**صورة تحتوي على شعار, رمز, علامة تجارية, دائرة

تم إنشاء الوصف تلقائياً**

**Faculty of Computer and Information Technology**

**Department of Computer Engineering**

**CPE 591: Graduation Project II Report**

**[AI-Based interactive digital content application: search, summarize, save, translate, integrate, chat and present]**

**Noor Saleh Sharkawi 143673**

**Aya Amjad Mahmoud 145227**

**Supervised By**

**Dr.Lo'ai Tawalbah**

**Winter 2024-2025**

**Table of content:**

1. [Acknowledgment](#_2._Acknowledgment)
2. [Abstract](#_3._Abstract)
3. [List of Figures](#_4.1_List_of)
4. [List of Tables](#_4.2_List_of)
5. [Introduction](#_5._Introduction)
6. [Project Planning and Task Definition.](#_6.Project_Planning_and)
7. [Preliminary Design](#_Chapter_8:_Preliminary)
8. [Implementation](#_Chapter_10:_Implementation)
9. [Results and Discussion](#_Chapter_11:_Results)
10. [Conclusion](#_Chapter_12:_Conclusion)
11. [Recommendations](#_Chapter_13:_Recommendations)
12. [List of References](#_List_of_References)
13. [Appendices](#_Appendices)
14. [Meeting Minutes](#_Meeting_Minutes)

## 2. Acknowledgment

We would like to express our deepest gratitude to Lo'ai Tawalbah, our supervisor, for his invaluable guidance, continuous support, and all the feedback he has provided throughout this project.

We would like to express our gratitude to Jordan University of Science and Technology (JUST) and the Computer Engineering Department (CPE) for providing us with the resources and knowledge that made this project a success.

A special thanks to our families and friends for their unwavering encouragement and patience throughout this journey. Their support has been a constant source of motivation.

Lastly, we are grateful to everyone who contributed, directly or indirectly, to the success of this project. Your help and guidance have played a significant role in bringing our vision to reality.

## 3. Abstract

In today's digital world, managing and accessing content efficiently is important for students, researchers, and professionals. This project introduces an AI-powered digital content application that focuses on file management and document interaction, rather than just reading files. The app includes a smart chatbot and a summarization feature powered by the Gemini API, making it easier to work with documents. The frontend is built with Flutter for a smooth user experience, while the backend uses Python with Flask for reliable performance.

The application supports searchable PDFs, allowing users to locate key information with highlighted results. Translation functionality is provided through a simple widget similar to Google Translate. To enhance productivity, users can manage notes and to-do lists, all synchronized using Firebase for real-time data management.

Security and user authentication are handled via Firebase Authentication, while Firestore serves as the primary database for storing user-generated content. The implementation of bookmarks and a ‘Continue Reading’ feature ensures an enhanced user experience tailored to content engagement.

This project aims to provide a smart and efficient solution for digital content organization, catering to students and professionals alike. By integrating AI and cloud-based services, the system offers a powerful and intuitive platform that simplifies document management, enhances productivity, and improves information retrieval.

## 4.1 List of Figures

//This section lists all the figures (diagrams, charts, UI screenshots, etc.) included in your report, with their corresponding page numbers.

**//Code screen shots with simple explanation for each main component**

[Code explanation with screen shots](https://docs.google.com/document/d/1P39gE-7bQQiSIWBBgBiJjs6xkQKfUJrQ/edit?usp=sharing&ouid=104868842639778942632&rtpof=true&sd=true)

**//Aya here put user manual**

## 4.2 List of Tables

//Similar to the list of figures but for tables (e.g., database schemas, performance comparisons).

[Database structure](https://drive.google.com/file/d/1d7gkTXexy2ZKvx6vCcyS9JSS70Dj8194/view?usp=drive_link)

## 5. Introduction

### 5.1 Statement of the Problem

**/\*** **This section defines the problem your project aims to solve \*/**

In today's digital world, working with many documents can be difficult. Most file readers only show content but don’t help with productivity. Users waste time trying to find important information, summarizing long texts, or organizing notes. Existing tools also don’t offer smart AI help for summarizing, or managing documents in a simple way all in one place.

### 5.2 Significance of the Project

**/\*This section explains why your project is important.\*/**

This app is designed to make working with documents much easier and more efficient. It uses AI to help users quickly summarize long texts and find important information through search. All files are stored securely in the cloud using Firebase, so you can access them from any device. Unlike regular e-readers, it includes useful productivity tools like to-do lists, bookmarking, and a feature that remembers where you stopped reading. Students will find it helpful for research, while professionals can use it to manage work documents more effectively. By combining these smart features, the app saves time and makes digital content more accessible for everyone.

### 5.3 Goals

**This section outlines the objectives of your project.**

The primary goal of this project is to develop an AI-Based Interactive Digital Content Application that enhances document management and content accessibility. The specific objectives include:

1. AI Summaries  :the app uses Google’s Gemini AI to create short, easy to read summaries of long documents.
2. Smart Chatbot : an AI assistant ( Google’s Gemini AI)helps answer questions .
3. Fast PDF Search  Find keywords quickly with highlighted results in PDFs.
4. Translation Tool  Instantly translate text into different languages.
5. Cloud Sync save notes, bookmarks, and to-do lists in Firebase, and access them on any device.
6. Secure Login & Storage  Your data stays safe with Firebase’s authentication and cloud storage.
7. Reading convenience  the app remembers where you left off and lets you bookmark important pages.
8. Easy to use :design a simple, clean interface built with Flutter for smooth navigation.

### 5.4 Contemporary Issues

**This section discusses current challenges related to digital content management and how your project addresses them.**

1. Dealing with Too Much Information

Today, people have to read so many documents that it's hard to find the important parts. Our app solves this problem by using AI to create short, clear summaries. This helps users understand documents faster without reading everything.

2. Finding Information Quickly

Many document apps don't have good search tools, so users waste time looking for specific information. Our app makes searching easier by highlighting keywords in PDFs. When you search for something, you'll see exactly where it appears in the document.

3. Helping You Stay Organized

Most document readers don't have tools for taking notes or making to-do lists. This means people need to use different apps for different tasks. Our app includes built-in note-taking, bookmarks, and task lists so everything you need is in one place.

4. Smart AI Help When You Need It

Not many document apps use AI to help users. Our app includes a helpful chatbot that can answer questions about your documents. It uses Google's Gemini AI technology to give you smart, useful answers.

By solving these common problems, our app makes working with digital documents easier for students and professionals. It saves time and helps people stay organized.

### 5.5 Impact on Society

**This section explains how your project benefits users and society.**

1. Saves Time for Students and Researchers

The app's AI summaries and search help users find important information quickly. Instead of reading whole documents, they can focus on the key points.

1. Works for Everyone

With translation tools and easy search features, the app is helpful for people who speak different languages or have different needs.

1. Keeps You Organized

Built in notes, to do lists, and bookmarks mean users don't need multiple apps. Everything works together in one place.

1. Modern Digital Tools  
   The app uses AI and cloud technology to support today's learning and work needs. It helps users work in smarter, more digital ways.

Makes Information Easier to Understand

By removing extra details and showing only what's important, the app helps users think more clearly about their work.

### 5.6 Initial Constraints

**This section lists the limitations faced during development.**  
The project faced several initial constraints that influenced the design and implementation:

1. AI API Limitations : The Google Gemini API may have rate limits or restricted access, affecting performance.
2. Cloud Storage Costs : Firebase storage and Firestore database have cost implications as data usage increases.
3. Computational Complexity :AI summarization and chatbot responses require efficient processing to maintain real-time performance.
4. Platform Compatibility : Ensuring seamless integration between Flutter (frontend) and Flask (backend) required careful optimization.
5. User Interface Design Challenges :Balancing feature richness with UI simplicity was a key consideration.

Despite these constraints, strategic planning and optimization techniques help mitigate these challenges, ensuring a smooth and functional user experience.

## 6.Project Planning and Task Definition.

### 6.1 Task Identification

**This section will involve identifying the core tasks required to complete your graduation project. Here’s a possible breakdown of tasks for your E-File reader app:**

Task Breakdown:

1. Research and Design Phase:
   * Define Requirements: Clarify the app’s functionalities (file management, summarization, chatbot, etc.).
   * UI/UX Design: Sketch wireframes for the app, focusing on the user interface and user experience.
   * System Architecture Design: Define the technical architecture (how Python and Flask will interact with Flutter, Firebase usage, etc.).
2. Development Phase:
   * Setup Development Environment: Install and configure all necessary tools (Flutter, Firebase, Python, Flask, etc.).
   * Create Core App Structure: Build the basic structure for navigation, layout, and integration between Flutter and Python.
   * Backend Development (Flask & Firebase):
     + Implement API endpoints for file uploads, user authentication, and productivity tools (notes, to-do lists).
     + Integrate Firebase Authentication and Firestore.
   * Frontend Development (Flutter):
     + Develop the user interface based on the UI/UX design.
     + Integrate PDF handling using Flutter packages (searchable PDFs).
     + Implement the "Continue Reading" feature, bookmarks, and PDF uploads.
   * AI Features:
     + Integrate Gemini API for summarization and chatbot functionality.
     + Test and refine AI responses for the chatbot.
   * Localization (Translation): Implement the translation widget using ( translator package: Free Google Translate API for Dart)
3. Testing Phase:
   * Unit Testing: Test individual components of both the backend and frontend.
   * Integration Testing: Ensure that the app works seamlessly across all features (Firebase, AI, PDF handling).
   * User Testing: Conduct testing with a small group of users to gather feedback on the app’s usability.
4. Deployment Phase:
   * Prepare for Launch: Finalize the app for deployment on relevant platforms.
   * Deploy the App: Deploy the app and ensure that Firebase services are connected.
5. Documentation:
   * Technical Documentation: Document the code and explain the app's features, structure, and APIs.
   * Project Report: Begin writing the final report, including chapters on planning, design, implementation, and results.

### 6.2 Timeline

Week 1-2: Research & Design

* Define app requirements and core functionalities.
* Start UI/UX design (wireframes, flow).
* Plan system architecture and tech stack.
* Research Firebase and AI integration.

Week 3-5: Backend & Frontend Setup

* Setup development environment (Flutter, Python, Firebase).
* Design and implement basic structure of the app (Flutter navigation, initial screens).
* Set up Firebase Authentication and Firestore.
* Develop initial backend APIs in Flask.

Week 6-8: Core Functionality Development

* Implement file upload, note-taking, to-do list ,translation, features.
* Start integrating AI features (summarization, chatbot with Gemini API).
* Develop PDF handling features (searchable PDFs, highlighting, etc.).

Week 9-10: Testing

* Conduct unit and integration tests.
* Refine AI chatbot responses.
* User testing for feedback and usability.

Week 11-12: Finalizing & Documentation

* Finalize all features, ensuring smooth integration.
* Prepare the app for deployment (test on actual devices).
* Complete technical documentation.
* Begin writing the final project report.

**7. Literature Review**

**/\*The Literature Review chapter will provide an overview of existing research, applications, and technologies related to your app. The purpose is to justify your project by identifying gaps in current solutions and explaining how your app improves upon them.\*/**

**7.1 Introduction**

The literature review explores prior work work in four key areas:

1.Ai base content tools =>summarization and chatbots like Gemini API

2. Document management system like search , book mark , continue reading , pdf annotation

3.translation and multilingual support using Google translator package integration

4.productivity tools like note taking and making to do list

**7.2 AI-Based Summarization and chatbots**

Summarization tools and chatbots are rarely embedded in productivity apps and

Lack integration with document management so for chatbots (*Adamopoulou & Moussiades, 2020*)

The Gaps Addressed: out app integrates Gemini API for real-time summarization and chat

All in one place unlike isolated tools.

**7.3 Document Management and search**

Apps like Adobe Acrobat support annotatios/bookmarks but lack of continue reading feature

So the Innovation is combines file uploads and session saving

**7.4 Translation in Productivity tools**

Google Translation API is widely used but typically as standalone service Few apps embed it alongside

Notes taking or taks lists

So Improvement is direct integration for Google translator package allows in app translation without switching plateforms

**7.5Note Taking and Task Management**

These productivity tools are also rare to be alongside document management so the improvement

Is to make them all in one platform + saving them not at the hive but at the firecloud Firestore data base

**7.6 conclusion**

The review reveals that a market gap for Multifunctional productivity apps thatMerge AI(summarization and chatbot) ,document management ,task tracking.Existing tools are fragmented , requiring users to switch between platforms.This project brudges these gaps by offering a cross functional ai (Gemini) , translation , seamless document handling like upload search deleting countunes reading add bookmarks And unified workspace like notes and tasks

## Chapter 8: Preliminary Design

### 8.1 Concept

The app is designed to be a simple yet powerful tool for managing digital documents. It uses AI to help users work faster and more efficiently. For example, it can generate quick summaries of long PDFs, translate text into different languages, and even answer questions about documents through an AI chatbot. Additionally, users can take notes, create to-do lists, and search through their PDFs with ease. To keep everything secure and up to date, the app integrates with Firebase, which ensures real-time syncing across devices and safe login authentication. Overall, the goal is to provide an intuitive, all-in-one platform that makes handling documents smoother and more productive.

### 8.2 Concept Evaluation and Selection

Several design concepts were carefully reviewed to create the most effective and user-friendly app. Key factors like **user experience, performance, scalability, security, and accessibility** were all considered. The goal was to make the interface smooth and engaging while ensuring fast AI responses and quick PDF loading. The app also needed to handle large documents efficiently and keep data secure, which is why **Firebase Authentication** was chosen for safe user management. Additionally, features like translation make the app more accessible to people using different languages. After comparing different options, the final design uses **Google Gemini API for AI tools** and **Firebase for backend services**, as this combination delivers the best mix of functionality, reliability, and affordability.

### 8.3 Design Constraints

The app’s initial design had to work within several important limitations. First, **real-time performance** was crucial AI features like summarization and chatbot responses needed to be fast, with minimal delay. Second  **network dependency**, since features like AI summarization, chatbot, open e-file require an active internet connection to function. Finally, **data security** was a top priority, which is why the app uses **Firebase Authentication** and strict **Firestore database rules** to protect user privacy. These constraints helped shape the app’s development to ensure speed, efficiency, and security.

### 8.4 Applicable Codes and Standards Used in the Design

The design adheres to the following standards and codes:

* **IEEE 829-2008:** For software test documentation.
* **ISO/IEC 9126:** Ensuring software quality and usability.
* **OAuth 2.0:** For secure authentication using Firebase.
* **WCAG 2.1:** Maintaining accessibility standards for all users.
* **JSON API Standards:** For efficient data exchange between Flutter and the backend.

### 8.5 Preliminary Analysis

Before development, we conducted thorough testing to ensure the app performs well and meets user needs. We checked the **accuracy and speed of AI summarization** using Google Gemini API benchmarks. Firebase was load-tested to confirm it handles heavy usage without synchronization issues. The interface was optimized to run smoothly across different devices, from older models to latest smartphones. We also built strong error handling with detailed logging to quickly identify and fix any problems. Based on these tests, the app is on track to deliver a fast, reliable experience with accurate AI assistance that users will find intuitive and helpful.

**Chapter 9: Detailed System Design**

**9.1 Engineering Analysis and Simulation**

The detailed system design of the App underwent extensive engineering analysis to ensure performance, reliability, and user satisfaction. Various simulations and performance tests were conducted, focusing on:

* AI Model Accuracy: Validating the summarization and question-answering capabilities of the Google Gemini API on diverse datasets.

• System Load Simulation: Stressing the backend in Firebase to observe database response time when there is heavy user load.

• User Interface Performance: Monitoring Flutter's rendering performance to deliver smooth interactions and minimize latency.

• Error Recovery and Logging: Strong error-handling mechanisms coupled with large-scale log generation for debugging.

**9.2 Layouts, Drawings, Equipment Specifications**

**System Architecture Diagram:**

****

**Component Overview:**

* **Frontend:** Developed using Flutter, featuring widgets for document viewing, note taking , task making , translation , document uploading,
* **Backend:** Built using Python with Flask for managing API requests and responses.
* **Database:** Firebase Firestore for real-time data storage and synchronization.
* **AI Integration:** Google Gemini API for summarization and chatbot assistance.

**Layout Design:**

* **Home Screen:** Displays uploaded files
* **Drawer** to access all the tools(add file, summary,translation,notes,todolist)

**9.3 Applicable Codes and Standards Used in the Design**

The app adheres to the following standards and best practices:

* **ISO/IEC 25010:** Ensuring software quality and maintainability.
* **IEEE 830:** Standard for software requirements specifications.
* **ISO 27001:** Implementing information security best practices.
* **OAuth 2.0:** For secure authentication and authorization.
* **REST API Standards:** Ensuring seamless communication between the backend and external services.

**9.4 Economic Analysis**

The economic viability of the app is calculated by estimating development, deployment, and maintenance costs. Development costs include Flutter (frontend) and Python Flask (backend) development costs, integration fees for Google Gemini API, and Firebase services (auth and Firestore database). Cloud and hosting costs are measured as a function of Firebase Firestore storage consumption and frequency of API calls, which increase with user growth. Post-launch maintenance costs involve regular updates, bug fixes, and potential feature enhancements.

For its sustainability, the app's revenue model can be premium subscription (e.g., for AI advanced features) or in-app ads. The project is economically viable as long as the costs are monitored and optimized with Firebase's pay-as-you-go pricing model and affordable scalability. Any future development will be decided by user reviews and market trends to balance profitability and functionality.

## Chapter 10: Implementation

### 10.1 Construction

The construction phase of the App involved assembling the various components and ensuring seamless integration. The primary components constructed include:

1. **Frontend:** Developed using Flutter for a cross-platform experience on Android . Components such as PDF viewers, AI interaction screens, note-taking, task lists, translations interfaces were built using customizable widgets.
2. **Backend:** Implemented using Python and Flask to handle API requests, data processing, and interactions with external AI services.
3. **Database:** Firebase Firestore was set up for real-time data storage and synchronization.
4. **AI Integration:** Integrated Google Gemini API for summarization and chatbot features.

The construction phase followed an agile methodology, allowing iterative testing and development.

### 10.2 Programming

The development process included several important stages to create a functional and user-friendly application. First, **the frontend** was built using Flutter, focusing on a clean and responsive design. The team used state management techniques to ensure smooth performance and created modular components, making future updates and maintenance easier. For **the backend**, API endpoints were developed using Flask to enable communication between the frontend and backend systems. Proper error handling and logging were implemented to improve reliability and make troubleshooting simpler.

The AI features were integrated using the Google Gemini API, which provided accurate text summarization and chatbot functionality. The team carefully managed API requests and responses to ensure efficiency. Firebase was used for real-time data storage with Firestore, while Firebase Authentication ensured secure user access.

Finally, thorough testing was conducted, including unit tests for individual components and integration tests to verify that the frontend, backend, and AI services worked together seamlessly. This step was crucial in delivering a stable and high quality application.

### 10.3 Validation

The validation phase ensured that the application met both its functional and non-functional requirements. Several key tests were conducted to verify performance, security, and usability.

**1.Functional Testing** focused on core features, including PDF uploads, AI-generated summarization, AI-generated chatbot response, and note taking functionality. Sample documents were used to confirm the accuracy and reliability of the AI summaries.

**2.Performance Testing** evaluated the app’s responsiveness across different devices and network conditions.

**3.Security Testing** confirmed that Firebase Authentication worked correctly, protecting user data and preventing unauthorized access.

Finally, **User Acceptance Testing (UAT)** involved real users testing the app to provide feedback. Their input helped identify usability issues, leading to iterative improvements before the final release.

The results of these tests confirmed that the application performed as intended, meeting all design specifications and delivering a seamless user experience.

## Chapter 11: Results and Discussion

### 11.1 Summary of Goals Met by the Design

The application successfully achieved its core objectives, delivering a robust and user-friendly platform for digital document management. By integrating the Google Gemini API, the app provides accurate and concise AI-powered summarization, significantly enhancing productivity. The built-in chatbot further improves usability by offering instant, relevant responses to user queries, making document interaction more intuitive and efficient. Document management capabilities were a key focus, with seamless PDF upload, viewing, and searching functionalities. The app supports searchable PDFs, allowing users to quickly locate and highlight important keywords within their documents. This feature streamlines information retrieval, saving users valuable time and effort.

To boost productivity, the app includes integrated notetaking and to-do list features, all synchronized in real-time via Firebase. The "Continue Reading" and bookmark functions enhance the user experience by enabling effortless navigation and progress tracking across documents.

Security and data integrity were prioritized through Firebase Authentication, which safeguards user accounts, and Cloud Firestore, which ensures reliable real-time data management. These features work together to create a secure and dependable environment for handling sensitive documents.

**Overall, the application combines advanced AI capabilities with practical productivity tools, all wrapped in a cross-platform solution**. Future developments will focus on expanding AI functionalities and further optimizing performance to meet evolving user needs. The successful implementation of these features demonstrates the app's potential as a comprehensive document management solution.

### 11.2 Summary of Constraints and Codes Met by the Design

The design adhered to multiple constraints and standards to ensure quality, security, and usability.

**Constraints Met:**

* **Performance:** Achieved low-latency AI responses and smooth PDF rendering.
* **Security:** Firebase Authentication ensures user data security.
* **Storage Management:** Implemented efficient storage and management of user data using Firebase Firestore.
* **Network Dependency:** API requests were optimized to minimize network load and improve response time.

**Codes and Standards Met:**

* **ISO/IEC 25010:** Ensured software quality, usability, and maintainability.
* **IEEE 830:** Followed requirements documentation standards.
* **ISO 27001:** Applied security best practices to protect user data.
* **OAuth 2.0:** Used for secure authentication and authorization.
* **REST API Standards:** Ensured consistent communication between the backend and APIs.

## Chapter 12: Conclusion

In this project, we aimed to design and implement an AI-based interactive digital content application that enhances the e-reading experience through intelligent features. By integrating AI-powered summarization, a chatbot for user assistance, and additional productivity tools, the application seeks to provide a seamless reading and management experience.

During the development process, we encountered and overcame various challenges, including API integration, ensuring real-time data synchronization, and designing an intuitive user interface. By leveraging technologies such as Flutter for the front-end, Python with Flask for the back-end, and Firebase for data management, we were able to create a scalable and efficient application.

The AI features, particularly the summarization and chatbot functionalities powered by the Gemini API, added significant value by helping users quickly comprehend content and obtain relevant information. Additionally, the implementation of a translation widget, note-taking capabilities, and to-do lists further enriched the application’s usability.

Based on user feedback and testing, the application demonstrated effectiveness in simplifying content consumption and enhancing productivity. However, there is always room for improvement. Future iterations may include expanding language support for translation, improving chatbot responses, and adding offline capabilities.

In conclusion, the project successfully met its objectives by delivering a functional and user-friendly digital content management platform. It showcases the potential of AI in enhancing digital experiences and paves the way for further innovations in the e-reading domain.

## Chapter 13: Recommendations

Based on the insights gained during the development and testing of the AI-Based Interactive Digital Content Application, the following recommendations are proposed for future improvements and expansions:

1. **Enhance AI Capabilities**: Improve the accuracy and context-awareness of the AI-powered chatbot and summarization feature and chatbot . Integrating more advanced natural language processing (NLP) models can enhance the user experience.
2. **Expand Language Support**: Extend the translation widget to support a wider range of languages, making the application more accessible to a global audience.
3. **Implement Offline Mode**: Enable offline access for certain features, such as viewing previously summarized content, saved notes, and bookmarked PDFs. This will enhance usability in areas with limited connectivity.
4. **Advanced Search and Filter Options**: Incorporate more sophisticated search capabilities, including keyword suggestions and voice search, to make finding content easier.
5. **User Customization**: Provide options for users to customize the app’s appearance, font size, reading themes, and layout preferences to ensure a personalized reading experience.
6. **Multi-Platform Support**: Develop desktop and web versions of the application, ensuring a consistent user experience across devices.
7. **Security Enhancements**: Implement additional layers of security, including two-factor authentication (2FA) and advanced data encryption, to ensure user data remains protected.
8. **User Feedback Integration**: Establish a feedback loop within the app, allowing users to report issues and suggest features for future updates.
9. **Video summarization** :Make users not to summarize files only but also videos

By implementing these recommendations, the application can offer a more robust, engaging, and user-centric experience, driving further adoption and satisfaction among its users.

## List of References

1. Google AI. (2024). *Gemini API Documentation*. Retrieved from <https://ai.google.dev/gemini-api/docs>
2. Firebase. (2024). *Firebase Documentation*. Retrieved from <https://firebase.google.com/docs>
3. Flutter. (2024). *Flutter Documentation*. Retrieved from <https://flutter.dev/docs>
4. Python Software Foundation. (2024). *Python Documentation*. Retrieved from <https://docs.python.org/3/>
5. Flask. (2024). *Flask Documentation*. Retrieved from <https://flask.palletsprojects.com/>
6. Stack Overflow. (2024). *Developer Community Discussions*. Retrieved from [https://stackoverflow.com](https://stackoverflow.com/)

## Appendices

**Appendix A: Project Timeline**

* Detailed Gantt chart showing the project schedule, milestones, and task completion.(from moday)

**Appendix B: System Requirements**

* //List of hardware and software requirements used during the development and testing phases.

**1. Development Environment**

**-Hardware Requirements:**

- Processor: Intel Core i5 or equivalent (minimum)

- RAM:8GB (minimum) / 16GB (recommended for smoother performance)

- Storage:10GB of free space (for IDEs, SDKs, and dependencies)

- Operating System: Windows 10 pro/11,

- **Software Requirements**:

- Flutter SDK (Flutter --v 3.24.5)

- Dart SDK (bundled with Flutter)

- Android Studio (for Android emulation) (Android SDK version 35.0.1)

- Firebase CLI & Tools(for authentication and Firestore integration)

- Python 3.13.0 & Flask (for backend API development)

- Google Gemini API Key (for AI-powered features)

**2. Testing Environment**

- Physical Devices:

- Android: Devices running Android 11+ (e.g., Samsung Galaxy , Google Pixel 7a)

- Network Conditions:

- Wi-Fi 4G /4G+/5G

- Cloud Services:

- Firebase Project (for authentication, Firestore, and hosting)

- Google Cloud Platform (GCP) (for Gemini API access)

**3. Deployment Requirements**

- Mobile Platforms:

- Google Play Store (for Android APK)

- Apple App Store (for iOS IPA)

- Backend Hosting:

- Cloud Services: Firebase Hosting / Render (for Flask API)

## Meeting Minutes

Starting with the first meeting where the team decided to use Flutter, Python, and Firebase while organizing preliminary AI research, the project followed a structured development process. Gathering requirements, talking about UI preferences, and defining features like chatbot and summarization were the main topics of later meetings. Progress reviews during development focused on frontend and backend issues, especially UI improvement and API integration. User feedback from testing sessions improved AI response accuracy and usability. After conducting a thorough review, the team prepared documentation and presentation materials for submission and made final adjustments based on supervisor feedback. Clear action items were included in every phase, guaranteeing consistent advancement toward project completion.