SW Engineering CSC 648/848 Fall 2024

GAITORGATE

Section 02

Team 3

Team Lead: Jared Aung (yaung2@sfsu.edu)

Front End Lead: Ulices Gonzalez

Back End Lead: Andre Dargani

Github Master & Front End Engineer: Mowtee Sailan

Front/Back End Engineer: Marco Barraza

Back End Engineer: Sergio Aguilar

Department of Computer Science, San Francisco State University

Professor. Henry Villar

March 11th, 2025

History Table

Version	Date Submitted	Date Revised	Description
1.0	03/11/2025		Initial Submission

1. **Functional Requirements**

Group 1 (Highest Priority):

- **User Authentication:**
    - Users shall be able to register, log in, and manage their accounts. **(1)**
- **Search Engine Functionalities:**
    - The search engine shall accept the following as relevant queries: text (keywords, natural language). **(1)**
    - The system shall retrieve related results from the database. **(1)**
    - The system shall rank the results in order of relevancy as default. **(1)**
    - The search engine shall recommend up to 5 related queries based on the user's last searches. **(1)**
- **Filters and Customization:**
    - The system shall allow the filtering of results by date, content type, relevancy, and recency. **(1)**
    - The user shall be able to change the filters or apply multiple filters. **(1)**
- **AI-Generated Summaries:**
    - The search engine shall generate concise summaries of the search results. The summary can be made longer and more detailed. **(1)**
    - The system shall use the LLaMA 3 model (or equivalent) to power AI-based summarization. **(1)**
- **Rate and Review System:**
    - The user shall be able to rate search results from 1 to 5. **(1)**
    - Users can write reviews about search results. **(1)**
    - The system shall prioritize higher-rated search results in search ranking. **(1)**
    - The system shall store reviews (rating, comment) and link them to both the user and the specific search result. **(1)**
- **Security and Privacy:**
    - The system shall provide role-based access control for different types of users (admins, regular users). **(1)**

Group 2 (Moderate Priority):

- **User Authentication:**
    - The system shall log user sessions (login/logout times) to help with security audits and analytics. **(2)**

- **Search Engine Functionalities:**
    - The system shall allow advanced or specialized queries (e.g., market research, job searches, or local events) by integrating relevant data sources. **(2)**
- **AI-Generated Summaries:**
    - The system shall cite the source of information in the summaries. **(2)**
- **User Interface (UI) and User Experience (UX):**
    - The search results page shall highlight relevant keywords from the query. **(2)**

Group 3 (Lower Priority / Optional Enhancements):

- **User Authentication:**
    - User accounts shall be protected by multi-factor authentication (MFA) during login. **(3)**
    - Users shall be able to update their profile settings and manage them. **(3)**
- **Search Engine Functionalities:**
    - The search engine shall accept the following as relevant queries: images and audio. **(3)**
    - The system shall analyze searches to understand the intent and guide users to correct resources. **(3)**
    - The system shall allow the filtering of local results by integrating location data (local events, job searches, etc.). **(3)**
- **ChatBot (Alli):**
    - The chatbot shall assist users by refining queries or suggesting more relevant questions. **(3)**
    - The chatbot shall answer questions using information from search results. **(3)**
    - The chatbot shall suggest additional material/resources. **(3)**
    - The chatbot shall support conversation through voice or text input. **(3)**
    - The chatbot shall support multi-turn conversations. **(3)**
    - The chatbot shall summarize the search results, and the summary shall contain links to the original articles or journals. **(3)**
    - The chatbot shall give recommendations and answers, using and citing the information in the search results and real-time data. **(3)**
    - The chatbot's core AI shall be powered by the LLaMA 3 model (or equivalent). **(3)**
- **User Interface (UI) and User Experience (UX):**

    - Users shall be able to switch between different UI themes (light mode, dark mode, etc). **(3)**
- **Feature and Comparison System:**
    - Users shall be able to delete their search histories. **(3)**
    - Users shall be able to opt out of having their history stored. **(3)**
    - The system shall allow administrators to manage global settings (feature toggles, default behaviors, etc.). **(3)**
    - System settings shall be stored in a key-value format and be changeable at runtime with minimal disruption. **(3)**
    - These settings shall influence search engine behavior. **(3)**
- **Logging and Notifications:**
    - The system shall log user activity for auditing and analytics. **(3)**
    - The system shall store notifications for each user (e.g., system updates, important alerts), allowing users to view or dismiss them. **(3)**
    - The system shall maintain a user activity log for searches, feedback, or changes in preferences. **(3)**
    - Users shall be able to opt out of having their history stored. **(3)**
- **Filters and Customization:**
    - Users shall be able to switch between different UI themes (light mode, dark mode, etc.). **(3)**
- **Feature and Customization:**
    - Users shall be able to switch between different UI themes (light mode, dark mode, etc.). **(3)**
- **Feature (Rate and Review / Comparison):**
    - Users shall be able to rate search results from 1 to 5. **(3)**
    - Users can write reviews about search results. **(3)**
    - The system shall generate side-by-side comparisons for selected results. **(3)**
    - The system shall highlight key differences. **(3)**
- **Security and Privacy:**
    - The system shall provide role-based access control for different types of users (admins, regular users). **(3)**
- **Logging and Notifications:**
    - The system shall store notifications for each user (e.g., system updates, important alerts), allowing users to view or dismiss them. **(3)**
    - The system shall maintain a user activity log for searches, feedback, or changes in preferences. **(3)**
    - Users shall be able to opt out of having their history stored. **(3)**

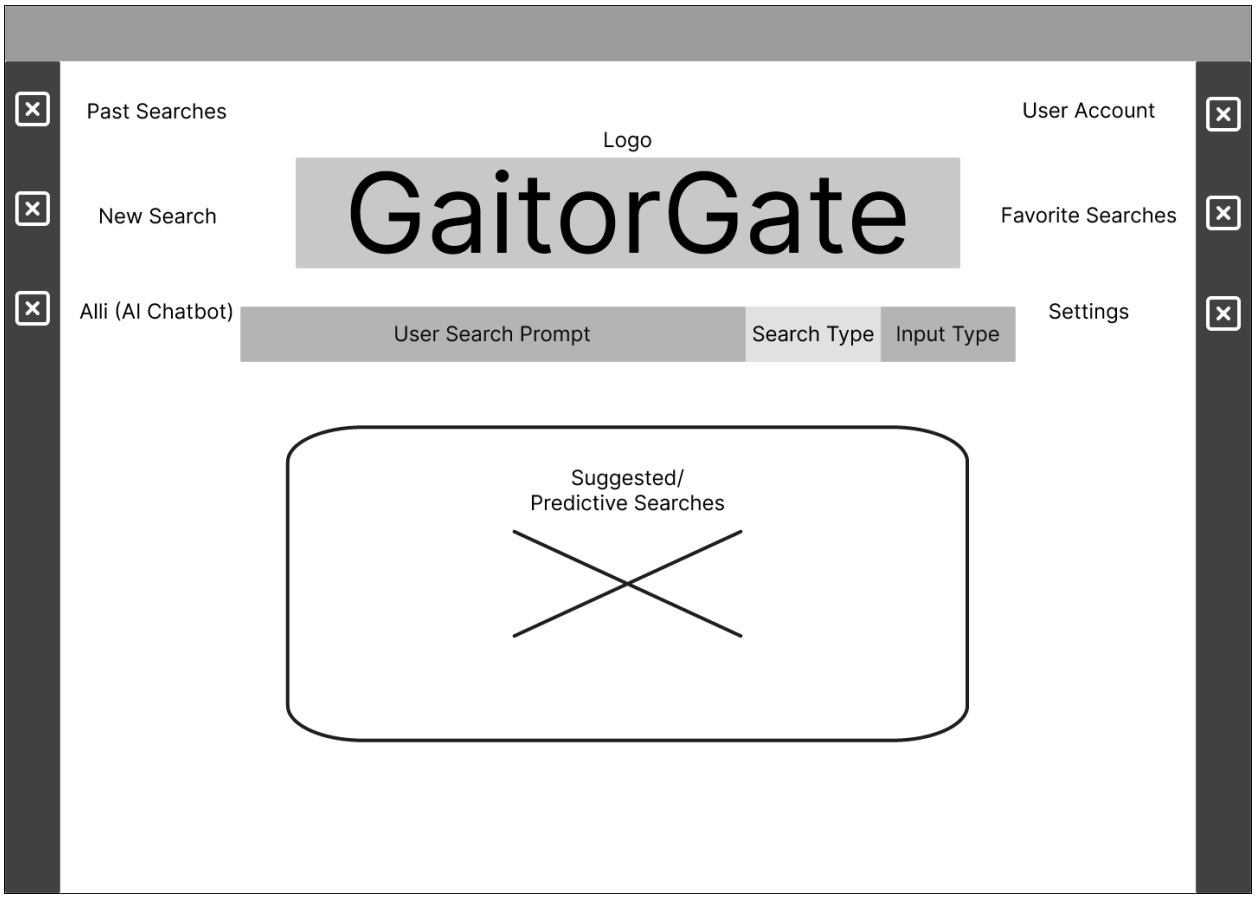
- **System Settings:**
    - System settings shall be stored in a key-value format and be changeable at runtime with minimal disruption. **(3)**
    - These settings shall influence search engine behavior, UI themes, and default behaviors. **(3)**
- **Filters and Customization:**
    - Users shall be able to switch between different UI themes (light mode, dark mode, etc.). **(3)**
- **Security and Privacy:**
    - The system shall log user activity (for auditing/analytics), respecting privacy settings. **(3)**
    - Users shall be able to delete their search histories at any time. **(3)**
- **Processing Queries and Integrations:**
    - The search engine shall accept the following as relevant queries: images and audio. **(3)**
    - The system shall analyze search queries to lead users to the correct resources. **(3)**
- **System Settings:**
    - Administrators shall manage global settings (feature toggles, defaults). **(3)**
    - Settings shall be stored in a key-value format and be changeable at runtime with minimal disruption. **(3)**
    - These settings shall influence search engine behavior. **(3)**
- **Feature and Review System:**
    - Users shall be able to rate search results from 1 to 5. **(3)**
    - Users can write reviews about search results. **(3)**
    - The system shall generate side-by-side comparisons for selected results. **(3)**
    - The system shall highlight the key differences. **(3)**
- **Search Engine Functionalities (Advanced):**
    - The search engine shall accept image and audio queries. **(3)**

1. **<u>UI Mockups and Storyboards: [Front End Team]</u>**

2.1. Landing Page

2.1.1. Page seen by user when first using or starting a new search on GaitorGate. Users can enter a prompt, change search types (by default this will be set to general search), and input type to be text input, voice, or even images for a reverse image search.

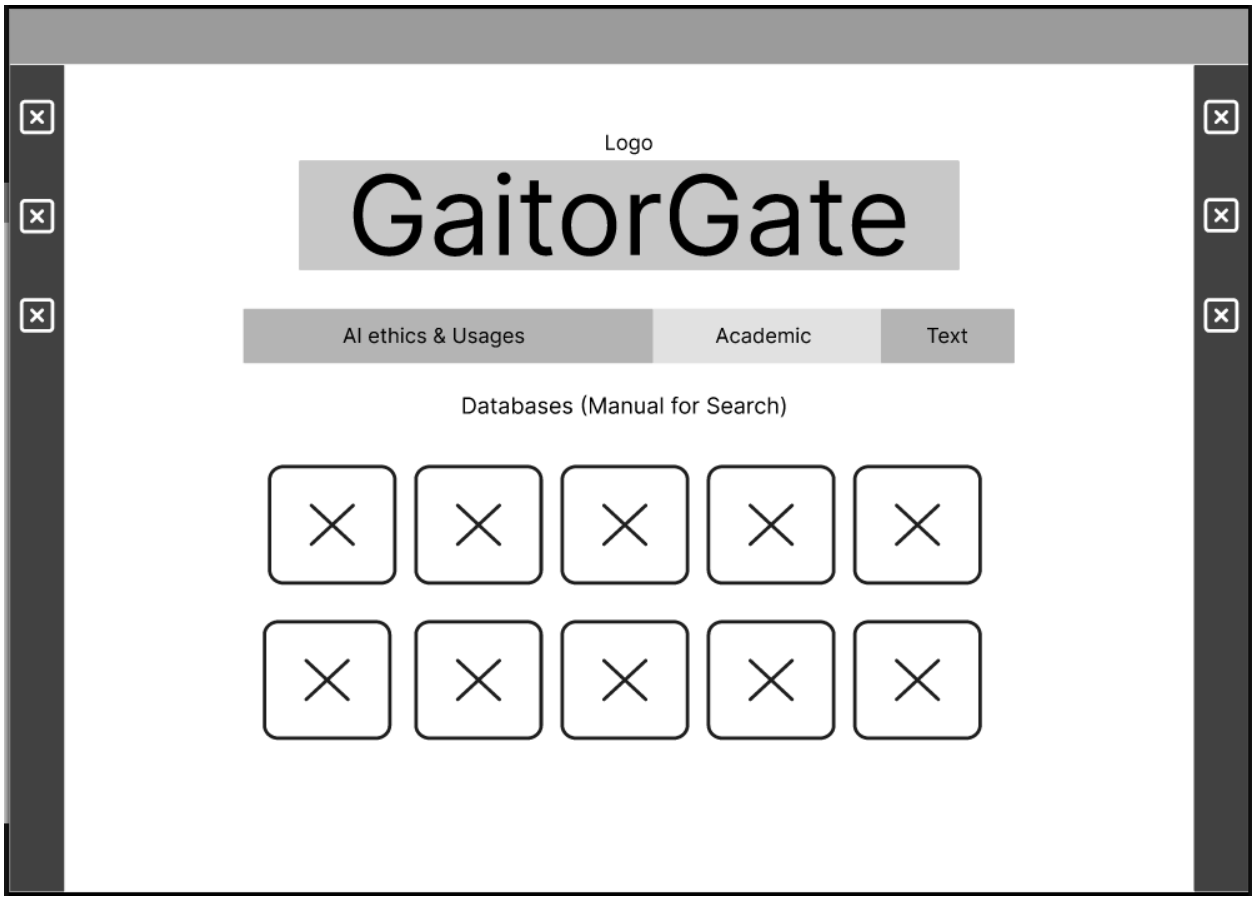
2.1.2. Users will also have button prompts to access their account and or create and log into one. Access favorite search, past searches, start a new search, change their settings, and even chat with Alli our AI Chatbot.



2.2. Academic Research Assistance

2.2.1. Article Search Page

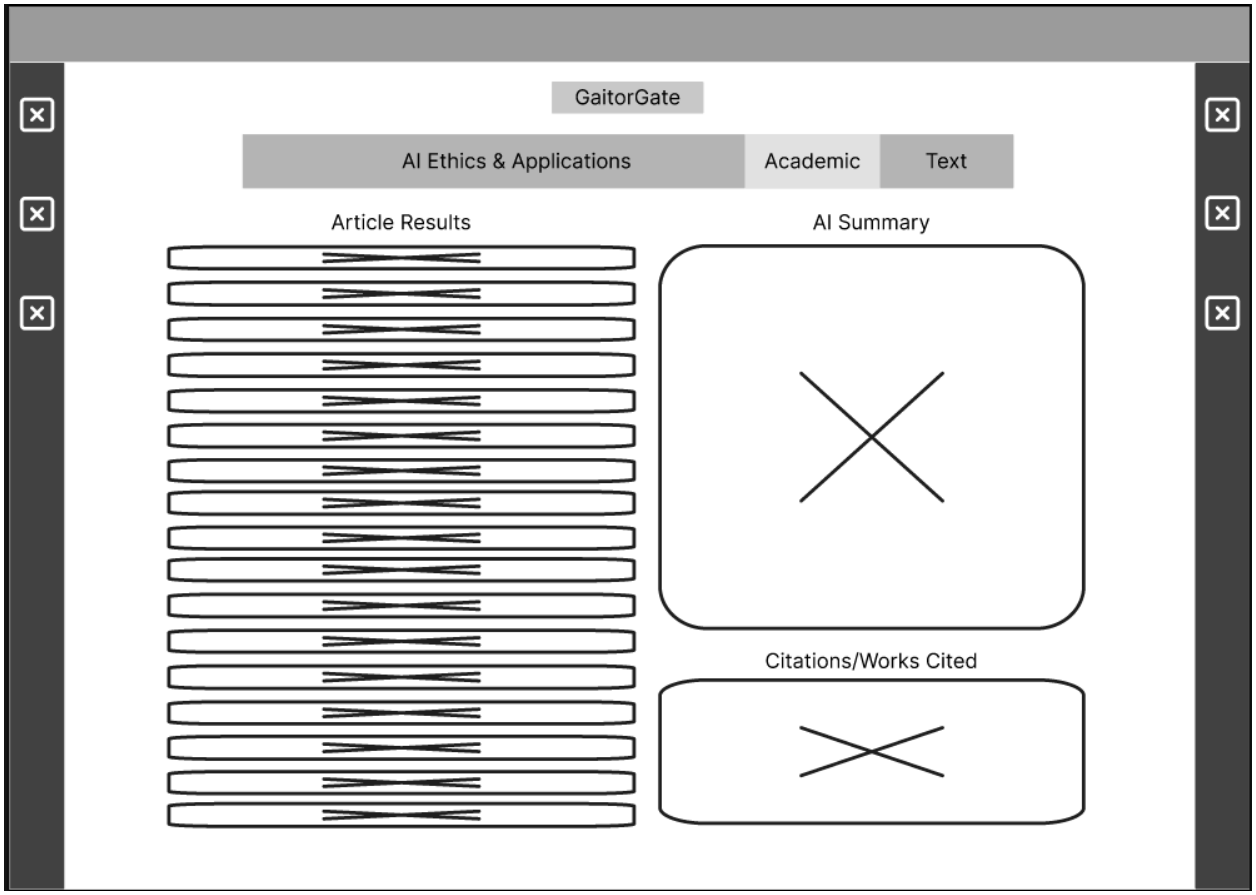
2.2.1.1. When the search type is changed to Academic for a user to find credible sources for a research project GaitorGate will adjust the landing page to show available databases for manual searches.



2.2.2. Academic Search Results

2.2.2.1. Upon entering their academic search the user will then be given a list of articles with data such as author info, publishing, date, topics, and other relevant information.

2.2.2.2. To the right of the results our integrated AI will generate a summary using the top results and will also list them below in a works cited and citations section.

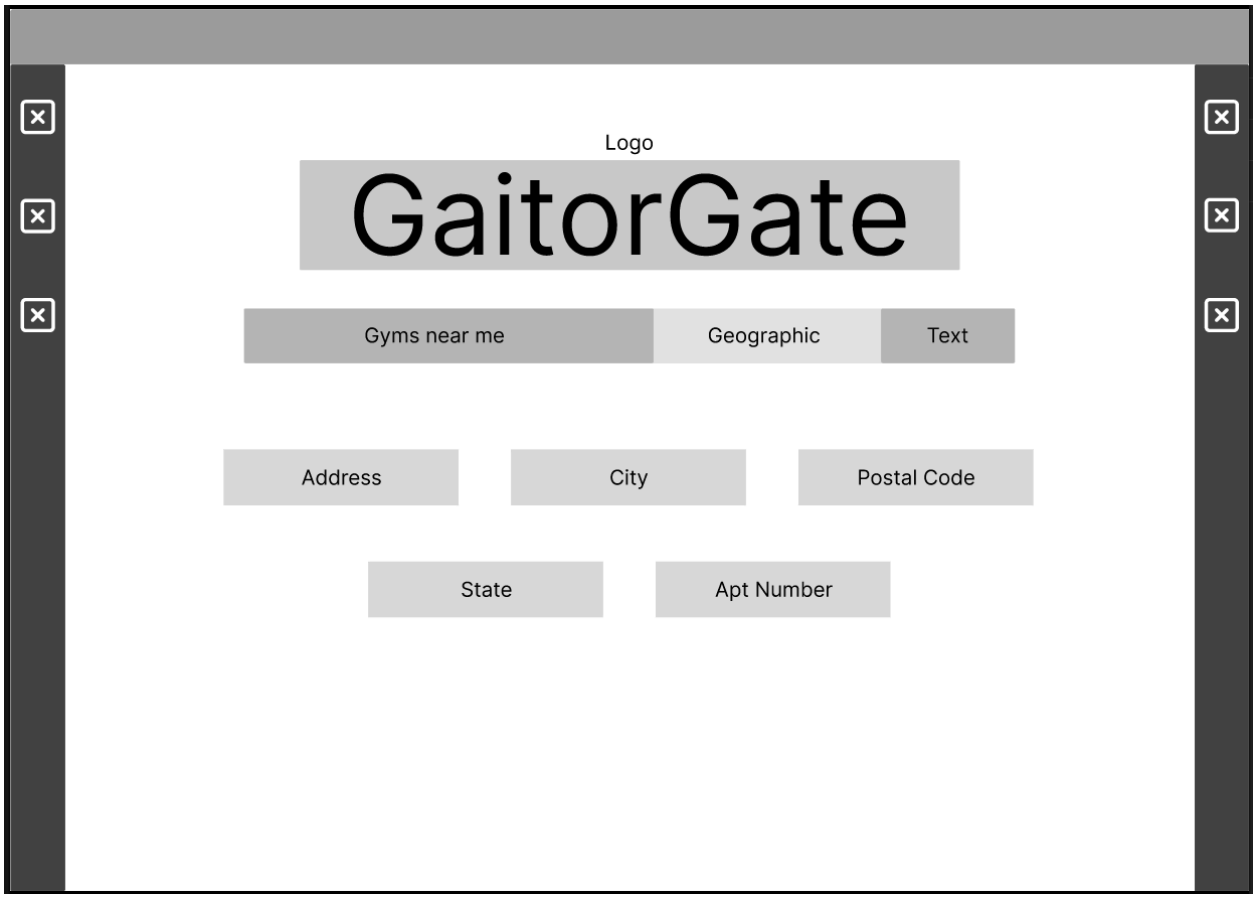


2.3. Sports Events And Fitness Facilities/Gyms Search

2.3.1. Geographic Search Page

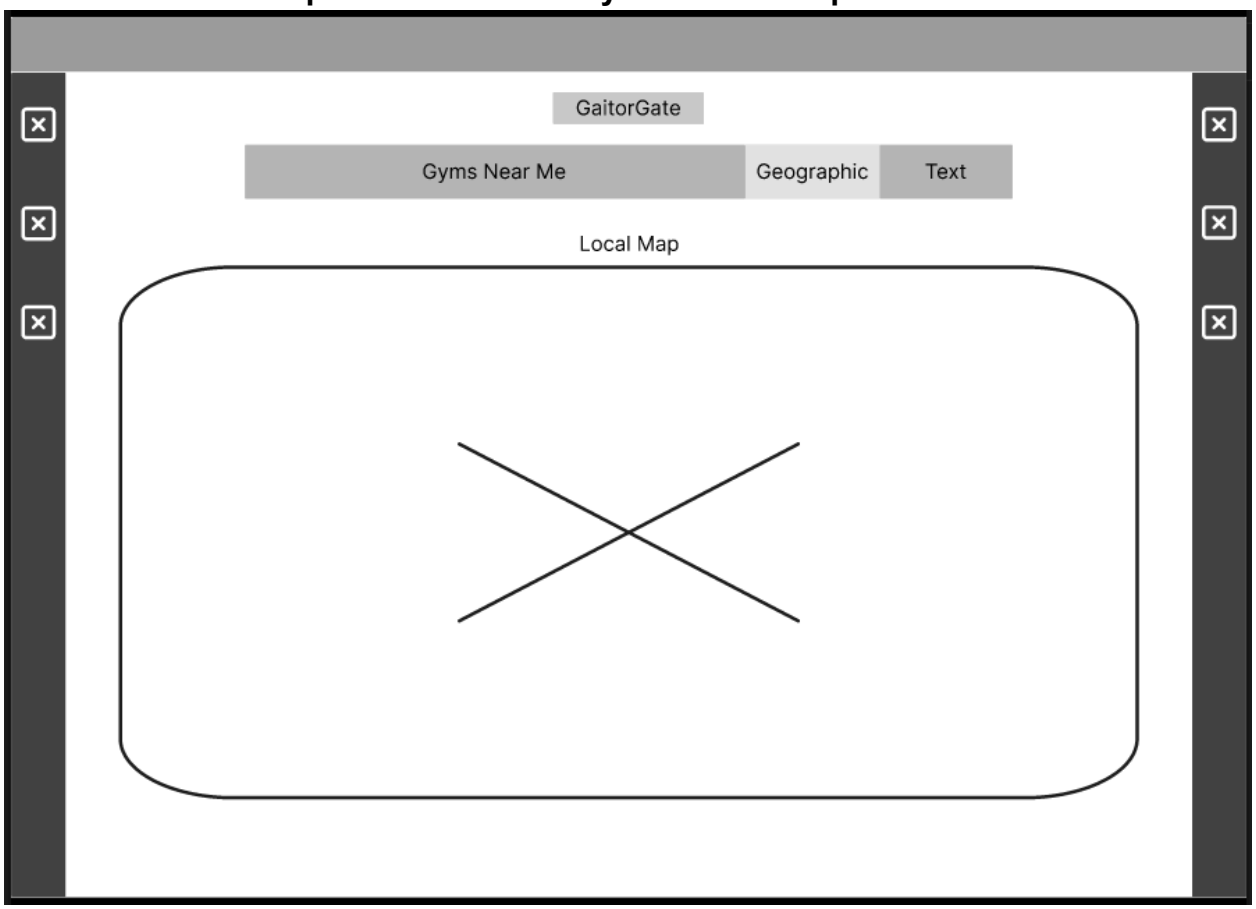
2.3.1.1. When changing the search type to geographic a user will be able to enter their address in order to help their search be localized.

2.3.1.2. A traveling sports journalist needs to find a gym and upcoming sports events to cover, they input their location and receive results when entered.



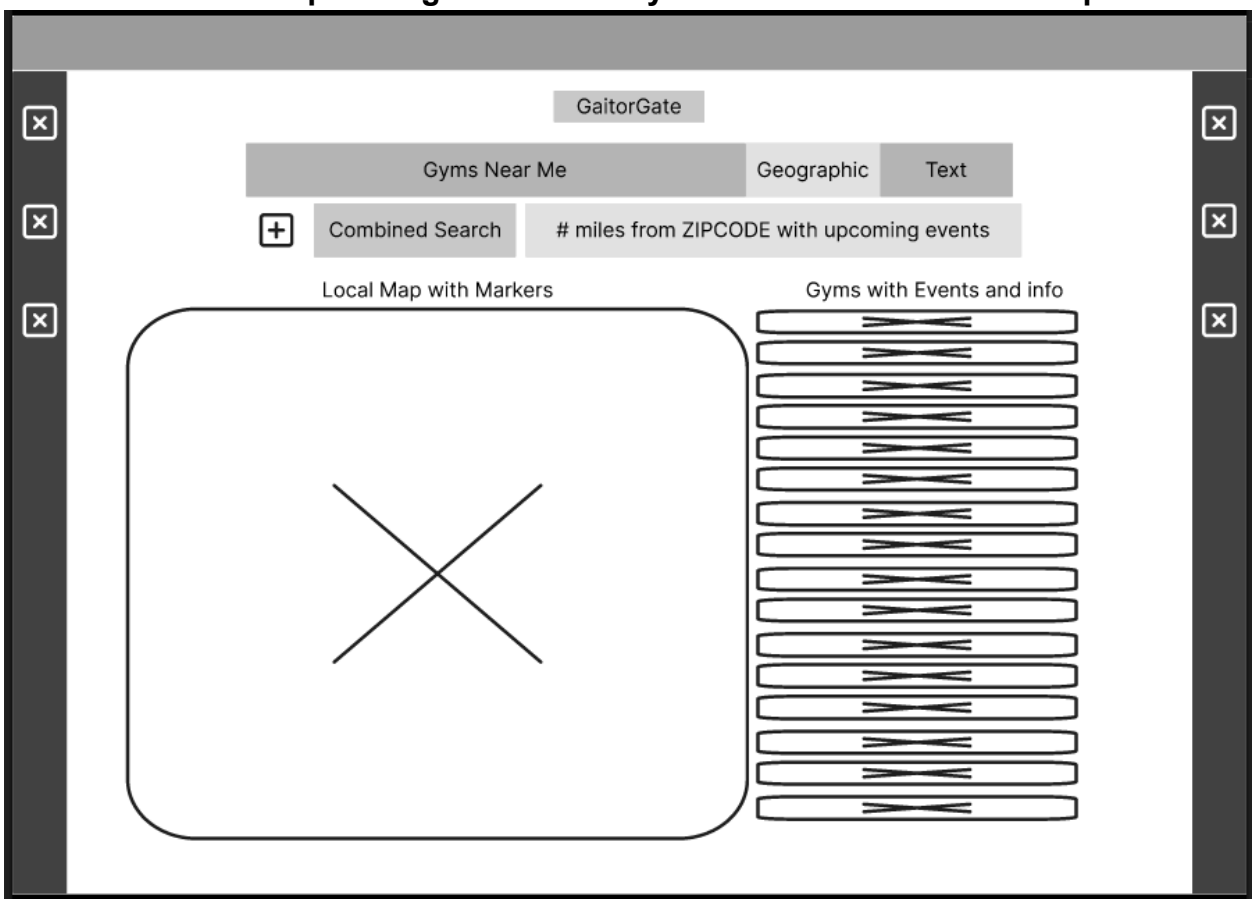
2.3.2. Geographic Search Results

2.3.2.1. After entering their search and location information they will receive AI recommendations for gyms, training centers, and sports venues nearby on a local map.



2.3.3. Geographic Search with Combined Results

2.3.3.1. After entering a combined search with a refining prompt for gyms, training centers, and sports venues nearby with upcoming events nearby with markers on a local map.

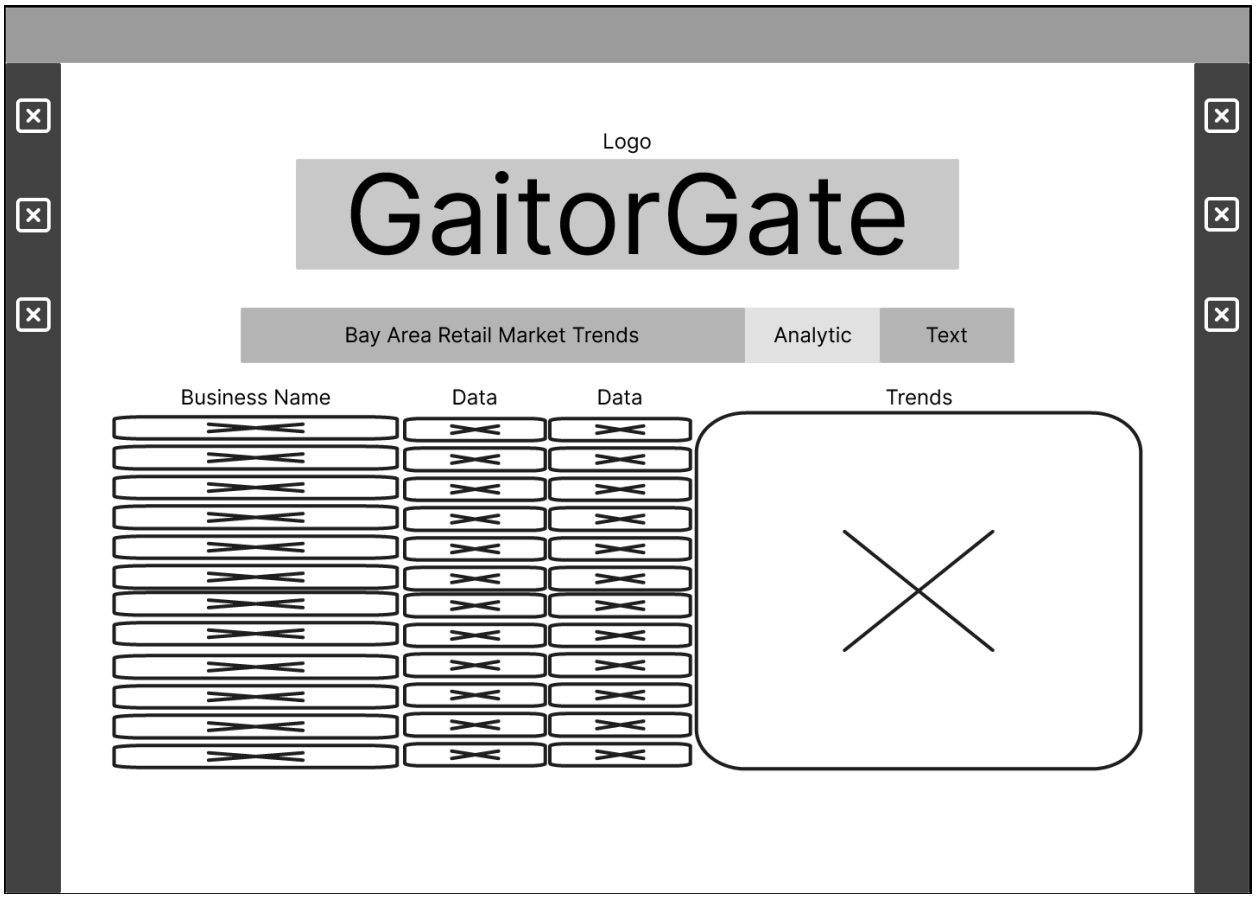


2.4. Market Research for Small Business Owners

2.4.1. Analytic Search Page

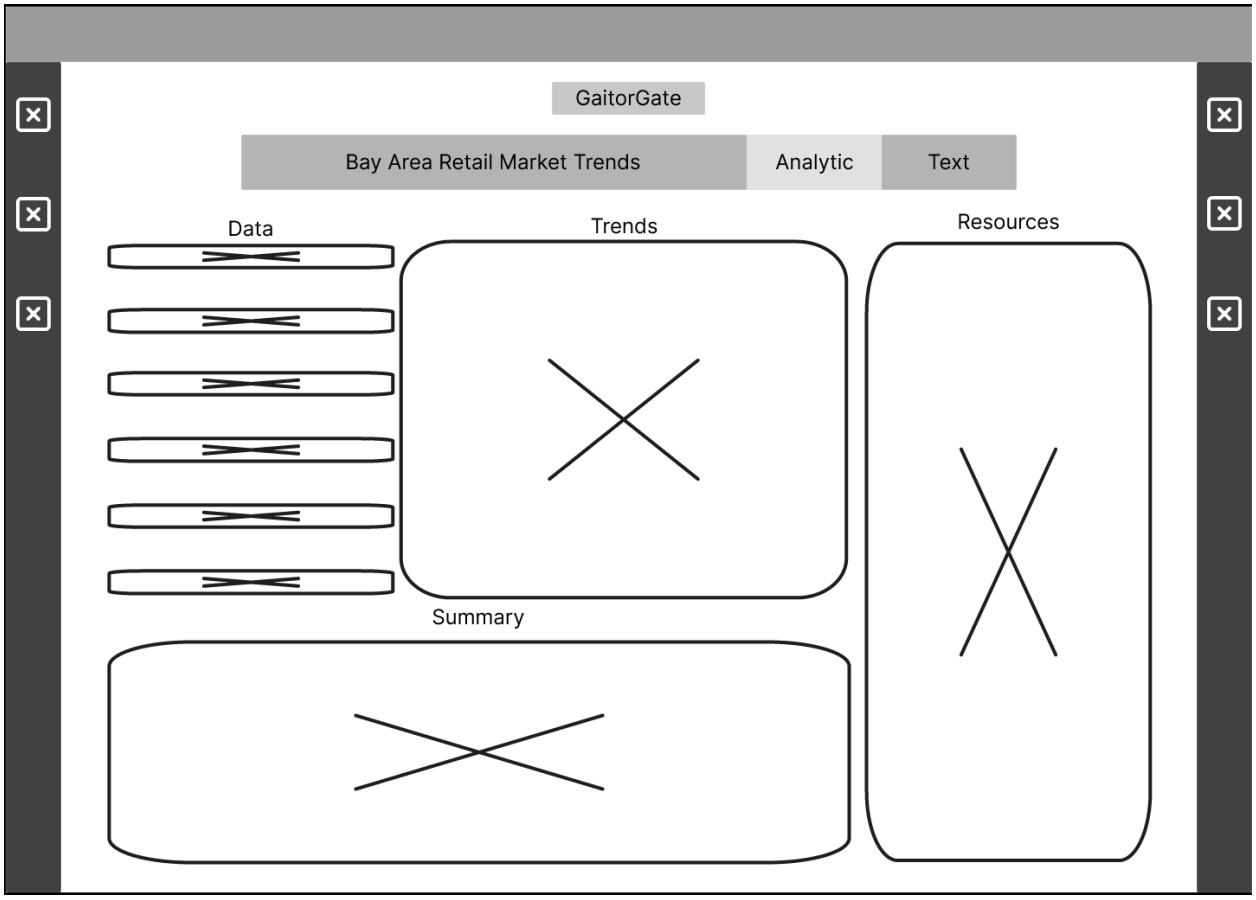
2.4.1.1. When changing the search type to be analytic the page will show general information on stocks and businesses along with a trend visualizer.

2.4.1.2. A retail business who wants to analyze market trends to improve their business can then search for market trends in their local area.



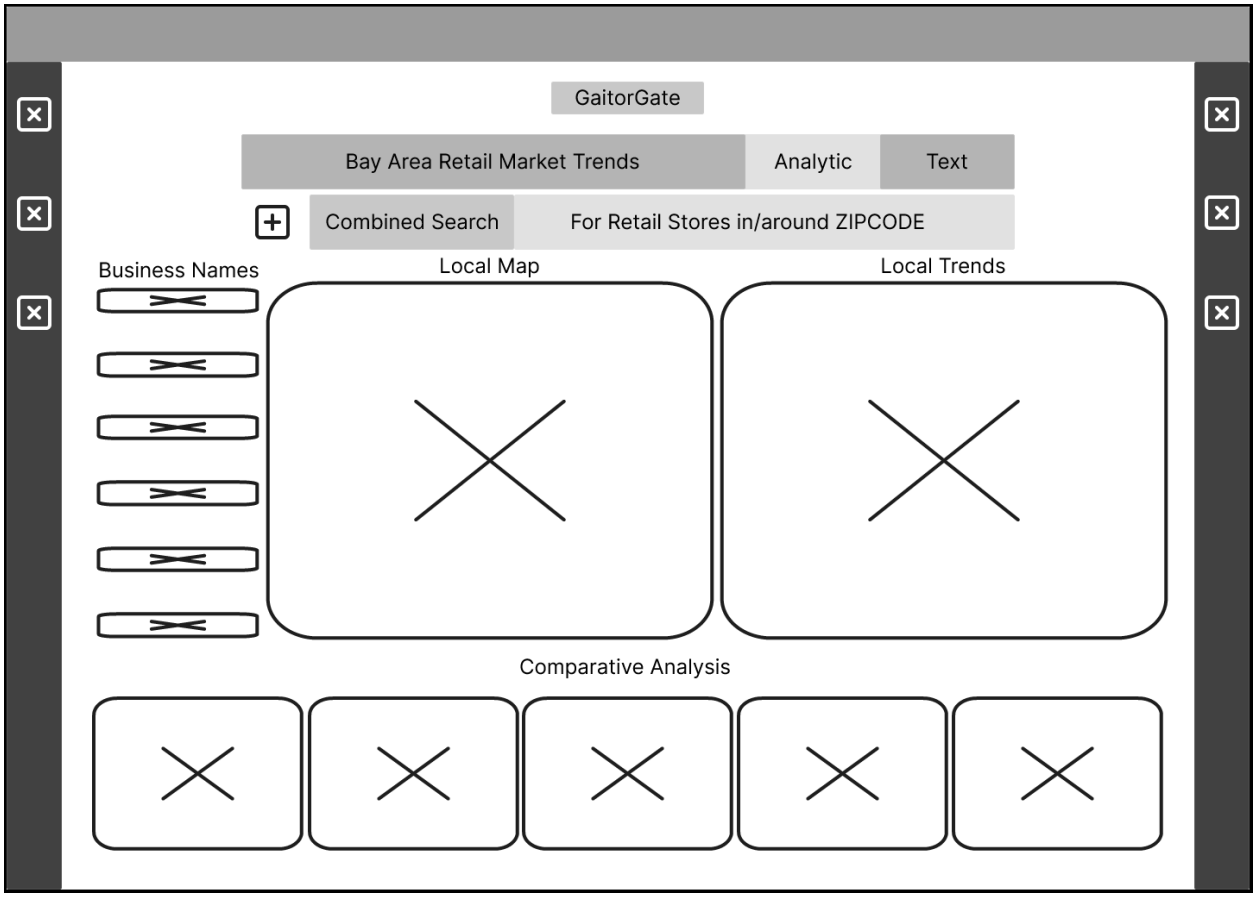
2.4.2. Analytic Search Result

2.4.2.1. Their search then provides AI reports and location-specific data which will enable informed business decisions.



2.4.3. Refined/Combined Analytic Search Result

2.4.3.1. When adding a combined search to refine it to retail stores (their competitors) in their area they will be provided more detailed information along with trends, a map, and an analysis.

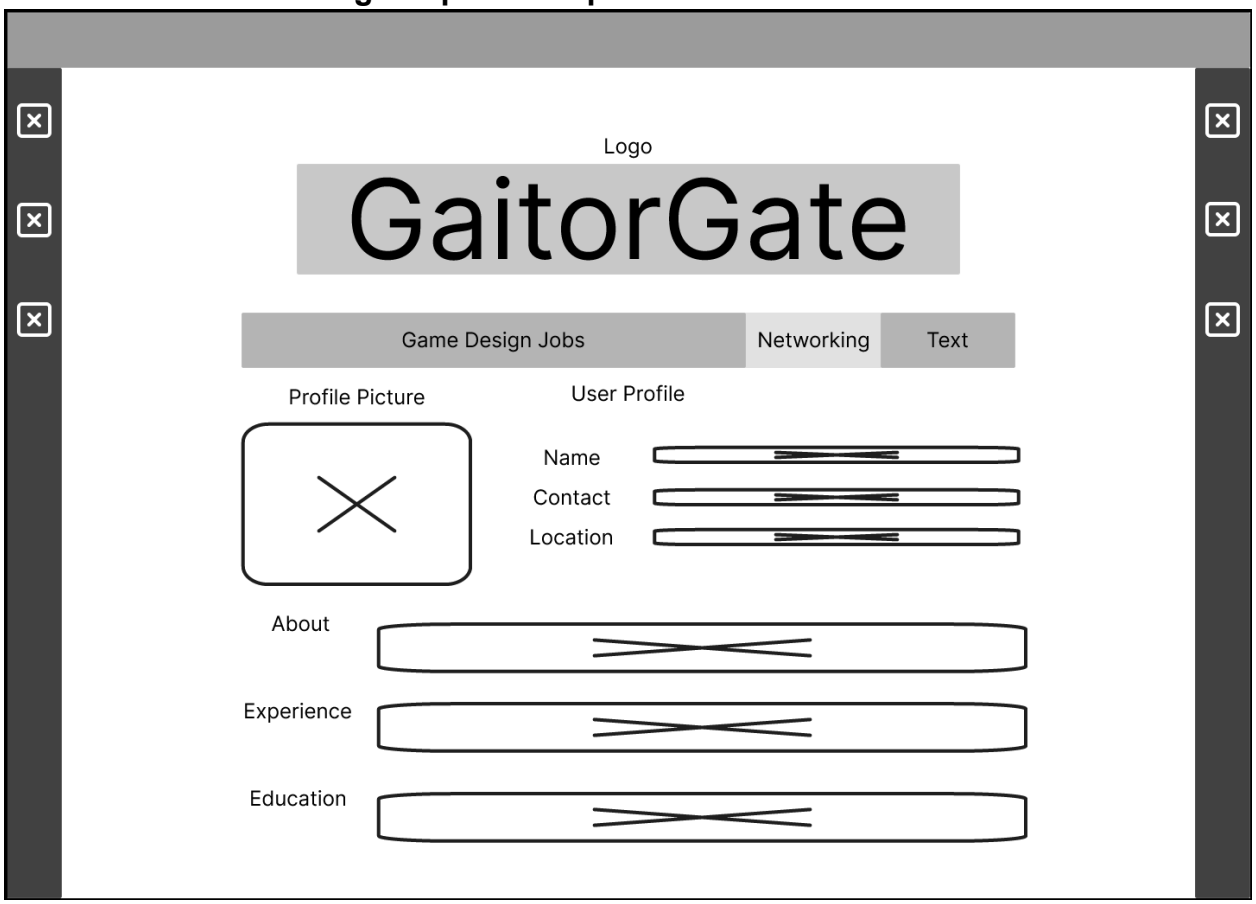


2.5. Career Search in Gaming

2.5.1. Networking/Career Search Page

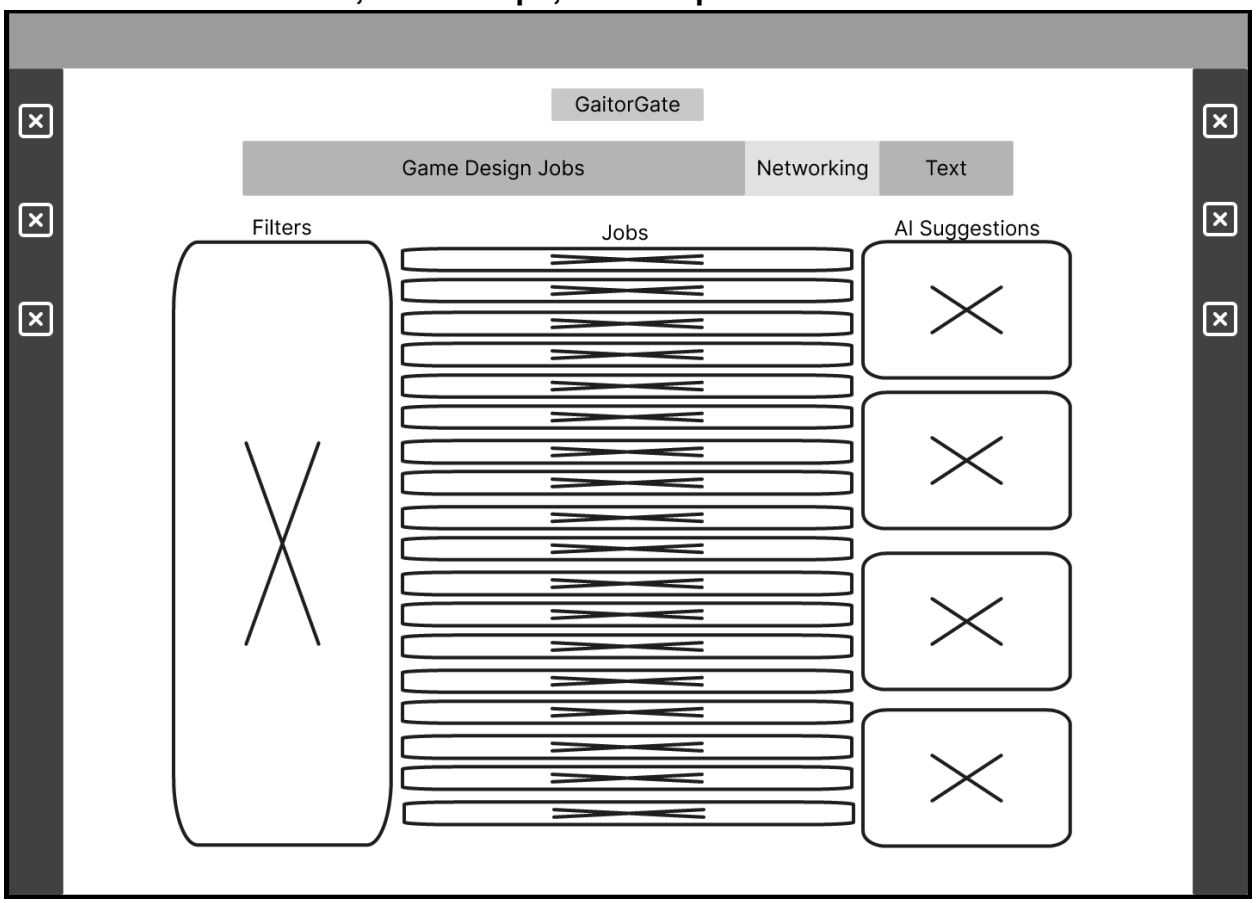
2.5.1.1. When changing the search type to networking the user will be prompted to enter information to build a professional profile/

2.5.1.2. A recent college graduate who is very passionate about game development wants to jump right into the job market post-grad so when prompted she can then search for Game Design jobs using her profile to provide better results.



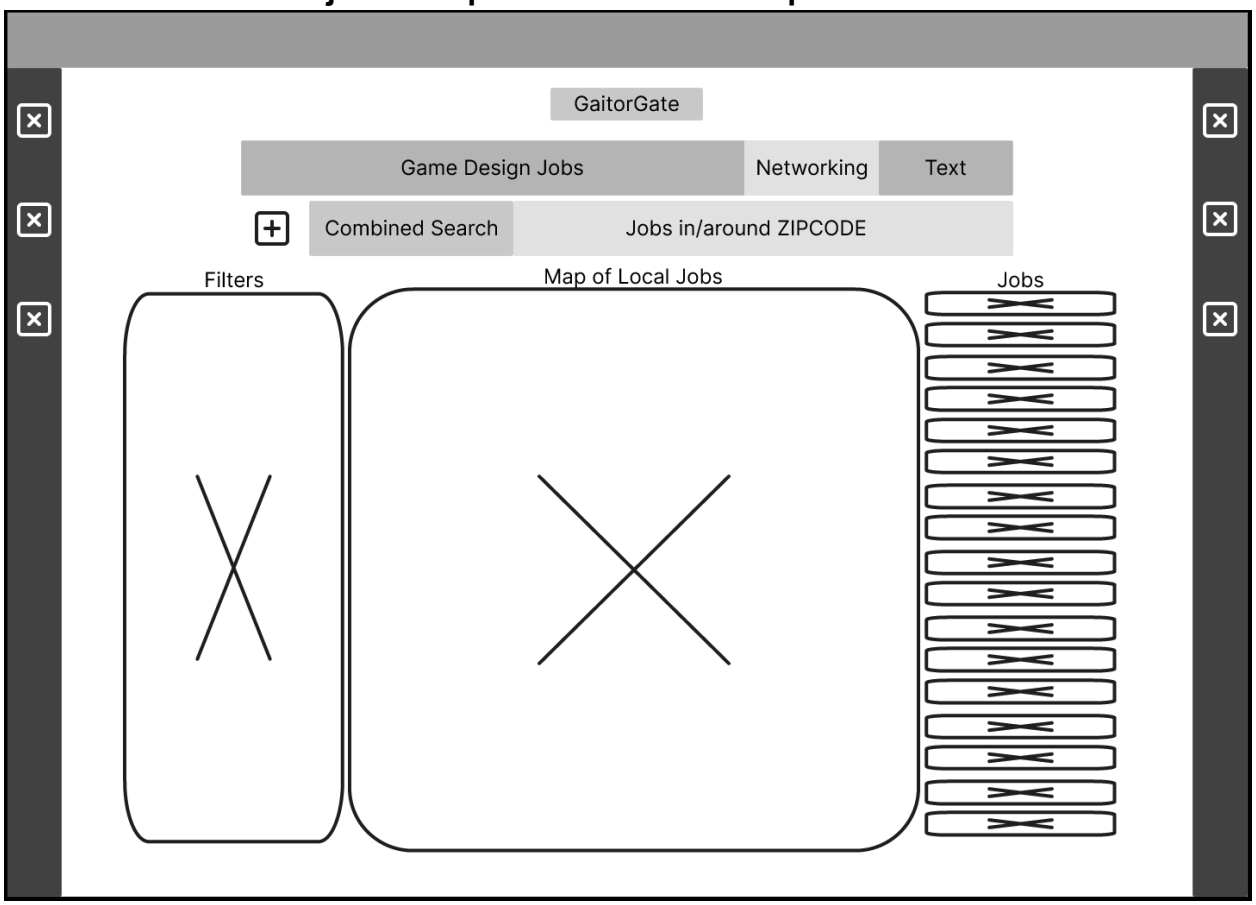
2.5.2. Job Results with Filtering

2.5.2.1. After entering her search she will then be given results based on industry trends, job postings, upcoming gaming career fairs, resume tips, and her profile.



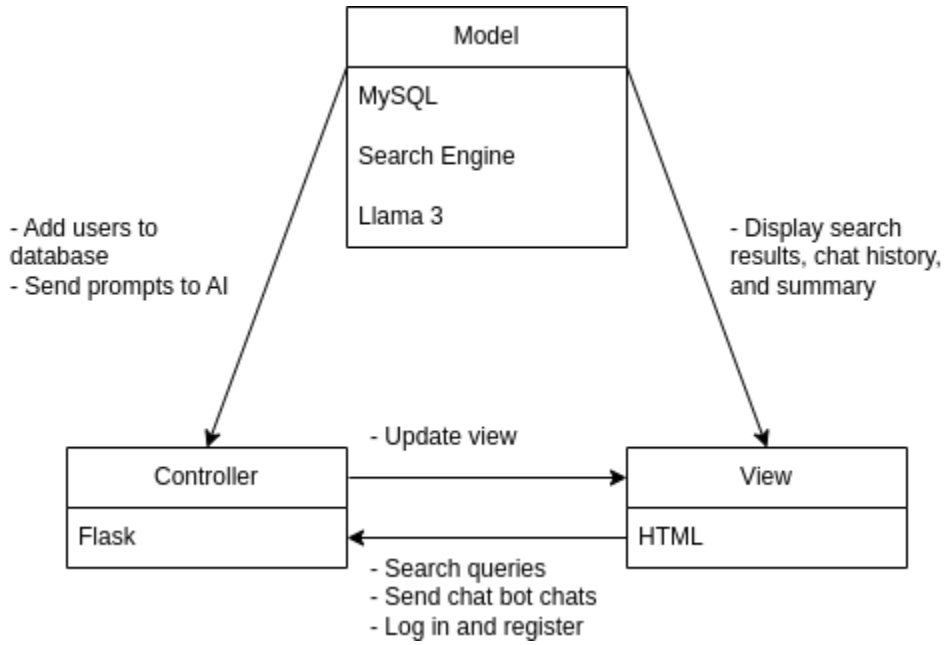
2.5.3. Local Job Results with Map

2.5.3.1. Lastly using a combined search she can then refine it to local jobs that will show markers on her local map along with a list of jobs and possible filters to help her refine further.



2. High Level Architecture

MVC Diagram



API Routes

Authentication:

- POST /auth/signup
- POST /auth/logout
- POST /auth/login

Queries:

- GET /search?query={QUERY}&filter={FILTER}
- POST /summarize

AI Chat:

- GET /chat
- POST /chat/message

Rating:

- POST /rating
- GET /rating

Chat AI Model:

- LLaMA 3

3. **Database Organization [Jared]**

1. Database Functional Requirements

User & Authentication

1. Users (Strong)
    a. A user shall perform zero or many search queries
    b. A user shall be classified as a guest or registered user.
    c. A user shall have a userID and the timestamp for when it was created.

2. Registered_User (Strong)
    a. A registered user shall be subclass of User
    b. A registered user shall have a unique email.
    c. A registered user shall have a password.
    d. A registered user shall write zero or many reviews.

3. Guest (Weak)
    a. A guest shall be a subclass of User.
    b. A guest shall be able to perform searches.
    c. A guest shall not have the same privileges (user profile, chatbot access, search history) as a registered user.

Search Engine Core Entities

4. Search_Query (Strong)
    a. A search query shall have a unique query text.
    b. A search query shall be performed by zero or many users.
    c. A search query shall retrieve zero or many indexed results.
    d. A search query shall have zero or many applied filters.

5. Index (Strong)
    a. An index shall reference one document.
    b. An index shall have zero or many keywords, tags.
    c. An index shall have zero or one category.
    d. An index shall be retrieved by zero or many queries.

6. Document (Strong)
    a. A document shall contain a URL, title and timestamp of when it was created.
    b. A document shall be referenced by zero or one index.

7. Filtering
    a. A filter shall be applied to zero or many queries.
    b. A filter shall be linked to many indexes.

10.1. Keywords (Weak)
    a. A keyword shall have one or many indexes.
10.2. Tags (Weak)
    a. A tag shall have one or many indexes.
10.3. Category (Weak)
    a. A category has one or many indexes.

2. Entities and Attributes

1. User
    a. UserID (PK, NN, AI (auto increment)) : INT
    b. Create_at : (Timestamp(Default to Current Time))
    c. User_type : ENUM (‘guest’, ‘registered user’)

2. Registered User
    a. UserID (PK, NN, AI (auto increment)) : INT
    b. Email (UQ,NN) : VARCHAR (40)
    c. Password (NN) : VARCHAR (40)
    d. Create_at : (Timestamp(Default to Current Time))
    e. User_type : ENUM (‘guest’, ‘registered user’)

3. Guest
    a. UserID (PK, NN, AI (auto increment)) : INT

4. Search_Query
    a. queryID (PK, AI) : INT
    b. Query_text (UQ, NN) : TEXT
    c. userID (FK) : INT
    d. created_at : TIMESTAMP

5. Index
    a. indexID(PK,AI) : INT
    b. documentID (FK, UQ, NN) : INT
    c. categoryID (FK, NN) : INT
    d. Created_at : TIMESTAMP
6. Document
    a. documentID (PK): INT
    b. URL (NN): TEXT
    c. Title (NN) : TEXT
    d. created_at : TIMESTAMP
7. Filter
    a. filterID (PK) : INT

    b. Filter_type : ENUM(‘keyword’, ‘tag’, ‘category’)
    c. Filter_text (UQ, NN): TEXT
8. Category
    a. FilterID (PK, FK to Filter) : INT
    b. Category_name : VARCHAR (255)

9. Keyword
    a. FilterID (PK, FK to Filter) : INT
    b. Keyword_name (UQ, NN) : VARCHAR (255)

10. Tag
    a. FilterID (PK, FK to Filter) : INT
    b. Tag_name (UQ, NN) : VARCHAR (255)

11. Review
    a. Review (PK, AI) : INT
    b. userID (FK to Registered User) : INT
    c. indexID (FK to Index) : INT
    d. Rating : INT
    e. Review_Text: TEXT
    f. created_at : TIMESTAMP

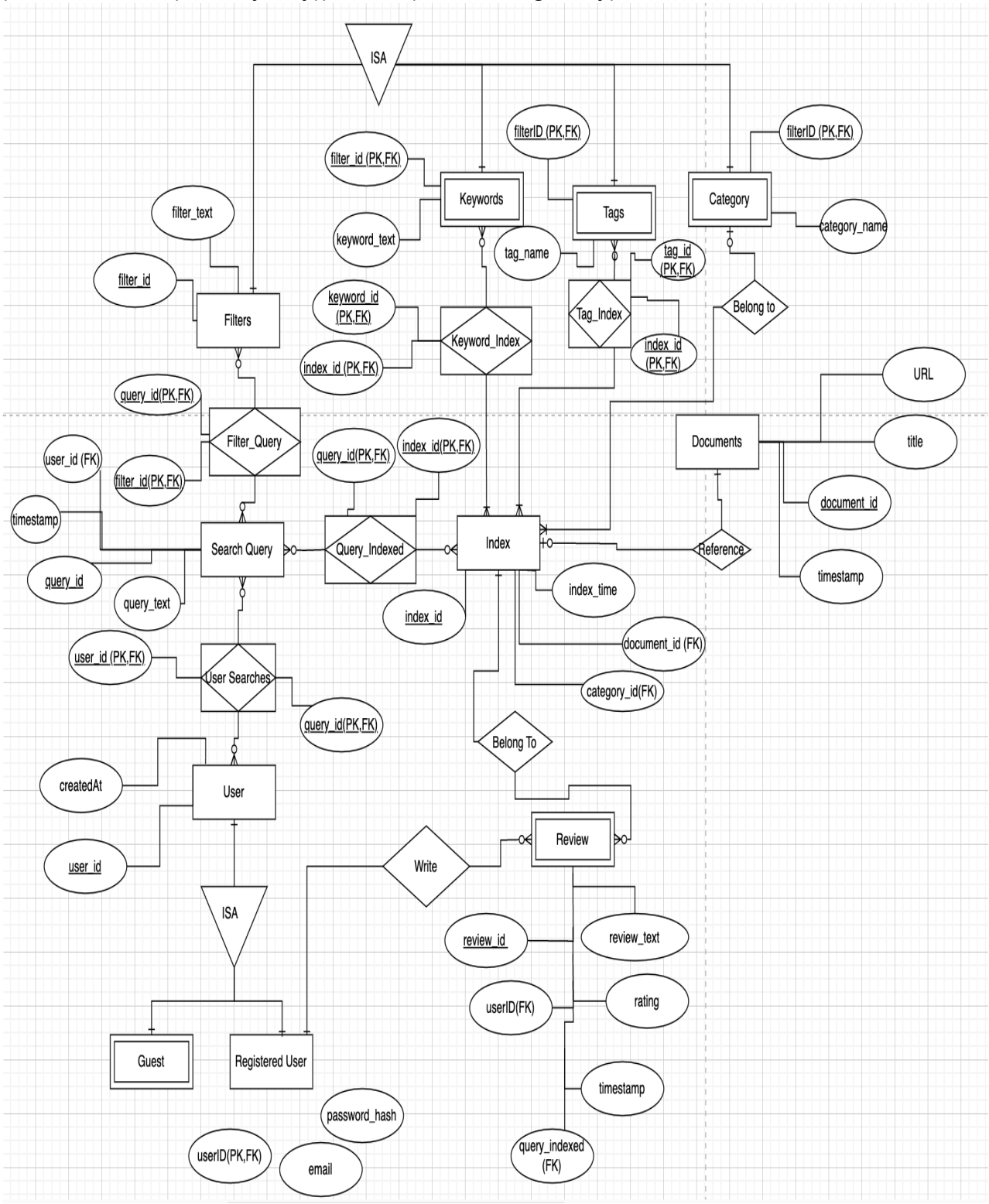
12. Query_Indexed
    a. queryID (PK, FK to Search Query) : INT
    b. indexID (PK, FK to Index) : INT

13. Query_Filtered
    a. queryID (PK, FK to Search Query) : INT
    b. filterID (PK, FK to Filter) : INT

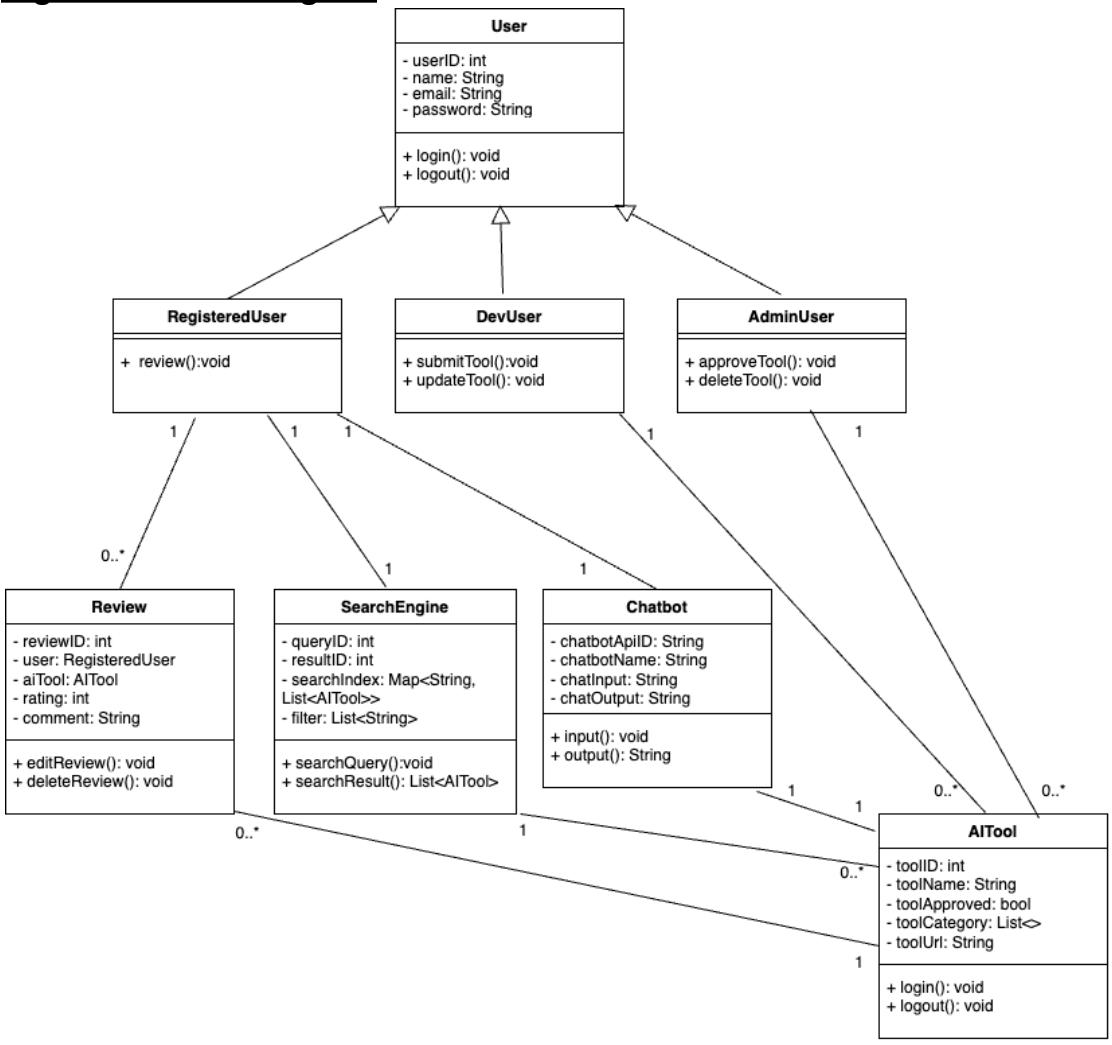
14. Indexed_Keyword
    a. indexID (PK, FK to Index) : INT
    b. keywordId (PK, FK to Keyword) : INT

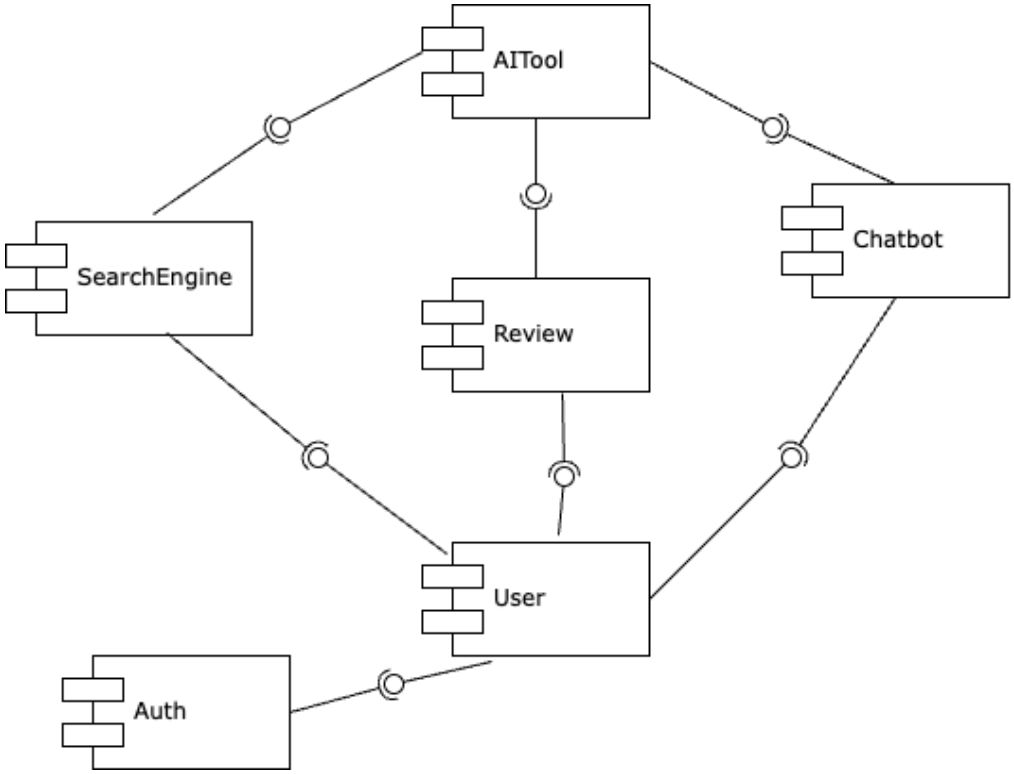
15. Indexed_Tag
    a. indexID (PK, FK to Index) : INT
    b. tagID (PK, FK to Tag) : INT

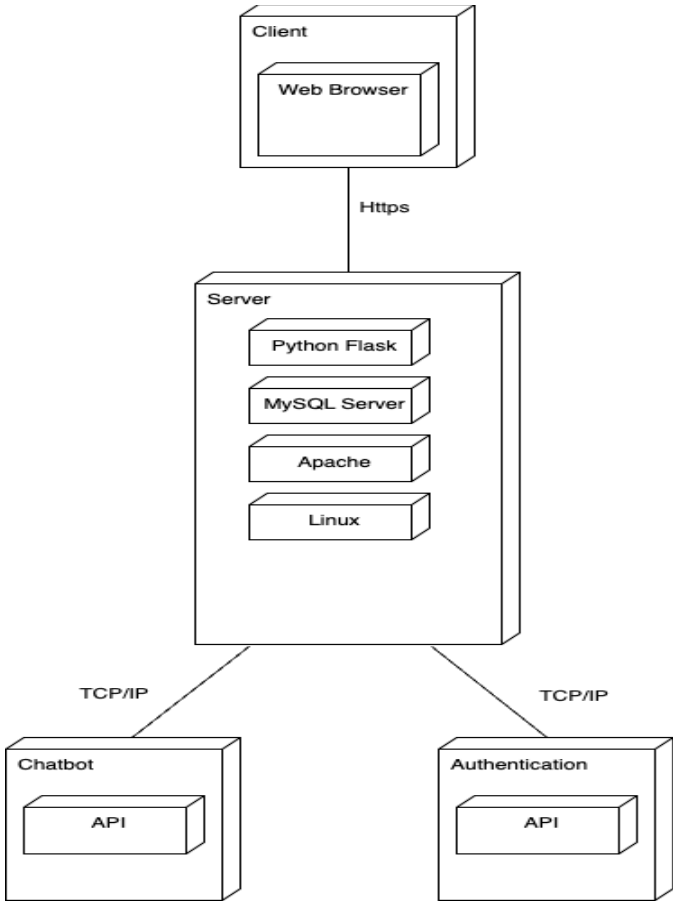
Description: Entity Relationship Diagram (ERD) for GaitorGate Database.
(underline = PK(Primary Key)) (FK = Foreign Key)



4. High Level UML Diagram




AITool
SearchEngine
Review
Chatbot
User
Auth


Client
Web Browser
Https
Server
Python Flask
MySQL Server
Apache
Linux
TCP/IP
TCP/IP
Chatbot
API
Authentication
API

5. **Identify actual key risk for your projects at this time**

- As of right now our front end deployment can serve users pages with styling as per usual HTML and CSS standards do however due to our functional requirements we will have a lot of assets that will need to be created and stored which will increase the workload of the server in order to populate and support our site.
    - A solution to this will be the use of handlebars/Jinja
- Jinja is still not entirely implemented to make use of Handlebars alternatives, this forces us to hard code many front end elements which until further notice may lead to conflicts in our GIT Repository especially during development and testing.
    - A solution that is planned for this is to create a Jinja branch after we deploy our vertical prototype in order to test and make sure Jinja runs while also making sure it can populate pages just as Handlebars would. From here we can then merge onto our development branch.
- The team has limited experience integrating and working with AI APIs, such as LLama3.
    - A solution would be to read up on documentation and watch tutorials on how to work with the API.
- Potentially getting behind schedule due to unpredictable bugs or meeting conflicts.

  A solution would be to have a clear timeline with extra time for testing. Breaking the work down into realistic chunks while having weekly meetings to track everyone's progress to make sure we are on track.

6. **Project Management**

In our group, we have decided to try to split up the work as fairly as possible. Tasks were assigned a few days after the last milestone so that everyone had plenty of time to work on their parts. The UI mockups and storyboards were assigned to the front-end team. And the rest were assigned as one person per topic. As the back-end didn't need as much collaboration as the front-end team, the back-end team worked more independently for this milestone. But in the future, we will try to find ways to meet more often and encourage collaborations and sharing ideas. We will also find a much more effective way to track everyone's progress through the milestone by using deadlines and frequent check-ins.