Applied Information Technology and Computer Science Vol. 4 No. 2 (2023) 978-996

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs

e-ISSN :2773-5141

Development of Natural Language Processing (NLP) Solution for Text Summarization from Web-pages

Leow Kar Yee[1], Mohammed Saeed Jawad[1*]

[1]Fakulti Sains Komputer dan Teknologi Maklumat,
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

DOI: https://doi.org/10.30880/aitcs.2023.04.02.056
Received 21 June 2023; Accepted 08 November 2023; Available online 30 November 2023

Abstract: Text summarization system using Natural Language Processing (NLP) solution was developed to handles problems faced by students. There are several problems that students faced without text summarization, such as students require to summarize texts to help them complete their assignments or projects, need to manually summarize text from web-pages, and difficult to memorise excessive number of articles. The objectives of this project are to analyse and design a text summarization system based on structured approach, develop a text summarization system using an NLP solution and evaluate the developed system. The prototype model is used in this project. This system is expected to store the records of students and summary results, and the students summarize texts, view summary history records, and generate summary report through this system. Therefore, this system can improve the effectiveness of text summarization in a quick way.

Keywords: Text Summarization, Natural Language Processing, Web-based system.

1. Introduction

With the rise of computers from the development of linguistics, the study of Natural language processing (NLP) has existed for more than 50 years [1]. NLP is a field of computer science and artificial intelligence that involves the development of algorithms and models to analyse, generate, and understand human language. Text summarization is a common application of NLP that involves generating a shorter version of a text while retaining its most important information. An NLP solution for text summarization from web pages can help students save time and improve efficiency and accuracy when reading and understanding large amounts of text. There are currently over 1.98 billion websites online, making an NLP solution for text summarization an important tool for efficiently navigating and understanding the vast amount of information available on the internet [2].

**Corresponding author: saeed@uthm.edu.my*

Students often struggle with reading and understanding large amounts of text, which can decrease their work efficiency and leave them with less time to do other tasks. They may attempt to speed read, but this can result in missing key points. An NLP solution for text summarization from web pages can help students save time, improve efficiency, and increase the accuracy of their work by automatically generating summaries of articles. This solution is particularly useful for students who need to read many articles on a particular topic, such as when researching a paper or preparing for an exam. It can help them quickly obtain the main points and essential ideas from web pages, making it easier for them to understand and remember the information.

An NLP solution for text summarization from web-pages is expected to be developed at the end of this project. The objectives of this project are to analyse and design a text summarization system based on structured approach, develop a text summarization system using a Natural Language Processing (NLP) solution and evaluate the developed system. The proposed system will be developed to be used for an administrator and students. They can register and login the system using their name, password, and email address. The administrator can manage the system's user information. Students can summarise texts from web-pages using a website link. Moreover, students can save website links they want to summarise and the summarised text. They are also able to view and delete saved summary results. The administrator can also manage the summary information. Next, the administrator and students can print and view summary reports. In short, the proposed system can computerise the system to improve the efficiency of the work of students.

This report consists of five parts, Section 1 explains the introduction of the project. Section 2 includes an explanation of the literature review. The system development methodology, system requirement analysis, system design, database design and interface design will be explained in Section 3. The implementation and testing phase will be conducted in Section 4. Lastly, Section 5 is about the conclusion.

2. Related Work

Natural language processing (NLP) is a field of computer science and linguistics focused on enabling computers to understand and interpret human language. It involves developing algorithms and models that can analyse and understand text and speech data. NLP is important in text summarization, a related field that involves generating a concise and coherent summary of a longer document or text. There are two main types of text summarization: extraction-based, which selects important sentences or phrases from the original text to create a summary, and abstractive, which generates a new summary that is a synthesis of the information in the original text. In this project, an NLP solution is used to enhance the text summarization process, aiming to provide users with accurate and informative summaries by leveraging the capabilities of natural language processing to understand and distil key information from web page content.

To study the functionality of the proposed system, a comparison was made between the proposed system and three other related systems, which are Quillbot Summarizer, Wordtune Summarizer and Text Compactor. QuillBot is an online tool for paraphrasing and rewriting that is designed to help improve students' paraphrasing skills and reduce instances of plagiarism in their writing [3]. A study published in 2021 found that the use of QuillBot significantly improved students' paraphrasing skills and reduced instances of plagiarism in their writing. Wordtune is an AI-powered digital writing assistant that provides rewrite options for text to help writers formulate their ideas into clear and concise sentences [4]. It is particularly useful for English as a Foreign Language (EFL) writers, but it should not be relied upon as a sole source for writing and editing. Text Compactor is a system for automatic text summarization, specifically for summarizing stories [5]. It works by calculating the frequency of each word in a passage and assigning a score to each sentence based on the frequency count of the words it contains. The results of testing the system on 10 stories of different sizes showed that it is

effective at generating summaries of stories at various levels of compaction. **Table 1** shows the results of the comparison.

Table 1: Comparison between the existing systems and the proposed system

Feature	Quillbot Summarizer	Wordtune Summarizer	Text Compactor	Proposed System
Method Used	Combination of extractive and abstractive method	Combination of extractive and abstractive method	Combination of extractive and abstractive method	Extractive method
Registration & Login Module	Yes	Yes	No	Yes
User Management Module	Yes	Yes	No	Yes
Summary Module	Yes	Yes	Yes	Yes
History Module	No	No	No	Yes
Report Module	Yes	Yes	No	Yes

3. Methodology

Methodology is a set of methods, practices, processes, techniques, procedures, and rules [6]. In this project, the prototype model is used. This model implements the main phases existing in the SDLC methodology, which are planning phase, analysis phase, design phase, implementation phase, prototype phase and implementation and testing phase. **Table 2** shows the system development workflow of each phase.

Table 2: System development workflow

Phase	Activity	Deliverable
Planning	• Task scheduling • Identify problem, scope, and objectives	• Gantt chart • Proposal
Analysis	• Collect and analyse the information	• Functions require to be in the system • Software and programming language suitable for use
Design	• Design interface of the whole system by using the correct programming language	• The interface of the system using Python programming, MySQL, and HTML
NLP Integration	• Incorporate NLP techniques and algorithms	• Integration of NLP libraries and algorithms
Implementation	• Conduct testing on the system and fix the errors occurred	• Python programming
Prototype 1	• Detect errors on the system and fix the existing system	• Prototype system
	Repeat from planning phase until implementation phase	
Prototype 2	• Detect errors again on the system and fix the existing system	• Prototype system
Presentation	• Present the system in front of the panels	• Final report • Complete system • Microsoft PowerPoint

System analysis and design refer to the process of identifying and implementing solutions to problems through the design and development of systems [7]. The processes involved are Data Flow Diagram (DFD), Entity Relationship Diagram (ERD), flowchart, database design and interface design.

A context diagram is a high-level view of a system that shows the system's boundaries and the interactions between the system and its environment. It is used to provide a broad understanding of the

system and its interaction with external entities. **Figure 1** shows the context diagram of the proposed system.

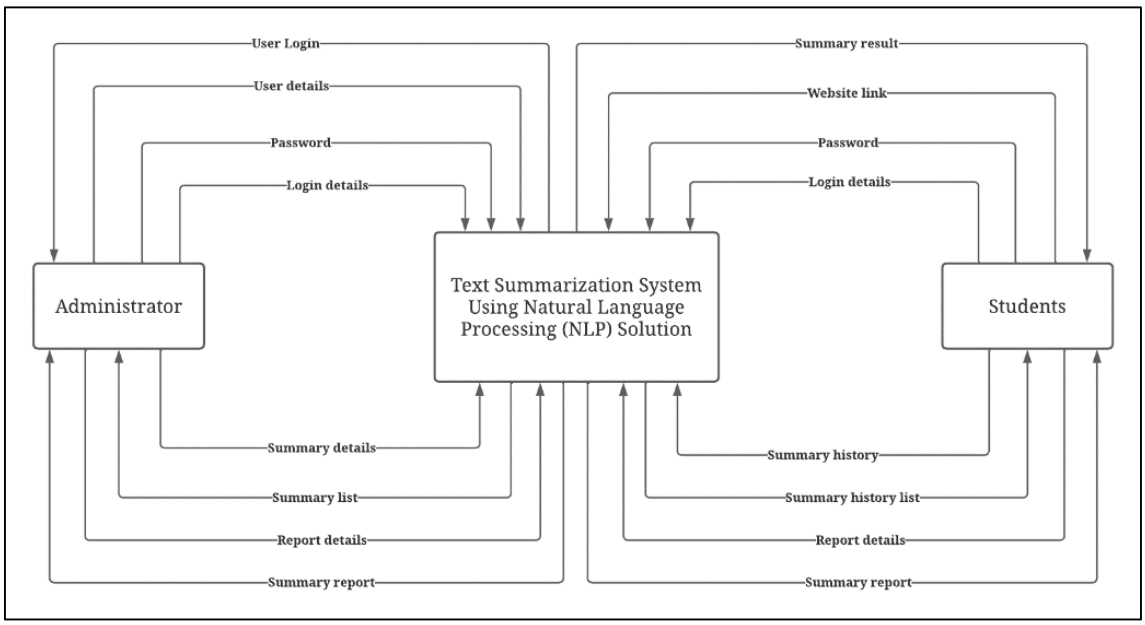


Figure 1: Context Diagram

There are two external entities of the system namely administrator and students. Login details, password, user details, user login, summary details, summary list, report details and summary report will be verified by the administrator. While login details, password, website link, summary result, summary history, summary history list, report details and summary report will be confirmed by the students.

A Data Flow Diagram (DFD) Level 0 is a high-level view of a system that shows the relationships between the system and its external entities, such as users, other systems, and external data sources. **Figure 2** shows the DFD Level 0 of the proposed system.

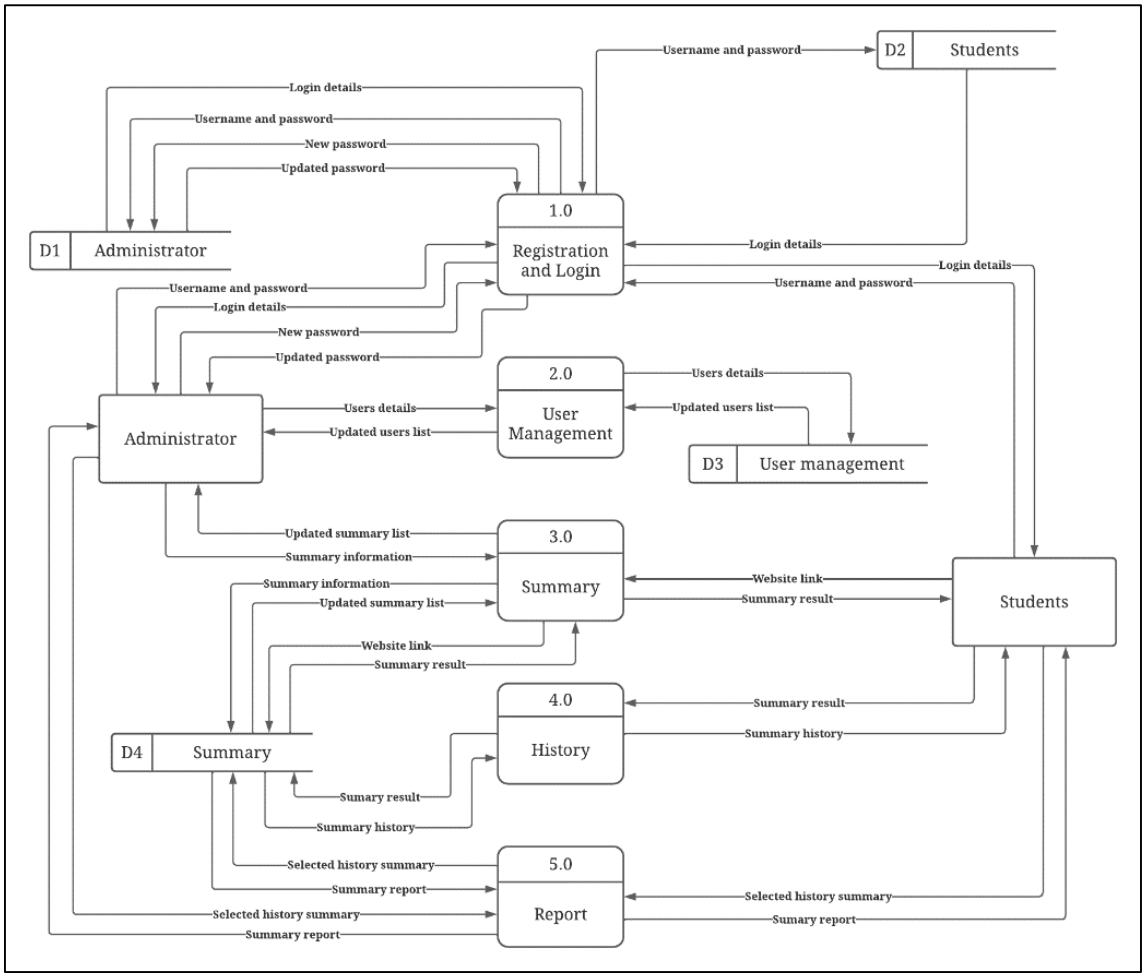


Figure 2: DFD Level 0

Based on Figure 2, the administrator and the users are required to provide a valid username and password to login the system in the registration and login process. The users must provide their name, email address, and password to register the system. The administrator can also change the password. All the data will be stored in a file. The second process is user management. The administrator is able to add, search, edit, or delete users' information. All new or updated users' information will be stored in a file.

In the third process, which is summary, the administrator can add, edit, or delete the summary, while the users can summarise texts using the NLP solution. All new or updated summary information will be kept on a file. The fourth process is history. The users can save the summary and view or delete the saved summarised results in the history record. All the saved summary information is kept on a file. The fifth process is report. The administrator and users can print and view the summary report. All the data will be stored in a file.

An Entity Relationship Diagram (ERD) is a visual representation of the relationships between entities in a database. A flowchart is a graphical representation of a process, system, or algorithm. It typically consists of a series of shapes connected by lines and arrows, which represent the flow of information or tasks. **Figure 3** and **Figure 4** shows the ERD and flowchart of the proposed system.

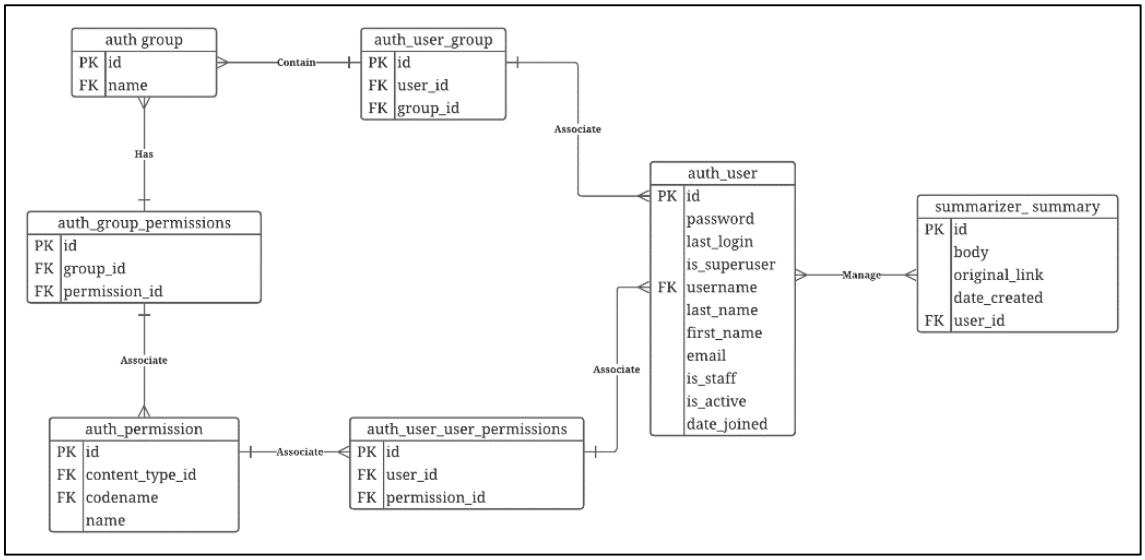


Figure 3: Entity Relationship Diagram

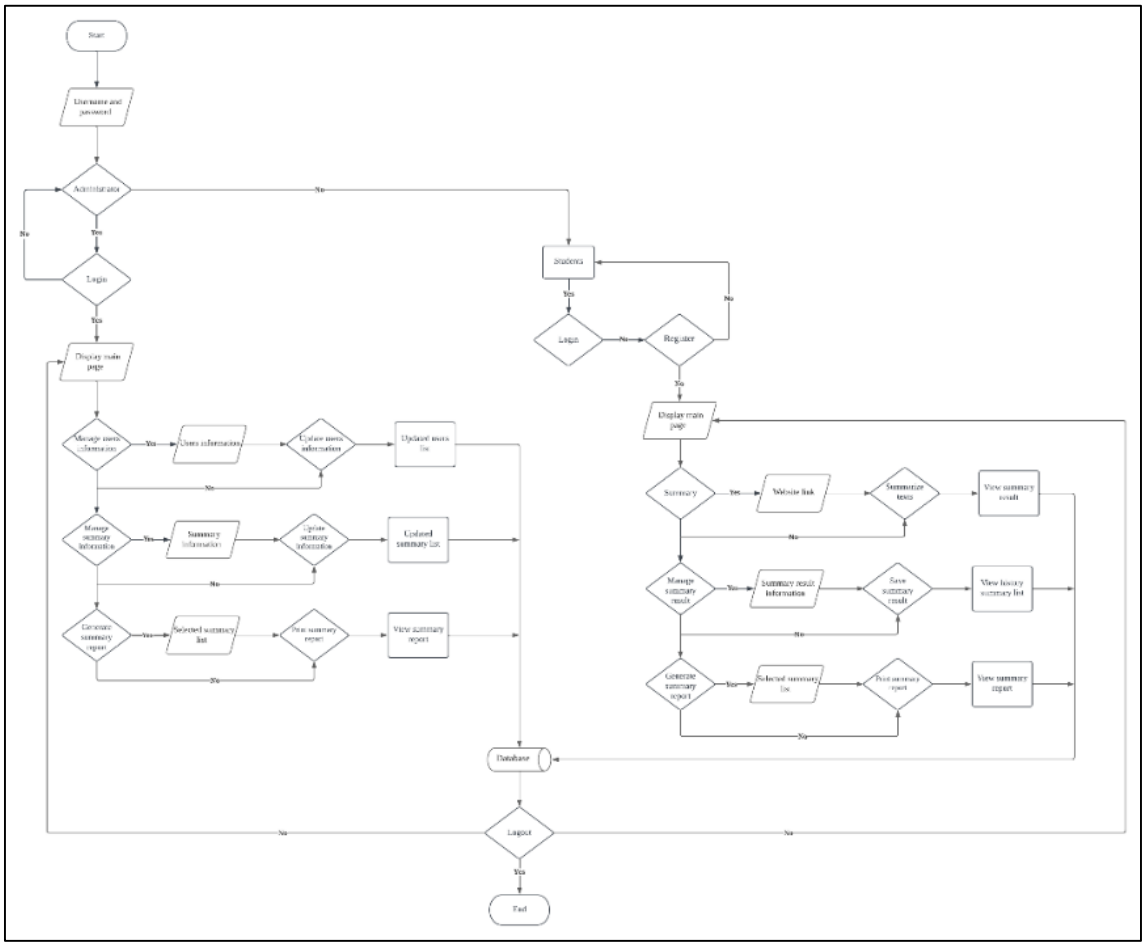


Figure 4: Flowchart

Web application architecture refers to the structure of a web application and the various components that make up the application. A web application typically includes a client-side component, which is the interface that users interact with, and a server-side component, which is responsible for handling requests from the client and providing a response. **Figure 5** shows the web application architecture.

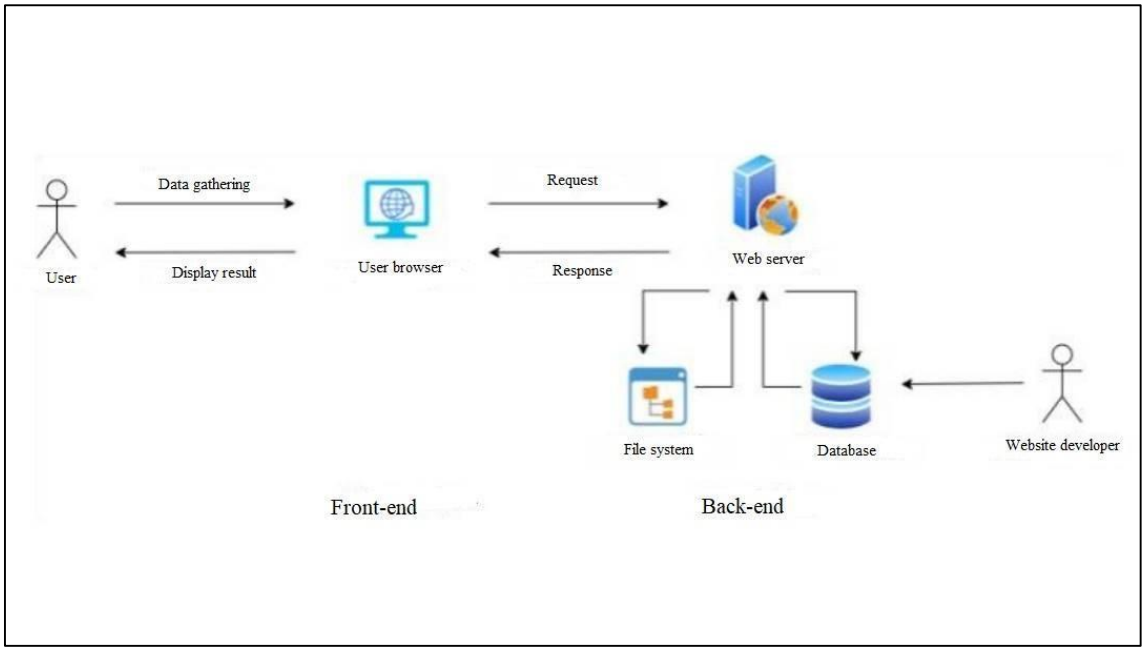


Figure 5: Web application architecture

The scheme tables for the database are listed as follows:

```
 i.   auth_group (id, name)
 ii.  auth_group_permissions (id, group_id, permission_id)
iii.  auth_permissions (id, name, content_type_id, codename)
 iv.  auth_user (id, password, last_login, is_superuser, username, first_name,
      last_name, email, is_staff, is_active, date_joined)
 v.   auth_user_groups (id, user_id, group_id)
 vi.  auth_user_user_permissions (id, user_id, permission_id)
vii.  summarizer_summary (id, body, original_link, date_created, user_id)
```

Interface design refers to the visual and interactive elements of a software application or website that allow users to interact with the system. A well-designed interface can make the system easy to use and understand.

Figure 6 shows the system login interface and the password change interface for the administrator. The administrator is required to fill in the correct username and password to login the system. To change the password, the administrator is required to fill in the old password, new password and confirm password. **Figure 7** shows the system register and login interface for users. Users are required to fill in their first name, last name, email, username, password, and confirm password to register the system. In order to login the system, they are required to fill in the correct username and password.

Figure 8 shows the user management interface for the administrator. The administrator can add, search, edit and delete users. **Figure 9** shows the summary interface for users. Users need to fill in a website link which contains the texts they want to summarise and choose how many sentences they want to summarise. **Figure 10** shows the summary interface for the administrator. The administrator can add, edit, and delete summaries.

Figure 11 shows the history interface for users. Users can save the website link and summary result, and view or delete the saved summary result. **Figure 12** shows the report interface for the administrator. The administrator can print and view the summary report. **Figure 13** shows the report interface for users. Users can print and view the summary report.

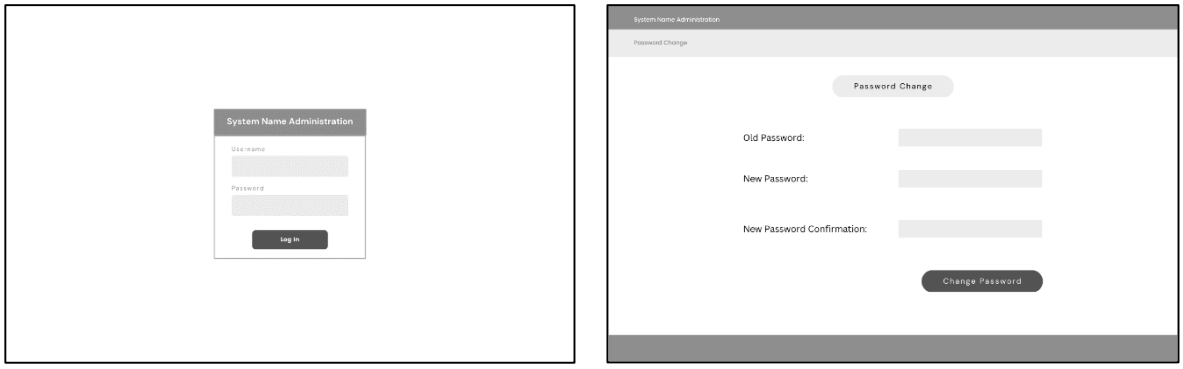


Figure 6: Registration and login module (Administrator)

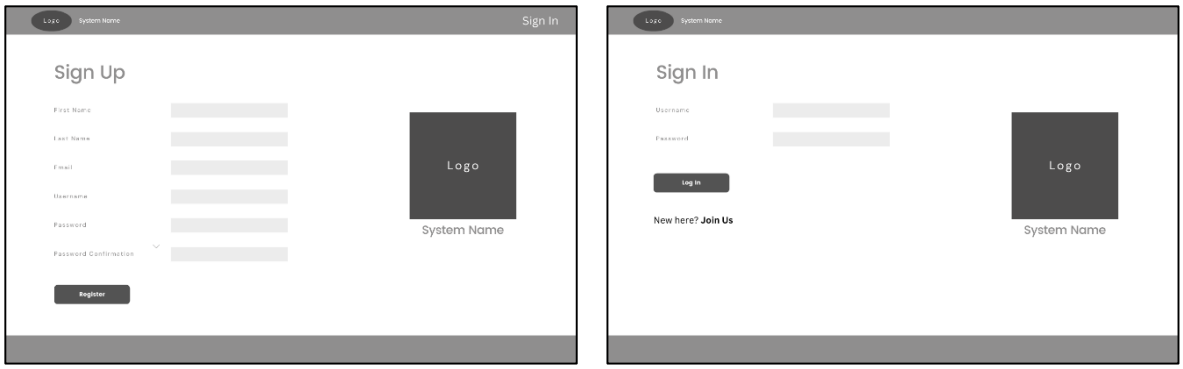


Figure 7: Registration and login module (Users)

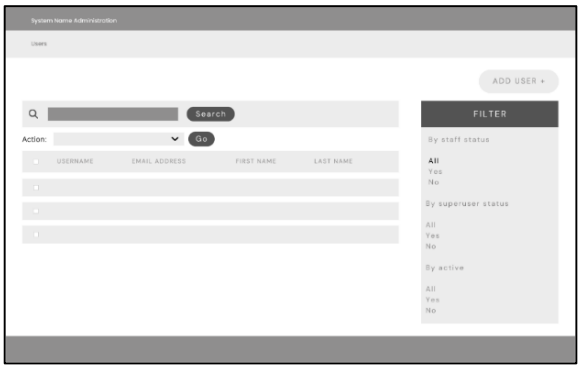


Figure 8: User management module

Figure 9: Summary module (Administrator)

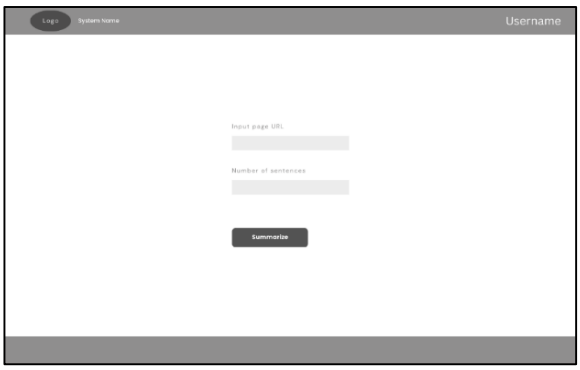


Figure 10: Summary modules (Users)

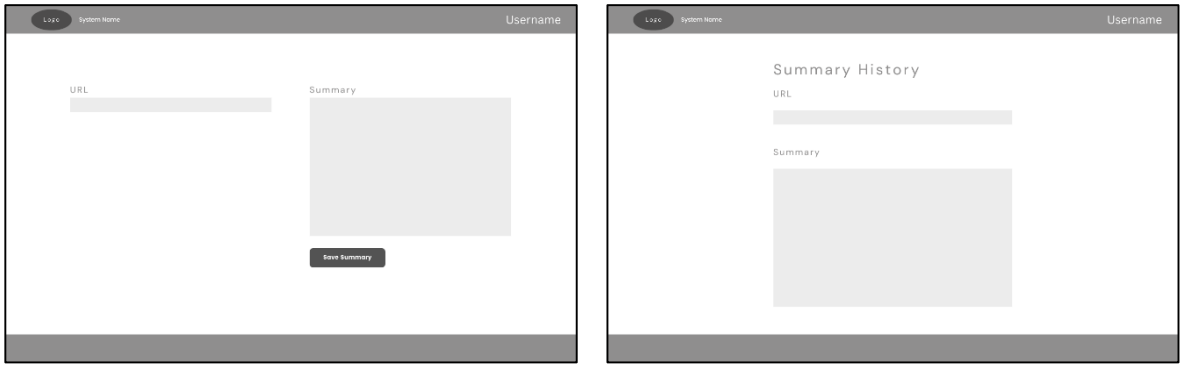


Figure 11: History module

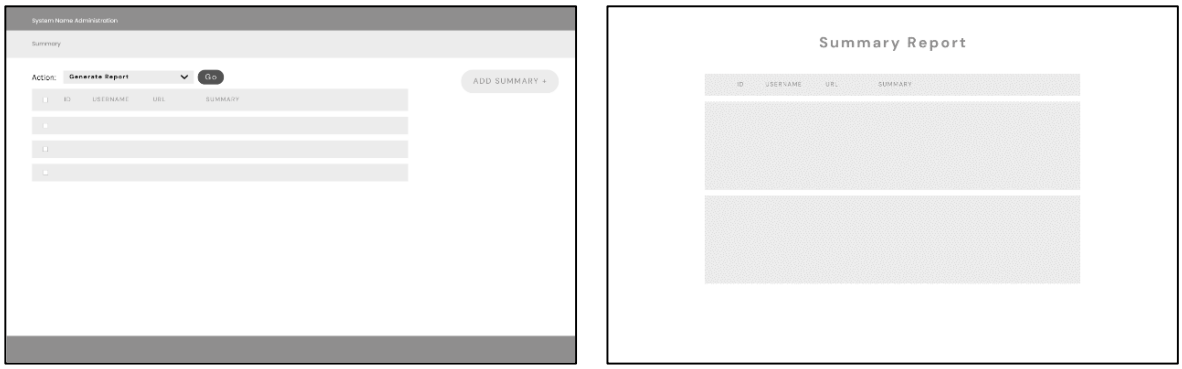


Figure 12: Report module (Administrator)

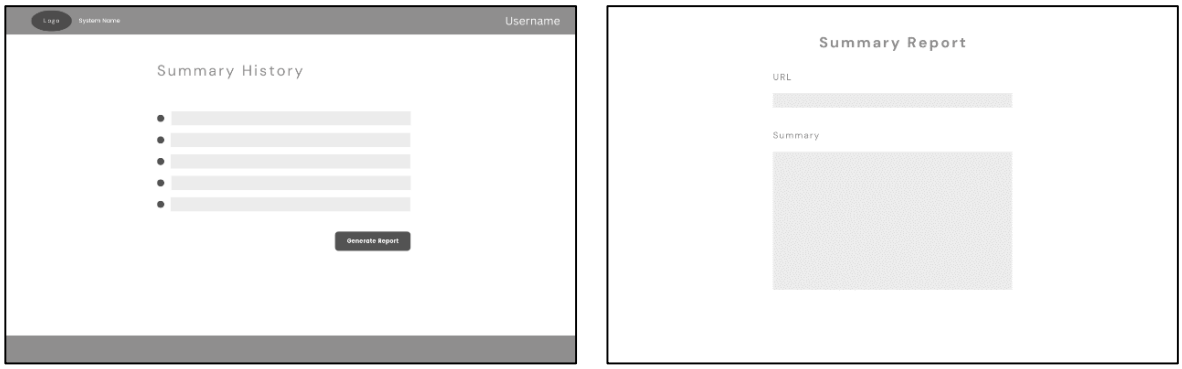


Figure 13: Report module (Users)

4. Implementation and Testing

This section explores the development of functional modules within the system and testing of each module.

4.1 System Implementation

The implementation of this project is website development. For the website, the coding will involve HTML, CSS, JavaScript, and Python programming languages. To facilitate the development process, XAMPP will be utilised as a distributor offering server and database modules, while Visual Studio Code will serve as the integrated development environment for efficient website building.

1) Registration and Login Module

Figure 14 shows the server-side coding of the account registration and login. While **Figure 15** shows the user interface of the account registration page. The user interface only consists of an input box which takes the first name, last name, email, username, password, password confirmation and a button. Next,

Figure 16 shows the user interface of the account login page. The user interface only consists of an input box which takes the username, password, and a button.

```
accounts > views.py > ...
1   # Create your views here.
2   from django.urls import reverse_lazy
3   from django.views import generic
4   from accounts.forms import RegisterForm
5   from django.contrib.auth import authenticate, login
6
7
8   class SignUp(generic.CreateView):
9       form_class = RegisterForm
10      success_url = reverse_lazy('index', current_app='summarizer')
11      template_name = 'registration/signup_form.html'
12
13      # To Automatically login after register
14      def form_valid(self, form):
15          response = super().form_valid(form)
16          # get the username and password
17          username = self.request.POST['username']
18          password = self.request.POST['password1']
19          # authenticate user then login
20          user = authenticate(username=username, password=password)
21          login(self.request, user)
22          return response
```

Figure 14: Account Registration Source Code

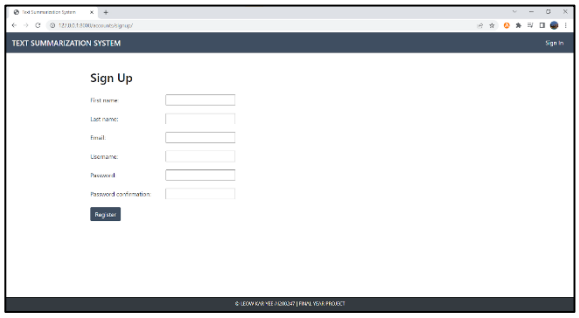


Figure 15: Account Registration User Interface

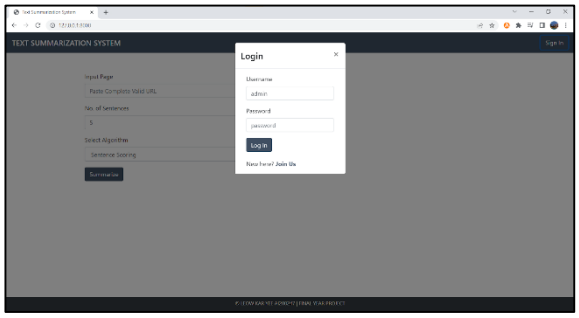


Figure 16: Account Login User Interface

2) User Management Module

Figure 17 and **Figure 18** show the server-side coding and the user interface of the user management. The user interface consists of a table which includes the groups, users and summaries. In this module, administrators are allowed to edit the information of the users and summaries.

```
1   from django.contrib import admin
2   from .models import Summary
3
4   admin.register(Summary)
```

Figure 17: User Management Source Code

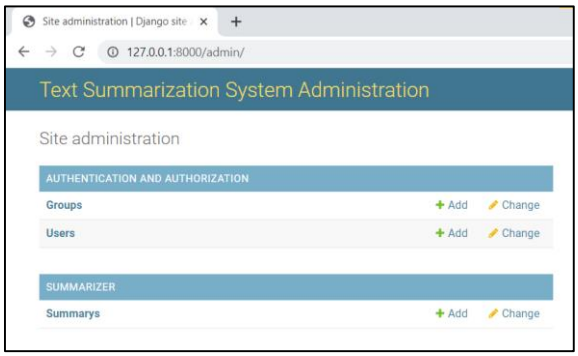


Figure 18: User Management User Interface

3) Summary Module

Figure 19 shows the server-side coding of the summary. While **Figure 20** shows the user interface of the summary page. The user interface only consists of an input box which takes the URL, number of sentences, selected algorithm, and a button. Next, **Figure 21** shows the user interface of the summary result page. The user interface consists of a URL link, a summary result, and a button.

```
23  def summarize_page(request):
24      url = request.GET.get('url')
25      long_text = request.GET.get('long-text')
26      sentence_no = int(request.GET.get('number'))
27      algorithm = request.GET.get('algorithm')
28      result_list = []
29
30      if url:
31          long_text = extraction.extract(url)  # text extraction using BS
32          original_text = url
33      else:
34          original_text = long_text
35
36      if algorithm == '1':
37          result_list = scoring_algorithm.scoring_main(long_text, sentence_no)
38      elif algorithm == '2':
39          result_list = frequency_algorithm.frequency_main(long_text, sentence_no)
40
41      summary = ' '.join(result_list)
42
43      context = {'data': summary, 'original_text': original_text}
44      return render(request, "summarizer/index.html", context)
```

Figure 19: Summary Source Code

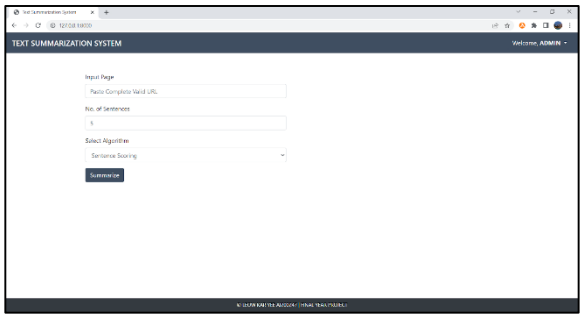


Figure 20: Summary User Interface

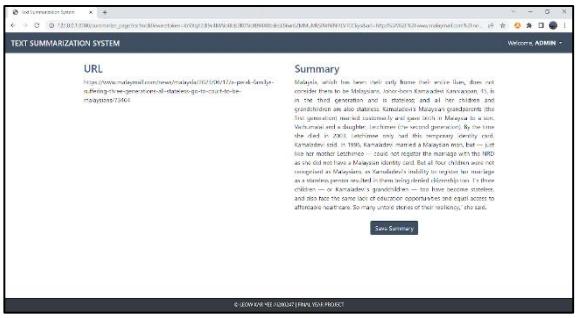


Figure 21: Summary Result User Interface

4) History Module

Figure 22 and **Figure 23** show the server-side coding and the user interface of the history. The user interface only consists of a URL and a summary result. In this module, users are allowed to save summary results, and view or delete history summaries.

```
86  def history(request):
87      summary = Summary.objects.filter(user=request.user).order_by('-id')
88      context = {'data': summary}
89      return render(request, "summarizer/history.html", context)
90
91
92  def history_topic(request):
93      if request.method == 'GET':
94          topic = request.GET.get('topic')
95          summary = request.GET.get('body')
96          context = {'topic': topic, 'body': summary}
97          return render(request, "summarizer/history_topic.html", context)
```

Figure 22: History Source Code

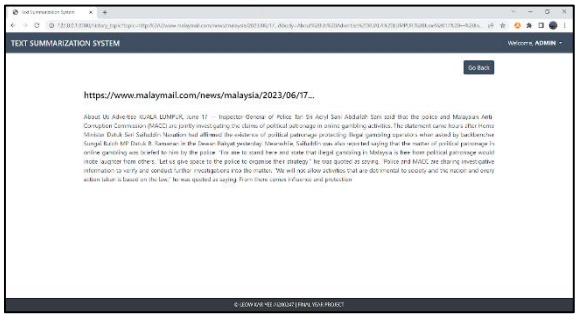


Figure 23: History Result User Interface

5) Report Module

Figure 24 shows the server-side coding of the report for administrators. While **Figure 25** shows the user interface of the report for administrators. The user interface consists of a table which includes the ID, username, original link, body, date created and an action to generate selected summarys. In this module, administrators are allowed to generate the report in PDF format. Next, **Figure 26** shows the generated report for administrators. The report consists of a table which includes the ID, username, URL, body and date created.

```python
def generate_selected_summary(modeladmin, request, queryset):
    # Create the PDF document
    response = HttpResponse(content_type='application/pdf')
    response['Content-Disposition'] = 'attachment; filename="summary_report.pdf"'

    # Create a PDF buffer
    buffer = BytesIO()

    # Set up the document
    #doc = SimpleDocTemplate(buffer, pagesize=A4)
    doc = SimpleDocTemplate(buffer, pagesize=A4, rightMargin=30, leftMargin=30, topMargin=30, bottomMargin=30)

    # Create a list to hold the document content
    elements = []

    # Add the title
    title_style = ParagraphStyle(
        name='Title',
        fontSize=20,
        fontName='Times-Bold',
        textColor=colors.black,
        spaceAfter=40,
        alignment=1  # Center alignment
    )
    title = Paragraph("Summary Report", style=title_style)
    elements.append(title)
```

```python
    # Create a list to hold the table data
    data = []

    # Add the table headers
    header = ['ID', 'User', 'URL', 'Body', 'Date Created']
    data.append(header)

    # Add the summary data
    for summary in queryset:
        row = [
            #str(summary.id),
            Paragraph(str(summary.id), style=ParagraphStyle('Normal', wordWrap='LTR')),
            #str(summary.user),
            Paragraph(str(summary.user), style=ParagraphStyle('Normal', wordWrap='LTR')),
            #summary.original_link,
            Paragraph(summary.original_link, style=ParagraphStyle('Normal', wordWrap='LTR')),
            #summary.body,
            Paragraph(summary.body, style=ParagraphStyle('Normal', wordWrap='LTR')),
            #str(summary.date_created),
            #Paragraph(str(summary.date_created), style=ParagraphStyle('Normal', wordWrap='LTR')),
            Paragraph(summary.date_created.date().isoformat(), style=ParagraphStyle('Normal', wordWrap='LTR')),
        ]
        data.append(row)

    # Create the table
    #table = Table(data)
    table = Table(data, colWidths=[40, 60, 120, 240, 70])
```

```python
    # Create the table
    #table = Table(data)
    table = Table(data, colWidths=[40, 60, 120, 240, 70])

    # Apply styles to the table
    table.setStyle(TableStyle([
        ('BACKGROUND', (0, 0), (-1, 0), colors.beige),
        ('TEXTCOLOR', (0, 0), (-1, 0), colors.black),
        ('ALIGN', (0, 0), (-1, -1), 'LEFT'),
        ('FONTNAME', (0, 0), (-1, 0), 'Times-Bold'),
        ('FONTSIZE', (0, 0), (-1, 0), 10),
        ('BOTTOMPADDING', (0, 0), (-1, 0), 12),
        ('BACKGROUND', (0, 1), (-1, -1), colors.white),
        ('GRID', (0, 0), (-1, -1), 1, colors.black),
        ('VALIGN', (0, 0), (-1, -1), 'MIDDLE'),  # Vertical alignment to middle
        ('WORDWRAP', (2, 1), (2, -1), 1),  # Enable word wrapping for the URL column
    ]))

    # Add the table to the elements list
    elements.append(table)

    # Build the document
    doc.build(elements)

    # Set the buffer position to the beginning
    buffer.seek(0)

    # Write the PDF buffer to the response
    response.write(buffer.getvalue())

    # Close the buffer
    buffer.close()

    return response
```

Figure 24: Report Source Code (Administrators)

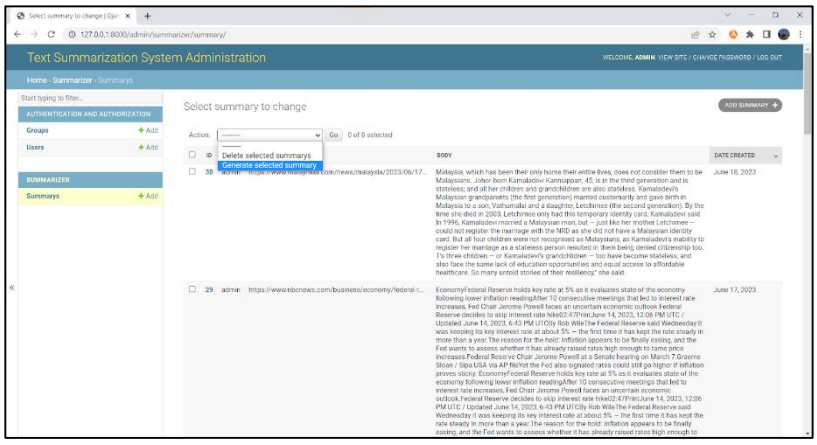


Figure 25: Report User Interface (Administrators)

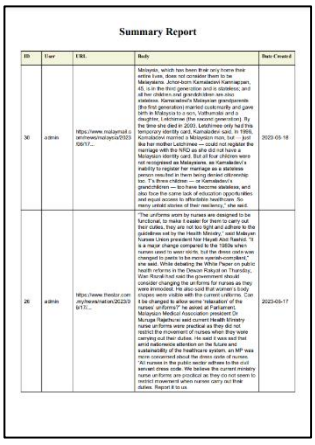


Figure 26: Generated Report (Administrators)

Figure 27 shows the server-side coding of the report for users. While **Figure 28** shows the user interface of the report for users. The user interface consists of a list which is the saved summaries and a button. In this module, users are allowed to generate the report in pdf format. Next, **Figure 29** shows the generated report for users. The report consists of a title, URL and summary result.

```python
def generate_report(request):
    if request.method == 'POST':
        summary_ids = request.POST.getlist('summary_ids')
        summaries = Summary.objects.filter(id__in=summary_ids, user=request.user)

        # Create the PDF document
        response = HttpResponse(content_type='application/pdf')
        response['Content-Disposition'] = 'attachment; filename="Summary Report.pdf"'

        buffer = io.BytesIO()
        doc = SimpleDocTemplate(buffer, pagesize=letter)

        elements = []

        # Define styles for the title
        title_style = ParagraphStyle('TitleText')
        title_style.fontName = 'Times-Bold'  # Customize the font name
        title_style.fontSize = 20  # Customize the font size
        title_style.alignment = TA_JUSTIFY
        title_style.spaceAfter = 15  # Add space after the title

        # Define styles for the topic
        topic_style = ParagraphStyle('TopicText')
        topic_style.fontName = 'Times-Bold'  # Customize the font name
        topic_style.fontSize = 12  # Customize the font size
        topic_style.textColor = 'blue'  # Customize the text color
        topic_style.spaceAfter = 6  # Customize the space after the topic

        # Define styles for the paragraphs
        body_style = ParagraphStyle('BodyText')
        body_style.fontName = 'Times'  # Customize the font name
        body_style.fontSize = 12  # Customize the font size
        body_style.alignment = TA_JUSTIFY
        body_style.spaceAfter = 12  # Customize the space after the topic

        # Add a title to the report
        title = Paragraph("Summary Report", title_style)
        elements.append(title)
        elements.append(Spacer(1, 12))

        # Add the summaries to the report
        for summary in summaries:
            topic = summary.original_link
            body = summary.body

            # Extract the domain from the URL
            parsed_url = urlparse(topic)
            domain = parsed_url.netloc

            # Create a clickable link with the full URL
            # link = f'<a href="{topic}">{domain}</a>'
            link = f'<a href="{topic}" style="font-weight: bold; color: blue; text-decoration: underline;">{domain}</a>'

            topic_paragraph = Paragraph(topic, topic_style)
            body_paragraph = Paragraph(body, body_style)

            elements.append(topic_paragraph)
            elements.append(body_paragraph)
            elements.append(Spacer(1, 12))

        doc.build(elements)
        response.write(buffer.getvalue())
        buffer.close()

        return response

    return HttpResponseBadRequest('Bad Request')
```

Figure 27: Report Source Code (Users)

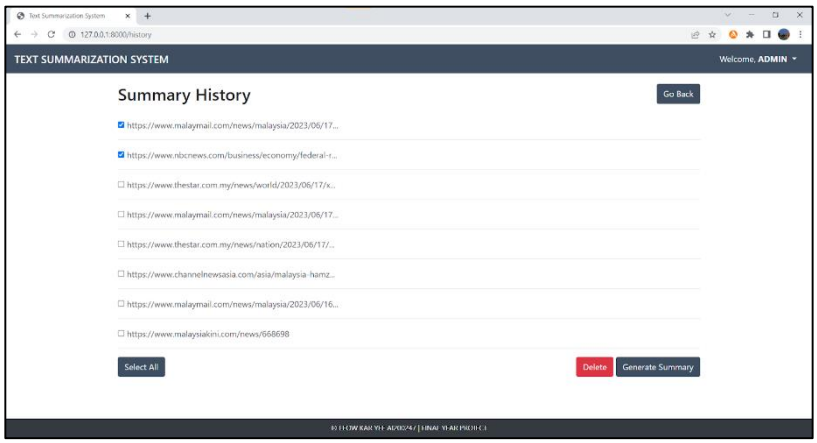


Figure 28: Report User Interface (Users)

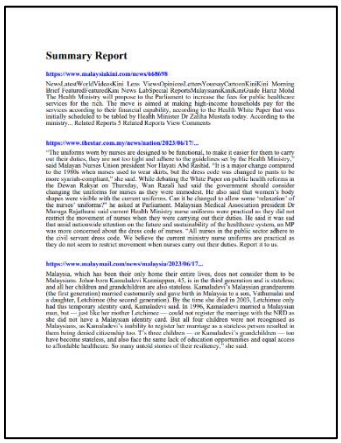


Figure 29: Generated Report (Users)

4.2 System Testing

This section conducts a comprehensive test to evaluate the functionality of each module. The testing process will employ the User Acceptance Test (UAT) method to ensure thorough assessment and validation. **Table 3** to **Table 7** show the results of the functional testing. The test results show that all functional modules can work correctly as intended.

Table 3: Test Cases for Account Registration and Login Module

Module: Account Registration and Login				
Test Case ID	Description	Expected Result	Actual	Pass/Fail
M1-1	To check whether an administrator can register for an account	The administrator should be able to create for an account	The administrator has successfully created for an account	Pass
M1-2	To check whether an administrator can login into the system	The administrator should be able to login into the system	The administrator has successfully logged into the system	Pass
M1-3	To check whether a user can register for an account	The user should be able to create for an account	The user has successfully created for an account	Pass
M1-4	To check whether a user can login into the system	The user should be able to login into the system	The user has successfully logged into the system	Pass
M1-5	To check whether the system will restrict login whenever a wrong credential is entered	The system should restrict login when an incorrect credentials has been entered	The system restricted the login when an incorrect or no credentials has been entered	Pass

Table 4: Test Cases for User Management Module

Module: User Management				
Test Case ID	Description	Expected Result	Actual	Pass/Fail
M2-1	To check whether an administrator can add new user information	The administrator should be able to add new user information	The administrator has successfully added new user information	Pass
M2-2	To check whether an administrator can update or edit user information	The administrator should be able to update or edit user information	The administrator has successfully updated or edited user information	Pass
M2-3	To check whether an administrator can delete existing user information	The administrator should be able to delete existing user information	The administrator has successfully deleted existing user information	Pass

Table 5: Test Cases for Summary Module

Module: Summary				
Test Case ID	Description	Expected Result	Actual	Pass/Fail
M3-1	To check whether an administrator can add new summary	The administrator should be able to add new summary	The administrator has successfully added new summary	Pass
M3-2	To check whether an administrator can update or edit summary	The administrator should be able to update or edit summary	The administrator has successfully updated or edited summary	Pass
M3-3	To check whether an administrator can delete existing summary	The administrator should be able to delete existing summary	The administrator has successfully deleted existing summary	Pass
M3-4	To check whether the user can summarise texts	The administrator should be able to summarise texts	The administrator has successfully summarised texts	Pass
M3-5	To check whether the system will summarise texts accurately	The system should be able to summarise texts accurately	The system has successfully summarised texts accurately	Pass

Table 6: Test Cases for History Module

Module: History				
Test Case ID	Description	Expected Result	Actual	Pass/Fail
M4-1	To check whether the user can save summary	The administrator should be able to save summary	The administrator has successfully saved summary	Pass
M4-2	To check whether a user can view the saved summaries	The user should be able to view the saved summaries	The user has successfully viewed the saved summaries	Pass
M4-3	To check whether a user can delete saved summaries	The user should be able to delete saved summaries	The user has successfully deleted saved summaries	Pass

Table 7: Test Cases for Report Module

Module: Report				
Test Case ID	Description	Expected Result	Actual	Pass/Fail
M5-1	To check whether an administrator can print the report	The administrator should be able to print the report	The administrator has successfully added print the report	Pass
M5-2	To check whether an administrator can view the report	The administrator should be able to view the report	The administrator has successfully viewed the report	Pass
M5-3	To check whether a user can print the report	The user should be able to print the report	The user has successfully added print the report	Pass
M5-4	To check whether a user can view the report	The user should be able to view the report	The user has successfully viewed the report	Pass

4.3 User Acceptance Test (UAT) Results

The purpose of UAT is to verify that the developed system functions smoothly and meets the specified requirements. During UAT, the system is tested by an administrator and users who try it out and assess its functionality and usability while performing various tasks. To facilitate the testing process, a Google Form is used as the response form for the administrator and users. The Google Form can be found in the Appendix. The user feedback collected from the Google Form includes responses from both the administrator and students. **Figure 30** and **Figure 31** shows the feedback from respondents, 90% were students, while the remaining 10% was an administrator.

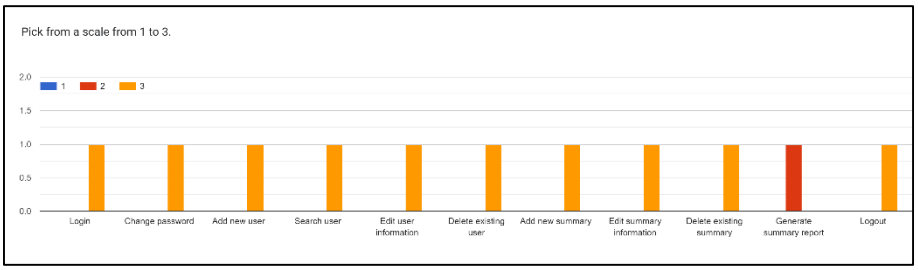


Figure 30: Feedback from an administrator

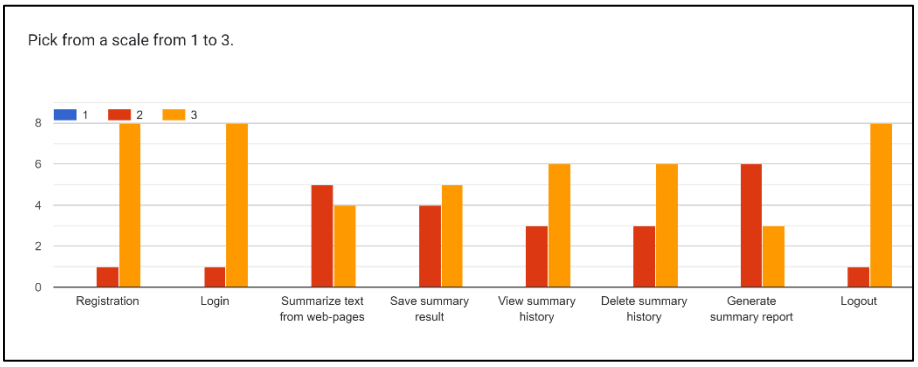


Figure 31: Feedback from students

Based on the figure, out of the 9 student responses, 8 students rated "Register the system" as functioning well, while only a student rated "Register the system" as functioning. When it came to "Login the system", it had the same result as "Register the system". For the administrator, the result indicated that "Login the system" and "Change password" were functioning well.

Next, the administrator stated that managing users, which involves adding new users, searching, or editing users, and deleting existing users, were functioning well.

Furthermore, there were 4 students rated "Summarise text from web-pages" as functioning well, while there are 5 students rated "Summarise text from web-pages" as functioning. For the administrator, the result indicated that managing summaries, which involves adding new summary, editing summary, and deleting summary were functioning well.

In addition, 4 students rated "Summarise text from web-pages" as functioning well, while there were 5 students rated "Summarise text from web-pages" as functioning. For the administrator, the result indicated that managing summaries, which involves adding new summary, editing summary, and deleting summary were functioning well. Lastly,

Moreover, there were 5 students rated "Save summary result" as functioning well, while 4 students rated "Save summary result" as functioning. When it came to "View summary history" and "Delete summary history", there were 6 students rated them as functioning well and 3 students rated them as functioning.

Lastly, there were 3 students rated "Generate report" as functioning well, while there were 6 students rated "Generate report" as functioning. For the administrator, "Generate report" was rated as functioning.

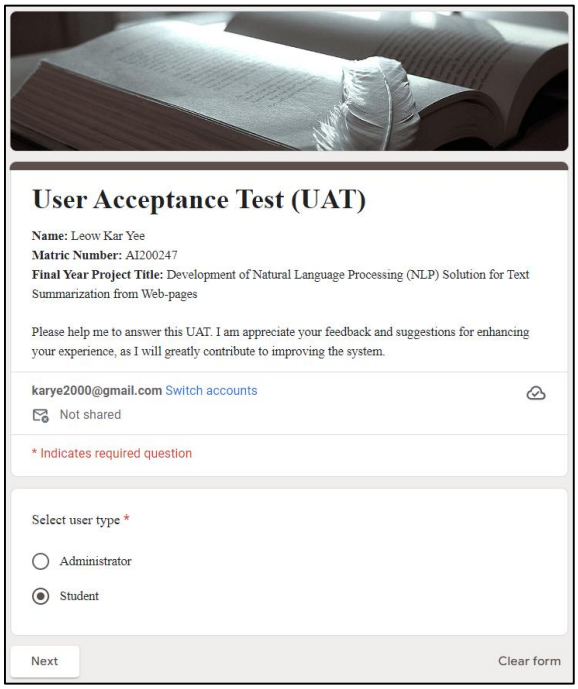
5. Conclusion

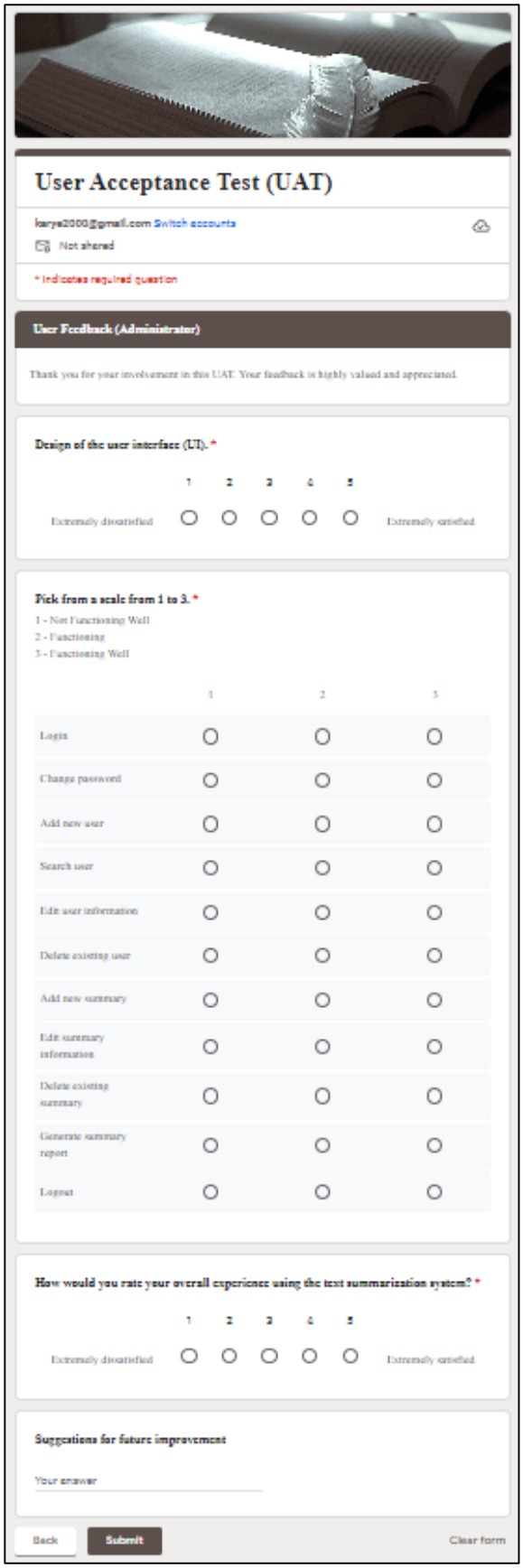
A Natural Language Processing (NLP) solution for text summarization from web-pages is developed at the end of this project. The system has several advantages for students, including enhanced efficiency, streamlined information access, and a simplified work process. However, the system has limitations that should be considered. It may struggle with accurately interpreting nuances and complexities of natural language, loss of context during summarization, and handling complex or ambiguous texts. For future work, the system can focus on enhancing linguistic comprehension, improving contextual understanding, and integrating the multi-modal information. By recognizing these limitations and investing in future research and development, the system can continue to evolve and provide valuable tools for students in their academic endeavours.

Acknowledgment

The author would like to express their gratitude to the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for their support and encouragement throughout the process of conducting this study.

Appendix





User Acceptance Test (UAT)

karya2000@gmail.com Switch accounts

Not shared

* Indicates required question

User Feedback (Administrator)

Thank you for your involvement in this UAT. Your feedback is highly valued and appreciated.

Design of the user interface (UI). *

	1	2	3	4	5	
Extremely dissatisfied	○	○	○	○	○	Extremely satisfied

Pick from a scale from 1 to 3. *

1 - Not Functioning Well
2 - Functioning
3 - Functioning Well

	1	2	3
Login	○	○	○
Change password	○	○	○
Add new user	○	○	○
Search user	○	○	○
Edit user information	○	○	○
Delete existing user	○	○	○
Add new summary	○	○	○
Edit summary information	○	○	○
Delete existing summary	○	○	○
Generate summary report	○	○	○
Logout	○	○	○

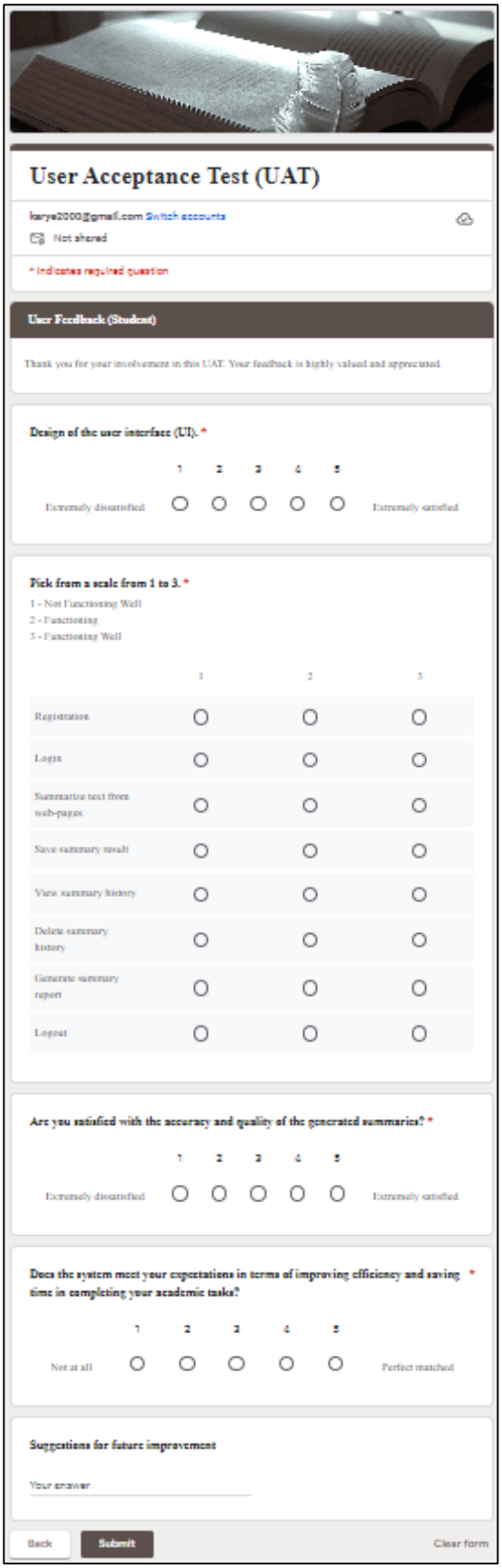
How would you rate your overall experience using the text summarization system? *

	1	2	3	4	5	
Extremely dissatisfied	○	○	○	○	○	Extremely satisfied

Suggestions for future improvement

Your answer

Back Submit Clear form



User Acceptance Test (UAT)

karya2000@gmail.com Switch accounts

Not shared

* Indicates required question

User Feedback (Student)

Thank you for your involvement in this UAT. Your feedback is highly valued and appreciated.

Design of the user interface (UI). *

	1	2	3	4	5	
Extremely dissatisfied	○	○	○	○	○	Extremely satisfied

Pick from a scale from 1 to 3. *

1 - Not Functioning Well
2 - Functioning
3 - Functioning Well

	1	2	3
Registration	○	○	○
Login	○	○	○
Summarize text from web-pages	○	○	○
Save summary result	○	○	○
View summary history	○	○	○
Delete summary history	○	○	○
Generate summary report	○	○	○
Logout	○	○	○

Are you satisfied with the accuracy and quality of the generated summaries? *

	1	2	3	4	5	
Extremely dissatisfied	○	○	○	○	○	Extremely satisfied

Does the system meet your expectations in terms of improving efficiency and saving time in completing your academic tasks? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	Perfect matched

Suggestions for future improvement

Your answer

Back Submit Clear form

References

[1] Jason Brownlee (2017, September 22). What is Natural Language Processing? Retrieved from https://machinelearningmastery.com/natural-language-processing/

[2] Ogi Djuraskovic (2022, September 28). How Many Websites Are There? – The Growth of The Web (1990–2022). Retrieved from https://firstsiteguide.com/how-many-websites/#:~:text=question%20people%20ask%3A-,How%20many%20websites%20are%20there%3F,over%201.98%20billion%20websites%20online.)

[3] Fitria, T. N. (2021). QuillBot as an online tool: Students' alternative in paraphrasing and rewriting of English writing. Englisia: Journal of Language, Education, and Humanities, 9(1), 183-196.

[4] Zhao, X. (2022). Leveraging Artificial Intelligence (AI) Technology for English Writing: Introducing Wordtune as a Digital Writing Assistant for EFL Writers. RELC Journal, 00336882221094089.

[5] Mahajan, D., Sawane, D., Mahendar, C. N., & BAMU, A (2016). Story Summarization using Text Compactor.

[6] Alexandra Cote (2022, October 10). Project Management Methods, Methodologies, and Frameworks – A Guide for Beginners. Retrieved from https://www.paymoapp.com/blog/project-management-methodologies/

[7] Indeed Editorial Team (2022, April 15). What Is System Analysis and Design? (Plus Benefits). Retrieved from https://www.indeed.com/career-advice/career-development/what-is-system-analysis-and-design#:~:text=System%20analysis%20and%20design%20is,related%20to%20growth%20and%20profitability.