



**TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING THAPATHALI
CAMPUS**

**A Minor Project Report
On
StudyBuddyyy - Your Personalized AI Learning Companion**

Submitted by:

Abinash Yadav [THA081BEI002]
Rupeshwar Sah [THA081BEI035]
Laxman Shakya [THA081BEI016]
Muhan Narshingh Shrestha [THA081BEI021]

Submitted to:

Department of Electronics and Computer Engineering
Thapathali Campus Kathmandu, Nepal

February, 2024

DECLARATION

We, the students who worked on the project “**StudyBuddyyy**”, are submitting this report to the **Department of Electronics and Computer Engineering at IOE, Thapathali Campus**. This project is one of the things we need to do to complete our first semester of **Bachelor of Engineering in Electronics and Communication Engineering**.

We confirm that this report is a real record of the work we did ourselves. We have not used this report to get any degree from any other university or college. This entire project and report are our own original work, and we have only used the resources that are listed in this report.

Abinash Yadav [THA081BEI002]

Rupeshwar Sah [THA081BEI035]

Laxman Shakya [THA081BEI016]

Muhan Narshingh Shrestha [THA081BEI021]

Date: March, 2025

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, a minor project work entitled “**StudyBuddyyy**” submitted by **Abinash Yadav, Rupeshwar Sah, Laxman Shakya** and **Muhan Narshingh Shrestha** in partial fulfillment for the award of Bachelor’s Degree in Electronics and Communication Engineering. The Project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of Bachelor’s degree of Electronics and Communication Engineering.

Project Supervisor
Prajwol Pakka,
Department of Electronics and Computer Engineering, Thapathali Campus

Er. Umesh Kanta Ghimire
Head of the Department,
Department of Electronics and Computer Engineering, Thapathali Campus
March, 2025

COPYRIGHT

The author has agreed that the library, Department of Electronics and Computer Engineering, Thapathali Campus, may make this report freely available for inspection. Moreover, the author has agreed that the permission for extensive copying of this project work for scholarly purpose may be granted by the professor/lecturer, who supervised the project work recorded herein or, in their absence, by the head of the department. It is understood that the recognition will be given to the author of this report and to the Department of Electronics and Computer Engineering, IOE, Thapathali Campus in any use of the material of this report. Copying of publication or other use of this report for financial gain without approval of the Department of Electronics and Computer Engineering, IOE, Thapathali Campus and author's written permission is prohibited.

Request for permission to copy or to make any use of the material in this project in whole or part should be addressed to department of Electronics and Computer Engineering, IOE, Thapathali Campus.

ACKNOWLEDGEMENTS

The successful development of the StudyBuddyyy project would not have been possible without the generous support and guidance of several individuals and resources. We would like to express our sincere gratitude to everyone.

We are particularly grateful to Professor Prajwal Pakka for his flexibility and understanding in allowing us to adopt a hybrid C and Python approach for StudyBuddyyy. His permission to leverage Python for the complex Artificial Intelligence and API handling tasks, despite the project's foundation in C, was crucial to its success. This strategic decision enabled us to effectively utilize Python's libraries and tools, enhancing both development efficiency and the overall functionality of StudyBuddyyy. Professor Pakka's open-mindedness and willingness to embrace innovative solutions have been truly inspiring and have greatly contributed to the technical capabilities and potential of this project.

Thank you, Professor Pakka, for your essential support and encouragement.

ABSTRACT

In today's educational landscape, many traditional study methods rely on a one-size-fits-all approach, providing static content that fails to address the unique learning styles of individual students. This lack of personalization often leads to information overload, disengagement, and inefficient learning processes. To tackle these challenges, we introduce StudyBuddyyy—a desktop application designed as a personalized AI learning companion. By leveraging advanced AI and large language models, StudyBuddyyy dynamically generates tailored study notes, interactive quizzes, concise YouTube video summaries, and visual mind maps that align with each student's unique needs. Integrating a user-friendly C-based interface with a robust Python-based content generation engine powered by the learning-toolbox library, StudyBuddyyy transforms traditional study methods into an engaging, adaptive, and efficient learning experience. This project outlines the system architecture, methodology, objectives, and expected deliverables that demonstrate its potential to revolutionize the way students learn by making education more personalized and effective.

Keywords: Educational Tool, Learning Assistant, learning-toolbox, Mind Map, Content Generation, Study Notes, YouTube Summarizer, Hybrid Application

TABLE OF CONTENTS

DECLARATION.....	i
CERTIFICATE OF APPROVAL	ii
COPYRIGHT	iii
ACKNOWLEDGEMENTS	iv
ABSTRACT.....	v
LIST OF FIGURE	viii
LIST OF ABBREVIATIONS	ix
1. INTRODUCTION.....	1
1.1 Background Introduction.....	1
1.2 Motivation.....	1
1.3 Problem Definition.....	1
1.4 Objectives.....	1
1.5 Project Scope and Applications	2
1.6 Report Organization	2
2. LITERATURE REVIEW.....	3
2.1 Traditional Learning Methods	3
2.2 Digital Learning Platforms and Tools.....	3
3. REQUIREMENT ANALYSIS.....	4
3.1 Operating System.....	4
3.2 C Compiler (GCC Recommended).....	4
3.3 Python 3.x	4
3.4 Python Package:.....	4
3.5 Node.js and npm (Node Package Manager)	4
3.6 Internet Connectivity	4
4. SYSTEM ARCHITECTURE AND METHODOLOGY	5
5. IMPLEMENTATION DETAILS	12
5.1 Core Application Structure (C Modules).....	12
5.2 Python API Handler (api_handler.py).....	16
6. RESULTS AND ANALYSIS.....	18
6.1 Content Generation Results (Study Notes and Quizzes)	18
6.1.1 Study Notes Example and Analysis	18
6.1.2 Quiz Example and Analysis	19
6.2 YouTube Summarization Results.....	20

6.3 Mind Map Generation Results.....	21
6.4 Error Analysis	22
7. FUTURE ENHANCEMENTS	23
8. CONCLUSION	25
9. APPENDICES	26
Appendix A: User Manual	26
Appendix B: Running StudyBuddyyy.....	28

LIST OF FIGURE

Figure 1 Block Diagram of Study Buddy	5
Figure 2 Flowchart of User Interaction and Content Generation	9
Figure 3 Core Application Structure	12
Figure 4 Generated Note	18
Figure 5 Generated Quiz.....	19
Figure 6 Generated Summary	20
Figure 7 Generated Mind Map.....	21
Figure 8 Authentication Menu	28
Figure 9 Main Menu	29
Figure 10 Topic Input.....	29
Figure 11 Select Content Type	30
Figure 12 Generated Note	30
Figure 13 Generated Quiz.....	30
Figure 14 Youtube Summary	31
Figure 15 Mind Map.....	31
Figure 16 File Structure	32

LIST OF ABBREVIATIONS

API	Application Programming Interface
UI	User Interface
MD	Markdown
PDF	Portable Document Format
LLM	Large Language Model

1. INTRODUCTION

1.1 Background Introduction

Traditional study methods relying on textbooks, lectures, and static resources have long provided a foundational approach to learning. However, these methods typically follow a one-size-fits-all strategy, often ignoring the unique learning styles and needs of individual students. With the emergence of large language models (LLMs) and AI-driven technologies, there is now an opportunity to transform education by personalizing learning experiences. StudyBuddyyy is designed to harness these advancements to create a tailored educational tool that moves beyond the limitations of traditional methods.

1.2 Motivation

Modern education demands flexibility and adaptation to individual learning preferences. Students often struggle with engagement and retention when faced with generic study materials that do not cater to their unique strengths or address their specific weaknesses. The motivation behind StudyBuddyyy is to bridge this gap by providing a personalized learning companion. By leveraging AI, this application generates study notes, quizzes, video summaries, and mind maps that are customized to each student's requirements, ultimately enhancing both comprehension and academic performance.

1.3 Problem Definition

The core issue lies in the one-dimensional approach of traditional study methods, which lack personalization. Standard textbooks and lectures offer the same content to every student regardless of their learning style or pace, leading to disengagement and ineffective learning. StudyBuddyyy addresses this problem by using advanced AI to dynamically generate personalized learning materials. The application adapts to individual topics and learning needs, ensuring that every student receives content that is both relevant and engaging.

1.4 Objectives

The primary objectives of the project are:

- **Personalization:** Develop an AI-powered tool that creates customized study materials tailored to each learner's unique needs.

- **Enhanced Engagement:** Increase student engagement and retention through adaptive content that matches individual learning styles.
- **Efficiency:** Streamline the study process by automating the generation of notes, quizzes, video summaries, and mind maps, thereby reducing the time and effort required for traditional studying.

1.5 Project Scope and Applications

Study Buddy is designed as a desktop application, making it easily accessible to students on their personal computers. Its scope is centered around providing core study assistance features: generating learning content, summarizing videos, and creating mind maps. The application is intended for individual students looking to enhance their study efficiency across various subjects. It can be particularly useful for:

- **Personal Study:** Students can use it to generate study materials for any topic they are learning, whether for school, college, or personal development.
- **Exam Preparation:** The quiz generation feature helps students test their knowledge and identify areas they need to focus on before exams. The summarized notes and mind maps can also serve as quick revision tools.
- **Quick Content Review:** For students who need to quickly understand the content of a YouTube video or get a summary of a topic, Study Buddy provides fast and efficient solutions.

1.6 Report Organization

This report is structured to give you a clear picture of the Study Buddy project. Following this introduction, the report would typically delve into:

- **Features:** A detailed look at each feature of Study Buddy, explaining how they work and how users can benefit from them.
- **System Design and Architecture:** An overview of the technical design of the application, including how the different components interact with each other (like the C application and the Python API handler).
- **Implementation Details:** A more technical section discussing the programming languages used, key algorithms, and how the features were actually built.
- **Testing and Evaluation:** Information on how Study Buddy was tested to ensure it works correctly and effectively.
- **Conclusion and Future Work:** A summary of the project's achievements, its limitations, and potential ideas for future improvements and expansions.

2. LITERATURE REVIEW

2.1 Traditional Learning Methods

Traditional educational approaches such as classroom lectures, textbooks, and standardized study guides have served as the cornerstone of learning for decades. However, these methods are inherently static and uniform. They often lack the flexibility needed to cater to different learning styles, leaving many students feeling disengaged or overwhelmed. This rigid approach fails to accommodate the unique pace and preferences of individual learners, highlighting the critical need for more adaptive and personalized study tools.

2.2 Digital Learning Platforms and Tools

Modern digital learning platforms have begun to introduce interactive and flexible learning experiences. Despite these advances, many still offer only limited personalization, often presenting pre-packaged content without adaptive customization. While digital tools provide improved multimedia experiences and interactivity, they frequently fall short of delivering content that truly adapts to the needs of each student. StudyBuddy fills this gap by integrating advanced AI and large language models to dynamically generate personalized learning materials—ensuring that every study session is uniquely tailored to the learner’s needs.

3. REQUIREMENT ANALYSIS

This section outlines the essential software components necessary for running and utilizing Study Buddy effectively. Understanding these requirements is crucial for ensuring the application functions as intended and is accessible to its target users.

3.1 Operating System: Windows (7, 8, 10, or 11) is the primary target operating system, as indicated by the project setup instructions and use of Windows-specific headers (windows.h).

3.2 C Compiler (GCC Recommended): A C compiler, preferably GCC (GNU Compiler Collection), is necessary to compile the C source code files (.c files) into an executable application (study_buddy.exe).

3.3 Python 3.x: Python 3.x must be installed and accessible in the system's PATH. Python scripts (api_handler.py) are used to handle API interactions for content generation, YouTube summarization, and mind map creation.

3.4 Python Package: learning-toolbox (version 0.2.0): This specific Python package is a dependency, and it needs to be installed using `pip install learning-toolbox==0.2.0`.

3.5 Node.js and npm (Node Package Manager): Node.js and npm are required, specifically for the mind map generation feature, and @mermaid-js/mermaid-cli package needs to be installed globally using `npm install -g @mermaid-js/mermaid-cli`.

3.6 Internet Connectivity: An active internet connection is essential for accessing external APIs (like the Gemini API through learning-toolbox) to generate content, summarize YouTube videos, and potentially for mind map generation if it also relies on online services.

4. SYSTEM ARCHITECTURE AND METHODOLOGY

Study Buddy employs a modular system architecture designed for clarity, maintainability, and efficient operation. This chapter details the system's block diagram, explaining each component and their interactions. Furthermore, we will illustrate the project's methodology using flowcharts to visualize the user interaction and content generation processes. This provides a comprehensive understanding of how Study Buddy is structured and how it functions to deliver its features.

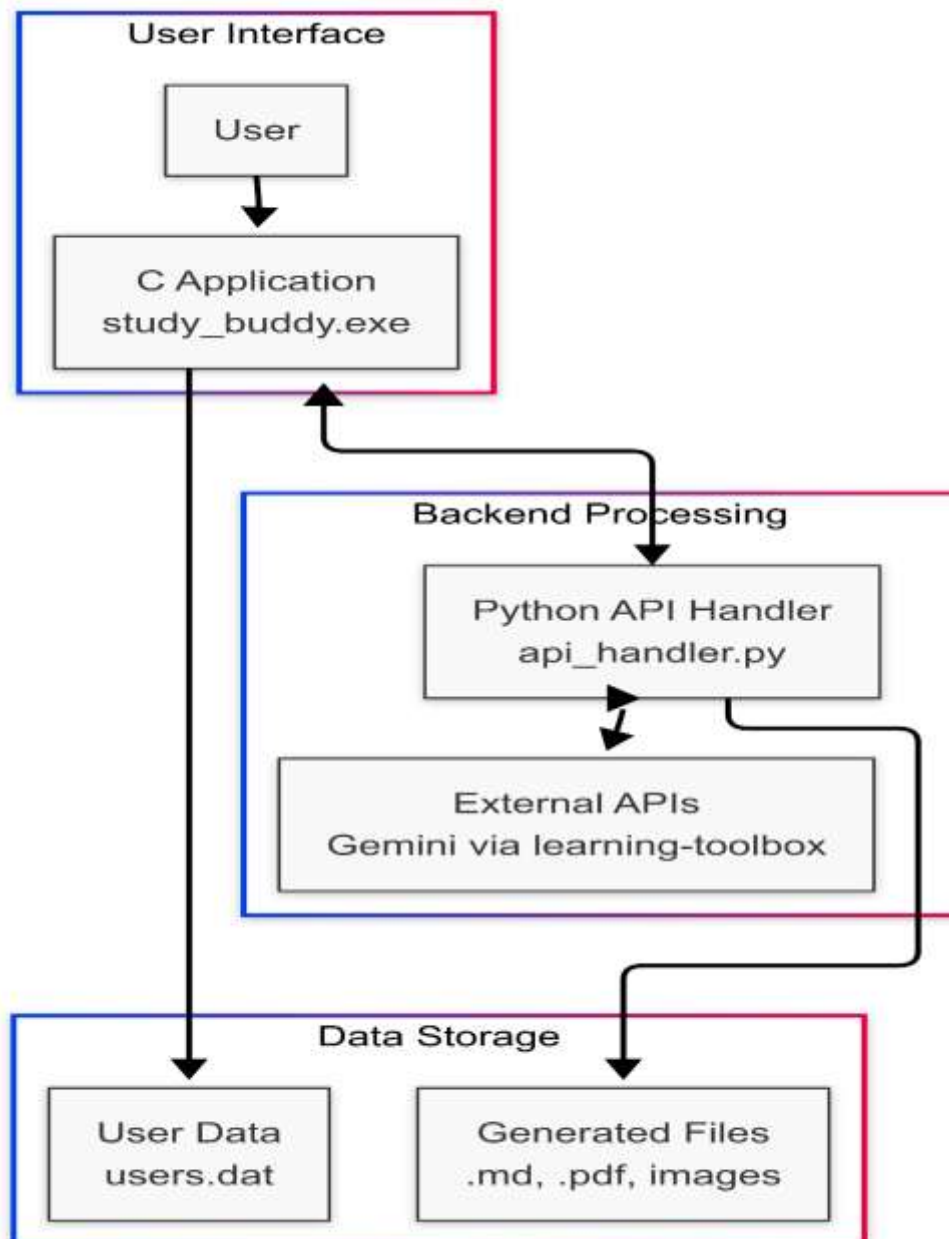


Figure 1 Block Diagram of Study Buddy

Figure 4.1 illustrates the system architecture of Study Buddy. The system is broadly divided into several key blocks, each with a specific purpose, working in concert to deliver the application's functionality:

- **User:** This represents the end-user interacting with the Study Buddy application. The user provides input through the command-line interface and receives output in the form of text, generated files, and visual displays.
- **C Application (study_buddy.exe):** This is the core of the Study Buddy application, written in C and compiled into an executable. It serves as the **User Interface and Core Logic** layer.
 - **Purpose:**
 - **User Interface:** Provides a text-based command-line interface (CLI) for user interaction. It displays menus, prompts for input, and shows output messages.
 - **Core Logic & Control Flow:** Manages the overall application flow, including authentication, menu navigation, feature selection, and calling relevant modules.
 - **Feature Modules:** Integrates specific feature modules implemented in C, such as:
 - **Authentication (auth.c):** Handles user registration and login, managing user credentials stored in users.dat.
 - **Notes & Quiz Generation (notes_quiz.c):** Manages user requests for study notes and quizzes, interacting with the Python API handler.
 - **YouTube Summarizer (youtube_summarizer.c):** Handles user requests for YouTube video summaries, interacting with the Python API handler.
 - **Mind Map Generator (mindmap_generator.c):** Manages user requests for mind map generation, interacting with the Python API handler and potentially external tools for rendering.
 - **Interaction:**
 - Receives user input from the command line.
 - Calls functions within its own modules (e.g., login_user() from auth.c).
 - Executes system commands to invoke the Python API handler (api_handler.py), passing user-provided parameters (topic, content type, YouTube URL).
 - Displays output to the user on the command line.
 - Reads and writes data to users.dat for user authentication.

- **Python API Handler (api_handler.py):** This Python script acts as a bridge between the C application and external services, specifically the AI APIs for content generation.
 - **Purpose:**
 - **API Interaction:** Handles communication with external APIs, likely the Gemini API via the learning-toolbox library.
 - **Data Processing:** Processes user requests received from the C application, prepares data for API calls, and processes the API responses.
 - **Content Generation & Summarization:** Utilizes the learning-toolbox library to generate study notes, quizzes, YouTube summaries, and mind map data by interacting with the AI model.
 - **File Saving:** Calls functions from learning-toolbox (like save_files) to save generated content in Markdown, (or image) and PDF formats.
 - **Interaction:**
 - Receives commands and parameters from the C application via command-line arguments (sys.argv).
 - Utilizes the learning-toolbox library and the provided API key to communicate with the external AI API.
 - Receives API responses containing generated content.
 - Saves generated content to files in the file system.
 - Prints content to stdout
- **External APIs (Gemini API via learning-toolbox):** This represents the external AI service that provides the core content generation and summarization capabilities. In this project, it is the Gemini API accessed through the learning-toolbox Python package.
 - **Interaction:**
 - Receives API requests from the Python API handler, including topics, content types, and API keys.
 - Processes these requests and returns API responses containing the generated content (text, quiz questions, mind map data).
- **Data Storage (users.dat):** This file is used for local storage of user authentication data.
 - **Interaction:**
 - Accessed by the auth.c module for user registration and login.
 - Data is read and written in binary format using file I/O operations in C.

- **File System (Generated Files: .md, .pdf, images):** The local file system is used to store all generated study materials.
 - **Purpose:**
 - **Persistent Storage:** Provides persistent storage for study notes, quizzes, YouTube summaries (in Markdown and PDF formats), and mind maps (as images).
 - **User Access:** Allows users to access and review the generated study materials after they are created.
 - **Interaction:**
 - The Python API handler saves generated files to the file system, in the application's working directory.
 - Users can access these files from folder name 'output_resources'.

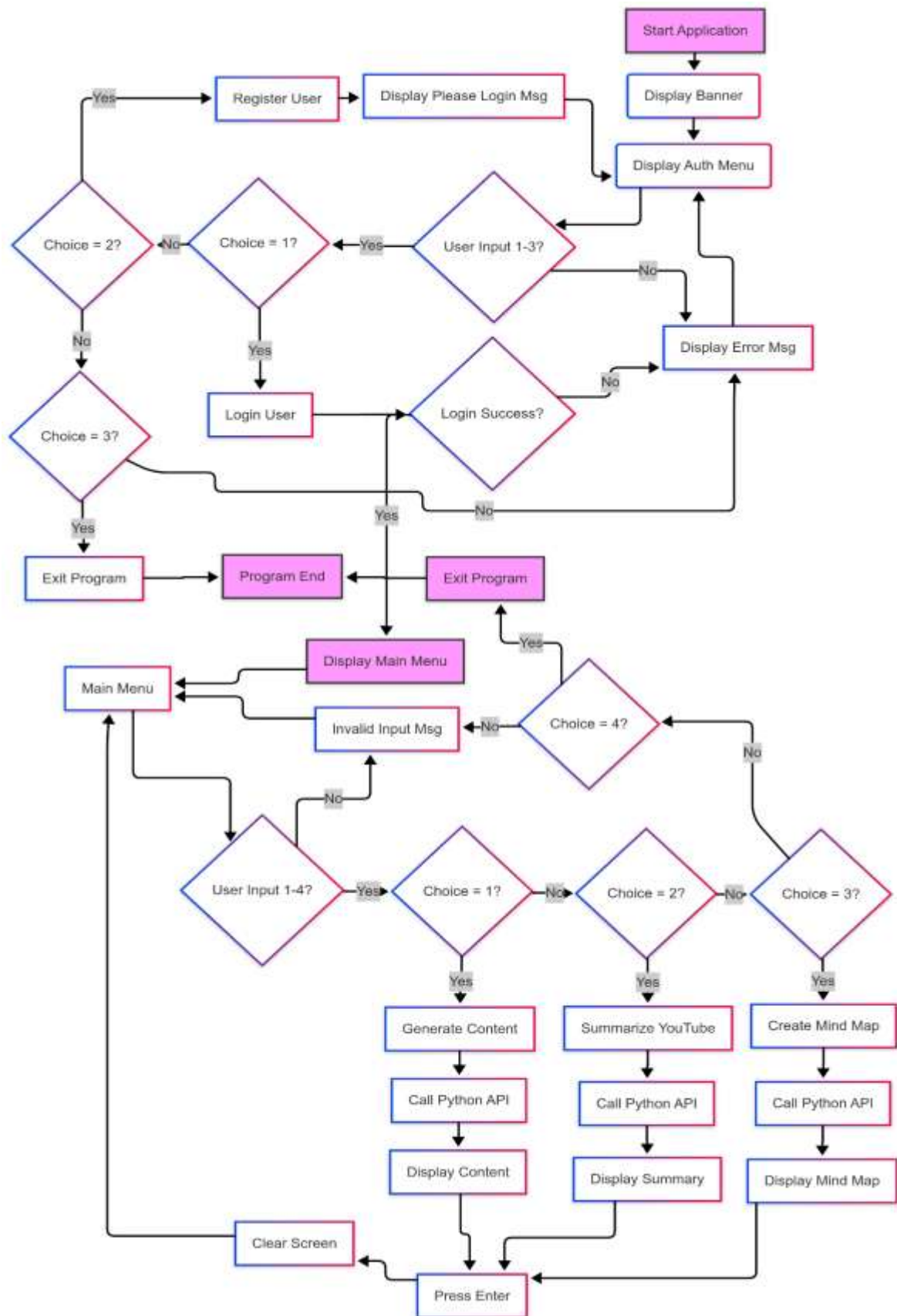


Figure 2 Flowchart of User Interaction and Content Generation

Figure 4.2 presents a flowchart illustrating the user interaction and content generation workflow in Study Buddy. This flowchart outlines the steps a user takes when interacting with the application, from initial startup to using its core features.

Flowchart Explanation:

1. **Start Application:** The application begins execution with `study_buddy.exe`.
2. **Display Banner:** The application first displays a welcome banner using the `banner.c` module and `printBanner()` function, enhancing the user experience with a visual introduction.
3. **Display Auth Menu:** The authentication menu is displayed, offering options for Login, Register, or Exit. This menu is generated by the `app.c` module and `printAuthMenu()` function.
4. **User Input (1-3)?:** The application prompts the user to choose an option from the authentication menu (1 for Login, 2 for Register, 3 for Exit).
5. **Choice Handling (Authentication):**
 - **Choice = 1 (Login):** Calls the `login_user()` function from `auth.c` to handle user login. Upon successful login, the application proceeds to the main menu. If login fails, an error message is displayed, and the user is returned to the authentication menu.
 - **Choice = 2 (Register):** Calls the `register_user()` function from `auth.c` to handle user registration. After successful registration, a success message is shown, and the user is prompted to login.
 - **Choice = 3 (Exit):** Exits the application gracefully.
6. **Logged In?:** After attempting login, the application checks if the login was successful. If yes, the application moves to the main menu; otherwise, it displays an error message and loops back to the authentication menu.
7. **Display Main Menu:** Once logged in, the main menu is displayed, offering options for Generate Learning Content, Summarize YouTube Video, Create Mind Maps, or Exit. This menu is generated by `app.c` and `printMenu()` function.
8. **User Input (1-4)?:** The application prompts the user to choose an option from the main menu (1-4).
9. **Choice Handling (Main Menu):**
 - **Choice = 1 (Generate Content):** Calls the `generate_content()` function from `notes_quiz.c`. This function prompts for a topic and content type (notes or quiz), and then calls the Python API handler.
 - **Choice = 2 (Summarize YouTube):** Calls the `summarize_youtube_video()` function from `youtube_summarizer.c`. This function prompts for a YouTube video URL and then calls the Python API handler.

- **Choice = 3 (Create Mind Map):** Calls the `create_mindmap()` function from `mindmap_generator.c`. This function prompts for a topic and then calls the Python API handler.
 - **Choice = 4 (Exit):** Exits the application with a thank you message.
10. **Call Python API (`api_handler.py`):** For choices 1, 2, and 3, the respective C functions execute a system command to run the `api_handler.py` script, passing necessary parameters (topic, content type, URL).
 11. **Display/Save Generated Content/Summary/Mind Map:** After the Python script executes and generates content (or retrieves a summary or mind map data via API), the results are typically saved to files (Markdown, PDF, image). The C application might display a confirmation message or minimal output to the console, but the primary output is the saved files. (Note: In the provided code, the C application doesn't seem to directly process or display the content printed by the Python script to stdout, it mainly triggers the Python script execution).
 12. **Press Enter to Continue:** After a feature is executed, the application prompts the user to press Enter to continue, allowing them to review output or prepare for the next action.
 13. **Clear Screen:** The screen is cleared to prepare for the next menu display, maintaining a clean user interface.
 14. **Back to Main Menu:** The application loops back to display the main menu, allowing the user to choose another feature or exit.

Algorithms and Methods:

- **System Calls for Python Execution:** Study Buddy uses system calls (via `system()` function in C) to execute the Python script `api_handler.py`. This is a simple method for integrating functionalities across different programming languages in a desktop application.
- **Text-Based User Interface:** The application employs a straightforward text-based menu system for user interaction. This approach is efficient for command-line applications and keeps the application lightweight.
- **File-Based Data Storage:** User credentials and generated study materials are stored in local files (`users.dat` and generated content files). This simplifies data management for a standalone desktop application but introduces security considerations (especially with plain text passwords).
- **Modular Design:** The separation of functionalities into C modules (for UI and core logic) and Python scripts (for API interaction and content generation) promotes modularity and makes the codebase easier to understand and maintain.

This chapter has provided a detailed overview of Study Buddy's system architecture and methodology, using a block diagram and flowchart to illustrate its structure and operational flow. This understanding is crucial for comprehending how the application functions and how its components interact to deliver the intended features.

5. IMPLEMENTATION DETAILS

This chapter delves into the nuts and bolts of Study Buddy, explaining how each software component is implemented and functions. We'll explore the inner workings of each module, the protocols they use to communicate, and how they all come together to create a fully functional study assistant. Understanding these implementation details will provide a clearer picture of how Study Buddy achieves its objectives.

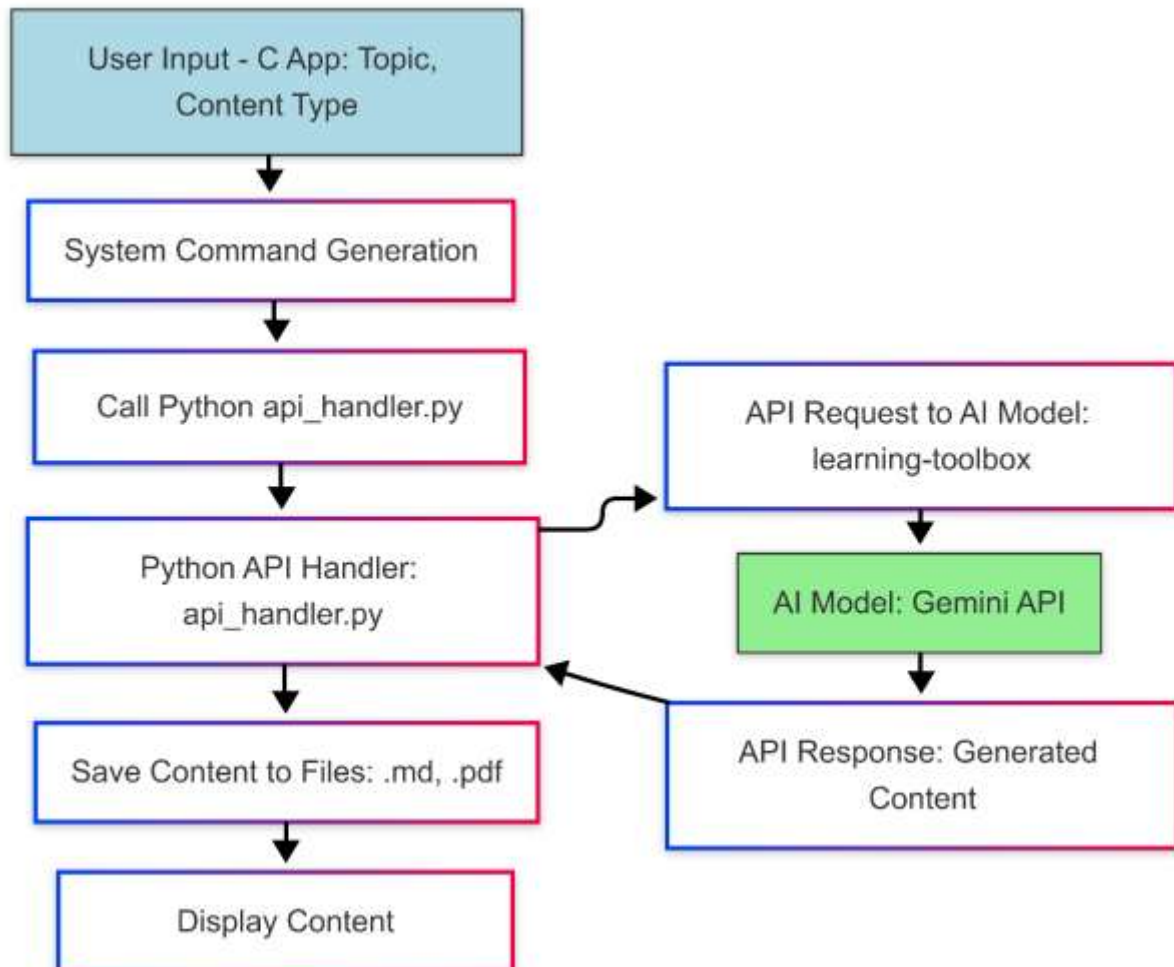


Figure 3 Core Application Structure

5.1 Core Application Structure (C Modules)

The heart of Study Buddy is built using C programming language, chosen for its efficiency and system-level capabilities. The C code is organized into several modules, each responsible for a specific part of the application's functionality.

- **app.c - The Main Application and User Interface:** This is the entry point of the Study Buddy application. It's like the conductor of an orchestra, directing all the other parts.
 - **Functionality:**
 - **main() function:** This is where the program execution begins. It initializes the application, sets up the character encoding for proper text display (`system("chcp 65001")`), and manages the main program loop.
 - **User Authentication Flow:** It controls the initial authentication process, presenting the login/register menu (`printAuthMenu()`), taking user input, and calling functions from `auth.c` (`login_user()`, `register_user()`) to handle authentication.
 - **Main Menu Navigation:** After successful login, `app.c` displays the main menu (`printMenu()`), allowing users to choose between generating content, summarizing YouTube videos, creating mind maps, or exiting. It takes user input for menu choices and uses a switch statement to direct the program flow to the appropriate feature.
 - **Feature Module Invocation:** Based on the user's choice in the main menu, `app.c` calls functions from other C modules (`notes_quiz.c`, `youtube_summarizer.c`, `mindmap_generator.c`) to initiate the requested feature.
 - **Screen Management:** Uses `clearScreen()` to clear the console for a cleaner user interface and `printBanner()` to display the application banner at the start and in menus.
 - **Protocols/Methods:**
 - **Menu-Driven Interface:** Uses `printf()` and `scanf()` functions for basic text-based menu interaction with the user.
 - **Function Calls:** Relies heavily on function calls to other C modules to delegate specific tasks.
 - **switch statement:** Uses switch statements to handle menu choices and direct program flow.
 - **System Calls:** Uses `system()` calls to execute external commands, specifically to invoke the Python script (`api_handler.py`).
- **auth.c and auth.h - User Authentication:** These files manage user registration and login functionalities.
 - **Functionality (auth.c):**
 - **register_user():** Handles new user registration. It prompts the user for a username and password, checks if the username already exists

(using `user_exists()`), and if not, appends the new user's information to the `users.dat` file.

- **login_user():** Handles user login. It prompts for a username and password, reads user data from `users.dat`, and checks if the entered credentials match a registered user.
- **user_exists():** A helper function that checks if a given username already exists in the `users.dat` file.
- **Data Storage (users.dat):**
 - Stores user data in binary format. Each user record is a `User` struct containing username and password (currently in plain text - a security concern).
 - Data is appended during registration and read during login.
- **Protocols/Methods:**
 - **File I/O:** Uses standard C file I/O functions (`fopen()`, `fread()`, `fwrite()`, `fclose()`) to interact with the `users.dat` file in binary mode ("rb", "ab").
 - **String Comparison:** Uses `strcmp()` to compare usernames and passwords.
 - **Struct (User):** Defines a `User` structure in `auth.h` to organize user data, making it easier to read and write user records to the data file.
- **notes_quiz.c and notes_quiz.h - Generate Notes and Quizzes:** These modules handle the "Generate Learning Content" feature.
 - **Functionality (notes_quiz.c):**
 - **generate_content():** This function is called from `app.c` when the user chooses option 1 from the main menu.
 - **Topic and Content Type Input:** It prompts the user to enter a topic they want to learn about and then asks them to choose between "Study Notes" or "Quiz".
 - **Python API Handler Invocation:** Constructs a system command to execute `api_handler.py`, passing the topic and content type as command-line arguments. For example: `python api_handler.py "Topic Entered by User" notes` or `python api_handler.py "Topic Entered by User" quiz`.
 - **Protocols/Methods:**
 - **User Input:** Uses `printf()` and `scanf()` to get topic and content type choices from the user.
 - **System Calls:** Uses `system()` to execute the Python script.

- **String Manipulation:** Uses strcpy() to set the content_type string based on user choice.
- **youtube_summarizer.c and youtube_summarizer.h - YouTube Video Summarization:** These modules implement the "Summarize YouTube Video" feature.
 - **Functionality (youtube_summarizer.c):**
 - **summarize_youtube_video():** Called from app.c when the user selects option 2 from the main menu.
 - **YouTube URL Input:** Prompts the user to enter a YouTube video URL.
 - **Python API Handler Invocation:** Creates a system command to run api_handler.py, passing the YouTube URL and content type "youtube_summary" as arguments. For example: python api_handler.py "YouTube URL Entered by User" youtube_summary.
 - **Protocols/Methods:**
 - **User Input:** Uses printf() and scanf() to get the YouTube URL from the user.
 - **System Calls:** Uses system() to execute the Python script.
 - **String Manipulation:** Uses snprintf() to safely construct the system command string.
- **mindmap_generator.c and mindmap_generator.h - Mind Map Generation:** These modules are responsible for the "Create Mind Maps" feature.
 - **Functionality (mindmap_generator.c):**
 - **create_mindmap():** Called from app.c when the user chooses option 3 from the main menu.
 - **Topic Input:** Prompts the user to enter a topic for the mind map.
 - **Python API Handler Invocation:** Constructs a system command to execute api_handler.py, passing the topic and content type "mindmap". For example: python api_handler.py "Topic Entered by User" mindmap.
 - **Protocols/Methods:**
 - **User Input:** Uses printf() and scanf() to get the topic from the user.
 - **System Calls:** Uses system() to execute the Python script.
 - **String Manipulation:** Uses snprintf() to safely construct the system command string.
- **Utility and UI Enhancement Modules:**

- **banner.c, banner.h:** Provides the `printBanner()` function to display the application's banner.
- **colors.c, colors.h:** Provides `setColor()` function to change the text color in the console, enhancing the visual appeal of the text-based interface. It uses Windows-specific API (`SetConsoleTextAttribute`).
- **config.h:** Defines configuration constants like `USER_DB` (filename for user database) and color codes (`COLOR_ERROR`, `COLOR_SUCCESS`, `COLOR_RESET`) for consistent styling.
- **utils.h:** Declares utility functions like `setColor()` and `clearScreen()`, used across different C modules.

5.2 Python API Handler (`api_handler.py`)

The `api_handler.py` script, written in Python, acts as the bridge to external AI services. It's called by the C application to handle content generation tasks.

- **Functionality (`api_handler.py`):**
 - **Command-line Argument Parsing:** Receives topic and content type (or YouTube URL and "youtube_summary", or topic and "mindmap") as command-line arguments via `sys.argv`.
 - **API Key Management:** Initially, it directly includes the `api_key` within the script. In a more secure setup, this should be handled differently (e.g., environment variables).
 - **Content Generation using learning-toolbox:**
 - Imports `generate_content` and `save_files` functions from the `learning-toolbox` library.
 - Calls `generate_content(topic, content_type, api_key=api_key)` to request content generation from the AI model (likely through the Gemini API, based on the placeholder key).
 - The `learning-toolbox` library handles the details of API communication and content retrieval.
 - **File Saving:**
 - Calls `save_files(topic, content, content_type, base_dir="")` from `learning-toolbox` to save the generated content as Markdown (.md) and PDF (.pdf) files in the application directory.
 - **Output to Console (stdout):**

- `print(content)`: Prints the generated content to the standard output (stdout). **However, in the current C application, this output is not actively processed or displayed by the C code.** It might be intended for debugging or future feature enhancements.
- **Protocols/Methods:**
 - **Command-line Arguments:** Receives input parameters via `sys.argv`.
 - **Function Calls (learning-toolbox):** Uses functions from the learning-toolbox library to interact with the AI API and save files.
 - **Standard Output (stdout):** Prints generated content to stdout.

5.2.1 Interconnection of Components

All these components work together in a coordinated manner to provide Study Buddy's functionality:

1. **User Interaction via app.c:** The user interacts with the text-based menus provided by `app.c`. They choose features, enter topics, URLs, and authentication details through the command-line interface.
2. **Feature Request to C Modules:** When a user selects a feature (like "Generate Learning Content"), `app.c` calls the corresponding function in modules like `notes_quiz.c`, `youtube_summarizer.c`, or `mindmap_generator.c`.
3. **Python API Handler Invocation:** These C feature modules construct system commands and use `system()` calls to execute the `api_handler.py` Python script. They pass the necessary data (topic, content type, URL) as command-line arguments to the Python script.
4. **API Interaction and Content Generation in Python:** `api_handler.py` receives these arguments, uses the learning-toolbox library to communicate with the external AI API (like Gemini), and requests content generation (notes, quiz, summary, mind map data).
5. **Content Saving by Python:** The Python script then uses learning-toolbox functions to save the generated content into Markdown and PDF files (and potentially images for mind maps) in the local file system.
6. **Output and Loop in app.c:** After the Python script finishes (system call returns), `app.c` typically clears the screen, displays the main menu again, and waits for the next user input, continuing the application loop. The generated files are now available in the application's directory for the user to access.

In essence, the C application manages the user interaction and orchestrates the process, while the Python script handles the more complex tasks of interacting with external AI services and managing data processing and file saving. This separation of concerns makes the system modular and easier to develop and maintain.

6. RESULTS AND ANALYSIS

This chapter presents and analyzes the results and outputs generated by Study Buddy. As a study companion tool, Study Buddy is designed to produce various types of learning materials, including study notes, quizzes, YouTube video summaries, and mind maps. The effectiveness of these outputs is crucial to the application's utility. This section will showcase examples of these outputs, analyze their characteristics, discuss their strengths and potential limitations, and explore sources of error or deviation from expected outcomes. While quantitative data might be less directly applicable, we will focus on a qualitative assessment of the generated content and its value for learning.

6.1 Content Generation Results (Study Notes and Quizzes)

Study Buddy's core feature is the generation of learning content, specifically study notes and quizzes, on user-defined topics. The quality and relevance of these generated materials are key to their usefulness for students.

6.1.1 Study Notes Example and Analysis

When a user requests study notes on a topic, for example, "Deeplearning," Study Buddy interacts with the AI API to generate structured notes.

Deeplearning

Overview

Deep Learning is a subfield of Machine Learning that is inspired by the structure and function of the human brain specifically, artificial neural networks. It's characterized by the use of **deep neural networks**, which are neural networks with multiple layers (typically more than three). These deep networks are capable of automatically learning intricate patterns and hierarchical representations from vast amounts of data without explicit programming of rules. This ability to learn complex features directly from raw data has led to breakthroughs in various fields like computer vision, natural language processing, speech recognition, and more.

In essence, Deep Learning automates feature engineering, a traditionally manual and time-consuming process in classical machine learning. Instead of hand-crafting features, deep learning algorithms learn optimal features directly from the data during the training process. This makes them incredibly powerful for tackling complex problems where feature extraction is not straightforward or intuitive.

Key Concepts

Here are some fundamental concepts crucial to understanding Deep Learning:

1. Neural Networks:

- **Structure:** At the heart of Deep Learning are neural networks. They are composed of interconnected nodes or neurons organized in layers. A typical neural network consists of:
 - **Input Layer:** Receives the raw input data. The number of neurons in this layer corresponds to the number of features in your input data.
 - **Hidden Layers:** One or more layers between the input and output layers. These layers perform the complex computations and feature extraction. "Deep" learning refers to networks with multiple hidden layers.
 - **Output Layer:** Produces the final output or prediction. The number of neurons in this layer depends on the task (e.g., 1 for regression, number of classes for classification).

Figure 4 Generated Note

Analysis of Study Notes Output:

- **Structure and Formatting:** The generated notes are well-structured using Markdown formatting. Headings, subheadings, bullet points, and bold text are used to organize the information logically and improve readability. This structured format is beneficial for quick review and understanding.
- **Content Relevance and Accuracy:** The example notes on deep learning cover the key aspects of the topic: overview, key concept, example, summary, and five practice question. The information presented is generally accurate and relevant to a basic understanding of deep learning.
- **Usefulness for Learning:** The notes provide a concise yet informative overview of the topic. Students can use these notes for initial learning, revision, or as a starting point for deeper study. The structured format aids in memorization and concept mapping.
- **Potential Limitations:**
 - **Depth of Information:** Depending on the AI model and prompt, the notes might be introductory and lack in-depth details required for advanced study.
 - **Accuracy and Completeness:** While generally accurate, AI-generated content can sometimes contain minor inaccuracies or omissions. Users should always cross-verify critical information with reliable sources, especially for academic purposes.
 - **Over-reliance on AI:** Students should use these notes as a study *aid*, not as a replacement for critical thinking and engagement with broader learning materials.

6.1.2 Quiz Example and Analysis

When a user requests a quiz on a topic, for instance, "Machine Learning," Study Buddy generates quiz questions. A conceptual example of a quiz output structure is as follows:

Here are 10 MCQs about Machine Learning in markdown format:

1. **Which of the following is NOT a type of Machine Learning?**
A. Supervised Learning B. Unsupervised Learning C. Reinforcement Learning D. Deep Learning
2. **What is the purpose of a validation set in Machine Learning?**
A. To train the model B. To evaluate the final performance of the model after training C. To tune hyperparameters and prevent overfitting during training D. To deploy the model to production
3. **Which algorithm is best suited for classification problems?**
A. Linear Regression B. K-Means Clustering C. Logistic Regression D. Principal Component Analysis
4. **What is 'overfitting' in Machine Learning?**
A. When a model performs poorly on both training and test data B. When a model performs well on training data but poorly on test data C. When a model performs well on both training and test data D. When a model is too simple to capture the underlying pattern in the data
5. **Which of the following evaluation metrics is most appropriate for imbalanced datasets?**
A. Accuracy B. Precision and Recall C. R-squared D. Mean Squared Error
6. **What is feature scaling?**
A. Selecting the most important features from the dataset B. Transforming features to have a similar range of values C. Creating new features from existing features D. Removing irrelevant features from the dataset
7. **Which of the following is an example of unsupervised learning?**
A. Spam email detection B. Image classification C. Customer segmentation D. Stock price prediction

Figure 5 Generated Quiz

Analysis of Quiz Output:

- **Question Format:** The quiz is presented in a clear, multiple-choice question format. This is a common and effective format for self-assessment and exam preparation.
- **Relevance to Topic:** The questions on cell machine learning are relevant to introductory concepts in this field. They test basic understanding of machine learning.
- **Usefulness for Learning:** Quizzes are valuable for active recall and self-testing. Students can use these quizzes to check their understanding of a topic after studying the notes or other materials. Identifying incorrect answers helps pinpoint areas needing further review.
- **Potential Limitations:**
 - **Question Quality and Depth:** The quality of quiz questions can vary. Some questions might be too simplistic or not effectively assess deeper understanding. The range of topics covered in a quiz might also be limited.
 - **Question Types:** The quizzes are currently limited to multiple-choice questions. Variety in question types (e.g., true/false, fill-in-the-blanks, short answer) could enhance the learning experience.

6.2 YouTube Summarization Results

Study Buddy's YouTube summarization feature aims to condense lengthy video content into concise text summaries, saving users time while still capturing the essence of the video.

Detailed Video Summary

Key Points

- **Colliding Blocks and Pi:** Two blocks colliding on a frictionless plane can be used to compute digits of pi based on the number of collisions.
- **Mass Ratio and Pi Digits:** As the mass ratio of the larger block to the smaller block increases by powers of 100, the number of collisions approximates pi to more digits.
- **Idealized Physics Puzzle:** This is presented as an idealized classical physics puzzle, ignoring real-world factors like energy loss, sound, and relativistic effects for large mass ratios.
- **Connection to Quantum Computing:** Surprisingly, this classical physics puzzle is secretly connected to quantum computing, specifically Grover's Algorithm for search.
- **Unsolved Problem Aspect:** The full connection to pi in this context is technically an unsolved mathematical problem, related to the digits of pi and potential off-by-one errors.
- **Problem-Solving Principles:** The video uses this puzzle to illustrate general problem-solving principles like listing relevant equations (conservation of energy and momentum), drawing pictures (state space), and respecting symmetries (transforming to a circular state space).
- **State Space and Geometry:** The problem is translated from physics into a geometric problem in a state space (velocity space), making it easier to visualize and solve.
- **Inscribed Angle Theorem:** The inscribed angle theorem from geometry is used to explain why the arcs in the state space diagram are of equal size, linking to the collision count.
- **Small Angle Approximation:** The small angle approximation ($\arctan(x) \approx x$ for small x) is crucial in connecting the mass ratio, the angle in the geometric representation, and the digits of pi.
- **Next Video: Quantum Computing Connection:** The video is the first part of a two-part series, with the next video explaining the connection to Grover's Algorithm and quantum computing.

Main Ideas

Figure 6 Generated Summary

Analysis of YouTube Summary Output:

- **Conciseness:** The output is designed to be a summary, providing a significantly shorter version of the original video content. This saves users time by allowing them to quickly grasp the main points.
- **Key Point Extraction:** The "Key Points" section highlights the most important takeaways from the video in a bulleted list, making it easy to scan and identify core concepts.
- **Overall Summary:** The overall summary provides a more narrative and cohesive overview of the video's content, offering a more complete understanding than just the key points alone.
- **Usefulness for Learning:** YouTube summaries are excellent for quickly reviewing video lectures or tutorials. They help students decide if a video is relevant to their study needs and allow for efficient revision of video content.
- **Potential Limitations:**
 - **Loss of Nuance and Detail:** Summaries, by nature, condense information. Some nuances, detailed explanations, examples, or visual demonstrations from the video might be lost in the summarization process.
 - **Dependency on Transcript Quality:** The accuracy of YouTube summaries heavily relies on the quality of the video's transcript (either auto-generated or human-provided). Inaccurate transcripts will lead to flawed summaries.

6.3 Mind Map Generation Results

Study Buddy's mind map feature aims to create visual representations of topics, helping users understand the relationships between concepts and improve information retention.

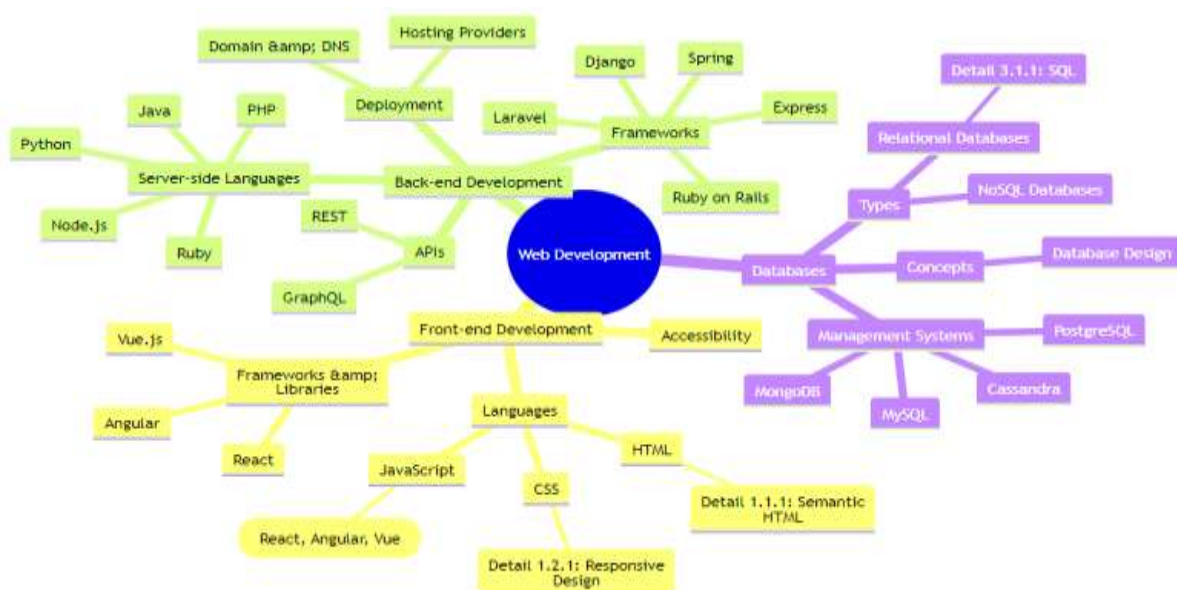


Figure 7 Generated Mind Map

Analysis of Mind Map Output:

- **Visual Organization:** Mind maps provide a visual and hierarchical organization of information, branching out from a central topic. This visual structure can significantly improve understanding and recall compared to linear text notes.
- **Relationship Mapping:** Mind maps effectively display the relationships between concepts and subtopics. Users can quickly see how different ideas connect to the main topic and to each other.
- **Usefulness for Learning:** Mind maps are highly beneficial for brainstorming, studying complex topics, and visualizing knowledge structures. They are particularly helpful for visual learners.
- **Potential Limitations:**
 - **Complexity of Topics:** For very complex or nuanced topics, automatically generated mind maps might oversimplify or misrepresent certain relationships. Human refinement and customization might be necessary to create truly effective mind maps for advanced subjects.
 - **Accuracy of Relationships:** The accuracy of the relationships depicted in the mind map depends on the AI's understanding of the topic and its ability to extract and structure relevant concepts. Inaccuracies in concept relationships can mislead the user.
 - **Customization and Interactivity:** While Study Buddy generates a mind map image, it might lack interactive features found in dedicated mind mapping software (e.g., collapsing/expanding branches, adding notes to nodes, dynamic editing).

6.4 Error Analysis

While Study Buddy aims to provide helpful learning materials, several potential sources of errors and deviations from expected outputs exist:

- **AI API Limitations:** The quality of generated content is directly tied to the capabilities and limitations of the underlying AI API (like Gemini) used by learning-toolbox. Inaccuracies, biases, or lack of depth in the AI model's knowledge base can translate into similar issues in Study Buddy's outputs.
- **Dependency on External Services:** Study Buddy relies on external APIs and internet connectivity. API downtime, changes in API functionality, or loss of internet connection will directly impact the application's ability to generate content.
- **learning-toolbox and Integration Issues:** Any bugs or limitations within the learning-toolbox Python package or in the way Study Buddy integrates with it can introduce errors or unexpected behavior.
- **User Input Errors:** Incorrectly entered topics, URLs, or API keys can lead to failed content generation or inaccurate results.

7. FUTURE ENHANCEMENTS

Study Buddy, in its current form, provides a solid foundation for a personal study companion. However, there are numerous avenues for technical improvement that could significantly enhance its functionality, user experience, and overall effectiveness as a learning tool. This chapter outlines several potential future enhancements categorized by area of improvement.

Here are some ideas for future enhancements:

- **Making it Safer (Security):** Right now, "StudyBuddyyy" stores usernames and passwords in a very simple way that isn't secure. A big improvement would be to use *password hashing and salting*. This is like scrambling the passwords so that even if someone got access to the users.dat file, they couldn't easily figure out the passwords. Another good idea would be to use a more robust database system (like SQLite) instead of a simple file.
- **Better Error Handling:** The application could be more helpful when things go wrong. Here's what could be improved:
 - **Checking for File Problems:** Make sure the application checks if files (like the user database) can be opened and read correctly.
 - **Handling Internet Issues:** If the application can't connect to the Google API, it should give a clear message to the user, like "Could not connect to the internet. Please check your connection."
- **Working on More Computers (Cross-Platform Compatibility):** Currently, "StudyBuddyyy" only works on Windows. To make it work on other operating systems (like macOS or Linux), you'd need to remove the parts that are Windows-specific (the windows.h code that controls the colors and text layout). There are cross-platform libraries that could be used instead.
- **Personalized Learning (User Profiles):** Imagine if "StudyBuddyyy" could adapt to your learning style! You could have a user profile where you set your preferences:
 - **Learning Style:** Do you prefer visual learning, reading, or doing exercises?
 - **Difficulty Level:** Beginner, intermediate, or advanced.
- **Tracking Your Progress:** The application could keep track of what you've studied and how well you've done on quizzes. This would help you see your progress and identify areas where you need more practice.
- **More Ways to Learn (More Content Types):** Besides notes, quizzes, summaries, and mind maps, "StudyBuddyyy" could also generate:
 - **Flashcards:** For memorizing key facts.
 - **Interactive Exercises:** More engaging ways to test your knowledge.

- **A Better Look (UI Improvements):** Instead of just typing in a console window, "StudyBuddyyy" could have a more user-friendly *graphical user interface (GUI)*. This would make it look more like a modern application, with buttons, menus, and windows.
- **Python Package Management:** To make sure the Python part of the code works correctly, it's important to manage the *dependencies* (the extra Python tools it uses). A `requirements.txt` file (or a virtual environment) would list all the required Python packages and their versions. This ensures that anyone can easily install the correct versions of the necessary libraries.
- **API Key Security:** Instead of putting the API key directly in the code, it's much safer to store it in a separate place, like an *environment variable* or a *configuration file*. This helps prevent accidental exposure of the key.
- **API Usage Limits (Rate Limiting):** The Google API probably has limits on how many requests you can send in a certain amount of time. "StudyBuddyyy" should be careful not to exceed these limits, or it might get temporarily blocked. *Rate limiting* is a way to control how often the application sends requests to the API.

These enhancements would make "StudyBuddyyy" a more powerful, secure, and user-friendly learning tool.

8. CONCLUSION

In conclusion, Study Buddy represents a significant step towards creating a truly personal and interactive study companion. This project set out to tackle the challenges students face in efficiently gathering, understanding, and retaining information in today's fast-paced learning environment. By integrating features for note generation, quiz creation, YouTube video summarization, and mind map visualization, Study Buddy offers a diverse toolkit designed to enhance the learning process.

The development of Study Buddy successfully demonstrated the feasibility of combining C-based application logic with Python-powered AI capabilities to deliver a functional desktop application. Key achievements include the implementation of a user authentication system, the integration with external AI APIs for content generation via the learning-toolbox library, and the successful creation of study notes, quizzes, YouTube summaries, and mind maps in various file formats. The modular architecture, separating the user interface in C from the backend AI processing in Python, provides a solid and maintainable foundation for future development.

However, it's important to acknowledge the limitations of the current version. The text-based command-line interface, while functional, lacks the user-friendliness of a graphical interface. The reliance on potentially imperfect AI-generated content means that users should always critically evaluate and cross-verify information. Furthermore, the current implementation has security weaknesses, most notably the plain text storage of passwords, which must be addressed in future iterations.

Despite these limitations, Study Buddy holds considerable promise. It provides a tangible demonstration of how AI can be leveraged to automate tedious study tasks and offer students personalized learning support. By saving time on note-taking and content summarization, Study Buddy empowers students to focus more on active learning and deeper understanding. The potential for future enhancements, including a graphical user interface, interactive quiz features, expanded AI integration, and improved security, is vast.

Ultimately, Study Buddy is more than just a code project; it's a stepping stone towards a future where technology can truly personalize and enhance the learning experience for everyone. It serves as a valuable platform for continued development and refinement, with the potential to evolve into an even more powerful and indispensable tool for students of all levels.

9. APPENDICES

This section provides supplementary materials that offer additional details and practical guidance related to Study Buddy. These appendices are designed to be helpful for users who want to understand more about the application or need practical instructions for using it effectively.

Appendix A: User Manual

This user manual provides a step-by-step guide on how to use Study Buddy. It covers everything from initial setup to utilizing each of its features.

Prerequisites

Before you begin, make sure you have the following software installed on your system:

- C Compiler: A C compiler is required to compile the application's source code. Common choices include:
- Python: Python 3 (version 3.6 or higher is recommended) is needed to run the content generation scripts. You can download it from the official Python website: <https://www.python.org/downloads/>
- Node.js and npm: Node.js is a JavaScript runtime, needed for Mermaid-cli, used to generate mindmap, and npm is a package manager, it comes with Node.js.

Installation and Setup

1. Clone the Repository:

Open a terminal or command prompt and use the git command to clone the StudyBuddyyy repository from GitHub:

```
git clone https://github.com/Avinash1286/Study-Buddy.git
```

This will create a directory named Study-Buddy containing the project files.

2. Navigate to the Project Directory:

Change your current directory to the newly cloned Study-Buddy directory:

```
cd Study-Buddy
```

3. Install the learning-toolbox Python Package:

Use pip to install the required learning-toolbox package:

```
pip install learning-toolbox==0.2.0
```

You can find more information about this package on PyPI: <https://pypi.org/project/learning-toolbox/0.2.0/>

4. Install Mermaid CLI (Globally):

Install mermaid cli using this command:

`npm install -g @mermaid-js/mermaid-cli`

1. Set up Your Google Gemini API Key:

- Obtain an API key from Google AI Studio:
<https://aistudio.google.com/app/apikey>
- Open the `api_handler.py` file in a text editor.
- Replace `GEMINI_API_KEY` with your actual API key:
- `api_key = 'YOUR_ACTUAL_API_KEY'`

Important: Keep your API key secret. Do not commit it to public repositories.

2. Compile the C Code:

Use your C compiler to compile the source code. Here's an example using GCC:

`gcc banner.c colors.c app.c notes_quiz.c auth.c youtube_summarizer.c mindmap_generator.c -o study_buddy`

This command will create an executable file named `study_buddy.exe`.

Appendix B: Running StudyBuddyyy

1. Start the Application:

- In your command prompt, navigate to the project directory and run the executable by typing `.\study_buddy` and pressing Enter.

2. Authentication Menu:

- The application will display a welcome banner and the Authentication Menu:

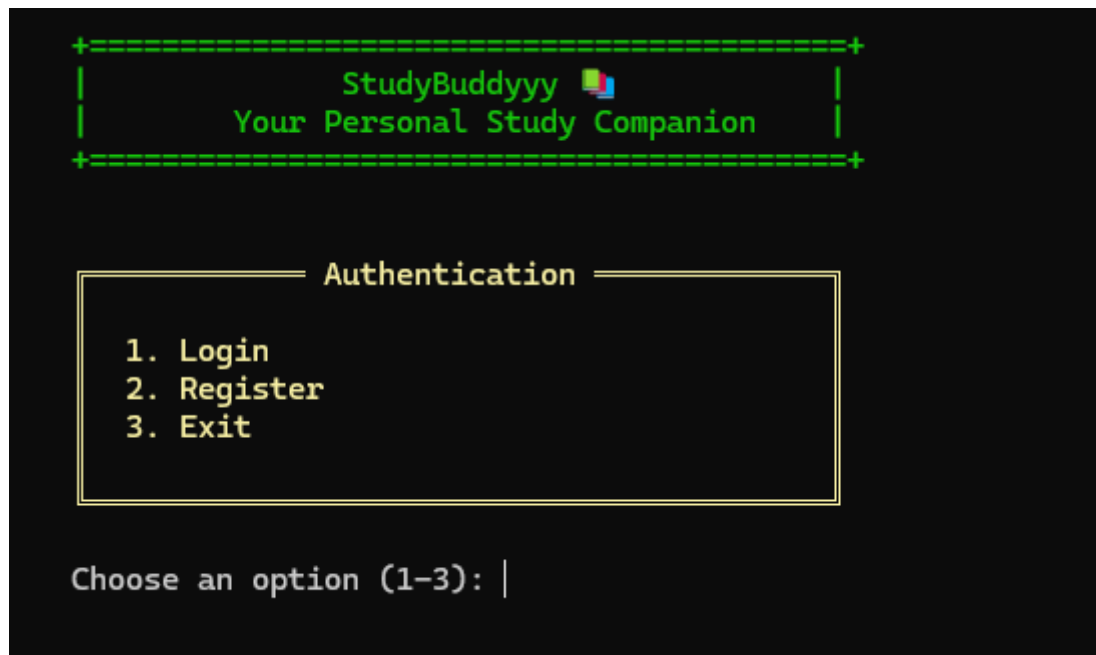


Figure 8 Authentication Menu

- **Register:** If you are a new user, choose option 2 (Register). Enter a username and password when prompted.
- **Login:** If you are a returning user, choose option 1 (Login). Enter your registered username and password.
- **Exit:** To close the application, choose option 3 (Exit).

Main Menu:

- After successful login, you will see the Main Menu:

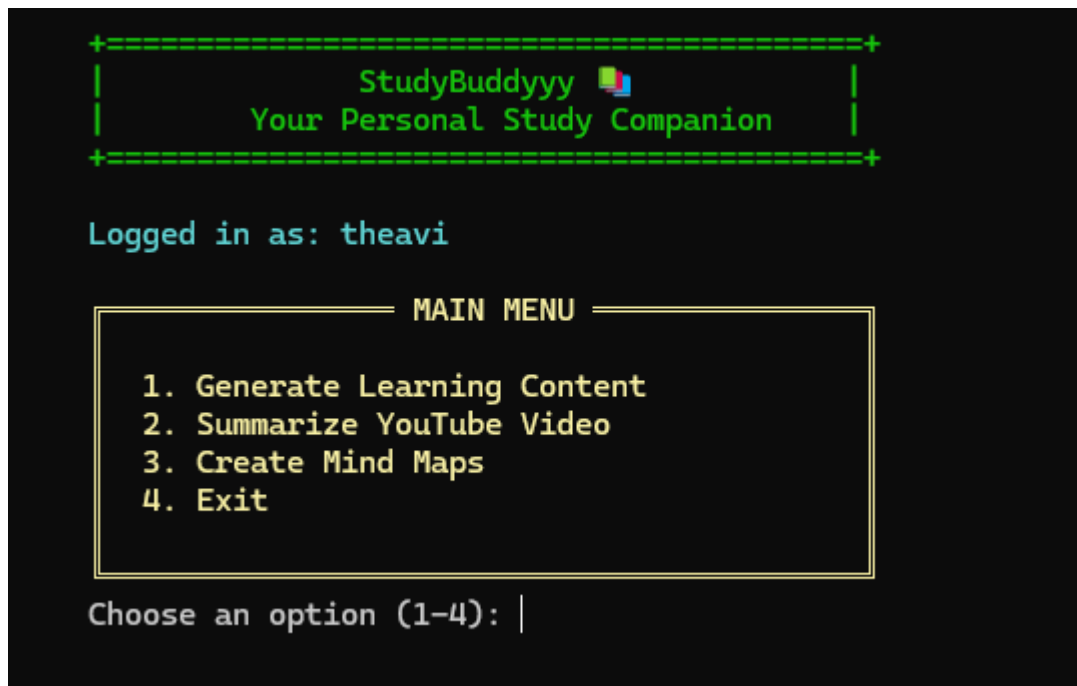


Figure 9 Main Menu

- **Generate Learning Content (Option 1):** Choose 1 to generate study notes or a quiz on a topic. You will be prompted to enter the topic and then select the content type (notes or quiz).

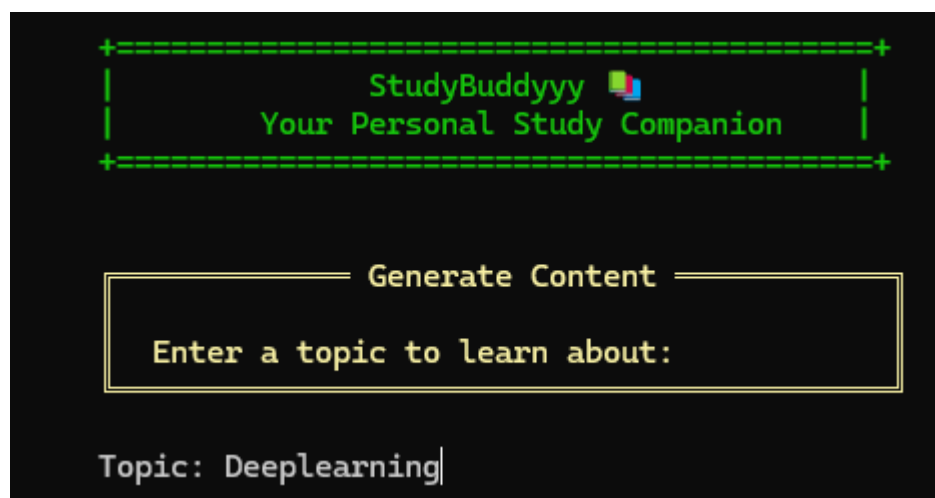


Figure 10 Topic Input

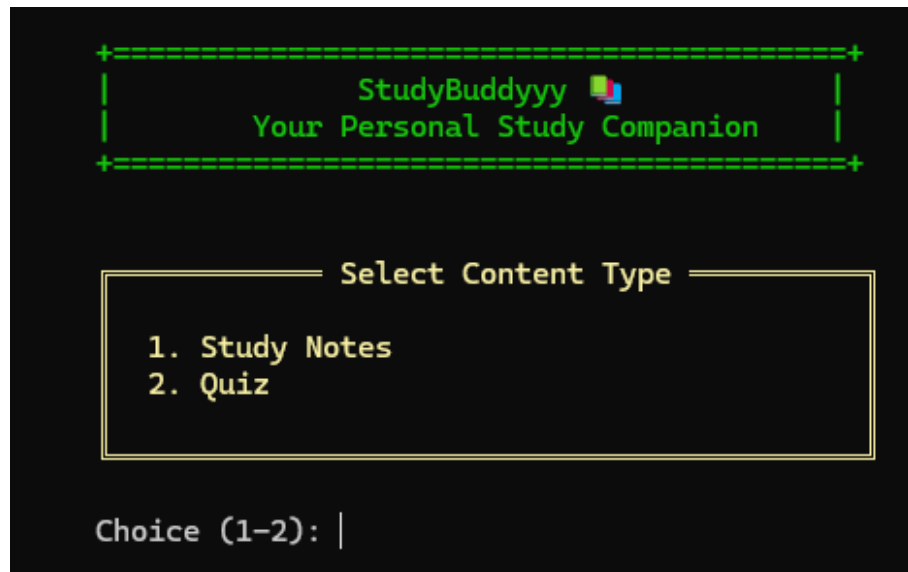


Figure 11 Select Content Type

```

Generating study notes for 'DeepLearning'...
WARNING: output_resource\notes\deeplearning_notes.md
PDF: output_resource\notes\deeplearning_notes.pdf
'''markdown
# DeepLearning

## Overview

Deep Learning is a subfield of Machine Learning that is inspired by the structure and function of the human brain specifically, artificial neural networks. It's characterized by the use of deep neural networks, which are neural networks with multiple layers (typically more than three). These deep networks are capable of automatically learning intricate patterns and hierarchical representations from vast amounts of data without explicit programming of rules. This ability to learn complex features directly from raw data has led to breakthroughs in various fields like computer vision, natural language processing, speech recognition, and more.

In essence, Deep Learning automates feature engineering, a traditionally manual and time-consuming process in classical machine learning. Instead of hand-crafting features, deep learning algorithms learn optimal features directly from the data during the training process. This makes them incredibly powerful for tackling complex problems where feature extraction is not straightforward or intuitive.

## Key Concepts

Here are some fundamental concepts crucial to understanding Deep Learning:

*1. Neural Networks:*

* Structure: At the heart of Deep Learning are neural networks. They are composed of interconnected nodes or neurons organized in layers. A typical neural network consists of:
    * Input Layer: Receives the raw input data. The number of neurons in this layer corresponds to the number of features in your input data.
    * Hidden Layers: One or more layers between the input and output layers. These layers perform the complex computations and feature extraction. "Deep" learning refers to networks with multiple hidden layers.
    * Output Layer: Produces the final output or prediction. The number of neurons in this layer depends on the task (e.g., 1 for regression, number of classes for classification).
    * Neurons/Nodes: The basic computational unit. Each neuron receives inputs, performs a weighted sum of these inputs, adds a bias, and then applies an activation function.
  
```

Figure 12 Generated Note

```

1. Which of the following is NOT a type of Machine Learning?
   A. Supervised Learning
   B. Unsupervised Learning
   C. Reinforcement Learning
   D. Deep Learning

2. What is the purpose of a validation set in Machine Learning?
   A. To train the model
   B. To evaluate the final performance of the model after training
   C. To tune hyperparameters and prevent overfitting during training
   D. To deploy the model to production

3. Which algorithm is best suited for classification problems?
   A. Linear Regression
   B. K-Means Clustering
   C. Logistic Regression
   D. Principal Component Analysis

4. What is 'overfitting' in Machine Learning?
   A. When a model performs poorly on both training and test data
   B. When a model performs well on training data but poorly on test data
   C. When a model performs well on both training and test data
   D. When a model is too simple to capture the underlying pattern in the data
  
```

Figure 13 Generated Quiz

- **Summarize YouTube Video (Option 2):** Choose 2 to summarize a YouTube video. You will be prompted to enter the YouTube video URL.

```

Choose an option (1-4): 2
Enter YouTube video URL: https://www.youtube.com/watch?v=6d7y0L1f0e
Generating Summary for Youtube Video 'https://www.youtube.com/watch?v=6d7y0L1f0e'...
Markdown: output_resource/youtube/youtube_6d7y0L1f0e_summary.md
PDF: output_resource/youtube/youtube_6d7y0L1f0e_summary.pdf
'''markdown
# Detailed Video Summary

## Key Points

* Colliding Blocks and Pi: Two blocks colliding on a frictionless plane can be used to compute digits of pi based on the number of collisions.
* Mass Ratio and Pi Digits: As the mass ratio of the larger block to the smaller block increases by powers of 100, the number of collisions approximates pi to more digits.
* Idealized Physics Puzzle: This is presented as an idealized classical physics puzzle, ignoring real-world factors like energy loss, sound, and relativistic effects for large mass ratios.
* Connection to Quantum Computing: Surprisingly, this classical physics puzzle is secretly connected to quantum computing, specifically Grover's Algorithm for search.
* Unsolved Problem Aspect: The full connection to pi in this context is technically an unsolved mathematical problem, related to the digits of pi and potential off-by-one errors.
* Problem-solving Principles: The video uses this puzzle to illustrate general problem-solving principles like listing relevant equations (conservation of energy and momentum), drawing pictures (state space), and respecting symmetries (transforming to a circular state space).
* State Space and Geometry: The problem is translated from physics into a geometric problem in a state space (velocity space), making it easier to visualize and solve.
* Inscribed Angle Theorem: The inscribed angle theorem from geometry is used to explain why the arcs in the state space diagram are of equal size, linking to the collision count.
* Small Angle Approximation: The small angle approximation ( $\arctan(x) \approx x$  for small  $x$ ) is crucial in connecting the mass ratio, the angle in the geometric representation, and the digits of pi.
* Next Video: Quantum Computing: The video is the first part of a two-part series, with the next video explaining the connection to Grover's Algorithm and quantum computing.

## Main Ideas

```

Figure 14 Youtube Summary

- **Create Mind Maps (Option 3):** Choose 3 to generate a mind map for a topic. You will be prompted to enter the topic for the mind map.



Figure 15 Mind Map

- **Exit (Option 4):** Choose 4 to exit the Study Buddy application.

Generated Files:







 mindmaps		3/14/2025 5:59 PM	File folder
 notes		3/14/2025 5:58 PM	File folder
 youtube		3/14/2025 5:58 PM	File folder

Figure 16 File Structure

- Study materials (notes and quizzes) are saved as both .md (Markdown) and .pdf files in the same directory as study_buddy.exe.
- Mind maps are saved as image files (.png) in the same directory.
- The filenames will be based on the topic you provided.