AI Integration with Agora

By:

Brenden Cabrera, Kyle Courounis, Hodo Duale, Ahmed Handulle, Kylie Wasserman

Table of Contents

Introduction

Currently, the world of academia relies solely on high-quality and relevant cited literature. Researchers and academics engage in the meticulous task of researching and implementing articles into their research, papers, and other written works. AI Integration with Agora recognizes the imminent need to introduce a middleware API designed to integrate AI-driven recommendations of cited articles. Agora's AI feature, Agnes, aids the user's work and research by generating cited sources based on the user's current documented writing.

Objective

Our objective is to build a middleware that utilizes OpenAI's API to assist scholars in writing academic papers by suggesting academic literature to the user based on the abstract, keywords, and existing sources of the paper they are writing.

Features

- Give users articles based on what they're writing
- Validate suggestions provided by AI using semantic scholar
- *A button that triggers a panel with the suggestions*
- Users can choose if they're writing a paper or just taking notes, changing the preciseness when calling the OpenAI API
- Ability to clipboard copy the returned sources with a click
- Users can click the link on each citation card to see the source
- Users choose the citation style: APA, MLA, Harvard, Chicago
- Ability to delete suggested citations or regenerate the full list of citations

Potential Users

The target audience of Agora as a whole is those who are in academia: students, professors, researchers, and scholars. Our specific project, AI Integration with Agora, is geared toward those who are interested in researching more about specific topics that they are currently writing about. Furthermore, our end-users of the software are those who are interested in learning more about their specific research topic.

Technology Stack

- NodeJS
- Express (Server)
- EJS (Page Rendering)
- Postgres (Database)

Challenges and Risks

1. **False/Inaccurate Information from the AI.**

   The challenge here is that the AI is not entirely reliable, so there will be cases where the AI will provide articles and information that is false or doesn't exist.

   We chose to validate every source OpenAI generates using Semantic Scholar. Using Semantic Scholar we validate the title of the source, the year and publisher, and the citation URL.

2. **Identifying if the user is writing notes or a paper.**

   The challenge here is that we do not want to generate sources and articles for the users who are just writing notes.

   Ultimately, the user selects the type of work they are writing in the Agora workspace, notes or paper. Depending on the choice, OpenAI is called differently to yield different results.

3. **Getting the information to prompt the AI.**

   The challenge here is determining where we are getting the information to prompt the AI to generate sources. The information has to come from somewhere and it needs to be sufficient enough so that the AI knows what articles to look for.

   We chose to get the information to prompt the AI through the first paragraph of at least 650 characters if it is a paper. If the user is writing notes, then all of the information that the user is writing is sent to the AI.

4. **How often to call the API.**

   The challenge here is determining how often to call the API/generate sources. If we choose something too frequent, then the risk is the cost going up and paying more for the API since it is not free.

   The solution chosen is to have a button that the user can press whenever they want to generate sources. This solution was determined as the best solution through implementation and testing.

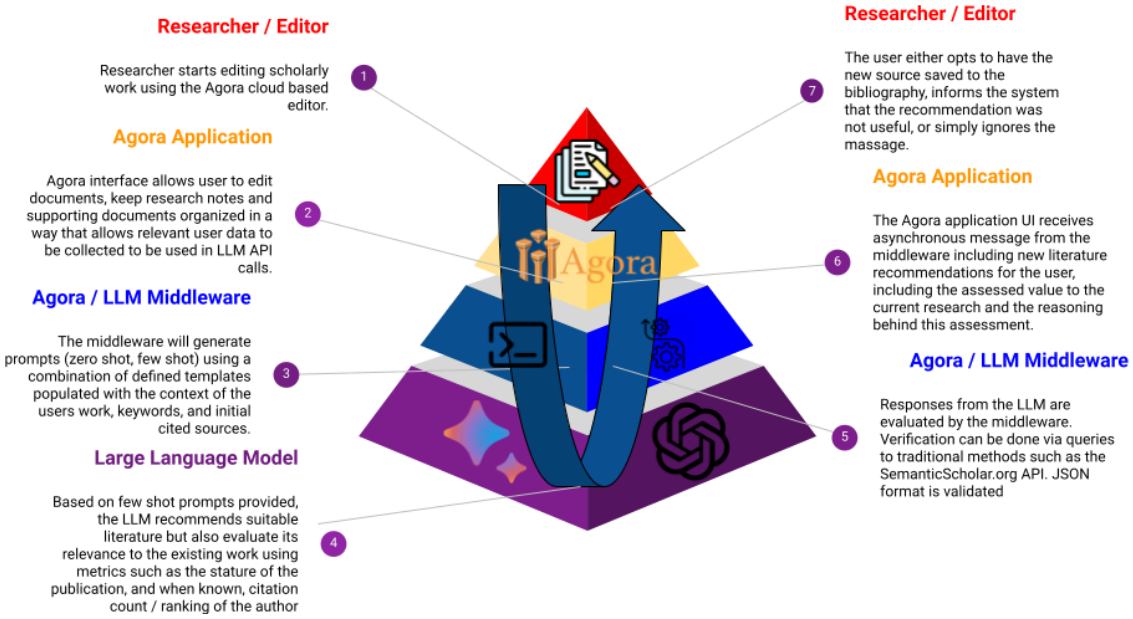
Requirements

Functional	**Description**
Literature Recommendations	Provide user's cited academic literature based on their written content
Source Validation	Validate AI-provided suggestions
Citations	Give the ability to cite recommended sources, reducing the manual effort
User-Friendly Interface	Intuitive user interface designed for a seamless user experience
Adaptive AI Prompting	Adapt OpenAI's prompts based on the user's type of work

Non-Functional	**Description**
Accuracy and Relevance	Ensuring accurate and relevant literature recommendations and sources
Performance Optimization	Minimizing latency in fetching from OpenAI's API for a seamless UX
Security and Privacy	Protecting user-generated content and interactions with the AI system
Documentation	Providing extensive documentation for effectively utilizing the AI integration feature

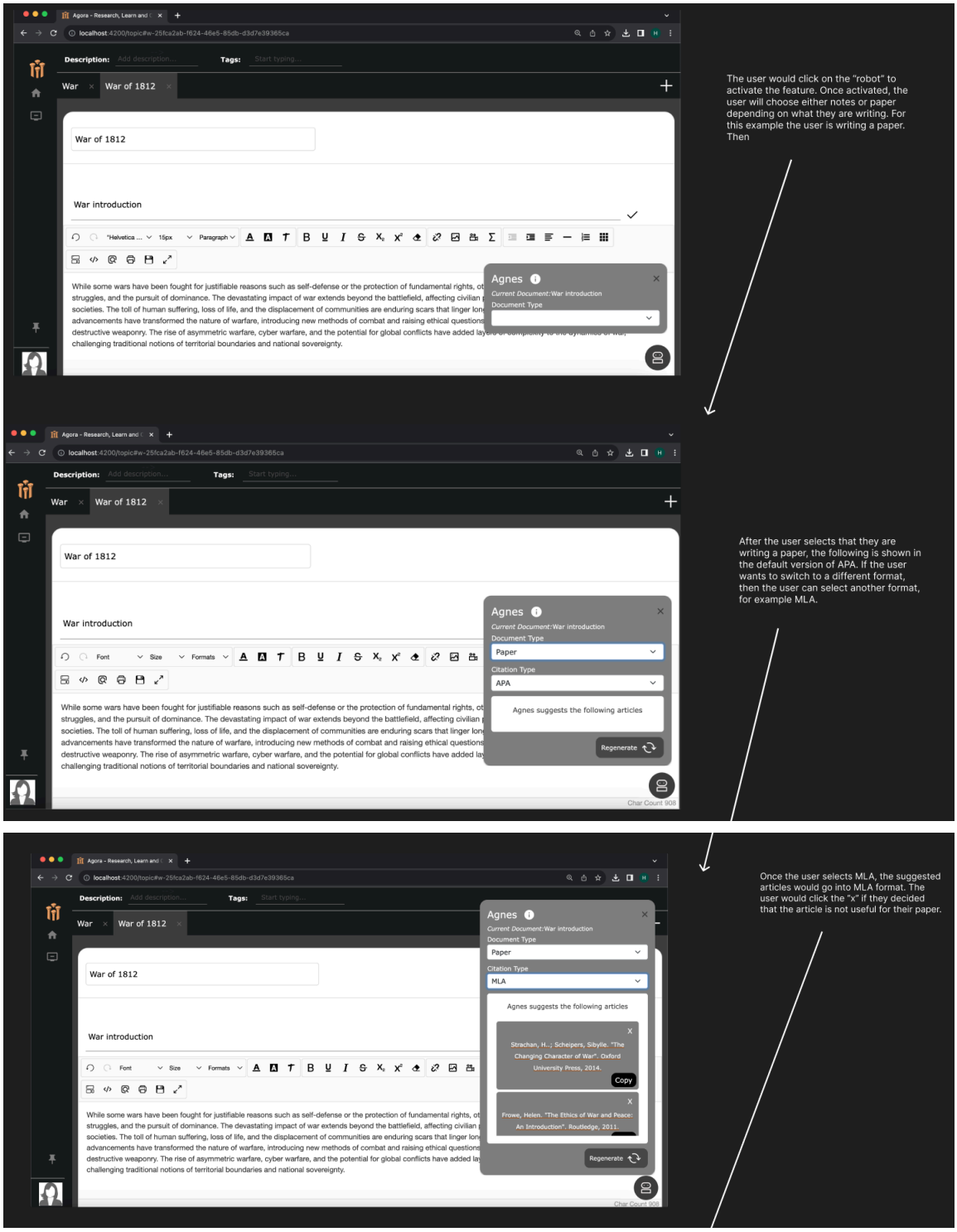
System Architecture

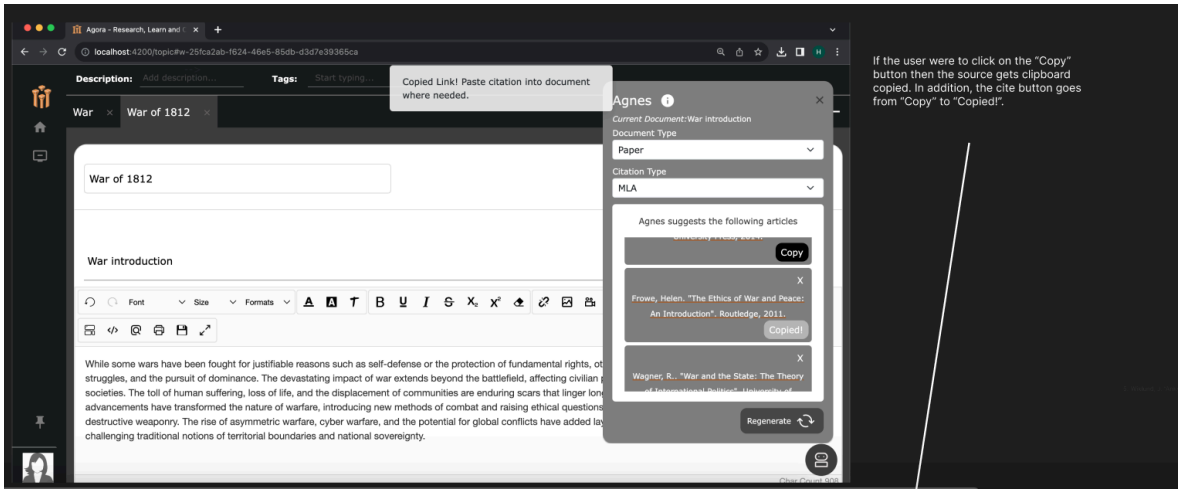
- NodeJS/Express
- EJS
- Postgres (Database)
- Figma (UI design)



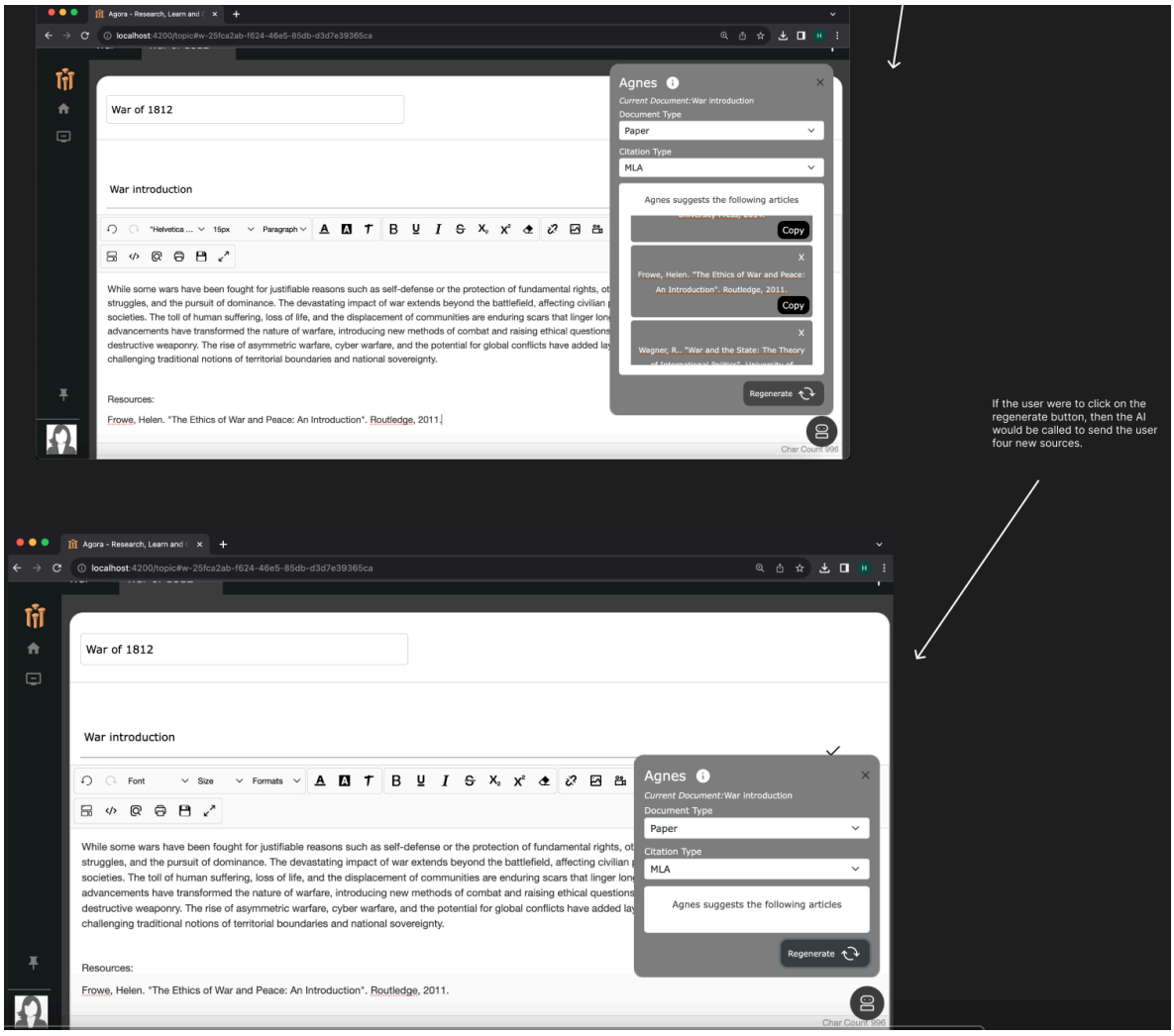
Interface Design

User Interface: Use case from the user creating a new workspace to the user adding a suggested resource from Agnes. (Reference: Figma UI Model)





If the user were to click on the "Copy" button then the source gets clipboard copied. In addition, the cite button goes from "Copy" to "Copied!".



If the user were to click on the regenerate button, then the AI would be called to send the user four new sources.

API Design

The prefix we will be using is '/ai/'. 2

Route	Parameters	Description
/ai/suggest	- mode (paper or notes) - workspace-id - selected-document	Call the server to call the OpenAI API, which generates suggestions, validates them, and then returns a JSON object with all of the valid suggestions. The workspace ID will be used to grab the workspace on the server and use its contents. The selected document parameter will specify which document in the workspace the cursor is currently in.

As for calling the OpenAI API, when the user tells the site they're writing a paper, we plan to use the few-shot prompt format that was developed from the research over the summer. If the user says they are just writing notes, we plan to develop a prompt that may provide resources on the topic at hand.

Installation and Setup

Before using Agnus there are some items to install in order to run locally.
This includes:

- Postgres
- Node.js

1. Install and configure both Node.js and Postgres.
2. To setup the database run either:
    a. "/server/db/createDatabase.sql"
    b. create a copy and modify the default "psql -U postgres -f server/db/createDatabase.sql (from the root directory of the project, default agora password is 'agora'"
3. Clone the git repo:
    a. git clone https://github.com/briangormanly/agora.git cd agora
4. Make a copy of the **.env.example** file call **.env** in the home directory
    a. Set a random session secret **SESSION_SECRET =**

    b. In the .env file add/edit the OPENAI_API_KEY and the SEMANTIC_SCHOLAR_API_KEY
    c. Save the .env file.
5. Install dependencies
    a. npm i
6. Run to start the server
    a. npm start
    b. Navigate to your browser to http://localhost:PORT [Default port is 4200].

How to Use

You can find Agnus in the bottom right corner of individual workspaces. The Agnus icon is a dark gray circle with a robot image.

Once you click on the Agnus icon, you are asked to select the document type that you are using. Select either paper or notes depending on your usage. To determine your current document, you are shown the document choice on the second line of the modal for Agnus.

To learn more information about Agnus please hover over the information icon on the first line.

After selecting the document type, you are shown the suggested articles in APA format. To change this selection, please choose another format from the list of choices under “Citation Type”.

The suggested articles are linked to the Semantic Scholar links. To get the semantic scholar link, click on the text of the suggested article.

To cite the article, click the “copy” button found in the bottom left corner of the individual suggestions. This button will copy the citation to your clipboard. You then need to paste the citation where needed in your document.

To remove a suggested article from the list of articles, click on the “x” button on the top left corner of the individual suggestions. This button removes the article from the list of suggested articles.

To regenerate the suggested articles, click on the “regenerate” button found on the bottom right of the Agnus modal. Once it is done loading, you will see your new suggested articles. Articles removed through the “x” button in earlier sessions will not be shown.

To close Agnus click on the “x” button on the top right of the Agnus modal, or click on the Agnus icon used to originally open Agnus. To reopen Agnus, click on the circle Agnus icon found on the bottom right of the screen.