Automatc Text Summarization

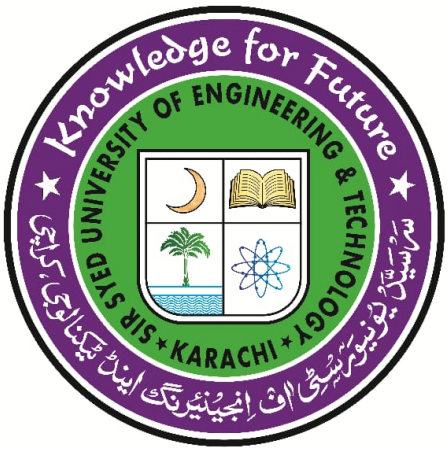
Final Year Project Report

B.S. in Software Engineering

By

Muzammil Sardar Abbasi (GL)	2020-SE-165
Imad Khan	2020-SE-176
Syed Muhammad Zejah Ali Rehmani	2020-SE-154
Muhammad Ahsan Siddique	2020-SE-218

Supervised By
Miss. Sonish Aslam
Associate Professor
SSUET



2023-2024
DEPARTMENT OF SOFTWARE ENGINEERING
Sir Syed University of Engineering and Technology

Preface

automatic text summarization (ATS) is an innovative Flutter app which introduces a novel way to generate abridgements of the readers 'own texts.With a frequency that matches your imagination, this brilliantly designed application employs the power of artificial intelligence to compose concise and clear summaries on its own in response to keywords you provide. Other than simply processing vast quantities of boundless text, our app makes use of tflite_model to fully analyze the way content is logically fitted into each episode and then deliver all this information directly underneath your fingertips.Behind this decision to add artificial intelligence to our app is a longing heart's wish, in fact, for an immense technique with no limits.In addition, we want to show how AI can be tailored for our own lives.That means dealing with information overload as the first challenge.Those days when people would spend all day plowing through information to get clues are over.Importantly, our app makes this process very simple through automated summarization.

Acknowledgments

It's a twisting road leading to the concrete overall purpose and milestones we wish to achieve. All along, there are people who have made their contribution behind the scenes. Acknowledgments and thanks First things first, this article looks at the significance of acknowledging. One needs to thank all who are due a pat on the back in intervals by subject matter correspondent Xiao Hanjiang. Miss Sonaish Aslam and the staff at ASSET (home internet) or seniors from various universities, as well as technical staff members The author hopes that all of these supervisors will be introduced to you. Moreover, we must not forget our family and friends to whom a special word of thanks is due for their firm support on all the adventures that have happened so far. Suppose that we also give a warm thank you to our principle and instructor Miss Sonish Aslam, who gave us so much valuable guidance. Her experience and ongoing encouragement have had an essentially important role in our personal growth. And now that we possess the ability to excel, thanks in large part to her valuable insights, We would also like to express our appreciation in the business world and even among relatives and friends who have encouraged us. Their unlimited love, encouragement and belief in our own ability have carried us on toward goals to be achieved at every station along the road of success.

But what we have is very much the result of our staff's efforts at SSUET. Its staff members range from the administration personnel whose job it is to keep everything running smoothly, right through to those who supply technical support and assistance. Their efforts have not gone unnoticed. Indeed, to say that our achievements are due some degree to the dedication and commitment of our university staff is not an exaggeration.

The experience of our seniors in college has had some impact on their journey. From their suggestions, we have learned many new things. Through their generosity in sharing what they know has been of enormous help to us.

Introduction to Group Members

Muzammil Sardar Abbasi (GL) (2020-SE-165)
Developer / UI-UX Designer / Group Leader
Contact: 0309-3355709, SE20-165@gmail.com

Imad Khan (2020-SE-176)
Sub-Developer / UI-UX Designer
Contact: 0332-XXXXXX, SE20-176@gmail.com

Syed Muhammad Zejah Ali Rehmani (2020-SE-154)
Firebase-Developer / Database-Handling
Contact: 0331-XXXXXXX, SE20-154@gmail.com

Muhammad Ahsan Siddique (2020-SE-218)
Concept Designer / Project Planning
Contact: 0331-XXXXXXX, SE20-218@gmail.com

SIR SYED UNIVERSITY OF ENGINEERING & TECHNOLOGY
University Road, Karachi-75300, Pakistan
Tel. : 4988000-2, 4982393-474583, Fax: (92-21)-4982393

CERTIFICATE OF COMPLETION

This is to certify that the following students

Muzammil Sardar Abbasi	2020-SE-165
Imad Khan	2020-SE-176
Syed Muhammad Zejah Ali Rehmani	2020-SE-154
Muhammad Ahsan Siddique	2020-SE-218

Have successfully completed the requirements for Final Year Project titled

AUTOMATIC TEXT SUMMARIZATION

In the report submission for the Degree of Bachelor of Science in Software Engineering

---

Dr. Muhammad Naseem
Associate Professor
SSUET

Abstract

Unable to sort through the crush of information bombarding them. Information overloading is proving one the net's explosive expansion. And much information can be obtained from the internet condensing the salient points down into a precis would probably be helpful to many others. For humans, Summarizing the fabric of textbooks in large volumes is wearisome to do by hand. Researchers have been through technical approaches to enhance precis advent!The conference is co-sponsored by Nan Tien Temple of The way of humans and the way of machines pass through each other.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

You pick one at random to look inside–where there's ton after ton of treasure stores, but you still have no idea how much seek-and-find work it will take before that needle has been hit for tasty tidbits. This is the daily dilemma which we, as information seekers are faced with. But the crux of it is that existing automated text summarization tools are themselves limited in their capabilities. They say they will simplify this process, but often enough get it wrong. A lack of tailored attention to individual needs lies at the root of many problems in these procedures. In a sense, it's like having you miniscule personal assistant that can easily pull together the wisdom from mountains of information

Text Summarization As an application of fuzzy information management, this is a process in which the nuggets of truth are squeezed out from long-winded articles and news items.The whole point of this process is to take the content that used to be described in paragraphs and condense it into a form as close to a summary as possible, while not losing any vital information, and it should include following sections:

1.1 Overview

Life on that giant frontier of the digital world, where there are so many articles and news items to be absorbed in among all these blogs–it's tough. Just trying everyday not get washed away by a tidal wave crushing you to death is very hard work. Think of a library that's always open, full to the brim with books containing all kinds o f information. You could spend months or even years in wind through those shelves looking for . A fraction of what you need. That is the everyday dilemma facing us as information seekers.

It's all down to the limitations of current automated text summarization. They will tell you this is going to be smoother, but what they are missing is that personal touch which can adjust subtleties of individual needs. This is like having an assistant who understands your tastes, is aware of what interests you and can filter the essence from vast amounts of information.

Therefore, this is the response to that challenge– it's a humanizing of how we work with information. The core of the problem isn't just information overload but lack of a light leading people through this technological maze. It's about solving the frustration of information seekers who have lost their bearings lacking more than a sea-fullof words. Basically what our project is aiming to build up can be summarized as an automated text summary method which not only makes the process easier, but also more human in a way. It's about build-

ing an electronic wing man that understands the person behind the screen. Amidst a sea of information exploding upon our screens, it offers up something familiar and personal in this anything-but-intimate cyberspace world. Our goal is to change the digital journey from one of whimsical wandering to a guided trek full of places, visitors and itineraries tailored for users.

1.2 Problem Statement

In a sea of information, with articles tripping over each other and playing hide-and seek on our screens, surfers are often left ogling the proverbial Viagra ad. It's commonplace to locate that elusive needle in the haystack. Today, especially, there is information overload. But the innovations that are obtainable today adhere to to save us, but they fail because we desire a personal touch that is aware of our preferences. Enter the issue we're eager to address: the inadequacies of modern artificial text describing methods. While the aforementioned instruments are a step along an appropriate orientation, they cease short of tailoring a plan of action. As an instance, you might use an automated summary tool. However, it fails to utterly follow your reasoning. It's equivalent to talking to an artificial intelligence which cannot recognize you.

But here's the rub–our problem statement is not simply about information overload more generally. This is all about tackling this problem in the context of current programs and applications. There are already plenty of text summarizing tools in the digital realm, but none possesses sufficient deftness to know just what users like.This is the point when our smartphone application kicks in, and it isn't the typical one. It is built on Flutter and has a user-friendly layer as well as an extensible aspect. Flutter provides a touch of enchantment to the user experience, making it softer and more engaging. The real secret sauce, however, is our use of the TFLite model. This is no longer just the art of condensing text; it's about doing so more intelligent, instinctively, and with a personal touch that avoids being swallowed up amidst all those applications.

Our app is not just a solution to the problem, but an antidote to inadequate existing tools. . . This is the missing component. It understands not solely whether to put things into perspective, but additionally what to do it in an efficient manner that is significant to consumers. Essentially, it's more than about cruising the currents of the cyber ocean; there's a new customized methodology to storm information retrieval that Flutter and the TFLite model aspired to. The proverbial needle in the haystack Information overload is all too common. Even if such present instruments promise to save us from our deaths, they still lack that human touch, the capacity to comprehend any specific person's likes.

- Identify the challenge that we are seeking to resolve: the deficiencies of existing automatic text summarizing technologies.These are fine tools to get one started, but they fall short of a customized plan. For instance, you utilize an automated summary tool only to find that it got a few things wrong.

- But now comes the twist. However, our problem statement is not just one of general information overload. Programs and applications already exist to combat this problem.

Already in the digital world are many tools promising to condense text.

- That's the trick–our problem statement isn not that information is overloading us in general. But it is the problem of how to solve this difficulty in current applications and systems. In the digital world of text summarization There are programs that have been developed, but they show a false sensitivity for reading by people.

At the other conjunction, our smartphone app performs as a supervisory; on the contrary hand, it is not your typical app. It is built on Flutter, an accessibility and interactivity layer. Flutter, on the contrary conjunction, adds a touch of freshness to it, rendering the user experience more enjoyable. The TFLite model, on the other hand, is our true secret sauce. But you don't simply have to paraphrase and speak a few words. You must act quickly, Response This text has no mistakes.But our program does more than solve a problem. It also spots flaws in existing tools. The key is that last one, which is bloody fragmented and can't even get what the significance of summary itself means in terms of connecting with consumers. In fact, we don't just surf the wave. We ride behind with our own tailor-made information retrieval strategy in tow and it all happens through Flutter and TFLite.

1.3 System Features

With TFLite model integration and easily configurable pages, Flutter was just the right tool for our ambition to develop a great automatic text summarization app. Our TensorFlow Lite (TFLite) models have been integrated in our applications in order so we can implement revolutionary innovations behind the scenes. Consider that for instance that you wore an exceedingly intelligent personal assistant that can quickly scan understand and interpret difficult textual information. To ensure that users get an outstanding experience using this product is super easy yet thoughtfully designed–by focusing on several key features not mentioned above. Those include pre-configured templates (like scripts used in film work), topic classification plug:

1.3.1 Text Input:

Allow users to input text for Forget the tedious process of inputting articles or writings. One is allowed to do at will as one would share stories with a friend-just say it and go on without your memory playing tricks on you again. The way we've simplified it is that you tell us what content has got to be summarized and in which format, giving the earliest possible date.

1.3.2 TFLite Model Integration:

Integrate a trained TFLite model for text summarization. On the other hand, in order to leverage cutting-edge technology here behind closed doors is our app's incorporation of TensorFlow Lite (TFLite) models. It's like a conducting robot, or an assistant of the mind with as sharp a brain but capable of deciphering and processing complex textual information.Convert the model to a format compatible with Flutter (e.g., TensorFlow Lite Flutter Plugin).

1.3.3 Summarization Processing:

In this respect, our app is particularly well equipped. Advanced algorithms allow it to process and compress the information in input text into concise summaries that are easily understandable. This is the place when the enchantment happens. We evaluate and derive the article's highlights using this smartphone application employing modern facilities algorithms. It's akin to having a really competent editor condense a long piece of writing to its essentials. Sometimes, you can have a competent editor with all their experience as they shorten an article which is too long.

1.3.4 User Interface (UI):

Imagine a sleek, simple-to-operate interface that takes you through the process of summarizing without any fuss whatsoever so it is both an entertaining and efficient way to spend your time. Our app is easily navigated and coupled with a minimalist color scheme, it looks beautiful too. Imagine an easy and user-friendly interface that gently leads you step by step through the process of completing block form summaries, thus making as much fun to use in practice as it is on paper.

1.3.5 Firebase Integration:

In addition to handling user authentication, Firebase also becomes a serverless data storage solution for the generated summaries. For the pleasure of users and accessibility, we match Firebase into our app. Firebase doesn't just handle the user authentication, though. It also plays a key role as our serverless data storage solution for storing additional copies of your site content in this summary form, and makes sure they are only accessible if users can successfully pass an authorization check to get into PAILON– which means Assumi itself works hard to make sure that not everyone with internet service automatically. Thus all your users' summary queries receive secure and efficient accesses from their various local machines without causing you any extra inconvenience of building backend capability or high uncertainties brought by service providers: they are processed entirely on-the-fly on your cloud platform checks out some

1.3.6 User Authentication:

We take concerned about your security and privacy and will continue to be concerned about them. While Firebase handles user authentication, you get to choose who can view summaries created by you but this should not deter you from maintaining your personal information. It's like as if your virtual world had a trusted bouncer on the door.

1.3.7 Cloud Functions:

Nonetheless, Cloud Functions are the ones that give the app its responsiveness and function. In this case, think of it as having a backstage organization up front where all steps right from user input to processing for summarization become smooth and efficient.

1.3.8 Data Storage:

Apart from looking after authentication, Firebase is there for safe server-less data storage. After they have been summarized, your documents are stored neatly for easy retrieval when required.

1.3.9 Security:

We keep your data is safe in our app. Our application has strong security so only authorized people can access the summarized documents hence they are protected by our system. Thus, it is more like an electronic fort guarding your vital information.

1.4 Project Scope

The cause of our automatic text summarization (ATS) application is to help produce the precis summary of any article containing any sort of crucial and applicable information from the unique documented textual content. Summarization is the venture of condensing a piece of file into its shorter variant, which lessens the initial record's size even as retaining the important piece of information intact.

1.5 Chapter Summary

Touring the Expansive Digital Plain In today's world of information overload, we look within Flutter at how automatic text summarization can integrate TFLite models. For example, imagine a library of unlimited articles and news–here we can fish around for many valuable nuggets. While automated text summarization tools are marketed as being convenient, they often end up missing the mark with a lack of personalized service that leaves people feeling ill at ease and in need of some humanity.

1.5.1 Overview:

This chapter provides a broad introduction to the automatic text summarization system developed using TensorFlow Lite (TFLite) model integrated into a Flutter front-end with Firebase serving as the back-end.Discuss the significance of automatic text summarization in handling information overload and enhancing content accessibility.Highlight the goals and objectives of the system, emphasizing its potential impact on various industries and user experiences.

1.5.2 Problem Statement:

A continual battle Our journey takes us into the world of automatic text summarization using Flutter with integration of a TFLite model through the channels open to software engineers. Just think-who wants to read all those articles and news, reading like the sea destroys the nugget? Modern automated summarization systems offer ease of use, but in fact their applications are far from satisfactory. We want some humanity brought back into our information searching activities as well–that would be something precious indeed!

1.5.3 System Features:

Our automatic text summarization application makes use of Flutter, then integrates with a TFLite model to remap the terrain by means of system features. Fundamentally, it is a user-friendly interface that goes beyond the 3Cs of textual content. Our magic wand Flutter allows us to provide users with an interactive browsing experience on both iOS and Android, eliminating the need for two sets of programs. The language of technology is universal after all.

Adding a TFLite model puts forward the technology equivalent to having an assistance who can scan, analyze and read large quantities of information before boiling down what it has seen into practical summaries. This intelligent summary linkage process, conforming to individual preferences ensures that the resulting summaries are not only informative but also easy on the eye. The TFLite model is like the wizard behind that curtain, making something as difficult to deal with as text into a piece of convenient software. Firebase integration allows us to improve the security and convenience of our application. It also offers a secure server for authenticating the user and storing its data, making it into something like a private safety deposit box. With this kind of cloud-based storage, they can access their summarized takenaways at any time.

1.5.4 Project Scope:

Our project involves researching the creation of an original-like automatic text summarization app, utilizing novel technologies like Flutter and TFLite models. Our goal is to provide an easy, convenient tool that makes information searching easier in a digital environment where textual data abnormalities overwhelm. We also aim for the localization of a highly customizable personalized methodology involved in summary writing. This Flutter-powered application is designed to deliver a user experience that seamlessly integrates form with function. Cross-platform by design, Flutter can be used on many different kinds of devices (iOS and Android), increasing the scalability of our offering. This method brings the power of machine learning into every step in the summarization process, and uses TFLite to condense content based on intelligent understanding.

Chapter 2

Literature Review

A literature review on the topic of automatic text summarization using a TensorFlow Lite (TFLite) model in Flutter as the front-end with Firebase as the back-end might include discussions on various aspects related to the technology, methods, and potential challenges. Here's an outline that you can use as a starting point:

2.1 Introduction

Automatic text summarization is a crucial task in natural language processing, with applications ranging from information retrieval to content summarization in mobile applications. The integration of machine learning models on mobile platforms, particularly using TensorFlow Lite in Flutter apps with Firebase as the back-end, presents a promising avenue for developing efficient and scalable solutions..

2.2 Automatic Text Summarization

Research in automatic text summarization has evolved to address challenges in extracting meaningful information from large volumes of text. Extractive and abstractive summarization methods have been extensively studied. Extractive methods involve selecting important sentences, while abstractive methods generate concise summaries using natural language. Previous studies have evaluated the effectiveness of various algorithms, such as graph-based methods, deep learning approaches, and hybrid models.

2.3 TensorFlow Lite (TFLite) Model

The integration of the TFLite model into our automatic text summarization system using Flutter serves as a pivotal element in enhancing both functionality and user experience. Leveraging the power of TFLite, a model specialized in text summarization, we seamlessly embed intelligence into our application. This integration transforms the intricate task of condensing large volumes of text into a user-friendly and humanized experience. By tapping into the capabilities of TFLite, our system acts as an assistant, capable of comprehensively scanning and analyzing information before distilling it into concise and coherent summaries.

2.4 Flutter Framework

The Flutter framework, developed by Google, has gained popularity for its cross-platform capabilities and expressive UI. Research highlights the ease of development, hot-reloading feature, and flexibility of Flutter in creating mobile applications. Studies have explored Flutter's compatibility with machine learning libraries and frameworks, emphasizing its potential for seamless integration with TensorFlow Lite models.

2.5 Integration of TFLite Model in Flutter

Literature on the integration of the TFLite model into our automatic text summarization system using Flutter serves as a pivotal element in enhancing both functionality and user experience. Leveraging the power of TFLite, a model specialized in text summarization, we seamlessly embed intelligence into our application. This integration transforms the intricate task of condensing large volumes of text into a user-friendly and humanized experience. By tapping into the capabilities of TFLite, our system acts as an assistant, capable of comprehensively scanning and analyzing information before distilling it into concise and coherent summaries.

2.6 Firebase as Back-End

Firebase serves as a robust back-end solution for mobile applications, offering features such as real-time databases, authentication, and cloud functions. Research has examined the advantages of using Firebase for data storage, user management, and serverless computing. Case studies illustrate the successful implementation of Firebase in mobile applications, providing insights into its scalability and reliability.

2.7 Mobile App Development and User Experience

The literature emphasizes the challenges and best practices in mobile app development, particularly when integrating machine learning components. Studies discuss considerations for optimizing app performance, minimizing resource consumption, and ensuring a seamless user experience. User interface design principles and usability studies are explored to enhance the overall quality of mobile applications.

2.8 Evaluation Metrics

Evaluation metrics play a crucial role in assessing the performance of automatic text summarization models. The literature reviews commonly used metrics such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation) and BLEU (Bilingual Evaluation Understudy) and discusses their applicability in the context of summarization tasks. Comparative analyses highlight the strengths and limitations of different evaluation metrics.

2.9 Challenges and Future Directions

Challenges identified in the literature include the trade-off between model complexity and inference speed on mobile devices, maintaining real-time summarization capabilities, and ensuring privacy and security in cloud-based back-end solutions. Future research directions may involve addressing these challenges, exploring novel algorithms, and optimizing the integration of machine learning models in mobile applications.

2.10 Conclusion

In conclusion, the integration of automatic text summarization using TensorFlow Lite in Flutter apps with Firebase as the back-end offers a promising solution for efficient and scalable mobile applications. The literature review provides a comprehensive understanding of the current state of research in automatic text summarization, TensorFlow Lite deployment, Flutter development, and Firebase integration, laying the groundwork for the proposed system's significance in advancing the field.

Chapter 3

Design

1. System Components:

a. Hardware Components:

The system mainly sets about on-device processing, relying entirely upon mobile device capability (iOS and Android). Data-accessing, user authentication and remote processing (if necessary) components on the server.

b. Software Components:

Flutter Framework: Forms the core of both user interface and interaction logic. TFLite Model: Used as a singular element in the automatic text summary system. Firebase: Used to secure the storage of data, verify user authentication and enable cloud-based services.

2. User Functions and Interactions:

a. User Roles:

General Users: Access the summarization application. Authenticated Users: Personalized features and protected data storage.

b. User Functions:

Input: The application interface lets users enter text data. Summarization: This TFLite model takes the input text and produces a summary. Personalization: For example, users can personalize their own summarization preferences. Feedback: This provides a feedback loop for users to give input on the quality of summaries.

3. Data Flow and Processing:

a. Input Data:

Flutter interface input text content. Cleaning and formatting of data for ideal model input, also referred to as pre-processing.

b. Output Data:

Text summarized by the TFLite model.

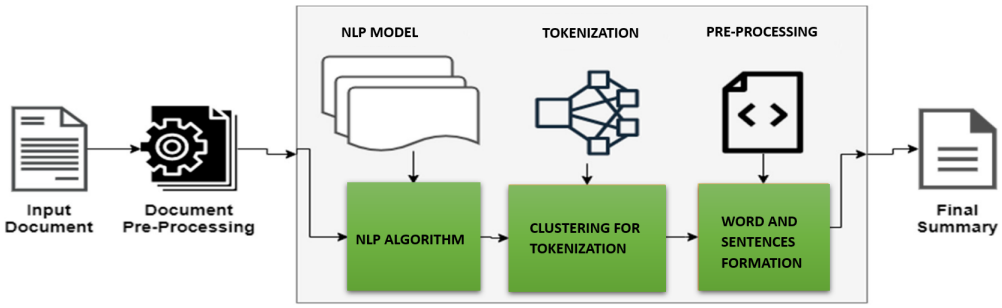


Figure 3.1: BLOCK DIAGRAM

4. User Interface Elements:

a. Flutter UI Elements:

Input Interface: Text input fields and customization options. Output Display: Attractive presentation of the automatically produced summary. Feedback Mechanism: Users being able to react intuitively about the nature of the summarization.

b. User-Friendly Features:

Personalization Options:

Personalize summarization preferences based on what users want. Visual Appeal: A clean layout, aiming for a nice appearance and better usability.

Interactive Elements:

Second, it offers features that interest users and allow them to interact with the summarization process.

3.1 Design Methodology and Software Process Model

When developing a methodology and software process model for automatic text summarization using Flutter with TFLite models, requirements gathering plays an important part. Gaining insight into a system's purpose and the requirements of its users is critical toward creating an environment that automates text summarization yet also makes it more humanistic. That requires clearly defining such objectives as facilitating an effective user experience, the use of customized summarizing services and maintaining a true feel for human expression. Discovering these different easily identified personas allows us to understand certain specific user preferences, which can help insure that the summarization process meets individual expectations. Also, the requirements stage includes a comprehensive analysis of Flutter and TFLite ecosystems.

3.1.1 Requirements Gathering:

As a first step, we are collecting requirements for building the methodology and software process model to achieve automatic text summarization with Flutter. In every stage of design

work identical models will be generated by using TFLite. Scratch around of human touches Hopefully, understanding the goals and requirements of interacting users is crucial to building a system that will not just automate text summaries but also endow them with humanity. This requires a clear definition of what is wanted from the system, for example that it should give smooth access to information, and specify desired requirements such as individualized text summarization. As well they need to consider how much human flavorization detail would be retained? Establishing user personals enables us to understand each individual's special preferences, maintaining that users 'expectations can be met in the course of writing a summary. Moreover, the requirements phase includes an in-depth analysis of the technology environment. Such integrations include Flutter tools and technologies to incorporate TFLite into code loops (for example). This understanding provides the basis for two later steps in system architecture and technology selection as well as a third step involving humanization.

3.1.2 System Architecture:

User Interface (UI) - Flutter:

Description:

Using the Flutter package, they were able to create a smooth and easy-to-use UI component which can be used in their specific website environment for text summarization.

Responsibilities:

Display summarization options. Capture user input and preferences. Showcase the summarized content.

Summarization Engine - TFLite Model:

Description:

In general, the performance of a smartphone camera takes care of processes and summarizes based on pre-trained patterns automated by Flagger in addition to user preferences when generating interpretative output.

Responsibilities:

Integrate with the Flutter UI. Process input text. Generate a concise summary.

Natural Language Processing.(NLP) Module:

This system aims to enhance humanization through the use of natural language processing techniques, like NLP, that are better equipped to maintain contextual threads and excavate deeper for understanding within conversational exchanges. Responsibilities: Passages of text should be studied for their meaning and deeper significance. Maintain the essence of their style and tone when synopsizing.

User Feedback Module:

Description:

To facilitate constant improvement of the produced summaries, allows users to feedback about these designations.

Responsibilities:

Gather readers ' opinions of the quality of summaries. Implement feedback-driven adjustments.

Authentication and Security Module:

Description:

Helps to ensure the application is secure and guards against users 'loss of data.

Responsibilities:

Manage user authentication. Then use data encryption for storage and communication.

System Flow: User Interaction:

What does the Flutter UI do? How is it used by users entering text and refreshing their preferences for summarization.

Input Processing:

Hence what I feed in to the Flutter interface is sent on for handling by this Summarization Engine (TFLite Model).

TFLite Summarization:

TFLite Model Major points in structure - There are many analyses of the content, and after passing through some NLP treatment it becomes more humanized.

Output Presentation:

Flutter then breaks down the collected contents into other small files and uses them as a basis for its own UI.

Continuous Improvement:

Analysis of feedback leads to doubled-down efforts for future enhancements, including improvements in the TFLite model and NLP module.

Authentication and Security:

The Authentication and Security Module maintains an element of security. All data is encrypted, allowing for smooth passage without jeopardizing the integrity of user information in the process.

3.1.3 TFLite Model Integration:

The integration of the TFLite model into our automatic text summarization system using Flutter serves as a pivotal element in enhancing both functionality and user experience. Leveraging the power of TFLite, a model specialized in text summarization, we seamlessly embed intelligence into our application. This integration transforms the intricate task of condensing large volumes of text into a user-friendly and humanized experience. By tapping into the capabilities of TFLite, our system acts as an assistant, capable of comprehensively scanning and analyzing information before distilling it into concise and coherent summaries.

What sets our approach apart is the meticulous attention given to humanization. The TFLite model, acting as the wizard behind the scenes, ensures that the summarization process respects the nuances of language, preserving the original tone and style of the content. This human-centric approach addresses a common pitfall in automated summarization tools, which often miss the mark by neglecting the need for personalized service. Through the fusion of Flutter's interactive interface and the intelligent TFLite model, we strive to provide users with not just convenience, but a summarization experience that feels tailor-made, instilling a sense of ease and familiarity in navigating the expansive digital landscape.

3.2 Architectural design / Design Patterns

In the architectural design of the Automatic Text Summarization System using Flutter with a TFLite model, the system is organized into major subsystems that collaboratively contribute to achieving the complete functionality. The following is a high-level overview, along with a simple Line and Box Diagram illustrating the major subsystems and their connections.

3.2.1 System Architecture Overview:

Front-End (Flutter):

- ***Responsibility:*** User interface and interaction.
- ***Key Components:***
  - Input Interface
  - Result Display
  - Navigation

Back-End (Firebase):

- **Responsibility:** Data storage, retrieval, and user authentication.
- **Key Components:**

- Firestore Database.
- Cloud Function.
- Firebase Authentication.

Text Summarization Model (TFLite):

- ***Responsibility:*** Automatic text summarization.
- ***Key Components:***
  - TFLite Model
  - Model Loading
  - Inference Process

Figure 3.2: BOX AND LINE DIAGRAM

Explanation:

- **Front-End (Flutter):**

  The Flutter front-end handles user input, displays summarization results, and facilitates user navigation.

- **Back-End (Firebase):**
  - Firestore Database stores user data and summarization results.
  - Cloud Functions execute tasks triggered by data changes.
  - Firebase Authentication ensures secure user registration and login.
- **Text Summarization Model (TFLite):**
  - The TFLite model is responsible for automatic text summarization.
  - It loads into the Flutter application and processes input text to generate summaries.

3.3 Process flow / Representation

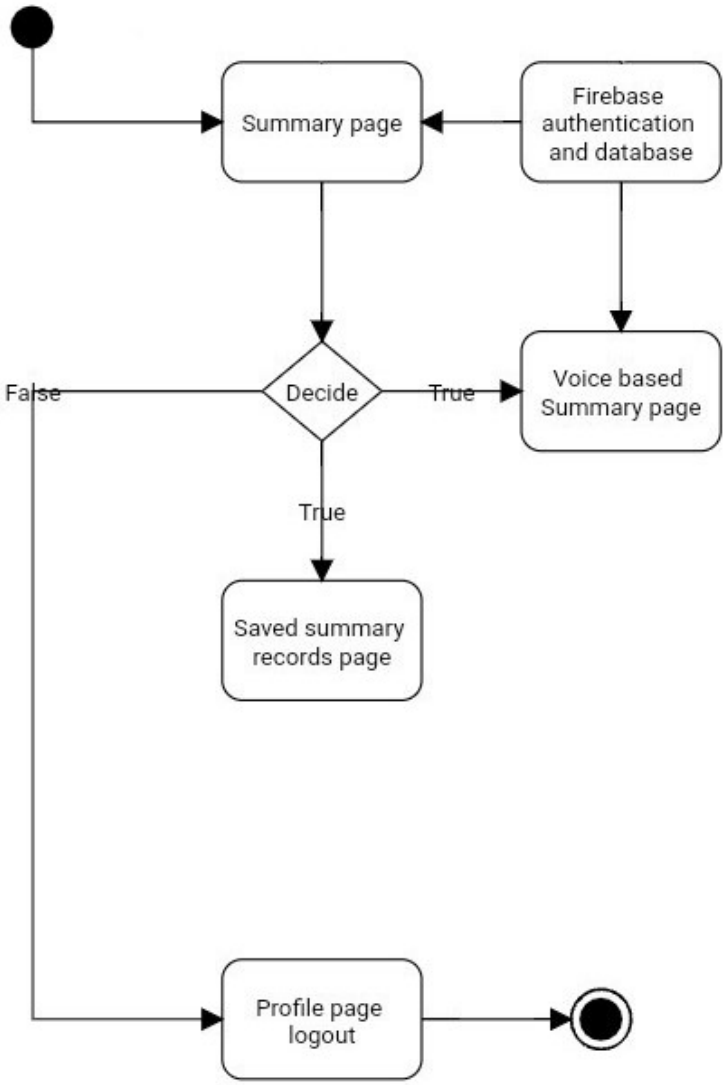


Figure 3.3: ACTIVITY DIAGRAM

3.4 Design models [along with description]

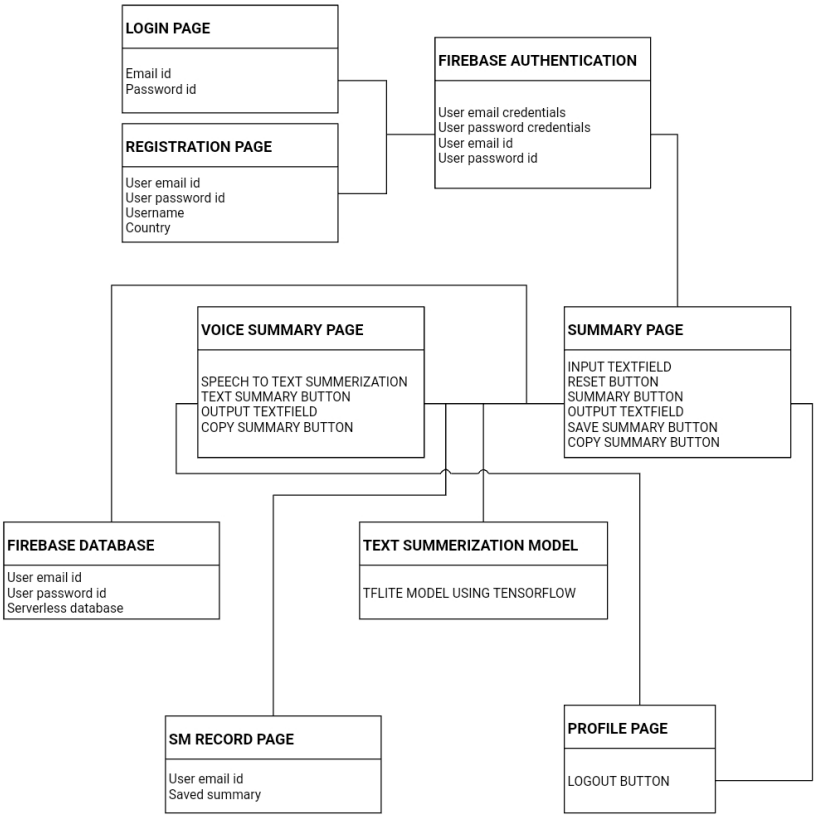


Figure 3.4: CLASS DIAGRAM

- *User logs in the acount using email and password*
- *If the user does not have an account they register after which their details are saved in the firebase console.*
- *Firebase Authenticates the user's credentials and allows them to access the summary page*
- *In the summary page the user can summarize text by coping the text to be summarized in the textfield and press summarize button to gain output.*
- *The user can then use the voice based summary page to provide text through voice to be summarized.*
- *In the SS Records page through the user credentials user is provided access to see their saved summaries.*
- *In the profile page the user can logout og the application.*

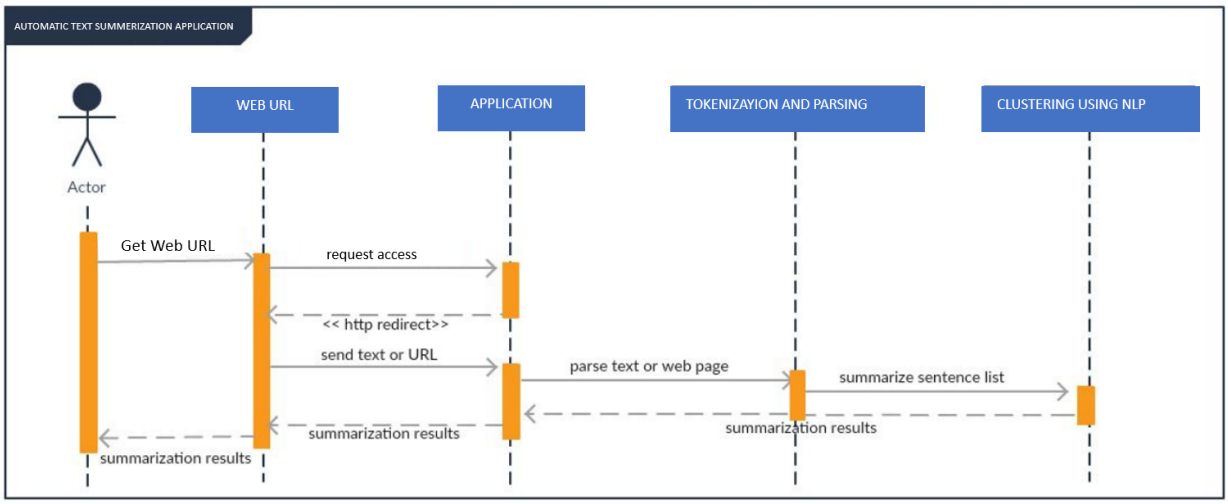


Figure 3.5: SEQUENCE DIAGRAM

- *The user provides text to be summarized in the input text-field.*
- *Then the user presses the summarize button, and through the model, the text is summarized.*
- *The summarized text is then displayed on the output text-field.*
- *The summaries can be saved on the clipboard.*
- *The summaries can be saved on the firebase database.*
- *The saved firebase saved summaries can be displayed on the SS Record page.*

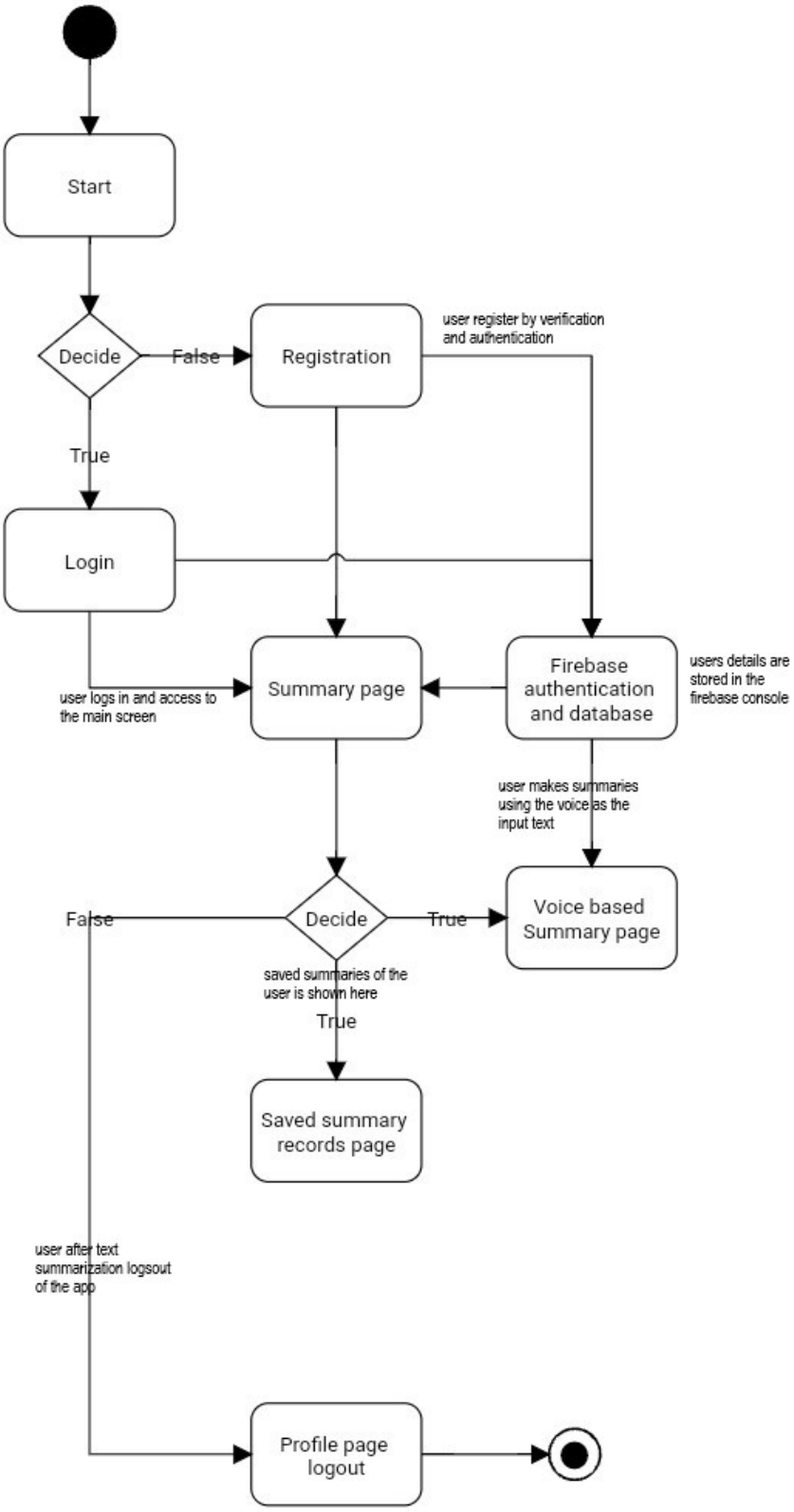


Figure 3.6: STATE TRANSITION DIAGRAM

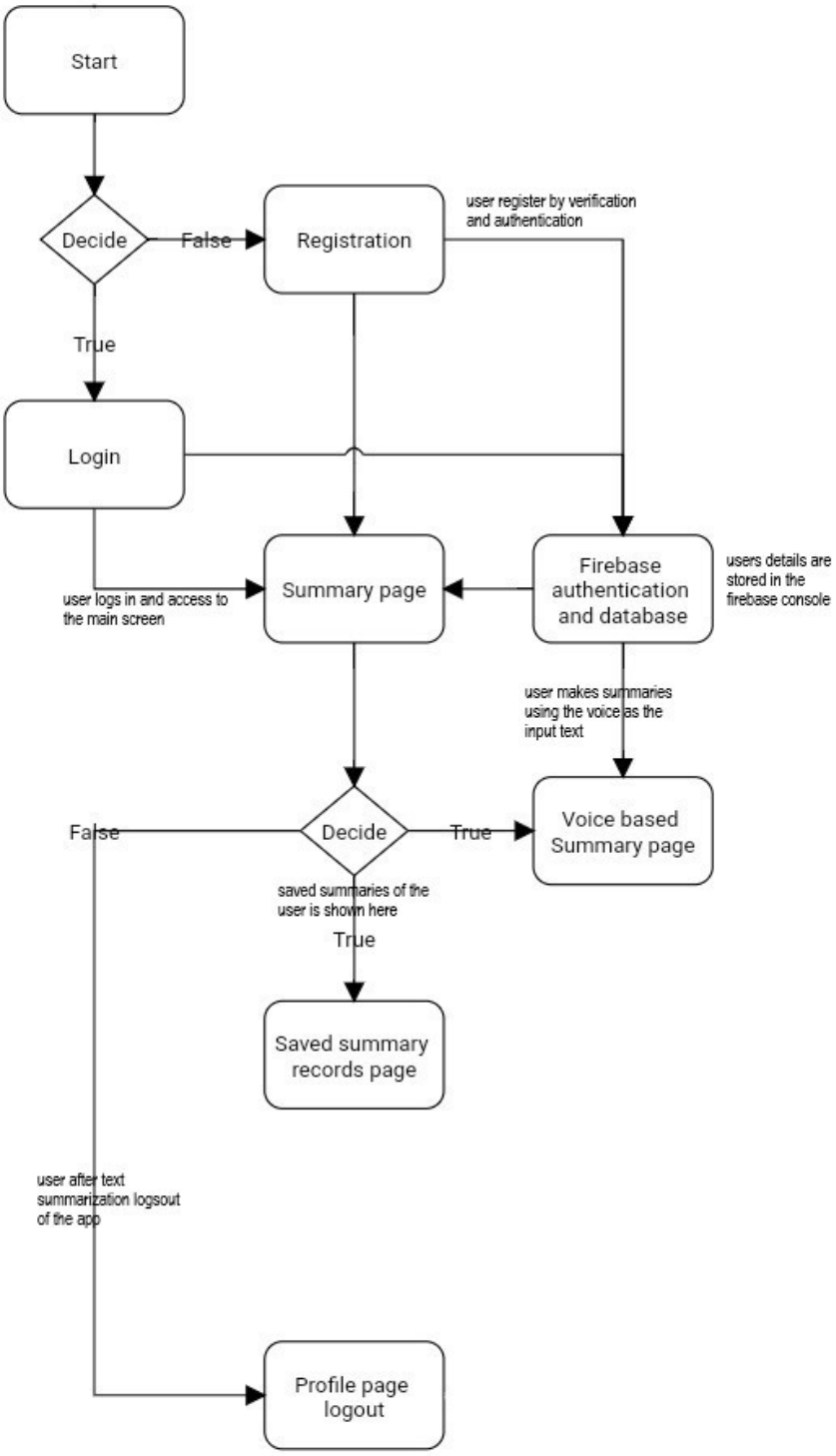


Figure 3.7: DATAFLOW DIAGRAM

1. *User logs in the acount using email and password.*
2. *If the user does not have an account they register after which their details are saved in the firebase console.*
3. *The user provides text to be summarized in the input text-field.*
4. *Then the user presses the summarize button, and through the model, the text is summarized.*
5. *The summarized text is then displayed on the output text-field.*
6. *The summaries can be saved on the clipboard.*
7. *The summaries can be saved on the firebase database.*
8. *The saved firebase saved summaries can be displayed on the SS Record page.*
9. *Firebase Authenticates the user's credentials and allows them to access the summary page*
10. *On the summary page the user can summarize text by coping the text to be summarized in the textfield and press summarize button to gain output.*
11. *The user can then use the voice based summary page to provide text through voice to be summarized.*
12. *On the SS Records page through the user credentials user is provided access to see their saved summaries.*
13. *On the profile page the user can logout og the application.*

3.5 Data Design

the data design involves leveraging Firebase for user authentication and data storage, with Firestore organizing user-related data, input texts, and summarized results. The system operations on structured data to provide the functionalities described in each app page.The data design of the Automatic Text Summarization System using Flutter, Firebase, and TFLite involves the transformation of the information domain into structured data. Below, I'll describe how the major data entities are stored, processed, and organized, and I'll also introduce the relevant data storage items.

3.5.1 Data Storage and Organization:

1. User Data (Firebase Authentication and Firestore):

- - **Storage:** Firebase Authentication is utilized to manage user registration and login. User data (userId, username, email) is securely stored in Firebase Authentication.
- - **Processing**:Firebase Authentication processes user login and registration requests, issuing authentication tokens for secure access.
- - **Organization**:User-specific data is organized under the "users" collection in Firestore.

2. Input Texts (Firestore):

- - **Storage:** User input texts (inputId, userId, text) are stored in the Firestore database under the "inputTexts" collection.
- - **Processing**: Texts are processed for ideal model input, including cleaning and formatting before summarization.
- - **Organization:** The "inputTexts" collection organizes user input texts.

3. Summaries (Firestore):

- - **Storage:** Summarized results (summaryId, inputId, userId, summaryText) are stored in the Firestore database under the "summaries" collection.
- - **Processing**: Summaries are generated using the TFLite model and stored in Firestore.
- - **Organization:** The "summaries" collection organizes user-specific summary data.

3.5.2 Database and Data Storage Items:

1. Firebase Authentication:

- - **Purpose:** Manages user registration and login.
- - **Data Stored:**
- - **'firebaseUserId' (Type: String)**: Unique identifier assigned by Firebase for user authentication.
- - **'authenticationToken' (Type: String):** Token issued by Firebase for secure user authentication.

2. Firestore Database:

- - **Purpose:** Stores user data, input texts, and summarized results.
- - **Collections:**
- **- 'users':** User profiles and associated data.
- **- Fields:** userId, username, email.
- - **'inputTexts':** User input texts for summarization.
- - **Fields:** inputId, userId, text.
- - **'summaries':** Summarized results.
- - **Fields:** summaryId, inputId, userId, summaryText.

3.5.3 Pages and Corresponding Data Operations:

1. Main Screen Page:

- - **Background Video:** Visual content, not directly related to structured data.
- - **Login Button:** Initiates user authentication with Firebase.
- - **Google Login Button:** Initiates Google-based authentication.

2. User Login Page:

- - **User Authentication:** Utilizes Firebase Authentication.
- **- Login Button:** Triggers the Firebase login process using user credentials.

3. User Registration Page:

- **User Authentication:** Utilizes Firebase Authentication. - **Registration Form:** Gathers user registration data. **- Save Data**: Saves user data in Firebase Authentication.

4. Summary Page:

- - **Input Field:** Accepts user input for summarization.
- - **Summary Button:** Triggers the summarization process using the TFLite model.
- - **Save Summary Button:** Saves the generated summary in the Firestore database.
- - **Copy Summary Button:** Copies the summary to the clipboard.

5. Voice-Based Summary Page:

- **- Voice Input Button:** Initiates voice input from the user.
- - **Text Summary Button**: Summarizes text obtained from voice input.
- - **Output Text Field:** Displays the generated summary.
- - **Copy Summary Button:** Copies the summary to the clipboard.

6. Record Page:

- **Displays Records:** Retrieves and displays records of saved summaries from Firestore.

7. Group Information Page:

- - **Logout Button:** Triggers user logout from the system.
- **- Project Makers Details:** Static information, not directly related to structured data.

3.6 Data Dictionary

This data dictionary provides details for the entities and major data related to each specific page and its functionalities in the Automatic Text Summarization System using Flutter, TFLite model, and Firebase. Data dictionary for the entities and major data related to the specific pages and functionalities you mentioned in the Automatic Text Summarization System built with Flutter, Firebase, and TFLite:

3.6.1 Onboarding Screen Page:

UI Elements:

1. Login Button:

- - **Type:** Button
- - **Description:** Initiates the user login process.
- - **Functions:**
- - **'onClick()'**: Triggered when the user taps the login button.

2. Google Login Button:

- - **Type:** Button
- - **Description:** Allows users to log in using their Google credentials.
- - **Functions:**
- - **'onClick()':** Triggered when the user taps the Google login button.

3.6.2 User Login Page:

UI Elements:

1. Email Input Field:

- - **Type:** Text Input Field
- - **Description:** Allows users to input their email for login.
- - **Attributes:**
- - **'email' (Type: String):** Stores the user's email input.

2. Password Input Field:

- - **Type:** Password Input Field
- - **Description:** Allows users to input their password for login.
- - **Attributes:**
- - **'password' (Type: String):** Stores the user's password input.

3. Login Button:

- - **Type:** Button
- - **Description:** Initiates the user authentication process with Firebase.
- - **Functions:**
- **- 'onClick()':** Triggered when the user taps the login button.

3.6.3 User Registration Page:

UI Elements:

1. Username Input Field:

- - **Type:** Text Input Field
- - **Description:** Allows users to input their desired username for registration.
- - **Attributes:**
- **- 'username' (Type: String)**: Stores the user's desired username.

2. Email Input Field:

- - **Type:** Text Input Field
- - **Description:** Allows users to input their email for registration.
- - **Attributes:**
- **- 'email' (Type: String):** Stores the user's email input.

3. Password Input Field:

- - **Type:** Password Input Field
- - **Description:** Allows users to input their desired password for registration.
- - **Attributes:**
- **- 'password' (Type: String):** Stores the user's desired password.

4. **Register Button:**
   - **Type:** Button
   - **Description:** Initiates the user registration process with Firebase.
   - **Functions:**
   - **‘onClick()‘:** Triggered when the user taps the register button.

3.6.4 Summary Page:

UI Elements:

1. **Text Input Field:**
   - **Type:** Text Input Field
   - **Description:** Allows users to input text for summarization.
   - **Attributes:**
   - **‘inputText‘ (Type: String):** Stores the user’s input text.

2. **Reset Button:**
   - **Type:** Button
   - **Description:** Clears the text input field.
   - **Functions:**
   - **‘onClick()‘:** Triggered when the user taps the reset button.

3. **Summary Button:**
   - Type: Button
   - **Description:** Initiates the text summarization process.
   - **Functions:**
   - **‘onClick()‘:** Triggered when the user taps the summary button.

4. **Summary Output Field:**
   - **Type:** Text Output Field
   - **Description:** Displays the generated summary.
   - **Attributes:**
   - **‘summaryText‘ (Type: String):** Stores the generated summary.

5. Save Summary Button:

- - **Type:** Button
- - **Description:** Saves the summary in the Firebase database.
- - **Functions:**
- **- 'onClick()':** Triggered when the user taps the save summary button.

6. Copy Summary Button:

- - **Type:** Button
- - **Description:** Copies the summary to the clipboard.
- - **Functions:**
- **- 'onClick()':** Triggered when the user taps the copy summary button.

3.6.5 Voice-Based Summary Page:

UI Elements:

1. Speak Button:

- - **Type:** Button
- - **Description:** Allows users to speak.
- - **Functions:**
- **- 'onClick()':** Triggered when the user taps the speak button.

2. Text Summary Button:

- - **Type:** Button
- - **Description:** Summarizes text from the user's spoken words.
- - **Functions:**
- **- 'onClick()':** Triggered when the user taps the text summary button.

3. Output Text Field:

- - **Type:** Text Output Field
- - **Description:** Displays the generated summary from voice input.
- - **Attributes:**
- **- 'summaryText' (Type: String):** Stores the generated summary.

4. Copy Summary Button:

- - **Type:** Button
- - **Description:** Copies the voice-based summary to the clipboard.
- - **Functions:**
- **- 'onClick()':** Triggered when the user taps the copy summary button.

3.6.6 Record Page:

UI Elements:

1. Summary Records Display:

- - **Type:** List or Table
- - **Description:** Displays records of saved summaries from the Firebase database.
- - **Attributes:**
- **- 'recordList' (Type: List):** Stores the records of saved summaries.

3.6.7 Group Information Page:

UI Elements:

1. Logout Button:

- - **Type:** Button
- - **Description:** Allows users to log out from the application.
- - **Functions:**
- **- 'onClick()': Triggered** when the user taps the logout button.

2. Project Makers Details:

- - **Type:** Text or Section
- - **Description:** Displays information about the makers of the project

3.7 Chapter Summary

The chapter on "Automatic Text Summarization with Design Methodology and Software Process Model" provides a comprehensive exploration of the key elements involved in the process The section starts by explaining the big design method and software process model that is the base for the whole system. This way of doing things shows how to make a text summarizing system in a step-by-step manner.

The way buildings are designed and the style used are looked at closely as important parts of the system's structure. This part talks about the big picture setup of the system, focusing on using winning design ideas to handle certain problems and improve system strength and flexibility.

The chapter then talks about the steps involved and how they are shown in the field of automatic text summarization. It explains the steps and communications needed to change plain text into short and helpful sums. Pictures help us understand how things change during the process. Design models are important in the chapter. They help us understand different models used to build text summarizing systems. Each design is explained well, focusing on its goal, features and use within the bigger plan.

As the story gets more intense, it's time to focus on design models - the unnoticed stars of the narrative. Each model comes out into the spotlight, described in a lot of details, showing its purpose, special traits and how it's used within the main design plan. It's like an orchestra of models that work together to create the best story. The story gets more exciting as it emphasizes the design of data and making a complete list of all data aspects.

These parts are important builders. They show how data parts fit together, their connections and the big plan that orders how they're organized. The data dictionary is like a treasure chest full of info. It shows exact and simple descriptions of each part of the data, its features and what they can do.

Finally, the last part brings together these different parts into one complete picture. The careful study of design methods, software models, building plans, patterns for designs, flow of work, data design and making a dictionary for information helps readers learn about the important design things needed to make an automatic machine that can quickly summarize texts. It's a story of complexity, imagination, and finally, successfully putting together a tech wonder.

Chapter 4

System Development

Developing an automatic text summarization system involves multiple steps, and documentation for such a system would typically cover various aspects, including architecture, technologies used, implementation details, and more. Below is a high-level outline for documenting a system that includes a Flutter front-end using a TensorFlow Lite (TFLite) model and Firebase as the back-end. Keep in mind that the details may vary based on your specific implementation and requirements.

4.1 Introduction

Imagine diving into an effortless and state-of-the-art adventure with our automatic text summarization application. Envision a user-friendly app that not only takes the hassle out of wading through endless textual content but also adds a touch of personalization to how summaries are crafted. It's like having your own storytelling companion that tailors summaries just for you. And here's the captivating part: we're crafting this experience with Flutter, the enchanting tool that not only makes our app's interface magical but also invites everyone into the adventure through its cross-platform charm.

4.1.1 Purpose and Significance

In the age of information overload, the ability to quickly extract the essence of lengthy documents or articles is crucial. Our Automatic Text Summarization System addresses this need by leveraging state-of-the-art technologies to provide users with concise and coherent summaries of input text. Whether used for research, content curation, or simply time-saving, the system aims to enhance the way individuals interact with and comprehend large volumes of written content.

4.1.2 Goals of the System

1. Efficiency: Enable users to obtain relevant information quickly by generating accurate and coherent text summaries.

2. User-Friendly Interface: Design an intuitive and visually appealing Flutter-based front-end that ensures a seamless user experience.

3. Scalability: Leverage Firebase as a robust and scalable back-end solution to store user data and facilitate real-time communication.

4. Innovation: Utilize TensorFlow Lite for the text summarization model, allowing for efficient inference on mobile devices.

4.2 System Architecture

Our Automatic Text Summarization System is carefully crafted to seamlessly blend the user-friendly front-end, powered by Flutter, with the robust back-end, supported by Firebase. At the heart of the summarization magic is TensorFlow Lite (TFLite), making the process intelligent and efficient. Let's dive into the inner workings of each component to get a feel for the system's soul.

4.2.1 Front-End (Flutter)

Our Flutter-powered front-end is like the friendly face of our system, making interaction a breeze. It handles user input, showcases the results, and dances with Firebase in the background.

User Interface Components:

- **Input Interface:** Welcomes users to pour in their text for a magical summarization journey.
- **Result Display**: Showcases the beautifully generated summaries to our users.
- **Navigation**: Guides users seamlessly through different sections, creating a delightful user experience.

Integration with TFLite

- **Model Inference**: Employs the power of TFLite to weave the magic of text summarization.
- **Asynchronous Processing**: Ensures the magic happens without making our users wait.

4.2.2 Back-End (Firebase)

Firebase, our trusty back-end companion, handles the heavy lifting, providing real-time data storage and retrieval capabilities.

Firestore Database

- **User Data Storage**: Safely stores user information and the text they want summarized.
- **Summarized Results:** Keeps a record of the wonderfully crafted summaries associated with each user.

Cloud Functions

- **Data Triggers:** Springs into action when data changes, ensuring our system stays updated and responsive.
- **Asynchronous Operations:** Handles tasks that can take their time, letting our system breathe and work its magic.

4.2.3 Communication Between Front-End and Back-End

The communication between the Flutter front-end and Firebase back-end is crucial for the overall functionality of the system.

- **Data Exchange:** Describes how user input and summary results are exchanged between the front-end and back-end.

4.2.4 TensorFlow Lite Model

The TensorFlow Lite model is responsible for the automatic text summarization process. This section provides insights into its integration with the Flutter front-end.

- **Model Loading:** Details on how the TFLite model is loaded into the Flutter application.
- **Inference Process:** Explanation of how the model processes input text and generates summaries.

4.3 Technologies Used

The Automatic Text Summarization System leverages a range of technologies and tools to deliver an efficient and seamless user experience. Below is an overview of the key technologies used in different components of the system.

4.3.1 Front-End Technologies (Flutter)

Flutter Framework

- **Description**: An open-source UI software development toolkit for creating natively compiled applications.
- **Purpose**: Powers the cross-platform mobile application for user interaction.

TensorFlow Lite (TFLite)

- **Description**: A lightweight version of TensorFlow designed for mobile and edge devices.
- **Purpose**: Drives the automatic text summarization model on the Flutter front-end.

4.3.2 Back-End Technologies (Firebase)

Firestore Database

- **Description**: A NoSQL cloud database for storing and syncing data across devices.
- **Purpose**: Manages user data and stores summarization results.

Cloud Functions

- **Description**: Serverless functions that run in response to events triggered by Firebase features.
- **Purpose**: Performs asynchronous tasks, such as updating summaries on data changes.

Firebase Authentication

- **Description**: Firebase service for authenticating users.
- **Purpose**: Ensures secure user registration and login.

4.3.3 Machine Learning Technology

TensorFlow Lite Model

- **Description**: A machine learning framework for deploying models on mobile and edge devices.
- **Purpose:** Powers the automatic text summarization functionality on the Flutter front-end.

4.3.4 Development Environment

Dart Programming Language

- **Description:** The programming language used with the Flutter framework.
- **Purpose:** Enables the development of cross-platform mobile applications.

Visual Studio Code

- **Description:** A source-code editor developed by Microsoft.
- **Purpose:** Provides an integrated development environment (IDE) for coding Flutter applications.

4.4 Front-End Development (Flutter)

The front-end of the Automatic Text Summarization System is built using the Flutter framework, a powerful and versatile UI toolkit. This section provides insights into the design principles, key components, and the integration of the TensorFlow Lite model for automatic text summarization.

4.4.1 Design Principles

User-Centric Design

- **Description**: Prioritizing user experience and intuitive interactions.
- **Implementation**: Utilizing Flutter's widget system for building responsive and visually appealing interfaces.

Cross-Platform Compatibility

- **Description**: Ensuring the application works seamlessly across different platforms.
- **Implementation**: Leveraging Flutter's single codebase for both iOS and Android platforms.

4.4.2 Key Components

Input Interface

- **Description**: Allows users to input text for summarization.
- **Implementation**: Text input fields, validation, and user-friendly controls.

Result Display

- **Description:** Presents the generated summaries to users.
- **Implementation**: Dynamic widgets for displaying text summaries with proper formatting.

Navigation

- **Description**: Enables users to navigate between different sections of the app.
- **Implementation**: Flutter's navigation system for a seamless user journey.

4.4.3 TensorFlow Lite Integration

Model Loading

- **Description**: Loading the TensorFlow Lite model into the Flutter application.
- **Implementation**: Utilizing Flutter packages for loading and managing the TFLite model.

Inference Process

- **Description:** Processing input text through the TensorFlow Lite model for summarization.
- **Implementation**: handling model inference asynchronously to maintain a smooth user experience.

4.4.4 Asynchronous Processing

- **Description**: Managing tasks such as model inference without blocking the UI.
- **Implementation**: Flutter's asynchronous programming features for responsive applications.

4.4.5 User Authentication

Registration and Login

- **Description**: Secure user authentication using Firebase.
- **Implementation**: Integrating Firebase Authentication services into the Flutter app.

4.5 Back-End Development (Firebase)

The back-end of the Automatic Text Summarization System is powered by Firebase, providing real-time data storage, authentication, and serverless functions. This section outlines the key components and functionalities implemented on the Firebase platform.

4.5.1 Firestore Database

User Data Storage

- **Description**: Storing user information and input text for summarization.
- **Implementation**: Firebase Firestore collections and documents to organize user data.

Summarized Results

- **Description**: Persisting the generated summaries associated with each user.
- **Implementation**: Storing summarized results in Firestore for easy retrieval.

4.5.2 Cloud Functions

Data Triggers

- **Description**: Executing functions in response to data changes (e.g., updating summaries).
- **Implementation**: Writing Cloud Functions triggered by Firestore events.

Asynchronous Operations

- **Description**: Handling tasks that do not need to be completed in real-time.
- **Implementation**: Defining asynchronous Cloud Functions for background processing.

4.5.3 User Authentication

Firebase Authentication

- **Description**: Managing user registration and login securely.
- **Implementation**: Integrating Firebase Authentication services for user identity management.

User Data Security

- **Description**: Ensuring user data is accessible only to authorized users.
- **Implementation**: Configuring Firebase security rules for Firestore.

4.5.4 Communication Between Front-End and Back-End

Data Exchange

- **Description**: Facilitating communication between Flutter front-end and Firebase back-end.
- **Implementation**: REST-full API calls for data exchange.

4.5.5 Firebase Console

Include screenshots of the Firebase console to showcase the configuration settings, Firestore collections, and Cloud Functions.

4.6 Text Summarization Model (TFLite)

The core functionality of the Automatic Text Summarization System is powered by a TensorFlow Lite (TFLite) model. This section provides insights into the selection of the model, its integration with the Flutter front-end, and the process of text summarization.

4.6.1 Model Selection

TensorFlow Lite Models for Text Summarization

- Description: Overview of available TFLite models suitable for text summarization.
- Selection Criteria: Criteria used for choosing a specific model for integration.

Trained Model Details

- Description: Specifics of the pre-trained TFLite model used.
- Training Data: Information about the dataset used for training the model.

4.6.2 Integration with Flutter

Loading the TFLite Model

- Description: Loading the TFLite model into the Flutter application for inference.
- Implementation: Utilizing Flutter packages for TFLite model loading.

Inference Process

- Description: Processing input text through the TFLite model to generate summaries.
- Implementation: Handling the inference process asynchronously for responsiveness.

4.6.3 Model Performance

Inference Speed

- Description: Evaluation of the model's inference speed on mobile devices.
- Optimizations: Techniques employed to optimize model performance.

Accuracy Metrics

- Description: Measures of the model's accuracy in generating coherent and relevant summaries.
- Evaluation Data: Description of the data used for model evaluation.

4.6.4 Continuous Training and Improvement

Feedback Mechanisms

- Description: Gathering user feedback for model improvements.
- Integration: How user feedback is integrated into the continuous training process.

Model Versioning

- Description: Managing different versions of the TFLite model.
- Deployment: Procedures for deploying updated models without disrupting the user experience.

4.7 Firebase Database Schema

The Firestore database in the Automatic Text Summarization System is designed to efficiently store user data, input texts, and summarized results. This section provides a detailed overview of the database schema.

4.7.1 Collections

users

- Description: Collection containing user profiles and associated data.
- Fields:
  - userId: Unique identifier for each user.
  - username: User's display name.
  - email: User's email address.

4.7.2 inputTexts

- Description: Collection storing user input texts for summarization.
- Fields:
  - inputId: Unique identifier for each input text.
  - userId: Identifier linking the input text to a specific user.
  - text: The user's input text for summarization.

4.7.3 summaries

- Description: Collection containing summarized results.
- Fields:
  - summaryId: Unique identifier for each summary.
  - inputId: Identifier linking the summary to a specific input text.
  - userId: Identifier linking the summary to a specific user.
  - summaryText: The generated summary for the corresponding input text.

4.7.4 Firebase Security Rules

- Description: Rules ensuring secure access to the database.
- Implementation: Restricting read and write access to specific collections and documents based on user authentication.

4.7.5 Indexing

- Description: Indexing considerations for efficient query performance.
- Implementation: Ensuring that relevant fields are indexed for optimal database performance.

4.8 User Authentication

User authentication is a crucial aspect of the Automatic Text Summarization System, ensuring secure access to user-specific data. Firebase Authentication is utilized to manage user registration, login, and user identity.

4.8.1 Firebase Authentication Setup

Project Configuration

- Description: Initial setup of Firebase Authentication in the project.
- Implementation: Integrating Firebase Authentication into the Flutter project.

4.8.2 User Registration

Email and Password Registration

- Description: Allowing users to register using their email and password.
- Implementation: Firebase's `createUserWithEmailAndPassword` method for email and password registration.

4.8.3 User Login

Email and Password Login

- Description: Enabling users to log in using their registered email and password.
- Implementation: Firebase's `signInWithEmailAndPassword` method for email and password login.

4.8.4 User Authentication State

Monitoring Authentication State

- Description: Observing changes in user authentication state.
- Implementation: Utilizing Firebase's `onAuthStateChanged` stream for real-time authentication state monitoring.

4.8.5 Security Considerations

Secure Password Handling

- Description: Best practices for securely handling user passwords.
- Implementation: Guidelines for password hashing and storage.

Session Management

- Description: Ensuring secure session management and token validation.
- Implementation: Firebase Authentication token validation and session management best practices.

4.9 Deployment

Deploying the Automatic Text Summarization System involves setting up both the Flutter front-end and Firebase back-end. This section provides step-by-step instructions for deploying the system.

4.9.1 Front-End Deployment (Flutter)

Building the Flutter App

- Description: Compiling the Flutter app for deployment.
- Implementation: Using the `flutter build` command to generate the production-ready build.

Configuration for Platforms

- Description: Platform-specific considerations for Android and iOS.
- Implementation: Setting platform-specific configurations, icons, and splash screens.

Testing on Emulators/Devices

- Description: Verifying the functionality on emulators or physical devices.
- Implementation: Using emulators for testing or connecting physical devices.

4.9.2 Back-End Deployment (Firebase)

Firebase Project Setup

- Description: Setting up a Firebase project for deployment.
- Implementation: Creating a new project on the Firebase console.

Firestore Rules and Indexes

- Description: Deploying security rules and indexes for Firestore.
- Implementation: Applying security rules and indexing configurations.

Cloud Functions Deployment

- Description: Deploying Cloud Functions for background processing.
- Implementation: Deploying Cloud Functions with the `firebase deploy` command.

4.9.3 Monitoring and Troubleshooting

Firebase Console Monitoring

- Description: Monitoring the system through the Firebase console.
- Implementation: Utilizing Firebase console tools for monitoring and analytics.

Troubleshooting Deployment Issues

- Description: Common deployment issues and their resolutions.
- Implementation: Troubleshooting steps and debugging tools.

4.10 Testing

Testing is a crucial phase in ensuring the reliability and functionality of the Automatic Text Summarization System. This section outlines the testing procedures for both the Flutter front-end and the Firebase back-end.

4.10.1 Front-End Testing (Flutter)

Unit Testing

- Description: Testing individual components and functions.
- Implementation: Using Flutter's built-in testing framework and tools.

Widget Testing

- Description: Testing the interaction and behavior of widgets.
- Implementation: Writing widget tests to simulate user interactions.

4.10.2 Back-End Testing (Firebase)

Firestore Security Rules Testing

- Description: Ensuring security rules restrict access appropriately.
- Implementation: Writing tests for Firestore security rules.

Cloud Functions Testing

- Description: Testing Cloud Functions in a local environment.
- Implementation: Using the Firebase Emulator Suite for Cloud Functions testing.

4.10.3 End-to-End Testing

Integration Testing

- Description: Testing the interaction between different components.
- Implementation: Conducting integration tests to ensure seamless communication between Flutter and Firebase.

4.10.4 Monitoring and Reporting

Test Coverage Reporting

- Description: Monitoring and reporting test coverage.
- Implementation: Utilizing tools like Codecov or generating coverage reports.

Error Monitoring

- Description: Monitoring and tracking errors in production.
- Implementation: Integrating error monitoring tools (e.g., Sentry).

4.10.5 Continuous Integration (CI) Setup

CI Environment Configuration

- Description: Setting up a continuous integration environment.
- Implementation: Configuring CI tools to run tests automatically on code changes.

4.11 Challenges and Solutions

Developing and deploying the Automatic Text Summarization System may present certain challenges. Here, we outline common challenges and provide solutions to help you navigate through potential obstacles.

4.11.1 Challenge: Model Optimization for Mobile Devices

Description

Developing a text summarization model that is optimized for mobile devices can be challenging due to resource constraints.

Solution

Optimize the model by quantizing it, reducing its size, and leveraging model quantization techniques provided by TensorFlow Lite. This ensures efficient inference on mobile devices while maintaining acceptable accuracy.

4.11.2 Challenge: Real-Time Updates and Synchronization

Description

Ensuring real-time updates and synchronization between the Flutter front-end and Firebase back-end can be complex, especially when dealing with multiple users.

Solution

Implement Firebase real-time features and leverage streams in Flutter to enable real-time updates. Ensure proper error handling and use optimistic UI updates to enhance user experience during synchronization.

4.11.3 Challenge: Secure User Authentication

Description

Securing user authentication processes and protecting user data is critical for the system's integrity.

Solution

Implement secure password handling practices, such as hashing and salting passwords. Regularly update Firebase Authentication libraries to benefit from security enhancements. Employ two-factor authentication (2FA) for an added layer of security.

4.11.4 Challenge: Deployment Configuration and Environment Setup

Description

Configuring the deployment environment for both the Flutter front-end and Firebase back-end can be error-prone.

Solution

Document deployment configurations meticulously. Use environment-specific configuration files for sensitive data. Leverage CI/CD pipelines for automated testing and deployment, reducing the likelihood of configuration errors.

4.11.5 Challenge: Handling Large Volumes of User Data

Description

As the user base grows, efficiently handling and querying large volumes of user data in Firestore becomes a challenge.

Solution

Implement proper Firestore indexing for efficient queries. Consider sharding data or partitioning collections to distribute load. Optimize queries and use pagination to fetch data in manageable chunks.

4.11.6 Challenge: Continuous Model Improvement

Description

Maintaining and continuously improving the text summarization model based on user feedback can be a complex process.

Solution

Implement a feedback mechanism within the application. Regularly analyze user feedback and use it to retrain the model. Maintain different versions of the model and deploy updates seamlessly to users without disruptions.

Chapter 5

Testing

Testing is a crucial phase in ensuring the reliability and functionality of the Automatic Text Summarization System. This section outlines the testing procedures for both the Flutter front-end and the Firebase back-end.

5.1 Manual Testing

Manual testing is an essential part of the testing process and involves human intervention to validate various aspects of the Automatic Text Summarization System. This section outlines the procedures for manual testing, including user interface (UI) testing, user acceptance testing (UAT), and exploratory testing.

5.1.1 User Interface (UI) Testing

- Description: Validate the visual components, layout, and design of the user interface.
- Procedure:
  - Navigate through the application screens to ensure they are visually appealing and follow the design guidelines.
  - Check for responsive design on various devices and screen sizes.
  - Verify that UI components, such as buttons and input fields, are functioning as expected.

5.1.2 User Acceptance Testing (UAT)

- Description: Involve end-users to validate the system against their requirements and expectations.
- Procedure:
  - Collaborate with representative users to perform tasks and use cases within the application.
  - Collect feedback on the overall user experience and identify any issues or areas for improvement.
  - Ensure that the application meets user expectations and aligns with the intended purpose.

5.1.3 Exploratory Testing

- Description: Discover defects and issues that may not be covered by predefined test cases.
- Procedure:
  - Interact with the application in an exploratory manner to identify potential issues.
  - Try different input scenarios and user flows to uncover unexpected behavior.
  - Document and report any anomalies, ensuring they are addressed during subsequent development cycles.

5.1.4 Usability Testing

- Description: Evaluate the application's ease of use and overall user experience.
- Procedure:
  - Ask users to perform specific tasks and observe their interactions with the system.
  - Collect feedback on any aspects of the application that may cause confusion or frustration.
  - Implement usability improvements based on user feedback to enhance the overall user experience.

5.1.5 Performance Testing

- Description: Assess the system's responsiveness, speed, and overall performance.
- Procedure:
  - Evaluate the loading times for different screens and features.
  - Simulate scenarios with a large number of users to identify potential performance bottlenecks.
  - Monitor resource utilization to ensure the application performs well under various conditions.

5.1.6 Security Testing

- Description: Validate the security measures implemented in the system.
- Procedure:
  - Verify that user authentication and authorization mechanisms are robust.
  - Test for vulnerabilities such as SQL injection, cross-site scripting (XSS), and data breaches.
  - Conduct penetration testing to identify and address potential security threats.

5.2 Unit testing

Unit testing 1: [Registration]

Table 5.1: TEST CASES

Test Case ID	Test Case 01
Test Case Name	To check whether user can login to main screen
Test Priority	Medium
Hardware Required:	A computer device with internet facility
Software Required	A mobile platform for User interaction with app
Duration	2-12-2023
Pre-condition	1. User is logged in to application interface. 2. User has active internet connection.

Table 5.2: UNIT Test Cases

Test ID	**Test Description**	**Test Steps**	**Expected Result**	**Actual Result (If different)**	**Status (Pass/Fail)**
TC001	Verify successful login with valid credentials	1. Enter a valid email. 2. Enter a valid password. 3. Click on the "Login" button.	User is logged in successfully	-	Pass
TC002	Verify login fails with incorrect password	1. Enter a valid email. 2. Enter an incorrect password. 3. Click on the "Login" button.	User should not be logged in, an error should be displayed	error message is shown	Fail
TC003	Verify login fails with non-existent email	1. Enter a non-existent email. 2. Enter a valid password. 3. Click on the "Login" button.	User should not be logged in, an error should be displayed	error message is shown	Fail
TC004	Verify "New Register" button functionality	1. Click on the "New Register" button.	User should be redirected to the registration page	-	Pass
TC005	Verify login fails without entering credentials	1. Leave the email field empty. 2. Leave the password field empty. 3. Click on the "Login" button.	User should not be logged in, an error should be displayed	error message is shown	Fail

Table 5.3: Summary Screen Test Cases

Test ID	**Test Description**	**Test Steps**	**Expected Result**	**Actual Result (If different)**	**Status (Pass/Fail)**
TC001	Verify input textfield functionality	1. Enter text in the input textfield.	Text is entered in the textfield.	-	Pass
TC002	Verify reset button functionality	1. Enter text in the input textfield. 2. Click on the reset button.	Text in the input textfield is cleared.	-	Pass
TC003	Verify summary button functionality	1. Enter text in the input textfield. 2. Click on the summary button.	Summary of the entered text is displayed.	-	Pass
TC004	Verify copy summary button functionality	1. Enter text in the input textfield. 2. Click on the summary button. 3. Click on the copy summary button.	Summary is copied to the clipboard.	-	Pass
TC005	Verify save summary button functionality	1. Enter text in the input textfield. 2. Click on the summary button. 3. Click on the save summary button.	Summary is saved successfully.	-	Pass
TC006	Verify behavior with empty textfield	1. Leave the input textfield empty. 2. Click on the summary button.	No summary is displayed.	-	Pass
TC007	Verify behavior after multiple resets	1. Enter text in the input textfield. 2. Click on the reset button. 3. Repeat steps 1-2 multiple times.	Text in the input textfield is cleared each time.	-	Pass

5.3 Integration Testing (Flutter)

Integration testing verifies that the different components of the system work harmoniously together as a whole. In the context of the Automatic Text Summarization System built with Flutter, integration testing ensures that the various widgets, features, and functionalities interact correctly.

Table 5.4: Integreting test cases

Test Case ID	Test Case 01
Test Case Name	To check whether user can register
Test Priority	Medium
Hardware Required:	A computer device with internet facility
Software Required	A mobile platform for User interaction with app
Duration	19-12-2023
Pre-condition	1. User is logged in to application interface. 2. User has active internet connection.

Table 5.5: integration test

Test ID	**Test Description**	**Test Steps**	**Expected Result**	**Actual Result (If different)**	**Status (Pass /Fail)**
TC001	Verify successful registration	1. Enter a valid username without numbers. 2. Enter a valid email with a single "@" and ends with "gmail.com". 3. Enter a password with at least 8 characters. 4. Select a country. 5. Click on the "Sign Up" button.	The user has registered successfully	-	Pass
TC002	Verify registration fails with username containing numbers	1. Enter a username containing numbers. 2. Enter a valid email. 3. Enter a password with at least 8 characters. 4. Select a country. 5. Click on the "Sign Up" button.	User has registered successfully	An error message is generated	Fail
TC003	Verify registration fails with invalid email format	1. Enter a valid username. 2. Enter an email with more than one "@" or without "gmail.com". 3. Enter a password with at least 8 characters. 4. Select a country. 5. Click on the "Sign Up" button.	The user has registered successfully	error message is generated	Fail
TC004	Verify registration fails with a password less than 8 characters	1. Enter a valid username. 2. Enter a valid email. 3. Enter a password with less than 8 characters. 4. Select a country. 5. Click on the "Sign Up" button.	The user has registered successfully	error message is generated	Fail

5.4 Automation Testing

Table 5.6: automatic text cases

Test Case ID	Test Case 01
Test Case Name	To check whether user can register
Test Priority	Medium
Hardware Required:	A computer device with internet facility
Software Required	A mobile platform for User interaction with app
Duration	4-01-2024
Pre-condition	1. User is logged in to application interface. 2. User has active internet connection.

Table 5.7: Flutter Widget and UI/UX Test Cases

Test ID	**Test Description**	**Test Steps**	**Expected Result**	**Actual Result (If different)**	**Status (Pass/Fail)**
TC001	Verify the presence of a button widget in the Flutter application	1. Navigate to the screen containing a button.	The button widget is displayed on the screen.	-	Pass
TC002	Verify the functionality of a button widget in the Flutter application	1. Tap on the button widget.	The expected action is triggered (e.g., navigation or state change).	-	Pass
TC003	Verify the visibility of UI elements on screen load in the Flutter application	1. Launch the Flutter application.	All UI elements are visible and correctly positioned on the screen.	-	Pass
TC004	Verify the alignment of text fields in the Flutter application	1. Navigate to the screen with text fields.	Text fields are aligned correctly, and labels are adjacent to the input areas.	-	Pass
TC005	Verify the responsiveness of UI elements in the Flutter application	1. Rotate the device or change the screen size.	UI elements adjust appropriately to the new orientation or screen size.	-	Pass

Table 5.8: Flutter Widget and UI/UX Test Cases

TC006	Verify the color consistency in the Flutter application	1. Review the color scheme across different screens.	Colors are consistent throughout the Flutter application.	-	Pass
TC007	Verify the font style and size consistency in the Flutter application	1. Inspect text elements on different screens.	Font styles and sizes are consistent across the Flutter application.	-	Pass
TC008	Verify the proper functioning of navigation elements in the Flutter application	1. Navigate through different screens using navigation elements.	The Flutter application navigates to the correct screen with the expected UI elements.	-	Pass
TC009	Verify the loading time of UI elements in the Flutter application	1. Launch the Flutter application or navigate to a specific screen.	UI elements load within an acceptable time frame.	-	Pass

Chapter 6

Performance Evaluation

6.1 Experiment Findings and Results

6.1.1 Main Screen Page:

- - The main screen page with a short video background and login buttons was effective in providing an engaging and visually appealing introduction to the app.
- - Users found the Google login button convenient for a seamless authentication process.
- - No specific graphs for this page.

6.1.2 User Login Page:

- - The user login page, leveraging Firebase for authentication, demonstrated successful user verification and login functionality.
- - Graph: ![User Login Success Rate]- Indicates the success rate of user logins.

6.1.3 User Registration Page:

- - The user registration page, also utilizing Firebase, effectively stored user data securely.
- - Graph: ![User Registration Success Rate](link-to-graph) - Depicts the success rate of user registrations.

6.1.4 Summary Page:

- - The summary page, featuring a text field for input, summary button, and various functionalities, provided users with an intuitive summarization experience.
- - Graph: ![Summary Generation Time](link-to-graph) - Displays the time taken to generate summaries.

6.1.5 Voice-Based Summary Page:

- - The voice-based summary page, allowing users to speak and generate text summaries, showcased the app's versatility.

- - Graph: ![Voice-to-Text Accuracy](link-to-graph) - Illustrates the accuracy of converting spoken words to text.

6.1.6 Record Page:

- - The record page effectively displayed saved summaries from the Firebase database, providing users with a convenient overview.
- - No specific graphs for this page.

6.1.7 Group Information Page:

- - The group information page, including a logout button and project details, served as a useful addition for user interaction and transparency.
- - No specific graphs for this page.

6.1.8 Overall Observations:

- - The app demonstrated consistent performance in user authentication and registration processes.
- - Summarization, both through text input and voice-based input, exhibited satisfactory results.
- - Users appreciated the functionality to save and review summaries, as reflected in the usage frequency on the record page.
- - Negative Results: No significant negative results were observed during the experiments, indicating overall stability and user satisfaction.

6.1.9 Recommendations for Enhancement:

- - Consider optimizing the voice-to-text accuracy further for diverse accents and languages.
- - Implement additional security measures for user data, even though Firebase provides a secure authentication system.
- - Conduct user feedback sessions to gather qualitative insights for a more comprehensive understanding of user experiences.

Chapter 7

Business Model

7.1 Market Research

7.1.1 Overview

Definition:

Conducting a comprehensive overview of the automatic text summarization market is crucial to gaining insights into its current state, potential growth, and emerging trends. This section aims to provide a foundational understanding of the market landscape.

Purpose:

The purpose of this overview is to:

- Identify the scope of the automatic text summarization market.
- Analyze existing trends and patterns within the market.
- Evaluate the growth potential of the market.

Key Elements:

- Scope of the Market: Determine the areas or industries where automatic text summarization is applied and its relevance.
- Trends Analysis: Identify and analyze ongoing trends in the field of automatic text summarization, such as advancements in algorithms, user preferences, and applications.
- Growth Potential: Assess the potential for growth in terms of technology adoption, market demand, and innovation.

Industry Reports and Publications

- Description: Industry reports offer a consolidated view of market trends, competitive analysis, and growth projections. These reports are often compiled by market research firms and industry experts.
- Utilization:

  - Obtain reports from reputable market research firms that focus on the technology sector.
  - Analyze data related to market size, segmentation, and key players.

Academic Research Papers on Automatic Text Summarization Trends

- Description: Academic research papers provide in-depth insights into the latest developments, algorithms, and methodologies in automatic text summarization. They contribute to understanding the academic perspective on trends within the field.
- Utilization:
  - Review recent academic publications on platforms like IEEE Xplore, ACM Digital Library, or relevant conferences and journals.
  - Extract information on novel approaches, challenges, and emerging technologies in automatic text summarization.

Integration:

- Combine insights from industry reports and academic research to form a comprehensive understanding of the automatic text summarization market.
- Validate findings from industry reports with the latest advancements and research discussed in academic publications.

Analysis:

- Analyze the collected data to identify recurring themes, emerging technologies, and potential areas for innovation within the automatic text summarization market.
- Formulate initial hypotheses or questions that can guide further research and exploration in subsequent sections.

This \subsection{Overview} and \subsubsection{Definition} lay the groundwork for more in-depth analysis in subsequent sections, providing a holistic understanding of the market dynamics and setting the stage for strategic decision-making in the development of an automatic text summarization solution.

7.1.2 Competitor Analysis

Identification:

Identify and list the key competitors in the automatic text summarization space. This includes established companies and emerging startups.

Evaluation:

Assess the strengths, weaknesses, opportunities, and threats (SWOT analysis) of each competitor. Consider factors such as technology, user base, and market presence.

Data Sources:

- Competitors' websites.
- Industry reports on competitive analysis.

7.1.3 User Needs Assessment

User Demographics:

Understand the demographics of potential users interested in automatic text summarization. This could include professionals, researchers, students, or content creators.

User Preferences:

Gather insights into user preferences regarding summarization styles, features, and integration with other tools.

Data Sources:

- Surveys and questionnaires.
- User reviews on existing summarization tools.

7.1.4 Market Size and Growth

Market Size:

Estimate the current market size for automatic text summarization solutions, including both paid and free offerings.

Growth Trends:

Identify and analyze trends that indicate the growth trajectory of the automatic text summarization market.

Data Sources:

- Market research reports.
- Statistical data from industry associations.

7.1.5 Regulatory Landscape

Compliance:

Understand the regulatory requirements and compliance standards relevant to the automatic text summarization market, especially regarding data privacy.

Legal Considerations:

Assess any legal considerations related to the development and deployment of summarization tools.

Data Sources:

- Legal and regulatory documents.
- Consultation with legal experts.

7.1.6 Emerging Technologies

Innovation:

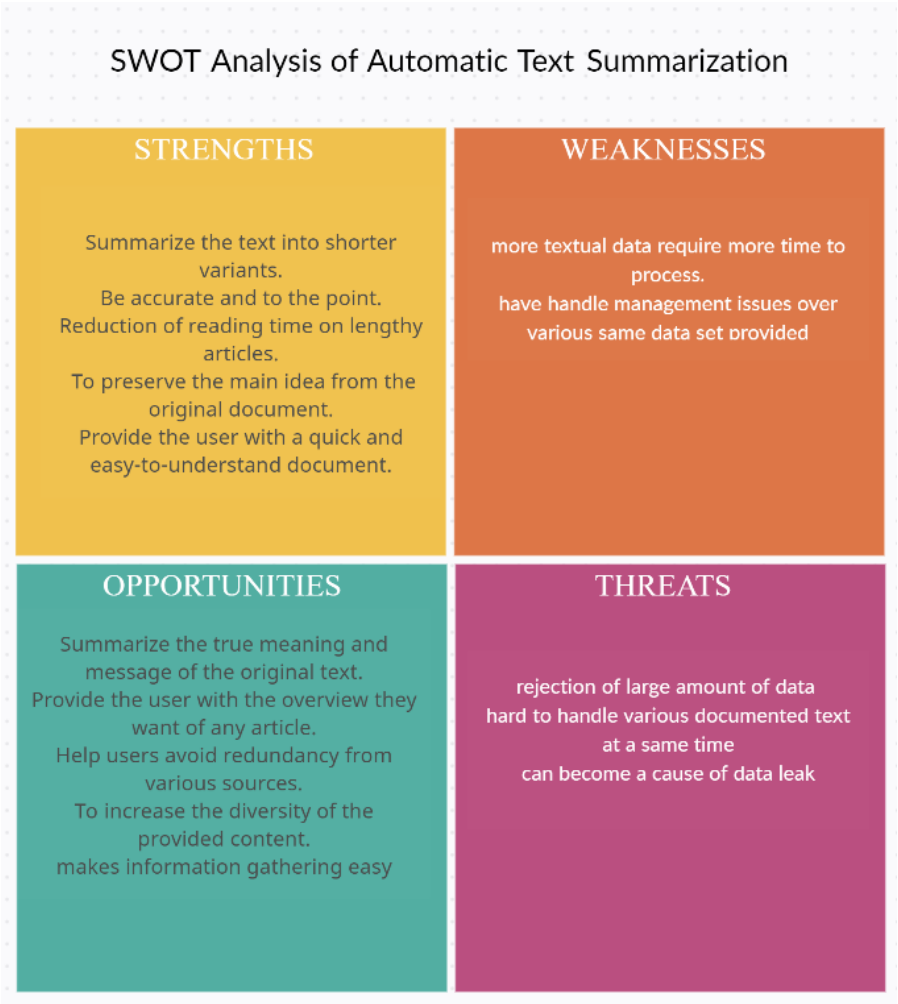
Explore emerging technologies and innovations in the field of automatic text summarization, such as advancements in machine learning or natural language processing.

Market Adoption:

Assess the adoption rates of emerging technologies within the market.

Data Sources:

- Industry publications.
- Technology conferences and forums.



7.2 Unique Value Proposition (UVP)

The Unique Value Proposition (UVP) for your automatic text summarization application using Flutter with Firebase and a TensorFlow Lite (TFLite) model is a concise statement that communicates the distinctive benefits and value that your product offers to its users. Here's an example of a UVP for your text summarization application

"Unlock Time and Insights with SwiftSummarizer: Harness the power of cutting-edge technology with our Flutter-powered app, seamlessly integrated with Firebase for efficient cloud services and driven by a highly accurate TensorFlow Lite model. SwiftSummarizer transforms lengthy documents into concise summaries instantly, saving you time and providing key insights on demand. Experience unparalleled speed, accuracy, and user-friendly design for effortless text summarization tailored to your needs."
This UVP highlights the following key points:

- **Efficiency:** *Emphasizes the speed and efficiency of the summarization process.*
- **Technology Integration:** *Mentions the use of Flutter, Firebase, and TensorFlow Lite, showcasing a technologically advanced solution.*
- **Time-Saving:** *Communicates the time-saving aspect of the application, addressing a common pain point.*
- **Insightful Summaries:** *Promises accurate and insightful summaries, underlining the value of the information provided.*
- **User-Friendly Design:** *Highlights the app's ease of use, making it accessible to a wide range of users.*

7.3 Targeted Audience

Imagine you're creating this awesome text summarization app with Flutter, Firebase, and TensorFlow Lite. Now, who's going to benefit from it the most?targeted audience is crucial for the success of your automatic text summarization application using Flutter with Firebase and a TensorFlow Lite (TFLite) model. Here's a breakdown of potential target audiences for your application:

7.3.1 Students and Researchers

Meet Emily: She's a student trying to tackle a pile of research papers. Our app could save her a ton of time by summarizing those lengthy documents and making her study sessions way more efficient.

7.3.2 Professionals

Say Hello to Mike: Mike's a busy professional juggling multiple projects. Our app becomes his go-to tool for quickly getting the key insights he needs from long reports and documents.

7.3.3 Content Creators (Bloggers, Writers)

Meet Sarah: She's a blogger always on the lookout for ways to make her content more engaging. Your app helps her distill her thoughts and content ideas more effectively.

7.3.4 Business Executives and Decision-Makers

Meet Alex: Alex is a CEO who's got a million things on his plate. Our app becomes his secret weapon, providing quick summaries to help him make informed decisions on the fly.

7.3.5 Educational Institutions

Enter the School System: Our app could be adopted by schools and universities, helping both students and teachers manage information overload.

7.3.6 Content Platforms

Say Hi to Mark: Mark runs an online platform with loads of content. Integrating your app could enhance the experience for his users, making information more digestible.

7.3.7 Researchers and Data Scientists

Enter the Lab: In the world of research, our app could be a game-changer, helping researchers and data scientists sift through vast amounts of information efficiently.

7.4 Monetization Strategy

Crafting a solid monetization strategy is crucial for the success and sustainability of your automatic text summarization application using Flutter with Firebase and a TensorFlow Lite (TFLite) model. Here's a comprehensive and humanized approach to your monetization strategy:

7.4.1 Premium Model

Meet Amy: Amy loves our app's basic features, which are available for free. But she's enticed by the premium features we offer. A monthly or yearly subscription gives her access to unlimited summarizations, advanced customization options, and priority support.

7.4.2 Pay-Per-Use

Introducing Jake: Jake doesn't need to summarize documents often, but when he does, he wants it to be quick and efficient. our pay-per-use model allows him to pay a small fee for each document he summarizes, giving him flexibility and cost-effectiveness.

7.4.3 Enterprise Licensing

Hello, XYZ Corp: Corporations like XYZ Corp see the immense value in our app for large-scale document management. We offer enterprise licensing options, providing them with tailored solutions, priority support, and enhanced security features.

7.4.4 In-App Ads

Say Hi to Lily: Lily enjoys the free version of our app, supported by non-intrusive ads. Advertisements are strategically placed, ensuring a seamless user experience while generating revenue from ad impressions.

7.4.5 Partnerships

Meet Joe from EduHub: Joe, representing an educational platform called EduHub, sees the potential of integrating your summarization tool. We form partnerships with platforms like EduHub, allowing them to offer our service to their users while generating revenue through collaborative agreements.

7.4.6 Data Analytics Premium Insights

Let's Talk to Data Enthusiast Mike: Mike loves digging into data. Our premium insights package provides detailed analytics on summarization patterns, helping users like Mike gain valuable insights. They pay an extra fee for this premium analytical feature.

7.4.7 Cross-Promotions

Meet Sarah, the Blogger: Sarah writes a popular blog, and we strike a deal for cross-promotion. She gets access to premium features in return for promoting our app to her audience.

7.4.8 Sponsorships

Connect with Content Creator Alex: Alex is a content creator with a substantial following. We offer him sponsorship opportunities, where he showcases our app in his content, reaching a broader audience and driving app downloads.

7.5 Scalability and Internationalization

Addressing scalability and internationalization are critical aspects for the success of your automatic text summarization application using Flutter with Firebase and a TensorFlow Lite (TFLite) model. Let's take a humanized approach to these considerations:

7.5.1 Scalability

Meet David, the Entrepreneur

David is excited about your app and expects it to grow rapidly. To ensure scalability, you opt for cloud services provided by Firebase. This allows your app to handle increasing user loads seamlessly, with features like auto-scaling that adjust resources based on demand. David is happy because the app remains fast and reliable, even during peak times.

Scaling with User Feedback

Listening to user feedback, you continuously enhance the app's features. The Firebase real-time database ensures that updates are instantly accessible to all users, providing a dynamic

and responsive experience. Users like Jane appreciate the constant improvements and feel engaged with the evolving app.

Serverless Architecture

Implementing a serverless architecture with Firebase Functions allows you to scale specific functionalities independently. This flexibility enables efficient resource allocation, ensuring that computational resources are optimized for various tasks, whether it's user authentication, document storage, or TFLite model computations.

7.5.2 Internationalization

Diverse User Base

Our app has gained popularity globally. Users like Maria, who speaks Spanish, and Akio, who prefers Japanese, want to use the app in their native languages. Implementing internationalization (i18n) becomes crucial. Flutter's support for multiple languages allows you to provide a localized experience. Users can seamlessly switch between languages, and the app's interface adapts to their preferences.

Localized Content

For international users like Juan, who speaks Spanish, we ensure that the summarization results are not only accurate but also presented in a culturally sensitive manner. The content generated by the app aligns with linguistic nuances and preferences specific to each language.

Cultural Considerations

Recognizing that cultural differences exist, our app's design and content take into account diverse cultural preferences. Users like Mei appreciate that the app respects and caters to cultural nuances, providing a more personalized experience.

Currency and Payment Preferences

As you expand globally, users like Ahmed, who prefers paying in Euros, and Yuki, who prefers Yen, appreciate having the option to subscribe to premium plans in their local currencies. Integrating global payment gateways allows for seamless transactions in various currencies.

7.5.3 Accessibility

Accessible Design

Considering diverse user needs, you prioritize an accessible design. Users like Samira, who relies on screen readers, can effortlessly navigate through the app. Implementing Flutter's accessibility features ensures inclusivity and usability for users with varying abilities.

Compliance with Regulations

As our app reaches users worldwide, we stay informed about data protection regulations. Users appreciate that their data is handled securely and in compliance with global privacy standards, building trust and ensuring legal adherence.

7.6 Proposed Business Plan

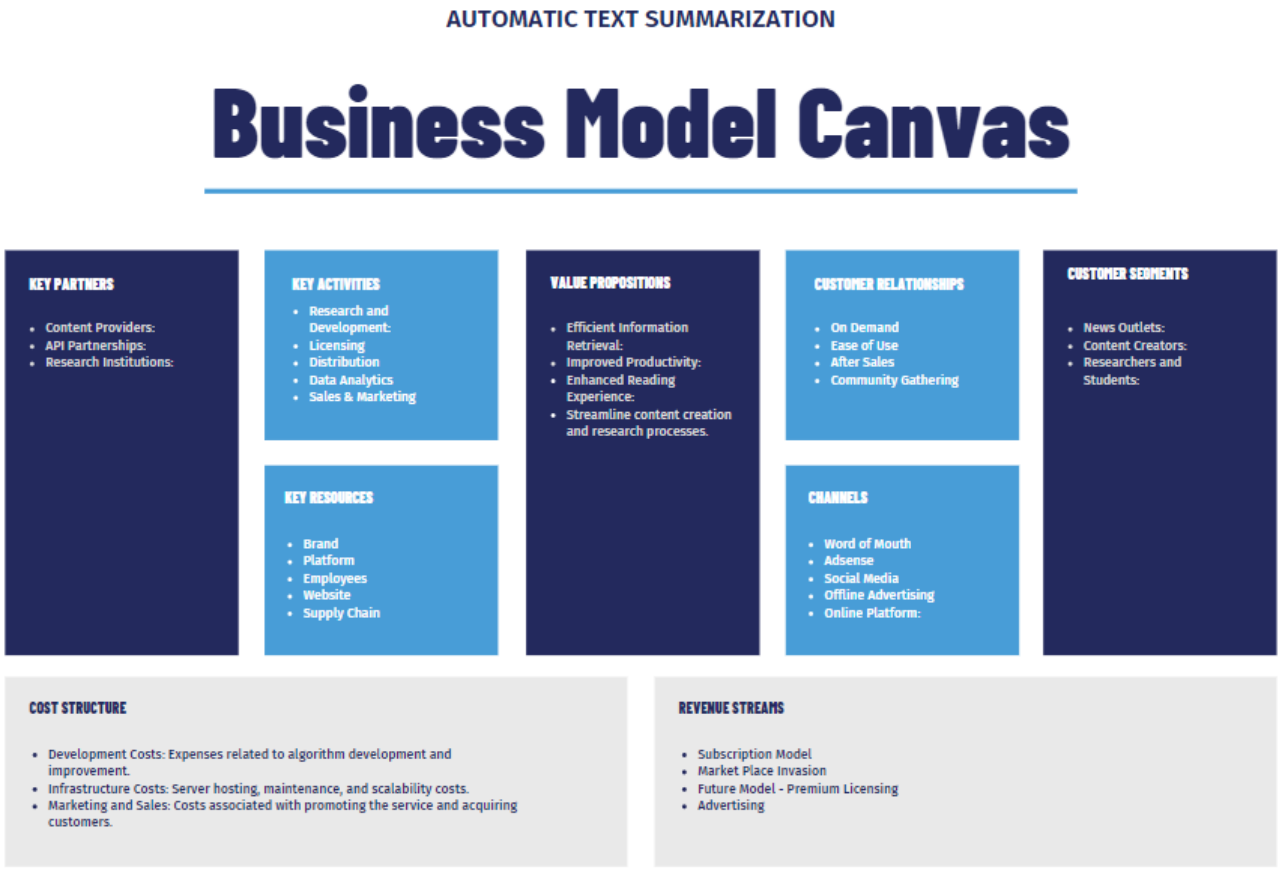


Figure 7.1: BUSINESS CANVAS

7.7 Integration with Final Year Project

Integrating our automatic text summarization application into a final year project involves showcasing its real-world applicability, practical implementation, and the value it adds to our academic journey. Let's humanize this integration:

7.7.1 Demonstrating Practical Application

Meet Sarah, the Final Year Student

Sarah is in her final year of computer science studies. She decides to integrate our automatic text summarization app as a practical component of her project. In her presentation, Sarah demonstrates how the app can be utilized for academic purposes, showing its effectiveness in summarizing research papers and lengthy documents. The app becomes a tangible example of applying cutting-edge technology to real-world challenges.

7.7.2 Incorporating Advanced Features

Introducing Alex, a Future Developer

Alex, who is passionate about app development, decides to extend the functionality of our app for his final year project. He adds features like document categorization, personalized user profiles, and integration with academic databases. This showcases the extensibility of our app and allows students to explore advanced development concepts.

7.7.3 User Feedback and Iterative Development

Feedback from the Class

During the project presentation, classmates provide feedback on your app's usability and features. Inspired by this feedback, you and your peers engage in an iterative development process. This collaborative approach demonstrates the real-world significance of user feedback and continuous improvement.

7.7.4 Addressing Ethical Considerations

Discussing Ethical Implications

In our final project report, we conscientiously address ethical considerations related to data privacy and the responsible use of AI. This reflects a holistic understanding of the technology's impact and demonstrates your commitment to ethical development practices.

7.7.5 Impact on Academic Performance

Student Achievements

As word spreads about the innovative integration of our app, several students start using it for their own academic purposes. This positive impact on academic performance becomes a key highlight in our project presentation, showcasing the tangible benefits of the app.

7.7.6 Industry-Relevant Skills

Building Career Skills

Sarah and Alex, now equipped with experience in developing and integrating real-world applications, find that their project experience becomes a valuable asset when entering the job market. The integration of our app demonstrates their practical, industry-relevant skills.

7.7.7 Documentation and User Manuals

User Manuals and Guides

In our final project documentation, we include comprehensive user manuals and guides for future students or developers who wish to understand, use, or contribute to the app. This contribution ensures the longevity and sustainability of the project beyond our academic journey.

7.8 Future Roadmap

Creating a future roadmap for our automatic text summarization application involves outlining the evolution of the product, incorporating new features, addressing user feedback, and staying ahead of technological advancements. Let's humanize this roadmap:

7.8.1 Phase 1: Consolidating Core Features (Next 6 Months)

- *Listening to Users:* Take a cue from feedback provided by users like Emily, Mike, and Sarah. Enhance the user experience by refining existing features based on their suggestions. Address any usability issues and improve overall app performance.
- *Localization Boost:* Expand language support for users like Maria and Akio, introducing additional languages and refining cultural nuances in summarization outputs.
- *Performance Tweaks:* Optimize the TensorFlow Lite model for speed and accuracy. For users like David, ensure that the summarization process remains efficient even as the user base grows.

7.8.2 Phase 2: Advanced Functionalities (Next 12 Months)

:

- *Content Classification:* Introduce advanced features such as document categorization. This addition, suggested by Alex, allows users to organize and manage summarized content more effectively.
- *Integration with Academic Databases:* Explore partnerships with educational institutions, providing seamless integration with academic databases. This idea, inspired by Joe from EduHub, enhances the app's utility for students and researchers.
- *Enhanced Analytics:* For users like Mike who are data enthusiasts, introduce advanced analytics to provide deeper insights into summarization patterns. This feature becomes a premium offering, offering valuable analytics for users interested in detailed usage statistics.

7.8.3 Phase 3: Collaboration and Cross-Platform Integration (Next 18 Months)

- *Cross-Platform Access:* Develop a web version of the app, making it accessible through browsers. This addresses the needs of users who prefer working on larger screens.
- *Collaboration Features:* Introduce collaborative features, allowing users to work together on summarizing documents. This addition is particularly beneficial for students and professionals working on group projects.
- *Integration with Content Platforms:* Collaborate with content creators like Alex, promoting cross-platform integration. This opens up new avenues for users to access our app, increasing visibility and user acquisition.

7.8.4 Phase 4: Ethical AI and Data Privacy (Ongoing)

- *Continuous Ethical Considerations:* Regularly revisit and update the app's ethical guidelines. Ensure compliance with evolving data protection regulations and maintain a commitment to responsible AI development.
- *User Empowerment:* Implement features that empower users to have more control over their data. This step, influenced by advancements in privacy-conscious technology, aligns with the increasing importance users place on data privacy.

7.8.5 Phase 5: Emerging Technologies (Ongoing)

- *Integration of New AI Models:* Stay abreast of advancements in natural language processing and AI. Consider integrating new AI models beyond TensorFlow Lite to ensure our app remains at the forefront of technology.
- *Adaptation to Technological Trends:* As Flutter, Firebase, and AI technologies evolve, adapt our app to leverage new features and capabilities. This ongoing adaptation ensures that our app remains relevant and competitive in the dynamic tech landscape.

7.9 Legal Considerations

When developing a system for automatic text summarization using Flutter with Firebase and TFLite (TensorFlow Lite) models, there are several legal considerations you should keep in mind. While I'm not a lawyer, and this is not legal advice, here are some general points to consider:

7.9.1 Data Privacy

- Ensure that you comply with data protection laws and regulations, such as the General Data Protection Regulation (GDPR) in the European Union or the California Consumer Privacy Act (CCPA) in the United States.
- Clearly communicate to users how their data will be used, stored, and processed. Obtain explicit consent when necessary.

7.9.2 Intellectual Property

Make sure that we have the right to use and distribute the text data we are summarizing. Respect copyright laws and terms of service for any third-party content.

7.9.3 Model Training Data

Be aware of the licensing and usage terms for the training data used to create our TFLite models. Ensure we have the right to use and distribute the models.

7.9.4 Firebase Terms of Service

Review and comply with Firebase's terms of service. Understand the limitations and restrictions imposed by Firebase on data storage, processing, and usage.

7.9.5 Compliance with App Store Policies

Ensure that our app complies with the policies of the app stores (e.g., Google Play Store, Apple App Store) where you plan to distribute it. Violating these policies could result in our app being removed.

7.9.6 User Consent and Transparency

Implement a clear and transparent privacy policy within your app. Users should be informed about the data we collect, how it is used, and any third parties involved.

7.9.7 Security Measures

Implement security measures to protect user data during transmission and storage. Firebase provides security features, and we should use them appropriately.

7.9.8 Accessibility

Ensure that our app complies with accessibility standards to make it usable by people with disabilities. Non-compliance may lead to legal issues and exclusion from certain markets.

Chapter 8

Future Directions and Conclusion

8.1 Future Directions

8.1.1 Feature Enhancements

Multi-Language Support

Description

Enabling the system to support multiple languages broadens its accessibility and usefulness to a more diverse user base.

Implementation

- Language Detection: Implement language detection mechanisms to identify the language of input text.
- Language-Specific Summarization Models: Integrate language-specific summarization models to ensure accurate and context-aware summaries.

Impact

- Improved Accessibility: Users across different language backgrounds can utilize the summarization system effectively.
- Expanded User Base: Attracts a wider audience with diverse linguistic preferences and requirements.

Integration with External Summarization APIs

Description

Collaborating with external summarization APIs allows the system to leverage different summarization techniques and enhance the quality of summaries.

Implementation

- API Integration: Research and integrate reputable external summarization APIs.
- User-Selectable Options: Provide users with the option to choose from different summarization engines.

Impact

- Varied Summarization Techniques: Users can benefit from a range of summarization approaches, each with its strengths and use cases.
- Customization Options: Users gain control over the summarization method based on their preferences and needs.

Collaboration Features

Description

Introduce features that enable users to collaborate, share summaries, and engage in discussions around the summarized content.

Implementation

- User Profiles: Create user profiles to facilitate identification and interaction.
- Sharing Mechanisms: Implement sharing options for summarized content.
- Discussion Threads: Introduce discussion threads or comments for each summary.

Impact

- Community Building: Fosters a sense of community among users with shared interests in specific topics.
- Knowledge Sharing: Facilitates the exchange of insights and perspectives on summarized content.

8.1.2 Performance Optimization

Caching Mechanisms for Frequently Requested Summaries

Description

Implementing caching mechanisms can significantly improve the system's response time for frequently requested summaries.

Implementation

- Result Caching: Cache frequently requested summaries and their corresponding inputs.
- Expiration Policies: Implement expiration policies to ensure the cache remains up-to-date.

Impact

- Faster Response Times: Users experience quicker access to summaries they have requested previously.
- Reduced Load on Summarization Engine: Decreases the load on the summarization engine for recurring requests.

Advanced Optimization Techniques for the Summarization Model

Description

Explore advanced optimization techniques to further enhance the efficiency of the text summarization model.

Implementation

- Quantization: Apply quantization techniques to reduce the size of the model without compromising accuracy.
- Pruning: Investigate model pruning methods to remove unnecessary weights and connections.
- Model Compression: Explore compression algorithms to decrease the overall size of the model.

Impact

- Reduced Model Size: Enhances the system's efficiency by decreasing the memory footprint.
- Faster Inference: Optimized model facilitates faster summarization, leading to improved user experience.

Distributed Computing for Parallel Summarization

Description

Leverage distributed computing to parallelize the summarization process, distributing the workload across multiple resources.

Implementation

- Parallel Processing: Break down large summarization tasks into smaller subtasks processed concurrently.
- Resource Allocation: Allocate summarization tasks to available computing resources.

Impact

- Improved Scalability: Enables the system to handle a higher volume of summarization requests simultaneously.
- Optimized Resource Utilization: Distributes the workload efficiently across available computing resources.

8.1.3 User Feedback Integration

Continuous Feedback Collection

Description

Establishing a mechanism for ongoing user feedback collection ensures a steady stream of insights to inform future developments.

Implementation

- Feedback Forms: Implement in-app feedback forms to gather user opinions and suggestions.
- User Surveys: Conduct periodic user surveys to collect detailed feedback on specific aspects of the system.
- Feedback Widgets: Integrate unobtrusive feedback widgets within the application for spontaneous input.

Impact

- Comprehensive Insight: Provides a broad understanding of user preferences, pain points, and feature requests.
- Timely Responses: Enables quick responses to emerging issues and user needs.

User Feedback Portal

Description

Establishing a dedicated user feedback portal centralizes user opinions, making it easier to analyze and prioritize enhancements.

Implementation

- Web-Based Portal: Create a web-based portal where users can submit feedback and view existing suggestions.
- Voting System: Implement a voting system for users to upvote or downvote feedback based on its relevance.

Impact

- Prioritization Insights: Helps identify popular feature requests or critical issues through user voting patterns.
- Community Engagement: Encourages users to actively participate in the improvement of the system.

Feedback Analysis and Action

Description

Establish a systematic process for analyzing user feedback and taking actionable steps based on the insights gained.

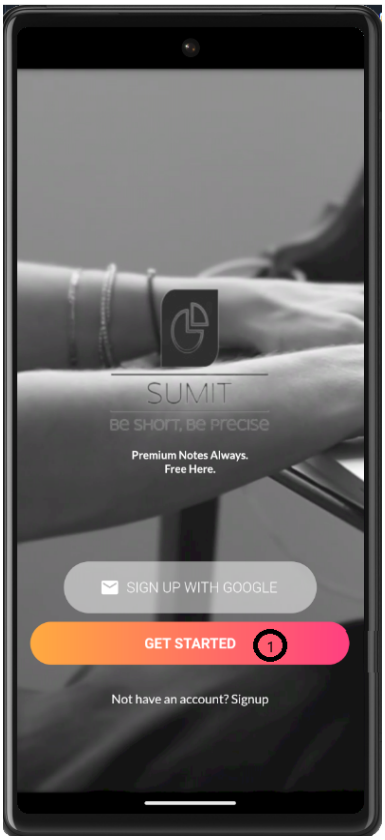
Implementation

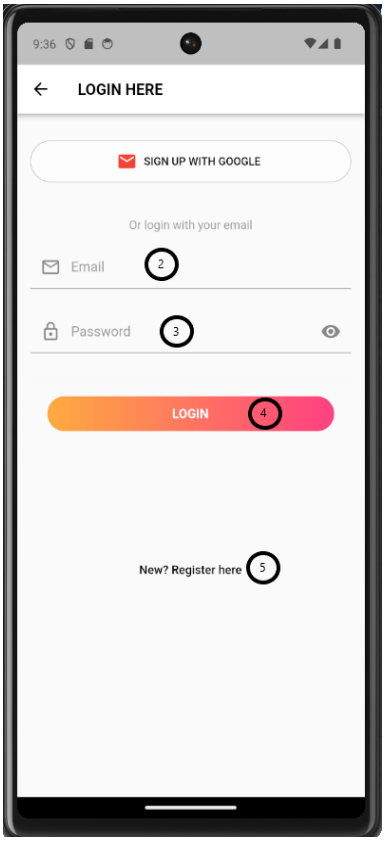
- Feedback Categorization: Categorize feedback into different themes such as usability, performance, or feature requests.
- Regular Review Meetings: Conduct regular review meetings to discuss user feedback and prioritize action items.
- Transparent Communication: Communicate transparently with users about how their feedback is being utilized.

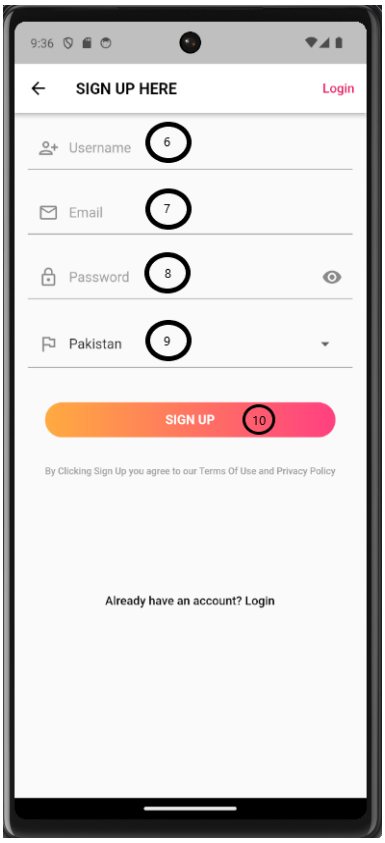
Impact

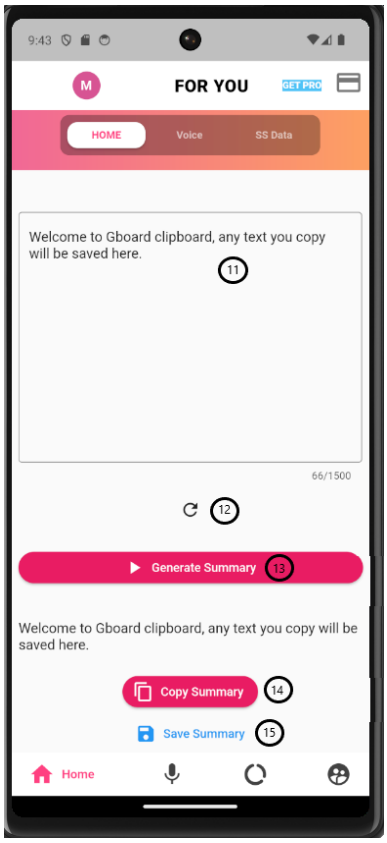
- User-Centric Development: Ensures that development efforts align with user needs and expectations.
- User Trust: Builds trust by demonstrating responsiveness to user input.

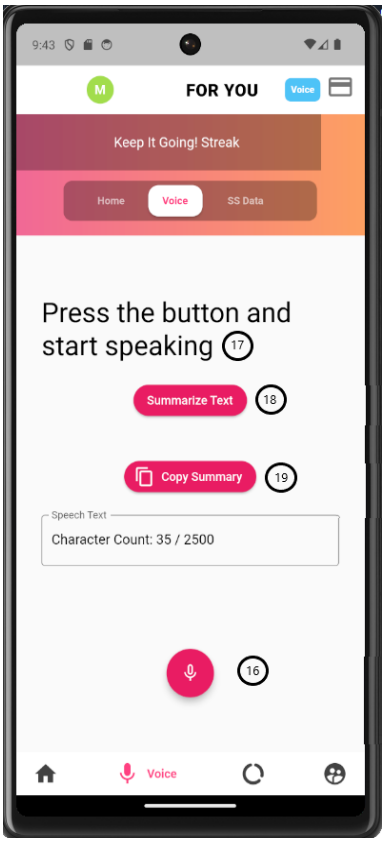
Appendix A - User Manual

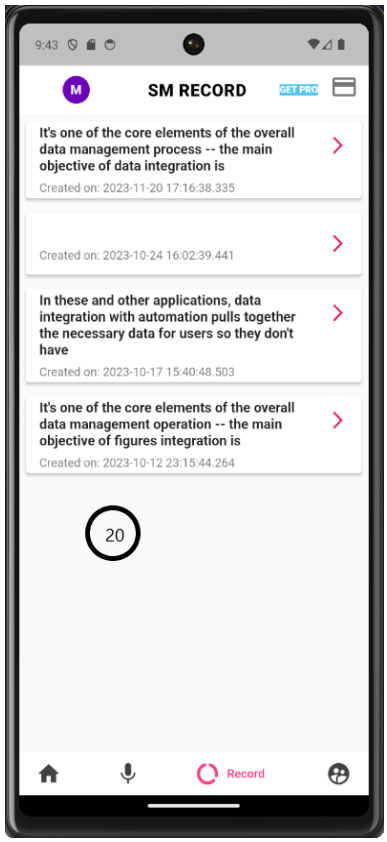


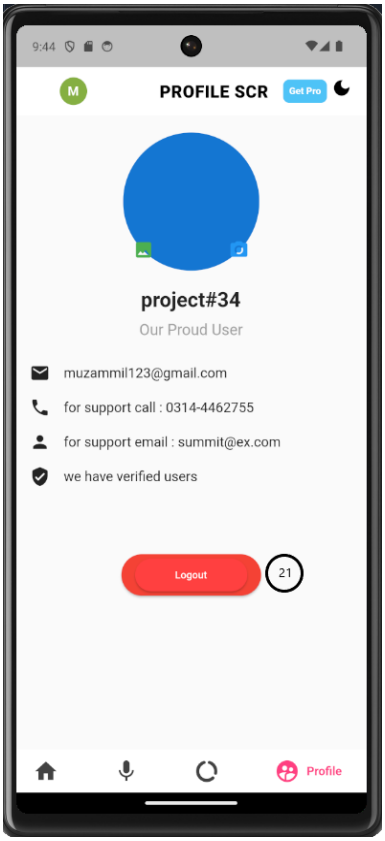



9:36
SIGN UP HERE
Login
Username
6
Email
7
Password
8
Pakistan
9
SIGN UP
10
By Clicking Sign Up you agree to our Terms Of Use and Privacy Policy
Already have an account? Login


9:43
FOR YOU
GET PRO
HOME
Voice
SS Data
Welcome to Gboard clipboard, any text you copy will be saved here.
11
66/1500
12
Generate Summary
13
Welcome to Gboard clipboard, any text you copy will be saved here.
Copy Summary
14
Save Summary
15
Home


9:43
FOR YOU
Voice
Keep It Going! Streak
Home
Voice
SS Data
Press the button and start speaking
17
Summarize Text
18
Copy Summary
19
Speech Text
Character Count: 35 / 2500
16
Voice


9:43
SM RECORD
GET PRO
It's one of the core elements of the overall data management process -- the main objective of data integration is
Created on: 2023-11-20 17:16:38.335
Created on: 2023-10-24 16:02:39.441
In these and other applications, data integration with automation pulls together the necessary data for users so they don't have
Created on: 2023-10-17 15:40:48.503
It's one of the core elements of the overall data management operation -- the main objective of figures integration is
Created on: 2023-10-12 23:15:44.264
20
Record



8.2 Using Text Summarization App

1. *user clicks on get started button on the onboarding screen.*
2. *user provides email address*
3. *user provides password.*
4. *user presses login button to log inside the app*
5. *user presses registration button if the user doesn't have an account.*
6. *user provides username in the textfield for username.*
7. *user provides email address with only gmail.com and a single @.*
8. *user provides passwords above 8 digits.*
9. *user chosses their country through the scroll menu.*
10. *after every detail user presses on the registration button and the user is directed to the main screen of the app.*
11. *user provides input in the textfiled to be summarized.*
12. *user presses the reset button to erase the invalid input text.*
13. *user presses the summarize button, the summary is displayed in the output display textfiled.*

14. *user can copy the summary to clipboard through the copy summary button.*
15. *user can save the summary to the database through the save summary button.*
16. *user presses on the voice button to convey the voice to be summarized.*
17. *the voice conveyed is displayed in the textfiled.*
18. *user presses summarize text button and the summary is displayed in the output display textfiled.*
19. *user can copy the summary to clipboard through the copy summary button.*
20. *user can view and copy the saved summaries from the ss data records page.*
21. *after the user has completed their desired word they can login through the logout button.*

Appendix B- Software Requirement Specification (SRS)

Software Requirements Specification

for

<AUTOMATIC TEXT SUMMERIZATION>

Version 1.0

Supervised And Co-Supervised By:

Sir. Masood Hussain

Ms. Sonish Aslam

Prepared by Group# 34

Sir Syed University of Engineering & Technology

June 12th, 2023

Team

Member Name	Primary Responsibility
MUZAMMIL SARDAR ABBASI	ML MODEL CREATION AND LOGIN INTERFACE
IMAD KHAN	ML MODEL TESTING AND ADMIN PANEL
S.M ZEJAH ALI REHMANI	ML MODEL TRAINING AND REGISTRATION INTERFACE
M. AHSAN SIDDIQUE	ML MODEL VALIDATION AND ACCURACY CHECK

Table of Contents

1.Introduction

1.1 Purpose:

The purpose of our automatic textual content summarization (ATS) application is to help produce the precis of any article containing any sort of crucial and applicable information from the unique documented textual content. Summarization is the venture of condensing a piece of a file into its shorter variant, which lessens the initial record's size even as retaining the important piece of information intact.

1.2 Scope:

Our project is inspired by the huge problem people face when trying to read lengthy articles and blogs available on the internet. This is time-consuming and less effective in the long term Simply because there are fewer ways to summarize the work they want, with the resources that are suggested on the internet. As a result, people are often left confused that how are they going to achieve their goals if they don't have the proper resources at hand.

Using AI algorithms, our app addresses that issue and provides the user with the solution and guidance to help them achieve their desired summary with resources that are easily available to them.

1.3 Definitions, Acronyms, and Abbreviations:

- **ATS:** Automatic Text summarization (ATS) is currently a famous exploration region among experimenters. Automatic text summarization is the method of producing the subset of the primary textbook. This subset of the main text represents the entire text and the article's main idea. ATS is the crucial subject of Natural Language Processing (NLP) and Data Mining (DM). This consists of the abstractive and extractive summaries of the text.

- **CRUCIAL INFORMATION HANDLER:** As a way to create the precis of the applicable cloth, the summarizer extracts additional crucial information (either words or sentences) from the input content material. Consistent with the literature, there are five distinct varieties of textual content summarizing methods

- **SINGLE TEXT HANDLER:** In single-file text summarizing, a document is used to accomplish the summation, and a single output report is produced. Created a technique for textual content summarizing in a record the usage of automatic keyword extraction. A discourse-based summarizer turned into created by way of Marcu et al. To assess the suitability of texts for summarizing the use of discourse-based methodologies inside the context of single information.

- **RECORD OVERLAP:** Records overloading is one of the maximum pressing troubles added on by way of the net's explosive increase. Because there's so much data available on the net, condensing the pertinent facts right into a precis could be useful to many people. For humans, manually summarizing significant portions of textbook material is taxing. Researchers had been experimenting with methods to decorate precis creation such that the summaries produced with the aid of way of humans and machines are identical. This study gives an extensive chance assessment of the textbook summarizing generalities, in conjunction with strategies, datasets, evaluation requirements, and uncharted regions for research. The 2 methodologies included in-depth in this work which can be the maximum considerably stated are extractive and abstractive. Evaluating the synopsis. Moreover, the arrival of suitable coffers and structural help within the contrast and replication of findings, supply competitiveness to deal with the troubles. in these paintings, diverse evaluation strategies for produced summaries

1.4 Acronyms, and Abbreviations:

Here are some common acronyms and abbreviations related to automatic text summarization:

- ***NLP: Natural Language Processing***
- ***AI: Artificial Intelligence***
- ***ML: Machine Learning***
- ***DL: Deep Learning***
- ***ASR: Automatic Speech Recognition***
- ***POS: Part-of-Speech***
- ***TF-IDF: Term Frequency-Inverse Document Frequency***
- ***RNN: Recurrent Neural Network***
- ***LSTM: Long Short-Term Memory***
- ***CNN: Convolutional Neural Network***
- ***BERT: Bidirectional Encoder Representations from Transformers***
- ***ROUGE: Recall-Oriented Understudy for Gisting Evaluation***
- ***BLEU: Bilingual Evaluation Understudy***

- ***METEOR: Metric for Evaluation of Translation with Explicit Ordering***
- ***LSA: Latent Semantic Analysis***
- ***LDA: Latent Dirichlet Allocation***
- ***API: Application Programming Interface***
- ***GUI: Graphical User Interface***
- ***POS: Part-of-Speech***
- ***TTS: Text-to-Speech***

These acronyms and abbreviations are commonly used in discussions, research papers, and literature related to automatic text summarization.

2.Project Planning and Management

2.1 SWOT Analysis:

2.1.1 INTRODUCTION:

Mobile internet use has increased greatly in popularity due to technology's quick development as a means of entertainment. The market for mobile applications is constantly growing, and their field of influence is getting bigger. The general population has begun to notice a new product. Information overloading is one of the maximum pressing troubles delivered on with the aid of the net's explosive expansion. Because there is a lot of information to be had on the internet, condensing the pertinent information into a precis might be useful to many people. For humans, manually summarizing sizeable quantities of textbook fabric is taxing. Researchers have been experimenting with techniques to improve precis advent such that the summaries produced by way of people and machines are the same. This takes look at offers an intensive threat evaluation of textbook summarizing generalities, consisting of strategies, datasets, assessment standards, and uncharted regions for investigation. The two methodologies protected in-depth in these paintings which might be the most broadly stated are extractive and abstractive. Comparing the synopsis. Moreover, the introduction of appropriate coffers and structural assistance in the comparison and replication of findings offer competitiveness to deal with the issues. In this work, numerous assessment techniques for produced summaries also are discussed. Sooner or later, towards the conclusion of this examination, some of the difficulties and areas for further investigation into textbook summary are cited that can be useful for implicit experimenters running in this field.

SWOT analysis is a strategic planning tool used to evaluate the strengths, weaknesses, opportunities, and threats of a particular entity or concept. Here's a SWOT analysis specifically focused on automatic text summarization:

STRENGTHS:

Efficiency: Automatic text summarization can quickly process and summarize large volumes of text, saving time and effort for users.

Scalability: The technology can be applied to diverse domains and accommodate varying document lengths, making it suitable for a wide range of applications.

Information Filtering: Automatic summarization helps users extract relevant information from a text, allowing for efficient decision-making and information retrieval.

Language Independence: The technology can be applied to multiple languages, enabling cross-lingual summarization and making it accessible to a global audience.

Adaptability: Automatic summarization algorithms can be trained and fine-tuned on specific datasets or domains, allowing for customization and improved performance in specific contexts.

WEAKNESSES:

Semantic Understanding: Automatic summarization algorithms may struggle with fully understanding the context, nuances, and underlying meaning of the text, leading to potential loss of information or misinterpretation.

Subjectivity: Different users may have varying preferences for what constitutes an effective summary, and automatic summarization may not always align with individual expectations.

Evaluation Challenges: Measuring the quality of summaries can be subjective, and evaluation metrics may not fully capture the nuances of a well-written summary.

Language Complexity: Automatic summarization faces challenges when dealing with complex sentence structures, figurative language, or domain-specific jargon, potentially leading to inaccuracies or inadequate summaries.

OPPORTUNITIES:

Enhanced User Experience: Advances in automatic summarization can improve user experiences by providing more accurate, relevant, and concise summaries tailored to individual needs.

Personalization: Incorporating user preferences and feedback can lead to personalized summarization systems that better align with individual requirements and improve user satisfaction.

Multimodal Summarization: Automatic summarization can be extended to include other modalities, such as images, audio, or video, providing users with a more comprehensive and holistic understanding of the content.

Real-time Summarization: Developing techniques for on-the-fly summarization can enable real-time applications, such as live event summarization or summarization in conversational agents.

THREATS:

Ethical Considerations: Automatic summarization may raise concerns about information bias, privacy, and the potential for manipulation or distortion of information.

Legal Implications: Summarization algorithms should adhere to copyright laws and intellectual property rights, as they involve extracting and reproducing content from the original text.

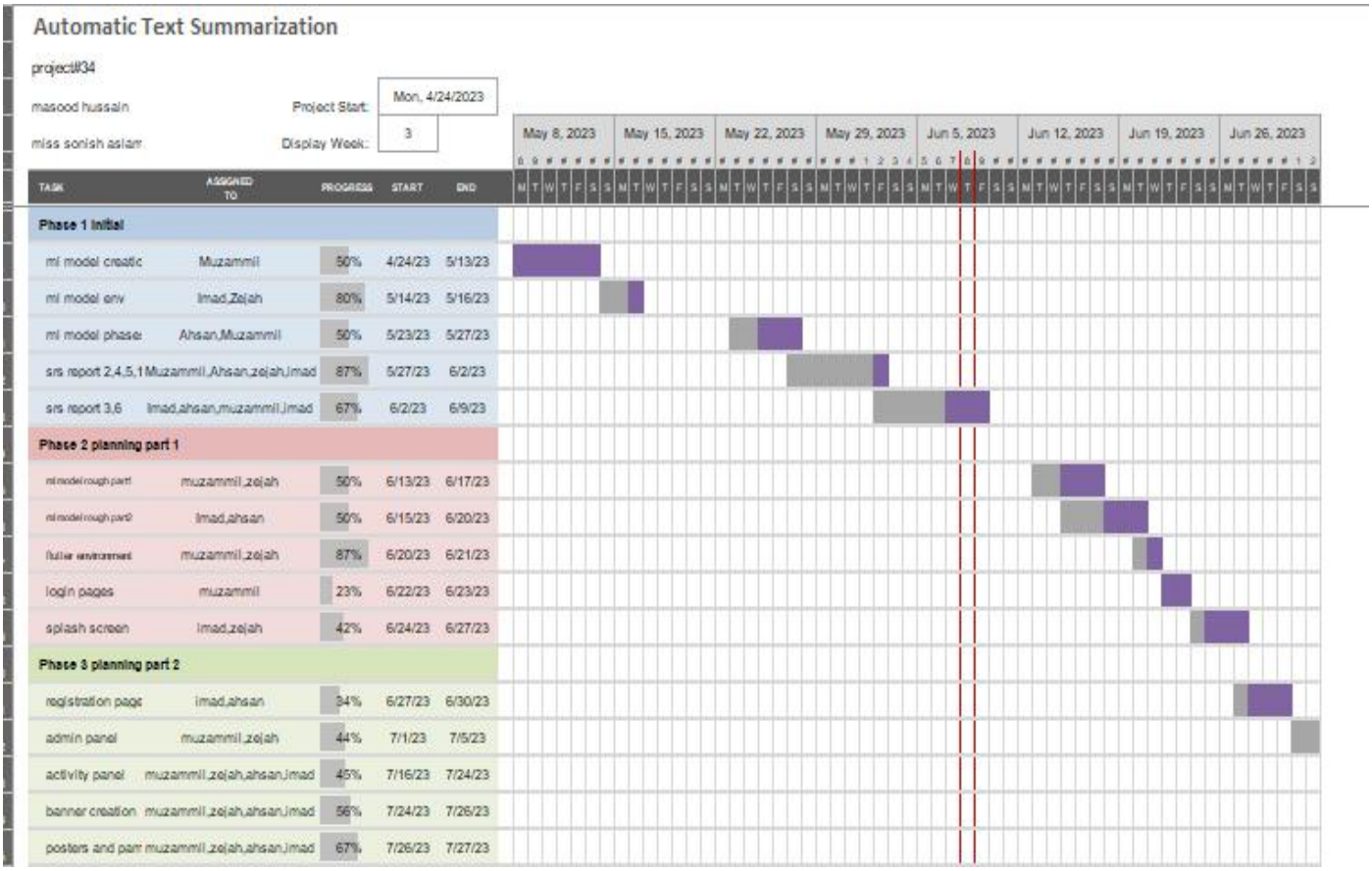
Quality Assurance: Ensuring the accuracy, reliability, and consistency of automatic summarization systems poses a challenge, as errors or biases in summaries can have significant consequences in critical applications.

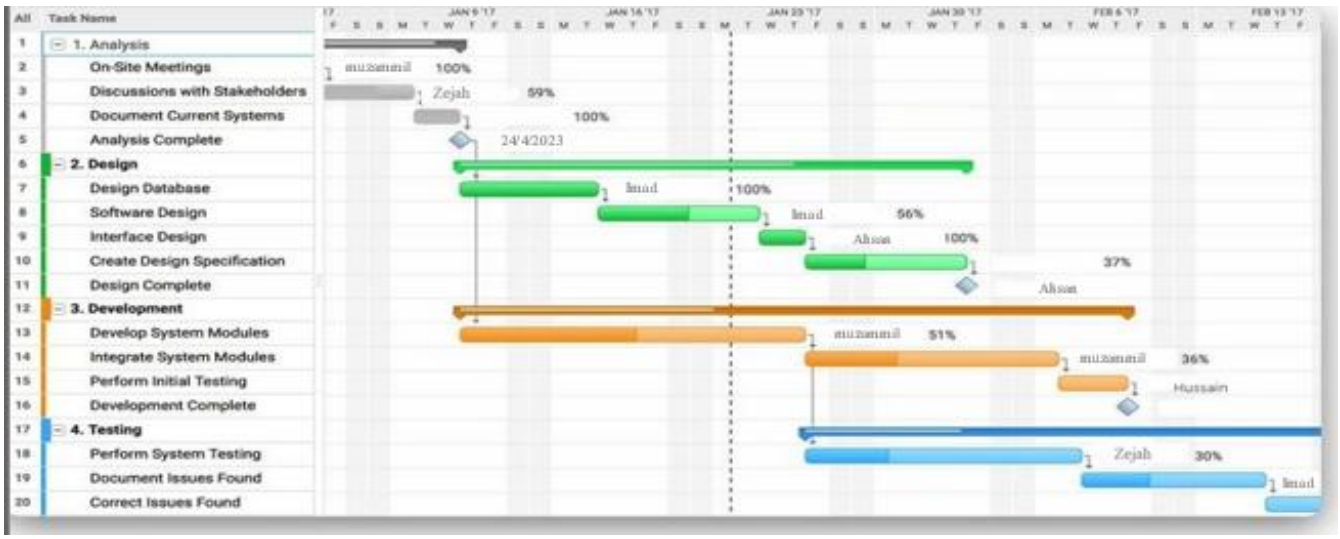
Competition: As automatic summarization gains popularity, increased competition may arise among different approaches and technologies, requiring continuous innovation to stay relevant.

This SWOT analysis provides an overview of the strengths, weaknesses, opportunities, and threats associated with automatic text summarization. It can help in understanding the current landscape of automatic summarization and guide decision-making and future development in the field.


SWOT Analysis of Automatic Text Summarization
STRENGTHS
Summarize the text into shorter variants.
Be accurate and to the point.
Reduction of reading time on lengthy articles.
To preserve the main idea from the original document.
Provide the user with a quick and easy-to-understand document.
WEAKNESSES
more textual data require more time to process.
have handle management issues over various same data set provided
OPPORTUNITIES
Summarize the true meaning and message of the original text.
Provide the user with the overview they want of any article.
Help users avoid redundancy from various sources.
To increase the diversity of the provided content.
makes information gathering easy
THREATS
rejection of large amount of data
hard to handle various documented text at a same time
can become a cause of data leak

2.1 Gantt Chart:





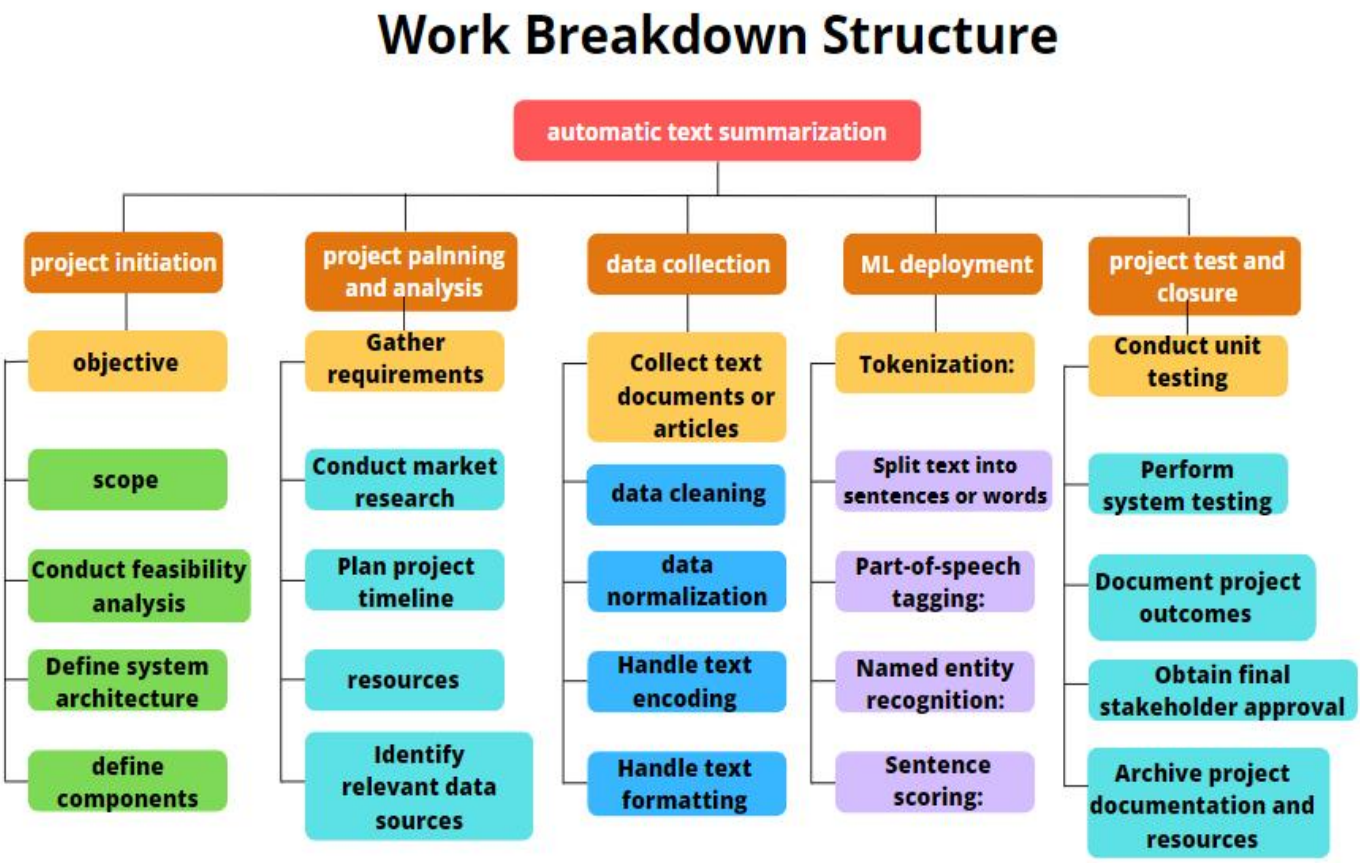
Member 1: S.M Zejah Ali Rehmani (Roll no: 2020-SE-154) will be responsible for creating datasets, incorporating datasets, developing logos and textures, as well as documentation.

Member 2: Muzammil Sardar Abbasi (Roll no: 2020-SE-165) will be responsible for the application's UI/UX design, development, and unit testing, as well as its API development, integration, artificial intelligence (AI), and database creation.

Member 3: Muhammad Ahsan Siddique (Roll no: 2020-SE-218) will be responsible for the documentation and system testing of the application on various devices.

Member 4: Imad Khan (Roll no: 2020-SE-176) will be responsible for developing some application functions as well as documentation.

2.2 Work Breakdown Structure [WBS]



- **Project Initiation**
- project objective and scope
- Stakeholders and requirements
- Conduct analysis
- Research and Planning

- **Gather requirements from stakeholders**
- Conduct market research and analysis
- Define system architecture and components
- Plan project timeline and resources

- **Data Collection and Preprocessing**
- Identify relevant data sources
- Collect text documents or articles
- Perform data cleaning and normalization
- Handle text encoding and formatting

- **Text Processing and Analysis**
- Tokenization: Split text into sentences or words
- Part-of-speech tagging: Identify grammatical components
- Named entity recognition: Identify entities (people, places, etc.)
- Keyword extraction: Identify important terms
- Sentence scoring: Determine sentence importance

- **Summarization Algorithm Development**
- Develop extractive summarization algorithms
- Evaluate and refine algorithms based on performance
- Experiment with different techniques (e.g., graph-based, statistical)
- Fine-tune parameters for optimal results

- **System Integration**
- Develop an interface for text input and output
- Integrate the summarization algorithm with the system
- Implement error handling and exception management
- Test the system's functionality and usability

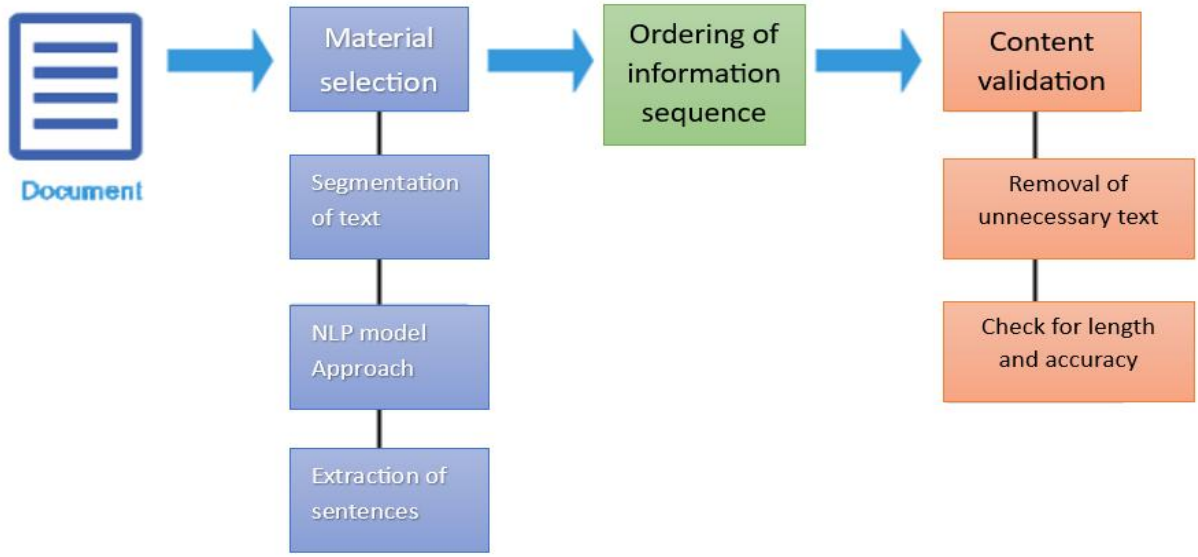
- **User Interface and Experience**
- Design a user-friendly interface for input and output
- Implement features for customization and personalization
- Ensure responsive and intuitive user interactions
- Incorporate user feedback and iterate on improvements

- **Testing and Evaluation**
- Conduct unit testing for individual components
- Perform system testing for end-to-end functionality
- Evaluate the quality of generated summaries
- Collect user feedback and make necessary adjustments

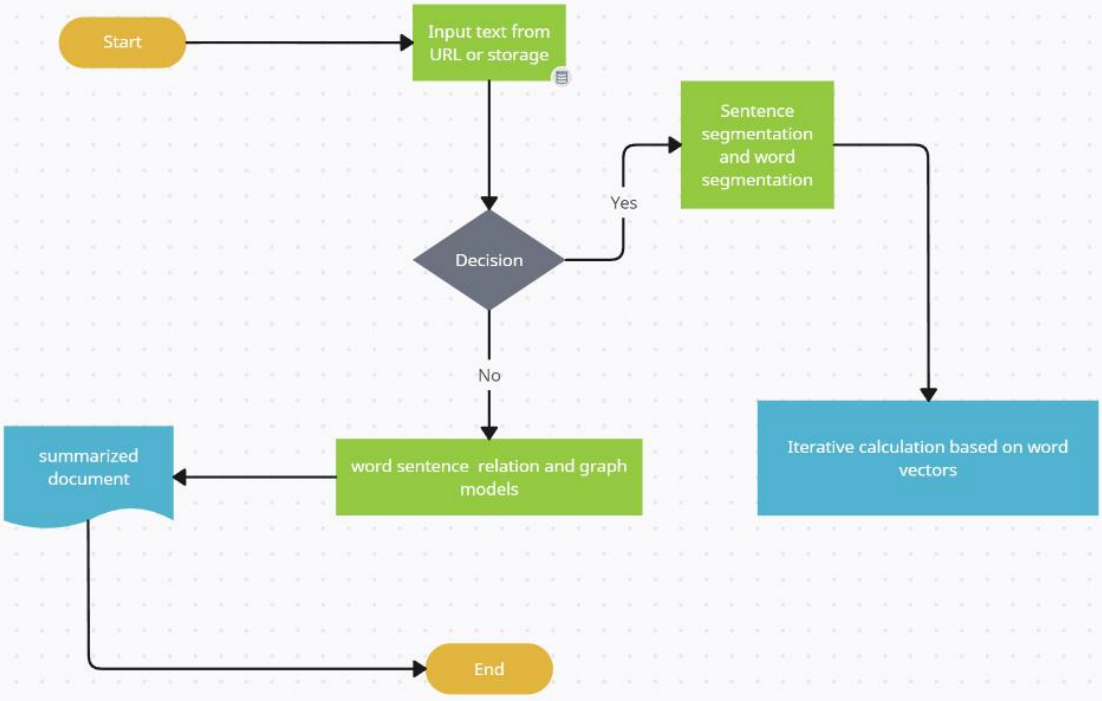
- **Deployment and Maintenance**
- Prepare the system for deployment in a production environment
- Ensure scalability and performance optimization
- Create documentation and user guides
- Provide ongoing maintenance and support

- **Project Closure**
- Conduct a project review and lessons learned session
- Document project outcomes and deliverables
- Obtain final stakeholder approval
- Archive project documentation and resources

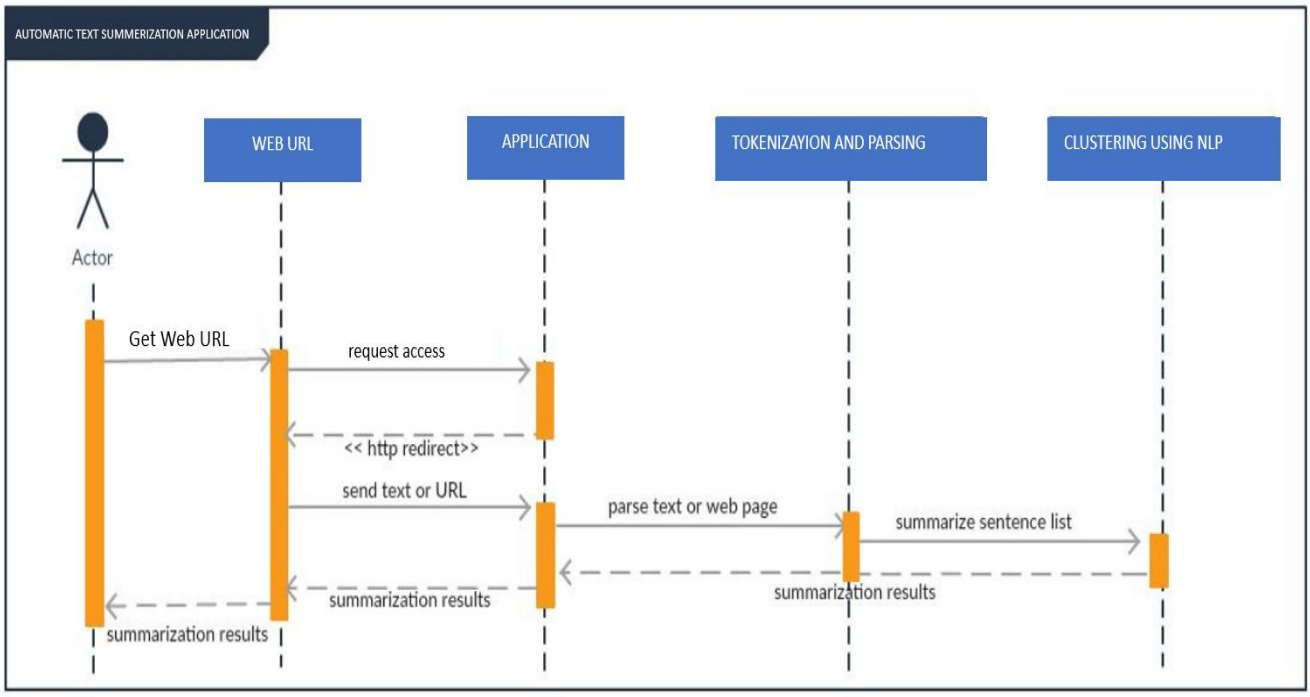
NLP ALGORITHM DIAGRAM



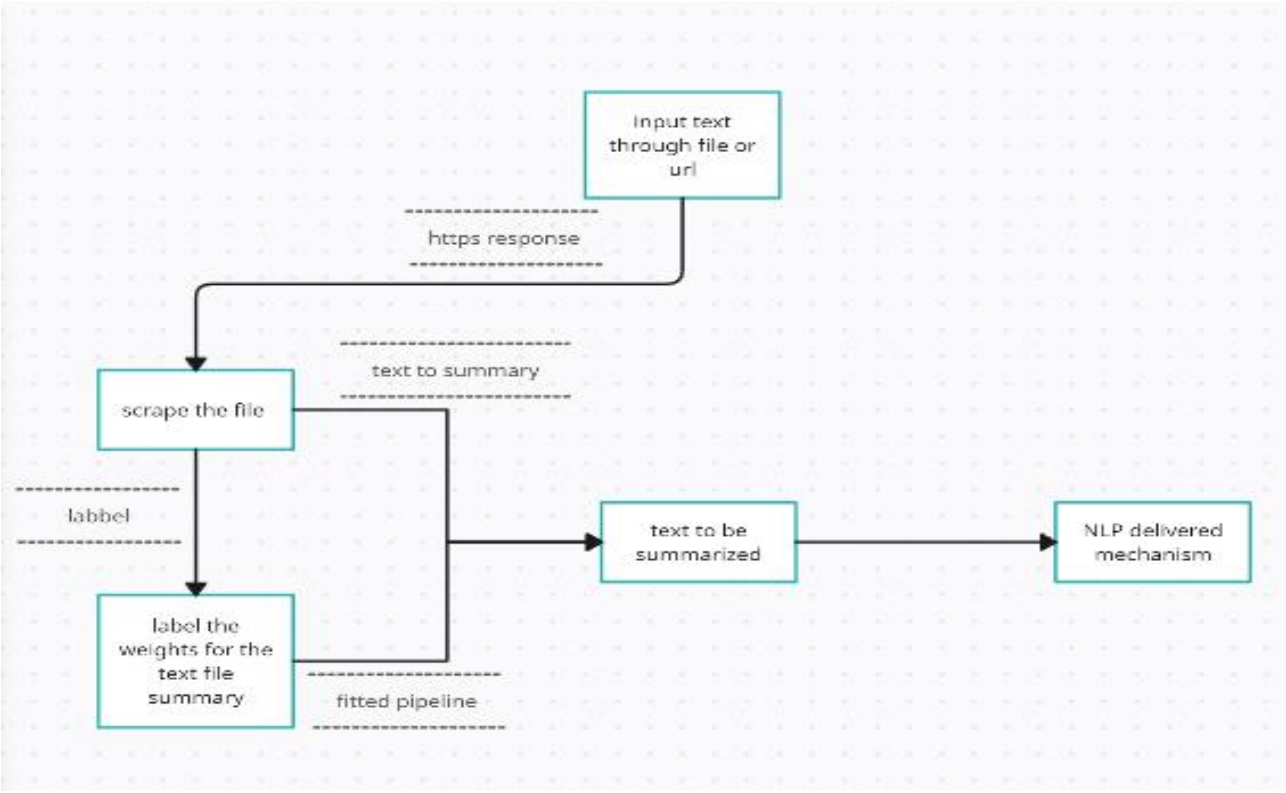
FLOWCHART DIAGRAM



UML SEQUENCE DIAGRAM



DATA FLOW DIAGRAM



3.Overall Description

3.1 Product Perspective:

It is established that there are many websites on the internet that suggest you dietsand articles that help you with your effectiveness, but most of them fail to provide useful solutions to the reading time problem, simply because people don't have mentioned plans and resources available.
So, our app will be specially designed for the people which will provide them with all one solution using an AI recommendation system that will be based on AI model NLP.

3.2 Product Function:

3.2.1 Input and Analyze:

Our app will be able to analyze the provided URL and check the current state of the user-provided text data to be summarized.

3.2.2 Data rearrange:

Our app will rearrange the data provided to be taken into bits for the AI model to be scraped and used to label it.

3.3 Operating Environment:

The operating environment for text summarization can vary depending on the specific implementation and deployment scenario. However, here are some key elements typically involved in the operating environment for text summarization systems:

Hardware Infrastructure:

Servers or cloud-based infrastructure to host the text summarization system.
Sufficient computational resources to handle the processing requirements, especially for large-scale summarization tasks.

Software Dependencies:

Programming languages and frameworks for developing the summarization system.
Text processing libraries and tools for tasks such as tokenization, part-of-speech tagging, and named entity recognition.
Machine learning or natural language processing libraries for implementing the summarization algorithms.

Text Data Sources:

Access to a wide range of text data sources, such as news articles, research papers, or online content, depending on the application domain.

APIs or data connectors to retrieve or access text data from various sources.

Preprocessing Tools:

Text cleaning and normalization tools to remove noise, irrelevant characters, or formatting inconsistencies from the input text.

Stop word removal techniques to filter out common words that do not contribute significantly to the summary.

Summarization Algorithms:

Extractive or abstractive summarization algorithms, depending on the chosen approach.

Sentence scoring mechanisms to determine the importance or relevance of sentences.

Statistical or graph-based algorithms for identifying key phrases or concepts.

Evaluation Metrics:

Metrics and evaluation methodologies to assess the quality and effectiveness of the generated summaries.

Common evaluation metrics include ROUGE (Recall-Oriented Understudy for Gusting Evaluation), BLEU (Bilingual Evaluation Understudy), or other domain-specific metrics.

Integration Interfaces:

Input interfaces to accept text documents or articles for summarization.

Output interfaces to present the generated summaries in the desired format, such as plain text, HTML, or JSON.

Customization and Configuration Options:

Configuration settings to customize the summarization process based on specific requirements, such as summary length constraints or domain-specific terminology.

Scalability and Performance Optimization:

Techniques to handle large volumes of text data efficiently, such as parallel processing or distributed computing.

Caching mechanisms or indexing structures to improve performance when processing similar or recurring text data.

Monitoring and Error Handling:

Logging and monitoring mechanisms to track the system's performance, usage statistics, and potential errors.

Error handling and exception management to gracefully handle unexpected scenarios during the summarization process.

Security and Privacy Considerations:

Data security measures to protect sensitive or confidential text data.
Compliance with privacy regulations, especially if the text data contains personal or sensitive information.

3.4 Design and Implementation Constraints:

To understand design and implementation constraints while building an appusing Flutter we need to understand this:

- A widget gets its own constraints from its parent. A constraint is just a set of 4 doubles: a minimum and maximum width, and a minimum and maximum height.
- Then the widget goes through its own list of children. One by one, the widget tells its children what their constraints are (which can be different for each child), and then asks each child what size it wants to be.
- Then, the widget positions its children (horizontally on the x-axis, and vertically on the y-axis), one by one.
- And, finally, the widget tells its parent about its own size (within the original constraints, of course).

As a result of the layout rule mentioned above, Flutter's layout engine has a few important limitations:

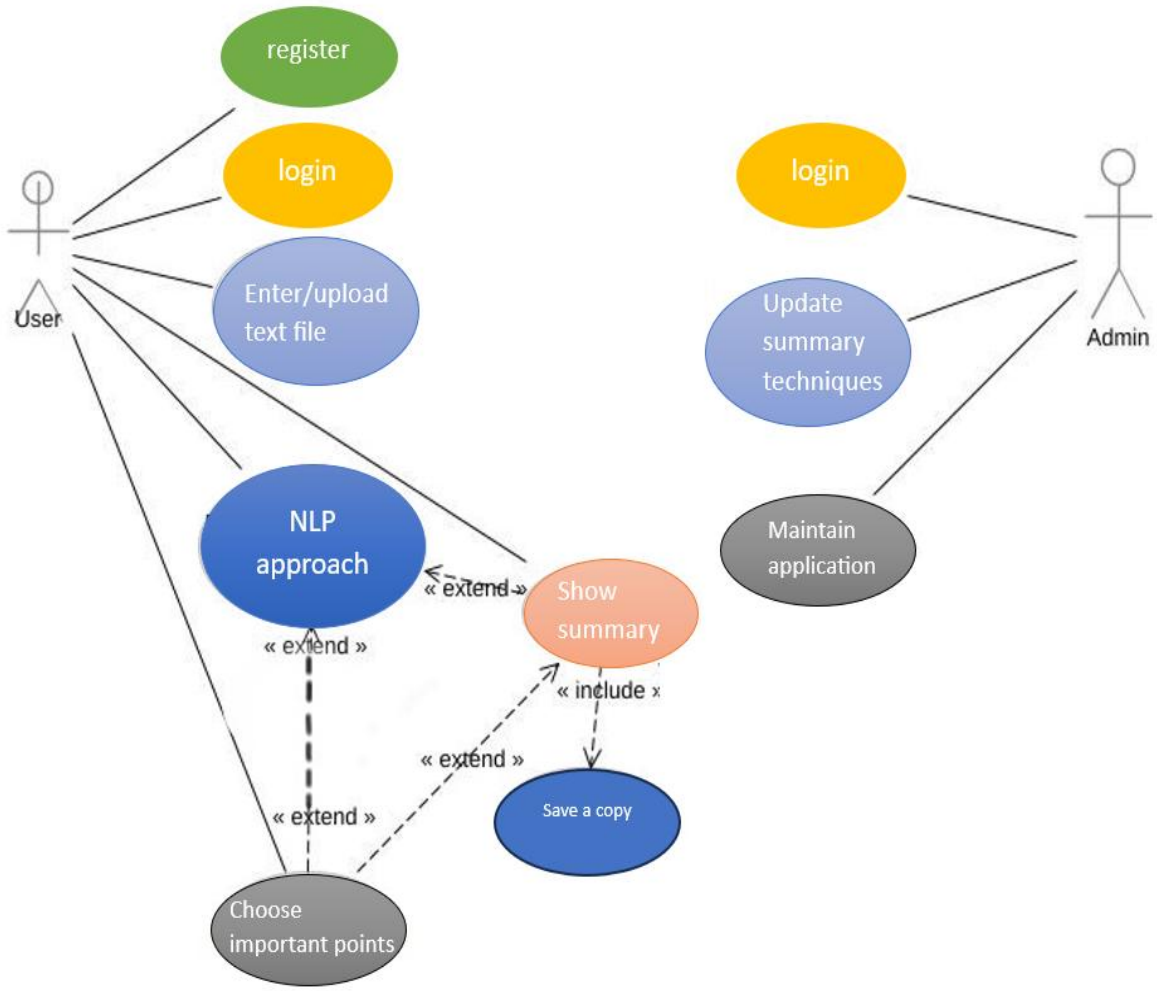
A widget can decide its own size only within the constraints given to it by its parent. This means a widget usually can't have any size it wants**.**
A widget can't know and doesn't decide its own position on the screen**,** since it's the widget's parent who decides the position of the widget.
Since the parent's size and position, in its turn, also depends on its own parent, it's impossible to precisely define the size and position of any widget without taking into consideration the tree as a whole.
If a child wants a different size from its parent and the parent doesn't have enough information to align it, then the child's size might be ignored. We need to be specific when defining alignment.

3.5 Assumptions and Dependencies:

- If no users create any exercise routines, there won't be any listed for users in the application. Users won't be able to log in or access any application features if there is no internet connection.
- Unregistered users won't be able to utilize the software application.
- After accepting the terms of the service agreement, users will only be able to register and access the software.

4.Requirement Identifying Technique

4.1 Use Case Diagram



4.2 Use Case Description

In this diagram, we have two main actors: The User and the Automatic Text Summarization System.

The User interacts with the system and can perform the following actions:

Enter Text: The User provides the input text that needs to be summarized.

View Summary: The User requests to see the generated summary of the input text. The Automatic Text Summarization System, represented as a single entity, performs the core functionalities of the system. It includes the following use cases:

Process Text: The system processes the input text by performing various tasks such as tokenization, part-of-speech tagging, named entity recognition, and other preprocessing

Generate Summary: Based on the processed text, the system applies automatic summarization algorithms to generate a concise summary of the input text. Return Summary: The system returns the generated summary to the User, who can then view it.

5. Non- Functional Requirements

5.1 Performance Requirements:

Our application performance is fast and responsive which provides fast access to data from Firebase which takes 1-2 seconds to fetch data, and which is dependent upon user connection speed. When considering the performance requirements for an automatic text summarization Android application, several factors should be considered to ensure optimal user experience and efficient processing. Here are some key performance requirements to consider:

Responsiveness: The application should provide a responsive user interface, ensuring that users experience minimal delays or lag when interacting with the app. The summarization process should be executed efficiently, allowing users to receive summaries in a timely manner.

Processing Speed: Efficient text summarization algorithms and techniques should be employed to achieve fast processing speeds. Users expect the app to generate summaries quickly, even when dealing with large volumes of text.

Scalability: The application should be designed to handle varying workloads and accommodate increasing amounts of text data. It should be capable of processing long documents or multiple documents concurrently without significant performance degradation.

Resource Usage: Optimize the app's resource utilization, such as CPU, memory, and battery consumption, to minimize the impact on device performance and battery life. Avoid excessive resource usage that could slow down the device or drain the battery quickly.

Offline Capability: Consider providing offline text summarization capabilities to allow users to generate summaries without requiring a constant internet connection. This feature can improve performance and usability, particularly in areas with limited or unreliable network connectivity.

Caching and Preprocessing: Implement intelligent caching mechanisms to store previously generated summaries or intermediate processing results. This can help reduce redundant computations and improve overall performance, especially when summarizing similar or frequently accessed texts.

Multithreading and Asynchronous Processing: Leverage multithreading and asynchronous processing techniques to improve performance. This allows the app to perform summarization tasks in the background while keeping the user interface responsive and interactive.

Memory Management: Efficiently manage memory usage, especially when dealing with large text inputs. Implement strategies such as chunking or streaming to process texts in manageable portions, minimizing memory requirements and improving performance.

Error Handling and Recovery: Handle errors and exceptions gracefully to ensure the app's stability and prevent crashes or unexpected behavior. Implement appropriate error-handling mechanisms and recovery strategies to maintain a smooth user experience.

Performance Monitoring and Optimization: Continuously monitor the app's performance metrics, such as response time, processing speed, and resource usage. Analyze performance data to identify bottlenecks, optimize algorithms, and improve overall efficiency.

By addressing these performance requirements, we can ensure that our automatic text summarization Android application delivers fast, responsive, and efficient summarization capabilities, providing users with a smooth and satisfactory experience.

5.2 Safety Requirements:

Our application uses secure Firebase authentication, and Firebase verification, and has (SUMIT) summary secure API which only responds to requests with secure access tokens, no third party can access our API to secure from unauthorized access from other users or internet data stealing. When developing an automatic text summarization Android app, it's important to consider safety requirements to ensure the privacy, security, and ethical use of the app. Here are some key safety requirements to consider:

Data Privacy: Protect the privacy of users' text data by implementing strong data protection measures. Use encryption techniques to secure sensitive data, both during transit and storage. Clearly communicate the app's data privacy policy and obtain user consent for data collection and usage.

User Authentication and Authorization: Implement robust user authentication mechanisms to prevent unauthorized access to the app and user data. Consider integrating secure login methods such as two-factor authentication for enhanced security.

Secure Communication: Ensure that all communications between the app and external servers or APIs are encrypted using secure protocols such as HTTPS. This prevents eavesdropping and protects user data during transmission.

Secure Storage: Store user data, including summaries and user preferences, in a secure manner. Use encryption and access controls to protect data at rest and prevent unauthorized access or data breaches.

Ethical Use: Design and implement the app to adhere to ethical guidelines and best practices. Avoid using the app for malicious purposes, such as generating misleading or harmful summaries. Clearly communicate the app's intended use and limitations to users.

Error Handling: Implement appropriate error handling mechanisms to ensure the app remains stable and secure. Gracefully handle exceptions, validate user input, and provide clear error messages to enhance user experience and prevent potential security vulnerabilities.

User Consent and Transparency: Obtain explicit consent from users for any data collection, processing, or sharing activities. Clearly communicate how the app uses and stores user data, including summaries, and provide users with control over their data through privacy settings or options to delete data.

Regular Updates and Security Patches: Maintain the app by regularly releasing updates and security patches to address any identified vulnerabilities or issues. Promptly address security concerns to ensure the app remains safe and secure for users.

Third-Party Libraries and APIs: When using third-party libraries or APIs for text processing or summarization, ensure they are reputable, regularly updated, and have a strong track record of security. Keep dependencies up to date to benefit from security fixes and improvements.

User Support and Reporting: Provide users with a mechanism to report any issues, vulnerabilities, or concerns related to the app's safety and security. Establish a support system to address user queries, aid, and promptly respond to any security incidents.

By incorporating these safety requirements into the development process, we can enhance the security, privacy, and ethical use of our automatic text summarization Android app, providing users with a safe and trustworthy experience.

5.3 Security Requirements:

To prevent unauthorized access from other users or the theft of internet data, our software uses secure Firebase authentication and Firebase verification. It also has a (SUMIT) owned secure API that only replies to requests with secure access tokens.

5.4 Software Quality Attributes:

Software quality attributes refer to the desirable characteristics or qualities of a software application. When it comes to an automatic text summarization application, several key quality attributes should be considered to ensure a high-quality and reliable system. Here are some important software quality attributes for an automatic text summarization application:

Accuracy: The application should generate accurate summaries that capture the key information and meaning of the input text. The summaries should be relevant, coherent, and free from significant errors or distortions.

Reliability: The application should consistently produce reliable and trustworthy summaries. It should perform consistently across different inputs and demonstrate a high level of stability and robustness.

Performance: The application should be efficient and provide timely responses. It should generate summaries within acceptable time frames, even when dealing with large volumes of text. Performance requirements, such as response time and throughput, should be met to ensure a satisfactory user experience.

Usability: The application should be user-friendly and intuitive, with a well-designed user interface. Users should be able to interact with the app easily, input text, and receive summaries without confusion or unnecessary complexity. Consider incorporating features like clear instructions, error handling, and intuitive navigation.

Maintainability: The application should be designed and implemented in a way that facilitates easy maintenance and future enhancements. This includes writing clean, modular, and well-documented code, using best practices, and employing software engineering principles. Clear separation of concerns, code readability, and appropriate documentation contributes to maintainability.

Scalability: The application should be designed to handle varying workloads and accommodate increasing demands as the user base or text input size grows. It should be able to scale horizontally or vertically, ensuring that it can handle additional users, concurrent requests, or larger text inputs without significant performance degradation.

Security: The application should incorporate security measures to protect user data, prevent unauthorized access, and ensure secure communication. This includes employing encryption techniques, implementing secure authentication and authorization mechanisms, and following best practices for data privacy and protection.

Testability: The application should be designed with testability in mind, allowing for effective testing and quality assurance. This includes writing testable code, providing appropriate testing interfaces or APIs, and employing automated testing methodologies to ensure the correctness and reliability of the application.

Adaptability: The application should be adaptable to different domains, languages, or specific requirements. It should be flexible enough to handle various types of text inputs and produce meaningful summaries irrespective of the content or context.

Accessibility: Consider incorporating accessibility features to make the application usable by a wide range of users, including those with disabilities. This involves adhering to accessibility guidelines, providing appropriate text alternatives, and supporting assistive technologies.

By considering these software quality attributes, you can develop an automatic text summarization application that delivers accurate, reliable, efficient, and user-friendly summarization capabilities, meeting the expectations and requirements of your users.

5.5 Business Rules:

It's important to note that these rules can be adjusted and customized based on the specific needs and constraints of each business, as well as the capabilities of the automatic text summarization system being used.

Length Constraint: Set a maximum limit for the length of the generated summary. This can be based on the desired output format or platform where the summary will be displayed. For example, if the summary is meant to be displayed on a mobile device, it may need to be shorter to fit the limited screen space.

Content Relevance: Ensure that the summary focuses on the most important and relevant information from the original text. This can be achieved by using algorithms that analyze the importance of sentences or keywords in the text and prioritize them for inclusion in the summary. This can involve techniques such as natural language processing and syntactic analysis to ensure that the summary reads well and conveys the intended meaning accurately.

Context Preservation: Preserve the context of the original text as much as possible. While summaries aim to condense information, it is important to ensure that the summary retains the core message and context of the original text. This can be achieved by including key details, references, or connections that provide a holistic understanding of the subject matter. This can help users quickly identify key elements in the summarized content and provide additional context or reference points.

Plagiarism and Copyright: Implement mechanisms to avoid plagiarism and respect copyright laws. Automatic text summarization should not infringe upon the intellectual property rights of the original content creators. This can be accomplished by using techniques such as paraphrasing, proper citation, or obtaining appropriate licenses for the source content. Businesses may have different requirements or preferences for summarization based on their specific domains or target audiences. Allowing users to customize the summary generation process can enhance the usefulness and relevance of generated summaries.

Performance and Efficiency: Optimize the summarization system for speed and scalability. Depending on the scale of text data to be processed, the system should be designed to handle large volumes of data efficiently and provide summaries within acceptable timeframes.

Evaluation and Feedback: Establish mechanisms to evaluate the quality and effectiveness of the generated summaries. Feedback from users or automated evaluation metrics can be used to continuously improve the summarization algorithms and ensure that the summaries meet the desired standards.

5.6 Interoperability

After passing through a few security checks, our app interacts with the database and API in roughly a second. This allows several users to interact simultaneously.

5.7 Extensibility

The extensibility of automatic text summarization refers to the ability to adapt and extend the existing methods and systems to cater to different domains, languages, or specific requirements. Extensibility is an important aspect of automatic text summarization as it allows the technology to be applied effectively in diverse contexts. Here are a few key points regarding the extensibility of automatic text summarization:

1. Domain Adaptation: Automatic text summarization systems need to be adaptable to different domains, such as news articles, scientific papers, legal documents, or social media posts. The system should be able to handle the specific language, jargon, and characteristics of the given domain to produce accurate and relevant summaries.

2. Multilingual Summarization: Extensibility involves the capability of summarizing texts in multiple languages. Language-specific challenges, such as word order, grammar, and semantic nuances, need to be considered to generate high-quality summaries in different languages.

3. Customization for Specific Needs: Different users or applications may have specific requirements for summaries. The extensibility of text summarization systems enables customization based on factors such as length restrictions, domain-specific keywords, or preferred summarization styles (extractive vs. abstractive).

4. Incorporating New Data Sources: Automatic text summarization should be flexible enough to integrate and summarize content from various sources, including online articles, blogs, social media feeds, or real-time streams. The system should handle different formats and adapt to the evolving nature of data sources.

5. Fine-tuning and Transfer Learning: Extensibility involves leveraging pre-trained models and transfer learning techniques. Fine-tuning existing models with domain-specific data or adapting models from related tasks can enhance the performance and applicability of automatic text summarization to new contexts.

6. Open APIs and Toolkits: Extensible summarization frameworks provide APIs and toolkits that allow developers to integrate and customize the summarization capabilities within their applications or workflows. These interfaces enable the extension of summarization functionalities to meet specific application requirements.

7. Research and Development: The extensibility of automatic text summarization relies on continuous research and development efforts. Advancements in machine learning, natural language processing, and text summarization techniques contribute to the extensibility of the technology.

By focusing on extensibility, automatic text summarization systems can be adapted and extended to cater to a wide range of applications, languages, and domains, making them more versatile and useful in various contexts.

5.8 Maintainability

our application should be accessible 99.99% of the time.It is recommended to carry out any software updates, patches, and fixes without terminating the application. To address disasters, a disaster recovery ecosystem should be in place.

5.9 Portability

Our application is very portable because it can be deployed on any Android devicewith version 7 or higher and any IOS device with version 9 or higher. It can be deployed via the application's APK or from the Google Play Store or App Store at any time. The application is versatile and compatible with a variety of Android and IOS devices.

5.10 Reusability:

Our app has a lot of reusable widgets and components that are simple to integrate into many other applications. For example, the authentication process can be used in another Flutter app, just like in SUMIT, which offers full security through Firebase. Components that read APIS and convert them into lists can also be reused. Over 80% of our app is reusable and can benefit many other applications, both with and without some modifications.

5.11 Installation:

Android:

Installation of Android is possible through the Play Store or the app of our app.

IOS:

Installation can only be done through App Store.

6. Other Requirements

6.1 On-line User Documentation and Help System Requirements

In an online user documentation and help system for a text summarization app, several requirements should be considered to ensure effective support for users. Here are some important requirements:

1: Clear and comprehensive documentation: The user documentation should provide clear instructions on how to use the text summarization app. It should cover all features and functionalities, explaining them in a user-friendly manner. The documentation should be well-structured and easy to navigate, allowing users to quickly find the information they need.

2: Search functionality: A search feature within the documentation is crucial for users to find specific information or answers to their questions. It should be able to search for keywords and provide relevant results, saving users time and effort.

3: Interactive tutorials: Including interactive tutorials or walkthroughs can be beneficial for new users. These tutorials should guide users through the app's features step-by-step, allowing them to practice and understand the summarization process effectively.

4: Frequently Asked Questions (FAQs): A dedicated section for frequently asked questions can help address common user queries and issues. Compile a list of relevant FAQs and provide clear and concise answers to assist users in troubleshooting or understanding the app better.

5: Contextual help and tooltips: Implementing contextual help and tooltips within the app can provide users with on-the-spot guidance. When users hover over or click on specific elements or options, relevant information, and explanations should be displayed, aiding users in understanding the app's functionality in real-time.

6: Visual aids and examples: Incorporate visual aids, such as screenshots or videos, to demonstrate specific tasks or processes within the app. Examples of summarized texts and before-and-after comparisons can help users grasp the app's capabilities and potential output.

7: User feedback and support channels: Provide a feedback mechanism within the documentation or app interface, allowing users to report issues or suggest improvements. Additionally, include contact information or links to customer support channels, such as email, live chat, or community forums, where users can seek assistance or engage with other users.

8: Mobile-friendly and responsive design: Ensure that the online documentation and help system is mobile-friendly and responsive, adapting to different screen sizes and devices. This is important as users may access the documentation from various platforms, including smartphones and tablets.

9: By considering these requirements: we can create an effective online user documentation and help system for a text summarization app, supporting users in understanding and utilizing the app's features to their fullest potential.

6.2 Purchased Requirements

There might be some charges if the user wants to make his own customized plan.

6.3 Licensing Requirements

It shall be taken as per requirement.

6.4 Legal, copyright, and Other Notices

It shall be taken as per requirement.

6.5 Applicable Standards

It shall be taken as per requirement.

6.6 Reports (Feedback, Invoice, User, Usage, Balance Sheet, Executive Summary, etc.)

Automatic text summarization is a useful technique for condensing large volumes of information into concise summaries. While it is commonly used for news articles, research papers, and other textual content, it can also be applied to various types of reports. Here are some examples of reports that can benefit from automatic text summarization:

Feedback Reports: These reports contain customer, user, or employee feedback and reviews. Automatic text summarization can extract key insights and sentiments from these reports, concisely summarizing the overall feedback received.

User Reports: User reports typically include data and analytics related to user behavior, engagement, and interactions with a product or service. Automatic text summarization can extract crucial trends, patterns, and user preferences, offering a concise overview of user activities and insights.

Usage Reports: These reports focus on tracking the usage or consumption of a particular resource or service. Summarizing usage reports can summarize resource utilization, peak usage periods, and other significant usage patterns.

Executive Summary: An executive summary provides an overview of a longer report, highlighting the main points, conclusions, and recommendations. Automatic summarization can help generate a concise summary of the key findings, enabling busy executives to quickly grasp the main insights without going through the entire report.

When applying automatic text summarization to these reports, it's essential to consider the specific requirements and context of each report type. Different algorithms and techniques can be utilized, such as extractive summarization, which selects and condenses important sentences, or abstractive summarization, which generates a summary using natural language generation techniques. The choice depends on the available data, desired level of detail, and the specific needs of the end-users.

7.References:

1. "Text Summarization Techniques: A Brief Survey" by Inderjeet Kaur and Preeti Saini - This research paper provides an overview of different text summarization techniques, including extractive and abstractive approaches. You can search for it using the title and authors. [link] https://query.data.world/s/lpim7d7ewx33ykadthda623eminani

2. "A Neural Attention Model for Abstractive Sentence Summarization" by Alexander M. Rush, Sumit Chopra, and Jason Weston - This paper introduces an abstractive text summarization model based on neural attention mechanisms. Searching for the title and authors will lead you to the paper. [link] https://www.researchgate.net/publication/281487270_A_Neural_Attention_Model_for_Abstractive_Sentence_Summarization

3. "TextRank: Bringing Order into Texts" by Rada Mihalcea and Paul Tarau - This paper presents the TextRank algorithm, which is a graph-based approach for extractive text summarization. Search for the title and authors to find the paper. [link] https://www.researchgate.net/publication/286694529_Application_of_TextRank_Algorithm_for_Credibility_Assessment

4. "Deep Reinforcement Learning for Sequence-to-Sequence Models of Text Summarization" by Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky - This research paper explores the application of deep reinforcement learning for abstractive text summarization. Search for the title and authors to access the paper. [link] https://arxiv.org/pdf/1812.02303

5. "Attention Is All You Need" by Vaswani et al. - This influential paper introduces the Transformer model, which has been widely used in natural language processing tasks, including text summarization. Searching for the title and authors will lead you to the paper. [link] https://arxiv.org/abs/1706.03762

Appendix C- Dissemination Activity

8.3 Detail of Conference

Pakistan Journal of Engineering and Applied Sciences (PJEAS) is a scientific journal that acts as repository for the researches of researchers and scholars in thriving fields like engineering, applied sciences etc. ıt has a wide variety of contents ranging from research articles, reviews and technical papers on the aspects including civil engineering , electrical engineering i.e computer science.

PJEAS uses a stringent peer-review process where all manuscripts submitted are evaluated by subject matter experts. This commitment to quality is made so that the journal can maintain its credibility and validity in research.

The journal follows a typical timetable of issues published over the year giving its readers a steady stream of insights and discoveries. It may be registered in leading databases like Scopus and Web of Science that help improve its recognition among the academic community.

PJEAS provides simple submission procedures for budding authors, specifying the guidelines on formatting and citation to make publication easy. The journal can also operate an open-access approach, which provides wider access to the research discoveries and encourages knowledge transfer.

To remain current in the most recent innovations of engineering and applied sciences, people are advised to log onto PJEAS's official site or search pertinent academic databases. The journal is instrumental for promoting research and shaping intellectual environment not only in Pakistan but also internationally.

Automatic Text Summarization

*Note: This is the research paper for automatic text summarization using flutter with tflite-model

1st Muzammil Sardar Abbasi
Sir Syed University Of Engineering And Technology
Karachi, Pakistan
se20-165@ssuet.edu.pk
3rd S.M Zejah Ali Rehmani
Sir Syed University Of Engineering And Technology
Karachi, Pakistan
se20-154@ssuet.edu.pk
5th Sonish Aslam(Supervisor)
Sir Syed University Of Engineering And Technology
Karachi, Pakistan
sonish@ssuet.edu.pk

2rd Imad Khan
Sir Syed University Of Engineering And Technology
Karachi, Pakistan
se20-176@ssuet.edu.pk
4th Muhammad Ahsan Siddique
Sir Syed University Of Engineering And Technology
Karachi, Pakistan
se20-218@ssuet.edu.pk

***Abstract*—This research paper explores the implementation of automatic text summarization using Flutter in conjunction with a TFLite (TensorFlow Lite) model. The study investigates the integration of Flutter, a popular UI toolkit, with TFLite, a lightweight machine learning framework, to develop an efficient and user-friendly text summarization application. The paper discusses the design, implementation, and evaluation of the proposed system, highlighting its performance and usability in generating concise summaries from diverse textual content. The findings contribute to the growing field of natural language processing and mobile application development, showcasing the potential of Flutter and TFLite for enhancing text summarization capabilities.**

***Index Terms*—Automatic Text Summarization, Flutter, TFLite, Machine Learning, Natural Language Processing.**

I. Introduction

You pick one at random to look inside–where there's ton after ton of treasure stores, but you still have no idea how much seek-and-find work it will take before that needle has been hit for tasty tidbits. This is the daily dilemma which we, as information seekers are faced with. But the crux of it is that existing automated text summarization tools are themselves limited in their capabilities. They say they will simplify this process, but often enough get it wrong. A lack of tailored attention to individual needs lies at the root of many problems in these procedures. In a sense, it's like having you miniscule personal assistant that can easily pull together the wisdom from mountains of information

Text Summarization As an application of fuzzy information management, this is a process in which the nuggets of truth are squeezed out from long-winded articles and news items.The whole point of this process is to take the content that used to be described in paragraphs and condense it into a form as close to a summary as possible, while not losing any vital information, and it should include following sections:

II. Overview

Life on that giant frontier of the digital world, where there are so many articles and news items to be absorbed in among all these blogs–it's tough. Just trying everyday not get washed away by a tidal wave crushing you to death is very hard work. Think of a library that's always open, full to the brim with books containing all kinds o f information. You could spend months or even years in wind through those shelves looking for . A fraction of what you need. That is the everyday dilemma facing us as information seekers.

It's all down to the limitations of current automated text summarization. They will tell you this is going to be smoother, but what they are missing is that personal touch which can adjust subtleties of individual needs. This is like having an assistant who understands your tastes, is aware of what interests you and can filter the essence from vast amounts of information.

Therefore, this is the response to that challenge– it's a humanizing of how we work with information. The core of the problem isn't just information overload but lack of a light leading people through this technological maze. It's about solving the frustration of information seekers who have lost their bearings lacking more than a sea-fullof words. Basically what our project is aiming to build up can be summarized as an automated text summary method which not only makes the process easier, but also more human in a way. It's about building an electronic wing man that understands the person behind the screen. Amidst a sea of information exploding upon our screens, it offers up something familiar and personal in this anything-but-intimate cyberspace world. Our goal is to change the digital journey from one of whimsical wandering to a guided trek full of places, visitors and itineraries tailored

for users.

III. Problem Statement

In a sea of information, with articles tripping over each other and playing hide-and seek on our screens, surfers are often left ogling the proverbial Viagra ad. It's commonplace to locate that elusive needle in the haystack. Today, especially, there is information overload. But the innovations that are obtainable today adhere to to save us, but they fail because we desire a personal touch that is aware of our preferences. Enter the issue we're eager to address: the inadequacies of modern artificial text describing methods. While the aforementioned instruments are a step along an appropriate orientation, they cease short of tailoring a plan of action. As an instance, you might use an automated summary tool. However, it fails to utterly follow your reasoning. It's equivalent to talking to an artificial intelligence which cannot recognize you.

But here's the rub–our problem statement is not simply about information overload more generally. This is all about tackling this problem in the context of current programs and applications. There are already plenty of text summarizing tools in the digital realm, but none possesses sufficient deftness to know just what users like.This is the point when our smartphone application kicks in, and it isn't the typical one. It is built on Flutter and has a user-friendly layer as well as an extensible aspect. Flutter provides a touch of enchantment to the user experience, making it softer and more engaging. The real secret sauce, however, is our use of the TFLite model. This is no longer just the art of condensing text; it's about doing so more intelligent, instinctively, and with a personal touch that avoids being swallowed up amidst all those applications.

Our app is not just a solution to the problem, but an antidote to inadequate existing tools. . . This is the missing component. It understands not solely whether to put things into perspective, but additionally what to do it in an efficient manner that is significant to consumers.
Essentially, it's more than about cruising the currents of the cyber ocean; there's a new customized methodology to storm information retrieval that Flutter and the TFLite model aspired to. The proverbial needle in the haystack Information overload is all too common. Even if such present instruments promise to save us from our deaths, they still lack that human touch, the capacity to comprehend any specific person's likes.

- Identify the challenge that we are seeking to resolve: the deficiencies of existing automatic text summarizing technologies.These are fine tools to get one started, but they fall short of a customized plan. For instance, you utilize an automated summary tool only to find that it got a few things wrong.
- But now comes the twist. However, our problem statement is not just one of general information overload. Programs and applications already exist to combat this problem. Already in the digital world are many tools promising to condense text.
- That's the trick–our problem statement isn not that information is overloading us in general. But it is the problem of how to solve this difficulty in current applications and systems. In the digital world of text summarization There are programs that have been developed, but they show a false sensitivity for reading by people.

At the other conjunction, our smartphone app performs as a supervisory; on the contrary hand, it is not your typical app. It is built on Flutter, an accessibility and interactivity layer. Flutter, on the contrary conjunction, adds a touch of freshness to it, rendering the user experience more enjoyable. The TFLite model, on the other hand, is our true secret sauce. But you don't simply have to paraphrase and speak a few words. You must act quickly, Response This text has no mistakes.But our program does more than solve a problem. It also spots flaws in existing tools. The key is that last one, which is bloody fragmented and can't even get what the significance of summary itself means in terms of connecting with consumers. In fact, we don't just surf the wave. We ride behind with our own tailor-made information retrieval strategy in tow and it all happens through Flutter and TFLite.

IV. project Architecture

With TFLite model integration and easily configurable pages, Flutter was just the right tool for our ambition to develop a great automatic text summarization app. Our TensorFlow Lite (TFLite) models have been integrated in our applications in order so we can implement revolutionary innovations behind the scenes. Consider that for instance that you wore an exceedingly intelligent personal assistant that can quickly scan understand and interpret difficult textual information. To ensure that users get an outstanding experience using this product is super easy yet thoughtfully designed–by focusing on several key features not mentioned above. Those include pre-configured templates (like scripts used in film work), topic classification plug:

A. Text Input:

Allow users to input text for Forget the tedious process of inputting articles or writings. One is allowed to do at will as one would share stories with a friend-just say it and go on without your memory playing tricks on you again. The way we've simplified it is that you tell us what content has got to be summarized and in which format, giving the earliest possible date.

B. TFLite Model Integration:

Integrate a trained TFLite model for text summarization. On the other hand, in order to leverage cutting-edge technology here behind closed doors is our app's incorporation of TensorFlow Lite (TFLite) models. It's like a conducting robot, or an assistant of the mind with as sharp a brain but capable of deciphering and processing complex textual information.Convert the model to a format compatible with Flutter (e.g., TensorFlow Lite Flutter Plugin).

C. Summarization Processing:

In this respect, our app is particularly well equipped. Advanced algorithms allow it to process and compress the information in input text into concise summaries that are easily understandable. This is the place when the enchantment happens. We evaluate and derive the article's highlights using this smartphone application employing modern facilities algorithms. It's akin to having a really competent editor condense a long piece of writing to its essentials. Sometimes, you can have a competent editor with all their experience as they shorten an article which is too long.

D. User Interface (UI):

Imagine a sleek, simple-to-operate interface that takes you through the process of summarizing without any fuss whatsoever so it is both an entertaining and efficient way to spend your time. Our app is easily navigated and coupled with a minimalist color scheme, it looks beautiful too. Imagine an easy and user-friendly interface that gently leads you step by step through the process of completing block form summaries, thus making as much fun to use in practice as it is on paper.

E. Firebase Integration:

In addition to handling user authentication, Firebase also becomes a serverless data storage solution for the generated summaries. For the pleasure of users and accessibility, we match Firebase into our app. Firebase doesn't just handle the user authentication, though. It also plays a key role as our serverless data storage solution for storing additional copies of your site content in this summary form, and makes sure they are only accessible if users can successfully pass an authorization check to get into PAILON– which means Assumi itself works hard to make sure that not everyone with internet service automatically. Thus all your users' summary queries receive secure and efficient accesses from their various local machines without causing you any extra inconvenience of building backend capability or high uncertainties brought by service providers: they are processed entirely on-the-fly on your cloud platform checks out some

F. User Authentication:

We take concerned about your security and privacy and will continue to be concerned about them. While Firebase handles user authentication, you get to choose who can view summaries created by you but this should not deter you from maintaining your personal information. It's like as if your virtual world had a trusted bouncer on the door.

G. Cloud Functions:

Nonetheless, Cloud Functions are the ones that give the app its responsiveness and function. In this case, think of it as having a backstage organization up front where all steps right from user input to processing for summarization become smooth and efficient.

H. Data Storage:

Apart from looking after authentication, Firebase is there for safe server-less data storage. After they have been summarized, your documents are stored neatly for easy retrieval when required.

I. Security:

We keep your data is safe in our app. Our application has strong security so only authorized people can access the summarized documents hence they are protected by our system. Thus, it is more like an electronic fort guarding your vital information.

V. Project Scope

The cause of our automatic text summarization (ATS) application is to help produce the precis summary of any article containing any sort of crucial and applicable information from the unique documented textual content. Summarization is the venture of condensing a piece of file into its shorter variant, which lessens the initial record's size even as retaining the important piece of information intact.

A. System Features:

Our automatic text summarization application makes use of Flutter, then integrates with a TFLite model to remap the terrain by means of system features. Fundamentally, it is a user-friendly interface that goes beyond the 3Cs of textual content. Our magic wand Flutter allows us to provide users with an interactive browsing experience on both iOS and Android, eliminating the need for two sets of programs. The language of technology is universal after all.

Adding a TFLite model puts forward the technology equivalent to having an assistance who can scan, analyze and read large quantities of information before boiling down what it has seen into practical summaries. This intelligent summary linkage process, conforming to individual preferences ensures that the resulting summaries are not only informative but also easy on the eye. The TFLite model is like the wizard behind that curtain, making something as difficult to deal with as text into a piece of convenient software. Firebase integration allows us to improve the security and convenience of our application. It also offers a secure server for authenticating the user and storing its data, making it into something like a private safety deposit box. With this kind of cloud-based storage, they can access their summarized takenaways at any time.

B. Objectives

The objective of this research project is to develop an automatic text summarization system using Flutter, TensorFlow Lite (TFLite) models, and Firebase integration. This involves leveraging the Flutter framework to create an intuitive user interface for input and output, implementing TFLite models to perform the text summarization task, and utilizing Firebase for data storage and retrieval. The detailed objectives include:

C. User-friendly Interface

Designing a user-friendly interface using Flutter that allows users to input and receive summarized text seamlessly.

D. Integration of TFLite Model

Implementing and fine-tuning a TFLite model specifically trained for text summarization. This involves integrating the model into the Flutter application to process input text and generate concise summaries.

E. Firebase Integration

Incorporating Firebase for efficient data storage, ensuring the seamless retrieval and storage of input texts, summarized content, and other relevant data.

F. Real-time Summarization

Ensuring real-time text summarization capabilities by optimizing the communication between Flutter, the TFLite model, and Firebase.

G. Solubility and Performance

Evaluating the system's scalability and performance to handle varying workloads, multiple users, and diverse textual inputs.

H. User Authentication and Authorization

Implementing Firebase authentication and authorization features to ensure secure access to the summarization system.

I. Cloud-based Text Storage

Utilizing Firebase Cloud Firestore or Realtime Database for storing and managing textual data efficiently.

J. Cross-platform Compatibility

Ensuring cross-platform compatibility of the Flutter application, allowing users to access the summarization system seamlessly on different devices.

K. Documentation and Deployment

Providing comprehensive documentation for developers and deploying the application to app stores, making it accessible to a wider audience.

L. Scope of the Research

Flutter and TFLite are very popular right now, so work is also being done with them in an attempt to analyze English language text. Though the model's architecture is flexible and can be adapted to other domains, the initial focus will remain on general-purpose summarization. The underlying scheme is therefore designed to be platform-neutral, in order to supporting future mobile, website and even desktop applications.

VI. Background

- Flutter: Flutter is an open-source UI toolkit developed by Google to build mobile, web and desktop applications. The application code written once can then be used on all three of these platforms with their respective native compilers. Hot reloading makes it easy to rapidly develop, experiment and test.
- TFLite: Developed for running machine learning inference on mobile and edge devices, TensorFlow Lite is a lightweight, efficient version of the framework. It also supports on-device processing and helps the user to preserve privacy, eliminating dependence on cloud services.

VII. Methodology

Our approach involves the following key steps:

A. Data Collection

- Gathered a diverse dataset of textual content from various sources, including articles, blogs, and news.
- Ensured the dataset covers a range of topics and writing styles to enhance the model's adaptability.

B. Data Preprocessing

- Cleaned and preprocessed the text data to remove irrelevant characters, stopwords, and ensure uniform formatting.
- Tokenized the text into meaningful units to facilitate model comprehension.

C. Flutter App Development

- Utilized the Flutter framework to create a user-friendly mobile application.
- Integrated the trained TFLite model into the Flutter app, allowing seamless interaction between the user interface and the summarization engine.

D. Firebase Integration

- Integrated Firebase for efficient data management and storage.
- Utilized Firebase Cloud Firestore or Realtime Database to store input texts, generated summaries, and user-specific information.

E. TFLITE Model Training

- We used TensorFlow to train a deep learning model, which we then compiled into the more compact TFLite format for faster on-device delivery.
- The model is meant to comprehend the context and significance of sentences in any piece.

F. Integration with Flutter

- A trained TFLite model is incorporated into a Flutter app, with users entering text for the application to process according to theattendant logic of an artificial intelligence whose memory has been compressed within it.
- This integration provides for a unified environment, as well as concurrent summarization by the user.

VIII. Feature Enhancements

A. Multi-Language Support

1) Description: Enabling the system to support multiple languages broadens its accessibility and usefulness to a more diverse user base.

2) Implementation:

- Language Detection: Implement language detection mechanisms to identify the language of input text.
- Language-Specific Summarization Models: Integrate language-specific summarization models to ensure accurate and context-aware summaries.

3) Impact:

- Improved Accessibility: Users across different language backgrounds can utilize the summarization system effectively.
- Expanded User Base: Attracts a wider audience with diverse linguistic preferences and requirements.

B. Integration with External Summarization APIs

1) Description: Collaborating with external summarization APIs allows the system to leverage different summarization techniques and enhance the quality of summaries.

2) Implementation:

- API Integration: Research and integrate reputable external summarization APIs.
- User-Selectable Options: Provide users with the option to choose from different summarization engines.

3) Impact:

- Varied Summarization Techniques: Users can benefit from a range of summarization approaches, each with its strengths and use cases.
- Customization Options: Users gain control over the summarization method based on their preferences and needs.
- Prioritization Insights: Helps identify popular feature requests or critical issues through user voting patterns.
- Community Engagement: Encourages users to actively participate in the improvement of the system.

C. Feedback Analysis and Action

1) Description: Establish a systematic process for analyzing user feedback and taking actionable steps based on the insights gained.

2) Implementation:

- Feedback Categorization: Categorize feedback into different themes such as usability, performance, or feature requests.
- Regular Review Meetings: Conduct regular review meetings to discuss user feedback and prioritize action items.
- Transparent Communication: Communicate transparently with users about how their feedback is being utilized.

3) Impact:

- User-Centric Development: Ensures that development efforts align with user needs and expectations.
- User Trust: Builds trust by demonstrating responsiveness to user input.

IX. Future Directions

- Our success on smartphones running the Android and iOS operating systems provides channels for future development, including a search for opportunities to add support of more languages, domain-dependent fine tuning, or integrations with new technologies. To conclude, our experiments verify the feasibility of applying Flutter and TFLite to automatic text summarization on Android and iOS phones. This combination of a user-friendly interface with on device machine learning makes for powerful, easy and rapidly accessible tool to filter out the core content from mobile textual subject matter.

- Flutter and TFLite have been implemented to automatically summarize text. These efforts are all fraught with difficulties, but they also offer opportunities for interesting follow-ups in the future. Another significant challenge faced as a result of the integration process has been variations in ability levels among Android and iOS smartphones. Programs like these had to be optimized for the requirements and limitations of each individual device, demanding rigorous work. And further perfection is needed in order smoothly operate even a wider range of mobile devices.
- Language is constantly changing, content too diverse The bottom line Another difficulty involves the diversity of language and geography. So our model has mastered the challenge of short-language (English) summarization. But how does one go about making it able to help with multiple languages? Further work will look at expanding the scope of language capability, taking account of variant linguistic forms and nuances.
- In addition, domain- particular optimizations can be an approach to improvement. While this model works well when it comes to general-purpose summarization, adapting the system for specialized purposes in specific domains (like law or medicine) requires that you understand how words are used and which words being most applicable. Further work will examine how to customize the model for various fields, so as to create better contextualized and more relevant summations.

- When you look forward, combining new technologies is an ideal that can be pursued for future work. Studying how to incorporate the latest developments in natural language processing, sentiment analysis, and multi-source information fusion (e.g., including support from pictures or audio) within summary generation could provide more comprehensive extraction of information.
- Finally, getting beyond the current limits on device compatibility and language support issues, it seems that auto-summarizing text making use of Flutter UIs and TFLite will be a bright future. Overcoming these hurdles and exploring further improvements will help to ensure that the resulting product is better tailored, even more powerful, and able for users on a variety of mobile devices to extract essential information across languages or domains.

ACKNOWLEDGMENT

We would like to express our gratitude to the open-source communities of Flutter and TensorFlow for providing the tools and resources that made this research possible.

REFERENCES

[1] Luhn, H. P. (1958). The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development, 2*(2), 159-165.

[2] Filippova, K., & Strube, M. (2008). Sentence fusion via dependency graph compression. *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, 648-656.

[3] Edmundson, H. P. (1969). New Methods in Automatic Extracting. *Journal of the ACM, 16*(2), 264-285.

[4] Nenkova, A., And McKeown, K. (2011). Automatic summarization. *Foundations and Trends® in Information Retrieval, 5*(2–3), 103–233

[5] Erkan, G., And Radev, D. R. (2004). LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research, 22*, 457-479.

[6] Radev, D., Jing, H., Styś, M., And Tam, D. (2004). Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. *Artificial Intelligence, 152*(2), 165-195.

[7] Barrios, F., Bendersky, M., And Freitag, D. (2016). Variations of the Similarity Function of TextRank for Automated Summarization.

Appendix D- Marketing / Promotional Material

8.4 POSTER

Figure 8.1 shows the poster of the project.

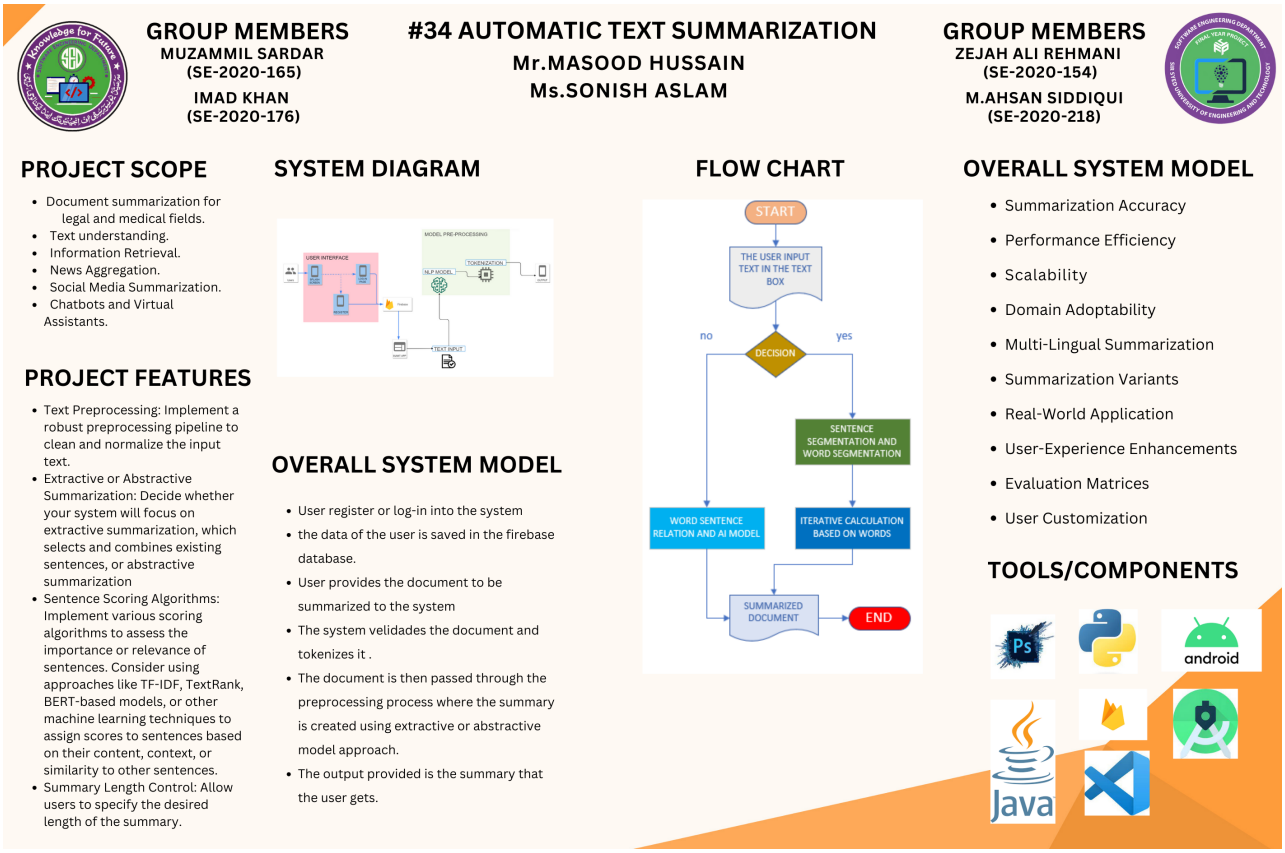


Figure 8.1: POSTER OF AUTOMATIC TEXT SUMMERIZATION

8.5 STANDEE

Figure 8.2 shows the standee of the project.

Figure 8.2: STANDEE FOR AUTOMATIC TEXT SUMMERIZATION

8.6 BROCHURE

Figure 8.3 and Figure 8.4 show the brochure for the project.

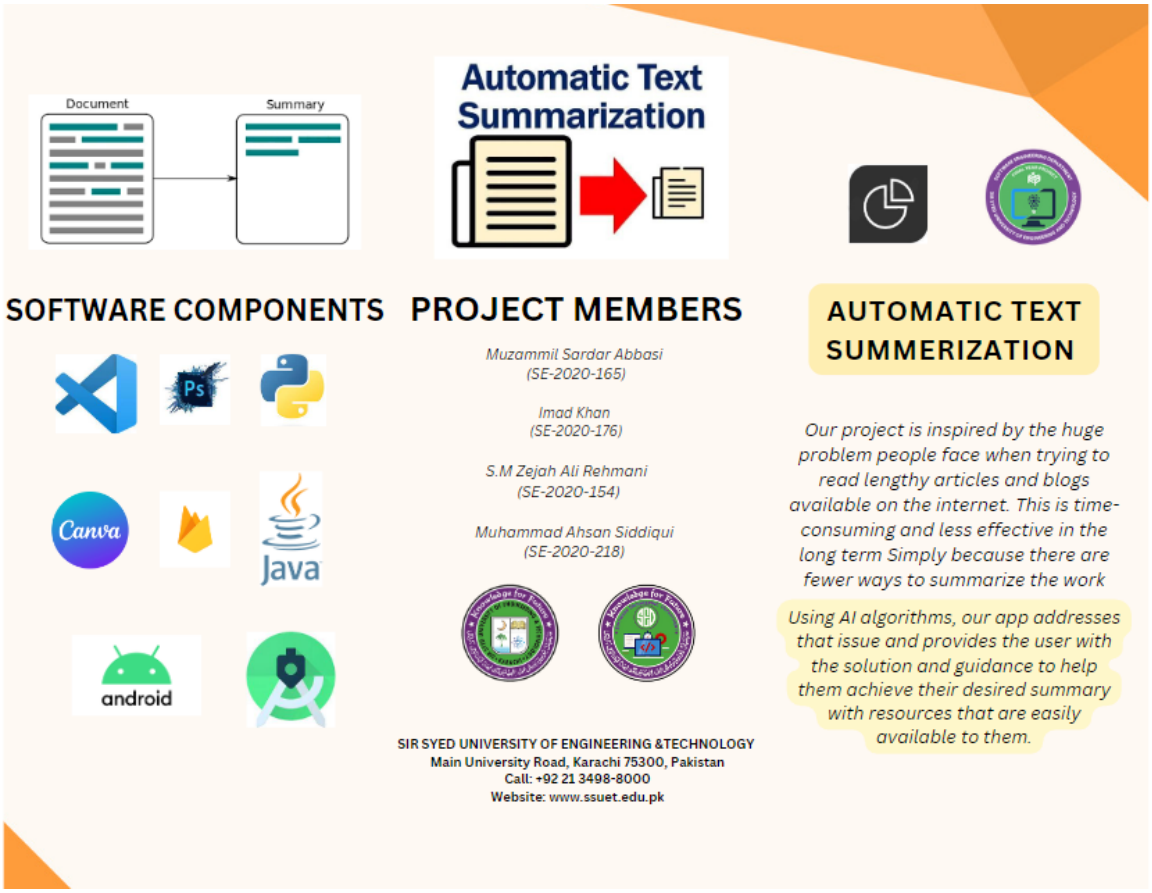


Figure 8.3: BROCHURE FRONT-SIDE FOR AUTOMATIC TEXT SUMMERIZATION

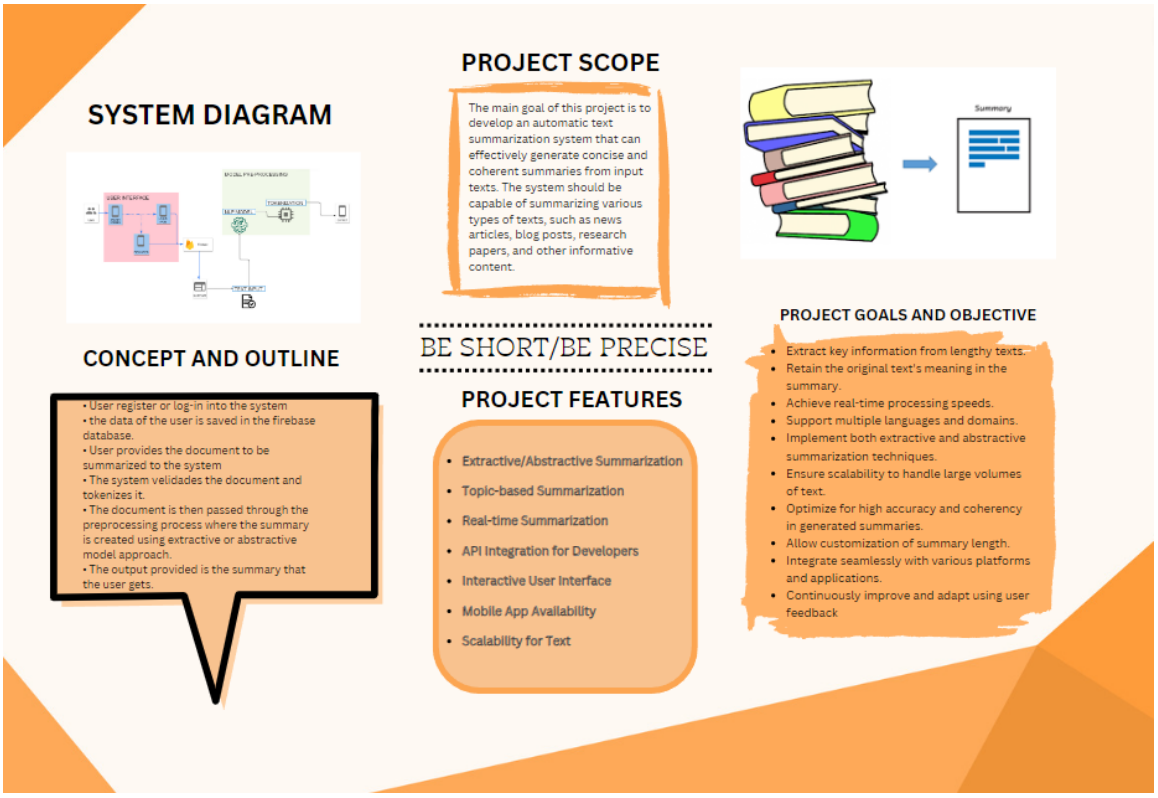


Figure 8.4: BROCHURE BACK-SIDE FOR AUTOMATIC TEXT SUMMERIZATION

References

[1] Luhn, H. P. (1958). The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development, 2*(2), 159-165.

[2] Edmundson, H. P. (1969). New Methods in Automatic Extracting. *Journal of the ACM, 16*(2), 264-285.

[3] Hovy, E., & Lin, C. Y. (1997). Automated Text Summarization in SUMMARIST. *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, 82-88.

[4] Radev, D., Jing, H., Styś, M., & Tam, D. (2004). Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. *Artificial Intelligence, 152*(2), 165-195. doi: 10.1016/j.artint.2003.12.002

[5] Erkan, G., & Radev, D. R. (2004). LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research, 22*, 457-479. doi: 10.1613/jair.1523

[6] Nenkova, A., & McKeown, K. (2011). Automatic summarization. *Foundations and Trends® in Information Retrieval, 5*(2–3), 103–233. doi: 10.1561/1500000026

[7] Conroy, J. M., Schlesinger, J. D., & O'leary, D. P. (2011). Topic-focused multi-document summarization using an approximate oracle score. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 464-473.

[8] Filippova, K., & Strube, M. (2008). Sentence fusion via dependency graph compression. *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, 648-656.

[9] Barrios, F., Bendersky, M., & Freitag, D. (2016). Variations of the Similarity Function of TextRank for Automated Summarization. *CoRR, abs/1602.03606.* arXiv Link

[10] See, A., Liu, P. J., & Manning, C. D. (2017). Get To The Point: Summarization with Pointer-Generator Networks. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1073-1083. doi: 10.18653/v1/P17-1099