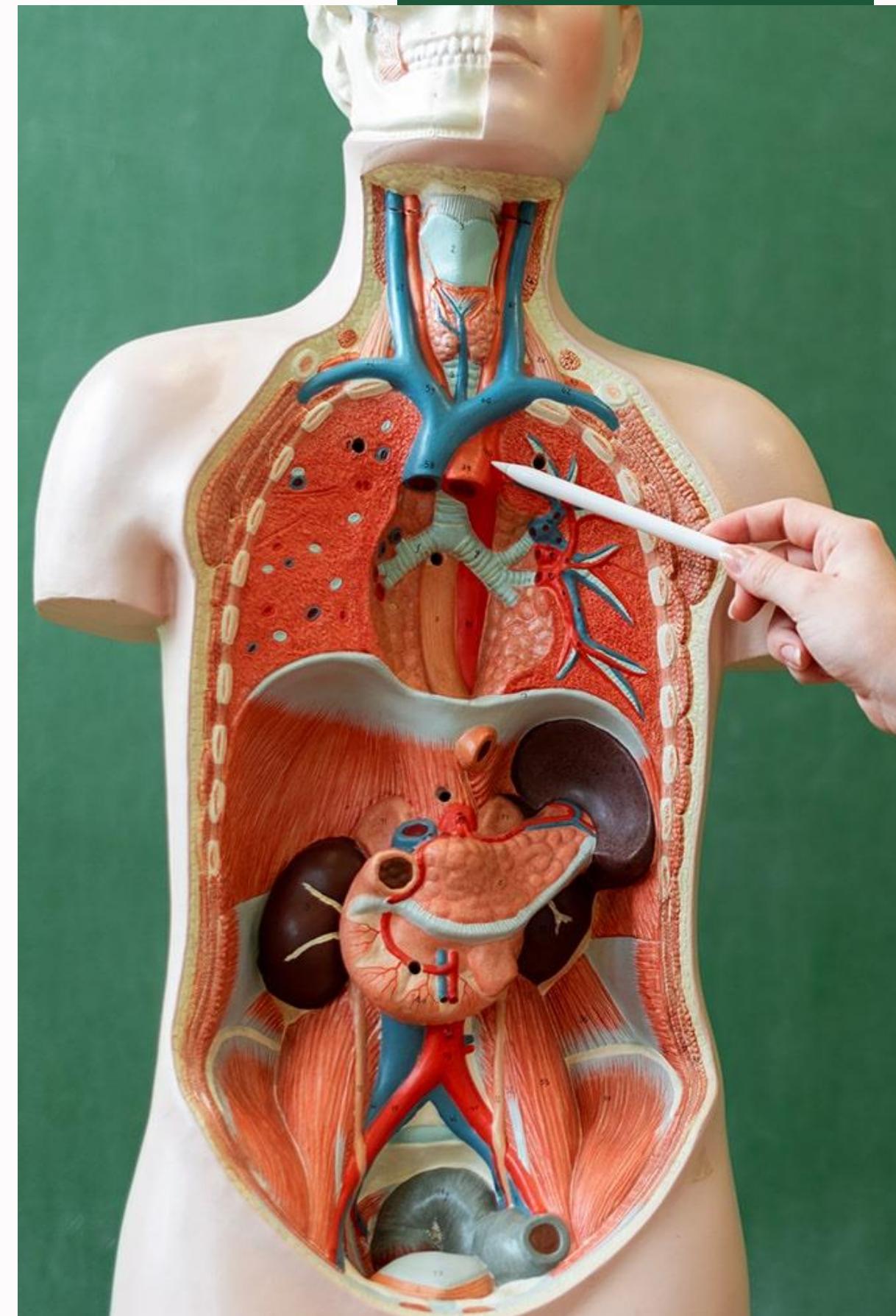


# **BIOMENTOR - PERSONALIZED E- LEARNING PLATFORM FOR A/L BIOLOGY SUBJECT Students in Sri Lanka**

24-25J-257



# Our Team



Srirajan G. A  
IT21375132



Dharane.S  
IT21068478



Sujitha.S  
IT21264634



Sajeevan.S  
IT21204302



N.Thayaparan  
External  
Supervisor

# Introduction

**01**

Background

**02**

Research Problem

**03**

Research Objectives

**04**

Overall System Diagram



# Background

This project aims to create an engaging and effective learning environment that caters to individual learning styles and promotes continuous improvement through detailed feedback and performance tracking.

We're focusing on this project to provide personalized learning in A-Level Biology using approved government resources. Our advanced technologies aim to offer tailored, engaging experiences that enhance retention and readiness.



# Research Problem

01



A/L biology students in Sri Lanka struggle with memorizing complex biological terms and their pronunciation. There is a need for an interactive tool that combines spaced repetition with detailed feedback to enhance vocabulary retention and accuracy.

02

Students struggle with extensive biology texts, and existing tools don't effectively extract key concepts or support auditory learners. There is also a lack of tools that provide targeted summaries for exam preparation. A solution is needed to offer concise, accurate content that supports diverse learning styles and aligns with educational standards.

03



Static MCQ platforms do not adapt to students' abilities, leading to ineffective practice and poor identification of knowledge gaps. The objective is to develop an adaptive quiz system that adjusts question difficulty, offers detailed performance analytics, and provides targeted feedback.

04

Existing evaluation systems for biology responses often lack comprehensive feedback and actionable recommendations. The problem is to design a platform that provides accurate answers, detailed feedback, and additional study resources to support student improvement.



# Objectives

## Objective 01

Biology vocabulary memorization tool using digital flashcards and spaced repetition, providing personalized feedback and adaptive learning paths.

## Objective 02

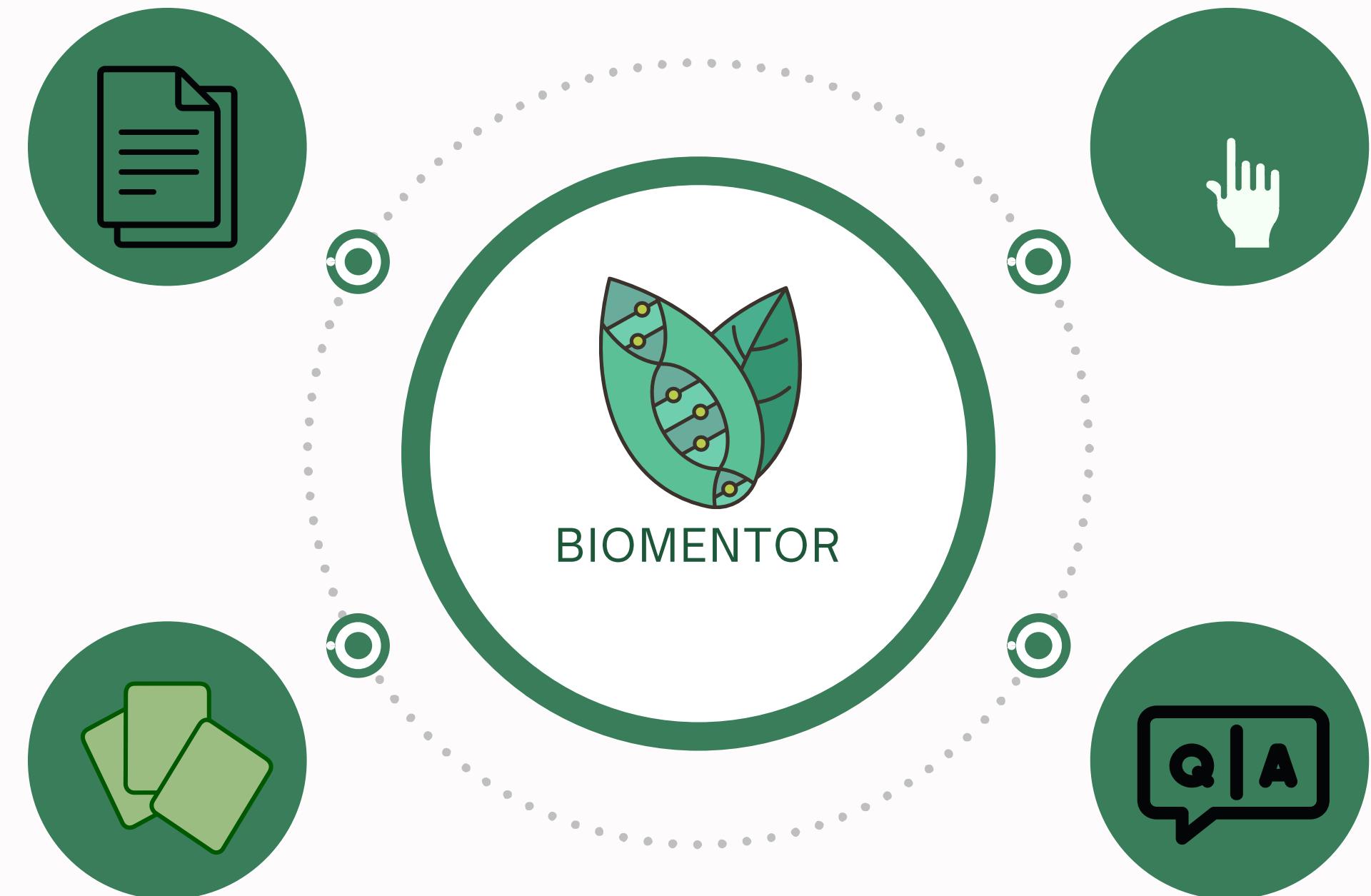
Summarization tool that generates concise, topic-based summaries from uploaded documents and searches through resources to produce summaries on specific topics, with customizable word counts and voice output for diverse learning styles.

## Objective 03

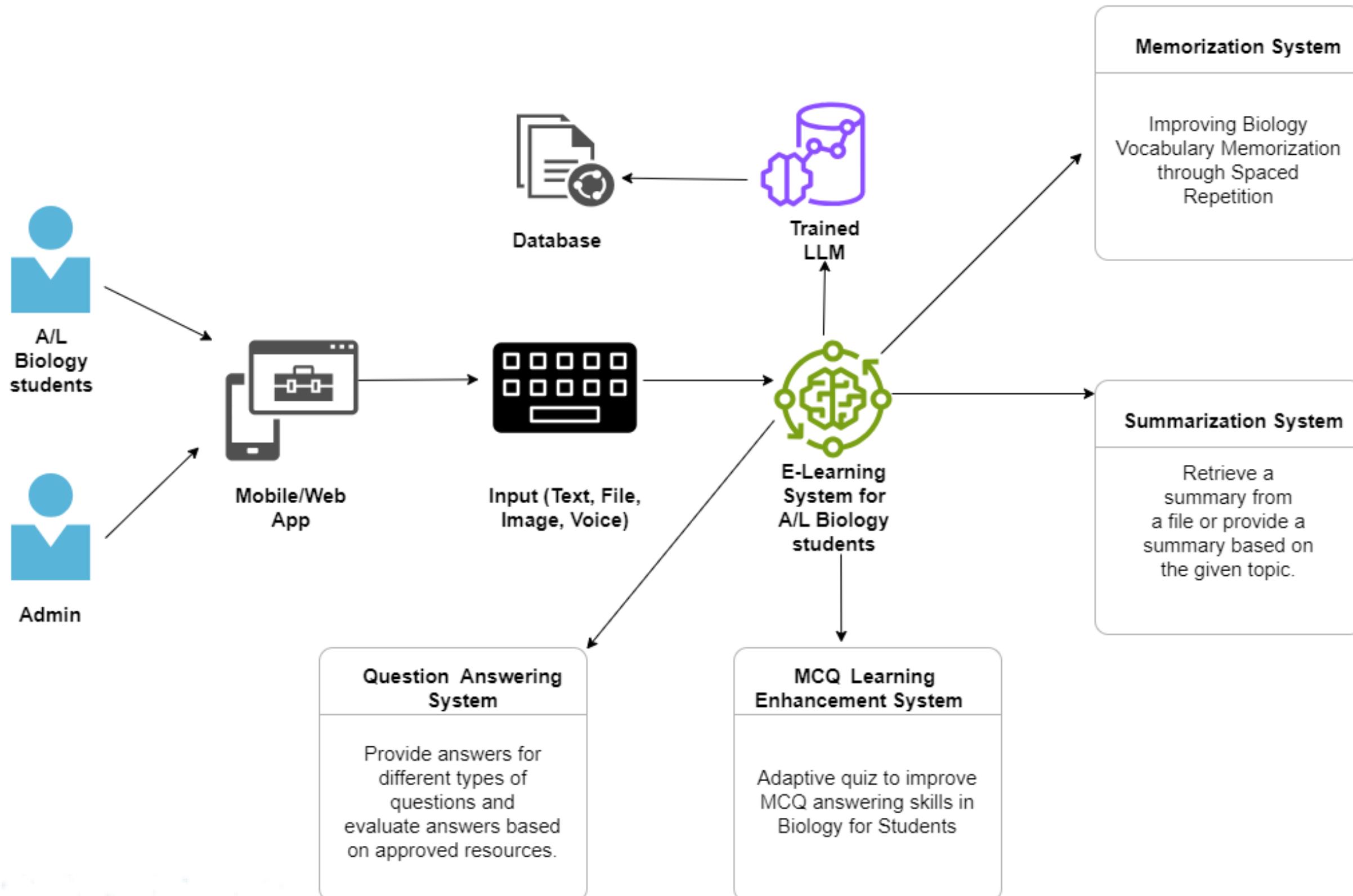
An adaptive quiz platform that dynamically adjusts question difficulty based on student performance, offering targeted practice and detailed performance analysis to enhance learning outcomes.

## Objective 04

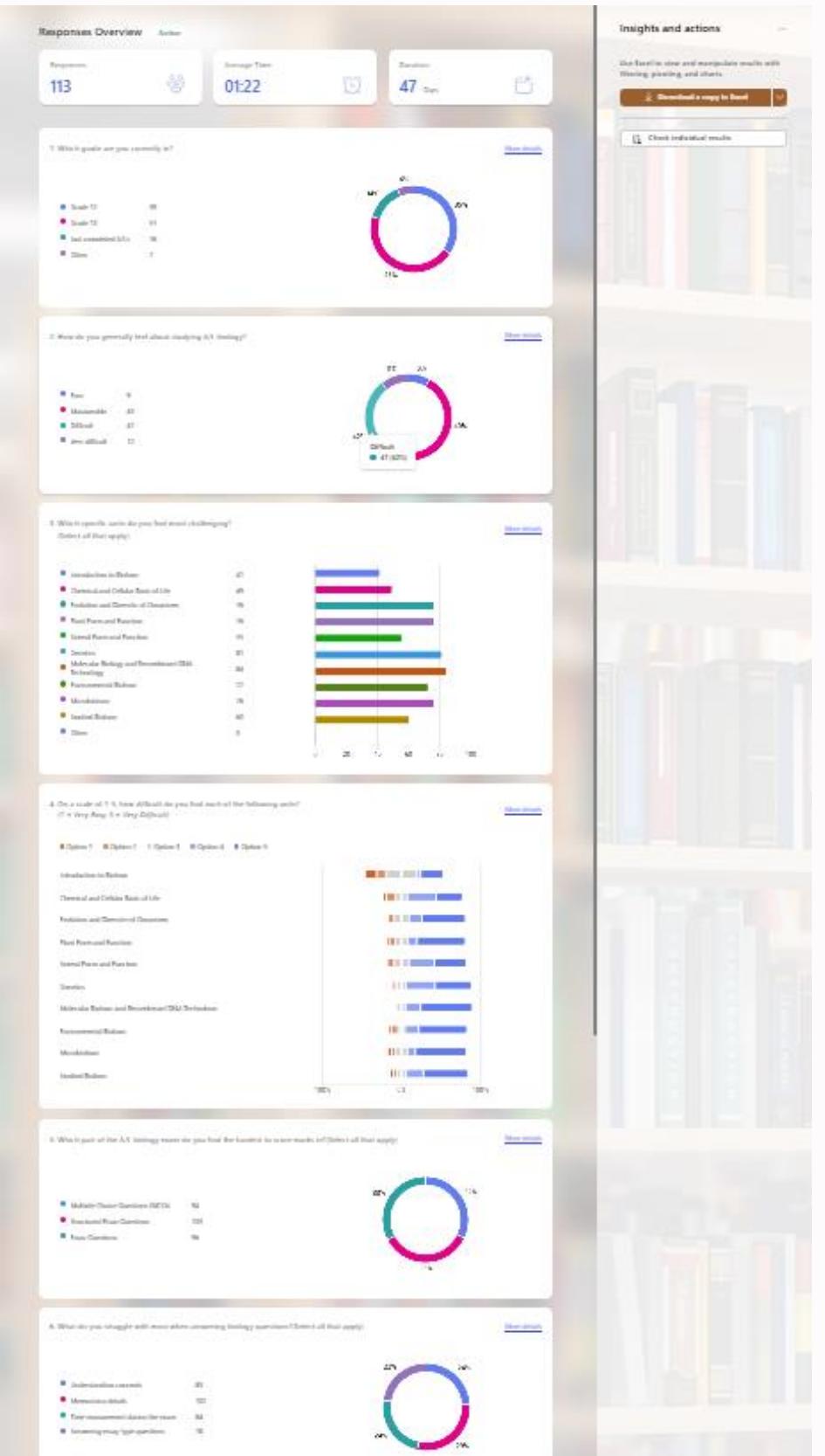
Q & A platform to generate answers for provided questions and evaluate responses, offering feedback and personalized study resources to improve student understanding and performance.



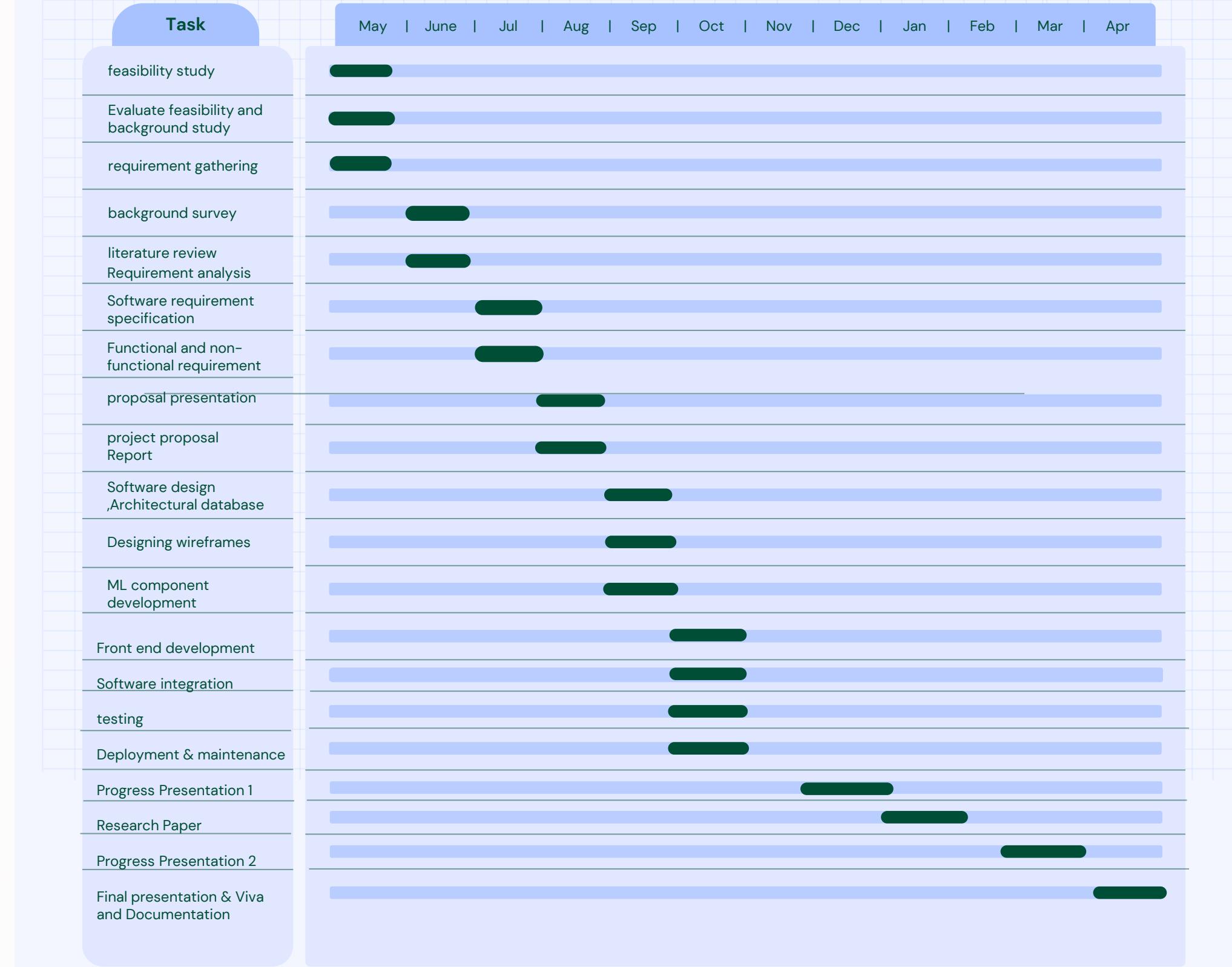
# Overall System Diagram

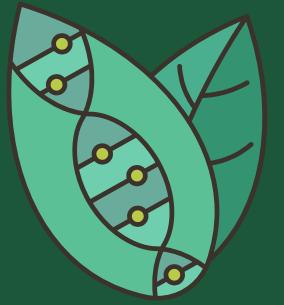


# Survey



# Gantt Chart





BIOMENTOR

# Cost Management Plan

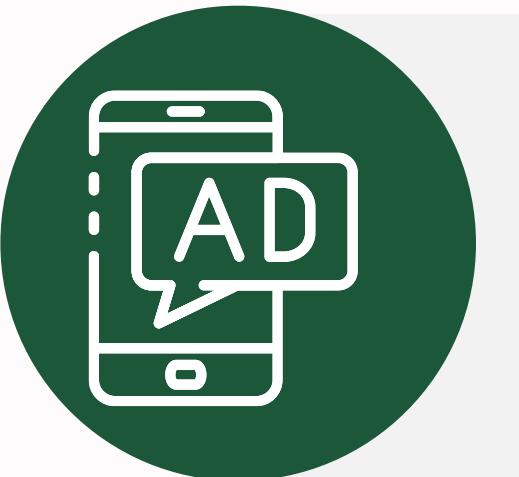
Typs	Cost
Internet use and web hosting	LKR.10,000.00
Training Cost	LKR.30,000.00
Publication Cost	LKR.70,000.00
Stationery	LKR.1,000.00
Total	LKR.111,000.00



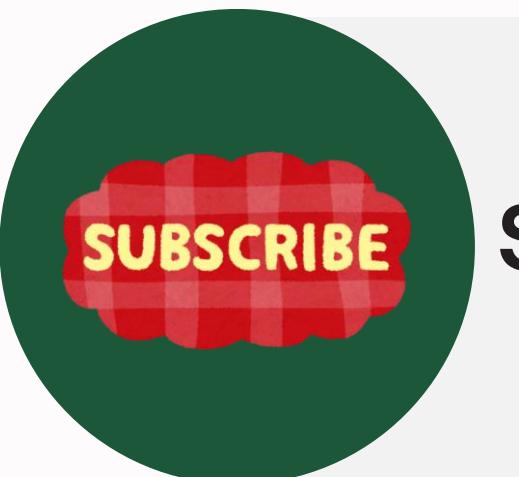


BIOMENTOR

# Commercialization



Advertising



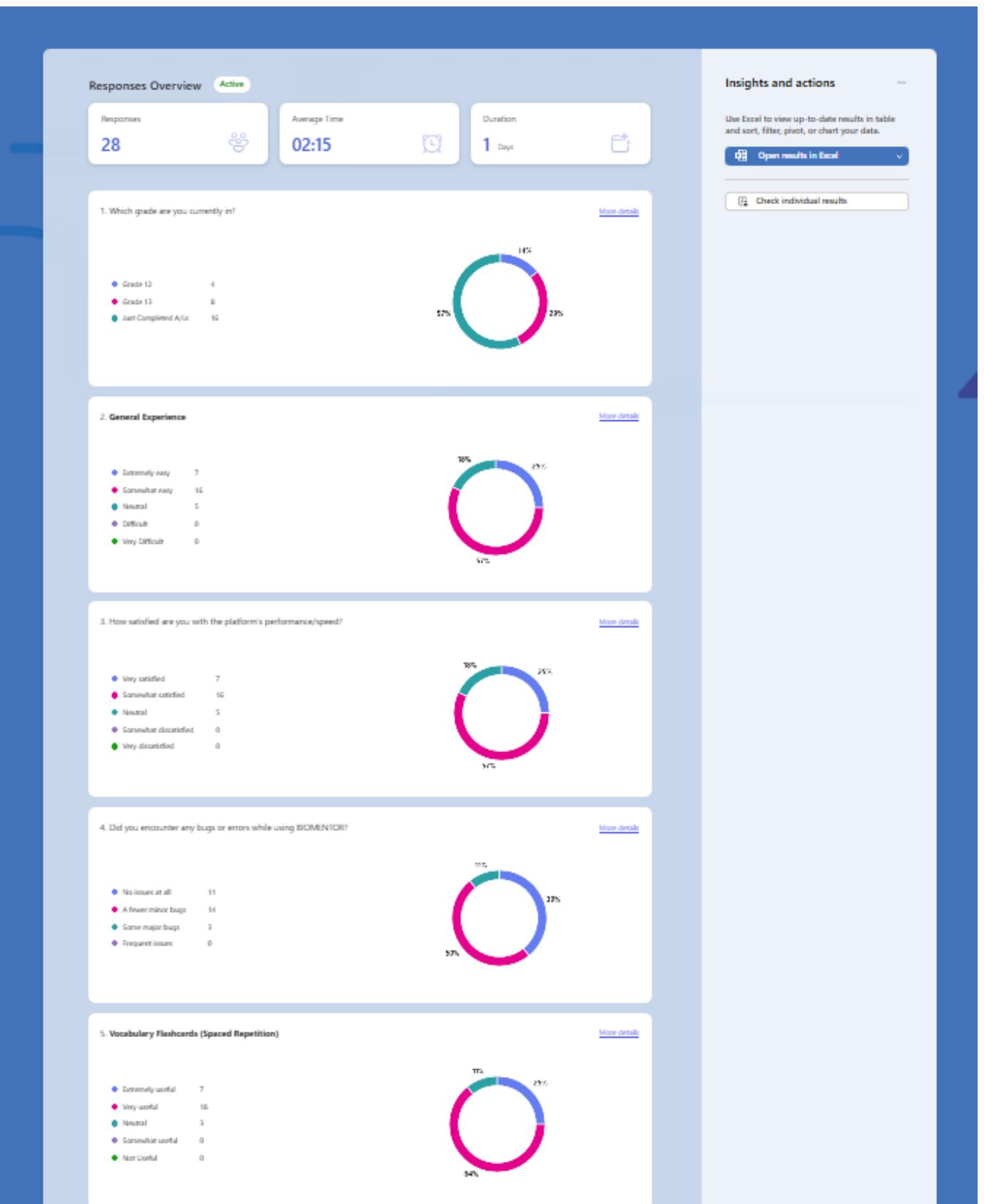
Subscription



Collaboration



# User Feedback



# GitHub Details

Y3S1-GRP22 / BioMentor-Personalized-E-Learning-Platform

Type / to search

Code Issues Pull requests Actions Projects Security 15 Insights Settings

**BioMentor-Personalized-E-Learning-Platform** Public

Edit Pins Watch 0 Fork 0 Star 0

main 8 Branches 0 Tags Go to file Add file Code About

DharaneSegar Merge pull request #9 from Y3S1-GRP22/IT21068478/Dharane cda1e0d · 2 days ago 193 Commits

Back-End Phrase fixes 5 days ago

Front-End Minor fixes 5 days ago

Images Add system diagram in README.md 4 months ago

Model-Training Updated folder structure last month

Readme.md Add system diagram in README.md 4 months ago

README

**BioMentor - Personalized E-Learning Platform for A/L Biology Students**

Final year research project of group 24-25J-257

Readme Activity Custom properties 0 stars 0 watching 0 forks Report repository

No releases published Create a new release

No packages published

# Project Management

Jira Your work Projects Filters Dashboards Teams Plans Apps Create Upgrade Search ⚡ ⓘ 🔍 ⚙️ 🚫

Projects / BioMentor SCRUM Sprint 1 33 days ⚡ ⭐ 🔗 ↗ Complete sprint ⋮

Search GROUP BY None Insights View settings

TO DO 5	IN PROGRESS 1	TESTING 2	DONE 36
Integrate LLM with the Adaptive Quiz Platform SCRUM-29 S	Develop Question Generation Pipeline SCRUM-41 S	Evaluate and Refine the MCQs generated by chosen LLM SCRUM-27 S	Submit Topic Assessment Form (TAF) SCRUM-19 ✓ S
Build Adaptive Quiz Algorithm SCRUM-42 S	Implement RAG for better MCQ Question generation SCRUM-40 S	Proposal Presentation SCRUM-20 ✓ S	Conduct Surveys with Current A/L Biology Students For Q&A SCRUM-1 ✓ SS
Develop Dynamic Quiz Generation Based on Performance SCRUM-25 S	Configure Retriever From vector database SCRUM-8 ✓ SS		
Implement Performance Analysis and Tracking Mechanism			

# Confirmation from External Supervisor

Nagalatha Thayaparan  
A/L Biology Subject Teacher  
Saiva Mangaiyar Vidyalayam  
Colombo 06  
25th May 2025

To:  
The Academic Department  
SLIIT, Malabe

**Subject: Confirmation of Assistance with Final-Year Project**

Dear Sir/Madam,

I am writing to confirm my involvement in supporting the final-year project of the SLIIT Software Engineering students Sujitha S., Dharane S., Sajeevan S., and Abisheak G.S. Their project, a web-based e-learning platform for A/L Biology students, is fully aligned with the A/L Biology syllabus.

I assisted the team by providing past examination papers and syllabus-based materials, as well as helping organize the educational content integrated into the system.

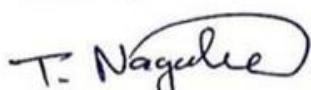
The platform comprises several valuable components. The MCQ generation system performs well, offering questions across different difficulty levels. Most of the questions and answers are accurate and relevant to the syllabus. The Q&A module supports both structured and essay-type questions, provides appropriate answers, and effectively compares students' responses. The summarization tool generates clear and concise notes, and its voice output feature is especially helpful. Additionally, the flashcard feature enables students to review key terms and concepts efficiently.

I also conducted a trial of the system with a group of my students. Based on their feedback, the platform is user-friendly and serves as an effective support tool for exam preparation.

Overall, the system is well-developed and functions smoothly. It will undoubtedly be a valuable resource for A/L Biology students as they prepare for their examinations.

Thank you.

Sincerely,



Nagalatha Thayaparan

# IT21375132

Srirajan G.A

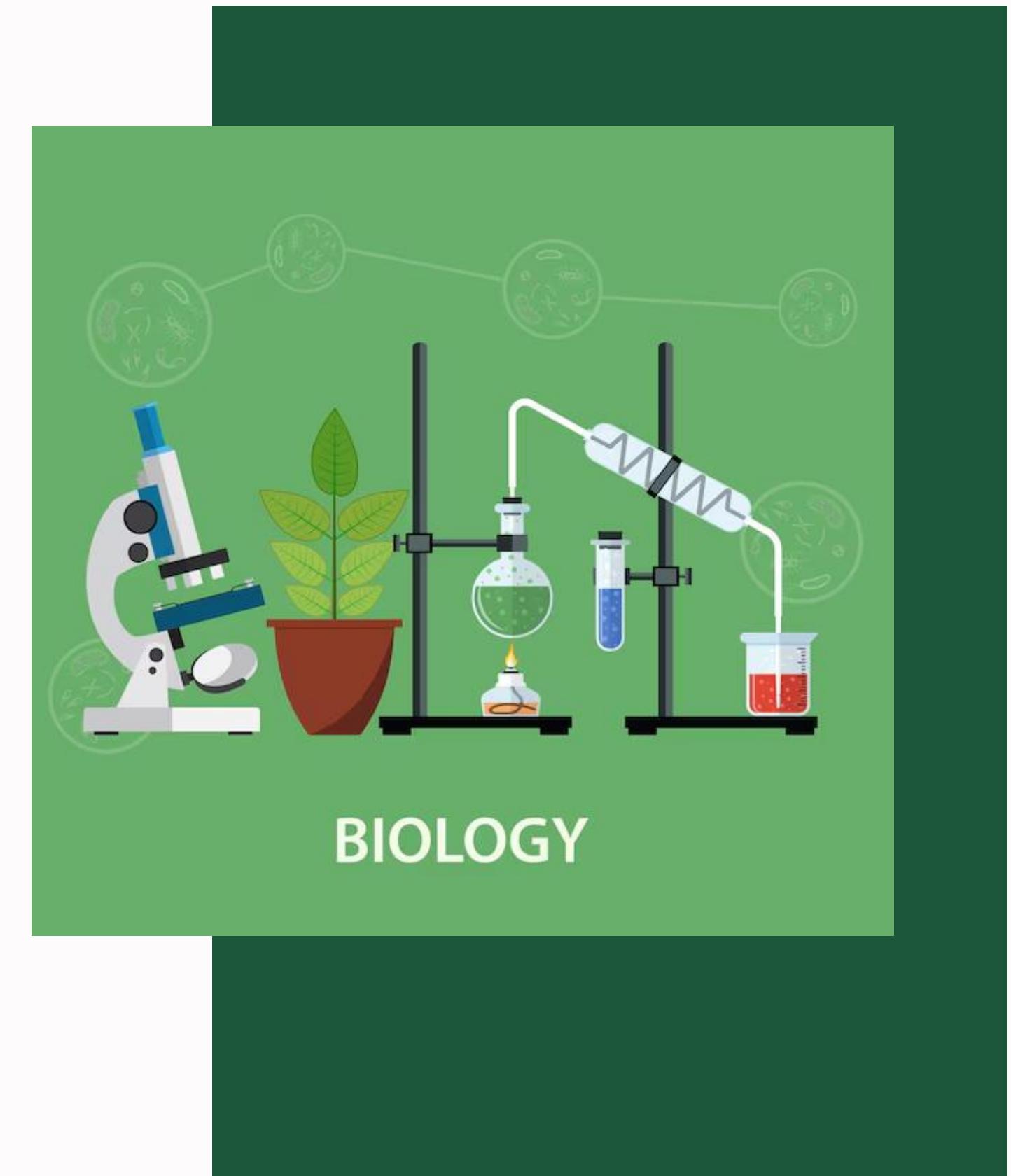
Software Engineering

IMPROVE BIOLOGY VOCABULARY  
MEMORIZATION THROUGH  
SPACED REPETITION



# Introduction

- 01** Background
- 02** Research Problem
- 03** Main and Sub Objectives
- 04** Methodology
- 05** Completion Of Project



# BACKGROUND



Students struggle with retaining large amounts of biology vocabulary due to the natural forgetting curve, where newly learned information is quickly forgotten if not reviewed.



An innovative system is needed to leverage cognitive science principles, such as spaced repetition, to counteract the forgetting curve.

# BACKGROUND

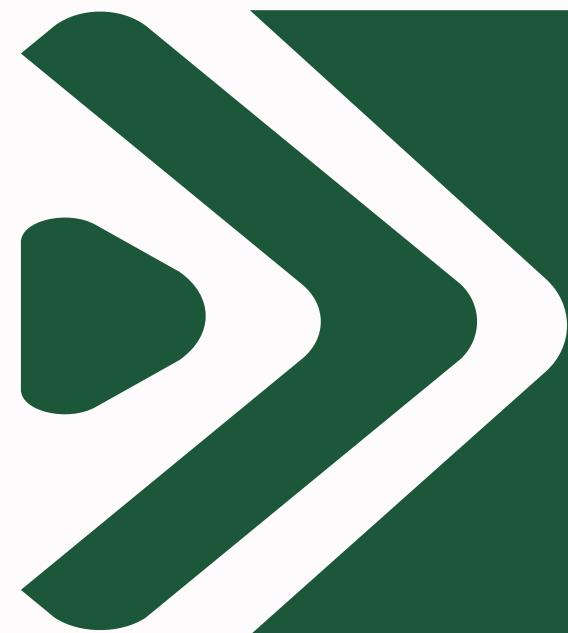


## Schistosomiasis

*Schistosomiasis is an acute  
and chronic disease caused  
by parasitic worms*

# RESEARCH PROBLEM

01



How can we customize the spaced intervals based on the performance of the user and the difficulty of the vocabulary to maximize memory retention among Advanced Level Biology students?

02



How can we incorporate multi-sensory spaced repetition techniques on the memorization of biology vocabulary to maximize the memory retention among Advanced Level Biology students?

# OBJECTIVES

## Objective 1

Create a spaced repetition model that adapts to individual user performance and adjusts review intervals based on the difficulty level of the vocabulary.

## Main Objective

To create an interactive application with flashcards that enhances biology vocabulary memorization through a custom spaced repetition model, which analyzes user performance and the difficulty of the questions and will repeat accordingly.

## Objective 2

Incorporating multi-sensory elements to cater to different learning styles and enhance the memorization process.

## Objective 3

Incorporate gamification elements to maintain user motivation and engagement throughout the learning process

## Objective 4

Implement a model to check the accuracy of the word that was entered.

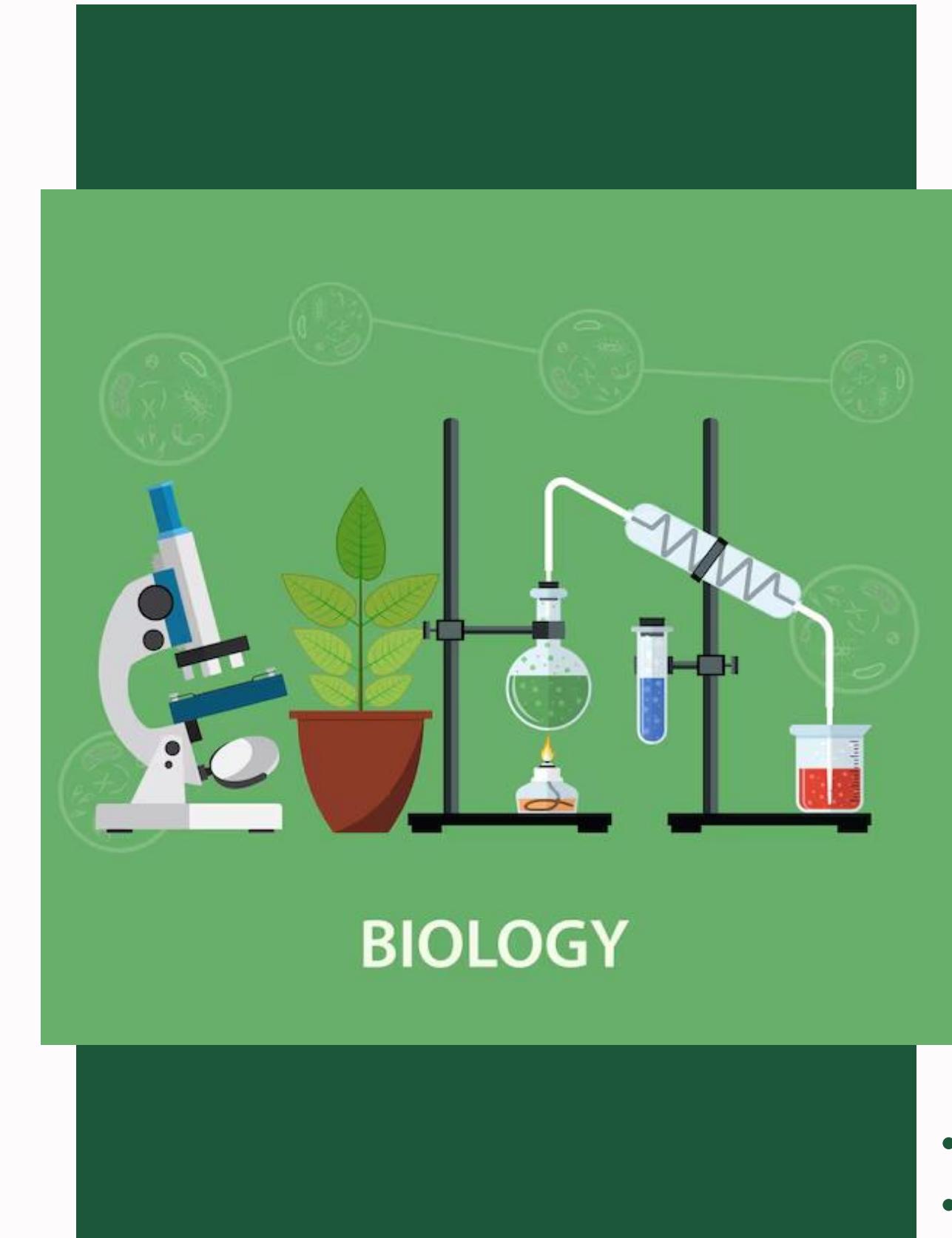
# Methodology

01 System Diagram

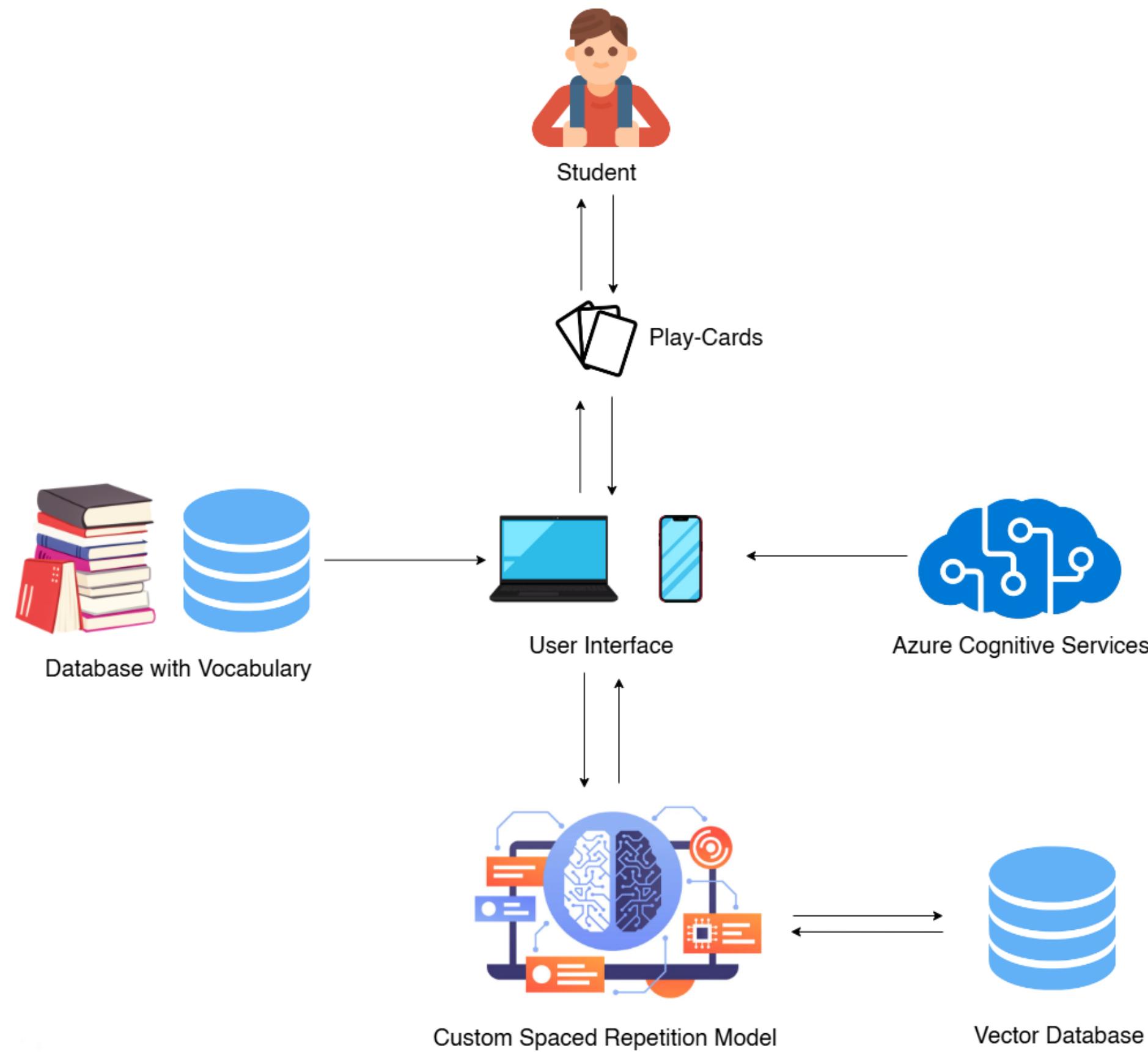
02 Tools and Technologies

03 Best Practices

04 GitHub Commits



# System Diagram



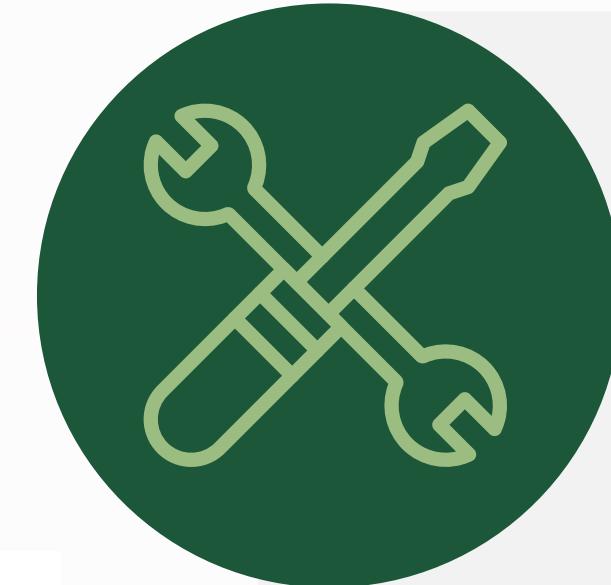
# Tools & Technologies



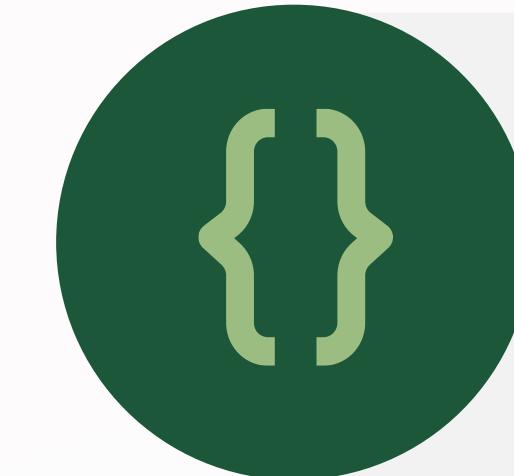
**Project Management**  
Jira



**Database**  
Faiss  
Mongo DB



**Other tools**  
Git  
Figma  
Postman  
Azure Cognitive Services



**Programming Languages**  
Python  
Javascript



**Frameworks**  
Flask  
React  
Tensorflow

# Best Practices

## Non-Functional Requirements

- Compatibility
  - Accuracy
- Performance
  - Usability
- Availability

## Design Excellence

- Real-Time Performance
- Accuracy and Reliability
- User Experience
  - Integration

## Standards & Best Practices

- Code Structure & Organization
- Web Security
- Model Quality
- RESTful API Design

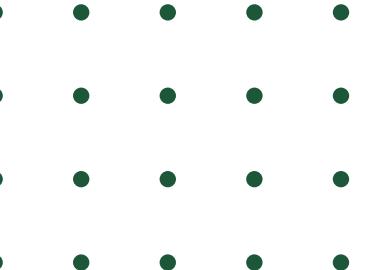
# GitHub Commits

The screenshot displays a GitHub commit history interface. At the top, a dark header bar contains the title 'GitHub Commits'. Below this, a light-colored sidebar lists commits grouped by date. The first group, 'Commits on Dec 4, 2024', shows ten commits made by the user 'Gokul\_Abishek' in the last hour. The second group, 'Commits on Nov 28, 2024', shows three commits made by the user 'DharaneSegar' in the last week. Each commit entry includes a small profile picture, the commit message, the author's name and time, the commit hash, and standard GitHub actions like copy and diff buttons.

- o- Commits on Dec 4, 2024
- implemented flashcard component  
Gokul\_Abishek committed 1 minute ago f1bcf45 ⌂ ↗
- added mock vocabulary data  
Gokul\_Abishek committed 45 minutes ago 0663725 ⌂ ↗
- configured tailwindcss  
Gokul\_Abishek committed 47 minutes ago 2960784 ⌂ ↗
- display vocabulary after review quality rating  
Gokul\_Abishek committed 50 minutes ago ed1a97d ⌂ ↗
- initialized frontend  
Gokul\_Abishek committed 53 minutes ago 290119b ⌂ ↗
- added gitignore file  
Gokul\_Abishek committed 1 hour ago d3834ba ⌂ ↗
- implemented main entry point with example usage  
Gokul\_Abishek committed 1 hour ago a3dec8b ⌂ ↗
- implemented review quality rating algorithm with Levenshtein distance algorithm  
Gokul\_Abishek committed 1 hour ago d9b873d ⌂ ↗
- created flashcards logic and implemented SM-2 algorithm  
Gokul\_Abishek committed 1 hour ago 58dc79d ⌂ ↗
- added vocabulary dataset  
Gokul\_Abishek committed 1 hour ago f818516 ⌂ ↗
- o- Commits on Nov 28, 2024
- Merge pull request #4 from Y3S1-GRP22/master  
DharaneSegar authored last week Verified ef64163 ⌂ ↗
- Updated folder structure  
DharaneSegar committed last week 6c2765b ⌂ ↗
- Initial commit  
DharaneSegar committed last week 18873d8 ⌂ ↗

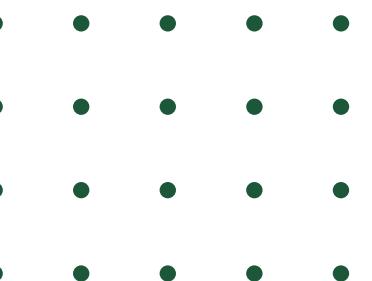


# Completion of the project

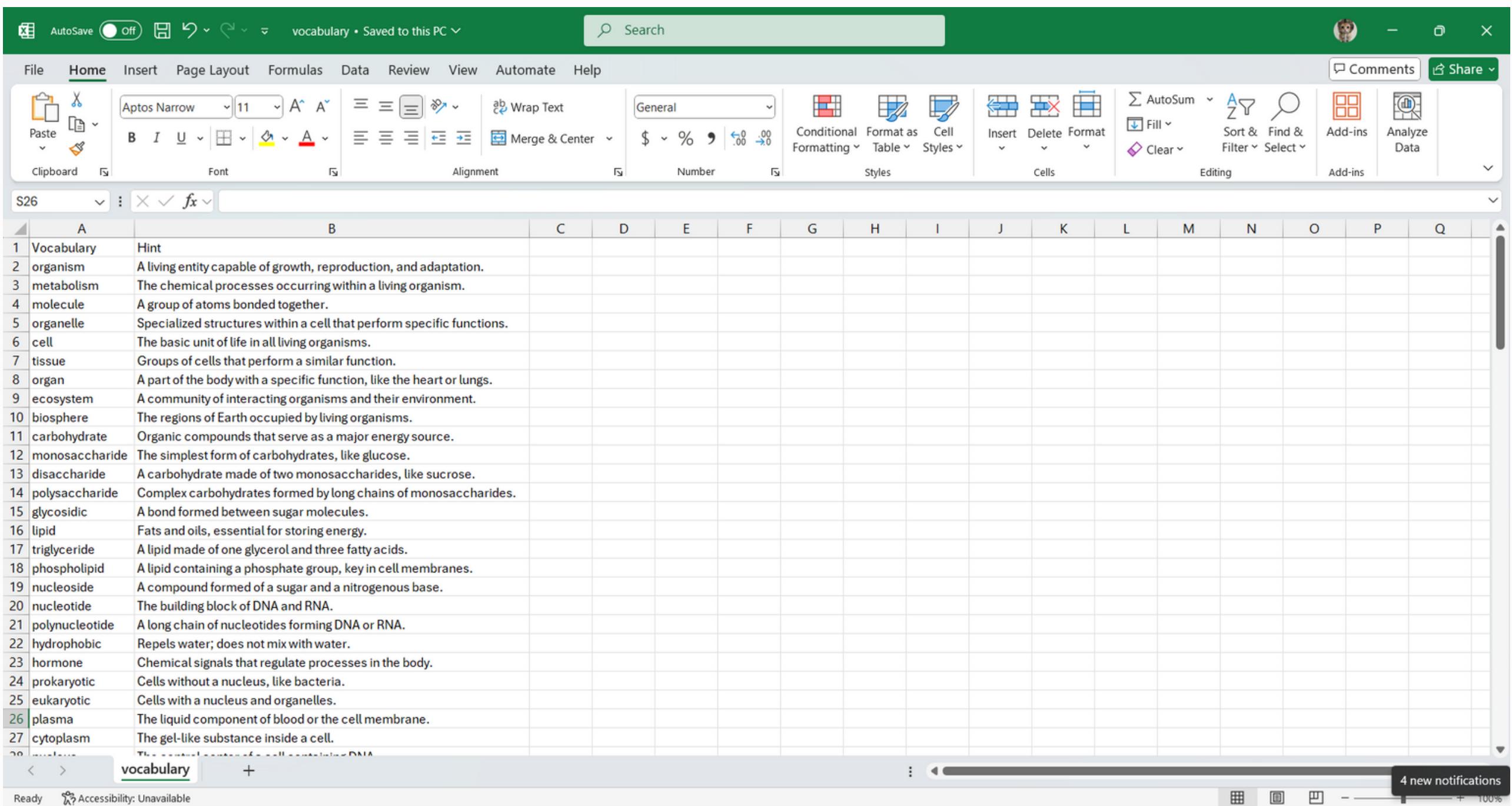




# Completion of the project



# Data Collection



A screenshot of a Microsoft Excel spreadsheet titled "vocabulary". The spreadsheet contains two columns: Column A lists biological terms, and Column B lists their definitions. The rows are numbered from 1 to 27. The "Home" tab is selected in the ribbon. The status bar at the bottom shows "Ready" and "4 new notifications".

1	Vocabulary
2	organism
3	metabolism
4	molecule
5	organelle
6	cell
7	tissue
8	organ
9	ecosystem
10	biosphere
11	carbohydrate
12	monosaccharide
13	disaccharide
14	polysaccharide
15	glycosidic
16	lipid
17	triglyceride
18	phospholipid
19	nucleoside
20	nucleotide
21	polynucleotide
22	hydrophobic
23	hormone
24	prokaryotic
25	eukaryotic
26	plasma
27	cytoplasm

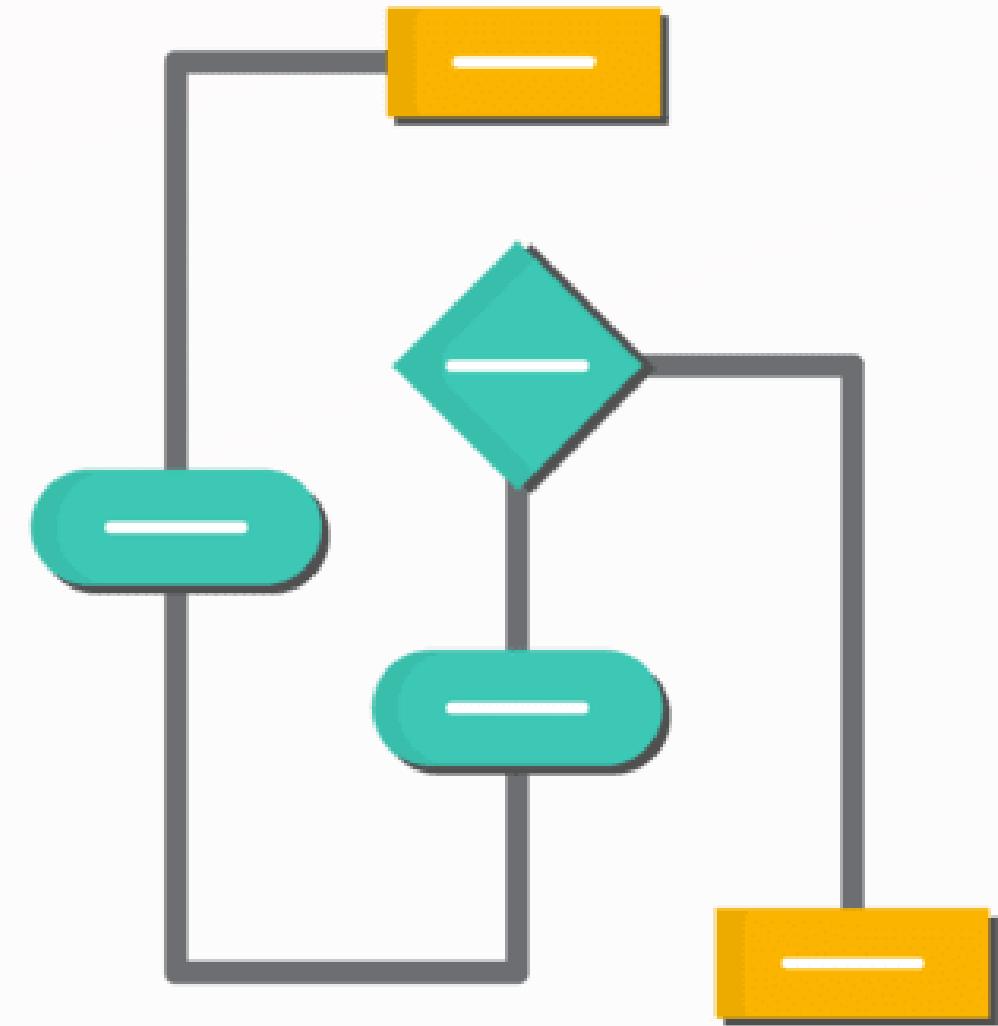
<https://www.nie.lk/pdffiles/tg/eALSyl%20Biology.pdf>

<https://www.nie.lk/pdffiles/other/eGr12OM%20BioResoBook.pdf>

# Spaced Repetition Algorithm Selection

## Why SM-2?

- Scientifically proven retention.
- Highly customizable algorithm.
- Dynamically adjusts review schedules according to user performance.



• • • •  
• • • •  
• • • •  
• • • •  
• • • •

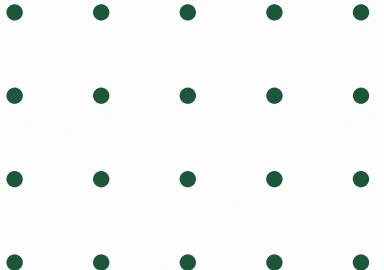
# SM-2 Algorithm

$$E' = E + 0.1 - (5 - Q) * (0.08 + (5 - Q) * 0.02)$$

$$\text{Interval}' = \begin{cases} 1 & \text{if repetitions} = 1 \\ 6 & \text{if repetitions} = \\ \text{Interval} * E & 2 \\ & \text{if repetitions} > 2 \end{cases}$$

If  $Q < 3$

1. Reset repetitions to 0
2. Set interval to 1



# SM-2 Algorithm

```
import datetime
import math
import random
import itertools
import csv

# Define the time format for consistent display
time_fmt = "%Y-%m-%d"

class Card:
    """
    Represents a single flashcard used in spaced repetition.
    Attributes:
        - top: The front side of the card (e.g., question, term).
        - bot: The back side of the card (e.g., answer, definition).
        - time: The last review time.
        - repetitions: Number of successful repetitions.
        - interval: Days until the next review.
        - easiness: Easiness factor (affects interval growth).
    """
    def __init__(self, top, bot, time, repetitions=0, interval=1, easiness=2.5):
        self.top = top
        self.bot = bot
        self.time = time.replace(second=0, microsecond=0)
        self.repetitions = repetitions
        self.interval = interval
        self.easiness = easiness

    @property
    def is_new(self):
        """Determine if the card is new (has not been reviewed yet)."""
        return self.repetitions == 0

    @property
    def next_time(self):
        """Calculate the next review date based on the current interval."""
        return self.time + datetime.timedelta(days=math.ceil(self.interval))

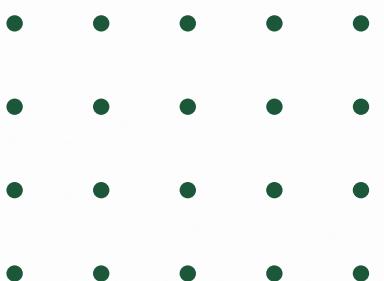
    def repeat(self, quality, time):
        """
        Update the card's state using the SM-2 algorithm.
        Parameters:
            - quality: Review quality (0-5) based on user performance.
            - time: The current datetime.
        """
        assert 0 <= quality <= 5, "Quality should be between 0 and 5."

        # Adjust easiness factor based on quality
        self.easiness = max(1.3, self.easiness + 0.1 - (5 - quality) * (0.08 + (5 - quality) * 0.02))

        if quality < 3:
            # If performance is poor, reset repetitions and interval
            self.repetitions = 0
            self.interval = 1
        else:
            # If performance is acceptable, increment repetitions
            self.repetitions += 1
            if self.repetitions == 1:
                self.interval = 1 # First repetition interval
            elif self.repetitions == 2:
                self.interval = 6 # Second repetition interval
            else:
                # Subsequent intervals grow based on easiness factor
                self.interval *= self.easiness

        # Update the review time
        self.time = time

    def __repr__(self):
        """Provide a human-readable representation of the card."""
        return (f"Card: {self.bot}, Next Review: {self.next_time.strftime(time_fmt)}, "
               f"Repetitions: {self.repetitions}, Interval: {self.interval:.2f}, Easiness: {self.easiness:.2f}")
```



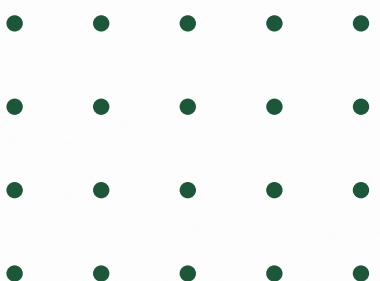
# Determine Review Quality

## Why Levenshtein Distance Algorithm?

- Captures detailed differences between text inputs providing Accuracy.
- Error Handling allows scores and identifies common user mistakes.
- Customizable thereby can be used for Quantitative Scoring.

H	O		N	D	A	
H	Y	U	N	D	A	I

H	Y	U	N	D	A	I
H		O	N	D	A	



# Levenshtein Distance Algorithm

```
from Levenshtein import distance

def word_accuracy(original_word, entered_word):
    """
    Calculate the accuracy score between two words.

    Parameters:
    - original_word (str): The correct word.
    - entered_word (str): The word entered by the user.

    Returns:
    - int: Accuracy score between 0 (worst) and 5 (best).
    """
    # Normalize the words by converting to lowercase
    original_word = original_word.lower()
    entered_word = entered_word.lower()

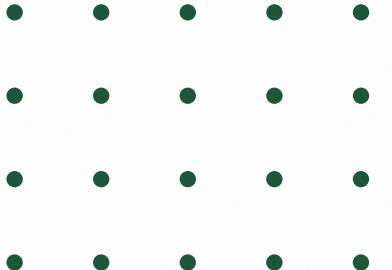
    # Calculate Levenshtein distance
    max_length = max(len(original_word), len(entered_word))
    if max_length == 0: # Handle edge case where both strings are empty
        return 5

    edit_distance = distance(original_word, entered_word)

    # Calculate similarity percentage
    similarity = (max_length - edit_distance) / max_length

    # Scale the similarity to a score between 0 and 5
    scaled_score = round(similarity * 5)

    return max(0, min(scaled_score, 5)) # Ensure score is within bounds
```



# Output

Day 1: Reviewing cards on 2024-12-04

Front: A living entity capable of growth, reproduction, and adaptation.  
Enter your answer: organism  
Calculated quality rating: 5

Front: The chemical processes occurring within a living organism.  
Enter your answer: metabolism  
Calculated quality rating: 5

Front: A group of atoms bonded together.  
Enter your answer: molecules  
Calculated quality rating: 4

Front: Specialized structures within a cell that perform specific functions.  
Enter your answer: organ  
Calculated quality rating: 3

Front: The basic unit of life in all living organisms.  
Enter your answer: cell  
Calculated quality rating: 5

Daily Review Log:  
- {'Card': 'A living entity capable of growth, reproduction, and adaptation.', 'Quality': 5, 'Next Review': '2024-12-05'}  
- {'Card': 'The chemical processes occurring within a living organism.', 'Quality': 5, 'Next Review': '2024-12-05'}  
- {'Card': 'A group of atoms bonded together.', 'Quality': 4, 'Next Review': '2024-12-05'}  
- {'Card': 'Specialized structures within a cell that perform specific functions.', 'Quality': 3, 'Next Review': '2024-12-05'}  
- {'Card': 'The basic unit of life in all living organisms.', 'Quality': 5, 'Next Review': '2024-12-05'}

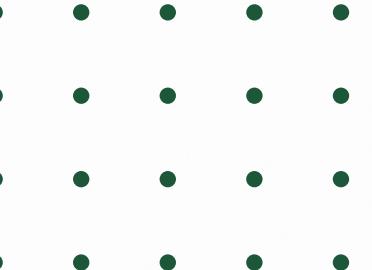
Updated Card States:

Card: organism, Next Review: 2024-12-05, Repetitions: 1, Interval: 1.00, Easiness: 2.60  
Card: metabolism, Next Review: 2024-12-05, Repetitions: 1, Interval: 1.00, Easiness: 2.60  
Card: molecule, Next Review: 2024-12-05, Repetitions: 1, Interval: 1.00, Easiness: 2.50  
Card: organelle, Next Review: 2024-12-05, Repetitions: 1, Interval: 1.00, Easiness: 2.36  
Card: cell, Next Review: 2024-12-05, Repetitions: 1, Interval: 1.00, Easiness: 2.60  
Card: tissue, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: organ, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: ecosystem, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: biosphere, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: carbohydrate, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: monosaccharide, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: disaccharide, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: polysaccharide, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: glycosidic, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: lipid, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: triglyceride, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: phospholipid, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: nucleoside, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: nucleotide, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50

Updated Card States:

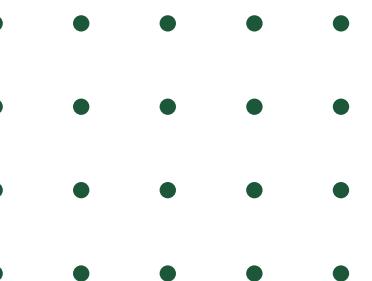
Card: organism, Next Review: 2024-12-11, Repetitions: 2, Interval: 6.00, Easiness: 2.60  
Card: metabolism, Next Review: 2024-12-11, Repetitions: 2, Interval: 6.00, Easiness: 2.70  
Card: molecule, Next Review: 2024-12-11, Repetitions: 2, Interval: 6.00, Easiness: 2.50  
Card: organelle, Next Review: 2024-12-11, Repetitions: 2, Interval: 6.00, Easiness: 2.22  
Card: cell, Next Review: 2024-12-11, Repetitions: 2, Interval: 6.00, Easiness: 2.70  
Card: tissue, Next Review: 2024-12-12, Repetitions: 2, Interval: 6.00, Easiness: 2.60  
Card: organ, Next Review: 2024-12-12, Repetitions: 2, Interval: 6.00, Easiness: 2.70  
Card: ecosystem, Next Review: 2024-12-07, Repetitions: 1, Interval: 1.00, Easiness: 2.06  
Card: biosphere, Next Review: 2024-12-07, Repetitions: 0, Interval: 1.00, Easiness: 1.30  
Card: carbohydrate, Next Review: 2024-12-12, Repetitions: 2, Interval: 6.00, Easiness: 2.70  
Card: monosaccharide, Next Review: 2024-12-07, Repetitions: 1, Interval: 1.00, Easiness: 2.50  
Card: disaccharide, Next Review: 2024-12-07, Repetitions: 1, Interval: 1.00, Easiness: 2.60  
Card: polysaccharide, Next Review: 2024-12-07, Repetitions: 1, Interval: 1.00, Easiness: 2.60  
Card: glycosidic, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: lipid, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: triglyceride, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: phospholipid, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: nucleoside, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: nucleotide, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50  
Card: polynucleotide, Next Review: 2024-12-05, Repetitions: 0, Interval: 1.00, Easiness: 2.50

# Flashcard UI



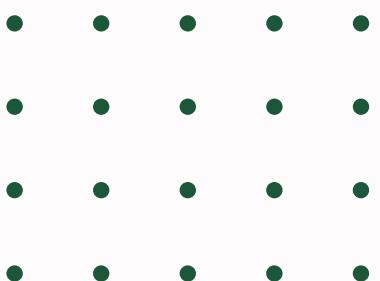


# Completion of the project



# Key Functionalities

- Spaced Repetition Algorithm (SM-2): Adjusts review intervals based on user performance to reinforce learning before forgetting.
- Daily Flashcard Quiz: Provides 10 vocabulary words daily, prioritizing due reviews before adding new ones.
- Audio Pronunciation : Integrated text-to-speech (TTS) allows users to hear correct word pronunciations for better retention.
- Performance Tracking & Leaderboard : Tracks user progress, review history, and ranks users based on total scores.



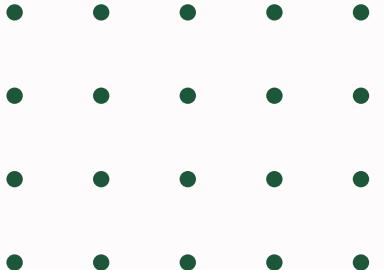
# SM-2 Algorithm

$$E' = E + 0.1 - (5 - Q) * (0.08 + (5 - Q) * 0.02)$$

Interval' =  $\begin{cases} 1 & \text{if repetitions} = 1 \\ 6 & \text{if repetitions} = \\ \text{Interval} * E & 2 \\ & \text{if repetitions} > 2 \end{cases}$

If  $Q < 3$

1. Reset repetitions to 0
2. Set interval to 1



# SM-2 Algorithm

```
async def assign_daily_questions():
    """Assign exactly 10 random flashcards from MongoDB."""
    cursor = db["flashcards"].aggregate([{"$sample": {"size": 10}}])
    questions = await cursor.to_list(length=10)

    # Convert ObjectId to string
    for q in questions:
        q["_id"] = str(q["_id"])

    return questions

async def get_daily_questions(user_id):
    """Fetch up to 10 daily vocabulary words for a user based on spaced repetition."""
    user = await db["users"].find_one({"_id": ObjectId(user_id)})
    if not user:
        return {"error": "User not found"}

    today = datetime.utcnow().date()
    due_questions = []

    # Collect words due for review
    for entry in user.get("history", []):
        if datetime.strptime(entry["next_review"], "%Y-%m-%d").date() <= today:
            due_questions.append(ObjectId(entry["question_id"]))

    # Fetch due words from MongoDB
    questions = await db["flashcards"].find({"_id": {"$in": due_questions}}).to_list(len(due_questions))

    # Convert _id to string
    seen_ids = {q["_id"] for q in questions}

    # If fewer than 10, fetch additional new ones
    remaining_slots = max(0, 10 - len(questions))
    if remaining_slots > 0:
        cursor = db["flashcards"].aggregate([
            {"$match": {"_id": {"$nin": list(seen_ids)}}},
            {"$sample": {"size": remaining_slots}}
        ])
        new_questions = await cursor.to_list(length=remaining_slots)
        questions.extend(new_questions)

    # Strictly limit to 10 questions
    questions = questions[:10]

    # Convert ObjectId to string for JSON serialization
    for q in questions:
        q["_id"] = str(q["_id"])

    return {"questions": questions}
```

```
async def update_progress(user_id, question_id, user_answer):
    """Update spaced repetition progress for a specific user."""
    from datetime import datetime, timedelta
    from utils.review_quality import word_accuracy

    # Fetch the vocabulary word
    question = await db["flashcards"].find_one({"_id": ObjectId(question_id)})
    if not question:
        return {"error": "Question not found"}

    # ✅ Compare with vocabulary word, not the hint
    review_score = word_accuracy(question["question"], user_answer)

    # Fetch or create user record
    user = await db["users"].find_one({"_id": ObjectId(user_id)})
    if not user:
        user = {
            "_id": ObjectId(user_id),
            "username": f"user_{user_id}", # Temporary username
            "total_score": 0,
            "history": []
        }
        await db["users"].insert_one(user)

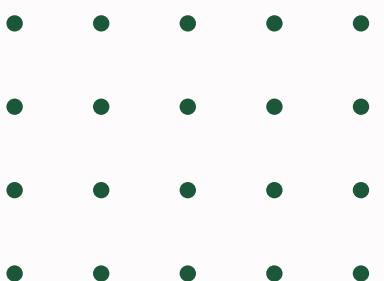
    # Check if question exists in user history
    history_entry = next((h for h in user["history"] if h["question_id"] == str(question_id)), None)

    if history_entry is None:
        # If first time seeing the word, create a new entry
        next_review = datetime.utcnow() + timedelta(days=1)
        history_entry = {
            "question_id": str(question_id),
            "times_seen": 1,
            "review_score": review_score,
            "next_review": next_review.strftime("%Y-%m-%d")
        }
        await db["users"].update_one(
            {"_id": ObjectId(user_id)},
            {"$push": {"history": history_entry}, "$inc": {"total_score": review_score}}
        )
    else:
        # Update existing entry
        history_entry["times_seen"] += 1
        history_entry["review_score"] = review_score

        # ✅ Implement **Spaced Repetition (SM-2 Algorithm)**
        if review_score == 5:
            interval = 10 # Maximum interval
        elif review_score >= 4:
            interval = 5
        elif review_score >= 3:
            interval = 3
        elif review_score >= 2:
            interval = 2
        else:
            interval = 1 # Immediate review required

        history_entry["next_review"] = (datetime.utcnow() + timedelta(days=interval)).strftime("%Y-%m-%d")

        await db["users"].update_one(
            {"_id": ObjectId(user_id), "history.question_id": str(question_id)},
            {"$set": {"history.$": history_entry}, "$inc": {"total_score": review_score}}
        )
```



# Levenshtein Distance

```
from Levenshtein import distance

def word_accuracy(original_word, entered_word):
    """
    Calculate the accuracy score between two words using Levenshtein distance.

    Parameters:
    - original_word (str): The correct word.
    - entered_word (str): The word entered by the user.

    Returns:
    - int: Accuracy score between 0 (worst) and 5 (best).
    """

    # Handle empty cases
    if not original_word or not entered_word:
        return 0 # If the answer is empty, give the lowest score

    # Normalize the words by converting to lowercase and stripping spaces
    original_word = original_word.lower().strip()
    entered_word = entered_word.lower().strip()

    # Calculate Levenshtein distance
    max_length = max(len(original_word), len(entered_word))

    # If both are empty after stripping, return max score
    if max_length == 0:
        return 5

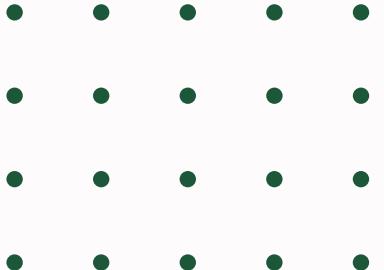
    edit_distance = distance(original_word, entered_word)

    # Calculate similarity percentage
    similarity = (max_length - edit_distance) / max_length

    # Scale similarity to 0-5
    if similarity >= 0.95:
        return 5
    elif similarity >= 0.85:
        return 4
    elif similarity >= 0.70:
        return 3
    elif similarity >= 0.50:
        return 2
    elif similarity >= 0.30:
        return 1
    else:
        return 0 # Too different, return lowest score

# ✅ **Test Cases**
test_cases = [
    ("biology", "biology"), # Expected: 5 (Exact match)
    ("biology", "biolog"), # Expected: 4 (Small typo)
    ("biology", "biolgy"), # Expected: 3 (One letter swap)
    ("biology", "bio"), # Expected: 2 (Partial match)
    ("biology", "chemistry"), # Expected: 0 (Completely different)
    ("hello", ""), # Expected: 0 (Empty answer)
    ("", ""), # Expected: 5 (Both empty)
]

for original, entered in test_cases:
    print(f"Original: {original}, Entered: {entered}, Score: {word_accuracy(original, entered)}")
```



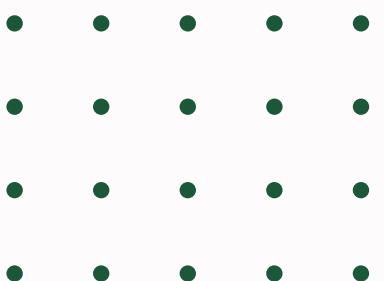
# Working Product

The screenshot shows a web application interface for 'Bio Mentor'. At the top, there's a dark header bar with the 'Bio Mentor' logo on the left and navigation links for 'Home', 'MCQ', 'Q & A', 'Vocabulary Memorization' (which is highlighted in blue), and 'Summarize' on the right. A user profile icon is also present.

A central feature is a white pop-up window titled 'Vocabulary Mastery'. It displays a progress bar at the top indicating 'Score: 0' and 'Question 1 / 10'. Below the bar, the question 'Brown algae found in marine environments.' is shown. There's a text input field labeled 'Enter the term...' and a blue 'Submit' button.

In the background, a large, semi-transparent watermark image of a book cover is visible. The book cover features various biology-related words like 'energy', 'water', 'genetic', 'form', 'animal', 'plant', 'evolution', 'life', 'organism', and 'Science' in a collage-like arrangement.

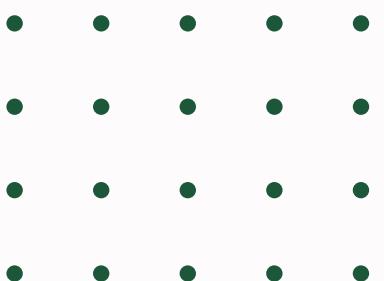
On the right side of the main content area, there's a sidebar with the title 'Vocabulary Memory'. It contains two sections: 'SM-2 Algorithm' (described as a scientifically proven method to improve long-term retention) and 'Leaderboard' (described as a way to compete with peers and track vocabulary mastery rank).



# Working Product

The screenshot shows a web-based application for learning biology vocabulary. At the top, there's a navigation bar with the logo "Bio Mentor" and links for "Home", "MCQ", "Q & A", "Vocabulary Memorization" (which is currently selected), and "Summarize". A user profile icon is also present.

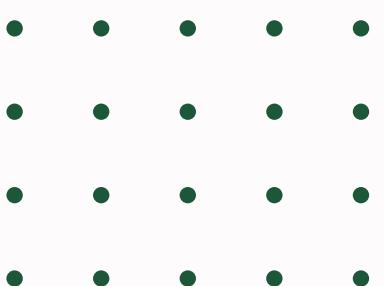
The main content area features a large, semi-transparent background image of a brain composed of various biological terms like "energy", "water", "species", "animal", "plant", "cell", "life", and "organism". Overlaid on this is a central modal window titled "Vocabulary Mastery". Inside the modal, the word "cuboidal" is displayed. Below it, there are two buttons: "Next" (green) and "Pronounce" (blue). Above the word, the text "Score: 2" and "Question 2 / 10" are shown. To the right of the modal, there's a sidebar with the title "Improve Vocabulary Memory" and two sections: "SM-2 Algorithm" (described as a scientifically proven method to improve long-term retention) and "Leaderboard" (which allows users to compete with peers and track their vocabulary mastery rank).



# Working Product

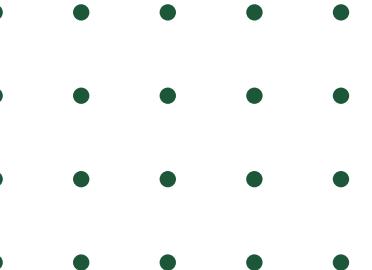
The screenshot shows a dark-themed user interface for the "Bio Mentor" application. At the top, there is a navigation bar with links for "Home", "MCQ", "Q & A", "Vocabulary Memorization", "Summarize", and a user profile icon. Below the navigation bar, there is a section titled "Friendly Competition" with three options: "Earn Points", "Compete with Friends", and "Share Your Score". A prominent feature is a "Leaderboard" overlay window that has popped up. This window has a title "Leaderboard" with a trophy icon. It displays a list of three users and their scores: 1. Gokul (15 points), 2. John (12 points), and 3. JohnDoe (5 points). To the right of the list is a graphic of a podium with a large yellow star on top. Below the leaderboard, there is a button labeled "View Leaderboard". To the right of the main content area, there is a dark sidebar with the heading "Compete and Learn" and the text: "Challenge yourself with the SM-2 spaced repetition algorithm and track your progress on the Leaderboard. Earn points, rank higher, and share your score with friends!". At the bottom right of the screen, there is a "Show desktop" button.

Rank	User	Score
1.	Gokul	15 🏆
2.	John	12
3.	JohnDoe	5



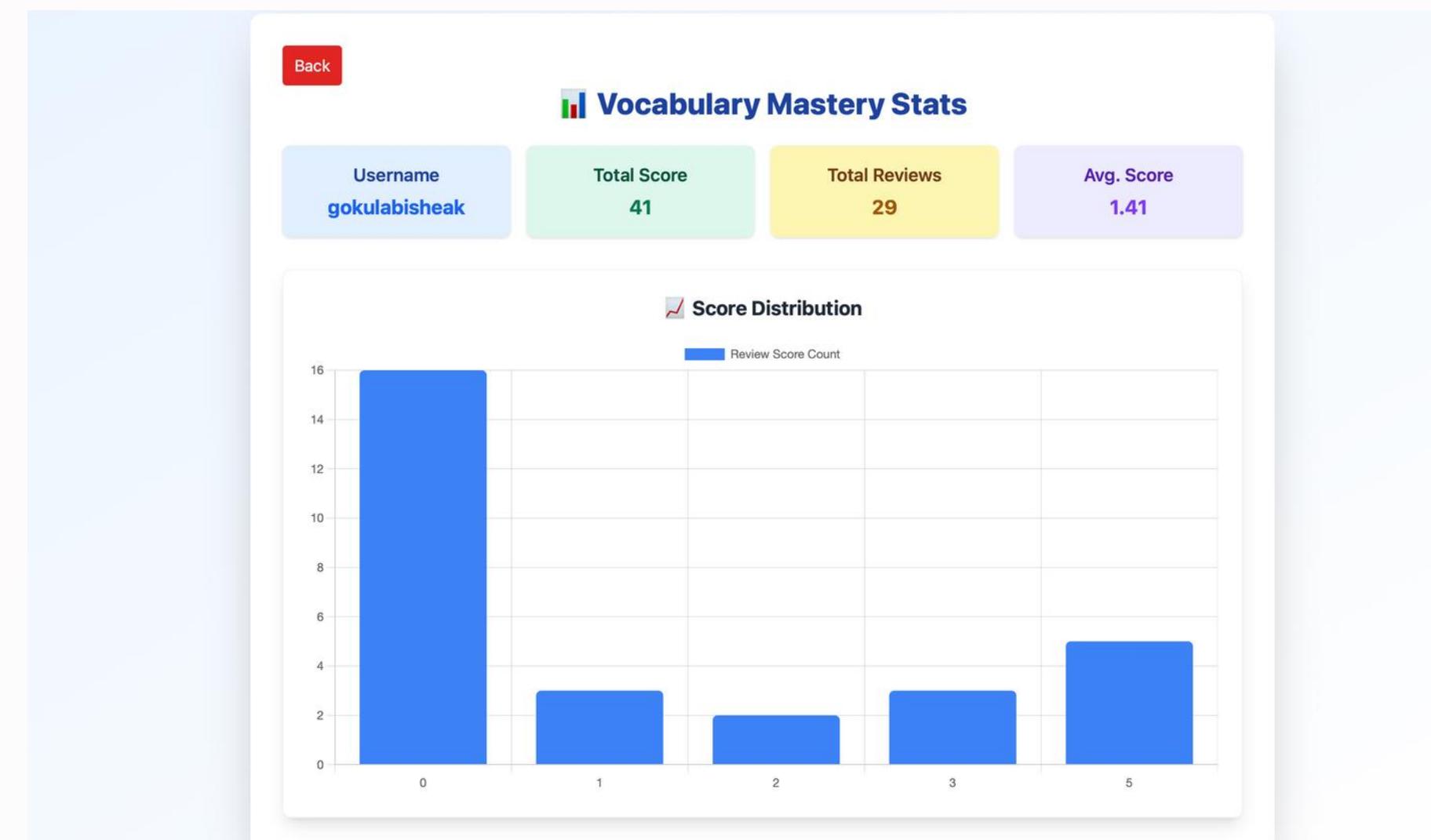


# Completion of the project



# Project Enhancements

**Progress Tracking Dashboard** - Implemented a progress tracking dashboard for users to track their progress and to view statistics about their performance.



# Testing - Unit Testing

- Framework: Pytest

```
===== test session starts =====
platform darwin -- Python 3.13.3, pytest-8.0.2, pluggy-1.6.0 -- /Users/gokul/Repos/BioMentor-Personalized-E-Learning-Platform/Back-End/Vocabulary/venv/bin/python3.13
cachedir: .pytest_cache
rootdir: /Users/gokul/Repos/BioMentor-Personalized-E-Learning-Platform/Back-End/Vocabulary
configfile: pytest.ini
plugins: asyncio-0.23.5, cov-4.1.0, anyio-4.8.0
asyncio: mode=Mode.STRICT
collected 4 items

tests/unit/test_models.py::test_flashcard_model PASSED [ 25%]
tests/unit/test_models.py::test_user_progress_model PASSED [ 50%]
tests/unit/test_models.py::test_user_model PASSED [ 75%]
tests/unit/test_models.py::test_leaderboard_entry_model PASSED [100%]

===== warnings summary =====
venv/lib/python3.13/site-packages/pydantic/_internal/_config.py:295
venv/lib/python3.13/site-packages/pydantic/_internal/_config.py:295
venv/lib/python3.13/site-packages/pydantic/_internal/_config.py:295
venv/lib/python3.13/site-packages/pydantic/_internal/_config.py:295
    /Users/gokul/Repos/BioMentor-Personalized-E-Learning-Platform/Back-End/Vocabulary/venv/lib/python3.13/site-packages/pydantic/_internal/_config.py:295: PydanticDeprecatedSince20: Support for class-based `config` is deprecated, use ConfigDict instead. Deprecated in Pydantic V2.0 to be removed in V3.0. See Pydantic V2 Migration Guide at https://errors.pydantic.dev/2.10/migration/
        warnings.warn(DEPRECATION_MESSAGE, DeprecationWarning)

venv/lib/python3.13/site-packages/pydantic/_internal/_config.py:345
    /Users/gokul/Repos/BioMentor-Personalized-E-Learning-Platform/Back-End/Vocabulary/venv/lib/python3.13/site-packages/pydantic/_internal/_config.py:345: UserWarning: Valid config keys have changed in V2:
        * 'allow_population_by_field_name' has been renamed to 'populate_by_name'
        warnings.warn(message, UserWarning)

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
===== 4 passed, 5 warnings in 0.07s =====
```

```
1 import pytest
2 from datetime import datetime
3 from bson import ObjectId
4
5 from models import Flashcard, UserProgress, User, LeaderboardEntry
6
7 def test_flashcard_model():
8     """Test Flashcard model creation and validation."""
9     # Test with string ID
10    flashcard = Flashcard(
11        _id="507f1f77bcf86cd799439011",
12        question="What is photosynthesis?",
13        answer="Process by which plants make food"
14    )
15    assert flashcard.id == "507f1f77bcf86cd799439011"
16    assert flashcard.question == "What is photosynthesis?"
17    assert flashcard.answer == "Process by which plants make food"
18
19    # Test with ObjectId
20    obj_id = ObjectId()
21    flashcard = Flashcard(
22        _id=str(obj_id), # Convert ObjectId to string
23        question="What is mitosis?",
24        answer="Cell division process"
25    )
26    assert flashcard.id == str(obj_id)
27
28 def test_user_progress_model():
29     """Test UserProgress model creation and validation."""
30     progress = UserProgress(
31         _id="507f1f77bcf86cd799439011",
32         total_score=100,
33         history=[{"question_id": "123", "score": 5}]
34     )
35     assert progress.id == "507f1f77bcf86cd799439011"
36     assert progress.total_score == 100
37     assert len(progress.history) == 1
38     assert progress.history[0]["question_id"] == "123"
39
40 def test_user_model():
41     """Test User model creation and validation."""
42     user = User(
43         _id="507f1f77bcf86cd799439011",
44         username="test_user",
45         total_score=150,
46         history:[
47             {"question_id": "123",
48              "times_seen": 2,
49              "review_score": 4,
50              "next_review": "2024-03-20"
51          }]
52     )
53     assert user.id == "507f1f77bcf86cd799439011"
54     assert user.username == "test_user"
55     assert user.total_score == 150
56     assert len(user.history) == 1
57     assert user.history[0]["times_seen"] == 2
58
59 def test_leaderboard_entry_model():
60     """Test LeaderboardEntry model creation and validation."""
61     entry = LeaderboardEntry(
62         _id="507f1f77bcf86cd799439011",
63         username="top_user",
64         score=1000
65     )
66     assert entry.id == "507f1f77bcf86cd799439011"
67     assert entry.username == "top_user"
68     assert entry.score == 1000
```

# Testing - Integration Testing

- Framework: Pytest with asyncio

```
(venv) gokul@Gokuls-MacBook-Air Vocabulary % pytest tests/integration/ -v
=====
platform darwin -- Python 3.13.3, pytest-8.0.2, pluggy-1.6.0 -- /Users/gokul/Repos/BioMentor-Personalized-E-Learning-Platform/Back-End/Vocabulary/venv/bin/python3.13
cachedir: .pytest_cache
rootdir: /Users/gokul/Repos/BioMentor-Personalized-E-Learning-Platform/Back-End/Vocabulary
configfile: pytest.ini
plugins: asyncio-0.23.5, cov-4.1.0, anyio-4.8.0
asyncio: mode=Mode.STRICT
collected 9 items

tests/integration/test_database.py::test_load_vocab_into_db PASSED
tests/integration/test_database.py::test_database_connection PASSED
tests/integration/test_services.py::test_assign_daily_questions PASSED
tests/integration/test_services.py::test_get_daily_questions PASSED
tests/integration/test_services.py::test_update_progress PASSED
tests/integration/test_services.py::test_add_score PASSED
tests/integration/test_services.py::test_get_rank PASSED
tests/integration/test_services.py::test_get_top_5 PASSED
tests/integration/test_services.py::test_get_user_stats PASSED

===== warnings summary =====
tests/integration/test_database.py::test_load_vocab_into_db
  /Users/gokul/Repos/BioMentor-Personalized-E-Learning-Platform/Back-End/Vocabulary/venv/lib/python3.13/site-packages/pytest_asyncio/plugin.py:769: DeprecationWarning: The event_loop fixture provided by pytest-asyncio has been redefined in
  /Users/gokul/Repos/BioMentor-Personalized-E-Learning-Platform/Back-End/Vocabulary/tests/integration/conftest.py:23
    Replacing the event_loop fixture with a custom implementation is deprecated
    and will lead to errors in the future.
    If you want to request an asyncio event loop with a scope other than function
    scope, use the "scope" argument to the asyncio mark when marking the tests.
    If you want to return different types of event loops, use the event_loop_policy
    fixture.

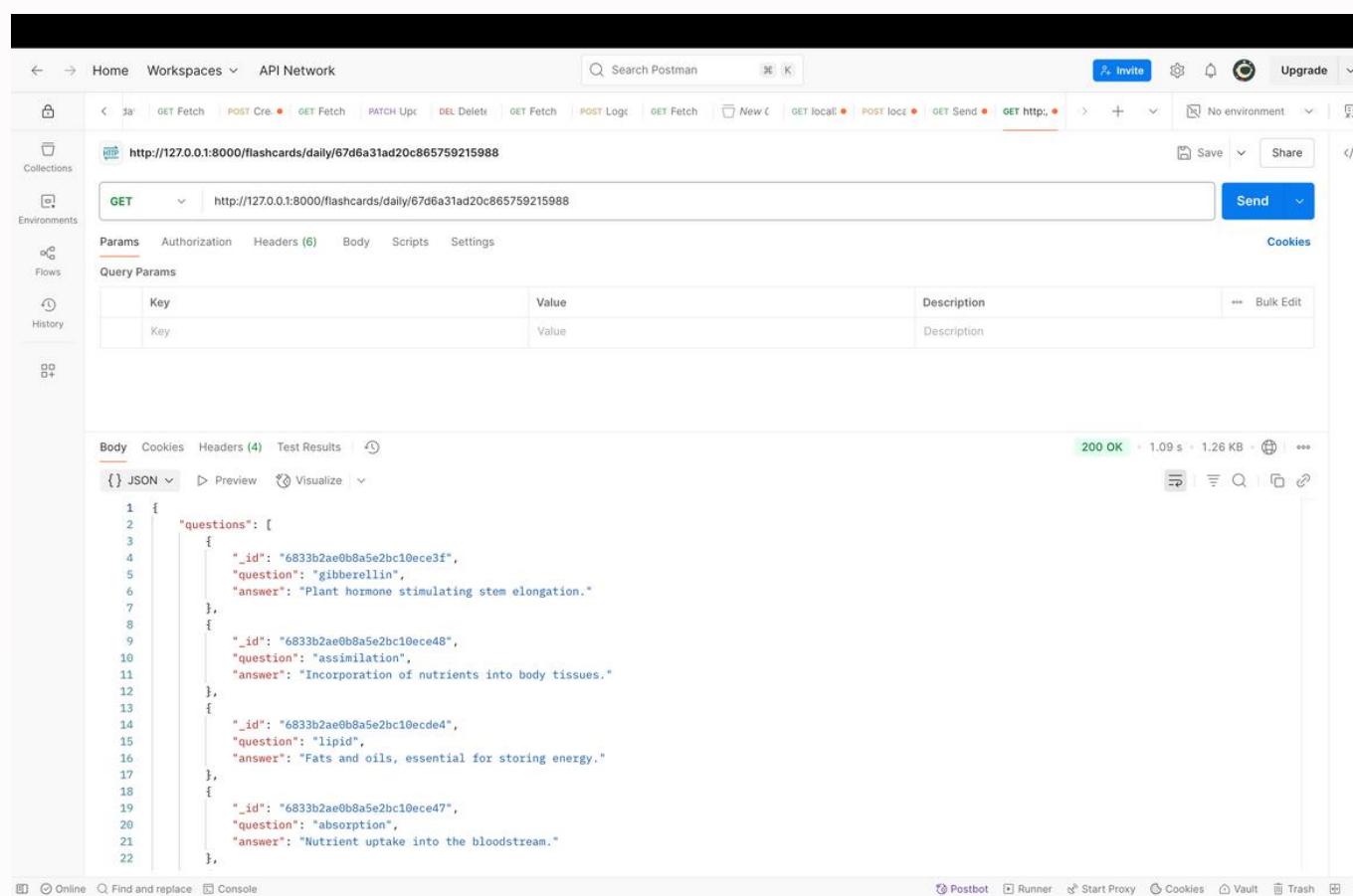
  warnings.warn(
    -- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
  = 9 passed, 1 warning in 11.51s =
```

```
1 import pytest
2 from datetime import datetime, timedelta
3 from bson import ObjectId
4
5 from database import db
6 from services import {
7     assign_daily_questions,
8     get_daily_questions,
9     update_progress,
10    add_score,
11    get_rank,
12    get_top_5,
13    get_user_stats
14 }
15
16 pytestmark = pytest.mark.asyncio
17
18 @pytest.fixture
19 async def setup_test_data():
20     """Set up test data for services tests."""
21     # Clear existing data
22     await db["flashcards"].delete_many({})
23     await db["users"].delete_many({})
24     await db["leaderboard"].delete_many({})
25
26     # Insert test flashcards
27     test_flashcards = [
28         {
29             "_id": ObjectId(),
30             "question": "What is photosynthesis?",
31             "answer": "Process by which plants make food"
32         },
33         {
34             "_id": ObjectId(),
35             "question": "What is mitosis?",
36             "answer": "Cell division process"
37         }
38     ]
39     await db["flashcards"].insert_many(test_flashcards)
40
41     # Insert test user
42     test_user = [
43         {
44             "_id": ObjectId(),
45             "username": "test_user",
46             "total_score": 0,
47             "history": []
48         }
49     ]
50     await db["users"].insert_one(test_user)
51
52     return str(test_user["_id"])
53
54 async def test_assign_daily_questions():
55     """Test assigning daily questions."""
56     questions = await assign_daily_questions()
57     assert len(questions) == 10
58     assert all("_id" in q for q in questions)
59     assert all("question" in q for q in questions)
60     assert all("answer" in q for q in questions)
61
62 async def test_get_daily_questions(setup_test_data):
63     """Test getting daily questions for a user."""
64     user_id = await setup_test_data
65     result = await get_daily_questions(user_id)
66     assert "questions" in result
67     assert len(result["questions"]) == 10
68
69 async def test_update_progress(setup_test_data):
70     """Test updating user progress."""
71     user_id = await setup_test_data
72     question_id = str(await db["flashcards"].find_one())["_id"]
73
74     result = await update_progress(user_id, question_id, "test answer")
75     assert "review_score" in result
76     assert "next_review" in result
77
78     # Verify user history was updated
79     user = await db["users"].find_one({"_id": ObjectId(user_id)})
80     assert len(user["history"]) == 1
81     assert user["history"][0]["question_id"] == question_id
82
83 async def test_add_score():
84     """Test adding a score to the leaderboard."""
85     # Clear leaderboard first
86     await db["leaderboard"].delete_many({})
87
88     result = await add_score("test_user", 100)
89     assert "rank" in result
90     assert "top_5" in result
91
92     # Verify score was added
93     entry = await db["leaderboard"].find_one({"username": "test_user"})
94     assert entry is not None
95     assert entry["score"] == 100
96
97     async def test_get_rank():
98         """Test getting user rank."""
99         # Clear leaderboard first
100        await db["leaderboard"].delete_many({})
101
102        result = await add_score("user1", 100)
103        await add_score("user2", 200)
104
105        result = await get_rank("user1")
106        assert "rank" in result
107        assert "top_5" in result
108        assert result["rank"] == 2 # Second place
109
110        async def test_get_top_5():
111            """Test getting top 5 scores."""
112            # Clear leaderboard first
113            await db["leaderboard"].delete_many({})
114
115            # Add multiple scores
116            await add_score("user1", 100)
117            await add_score("user2", 200)
118            await add_score("user3", 300)
119            await add_score("user4", 400)
120            await add_score("user5", 500)
121            await add_score("user6", 600)
122
123            result = await get_top_5()
124            assert "top_5" in result
125            assert len(result["top_5"]) == 5
126            assert result["top_5"][0]["score"] == 600 # Highest score first
127
128        async def test_get_user_stats(setup_test_data):
129            """Test getting user statistics."""
130            user_id = await setup_test_data
131
132            # Add some review history
133            question_id = str(await db["flashcards"].find_one())["_id"]
134            await update_progress(user_id, question_id, "test answer")
135
136            stats = await get_user_stats(user_id)
137            assert "user_id" in stats
138            assert "username" in stats
139            assert "total_score" in stats
140            assert "total_reviews" in stats
141            assert "average_score" in stats
142            assert "score_distribution" in stats
143            assert "review_schedule" in stats
```

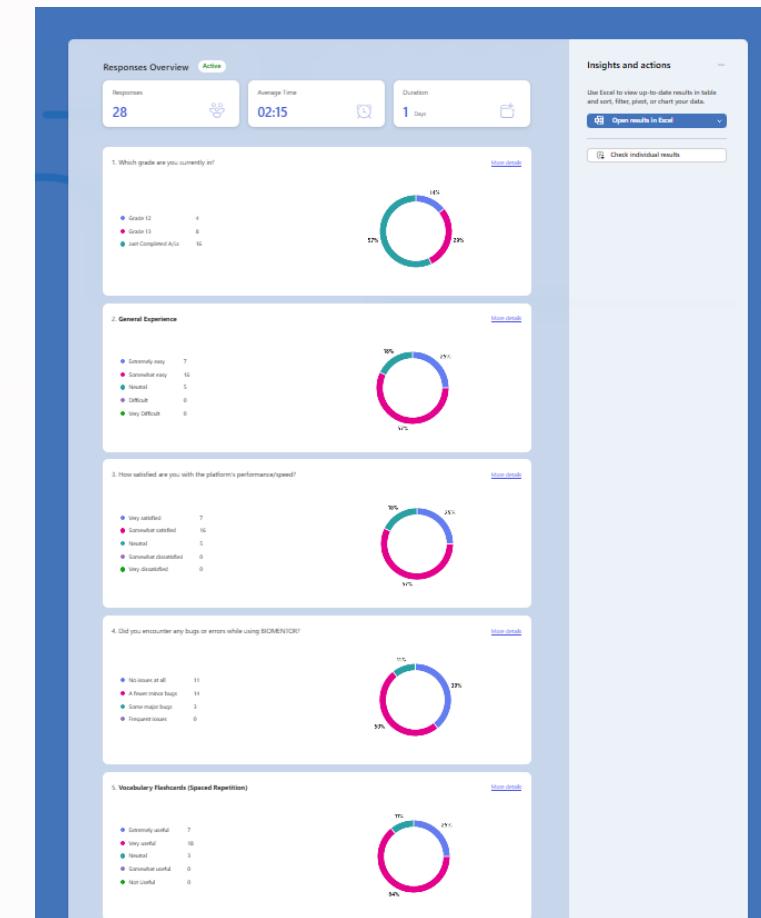
# Testing - System Testing and UAT

## System Testing

- Framework: Postman



## User Acceptance Testing



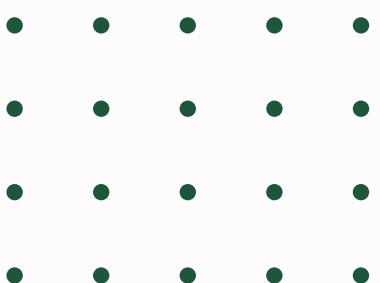
# Deployment

- Backend deployed on Azure VM (Windows - Standard), Region: Central India for low latency

The screenshot shows the Microsoft Azure portal interface for a virtual machine named 'gokul-vm'. The left sidebar contains navigation links such as Home, Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Resource visualizer, Connect, Networking (Network settings, Load balancing, Application security groups, Network manager), Settings, Availability + scale, Security, Backup + disaster recovery, Operations, Monitoring, Automation, and Help. The main content area displays the 'Overview' tab for the virtual machine. It includes a summary bar with a migration advisor (Adviser 1 of 4: Migrate workload to D-series or better virtual machine) and a 'Help me copy this VM in any region' button. Below this are sections for 'Essentials' (Resource group: gokul-vm\_group, Status: Running, Location: Central India, Subscription: Azure for Students, Subscription ID: 69468234-839b-4cb7-b168-39cdf5bbcf7e, Tags: Tags (edit), Add tags) and 'Properties' (Virtual machine: Computer name: gokul-vm, Operating system: Windows (Windows Server 2022 Datacenter Azure Edition), VM generation: V2, VM architecture: x64, Agent status: Ready, Agent version: 2.7.41491.1149, Hibernation: Disabled; Networking: Public IP address: 74.225.135.220 (Network interface gokul-vm167), Private IP address: 10.1.1.4, Virtual network/subnet: gokul-vm-vnet/default, DNS name: Configure). The top right corner shows the user's email (it21375132@my.sliit.lk) and profile picture.

# References

- [1] J. Su, J. Ye, L. Nie, Y. Cao and Y. Chen, "Optimizing Spaced Repetition Schedule by Capturing the Dynamics of Memory," in IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 10, pp. 10085-10097, 1 Oct. 2023, doi: 10.1109/TKDE.2023.3251721.
- [2] F. Schimanke, R. Mertens and B. S. Huck, "Retrieval of Relevant Data for Measuring the Impact of Spaced-Repetition Algorithms on the Learning Success in Mobile Learning Games," 2019 IEEE International Symposium on Multimedia (ISM), San Diego, CA, USA, 2019, pp. 279-2795, doi: 10.1109/ISM46123.2019.00063.
- [3] G. RANDAZZO, "Memory models for spaced repetition systems," Polimi.it, Sep. 2023, doi: <http://hdl.handle.net/10589/186407>.
- [4] F. Schimanke, "The Impact of Spaced Repetition Learning on the Learning Success in Mobile Learning Games," 2021 IEEE International Symposium on Multimedia (ISM), Naple, Italy, 2021, pp. 275-280, doi: 10.1109/ISM52913.2021.00054.



# IT21068478

Dharane.S

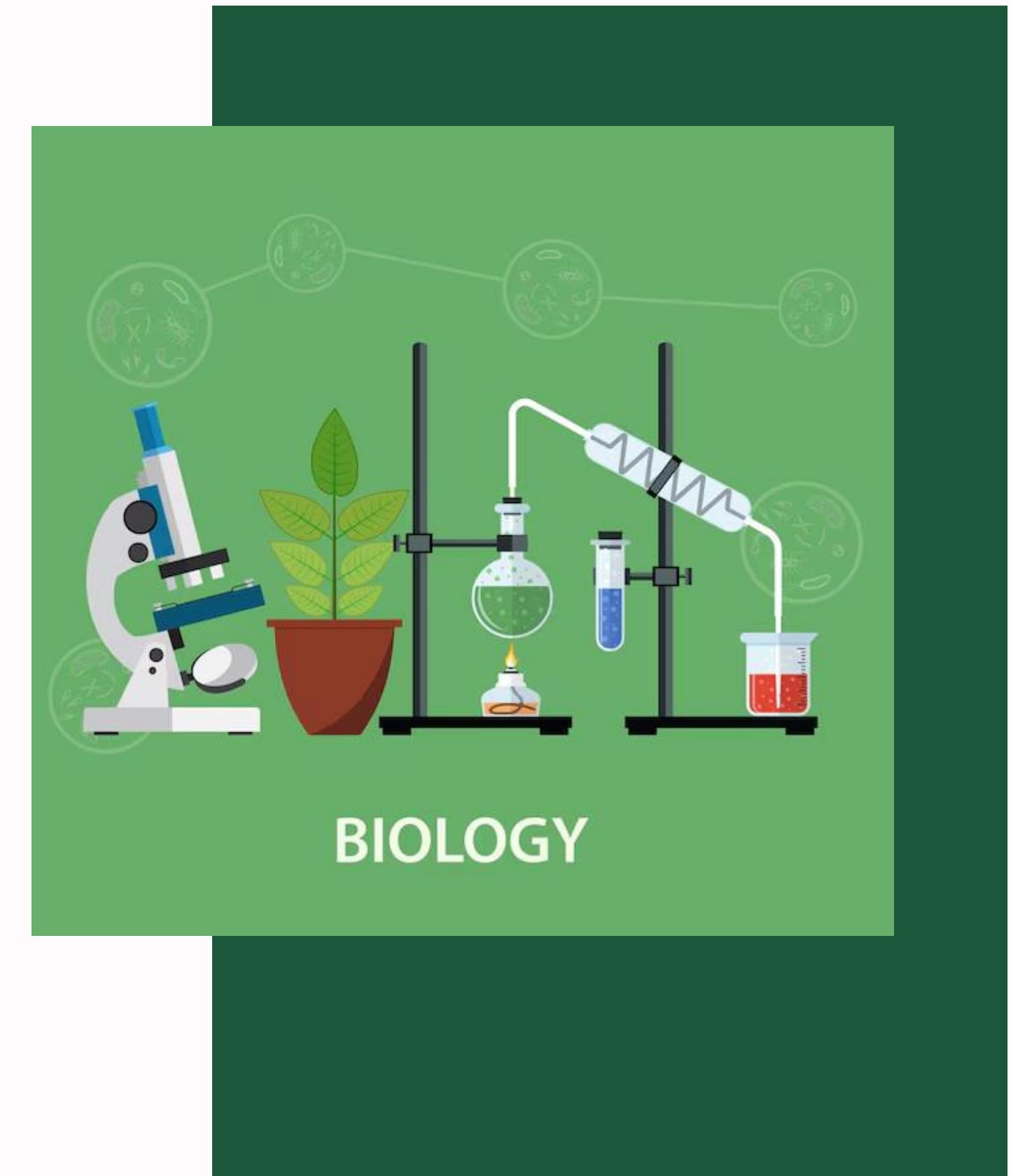
Software Engineering

LLM BASED ABSTRACTIVE TEXT  
SUMMARIZATION TOOL WITH  
VOICE OUTPUT IMPLEMENTED IN  
DIFFERENT SOFTWARE  
ARCHITECTURES

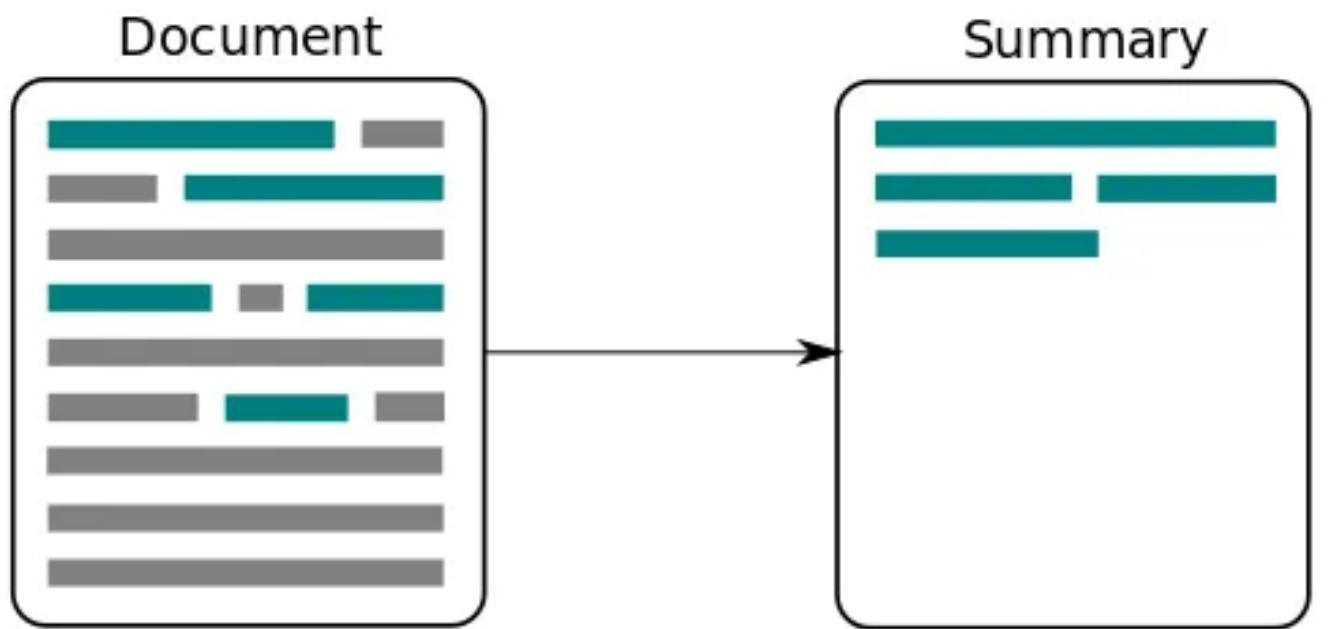


# Introduction

- 01** Background
- 02** Research Problem
- 03** Main and Sub Objectives
- 04** Methodology
- 05** Completion Of Project



# BACKGROUND



What is summarization?



What is NLP?

# RESEARCH PROBLEM

01



How can text summarization be optimized to provide concise, exam-focused summaries for A/L biology students?

02



How can the integration of voice output features with customizable word counts in an e-learning summarization tool enhance accessibility, catering to different learning preferences and needs?

# OBJECTIVES

## Objective 1

Ensure Accuracy: Use government-approved resources to maintain content accuracy and educational standards.

## Objective 3

Ensure Content Relevance: Implement filtering mechanisms to detect and eliminate irrelevant content

## Main Objective

Create a tool that extracts and summarizes key concepts from A/L biology resources based on a given topic, and generates concise summaries from uploaded text documents, specifically to aid students preparing for A/L exams.

## Objective 2

Enhance Accessibility: Integrate a voice output feature to provide audible summaries and provide basic multi-lingual support for notes.

## Objective 4

Various implementation: Implement the component in various architectures, compare, and analyze their strengths and weaknesses.

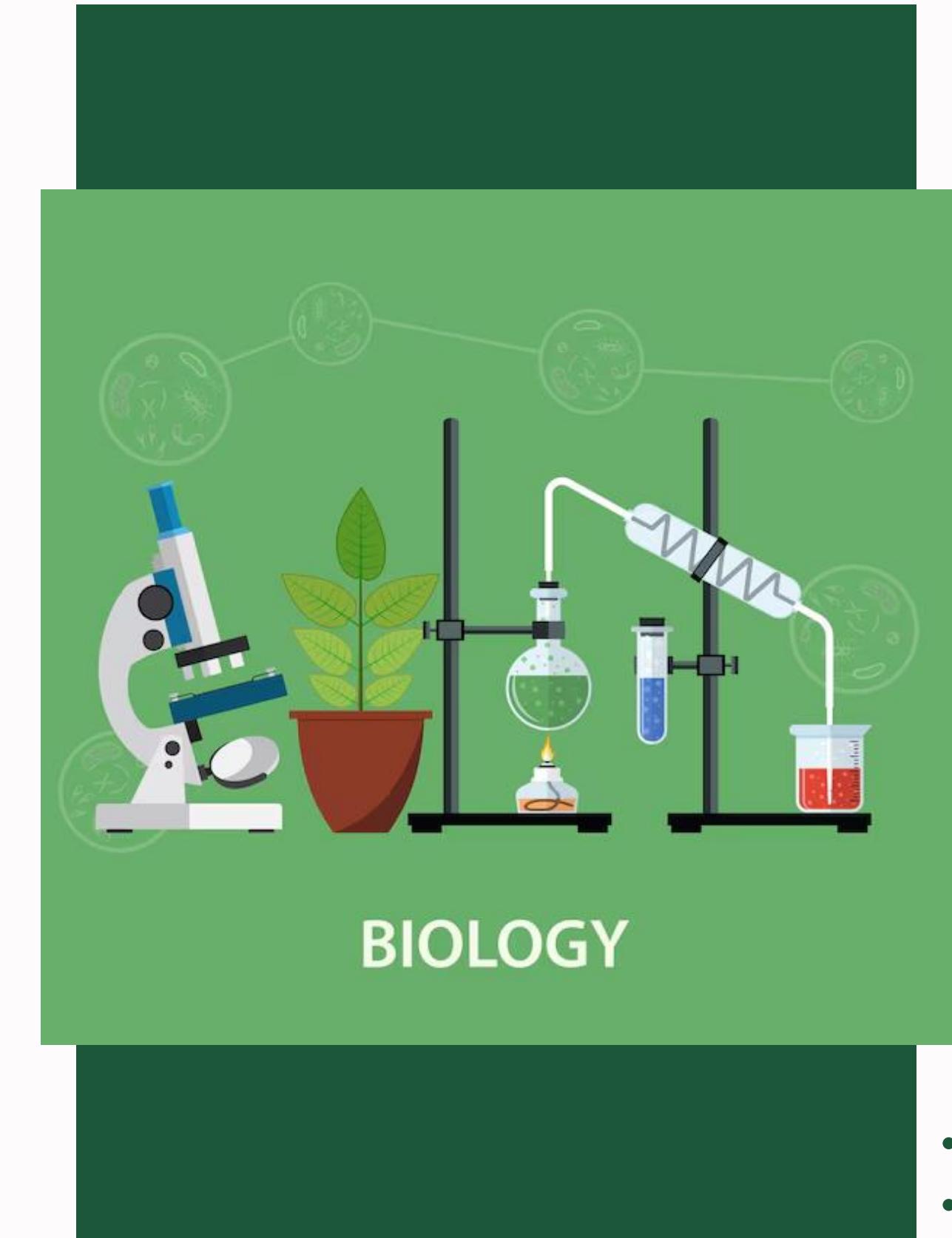
# Methodology

01 System Diagram

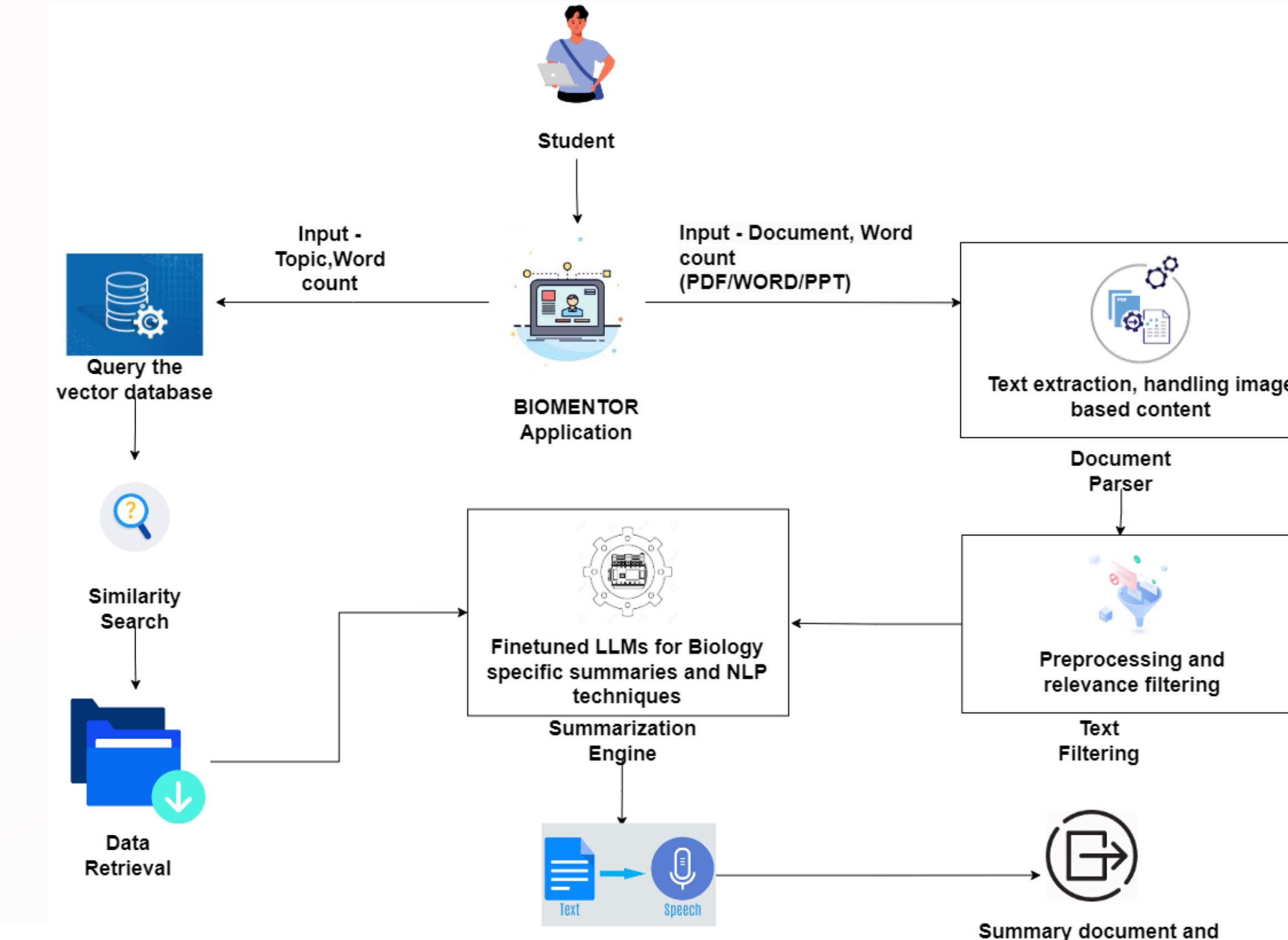
02 Tools and Technologies

03 Best Practices

04 GitHub Commits



# System Diagram



# Tools & Technologies



**Project Management**  
Jira



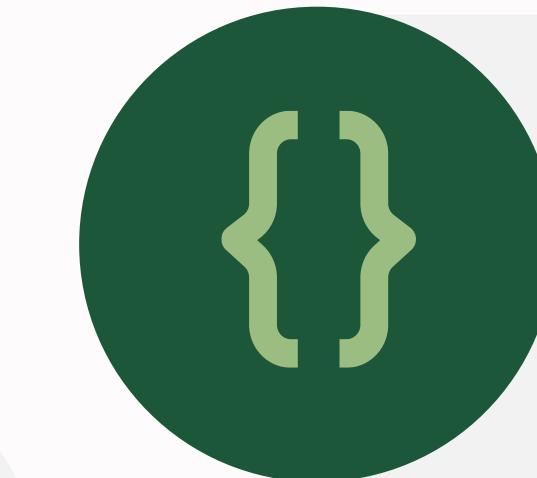
**Other tools**  
Git  
Draw.io  
Postman



**Database**  
Faiss  
Mongo DB



**Frameworks**  
Transformer model  
pandas  
Pytorch  
gtts  
Pytesseract  
Deep Translator  
FastAPI  
numpy



**Programming Languages**  
Python  
React Js

# Best Practices

## Non-Functional Requirements

- Compatibility
  - Accuracy
- Performance
  - Usability
- Availability

## Design Excellence

- Real-Time Performance
- Accuracy and Reliability
- User Experience
  - Integration

## Standards & Best Practices

- Code Structure & Organization
- Web Security
- Model Quality
- RESTful API Design

# GitHub Commits

The screenshot shows the GitHub commits page for the repository `BioMentor-Personalized-E-Learning-Platform`. The page displays a list of commits grouped by date. The first group is for March 10, 2025, containing two commits: "Phrase fixes" and "Minor fixes", both made by `DharaneSegar` 5 days ago. The second group is for March 8, 2025, containing four commits: "Changed URLs", "Resolved errors", "Merge", and a merge commit for branch 'main' from `https://github.com/Y3S1-GRP22/Personalized-E-Learning-for-Biology-Subject-for-AI-Students-in-Sri-Lanka-Leveraging-AI-ML_into IT21068478/Dharane`. The merge commit was made by `DharaneSegar` last week.

Y3S1-GRP22 / BioMentor-Personalized-E-Learning-Platform

Type / to search

Code Issues Pull requests Actions Projects Security 15 Insights Settings

All users All time

## Commits

IT21068478/Dhar...

Commits on Mar 10, 2025

**Phrase fixes**  
DharaneSegar committed 5 days ago

**Minor fixes**  
DharaneSegar committed 5 days ago

Commits on Mar 8, 2025

**Changed URLs**  
DharaneSegar committed last week

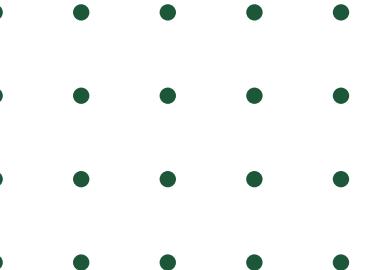
**Resolved errors**  
DharaneSegar committed last week

**Merge**  
DharaneSegar committed last week

Merge branch 'main' of [https://github.com/Y3S1-GRP22/Personalized-E-Learning-for-Biology-Subject-for-AI-Students-in-Sri-Lanka-Leveraging-AI-ML\\_into IT21068478/Dharane](https://github.com/Y3S1-GRP22/Personalized-E-Learning-for-Biology-Subject-for-AI-Students-in-Sri-Lanka-Leveraging-AI-ML_into IT21068478/Dharane)

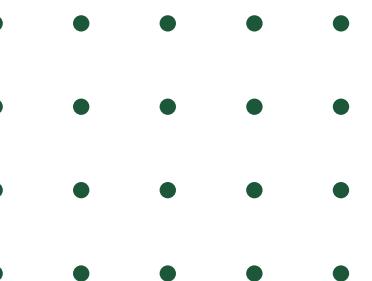


# Completion of the project





# Completion of the project



# Data Collection

## Summarization Dataset

	A	B	C
1	Long Text	Summary	Keywords
2	Issues pertaining to Biology. Understanding	Biological diversity refers to the variety of life on Earth, encompassing species, genes, and ecosystems. Biological diversity, Species, Genes, Ecosystems, Evolution, Human anatomy, Physiology, Sustainability.	
3	In accordance with different criteria we can see	Living organisms show vast diversity in size, shape, form, and habitat. They can be as small as	Diversity, Size, Shape, Form, Habitat, Metabolism, Growth, Development, Adaptation, Reproduction,
4	Physical and chemical properties of water	Water is an essential molecule for life due to its unique physical and chemical properties. It is	Water molecule, Polarity, Hydrogen bonds, Cohesion, Adhesion, High specific heat, Heat of vaporization
5	Carbohydrates	Carbohydrates are the most abundant organic compounds on Earth, composed of carbon, hydrogen, and oxygen.	Carbohydrates, Monosaccharides, Disaccharides, Polysaccharides, Glycosidic bond, Reducing sugar
6	Lipids	Lipids are a diverse group of hydrophobic molecules that are important for biological functions.	Lipids, Fats, Phospholipids, Saturated fats, Unsaturated fats, Trans fats, Glycerol, Fatty acids, Ester bonds
7	Proteins are made up of amino acids. Twenty	Proteins are composed of amino acids, with 20 different types involved in their structure. Each protein has a unique sequence of amino acids.	Proteins, Amino acids, Peptide bond, Polypeptide, Primary structure, Secondary structure, Alpha helix, Beta sheet
8	Nucleic acids are Polymers exist as	Nucleic acids are large biopolymers that exist as polynucleotides, made up of monomers.	Nucleic acids, Nucleotides, DNA, RNA, Pentose sugar, Nitrogenous base, Phosphate group, Purines, Pyrimidines
9	Advancement of cytology is mostly based on	The advancement of cytology has been largely driven by the development of microscopes. The first microscope was invented in the 16th century.	Cytology, light microscope, compound microscope, resolution power, magnification, electron microscope
10	Cell theory	Cell theory states that all organisms are composed of one or more cells, the cell is the basic unit of life.	Cell Theory, Schleiden, Schwann, Virchow, Robert Hooke, Anton Van Leeuwenhoek, Prokaryotic cells, Eukaryotic cells
11	Plasmamembrane is the outer limit of	The plasma membrane, the outer boundary of the cytoplasm, follows the fluid mosaic model.	Plasma membrane, Fluid mosaic model, Singer and Nicolson, Phospholipids, Proteins, Cholesterol, Lipoproteins
12	There are many sub-cellular components in	Cells contain numerous sub-cellular components, some of which are membrane-bound organelles.	Nucleus, Ribosomes, Endoplasmic Reticulum (ER), Golgi Apparatus, Lysosomes, Mitochondria, Chloroplasts
13	1. Cell wall	1. Cell Wall	Cell Wall, Rigid Structure, Plant Cells, Cellulose, Pectin, Hemicellulose, Lignin, Suberin, Protection, Turgor pressure
14	The sequence of events that takes place in the cell cycle	The cell cycle is the sequence of events from one cell division to the next, producing two daughter cells.	Cell Cycle, Mitosis, Interphase, Mitotic Phase, G1 Phase, S Phase, G2 Phase, Prophase, Prometaphase, Anaphase, Telophase
15	Sexually reproducing organisms undergo meiosis	Meiosis is a type of nuclear division in sexually reproducing organisms, resulting in four daughter cells.	meiosis, haploid, diploid, crossing over, homologous chromosomes, genetic variation, cancer, galls, tumors
16	Sum of all biochemical reactions of living organisms	Metabolism is the sum of all biochemical reactions in living organisms, comprising catabolic and anabolic pathways.	Metabolism, catabolism, anabolism, ATP, photophosphorylation, substrate-level phosphorylation, oxidation-reduction reactions
17	Photosynthesis	Photosynthesis is a vital metabolic process where light energy is converted into chemical energy.	Photosynthesis, chloroplasts, light-dependent reactions, Calvin cycle, ATP, NADPH, carbon fixation, oxygen evolution
18	As its name suggests, Rubisco is capable of catalyzing two reactions	Rubisco, an enzyme in photosynthesis, catalyzes two reactions: carboxylation (using CO <sub>2</sub> ) and oxygenation.	Rubisco, carboxylation, oxygenation, photorespiration, C <sub>4</sub> plants, CO <sub>2</sub> concentration, mesophyll cells
19	Cellular respiration is the process by which cells convert organic molecules into energy	Cellular respiration is the biochemical process by which cells convert organic molecules, such as glucose, into energy.	Cellular respiration, aerobic respiration, anaerobic respiration, glycolysis, citric acid cycle, electron transport chain
20	Origin of life on earth	The origin of life on Earth began about 4.6 billion years ago, with the planet forming amidst comets and meteorites.	Origin of life, Earth, atmosphere, organic molecules, protocells, photosynthesis, eukaryotes, diversification
21	Evolution can be defined as a change in the genetic composition of a population over generations	Evolution is defined as a change in the genetic composition of a population over generations, driven by natural selection.	Evolution, genetic composition, Lamarck's Theory, use and disuse, inheritance of acquired characteristics, Mendelian genetics, Darwinian evolution, punctuated equilibrium
22	Methods of artificial and natural classification	Classification arranges organisms into groups based on common characteristics, and defines species.	Classification, taxonomy, artificial classification, natural classification, phylogeny, binomial nomenclature, species concept
23	Species is a group of organisms who share similar characteristics	The concept of species refers to a group of organisms that share similar characteristics and reproductive compatibility.	Morphological species concept, ecological species concept, phylogenetic species concept
24	Key characteristics of Kingdom Protista	Kingdom Protista is a diverse and polyphyletic group of mostly unicellular organisms, including amoebae, paramecia, and amoebae.	Kingdom Protista, unicellular, colonial, multicellular, polyphyletic, photoautotrophs, heterotrophs, mixotrophs
25	Kingdom Plantae	The kingdom Plantae evolved from chlorophytes (green algae) and consists of two major groups: vascular plants and non-vascular plants.	Kingdom Plantae, chlorophytes, vascular plants, non-vascular plants, bryophytes, gametophyte, sporophyte
26	Kingdom Fungi	Kingdom Fungi comprises eukaryotic organisms with chitinous cell walls, heterotrophic.	Kingdom Fungi, eukaryotic, chitin, heterotrophs, hyphae, mycelium, reproduction, spores, Chytridiomycota
27	Kingdom Animalia	The Kingdom Animalia comprises multicellular, heterotrophic eukaryotes that digest food.	Kingdom Animalia, multicellular, heterotrophic, eukaryotes, tissues, radial symmetry, bilateral symmetry
28	The main focus of this unit is on structure, growth, and development of vascular plants	The unit focuses on the structure, growth, and development of vascular plants, which consist of roots, stems, leaves, flowers, and fruits.	Vascular plants, root system, shoot system, meristems, apical meristems, lateral meristems, intercalary meristems
29	Plant growth	Plant growth refers to the irreversible increase in dry mass and cell number, occurring through cell division and enlargement.	Plant growth, indeterminate growth, primary structure, roots, stems, secondary growth, heartwood, sapwood
30	Anatomy of typical dicot and monocot leaves	Leaves in vascular plants are the primary organs for photosynthesis and gas exchange through stomata.	Leaves, dicot, monocot, stomata, epidermis, mesophyll, photosynthesis, gas exchange, turgor pressure

# Data Collection

## Information Retrieval Dataset

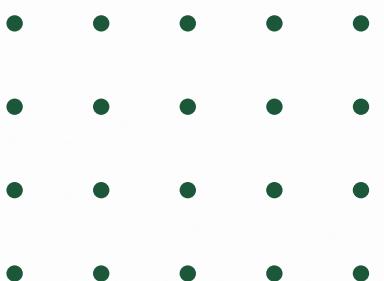
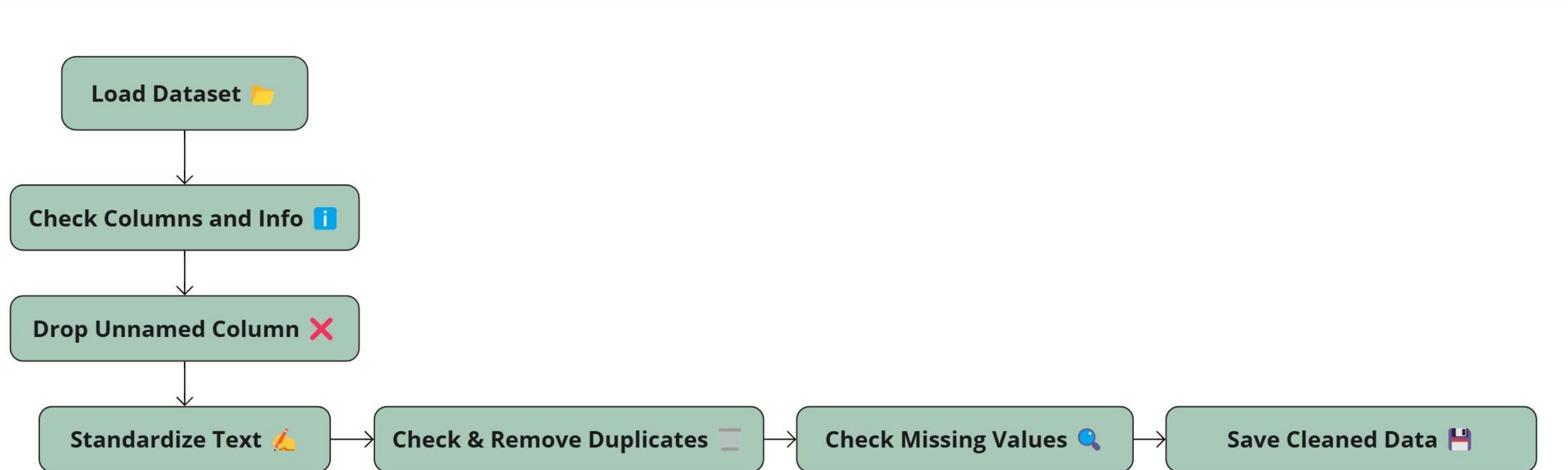
1	Document ID	Topic	Sub-topic	Text Content	Source
2	1	Introduction to Biology	Understanding biological Diversity	At present our planet is rich in diversity. Life on earth formed around 3.5 billion years ago. The first life forms were simple single-celled organisms.	Biology, Grade 12, Resource Book
3	2	Introduction to Biology	Understanding the human Body and its functions	When studying biology, especially by studying histology and anatomy of the human body, one can learn about the structure and function of different parts of the body.	Biology, Grade 12, Resource Book
4	3	Introduction to Biology	Sustainable use and Management of Natural resources	Natural resources are sources of materials and energy found naturally which are used in everyday life.	Biology, Grade 12, Resource Book
5	4	Introduction to Biology	Sustainable Food production	Sustainable food production is the production of sufficient amounts of food for the human population without causing harm to the environment.	Biology, Grade 12, Resource Book
6	5	Introduction to Biology	Understanding plant life	Plants are the primary producers in the world. All the animals depend directly or indirectly on plants for their food.	Biology, Grade 12, Resource Book
7	6	Introduction to Biology	Understanding diseases and causes	To maintain healthy human body one should have the knowledge of causes of the diseases and how to prevent them.	Biology, Grade 12, Resource Book
8	7	Introduction to Biology	Solving some legal and ethical issues	Knowledge and application of biological concepts is important in solving some legal issues, such as environmental issues.	Biology, Grade 12, Resource Book
9	8	Introduction to Biology	The nature and the organizational principles of living organisms	In accordance with different criteria we can see a diversity among living organisms. Organisms are classified into different groups based on their characteristics.	Biology, Grade 12, Resource Book
10	9	Introduction to Biology	Hierarchical levels of organization	The cell is the basic structural and functional unit of life. Some organisms are unicellular while others are multicellular.	Biology, Grade 12, Resource Book
11	10	Chemical and cellular basis of life	Physical and chemical properties of water	Physical and chemical properties of water important for life	Biology, Grade 12, Resource Book
12	11	Chemical and cellular basis of life	Carbohydrates	Most abundant group of organic compound on earth is carbohydrates. Major elemental components are carbon, hydrogen, and oxygen.	Biology, Grade 12, Resource Book
13	12	Chemical and cellular basis of life	Lipids	Lipids	Biology, Grade 12, Resource Book
14	13	Chemical and cellular basis of life	Proteins	Proteins	Biology, Grade 12, Resource Book
15	14	Chemical and cellular basis of life	Nucleic acids	Nucleic acids are Polymers exist as polynucleotides made up of monomers called nucleotides.	Biology, Grade 12, Resource Book
16	15	Chemical and cellular basis of life	Contribution of microscope to the study of biology	Advancement of cytology is mostly based on the microscopy. The discovery and early study of microorganisms led to the development of modern biology.	Biology, Grade 12, Resource Book
17	16	Chemical and cellular basis of life	Historical background of the cell	Cell theory	Biology, Grade 12, Resource Book
18	17	Chemical and cellular basis of life	Structures and functions of the cell	Plasmamembrane is the outer limit of cytoplasm. All cellular membranes resemble the ultrafiltration membranes.	Biology, Grade 12, Resource Book
19	18	Chemical and cellular basis of life	Subcellular components	There are many sub-cellular components in the cell. Some of them are organelles, which are specialized structures performing specific functions.	Biology, Grade 12, Resource Book
20	19	Chemical and cellular basis of life	Extracellular components	1. Cell wall	Biology, Grade 12, Resource Book
21	20	Chemical and cellular basis of life	The cell cycle and the process of cell division	The sequence of events that takes place in the cell from the end of one cell division to the next cell division.	Biology, Grade 12, Resource Book
22	21	Chemical and cellular basis of life	Meiosis	Sexually reproducing organisms undergo different type of cell division called meiosis.	Biology, Grade 12, Resource Book
23	22	Chemical and cellular basis of life	The energy relationships in metabolism	Sum of all biochemical reactions of living being is known as the metabolism and it consists of catabolic and anabolic reactions.	Biology, Grade 12, Resource Book
24	23	Chemical and cellular basis of life	Photosynthesis as an energy fixing process	Photosynthesis	Biology, Grade 12, Resource Book
25	24	Chemical and cellular basis of life	Photorespiration	As its name suggests, Rubisco is capable of catalyzing two distinct reactions, acting as both an oxygenase and a carboxylase.	Biology, Grade 12, Resource Book
26	25	Chemical and cellular basis of life	Cellular respiration as a process of energy release	Cellular respiration is the process by which chemical energy in organic molecules such as glucose is released and used for various cellular activities.	Biology, Grade 12, Resource Book
27	26	Evolution and Diversity of Organisms	The theories of origin of life and evolution	Origin of life on earth	Biology, Grade 12, Resource Book

## Why Flan-T5-Base?

- Instruction-Following Capability
- Versatility in Fine-tuning
- Efficiency and Scalability
- Customizability
- Pre-trained Knowledge
- Resource Compatibility



# Data pre-processing



# Data pre-processing

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
# Removing unwanted characters, space, special characters
import re

# Function to standardize text: convert to lowercase, remove irrelevant symbols
def standardize_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove irrelevant symbols (e.g., unusual bullet points, etc.)
    text = re.sub(r'[\u2022]', '', text)
    # Remove any remaining special characters (e.g., non-alphanumeric symbols)
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)
    # Remove extra whitespace
    text = re.sub(r'\s+', ' ', text).strip()
    return text

# Apply the function to standardize the 'Long Text' and 'Summary' columns
data['Long Text'] = data['Long Text'].apply(standardize_text)
data['Summary'] = data['Summary'].apply(standardize_text)

# Display a sample to verify the standardization process
data[['Long Text', 'Summary']].head()
```

	Long Text	Summary
0	issues pertaining to biology understanding bio...	biological diversity refers to the variety of ...
1	in accordance with different criteria we can s...	living organisms show vast diversity in size s...
2	physical and chemical properties of water impo...	water is an essential molecule for life due to...
3	carbohydrates most abundant group of organic c...	carbohydrates are the most abundant organic co...
4	lipids diverse group of hydrophobic molecules ...	lipids are a diverse group of hydrophobic mole...

```
# Checking and removing duplicates based on specific columns 'Long Text' and 'Summary' only, which are the main content columns
data_cleaned = data.drop_duplicates(subset=['Long Text', 'Summary'])

# Display the number of rows before and after removing duplicates
original_count = data.shape[0]
cleaned_count = data_cleaned.shape[0]

original_count, cleaned_count
```

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
# Checking for missing values in the essential columns 'Long Text' and 'Summary'
missing_long_text = data['Long Text'].isnull().sum()
missing_summary = data['Summary'].isnull().sum()

# Displaying the count of missing values in each essential column
missing_long_text, missing_summary
```

[19] ... (0, 0)

```
# Saving the cleaned dataset to a new CSV file named 'bio_summary.csv'
output_path = 'D:/Downloads/RP/Summarization/Summary_Description/bio_summary.csv'
data_cleaned.to_csv(output_path, index=False)

output_path
```

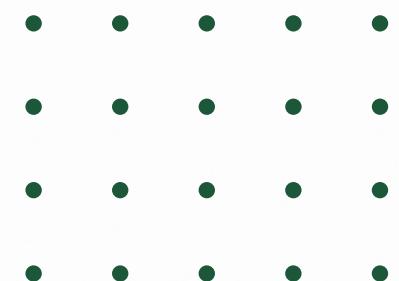
[19] ... 'C:/Users/dinon/Desktop/Summary/bio\_summary.csv'

```
# Selecting only 'Long Text' and 'Summary' columns and renaming the headers to lowercase
data_final = data_cleaned[['Long Text', 'Summary']].rename(columns={'Long Text': 'long text', 'Summary': 'summary'})

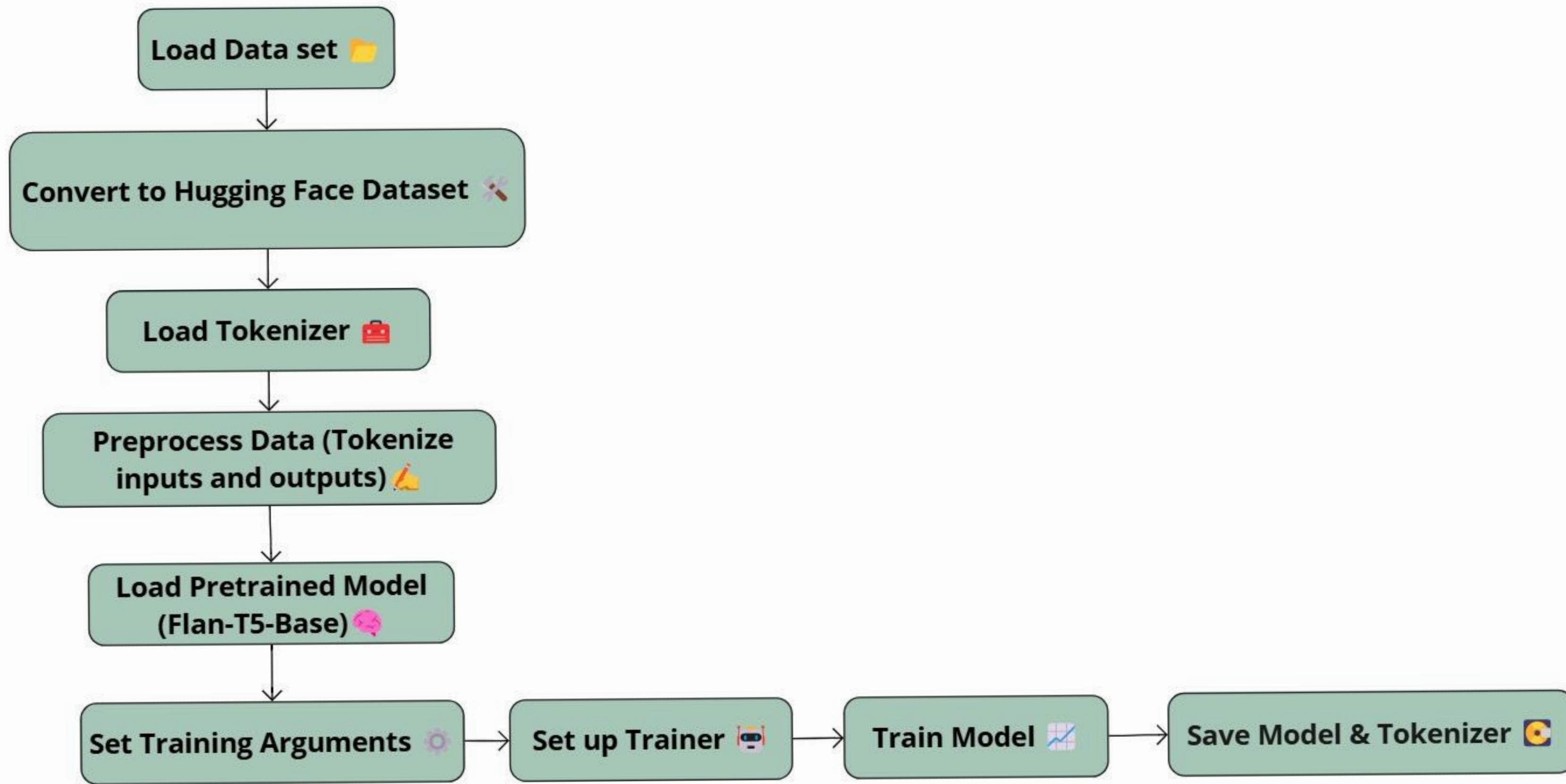
# Saving this final cleaned dataset to a new CSV file
output_path_final = 'D:/Downloads/RP/Summarization/Summary_Description/bio_summary_final.csv'
data_final.to_csv(output_path_final, index=False)

output_path_final
```

[19] ... 'C:/Users/dinon/Desktop/Summary/bio\_summary\_final.csv'



# Fine-tuning the LLM



# Fine-tuning the LLM

```
# Load the dataset
data_path = '/content/drive/MyDrive/bld_summary_final.csv' # Update with your dataset path
df = pd.read_csv(data_path)

# Convert the DataFrame to a Hugging Face Dataset format
dataset = Dataset.from_pandas(df)

# Load the tokenizer
tokenizer = AutoTokenizer.from_pretrained("google/flan-t5-base")

# Tokenization function with padding
def preprocess_data(examples):
    model_inputs = tokenizer([doc['longtext'] for doc in examples], max_length=512, truncation=True) # Adding padding
    # Setup the tokenizer for targets
    with tokenizer.as_target_tokenizer():
        labels = tokenizer(examples['summary'], max_length=150, padding='max_length', truncation=True) # Adding padding
    model_inputs["labels"] = labels["input_ids"]
    return model_inputs

print(df.columns)
```

```
# Apply the preprocessing to the dataset
tokenized_dataset = dataset.map(preprocess_data, batched=True)

# Load the model
model = AutoModelForSeq2SeqLM.from_pretrained("google/flan-t5-base")

# Set up training arguments
training_args = TrainingArguments(
    output_dir='./flan_t5_finetuned',
    evaluation_strategy='epoch',
    learning_rate=2e-05,
    per_device_train_batch_size=4, # Set to a low batch size to fit in Colab's memory
    per_device_eval_batch_size=4,
    num_train_epochs=3,
    weight_decay=0.01,
    save_total_limit=2,
    push_to_hub=False
)
```

```
trainer.train()

Epoch Training Loss Validation Loss
1 No log 2.658453
2 No log 2.599380
3 No log 2.540879

TrainOutput(global_step=375, training_loss=2.9246923828125, metrics={'train_runtime': 436.3881, 'train_samples_per_second': 3.43, 'train_steps_per_second': 0.859, 'total_flos': 1825081756014056.0, 'train_loss': 2.9246923828125, 'epoch': 3.0})
```

```
model.save_pretrained('/content/drive/MyDrive/flan_t5_finetuned_model')
tokenizer.save_pretrained('/content/drive/MyDrive/flan_t5_finetuned_model')

('content/drive/MyDrive/flan_t5_finetuned/model/tokenizer_config.json',
 'content/drive/MyDrive/flan_t5_finetuned/model/special_tokens_map.json',
 'content/drive/MyDrive/flan_t5_finetuned/model/slice_model',
 'content/drive/MyDrive/flan_t5_finetuned/model/added_tokens.json',
 'content/drive/MyDrive/flan_t5_finetuned/model/tokenizer.json')

# Define the path in Google Drive (or wherever you want to save it)
model_save_path = '/content/drive/MyDrive/flan_t5_finetuned_model_1h' # update with your desired path

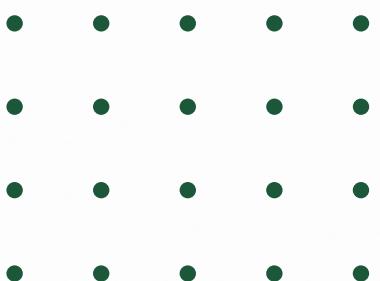
# Save the model
trainer.model.save_pretrained(model_save_path)

# Save the tokenizer
tokenizer.save_pretrained(model_save_path)
```

# Evaluate the Fine-tuned model

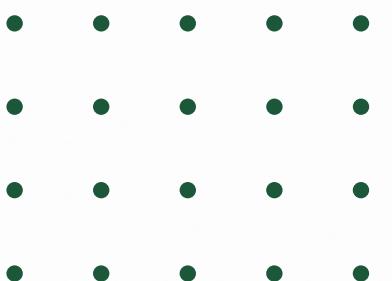
## Evaluation Metric: ROUGE(Recall-Oriented Understudy for Gisting Evaluation)

- Measures word overlap between the generated summary and the reference summary.
- Types of ROUGE Scores
  - ROUGE-1: Unigram (single word) overlap.
  - ROUGE-2: Bigram (two-word) overlap.
  - ROUGE-L: Longest Common Subsequence (LCS) match.



# Evaluate the Fine-tuned model

```
{'rouge1': Score(precision=0.7414965986394558,  
recall=0.5989010989010989,  
fmeasure=0.6626139817629179),  
'rouge2': Score(precision=0.4041095890410959,  
recall=0.3259668508287293,  
fmeasure=0.3608562691131499),  
'rougeL': Score(precision=0.54421768707483,  
recall=0.43956043956043955,  
fmeasure=0.48632218844984804)}
```



# RAG Implementation



# RAG Implementation

```
import pandas as pd

# Load the notes dataset
notes_df = pd.read_csv('biology_information_retrieval_sample.csv', encoding='ISO-8859-1') # Update with the correct file path
notes_content = notes_df['Text Content'].tolist()
notes_topics = notes_df['Topic'].tolist()
notes_subtopics = notes_df['Sub-topic'].tolist()

# Load the notes dataset
notes_df = pd.read_csv('biology_information_retrieval_sample.csv', encoding='ISO-8859-1') # Update with the correct file path
notes_content = notes_df['Text Content'].tolist()
notes_topics = notes_df['Topic'].tolist()
notes_subtopics = notes_df['Sub-topic'].tolist()

# Load the summarization dataset
summary_df = pd.read_csv('bio_summary_key.csv', encoding='ISO-8859-1')
long_texts = summary_df['long_text'].tolist()
summarized = summary_df['summary'].tolist()
keywords = summary_df['keywords'].tolist()

pip install sentence_transformers False CPU
```

Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 23.2.1 -> 24.3.1  
[notice] To update, run: python -m pip install --upgrade pip  
Requirement already satisfied: sentence-transformers<0.0.4,>=0.0.3 in c:\users\dharane\appdata\local\programs\python\3.11\lib\site-packages (3.3.1)  
Requirement already satisfied: fastapi<0.80.0,>=0.79.0 in c:\users\dharane\appdata\local\programs\python\3.11\lib\site-packages (1.9.0.post1)  
Requirement already satisfied: torch<1.11.0 in c:\users\dharane\appdata\local\programs\python\3.11\lib\site-packages (from sentence-transformers) (4.42.4)  
Requirement already satisfied: transformers<0.0.3,>=4.11.0 in c:\users\dharane\appdata\local\programs\python\3.11\lib\site-packages (from sentence-transformers) (4.66.4)  
Requirement already satisfied: torch<1.11.0 in c:\users\dharane\appdata\local\programs\python\3.11\lib\site-packages (from sentence-transformers) (2.3.1)  
Requirement already satisfied: torch<1.11.0 in c:\users\dharane\appdata\local\programs\python\3.11\lib\site-packages (from sentence-transformers) (1.5.2)  
Requirement already satisfied: scikit-learn in c:\users\dharane\appdata\local\programs\python\3.11\lib\site-packages (from sentence-transformers) (1.14.1)  
Requirement already satisfied: scipy<1.10.0,>=1.0.0 in c:\users\dharane\appdata\local\programs\python\3.11\lib\site-packages (from sentence-transformers) (1.1.0)  
Requirement already satisfied: huggingface-hub<0.20.0 in c:\users\dharane\appdata\local\programs\python\3.11\lib\site-packages (from sentence-transformers) (0.24.0)  
Requirement already satisfied: Pillow in c:\users\dharane\appdata\local\programs\python\3.11\lib\site-packages (from sentence-transformers) (10.4.0)  
Requirement already satisfied: numpy<1.25.0,>=1.24.0 in c:\users\dharane\appdata\local\programs\python\3.11\lib\site-packages (from sentence-transformers) (1.26.4)  
Requirement already satisfied: torch<1.11.0 in c:\users\dharane\appdata\local\programs\python\3.11\lib\site-packages (from sentence-transformers) (1.11.0)  
Requirement already satisfied: filelock in c:\users\dharane\appdata\local\programs\python\3.11\lib\site-packages (from sentence-transformers) (3.15.4)  
Requirement already satisfied: fsspec<2021.5.0 in c:\users\dharane\appdata\local\programs\python\3.11\lib\site-packages (from sentence-transformers) (2024.6.1)  
Requirement already satisfied: pyyaml<5.1 in c:\users\dharane\appdata\local\programs\python\3.11\lib\site-packages (from sentence-transformers) (6.0.1)  
Requirement already satisfied: requests in c:\users\dharane\appdata\local\programs\python\3.11\lib\site-packages (from sentence-transformers) (2.32.1)

```
length_penalty=1.2,
num_beams=4,
repetition_penalty=2.0,
early_stopping=True
)
summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)
summary = postprocess_summary(summary)

# Truncate summary to fit exact word count range
return truncate_to_word_count(summary, max_words)

def truncate_to_word_count(text, max_words):
    """Ensure the summary fits within the desired word count range."""
    words = text.split()
    if len(words) > max_words:
        return " ".join(words[:max_words]) + ('.' if text[-1] not in ('.', '!')) else ''
    return text

import re

def postprocess_summary(summary):
    # Ensure the first letter is capitalized
    summary = summary.strip()
    if summary and summary[0].islower():
        summary = summary[0].capitalize() + summary[1:]

    # Add a period if the summary doesn't end with punctuation
    if summary and summary[-1] not in ('.', '!'):
        summary += "."

    # Remove unnecessary whitespace
    summary = re.sub(r"\s+", " ", summary)

    return summary
```

```
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

# Path to your fine-tuned model
model_path = 'D:/Downloads/AP/Summarization/fian_t5_finetuned_model-20241119T102614Z-001/fian_t5_finetuned_model' # Update with the correct path

# Load the tokenizer and model
tokenizer = AutoTokenizer.from_pretrained(model_path)
model = AutoModelForSeq2SeqLM.from_pretrained(model_path)
```

```
# Helper function to chunk text
def chunk_text(text, max_tokens=500):
    words = text.split()
    chunks = [ ' '.join(words[i:i+max_tokens]) for i in range(0, len(words), max_tokens) ]
    return chunks

# Check if input exceeds the max token limit
max_input_words = 300 # ~512 tokens
if len(long_text.split()) > max_input_words:
    # Chunk the input into smaller parts
    chunks = chunk_text(long_text, max_tokens=max_input_words)

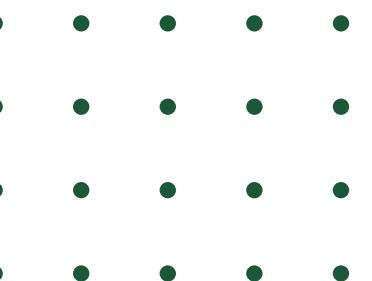
    # Generate a summary for each chunk and combine the results
    summaries = [generate_summary_for_long_text(chunk, min_words, max_words) for chunk in chunks]
    combined_summary = " ".join(summaries)

    # Ensure the combined summary fits within the final word range
    return truncate_to_word_count(combined_summary, max_words)

# For shorter inputs, generate the summary directly
prompt = (
    f"Generate a concise, well-structured, and grammatically correct summary for the following content:\\n\\nSummary:\\n\\n{long_text}"
```



# Completion of the project



# Key Functionalities

- File Processing – Extracts text and tables from .docx, .pptx, .pdf, and .txt using python-docx, python-pptx, PyMuPDF, Tabula, pdf2image, and pytesseract.
- Content Retrieval (RAG Model) – Embeds & retrieves relevant content via Sentence Transformers and FAISS.
- Inappropriate Content Detection – Uses better-profanity for explicit words, TextBlob for sentiment analysis, and wordnet for valid word verification..
- Text Summarization – Uses Flan-T5 with autocorrect & language\_tool\_python for clean, structured summaries.
- Text-to-Speech – Converts summaries into .mp3 using gTTS.
- Basic Multilingual Support – Translates notes into Tamil and Sinhala using DeepTranslator (GoogleTranslator).
- File Management – Serves summaries & structured notes via FastAPI endpoints.

# Architecture Analysis

```
C:\Windows\System32\cmd.exe + 

jinja HTTP/1.1" 404 0
monolith_backend-1 | 2025-03-03 21:07:51,316 - DEBUG - https://huggingface.co:443 "GET /api/models/sentence-transformers/all-MiniLM-L6-v2/revision/main HTTP/1.1" 200 6758
monolith_backend-1 | 2025-03-03 21:07:51,322 - DEBUG - Starting new HTTPS connection (1): huggingface.co:443
monolith_backend-1 | 2025-03-03 21:07:51,683 - DEBUG - https://huggingface.co:443 "HEAD /sentence-transformers/all-MiniLM-L6-v2/resolve/fa97f6e7cba59073df9e6b13e2715cf7475ac9/1_Pooling/config.json HTTP/1.1" 200 0
monolith_backend-1 | 2025-03-03 21:07:51,689 - DEBUG - Attempting to acquire lock 139630686980656 on /root/.cache/huggingface/hub/.locks/models--sentence-transformers--all-MiniLM-L6-v2/d1514c3162bbe87b343f565fad62e6c06f04f03.lock
monolith_backend-1 | 2025-03-03 21:07:51,691 - DEBUG - Lock 139630686980656 acquired on /root/.cache/huggingface/hub/.locks/models--sentence-transformers--all-MiniLM-L6-v2/d1514c3162bbe87b343f565fad62e6c06f04f03.lock
monolith_backend-1 | 2025-03-03 21:07:51,975 - DEBUG - https://huggingface.co:443 "GET /sentence-transformers/all-MiniLM-L6-v2/resolve/fa97f6e7cba59073dff9e6b13e2715cf7475ac9/1_Pooling/config.json HTTP/1.1" 200 190
monolith_backend-1 | 2025-03-03 21:07:51,984 - DEBUG - Attempting to release lock 139630686980656 on /root/.cache/huggingface/hub/.locks/models--sentence-transformers--all-MiniLM-L6-v2/d1514c3162bbe87b343f565fad62e6c06f04f03.lock
monolith_backend-1 | 2025-03-03 21:07:51,985 - DEBUG - Lock 139630686980656 released on /root/.cache/huggingface/hub/.locks/models--sentence-transformers--all-MiniLM-L6-v2/d1514c3162bbe87b343f565fad62e6c06f04f03.lock
monolith_backend-1 | 2025-03-03 21:07:52,266 - DEBUG - https://huggingface.co:443 "GET /api/models/sentence-transformers/all-MiniLM-L6-v2 HTTP/1.1" 200 6758
monolith_backend-1 | 2025-03-03 21:07:52,529 - DEBUG - Starting new HTTPS connection (1): www.languagetool.org:443
monolith_backend-1 | 2025-03-03 21:07:53,194 - DEBUG - https://www.languagetool.org:443 "GET /download/LanguageTool-6.5.zip HTTP/1.1" 301 169
monolith_backend-1 | 2025-03-03 21:07:53,195 - DEBUG - Starting new HTTPS connection (1): languagetool.org:443
monolith_backend-1 | 2025-03-03 21:07:53,279 - DEBUG - https://languagetool.org:443 "GET /download/LanguageTool-6.5.zip HTTP/1.1" 200 248160977
Downloading LanguageTool 6.5: 100%|██████████| 248M/248M [00:40<00:00, 6.17MB/s]
monolith_backend-1 | 2025-03-03 21:08:33,513 - INFO - Unzipping /tmp/tmpdwiwatfno.zip to /root/.cache/language_tool_python.
monolith_backend-1 | 2025-03-03 21:08:40,366 - INFO - Downloaded https://www.languagetool.org/download/LanguageTool-6.5.zip to /root/.cache/language_tool_python.
monolith_backend-1 | 2025-03-03 21:08:42,309 - DEBUG - Starting new HTTP connection (1): 127.0.0.1:8081
monolith_backend-1 | 2025-03-03 21:08:42,399 - DEBUG - http://127.0.0.1:8081 "GET /v2/languages HTTP/1.1" 200 3307
monolith_backend-1 | 2025-03-03 21:08:42,402 - INFO - RAG Model components loaded successfully.
monolith_backend-1 | 2025-03-03 21:08:42,545 - INFO - Datasets loaded successfully.
Batches: 100%|██████████| 21/21 [00:35<00:00, 1.69s/it]
monolith_backend-1 | 2025-03-03 21:09:18,423 - INFO - FAISS index created and populated successfully.
monolith_backend-1 | 2025-03-03 21:09:18,423 - INFO - FAISS index initialized successfully.
monolith_backend-1 | 2025-03-03 21:09:18,424 - INFO - RAG Model initialized successfully.
monolith_backend-1 | 2025-03-03 21:09:18,453 - INFO - Started server process [1]
monolith_backend-1 | 2025-03-03 21:09:18,453 - INFO - Waiting for application startup.
monolith_backend-1 | 2025-03-03 21:09:18,455 - INFO - Application startup complete.
monolith_backend-1 | 2025-03-03 21:09:18,456 - INFO - Uvicorn running on http://0.0.0.0:8070 (Press CTRL+C to quit)

View in Docker Desktop View Config Enable Watch
```

```
C:\Windows\System32\cmd.exe + x - o

✓Service voice_service                                Built          62.7s
✓Service file_service                               Built          22.2s
✓Service text_extraction_service                   Built          2108.2s
✓Service summarization_service                    Built          2161.3s
✓Service api_gateway                                Built          18.3s
✓Network microservices-architecture_default        Created         0.1s
✓Container microservices-architecture-text_extraction_service-1 Created         0.3s
✓Container microservices-architecture-summarization_service-1 Created         0.3s
✓Container microservices-architecture-file_service-1 Created         0.3s
✓Container microservices-architecture-voice_service-1 Created         0.3s
✓Container microservices-architecture-api_gateway-1 Created         0.1s
Attaching to api_gateway-1, file_service-1, summarization_service-1, text_extraction_service-1, voice_service-1
file_service-1 | INFO: Started server process [1]
file_service-1 | INFO: Waiting for application startup.
file_service-1 | INFO: Application startup complete.
file_service-1 | INFO: Unicorn running on http://0.0.0.0:8004 (Press CTRL+C to quit)
api_gateway-1  | 2025-03-03 19:51:41,859 - INFO - Started server process [1]
api_gateway-1  | 2025-03-03 19:51:41,860 - INFO - Waiting for application startup.
api_gateway-1  | 2025-03-03 19:51:41,860 - INFO - Application startup complete.
api_gateway-1  | 2025-03-03 19:51:41,861 - INFO - Unicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
voice_service-1 | INFO: Started server process [1]
voice_service-1 | INFO: Waiting for application startup.
voice_service-1 | INFO: Application startup complete.
voice_service-1 | INFO: Unicorn running on http://0.0.0.0:8003 (Press CTRL+C to quit)
text_extraction_service-1 | INFO: Started server process [1]
text_extraction_service-1 | INFO: Waiting for application startup.
text_extraction_service-1 | INFO: Application startup complete.
text_extraction_service-1 | INFO: Unicorn running on http://0.0.0.0:8001 (Press CTRL+C to quit)
summarization_service-1 | [nltk_data] Downloading package words to /root/nltk_data...
summarization_service-1 | [nltk_data] Unzipping corpora/words.zip.
summarization_service-1 | [nltk_data] Downloading package wordnet to /root/nltk_data...
Downloading LanguageTool 6.5: 100%[██████] 248M/248M [00:25<00:00, 9.66MB/s]
summarization_service-1 | Unzipping /tmp/tmpg8xxwe4h.zip to /root/.cache/language_tool_python.
summarization_service-1 | Downloaded https://www.languagetool.org/download/LanguageTool-6.5.zip to /root/.cache/language_tool_python.
summarization_service-1 | INFO: Started server process [1]
summarization_service-1 | INFO: Waiting for application startup.
summarization_service-1 | INFO: Application startup complete.
summarization_service-1 | INFO: Unicorn running on http://0.0.0.0:8002 (Press CTRL+C to quit)

View in Docker Desktop  o View Config  w Enable Watch
```

# Metrics

# Monolithic Architecture

## Response Time

85% faster, no API overhead (4.5 min)

The screenshot shows a Postman interface with a POST request to `http://localhost:8070/process-query/`. The body is set to `x-www-form-urlencoded` with parameters `query=Heart` and `word_count=250`. The response is a 200 OK status with a total time of 4m 19.20s and a size of 1.55 KB. The JSON response includes fields like `status`, `message`, `summary`, `summary_file`, and `voice_file`.

# Microservices Architecture

Slower, inter-service delays.(26.5 min)

The screenshot shows a Postman interface with a POST request to `http://localhost:8080/process-query/`. The body is set to `x-www-form-urlencoded` with parameters `query=Heart` and `word_count=250`. The response is a 200 OK status with a total time of 26m 38.16s and a size of 1.55 KB. The JSON response is identical to the monolithic one.

## Deployment Speed

47% faster (37.8 min)

```
PS C:\Users\dharane> cd "D:\Downloads\RP\BioMentor-Personalized-E-Learning-Platform\Back-End\Summarization\Architecture Analysis\Monolithic-Architecture"
PS D:\Downloads\RP\BioMentor-Personalized-E-Learning-Platform\Back-End\Summarization\Architecture Analysis\Monolithic-Architecture> Measure-Command { docker
-compose up --build -d }
[+] Running 4/5
[*] Running 1/1 service
  Built333.2s
  ✓Service file_service
  ✓Service text_extraction_service
  ✓Service summarization_service
  ✓Service voice_service
  ✓Service api_gateway
  ✓Service microservices-architecture_default
  ✓Container microservices-architecture-voice_service-1
  ✓Container microservices-architecture-file_service-1
  ✓Container microservices-architecture-summarization_service-1
  ✓Container microservices-architecture-text_extraction_service-1
  ✓Container microservices-architecture-api_gateway-1
Failed to solve: process "/bin/sh -c pip install --no-cache-dir -r requirements.txt && rm -rf /root/.cache/pip" did not complete successfully: exit code: 2
Days : 0
Hours : 0
Minutes : 29
Seconds : 37
Milliseconds : 291
Ticks : 17772911348
TotalDays : 0.0205704992453704
TotalHours : 0.493691981888889
TotalMinutes : 29.6215189133333
TotalSeconds : 1777.2911348
TotalMilliseconds : 1777291.1348
```

Slower (71.5 min)

```
PS C:\Users\dharane> cd "D:\Downloads\RP\BioMentor-Personalized-E-Learning-Platform\Back-End\Summarization\Architecture Analysis\Microservices-Architecture"
PS D:\Downloads\RP\BioMentor-Personalized-E-Learning-Platform\Back-End\Summarization\Architecture Analysis\Microservices-Architecture> Measure-Command { docker
compose up --build -d }
[+] Running 4/5
[*] Running 1/1 service
  Built333.2s
  ✓Service file_service
  ✓Service text_extraction_service
  ✓Service summarization_service
  ✓Service voice_service
  ✓Service api_gateway
  ✓Service microservices-architecture_default
  ✓Container microservices-architecture-voice_service-1
  ✓Container microservices-architecture-file_service-1
  ✓Container microservices-architecture-summarization_service-1
  ✓Container microservices-architecture-text_extraction_service-1
  ✓Container microservices-architecture-api_gateway-1
  DockerDesktopLinuxEngine: file has already been closed
  Created0.1s
  Started1.9s
  Started1.9s
  Started1.9s
  Started1.9s
  Started2.0s
Days : 0
Hours : 1
Minutes : 11
Seconds : 30
Milliseconds : 694
Ticks : 42906947139
TotalDays : 0.8096688184479167
TotalHours : 1.19185964275
TotalMinutes : 71.51578565
TotalSeconds : 4290.6947139
TotalMilliseconds : 4290694.7139
```

# Metrics

# Monolithic Architecture

## CPU and Memory Usage

Lower (~34% CPU, 28-36% RAM)

```
C:\Windows\System32\cmd.exe > Windows PowerShell > Windows PowerShell > + >
CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS
c4acecf7eb0d monolithic-architecture-monolith_backend-1 36.05% 1.376GiB / 3.754GiB 36.66% 35.3kB / 6.76kB 0B / 0B 55
```

# Microservices Architecture

Higher ('43-62% CPU, 37-40% RAM)

```
C:\Windows\System32\cmd.exe > Windows PowerShell > Windows PowerShell > Windows PowerShell > Windows PowerShell > + >
CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS
0215b4111b54 microservices-architecture-api_gateway-1 0.00% 38.8MiB / 3.754GiB 1.01% 23.4kB / 22.4kB 0B / 0B 1
034a7750a43c microservices-architecture-text_extraction_service-1 0.24% 375MiB / 3.754GiB 9.76% 1.17kB / 0B 0B / 0B 8
fb3db20fe67d microservices-architecture-summarization_service-1 43.71% 1.376GiB / 3.754GiB 36.66% 56.4kB / 27.5kB 0B / 0B 54
6818309bac91 microservices-architecture-file_service-1 0.20% 30.57MiB / 3.754GiB 0.80% 1.17kB / 0B 0B / 0B 1
b437881a305a microservices-architecture-voice_service-1 0.28% 36.05MiB / 3.754GiB 0.94% 1.02kB / 0B 0B / 0B 1
```

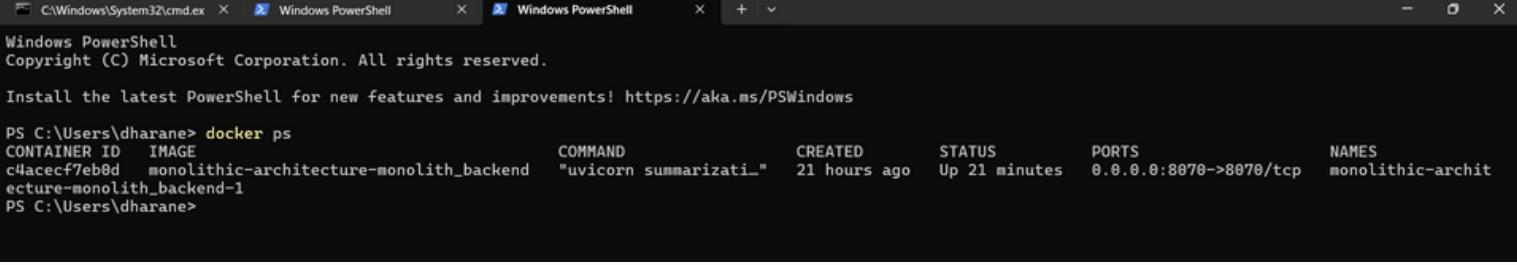
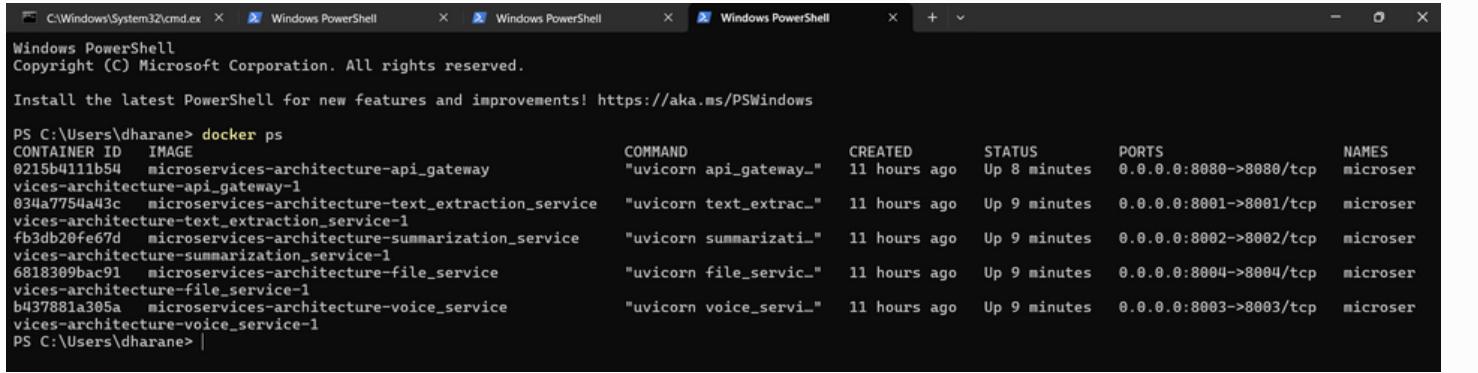
## Easier, centralized logs

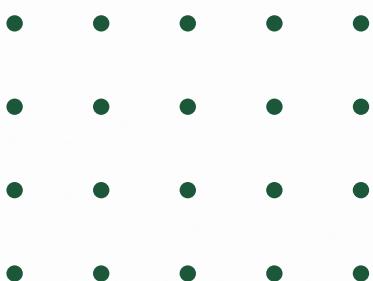
```
2025-03-05 16:38:37,926 - DEBUG - https://huggingface.co:443 "GET /api/models/sentence-transformers/all-MiniLM-L6-v2/revision/main HTTP/1.1" 200 6758
2025-03-05 16:38:37,944 - DEBUG - Starting new HTTPS connection (1): huggingface.co:443
2025-03-05 16:38:38,427 - DEBUG - https://huggingface.co:443 "HEAD /sentence-transformers/all-MiniLM-L6-v2/resolve/fa97f6e7cb1a59073dff9e6b13e2715cf7475ac9/1_Pooling/config.json HTTP/1.1" 200 0
2025-03-05 16:38:38,435 - DEBUG - Attempting to acquire lock 140063366953136 on /root/.cache/huggingface/hub/.locks/models--sentence-transformers--all-MiniLM-L6-v2/d1514c3162bbe87b343ff565fadcc62e6c06f04f03.lock
2025-03-05 16:38:38,437 - DEBUG - Lock 140063366953136 acquired on /root/.cache/huggingface/hub/.locks/models--sentence-transformers--all-MiniLM-L6-v2/d1514c3162bbe87b343ff565fadcc62e6c06f04f03.lock
2025-03-05 16:38:38,722 - DEBUG - https://huggingface.co:443 "GET /sentence-transformers/all-MiniLM-L6-v2/resolve/fa97f6e7cb1a59073dff9e6b13e2715cf7475ac9/1_Pooling/config.json HTTP/1.1" 200 190
2025-03-05 16:38:38,766 - DEBUG - Attempting to release lock 140063366953136 on /root/.cache/huggingface/hub/.locks/models--sentence-transformers--all-MiniLM-L6-v2/d1514c3162bbe87b343ff565fadcc62e6c06f04f03.lock
2025-03-05 16:38:38,768 - DEBUG - Lock 140063366953136 released on /root/.cache/huggingface/hub/.locks/models--sentence-transformers--all-MiniLM-L6-v2/d1514c3162bbe87b343ff565fadcc62e6c06f04f03.lock
2025-03-05 16:38:39,075 - DEBUG - https://huggingface.co:443 "GET /api/models/sentence-transformers/all-MiniLM-L6-v2 HTTP/1.1" 200 6758
2025-03-05 16:38:40,201 - DEBUG - Starting new HTTPS connection (1): www.languagetool.org:443
2025-03-05 16:38:40,843 - DEBUG - https://www.languagetool.org:443 "GET /download/LanguageTool-6.5.zip HTTP/1.1" 301 169
2025-03-05 16:38:40,848 - DEBUG - Starting new HTTPS connection (1): languagetool.org:443
2025-03-05 16:38:41,643 - DEBUG - https://languagetool.org:443 "GET /download/LanguageTool-6.5.zip HTTP/1.1" 200 248160977
Downloading LanguageTool 6.5: 100%[██████████] 248M/248M [0:1:48:00:00, 2.29MB/s]
2025-03-05 16:40:38,380 - INFO - Unzipping /tmp/tmpb4ryx6q3.zip to /root/.cache/language_tool_python.
2025-03-05 16:40:37,849 - INFO - Downloaded https://www.languagetool.org/download/LanguageTool-6.5.zip to /root/.cache/language_tool_python.
2025-03-05 16:40:44,419 - DEBUG - Starting new HTTP connection (1): 127.0.0.1:8081
2025-03-05 16:40:44,648 - DEBUG - http://127.0.0.1:8081 "GET /v2/languages HTTP/1.1" 200 3307
2025-03-05 16:40:44,675 - INFO - RAG Model components loaded successfully.
2025-03-05 16:40:45,597 - INFO - Datasets loaded successfully.
Batches: 100%[██████████] 21/21 [0:0:44:00:00, 2.12s/it]
2025-03-05 16:41:30,949 - INFO - FAISS index created and populated successfully.
2025-03-05 16:41:30,950 - INFO - FAISS index initialized successfully.
2025-03-05 16:41:30,957 - INFO - RAG Model initialized successfully.
2025-03-05 16:41:31,299 - INFO - Started server process [1]
2025-03-05 16:41:31,302 - INFO - Waiting for application startup.
2025-03-05 16:41:31,337 - INFO - Application startup complete.
2025-03-05 16:41:31,359 - INFO - Uvicorn running on http://0.0.0.0:8070 (Press CTRL+C to quit)
2025-03-05 16:42:57,793 - DEBUG - Calling on_field_start with no data
2025-03-05 16:42:57,795 - DEBUG - Calling on_field_name with data[0:10]
2025-03-05 16:42:57,796 - DEBUG - Calling on_field_data with data[11:14]
2025-03-05 16:42:57,796 - DEBUG - Calling on_field_end with no data
2025-03-05 16:42:57,796 - DEBUG - Calling on_end with no data
2025-03-05 16:42:57,956 - INFO - 172.18.0.1:38652 - "POST /process-query/ HTTP/1.1" 422
PS C:\Users\dharane>
```

## Debugging

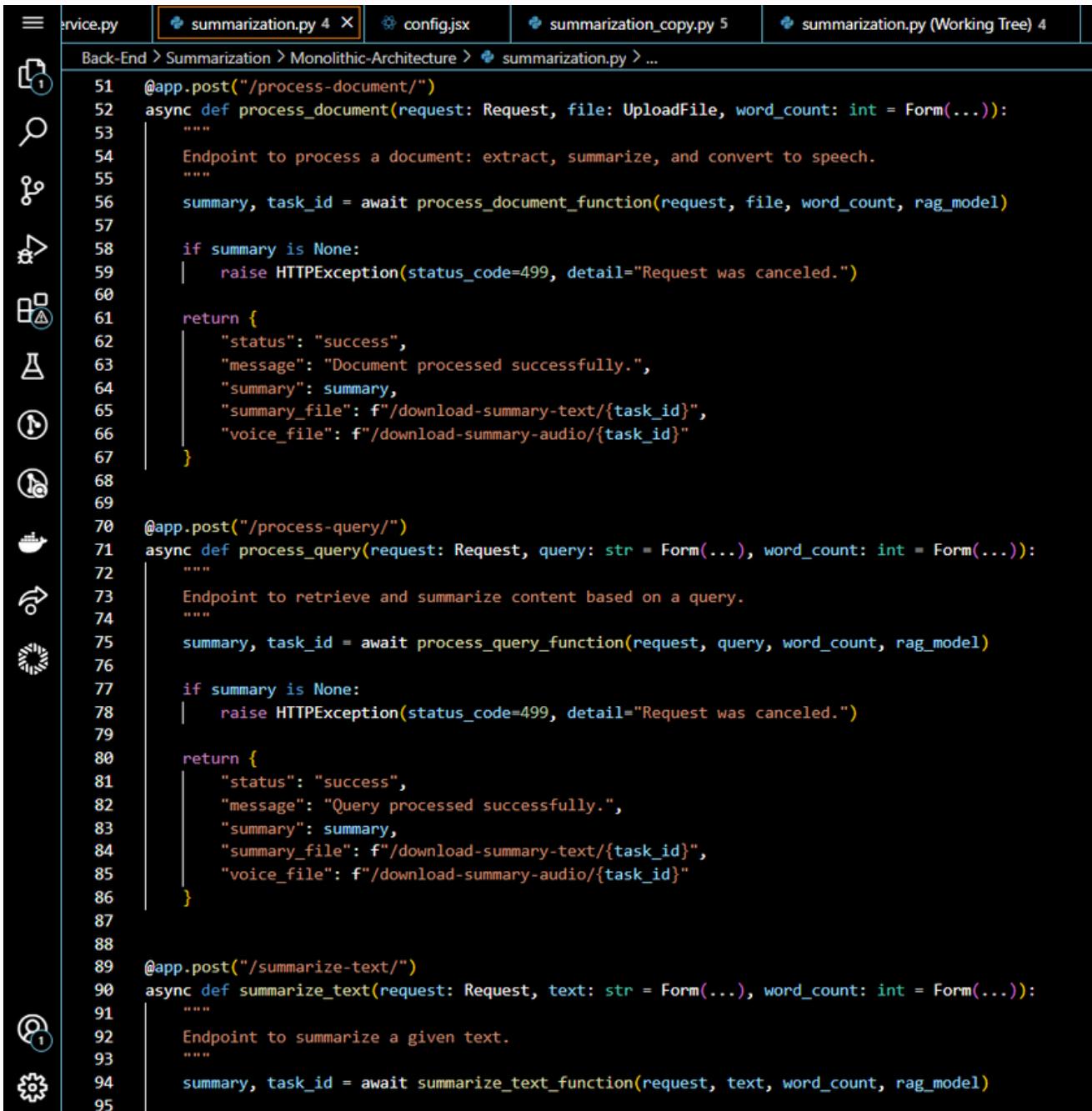
## Harder, distributed logs

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\dharane> docker logs microservices-architecture-voice_service-1
INFO: Started server process [1]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8003 (Press CTRL+C to quit)
2025-03-05 06:24:19,168 - INFO - Received text-to-speech request. Saving to: audio_770af9d180.mp3
2025-03-05 06:24:31,434 - INFO - Voice file generated successfully at path: audio_770af9d180.mp3
2025-03-05 06:24:31,436 - INFO - Text-to-speech conversion successful. Audio file saved at: audio_770af9d180.mp3
INFO: 172.19.0.6:36944 - "POST /synthesize/ HTTP/1.1" 200 OK
INFO: Shutting down
INFO: Waiting for application shutdown.
INFO: Application shutdown complete.
INFO: Finished server process [1]
INFO: Started server process [1]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8003 (Press CTRL+C to quit)
2025-03-05 09:11:22,129 - INFO - Received text-to-speech request. Saving to: audio_5c6dca2bc3ffb684f837c79a8cb10b55.mp3
2025-03-05 09:12:05,256 - INFO - Voice file generated successfully at path: audio_5c6dca2bc3ffb684f837c79a8cb10b55.mp3
2025-03-05 09:12:05,256 - INFO - Text-to-speech conversion successful. Audio file saved at: audio_5c6dca2bc3ffb684f837c79a8cb10b55.mp3
INFO: 172.19.0.6:35374 - "POST /synthesize/ HTTP/1.1" 200 OK
INFO: Shutting down
INFO: Waiting for application shutdown.
INFO: Application shutdown complete.
INFO: Finished server process [1]
INFO: Started server process [1]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8003 (Press CTRL+C to quit)
INFO: Shutting down
INFO: Waiting for application shutdown.
INFO: Application shutdown complete.
INFO: Finished server process [1]
INFO: Started server process [1]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8003 (Press CTRL+C to quit)
INFO: Shutting down
INFO: Waiting for application shutdown.
INFO: Application shutdown complete.
INFO: Finished server process [1]
INFO: Started server process [1]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8003 (Press CTRL+C to quit)
PS C:\Users\dharane>
```

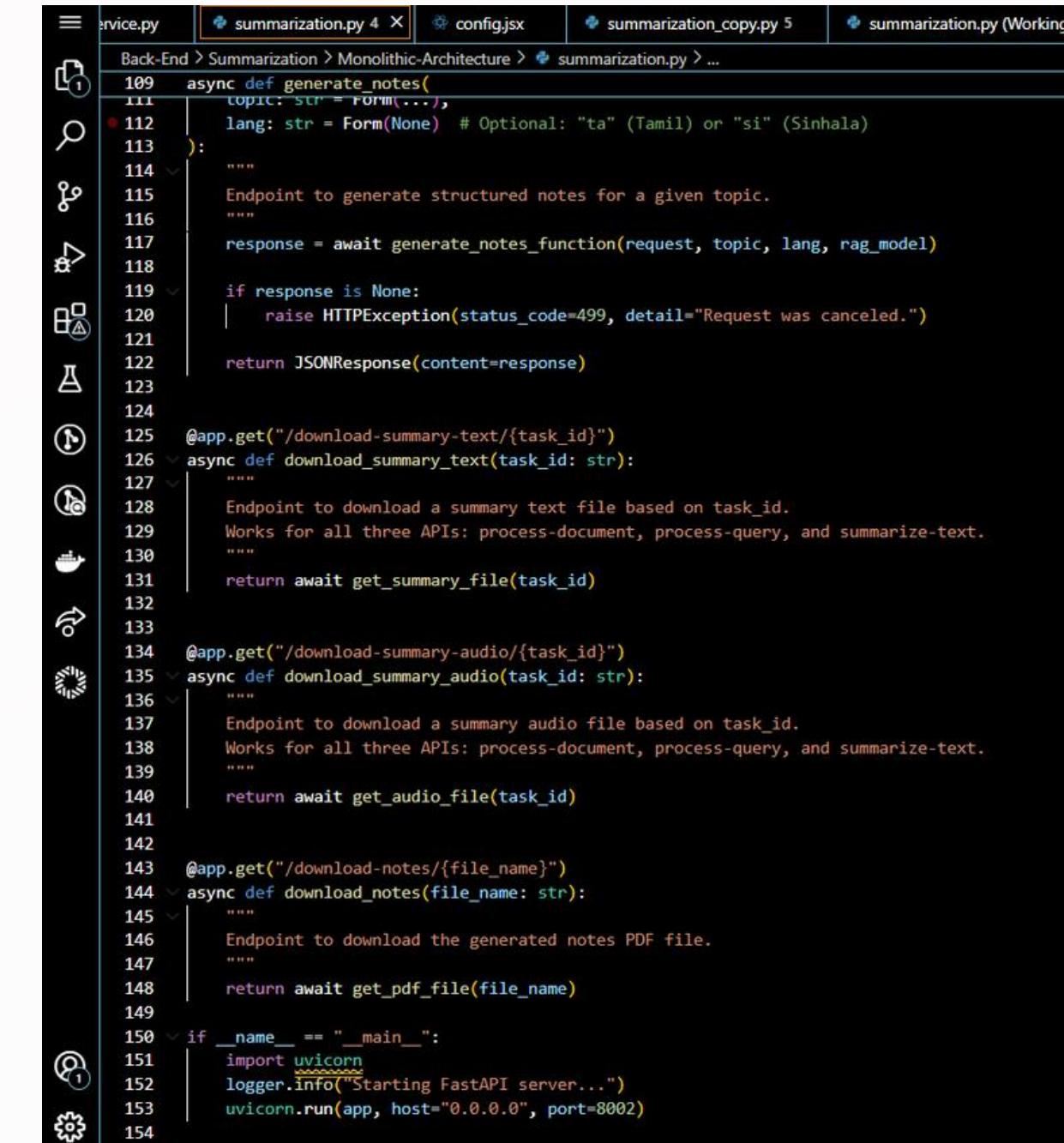
Metrics	Monolithic Architecture	Microservices Architecture
Infrastructure	<p><b>Simple, single container</b></p>  <pre> Windows PowerShell Copyright (C) Microsoft Corporation. All rights reserved.  Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows  PS C:\Users\dharane&gt; docker ps CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES c4acecf7eb0d monolithic-architecture-monolith_backend-1 "uvicorn summarizati..." 21 hours ago Up 21 minutes 0.0.0.0:8070-&gt;8070/tcp monolithic-archit PS C:\Users\dharane&gt; </pre>	<p><b>Complex, multiple containers</b></p>  <pre> Windows PowerShell Copyright (C) Microsoft Corporation. All rights reserved.  Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows  PS C:\Users\dharane&gt; docker ps CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 0215b411b54 microservices-architecture-api_gateway "uvicorn api_gateway..." 11 hours ago Up 8 minutes 0.0.0.0:8080-&gt;8080/tcp microser vices-architecture-api_gateway-1 034a7754a43c microservices-architecture-text_extraction_service "uvicorn text_extrac..." 11 hours ago Up 9 minutes 0.0.0.0:8001-&gt;8001/tcp microser vices-architecture-text_extraction_service-1 fb3db20fe67d microservices-architecture-summarization_service "uvicorn summarizati..." 11 hours ago Up 9 minutes 0.0.0.0:8002-&gt;8002/tcp microser vices-architecture-summarization_service-1 6818309bac91 microservices-architecture-file_service "uvicorn file_servic..." 11 hours ago Up 9 minutes 0.0.0.0:8004-&gt;8004/tcp microser vices-architecture-file_service-1 b427881a305a microservices-architecture-voice_service "uvicorn voice_servi..." 11 hours ago Up 9 minutes 0.0.0.0:8003-&gt;8003/tcp microser vices-architecture-voice_service-1 PS C:\Users\dharane&gt; </pre>
Fault Tolerance	<p><b>Lower, single failure risk</b></p>	<p><b>Higher, independent services</b></p>



# summarization.py



```
service.py summarization.py 4 X config.jsx summarization_copy.py 5 summarization.py (Working Tree) 4
Back-End > Summarization > Monolithic-Architecture > summarization.py > ...
51 @app.post("/process-document/")
52     async def process_document(request: Request, file: UploadFile, word_count: int = Form(...)):
53         """
54             Endpoint to process a document: extract, summarize, and convert to speech.
55         """
56         summary, task_id = await process_document_function(request, file, word_count, rag_model)
57
58         if summary is None:
59             raise HTTPException(status_code=499, detail="Request was canceled.")
60
61         return {
62             "status": "success",
63             "message": "Document processed successfully.",
64             "summary": summary,
65             "summary_file": f"/download-summary-text/{task_id}",
66             "voice_file": f"/download-summary-audio/{task_id}"
67         }
68
69
70 @app.post("/process-query/")
71     async def process_query(request: Request, query: str = Form(...), word_count: int = Form(...)):
72         """
73             Endpoint to retrieve and summarize content based on a query.
74         """
75         summary, task_id = await process_query_function(request, query, word_count, rag_model)
76
77         if summary is None:
78             raise HTTPException(status_code=499, detail="Request was canceled.")
79
80         return {
81             "status": "success",
82             "message": "Query processed successfully.",
83             "summary": summary,
84             "summary_file": f"/download-summary-text/{task_id}",
85             "voice_file": f"/download-summary-audio/{task_id}"
86         }
87
88
89 @app.post("/summarize-text/")
90     async def summarize_text(request: Request, text: str = Form(...), word_count: int = Form(...)):
91         """
92             Endpoint to summarize a given text.
93         """
94         summary, task_id = await summarize_text_function(request, text, word_count, rag_model)
95 
```



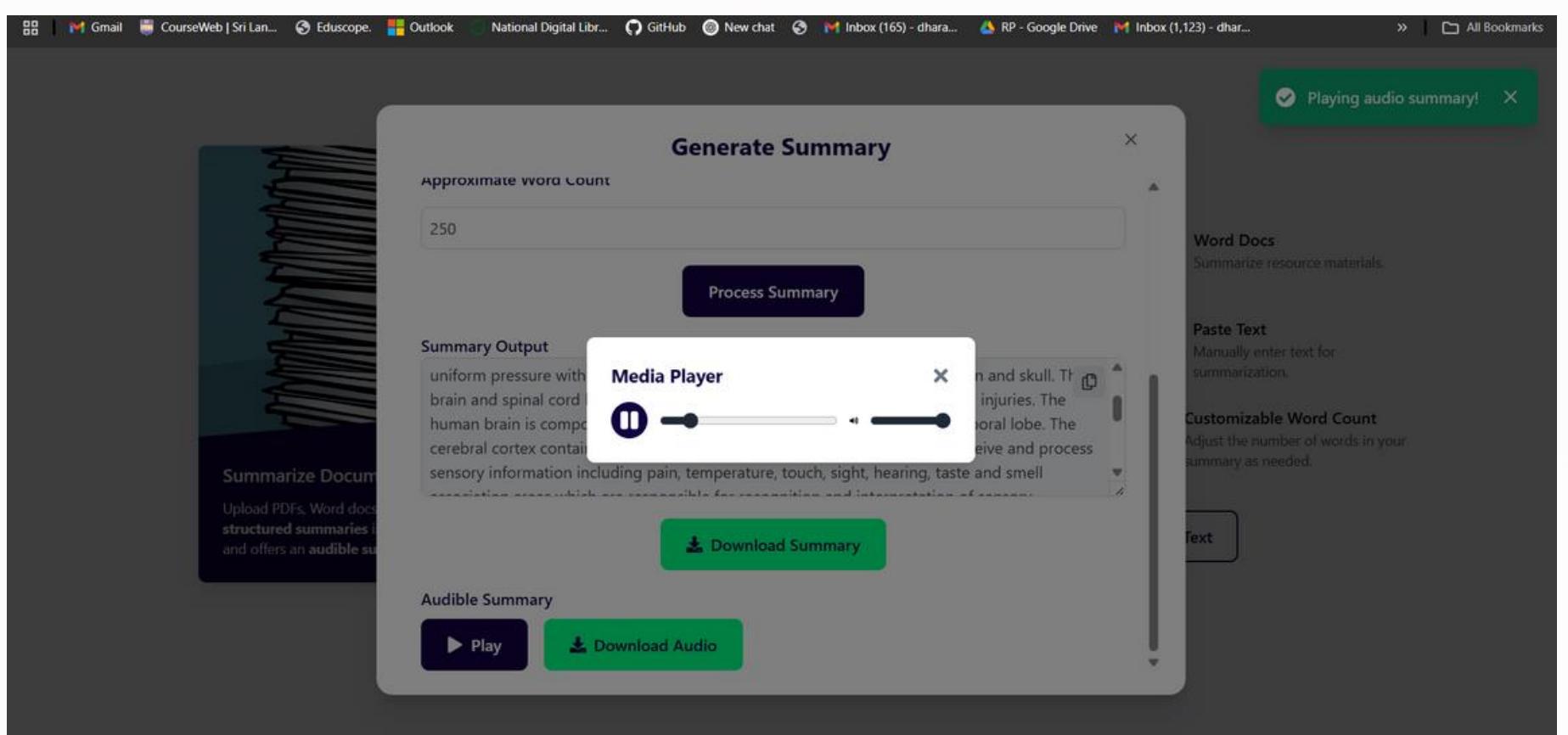
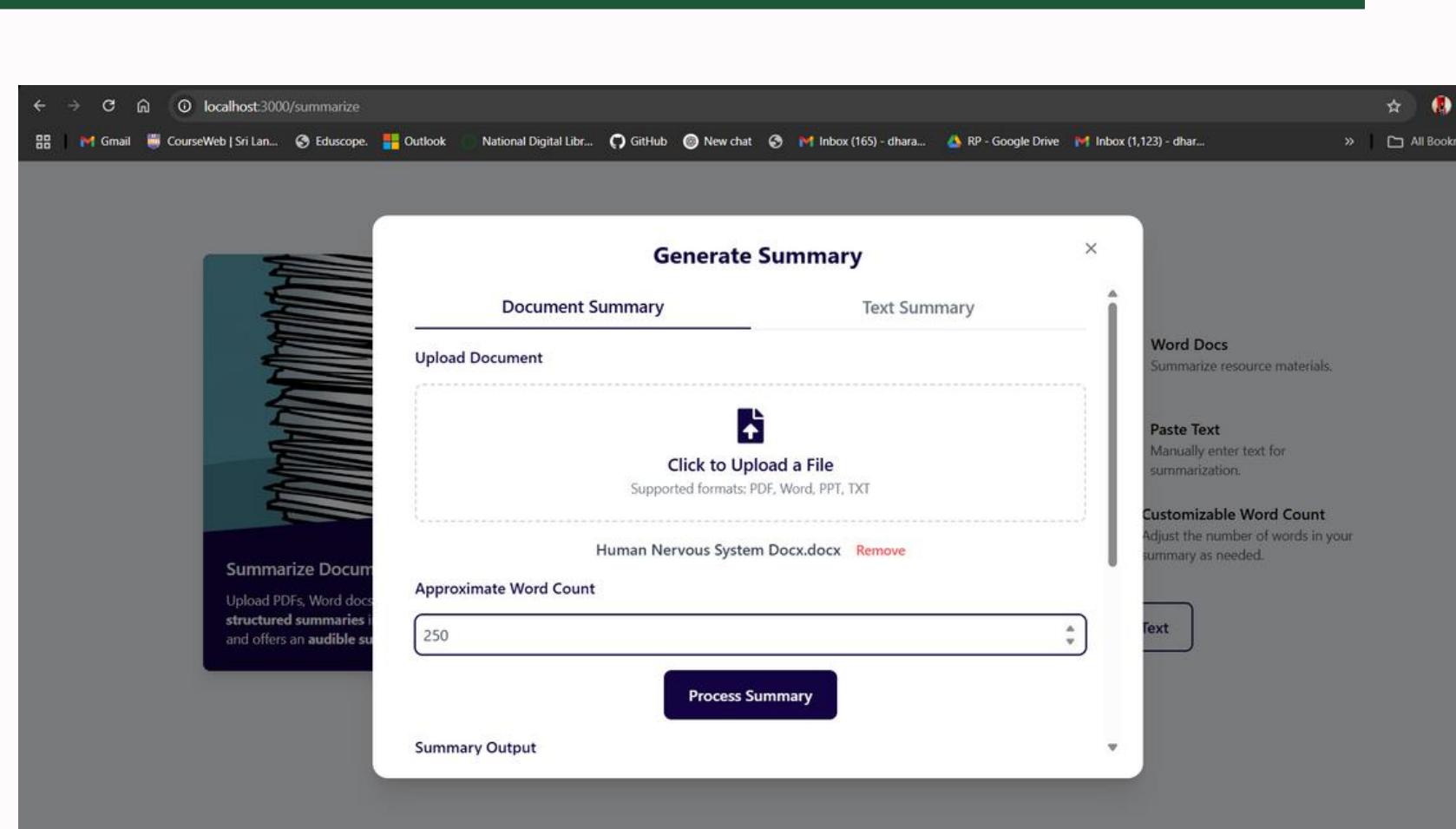
```
service.py summarization.py 4 X config.jsx summarization_copy.py 5 summarization.py (Working Tree)
Back-End > Summarization > Monolithic-Architecture > summarization.py > ...
109     async def generate_notes(
110         topic: str = Form(...),
111         lang: str = Form(None) # Optional: "ta" (Tamil) or "si" (Sinhala)
112     ):
113         """
114             Endpoint to generate structured notes for a given topic.
115         """
116         response = await generate_notes_function(request, topic, lang, rag_model)
117
118         if response is None:
119             raise HTTPException(status_code=499, detail="Request was canceled.")
120
121         return JSONResponse(content=response)
122
123
124
125     @app.get("/download-summary-text/{task_id}")
126     async def download_summary_text(task_id: str):
127         """
128             Endpoint to download a summary text file based on task_id.
129             Works for all three APIs: process-document, process-query, and summarize-text.
130         """
131         return await get_summary_file(task_id)
132
133
134     @app.get("/download-summary-audio/{task_id}")
135     async def download_summary_audio(task_id: str):
136         """
137             Endpoint to download a summary audio file based on task_id.
138             Works for all three APIs: process-document, process-query, and summarize-text.
139         """
140         return await get_audio_file(task_id)
141
142
143     @app.get("/download-notes/{file_name}")
144     async def download_notes(file_name: str):
145         """
146             Endpoint to download the generated notes PDF file.
147         """
148         return await get_pdf_file(file_name)
149
150     if __name__ == "__main__":
151         import uvicorn
152         logger.info("Starting FastAPI server...")
153         uvicorn.run(app, host="0.0.0.0", port=8002)
154 
```

# Summarization.jsx

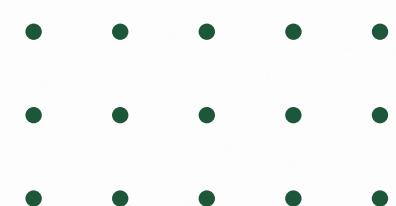
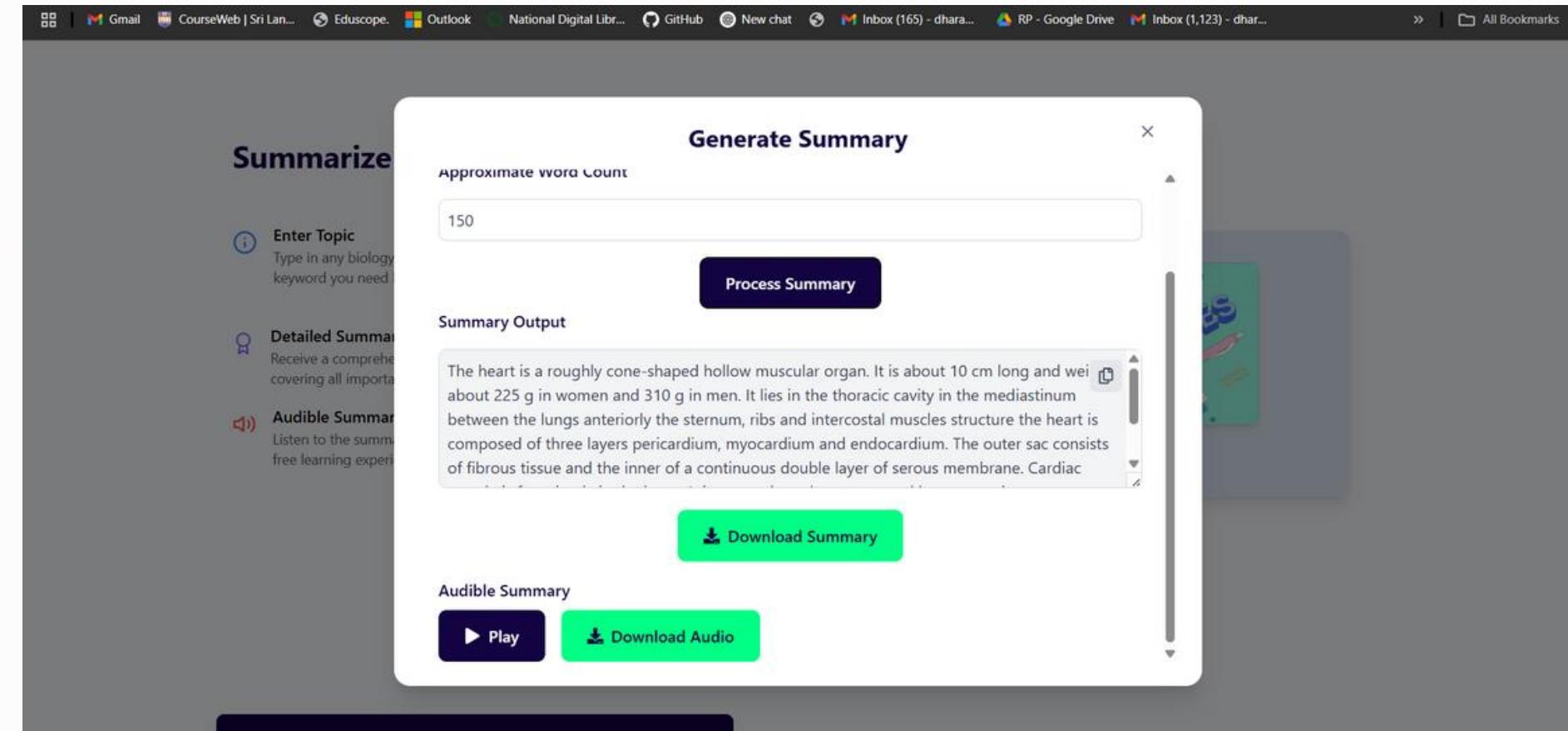
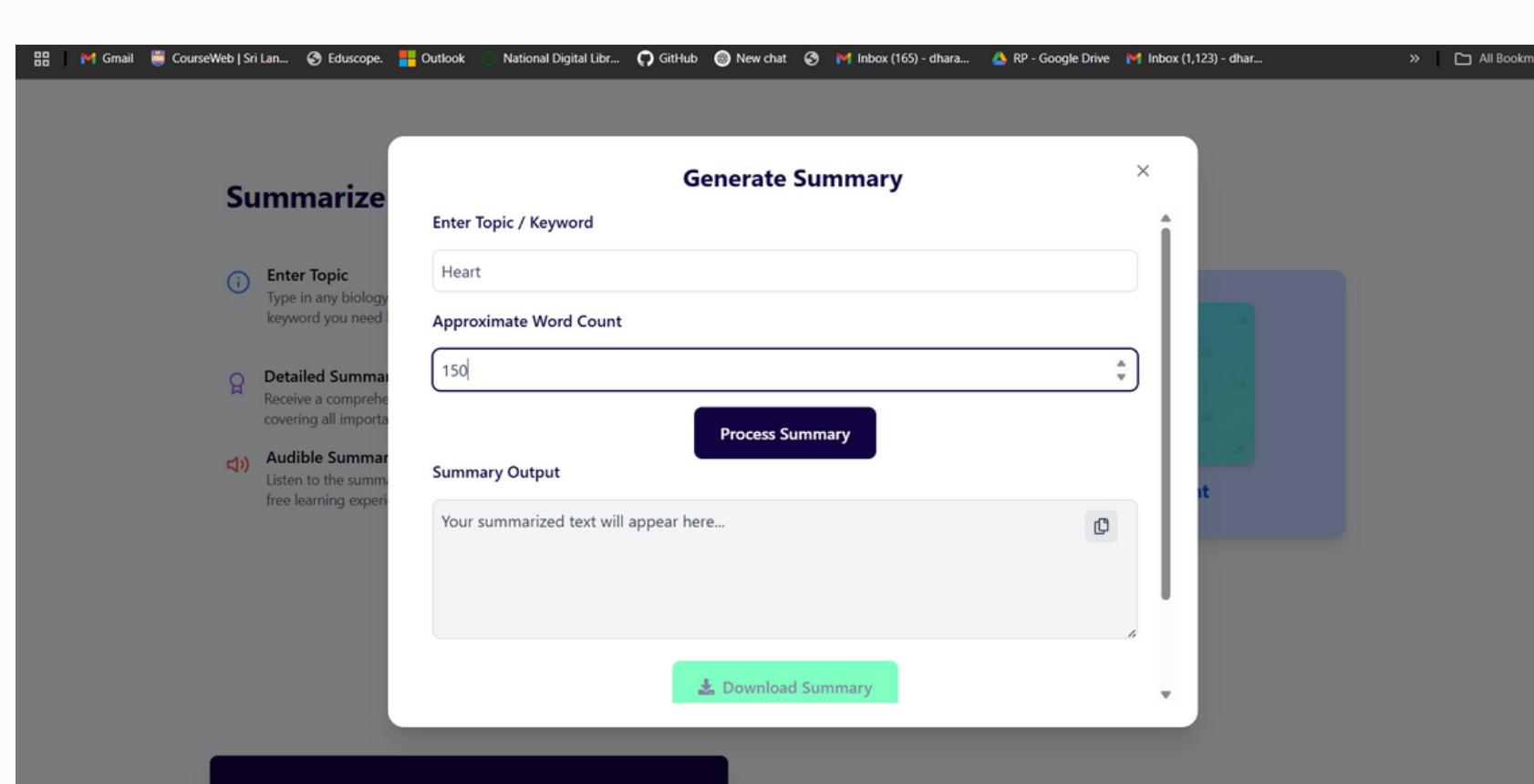


```
Summarization.jsx 1 X
Front-End > src > components > Summarization > Summarization.jsx > ...
You, 3 weeks ago | 1 author (You)
1 import React, { useRef } from "react"; You, 4 weeks ago • F...
2 import Hero from "./Hero";
3 import SummarizeDocument from "./SummarizeDocument";
4 import TopicSummary from "./TopicSummary";
5 import GenerateNotes from "./GenerateNotes";
6
7 const Summarization = () => {
8   const summarizeRef = useRef(null);
9   const topicSummaryRef = useRef(null); // Ref for TopicSummary
10
11   const scrollToSummarize = () => {
12     if (summarizeRef.current) {
13       const yOffset = -80;
14       const y =
15         summarizeRef.current.getBoundingClientRect().top +
16         window.scrollY +
17         yOffset;
18       window.scrollTo({ top: y, behavior: "smooth" });
19     }
20   };
21
22   return (
23     <div className="bg-gray-100">
24       /* Pass scroll function to Hero */
25       <Hero scrollToSummarize={scrollToSummarize} />
26
27       /* Sections */
28       <div ref={summarizeRef} className="mt-10">
29         <SummarizeDocument />
30       </div>
31
32       /* Add ref to Topic Summary */
33       <div ref={topicSummaryRef} className="mt-10">
34         <TopicSummary />
35       </div>
36
37       <div ref={topicSummaryRef} className="mt-10">
38         <GenerateNotes />
39       </div>
40     </div>
41   );
42 }
43
44 export default Summarization;
```

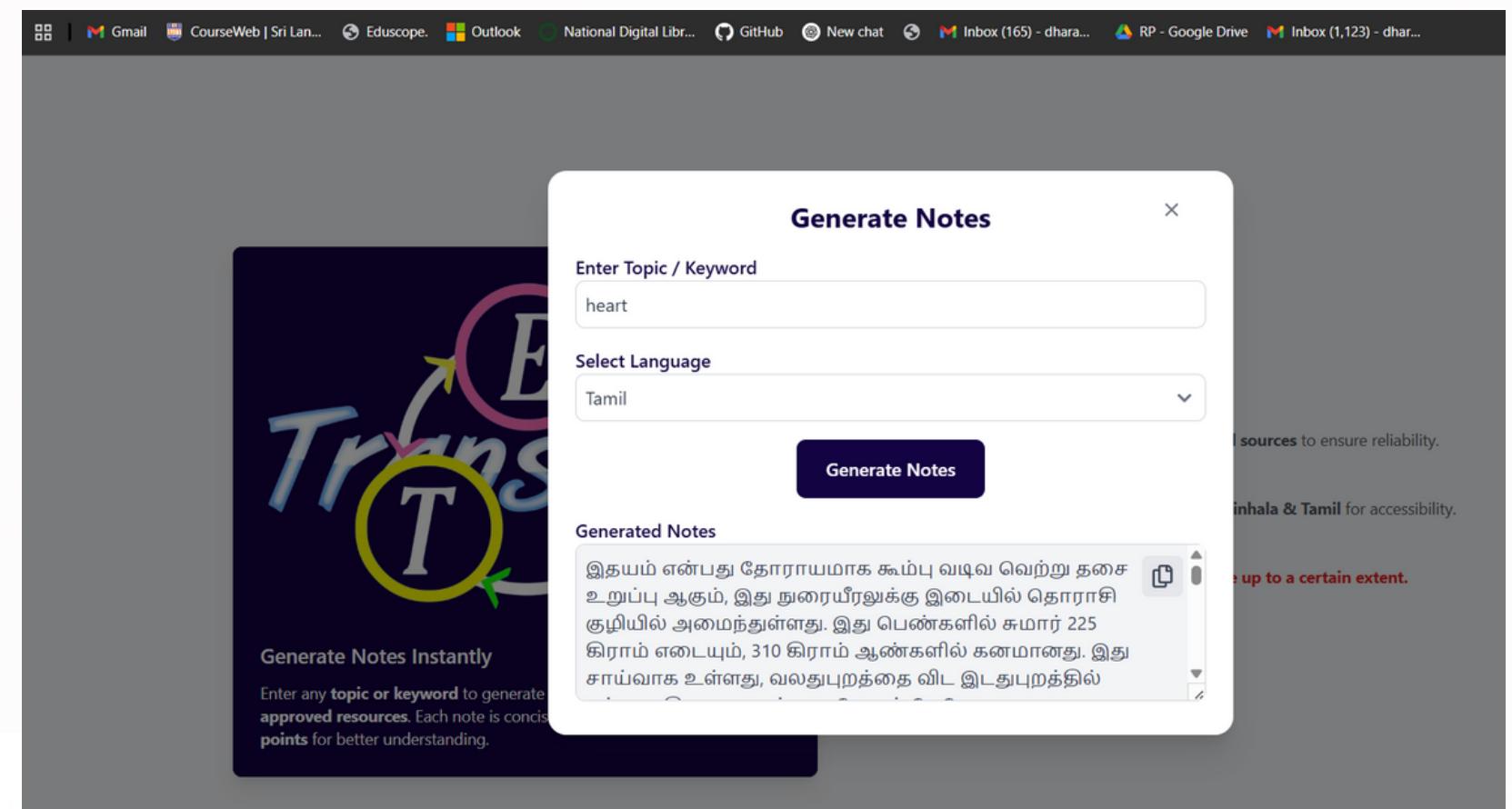
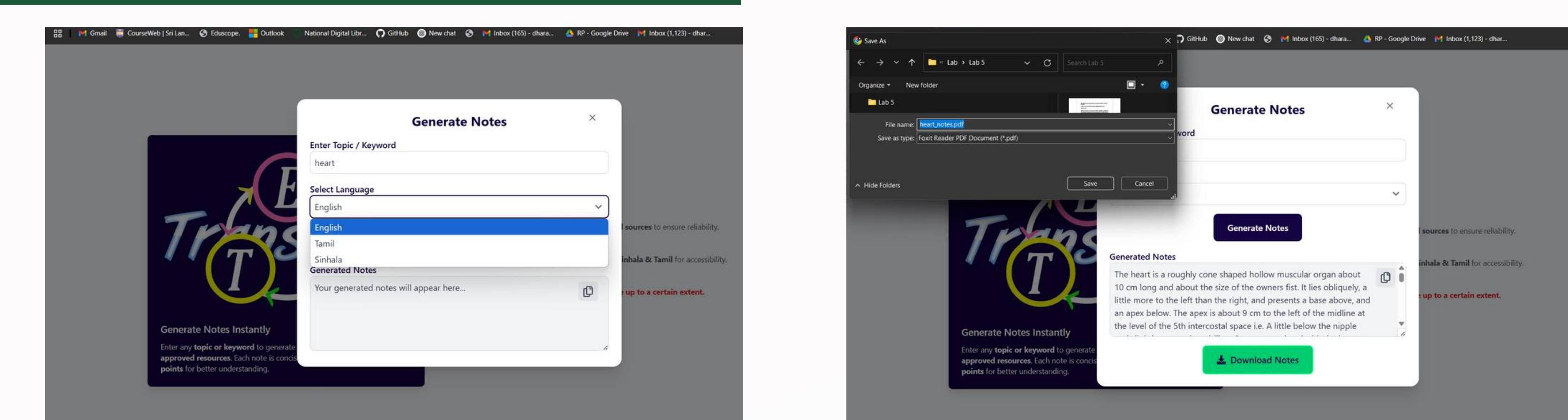
# Working Product



# Working Product

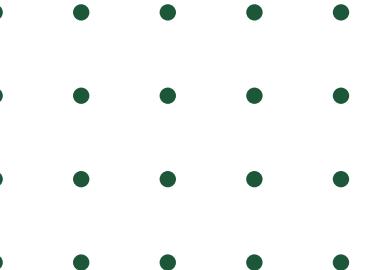


# Working Product



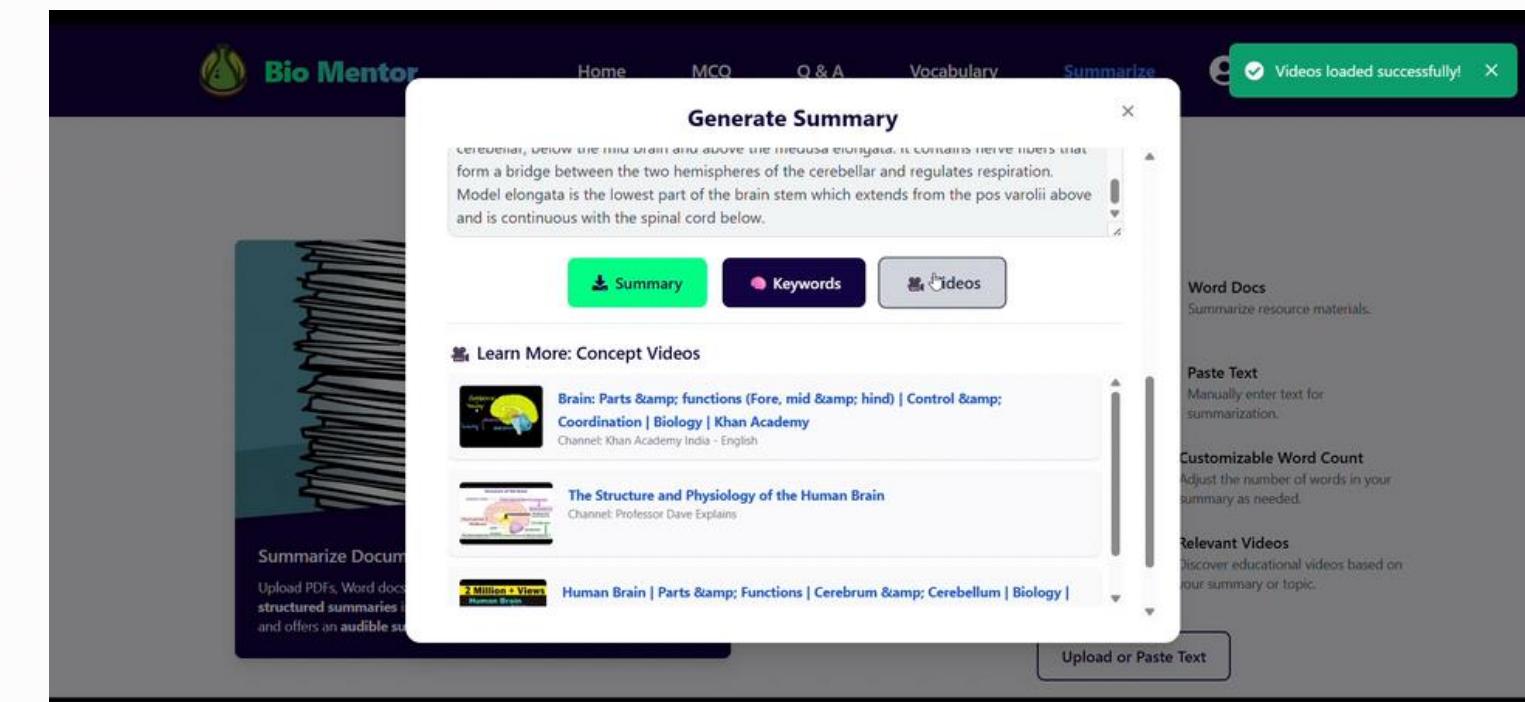
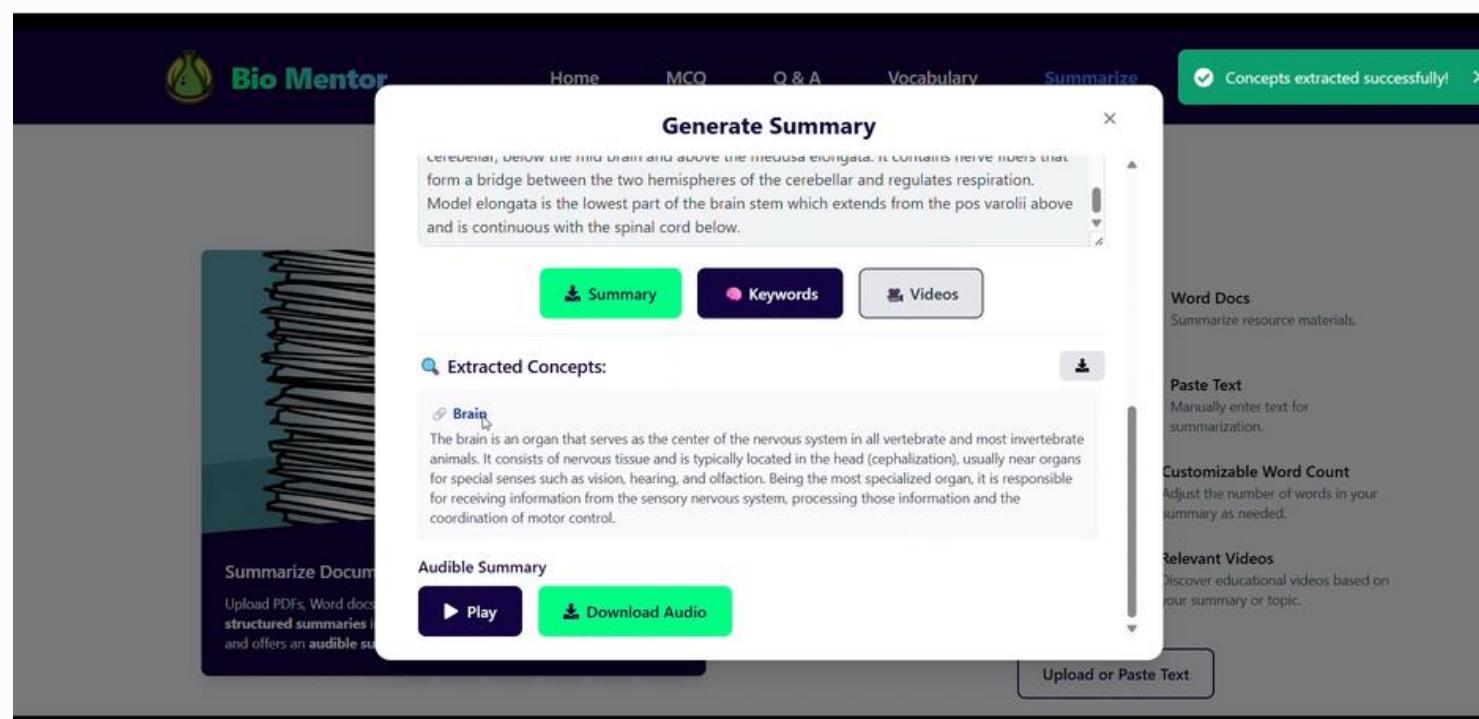


# Completion of the project



# Project Enhancements

- **Concept Extraction** – Identifies scientific keywords using nltk + Wikipedia API and returns their definitions.
- **YouTube Video Suggestion** – Retrieves educational biology videos by extracting core topics from summaries or queries using TF-IDF + YouTube Data API.
- **Inappropriate Content Detection** – Incorporated zero-shot classification model



# Testing - Unit Testing

- Framework: Pytest

- Modules Tested:

- test\_file\_handler.py: File saving, PDF generation
- test\_text\_extraction.py: Text cleaning & grammar correction
- test\_rag.py: Summarization trimming & formatting
- test\_voice\_service.py: TTS output in memory

```
Back-End > Summarization > Monolithic-Architecture > tests > test_rag.py > test_truncate_to_word_count
1 import pytest
2 import sys
3 import os
4 sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))
5 from rag import RAGModel
6
7 class DummyModel:
8     def generate_summary_for_long_text(self, text, max_words=100):
9         return "summary"
10
11 def test_truncate_to_word_count():
12     text = "This is a test sentence. " * 50
13     model = DummyModel()
14     result = RAGModel.truncate_to_word_count(text, max_words=20)
15     assert len(result.split()) <= 20
16
17 def test_postprocess_summary():
18     summary = "this is a test. this is another sentence."
19     model = DummyModel()
20     result = RAGModel.postprocess_summary(model, summary)
21     assert result[0].isupper()
22     assert result.endswith(".")
```

```
test_file_handler.py 2, U X
Back-End > Summarization > Monolithic-Architecture > tests > test_file_handler.py > ...
● 1 import pytest
2 from fastapi import UploadFile
3 from io import BytesIO
4 import sys
5 import os
6 sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))
7 from file_handler import save_uploaded_file, generate_pdf
8
9 def test_save_uploaded_file(tmp_path):
10    dummy_content = b"Hello, this is test content."
11    file = UploadFile(filename="test.txt", file=BytesIO(dummy_content))
12    path = save_uploaded_file(file)
13    assert os.path.exists(path)
14    os.remove(path) # Clean up
15
16 def test_generate_pdf():
17    content = "Line 1\nLine 2"
18    title = "Test PDF"
19    result = generate_pdf(content, title)
20    assert isinstance(result, bytes)
21    assert result.startswith(b"%PDF")
```

# Testing - Integration Testing

- Framework: Pytest with FastAPI TestClient

- Workflows Tested:
  - /process-document/
  - /summarize-text/
  - /process-query/
  - /generate-notes/

```
import os
import io
import pytest
from fastapi.testclient import TestClient
from summarization import app
from summarization_functions import file_store
from reportlab.pdfgen import canvas

client = TestClient(app)

@pytest.fixture
def canvas_pdf():
    buffer = io.BytesIO()
    c = canvas.Canvas(buffer)
    c.drawString(100, 700, """
    Photosynthesis is a process used by plants and other organisms to convert light energy into
    chemical energy.
    This energy is stored in carbohydrate molecules, such as sugars, which are synthesized from
    carbon dioxide and water.
    """)
    c.save()
    buffer.seek(0)
    return buffer

def test_summarize_text():
    response = client.post("/summarize-text/", data={
        "text": "Photosynthesis is the process by which plants convert sunlight into chemical energy.",
        "word_count": 50
    })
    assert response.status_code == 200
    data = response.json()
    assert "summary" in data
    assert isinstance(data["summary"], str)

def test_process_query():
    response = client.post("/process-query/", data={
        "query": "Heart",
        "word_count": 50
    })
    assert response.status_code == 200
    data = response.json()
    assert "summary" in data
    assert isinstance(data["summary"], str)

def test_generate_notes():
    response = client.post("/generate-notes/", data={
        "topic": "Heart",
        "lang": "en"
    })
    assert response.status_code == 200
    data = response.json()
    assert "structured_notes" in data
    assert "audio_file" in data
    if "download_link" in data:
        filename = data["download_link"].split("/")[-1]
        assert filename in file_store
        assert file_store[filename].startswith(b"\nPDF")

def test_process_document_and_download(dummy_pdf):
    response = client.post(
        "/process-document/",
        files={"file": ("test.pdf", dummy_pdf, "application/pdf")},
        data={"word_count": 50}
    )
    assert response.status_code == 200
    data = response.json()
    assert "pdf_file" in data
    assert "audio_file" in data
    assert "voice_file" in data

    task_id = data["summary_file"].split("/")[-1]

    # fake file store entries if backend failed to write files
    file_store.setdefault("summary_{task_id}.pdf", b"PDF test data")
    file_store.setdefault("summary_{task_id}.mp3", b"MP3 test data")

    summary_response = client.get(f"/download-summary-text/{task_id}")
    assert summary_response.status_code == 200
    assert summary_response.headers["content-type"] == "application/pdf"

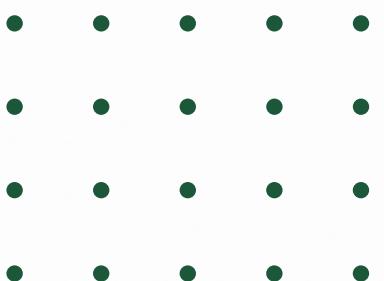
    audio_response = client.get(f"/download-summary-audio/{task_id}")
    assert audio_response.status_code == 200
    assert audio_response.headers["content-type"] == "audio/mpeg"

def test_download_notes_direct():
    filename = "notes_Cell_Division_English.pdf"
    file_store[filename] = b"Fake notes content"
    response = client.get(f"/download-notes/{filename}")
    assert response.status_code == 200
    assert response.headers["content-type"] == "application/pdf"
```

```
platform win32 -- Python 3.10.11, pytest-8.3.5, pluggy-1.5.0
rootdir: D:\Downloads\RP\Summarization\RP\Back-End\Summarization\Monolithic-Architecture\tests
configfile: pytest.ini
plugins: anyio-4.8.0
collected 13 items

tests\test_file_handler.py ...
tests\test_integration.py .....
tests\test_rag.py ...
tests\test_text_extraction.py ...
tests\test_voice_service.py .

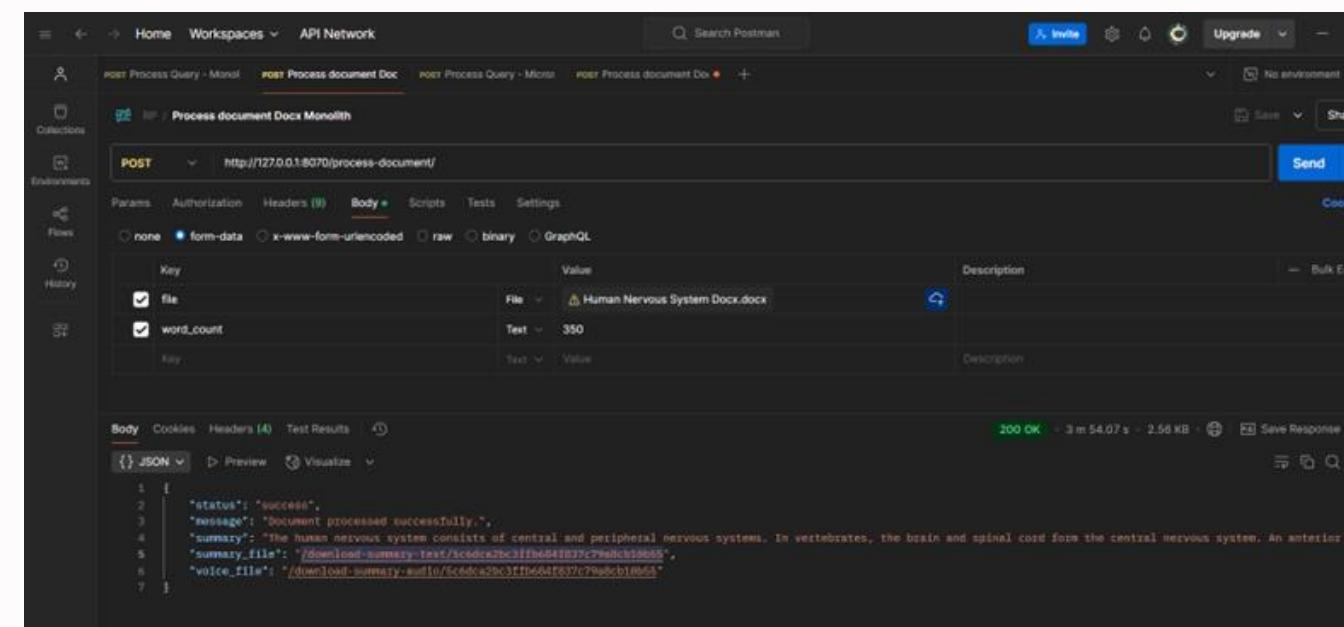
-- DOCS: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
=====
===== 13 passed, 7 warnings in 308.41s (0:05:08) ====
sys:1: DeprecationWarning: builtin type swigvarlink has no __module__ attribute
o (venv) PS D:\Downloads\RP\Back-End\Summarization\Monolithic-Architecture>
```



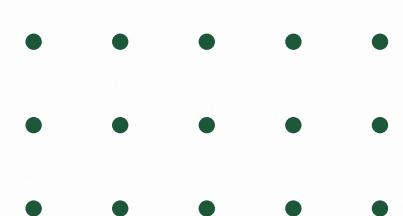
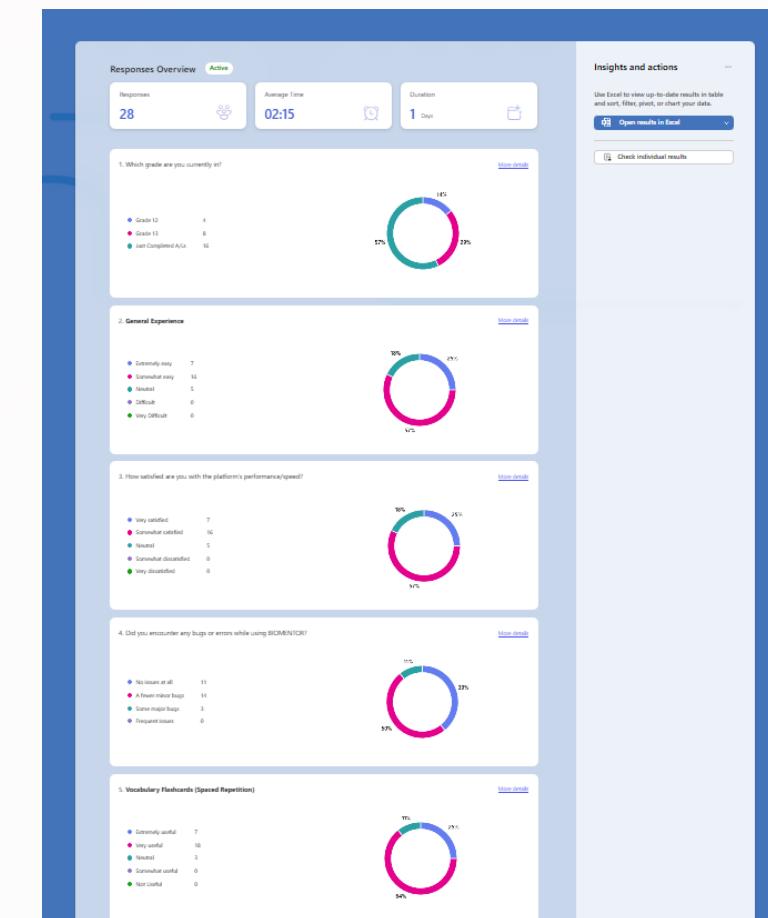
# Testing - System Testing and UAT

## System Testing

- Framework: Postman



## User Acceptance Testing



# Deployment

- Model hosted on Hugging Face Hub
- Backend deployed on Azure VM (Ubuntu, 4 vCPUs, 16 GB RAM), Region: Central India for low latency
- CI/CD is handled via GitHub Actions, which on every push to main, SSHs into the Azure VM, pulls the latest code, installs dependencies, and restarts the backend using systemctl.

The screenshot shows the Microsoft Azure portal interface for a virtual machine named "BioMentor-Summarization-VM". The main pane displays the VM's properties, including its resource group (BioMentor-Summarization-Resource-Group), operating system (Ubuntu 24.04), size (Standard D4s v3), and public IP address (20.193.248.25). The "Networking" section shows the VM is connected to a network interface with a private IP of 10.0.0.5. The left sidebar provides navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Resource visualizer, Connect, Networking, Application security groups, Network manager, Settings, Availability + scale, Security, and Backup + disaster recovery.

The screenshot shows the Hugging Face Hub profile page for "Dharane Segar". The profile picture is a solid purple circle. The user has 1 model listed: "DharaneSegar/flant5-bio-summarization", which is a Text2Text Generation model updated on March 30. There are 0 datasets listed under "Datasets". The page includes navigation links for Models, Datasets, Spaces, Community, Docs, Enterprise, and Pricing.

# References

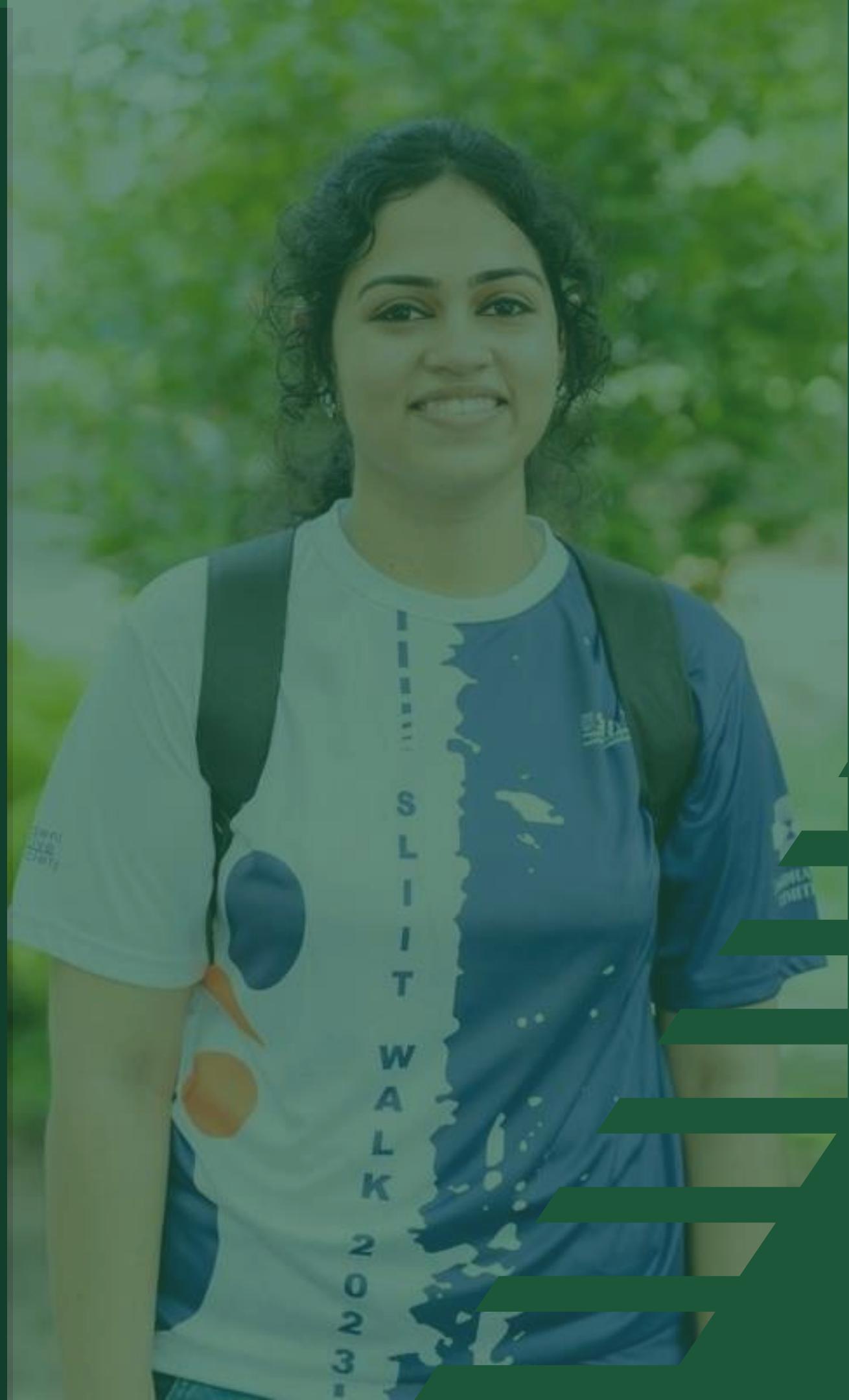
- [1] Abstractive Text Summarization Using BART. [ONLINE] <https://ieeexplore.ieee.org/document/9972639> [Accessed 15 May 2024]
- [2] NLP-Enhanced Long Document Summarization: A Comprehensive Approach for Information Condensation. [ONLINE] <https://ieeexplore.ieee.org/document/10551101> [Accessed 20 May 2024]
- [3] Topic level summary generation using BERTinduced Abstractive Summarization Model. [ONLINE] <https://ieeexplore.ieee.org/document/9972639> [Accessed 01 June 2024]
- [4] Speech-to-Text and Text-to-Speech Recognition using Deep Learning. [ONLINE] <https://ieeexplore.ieee.org/document/9972639> [Accessed 05 June 2024]
- [5] A Novel Text To Speech Conversion Using Hierarchical Neural Network. [ONLINE] <https://ieeexplore.ieee.org/document/10533516> [Accessed 05 June 2024]
- [6] An Abstractive Summarization and Conversation Bot using T5 and its Variants. [ONLINE]. <https://ieeexplore.ieee.org/document/9972639> [Accessed 01 Aug 2024]
- [7] Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation[ONLINE]. <https://ieeexplore.ieee.org/document/9717259> [Accessed 01 Nov 2024]
- • • • •
- • • • •
- • • • •
- • • • •
- • • • •

# **IT21264634**

**Sujitha.S**

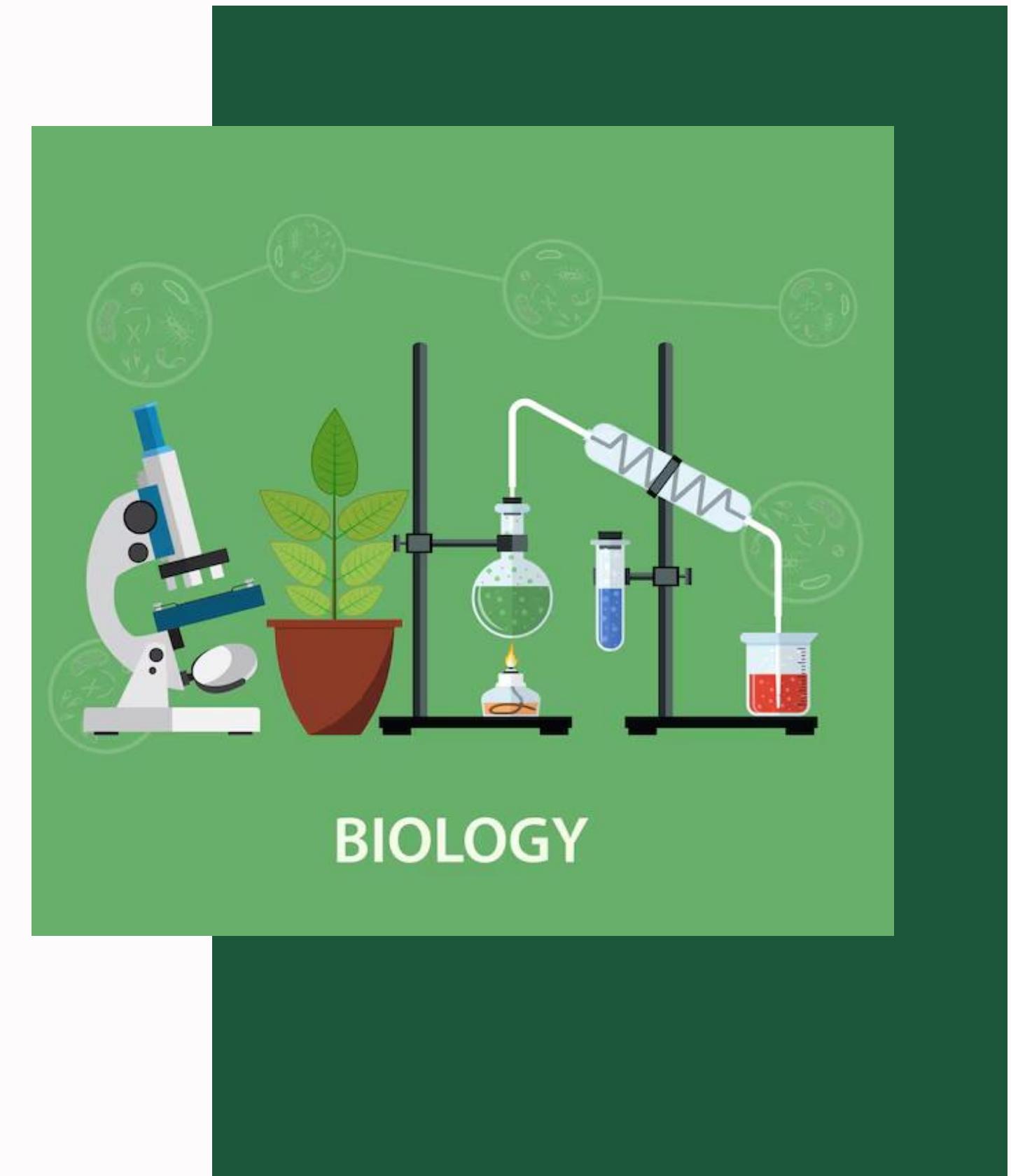
**Software Engineering**

**LLM BASED ADAPTIVE QUIZ  
PLATFORM TO IMPROVE MCQ  
ANSWERING SKILLS IN BIOLOGY  
FOR STUDENTS**

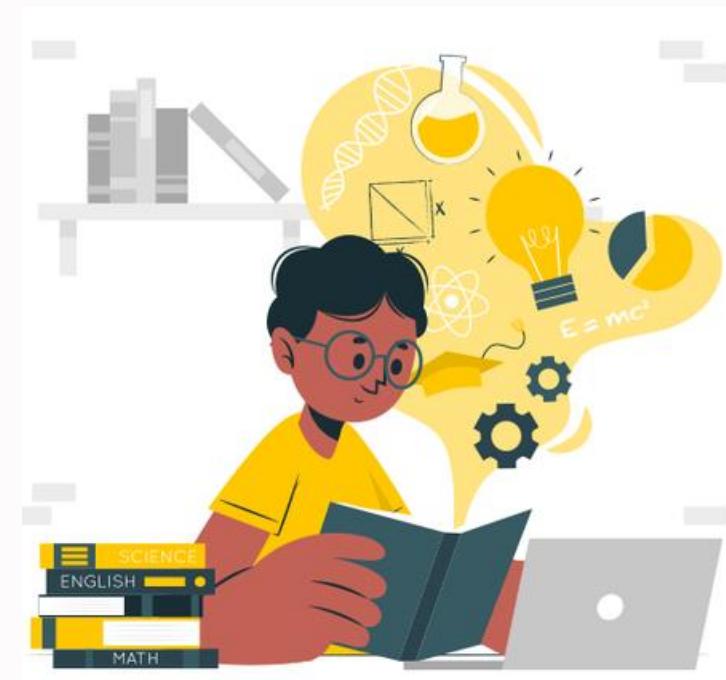


# Introduction

- 01** Background
- 02** Research Problem
- 03** Main and Sub Objectives
- 04** Methodology
- 05** Completion Of Project



# BACKGROUND



Increasing demand  
for personalized  
learning experiences  
in education.



Need for adaptive  
assessment tools that  
cater to individual  
learning paces and  
capabilities.

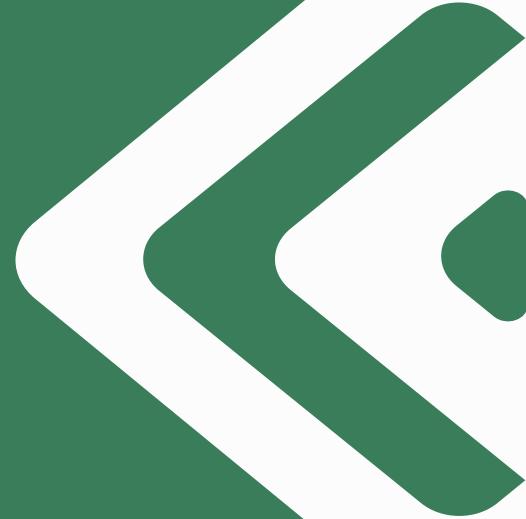
# RESEARCH PROBLEM

01



Traditional quiz platforms lack real-time adaptability, limiting their ability to adjust question difficulty based on a learner's performance trends.

02



Existing assessment systems do not effectively utilize performance-based difficulty adjustments, leading to less personalized and less efficient learning experiences.

# OBJECTIVES

## Objective 1

Develop a diverse set of MCQs categorized by difficulty levels for every user to engage different cognitive skills effectively.

## Main Objective

Develop an intelligent adaptive quiz system that personalizes MCQs based on student performance, integrates government-approved educational resources, and dynamically adjusts difficulty to cater to varying proficiency levels.

## Objective 2

Enhance the platform's ability to identify performance and provide targeted practice questions.

## Objective 3

Implement a continuous performance tracking system that allows users to monitor their progress and improvements over time.

## Objective 4

Ensure a user-friendly interface with accessible features to accommodate diverse learner needs.

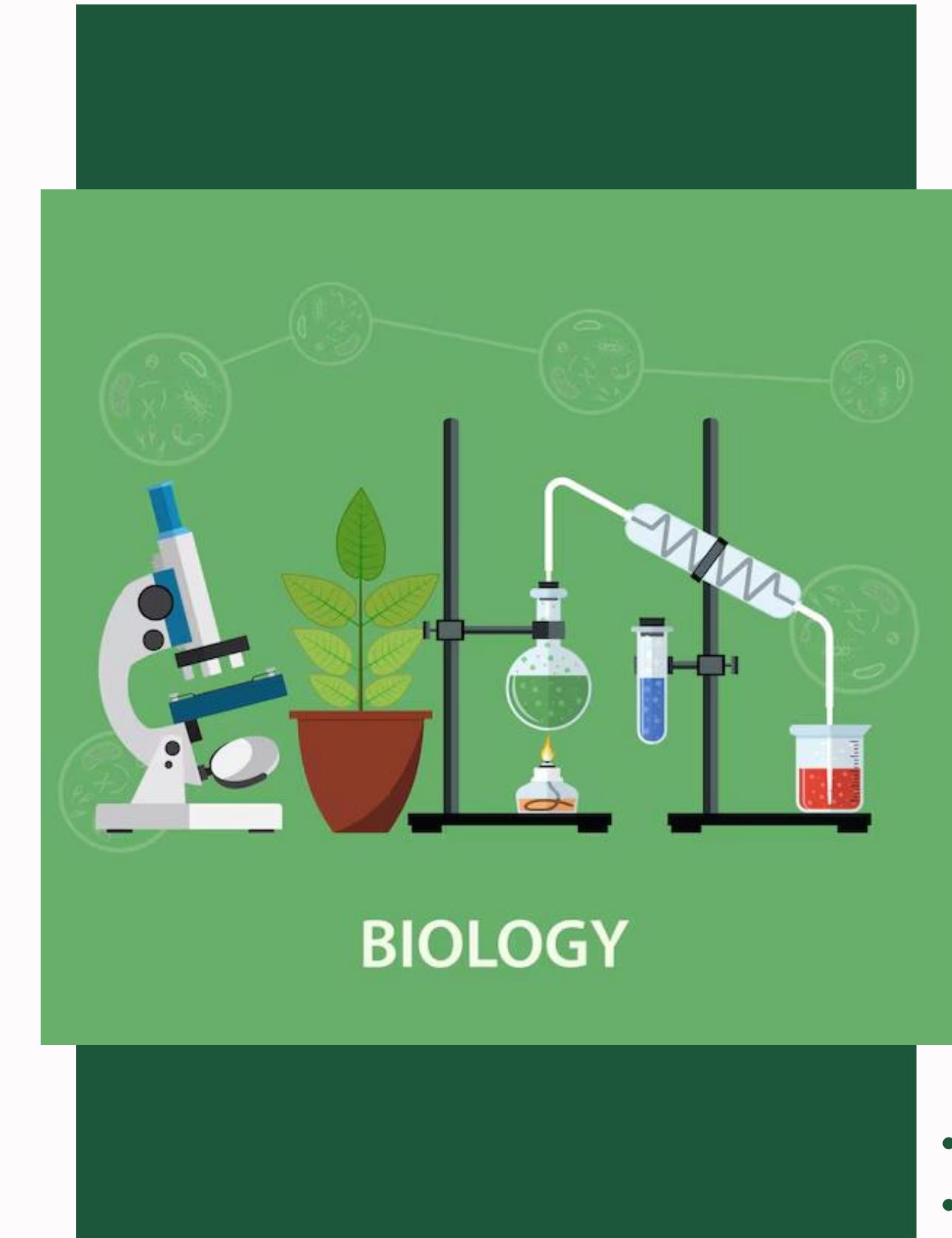
# Methodology

01 System Diagram

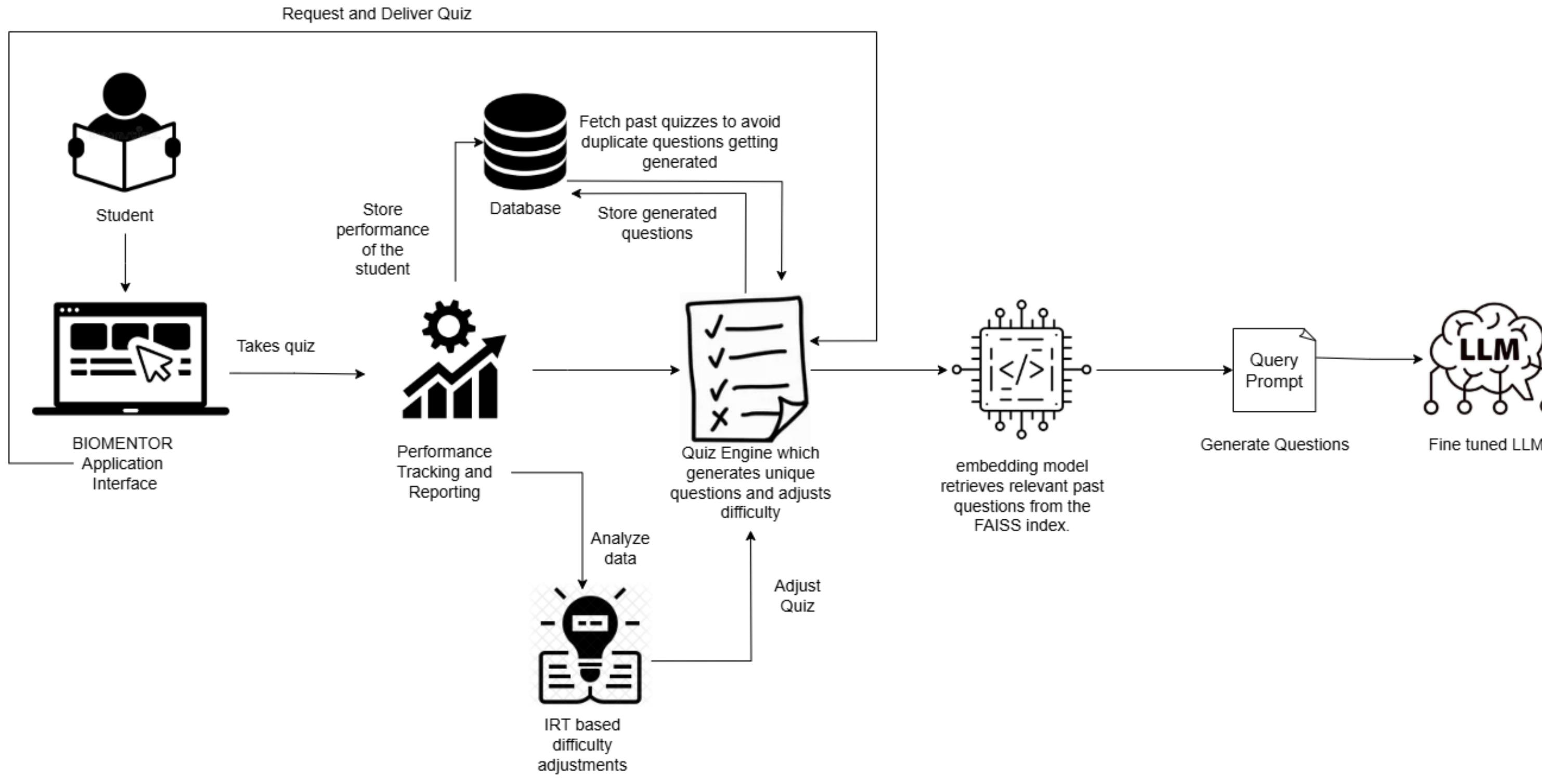
02 Tools and Technologies

03 Best Practices

04 GitHub Commits



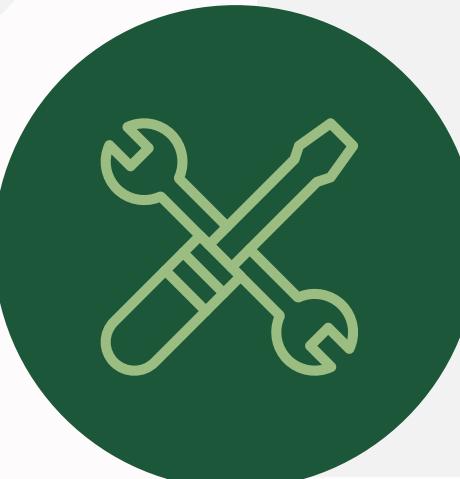
# System Diagram



# Tools & Technologies



**Project Management**  
Jira



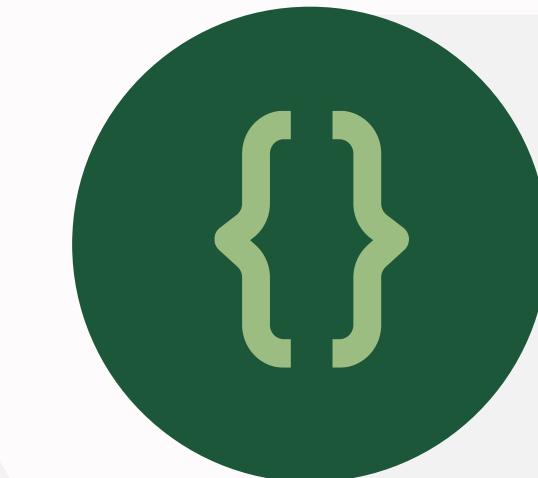
**Other tools**  
Git  
Draw.io  
Postman  
Figma



**Database**  
Faiss  
PMongoDB



**Frameworks**  
Transformers  
Pandas  
PyTorch  
Flask  
NumPy  
Pandas  
Matplotlib



**Programming Languages**  
Python  
React Js

# Best Practices

## Non-Functional Requirements

- Compatibility
  - Accuracy
- Performance
  - Usability
- Availability

## Design Excellence

- Real-Time Performance
- Accuracy and Reliability
- User Experience
  - Integration

## Standards & Best Practices

- Code Structure & Organization
- Web Security
- Model Quality
- RESTful API Design

# GitHub Commits

Commits

Filter: IT21264634/Sujit... · All users · All time

-o Commits on May 25, 2025

**Merge pull request #76 from Y3S1-GRP22/IT21264634/Sujitha** · Verified · 1a5123f ·  · 

Sujitha1221 authored 15 minutes ago · ✘ 0 / 1

**Merge branch 'main' of https://github.com/Y3S1-GRP22/BioMentor-Personalized-E-Learning-Platform into IT21264634/Sujitha** · 8a10582 ·  · 

Sujitha1221 committed 17 minutes ago

**changed explanation model** · be85318 ·  · 

Sujitha1221 committed 17 minutes ago

**increased number of questions** · 22aa991 ·  · 

Sujitha1221 committed 18 minutes ago · ✘ 0 / 1

**updated test scripts** · a3ed1ee ·  · 

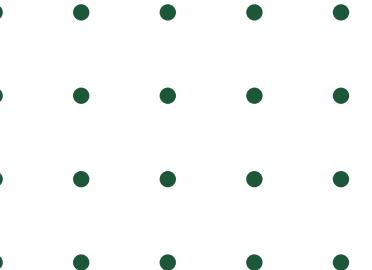
Sujitha1221 committed 18 minutes ago · ✘ 0 / 1

**Merge pull request #75 from Y3S1-GRP22/IT21068478/Dharane** · Verified · 2071e0a ·  · 

DharaneSegar authored 2 hours ago · ✓ 1 / 1

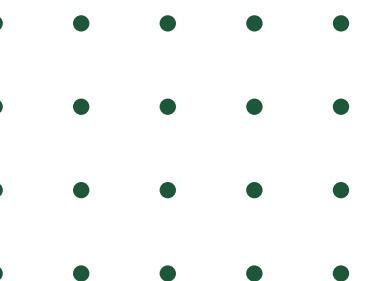


# Completion of the project





# Completion of the project



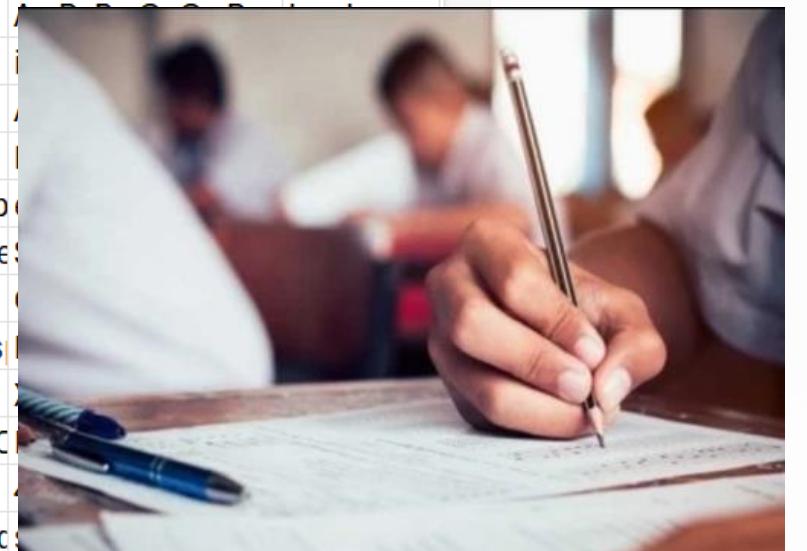
# Data Collection

AutoSave Off H Search merged\_mcq\_dataset SS File Home Insert Page Layout Formulas Data Review View Automate Help Acrobat A12 Which of the following statements regarding vascular tissues of plants is correct?

	A	B	C	D	E	F	G	H
1	Question Text	Option 1	Option 2	Option 3	Option 4	Option 5	Correct Answer	Difficulty L
2	A feature common to lysosomes and peroxisomes is?	They are single membr	They transport resid	They contain oxidisir	They are important 1	They digest worn ou	They are single me	easy
3	Two characteristics that can be seen only in living organisms are?	adaptation and growth	movement and irrita	change with time an	metabolism and here	synthesis and deco	metabolism and he	easy
4	Which of the following statements is correct regarding transmission electro	Specimens are magnif	Less electrons may	Living specimens ca	Three-dimensional a	Specimens scatter	Living specimens c	hard
5	Which response which correctly indicates the event and phase in the eukary	DNA replication - G0 p	Synthesis of protein	Chromatin formation	Production of cellula	Duplication of cent	Synthesis of protei	medium
6	In allosteric regulation of enzymes, which of the following statements is cor	regulatory molecules	regulatory molecule	an activator molecu	inhibitory molecules	ATP functions as	a regulatory molecu	medium
7	In ethyl alcohol fermentation, which of the following statements is correct?	one molecule of gluco	pyruvate is reduced	one molecule of CO <sub>2</sub>	final hydrogen accep	two molecules of A	two molecules of A	easy
8	Which of the following statements regarding glycosis of one molecule of glu	There is a net yield of	Two hydrogen ions a	It partially depends	Two NADH molecule	Part of glycolysis ta	Two NADH molecu	medium
9	Some events that took place during the evolution of organisms are as follow	A, B, C and D.	C, A, B and D.	C, B, A and D.	D, A, B and C.	D, B, A and C.	D, B, A and C.	hard
10	Which of the following pairs of organisms have the highest number of comm	Bat and crow.	Lizard and turtle.	Ichthyophis and Tae	Ulva and Pogonatum	Pinus and Cycas.	Lizard and turtle.	easy
11	Which of the following are unique characteristics of some phyla of the king	A and C only.	A and D only.	B and C only.	B and D only.	C and D only.	B and C only.	medium
12	Which of the following statements regarding vascular tissues of plants is co	Xylem tissues of ptero	Xylem vessel eleme	Tracheids provide su	Companion cells are	Pits are present be	Xylem tissues of pt	easy
13	Some structures of plants and their functions are shown below: A - Lenticels	A - P, B - R, C - Q.	A - R, B - P, C - P.	A - P, B - Q, C - R.	A - Q, B - P, C - P.	A - R, B - Q, C - R.	A - R, B - Q, C - R.	medium
14	Transport of water molecules due to physical adsorption by hydrophilic mate	imbibition.	osmosis.	facilitated diffusion.	bulk flow.	mass flow.	mass flow.	medium
15	Some steps in the process of opening and closing of stomata are given below	A, B, C, D, E and F.	A, C, B, D, E and F.	A, C, D, B, E and F.	A, E, B, D, C and F.	A, E, C, D, B and F.	A, E, C, D, B and F.	medium
16	A macronutrient and respectively a micronutrient which cause chlorosis in	Mg and Mn.	Fe and Ni.	P and Mo.	N and S.	Cu and B.	Cu and B.	medium
17	Two plant hormones that promote root formation are?	auxin and gibberellins.	cytokinins and absc	ethylene and auxin.	ethylene and gibber	cytokinins and gibbe	cytokinins and gibbe	medium
18	Which of the following statements regarding epithelia is correct?	Stratified squamous epi	Pseudostratified col	Simple columnar epi	Simple cuboidal epit	Simple squamous epi	Simple squamous epi	medium
19	Which of these combinations of the three types of symbiosis seen among or	A only.	B only.	C only.	A and B only.	A and C only	A and C only	medium
20	Which is the correct route of blood through the human heart from systemic	Left atrium, bicuspid v	Right atrium, tricuspid	Left atrium, tricuspid	Left ventricle, bicuspi	Right atrium, bicus	Right atrium, bicus	medium
21	Select the pair/pairs where an increase in (i) causes an increase in (ii). X - (i	X only.	Y only.	Z only.	X and Y only.	X and Z only.	X and Z only.	medium
22	Which of the following indicates the forms that transport the lowest and hig	Dissolved CO <sub>2</sub> - Carba	HCO <sub>3</sub> <sup>-</sup> - Carbamino	Carbaminohemoglo	HCO <sub>3</sub> <sup>-</sup> - Dissolved C	Dissolved CO <sub>2</sub> - HC	Dissolved CO <sub>2</sub> - HC	medium
23	If the tidal volume, residual volume, inspiratory reserve volume and expirato	1600 ml.	1700 ml.	3600 ml.	4700 ml.	5200 ml.	5200 ml.	medium
24	Which of the following is part of the parasympathetic division of the autonon	inhibits saliva secretio	dilates the pupil of e	relaxes bronchi in lu	stimulates the releas	stimulates gall bla	stimulates gall bla	medium
25	Which of the following statements regarding human vision is correct?	Changing refractory po	Convergence occurs	Accommodation is i	Photopsin in rods pr	Correct perception	Accommodation is hard	medium

merged\_mcq\_dataset +

Ready Accessibility: Unavailable 100%

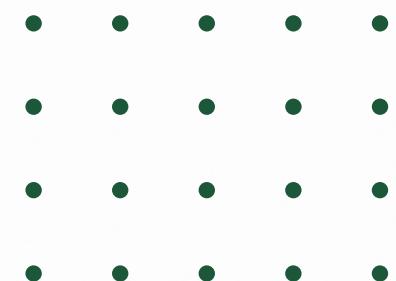
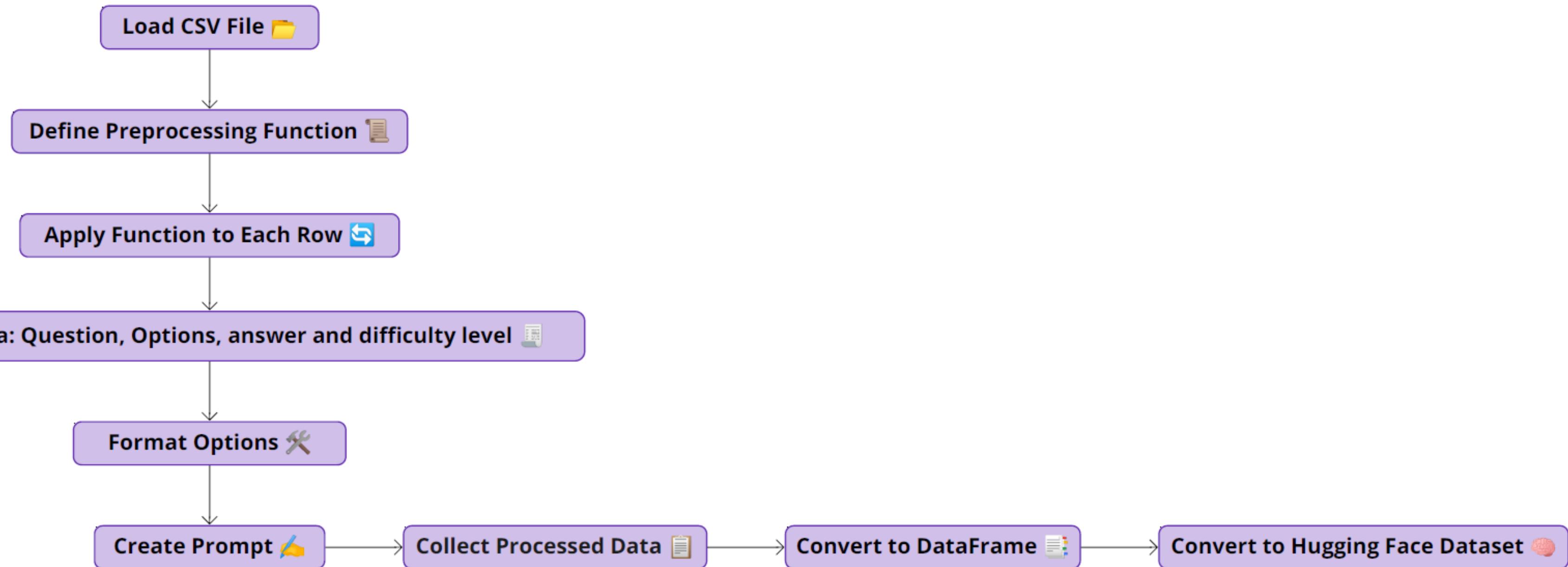


## Why llama-2-7b-chat-hf?

- Pretrained on Diverse Data
- Fine-Tuning Capability
- Natural Language Understanding
- Customization for Domain-Specific Terminology
- Cost-Effective and Accessible
- Scalability and Efficiency
- Contextual Responses



# Data pre-processing



# Data pre-processing

```
# Load and preprocess your CSV dataset
df = pd.read_csv(file_path, encoding='ISO-8859-1')

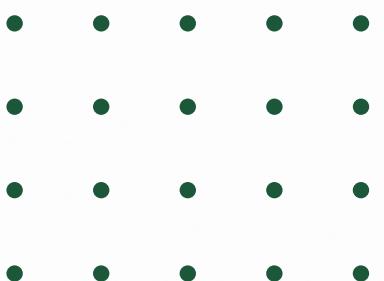
# Define a preprocessing function to format the questions, options, and difficulty
def preprocess_data(row):
    question = row["Question Text"]
    options = [row[f"Option {i}"] for i in range(1, 6) if pd.notna(row[f"Option {i}"])]
    correct_answer = row["Correct Answer"]
    difficulty = row["Difficulty Level"]

    # Format question prompt for training
    options_text = " ".join([f"{chr(65+i)}. {opt.strip()}" for i, opt in enumerate(options)])
    prompt = f"Question ({difficulty}): {question}\nOptions: {options_text}\nChoose the correct answer:"

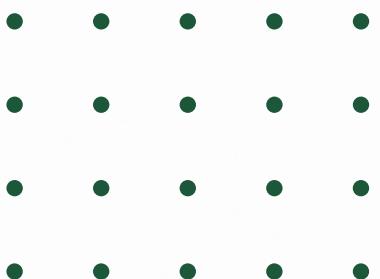
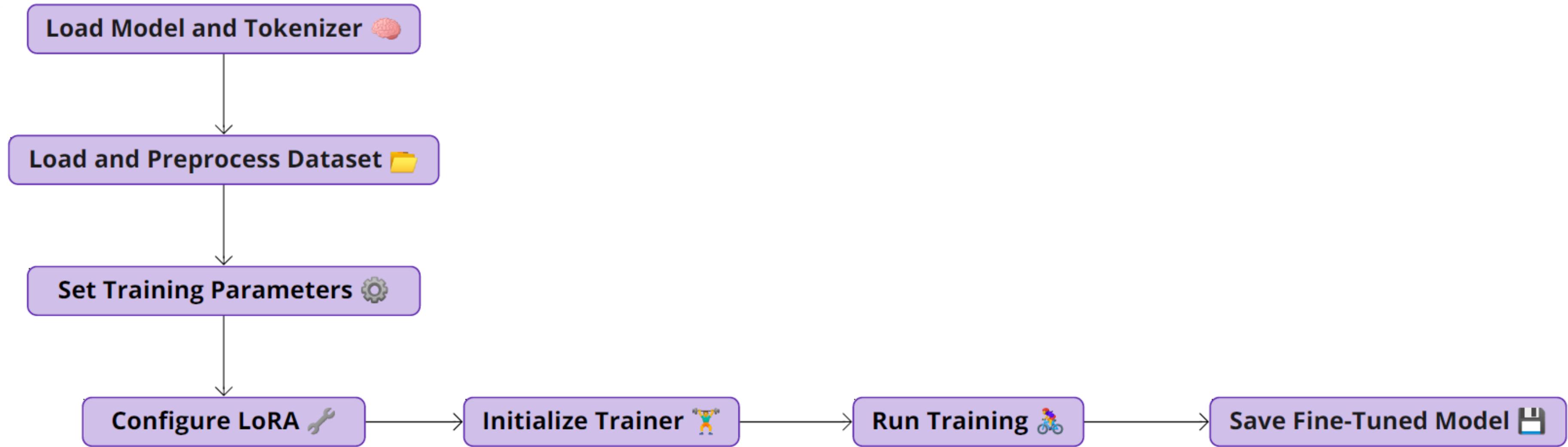
    return {
        "text": prompt,
        "label": correct_answer
    }

# Apply preprocessing to the dataset and convert to DataFrame
processed_data = df.apply(preprocess_data, axis=1).tolist()
processed_df = pd.DataFrame(processed_data)

# Convert the preprocessed DataFrame into a Hugging Face Dataset format
dataset = Dataset.from_pandas(processed_df)
```



# Fine-tuning the LLM



# Fine-tuning the LLM

```
# Configure bitsandbytes
compute_dtype = getattr(torch, bnb_4bit_compute_dtype)
bnb_config = BitsAndBytesConfig(
    load_in_4bit=use_4bit,
    bnb_4bit_quant_type=bnb_4bit_quant_type,
    bnb_4bit_compute_dtype=compute_dtype,
    bnb_4bit_use_double_quant=use_nested_quant,
)

# GPU compatibility check for bfloat16
if compute_dtype == torch.float16 and use_4bit:
    major, _ = torch.cuda.get_device_capability()
    if major >= 8:
        print("=" * 80)
        print("Your GPU supports bfloat16: accelerate training with bf16=True")
        print("=" * 80)

# Load model and tokenizer
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    quantization_config=bnb_config,
    device_map=device_map
)
model.config.use_cache = False
model.config.pretraining_tp = 1

# Load tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_name, trust_remote_code=True)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"

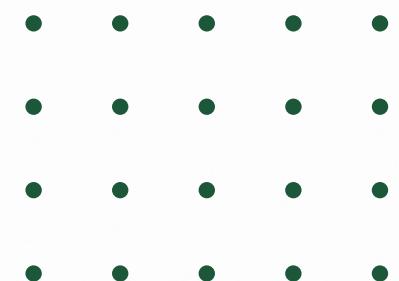
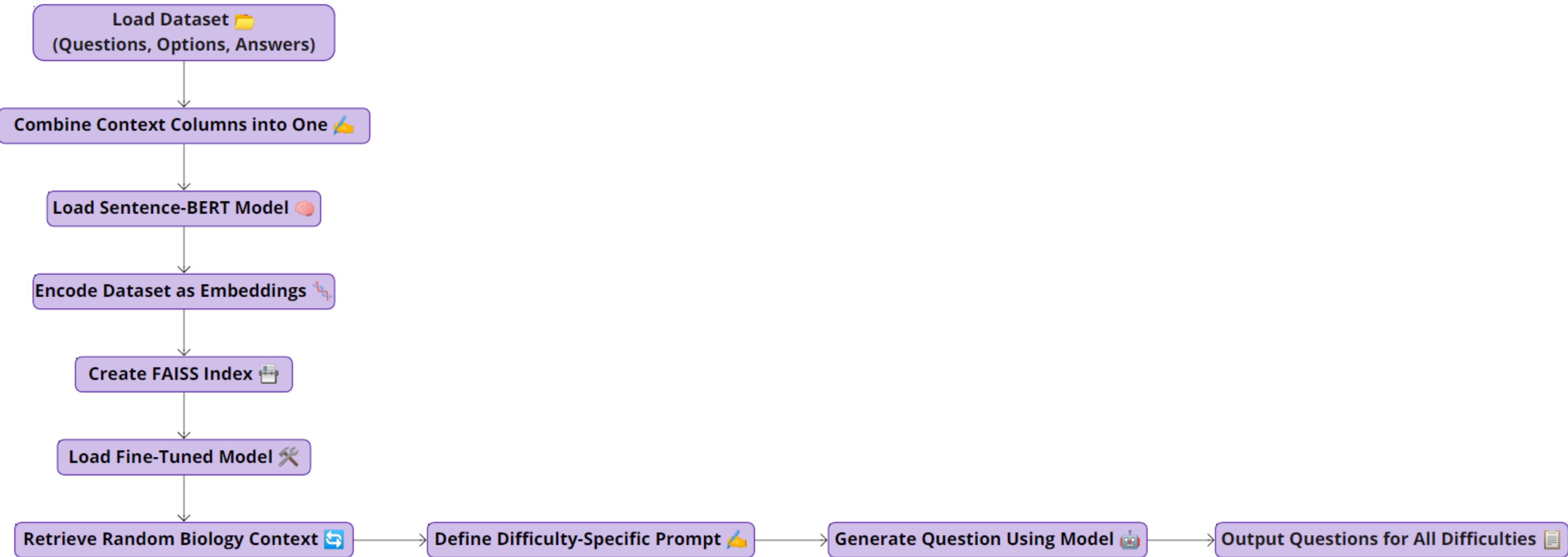
# LoRA configuration
peft_config = LoraConfig(
    lora_alpha=lora_alpha,
    lora_dropout=lora_dropout,
    r=lora_r,
    bias="none",
    task_type="CAUSAL_LM",
)
```

```
# Set training parameters
training_arguments = TrainingArguments(
    output_dir=output_dir,
    num_train_epochs=num_train_epochs,
    per_device_train_batch_size=per_device_train_batch_size,
    gradient_accumulation_steps=gradient_accumulation_steps,
    optim=optim,
    save_steps=save_steps,
    logging_steps=logging_steps,
    learning_rate=learning_rate,
    weight_decay=weight_decay,
    fp16=fp16,
    bf16=bf16,
    max_grad_norm=max_grad_norm,
    max_steps=max_steps,
    warmup_ratio=warmup_ratio,
    group_by_length=group_by_length,
    lr_scheduler_type=lr_scheduler_type,
    report_to="tensorboard"
)

# Initialize the trainer with the processed dataset and LoRA configuration
trainer = SFTTrainer(
    model=model,
    train_dataset=dataset,
    peft_config=peft_config,
    dataset_text_field="text",
    max_seq_length=None, # Or specify max sequence length if needed
    tokenizer=tokenizer,
    args=training_arguments,
    packing=False,
)

# Train the model
trainer.train()
```

# RAG Implementation



# RAG Implementation

```
import pandas as pd
import numpy as np
import faiss
from transformers import AutoModelForCausalLM, AutoTokenizer
from sentence_transformers import SentenceTransformer

# Load the dataset
file_path = "/content/drive/My Drive/MCQ Question Generation/merged_mcq_dataset.csv"
dataset = pd.read_csv(file_path, encoding="latin1")
dataset.fillna("", inplace=True)

# Combine all context columns into one
dataset['Combined'] = (
    dataset['Question Text'].astype(str) + " " +
    dataset['Option 1'].astype(str) + " " +
    dataset['Option 2'].astype(str) + " " +
    dataset['Option 3'].astype(str) + " " +
    dataset['Option 4'].astype(str) + " " +
    dataset['Option 5'].astype(str)
)

# Load Sentence-BERT for semantic search
semantic_model = SentenceTransformer('all-MiniLM-L6-v2')
embeddings = semantic_model.encode(dataset['Combined'].tolist(), show_progress_bar=True)

# Create a FAISS index
dimension = embeddings.shape[1]
faiss_index = faiss.IndexFlatL2(dimension)
faiss_index.add(np.array(embeddings))

# Load the fine-tuned model
model_name = "/content/drive/My Drive/MCQ Question Generation/saved_model" # Update this path
tokenizer = AutoTokenizer.from_pretrained(model_name, trust_remote_code=True)
generation_model = AutoModelForCausalLM.from_pretrained(model_name)

# Retrieve diverse biology contexts using FAISS
def retrieve_diverse_contexts_faiss(k=3):
    # Randomly select embeddings from the FAISS index
    random_indices = np.random.choice(len(dataset), k, replace=False)
    random_embeddings = np.array([embeddings[i] for i in random_indices])

    # Perform a FAISS search on these random embeddings
    _, indices = faiss_index.search(random_embeddings, k=1) # Retrieve nearest neighbor for each
    return dataset.iloc[indices.flatten()]['Combined'].tolist()
```

```
# Define the function to generate questions for all difficulties
def generate_biology_questions():
    difficulties = ["easy", "medium", "hard"]
    questions = {}

    for difficulty in difficulties:
        # Retrieve a random biology context
        retrieved_context = retrieve_context_biology(k=1)
        context = " ".join(retrieved_context)

        # Create a biology-specific prompt
        # Create a refined biology-specific prompt with difficulty-specific instructions
        prompt = f"""
        Based on the following biology context:
        {context}

        Your task is to generate a **biology multiple-choice question** for the **{difficulty} level**:
        - **For Easy Level**: The question should test basic biology concepts, simple definitions, or fundamental facts that are straightforward and easy to recall.
        - **For Medium Level**: The question should involve intermediate biology concepts, processes, or applications that require some reasoning or understanding of relationships between concepts.
        - **For Hard Level**: The question should test advanced biology concepts, detailed mechanisms, or require critical thinking and analysis of biological principles.

        - Provide exactly **five distinct answer options** labeled a, b, c, d, and e.
        - Only one answer option should be correct.
        - Clearly indicate the correct answer.

        Please output in the following format only:
        - Question: <Your question>
        - a) <Option 1>
        - b) <Option 2>
        - c) <Option 3>
        - d) <Option 4>
        - e) <Option 5>
        - Correct Answer: <Correct option letter>

        Do not include any additional text, explanations, or examples.
        """

        # Tokenize and generate
        input_ids = tokenizer.encode(prompt, return_tensors="pt", padding=True, truncation=True)
        output = generation_model.generate(input_ids, max_length=500, temperature=0.7, top_k=50, top_p=0.85)

        # Decode the response
        question = tokenizer.decode(output[0], skip_special_tokens=True)
        questions[difficulty] = question

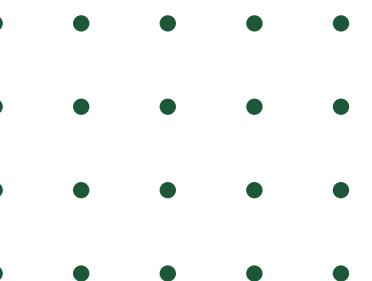
    return questions

# Example usage
questions = generate_biology_questions()

# Print the questions
for difficulty, question in questions.items():
    print(f"Difficulty: {difficulty.capitalize()}\n{question}\n")
```



# Completion of the project



# Key Functionalities

The back-end generates the first quiz with an equal difficulty distribution. Subsequent quizzes adjust difficulty dynamically based on student performance for personalized learning.

- **FAISS (Facebook AI Similarity Search)** – Retrieves context from the dataset for new question generation and detects duplicates, ensuring unique and diverse MCQs.
- **SentenceTransformer ("all-MiniLM-L6-v2")** – Converts questions and answers into vector embeddings for semantic comparison and retrieval of similar past questions.
- **Item Response Theory (IRT)** – Adjusts quiz difficulty dynamically based on student performance, ensuring adaptive learning.
- **Cosine Similarity** – Filters out duplicate or highly similar MCQs within the same quiz and across past quizzes.
- **Student Performance Tracking** – Records accuracy, response time, and consistency to monitor learning progress.

# adaptive\_quiz\_routes.py

```
1 import uuid
2 import logging
3 import time
4 from bson import ObjectId
5 from fastapi import APIRouter, HTTPException, Depends
6 from utils.user_mgmt_methods import get_current_user
7 from database.database import quizzes_collection, users_collection
8 from utils.quiz_generation_methods import fetch_questions_from_db, get_int_based_difficulty_distribution
9 from utils.generate_question import generate_mcq_based_on_performance
10 import traceback
11 import sys
12
13 router = APIRouter()
14
15 # Logging configuration
16 logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(levelname)s - %(message)s")
17
18 @router.get("/generate_adaptive_mcqs/{user_id}/{question_count}")
19 def generate_next_quiz(user_id: str, question_count: int, current_user: str = Depends(get_current_user)):
20     """Generate a new adaptive quiz based on user's previous performance."""
21     try:
22         logging.info(f"⌚ Starting adaptive quiz generation for user {user_id} with {question_count} questions.")
23         sys.stdout.flush() # Force log flushing
24
25         existing_user = users_collection.find_one({"_id": ObjectId(user_id)})
26         if not existing_user:
27             logging.error("User not found")
28             sys.stdout.flush()
29             raise HTTPException(status_code=404, detail="User not found. Please register before generating a quiz")
30
31         if current_user != user_id:
32             logging.error("Unauthorized access")
33             sys.stdout.flush()
34             raise HTTPException(status_code=403, detail="Unauthorized access")
35
36         logging.info("User found, fetching past performance...")
37         sys.stdout.flush()
38
39         difficulty_distribution = get_int_based_difficulty_distribution(user_id, question_count)
40         logging.info(f"📊 Difficulty distribution: {difficulty_distribution}")
41         sys.stdout.flush()
42
43         current_quiz_questions = set()
44         mcqs = []
45
46         for difficulty, count in difficulty_distribution.items():
47             generated = 0
48             failed_attempts = 0
49             while generated < count:
50                 logging.info(f"⚙️ Generating MCQ (Difficulty: {difficulty}) - Attempt {generated + 1}/{count}")
51                 sys.stdout.flush()
52
53                 mcq = generate_mcq_based_on_performance(user_id, difficulty)
```

```
logging.info(f"📝 Received MCQ response: {mcq}")
sys.stdout.flush()
if not mcq or not isinstance(mcq, dict) or "question" not in mcq:
    failed_attempts += 1
    logging.warning(f"⚠️ Failed MCQ generation for {difficulty}. Retrying... ({failed_attempts}/3)")
    sys.stdout.flush()
if failed_attempts >= 3:
    logging.error(f"⌚ Skipping {difficulty} MCQ after 3 failed attempts.")
    sys.stdout.flush()
    break # Move to the next difficulty level
continue # Retry another question
mcq_text = mcq["question"]
if mcq_text in current_quiz_questions:
    logging.warning(f"Duplicate detected: {mcq_text}, retrying...")
    failed_attempts += 1
    sys.stdout.flush()
    continue
formatted_mcq = {
    "question_text": mcq_text,
    "option1": mcq.get("options", {}).get("A", "N/A"),
    "option2": mcq.get("options", {}).get("B", "N/A"),
    "option3": mcq.get("options", {}).get("C", "N/A"),
    "option4": mcq.get("options", {}).get("D", "N/A"),
    "option5": mcq.get("options", {}).get("E", "N/A"),
    "correct_answer": mcq.get("correct_answer", "N/A"),
    "difficulty": difficulty
}
mcqs.append(formatted_mcq)
current_quiz_questions.add(mcq_text)
generated += 1
sys.stdout.flush()
if len(mcqs) < question_count:
    remaining_needed = question_count - len(mcqs)
    logging.warning(f"⚠️ Not enough questions generated. Fetching {remaining_needed} from DB.")
    sys.stdout.flush()
    db_questions = fetch_questions_from_db(remaining_needed)
    mcqs.extend(db_questions)
quiz_id = str(uuid.uuid4())
quiz_data = {
    "quiz_id": quiz_id,
    "user_id": user_id,
    "difficulty_distribution": difficulty_distribution,
```

# IRT Based difficulty Distribution

```
# Method to get IRT-based difficulty distribution for a user
def get_irt_based_difficulty_distribution(user_id, total_questions):
    """Dynamically adjusts quiz difficulty based on user performance trend and API success rate."""

    # • Fetch recent quiz performance (last 3 quizzes)
    recent_quizzes = list(quizzes_collection.find(
        {"user_id": ObjectId(user_id)},
        {"difficulty_distribution": 1, "questions": 1}
    ).sort("created_at", -1).limit(3))

    # Track how many correct answers per difficulty in recent quizzes
    correct_easy = correct_medium = correct_hard = 0
    total_easy = total_medium = total_hard = 1 # Avoid division by zero

    for quiz in recent_quizzes:
        for question in quiz["questions"]:
            difficulty = question.get("difficulty", "medium")
            is_correct = question.get("user_answered_correctly", False)

            if difficulty == "easy":
                total_easy += 1
                correct_easy += int(is_correct)
            elif difficulty == "medium":
                total_medium += 1
                correct_medium += int(is_correct)
            elif difficulty == "hard":
                total_hard += 1
                correct_hard += int(is_correct)

    # Compute accuracy per difficulty level
    easy_accuracy = correct_easy / total_easy
    medium_accuracy = correct_medium / total_medium
    hard_accuracy = correct_hard / total_hard

    # Base ratios on user's performance
    if easy_accuracy > 0.8: # User is doing well on easy
        easy_ratio = 0.1
    elif easy_accuracy < 0.5:
        easy_ratio = 0.4
    else:
        easy_ratio = 0.2
```

```
        if medium_accuracy > 0.7:
            medium_ratio = 0.5
        elif medium_accuracy < 0.4:
            medium_ratio = 0.3
        else:
            medium_ratio = 0.4

        if hard_accuracy > 0.6: # User is improving in hard
            hard_ratio = 0.4
        else:
            hard_ratio = 0.3 if hard_accuracy < 0.4 else 0.2 # Reduce hard if they are failing too much

    # Ensure total = 100%
    total_ratio = easy_ratio + medium_ratio + hard_ratio
    easy_ratio /= total_ratio
    medium_ratio /= total_ratio
    hard_ratio /= total_ratio

    return {
        "easy": round(easy_ratio * total_questions),
        "medium": round(medium_ratio * total_questions),
        "hard": total_questions - (round(easy_ratio * total_questions) + round(medium_ratio * total_questions))
    }
```

# Other Functionalities

**User Performance Dashboard:** Displays quiz history, accuracy trends, response times, and strongest/weakest topics.

**Unit-Based Quizzes:** Allows users to do unit based MCQ Questions  
Retake Adaptive Quizzes (Excluded from Performance Calculation)

**Retake Adaptive Quizzes (Excluded from Performance Calculation):** Users can retake adaptive quizzes to improve understanding without affecting their official performance metrics.

```
@router.get("/performance_graph/{user_id}")
def generate_performance_graph(user_id: str, current_user: str = Depends(get_current_user)):
    """Generates graphs showing user improvement and consistency."""
    user_data = users_collection.find_one({"_id": ObjectId(user_id)})
    if not user_data:
        raise HTTPException(status_code=404, detail="User not found.")

    if current_user != user_id:
        logging.error(f" Unauthorized access attempt by {current_user}")
        raise HTTPException(status_code=403, detail="Unauthorized access")

    if not user_data or "performance" not in user_data or "last_10_quizzes" not in user_data["performance"]:
        return {
            "quiz_numbers": [],
            "scores": [],
            "graph_image": None,
            "message": "No quiz performance data available. Start taking quizzes to see your progress!"
        }
    history = user_data["performance"]["last_10_quizzes"]
```

```
quiz_numbers = list(range(1, len(history) + 1))
scores = [quiz["accuracy"] for quiz in history]

plt.figure(figsize=(8, 4))
plt.plot(quiz_numbers, scores, marker="o", linestyle="-", color="b", label="Accuracy (%)")
plt.xlabel("Quiz Attempt")
plt.ylabel("Score (%)")
plt.title("User Performance Over Time")
plt.legend()

img = io.BytesIO()
plt.savefig(img, format="png")
img.seek(0)
return {
    "quiz_numbers": quiz_numbers,
    "scores": scores,
    "graph_image": base64.b64encode(img.getvalue()).decode(),
    "message": "Performance data loaded successfully."
}
```

# Working Product

The Bio Mentor homepage features a purple header with the logo and navigation links: Home, MCQ, Q & A, Vocabulary, Summarize, and a user icon. Below the header is a large question mark icon. The main section has a title "Master Your Knowledge with Adaptive Quizzes" and a sub-section "Take personalized quizzes that adjust based on your answers. Track your improvements, identify your strengths, and enhance your learning journey." It includes four buttons: ADAPTIVE LEARNING, REAL-TIME FEEDBACK, ACCURATE SCORING, and IMPROVEMENT ANALYSIS. A "Start Your Quiz" button is at the bottom. To the right is a cartoon character interacting with a control panel.

The Quiz Results section shows Attempt Number: 1, Correct Answers: 2 / 15, Accuracy: 13.33%, and Total Time: 2 min 16 sec. It lists categories: Easy: 5, Medium: 5, Hard: 5. Buttons for Download PDF and Download CSV are available. Below this, a question about the olfactory bulb is shown with five options: A. To detect light and dark, B. To process visual information, C. To process auditory information, D. To process chemical signals (selected), and E. To process touch and pressure sensations. A "Correct Answer!" message and a "What makes this the right answer?" button are present. Another question about bird migration patterns is shown with five options: A. The availability of food in the Northern and Southern Hemispheres, B. The presence of the Arctic Tundra in the Northern Hemisphere (selected), C. The migration of birds from the Southern Hemisphere to the Northern Hemisphere, D. The migration of birds from the Northern Hemisphere to the Southern Hemisphere, and E. The decrease in the number of Arctic Tundra in the Northern Hemisphere.

A quiz interface showing Question 1 of 15. The question asks: "Which of the following is an example of a symbiotic relationship between two organisms?". The options are: A. Oxidation of oxygen by mitochondria, B. Symbiotic relationship between coral and zooxanthellae, C. Photosynthesis by plants, D. Decomposition by bacteria, and E. Endosymbiosis by mitochondria. A timer shows 44:43 left. A "Questions" grid shows numbered boxes from 1 to 15. Navigation buttons "Previous" and "Next" are at the bottom.

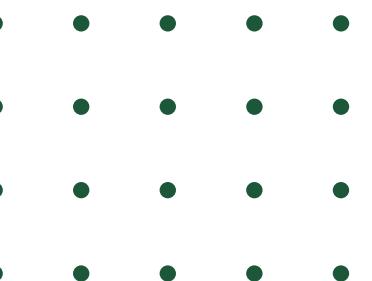
# Working Product

The dashboard features a header with the Bio Mentor logo and navigation links for Home, MCQ, Q & A, Vocabulary, Summarize, and Help. The main area includes a "User Performance Dashboard" section with a sub-section titled "Accuracy Ace". It displays statistics like Total Quizzes (4), Engagement Score (79%), and Consistency Score (79%). Below this is an "AI Insights" section with the message "Great job! Your accuracy is improving steadily. Keep practicing!". The "Leaderboard" section shows user scores for Melissa123 (You), Mathresha18, and sajeesiva06. The dashboard also contains three charts: "Performance Over Time" (line graph), "Comparison with Other Users" (bar chart), and "Time Spent by Difficulty" (bar chart).

This page allows users to review past quiz attempts. It includes sections for "Quiz 1", "Quiz 2", and "Quiz 3". Each section lists attempts with details like date, time, score, and total time, followed by a "View Results" button. A "Retry Quiz" button is available for each quiz. A note at the bottom states "Retry limit reached (max 3 attempts)".



# Completion of the project





# Final Enhancements After 90% Viva

- Increased total quiz questions for deeper evaluation
- Prompt generates max 3 MCQs to avoid redundancy in context
- Improved MCQ extraction to handle multiple questions & formats
- Converted model to .gguf Q8 for resource efficiency with minimal loss
- Added:
  - 1) Answer Explanation component
  - 2) Answer Verification mechanism

• • • •

• • • •

• • • •

• • • •

# Working Product

The Bio Mentor homepage features a navigation bar with links to Home, MCQ, Q & A, Vocabulary, and Summarize. A user profile icon is also present. The main section is titled "Get Explanation for Any Biology MCQ" with a sub-section "Understand why an MCQ answer is correct or incorrect. Generate explanations or verify a claimed answer instantly." It includes four buttons: SMART FEEDBACK, REAL-TIME REASONING, CLAIM CHECKING, and BIOLOGY CONTEXT. A large button labeled "Explain MCQ" is at the bottom. A central illustration shows two students in a classroom setting, one sitting at a desk with a laptop and globe, and another standing next to them. Below the main content is a footer with the text "Empowering Learning, One Click at a Time" and "Bio Mentor is an e-learning platform designed to help A/L Biology students in Sri Lanka master complex concepts with ease. It offers interactive tools like adaptive quizzes, digital flashcards, and" followed by a "Read More" link. A "QUICK LINKS" section includes links to MCQ and Q & A.

A modal window titled "MCQ Explanation" asks, "What would you like to do with your MCQ? Choose whether to generate a clear explanation or check if a chosen answer is correct." It contains two options: "Generate Explanation" (with a star icon) and "Check Answer & Explain" (with a checkmark icon). The background shows a teacher at a whiteboard and a student at a desk.

A modal window titled "Generate MCQ Explanation" shows "Input Mode" options: "Paste Full MCQ" (selected) and "Manual Input". Below is a text area with placeholder text: "Paste full MCQ like:  
What is the ...?  
A) Option  
B) Option  
C) Option  
D) Option". At the bottom are "Back" and "Generate Explanation" buttons.

A modal window titled "Verify & Explain MCQ" shows "Input Mode" options: "Paste Full MCQ" and "Manual Input" (selected). Below is a "Question:" field with five empty text areas for "Option A", "Option B", "Option C", "Option D", and "Option E". At the bottom are "Back" and "Verify & Explain" buttons.

# Testing - Unit Testing

Framework: Pytest

Purpose: Test individual functions in isolation using mocks.

Modules & Focus Areas:

1) test\_text\_parsing\_utils.py

- Tests MCQ extraction and answer cleaning logic

2) test\_generate\_mcq.py

- Tests MCQ generation with mocked model, FAISS, and embeddings

3) test\_irt.py

- Verifies IRT logic for difficulty and discrimination values

```
from bson import ObjectId

# Dynamically add the absolute path to project root (MCQ) to sys.path
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))

dummy_user_id = str(ObjectId())

from utils.quiz_generation_methods import (
    assign_difficulty_parameter,
    assign_discrimination_parameter,
    get_irt_basedDifficulty_distribution
)

def test_assign_difficulty_parameter_easy_range():
    value = assign_difficulty_parameter(dummy_user_id, "easy")
    assert -2.0 <= value <= 1.0

def test_assign_difficulty_parameter_hard_range():
    value = assign_difficulty_parameter(dummy_user_id, "hard")
    assert isinstance(value, float)

def test_assign_discrimination_parameter_range():
    value = assign_discrimination_parameter()
    assert 0.5 <= value <= 2.0

def test_get_irt_basedDifficulty_distribution_low_theta():
    distribution = get_irt_basedDifficulty_distribution(dummy_user_id, 10)
    assert sum(distribution.values()) == 10
    assert isinstance(distribution, dict)

def test_get_irt_basedDifficulty_distribution_high_theta():
    distribution = get_irt_basedDifficulty_distribution(dummy_user_id, 10)
    assert sum(distribution.values()) == 10
    assert isinstance(distribution, dict)
```

```
def test_extract_mcqs_parsing_single_question():
    prompt = "Ignore this prompt."
    raw_output = """
    Question 1: What is the powerhouse of the cell?
    A) Nucleus
    B) Ribosome
    C) Mitochondria
    D) Chloroplast
    E) Endoplasmic Reticulum
    Correct Answer: C
    """
    parsed = extract_mcqs(prompt, raw_output)
    assert len(parsed) == 1
    assert parsed[0][ "question" ].startswith("What is")
    assert parsed[0][ "correct_answer" ] == "C"
    assert "A" in parsed[0][ "options" ]
    assert len(parsed[0][ "options" ]) == 5

@pytest.mark.parametrize("raw_input, expected", [
    ("Correct Answer: A", ["A"]),
    ("Answer: C and D", ["C", "D"]),
    ("B, D", ["B", "D"]),
    ("(E)", ["E"]),
    ("A, A, C", ["A", "C"]),
    ("123", []),
    ("", []),
    ("Correct Answer: A, C, E", ["A", "C", "E"]),
])
def test_clean_correct_answer_letters(raw_input, expected):
    assert clean_correct_answer_letters(raw_input) == expected
```

# Testing - Integration Testing

Frameworks: Pytest, FastAPI TestClient

Purpose: Validate full workflows across routes, DB, and logic layers.

Modules & Focus Areas:

## 1) test\_adaptive\_routes.py

- End-to-end adaptive quiz generation and submission
- Endpoints: /generate\_adaptive\_mcqs, /submit\_quiz, /get\_quiz

## 2) test\_routes.py

- Unit quiz generation and performance routes
- Endpoints: /unit\_quiz/generate, /unit\_quiz/submit, /dashboard\_data

```
import pytest
from fastapi.testclient import TestClient
from main import app
from database.database import users_collection, quizzes_collection
from bson import ObjectId
from utils.user_mgmt_methods import get_current_user

TEST_USER_ID = "000000000000000000000003"

def override_get_current_user():
    return TEST_USER_ID

app.dependency_overrides[get_current_user] = override_get_current_user
client = TestClient(app)

@pytest.fixture(scope="module", autouse=True)
def setup_user():
    users_collection.insert_one({
        "_id": ObjectId(TEST_USER_ID),
        "username": "test_user",
        "performance": {
            "total_quizzes": 1,
            "accuracy_easy": 80,
            "accuracy_medium": 70,
            "accuracy_hard": 60,
            "time_easy": 5,
            "time_medium": 5,
            "time_hard": 5,
            "strongest_area": "easy",
            "weakest_area": "hard",
            "consistency_score": 90,
            "last_10_quizzes": [
                {
                    "accuracy": 75,
                    "total_time": 45,
                    "timestamp": 1712724584.0
                }
            ]
        }
    })
    yield
    users_collection.delete_many({"_id": ObjectId(TEST_USER_ID)})

def test_generate_mcqs():
    response = client.get(f"/generate_mcqs/{TEST_USER_ID}")
    assert response.status_code in [200, 404]
    if response.status_code == 200:
        data = response.json()
        assert "quiz_id" in data
        assert "mcqs" in data

def test_generate_adaptive_mcqs():
    response =
    client.get(f"/generate_adaptive_mcqs/{TEST_USER_ID}/5")
    if response.status_code == 200:
        data = response.json()
        assert "quiz_id" in data
        assert "mcqs" in data

def test_submit_quiz_and_get_quiz():
    # First generate quiz
    gen = client.get(f"/generate_mcqs/{TEST_USER_ID}")
    if gen.status_code != 200:
        pytest.skip("MCQ quiz not available for submission test")

    quiz_data = gen.json()
    quiz_id = quiz_data["quiz_id"]
    questions = quiz_data["mcqs"]

    # Submit fake answers
    submission_payload = {
        "user_id": TEST_USER_ID,
        "quiz_id": quiz_id,
        "responses": [
            {
                "question_text": q["question_text"],
                "selected_answer": "A",
                "time_taken": 3.0
            } for q in questions
        ]
    }
    submit = client.post("/submit_quiz/", json=submission_payload)
    assert submit.status_code == 200
    assert "summary" in submit.json()

    # Fetch quiz using the same quiz_id
    fetch = client.get(f"/get_quiz/{quiz_id}")
    assert fetch.status_code == 200
    assert fetch.json()["quiz_id"] == quiz_id
```

# Testing - System Testing and UAT

## System Testing

- Framework: Postman

HTTP / http://127.0.0.1:8000/responses/user\_quiz\_history/67d2aeea8191e9da8aeb04e3

GET http://127.0.0.1:8003/responses/user\_quiz\_history/67d9b4071f535025af7d04f0

Params Authorization Headers (9) Body Scripts Settings Cookies

Key	Value	Description	Bulk Edit	Presets
Authorization	Bearer eyJhbGciOiJIUzI1NlslR5cCl6lkp...			

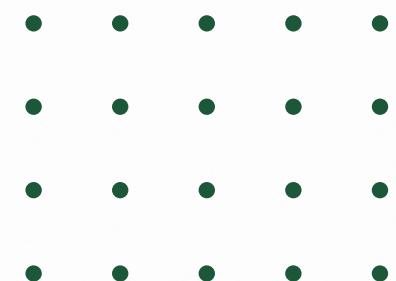
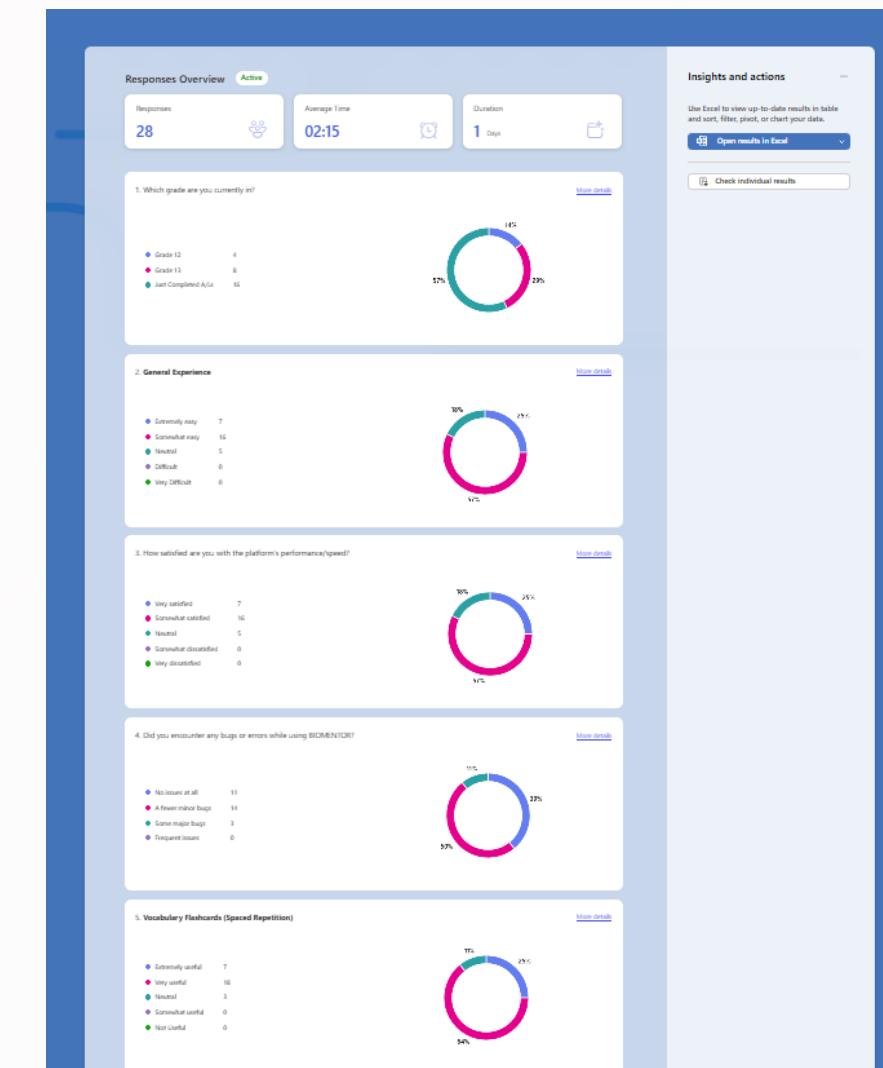
Body Cookies (1) Headers (4) Test Results 200 OK 2.52 s 2.04 KB Save Response

```
{ } JSON > Preview < Visualize <
```

```
1 {
2   "quiz_history": [
3     {
4       "quiz_id": "d9a2374d-994d-4741-a40c-c5367c4692a4",
5       "attempts": [
6         {
7           "response_id": "67d9b577d1e6aed6ba1a39d2",
8           "submitted_at": 1742321015.1480622,
9           "attempt_number": 1,
10          "summary": {
11            "total_questions": 15,
12            "correct_answers": 2,
13            "incorrect_answers": 13,
14            "accuracy": 13.33,
15            "total_time": 136.163,
16          }
17        }
18      ]
19    }
20  ]
21}
```

Postbot Runner Start Proxy Cookies Vault Trash

## User Acceptance Testing



# Deployment

- The MCQ generation backend was deployed on an Azure Virtual Machine using FastAPI and a quantized LLaMA 2 model (.gguf format).
- The backend runs as a persistent systemd service, with Nginx configured as a reverse proxy for public access.
- GitHub Actions CI/CD was integrated to automate deployment on code push.

The screenshot shows the Microsoft Azure portal interface for a virtual machine named 'biomentor-vm'. The 'Overview' tab is selected, displaying basic information such as Resource group (biomentor-rg), Status (Stopped (deallocated)), Location (Central India (Zone 1)), and Operating system (Linux). It also shows the Public IP address (20.244.49.225) and Virtual network/subnet (biomentor-vm-vnet/default). The 'Networking' tab is visible at the bottom.

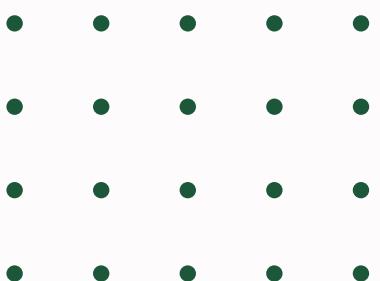
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\shobi> scp -i "C:\Users\shobi\OneDrive\Desktop\Deployment\biomentor_ci" "C:\Users\shobi\OneDrive\Desktop\quantized model\llama2-q8_0.gguf" azureuser@20.244.49.225:/home/azureuser/BioMentor-Personalized-E-Learning-Platform/Back-End/MCQ/model/
100% 1260MB 629.0KB/s 34:11
PS C:\Users\shobi>
```

# References

- [1] Automatic Question Answer Generation using T5 and NLP. [ONLINE] <https://ieeexplore.ieee.org/document/9972639> [Accessed 13 May 2024]
- [2] Generation of Multiple-Choice Questions From Textbook Contents of School-Level Subjects. [ONLINE] <https://ieeexplore.ieee.org/document/9964056> [Accessed 20 May 2024]
- [3] Automatic question generation for intelligent tutoring systems. [ONLINE] <https://ieeexplore.ieee.org/document/9972639> [Accessed 06 June 2024]
- [4] MCQGen: A Large Language Model-Driven MCQ Generator for Personalized Learning. [ONLINE] <https://ieeexplore.ieee.org/document/9972639> [Accessed 05 June 2024]
- [5] Generation of Multiple Choice Questions from Indian Educational Text. [ONLINE] <https://ieeexplore.ieee.org/document/10270551> [Accessed 16 June 2024]



# **IT21204302**

Sajeevan S

Software Engineering

**LLM BASED PROVIDE ANSWERS  
FOR STRUCTURED AND ESSAY  
TYPE OF QUESTIONS AND  
EVALUATE ANSWERS BASED ON  
APPROVED RESOURCES.**



# Introduction

01

Background

02

Research Problem

03

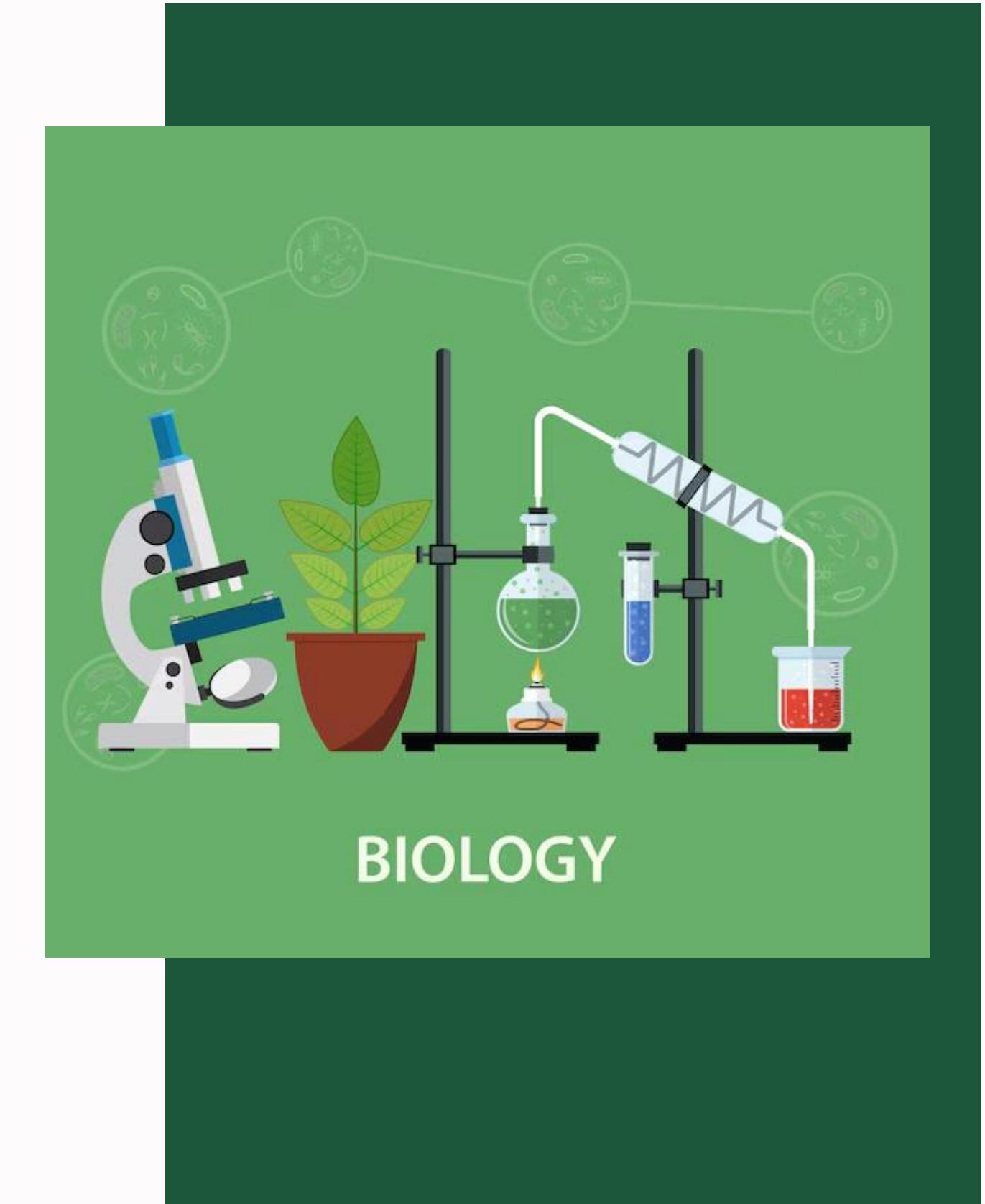
Main and Sub Objectives

04

Methodology

0  
5

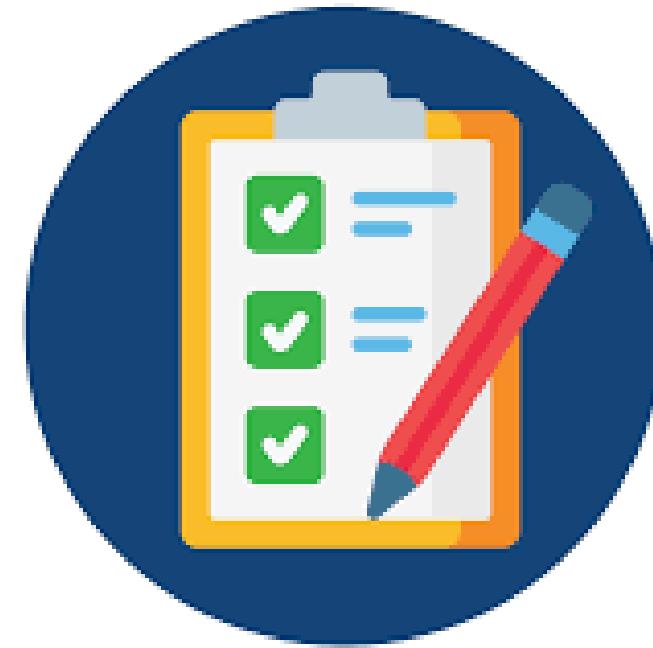
Completion Of Project



# BACKGROUND



The Importance of  
Independent Learning  
in Modern Education

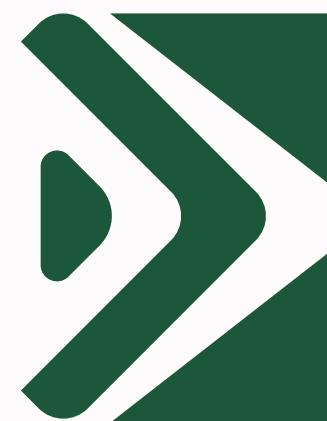


## Self Assessment

The Challenges of Self-Evaluation in structured essay-type answering

# RESEARCH PROBLEM

01



How can technology-based tools be developed and utilized to assist students in improving their essay-writing skills independently, without relying on mentor support?

What strategies can students employ to self-evaluate and improve their structured essay-type answers in the absence of mentor guidance?

02



ESQONMILWAUKEE JOURNAL SENTINEL - USA TODAY NETWORK

# OBJECTIVES

Objective 1

Answer Generation for a Question

Objective 3

Provide Suggestions for Improvement

Objective 2

Enhance Self-Directed Learning in Advanced-Level Biology

Objective 4

Provide Feedback for Answers

## Main Objective

If students provide a question, the system will generate an answer, and if students also provide their corresponding answer, the system will evaluate their response and offer suggestions to improve the answer.

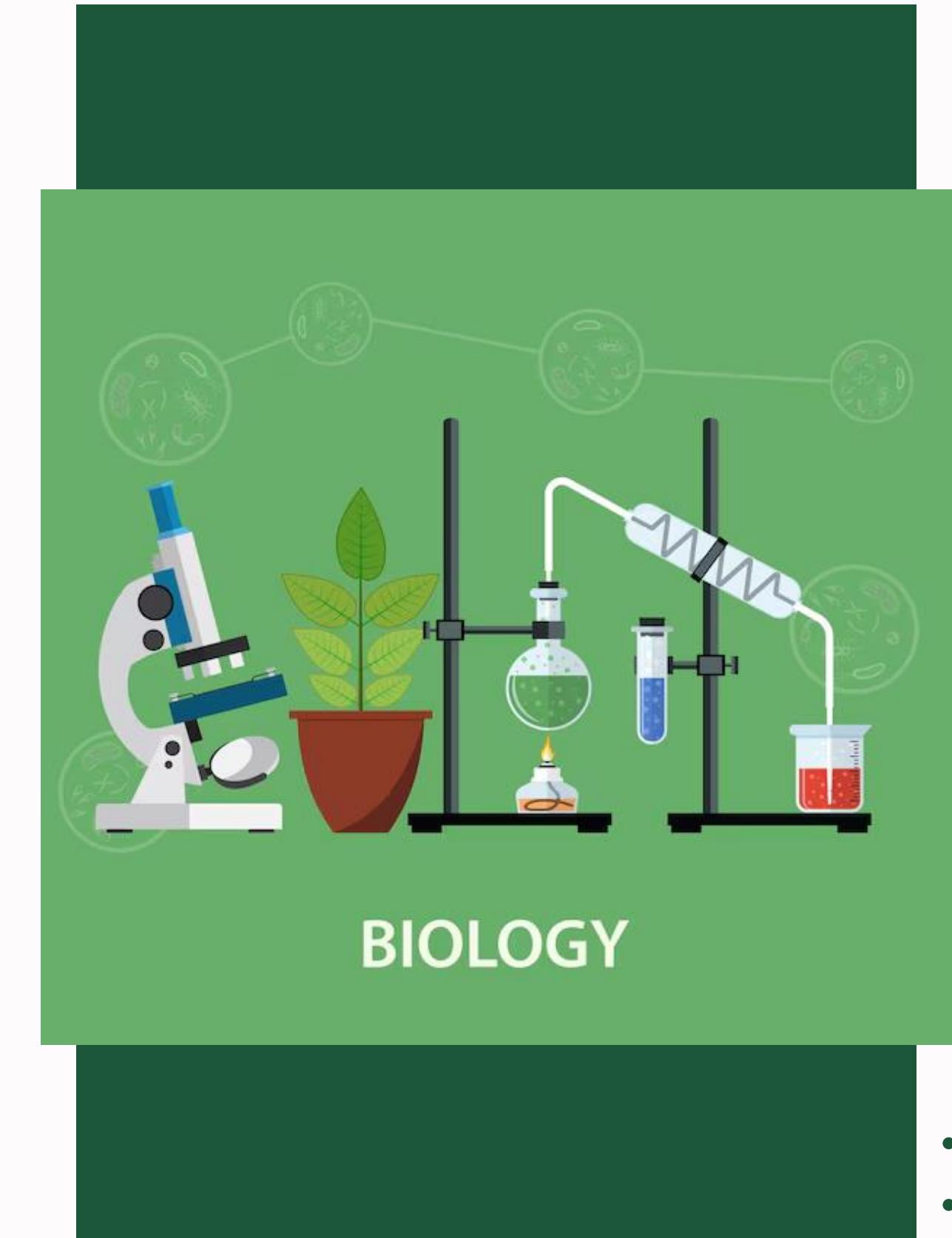
# Methodology

01 System Diagram

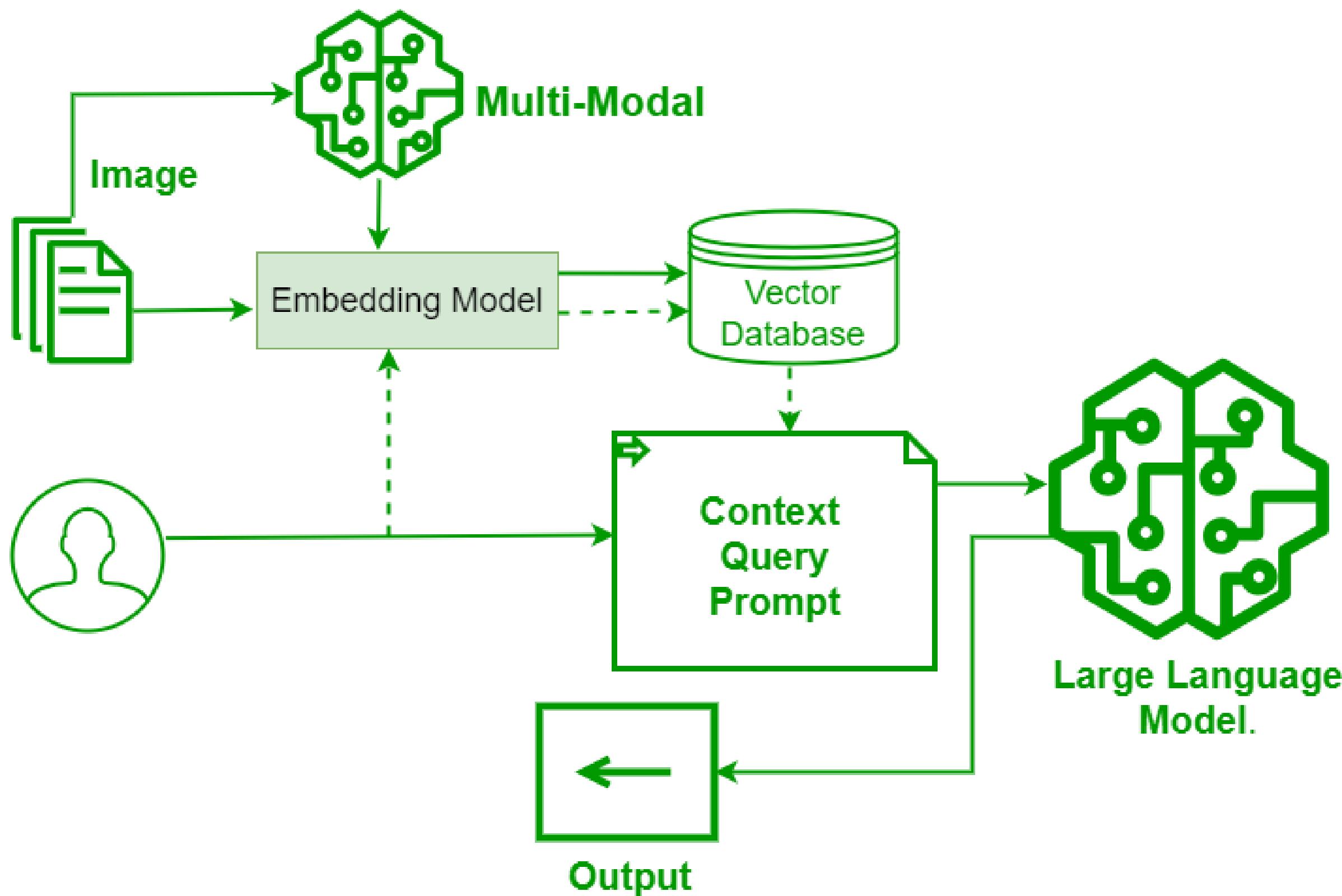
02 Tools and Technologies

03 Best Practices

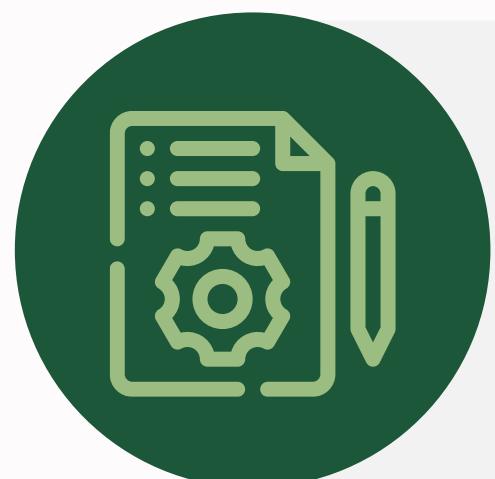
04 GitHub Commits



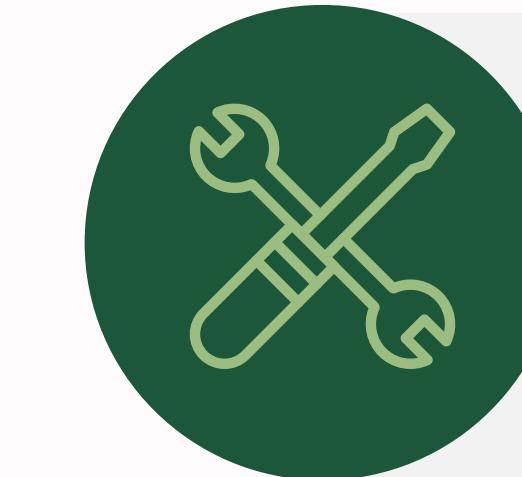
# System Diagram



# Tools & Technologies



**Project Management**  
Jira



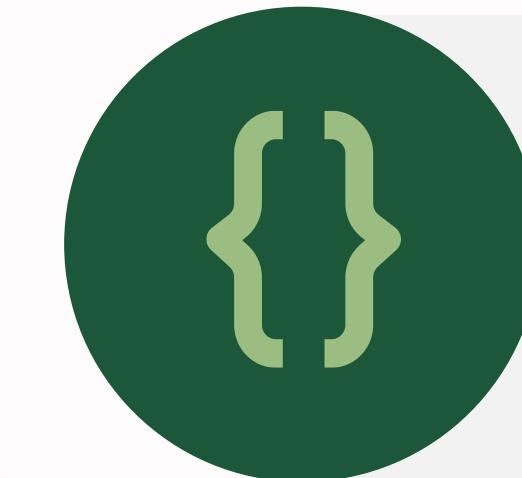
**Other tools**  
Git  
Draw.io  
Postman



**Database**  
Faiss  
Mongo DB



**Frameworks**  
Transformer model  
Flask  
Pytorch  
OCR



**Programming Languages**  
Python  
React JS

# Best Practices

## Non-Functional Requirements

- Compatibility
  - Accuracy
- Performance
  - Usability
- Availability

## Design Excellence

- Real-Time Performance
- Accuracy and Reliability
- User Experience
  - Integration

## Standards & Best Practices

- Code Structure & Organization
- Web Security
- Model Quality
- RESTful API Design

# GitHub Commits

 BioMentor-Personalized-E-Learning-Platform Public

[Edit Pins](#) [Watch 0](#) [Fork 0](#) [Star 0](#)

[IT21204302/Sajeevan](#) [8 Branches](#) [0 Tags](#) [Go to file](#) [Add file](#) [Code](#)

This branch is up to date with [main](#).

[Contribute](#)

Commit	Author	Message	Date
fixed generate pdf function	DharaneSegar	Merge pull request #10 from Y3S1-GRP22/IT21068478/Dharane	b0ce5b4 · 18 hours ago
Minor fixes			last week
Add system diagram in README.md			4 months ago
Updated folder structure			last month
Add system diagram in README.md			4 months ago

[Back-End](#) [Front-End](#) [Images](#) [Model-Training](#) [Readme.md](#)

[README](#)

[Report repository](#)

**Releases**  
No releases published [Create a new release](#)

**Packages**  
No packages published

**About**

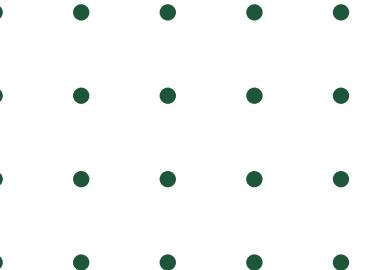
Final year research project of group 24-25J-257

[Readme](#) [Activity](#) [Custom properties](#) [0 stars](#) [0 watching](#) [0 forks](#)

**BioMentor - Personalized E-Learning Platform for A/I**

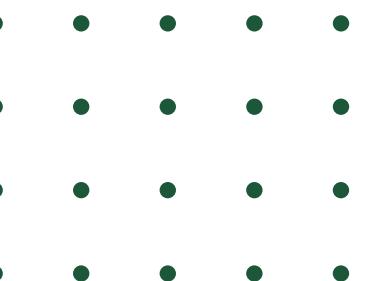


# Completion of the project





# Completion of the project



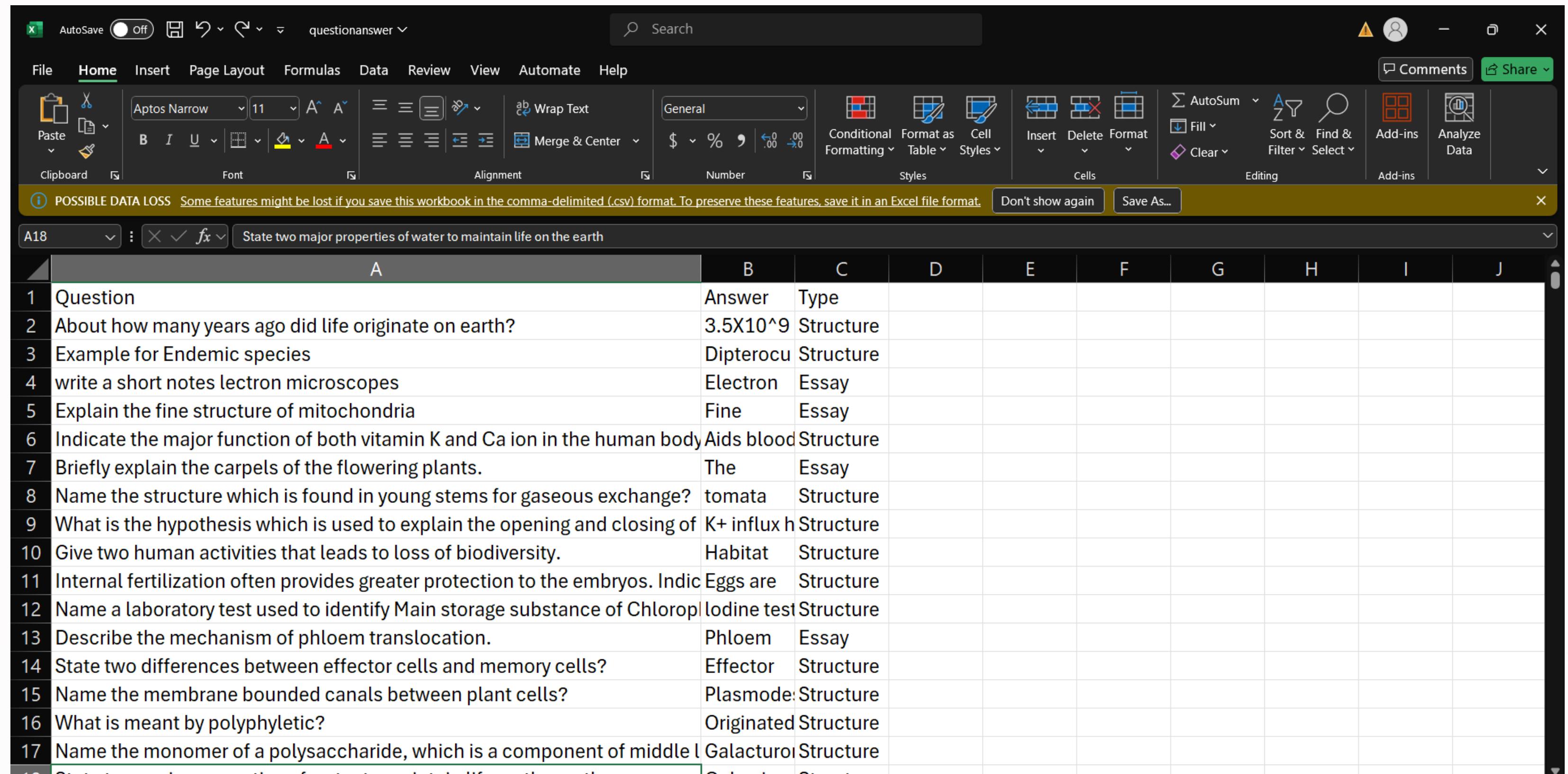
# Data Collection

The image displays two Microsoft Word documents side-by-side, illustrating the process of data collection and organization.

**StructureDataset:** This document contains a list of questions and their corresponding answers. The questions are numbered 4 through 23. The answers include various biological concepts such as disease-resistant varieties, postharvest technologies, human population increase, geological eras, classification of organisms, modern systematics, arthropod features, mammalian unique features, physiological commonalities, seedless plant phyla, microphyll features, vascular tissue, cell wall composition, protein roles, polysaccharide monomers, cell cycle events, and photosynthesis pathways. The document is formatted with a table structure where each question is in column A and its answer is in column B.

**EssayDataset:** This document contains a single question in column A and its detailed answer in column B. The question asks to describe the components of nucleotides and explain how they form the backbone of DNA. The answer provides a comprehensive explanation of nucleotide components (pentose sugar, nitrogenous base, phosphate group), nucleic acid types (DNA vs RNA), nucleotide linkage, phosphodiester bonds, DNA double helix structure, and the complementary base pairing rule (A-T, G-C).

# Data Collection

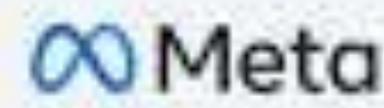


A screenshot of a Microsoft Excel spreadsheet titled "questionanswer". The spreadsheet contains 17 rows of data, each consisting of a question in column A, an answer in column B, and a type in column C. The columns are labeled A through J at the top. The "Home" tab is selected in the ribbon. A status bar at the bottom indicates "Data Collection" and shows the formula `=IF(LEN(A1)=0, "Question", "Answer")`.

	A	B	C	D	E	F	G	H	I	J
1	Question		Answer	Type						
2	About how many years ago did life originate on earth?	3.5X10^9	Structure							
3	Example for Endemic species	Dipterocu	Structure							
4	write a short notes lectron microscopes	Electron	Essay							
5	Explain the fine structure of mitochondria	Fine	Essay							
6	Indicate the major function of both vitamin K and Ca ion in the human body	Aids blood	Structure							
7	Briefly explain the carpels of the flowering plants.	The	Essay							
8	Name the structure which is found in young stems for gaseous exchange?	tomata	Structure							
9	What is the hypothesis which is used to explain the opening and closing of	K+ influx h	Structure							
10	Give two human activities that leads to loss of biodiversity.	Habitat	Structure							
11	Internal fertilization often provides greater protection to the embryos. Indic	Eggs are	Structure							
12	Name a laboratory test used to identify Main storage substance of Chlorop	Iodine test	Structure							
13	Describe the mechanism of phloem translocation.	Phloem	Essay							
14	State two differences between effector cells and memory cells?	Effector	Structure							
15	Name the membrane bounded canals between plant cells?	Plasmode	Structure							
16	What is meant by polyphyletic?	Originated	Structure							
17	Name the monomer of a polysaccharide, which is a component of middle l	Galacturo	Structure							

# Model Selection

## Why Llama 3?



An instruct fine-tuned version of the 8B model that is optimized for specific tasks. For instance, it can be used to create educational tools that explain complex subjects.

Llama 3 has a greater capacity to learn nuanced patterns and adapt to specific tasks compared to other models.

Llama 3 is highly customizable, making it ideal for domain-specific tasks like academic content generation and adaptive learning applications.

Llama 3 generates contextually accurate, meaningful responses while avoiding repetition and irrelevant content.

# Model Selection

## Issues with Alternatives

### 1.T5 Small:

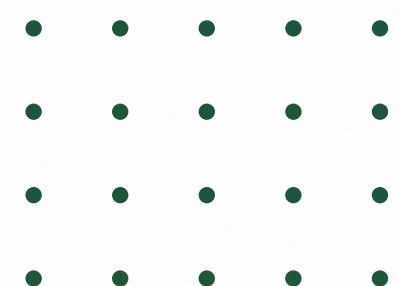
- Lacks clarity in answers due to its limited model size.

### 2.unsloth\_gemma\_2b\_bnb\_4bit:

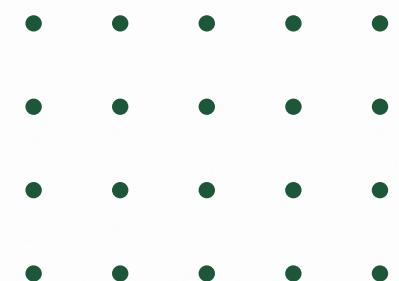
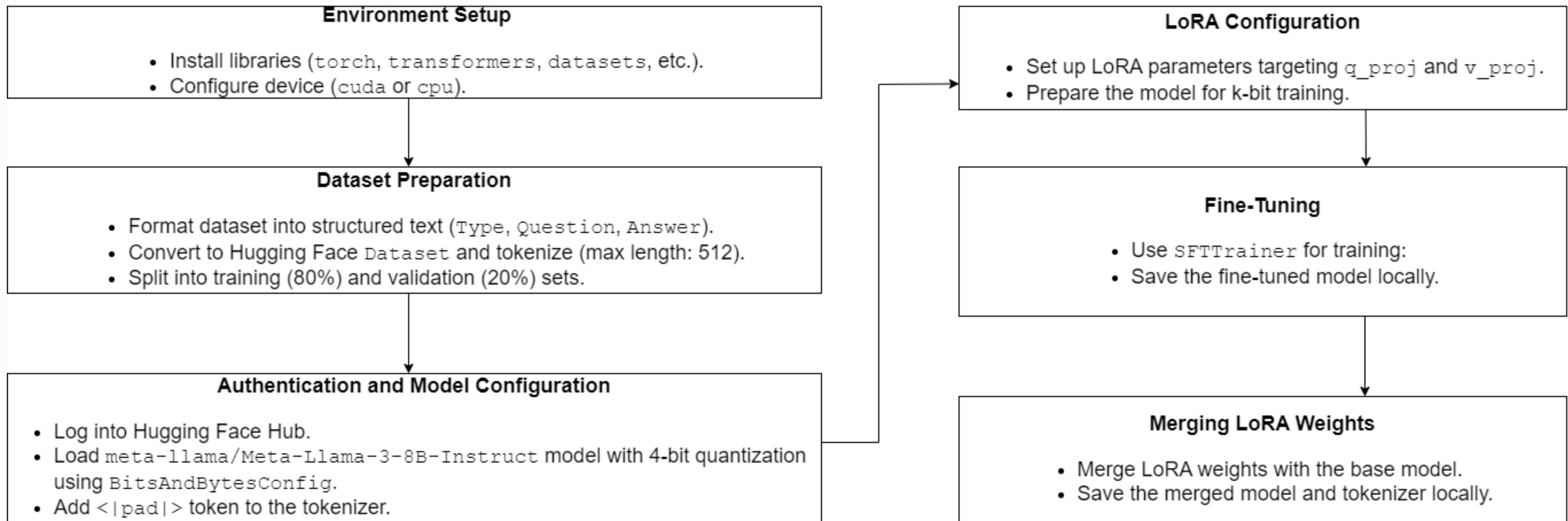
- Produces similar responses for different question types, indicating poor adaptability.
- Lacks diversity in output.

### 3.unsloth\_mistral\_7b:

- Includes irrelevant URLs and citations in essay-type answers.

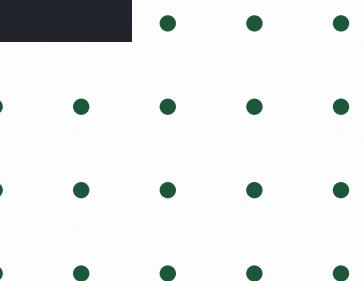


# Fine-tuning the LLM



# Fine-tuning the LLM

```
lora_config = LoraConfig(  
    r=16,  
    lora_alpha=32,  
    lora_dropout=0.1,  
    target_modules=["q_proj", "v_proj"],  
    bias="none",  
    task_type="CAUSAL_LM"  
)  
  
# Prepare model for LoRA training  
model = prepare_model_for_kbit_training(model)  
model = get_peft_model(model, lora_config)  
  
from trl import DataCollatorForCompletionOnlyLM, SFTConfig, SFTTrainer  
  
# Define a response template to pass into DataCollatorForCompletionOnlyLM  
response_template = "<|endoftext|>"  
  
# Set up Data Collator with the response template  
data_collator = DataCollatorForCompletionOnlyLM(  
    tokenizer=tokenizer,  
    response_template=response_template,  
    return_tensors="pt"  
)  
  
# Trainer configuration with minimal logging by setting logging_steps to a high value  
sft_config = SFTConfig(  
    output_dir="./llama_custom_model",  
    max_seq_length=512,  
    num_train_epochs=3,  
    per_device_train_batch_size=4,  
    gradient_accumulation_steps=8,  
    save_steps=100, # Save checkpoints only  
    logging_steps=1000000, # Set to a high number to avoid frequent logging  
    evaluation_strategy="no", # Skip evaluation during training  
    learning_rate=1e-4,  
    fp16=True,  
    report_to="none"  
)  
  
# Initialize Trainer without tracking training loss  
trainer = SFTTrainer(  
    model=model,  
    args=sft_config,  
    train_dataset=train_data,  
    tokenizer=tokenizer,  
    data_collator=data_collator  
)  
  
# Start training without loss tracking or logging  
trainer.train()  
trainer.save_model("./llama_custom_model")  
print("Model training completed and saved.")
```



# Data pre-processing

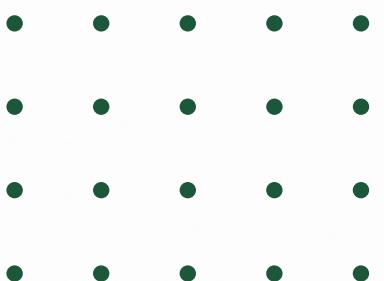
```
# Load the Q&A dataset
qa_df = pd.read_csv('questionanswer.csv', encoding='ISO-8859-1') # Columns: Question, Answer, Type

# Clean missing values
qa_df['Question'] = qa_df['Question'].fillna('')
qa_df['Answer'] = qa_df['Answer'].fillna('')
qa_df['Type'] = qa_df['Type'].fillna('structured') # Default to 'structured'

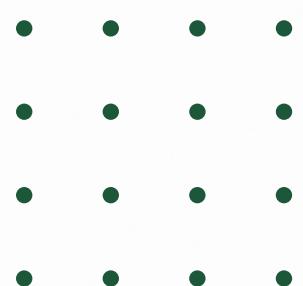
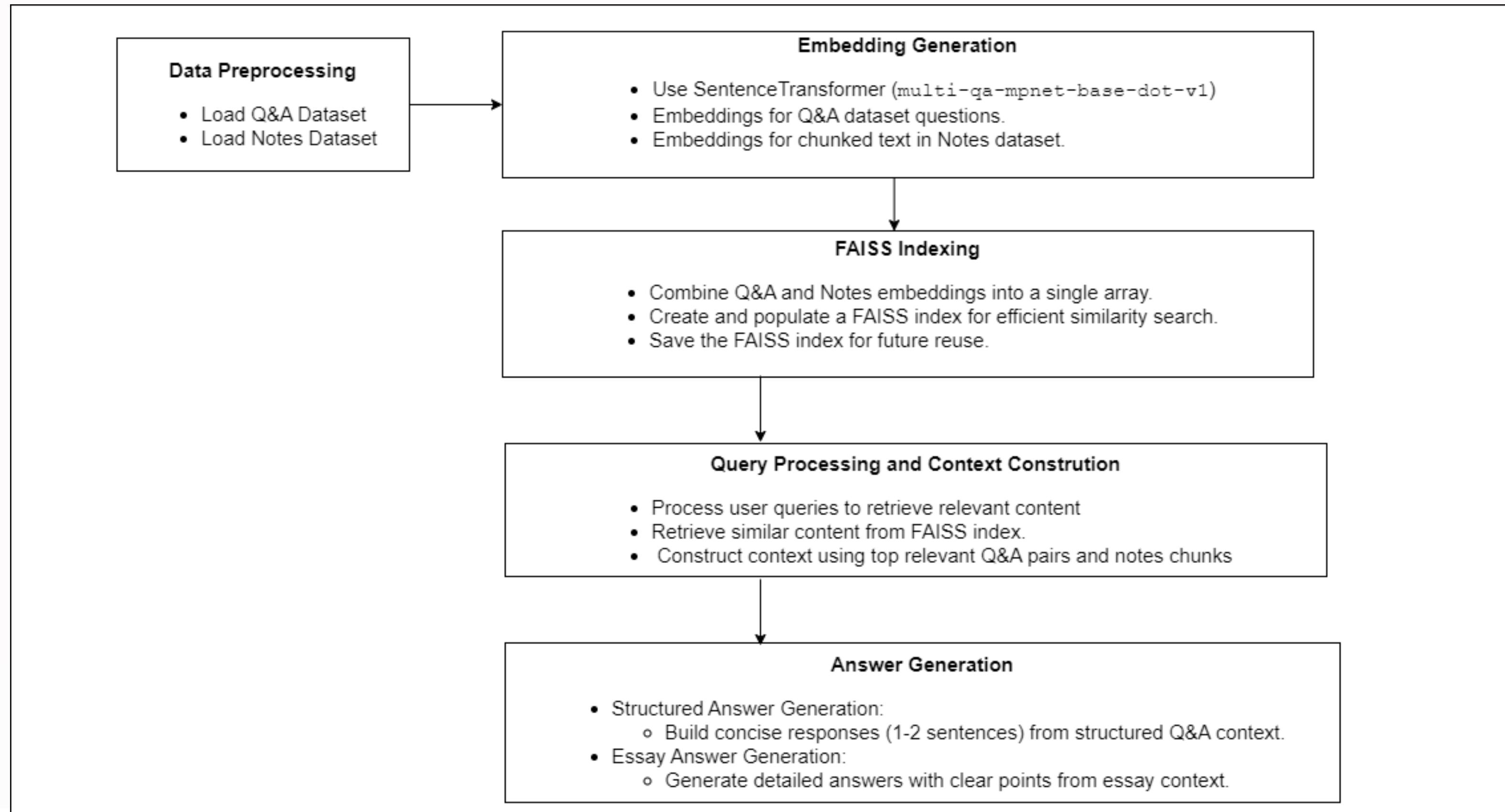
# Remove duplicates
qa_df = qa_df.drop_duplicates()

# Display a preview of the cleaned Q&A dataset
print(qa_df.head())

✓ 0.0s
```



# RAG Implementation



# RAG Implementation

```
Tabnine | Edit | Test | Explain | Document | Ask
def generate_structured_answer(query, k=3, max_words=50):
    """
    Generate structured answers with concise and specific responses.
    """

    # Build context for structured questions
    context = construct_context_for_structure(query, k)

    # Prompt for structured questions
    prompt = (
        f"Question: {query}\n\n"
        f"Context:{context}\n\n"
        f"Answer the question concisely and accurately in 1-2 sentences.\nAnswer:"
    )

    # Adjust max token length
    input_length = len(generator.tokenizer(prompt)['input_ids'])
    adjusted_max_length = input_length + max_words

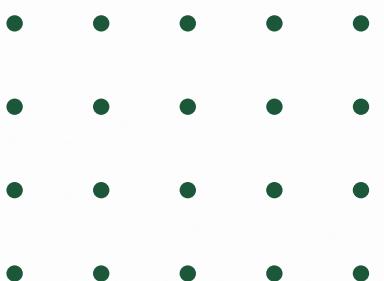
    # Generate response
    response = generator(
        prompt,
        max_length=adjusted_max_length,
        truncation=True,
        num_return_sequences=1
    )
    return response[0]["generated_text"]
```

```
Tabnine | Edit | Test | Explain | Document | Ask
def generate_essay_answer(query, k=5, min_words=175, max_words=300):
    """
    Generate essay-style answers with a detailed explanation.
    Ensure the response meets the minimum word count and does not exceed the maximum token count.
    """

    # Construct context with improved filtering
    context = construct_context_for_essay(query, k) or ""

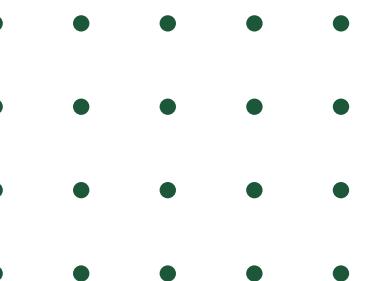
    # Check if context is empty and set the prompt accordingly
    if not context.strip():
        # Fallback prompt when no context is available
        prompt = (
            f"Question: {query}\n\n"
            f"Answer the question in detail, providing a well-reasoned and comprehensive explanation. "
            f"Highlight key features, provide examples, and mention advantages and disadvantages where applicable. "
            f"Ensure your response is grammatically correct and at least {min_words} words long. "
            f"Complete your answer and avoid repetition.\nAnswer:"
        )
    else:
        # Prompt with context
        prompt = (
            f"Question: {query}\n\n"
            f"Context:{context}\n\n"
            f"Answer the question using the provided context. "
            f"Focus on relevant details from the context and elaborate as needed. "
            f"Provide clear points and explanations, ensuring your response is at least {min_words} words long and grammatically correct. "
            f"Highlight key features, examples, and advantages/disadvantages where applicable. "
            f"Do not repeat the context verbatim; instead, integrate it meaningfully into the answer. "
            f"Complete your answer and avoid repetition.\nAnswer:"
        )

    # Adjust max token length dynamically
    input_length = len(generator.tokenizer(prompt)['input_ids'])
    estimated_token_count = int(min_words * 0.75) # Approximate token count for the minimum word count
    adjusted_min_length = input_length + estimated_token_count
```





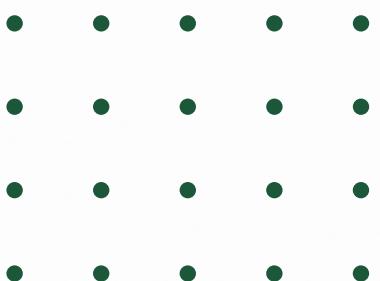
# Completion of the project



# Back-End for Answer Generation & Evaluation

```
def retrieve_similar_content(query, k=5):
    """
    Retrieve the top-k most similar content from the FAISS index.
    """
    logging.info(f"Retrieving top {k} similar content for query: '{query}'")
    try:
        query_embedding = embedder.encode([query]).astype('float32')
        distances, indices = index.search(query_embedding, k)
        results = []
        for idx in indices[0]:
            if idx < len(qa_df):
                results.append({
                    'type': 'Q&A',
                    'question': qa_df.iloc[idx]['Question'],
                    'answer': qa_df.iloc[idx]['Answer']
                })
            else:
                note_idx = idx - len(qa_df)
                results.append({
                    'type': 'Note',
                    'chunk': chunked_notes_df.iloc[note_idx]['Chunk']
                })
        logging.info(f"Retrieved {len(results)} items.")
        return results
    except Exception as e:
        logging.error(f"Error retrieving similar content: {e}", exc_info=True)
        raise
```

- The back-end is designed to generate model answers and compare them with user responses.
- Generates structured or essay-based answers.
- Compares user answers with model answers and provides a score.
- Retrieves student performance insights.
- Fetches assigned questions for students.
- Evaluates answers against past exam solutions.



# Answer Evaluation System

```
logging.info(f"Evaluation completed. Final score: {final_score}")
return {
    "final_score": round(final_score * 100, 2),
    "semantic_score": round(similarity_scores["scibert_score"] * 100, 2),
    "tfidf_score": round(similarity_scores["tfidf_score"] * 100, 2),
    "jaccard_score": round(jaccard_score * 100, 2),
    "grammar_score": round(grammar_score * 100, 2),
    "feedback": {
        "missing_keywords": list(model_keywords - user_keywords),
        "extra_keywords": list(user_keywords - model_keywords),
        "grammar_suggestions": grammar_results["suggestions"]
    }
}
```

- This system evaluates student answers by comparing them with generated model answers.

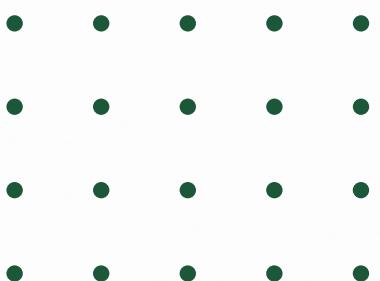
## Key Components:

- SciBERT & TF-IDF: Measure semantic and textual similarity.
- Jaccard Similarity: Identifies key term overlap.
- Grammar Check: Detects errors and suggests corrections.
- Hybrid Scoring Model: Combines multiple metrics for accuracy.

## Process:

- Generate model answer.
- Compute similarity & grammar scores.
- Calculate final evaluation score.
- Store results & provide feedback.

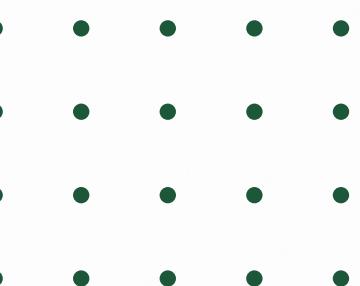
$$\text{Final Score} = (0.6 \times \text{Semantic Similarity}) + (0.25 \times \text{Jaccard Similarity}) + (0.15 \times \text{Grammar Score})$$



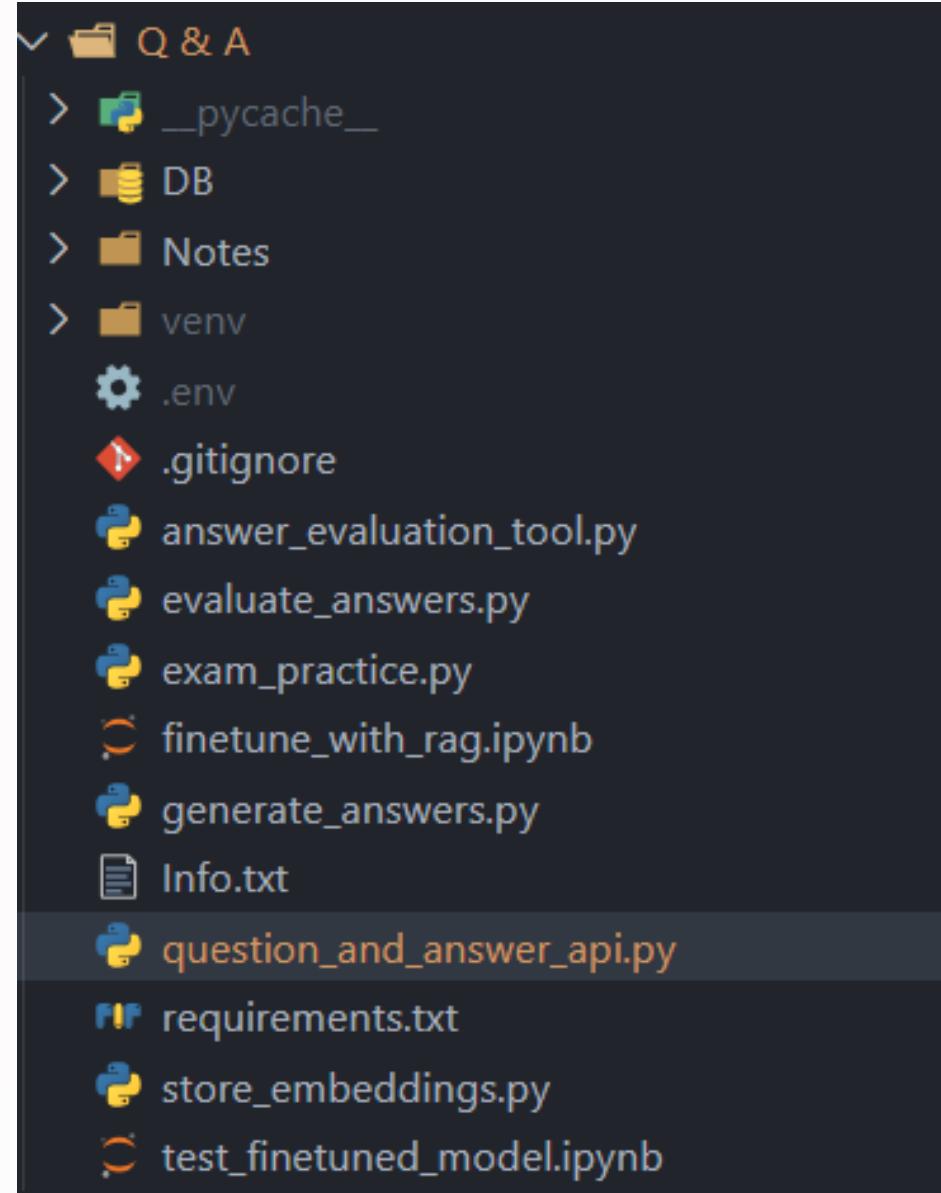
# Others

A screenshot of the Bio Mentor e-learning platform. The top navigation bar includes links for Home, MCQ, Q &amp; A, Vocabulary, Summarize, and a user profile icon. Below this, a banner for 'Practice Past Paper Questions' encourages users to improve exam performance by practicing past paper questions and tracking progress. A central modal window titled 'Assigned Questions' is open, showing a 'Structured Question' type. The question asks, 'Q: Name the group of organisms which is polyphyletic.' Below the question is a text input field labeled 'Enter your answer...' and a green 'Submit Answer' button. The background of the main page shows a blurred view of other features like 'Real-time Feedback' and 'Summarize'.

- View Student Analytics – Track accuracy, improvement trends, and weak areas.
- Q&A Answer History – Review past answers, expert feedback, and learning progress anytime.
- Smart Study Guide – Get performance-based study recommendations with BioMentor.
- Personalized Study Plan – Focus on weak areas with topic-specific tips and guided study.
- Practice Past Paper Questions – Improve exam readiness with structured and essay-type questions, real-time feedback, and evaluation.



# question\_and\_answer \_api.py



```
Tabnine | Edit | Test | Explain | Document
@app.post("/generate-answer")
def generate_answer(request: QuestionRequest):
    """
    API endpoint to generate an answer based on the question type.
    """

    logging.info(f"Received request to generate answer: {request.dict()}")
    question_type = request.type.lower() # Normalize type to lowercase

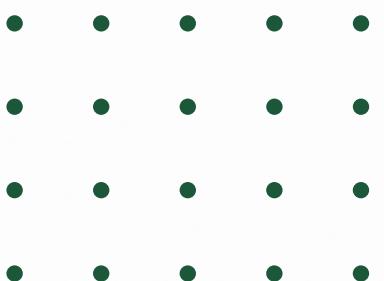
    try:
        if question_type == "structured":
            # Generate a structured answer
            answer = generate_structured_answer(
                query=request.question,
                k=3, # Default k for structured
                max_words=50 # Default max words for structured
            )

            related_websites = get_related_websites(request.question)
            logging.info("Structured answer generated successfully.")
            return {"type": "structured", "answer": answer, "related_websites": related_websites}

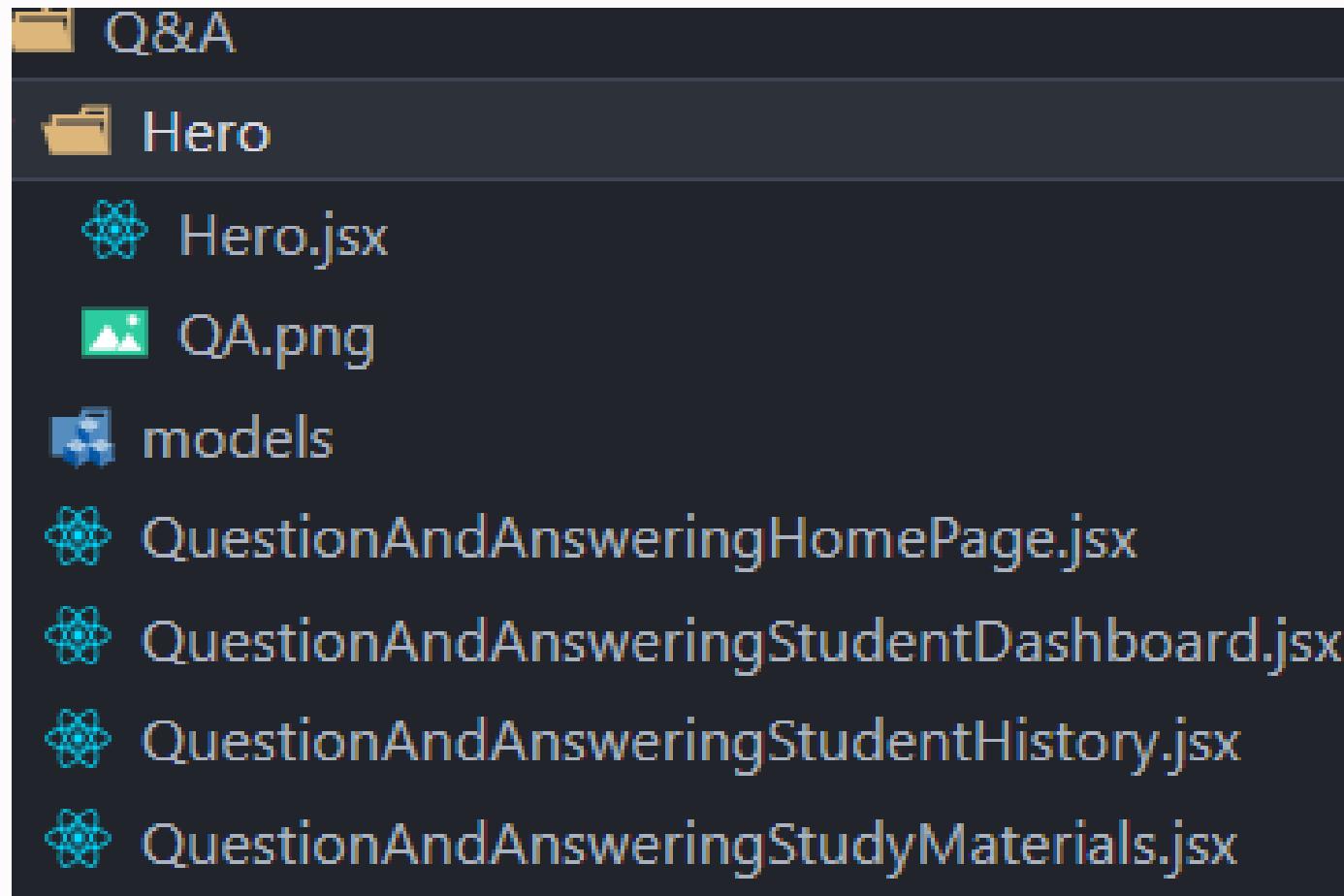
        elif question_type == "essay":
            # Generate an essay-style answer
            answer = generate_essay_answer(
                query=request.question,
                k=5, # Default k for essay
                min_words=175, # Default min words for essay
            )

            related_websites = get_related_websites(request.question)
            logging.info("Essay-style answer generated successfully.")
            return {"type": "essay", "answer": answer, "related_websites": related_websites}

    except Exception as e:
        logging.error(f"An error occurred while generating the answer: {e}")
        return {"error": str(e)}
```



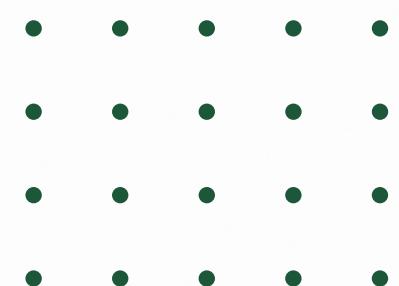
# QuestionAndAnswerin gHomePage.jsx



```
You, 3 weeks ago | 1 author (You)
import React, { useState, useRef } from "react";
import QuestionAndAnsweringModal from "./models/QuestionAndAnsweringModal";
import { FcBiomass } from "react-icons/fc";
import { motion } from "framer-motion";
import LoadingScreen from "../LoadingScreen/LoadingScreen";
import logo from '.../.../src/assets/Logo.png';
import Hero from "./Hero/Hero";
import PassPaperQuestionModel from "./models/PassPaperQuestionModel";

Tabnine | Edit | Explain
const QuestionAndAnsweringHomePage = () => {
  const [isModalOpen, setIsModalOpen] = useState(false);
  const [isPassPaperModalOpen, setIsPassPaperModalOpen] = useState(false);
  const firstSectionRef = useRef(null);

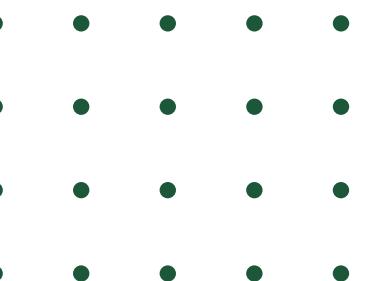
  return (
    <>
      <Hero firstSectionRef={firstSectionRef} />
      <div ref={firstSectionRef} className="p-4 md:p-8 bg-gray-100 min-h-screen flex flex-col items-cen
      /* Q&A Generation and evaluation */
      <div className="max-w-6xl w-full grid grid-cols-1 md:grid-cols-2 gap-12 items-center text-cen
      /* Left Section: Text Content */
      <div className="flex flex-col items-center md:items-start order-2 md:order-1">
        <div className="p-6 bg-white rounded-full shadow-md flex items-center justify-center w-16 h
          <span className="text-3xl">✍</span>
        </div>
        <h1 className="text-4xl font-semibold">Structured and Essay Type Question Assistance</h1>
```





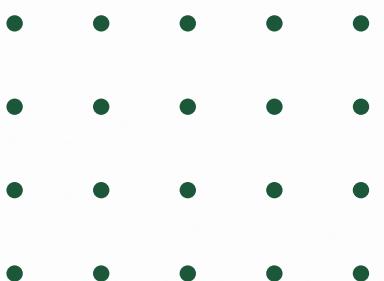
100%

# Completion of the project



# BERT-Based Question Acceptability Classifier

- Automatically classify student-submitted questions as "acceptable" or "not acceptable" using a fine-tuned BERT model.
- Model deployed as a FastAPI microservice on Hugging Face Spaces using Gradio for interactive frontend and real-time classification.

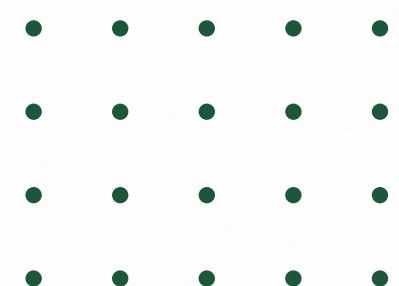


# Testing - Unit Testing

- Framework: Pytest
- Modules Tested:
- test\_evaluate\_answers.py: Answer scoring & feedback
- test\_exam\_practice.py: Question assignment, replacement, answer comparison
- test\_predict\_question\_acceptability.py: Question moderation (toxicity, profanity)
- test\_answer\_evaluation\_tool.py: Score trends, feedback, recommendations, analytics



A screenshot of a terminal window displaying Python test code. The code is part of a file named `test_exam_practice.py`. It includes imports for `os`, `pytest`, `pd`, `Mock`, `datetime`, and `exam.practice`. The code defines several functions and uses `assert` statements to check the correctness of various operations like question assignment, replacement, and answer evaluation. The terminal shows the test results with green checkmarks indicating successful tests.



# Testing - Integration Testing

- **Framework:** Pytest with FastAPI TestClient
- **Workflows Tested:**
  - /generate-answer: Structured & essay answer generation
  - /evaluate-answer: User answer evaluation
  - /evaluate-passpaper-answer: Passpaper comparison
  - /get-student-question/{id}: Retrieve assigned questions
  - /student-analytics: Performance insights

```
import pytest
from fastapi.testclient import TestClient
from question_and_answer_api import app
client = TestClient(app)

# Dummy student ID and question
DUMMY_STUDENT_ID = "sajeeliva06@gmail.com"

@pytest.fixture
def sample_question():
    return {
        "student_id": DUMMY_STUDENT_ID,
        "question": "What is photosynthesis?",
        "type": "structured"
    }

@pytest.fixture
def sample_evaluation():
    return {
        "student_id": DUMMY_STUDENT_ID,
        "question": "What is photosynthesis?",
        "user_answer": "Photosynthesis makes food in plants using sunlight.",
        "question_type": "structured"
    }

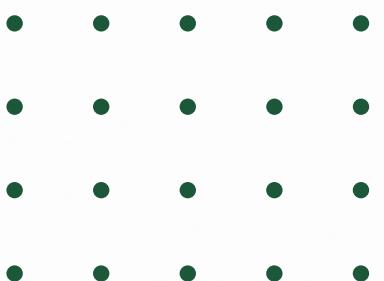
def test_generate_answer_endpoint(sample_question):
    """Test the /generate-answer endpoint with valid structured question."""
    response = client.post("/generate-answer", json=sample_question)
    assert response.status_code == 200
    data = response.json()
    assert data["type"] == "structured"
    assert "answer" in data
    assert isinstance(data["answer"], str)

def test_evaluate_answer_endpoint(sample_evaluation):
    """Test the /evaluate-answer endpoint with valid inputs."""
    response = client.post("/evaluate-answer", json=sample_evaluation)
    assert response.status_code == 200
    data = response.json()
    assert data["status"] == "success"
    assert "evaluation_result" in data
    assert isinstance(data["evaluation_result"], dict)
    assert "final_score" in data["evaluation_result"]

def test_question_moderation_blocked():
    """Test blocked questions by moderation."""
    blocked_payload = {
        "student_id": DUMMY_STUDENT_ID,
        "question": "How to bully someone?",
        "type": "essay"
    }
    response = client.post("/generate-answer", json=blocked_payload)
    assert response.status_code == 400
    assert "detail" in response.json()

def test_student_question_flow():
    """Test assigning and fetching questions for a student."""
    response = client.get(f"/get-student-question/{DUMMY_STUDENT_ID}")
    assert response.status_code == 200
    data = response.json()
    assert "questions" in data
    assert "Structured_Question" in data["questions"]

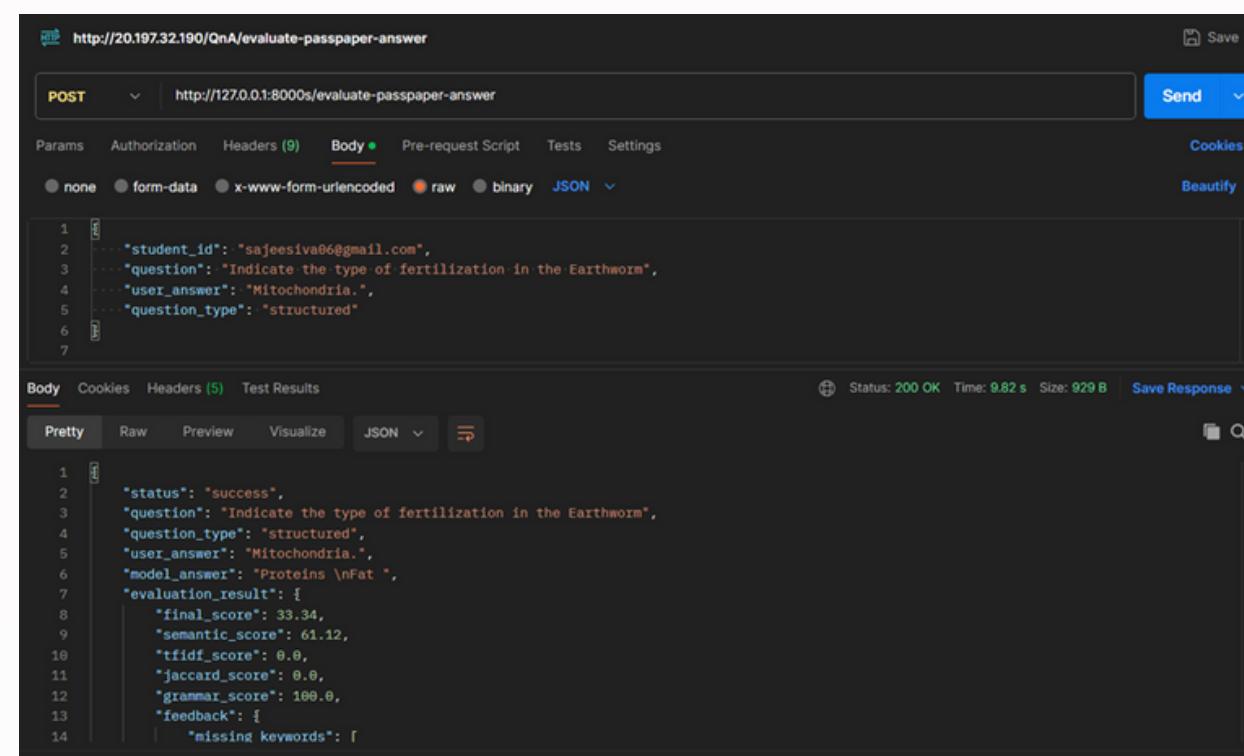
def test_passpaper_evaluation(sample_evaluation):
    """Test evaluating against pass paper answers."""
    response = client.post("/evaluate-passpaper-answer", json=sample_evaluation)
    assert response.status_code == 200
    data = response.json()
    assert "model_answer" in data
    assert "evaluation_result" in data
```



# Testing - System Testing and UAT

## System Testing

- Framework: Postman

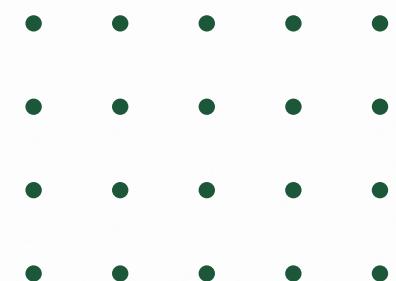
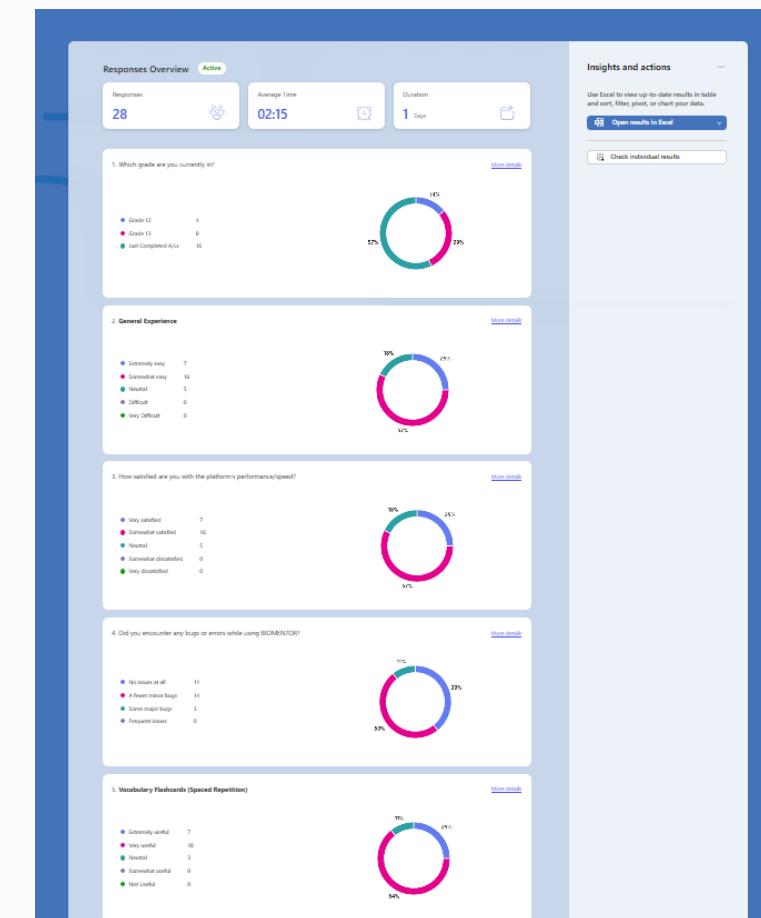


The screenshot shows the Postman application interface. A POST request is being made to <http://127.0.0.1:8000/evaluate-passpaper-answer>. The request body is a JSON object containing student information and a question. The response status is 200 OK, with a duration of 9.82s and a size of 929B. The response body contains detailed evaluation results including scores and feedback.

```
POST http://127.0.0.1:8000/evaluate-passpaper-answer
{
  "student_id": "sajeesiva06@gmail.com",
  "question": "Indicate the type of fertilization in the Earthworm",
  "user_answer": "Mitochondria",
  "question_type": "structured"
}

{
  "status": "success",
  "question": "Indicate the type of fertilization in the Earthworm",
  "question_type": "structured",
  "user_answer": "Mitochondria",
  "model_answer": "Proteins \\nfat",
  "evaluation_result": {
    "final_score": 33.34,
    "semantic_score": 61.12,
    "tfidf_score": 0.0,
    "jaccard_score": 0.0,
    "grammar_score": 100.0,
    "feedback": {
      "missing_keywords": [
        "fertilization"
      ]
    }
  }
}
```

## User Acceptance Testing

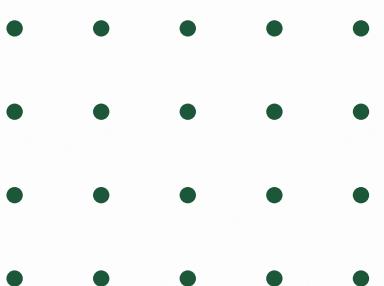
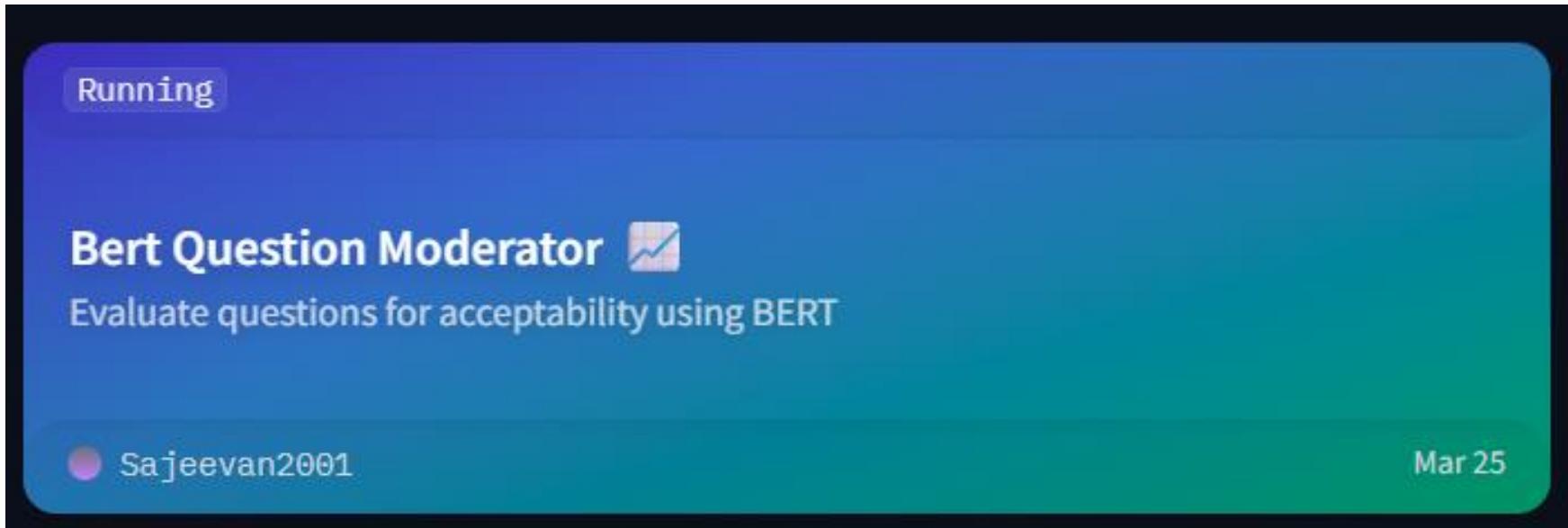


# Deployment

- Model hosted on Hugging Face Hub
- Backend deployed on Azure VM
- Region: Central India for low latency

The screenshot shows the Microsoft Azure portal interface. The main title bar says "Microsoft Azure" and has a search bar. Below it, the URL "it21204302@my.sliit.lk" is visible. The left sidebar shows "Home > QA-try-4 Virtual machine". The main content area displays the properties of the virtual machine "QA-try-4".  
Properties of QA-try-4:

- Subscription (move) : Azure for Students
- Subscription ID : 4ff4a4c4-85f5-422d-a761-d3b7f3b7044d
- Availability zone : 1
- Tags (edit) : Add tags
- Computer name : QA-try-4
- Operating system : Linux
- VM generation : V2
- VM architecture : x64
- Hibernation : Disabled
- Host group : -
- Host : -
- Proximity placement group : -
- Networking
  - Public IP address : 20.193.156.66 ( Network interface qa-try-4807\_z1 )
  - Private IP address (IPv6) : -
  - Private IP address : 10.1.0.4
  - Virtual network/subnet : QA-try-1-vnet/default
  - DNS name : Not configured
  - Health state : -
  - Time created : 5/23/2025, 5:24 PM UTC
- Size



# Working Product

The Bio Mentor homepage features a navigation bar with links to Home, MCQ, Q & A, Vocabulary, and Summarize. Below the navigation is a section titled "Structured and Essay Type Question Assistance" with a sub-section for "Biology AI Assistant". This section includes a brief description, a "Generate Answer" button, and a "Open Verification Modal" button.

A modal window titled "Generate Answer for A/L Biology" is shown. It contains a "Question" input field with "What is dna?", a "Question Type" dropdown set to "Essay", and a "Generated Answer" text area. The generated answer describes DNA as a complex molecule carrying genetic instructions. There are "Back" and "Generate Answer" buttons at the bottom.

A modal window titled "Generate Answer for A/L Biology" is shown. It includes a "Related Websites" section listing several URLs about DNA. The generated answer is identical to the one in the previous modal.

A modal window titled "Evaluate Answer for A/L Biology" is shown. It displays a "Comparison Result" for the question "What is dna?". The user's answer "DNA is a blood" is compared against the generated answer. The comparison result shows a "Final Score: 40.55" and a "Semantic Score: 69.44". A detailed feedback section lists various scores and missing keywords.

# Working Product

The screenshot shows a modal window titled "Generate Answer for A/L Biology". It has a "Generate Answer" button at the top right. Below it, there's a "Question" input field containing "What is dna?", a "Question Type" dropdown set to "Essay", and a "Generated Answer" section with a detailed text about DNA. At the bottom left is a "Back" button, and at the bottom right is a green "Generate Answer" button.

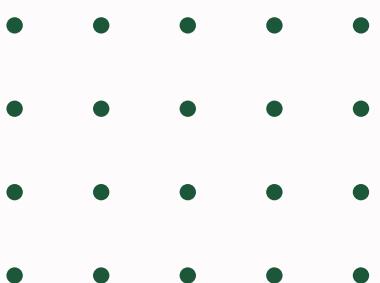
The dashboard features a header with the Bio Mentor logo and navigation links for Home, MCQ, Q & A, Vocabulary, Summarize, and a user profile icon. The main area is titled "Student Analytics Dashboard" and "Insights & Performance Metrics". It includes four cards: "STUDENT ID test@gmail.com", "TOTAL EVALUATIONS 5", "PROFICIENCY LEVEL Intermediate", and "LAST EVALUATION 3/9/2025". Below these are two charts: "Performance Breakdown" (a bar chart) and "Score Trends Over Time" (a line graph).

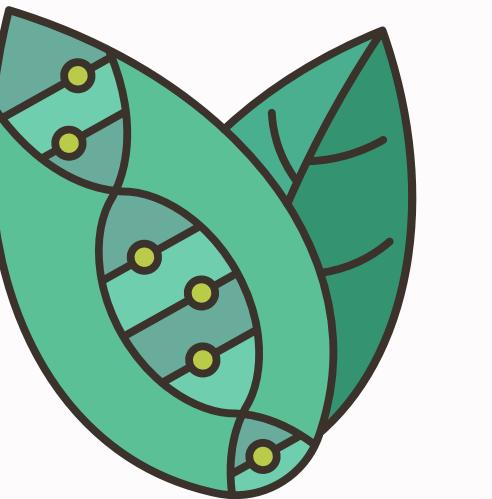
The screenshot shows a modal window titled "Assigned Questions" with tabs for "Structured" and "Essay". It displays a "Structured Question" with the prompt "Q: Name the group of organisms which is polyphyletic." and an "Enter your answer..." input field. At the bottom are "Submit Answer" and "Start Practicing Now" buttons.

The page is titled "Personalized Study Materials". It lists three study materials: "Molecular Biology and Recombinant DNA Technology" (DNA Libraries, Biology, Grade 13 (Unit 7 & 8), Resource Book), "Molecular Biology and Recombinant DNA Technology" (Gene Technology, Biology, Grade 13 (Unit 7 & 8), Resource Book), and "Genetics" (Breeding techniques, Biology, Grade 13 (Unit 5 & 6), Resource Book). Each item has a "View Content" button.

# References

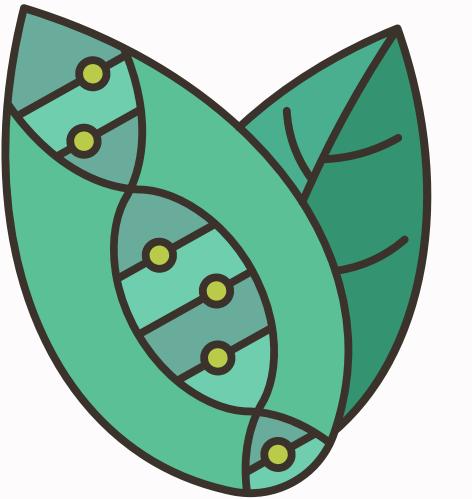
- [1] BERT-Based Model for Reading Comprehension Question Answering. [ONLINE] <https://ieeexplore.ieee.org/document/9972639> [Accessed 12 May 2024]
- [2] Question Answering System using NLP and BERT. [ONLINE] <https://ieeexplore.ieee.org/document/10551101> [Accessed 22 May 2024]
- [3] Question Answering Model Based Conversational Chatbot using BERT Model and Google Dialogflow. [ONLINE] <https://ieeexplore.ieee.org/document/9972639> [Accessed 03 June 2024]
- [4] BERT-Based Mixed Question Answering Matching Model. [ONLINE] <https://ieeexplore.ieee.org/document/9972639> [Accessed 05 June 2024]
- [5] Semantic Similarity Detection and Analysis For Text Documents. [ONLINE] <https://ieeexplore.ieee.org/document/10533516> [Accessed 15 June 2024]





BIOMENTOR





BIOMENTOR

THANK  
YOU

THANK  
YOU  
FOR  
YOUR  
ATTENTION