KRISHNA MADHWANI

Project Title: Refactoring and Publishing EduAid
Project Size: Large (350 hours)

Mentor:
Prarabdh Shukla
Harsh Mishra
Divyanshu Singh

Contact Information

Email: krishna.madhwani.cse21@itbhu.ac.in
Github profile: https://github.com/Roaster05
Slack: **Krishna Madhwani**
Mobile: **+917974032798**
TimeZone: **+5:30 GMT**
LinkedIn: https://www.linkedin.com/in/krishna-madhwani-978798223/
Resume/CV: **Krishna Madhwani.pdf**
Discord: **NoobKido (noobkiddo1213)**

My GSoC'23 (SoR) Project

Organization: UCOSPO

Chosen Project: Public Artifact and Data Visualization

Domain: Web Development, CLI Development, Visualisation Tools etc.

My Blog Page: https://ucsc-ospo.github.io/author/krishna-madhwani/

Education

University: *Indian Institute of Technology (BHU), Varanasi*

Degree: B.Tech

Expected Graduation Date: May 2025

Courses and Your Performance: Scored 'A' Grade in both the subjects of *Natural Language Processing* and *AI.*

Commitment

How many hours will you work per week on your GSoC project?

Considering my previous experience in GSoC, I know the consistency and the efforts it requires, and hence I can commit a total of **30-35** hours each week.

Other Commitments: *No*

Project Summary

Chosen Project: *EduAid*

Chosen Idea: *Refactoring and Publishing EduAid*

Vision:

The proposal aims to improve EduAid by adding **diverse question types**, enhancing help functionality through the **SidePanel,** and **generating answer keys and summarized notes.** This includes **Multiple Choice, Single Correct, and Boolean Questions** for versatile assessment creation. Leveraging Google Chrome's SidePanel API ensures easy navigation and uninterrupted browsing. Automated generation of answer keys and notes helps students prepare effectively. A **uniform user interface** will give the extension a professional look, and allow the user to take content from **various mediums of Data containing files.** These enhancements aim to revolutionize educational content creation and delivery for a more engaging learning environment.

Motivation:

The reason I believe I am the best candidate for leading this project this summer lies in the given two facts:-

Contribution:

- Made the **Highest amount of contributions** (almost 2x of the Second best contributor) by Actively contributing to the codebase by making a total of **9 Pull Requests,** raising **8 Issues,** and contributing a **UI interface for the same**.
- Invested **2-3 months** in comprehensive exploration of the proposal's intricacies.
- **Already developed an advanced model** for question categorization, prioritizing efficiency, speed, and accuracy which turns out to be the best from what other contributors have proposed
- **Designed a visually appealing UI** to elevate the user experience and professionalism.

Experience:

- Qualified for **GSoC'23**, showcasing proficiency in Open Source programs and project management.
- Demonstrated competence in navigating Open Source environments and fostering effective team collaboration through **various Podium Winning positions** in Hackathons (**Bronze at InterIIT Tech Meet 12.0**).

Past Experience in Software Development

- **FinHealth :** (A **Chrome Extension** based project)

  GitHub Link: https://github.com/Roaster05/FinHealth

  Description:

  - Chrome Extension: Tracks expenses during online shopping with one-click access on listed e-commerce platforms, providing real-time tracking convenience.
  - Database Storage: Securely stores all expenses for comprehensive spending history and analysis.
  - Integration Features: Seamlessly combines Gmail integration, price comparison across platforms, and an AI chatbot for comprehensive expense tracking and financial management.
  - User Benefits: Effortless expense monitoring, real-time insights, and historical analysis for better budget management.

  Technologies and programming languages used:
  Chrome API, Django, Python, JavaScript, HTML, CSS

- **Trumio:** (**Inter IIT 12.0 Bronze** Winning Project)

  Deployed Link : https://trumio-interiit12.vercel.app/

  GitHub Link: **Due to constraints of the Institution, the repository is Private.**

  Description:

  - Participated in the Product Development Challenge of Trumio in Inter IIT Tech Meet 12.0 and led the development team for my college in this PS.
  - Developed a Full-Stack product containing a total of 5 AI and recommendation features, which used a combination of RAG-based LLMs and other ML algorithms for serving the purpose.
  - Our developed product received the highest points in the development phase for our technical innovations in the development of these AI features.

  Technologies and programming languages used:
  Nextjs, MongoDB, Flask, OpenAI + RAG, Recommendation Engines etc.

Contributions:

*I have Made a Total of 9 Good PR's along with their proper documentation, which is by far the **Highest in terms of numbers as well as quality** for this project. I am listing my PR's here:*

- **Figma Design:** Proposed a Figma UI for the extension in issue #2 and provided a detailed explanation of all the Wireframes along with the flow of information between the different wireframes.

- **Pull Request #32:** [Improvement] Revamped Question Generator Model: Issue Linked: #31:
    - This PR introduces **a revamped question generator model** with methods for generating multiple-choice, boolean, and short-answer questions. It includes endpoints for automated quiz generation and answer key generation, utilizing advanced natural language processing techniques and allowing customization of the number of questions generated.

- **Pull Request #30:** [Improvement] Leveraging SidePanel API for better Visualization of Quiz: Issue Linked: #29:
    - This PR enhances quiz **visualization by integrating SidePanel's API** for improved user experience. It introduces a background script (*serviceWorker.js*) and adjusts files for seamless integration, including an alert feature for convenient access to generated questions.

- **Pull Request #28:** [Feature] Storing Previously Generated Quizzes: Issue Linked: #27:
    - This PR introduces a **dashboard** accessible from *index.html*, featuring *previous.html* for managing generated quizzes. It includes a title input field in *text_input.html* and displays quiz titles with metadata on hover for an improved user experience.

- **Pull Request #19:**[Improvement] Revamped UI/UX for the Extension: Issue Linked: #2:
    - This PR **refines the UI** by integrating *Tailwind CSS*, improving the layout in *index.html*, enhancing user input in *text_input.html* with responsive features, and introducing a modal interface in *question_generation.html* for displaying questions. Public assets are centralized for easier access.

- **Pull Request #18:** [Improvement] Adding Context-Menu option for Generating Quiz: Issue Linked: #17:
    - This PR enhances user experience by adding a **context-menu option** for quiz generation from selected text. It includes a background script (*service_worker.js*) to define context menu metadata, storing highlighted text in browser local storage for retrieval by *text_input.js*, and improving interaction in the extension's popup window.

- **Pull Request #15:** [Bug] Upload PDF not working: Issue Linked: #14:
    - This PR incorporates external scripts (*pdfjsLib.js and pdfjsLib.worker.js*) into the scripts folder. These scripts are then imported into the *text_input.html* file for use.

- **Pull Request #13:** [Improvement] Interactive Question and Answer Sheet Generation: Issue Linked: #12
    - This PR introduces **Interactive Quiz and Answer Key generation** using *pdf-lib.js*, adjusting *question_generation.html* to include a title input field and a download button for the answer key. Additionally,

question_generation.js is modified to implement functions for mapping the generated quiz in PDF format.

- **Pull Request #39:** [Improvement] Customisation in Generated Quiz: Issue Linked: #38
    - This PR incorporates features for modifying, deleting, and adding Question-Answer pairs in the generated quiz set, empowering users to customize content and address specific requirements for improved user satisfaction and flexibility.

- **Pull Request #37:** [Feature] Added Google Docs Support: Issue Linked: #36
    - To enhance our extension's functionality, this PR integrates a Google Docs URL Parser utilizing the Google Docs API, allowing users to input Google Docs URLs directly. This parser would fetch content for Quiz Generation, potentially employing web scraping to extract text. By bridging this gap, we enable users to utilize Google Docs content seamlessly within our extension, improving its versatility and usability.

Proposal:

1. Enhanced Question Diversity (Key Feature)

The Question Generator Model I have been developing represents a groundbreaking advancement in educational content creation. With a **response time under 25 seconds** for each question type, it **outperforms previous PRs by other contributors by a significant margin, which typically took over 2 minutes.** This rapid generation is coupled with high-quality questions featuring meticulously crafted distractors, ensuring an engaging and effective learning experience. Additionally, the incorporation of a contest-based answer key feature provides precise explanations alongside correct answers, enhancing comprehension and mastery for learners.

I have **already submitted a Pull Request (PR) for your review,** which includes the attached **screencasts** for a better understanding of our accomplishment.

WHAT

The extension broadens its question types to include **Multiple Choice Questions (MCQ), Single Correct, and Boolean Questions**, enriching its coverage for more comprehensive assessment creation.

WHY

1. **Enhanced Assessment Formation:**
   The addition of MCQs, single correct, and boolean questions offers flexibility in assessment design, enabling a nuanced evaluation of student understanding across cognitive levels.

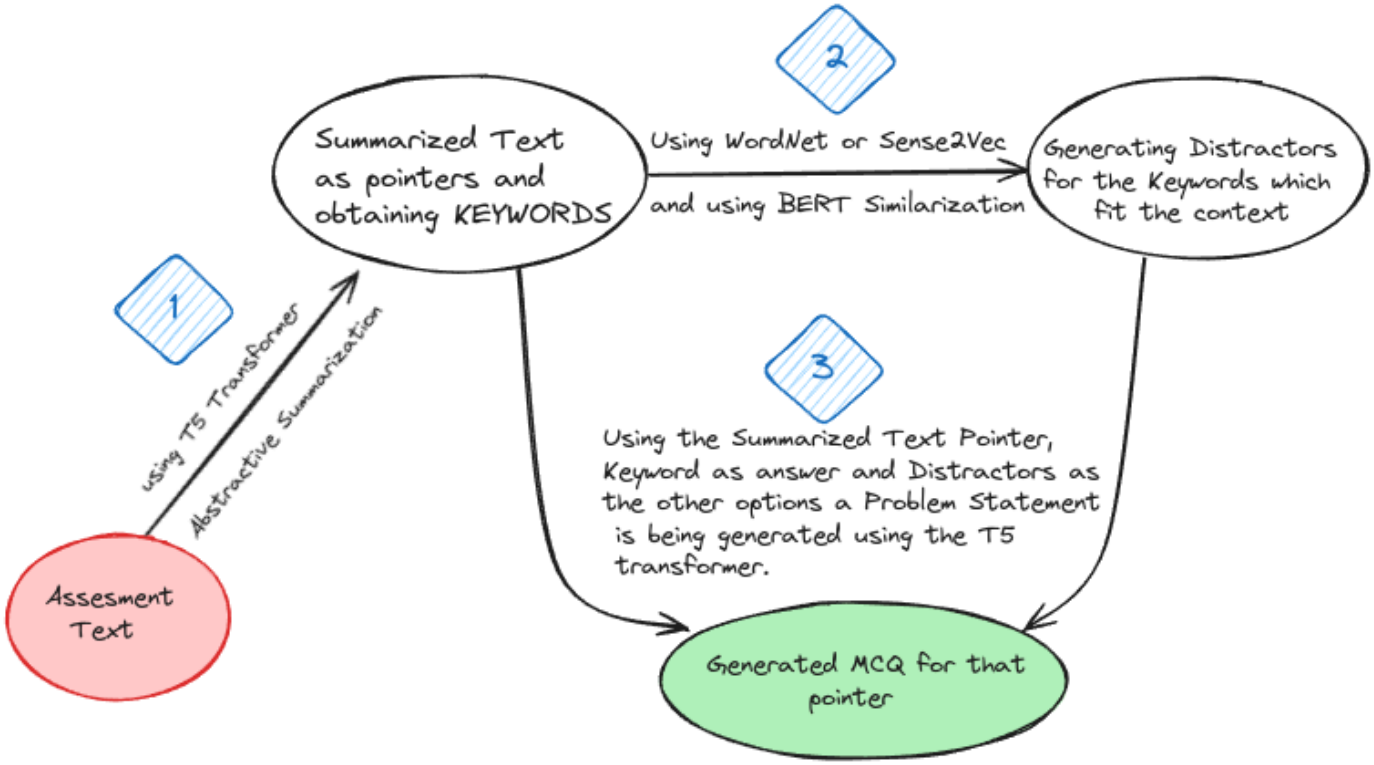
2. **Improved Question Generation Pipeline:**
   Diversifying question types creates a stronger pipeline for generating aligned assessment items, ensuring higher quality and relevance.

3. **Increased Acceptance Rate:**
   Offering varied question types boosts acceptance rates, reduces manual intervention, and saves teachers' time.

4. **Faster Response Time:**
   A wider range of question types streamlines assessment creation, leading to quicker response times for both teachers and learners.

HOW

- Multiple Choice Question



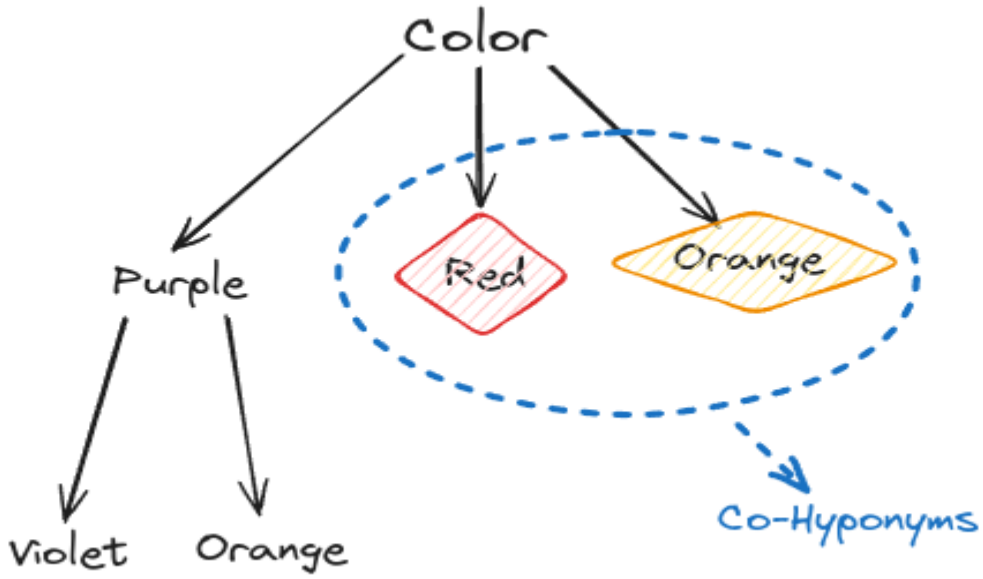
Step 1: Point-based Abstractive Summarization

- Integrating a pre-trained **abstractive summarization model T5** (Text-to-Text Transfer Transformer) to generate a concise summary of the input text.

- We segment the text into several pointers and then rank them using the techniques of **TF-IDF to rank** them in order of relevance.

- For each pointer, we obtain an associated keyword with that pointer using the **Entity Recognition** technique and associate a pointer with a keyword, which will act as the answer for the MCQ generated with that pointer.

Step 2: Generating Distractors from Keywords



- We generate the **Co-Hyponyms for the keyword** by accessing the umbrella class of the keyword.

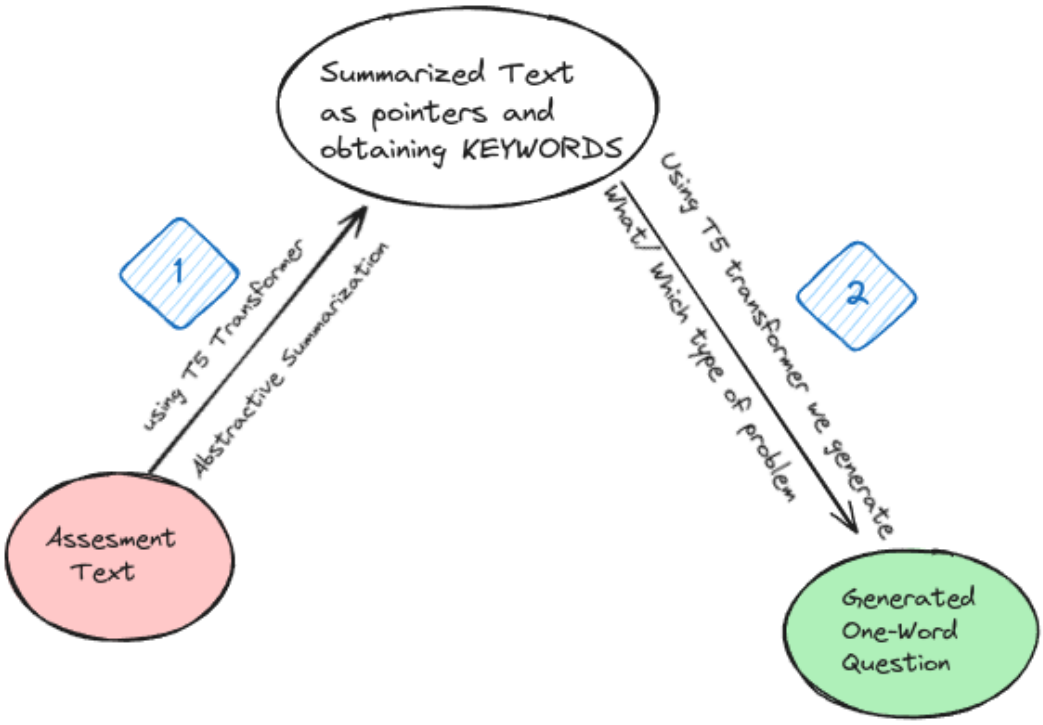
- We will utilize the libraries `nltk.corpus.wordnet` or `gensim.models.word2vec` to find synonyms/related terms for the answer keyword.

Step 3: Generating MCQ statement from the pointer and the keyword

- Using the **Stanford Question Answering Dataset and T5 Transforme**r, we generate an MCQ statement and ensure that it fits the context of the distractors and the keyword.

- Shuffle the answer choices to prevent bias and ensure they are grammatically correct.

- One Word Question



Step 1: Obtaining Keywords

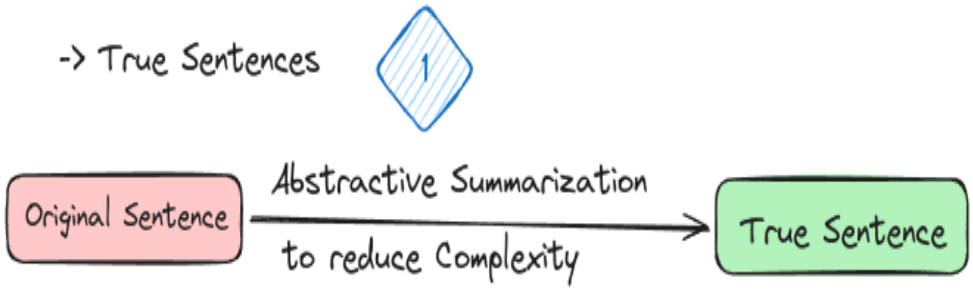
- Using the **T5 Transformer and the Multipartite Rank**, we obtain the keywords based on their relevance.

Step 2: Forming the problem statement

- Using the Stanford Question Answering Dataset and T5 Transformer, we generate a One Word question statement and ensure that it fits the context of the keyword

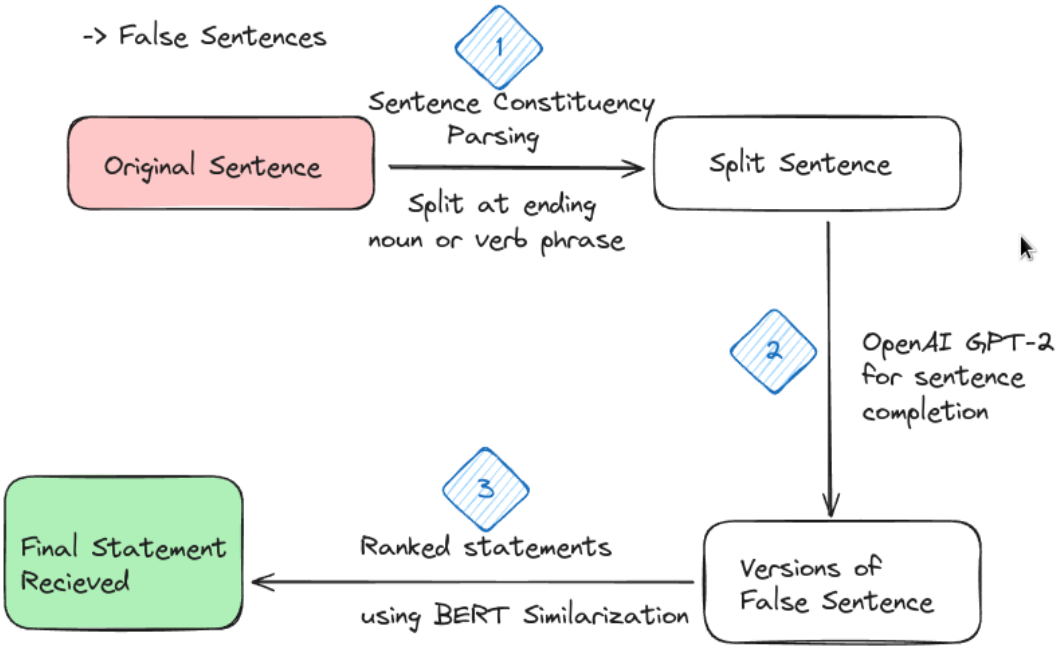
- Boolean Questions

True Statements:



- We reduce the complexity of the original sentence by summarizing it, which will act as a true statement.

False Statements:



- Leverage **Natural Language Processing (NLP) libraries like NLTK or SpaCy** to analyze sentence structure. These tools will help identify and split sentences based on the grammatical structure of their ending noun phrases.

- Utilize **OpenAI GPT-2,** a large language model, to generate various alternative phrasings for the ending portion of the base sentence. By employing techniques like beam search or nucleus sampling, you can ensure diversity and coherence within the generated options.

- To assess the "falsehood" of generated options, leverage **pre-trained BERT models.** These models generate sentence embeddings that capture the meaning of the sentence. Calculate the cosine similarity between the original sentence (base + ending phrase) and each generated option using their respective embeddings. The option with the lowest similarity score is considered the least similar and, therefore, the most likely "false" option.

2. User Interface (UI)

A **Pull Request (PR) has already been submitted** for the same, showcasing the integration of some pop-ups that support the previous version of the Chrome extension. Take a moment to review it here, and I am attaching the **Figma File** here

WHAT

Implementing a consistent user interface (UI) design throughout the extension, focusing on uniformity in layout, color scheme, and overall aesthetics.

WHY

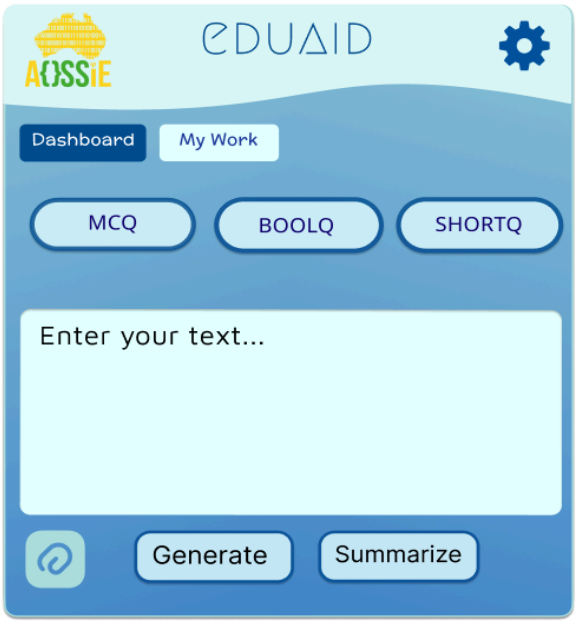
- **Improving User Experience:** Previous UI lacks a production-level feel, leading to inconsistencies in design elements and user interaction. By introducing a uniform UI, users can navigate the extension seamlessly, enhancing their overall experience.

- **Professional Appearance:** A consistent UI design establishes a professional appearance, instilling confidence in users and creating a polished impression of the extension.

- **Ease of Use:** Uniformity in layout and color scheme enhances usability by providing visual cues and maintaining predictability, resulting in smoother navigation and task completion for users.

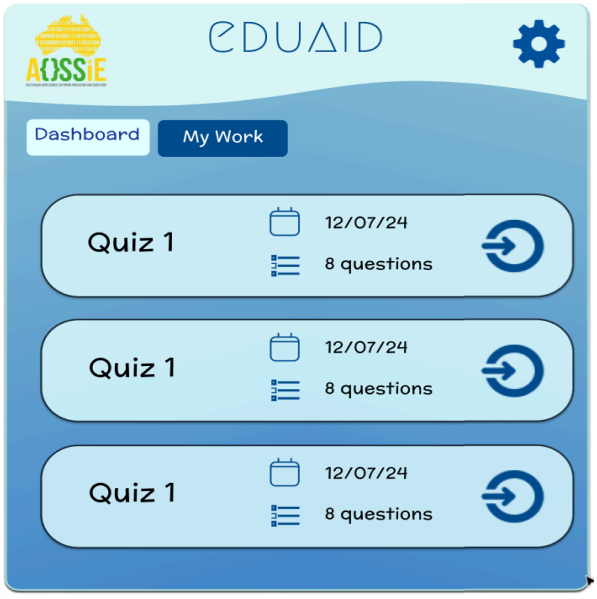
HOW

- Dashboard (Extension Popup)



The initial screen serves as the primary interface upon clicking the extension icon. Here, users can effortlessly create quizzes by inputting text and specifying parameters for each question numerically. Additionally, the screen features a settings icon and provides the option to upload a PDF document for text input convenience.

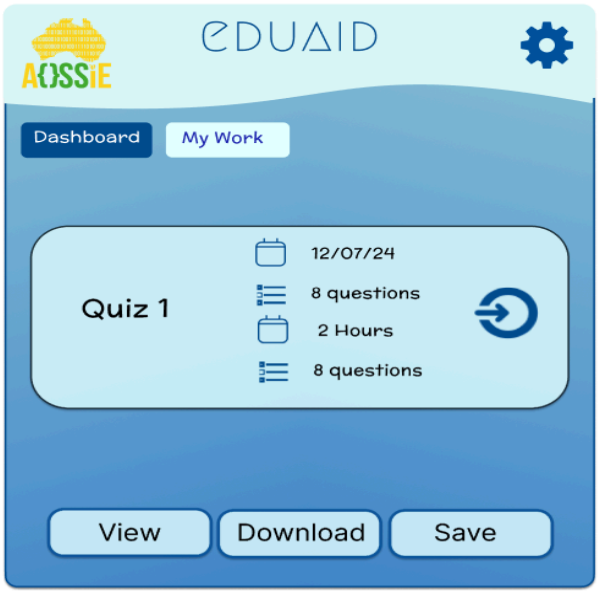
- My Work (Extension Popup)



This screen functions as a hub where **previously generated quizzes are showcased** in a structured tabular format, accompanied by metadata such as question numbers and creation dates. Furthermore, it offers a gateway button for accessing individual quizzes, enabling users to delve into each quiz and review its associated problems.

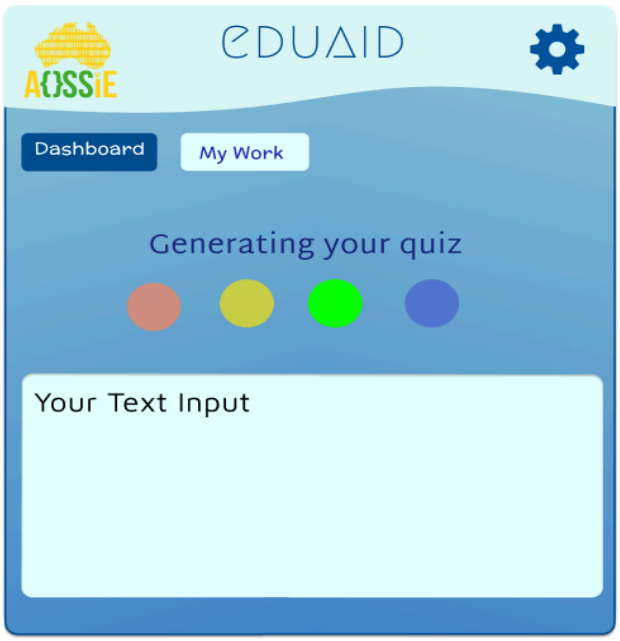
- Generated Quiz Display (Extension Popup)

  This interface **showcases the metadata linked to the generated quiz,** such as its title, the quantity of different question types generated, the date of creation, and provides options for viewing, downloading, and saving the quiz.



- Loading Screen(Extension Popup)

  This screen **serves as a loading indicator**, featuring an animated sequence of dots arranged to mimic a wave-like pattern. Each dot represents varying shades of RGB colors.



- [Quiz Display (Side Panel)]

This screen **harnesses the power of the sidePanel API, presenting a dynamic display of quizzes in various formats.** Users can effortlessly navigate through different question types while accessing convenient options such as saving the quiz, regenerating it, and exporting it as a PDF.

3. Enhanced Question Visualization (Leveraging SidePanel API)

A **Pull Request (PR) has already been submitted** for the same which demonstrates the use of SidePanel in the current version of the chrome extension and how it leverages the visualization of the generated questions

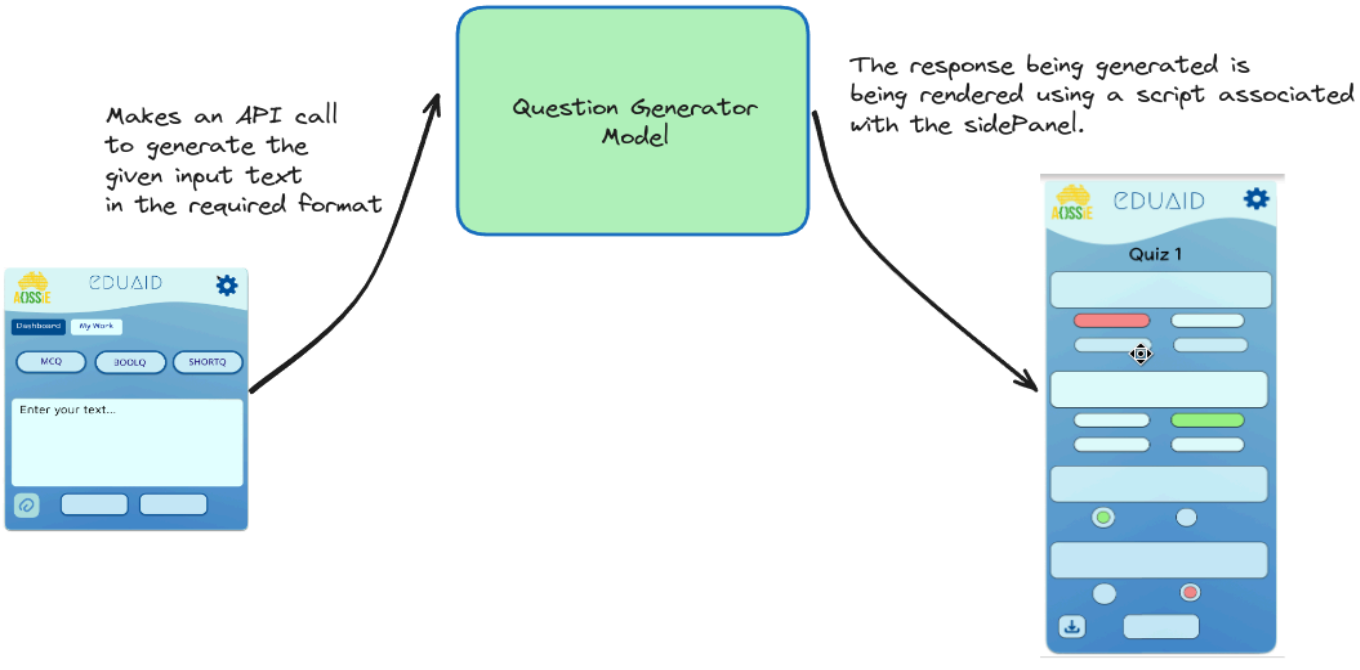
WHAT

Implementing the **SidePanel feature** to leverage its **size and vertical orientation** for displaying quizzes and providing additional functionalities such as saving, regenerating, and exporting quizzes.

WHY

- **Ample Space**: Side panels offer more room than pop-ups, accommodating multiple questions and options comfortably.
- **Vertical Layout**: The side panel's vertical orientation aligns well with the reading flow, facilitating easy navigation through quiz content.
- **Consistent Positioning**: Unlike pop-ups, side panels maintain a steady position alongside the main content, ensuring uninterrupted browsing and quiz-taking.
- **Enhanced Visibility**: Positioned adjacent to the main content, side panels improve quiz visibility, eliminating the need for window adjustments and reducing cognitive load.

HOW

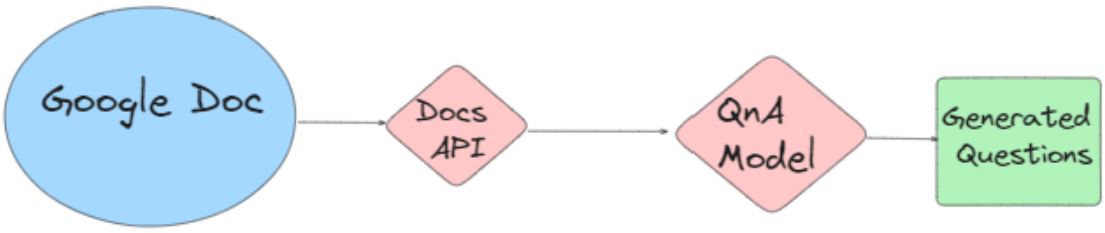


- Google Chrome recently introduced the **SidePanel's API,** which can be accessed by the Chrome extensions. We will be leveraging its introduction by creating a **separate HTML file for its content and mentioning the same in the manifest.json file;** only then we will associate a script with it as well and provide accessibility by mentioning it in the manifest.json file.

- When a request for quiz generation is made from the extension popup, the request is directed to the backend model, which in turn sends the desired response. This desired **response is then accessed by the SidePanel's script and then populates the page with its content.**

4. Incorporating Different Various Input Mediums

- **Google Docs:** Considering Google Docs is the primary tool used to create online content, a parser that can extract text from these documents and utilize it to generate questions would satisfy our requirement to use Google Docs for content generation. To accomplish this, we will use the **Docs API** in conjunction with a scraping method that reaches the child nodes to extract text content.
  This has **already been implemented** in this PR:
  [Feature] Added Google Docs Support:



- **Educational YouTube Videos:** Since YouTube videos are the most widely used resource for education, it will be vital and important for our aspects to incorporate a method that can perform this process of generating a quiz from the content being taught in the video. To achieve this, we will first access the video using the URL provided; once there, it will be **split into multiple sections of audio input using metric-based techniques,** which are then being **converted into text.**

-

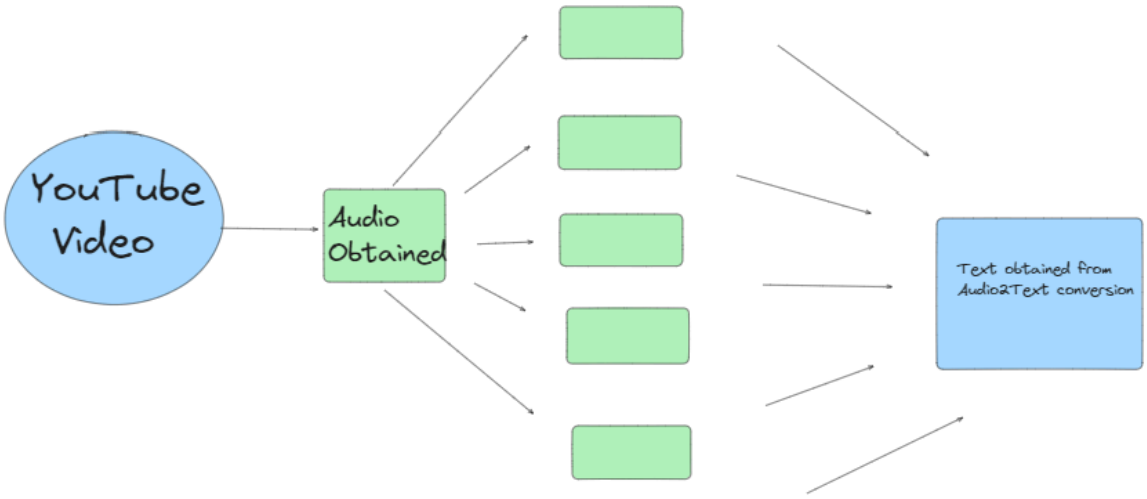


Figure: The Audio obtained from the Youtube Video is then being splitted in various .wav files which are then one by one being converted into a single text statement, the audio splitting is done in such a way that the text segmentation has a logical meaning assigned to it.

- **Using Audio Formats:** These days, there has been an increasing trend of audiobooks becoming a common source of knowledge. Hence, implementing a feature that can generate quizzes using the audio provided will prove to be quite beneficial; hence, implementing a plugin that does audio to text conversion would serve the purpose of implementing this feature.

5. Intelligent Assessment Aid (Generating Answer Keys and Summarized Notes for the Quiz)

WHAT

Generating the **answer key and summarizing notes** related to the assessment being created.

WHY

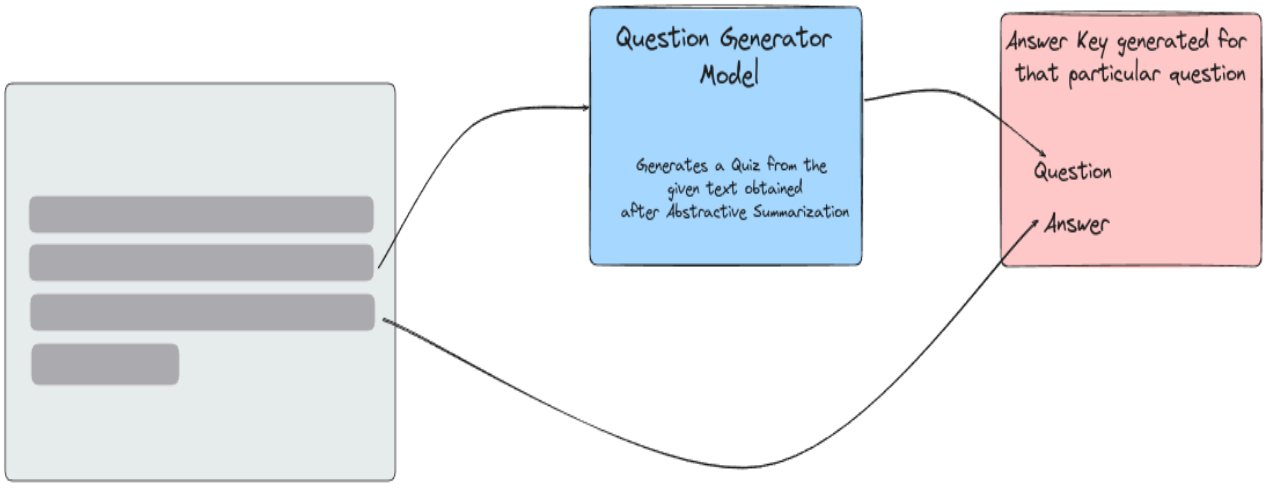
This feature is designed to empower students in their learning journey by providing them with valuable tools for preparation and self-assessment. By **automatically**

generating answer keys, students gain insight into the correct solutions, allowing them to compare their responses and identify areas for improvement. Additionally, the **summarized notes offer a concise overview of the assessment content**, enabling students to review key concepts and reinforce their understanding. This proactive approach not only enhances student preparedness but also facilitates effective self-assessment, ultimately fostering a deeper grasp of the topics being taught.

HOW

- For generating the summarized notes, we'll leverage advanced natural language processing techniques, **specifically the T5 transformer model. This model excels at abstractive summarization**, condensing input text into concise and coherent summaries suitable for student study materials. Additionally, we'll employ **Multi-Partite Ranking to rank** the generated summaries based on relevance and importance.

- In answer key generation, **the pointers derived from abstractive summarization serve a dual purpose: not only do they aid in formulating problem statements, but they also serve as the solution key for each specific problem.** By leveraging these pointers as the foundation for both question generation and solution derivation, we ensure alignment between the problems posed and their corresponding correct solutions. This integrated approach streamlines the process of generating comprehensive assessment materials, offering students clear and accurate guidance through their learning journey.



6. Generating Form-Based Quiz and AnswerKey

A **Pull Request (PR)** has been filed, presenting a method for generating a PDF file designed in a format resembling a 'FORM'. This format offers an interactive quiz experience and mimics the appearance of an answer sheet.

WHAT

Generating **questions and their answer keys in an editable form like format** that is suitable for serving as an answer sheet.

WHY:

Enhanced User Experience: The implementation of an interactive answer sheet transforms the quiz-taking process from a passive activity to an engaging and user-friendly experience. By allowing quiz takers to input their answers directly into a form-like interface, the system promotes interactivity and convenience, ultimately enhancing user satisfaction.

Immediate Score Generation: With an interactive answer sheet, quiz takers receive immediate feedback on their responses. This feature empowers learners to gauge their understanding and progress instantly, fostering a more dynamic and engaging learning experience.

Streamlined Quiz Administration: Incorporating an interactive answer sheet streamlines the quiz administration process by centralizing question submission and answer collection within a single platform.

HOW:

1. **Integration of Library**: Incorporate `**pdf-lib.js**` into the project environment, ensuring that it is accessible within the codebase and **does not violate the content security policy** of the Chrome extensions.

2. **HTML Modification**: Update the HTML file responsible for generating the quiz interface to accommodate additional elements required for interactivity. This may involve **adding input fields for titles and buttons for downloading answer keys** so that the user actions respond with a generated response

3. **JavaScript Implementation:** Modify the JavaScript file responsible for generating the quiz logic. **Integrate functions from the pdf-lib.js** library to create the answer key and questions in a form-like format. This involves utilizing functions such as **createAnswerKey()** and **createQuestions()** to map the quiz content appropriately.

7. Publishing the Extension

Deploying the FLASK Application:

We will consider deploying our FLASK application containing ML Model to production levels, which will be a necessary step for deploying our Chrome extension to the Chrome web store, so for that, depending on the scale of use and our requirements, we will be using different sets of deployment platforms and listing them here along with their cost and when to upgrade.

Local System (Initial Phase):

Advantages: For the initial phase of deployment and testing, we will be deploying our FLASK applications locally so that they can be tested with good speed.

Cost: Free of cost in the development phase

Due to local availability, we will have to upgrade to a deployable method.

Pythonanywhere (Pre-Production Phase):

Advantages: When the user base of the extension is based on a set of developers or a set of initial users, we can go well around with this deployment platform; it allows us to get deployment-level endpoints.

Cost: Free of cost

Due to the limitations on the API requests and the efficiency, there will be a need to upgrade the deployment platform to a more reliable platform that can handle a larger user base.

Heroku (Production Phase):

Advantages: When the user base of the extension is large enough so that it has to cater to a large number of requests at each moment, we should upgrade to Heroku as it provides a reliable and more efficient platform for production-level deployments. Also, the rate of a server breakdown, which hosts the FLASK application, is much lower than the previous free methods.

Cost: Approximately it will go from $5 per month to approximately $15 per month depending on the type of deployment chosen among the available options.

Amazon AWS (Enterprise Level):

Advantages: If the user base of the extension has reached a full fledged ` enterprise level, then due to reasons of user base analysis along with various features accompanying the deployment of FLASK application, like a cloud storage capacity, along with various other integration availability.

Cost: The cost of deployment of an AWS ML prediction model comes to around $25-30$ on a per-month basis, and this is based on a prediction made on the maximum expected user base and the size of the models.

Deploying the Extension in Chrome Webstore:

In order to publish our Chrome extension on Chrome Webstore, one has to become an authorized developer, and a form has to be filled out, which also involves paying a one-time fee of $5. Then we have to develop our store content along with formal documentation involving various things like justifications for the allower permissions being accessed, the level of DOM content access any script can do, and whether it invades the privacy of the user in any sense, then after a formal review process of around 30 days, we receive a reply from Google's team regarding the request to publish our extension on the store.

Timeline

Community Bonding (4 weeks):

- Establish communication channels with mentors and project collaborators.
- Discuss project goals, requirements, and expectations.
- Familiarize yourself with existing project documentation, codebase, and relevant technologies.
- Identify potential challenges and devise strategies to address them.
- Plan regular check-ins and updates with mentors to track progress and address any issues.

Phase 1 Model Enhancement and Deployment (6 Weeks)

Week 1-2: Model Modification and Optimization

- Review existing Question Generator model pipeline and identify areas for improvement.
- Implement modifications to enhance question diversity and generation speed.
- Optimize ML models by fine-tuning parameters and exploring ensemble techniques.
- Conduct thorough testing to ensure the stability and efficiency of the updated model.

Week 3-4: Flask Application Development

- Develop a Flask application to serve as the backend for the Chrome extension.
- Create endpoints for handling various functionalities, including question generation and retrieval.
- Implement authentication and authorization mechanisms to ensure secure access to the application.
- Integrate with external APIs and services as needed for data processing and storage.

Week 5-6: Hosting and Deployment

- Set up hosting infrastructure for the Flask application on a cloud platform like AWS or Heroku.
- Configure deployment pipelines for automated deployment and continuous integration.
- Perform load testing and scalability assessments to ensure the application can handle expected traffic.
- Monitor performance metrics and optimize resource utilization for cost efficiency.

Phase 1 Evaluation

Phase 2 Extension Development and Deployment (6 Weeks)

Week 7-8: Chrome Extension Development

- Design the user interface for the Chrome extension, focusing on usability and accessibility.
- Implement frontend components using HTML, CSS, and JavaScript frameworks.
- Integrate with the Flask backend to enable seamless communication and data exchange.
- Test the extension across different browsers and devices to ensure compatibility.

Week 9-10: Testing and Quality Assurance

- Conduct comprehensive testing of the Chrome extension to identify and fix any bugs or issues.
- Perform user acceptance testing with a group of beta testers to gather feedback and suggestions.
- Implement unit tests and end-to-end tests to ensure code quality and reliability.
- Refactor code as needed to improve maintainability and scalability.

Week 11-12: Deployment and Documentation

- Prepare documentation for the Chrome extension, including installation instructions and usage guidelines.
- Finalize deployment process for publishing the extension to the Chrome Web Store.
- Submit the extension for review and approval by the Chrome Web Store team.
- Address any feedback or requests for changes from the review process.

Week 13: Finalization and Wrap-Up

- Conduct a final review of the project to ensure all requirements have been met.
- Celebrate the successful completion of the project with mentors and collaborators.
- Reflect on lessons learned and areas for personal and professional growth.
- Prepare for project handover and transition, including documentation and knowledge transfer.

Final Evaluation