Thangal Kunju Musaliar College of Engineering

Department of Computer Science and Engineering, Kollam Kerala
June 2023

ECHO: VOICE BASED EMAIL ASSISTANT FOR THE VISUALLY IMPAIRED

MINI PROJECT REPORT

Submitted By

ASHWIN BINU ABRAHAM (TKM20CS031)
DEEPAK P (TKM20CS041)
FATHIMA NOOHA KOTTANGODAN (TKM20CS052)
SALIHU AHAMED (TKM20CS116)

to

APJ Abdul Kalam Technological University
in partial fulfilment of the requirements for the award of B.Tech Degree
in Computer Science and Engineering

DECLARATION

We undersigned hereby declare that the project report on “Echo: Voice based email application for visually impaired”, submitted as part of our curriculum, Mini Project under APJ Abdul Kalam University, Kerala is a bonafide work done by us under supervision of Dr. Dimple A Shajahan, Project Coordinator and Head of the Department of Computer Science and Engineering, Prof. Jesna JS, Prof. Reena Mary George, and Prof. Nisa AK, Assistant Professor, Department of Computer Science and Engineering, TKMCE.

This submission represents our ideas in our own words and from other sources that have been adequately and accurately cited and referenced. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission.

We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Kollam
Date: 22/07/2022

Mr. Ashwin Binu Abraham
Mr. Deepak P
Ms. Fathima Nooha Kottangodan
Mr. Salihu Ahamed

Thangal Kunju Musaliar College of Engineering

Department of Computer Science and Engineering, Kollam Kerala
June 2023

CERTIFICATE

This is to certify that the report titled "**Echo: Voice based Email assistant for the blind**" submitted by **Ashwin Binu Abraham**, **TKM20CS031**; **Deepak P**, **TKM20CS041**; **Fathima Nooha Kottangodan**, **TKM20CS052**; **Salihu Ahamed**, **TKM20CS116** to the APJ Abdul Kalam Technological University in completion of the requirements for the award of Bachelor of Technology Degree in Computer Science and Engineering during 2022 – 2023 is a bonafide record of the **Mini Project** work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Project Coordinator

Head of the Department

Examiner

ACKNOWLEDGEMENT

We take this opportunity to express our deep sense of gratitude to the Almighty and sincere thanks to all who helped me to complete the project successfully.

We express our sincere gratitude to **Dr. T. A. Shahul Hameed**, Principal, TKM College of Engineering, for providing us with all the necessary facilities and support for doing the project.

We are extremely grateful to **Dr. Dimple A Shajahan**, Project Coordinator and Head of the Department of Computer Science and Engineering, along with **Prof. Jesna JS**, Assistant Professor, Department of Computer Science and Engineering, **Prof. Reena Mary George**, Assistant Professor, Department of Computer Science and Engineering, and **Prof. Nisa AK**, Assistant Professor, Department of Computer Science and Engineering, for their constructive guidance, advice, constant support and technical guidance provided throughout the making of this project. Without their intellectual support and apt suggestions at the perfect time, this project work would not be possible.

We extend our immense gratitude to all Faculties and Technical Staffs in the Department of Computer Science and Engineering, for their help and necessary facilities to complete the project. Our humble gratitude and heartiest thanks also go to our parents and friends, who have supported and helped me on the course of this work.

Mr. Ashwin Binu Abraham

Mr. Deepak P

Ms. Fathima Nooha Kottangodan

Mr. Salihu Ahamed

ABSTRACT

Echo is a voice-based email application developed using Flutter, designed to enhance email accessibility for visually impaired individuals. The app targets the global community of approximately 285 million visually impaired people, aiming to provide them with more accessible communication technology. Echo features a user-friendly voice-based interface, allowing users to perform essential email tasks using natural language commands. The application includes core email functionalities such as composing, reading, and replying to emails, along with advanced features like email filtering, sorting, voice-based reminders, and notification alerts. By leveraging natural language processing and speech-to-text technology, Echo delivers a seamless and inclusive user experience. Integrating with popular email providers, the app empowers visually impaired users to independently manage their emails, fostering social inclusion and improving employment prospects. The development process follows the agile methodology, emphasizing rigorous testing to ensure compliance with desired specifications before deployment on suitable platforms.

The Echo app is designed to work on Android smartphones with an API level above 21. It requires an internet connection either through Wi-Fi or mobile data for optimal performance. The device should also have a good-quality microphone, speaker, and camera to support the app's functionalities. The app is built using the Flutter framework for the front end and Django for the back end. Several Flutter plugins are utilized to implement the various features of the app.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abbreviations

Abbreviations	Full Form
STT	Speech-to-Text
TTS	Text-to-Speech
API	Application Programming Interface
UI/UX	User Interface and User Experience

1. Introduction

In the rapidly evolving world of mobile applications, creating solutions that are inclusive and accessible to all users is of paramount importance. The Voice-Assisted Email Client App represents an innovative project specifically tailored to empower visually impaired and blind individuals in their digital communication journey. By using the power of voice commands, this app revolutionizes how users with visual impairments interact with their email accounts.

1.1 Motivation

The inspiration behind our project, a voice-assisted email client app built on the Flutter platform, is deeply rooted in the mission to enhance the lives of visually impaired and blind individuals in the realm of digital communication. We firmly believe that technology should be an equalizer, breaking barriers and providing accessible solutions for all users, regardless of their abilities. For visually impaired individuals, traditional email interfaces often pose significant obstacles, impeding their full participation in the digital age.

Through this project, we are motivated to revolutionize the way visually impaired users interact with email by leveraging the power of voice commands. By developing a user-friendly and intuitive mobile application, we aim to empower our target users with the ability to effortlessly manage their emails, access important information, and engage in meaningful communication. The overarching goal is to enable independent and seamless email interactions, enhancing their productivity, social connections, and overall quality of life.

Our team is driven by the potential to make a tangible impact on the lives of visually impaired individuals, transforming their daily digital experiences from frustration to empowerment. By embracing inclusivity and embracing cutting-edge technology, we envision a future where everyone can harness the benefits of email communication, irrespective of visual impairments. With utmost dedication and a passion for creating positive change, we embark on this journey to contribute to a more accessible and inclusive world, one voice command at a time.

1.2 Objectives

- Develop a highly accessible and user-friendly mobile application to cater specifically to visually impaired and blind users, enhancing their email communication experience with seamless voice-assisted interactions.

- Implement a state-of-the-art voice recognition system with exceptional accuracy, capable of comprehending diverse natural language voice commands to perform various email actions effectively.

- Enable visually impaired users to efficiently manage their emails through intuitive voice commands, encompassing tasks such as sending, receiving, reading aloud, composing, and organizing messages.

- Integrate a comprehensive speech feedback mechanism that provides clear and informative auditory cues, guiding users throughout email interactions and ensuring smooth navigation.

- Conduct rigorous user testing and engagement with visually impaired individuals to gather invaluable feedback, enabling iterative refinements to optimize the app's usability, accessibility, and overall user experience.

- Prioritize data security and user privacy by implementing robust encryption and authentication protocols, ensuring the utmost protection of sensitive email information.

- Create comprehensive documentation and user support resources to empower visually impaired users in utilizing the app independently and fostering a sense of confidence and self-sufficiency.

- Foster inclusivity and advocate for digital accessibility by adhering to industry best practices and guidelines, setting an example for the development of future accessible applications.

- Establish a scalable infrastructure and consider potential future enhancements to expand the app's capabilities and cater to evolving needs and advancements in voice technology.

1.3 Organization of the report

The report is structured into two main sections. The first section covers the development process and key features of the Echo, the voice-based email application, with a focus on improving email accessibility for visually impaired individuals. The subsequent sections include a comprehensive literature review, analyzing existing research and technologies related to email accessibility for the visually impaired, while also addressing the challenges faced and the approaches taken by previous solutions to mitigate these obstacles. The methodology section details the development approach, incorporating agile methodology and the integration of text-to-speech and speech-to-text technology, emphasizing the significance of continuous testing and feedback during the process. Subsequently, the design and implementation section delves into the app's user interface, core email functionalities, additional features, and seamless integration with popular email providers, while prioritizing user data privacy and security. The testing and evaluation segment includes rigorous developer-side testing to identify and address issues related to performance, functionality, and accessibility, ensuring a smooth and effective user experience for visually impaired individuals.

It concludes by summarizing the project's findings, underscoring the importance of accessible communication technologies, and suggesting future enhancements for the app. The report concludes with a list of references and appendices containing supplementary information, providing a comprehensive overview of the Echo project and its contributions to improving email accessibility for the visually impaired.

2. Literature review

Email accessibility for visually impaired individuals has been a subject of growing interest, driven by the need to provide inclusive communication technologies for this section of the population. Several studies have explored the challenges faced by visually impaired users when accessing and managing emails, highlighting the importance of developing accessible email applications like Echo.

Wenqing Zhao, et al.'s work on "A Survey of Automatic Speech Emotion Recognition: Tasks, Methods, and Datasets" provides valuable insights into automatic speech emotion recognition, which holds significance in enhancing email accessibility for visually impaired users, as voice-based interactions play a crucial role in their communication. Additionally, K. Sreenivasa Rao and K. Satya Prasad's "Text-to-Speech Synthesis: A Review" offers pertinent information about text-to-speech synthesis, a technology essential in converting written emails into audible formats to ensure effective communication for visually impaired individuals.

Several studies have investigated the challenges faced by visually impaired users while accessing and managing emails. Alapetite, A., Vermeulen, J., & de Graaf, M. (2015) explored email access through voice for visually impaired and illiterate individuals, shedding light on the difficulties faced by this user group. Such research underscores the need for email solutions that incorporate natural language processing and voice-based interactions, exemplified by Yousif, S., Elyas, E., & Salih's (2019) work on a voice-based email system using natural language processing. This system enables visually impaired users to send and receive messages through voice commands without relying on traditional visual interfaces.

In the process of selecting an appropriate framework for developing an accessible email application, "Flutter: A Portable Framework for Building High-Performance Mobile Apps" by Eric Seidel, et al., and Singh, A., Mishra, S., & Goyal, A.'s (2021) study on "Flutter for Cross-platform Mobile Application Development" played a critical role in choosing Flutter as the framework for the Echo application.

While some existing email apps like Easy Voice Mail, Gmail Voice, HeyMail, TalkMail, Hound, and Voiceitt have made progress in addressing email accessibility for the visually impaired, they still have limitations that highlight the necessity for a cross-platform solution like Echo. These existing apps often suffer from platform dependency, which restricts users to specific operating systems, limiting device usage flexibility. Additionally, they might lack essential features and customization options crucial for catering to the diverse preferences and needs of visually impaired users. The learning curve associated with certain apps can be steep for new users, hindering their adoption. Data migration between different apps can also be challenging.

A cross-platform solution like Echo would mitigate these issues by offering a consistent and intuitive user interface, simplifying the learning process, and ensuring seamless data synchronization across devices. Moreover, Echo could prioritize compatibility with a broader range of assistive technologies and adhere strictly to international accessibility standards like WCAG, thus overcoming the limitations of existing apps and fostering a unified user experience. Such an approach has the potential to empower visually impaired individuals, enabling them to independently and efficiently manage their emails across various platforms while benefiting from a more engaged and supportive community.

Despite the progress made in email accessibility research, there are still some research gaps that need to be addressed. Limited studies have explored the impact of email accessibility on social inclusion and

employment opportunities for visually impaired individuals. The present project, with its focus on the potential social and economic benefits of the Echo app, aligns with these underexplored areas and offers the potential to make a positive impact on the lives of visually impaired individuals.

3. Requirement Analysis

3.1 Stakeholders

The 'Echo' project relies on the collective efforts and support of various stakeholders who have played vital roles in its development and impact. Our project team worked collaboratively to design and implement the 'Echo' app. Under the guidance of our project supervisor, their expertise and feedback shaped the app's direction. Our fellow students, friends, and other college community members served as end users, actively testing and providing valuable feedback during the development process. We appreciate the interest shown by the computer science department in recognizing the potential impact of the project on society. Peers, classmates, and some alumni contributed to the project by offering insights and suggestions. In projects with community outreach, local community members provided valuable perspectives.

3.2 Requirement Elicitation

To capture comprehensive requirements, we employed a range of effective elicitation techniques. Through some online surveys distributed to a wider audience, enabling us to gather both quantitative and qualitative feedback on the app's potential features and usability. Interactive brainstorming sessions fostered creativity and generated innovative ideas for the app. Prototyping allowed us to gather early feedback and visualize the app's potential interface. Feedback sessions further helped refine prototypes and requirements based on stakeholders' inputs.

As a result of the requirement elicitation phase, we identified several crucial requirements for the "Echo" app. Ensuring accessibility was a top priority, and the app must feature a voice-based interface and functionalities tailored to meet the needs of visually impaired users. Core email functionalities, such as composing, reading, and replying to emails using speech-to-text and text-to-speech technologies, were also fundamental requirements. In addition, we aimed to enhance the user experience by incorporating features such as email filtering, sorting, voice-based reminders, and notification alerts. The app's seamless integration with popular email service providers was deemed essential to allow users to access their existing accounts without difficulty. To ensure a stable and reliable app, rigorous testing and bug-fixing procedures were included in the requirements. Furthermore, adopting an agile development approach would facilitate iterative development and adaptability to evolving requirements.

3.3 Requirement Specification

The requirement specification phase outlines the functional and nonfunctional requirements essential for the successful development of the "Echo" app. This section identifies and describes the functional requirements, which specify the features and capabilities of the app, as well as the nonfunctional requirements, which address aspects such as performance, safety, and security.

Functional Requirements

- The app should be compatible with Gmail and Outlook email providers.
- The app shall provide users with the option to set up their email accounts using voice commands.
- The app shall authenticate user credentials for the email provider through the user's voice command.
- The application should have a voice recognition system capable of transcribing spoken text accurately
- The application should be able to format the transcribed text into a readable email format
- The application should allow users to edit the transcribed text if necessary
- The application should provide users with a confirmation option before sending the email
- The system must be able to detect new incoming emails in the user's email account and extract the text content of the email.
- The system must have a built-in text-to-speech engine capable of converting the email's text content into an audio format.
- The user must be able to control the speed and volume of the speech output through the system's user interface.
- The system must provide a clear and natural-sounding audio output that is easily understandable by the user.
- The system should have the ability to recognize voice commands for editing emails.
- The system should have the ability to save drafts of emails in progress.
- The system should provide error messages if the user tries to send an email without a recipient or with an invalid email address.

Nonfunctional Requirements

- Performance Requirements
    - System should respond to user input within 2 seconds
    - System should load images and data within 5 seconds
    - System should handle 100 simultaneous users without performance degradation

- Safety Requirements
    - System should not allow users to upload or access malicious files
    - System should encrypt all user data in transit and at rest
    - System should have a failsafe mechanism to prevent catastrophic failures

- Security Requirements
    - System should have multi-factor authentication for user accounts
    - System should have role-based access control to restrict user access to certain features
    - System should be compliant with GDPR and other privacy regulations

3.4 Verification

The SRS (Software Requirement Specification) Document was documented and verified. Several updates were made to SRS based on decisions made by the project team in agile scrum meetings.

4. Proposed System Design

4.1 SDLC

The Software Development Life Cycle (SDLC) for the Voice-Assisted Email Client App for Visually Impaired Users adopts an agile-waterfall hybrid model. The waterfall methodology follows a linear and sequential approach, dividing project activities into distinct phases that rely on the outputs of the previous phase. On the other hand, agile project management emphasizes continuous improvement throughout the project's life cycle, allowing for flexibility and responsiveness to changes. In the case of ScribbleInspect, a hybrid project management approach was implemented. This hybrid model integrates agile methodologies within a larger waterfall structure, providing the benefits of both approaches. It enables quick innovation and adaptability in certain areas, while maintaining a structured and predictable process for critical aspects.

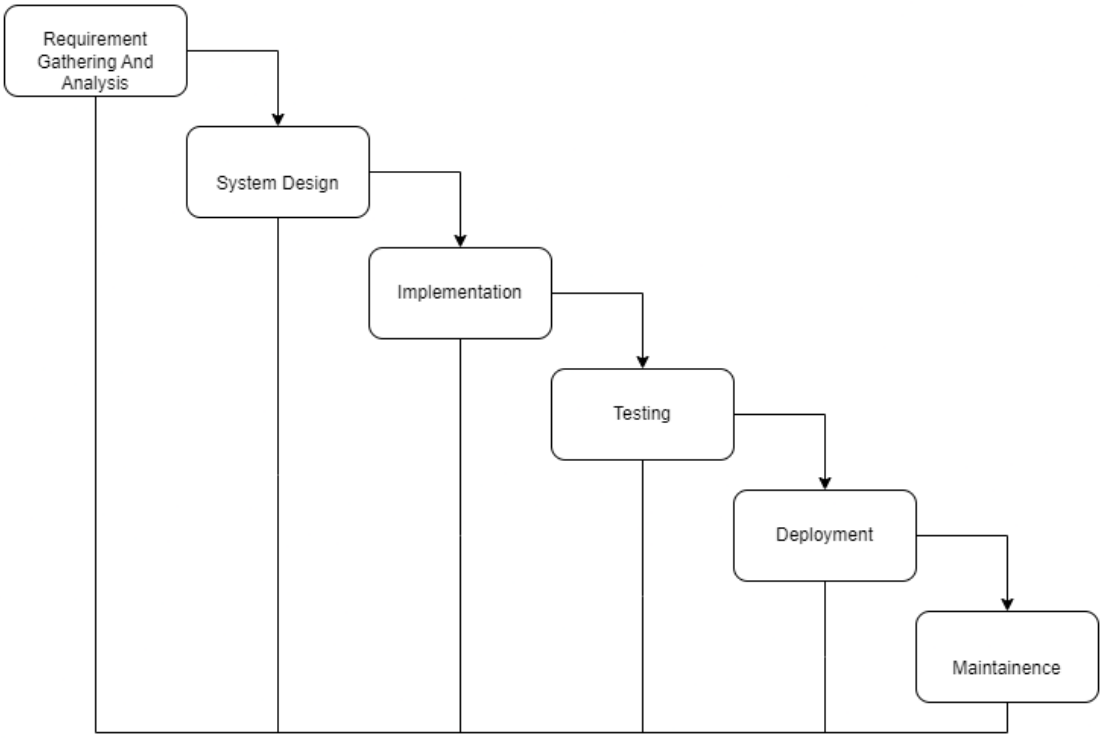


Figure 1.0 SDLC

4.2 Design Details Summary

System Architecture Design

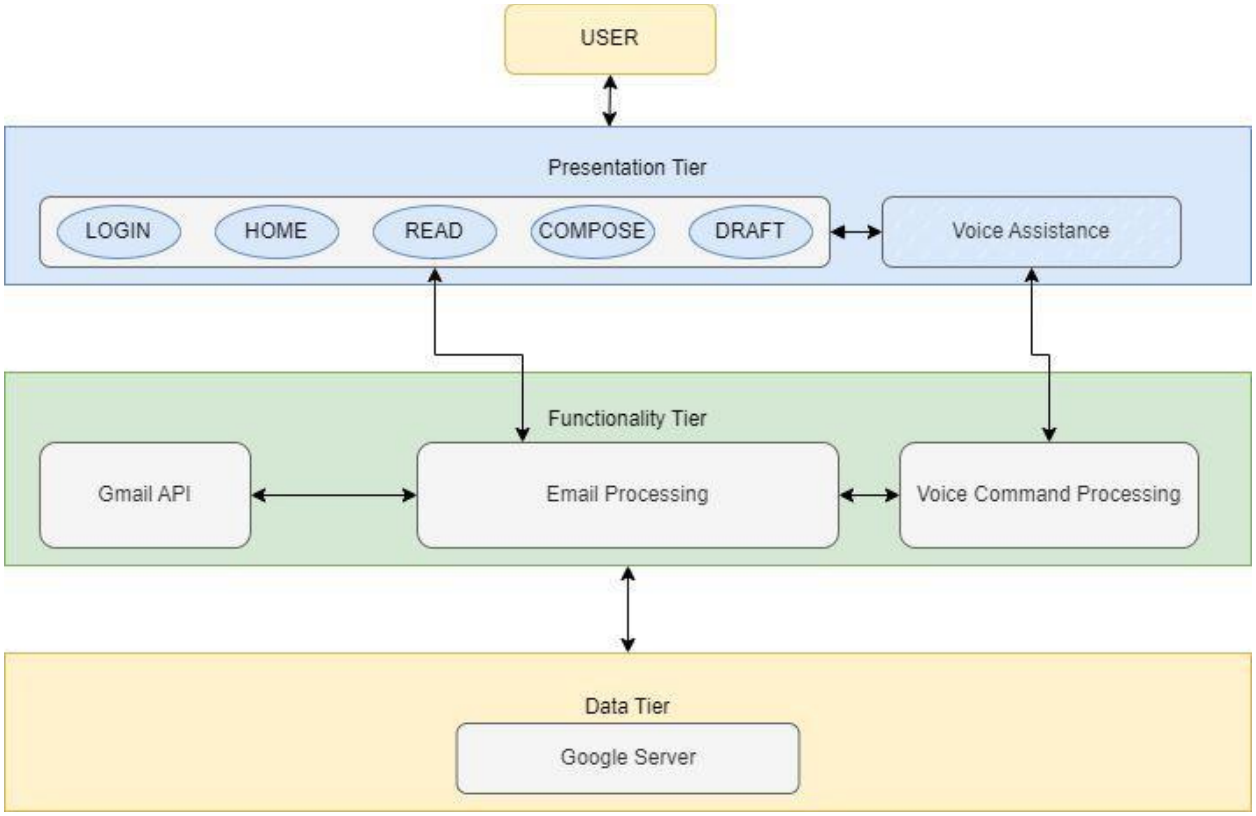


Figure 2.0 : System Architecture

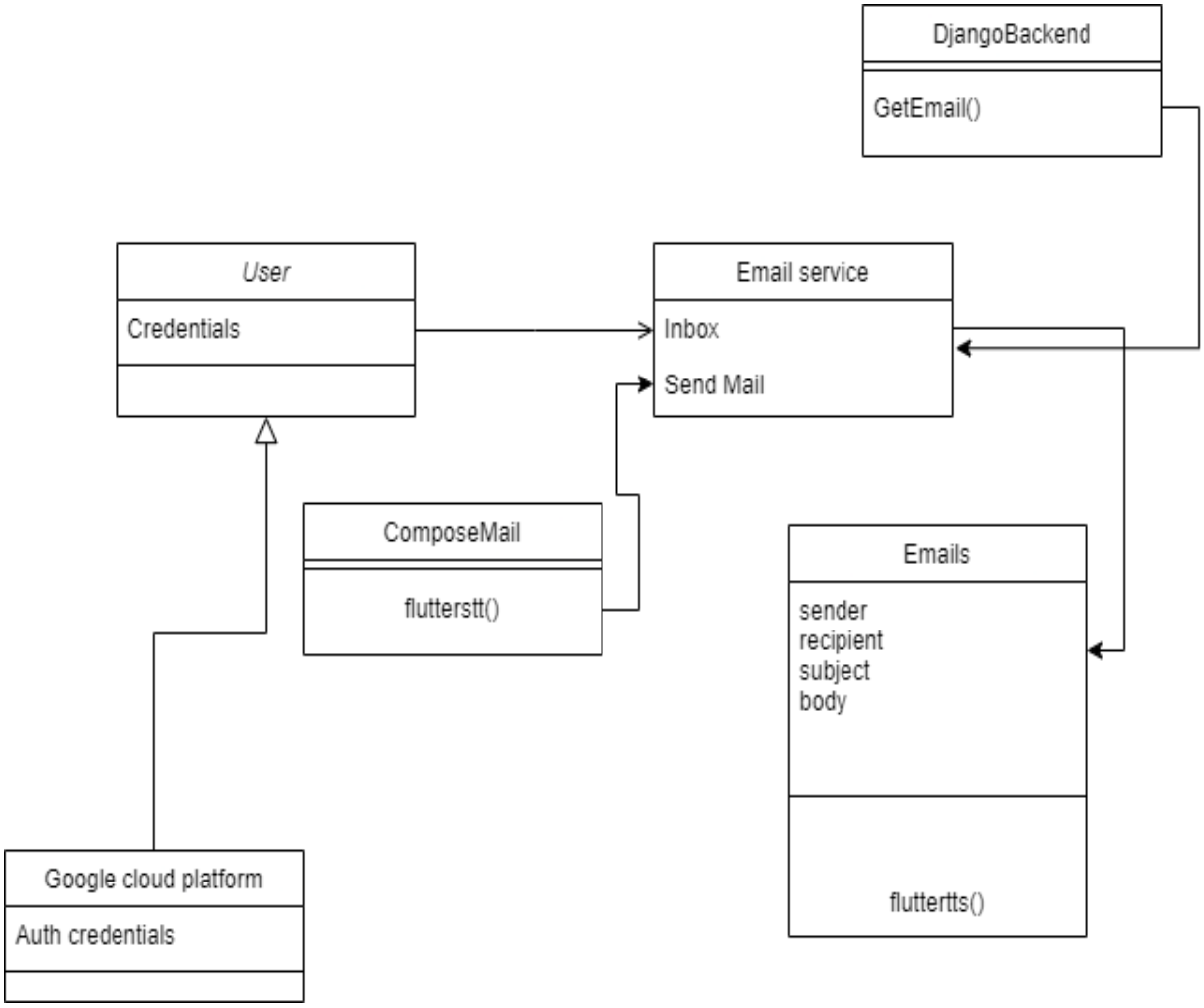
The architecture consists of three main tiers: the Presentation Tier, the Functionality Tier, and the Data Storage Tier. The Presentation Tier is responsible for the user interface and screens, which are developed using Flutter, and the voice command plugin.

The Functionality Tier is responsible for the email service, which handles communication with the Gmail API to perform email operations such as sending, receiving, and reading emails, as well as the email processor, which processes received emails. The Functionality Tier also includes the Flutter plugin for enabling voice commands.

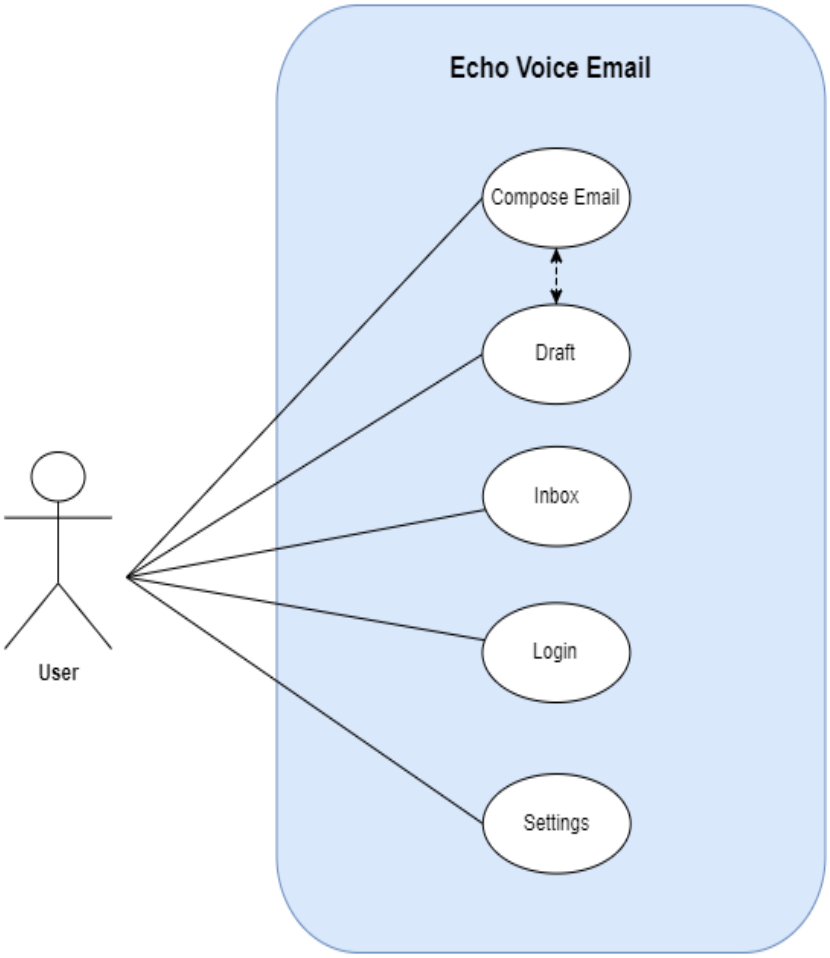
The Data Storage Tier includes a Google server. Mails are retrieved from the Google server. These three tiers work together to provide the complete functionality of the system, allowing visually impaired people to perform basic email operations through voice commands using the mobile app

1.1 UML Diagrams

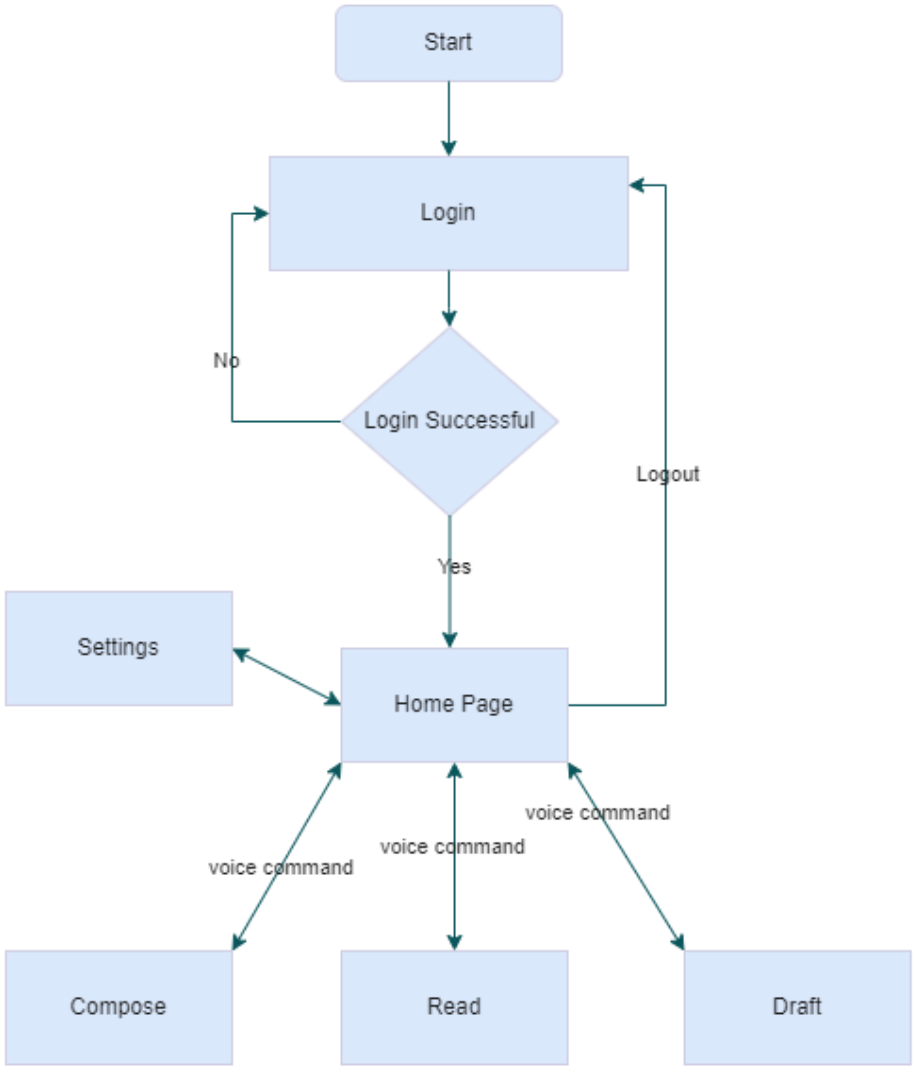
4.2.1 Class Diagrams



4.2.2 Use Case Diagrams



4.2.3 Control Flow Diagram


Start
Login
No
Login Successful
Logout
Yes
Settings
Home Page
voice command
voice command
voice command
Compose
Read
Draft

5. Implementation

5.1 Technology Stack

The Voice-Assisted Email Client App utilizes a comprehensive technological stack to deliver a seamless and empowering email communication experience for visually impaired and blind users. The technological stack encompasses both the front-end and back-end components, ensuring robust functionality and cross-platform compatibility.

1. Front-End: Developed using Flutter, a versatile UI software development kit (SDK) by Google, providing visually appealing and responsive user interfaces. Ensures cross-platform compatibility for Android and iOS.

2. Back-End: Built on Django, a high-level Python web framework, offering rapid development, security, and scalability. Simplifies data management and database operations.

3. Voice Assistance and Command: Integrates specialized Flutter voice recognition plugin for accurate transcription of voice commands, enabling hands-free interactions.

4. Text-to-Speech Conversion: Incorporates Flutter text-to-speech plugin to provide real-time audio feedback, confirming email actions and guiding users through the app.

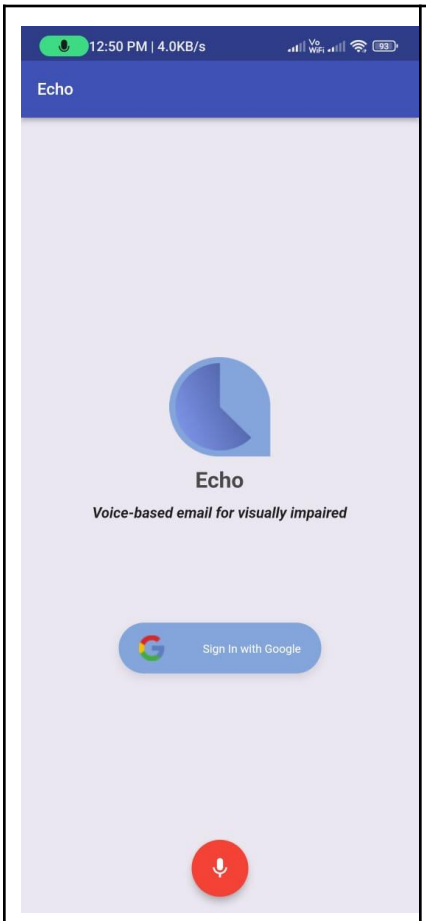
5. Email Integration: To provide seamless email management, the app leverages the Gmail API, a powerful interface provided by Google. The Gmail API allows the app to fetch and send emails from users' Gmail accounts in real-time, ensuring synchronization and enabling efficient email management.

5.2 GUI

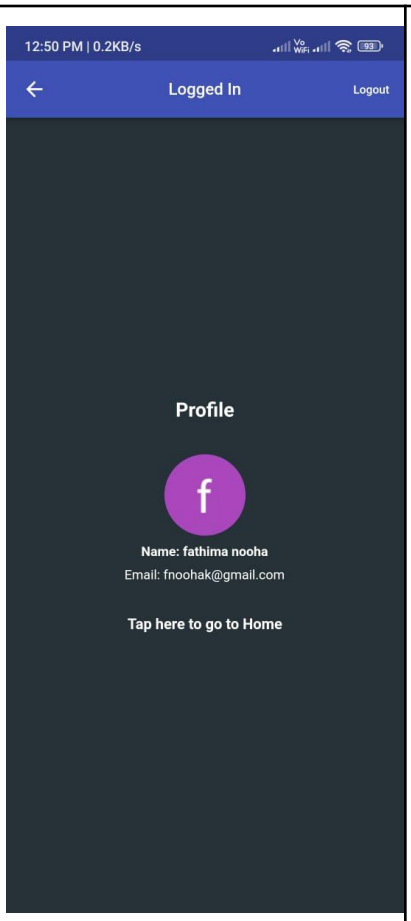
The GUI for "Echo" showcases a user-friendly interface with several key features. The splash screen welcomes users and provides an intuitive "Login with Gmail" option to access the Gmail login functionality seamlessly. Upon successful login, the Profile page confirms the authenticated user's identity and offers an option to logout securely.

The Home page serves as a central hub, enabling users to navigate effortlessly to essential sections such as the compose page, inbox page, and draft page. Additionally, it conveniently displays and reads out the latest unread emails, keeping users informed about their email activity.

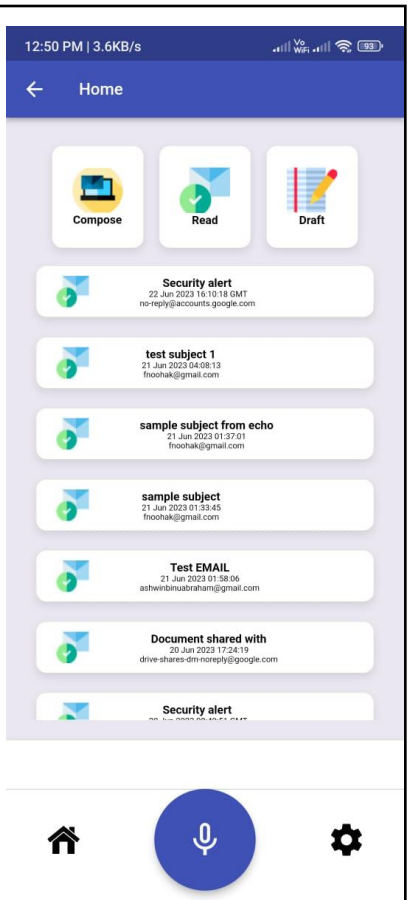
The Settings page empowers users with customization options, allowing them to adjust the speed of transcript reading for emails. This feature ensures that users can personalize their email listening experience to suit their preferences.



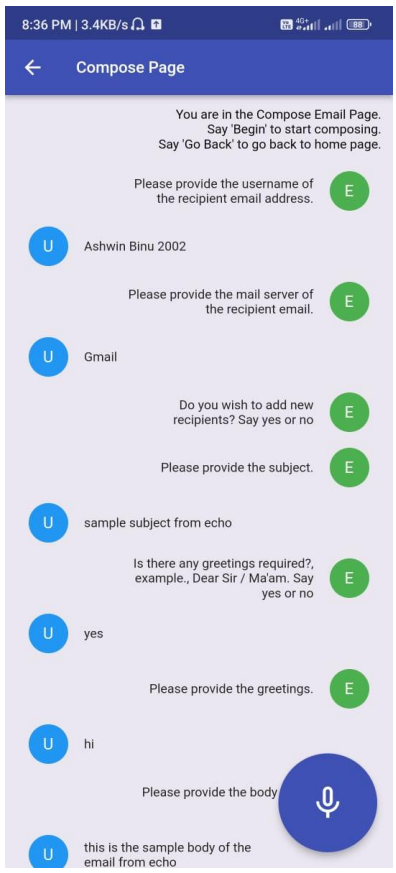
Splashscreen



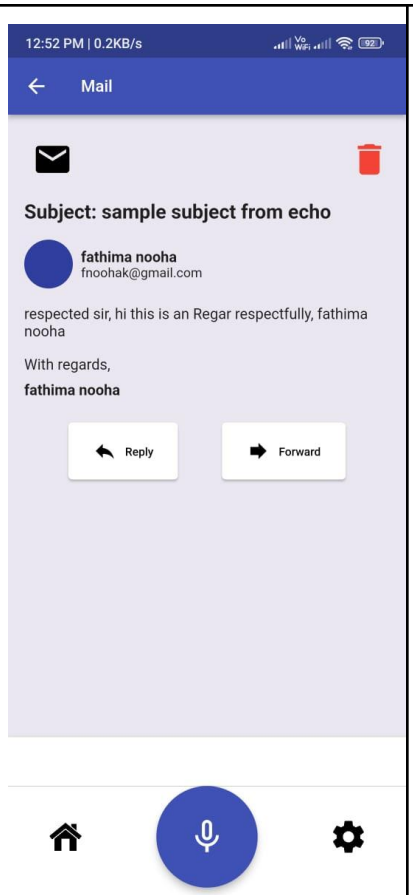
Profile page



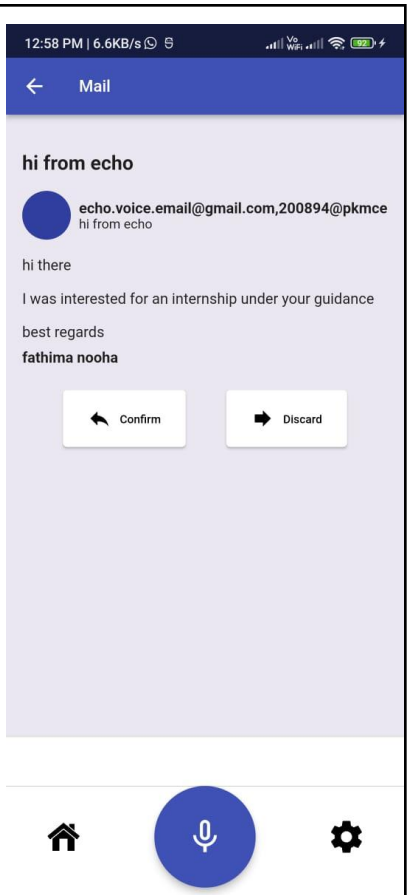
Home



Compose page

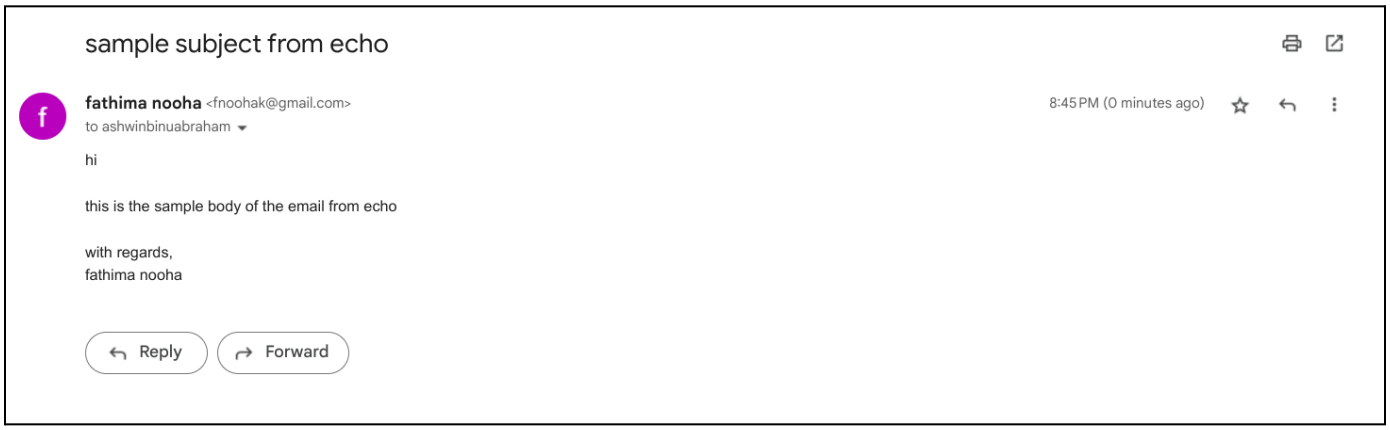


Inbox page

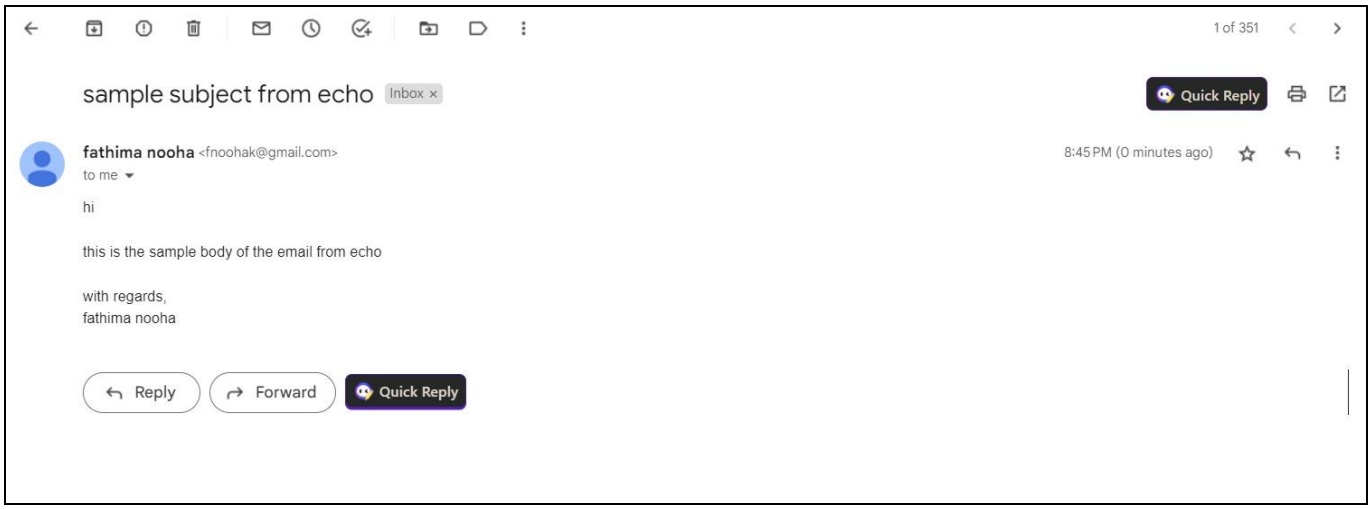


Draft mail page

5.3 Result



Send email



Received email

The development of the "Echo" app has been successful, as it has demonstrated the seamless sending and receiving of emails. Through the implementation of the specified functional requirements, users can compose emails using voice commands, edit and confirm their content, and send them to recipients with ease. The app's voice recognition and transcription capabilities accurately capture user inputs, facilitating effortless email composition. Additionally, the email reception and text-to-speech functionalities ensure that users receive incoming emails in a timely manner and can listen to the text content in a clear and natural-sounding audio format. The hybrid SDLC approach combining elements of the waterfall and agile methodologies has allowed for effective planning, development, and maintenance, ensuring that the "Echo" app meets the needs of visually impaired users while adhering to high standards of performance and usability.

6. Testing

Black Box Testing:

1. Functional Testing:
    a. Ensured successful user authentication with voice commands for email accounts.
    b. Tested email composition and sending, validating different recipient, subject, and content scenarios.
    c. Verified accurate text-to-speech conversion when reading incoming emails aloud.
    d. Tested the ability to save, retrieve, and edit email drafts.
    e. Validated various actions like marking emails as read, deleting them, and other necessary functionalities.

2. Usability Testing:
    a. Evaluated the user interface's accessibility and intuitiveness, focusing on visually impaired users.
    b. Assessed the voice recognition accuracy and system responsiveness to voice commands.
    c. Tested speech-to-text conversion accuracy and responsiveness to user inputs.
    d. Validated customization options in the settings module, such as speech speed adjustment.

3. Compatibility Testing:
    a. Verified the app's compatibility with different Android versions and email service providers like Gmail and Outlook.
    b. Tested the app's performance under various network conditions, including different Wi-Fi and mobile data connections.

White Box Testing:

1. Unit Testing:
    a. Conducted unit tests on individual functions and methods to ensure their correctness and behavior.
    b. Tested the speech recognition module for accurate voice command transcription.
    c. Verified proper audio output generation in the text-to-speech module.

2. Integration Testing:
    a. Tested the interaction between the front-end (Flutter) and back-end (Django) components.
    b. Verified the communication between the app and the Gmail API or other email service providers.

   c. Validated the integration between the speech recognition and text-to-speech modules.

3. Performance Testing:
   a. Measured the system's response time for various operations, such as email composition, reading, and navigation.

4. Security Testing:
   a. Conducted security testing to identify and address potential vulnerabilities in authentication and data handling processes.
   b. Verified data encryption during transmission and storage.

7. Deployment and Maintenance

Deployment:

The Echo mobile application is set to be deployed on the Google Play Store, providing Android users with easy access to download and install the app on their devices. The deployment process will involve adhering to the Play Store's specific requirements and guidelines to ensure successful publication.

To ensure an optimal user experience and performance, the Echo app will receive regular updates containing bug fixes and exciting new features. These updates will be seamlessly pushed to the Play Store, and users will be promptly notified of their availability on their Android devices. Additionally, the app will integrate a built-in feedback mechanism, empowering users to report bugs and propose new features, enhancing the app's ongoing development.

Regarding security, the Echo app will implement industry-standard security practices to safeguard user data and prevent unauthorized access. This comprehensive approach includes secure storage and transmission of user data, coupled with stringent access control measures for the app's backend services.

To ensure scalability and availability, the Echo app will be designed to work efficiently with a scalable cloud infrastructure. This architecture enables the app to handle increasing numbers of users and requests without compromising on performance or availability. Furthermore, regular monitoring of the app's cloud infrastructure will be carried out to ensure optimal functioning.

Finally, the deployment design includes a meticulous maintenance plan to ensure the app continues to operate as intended over time. This plan encompasses ongoing performance monitoring and user feedback analysis, as well as regular updates to maintain compatibility with new operating system versions and other dependencies, ensuring a seamless user experience.

Maintenance:

This section outlines the strategy for maintaining the voice-based email application catering to visually impaired users, ensuring its continuous functionality, security, and reliability. The maintenance team will be responsible for implementing regular updates, scheduled quarterly, with the possibility of emergency maintenance when required.

Key Strategies for Effective Maintenance:

- Regular Updates: The application will undergo regular updates to maintain compatibility with the latest screen reader software and email providers. Bug fixes and user feedback incorporation will be prioritized.

- Backward Compatibility: Ensuring backward compatibility with previous versions, allowing uninterrupted use for users who haven't upgraded to the latest version.

- User Feedback Mechanism: Implementation of a user feedback mechanism to gather insights and inform future updates and enhancements.

- Data Backup and Recovery: Regular backups of user data will be conducted to safeguard against data loss in case of system failure. A recovery plan will be in place to restore data if needed.

- Security Updates: Continuous monitoring for security vulnerabilities, prompt updates to ensure application remains secure and protects user data.

- User Training: User training materials will be updated in sync with application updates, ensuring users are informed about changes and adept at using the application effectively.

Change Management:
Formal change management process will be followed, involving change requests, thorough review, and implementation. Changes will be requested through a designated ticketing system and meticulously reviewed and approved by the development team to maintain software integrity.

Version Control:
GitHub will be employed for version control, facilitating tracking of changes, revisions, and deployment of updates. Changes will undergo rigorous review and testing before deployment to ensure system stability and reliability.

Documentation:
Comprehensive documentation will be maintained, including user manuals, technical documentation, and release notes. The maintenance team will ensure the accuracy and currency of all documentation, empowering end-users with necessary information for optimal software usage.

8. Conclusion and Future Scope

The project aims to address the challenges faced by visually impaired individuals in managing their email communications through the development of a voice-based email assistant called Echo. The app provides features such as voice-based email composition, reading, and management, catering specifically to the needs of visually impaired users.

Future Works:

1. Enhancing Platform Support: Currently, the Echo app is designed for Android smartphones with API levels above 21. Expanding the app's compatibility to other platforms, such as iOS, would increase its reach and impact on visually impaired users.

2. Improving Speech Recognition Accuracy: Although the app utilizes speech recognition technology, it is important to continuously improve its accuracy, especially for users with different accents or speech patterns. Integrating advanced speech recognition algorithms and leveraging machine learning techniques can help enhance the accuracy and user experience.

3. Integration with More Email Service Providers: While the app currently supports Gmail, integrating with other popular email service providers would make it more versatile and accessible to a wider range of users.

4. Collaboration with Assistive Technology Companies: Partnering with assistive technology companies can lead to bundled product and service offerings, expanding the app's reach and impact on visually impaired individuals. Collaboration can also provide opportunities for integrating other assistive technologies or devices to further enhance the user experience.

By addressing these future works, the Echo app can continue to evolve, offering an even more robust and user-friendly voice-based email assistant for visually impaired individuals, enabling them to manage their email communications independently and effortlessly.

9. References

Document Name	**Document URL**	**Issuance Date**
"Building Accessible Android Apps" by Google Developers	https://developer.android.com/guide/topics/ui/accessibility/principles	September 2021
"Best Practices for Developing Accessible Mobile Applications" by the World Wide Web Consortium (W3C)	https://www.w3.org/WAI/standards-guidelines/mobile/	December 2022
"Voice User Interface Design" by the Google Developers	https://developers.google.com/assistant/conversation-design	January 2023

APPENDIX I

Software Requirements Specification

Echo

Version 1.0

Prepared by

Ashwin Binu Abraham
Deepak P
Salihu Ahamed
Fathima Nooha Kottangodan

TKM College of Engineering

5th March 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Salihu Ahamed	25/02/2023	Initial draft prepared	1.0
Ashwin Binu	26/02/2023	Added additional functional requirements	1.0
Fathima Nooha	28/02/2023	Added overall description section	1.0
Deepak P	28/02/2023	Edited external interface requirements	1.0
Fathima Nooha	02/03/2023	Updated performance requirements	1.0
Ashwin Binu	03/03/2023	Edited System features part	1.0
Deepak P	04/03/2023	Added security requirements	1.0
Salihu Ahamed	05/03/2023	Finalized document for submission	1.0

1. Introduction

1.1 Purpose

The purpose of this project is to develop a mobile application that enables visually impaired users to easily read and send emails using voice commands. The application aims to provide visually impaired users with a solution that is tailored to their unique needs and challenges, allowing them to use email in a more intuitive and efficient way. The project also aims to improve the quality of life for visually impaired users by providing them with a tool that increases their independence and participation in society.

1.2 Document Conventions

This document is based upon:

- IEEE 830-1998 - IEEE Recommended Practice for Software Requirements Specifications
- IEEE 830-1984 - IEEE Guide for Software Requirements Specifications

1.3 Intended Audience and Reading Suggestions

Intended audience includes college professors, developers, industry experts, and other stakeholders who may be involved in the development or evaluation of the system

Review the overall design and system description, then focus on specific requirements and any diagrams or visual aids that can help clarify system behavior. Note any areas of interest or concern that could impact involvement or use of the system.

1.4 Product Scope

The mobile application aims to provide an accessible and convenient email client for visually impaired individuals by allowing them to manage their Gmail and Outlook accounts through voice commands. The purpose of the application is to provide visually impaired individuals with a user-friendly and efficient means of managing their email, which is essential in today's digital world. The key objectives of the application include:

- To provide a hands-free and intuitive email management experience through voice commands.
- To enable speech-to-text and text-to-speech functionality for ease of use.
- To offer compatibility with assistive technology, making it fully accessible for visually impaired individuals

To ensure the success and sustainability of the application, the following business strategies will be implemented:

- Partnership with assistive technology companies: Collaboration with assistive technology companies can enable bundled product and service offerings, expanding the app's reach to the target market.

- Targeted marketing: Marketing the application through channels that cater to the visually impaired community, such as organizations and support groups, will help reach the intended audience more effectively.

1.5 References

- Gmail API: https://developers.google.com/gmail/api/guides
- Outlook API: https://learn.microsoft.com/en-us/outlook/rest/
- Flutter Text To Speech Plugin: https://pub.dev/packages/flutter_tts
- Flutter Speech To Text Plugin: https://pub.dev/packages/speech_to_text
- Wake Word Detection Plugin: https://pub.dev/packages/porcupine_flutter/versions/2.0.1
- UI/UX References: https://dribbble.com/search/voice-email-app
- A survey of blind users on the usability of email applications by https://link.springer.com/article/10.1007/s10209-012-0285-9
- Voice Based Mail System for Visually Impaired: https://www.ijert.org/voice-based-mail-system-for-visually-impaired
- GeekForGeeks: https://www.geeksforgeeks.org/how-to-write-a-good-srs-for-your-project/
- Perforce blog by Gerhard Krüger and Charles Lane: https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document

2. Overall Description

2.1 Product Perspective

The voice-based email application is designed to provide an alternative means of email communication for individuals with visual impairments. The application will interact with various email providers through APIs to ensure compatibility and access to existing email accounts. The application will also integrate natural language processing (NLP) and speech-to-text (STT) technology to interpret user voice commands and generate text-based emails.

- **System Interfaces**

The voice-based email application will interface with the iOS and Android operating systems and the associated software and hardware. This includes access to system libraries and resources, such as audio and speech recognition libraries.

- **User Interfaces**

The user interface of the application will be designed to be simple and intuitive, with large buttons and clear audio feedback. The interface will be accessible for visually impaired users, allowing them to navigate the app easily and use voice commands to compose and read emails.

- **Hardware Interfaces**

The application will interface with the microphone and speaker of the user's device to receive voice commands and provide audio feedback. The user will speak commands into the microphone, which will be interpreted by the app's NLP and STT technology. The app will then provide audio feedback via the speaker, reading incoming emails aloud to the user.

- **Software Interfaces**

The voice-based email application will integrate with various email providers through APIs to ensure compatibility and access to existing email accounts. The application will be designed to integrate seamlessly with popular email providers, such as Gmail and Outlook, to ensure that users can access their email accounts without having to switch to a different email client.

- **Communications Interfaces**

The voice-based email application will use Wi-Fi or cellular data to communicate with email servers and API endpoints. The app will use standard email protocols such as SMTP, POP3, and IMAP to send and receive emails. The app will also use APIs to connect to email providers and to integrate with other services such as speech-to-text and text-to-speech conversion services.

2.2 Product Functions

- Login and authentication
- Voice-based email composition
- Voice-based email reading
- Integration with popular email providers
- User interface
- Compatibility with iOS and Android devices
- Settings and customization
- Error handling and notification

2.3 User Classes and Characteristics

Different users are identified on the basis of the review process of the document. It is listed as follows:

1. Customers

a. Educational institutions: The solution can be used in educational institutions to facilitate communication between students and teachers with visual or hearing impairments.

b. Non-profit organizations: Non-profit organizations that serve the disabled community can use the solution to enhance accessibility for their clients.

2. End Users

a. Visually Impaired Users: These are the primary users of the application who have visual impairments and face challenges in accessing and using email. They may have varying degrees of visual impairments and may require different levels of assistance from the application.

b. Regular Users: These are users who do not have visual impairments but may still benefit from the application's voice-based email composition and reading functions. They may also use the application to send emails hands-free while driving or doing other tasks.

c. Technical Users: These users may have technical expertise and require advanced features of the application, such as integration with custom email providers or customization of voice commands.

2.4 Operating Environment

The voice-based email application will be designed to operate on the following environments:

a. Mobile Devices: The application will operate on mobile devices running iOS 13 or later and Android 10 or later.
b. Internet Connectivity: The application will require an active internet connection to interact with the email server and API endpoints.
c. Hardware Requirements: The application will require a microphone and speaker to enable voice-based commands and audio feedback.
d. Software Requirements: The application will require access to speech-to-text and text-to-speech engines for voice interpretation and audio feedback, respectively.
e. Power Requirements: The application will require sufficient battery power to ensure uninterrupted operation.

2.5 Design and Implementation Constraints

- Hardware Constraints
    - The application requires the microphone and speaker of the user's device to be in good working condition.
    - The application is designed to work on both iOS and Android devices.
    - The Android API level should be above 21.
    - The device must have a good camera, speaker, and microphone.
    - The application works only on Wi-Fi or mobile data.

- Regulatory Constraints
    - The application must comply with the General Data Protection Regulation (GDPR).

- Functional Constraints
    - The application must use Flutter plug-ins and the Flask framework for front-end and back-end development respectively.
    - The application must be developed using Android Studio or VS Code IDE.
    - The user interface should be simple and easy to understand, with readable text and smooth transitions.
    - The application should be space-efficient and use a readable and maintainable code.

- The variables should be named using camel case, and the class names should start with a capital letter.
- The application should follow proper exception handling mechanisms.

- Language Requirements
  - The front-end should be developed using Dart programming language.
  - The back-end should be developed using the Python programming language.

2.6 User Documentation

- A user guide will be distributed to customers and developers as soft-copy in pdf format.
- An online forum will be created to allow users to connect with each other and ask questions, share tips, and provide feedback on the application.
- The end users (target) would be provided a tutorial video available in the Google Play Store while installing the app.

2.7 Assumptions and Dependencies

Assumptions:

- Users have access to a compatible device with the necessary hardware (microphone, speaker, camera) and software (Android or iOS operating system, internet connectivity).
- Users have basic knowledge of using mobile devices and accessing email.
- The email service providers' APIs remain stable and compatible with the application.

Dependencies:

- Flutter framework for cross-platform mobile application development.
- Speech-to-text plugin for converting spoken words to text.
- Text-to-speech plugin for converting text to spoken words.
- tflite_flutter and tflite_flutter_helper plugins for integrating machine learning models into the application.
- Email provider APIs for accessing and managing email accounts.

3. External Interface Requirements

3.1 User Interfaces

At initial startup the homepage will contain the login to the email of users.Then on homepage will show buttons for:

Inbox
Compose
Starred
Settings

- A button to read out the text on the screen(Text to speech) and provide audio assistance will be placed on the bottom of the screen.
- A button for speech to text option will be available in the compose email section.
- On every new page open an audio will be spoken to detail the current state of the page and to ask the instruction from the user.

3.2 Hardware Interfaces

The application is user-friendly.Buttons can be used for mode of transactions.The application mainly deals with audio data inputs.To record the user's voice instructions and send them to the app for processing, microphone is used.A speaker or headphones is used to receive auditory feedback. To connect with external hardware devices like Bluetooth headsets or hearing aids, the app needs Bluetooth connectivity.

3.3 Software Interfaces

TTS flutter plugin is used for text-to-speech features.Speech_to_text plugin is used for speech-to-text conversion.

4. System Features

4.1 Integration with Email Providers

4.1.1 Description and Priority

The mobile application is designed to be compatible with two popular email providers, Gmail and Outlook. This means that users with Gmail or Outlook email accounts will be able to use the app to access and manage their email

The compatibility with email providers feature is of high priority as it is a fundamental functionality of the application. Without this feature, the application would not be able to serve its purpose of providing a user-friendly email client for visually impaired individuals.

4.1.2 Stimulus/Response Sequences

1. User launches the application and selects the option to add an email account.
2. Users are prompted to enter their email address and password through voice commands.
3. System verifies the entered credentials and fetches the user's email account information.
4. User selects the email provider from the list of supported providers.
5. System prompts the user to grant permission for the application to access their email account.
6. User grants permission through voice commands.
7. System completes the integration process and displays the user's inbox on the app's home screen.

For first-time users, the following steps are added:

1. Download and install the mobile application from the app store.
2. Launch the application and grant necessary permissions, such as microphone and accessibility access.
3. Upon opening the application, users will be prompted to enter their email credentials for Gmail or Outlook.
4. The application will verify the credentials and connect to the email account.
5. Users will be able to start managing their emails through voice commands immediately.

4.1.3 Functional Requirements

REQ-1: The app should be compatible with Gmail and Outlook email providers.
REQ-2: The app shall provide users with the option to set up their email accounts using voice commands.
REQ-3: The app shall authenticate user credentials for the email provider through the user's voice command.

4.2 Voice-based email composition

4.2.1 Description and Priority

This feature allows users to compose emails using voice commands, making it more accessible and convenient for individuals with visual impairments or those who prefer a hands-free approach to email composition.

The priority of the voice-based email composition feature is high, as it is a key functionality of the application that enables visually impaired users to compose and send emails hands-free. It is a primary reason why users would choose to use this application over other email clients.

4.2.2 Stimulus/Response Sequences

1. User says "Compose email".
2. System prompts the user to dictate the recipient's email address.
3. User dictates the recipient's email address.
4. System confirms the recipient's email address and prompts the user to dictate the subject of the email.
5. User dictates the subject of the email.
6. System confirms the subject of the email and prompts the user to dictate the body of the email.
7. User dictates the body of the email.
8. System confirms the body of the email and prompts the user to confirm whether they want to send the email.
9. User confirms they want to send the email.
10. System sends the email to the recipient's email address and confirms that the email has been sent.

4.2.3 Functional Requirements

REQ-1: The application should have a voice recognition system capable of transcribing spoken text accurately
REQ-2: The application should be able to format the transcribed text into a readable email format
REQ-3: The application should allow users to edit the transcribed text if necessary
REQ-4: The application should provide users with a confirmation option before sending the email

4.3 Text-to-speech for reading emails

4.3.1 Description and Priority

The text-to-speech feature for reading incoming emails is a high-priority requirement of the system, as it is essential for visually impaired or blind users who rely on auditory feedback to access their emails.

This feature enables the system to convert incoming email text content into audio format, which can then be read aloud to the user by the system's built-in text-to-speech engine. The user can also control the speed and volume of the speech output, as well as pause, rewind, or skip the audio playback as needed.

4.3.2 Stimulus/Response Sequences

The stimulus for this feature is an incoming email that has been received by the user's email account. When a new email arrives, the system should automatically trigger the text-to-speech feature to read the email's contents aloud to the user.

The response sequence for this feature involves several steps:

1. The system identifies the new email in the user's inbox and extracts the email's text content.
2. The system sends the email's text content to the text-to-speech engine for conversion into audio format.
3. The system plays the audio output through the user's device speakers or headphones.
4. The user can control the audio playback and adjust the speech speed and volume as needed.

4.3.3 Functional Requirements

REQ-1: The system must be able to detect new incoming emails in the user's email account and extract the text content of the email.
REQ-2: The system must have a built-in text-to-speech engine capable of converting the email's text content into an audio format.
REQ-3: The user must be able to control the speed and volume of the speech output through the system's user interface.
REQ-4: The user must be able to pause, rewind, or skip the audio playback as needed through the system's user interface.
REQ-5: The system must provide a clear and natural-sounding audio output that is easily understandable by the user.

4.4 Ability to edit and delete emails

4.4.1 Description and Priority

The ability to edit and send delete is a crucial feature of any email application. This feature allows users to manage their email communications effectively by composing new emails, modifying the content of existing emails, and deleting unwanted emails. The feature provides a user-friendly interface to perform these actions with ease and efficiency.

This feature has high priority as it is essential for the core functionality of the email application. Without this feature, the application would not be useful for managing email communications.

4.4.2 Stimulus/Response Sequences

1. When the user says "edit email," the system prompts the user to select an email to edit.
2. The user selects the email and the system opens the email for editing.
3. The user can then edit the email by using voice commands such as "insert text," "delete text," or "change text."
4. When the user has finished editing, they can say "send email" to send the updated email.

4.4.3 Functional Requirements

REQ-1: The system should have the ability to recognize voice commands for editing emails.
REQ-1: The system should provide audio feedback to confirm user actions, such as deleting or changing text.
REQ-1: The system should allow the user to undo and redo changes made to the email.
REQ-1: The system should have the ability to save drafts of emails in progress.
REQ-1: The system should provide error messages if the user tries to send an email without a recipient or with an invalid email address.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The system should respond to user input within 2 seconds.
- The system should load images and data within 5 seconds.
- The system should be able to handle 100 simultaneous users without performance degradation.

5.2 Safety Requirements

- The system should not allow users to upload or access malicious files.

- The system should encrypt all user data in transit and at rest.
- The system should have a failsafe mechanism to prevent catastrophic failures.

5.3 Security Requirements

- The system should have multi-factor authentication for user accounts.
- The system should have role-based access control to restrict user access to certain features.
- The system should be compliant with GDPR and other privacy regulations.

5.4 Software Quality Attributes

- Capacity: The storage requirements of the app can be minimal since no information about the user is stored.
- Usability: The product is very simple and easy to use as it has an AI assistant to assist the disabled people. It will be required for better communication.
- Compatibility: Compatible with any version of the android which supports voice assistants.
- Response Time: Each page loads within 10-15 seconds and the output is expected to have the same response time.
- Reliability: Chances of failure or system crash is low compared to other heavy applications that require huge storage. However, the application is difficult to work without Wi-Fi or mobile data.
- Availability: The app will be available to the user at all times except times of maintenance. It will be made available in the Play Store for installation.
- Efficiency: The app is very efficient as the user can reach their goal within minimal time.

5.5 Business Rules

- Only registered users should be able to access certain features.
- Users should not be able to upload copyrighted or illegal material.
- Users should be able to report any inappropriate content or behavior.

6. Other Requirements

- Legal Requirements

The system shall comply with all applicable laws and regulations regarding user privacy and data protection, including the General Data Protection Regulation (GDPR).

- Reuse Objectives

The system shall be designed with a modular architecture to facilitate code reuse and extensibility. The system shall be designed using standard software engineering principles to maximize reuse of existing software libraries and frameworks.

- Accessibility Requirements

The system shall comply with accessibility standards and guidelines, including the Web Content Accessibility Guidelines (WCAG) 2.1. The system shall be designed to be accessible to users with disabilities, including visual, auditory, and motor impairments.

- Performance Requirements

The system shall be designed to handle a high volume of concurrent users, with a target response time of 2 seconds or less for all user interactions. The system shall be scalable to support future growth and increasing user loads.

- Usability Requirements

The system shall be designed to be intuitive and easy to use, with a modern and responsive user interface. The system shall be designed to minimize user errors and to provide clear and concise error messages when errors occur.

- Maintenance Requirements

The system shall be designed to be maintainable and easy to update, with clear separation of concerns and modular design. The system shall be designed to minimize downtime during maintenance and to provide easy rollback in case of errors.

- Testing Requirements

The system shall be designed with testing in mind, with a comprehensive suite of automated tests to ensure functionality, performance, and security. The system shall be designed to be testable, with clear separation of concerns and well-defined interfaces.

Appendix A: Glossary

TTS: Text-to-Speech
STT: Speech-to-Text
API: Application Programming Interface
UI/UX: User interface and user experience

APPENDIX II

Software Design Document

Echo

Version 1.0

Prepared by

Ashwin Binu Abraham
Deepak P
Salihu Ahamed
Fathima Nooha Kottangodan

TKM College of Engineering

3rd April 2023

Table of Contents

List of Figures

List of Tables

1. Introduction

This Software Design Document (SDD) is intended to provide a comprehensive overview of the design and architecture of a software system. The document is aimed at developers, architects, and stakeholders involved in the development of the system. It outlines the key design decisions, architectural patterns, and technologies used to build the system.

The SDD covers various aspects of the software design, including system architecture design, application architecture design, GUI design, API design, and technology stack. The document provides detailed descriptions of the design decisions and provides an overview of the system architecture, the application architecture, and the user interface design. Additionally, it provides details about the API design and the technologies used to build the system.

This document aims to serve as a reference guide for the development team throughout the development process. It will help to ensure that the team remains aligned with the design goals and can make informed decisions regarding the implementation of the system.

1.1 Purpose

The purpose of the Software Design Document (SDD) is to provide a comprehensive overview of the design of the software system being developed. It serves as a guide for the developers, testers, and stakeholders involved in the project, and provides a common understanding of the system architecture, application architecture, GUI design, API design, database design, and technology stack. The document outlines the technical aspects of the software and the design decisions that were made during the development process, ensuring that the software meets the specified requirements and performs as expected. The SDD also helps to identify potential issues and risks that may arise during the development and implementation phases, enabling proactive measures to be taken to mitigate them. Overall, the SDD serves as a critical reference document for the development team throughout the software development lifecycle.

1.2 Scope

The scope of this Software Design Document (SDD) is to provide a comprehensive overview of the design and architecture of the proposed application. It covers the system architecture design, application architecture design, GUI design (mockups), API design, and the technology stack that will be used to develop the application. The SDD aims to provide a clear understanding of the design and architecture of the application to the development team and stakeholders involved in the project. It will also serve as a reference guide for the development team throughout the development process. The SDD is intended to be a living document and will be updated as needed throughout the project lifecycle.

1.3 Intended Audience

The intended audience for this SDD includes software developers, College Professors, and other stakeholders involved in the software development process. This document is intended to provide a detailed design of the system architecture, application architecture, GUI design, API design, and technology stack, as well as any other relevant technical details related to the development of the software application.

The audience can review the overall design and technical description, and focus on specific components such as the system architecture, application architecture, GUI design, API design, and technology stack.

Any concerns or areas of interest that could impact the development or evaluation of the software application can be noted for further discussion and clarification

1.4 References

- IEEE Standards Association. IEEE Std 1016-2009, IEEE Standard for Information Technology--Systems Design--Software Design Descriptions. IEEE, 2009.
- Martin, Robert C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017.

2. System Architecture Design

2.1 Description

The architecture consists of three main tiers: the Presentation Tier, the Functionality Tier, and the Data Storage Tier. The Presentation Tier is responsible for the user interface and screens, which are developed using Flutter, and the voice command plugin. The Functionality Tier is responsible for the email service, which handles communication with the Gmail API and Outlook API to perform email operations such as sending, receiving, and reading emails, as well as the email processor, which processes received emails and stores them locally in the device storage for offline access. The Functionality Tier also includes the Flutter plugin for enabling voice commands. The Data Storage Tier is responsible for locally storing the fetched emails for offline access, and the emails are stored in the device's local storage.

These three tiers work together to provide the complete functionality of the system, allowing visually impaired people to perform basic email operations through voice commands using the mobile app

2.2 Architecture

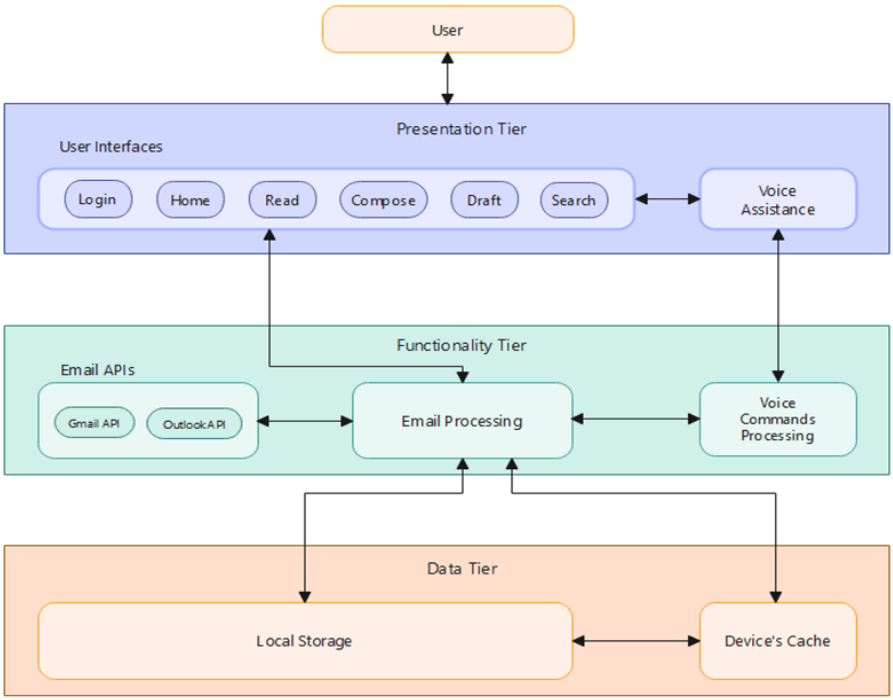


Figure 1.0: Architectural Overview

3. Application Architecture Design

The Echo app will be a mobile email client designed for visually impaired users. The application architecture will consist of two main components: a frontend built using the Flutter framework, and a backend built using the Django web framework.

The frontend will handle the user interface and the voice command processing. The interface will be designed with accessibility in mind, with high-contrast and large text options available. The voice command processing will be implemented using Flutter plugins.

The backend will handle the email operations and will integrate with the Gmail and Outlook APIs for email retrieval, sending, and management. The backend will also implement the authentication flow for both Gmail and Outlook.

Overall, the application architecture is designed to provide a smooth and accessible email experience for visually impaired users, utilizing the latest technologies for mobile app development and API integration

4. GUI Design

4.1 User Interface Design

a. Splash Screen
Displaying the application logo and name for 1.5 seconds.

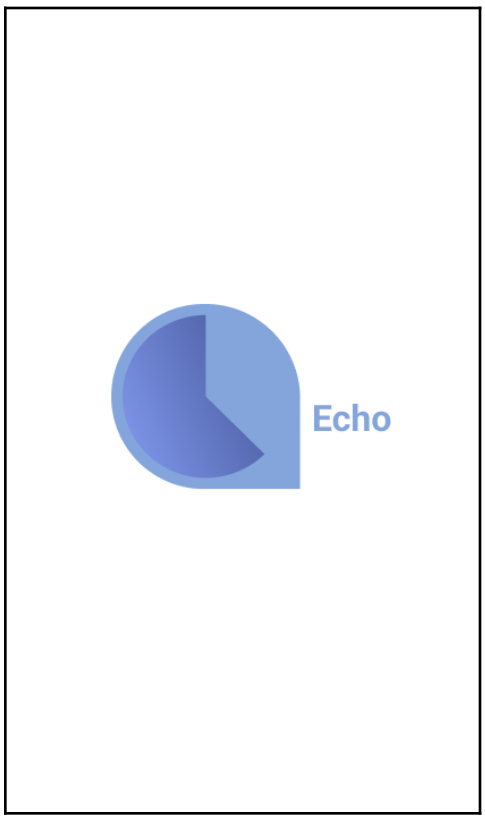


Figure 2.0: Splash Screen GUI

b. Login Screen
A voice-over provides instructions on how to login. The text boxes are filled as per the voice input from users.

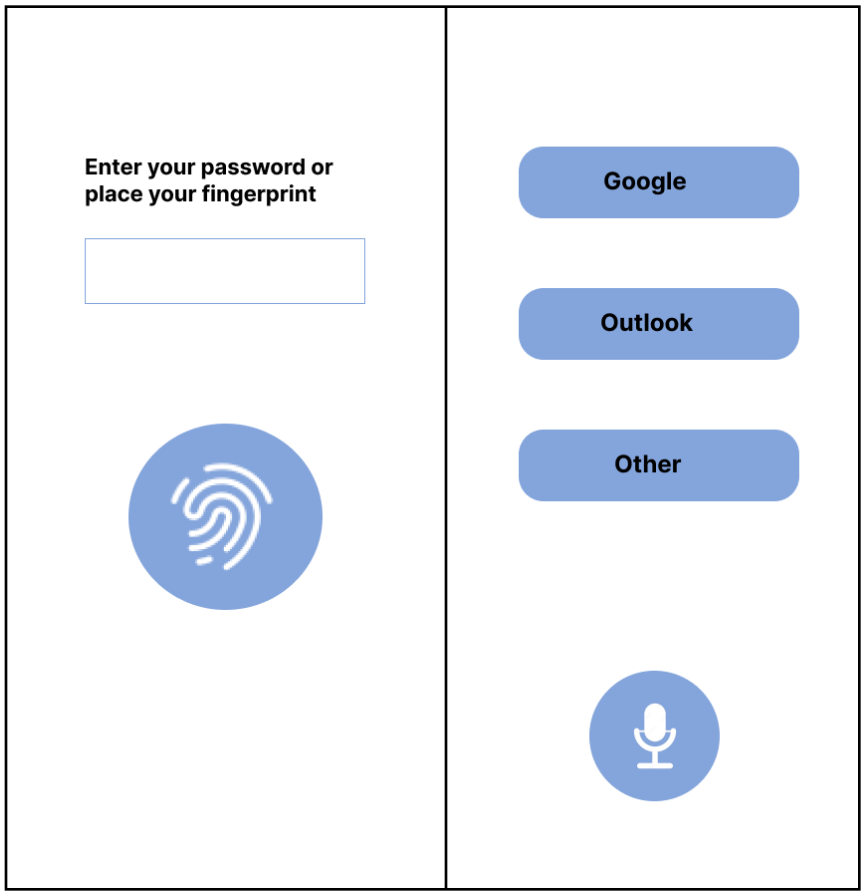


Figure 3.0:Login Screen GUI

c. Home Screen

- The home screen includes three buttons, Compose, Read and Draft which is announced by a voice-over.
- The user can navigate using voice commands.
- A microphone icon in the bottom center can be used to provide voice input.
- To the right of the microphone icon is the settings icon. (Users can also select settings by saying settings to the input).

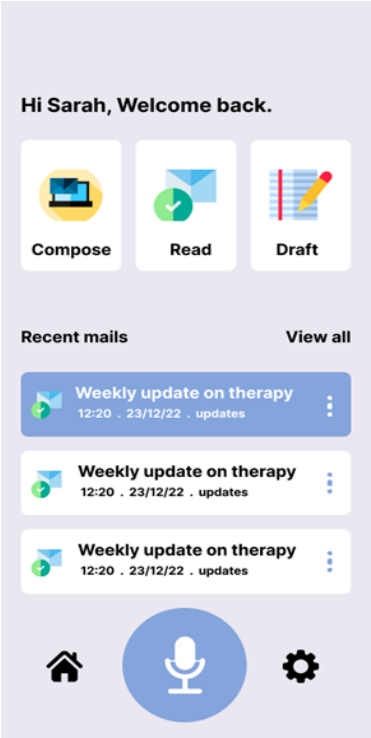


Figure 4.0: Home Screen GUI

d. Compose Screen

- The compose screen includes voice-over instructions on how to compose a new email.
- The user can input the recipient's email address, subject, and message using voice commands.
- To the right of the microphone icon is the settings icon and to the left is the home screen icon.

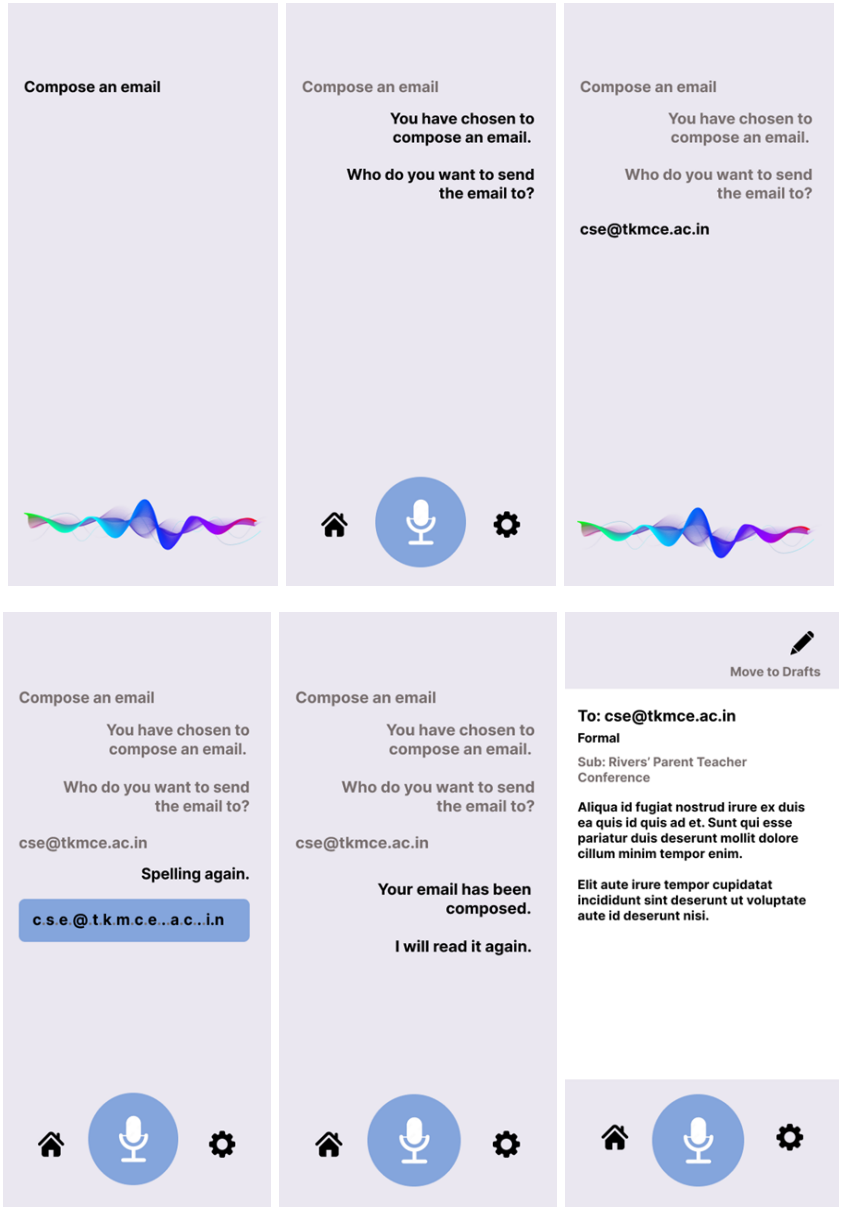


Figure 5.0: Compose Screen

e. Read Email Screen

- The read email screen includes a list of unread emails. The user can view all emails by clicking the view all button or saying it to the input.
- Unread emails are announced by voice-over. Email is selected as per the user input and the email's content is read out loud by a voice-over.
- There are voice commands and buttons to go back to the inbox, delete the email, reply, forward etc
- To the right of the microphone icon is the settings icon and to the left is the home screen icon.

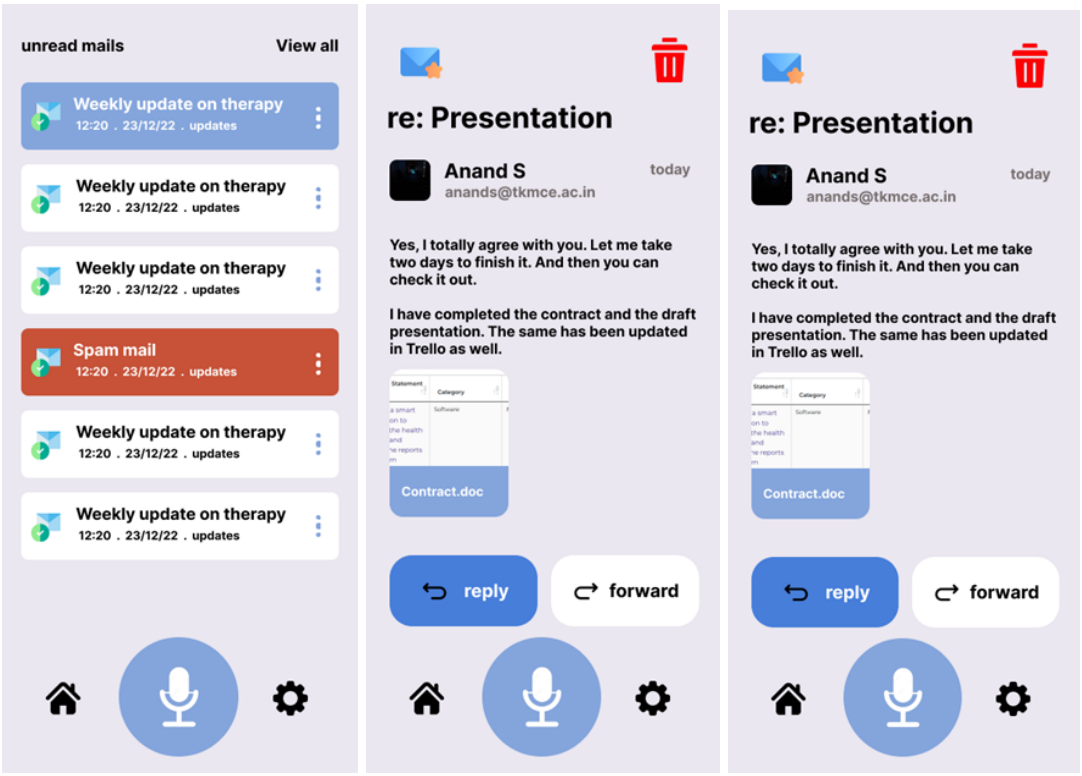


Figure 6.0: Read Email Screen

f. Draft Screen

- The drafts are listed by subject and can be edited or sent using voice commands.
- To edit a draft email, the Compose screen is used, with To and subject prefilled.
- To the right of the microphone icon is the settings icon and to the left is the home screen icon.

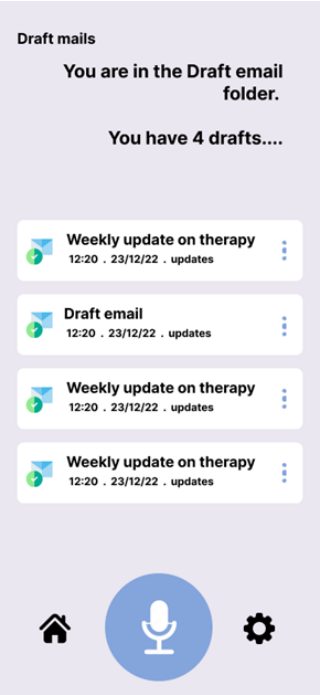


Figure 7.0: Draft Screen

g. Settings Screen

- Buttons for Account settings, Audio control, Sound control and support.
- As per the chosen option, corresponding fields are displayed.

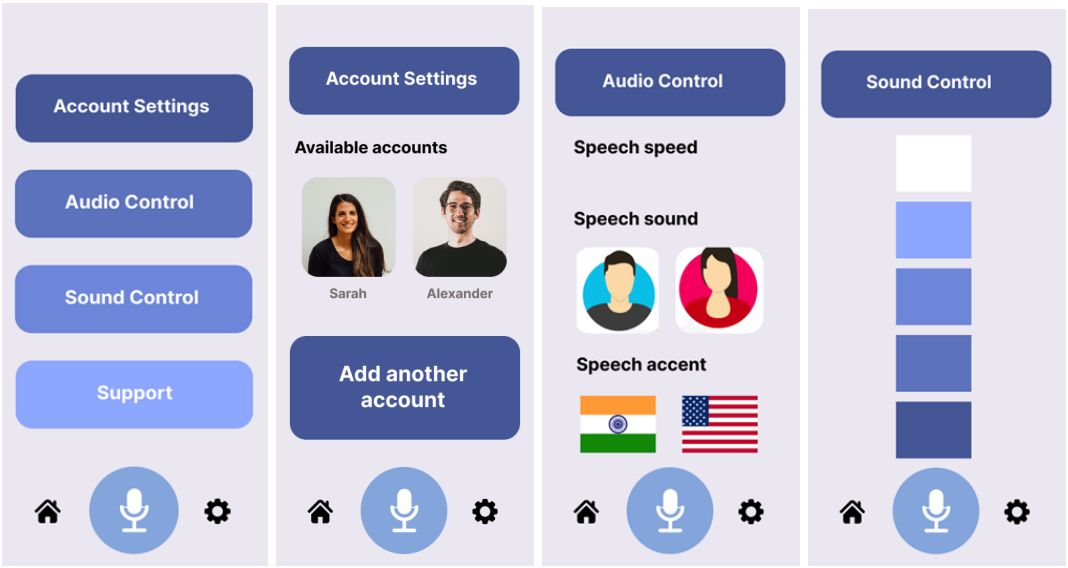


Figure 8.0: Settings Screen

4.2 Visual Design

- The GUI design will use high-contrast colors and larger font sizes for better visibility and readability.

- Icons and buttons will have descriptive labels in addition to their visual representation.
- Images will be accompanied by descriptive voice-over text where necessary.

4.3 Navigation Design

- The application will use a consistent navigation structure throughout
- Navigation will be designed with clear and descriptive labels to aid users with limited vision.

5. API Design

The Echo application utilizes two main APIs: Flutter and Django. Flutter provides built-in packages for text to speech and speech to text conversion, while Django is used as the back-end framework for the application.

5.1 Flutter API

Flutter provides a set of built-in APIs for text to speech and speech to text conversion. These APIs will be used to enable users to interact with the Echo application using their voice.
The Text to Speech API will convert text-based emails into audible speech, which can be listened to by the user. The Speech to Text API will convert the user's spoken commands into text format, which can then be used to navigate the application, compose new emails, and reply to existing ones.

5.2 Django API

Django is used as the back-end framework for the Echo application. The Django API is responsible for handling user requests and managing email data.
The Django API consists of several endpoints, including:

Authentication endpoint: Used to authenticate users and grant access to their email accounts.
Email retrieval endpoint: Used to retrieve email data from the email server.
Email composition endpoint: Used to compose and send new emails.
Email deletion endpoint: Used to delete emails from the user's inbox.

5.3 Natural Language Processing (NLP) API

The Echo application also plans to incorporate NLP functionality within Django to enhance the accuracy of speech to text conversion. NLP will be used to analyze the user's speech patterns, identify common phrases, and improve the overall accuracy of voice recognition within the application.

6. Technological Stacks

- Flutter framework:

  Flutter is an open-source UI framework for building high-performance, cross-platform mobile applications. It provides a fast development experience and supports a wide range of mobile platforms

- Dart programming language

  Dart is the primary programming language used in Flutter development. It is a modern, object-oriented language with features like garbage collection, asynchronous programming, and strong type checking.

- Speech recognition and synthesis libraries

  To enable voice-based interaction with the email application.We speech recognition and synthesis libraries within the flutter framework. Flutter provides plugins for both Google's Speech-to-Text API and Text-to-Speech API that can be used to implement these features.

- Email service provider API

  Email Service Provider (ESP) API is an interface provided by an email service provider such as gmail,outlook etc that allows developers to programmatically interact with the ESP's email sending and management capabilities.

- Dart Speech

  The Dart Speech package also provides a set of utilities for speech recognition and synthesis

7. Component Design

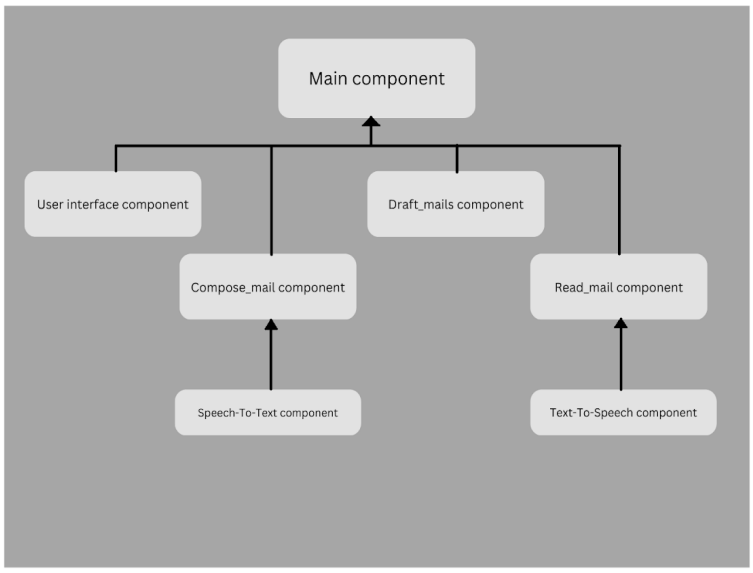


Figure 9.0: Component Design

- User interface component

  A user interface component is needed to provide feedback to the user and allow them to interact with the system even though primary input for the system is voice input.

- Compose_mail component

  This component is responsible for providing the user with a user-friendly interface for composing new email messages.It includes the features user Input ,confirmation and send.
  Speech-To-Text component is integrated with compose_mail component for the conversion of user voice inputs to text.

- Draft_mails component

  This component deals with draft mails.Users can review their unsend and incomplete compose mails and take actions.

- Read_mail component

  It is responsible for providing the user with a user-friendly interface for reading and managing their incoming email messages.
  Read_mail component should be able to convert the text of the email message into spoken words using a Text-To-Speech component.
  It also contains other features including filtering,mark as read/unread and delete.

8. Algorithm Design

Initialize the app:
a. Load the necessary libraries and modules for the app to function
b. Establish a connection to the server using Django framework
c. Authenticate the user and retrieve their email credentials

Main Loop:
a. Display the main menu options for the user:
    i. Compose a new email
    ii. Read incoming emails
    iii. Write Drafts
    iv. Navigate to a specific email
    v. Exit the app

b. Wait for user input, either through voice commands or GUI buttons

c. If user selects "ECHO COMPOSE":
    i. Prompt the user to speak the recipient's email address
    ii. Convert the speech input to text
    iii. Verify the email address is valid and prompt the user to confirm
    iv. Prompt the user to speak the email subject
    v. Convert the speech input to text
    vi. Prompt the user to speak the email body
    vii. Convert the speech input to text
    viii. Send the email

d. If user selects "ECHO READ":
    i. Retrieve the list of new emails from the email server
    ii. For each email, extract the subject, sender, and message body
    iii. convert the message body to speech
    iv. Display the email information to the user using GUI interface and also read the email information through voice

e. If user selects "ECHO LIST":
    i. Retrieve the list of sent emails from the email server
    ii. For each email, extract the subject, recipient, and message body
    iii. Display the email information to the user using GUI interface and also read the email information through voice

f. If user selects "ECHO DRAFT":
    i. Prompt the user to speak the email's subject
    ii. Convert the speech input to text
    iii. Prompt the user to speak the email's body

iv. Convert the speech input to text

v. Display the email information to the user using GUI interface and also read the email information through voice

g. If user selects "ECHO SPELL":

i. Display the email information letter by letter to the user using GUI interface and also read the spelling of email information through voice

h. If user selects "ECHO EXIT":

i. Terminate the main loop and exit the app

9. Data Design

- Data Storage
    - The application will need to store various types of data, including user account information and email messages. To do this, we will use a relational database management system (RDBMS) such as MySQL or PostgreSQL. The database will have the following tables:
    - Users: Stores user account information such as username and password.
    - Emails: Stores email messages along with metadata such as sender, recipient, subject, and timestamp.
    - We will use Django's built-in Object-Relational Mapping (ORM) framework to interact with the database. This will allow us to use Python code to define the structure of the database and interact with it, instead of writing raw SQL queries.

- Data Access
    - To access the data stored in the RDBMS, the Django ORM provides several APIs such as QuerySet, Model and Manager. This will allow us to access the data using object-oriented programming techniques.

- Data Transfer
    - The application will need to transfer data between the user's device and the email server. To do this, we will use the Simple Mail Transfer Protocol (SMTP) for sending email messages and the Internet Message Access Protocol (IMAP) for receiving email messages. We will use Python's built-in smtplib and imaplib modules to communicate with the email server.

- Data Backup and Recovery
    - To ensure that user data is not lost in the event of a system failure, we will implement regular backups of the database. We will also develop a recovery plan in case of a disaster, such as restoring the database from a backup or using a replication solution to maintain multiple copies of the data. We will use Django's built-in dumpdata and loaddata commands to backup and restore the database.

- Speech-to-Text Conversion
    - The application will use speech-to-text conversion to allow users to enter details, send and read emails. We will use a third-party API such as Google Cloud
    - Speech-to-Text or Amazon Transcribe to convert speech input to text.

- Speed Control
    - The application will allow users to control the speed of voice replies. This will be stored as a configuration setting for each user. However, since there are no other configuration settings, this will be stored in the Users table as a field.

10. Error Handling Design

- Network errors: In a voice-based email system, network errors such as slow internet connections can be common.The system handles these errors, with appropriate error messages and automatic retries when possible.Also when sending a mail in the absence of internet, the mail should be transferred to draft automatically.

- Error reporting: Whenever an error or crash occurs it will be reported to the server.

- Hardware error checking: Users can test the speakers and mic of their system to ensure they are fully operational.

- Clear and concise error messages: Since users will be interacting with the system using their voice, error messages should be clear, concise, and easy to understand.

- Person names,slangs etc handling: When we are inputting or reading such words the system cannot identify them.At this case the user can input or the system can read them character by character.

11. Performance Design

The performance of the Echo application is critical to ensure a smooth and efficient user experience. The following performance design considerations have been taken into account during the development of the application:

- Text to Speech and Speech to Text Conversion:
  The performance of the voice conversion feature is critical to ensure that the user can efficiently interact with the application. We will be using Flutter's built-in package for text to speech and speech to text conversion to ensure a fast and reliable conversion process. This package is known for its speed and accuracy, which will provide a seamless experience to the users. Additionally, we are also planning to implement NLP (Natural Language Processing) inside Django to further improve the accuracy of the conversion.

- Backend Framework:
  Django has been chosen as the backend framework for the Echo application. Django is a high-performance and scalable framework that can handle a large volume of requests. We will be optimizing the backend for better performance by using caching techniques and optimizing the database queries. The use of Django will ensure that the application can handle a high volume of requests without any performance issues.

- Frontend Framework:
  Flutter has been chosen as the frontend framework for the Echo application. Flutter is a high-performance framework that enables the creation of fast and efficient mobile applications. We will optimize the application's UI to ensure that it is lightweight and fast. This will ensure that the application is responsive and provides a smooth user experience.

- Server and Database:
  The performance of the server and database is critical to ensure that the application can handle a high volume of requests. We will be using a high-performance server and database to ensure that the application can handle a large volume of requests without any performance issues. We will also be optimizing the database queries to ensure that they are fast and efficient.

- Caching:
  Caching is a crucial technique to improve application performance. We will use caching to store frequently accessed data to reduce the number of database queries and improve application performance. This will ensure that the application is fast and responsive.

- Testing and Optimization:

We will be conducting regular performance testing to ensure that the application is performing optimally. Any performance issues that are identified during testing will be addressed immediately to ensure a smooth and efficient user experience. We will also be optimizing the application to ensure that it is as fast and efficient as possible.

12. Security Design

- Secure authentication: It is ensured that only authorized users can access the voice-based email system. This might include password-based authentication or biometric authentication.

- Storage security: The system mainly uses the data from the database of the email provider.

- Compliance with data privacy regulations: The voice-based email system is subject to data privacy regulations such as GDPR.

- Scam protection: Using the email service provider features the software will let the user to ignore the scams and ad mails.

13. Testing Design

13.1 Testing Approach

The overall approach for testing the application will be a combination of manual and automated testing. Manual testing will be used to verify the user interface and user experience, while automated testing will be used to verify functionality, integration, and performance. Each feature of the application will be tested separately before testing the application as a whole.

13.2 Testing Scope

The scope of testing for this application will include the following types of testing:

1. Functional testing: To ensure that the application meets its functional requirements
2. Integration testing: To ensure that all components of the application work together seamlessly.
3. Performance testing: To ensure that the application performs within acceptable limits and responds to user input within 2 seconds and loads images and data within 5 seconds
4. Security testing: To verify that the application is secure and complies with GDPR and other privacy regulations
5. Usability testing: To test the application's ease of use and accessibility for blind and visually impaired users.

13.3 Test Cases

We will use the following test cases to verify that the application meets its requirements:

1. **Functional Testing**
    a. Test Case 1 - Email Account Setup
        Expected Result: Users should be able to set up their email accounts and log in to their Gmail or Outlook accounts and authenticate their credentials through voice commands.
    b. Test Case 2 - Voice Recognition and Email Formatting
        Expected Result: The application should be able to transcribe spoken text accurately, format the transcribed text into a readable email format, allow users to edit the transcribed text if necessary, and provide users with a confirmation option before sending the email.
    c. Test case 3 - Edit the transcribed text
        Expected Result: The user is able to edit the transcribed text before sending the email.
    d. Test case 4 - Confirming the email before sending
        Expected Result: The user is able to confirm the email before sending it.
    e. Test Case 5 - Voice Recognition Accuracy
        Expected Result: The application should accurately transcribe spoken text, even in noisy environments or with accents or speech impediments. Tested by speaking with different accents, using different volumes or pitches, or speaking in noisy environments.
    f. Test Case 6 - Email Attachment

Expected Result: The application should allow users to add attachments to their emails, either through voice commands or manual selection.

g. Test Case 7 - Email Forwarding

Expected Result: The application should allow users to forward received emails to other recipients using voice commands. Test this by receiving an email and forwarding it to a different email address or contact.

h. Test Case 8 - Email Reply

Expected Result: The application should allow users to reply to received emails using voice commands. Test this by receiving an email and replying to it with a voice command.

i. Test Case 9 - Email Deletion

Expected Result: The application should allow users to delete received or sent emails using voice commands. Test this by deleting received or sent emails using voice commands.

2. **Integration Testing**
   a. Test Case 1 - Detect new incoming emails

      Expected Result: The system extracts the text content of new emails as they arrive in the user's inbox.

   b. Test Case 2 - Convert the email's text content into an audio format.

      Expected Result: The system converts the email's text content into an audio format that is easily understandable by the user.

   c. Test Case 3 - Control the speed and volume of the speech output

      Expected Result: Users are able to control the speed and volume of the speech output through the system's user interface. Users are also able to pause, rewind, or skip the audio playback as needed through the system's user interface.

   d. Test Case 4 - Audio output is clear and natural-sounding

      Expected Result: The audio output is clear and natural-sounding and easily understandable by the user.

   e. Test Case 5 - Handle multiple user accounts

      Expected Result: The system can handle multiple user accounts and switch between them seamlessly, without requiring the user to log out and log back in.

3. **Performance Testing**
   a. Test Case 1 - Email Download Speed

      Expected Result: The application should download emails from the user's email provider in a timely manner.

   b. Test Case 2 - Response Time

      Expected Result: The application should respond to user input (e.g. voice commands) within acceptable limits and responds to user input within 2 seconds and load images and data within 5 seconds.

   c. Test Case 3 - Concurrent Users

      Expected Result: The application should be able to handle multiple users simultaneously without a significant decrease in performance.

   d. Test Case 4 - Load Testing

Expected Result: The application should be able to handle a high volume of traffic without a significant decrease in performance.

e. Test Case 5 - Stress Testing

Expected Result: The application should be able to handle a sudden increase in traffic without crashing or experiencing significant performance degradation.

f. Test Case 6 - Resource Utilization

Expected Result: The application should use system resources efficiently and not consume an excessive amount of memory or CPU usage.

4. **Security testing**

a. Test Case 1 - Authentication

Expected Result: The application should require users to authenticate with a strong password or other secure authentication methods before accessing their email accounts.

b. Test Case 3 - Encryption

Expected Result: The application should encrypt user data in transit and at rest to protect against unauthorized access or interception.

c. Test Case 5 - Session Management

Expected Result: The application should manage user sessions securely to prevent unauthorized access or session hijacking. It is tested by attempting to access an active session from a different device or location.

d. Test Case 6 - Error Handling

Expected Result: The application should handle errors and exceptions securely to prevent information disclosure or other security vulnerabilities.

e. Test Case 7 - Denial of Service (DoS) Attacks

Expected Result: The application should be able to handle high volumes of traffic and prevent DoS attacks that could result in a loss of service.

f. Test Case 8 - Vulnerability Scanning

Expected Result: The application should undergo regular vulnerability scanning to identify and address potential security vulnerabilities.

5. **Usability testing**

a. Test Case 1 - Navigation:

Expected Result: Users can easily navigate through the application using voice commands and are able to access all the features and options without confusion or difficulty.

b. Test Case 2 - Voice Recognition Accuracy:

Expected Result: The application should accurately recognize and transcribe user voice commands, even in noisy environments or with accents or speech impediments.

c. Test Case 3 - Feedback on User Actions:

Expected Result: The application should provide clear and timely feedback to the user when an action is performed, such as confirming the user's command, indicating the status of a process, or providing error messages when necessary.

d. Test Case 4 - Help and Documentation:

Expected Result: The application should provide clear and easy-to-understand help and documentation that is accessible through voice commands or the user interface.

e. Test Case 5 - Learnability:

Expected Result: The application should be easy to learn and use for users with varying levels of experience and technical knowledge.

f. Test Case 6 - Error Handling:

Expected Result: The application should provide clear and helpful error messages when something goes wrong, and users should be able to easily recover from errors.

g. Test Case 7 - Customization:

Expected Result: Users should be able to customize the application's settings, such as language, speech rate, or email display format, to suit their preferences.

h. Test Case 8 - Accessibility:

Expected Result: The application should be accessible to users with disabilities, such as visual or motor impairments, and provide alternative ways to interact with the system.

13.4 Testing Tools

We will use the following testing tools to automate and facilitate testing:

A. Screen Reader: Use a screen reader tool such as NVDA (Non-Visual Desktop Access) to ensure that the application can be navigated and used by blind users who rely on screen readers.
B. Speech Recognition Software: Use speech recognition software such as Dragon Naturally Speaking to ensure that the application accurately transcribes voice commands and inputs.
C. Accessibility Testing Tools: Use accessibility testing tools such as a colour contrast analyzer, keyboard-only navigation testing tool, and screen magnification software to ensure that the application meets accessibility standards for users with low vision or motor disabilities.
D. Test Automation Tools: Use test automation tools such as Selenium and Appium to automate repetitive testing tasks such as login, email composition, and email reading.
E. Load Testing Tools: Use load testing tools such as JMeter or LoadRunner to simulate a large number of users accessing the application simultaneously and verify that the system can handle the load without crashing or slowing down.
F. Security Testing Tools: Use security testing tools such as OWASP, ZAP to perform penetration testing and identify potential security vulnerabilities in the application.
G. User Feedback Tools: Use user feedback tools such as Usabilla or UserTesting to gather feedback from blind users on the application's usability, accessibility, and overall user experience.

13.5 Test Environment

The test environment for Echo will include the following hardware, software, and network configurations:

A. Hardware: The app will be tested on a variety of devices, including smartphones and tablets running on iOS and Android operating systems.
B. Software: The app will be tested on the latest versions of iOS and Android operating systems.

C. Network: The app will be tested on both cellular and Wi-Fi networks with different signal strengths to ensure that it can function properly under different network conditions.

13.6 Test Data

A. Test email accounts:

A set of test email accounts will be used to verify the functionality of the email setup feature. These email accounts will include both Gmail and Outlook accounts with varying authentication settings.

B. Sample emails:

A set of sample emails will be used to verify the email transcription, formatting, and sending features. These sample emails will include different types of content such as text, images, and attachments, and will be used to test the accuracy of the email transcription feature as well as the ability to edit and send emails using voice commands.

C. Noise data:

A set of noise data will be used to test the accuracy of the voice recognition feature in noisy environments. This data will include recordings of different types of noise such as traffic, background conversations, and music, and will be used to simulate the types of environments in which the app might be used.

D. Accent data:

A set of accent data will be used to test the accuracy of the voice recognition feature for different accents. This data will include recordings of people with different accents, including but not limited to American, British, Australian, and Indian accents.

E. Speech impediment data:

A set of speech impediment data will be used to test the accuracy of the voice recognition feature for people with speech impediments. This data will include recordings of people with different types of speech impediments, including but not limited to stuttering, lisps, and voice tremors.

13.7 Acceptance Criteria

The following acceptance criteria must be met for the application to be considered ready for release:

- All test cases have been executed and passed
- Performance benchmarks have been met, including page load times and response times
- Security requirements have been met, including data encryption and user authentication
- Usability metrics have been met, including user satisfaction and ease of use.
- Compatibility with different operating systems and devices
- Compatibility with different email providers (e.g., Gmail, Outlook, Yahoo)
- Compliance with accessibility guidelines
- Compliance with industry standards and regulations
- Integration with other assistive technologies

13. Maintenance Design

This section outlines the plan for maintaining the voice-based email application for visually impaired users. The purpose of this section is to provide a framework for the team responsible for maintenance to ensure the software remains up-to-date, secure, and reliable.

Maintenance Plan

The voice-based email application will undergo regular maintenance updates to ensure its continued functionality and usability. The maintenance team will be responsible for implementing these updates, which will occur on a quarterly basis. In the event of urgent updates, emergency maintenance may be performed as needed.

The following strategies will be implemented to ensure effective maintenance:

1. Regular updates: The application will be updated regularly to ensure that it remains compatible with the latest screen reader software and email providers. This will also include fixing any bugs that may arise and addressing user feedback.
2. Backward compatibility: The application will be designed to maintain backward compatibility with previous versions of the application to ensure that users who have not upgraded to the latest version can continue to use the application without interruption.
3. User feedback: A mechanism will be put in place to gather user feedback, which will be used to inform future updates and improvements to the application.
4. Data backup and recovery: Regular backups of user data will be performed to ensure that user data is not lost in the event of a system failure. A recovery plan will also be in place to restore user data in the event of a catastrophic failure.
5. Security updates: The application will be monitored regularly for security vulnerabilities, and updates will be made as necessary to ensure that the application remains secure and protects user data.
6. User training: As the application is updated and new features are added, user training materials will be updated to ensure that users are aware of these changes and know how to use the application effectively.

Change Management

Changes to the voice-based email application will be managed through a formal process that includes requesting changes, reviewing changes, and implementing changes. Change requests will be submitted through a designated ticketing system, which will be reviewed by the development team. Changes will be implemented following a thorough review and approval process to ensure the integrity of the software.

Version Control

The voice-based email application will use GitHub to track changes to the software system. The team responsible for maintenance will utilize this system to manage code changes, track revisions, and deploy updates. Any changes made to the software will be tracked, reviewed, and tested before being deployed to ensure the stability and reliability of the system.

Documentation

The voice-based email application will maintain up-to-date documentation, including user manuals, technical documentation, and release notes. The team responsible for maintenance will ensure that all documentation is accurate and up-to-date, providing end-users with the information they need to effectively use the software system.

14. Deployment Design

The mobile application will be deployed to the Google Play Store and Apple App Store, which will allow users to download and install the app on their Android and iOS devices. The deployment process involves publishing the app to the app stores and ensuring that it meets their respective requirements and guidelines.

To ensure optimal performance and user experience, the app will be regularly updated with bug fixes and new features. These updates will be pushed to the app stores, and users will be notified of the availability of the updates on their devices. The app will also include a built-in feedback mechanism to allow users to report bugs and suggest new features.

In terms of security, the app will employ industry-standard security practices to protect user data and prevent unauthorized access. This will include secure storage and transmission of user data, as well as strict access control measures for the app's backend services.

To ensure scalability and availability, the Echo app will be designed to work with a scalable cloud infrastructure. This will allow the app to handle increasing numbers of users and requests without compromising performance or availability. Additionally, the app's cloud infrastructure will be monitored regularly to ensure that it is operating optimally.

Finally, the deployment design will include a maintenance plan to ensure that the app continues to function as intended over time. This will involve ongoing monitoring of the app's performance and user feedback, as well as regular updates to ensure that the app remains compatible with new versions of the operating systems and other dependencies.

Appendix A: Record of Changes

Version	Date	Author	Description of change
1.0	30/03/2023	Salihu Ahamed	Added System and Application Architecture
1.0	30/03/2023	Ashwin Binu	Added Data and Algorithm Design
1.0	31/03/2023	Fathima Nooha	Started GUI Design and Testing Design
1.0	01/04/2023	Deepak P	Edited Component and Error Handling Design
1.0	01/04/2023	Fathima Nooha	Updated Maintenance Design
1.0	01/04/2023	Ashwin Binu	Edited Performance Design part
1.0	02/04/2023	Deepak P	Added Security Design
1.0	02/04/2023	Salihu Ahamed	Finalized document for submission

Appendix B: Acronyms

Acronym	Literal translation
API	Application Programming Interface
GUI	Graphical User Interface
NLP	Natural Language Processing
UI	User Interface
ESP	Email Service Provider
RDBMS	Relational Database Management System
ORM	Object-Relational Mapping
SQL	Structured Query Language
SMTP	Simple Mail Transfer Protocol
IMAP	Internet Message Access Protocol
GDPR	General Data Protection Regulation

Appendix C: Referenced Documents

Document Name	Document URL	Issuance Date
"Building Accessible Android Apps" by Google Developers	https://developer.android.com/guide/topics/ui/accessibility/principles	September 2021
"Best Practices for Developing Accessible Mobile Applications" by the World Wide Web Consortium (W3C)	https://www.w3.org/WAI/standards-guidelines/mobile/	December 2022
"Voice User Interface Design" by the Google Developers	https://developers.google.com/assistant/conversation-design	January 2023

APPENDIX III

FULFILLMENT OF COURSE OUTCOMES AND PROGRAM OUTCOMES

Fulfilling the Course Outcomes:

CO1. Identify technically and economically feasible problems

I. The project was chosen with a social cause of empowering the visually impaired population by helping them in their professional email chores.

CO2. Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes

I. A detailed online survey was taken up for our project.
II. On the basis of the surveys, the functional and non-functional requirements were established.

CO3. Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions of minimal complexity by using modern tools & advanced programming techniques

I. A detailed requirement analysis was performed and documented in the form of SRS Document.
II. Flutter built-in plugins were used in order to improve the functionality of the application.

CO4. Prepare technical report and deliver presentation

I. The technical report with all the requirements was structured and the presentation was also created.

CO5. Apply engineering and management principles to achieve the goal of the project

I. Waterfall Software model was used to achieve the goal of the project

Fulfilling the Program Outcomes:

PO1. Engineering Knowledge

Detailed knowledge in the field of computer science was used for this project, including the latest tech stack for mobile development: Flutter and the plugins associated with it. ML model was also implemented in the ML model.

PO2. Problem analysis

With the help of detailed online surveys, the key requirements for our target consumers were identified and according to that, the requirements were captured and written in the SRS Document.

PO3. Design/ development of solution

The wireframe of the mobile application was built using Figma. The main aim in this phase was to choose a simple design that is user-friendly for the customers.

PO4. Conduct investigation for complex problems

A detailed research was done to find an apt ML model for the application. The plugins and their pros and cons were also researched before using it. Various test cases were implemented to check the reliability of the mobile application.

PO5. Modern tool usage

The latest and most widely used technologies were used for the project. Flutter and its plugins were used to build the mobile application.

PO6. The engineer and the society

The project was chosen to contribute towards a relevant cause and help the disabled population to have self confidence and empower them.

PO7. Environment and sustainability

The mobile app is made open source and hence can be reused for any other purpose. The mobile application does not use mobile storage more than 13mb. Hence, it is light and does not require much space.

PO8. Ethics

No malpractices were performed during the development process. References contain the list of all websites that were referred for documentation and implementation of the project. None of the team members copied from other fellow batchmates. There was no kind of internal abuse or disputes that were not handled and hence the teammates implemented the project showcasing their professional ethics.

PO9. Individual and teamwork

Each individual was given tasks and milestone checks were conducted on a weekly basis. The team leader made sure that the milestones were met and the project went smoothly without much delay. Every member was ready to assist each other whenever in need of help, hence, a spirit of teamwork was always present among the members. There was proper communication and equal contribution from all members, based on their caliber and strengths.

PO10. Communication:

Established effective communication channels within the team, including regular offline meetings and online collaboration platforms. Ensured seamless coordination, shared progress updates, and addressed any project-related issues promptly.

PO11. Project Management and Finance:

Utilized a structured project management approach, considering Agile-Waterfall hybrid methodologies. Accounted for deployment and maintenance costs, including expenses associated with hosting the database and required tools. Ensured proper financial planning and resource allocation.

PO12. Life-long Learning:

Embraced a growth mindset and cultivated a culture of continuous learning. Adapted to new technologies, programming languages, and development practices throughout the project. Encountered challenges as learning opportunities and enhanced skills in web development and related domains

APPENDIX IV

PROJECT TIMELINE

Phase	Feb	Mar	Apr	May	Jun
Research and Planning					
Learning and UI/UX Design					
Development					
Testing and Deployment					