Criterion B: Design Stage

01) Page Overview:

The functionalities will be separated into the following pages.

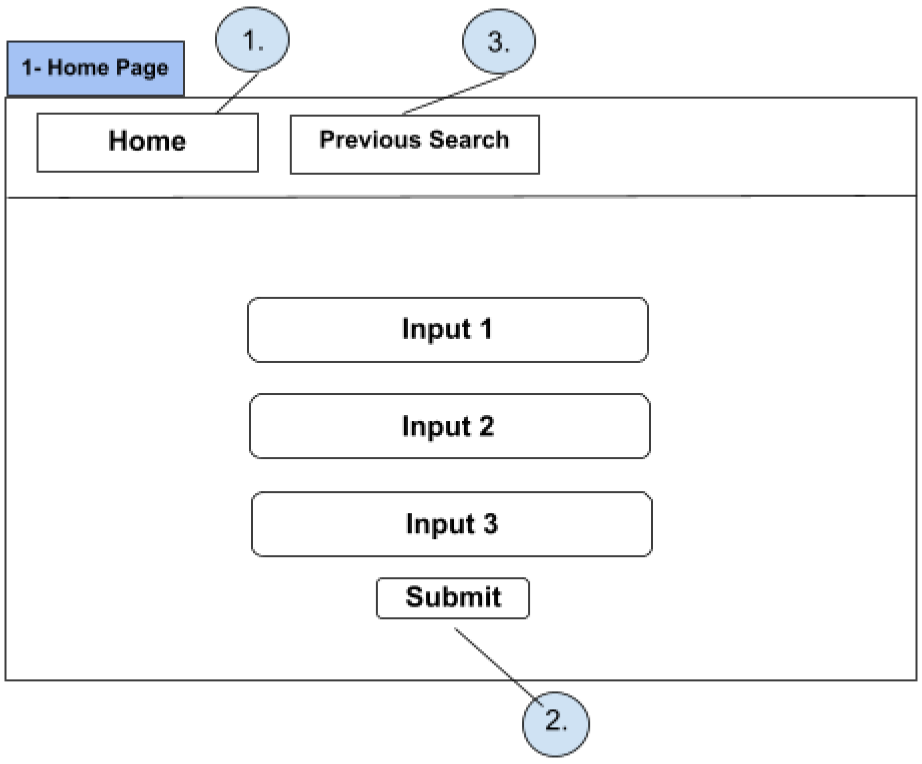
No.	Page	HTML template	URL	Description
1	Home page	main.html	/	Main page where user can find all functionalities
2	Loading page	Loading.html	/loading	A loading page with spinner running as backend processing of inputs is done
3	Previous Inputs page	previous_inputs.html	/previous_inputs	Display of past input records
4	Podcast Recommendation page	result.html	/result	Displays the recommended podcast and other results including further functionalities
5	Word Cloud page	word_cloud.html	/word_cloud	Displays an animated word cloud of associated keywords relating to user search from previous searches page

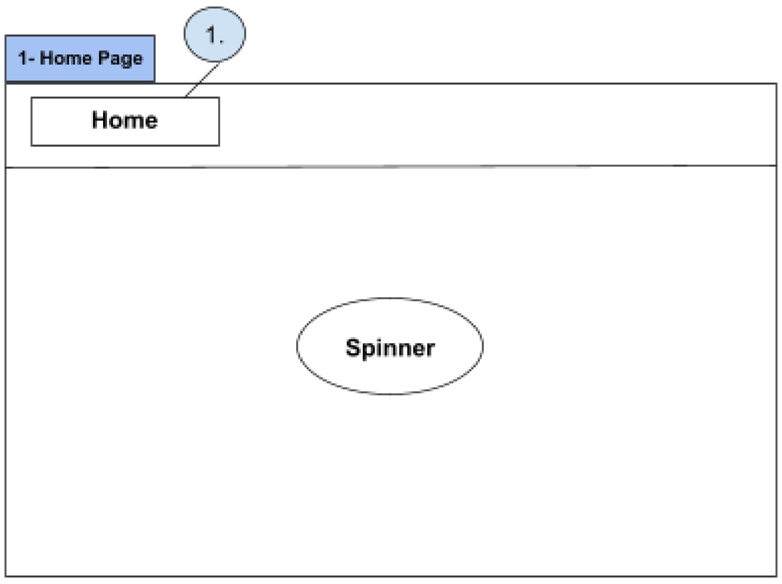
02: General Preview of the Graphic User Interface(GUI):

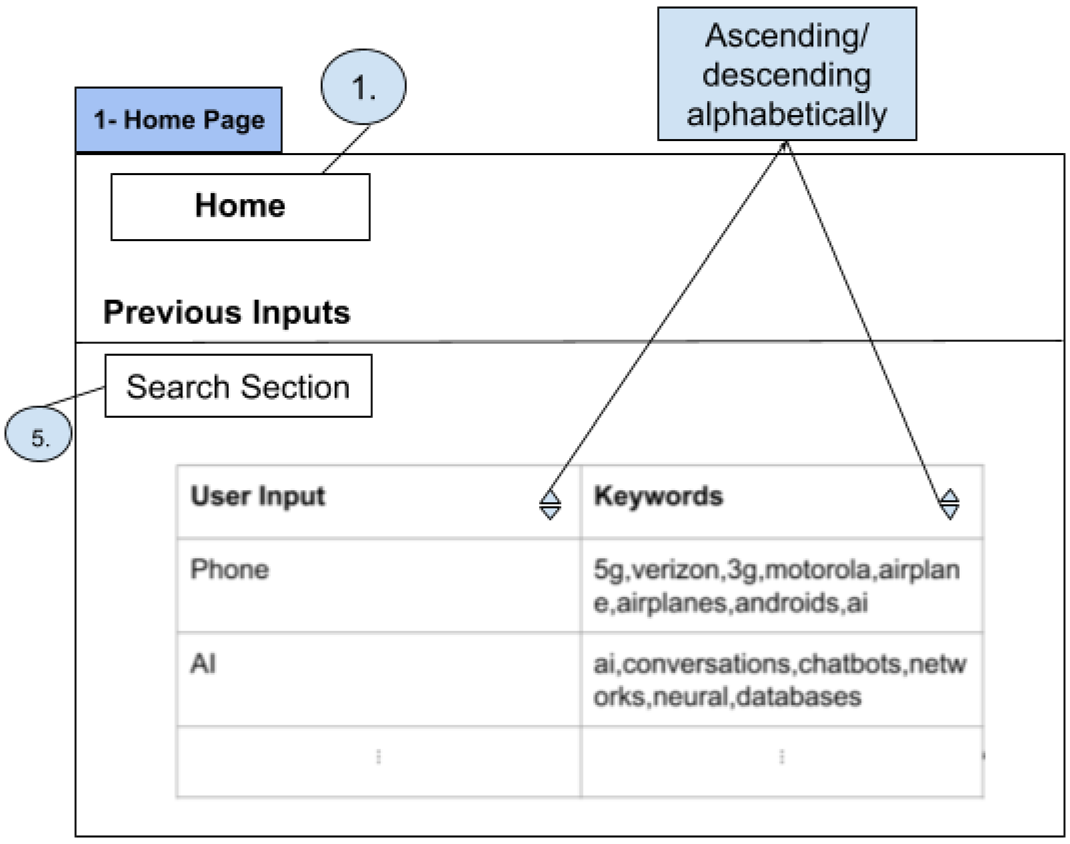
Clicking on each of the rectangles indicated by the **blue circles** would link to other pages within the website. Below, we will go through each section in detail, each of which has been approved as well.

<table>
<tr><th>Pages and Features</th><th>Wireframe</th></tr>
<tr><td>1.

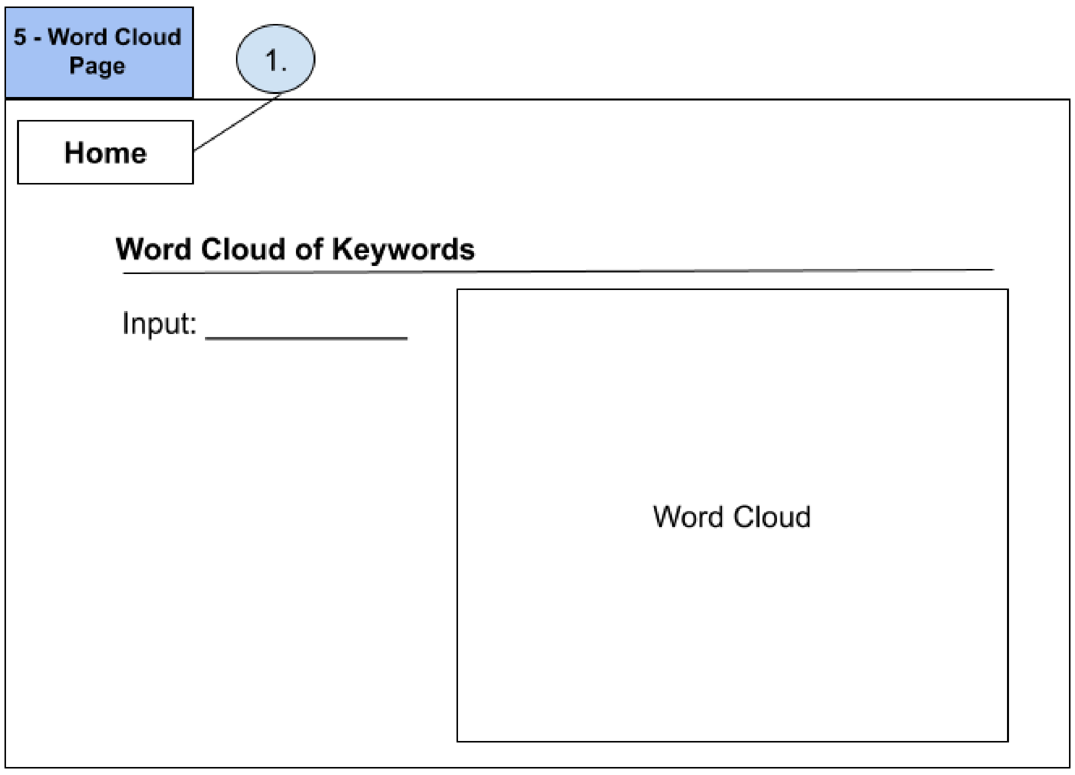
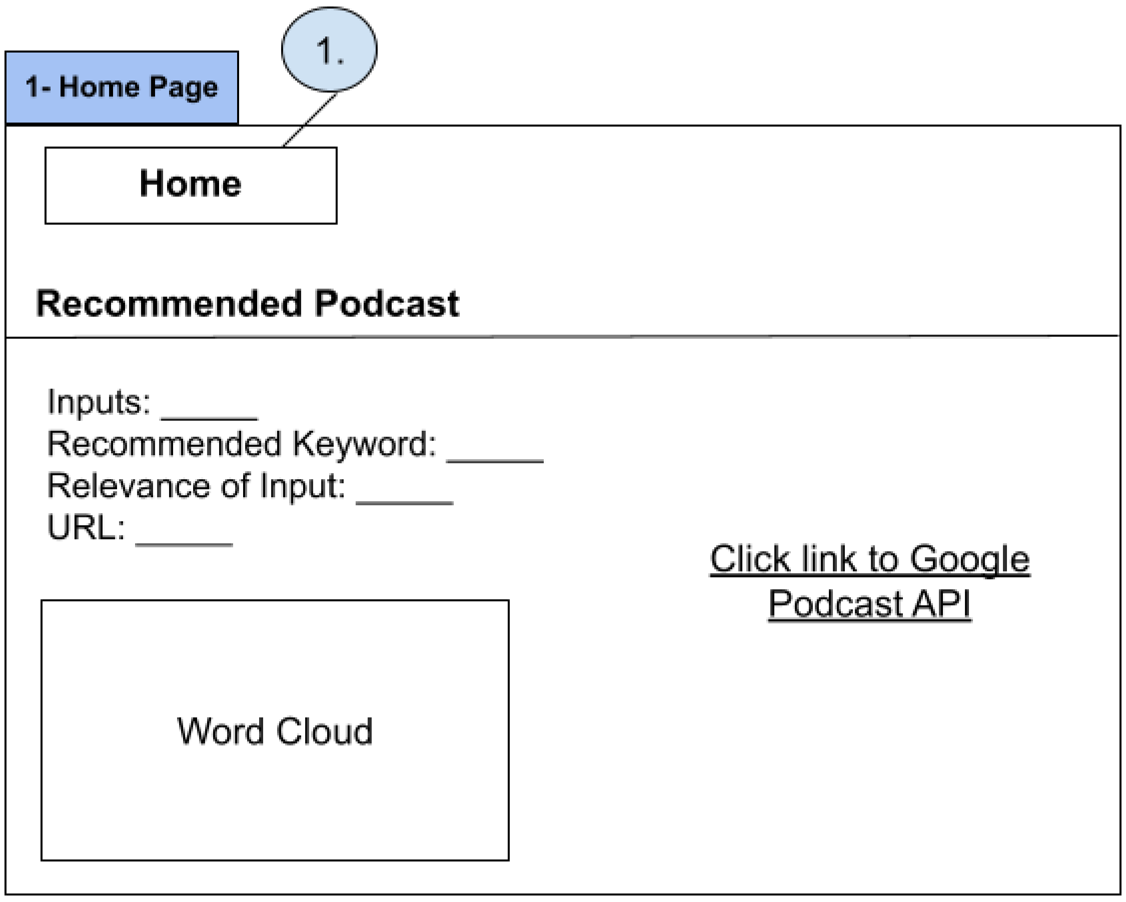
- Input triggers several function:
  - Check format
  - Input validation
  - Loading screen
  - Process input (web-scraping and NLP)

- Layout:
  - Flask form</td><td>
</td></tr>
<tr><td>2.

- Loading screen while input is being processed</td><td>
</td></tr>
<tr><td>3.

- Search section filters results in data table
- Submitting search query leads to Word Cloud page
- Columns of data table can be sorted alphabetically</td><td>

<table>
<tr><th>User Input</th><th>Keywords</th></tr>
<tr><td>Phone</td><td>5g,verizon,3g,motorola,airplane,airplanes,androids,ai</td></tr>
<tr><td>AI</td><td>ai,conversations,chatbots,networks,neural,databases</td></tr>
<tr><td>⋮</td><td>⋮</td></tr>
</table></td></tr>
</table>

4. - Word Cloud	 5 - Word Cloud Page 1. Home Word Cloud of Keywords Input: ____________ Word Cloud
5. - **Info on recommended podcast** - **Wordcloud** - **Link to podcast**	 1- Home Page 1. Home Recommended Podcast Inputs: ______ Recommended Keyword: ______ Relevance of Input: ______ URL: ______ Word Cloud Click link to Google Podcast API

03: File Structure

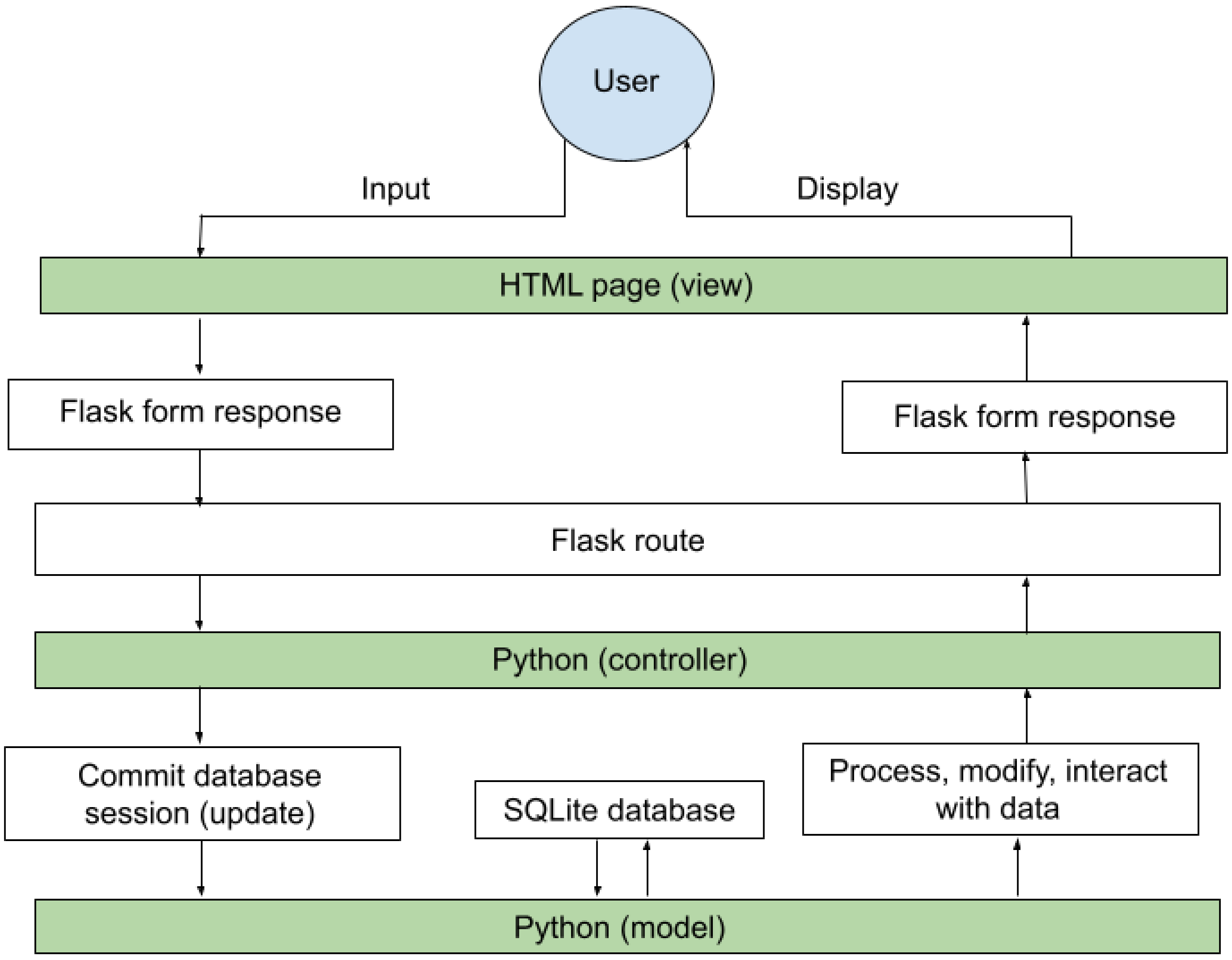
The following table denotes the file structure of the project, as well as the purpose of each folder and file. Folder names are marked in bold.

Podcast_finder	Main project folder
| main.py	Python file containing all the main functions and is called to launch the website
| | etl.py	Stands for “extract, transform, load”: used for functionalities related to web crawling, interaction with the BERT model, and loading of processed data to SQL database
| | nlp.py	Python file used for functionalities related to NLP processes

	like vectorisation, and centroid-clustering to find the most relevant keyword
| | -------- Data	
| | | KEYWORD_MAP.db	SQLite database
| | | Stopwords.pickle	Pickle file containing stopwords
| | | glove.6B.300d.txt	Text file containing pre-trained vector representations for words
| | | df_result.pickle	Pickle file containing pre-trained/pre-processed data from various user inputs
| | -------- Static	
| | | main.css	Style sheet for all HTML pages.
| | | | form_display.css	Style sheet for form.html
| | | | loading.css	Style sheet for loading.html
| | | | button.css	Style sheet for the home button
| | | | result_style.css	Style sheet for result.html
| | | | embedding_projector.css	Style sheet for embedding_projector.html
| | | | https://unpkg.com/gridjs/dist/theme/mermaid.min.css	Online Style sheet for displaying database
| | | | wordcloud.js	JavaScript file for animating word cloud
| | -------- Templates	
| | | form.html	Template for homepage form
| | | loading.html	Template for loading page
| | | result.html	Template for results page
| | | previous_inputs.html	Template for displaying database of previous inputs and relevant data
| | | embedding_projector.html	Template for displaying word cloud

Code Model

The below diagram follows the MVC (model, view, controller) framework for building web applications. The communication within the different logical components of the application is clearly set out:



Python Libraries Used

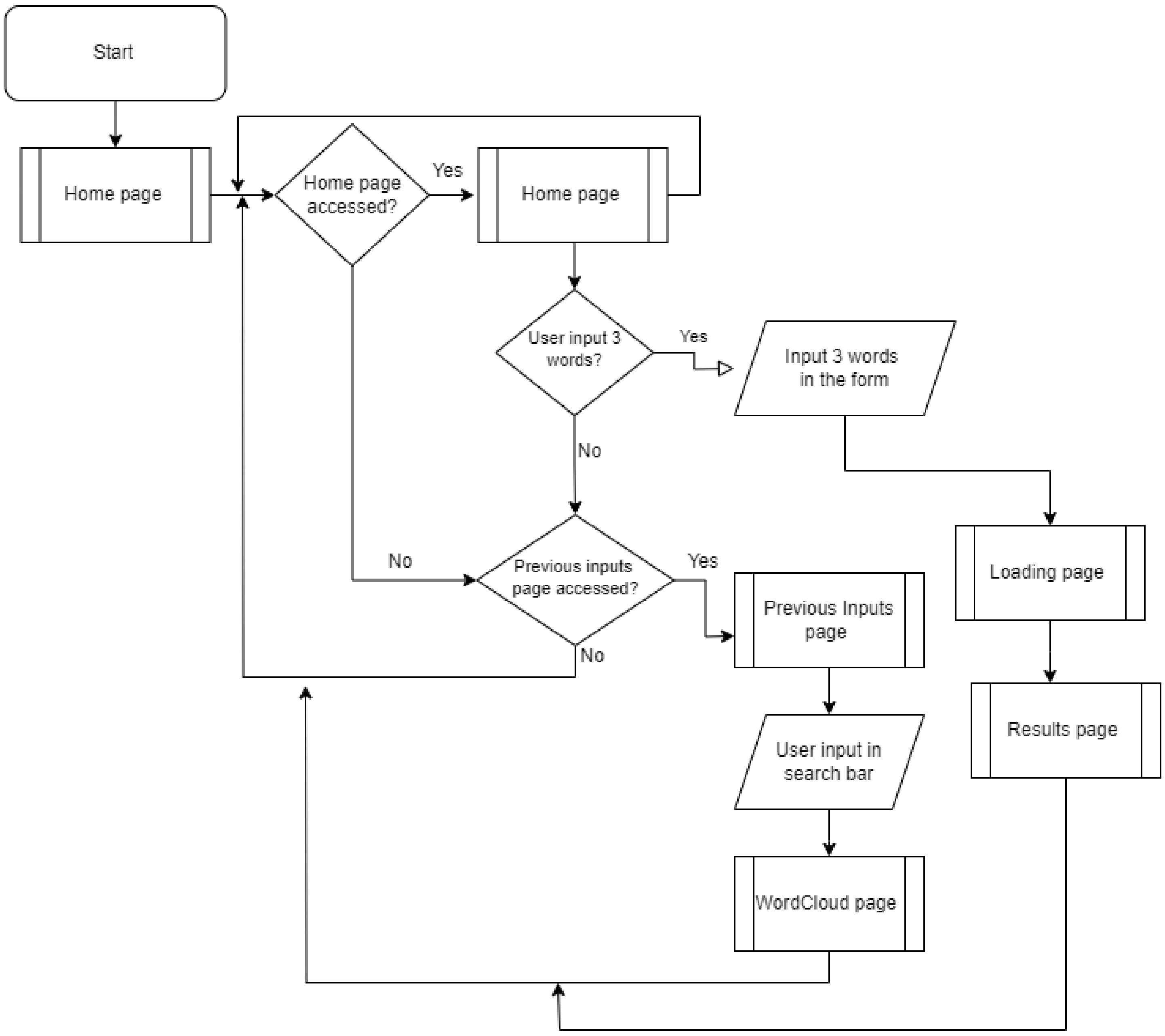
Python Library	Version	Purpose
Beautifulsoup4	4.12.2	Easy way to to perform web scraping for requested HTML and XML files
Pandas	2.1.1	Module contains many functions for manipulation of data structures which is used when fetching and updating data from the SQL database
Podcastparser	0.6.10	Easy tool for parsing podcast RSS feeds to search through, fetch and process data
Urllib3	2.0.5	A powerful, user-friendly HTTP client for Python used in requesting data from HTML and XML files
KeyBert	0.7.0	A pretrained model that uses tone, word frequency, and other extraction techniques to create keywords most similar to podcast descriptions
Db-sqlite3	0.0.1	Allows easy access to embedded SQL database engine and reads and writes data to the disk file
Numpy	1.26.0	Allows efficient methods in operating with multi-dimensional arrays
Gensim	4.3.2	Allows access to large pre-trained word to vector models that can be used to represent keywords as semantic

		vectors
Python-math	0.0.1	Used to perform mathematical calculations efficiently like .mean() and also utilise functions like math.random for pre-training centroid data
Regex	2023.8.8	Used to specify a search pattern in validating the syntax of keywords from each podcast
Requests	2.31.0	Allows easy sending of HTTP requests to access data from websites
Flask	2.3.3	Allows a simple way to efficiently build web applications including techniques like routing between different URLs, rendering HTML templates, and easy fetching and handling of request and post data.
NLTK	3.8.1	(Natural Language Toolkit) allows access to specific tools such as tokenisation and lemmatisation in language processing; this is useful for validating the user inputs in homepage form
Wordnet	0.0.1b2	Used in conjunction with NLTK library for finding word meanings and lemmatisation

04: Program, GUI and Input Flow – User & Client Perspective

The following materials used to explain the program basics from the client/user perspective.

User Flowchart



1. Data format validation:

Component	Format	Reason for format
1) Homepage user input	string	Straightforward format to easily process user inputs. The user can type any input as long as a result comes up through a connection with Google Podcast API. This is considered the most efficient form of validation for this specific input.
2) Previous searches page user input	string	Entered string can be easily compared with database records of previous user input.

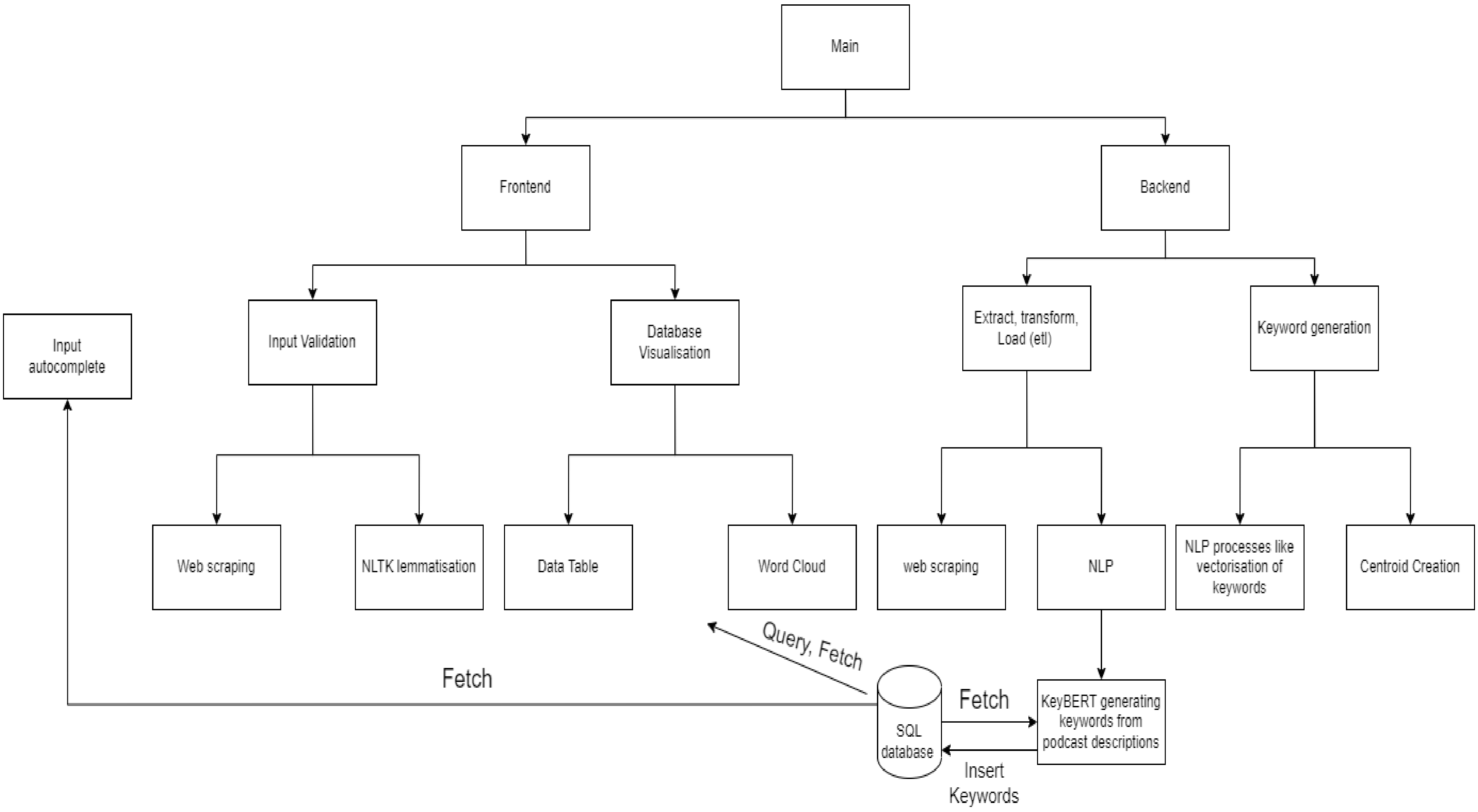
For Homepage user input:

Extreme Cases	**Processed input/response**
'Sdkjfljddj'	Error: no podcast result returned (returns to homepage automatically)
'Bless'	'bless'
'Governments'	'government'
'Government'	'government'
'the'	Error: input is empty (returns to homepage automatically)

Subsequent functions performing these validation checks will be explored in Criteria C.

05: Developer Perspective (backend):

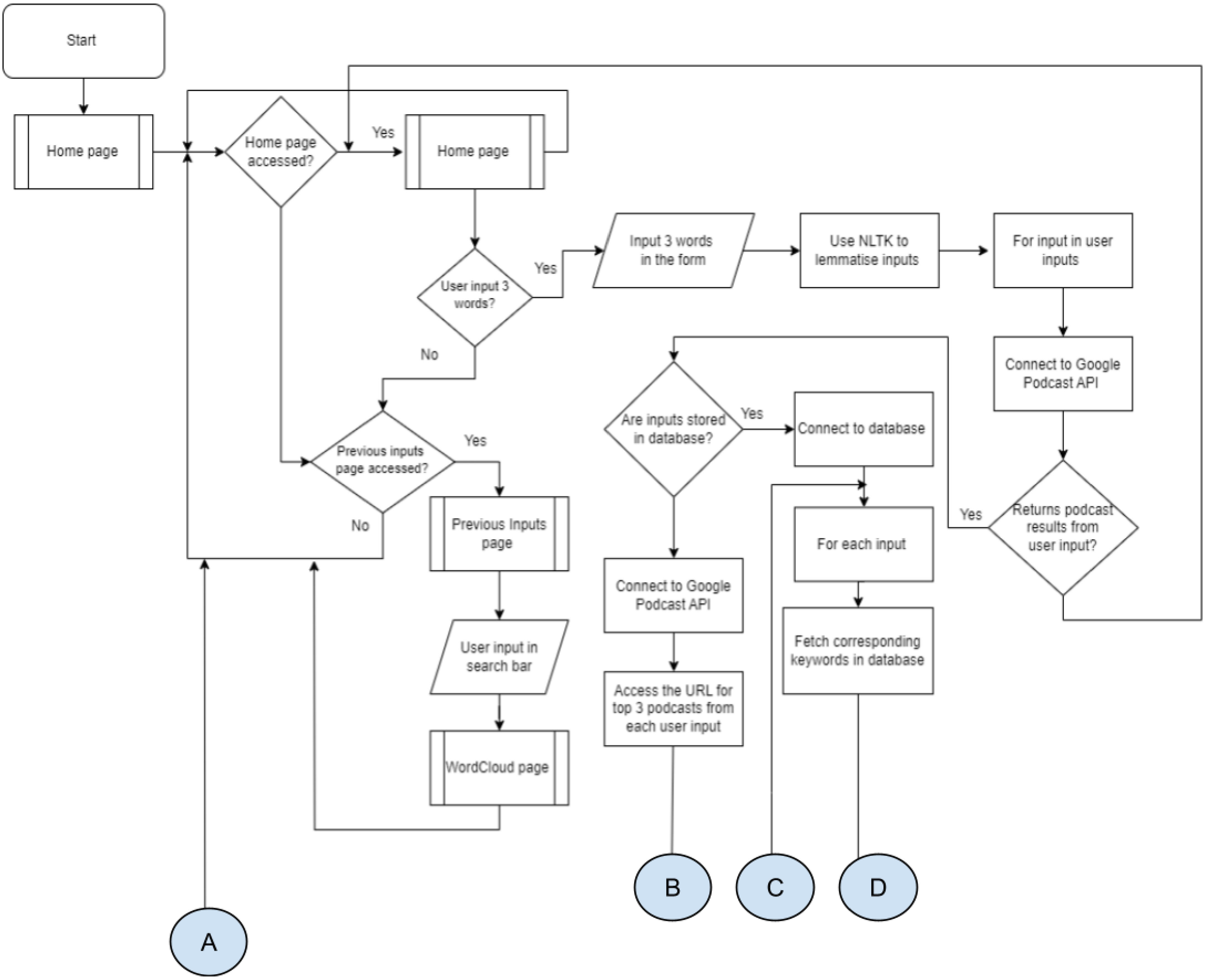
The hierarchy chart shows the main functionalities of the program split into the frontend and backend functionalities that interact with the SQL database.

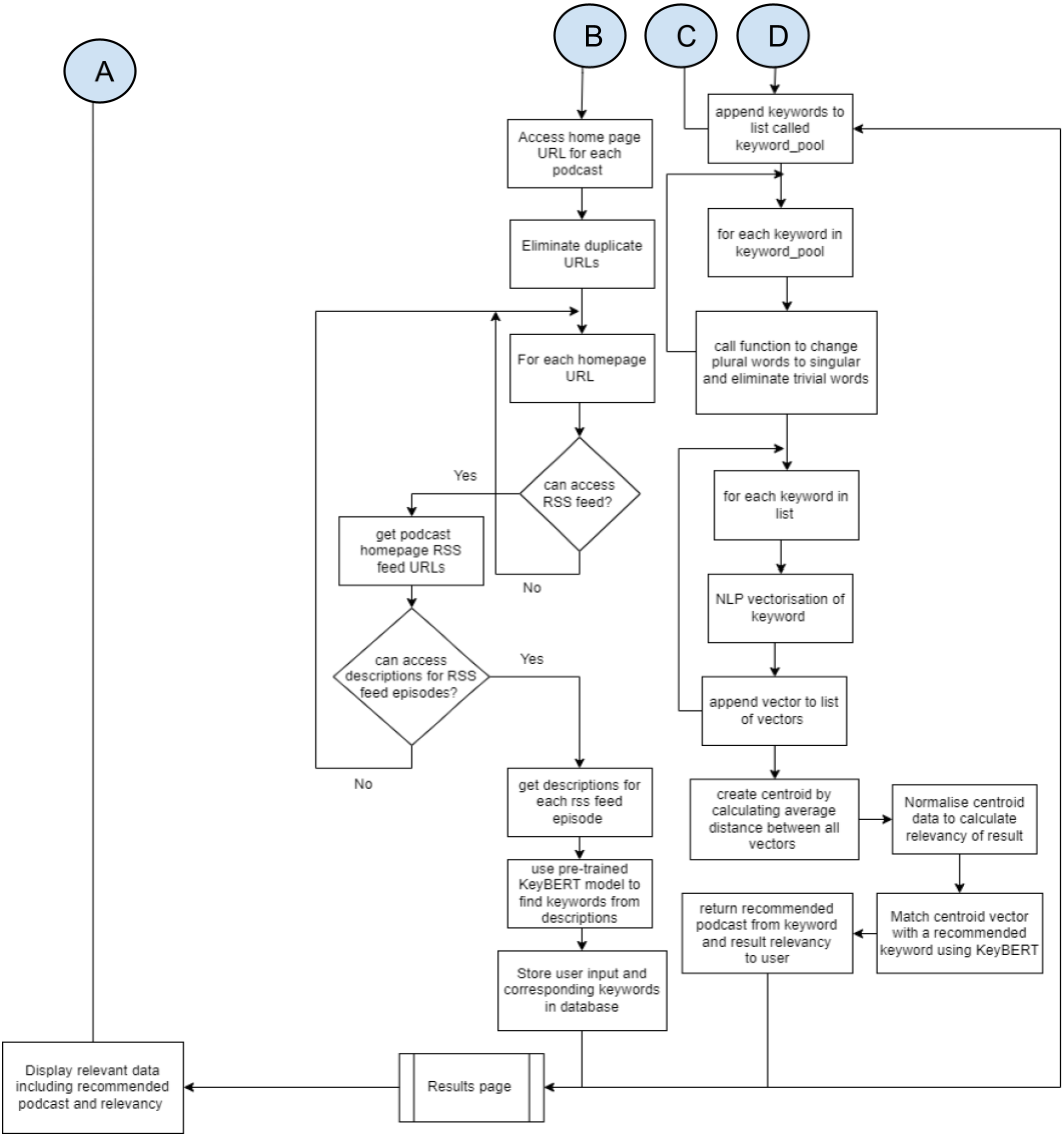


05.1) backend functionalities flowchart

From the homepage, when the user enters 3 words, input validation is performed through NLTK lemmatization and connection to Google Podcast API. If results are returned then inputs are valid. After lemmatization, if the inputs are stored in the database, then their corresponding keywords are fetched to be directly processed by NLP vectorisation using the Word2Vec model trained by GloVe. A centroid is created and confidence of recommendation is calculated from distance of keywords to the centroid. The results are returned and displayed in the podcast recommendation page.

Otherwise, if inputs are not stored in the database, web-scraping process is initiated such that podcast episode URLs are scraped. From these URLs, the homepage URLs are accessed using BeautifulSoup. Connection to an API allows the programme to get the RSS feeds which can then be used to append all the podcast episode descriptions into a nested list. These descriptions are processed using a pre-trained keyBERT model to find keywords from each podcast. The keywords are validated and the NLP process is repeated to return centroid and relevance score.


Start
Home page
Home page accessed?
Yes
Home page
User input 3 words?
Yes
No
Input 3 words in the form
Use NLTK to lemmatise inputs
For input in user inputs
Connect to Google Podcast API
Returns podcast results from user input?
Yes
Are inputs stored in database?
Yes
Connect to database
For each input
Fetch corresponding keywords in database
Connect to Google Podcast API
Access the URL for top 3 podcasts from each user input
Previous inputs page accessed?
Yes
No
Previous Inputs page
User input in search bar
WordCloud page
A
B
C
D


A
B
C
D
Access home page URL for each podcast
Eliminate duplicate URLs
For each homepage URL
can access RSS feed?
Yes
No
get podcast homepage RSS feed URLs
can access descriptions for RSS feed episodes?
Yes
No
get descriptions for each rss feed episode
use pre-trained KeyBERT model to find keywords from descriptions
Store user input and corresponding keywords in database
append keywords to list called keyword_pool
for each keyword in keyword_pool
call function to change plural words to singular and eliminate trivial words
for each keyword in list
NLP vectorisation of keyword
append vector to list of vectors
create centroid by calculating average distance between all vectors
Normalise centroid data to calculate relevancy of result
Match centroid vector with a recommended keyword using KeyBERT
return recommended podcast from keyword and result relevancy to user
Results page
Display relevant data including recommended podcast and relevancy

05.2) Main Program Functions:

This mainly concerns functions operating in the backend

Function	Colour Code	Purpose
main_input_processing()	🟦	Fetches the keywords generated from KeywordExtractor() function and processes them by eliminating stop words as well as changing plural words to their singular form.
KeywordExtractor()	🟥	Generates the keywords based on user input by either going through a web-scraping process and using KeyBERT model to extract keywords based on podcast descriptions or by fetching corresponding keywords to inputs already stored in a SQL database.
create_recommendation()	🟪	Generates a keyword from finding the most similar word based on the closest vector to the centroid. Moreover, it also calculates the relevancy of the centroid to podcast keywords using normalisation with pre-trained data.
get_centroid_2()	🟧	calculates centroid based on the average of all the vectorised keywords from each podcast homepage and calculates the total distance of all the keywords to the centroid
query_user_input()	🟪	Fetches user query data from previous search page and finds corresponding keywords to display word cloud

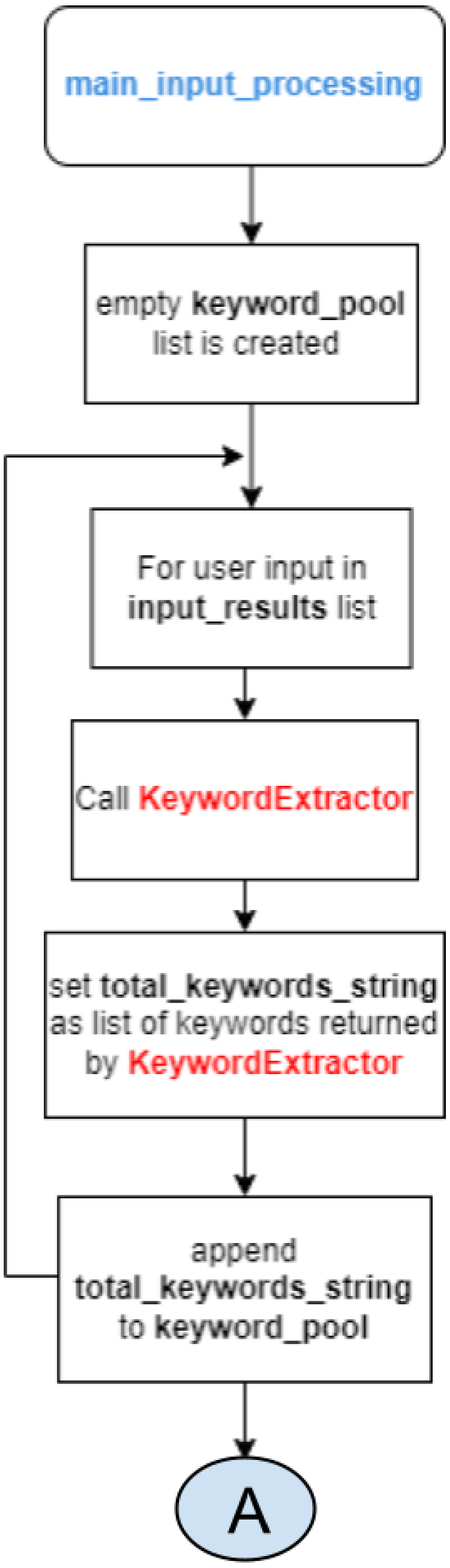
05.3) Variables:

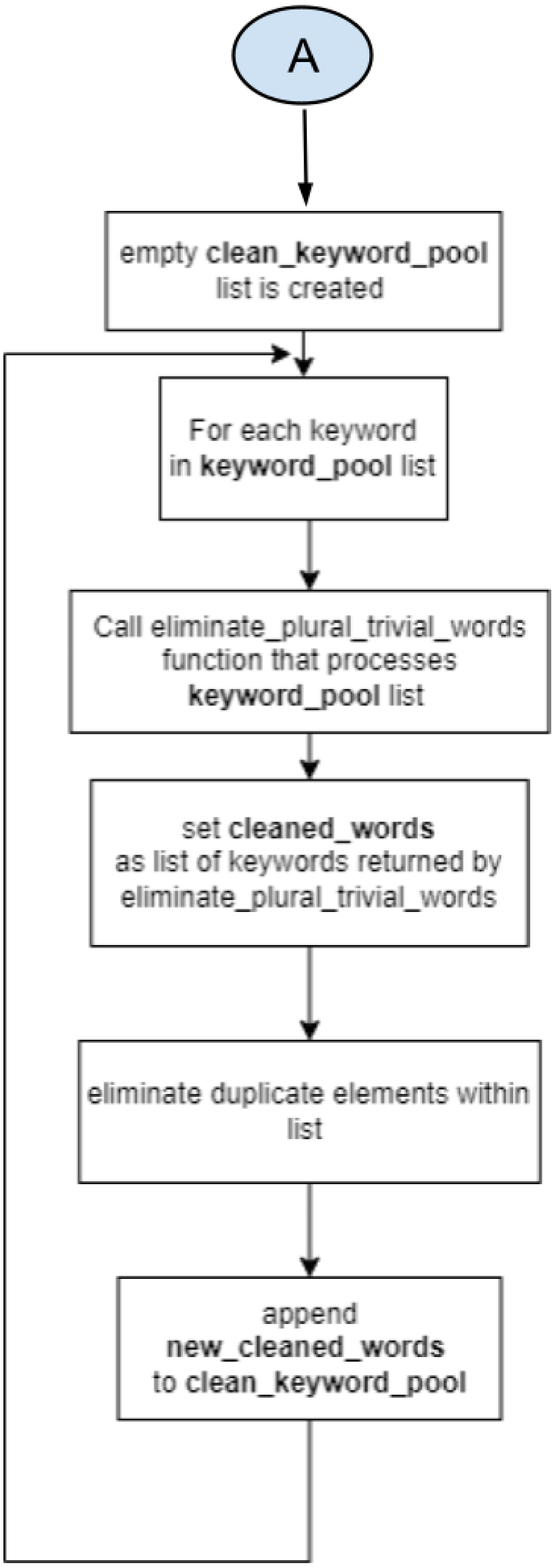
To accommodate the outputs of these major functions, the main variables I must create for the program include:

Variable Name	Variable Type	Purpose
user_inputs	List	Holds the 3 user inputs in homepage form
podcast_urls	List	Holds the URLs of each podcast item
homepage_urls	List	Holds the homepage URLs from each podcast item
keyword_pool	List (2D)	List of list holding the keywords from each input
rss_url	String	Holds the link to the RSS feed of accessed homepage URL
parsed	Dictionary	Holds the parsed podcast RSS feed URL
descriptions	Dictionary	For each homepage title stored as a key, it would hold the descriptions concatenated together from each podcast episode of the parsed RSS feed of homepage URLs
total_keywords	List (2D)	List of List storing the keywords generated from the descriptions of each podcast homepage
clean_keyword_pool	List (2D)	Holds the keywords from each podcast homepage after the elimination of stopwords and the changing of plural words to its singular form
centroid_2	NDArray	(1X300) array storing the vector data from the mean of the vectors for all keywords
avg_distance	Float	Average distance of all the keyword vectors in space to the centroid
c2_relevance	Float	Relevancy of centroid to podcast keywords calculated from normalisation with pre-trained data
centroid_input1	NDArray	Contains array of most similar words by vector to the centroid
similar_word	String	Most relevant keyword generated based on the centroid vector
query_keywords	List	Contains the corresponding keywords to the input that the user queries in the previous search page

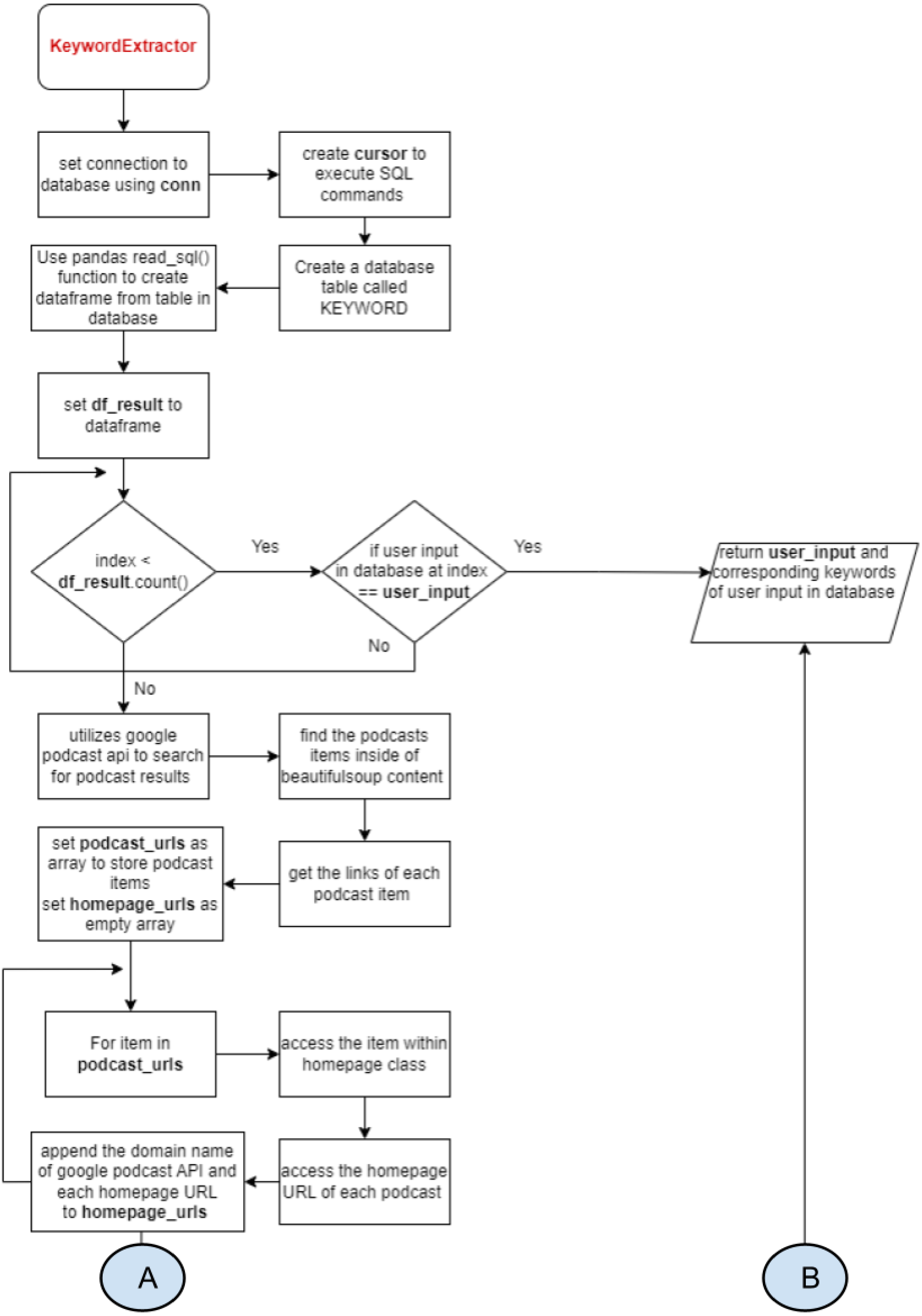
05.4) Flowcharts for main functions

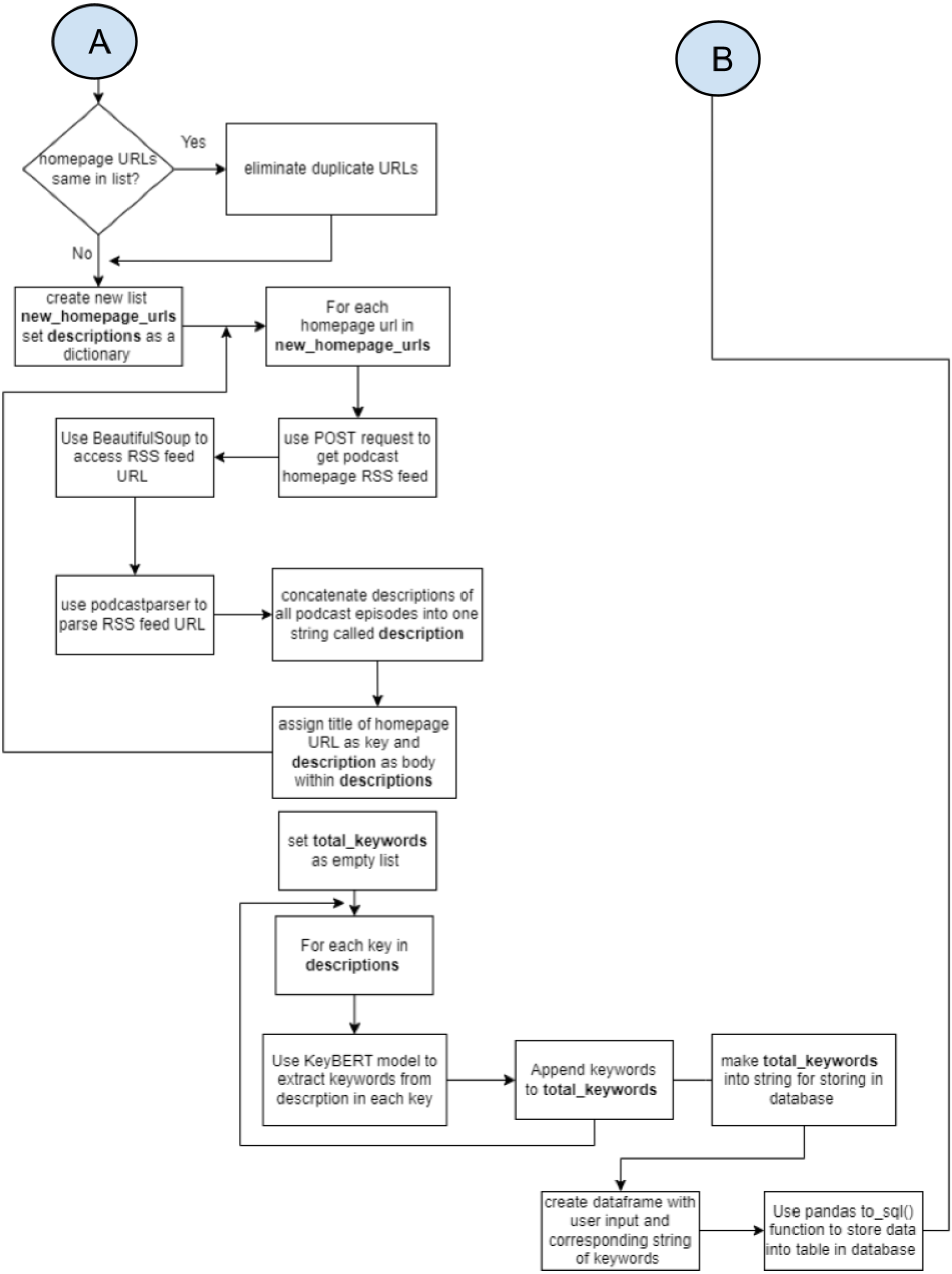
main_input_processing()


main_input_processing
empty keyword_pool list is created
For user input in input_results list
Call KeywordExtractor
set total_keywords_string as list of keywords returned by KeywordExtractor
append total_keywords_string to keyword_pool
A

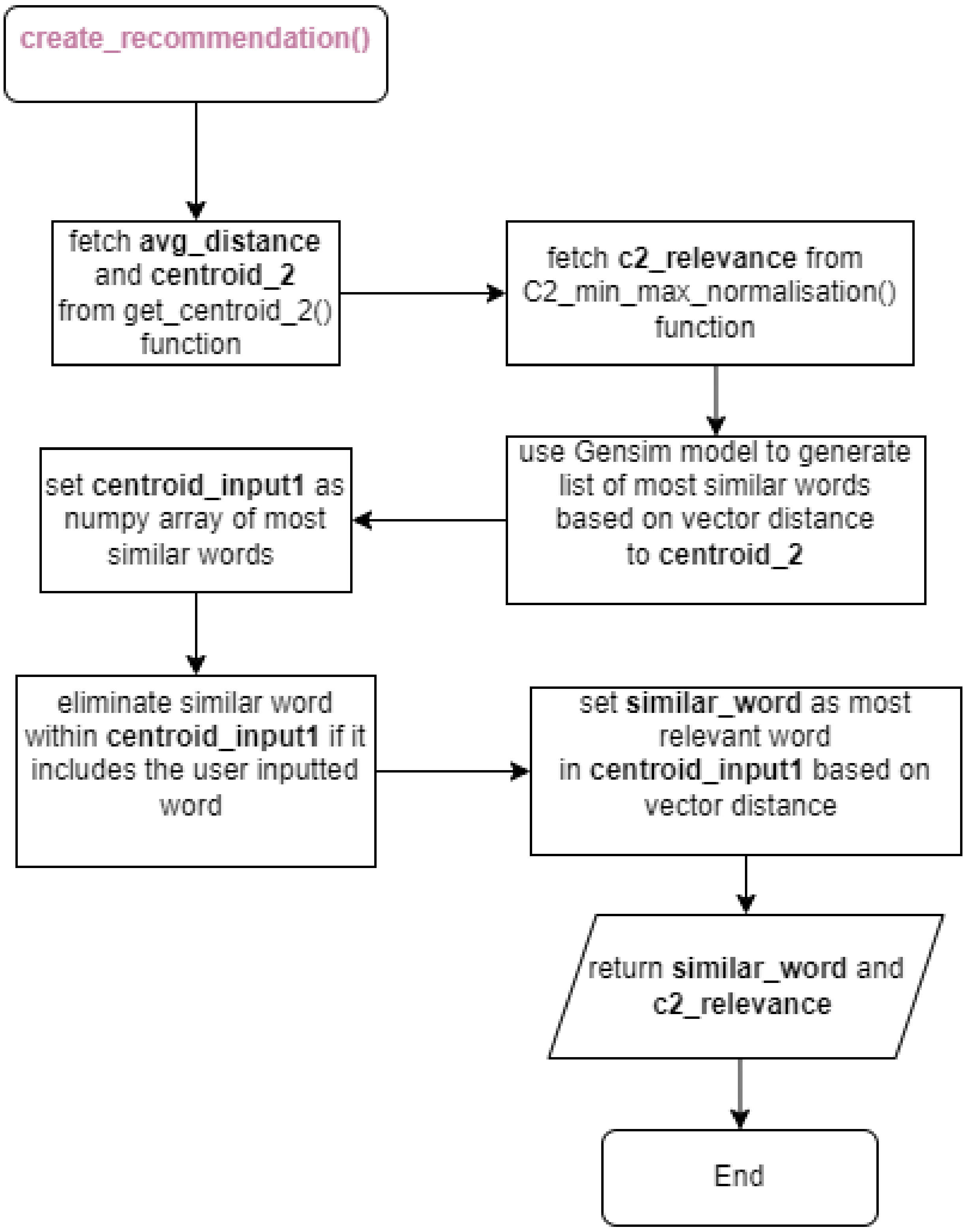

A
empty **clean_keyword_pool** list is created
For each keyword in **keyword_pool** list
Call eliminate_plural_trivial_words function that processes **keyword_pool** list
set **cleaned_words** as list of keywords returned by eliminate_plural_trivial_words
eliminate duplicate elements within list
append **new_cleaned_words** to **clean_keyword_pool**

KeywordExtractor()

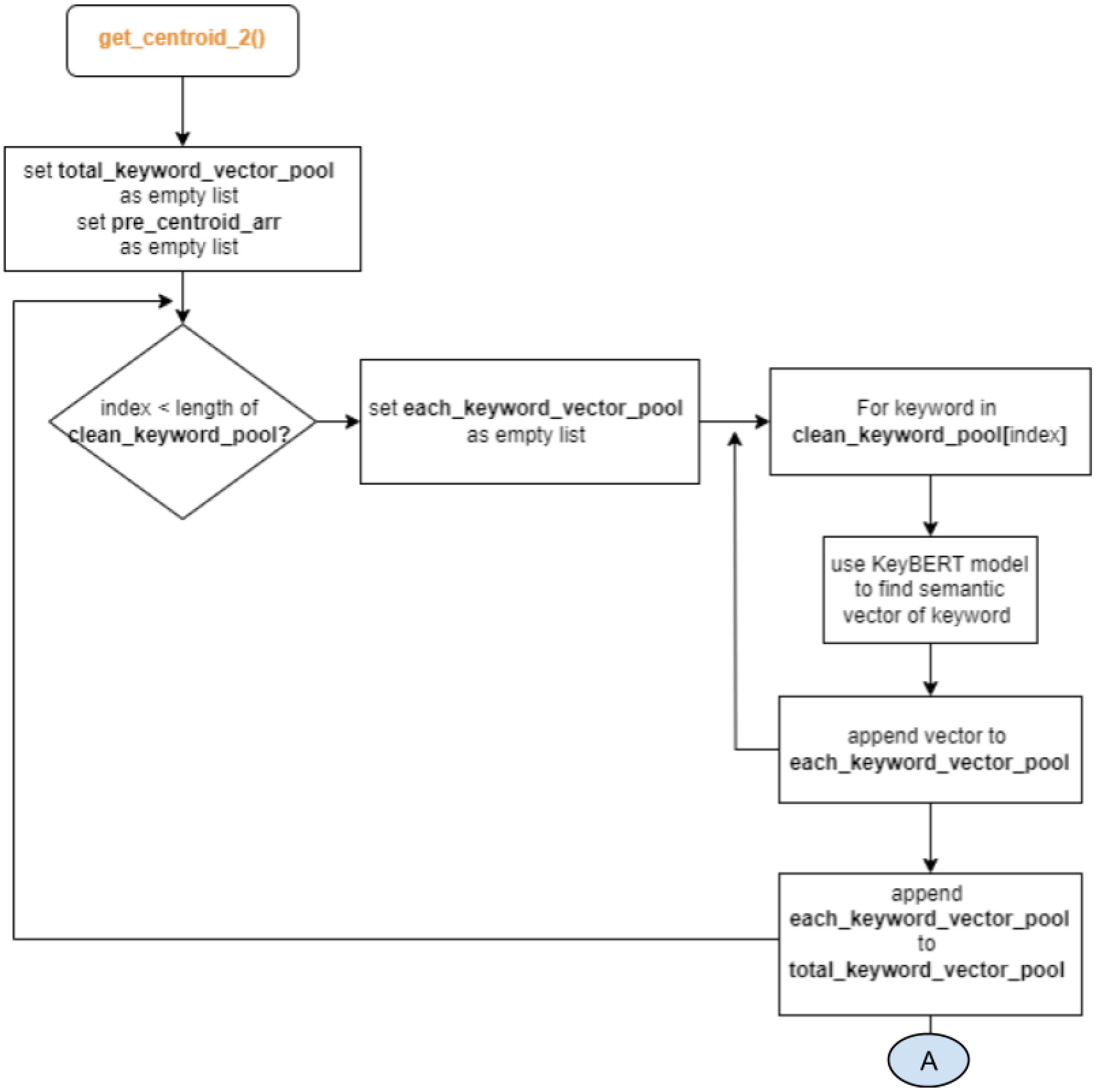

KeywordExtractor
set connection to database using conn
create cursor to execute SQL commands
Create a database table called KEYWORD
Use pandas read_sql() function to create dataframe from table in database
set df_result to dataframe
index < df_result.count()
Yes
if user input in database at index == user_input
Yes
return user_input and corresponding keywords of user input in database
No
No
utilizes google podcast api to search for podcast results
find the podcasts items inside of beautifulsoup content
get the links of each podcast item
set podcast_urls as array to store podcast items
set homepage_urls as empty array
For item in podcast_urls
access the item within homepage class
access the homepage URL of each podcast
append the domain name of google podcast API and each homepage URL to homepage_urls
A
B

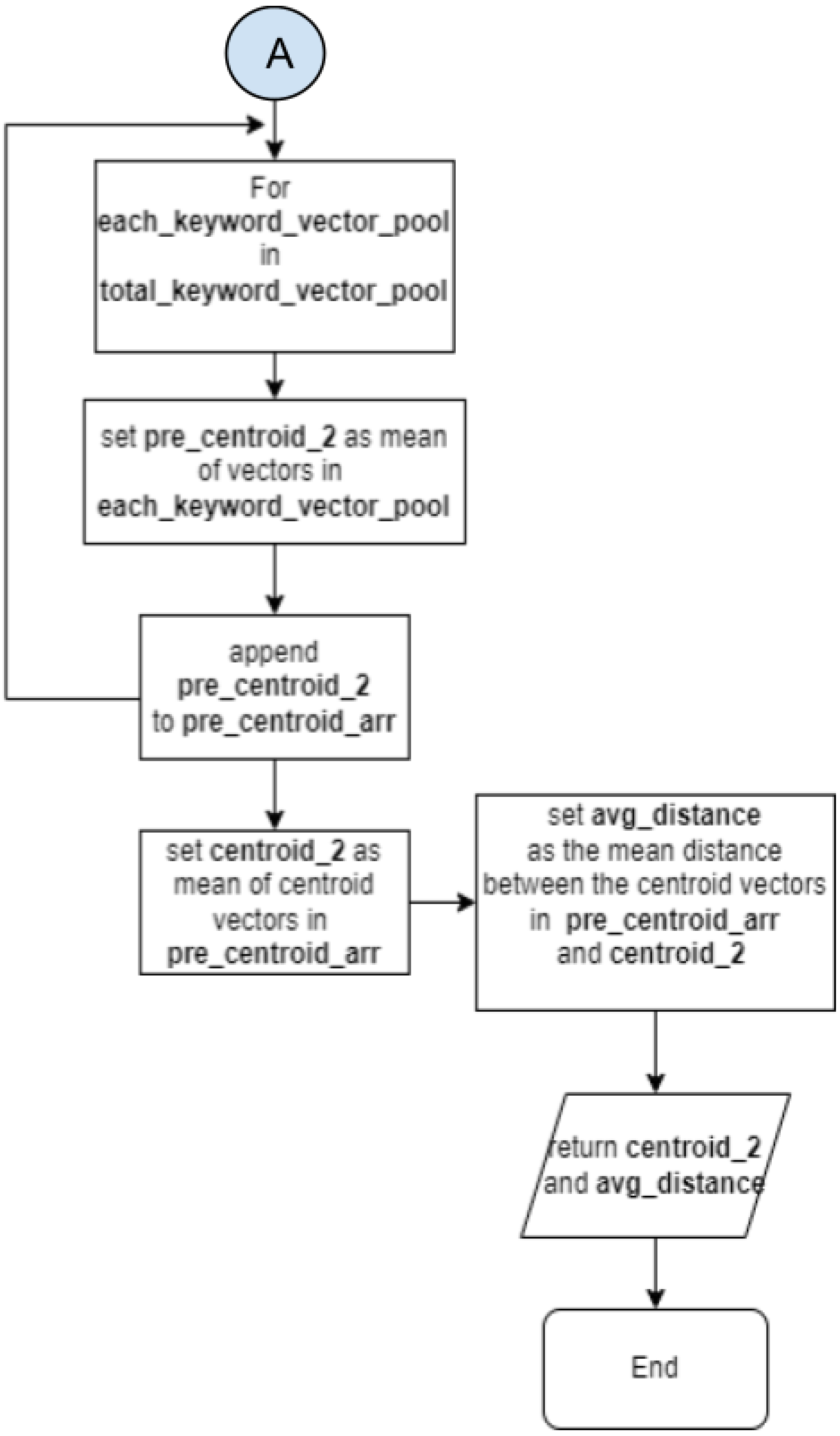

A
homepage URLs same in list?
Yes
eliminate duplicate URLs
No
create new list **new_homepage_urls** set **descriptions** as a dictionary
For each homepage url in **new_homepage_urls**
use POST request to get podcast homepage RSS feed
Use BeautifulSoup to access RSS feed URL
use podcastparser to parse RSS feed URL
concatenate descriptions of all podcast episodes into one string called **description**
assign title of homepage URL as key and **description** as body within **descriptions**
set **total_keywords** as empty list
For each key in **descriptions**
Use KeyBERT model to extract keywords from descrption in each key
Append keywords to **total_keywords**
make **total_keywords** into string for storing in database
create dataframe with user input and corresponding string of keywords
Use pandas to_sql() function to store data into table in database
B

create_recommendation()

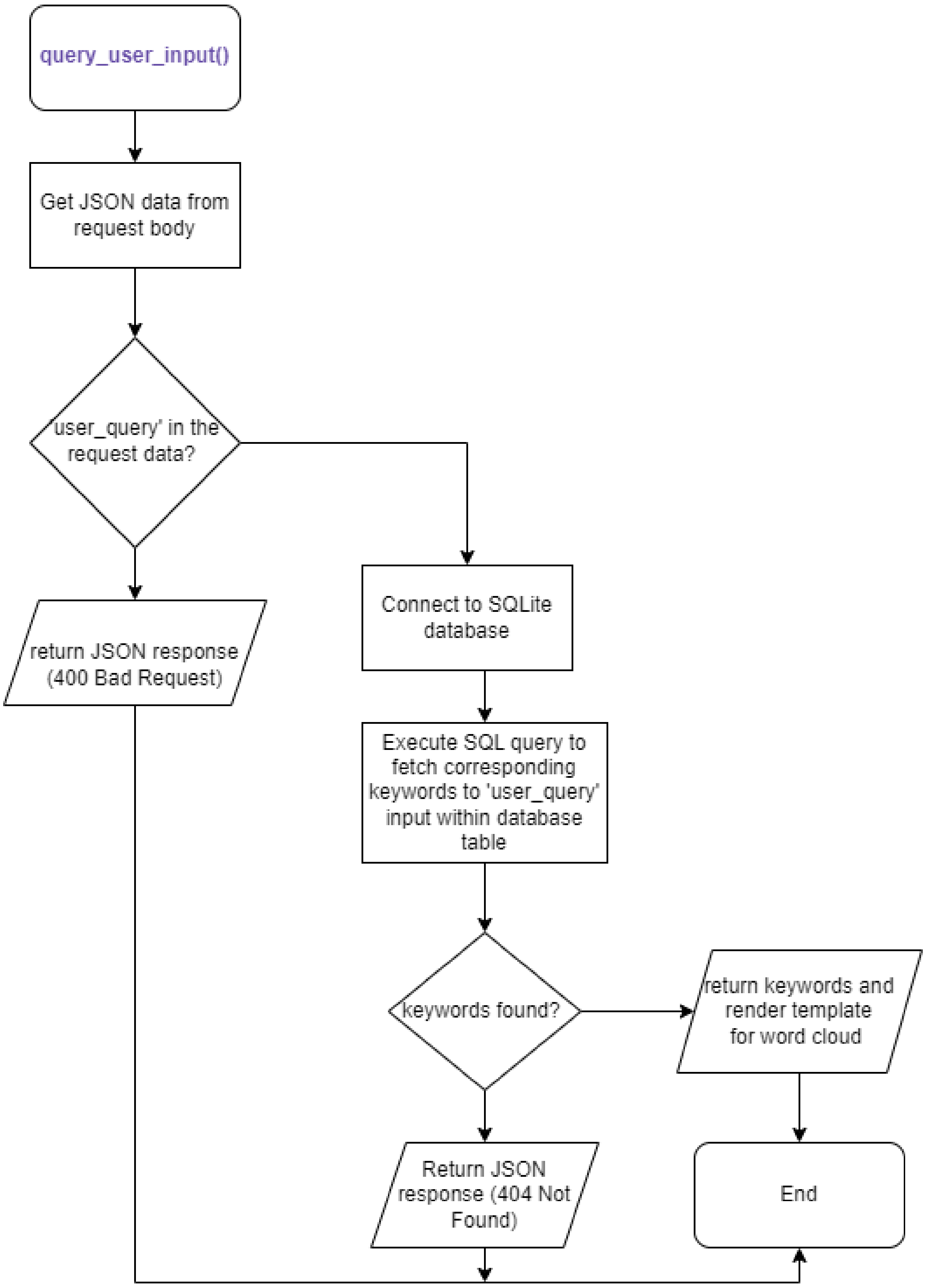

create_recommendation()
fetch **avg_distance** and **centroid_2** from get_centroid_2() function
fetch **c2_relevance** from C2_min_max_normalisation() function
use Gensim model to generate list of most similar words based on vector distance to **centroid_2**
set **centroid_input1** as numpy array of most similar words
eliminate similar word within **centroid_input1** if it includes the user inputted word
set **similar_word** as most relevant word in **centroid_input1** based on vector distance
return **similar_word** and **c2_relevance**
End

get_centroid_2()


get_centroid_2()
set total_keyword_vector_pool as empty list set pre_centroid_arr as empty list
index < length of clean_keyword_pool?
set each_keyword_vector_pool as empty list
For keyword in clean_keyword_pool[index]
use KeyBERT model to find semantic vector of keyword
append vector to each_keyword_vector_pool
append each_keyword_vector_pool to total_keyword_vector_pool
A


A
For **each_keyword_vector_pool** in **total_keyword_vector_pool**
set **pre_centroid_2** as mean of vectors in **each_keyword_vector_pool**
append **pre_centroid_2** to **pre_centroid_arr**
set **centroid_2** as mean of centroid vectors in **pre_centroid_arr**
set **avg_distance** as the mean distance between the centroid vectors in **pre_centroid_arr** and **centroid_2**
return **centroid_2** and **avg_distance**
End

query_user_input()


query_user_input()
Get JSON data from request body
'user_query' in the request data?
return JSON response (400 Bad Request)
Connect to SQLite database
Execute SQL query to fetch corresponding keywords to 'user_query' input within database table
keywords found?
return keywords and render template for word cloud
Return JSON response (404 Not Found)
End

06) Database:

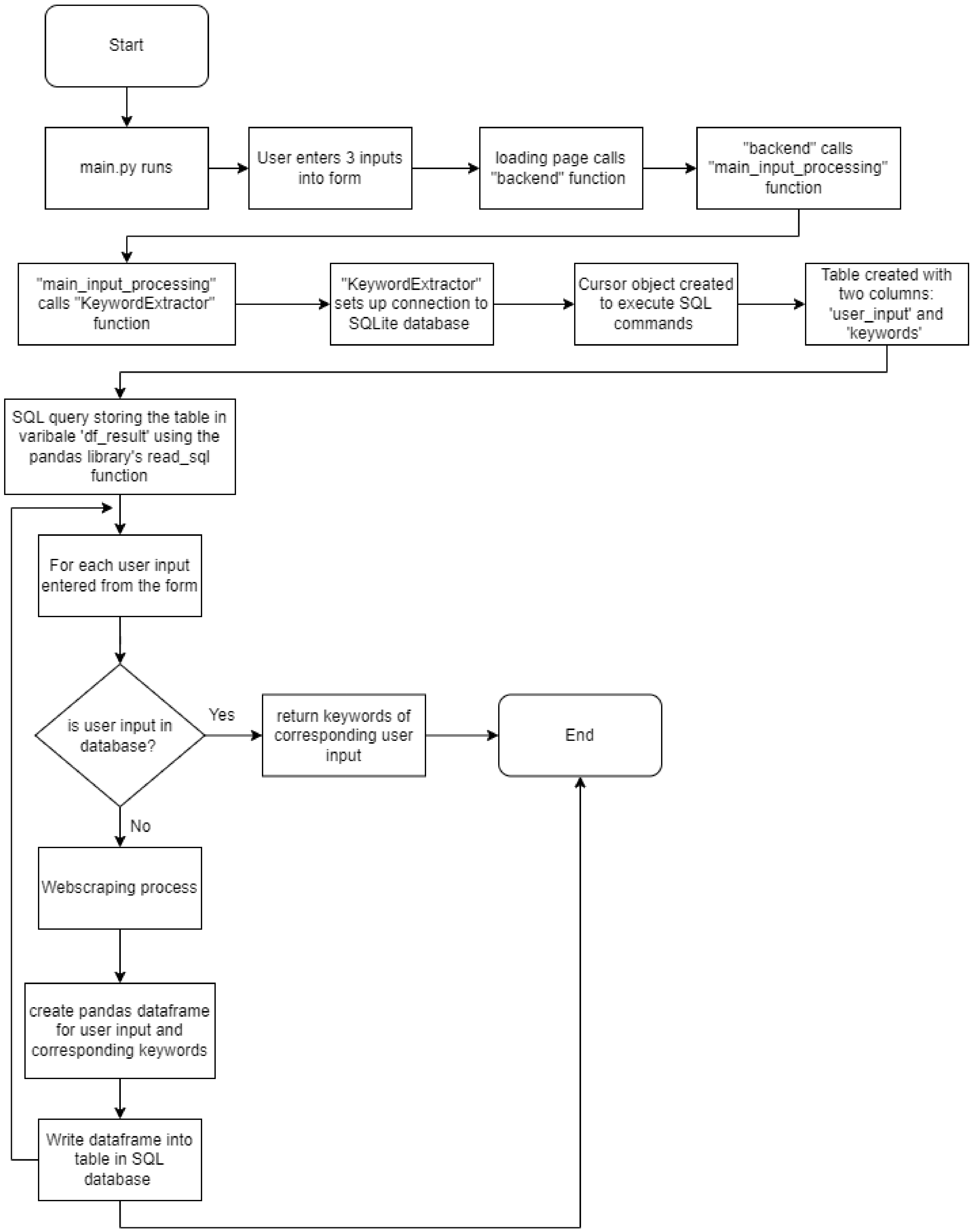
The website uses a SQLite database, “KEYWORD_MAP.db” in the project folder. It stores past user inputs, allowing easy and efficient access to corresponding keywords. This eliminates the more tedious and time-consuming web-scraping + NLP process to gather keywords.

Why not use a dataframe instead of a database?

Pandas dataframes have limited memory capacity; Sql database provides large storage and more efficient retrieval with more advanced querying techniques and optimised indexing. However, the simplicity of the database might mean there won’t be a significant difference.

Table - KEYWORD		
Column	**Type**	**Purpose**
user_input	String	Word user inputted
keywords	String	Corresponding keywords extracted from related podcasts

The following flowchart outlines how the database is linked to the website and how data insertion occurs to the database:


Start
main.py runs
User enters 3 inputs into form
loading page calls "backend" function
"backend" calls "main_input_processing" function
"main_input_processing" calls "KeywordExtractor" function
"KeywordExtractor" sets up connection to SQLite database
Cursor object created to execute SQL commands
Table created with two columns: 'user_input' and 'keywords'
SQL query storing the table in varibale 'df_result' using the pandas library's read_sql function
For each user input entered from the form
is user input in database?
Yes
return keywords of corresponding user input
End
No
Webscraping process
create pandas dataframe for user input and corresponding keywords
Write dataframe into table in SQL database

07) Hypothesis Testing:

- To check the statistical significance of test results:

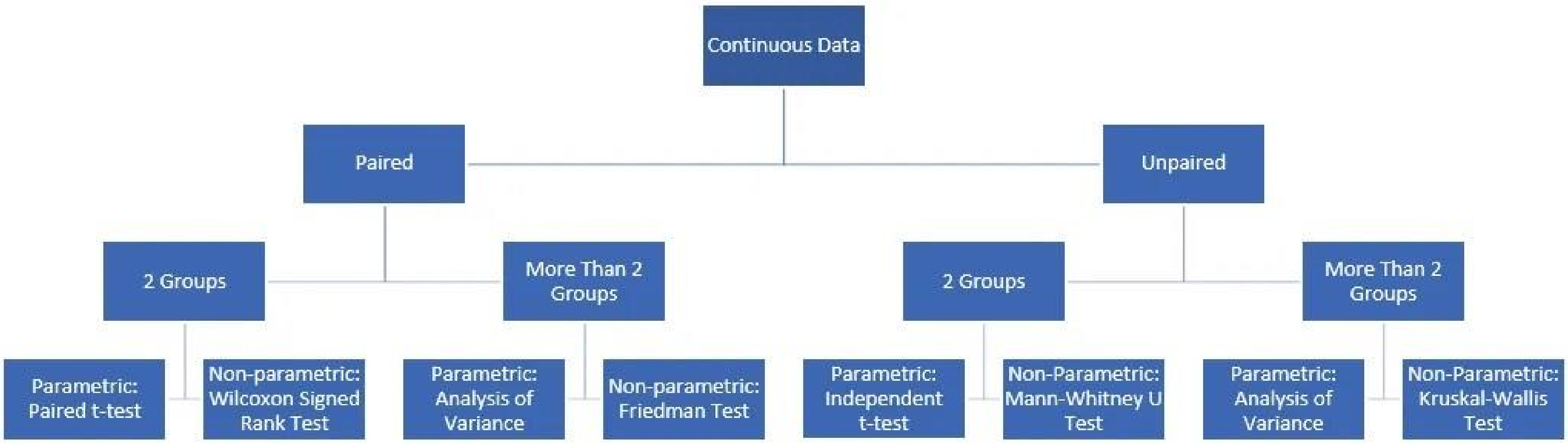
  Hypothesis: vector distances between the keyword pool vectors and the centroid calculated through certain algorithms are smaller than the vector distances between random scatter of data points and the centroid.

  The p-value calculated will suggest if this assumption is correct, meaning that centroid calculation is a good measure of the statistical relevance of the keyword represented by that centroid vector.

Assumption check:

- To decide whether to use parametric or non-parametric version of test using requirements below:
  - Observations in each sample are independent and identically distributed
  - Observations in each sample are normally distributed
  - Observations in each sample have the same variance

Select test:


Continuous Data
Paired
Unpaired
2 Groups
More Than 2 Groups
2 Groups
More Than 2 Groups
Parametric: Paired t-test
Non-parametric: Wilcoxon Signed Rank Test
Parametric: Analysis of Variance
Non-parametric: Friedman Test
Parametric: Independent t-test
Non-Parametric: Mann-Whitney U Test
Parametric: Analysis of Variance
Non-Parametric: Kruskal-Wallis Test

Definitions:

- Alpha (a) = 0.05
- Centroid distance for similar inputs = S
- Centroid distance for random inputs = R
- H0 (Null hypothesis): S<R
- H1 (Alternative hypothesis): S>=R
- Distance to Centroid 1: calculated using the mean distance of all input keywords to centroid
- Distance to Centroid 2: calculated using the mean distance of the centroid to the three centroids created from keyword pool of each user input

Results:

- Data is unpaired because the input obtained and performed on are different and random.

From fig 1 and fig 2, for both centroid 1 and centroid 2, the euclidean difference of the distances between similar and dissimilar words follow a normal distribution, suggesting that there is a strong correlation between similarity of inputs and the distance to both centroid 1 and 2. Thus, euler distance can be used as a measure of the similarity between user inputs and hence determine the confidence level of the recommended podcast to the user.

Fig 1. Euclidean Difference Between Similar and Non-similar Inputs Under Centroid 1

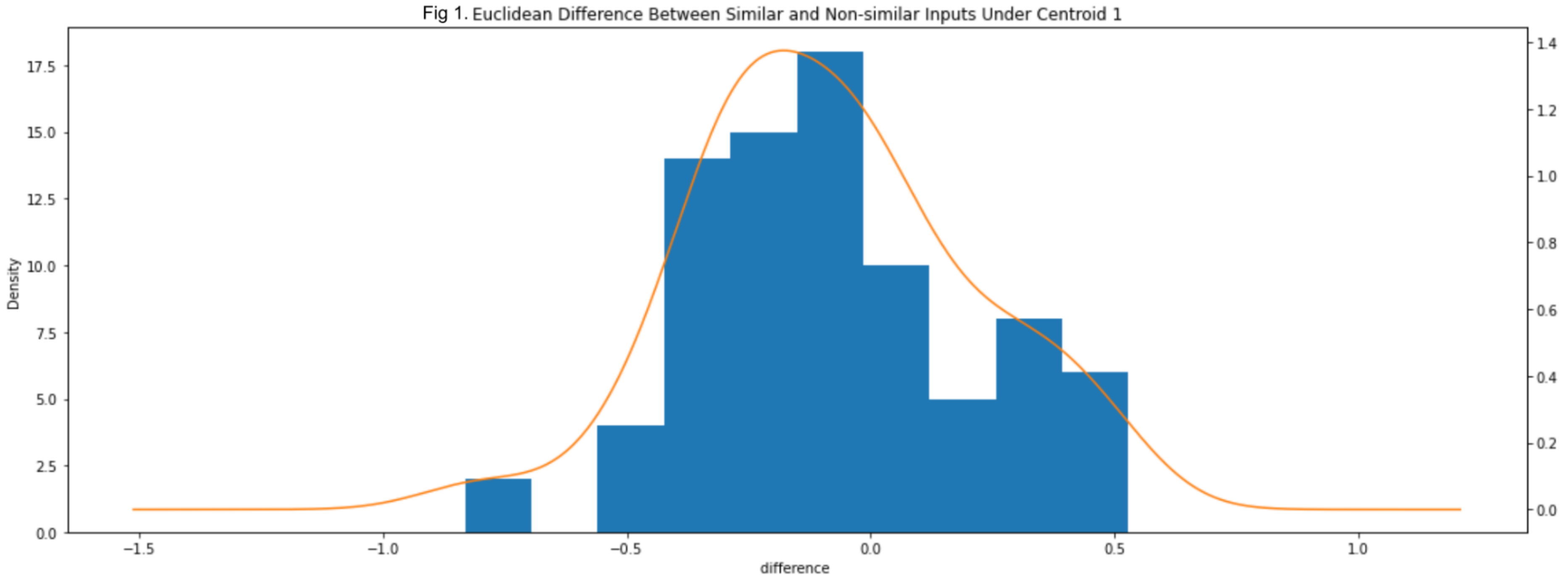
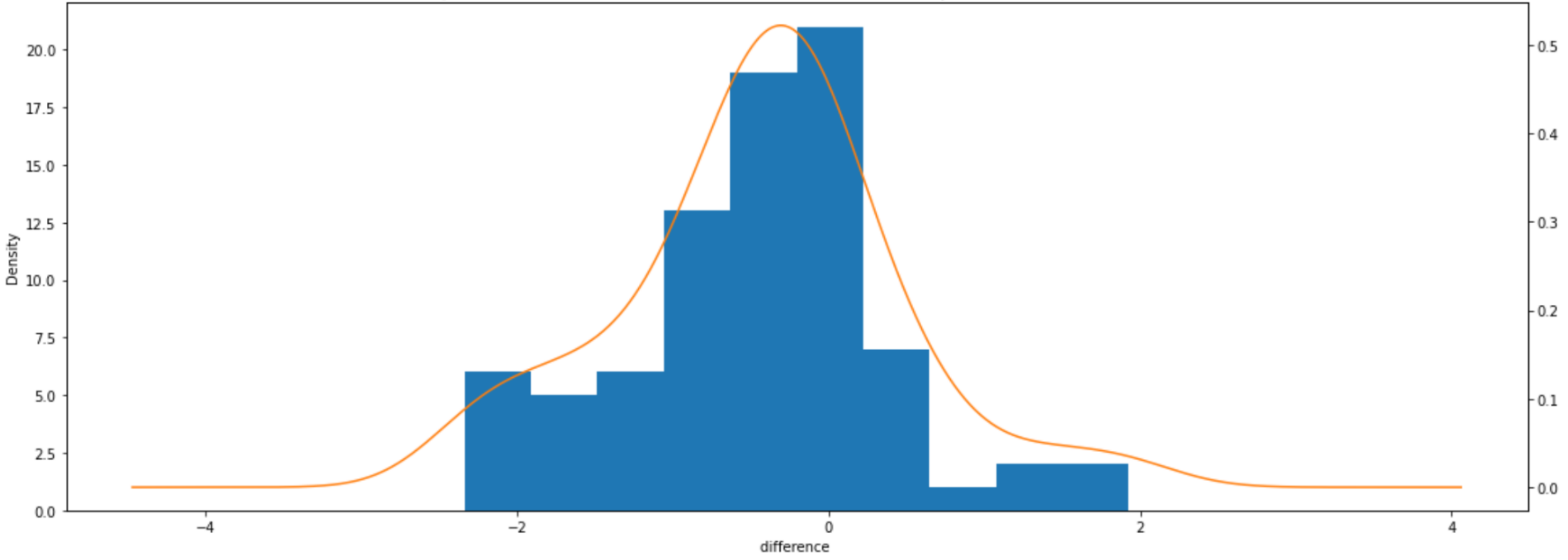


Fig 2. Euclidean Difference Between Similar and Non-similar Inputs Under Centroid 2



08) Developer Test Plan

Test #	Success Criteria	Purpose	Test Action	Expected Result
1	1) b)	Website successfully runs, can be accessed on all major browsers	Execute “main.py” to initiate website on local system Access “http://localhost:5000/” on Google Chrome, Microsoft Edge, Firefox	Website can be accessed at “http://localhost:5000/” on all browsers
2	1) d) i)	Corresponding keywords from user input can be fetched from SQL database	User inputs a word that is already in the database	Database returns corresponding keywords of said user input
3	1) c) and d)	Form data is posted when submit button is pressed	Input 3 words into the search boxes in form and press the submit button or press ‘enter’	Upon pressing button or ‘enter’ key, website redirects to loading page suggesting that user inputs have been sent for processing using the POST method
4	1) d) (iii) (2)	Website redirects user back to home page when user input has no search result	Input random letters into the search boxes in the homepage form and press submit or press ‘enter’ key	Upon pressing submit, the website should redirect the user back to the homepage, suggesting that the user input does not have a corresponding search result when connected to the Google Podcast API.
5	1) d) (iii) (1)	Website redirects user back to home page when user input has no search result	Input singular and plural words into the search boxes in the homepage form and press submit or press ‘enter’ key	The keywords returned for both user inputs should be the same from the backend point of view.
5	1) c) & 5) b)	User input and corresponding keywords can be added to SQL database	Enter user inputs into form in the homepage	User input and corresponding keywords are updated in the database which can be viewed upon search in the previous searches page
6	2) h)	(Error)Podcast homepage URL RSS feed not found	Enter a word that returns a podcast homepage where its RSS feed cannot be accessed	Outputs a message stating that there is an error parsing the RSS feed
7	2) g)	404 (Page not found) error handling	Input a URL of a page which does not exist	Redirects user to an error 404 page not found page

8	1) d) iii) (1)	Stopwords are eliminated and plural words are changed to its singular form from list of keywords for each input	Print out keywords before and after processing to check if stopwords and plural words have been cleaned	Expect that stopwords are eliminated and plural words are changed to its singular form in keyword list
9	5) d)	Home button routes to homepage of website	Press home button	Upon pressing Home button, website directs user to homepage of website
10	1) b)	User-friendly GUI (minimalistic, easy to understand and easy to use)	Homepage easily naturally directs user to input in form, otherwise, user presses button that routes to previous_inputs page	User should have no problems using the product
11	1) a)	Requires Minimum Input	Input only 3 words into search boxes in form	Program should process inputs and redirect user to results page after loading is complete
12	1) c)	Correctly reads user inputs into a session after request for inputs	Input 3 words into form on homepage	Print out the datalog from the session to see if it matches what user has inputted
13	1) c)	Directs to results page after loading is complete	Input 3 words into form on homepage	Directs to results page after loading is complete and displays processed data and embedded frame from Google Podcast API
14	5) a)	Embedded frame is rendered and shows established connection to Google Podcast API	Input 3 words into form on homepage	embedded frame is connected to Google Podcast website
15	5) a)	Embedded frame directs correctly to URL of recommended search keyword	Check if recommended keyword from processed data in results page match the search request for the URL in the embedded frame	Recommended keyword matches search request within URl of embedded frame
16	5) a)	Results page contains processed data that correctly shows user inputs, relevance	Enter the same inputs twice and check if the processed and rendered data on results page are shown the same	All elements both times are the same and user inputs match the user inputs shown on the results webpage

		of recommended keyword, the recommended keyword and a word cloud		
17	5) b)	Previous input page correctly renders a data table with record of previous user inputs and corresponding keywords	Click on button to direct to Previous Input page	Upon clicking the button, the user is directed to a new page with a displayed data table of user inputs and corresponding keywords.
18	5) c)	Previous input page directs correctly to the wordcloud page when user enters input into search box.	Enter a string into the search box	Upon submitting the user input, the page redirects to another page with an interactive word cloud of corresponding keywords
19	5 d)	Each page contains the button that redirects back to the homepage	Click on each page of the website.	Upon clicking the button that directs to each page, the homepage button still appears for each page.
20	4) a)	Upon clicking the header of either the inputs or the keywords column, it will sort the rows of the pressed column alphabetically.	Click on either column	The rows of that column are sorted alphabetically.
21	4) a)	Upon clicking the same header of a specific column again, it will reverse the sorted order of the column rows.	Click on the same column of the data table again.	The rows of that column are displayed in reversed order.
22	3) a)	Keywords are created and stored in the database for a specific user input	Enter a new input in the input box of the homepage, and wait for it to be processed. Then, go to the Previous inputs page and enter the word you entered, checking if corresponding keywords are produced for that	Corresponding keywords for that specific input is displayed in datatable of the Previous inputs page.

			user input.	
23	3) b) i)	A high relevance score should be displayed for 3 similar user inputs.	Enter 3 similar words into the input box in the homepage of the website.	A high relevance score (>80%) should be displayed in the Podcast recommendation page.
24	3) b) i)	A low relevance score should be displayed for 3 dissimilar user inputs.	Enter 3 dissimilar words into the input box in the homepage of the website.	A low relevance score (<50%) should be displayed in the Podcast recommendation page.