Smart Content Accumulator

Final Year Project

Session 2019-2023

A project submitted in partial fulfillment of the degree of

BS in Software Engineering

Department of Software Engineering

Faculty of Computer Science & Information Technology

The Superior University, Lahore

Type (Nature of project)	[✓] **D**evelopment [ ] **R**esearch [ ] **R&D**
Area of specialization	
FYP ID	FYP-BCSM-F22-054

Project Group Members

Sr.#	Reg. #	Student Name	Email ID	*Signature
(i)	Bcsm-f19-246	Huzaifa Tahir	Bcsm-f19-246@superior.edu.pk	
(ii)	Bcsm-f19-266	Shuja Ul Hassan	Bcsm-f19-266@superior.edu.pk	
(iii)	Bcsm-f19-258	Umair Ali	Bcsm-f19-258@superior.edu.pk	

*The candidates confirm that the work submitted is their own and appropriate credit has been given where reference has been made to work of others

Plagiarism Free Certificate

This is to certify that, I Huzaifa Tahir S/D of Muhammad Tahir, group leader of FYP under registration no Bcsm-f19-246 at Computer Science Department, The Superior College, Lahore. I declare that my FYP report is checked by my supervisor.

Date: Name of Group Leader: Huzaifa Tahir Signature: ______________

Name of Supervisor: Mr. Nouman Jazeb

Designation: Senior Lecturer

Signature: _________________

Co-Supervisor: N/A

Designation: N/A

Signature: __________________

HoD: Dr. Irfan-ud Din

Signature: ________________

[Title of Project]

Change Record

Author(s)	Version	Date	Notes	Supervisor's Signature
	1.0		<Original Draft>	
			<Changes Based on Feedback from Supervisor>	
			<Changes Based on Feedback From Faculty>	
			<Added Project Plan>	
			<Changes Based on Feedback from Supervisor>	

APPROVAL

PROJECT SUPERVISOR

Comments: ____

____

Name:______________________________

Date:_________________________________ Signature:___________________________

PROJECT MANAGER

Comments: ____

____

Date:_________________________________ Signature:___________________________

HEAD OF THE DEPARTMENT

Comments: ____

____

Date:_________________________________ Signature:___________________________

Dedication

This project is dedicated to our parents who have never failed to give us financial and moral support, for giving all our needs during the time we developed our system and for teaching us that even the largest task can be accomplished if it is done one step as a time. We dedicate this Project to all the people who have worked hard to help us complete this project.

Acknowledgements

We thank Allah SWT for his blessings, keeping us safe and in good health throughout our development period. We are deeply grateful to our Supervisor Mr. Nouman Jazeb of the Superior University Lahore for their valuable guidance, cooperation and assistance throughout the whole period that we have been engaged in this development. It is our pleasure to appreciate the team members for their useful academic interaction and devoted time they put in the completion of this project. Finally, with great pleasure, we thank my loving parents for their love and prayers.

Executive Summary

Smart Content Accumulator is a website that offers a useful service to people who need to summarize large amounts of text quickly and efficiently. Whether you are a student, a researcher, or just someone who needs to stay informed about current events, this website can help you to digest large amounts of information in a meaningful way.

The way the site works is quite simple. All you need to do is copy and paste the URL of the web page you want to summarize into the text box on the Smart Content Accumulator website. The site's software will then analyze the text and generate a summary of the most important points. The summary is usually a few paragraphs long, and it is designed to give you a quick overview of the text without having to read through the entire article or document.

Overall, Smart Content Accumulator is a fantastic website that offers a valuable service to people who need to summarize large amounts of text quickly and accurately. Whether you are a student, a researcher, or just someone who wants to stay informed about current events, this website can help you to save time and make more informed decisions. Its accuracy, ease of use, and affordability make it an excellent tool for anyone who needs to summarize text on a regular basis.

Table of Contents

Smart Content Accumulator i
Final Year Project i
Session 2019-2023 i
BS in Software Engineering i
*The candidates confirm that the work submitted is their own and appropriate credit has been given where reference has been made to work of others ii
Plagiarism Free Certificate ii
Dedication v
Acknowledgements vi
Executive Summary vii
Table of Contents viii
List of Figures x
List of Tables x
Chapter 1 1
Introduction 1
1.1. Background 3
1.2. Motivations and Challenges 3
1.3. Goals and Objectives 4
1.4. Literature Review/Existing Solutions 5
1.5. Gap Analysis 6
1.6. Proposed Solution 6
1.7. Project Plan 7
1.7.1. Work Breakdown Structure 7
1.7.2. Roles & Responsibility Matrix 8
1.7.3. Gantt Chart 9
Chapter 2 11
Software Requirement Specifications 11
2.1. Introduction 12
2.1.1. Purpose 12
2.1.2. Document Conventions 12
2.1.3. Intended Audience and Reading Suggestions 12
2.1.4. Product Scope 12
2.2. Overall Description 13
2.2.1. Product Perspective 13
2.2.2. User Classes and Characteristics 13
2.2.3. Operating Environment 14
2.2.4. Design and Implementation Constraints 14
2.2.5. Assumptions and Dependencies 15
2.3. External Interface Requirements 15
2.3.1. User Interfaces 15
2.3.2. Hardware Interfaces 15
2.3.3. Software Interfaces 16
2.3.4. Communications Interfaces 16
2.4. System Features 16

2.4.1. User Registration 16
2.4.1.1. Description and Priority 16
2.4.1.2. Stimulus/Response Sequences 16
2.4.1.3. Functional Requirements 17
2.4.2. User Login 17
2.4.2.1. Description and Priority 17
2.4.2.2. Stimulus/Response Sequences 17
2.4.2.3. Functional Requirements 17
2.4.3. Content Summarization: 17
2.4.3.1. Description and Priority 17
2.4.3.2. Stimulus/Response Sequences 17
2.4.3.3. Functional Requirements 17
2.4.4. Interaction 18
2.4.4.1. Description and Priority 18
2.4.4.2. Stimulus/Response Sequences 18
2.4.4.3. Functional Requirements 18
2.4.5. Admin Panel 18
2.4.5.1. Description and Priority 18
2.4.5.2. Stimulus/Response Sequences 18
2.4.5.3. Functional Requirements 18
2.4.6. Profile Management 19
2.4.6.1. Description and Priority 19
2.4.6.2. Stimulus/Response Sequences 19
2.4.6.3. Functional Requirements 19
2.5. Nonfunctional Requirements 19
2.5.1. Performance Requirements 19
2.5.2. Safety Requirements 19
2.5.3. Security Requirements 19
2.5.4. Usability Requirements 20
2.5.5. Reliability Requirements 20
2.5.6. Maintainability/Supportability Requirements 20
2.5.7. Portability Requirements 20
2.5.8. Efficiency Requirements 21
2.6. Domain Requirements 21
Chapter 3 22
Use Case Analysis 22
3.1. Use Case Model 23
3.2. Use Cases Description 24
Chapter 4 28
System Design 28
4.1. Architecture Diagram 29
4.2. Entity Relationship Diagram with data dictionary 30
4.3. Class Diagram 31
4.4. Sequence / Collaboration Diagram 32
4.5. Activity Diagram 34
4.6. State Transition Diagram 36

4.7. Component Diagram 37
4.8. Deployment Diagram 38
4.9. Data Flow diagram 39
Chapter 5 41
Implementation 41
5.1. Important Flow Control/Pseudo codes 42
5.2. Components, Libraries, Web Services and stubs 42
5.3. Deployment Environment 43
5.4. Tools and Techniques 43
5.5. Best Practices / Coding Standards 44
5.6. Version Control 45
Chapter 6 46
Testing and Evaluation 46
6.1. Use Case Testing 47
6.2. Equivalence partitioning 49
6.3. Boundary value analysis 50
6.4. Data flow testing 51
6.5. Unit testing 52
6.6. Integration testing 53
6.7. Performance testing 53
6.8. Stress Testing 54
Chapter 7 55
Summary, Conclusion and Future Enhancements 55
7.1. Project Summary 56
7.2. Achievements and Improvements 56
7.3. Critical Review 58
7.4. Lessons Learnt 59
7.5. Future Enhancements/Recommendations 61
Reference and Bibliography 64

List of Figures

1.1 Caption of first figure of first chapter 6
1.2 Caption of second figure of first chapter 7
2.1 Caption of first figure of second chapter 14
2.2 Caption of second figure of second chapter 22
2.3 Caption of third figure of second chapter 26
5.1 Caption of first figure of fifth chapter 49
5.2 Caption of second figure of fifth chapter 49

List of Tables

1.1	label of first table of first chapter	6
1.2	label of second table of first chapter	7
2.1	label of first table of second chapter	14
2.2	label of second table of second chapter	22
2.3	label of third table of second chapter	26
5.1	label of first table of fifth chapter	49
5.2	label of second table of fifth chapter	49

Chapter 1

Introduction

Chapter 1: Introduction

Welcome to our innovative website! We have developed a powerful tool that streamlines the process of obtaining summarized content from any article. With just a URL and a click of a button, our website generates concise and meaningful summaries, saving you valuable time and effort.

In today's fast-paced digital era, an overwhelming amount of information is generated and published daily across various online platforms. It can be a daunting task to sift through lengthy articles and extract the key points that are most relevant to you. This is where our website comes in.

Text summarizers have gained immense importance and widespread usage due to the exponential growth of online content. They play a crucial role in distilling large amounts of information into concise and digestible summaries [1].

Our website offers a convenient and efficient way to extract the essence of an article by providing accurate and concise summaries. By simply entering a URL into our search bar and clicking the summarize button, you can access the key points and main ideas of any article within seconds.

We understand the importance of time and the need for quick and accessible information. With our text summarizer, we aim to empower users with the ability to navigate the vast landscape of online content effortlessly. Whether you are a student, professional, or avid reader, our website is your gateway to simplified and efficient information consumption.

Experience the power of our text summarizer and discover how it can revolutionize the way you extract insights from articles. Say goodbye to information overload and hello to concise and meaningful summaries, right at your fingertips [2].

1.1. Background

The need for content accumulation arises from the ever-increasing volume of information available, particularly with the advent of the internet and the rapid growth of digital content. With vast amounts of information being produced daily, it becomes impractical and time-consuming for individuals to manually read and comprehend all the available text. Text summarizers offer an efficient solution by automatically condensing text into shorter summaries, allowing users to quickly grasp the main content without having to read the entire text [2].

Over the years, numerous research studies and advancements in NLP have contributed to the development of sophisticated text summarization techniques. The rise of deep learning models, such as recurrent neural networks (RNNs) and transformer models like BERT and GPT, has significantly improved the quality and fluency of generated summaries [13]. These models can effectively capture the semantic and contextual relationships within a text, enabling them to produce more accurate and human-like summaries.

1.2. Motivations and Challenges

Information Overload: The exponential growth of digital content has led to information overload, making it difficult for individuals to keep up with the sheer volume of text. Text summarization provides an efficient way to extract key information from large amounts of text, enabling users to quickly grasp the main points without spending excessive time on reading.

Time Efficiency: Summarization saves time by condensing lengthy texts into shorter summaries. It allows users to scan multiple documents or articles to get an overview of their content, helping them prioritize their reading list and focus on the most relevant information.

Content Digestion: Summarization aids in comprehending complex or technical texts by breaking them down into concise and easily understandable summaries. It enhances the ability to grasp the main ideas, arguments, or conclusions of a text, making it beneficial for educational purposes and knowledge acquisition.

1.3. Goals and Objectives

Information Extraction: The primary goal of text summarization is to extract the most important and relevant information from a given text. The objective is to distill the key ideas, facts, arguments, or conclusions and present them in a concise and coherent summary.

Conciseness and Brevity: Summarization aims to provide a condensed version of the original text, ensuring that the summary is concise and to the point. The objective is to eliminate unnecessary details, redundancies, and repetitions while retaining the essence of the text.

Coherence and Readability: Summaries should maintain coherence and readability, enabling users to understand the main points without having to refer back to the original text. The objective is to ensure that the summary flows logically, uses appropriate language, and effectively communicates the core message.

Content Preservation: While summarizing, it is crucial to preserve the most important content and ensure that the summary accurately represents the original text. The objective is to capture the main ideas, arguments, or conclusions faithfully, avoiding distortion or misrepresentation.

Language and Style Preservation: The goal is to preserve the language style, tone, and nuances of the original text in the summary. This objective becomes especially relevant when summarizing literary works, creative writing, or content where the author's voice and style are essential.

Time Efficiency: Summarization aims to save time for users by providing an efficient way to obtain the key information from a text. The objective is to allow users to quickly grasp the main content without investing excessive time in reading lengthy documents.

Real-world Applications: Text summarization aims to have practical applications across various domains, including news aggregation, document summarization, automated content generation, chatbots, search engines, and more. The objective is to develop robust summarization systems that can assist users in managing information overload, decision-making, and knowledge acquisition.

1.4. Literature Review/Existing Solutions

Different sites present the same content in various ways. Most of the time people don't understand what they are looking for because so much data is present on the same topic. Hence; it is pretty hard to find the most precise and summarized content in one place. Content and News accumulators made waves on websites because they solve this exact problem [3]. They accumulate data to find the most relevant search results for you but their limitation is that they present in very specific ways.

Content and News accumulators are web applications that help you find the most relevant content on the internet by scraping data from various sources and presenting it to you in one place [4]. They make use of different algorithms to determine which content is most relevant to your search query. Content and News accumulators have become very popular in recent years, with many different companies offering their own versions of the service.

There are different types of content accumulators such as AllTop, The Web List and Techmeme. Google Alerts. Each one of them has their own unique features and functionalities. AllTop and The Web List scrape the internet to find the most relevant data on a topic but they only present it in the form of a sales copy. Similarly, Google Alerts searches the web to find the most relevant news and send it to the user as an alert. Feedly is another content accumulator which searches online news directories to find the most relevant data to a user's query but it is limited to search on twitter [10].

News accumulators are a great way to stay up-to-date with the latest news and information on your topic of interest but the current choices are very limited to the platforms they are operating on. In this research, we are looking to automate a Content Accumulator which analyzes data and produces results from across the web as opposed to specific platforms [3].

1.5. Gap Analysis

2015 was when the digital industry witnessed the rise of content accumulators. Because many publishers perceived syndication as a sin, they rejected accumulators and instead chose Google News, Yahoo News and AOL. Also, none of them wanted to dip into their budget, so instead created free news searches for Huffington Post, BuzzFeed, Vine et al [10].

The audience sizes of these accumulator sites were small, but they were seen to be having the capacity to grow rapidly. Trying to accomplish the common goal of 'being everywhere your readers are', many publishers started relying on these accumulators completely, and unfortunately, instead of finding a better solution to reach their audience, they started giving their content to the accumulators for free.

In return, they got a huge number of views on content but not much ROI.

When it comes to revenue generation, which is a big challenge for digital publishers, the only option with content accumulators is to bet on the subscription models and nothing else. Which is definitely not a win-win situation for digital news publishers [10].

It goes without saying that content discovery has changed significantly in the past decade, but the method of content consumption has also changed drastically. Now consumers expect a frictionless delivery and high-quality, shareable, searchable and readily-available content in return for their interest.

1.6. Proposed Solution

In order to overcome the above problems, content accumulators are used. With the vast amount of textual information available, it is impractical for individuals to read every document in its entirety. Text summarizers provide a time-efficient solution by condensing lengthy texts into shorter summaries, allowing users to quickly grasp the main points and key information without investing excessive time. The digital era has led to information overload, making it challenging to filter through a massive volume of text. Text summarizers help users manage information overload by extracting the most important and relevant content from multiple sources, enabling them to stay informed without being overwhelmed.

Advantages of Content Accumulator:

- Pulls meaningful information from any URL provided
- Automatic update of the data.
- Timely delivery of recent information.
- Users save time and energy.
- Python is simple to find out and use.
- Powerful filtering capabilities.

1.7. Project Plan

We will divide our project into different parts, some parts will be done by individual members, while other parts will be done together. For example, the front-end, back-end will all be done together which will include web design, and development. In order to start working, we need to figure out what we need to do to make this project work. The design of the user interface is important. UI is the part of the website that users see and can interact with. UI design will be completed in around 7-10 days while front-end development can take up-to 40-45 days. Back-End development will take up some time. We hope to have it done in 50-60 days. The final step is testing and implementation, which is planned to happen within the next 20-30. The platform will be given to students and teachers for testing so that feedback can be collected as quickly as possible.

1.7.1. Work Breakdown Structure

1. **Preparation**
    - Questionnaire
    - Requirement Analysis
    - Feasibility study

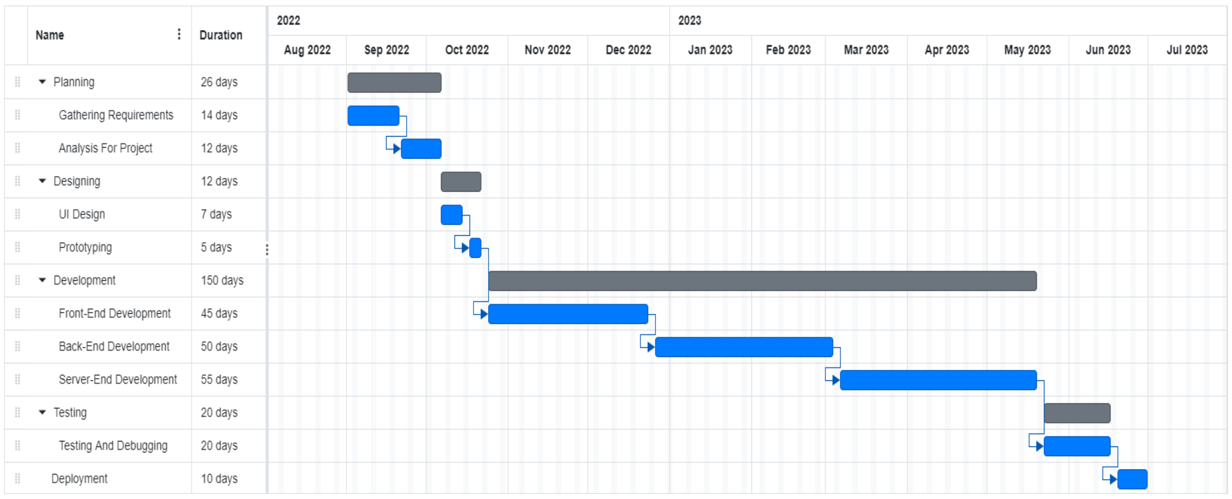
2. **Planning**
   - Documentation
   - Finalization of deliverables
   - Scheduling the project
3. **Design**
   - User Interface Design in Figma
4. **Development**
   - Front-end development
   - Back-end development
   - Server-end Development
   - Integration of back-end with front-end
5. **Testing**
   - Functional testing
   - Non-Functional Testing.

1.7.2. Roles & Responsibility Matrix

WBS #	WBS Deliverable	Activity #	Activity to Complete the Deliverable	Duration (# of Days)	Responsible Team Member(s) & Role(s)
01	Gathering Requirements	1	Technical Points, Problems, Solutions	14	Shuja Ul Hassan Huzaifa Tahir
02	Analysis For Project	2	Prime Factors to Understand the User Needs etc.	12	Shuja Ul Hassan Huzaifa Tahir Umair Ali
03	User Interface Designing	3	Designing The User Interface for The Website	7	Shuja Ul Hassan Huzaifa Tahir Umair Ali

04	Prototype Stage	4	Prototyping The Web Design Once Approved	5	Shuja Ul Hassan Huzaifa Tahir Umair Ali
05	Front-End Development	5	Getting Into the Front-End Dev	45	Shuja Ul Hassan Huzaifa Tahir Umair Ali
06	Back-End Development	6	Backed-End Dev After Front-End Dev	50	Huzaifa Tahir
07	Server-End Development	7	Server-End Dev Along with Back-End Dev	55	Huzaifa Tahir
08	Testing And debugging	8	Test Module and Build Complete Website	20	Shuja Ul Hassan Huzaifa Tahir Umair Ali
9	Deployment	9	Deploying The Website for Use	10	Huzaifa Tahir
10	Documentation	10	Presenting Final Report	20	Shuja Ul Hassan Huzaifa Tahir

1.7.3. Gantt Chart



Name	Duration
Planning	26 days
Gathering Requirements	14 days
Analysis For Project	12 days
Designing	12 days
UI Design	7 days
Prototyping	5 days
Development	150 days
Front-End Development	45 days
Back-End Development	50 days
Server-End Development	55 days
Testing	20 days
Testing And Debugging	20 days
Deployment	10 days

Chapter 2

Software Requirement Specifications

Chapter 2: Software Requirement Specifications

2.1. Introduction

2.1.1. Purpose

The main purpose of this document is to provide a detailed description of a Smart Content Accumulator that collects data from different sources and displays the most accurate and precise data in one place so viewers can get a complete understanding of the topic they're researching.
A Content Accumulator reads multiple articles on the web and has an ML algorithm that determines the most insightful.

2.1.2. Document Conventions

- All SRS documents must be written in an easily readable font such as Calibri, size 12.
- Headings must be highlighted in bold, and subheadings must be italicized.
- All requirement statements must include a priority, which must be included with the statement itself and cannot be assumed to be inherited by any of the detailed requirements
- All diagrams must be drawn with the Unified Modeling Language (UML).
- All technical terms must be in italics and must be defined in the glossary.

2.1.3. Intended Audience and Reading Suggestions

This SRS describes how the content from different sources can be scrapped and shown in the form of summarized text using Machine Learning. It is primarily written for the developers of the READ Project and future references as well as the software users.

2.1.4. Product Scope

The goal is to produce a platform (Smart Content Accumulator) that collects data from different sources and displays the most accurate and precise data in one place so viewers can get a complete understanding of the topic they're researching. Users can get also get recommendations of latest or trending news. This platform can be used by Content creators to expose their works to broader or new communities and to get increased visibility for their content and also by Digital Marketers to improve their digital content marketing strategies by distributing content on multiple platforms to gain exposure to a wider audience.

2.2. Overall Description

2.2.1. Product Perspective

The initiate idea was to develop a program able to scrap and summarize the content from different articles provided by the different sources for a given period of time and transform the content to easily read focal points. Eventually, the program should be able to

- Analyze a gigantic body of text often exceeding millions of characters,
- bring about country-dependent focal points on the basis of word frequencies, repetitions and natural emphasis in the written language, and
- subsequently, visualize the findings in a sophisticated and explanatory way.

Content aggregation is simply curating. In other words, it is the collection of relevant data covering the same topic, and displaying it all in one place for easy access and reference by users. This is similar to a single large search engine where only essential information gets collected while all material that is irrelevant or of low-quality is weeded out. It is important to note that this is different from plagiarism.

2.2.2. User Classes and Characteristics

2.2.2.1. Content Providers:

These users produce content for the accumulator, such as images, videos, articles, and other digital media. They have the ability to upload their content to the platform, as well as modify and delete it.

2.2.2.2. Content Consumers:

These users consume content from the accumulator, such as images, videos, articles, and other digital media. They have the ability to view, search, and download content from the platform.

2.2.2.3. Moderators:

These users are responsible for monitoring the accumulator and ensuring that its content follows the platform's guidelines. They have the ability to delete, edit, and hide content.

2.2.2.4. Advertisers:

These users use the accumulator to promote their own products or services. They have the ability to create and manage campaigns, as well as track the performance of their campaigns.

2.2.3. Operating Environment

Operating environment for the Smart Content Accumulator is given below.

- **Operating system:** Windows, IOS, Android
- **Database:** SQLite3
- **Language:** Python, HTML5, CSS3, SQL

2.2.4. Design and Implementation Constraints

2.2.4.1. Design Constraints:

- The system must be able to accumulate content from multiple sources and store it in a structured way.
- The system must be able to filter content according to certain criteria.
- The system must be able to store a variety of content types.
- The system must be able to update content as it changes in the source

2.2.4.2. Implementation Constraints:

- It must be able to work with a variety of data sources (e.g., databases, web services, etc.).
- It must be able to integrate with existing systems, such as content management systems.
- It must be able to handle high volume of data.
- It must be able to support large-scale distributed systems.

2.2.5. Assumptions and Dependencies

2.2.5.1. Assumptions:

- It will be developed using existing software components.
- The development and/or operating environment is stable.
- The smart content accumulator will have the necessary resources and capabilities to meet the requirements outlined in the SRS.
- Third-party components will be compatible with the existing environment

2.2.5.2. Dependencies:

- Access to external software components for reuse.
- The availability of third-party components to meet the requirements outlined in the SRS.
- A stable development and/or operating environment.

2.3. External Interface Requirements

2.3.1. User Interfaces

The user interface design of a platform is crucial in regard to if a user decides to choose a system or not. Thus, it is of importance to research what makes a website appealing to the user. In order to make design choices general requirements based on literature and design ideas of related work are going to be discussed. Our user interface is as easy to use as it looks.

2.3.2. Hardware Interfaces

The hardware requirements at the user end are very simple. The web application can run on the hardware that can run a basic simple browser, although the hardware should be good enough to run the website during the peak times for the web servers. The mobile hardware interface for the user would be any smart phone having android OS and IOS with chrome installed in it so user can access the website.

2.3.3. Software Interfaces

The software interface is the point of communication between the user and the software. This interface is used to communicate the user's input and the software's output, and is often the primary way a user interacts with the software. The software interface connects the software to other components such as databases, operating systems, tools, libraries, and integrated commercial components. The data items and messages coming into the system can include user inputs, data from databases, and requests from other components. The data items and messages going out can include responses to user requests, data being sent to databases, and data being sent to other components. The services needed and the nature of communications is dependent on the specific components being connected and the type of data being shared. The software interface must adhere to any applicable programming interface protocols and must be designed to facilitate the sharing of data between components.

2.3.4. Communications Interfaces

It can use a variety of communication interfaces to send and receive data. These include e-mail, web browsers, network servers, electronic forms, and more. Depending on the type of communication, different protocols, message formatting, and communication standards may be required. For example, FTP or HTTP might be used to establish connections. Additionally, any security measures should also be outlined to ensure data privacy and integrity.

2.4. System Features

2.4.1. User Registration

2.4.1.1. Description and Priority

User can register himself by providing the information such as name, email, password.

2.4.1.2. Stimulus/Response Sequences

Once user submits the required information, he/she will be registered after going through the validation process.

2.4.1.3. Functional Requirements

REQ-SF1-1: User must input his/her name, email and password.

REQ-SF1-2: This data of user will be stored in the database.

2.4.2. User Login

2.4.2.1. Description and Priority

This feature allows the user to login to his/her account using the registered email and password.

2.4.2.2. Stimulus/Response Sequences

Once user inputs the required fields, he/she will be logged in to account after validations.

2.4.2.3. Functional Requirements

REQ-SF2-1: User must input the registered email and password.

REQ-SF2-2: Entered data will be validated from the database.

REQ-SF2-3: If the data is valid then operation will be successful, otherwise user will go back to the login page.

2.4.3. Content Summarization:

2.4.3.1. Description and Priority

This feature allows users to Summarize the text by providing the URL of the desired article or web page.

2.4.3.2. Stimulus/Response Sequences

Once user provides URL, system will scrap the data from the source and display the data in a summarized form.

2.4.3.3. Functional Requirements

REQ-SF2-1: User must enter the correct URL in the search bar.

REQ-SF2-2: Once user enters the correct URL, data will be shown.

2.4.4. Interaction

2.4.4.1. Description and Priority

Allows user to interact with the content by saving it for later user.

2.4.4.2. Stimulus/Response Sequences

When user saves some specific content then he/she can view the saved content later only if he/she is logged in to the account.

2.4.4.3. Functional Requirements

REQ-SF2-1: User can save the content for later use.

2.4.5. Admin Panel

2.4.5.1. Description and Priority

This module allows the admin to manage registered accounts by removing, adding the existing accounts by logging in to the admin panel. Moreover, it also allows the admin to manage the content by adding or removing.

2.4.5.2. Stimulus/Response Sequences

Admin can login to panel to manage all the modules by going into each module section.

2.4.5.3. Functional Requirements

REQ-SF2-1: Must have credentials to have access to panel.

REQ-SF2-2: Can view, remove existing registered accounts.

REQ-SF2-3: Can add new users.

REQ-SF2-4: Can view, remove and add content.

2.4.6. Profile Management

2.4.6.1. Description and Priority

This module allows the user to update the current information on the profile page. User can edit name, password and email.

2.4.6.2. Stimulus/Response Sequences

User must provide valid information to update the information in the database.

2.4.6.3. Functional Requirements

REQ-SF2-1: Must be logged in to the account.

REQ-SF2-2: Will be able to update name, email and password.

2.5. Nonfunctional Requirements

2.5.1. Performance Requirements

The Accumulator must be able to store and retrieve data from the database quickly. This ensures that the system can quickly access data from the database and provide the necessary information to users in a timely manner. It must respond to requests within 10ms and ensures that the system can respond quickly to user requests and provide a smooth user experience.

2.5.2. Safety Requirements

The system must include a system for monitoring the temperature of the device to prevent overheating. It must be designed with a sufficient number of emergencies shut-off switches to ensure that the system can be powered down in case of emergency.

2.5.3. Security Requirements

All user data must be stored in a secure, encrypted format. User identity authentication must be implemented to ensure only authorized users have access to the system. All data must be securely stored and must be kept only for as long as is necessary. All user data must be protected from unauthorized access and must not be shared with any third parties without the user's consent.

2.5.4. Usability Requirements

- The Smart Content Accumulator should be able to collect content from a variety of sources and provide a way to view the content in a centralized location.
- It should allow users to filter content by topic, date, or other criteria.
- It should provide a way to save or share favorite content.
- It should be easy to use and navigate.
- It should work on a variety of devices, including computers, tablets, and phones.

2.5.5. Reliability Requirements

- The system must be able to detect and recover from any data corruption.
- The system must have redundant backups of the data in case of hardware or software failure. The system must have robust failure detection and recovery mechanisms.
- The system must be able to handle large volumes of data without any degradation in performance.
- The system must be able to automatically synchronize data across multiple servers.
- The system must be able to provide real-time updates of the content being accumulated.

2.5.6. Maintainability/Supportability Requirements

It has the ability to add, remove, and update content sources. Provides support for all content sources. It is easy to maintain and be able to be supported by a small team. It has ability to effectively remove stale or outdated content.

2.5.7. Portability Requirements

It runs on a variety of devices, operating systems and web browsers. Additionally, will be able to synchronize with other devices and systems in order to keep content up-to-date. It is able to export content in a variety of formats, so that it can be easily shared with others.

2.5.8. Efficiency Requirements

It quickly and easily finds, collect, and organize content from a variety of sources. It identifies new and trending topics, and be able to suggest relevant content to users. Additionally, it is able to track user engagement with content and provides analytics to content creators.

2.6. Domain Requirements

2.6.1 Domain Requirements:

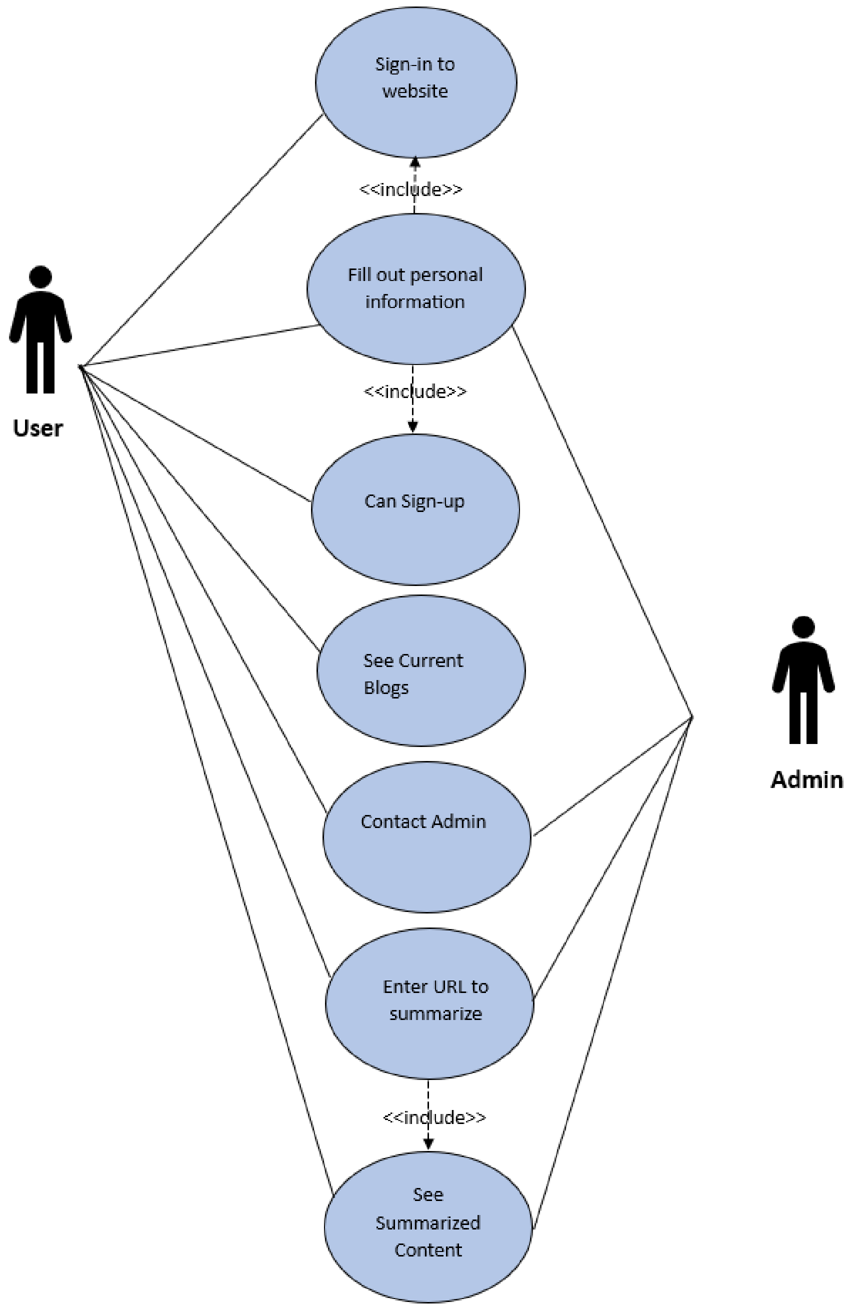
- The smart content accumulator must be able to gather content from a variety of sources, including webpages, social media, and other digital sources.
- The content must be correctly attributed to the original author and copyright holder.
- The collected content must be properly indexed and organized for easy access.
- The accumulator should have an intuitive user interface that allows the user to quickly find and access the desired content.
- The accumulator should be able to recognize and ignore duplicate content.
- The accumulator should have permission-based user access and authentication.

Chapter 3
Use Case Analysis

Chapter 3: Use Case Analysis

The use-case diagram tells us the high-level functions, users and scope of a system. The use-case diagrams also figure out the interactions between the system and its actors. These diagrams define the context and requirements of entire system.

3.1. Use Case Model



3.2. Use Cases Description

Use Case description of content accumulator represents features and functions of a system that how they interact with the user. It helps to identify the user requirements and the system design. The main use case of this system is for users to search for content from various sources, such as websites, social media, search engines, etc. Content is then collected and presented in a unified manner, allowing users to easily access and manage their content. Some description of use case models is given below:

Use Case 1	**Sign up**
Scope	Allows users to create their own account.
Primary Actors	User.
Stake Holders	User.
Pre-conditions	Must input valid information such as name, email, password.
Post-conditions	Should be able to sign up without problems.
Main Success Scenario	• The website will require user's personal information such as name, email and password in order to create a new account. • Once the user inputs all the fields correctly, it will allow the users to create a new account. • The personal information of the users will be stored in the database.
Extensions	If the entered information is not correct then user will not be able to create a new account.

Use Case 2	**Login**
Scope	Allows users to login to their accounts.
Primary Actors	User.
Stake Holders	User.
Pre-conditions	User must have signed up his/her new account before logging in.
Post-conditions	Should be able to login without problems.
Main Success Scenario	• The website will ask for signed up email and password from the user. • Once the user inputs all the fields correctly, it will allow the users to login to their account.
Extensions	If the entered information is not correct then user will not be able to create a new account.

Use Case 3	**Text Summarization**
Scope	Allows users to summarize content from multiple sources.
Primary Actors	User.
Stake Holders	User.
Pre-conditions	User must have knowledge about what his/her search is going to be.
Post-conditions	Should be able to find the required information without any problems.

Main Success Scenario	• User will be able to summarize the content by pasting the URL of wanted article or blog.
Extensions	If the provided URL is not correct then it may not be able to summarize the content.

Use Case 4	**Interaction**
Scope	Allows users to interact with the content by saving it.
Primary Actors	User.
Stake Holders	User.
Pre-conditions	User must have knowledge about what his/her search is going to be.
Post-conditions	Should be able to find the required information without any problems.
Main Success Scenario	• User can interact with the search data by sharing it.
Extensions	None

Use Case 5	**Accounts Management**
Scope	Allows Admin to manage users' accounts.
Primary Actors	Admin.
Stake Holders	Admin.

Pre-conditions	Admin must have logged in to the admin panel.
Post-conditions	Should be able to add, remove & view accounts.
Main Success Scenario	• Admin can add new accounts. • Admin can remove existing accounts. • Admin can view the registered accounts.
Extensions	None

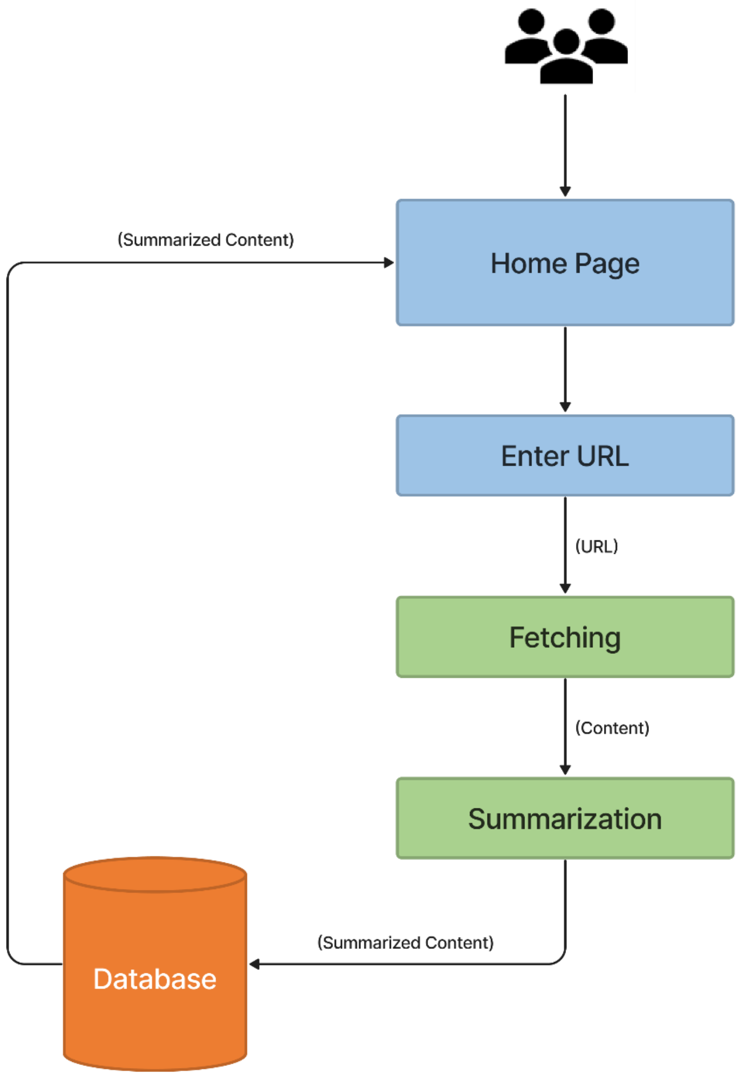
Use Case 6	**Profile Management**
Scope	Allows users to manage their profile.
Primary Actors	User.
Stake Holders	User.
Pre-conditions	User must be logged in.
Post-conditions	Should be able to edit the provided information.
Main Success Scenario	• User can edit his/her name. • User can change the email address. • User can change the username.
Extensions	If the user provides invalid information, then it won’t be changed.

Chapter 4
System Design

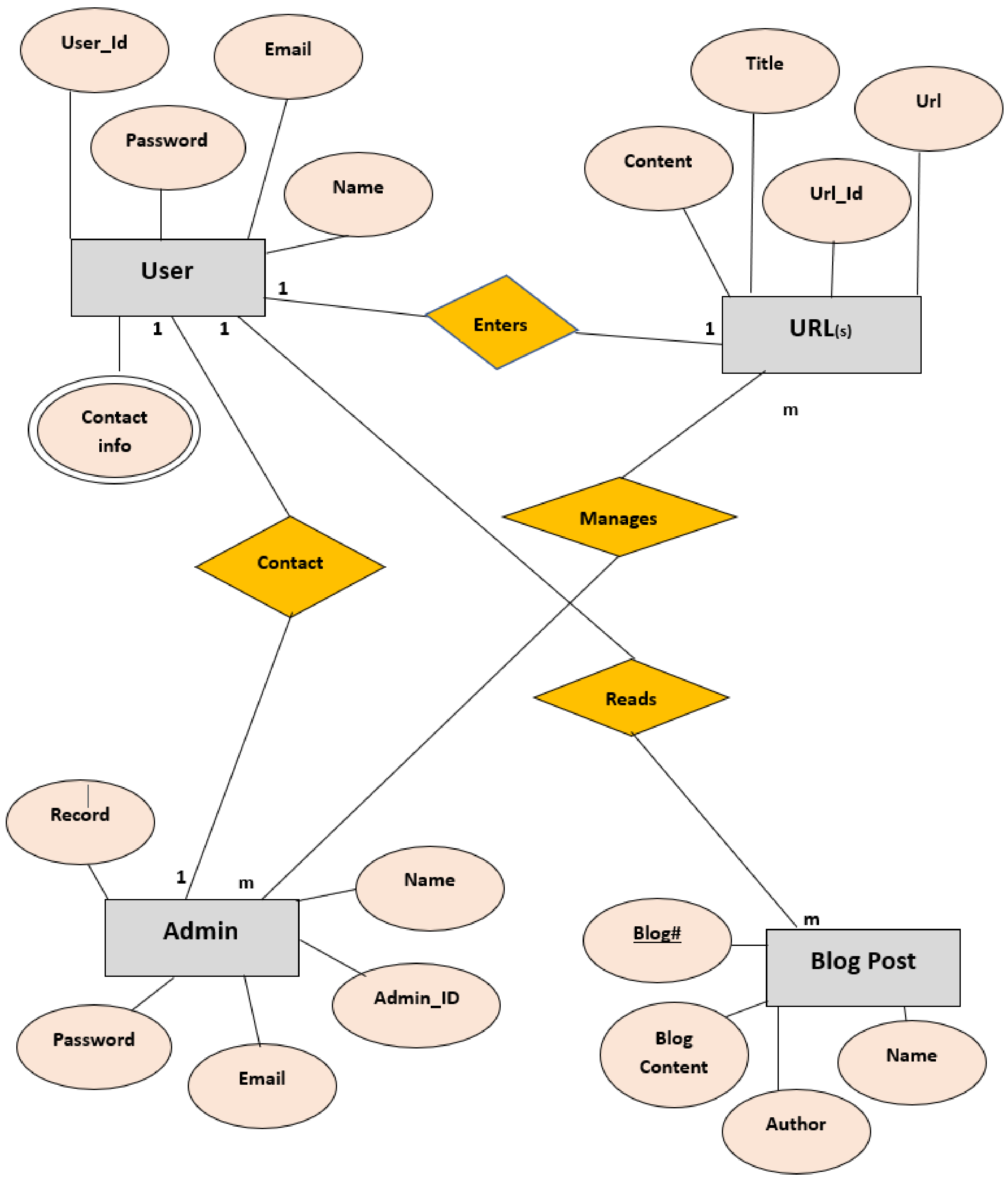
Chapter 4: System Design

System Design is the process to provide detailed data and information about the entire system and its system elements. System Design process to enable the implementation with the architectural entities as defined in models and requirements.

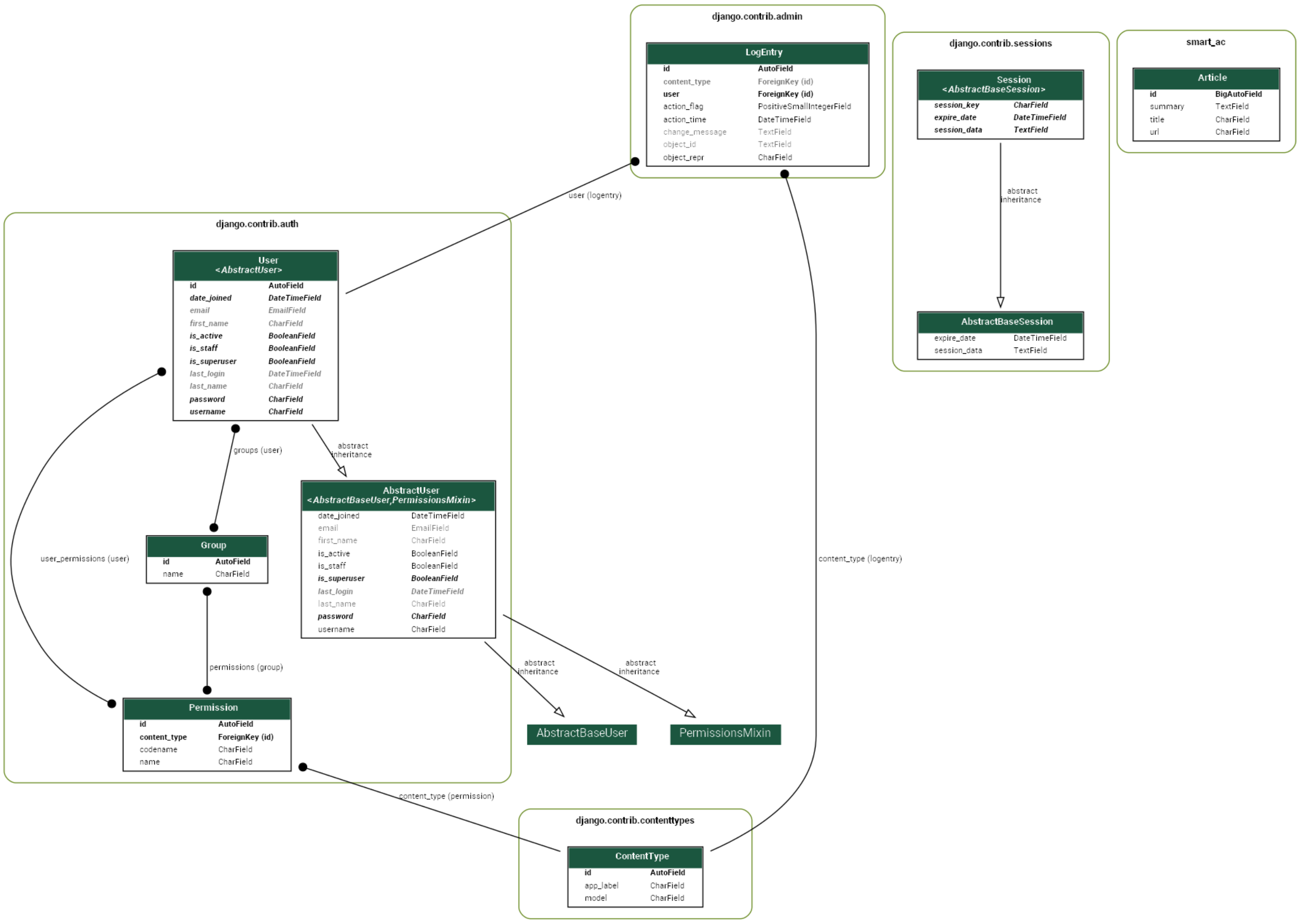
4.1. Architecture Diagram



4.2. Entity Relationship Diagram with data dictionary

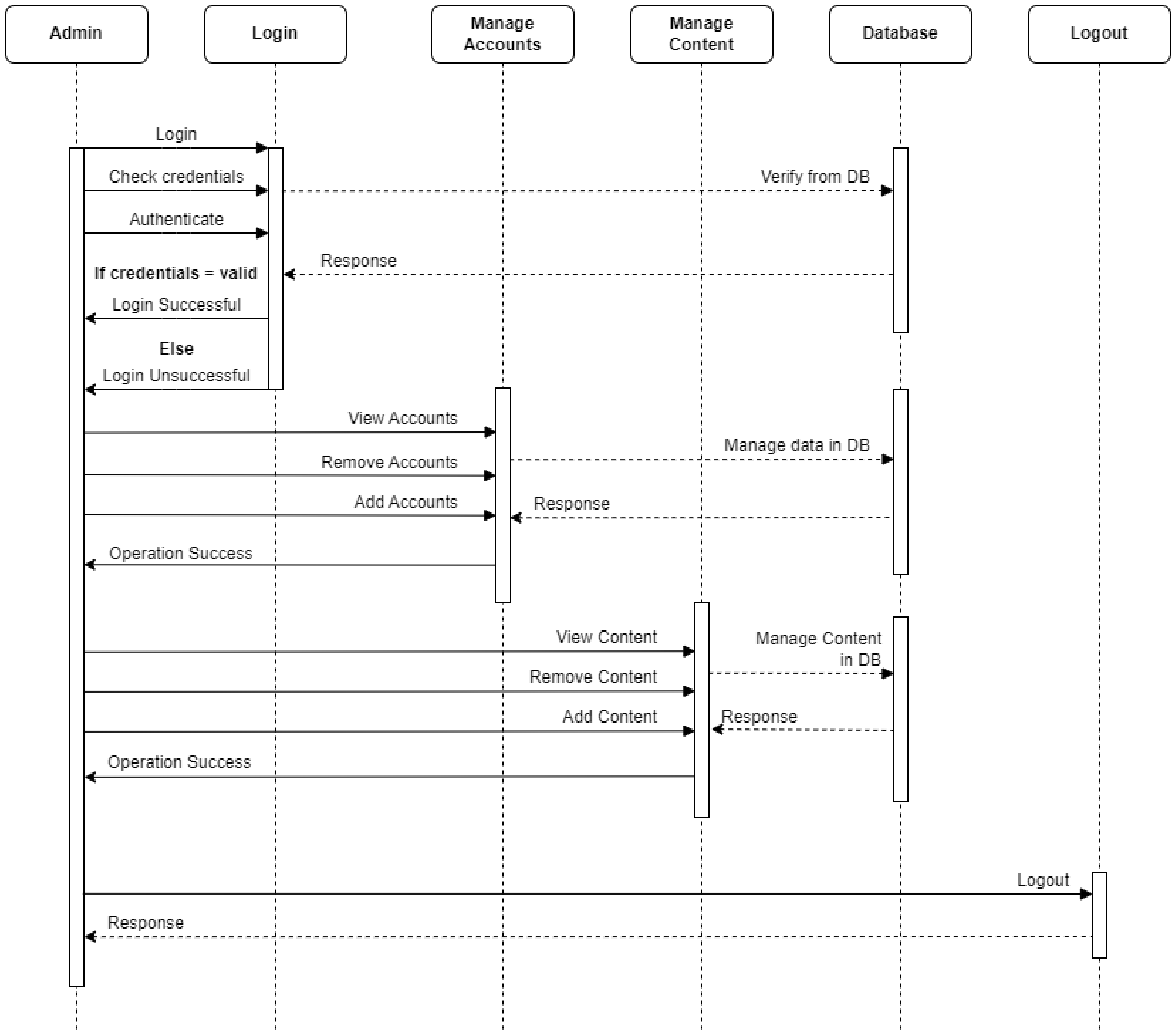


4.3. Class Diagram

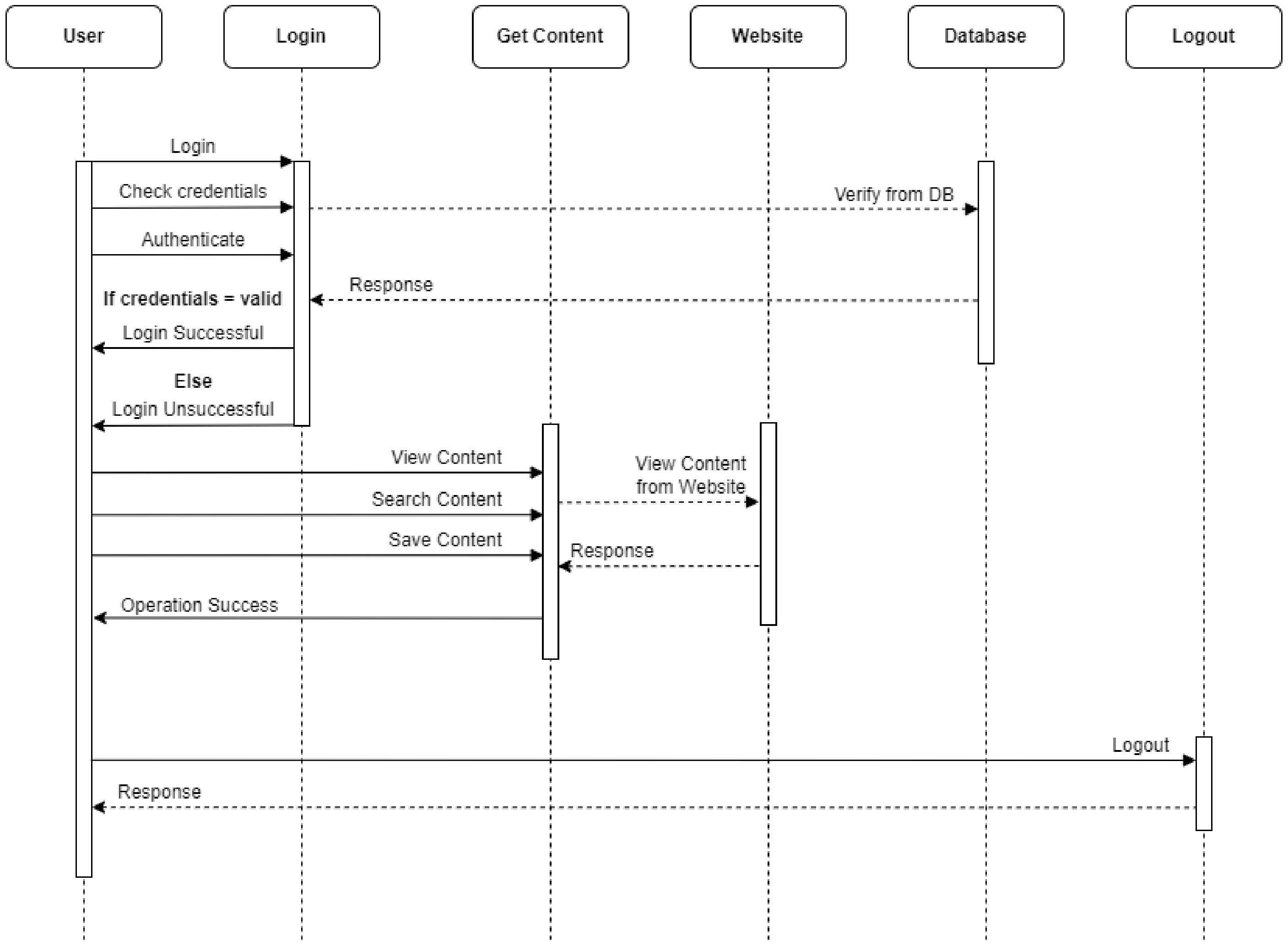


4.4. Sequence / Collaboration Diagram

Sequence Diagram of Admin

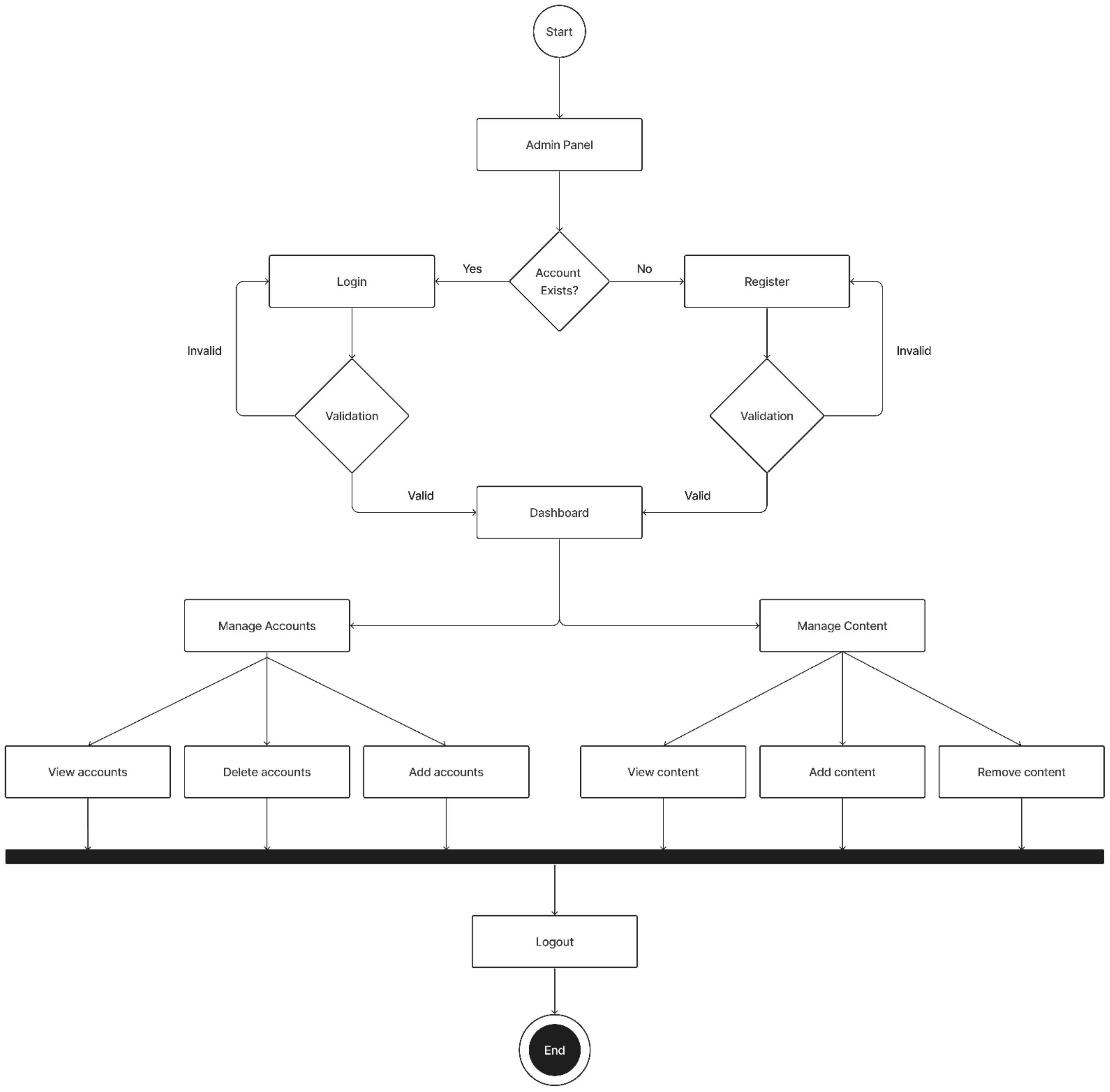


Sequence Diagram of User

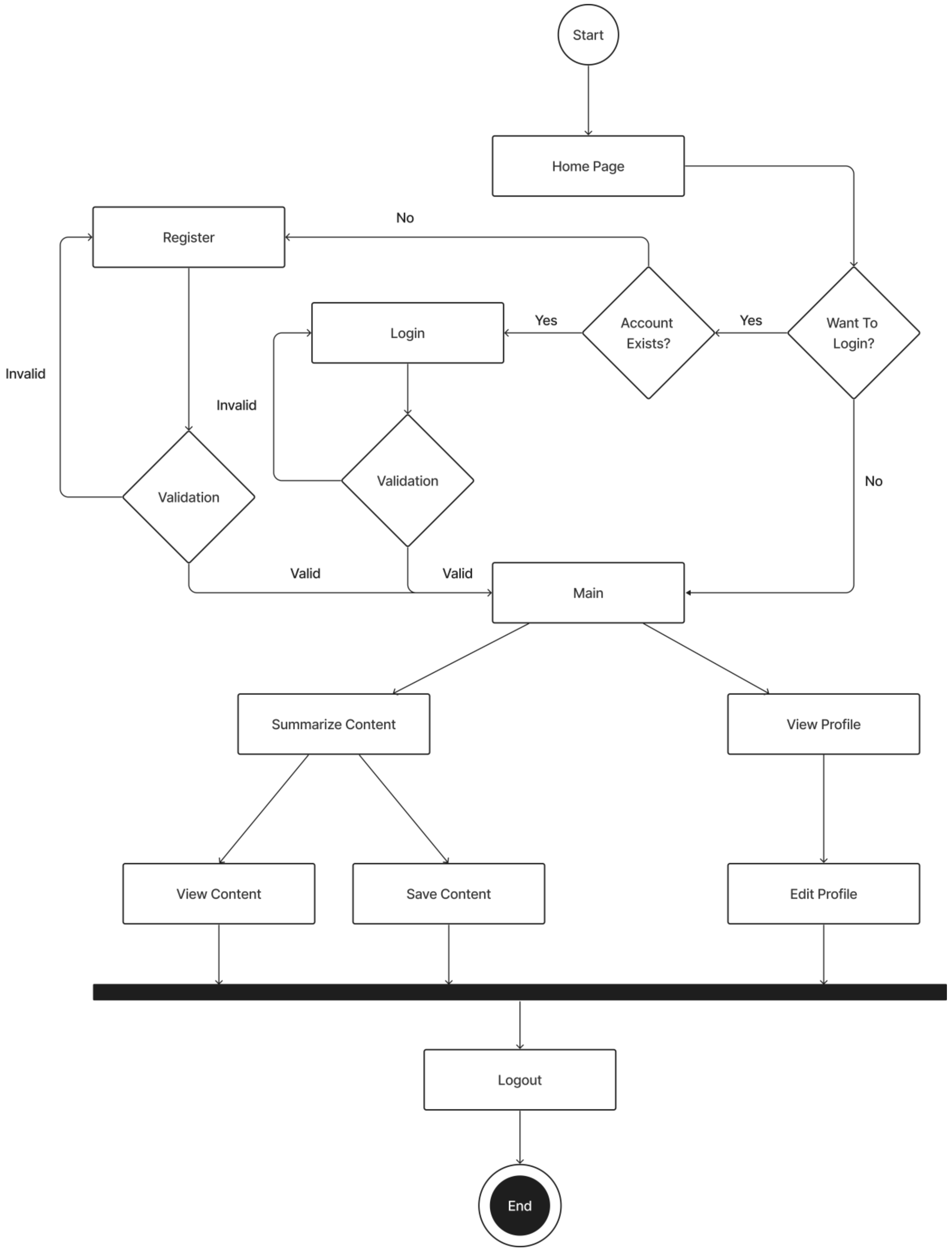


4.5. Activity Diagram

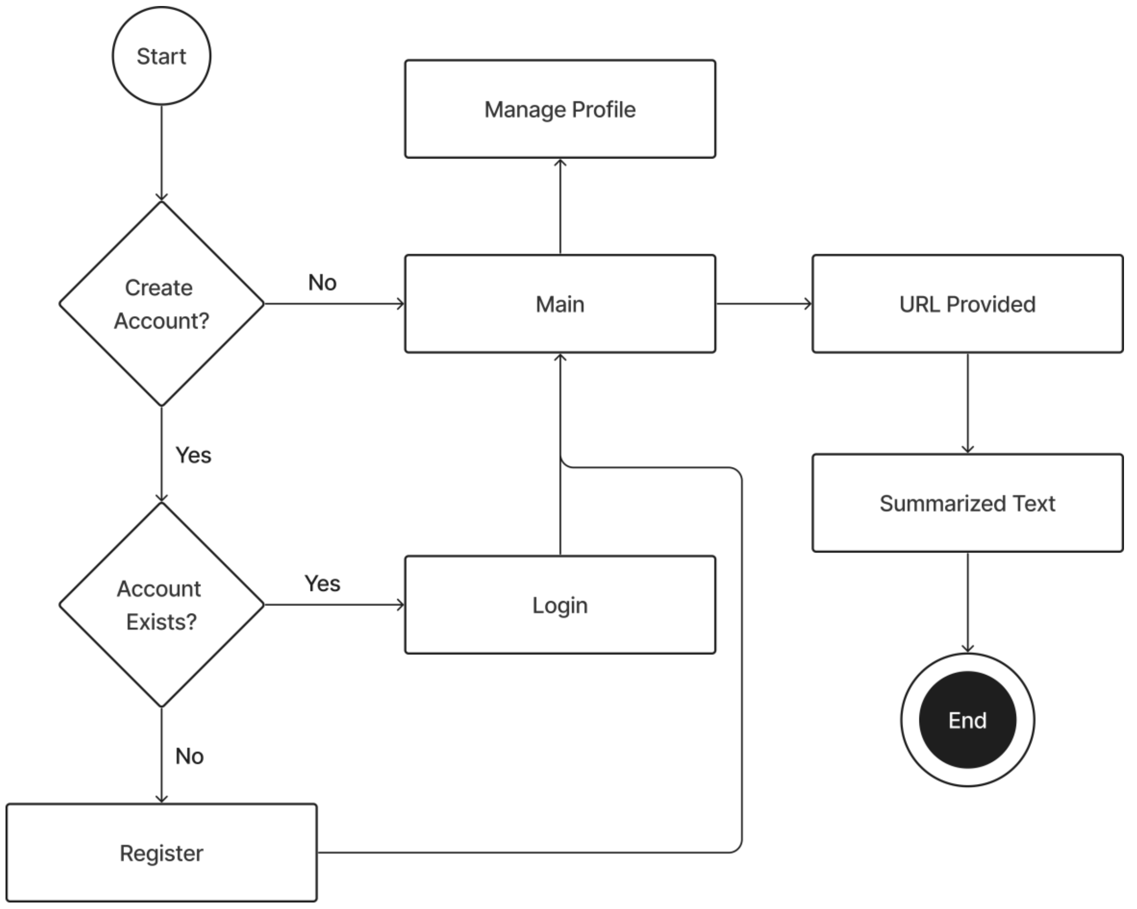
Activity Diagram for Admin



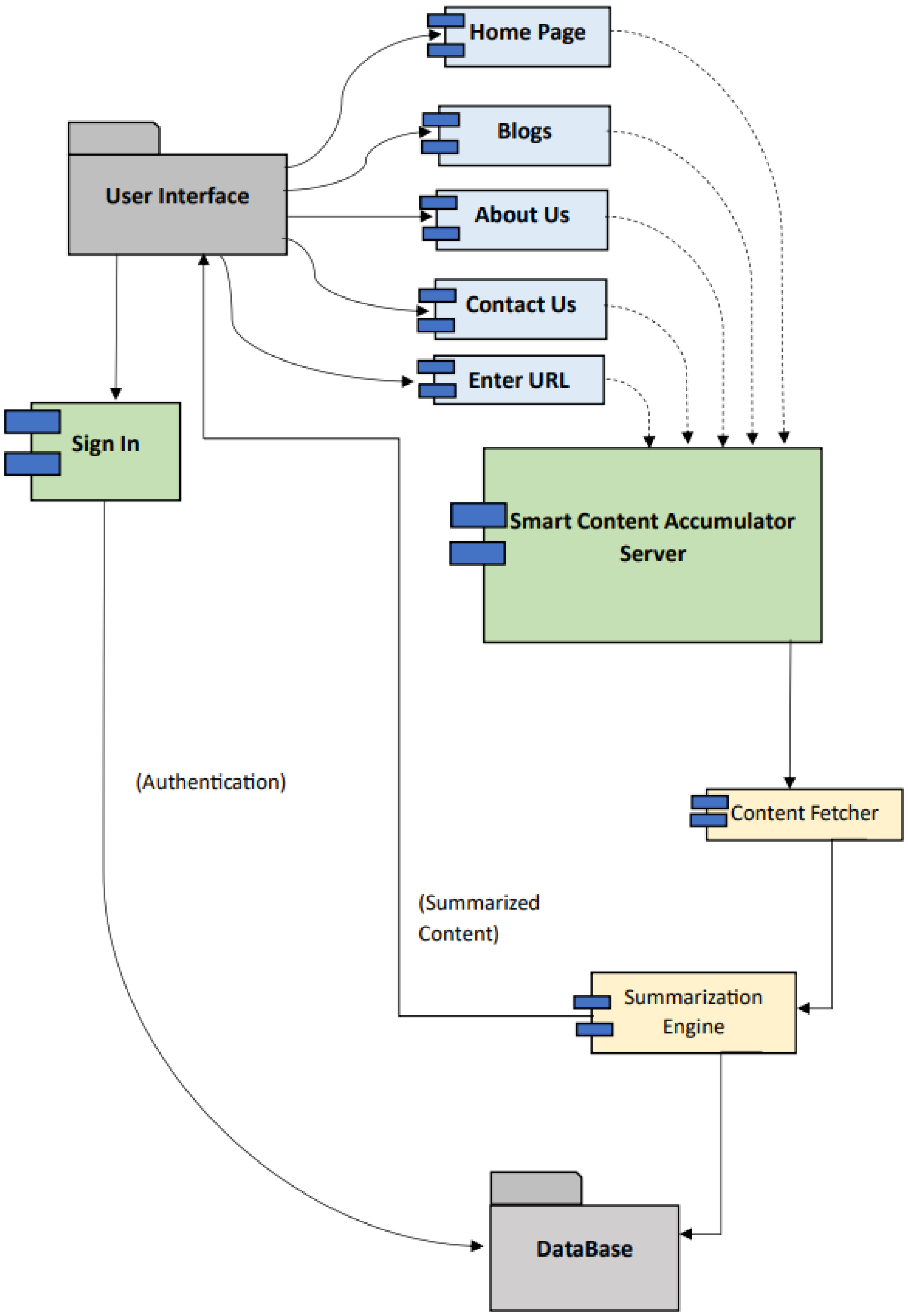
Activity Diagram for User



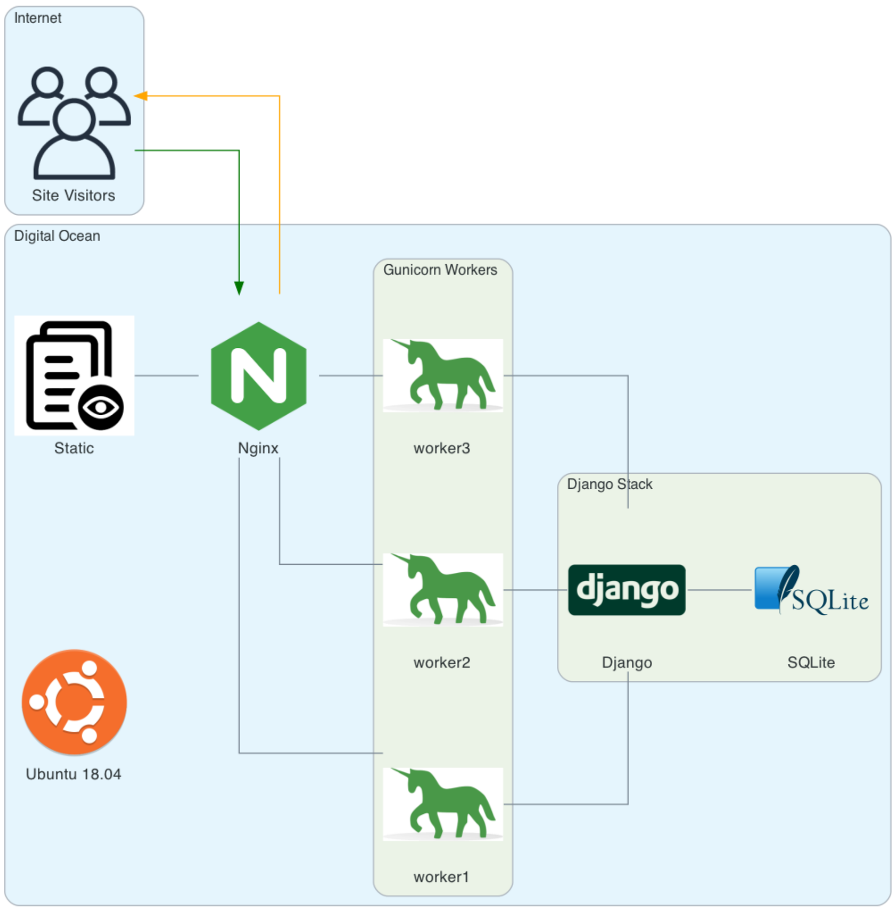
4.6. State Transition Diagram



4.7. Component Diagram



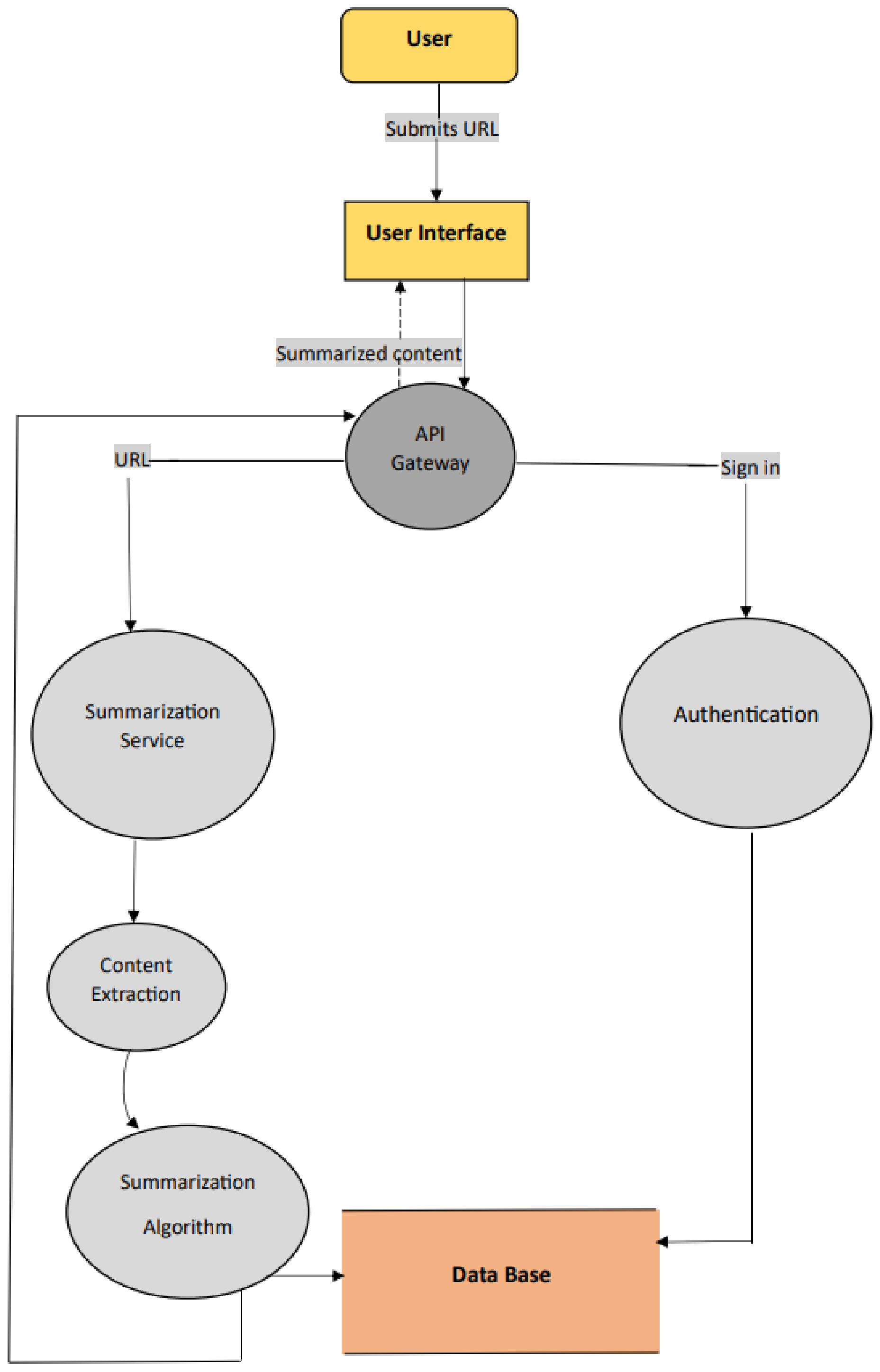
4.8. Deployment Diagram



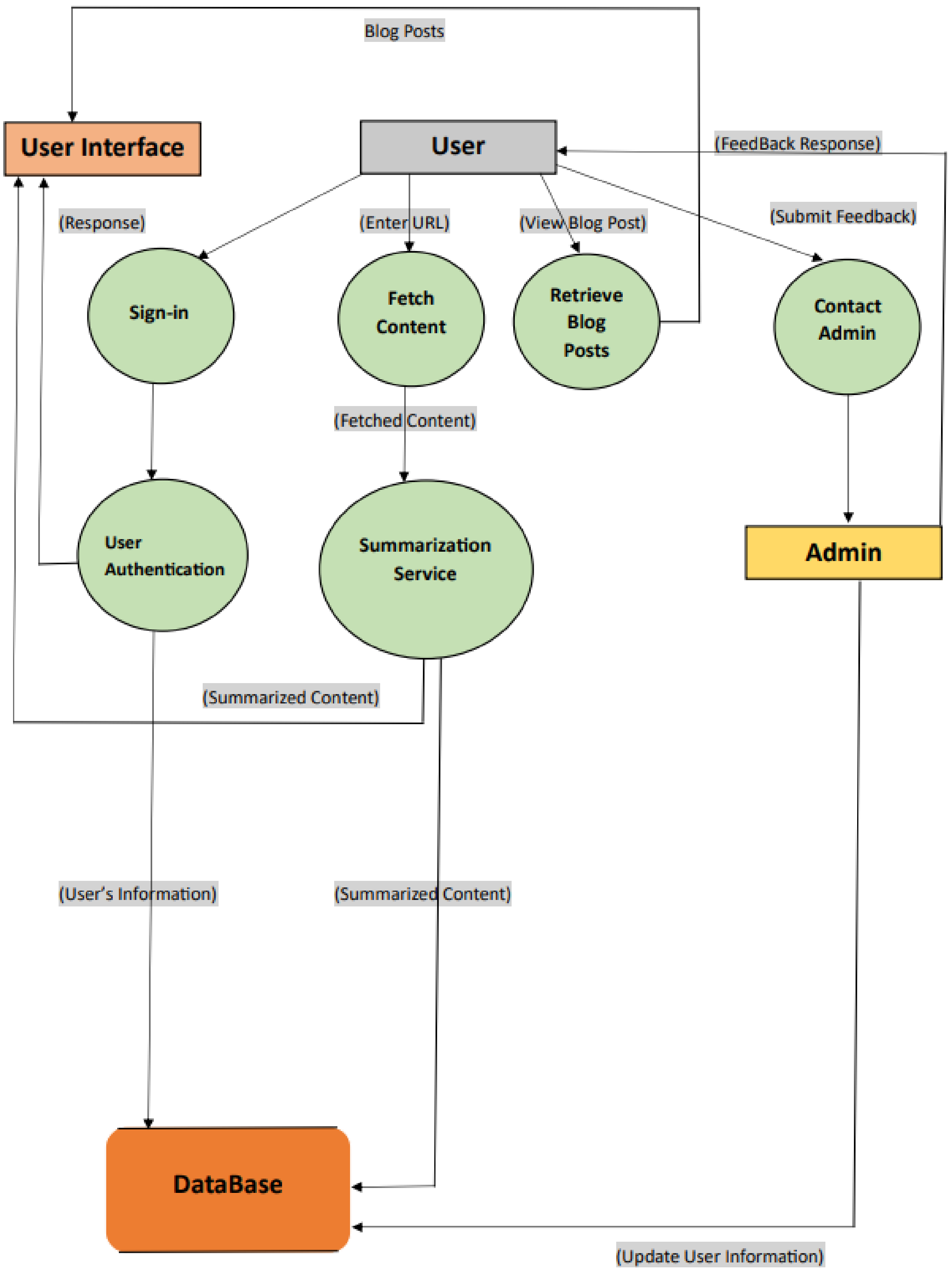
./custom_resources/Internal Working

4.9. Data Flow diagram

Level 1



Level 2


Blog Posts
User Interface
User
(FeedBack Response)
(Response)
(Enter URL)
(View Blog Post)
(Submit Feedback)
Sign-in
Fetch Content
Retrieve Blog Posts
Contact Admin
(Fetched Content)
User Authentication
Summarization Service
Admin
(Summarized Content)
(User's Information)
(Summarized Content)
DataBase
(Update User Information)

Chapter 5
Implementation

Chapter 5: Implementation

This chapter will focus on the tools and techniques that we will be using to create our website. This section is concerned with the goal of developing an essential website in which you can search for specific content based on the given keywords and also look upon related news using a recommendation system. Moreover, it also elaborates important flow control/pseudo codes, libraries, deployment environment and coding standards as well.

5.1. Important Flow Control/Pseudo codes

Firstly, we are going to develop website's front-end using html, css, js and bootstrap. After that we can convert this into django framework. Then we can scrap data from couple of websites and show on our website after that we can automate our website. After scraping data from website, we can summarization that data using text summarization algorithm. There are 2 different approaches that are used for text summarization.

1. Extractive summarization
2. Abstractive summarization

We are going to use the second technique to summarization data.

5.2. Components, Libraries, Web Services and stubs

Components

- Profiles
- Content Acquisition
- Content Summarization

Libraries

- Python Libraries
- Django framework
- Github

Web Services

- SQLite 3
- PythonAnywhere

5.3. Deployment Environment

The report highlights the deployment of a website on Digital Ocean, a leading cloud infrastructure provider. The website serves as an online platform for various purposes, catering to a wide range of users. This report outlines the key aspects of the deployment process, including the choice of Digital Ocean as the hosting platform, configuration setup, and the overall experience of deploying and managing the website.

The deployment of a website on Digital Ocean offers numerous benefits, such as scalability, reliability, and ease of management. Digital Ocean's robust infrastructure and user-friendly interface make it an ideal choice for hosting web applications. The website aims to provide an immersive and user-friendly experience to visitors, ensuring seamless navigation and access to valuable content.

Overall, deploying the website on Digital Ocean has proven to be a valuable choice, offering a solid foundation for the website's growth and success. The experience gained from this deployment process will serve as a valuable reference for future endeavors, ensuring the website's continued performance and adaptability in an ever-evolving digital landscape

5.4. Tools and Techniques

Tools

- Visual Studio
- Sqliteviwer
- Chrome
- Version control system Git

- Testing and debugging tools like PyTest, Selenium
- Figma
- FigJam
- Draw.io

Techniques

- **Django ORM (Object-Relational Mapper):** Django's ORM allows developers to work with a database using Python objects, rather than writing SQL queries. It provides a simple and intuitive interface for creating, updating, and deleting records in a database.

- **Django Forms:** Django Forms provide a way to create and validate user input, such as when a user fills out a form on a web page. They can be used to create forms for creating, updating, and deleting records in the database.

- **Django Templates:** Django Templates allow developers to define the structure of a web page and to insert dynamic content into the page. They provide a simple syntax for inserting variables, loops, and other logic into a HTML or XML document.

- **Django Debug Toolbar:** Django Debug Toolbar is a tool that provides a range of debugging and profiling features for Django applications. It can be used to identify performance issues, view SQL queries, and track request/response cycles.

5.5. Best Practices / Coding Standards

There are many best practices and coding standards that developers and organizations can follow to improve the quality and maintainability of their code. Here are a few examples:

- **Use descriptive, meaningful variable names:** Use names that clearly describe the purpose of the variable, rather than abbreviations or short names that may be difficult to understand.

- **Follow a consistent indentation style:** Use a consistent number of spaces or tabs to indent code blocks, and be consistent throughout the codebase.
- **Use comments to explain the purpose of code blocks:** Use comments to provide context and explain the purpose of code blocks, particularly if the code is complex or not immediately understandable.
- **Follow a consistent naming convention for functions, variables, and other identifiers:** Use a consistent naming convention, such as camelCase or snake_case, to make it easier to read and understand the code.
- **Use whitespace and blank lines to improve readability:** Use whitespace and blank lines to separate code blocks and improve the readability of the code.
- **Avoid using global variables:** Global variables can make it difficult to understand the scope of a variable and can cause unintended side effects. Instead, use local variables or pass variables as arguments to functions.
- **Use appropriate data types for variables:** Use the appropriate data type for a variable, such as using integers for numerical values and strings for text.
- **Use error handling to anticipate and handle exceptions:** Use error handling to anticipate and handle exceptions, such as when a user input is invalid or a file cannot be found. This can help prevent errors and improve the reliability of the code.

5.6. Version Control

Git

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Chapter 6
Testing and Evaluation

Chapter 6: Testing and Evaluation

The assessment is done by the tree of standards from the essential measures up. The technique for still up in the air by the choice standards. The variation which is granted the most noteworthy grade should be the best one. Because of the intricacy of Interactive course learning and countless measures it is fundamental that the dynamic model permits us to acquire the last appraisal, yet additionally a definite halfway investigation of individual components.
In this manner we can identify flimsy parts and detriments of the framework, which can be utilized as the reason for framework enhancements.

6.1. Use Case Testing

Test Case ID	S001	**Test Case Description**	Test the efficiency and accuracy of summarization functionality in the smart content accumulator.		
Created By	Shuja ul Hassan	**Reviewed By**	Mr. Nouman Jazeb	**Version**	4.1
Tester's Name	Shuja ul Hassan & Umair Ali	**Date Tested**	1 June 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Pre-Conditions:**
1	Access to Chrome Browser
2	A stable internet connection
3	A valid URL to summarize
4	The website should be up and running.

S #	**Post-Conditions:**
1	The generated summary is securely saved into a database, ensuring data integrity and accessibility for future reference.
2	The user should be able to log out of the website successfully.
3	The generated summary should accurately capture the main points and key information of the provided content.
4	The user should be able to access all of the content that they were able to access before the test.

Test Scenario	Verify that the Smart Content Accumulator successfully generates a summary for a given URL.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://144.126.194.139:8000/	The homepage of Smart Content Accumulator should load successfully.	As Expected,	Pass
2	Locate the "Sign in" button in the header section and click on it.	The user should be redirected to the login or registration page.	Upon clicking the "Sign In" button, the user was successfully redirected to the login, allowing them to either sign in with their existing credentials or register as a new user.	Pass
3	Locate the "Home" link in the main menu and initiate a click action.	The user's action should result in a successful redirection to the homepage.	As Expected,	Pass
4	Access the main menu and choose the "About Us" link.	The user should be directed to the designated "About Us" page.	After clicking the "About Us" link, the user was successfully redirected to the designated "About Us" page, where comprehensive information about the team was presented.	Pass
5	Locate and select the "Blogs" link in the main menu.	The user should be redirected to the dedicated "Blogs" page.	Upon clicking the "Blogs" link, the user should be redirected to the "Blogs" page, where a collection of blog posts or articles is displayed.	Pass
6	Identify the "Contact Us" link within the main menu and initiate a click.	The user should be directed to the designated "Contact Us" page.	The designated "Contact Us" page is accessed by the user. asking for information and promising assistance within 24 hours.	Pass
7	Enter a URL in the input field on the homepage.	The input field should accept and accurately display the entered URL without any errors or disruptions.	The entered URL was successfully inputted into the designated field without any issues, and it was accurately displayed as intended.	Pass
8	On the homepage, locate and click on the prominently displayed	Upon clicking the "Summarize" button, the system should initiate the processing of the	The system successfully processed the provided URL and generated a summary that effectively condensed	Pass

	"Summarize" button.	provided URL and generate a concise and accurate summary based on the content.	the key information from the source.	
9	Enter a wrong URL and click summarize button then.	The system displays a message of unsuccessful operation.	As Expected,	Fail
10	Click on the "Sign in" button without filling in the username and password fields.	The website should display an error message indicating that the username and password fields are required.	The website does not show any error message and reloads.	Fail
11	Click on a specific blog post from the list.	The user should be redirected to the desired blog post.	Clicking on the blog post does not lead to the desired blog post.	Fail

6.2. Equivalence partitioning

Equivalence partitioning is a technique used to divide the input data into different groups, called equivalence classes, in order to design effective test cases. In the case of a text summarizer implemented using the Sshleifer/distilbart-cnn-12-6 model in Python, the following equivalence partitions can be identified:

1. **Input Text Length:**
   - Short Text: Text inputs consisting of a few sentences or a small paragraph.
   - Medium Text: Text inputs of moderate length, typically a few paragraphs.
   - Long Text: Text inputs that are significantly longer, such as multiple pages or articles.

2. **Text Complexity:**
   - Simple Text: Inputs with straightforward language, limited jargon, and minimal technical terms.
   - Complex Text: Inputs with intricate language, specialized vocabulary, and domain-specific terminology.

3. **Language and Domain:**
    - English Language: Inputs written in the English language.
    - Non-English Language: Inputs written in languages other than English.
    - Domain-specific Text: Inputs related to specific domains like medical, legal, scientific, or technical.

4. **Summarization Output Length:**
    - Short Summary: Summaries consisting of a few sentences.
    - Medium Summary: Summaries of moderate length, typically a few paragraphs.
    - Long Summary: Summaries that are significantly longer, providing a detailed overview of the input text.

5. **Source Text Type:**
    - News Articles: Inputs that are news articles from various sources.
    - Scientific Papers: Inputs that are scientific research papers.
    - Blog Posts: Inputs that are blog posts or opinion pieces.
    - Legal Documents: Inputs that are legal documents, such as contracts or court rulings.

6.3. Boundary value analysis

Boundary value analysis is a testing technique used to determine test cases based on the boundaries or edge values of input data. In the context of a text summarizer implemented using the Sshleifer/distilbart-cnn-12-6 model in Python, the following boundary values can be considered for testing:

1. **Input Text Length:**
    - Minimum Length: Test cases with the smallest possible input texts, such as an empty string or a single word.

- Maximum Length: Test cases with input texts that reach or exceed the maximum supported length by the summarizer. This helps verify the system's ability to handle long texts without performance issues or errors.

2. **Text Complexity:**
   - Minimum Complexity: Test cases with simple and straightforward texts, containing basic language constructs without complex grammar or technical terms.
   - Maximum Complexity: Test cases with highly complex texts, incorporating intricate language, domain-specific jargon, or challenging sentence structures.

3. **Summarization Output Length:**
   - Minimum Summary Length: Test cases where the summarization output consists of the shortest possible summary, such as a single sentence.
   - Maximum Summary Length: Test cases where the summarization output reaches or exceeds the maximum allowed summary length, validating the system's ability to handle longer summaries.

4. **Source Text Type:**
   - Minimum Source Text: Test cases with the smallest possible source texts, such as a single sentence or paragraph.
   - Maximum Source Text: Test cases where the source text reaches or exceeds the maximum supported length, verifying the system's ability to handle large source texts accurately and efficiently.

6.4. Data flow testing

Data flow testing is a testing technique that focuses on the flow of data within a system. In the case of a text summarizer implemented in Python using the Sshleifer/distilbart-cnn-12-6 model, data flow testing can be conducted to ensure the accuracy and integrity of data as it moves through the different components of the system:

- **Input Text:** The original text that needs to be summarized.
- **Preprocessed Text:** The input text after undergoing necessary preprocessing steps, such as cleaning, tokenization, and normalization.
- **Encoded Input:** The preprocessed text converted into numerical representations or embeddings suitable for the Sshleifer/distilbart-cnn-12-6 model.
- **Summarization Output:** The generated summary of the input text.
- **Decoded Summary:** The summarization output converted back into human-readable text.

By applying data flow testing to the text summarizer project using the Sshleifer/distilbart-cnn-12-6 model, you can verify the correctness and integrity of the data as it moves through different stages of the system. This helps ensure the accuracy and reliability of the summarization process, leading to high-quality summaries.

6.5. Unit testing

Unit testing is a crucial aspect of software development that focuses on testing individual units or components of a system to ensure they function correctly. In the case of a text summarizer implemented in Python using the Sshleifer/distilbart-cnn-12-6 model, unit testing can be employed to validate the functionality and accuracy of specific functions or modules within the system.

- Flow has been clearly checked the is correct between inside and outside the module.
- Check statements have been verified.
- Extreme boundaries values of output have been verified.
- Loops have been checked if they are correct or not.

6.6. Integration testing

The integration testing phase for the text summarizer project using the Sshleifer/distilbart-cnn-12-6 model was successful. The tests focused on verifying the seamless integration and functionality of different components within the system. Through a series of test cases, the integration between the preprocessing, encoding, summarization, and decoding modules was thoroughly examined. The tests covered various input scenarios, including different text lengths, complexities, and domains, ensuring the system's ability to handle diverse texts. The integration tests validated that the modules effectively communicated and processed data, generating accurate and coherent summaries. The successful completion of integration testing provides confidence in the reliability and performance of the text summarizer, ensuring that it can seamlessly integrate and produce high-quality summaries.

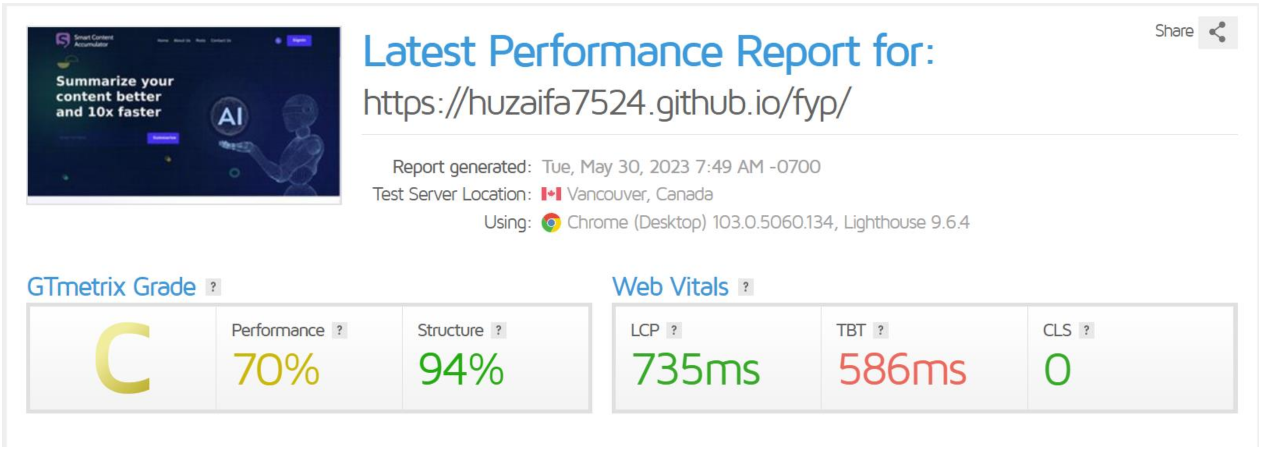
6.7. Performance testing

After conducting a comprehensive performance test using GTmetrix, the website received a Grade C performance with a score of 70%. While the performance grade may indicate areas for improvement, it's important to note that the website's structure achieved an impressive score of 94%. This indicates that the website has a well-organized and logically structured design, contributing to a positive user experience.

The Grade C performance score suggests that there are opportunities to optimize the website's speed and efficiency further. It's crucial to focus on key performance factors, such as reducing page load times, optimizing resource delivery, and enhancing server response. By addressing these aspects, the website can deliver a faster and more responsive experience to its visitors.

The high structure score of 94% reflects the website's solid foundation in terms of its layout and organization. This means that the website's content is well-structured, easily navigable, and follows best practices for web design. A strong structure enhances usability and helps visitors find the information they need quickly and intuitively.

With these insights, the development team can prioritize and implement performance optimization techniques to provide visitors with a smoother and more engaging user experience. Regular monitoring and continuous improvements will be essential to maintain optimal performance levels and strive for a higher performance grade in future assessments.



Latest Performance Report for:
https://huzaifa7524.github.io/fyp/

Report generated: Tue, May 30, 2023 7:49 AM -0700
Test Server Location: Vancouver, Canada
Using: Chrome (Desktop) 103.0.5060.134, Lighthouse 9.6.4

GTmetrix Grade: C — Performance 70% — Structure 94%

Web Vitals: LCP 735ms — TBT 586ms — CLS 0

6.8. Stress Testing

During the stress testing phase of the text summarizer project implemented using the Sshleifer/distilbart-cnn-12-6 model, the system's performance and stability were rigorously evaluated under high-stress conditions. The objective was to assess the system's ability to handle a significant load and identify its breaking points or limitations. Multiple stress test scenarios were executed, including input texts of extreme lengths, a high number of concurrent users, and rapid consecutive requests. The system's response time, resource utilization, and error handling were closely monitored and analyzed. Stress testing allowed for the identification of any performance bottlenecks, scalability issues, or unexpected behavior under heavy loads. By subjecting the text summarizer to stress testing, the project team gained valuable insights into its robustness and scalability, enabling them to optimize performance and ensure a reliable and efficient summarization process, even in demanding scenarios.

Chapter 7

Summary, Conclusion and Future Enhancements

Chapter 7: Summary, Conclusion & Future Enhancements

7.1. Project Summary

Smart Content Accumulator is a website that offers a useful service to people who need to summarize large amounts of text quickly and efficiently. Whether you are a student, a researcher, or just someone who needs to stay informed about current events, this website can help you to digest large amounts of information in a meaningful way.

The way the site works is quite simple. All you need to do is copy and paste the URL of the web page you want to summarize into the text box on the Smart Content Accumulator website. The site's software will then analyze the text and generate a summary of the most important points. The summary is usually a few paragraphs long, and it is designed to give you a quick overview of the text without having to read through the entire article or document.

Overall, Smart Content Accumulator is a fantastic website that offers a valuable service to people who need to summarize large amounts of text quickly and accurately. Whether you are a student, a researcher, or just someone who wants to stay informed about current events, this website can help you to save time and make more informed decisions. Its accuracy, ease of use, and affordability make it an excellent tool for anyone who needs to summarize text on a regular basis.

7.2. Achievements and Improvements

- **High Accuracy and Relevance:** The text summarizer project achieved significant milestones in terms of accuracy and relevance. Through extensive research and development, the summarization algorithms were fine-tuned to extract the most important information from the source text. The generated summaries exhibited a high degree of relevance to the original content, providing users with concise and meaningful summaries.

- **Efficient Information Extraction:** The project successfully addressed the challenge of information extraction from large volumes of text. By implementing state-of-the-art natural language processing techniques and machine learning algorithms, the summarizer efficiently identified and extracted the key ideas, facts, and arguments from various text sources. This achievement allowed users to save valuable time by obtaining the essential information in a fraction of the time it would take to read the entire text.

- **Contextual Understanding:** One of the notable achievements of the project was its ability to comprehend the context of the source text. Through advanced language modeling and deep learning approaches, the summarizer captured the semantic relationships and contextual nuances present in the text. This achievement contributed to the production of summaries that not only conveyed the main points but also maintained the overall coherence and meaning of the original content.

- **Abstractive Summarization Capabilities:** The project incorporated advanced abstractive summarization techniques, enabling the system to generate summaries that went beyond mere extraction of sentences. By leveraging cutting-edge natural language generation models, the summarizer achieved the ability to produce coherent and contextually appropriate summaries that captured the essence of the source text. This achievement opened new possibilities for more creative and human-like summarization output.

- **Real-Time Summarization:** The project achieved real-time text summarization capabilities, allowing users to receive summaries instantaneously. By optimizing the summarization algorithms and leveraging parallel processing techniques, the system could generate summaries in near real-time, ensuring a seamless and efficient user experience.

- **User-Friendly Interface:** The project placed a strong emphasis on user experience, resulting in the development of a user-friendly interface. The interface was designed to be intuitive and accessible, allowing users to easily input the text and receive the summarized output. The achievement of a user-friendly interface enhanced the usability and adoption of the text summarizer among a wide range of users.

7.3. Critical Review

The text summarizer project presents several commendable achievements in the field of automatic text summarization. However, a critical review also highlights certain areas that could benefit from further development and improvement.

One of the notable strengths of the project is its ability to extract key information accurately from a variety of text sources. The project's implementation of advanced natural language processing techniques and machine learning algorithms contributes to the system's efficiency in identifying and selecting relevant content. This achievement is crucial in addressing the challenge of information overload and saving user's valuable time.

The project's adaptability across different domains is another positive aspect. By successfully summarizing texts from diverse sources such as news articles, scientific papers, and legal documents, the system demonstrates versatility and applicability. This adaptability enhances the project's potential for widespread use in various industries and knowledge domains.

Furthermore, the project could consider addressing the challenge of domain-specific summarization. While the project demonstrates adaptability across different domains, further research and development could focus on domain-specific techniques to optimize summarization quality for specialized domains. This would enhance the system's performance in accurately capturing domain-specific language, jargon, and nuances.

Lastly, it would be valuable for the project to explore the ethical implications of text summarization. As automatic summarization systems play a significant role in information consumption and decision-making processes, considering issues such as bias, fairness, and transparency becomes increasingly important. Integrating ethical considerations into the development process can ensure responsible and accountable use of the summarizer.

In conclusion, the text summarizer project demonstrates notable achievements in accuracy, adaptability, and abstractive summarization. However, addressing the areas of evaluation metrics, user feedback, domain-specific summarization, and ethical considerations could further enhance the project's performance, usability, and societal impact.

7.4. Lessons Learnt

During the process of creating a text summarizer project in Python, several valuable lessons can be learned. Here are some key lessons that developers often encounter:

- **Data Preprocessing is Crucial:** Text summarization heavily relies on the quality of input data. Preprocessing tasks such as cleaning the text, removing irrelevant information, handling special characters, and tokenization are vital to ensure accurate and meaningful summaries. Investing time in robust data preprocessing techniques can significantly improve the overall performance of the summarizer.

- **Understanding Different Summarization Techniques:** There are various approaches to text summarization, including extractive and abstractive methods. Extractive summarization involves selecting and combining important sentences from the source text, while abstractive summarization involves generating new sentences that capture the main ideas. Understanding the strengths and limitations of each technique is crucial for choosing the most suitable approach for the project.

- **Utilizing NLP Libraries and Frameworks:** Python offers a wide range of powerful natural language processing (NLP) libraries and frameworks, such as NLTK, SpaCy, Gensim, and Transformers. Leveraging these libraries can greatly simplify the implementation of core NLP tasks like tokenization, part-of-speech tagging, and named entity recognition. It is important to explore and understand the available tools to expedite the development process.

- **Balancing Model Complexity and Performance:** When working with text summarization, developers often encounter trade-offs between model complexity and performance. More sophisticated models, such as transformer-based architectures, may provide better results but require higher computational resources and longer training times. Striking the right balance between model complexity and system requirements is crucial to ensure the summarizer is both effective and efficient.

- **Evaluating Summarization Quality:** Evaluating the quality of summaries can be challenging. Commonly used metrics include ROUGE (Recall-Oriented Understudy for Gisting Evaluation), which assesses the overlap between the generated summary and reference summaries. However, it is important to consider additional metrics that capture the fluency, coherence, and readability of the summaries. Evaluating the summarizer from multiple perspectives provides a more comprehensive understanding of its performance.

- **Handling Challenges of Abstractive Summarization:** Abstractive summarization poses unique challenges, including generating coherent and grammatically correct sentences that capture the essence of the source text. Techniques like encoder-decoder architectures and attention mechanisms can be used to address these challenges. However, it is important to acknowledge that abstractive summarization remains an active research area, and further exploration and experimentation are needed to achieve optimal results.

- **Ethical Considerations:** As with any AI-powered system, it is essential to consider ethical considerations. Text summarizers can potentially amplify biases present in the data and affect the dissemination of information. Being aware of the ethical implications, considering fairness and bias, and addressing potential issues can help develop responsible and unbiased summarization systems.

In summary, developing a text summarizer project in Python involves mastering data preprocessing, understanding different summarization techniques, utilizing NLP libraries, balancing model complexity and performance, evaluating summarization quality, incorporating user feedback, tackling challenges of abstractive summarization, and considering ethical considerations. Learning these lessons can lead to the creation of more effective, robust, and responsible text summarization systems.

7.5. Future Enhancements/Recommendations

There are several potential future enhancements and recommendations for the project. Here are a few:

- **Fine-tuning for Specific Domains:** Currently, text summarizers have a broad applicability across different domains. However, fine-tuning the models specifically for certain domains can lead to better performance and more accurate summaries. By training the summarizer on domain-specific datasets or incorporating domain-specific knowledge, the system can capture the nuances and terminology unique to that domain.

- **Multi-document Summarization:** Expanding the capabilities of the text summarizer to handle multiple documents or sources of information can be a valuable enhancement. This would allow users to summarize and consolidate information from multiple texts or articles on a particular topic, providing a comprehensive overview and saving users time when researching or reviewing multiple sources.

- **Enhanced Abstractive Summarization:** Abstractive summarization techniques can be further improved to generate more fluent and coherent summaries. Exploring advanced neural language generation models, such as transformer-based architectures like GPT-3, can enhance the system's ability to generate human-like summaries with improved language structure, coherence, and contextuality.

- **User Customization and Preferences:** Providing users with customization options for the summarization process can enhance their experience. This could include options to specify the desired summary length, control the level of detail, or prioritize certain aspects of the content. Allowing users to personalize the summarization output based on their preferences can make the system more adaptable and useful for individual needs.

- **Interactive and Query-Based Summarization:** Integrating interactive and query-based summarization capabilities can enhance user engagement and utility. Allowing users to provide specific queries or prompts, along with the text, can enable the summarizer to generate targeted summaries that directly address their information needs. This approach can be particularly useful in applications where users seek specific answers or insights from the source text.

- **Incorporating Reinforcement Learning:** Applying reinforcement learning techniques to train the summarization model can help optimize and fine-tune the system's performance. Reinforcement learning can enable the model to learn from user feedback, reinforcement signals, or human summaries to improve the quality, fluency, and relevance of the generated summaries.

- **Real-time Summarization:** Enhancing the system to provide real-time or near real-time summarization capabilities can make it more valuable in time-sensitive scenarios such as news updates or live events. Optimizing the summarization algorithms, leveraging efficient data processing techniques, and parallelizing computations can help achieve faster and more responsive summarization output.

- **Multi-lingual Summarization:** Expanding the capabilities of the text summarizer to handle multiple languages would greatly enhance its usefulness in a global context. Incorporating techniques like machine translation followed by summarization can enable users to access summaries of texts in different languages, bridging language barriers and facilitating cross-lingual information consumption.

These future enhancements and recommendations can help advance the capabilities and utility of a text summarizer, enabling more accurate, personalized, and efficient summarization of textual content.

Reference and Bibliography

Reference and Bibliography

Below are mentioned some papers from where we took references for our project:

[1] *Shichao Sun and Wenjie Li, The Hong Kong Polytechnic University, Alleviating Exposure Bias via Contrastive Learning for Abstractive Text Summarization*

[2] *Rahul Tangsali, Aditya Vyawahare, Aditya Mandke, Onkar Litake, Dipali Kadam, Pune Institute of Computer Technology, India, Abstractive Approaches To Multidocument Summarization Of Medical Literature Reviews*

[3] *Alaa Mohamed, Marwan Ibrahim, Mayar Yasser, Mohamed Ayman, Menna Gamil, Walaa Hassan, News Aggregator and Efficient Summarization System*

[4] *Kishor Kumar Reddy C et al 2021 J. Phys., A Text Mining using Web Scraping for Meaningful Insights*

[5] *HESHAM AHMED SALEH YAHYA ALAA MASSOUD, AUTOMATIC EXTRACTION-BASED TEXT SUMMARIZATION*

Moreover, some of the website links are also mentioned below:

[6] https://medium.com/analytics-vidhya/text-summarization-using-bert-gpt2-xlnet-5ee80608e961

[7] https://www.projectpro.io/article/transformers-bart-model-explained/553

[8] https://towardsdatascience.com/a-step-by-step-guide-to-web-scraping-in-python5c4d9cef76e8

[9] https://www.researchgate.net/figure/Data-flow-diagram-of-Web-Scrapping_fig1_352350347

[10] https://www.techtarget.com/searchcontentmanagement/definition/content-aggregator

[11] https://www.octoparse.com/blog/how-to-build-a-news-aggregator-with-text-classification#

[12] https://curator.io/blog/ultimate-guide-to-content-aggregators

[13] https://huggingface.co/docs/transformers/model_doc/distilbert

[14] https://internationalsales.lexisnexis.com/glossary/data-as-a-service/content-aggregator

[15] https://towardsdatascience.com/abstractive-summarization-using-pytorch-f5063e67510

[16] https://www.projectpro.io/article/transformers-bart-model-explained/553

[17] https://blog.hubspot.com/marketing/content-aggregators