Table of Contents

[**Introduction** 1](#_Toc170562878)

[**Aim** 1](#_Toc170562879)

[**Targeted** **Audience** 1](#_Toc170562880)

[**Use/Problem that it solves** 1](#_Toc170562881)

[**System Overview** 2](#_Toc170562882)

[**Requirements** 2](#_Toc170562883)

[**Functional Requirements:** 2](#_Toc170562884)

[**Non-Functional Requirements:** 2](#_Toc170562885)

[**External Interface Requirements** 3](#_Toc170562886)

[**Limitations** 3](#_Toc170562887)

[**System** **Architecture** 3](#_Toc170562888)

[**User Interface (UI)** 3](#_Toc170562889)

[**Detailed Design** 3](#_Toc170562890)

[**Data Access Layer** 3](#_Toc170562891)

[**Communicator** 4](#_Toc170562892)

[**User Interface** 4](#_Toc170562893)

[**Bibliography** 5](#_Toc170562894)

# **Introduction**

## **Aim**

This project aims to create a fully functioning **prototype[[1]](#footnote-1)** of a searchable database of resources, contests, and other such academic content. It shall contain resources for many subjects such as math, science, public speaking, and history.

## **Targeted** **Audience**

The database will be aimed at gifted children but be available for the public.

## **Use/Problem that it solves**

This is useful and solves a problem because gifted children face challenges to find educational resources that suit them (Veno, 2023); (NAGC, n.d.), as they learn extremely fast and standard education systems are not designed to cope with them. Many gifted children also enjoy competing because of the challenge it gives them but can’t find places to do so. This project tries to centralize all these things and make finding contests and resources easy for everyone.

# **System Overview**

The database will have options to filter out search results based on age, grade, time, and location. It will have a simple interface so that anyone can use it, and all menu buttons will be located immediately on the front page.

# **Requirements**

## **Functional Requirements:**

* The database will have filters on the search functionality, such as age, grade, subject, and location (more complex filters such as location will only be implemented if I have time). These filters might be applied with double-ended sliders, dropdown menus, or an embedded map.
* The search results will be prioritized based on things such as how close the event is, what grade it is and if it is in the grade range selected by the user, etc.
* The database will be hosted on the web.
* The database will start returning results for search queries within 5 seconds under normal load conditions.
* Updates to the database will not cause significant lag to the UI and the database will be easy to update.
* The database will include information about topics such as math, science, public speaking, and history.[[2]](#footnote-2)
* The database will be lightweight enough that a high-end non-server computer should be able to handle at least 100,000 users at any time while still returning results within 10 seconds (where “returning results” is defined in the same way as it was for the condition “The database will start returning results within 5 seconds”.)

## **Non-Functional Requirements:**

* The database should be screen scraper resistant to avoid piracy and other such problems.
* The database should comply with industry standard security, such as AES-256 for encryption and SSL for data transmission. If SQL is used, the data should be sanitized to avoid SQL injection (I’m looking at you, Little Bobby Tables[[3]](#footnote-3)).

These Requirements are grouped by difficulty, where the green requirements are the easiest to implement and the red are the hardest to implement. The underlined requirements signify mandatory requirements.

## **External Interface Requirements**

* The database will use a MySQL database or a CSV to store data about contests. The choice between the two depends on which suits the project better (this will become clear after development begins). It seems more likely that I will go with CSVs, as I have used them more than I have used SQL
* The database will be accessible from a Flask frontend

## **Limitations**

* The project should not exceed a budget of $5000, including hardware, software licenses, hosting services, etc.
* The project should be easily maintainable by a single person, requiring no more than one hour of maintenance per day. This means that updates should mostly be handled automatically using techniques like web scraping and automating data updates through web scraping and API integrations where possible. The system should feature comprehensive documentation and a user-friendly administrative interface to facilitate ongoing maintenance and updates.

## **System** **Architecture**

* The database will be implemented in python, using MySQL or a CSV file as the database. MySQL has a python wrapper called MySQLdb, which is convenient as I will be using python for the rest of my code as well. Python also has a built-in module for dealing with CSV files.
* The code will use Flask, another third part module for web apps, to make the UI. As both the frontend and the database will be implemented in python, the data transmission will be as simple as importing the Data Access Layer (DAL) and querying it using python methods.

## **User Interface (UI)**

As this is only a prototype, the UI will be very minimal and will be focused on functionality and simplicity so that it is obvious to use. Searching will send you to a different page (this is similar to Google – they have a home page and they have a “/search” page that takes a query). This page will list all results as hyperlinks which link to a page that has more details about that contest/resource.

## **Detailed Design**

The code will be split into 3 entities:

## **Data Access Layer**

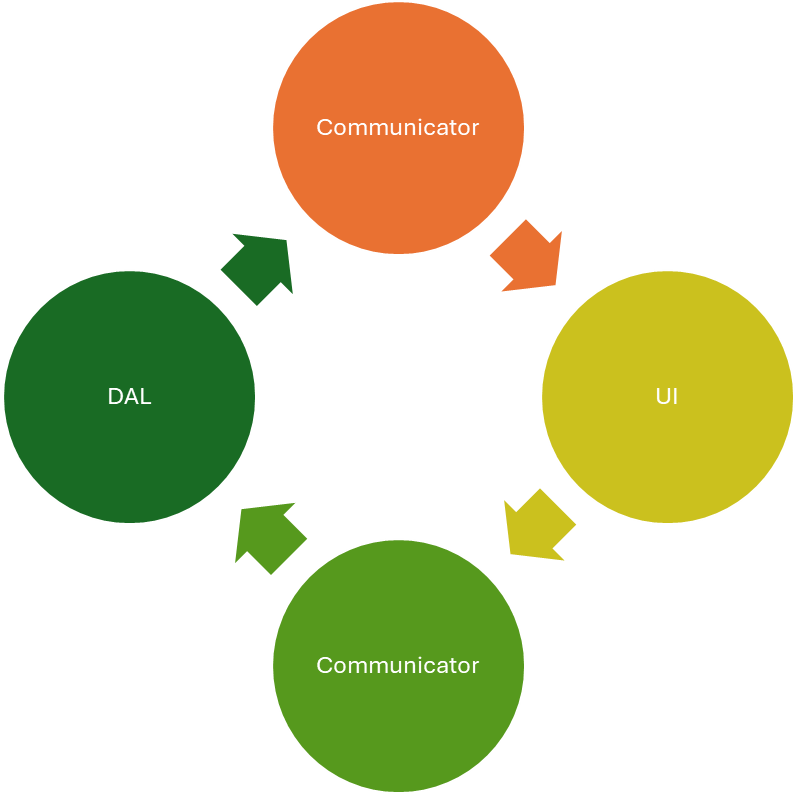
This accesses and manipulates the data. It also handles searching through the data. Some SEO might be needed here.[[4]](#footnote-4)

## **Communicator**

Manages communication between the Data Access Layer and the UI. It will reformat the UI data such as the selection of the dropdown menus to a format understandable by the DAL

## **User Interface**

The part of the app that the User Interacts with. This will be constructed using Flask.



UI sends the User’s Selection out of the filters such as Age, Subject, and Location to the Communicator

Reformatted criteria (Instead of grades: (9, 12), convert to grade\_minimum: 9, grade\_maximum:12)

Search Results are sent as Raw Data to the Communicator

The Communicator formats the Results as HTML and sends them to the UI

Below is a flow chart demonstrating how they will interact with each other:

# **Bibliography**

NAGC. (n.d.). *Why Are Gifted Programs Needed?* Retrieved from National Association For Gifted Children: https://nagc.org/page/why-are-gifted-programs-needed

Veno, E. (2023, July 7). *Gifted Kids Face These 6 Challenges, According to Science*. Retrieved from Prisma: https://www.joinprisma.com/blog/challenges-gifted-kids-face

1. Emphasis should be placed on “prototype”. This is only a design document for the prototype, which will be much more simplified than the real thing. [↑](#footnote-ref-1)
2. As the database grows, more subjects will be added [↑](#footnote-ref-2)
3. “Little Bobby Tables” is a reference to xkcd #327. The URL for this comic is: <https://xkcd.com/327/> [↑](#footnote-ref-3)
4. SEO might be needed in the DAL to generate better search results for the user’s queries. [↑](#footnote-ref-4)