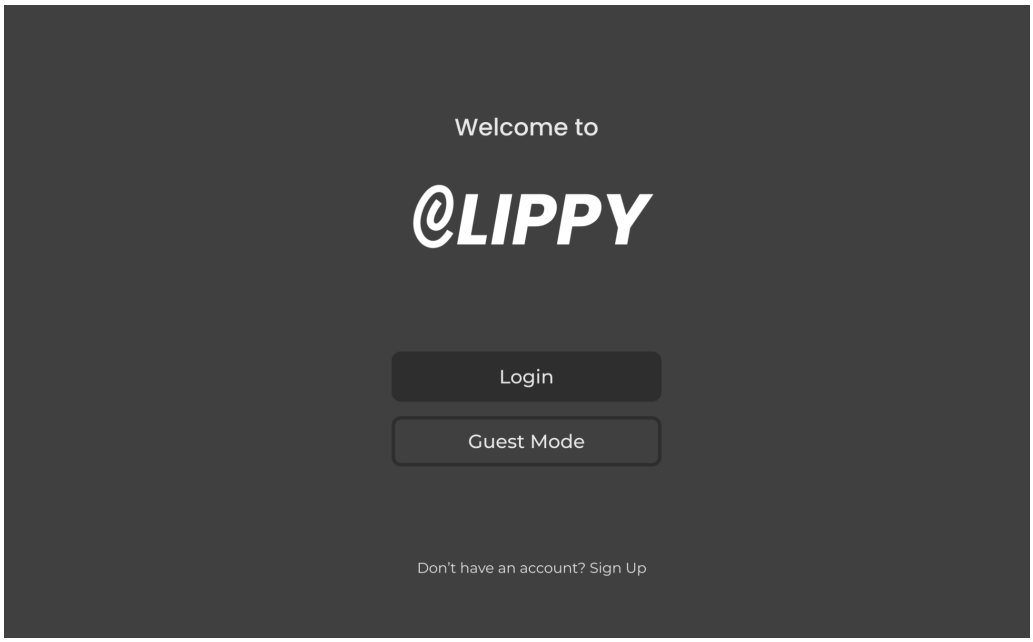

Figure 17: Screenshot of Welcome page

When our application is accessed through a web url. This page is loaded first. A user can sign in in Guest Mode, Login or Sign Up.
A user needs to sign up if she doesn’t already have an account. After that, a user can login. A logged in user has the option to view history which is not available in Guest Mode.

Home Page

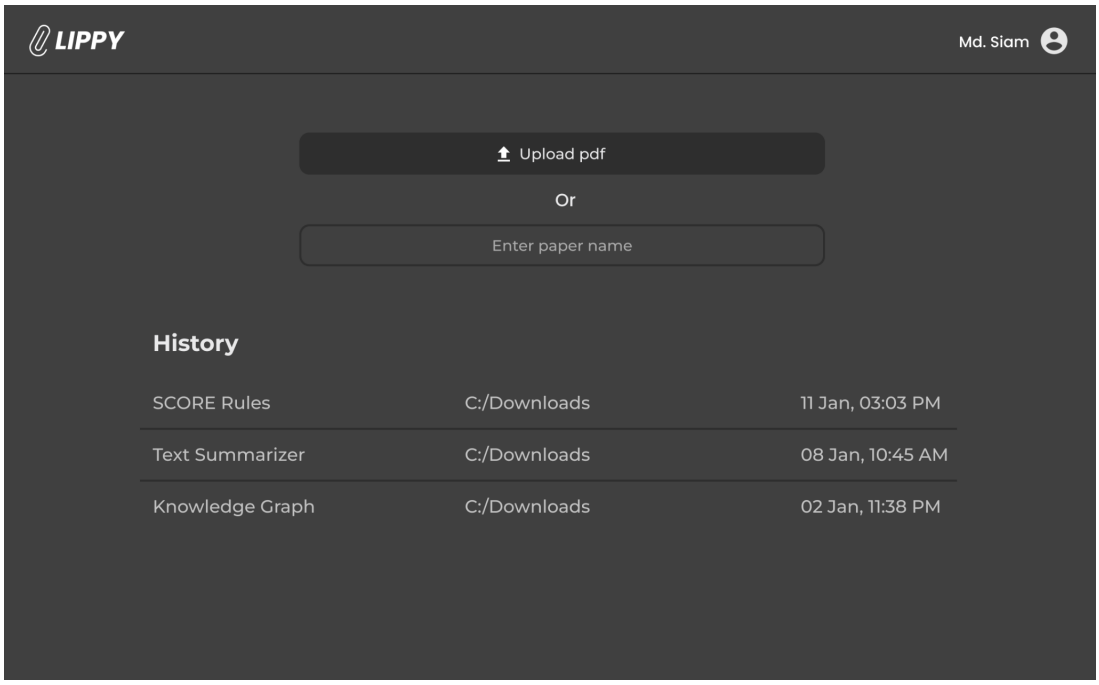

Figure 18: Screenshot of Home page

Here, we can see that a user named ‘Md Siam’ has signed in.

Our Project supports two user modes: Guest mode and Signed in mode. A signed in user has access to his history of previous opened PDFs. The history module can not be accessed in Guest Mode

PDF Viewer Page

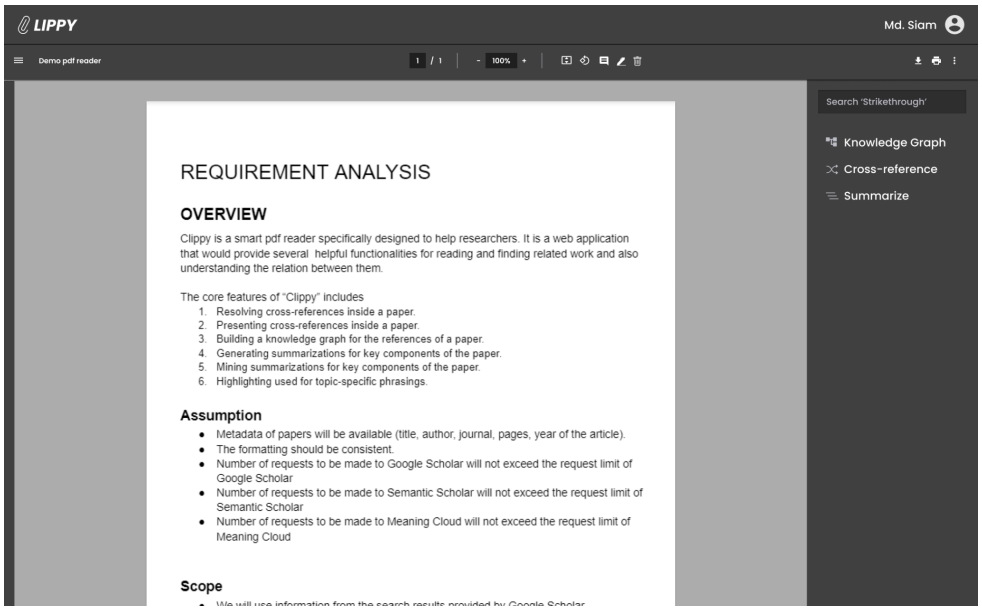

Figure 19: Screenshot of PDF Viewer page

When a PDF is opened, the user gets the above page. A user get access to regular PDF viewer functions. The extra features are shown in the side bar which has the buttons - ‘ Citation Graph (Knowledge Graph)’, ‘Cross-Referance’, ‘Summarization’.

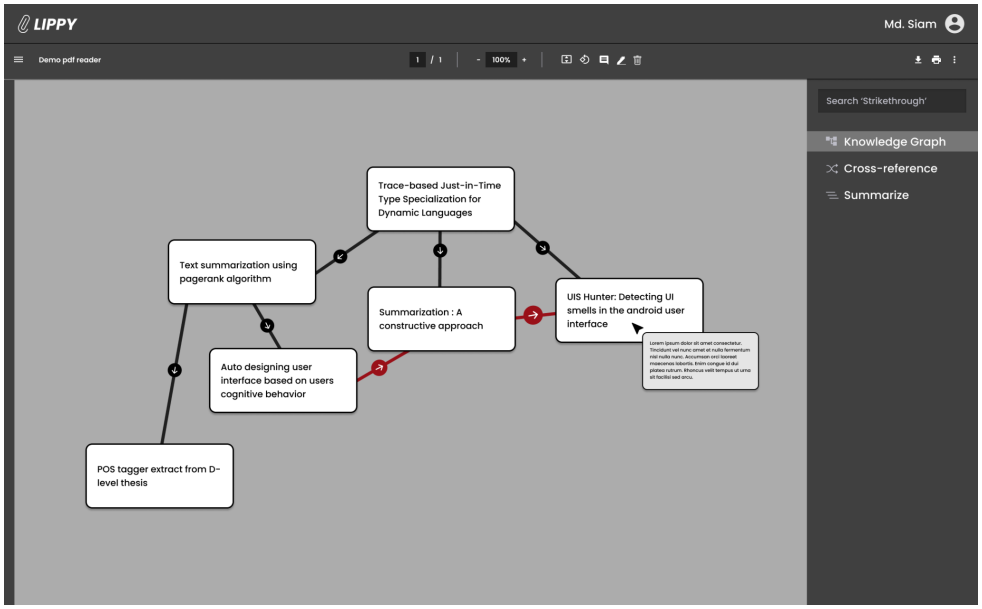

Figure 20: Screenshot of Knowledge Graph feature

The above figure shows the Citation Graph feature where the central node creates a one level Graph for all the citations of the PDF. A user can get hover over the nodes and see the Research Papers Title.

Cross Reference Feature

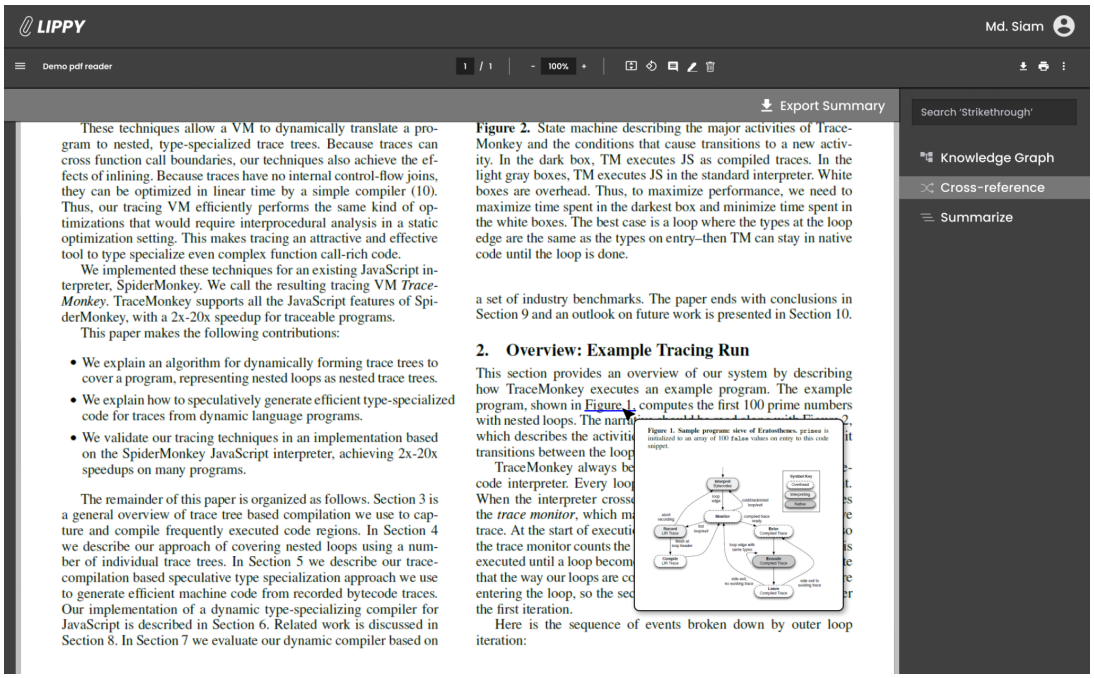

Figure 21: Screenshot of Cross-reference feature

The objects for which Cross-Referencing is done are ‘Hyperlink’, ‘Image’ & ‘Table’. Whenever, a user hovers over the keywords ‘Figure {number}’, ‘Table {number}’, ‘Fig {number}` then a pop up will be shown for that object. For hyperlink, whenever a page is referred then the user can get the information by hovering over the hyperlink.

Summarization & Highlighting

Figure 22: Screenshot of Summarize feature

The above feature is Extractive Summarization. Whenever the user clicks the Extractive Summarization button, then the most important sentences of the PDF gets highlighted.

When the user clicks the Abstractive Summarization button, a side bar is loaded which contains the abstract of the Paper, and it also contains the abstractive summary of each paragraph of the paper from 'Abstract' till 'Reference'.

8. Verification and Validation Activities

Verification is a process of determining if the software is designed and developed as per the specified requirements. Validation is the process of checking if the software (end product) has met the client's true needs and expectations.

1. Inconsistencies, logical flaws, incompleteness, and other problematic details were thoroughly examined in the documentation pertaining to requirements, design, methodologies, and test cases.
2. Unit, integration, and system testing are the three phases of software testing that we did. We carefully examined the input for white-box testing to direct them to independent paths to view their control flow for more thorough testing. To create unit test cases in the frontend, we used Jasmine & Karma libraries which are built in Angular. .
3. After performing manual exploratory testing, we built test cases for both backend and frontend.
4. We used Postman to do API testing and inspected response headers to find defects.
5. After all modules are finished, we need to conduct additional testing on usability issues, integration, etc., as well as some regression testing.
6. A short statistics of our implemented 'Manual Cross Referencing (MCR) ' feature if given below:

Paper Examined	Figure Count	Figure Extracted	Figure Accuracy	Table Count	Table Extracted	Table Extraction Accuracy
67 Research papers	419	345	82.33%	297	214	72.05%

Table 03: short statistics of 'Manual Cross Referencing' (MCR)

```
it(`shoulde have isActive to be True`, () => {
  expect(component.isActive).toBeTruthy();
});
it(`should have optionValue to be False `, () => {
  expect(component.optionsActive).toBeFalsy();
});
it('should have extractive false by default', () => {
  expect(component.extractiveOn).toBeFalsy();
});
```

Figure 22: Screenshot of Test cases from Frontend

8.1 Github Link: https://github.com/jaf107/Clippy.git

Here is given the user manual of this project:

- You can sign up/ sign in to the software or enter as guest mode.
- By signing in to the software, you can see the history of your pdf files.
- You can easily upload your pdf by clicking on the 'Upload pdf' button. Or you can search the paper by entering 'paper ID' or 'Paper title'.
- After that you can view the pdf. In the navbar, you can download the pdf, upload new pdf, zoom in , zoom out the pdf.
- You can view the section wise summarized part of the pdf. If you click the extractive summary button, this will highlight the important information of the pdf.
- You can also view the summarized portion on the right side of the pdf.
- You can explore the abstractive summarization on the right side of the pdf by clicking on the 'abstractive summarization' button.
- To see the referenced part of a pdf file just hover the mouse cursor on the referenced keyword.
- To see the citation graph by clicking on the citation graph option in the sidebar. Click again to go back to the pdf.

10. Challenges and Lessons Learned

10.1. Challenges

We faced a lot of challenges in doing this project. Some of the challenges for different aspects are given below sections by sections:

Backend:
Cross Reference
Cross Referencing of Objects was the most challenging part among all the requirements.

- Extracting Image from PDFs properly handling the corner cases such as
- Extracting Tables from PDFs was tougher as the chunk heights got matched with general text heights. So no significant distinctions could not be found. Thus we had to change our approach and extracted the reference keywords being referred from this paper.
- Manual referencing annotations for hyperlink.
- Customizing the library to extract out hyperlinks
- Creating custom library for hyperlink extraction ~ ng2-pdf-viewer

Once the chunk heights were aligned with standard text heights, extracting images from PDFs was more difficult in corner instances like extracting tables from PDFs. Hence, no notable differences could be discovered. As a result, we had to adapt our strategy and extract the reference keywords from this publication.

manual annotations for hyperlink referencing.

Making changes to the library to remove linkages
ng2-pdf-viewer: custom library for hyperlink extraction

Summarization

- Using chunks by parsing PDFs was very challenging as we never used inputs like this.
- Preprocessing from parsed chunks
- Extracting out title and general text from parsed chunks based on frequency and height analysis
- Highlighting extractive summary using chunks thorough segmentation and creating segments for highlighting.

Frontend:

- Making the UI responsive and fast
- Integrating PopUp PDF Viewer with Regular PDF viewer.
- Handling collision in multiple event bus for multiple pdf viewers.
- Creating a separate custom event for showing the popup viewer
- Reverse engineering the custom pdf viewer to get reference object from page number
- Wrapping the reference keyword in a rendered pdf page.
- Cross Referencing of Objects
- Creative Pop over for keywords.

enhancing the UI's responsiveness and speed
combining the PopUp PDF Viewer with the default PDF viewer.
multi-event bus collision handling for multiple pdf readers.
the popup viewer being displayed through a different custom event
Using reverse engineering to obtain the reference object from the page number for the custom PDF viewer
rendering a PDF page and enclosing the reference term in it.
Cross-Reference between Objects
Creative For keywords, click over.

Moreover, we faced the following challenges as well.

1. Integrating API usage all together to make a complete product.
2. Highlighting particular PDF pages.
3. Working with this form of input was difficult because we hadn't used PDF previously.
4. Extracting keywords from PDF document.

Post Mortem Analysis with Project Reflection

The core f
eatures of the projects were:

- resolving and presenting cross-references inside a paper
    - Our software is able to resolve and present cross-references as pop ups when a user hover overs the objects (‘hyperlink’, ‘image’, ‘table’)
- building a knowledge graph for the citations of a paper
    - Our software shows 1 level knowledge graph for all the papers cited in the paper
- Generating/mining summarizations for key components of the paper
    - Our software provides summarizations for key components(All the paragraphs from Abstract till Reference) of the paper.
- (optionally) highlighting used, topic-specific phrasings.
    - Our software provides extractive summarization that highlights the important sentences of the paper. The important sentences can be used as a learning step for novice researchers.

Study of the after-action using project reflection
The projects' primary features included: resolving and showing cross-references inside a publication.
When a user hovers over the objects (a "hyperlink," a "picture," or a "table," for example), our program can resolve and provide cross-references as pop-up windows.
creating a knowledge graph for a paper's citations
For each paper that is cited in the publication, our program displays a knowledge graph at one level.
creating/mining summaries of the paper's important sections
The essential paragraphs of the article, including every paragraph from the abstract to the reference, are summarized by our program.
Highlighting commonly-used, subject-specific phrases is optional.
Our program offers extractive summary, which draws attention to the crucial passages in the essay. For beginning researchers, using the significant sentences as a learning tool might be helpful.

10.2. Lessons Learned

We learned a lot of lessons as well. They are:

1. We know very little about research. Through the course of this endeavor, we obtained a great deal of data for our research. We developed our teamwork skills.
2. We discovered how to combine several APIs to offer services and a fully functional application.

11. Conclusion

We enjoyed working on this project very much, and it provided us with fresh learning opportunities. We have made an effort to use the best software engineering techniques while implementing as many features as we can. Before beginning to write the code, we went through the design phases in great detail. The

requirements design and architectural design both show the effects of our design processes. There is still much room for expansion of this project.

We would like to thank Professor Xiaoning DU for helping us from time to time by answering our queries.

List of libraries that are used to complete the project

SI No.	Package / Library Name	Function	Version
1	pdf.js	viewing and parsing pdf	
2	cytoScape	Showing edge and nodes	3.23.0
3	ng2-pdf-viewer	custom library to show PopUp	
4	bootstrap-icons		1.10.3
5	cytoscape-dagre		2.5.0
6	ngx-toastr		16.0.2
7	pdfjs-dist		2.14.305
8	rxjs		7.5.0
9	sweetalert2		11.7.2
10	tslib		2.5.0
11	zone.js		0.12.0
12	file-saver		2.0.5
13	Bootstrap	Styling	
14	pdf2json	used for metadata extraction	
15	axios	For api request	1.3.4

16	oneai	Generating Abstractive Summary	0.7.9
17	b64-to-blob	Generating blob data from pdf	1.2.19
18	body-parser	Parsing request body as JSON	1.20.2
19	cloudinary		1.34.0
20	cookie-parser		1.4.6
21	cookie-session		2.0.0
22	express		4.17.2
23	jsonschema		1.4.0
24	express-fileupload		1.4.0
25	crawler-request		1.2.2
26	form-data"		4.0.0
27	dotenv		10.0.0
28	fs		0.0.1-security
29	cors		2.8.5
30	jsonwebtoken,		9.0.0
31	download		8.0.0
32	mongoose		6.1.4
33	multer		1.4.5-lts.1
34	node-downloader-helper		2.1.6
35	path		0.12.7
36	pdf-parse		1.1.1
37	pdf2json		3.0.3
38	swagger-ui-express		4.6.0
39	pdfjs-dist		3.4.120
40	swagger-autogen		2.23.1
41	nodemon		2.0.20

References

1. Gupta, Vishal & Lehal, Gurpreet. (2010). A Survey of Text Summarization Extractive Techniques.
2. Journal of Emerging Technologies in Web Intelligence. 2. 10.4304/jetwi.2.3.258-268.
3. Prateek Joshi. (2018). An Introduction to Text Summarization using the TextRank Algorithm.
4. Beáta Megyes, Brill's rule-based PoS tagger-Extract from D-level thesis (section 3)
5. Extracting Figures, Tables and Captions from Computer Science Papers