SOFTWARE DESIGN SPECIFICATION

Project: **LiveSearch**

Abdelrahman Abdelhaq

Winter Experiential Honors Directed Design- ENGR 493

Winter 2020

4/19/2020

[1.0 Introduction 5](#_Toc38319410)

[1.1 Goals and objectives 5](#_Toc38319411)

[1.2 Statement of scope 5](#_Toc38319412)

[1.3 Software context 5](#_Toc38319413)

[1.4 Major constraints 5](#_Toc38319414)

[2.0 Data design 6](#_Toc38319415)

[2.1 Internal software data structure 6](#_Toc38319416)

[2.2 Global data structure 6](#_Toc38319417)

[2.3 Temporary data structure 6](#_Toc38319418)

[2.4 Database description 6](#_Toc38319419)

[3.0 Architectural and component-level design 6](#_Toc38319420)

[3.1 Program Structure 6](#_Toc38319421)

[3.1.1 Architecture diagram 8](#_Toc38319422)

[3.1.2 Alternatives 8](#_Toc38319423)

[3.2 Description for Components 8](#_Toc38319424)

[3.2.1.1 Processing narrative (PSPEC) for component Homepage 8](#_Toc38319425)

[3.2.2.1 Component Homepage interface description. 9](#_Toc38319426)

[3.2.1.2 Processing narrative (PSPEC) for component SearchPage 9](#_Toc38319427)

[3.2.2.2 Component SearchPage interface description. 9](#_Toc38319428)

[3.2.3.2.1 Sub-Component SearchPage.LiveSearch processing detail 9](#_Toc38319429)

[3.2.3.2.1.1 Interface description 9](#_Toc38319430)

[3.2.3.2.1.2 Algorithmic model (e.g., PDL) 9](#_Toc38319431)

[3.2.3.2.1.3 Restrictions/limitations 10](#_Toc38319432)

[3.2.3.2.1.4 Local data structures 10](#_Toc38319433)

[3.2.3.2.1.5 Performance issues 10](#_Toc38319434)

[3.2.3.2.1.6 Design constraints 10](#_Toc38319435)

[3.2.3.2.2 Sub-Component SearchPage.Subjects processing detail 10](#_Toc38319436)

[3.2.3.2.2.1 Interface description 10](#_Toc38319437)

[3.2.3.2.2.2 Algorithmic model (e.g., PDL) 10](#_Toc38319438)

[3.2.3.2.2.3 Restrictions/limitations 11](#_Toc38319439)

[3.2.3.2.2.4 Local data structures 11](#_Toc38319440)

[3.2.3.2.2.5 Performance issues 11](#_Toc38319441)

[3.2.3.2.2.6 Design constraints 11](#_Toc38319442)

[3.3 Software Interface Description 11](#_Toc38319443)

[3.3.1 External machine interfaces 11](#_Toc38319444)

[3.3.2 External system interfaces 12](#_Toc38319445)

[3.3.3 Human interface 12](#_Toc38319446)

[4.0 User interface design 12](#_Toc38319447)

[4.1 Description of the user interface 12](#_Toc38319448)

[4.1.1 Screen images 12](#_Toc38319449)

[4.1.2 Objects and actions 16](#_Toc38319450)

[4.2 Interface design rules 19](#_Toc38319451)

[4.3 Components available 19](#_Toc38319452)

[4.4 UIDS description 19](#_Toc38319453)

[5.0 Restrictions, limitations, and constraints 19](#_Toc38319454)

[6.0 Testing Issues 19](#_Toc38319455)

[6.1 Classes of tests 19](#_Toc38319456)

[6.2 Expected software response 20](#_Toc38319457)

[6.3 Performance bounds We will want to ensure that the performance of the application remains above 30 fps while also maintaining a strong connection to the database. This will ensure minimal lag and short loading times for downloading and uploading scenarios and landscapes. 20](#_Toc38319458)

[6.4 Identification of critical components 20](#_Toc38319459)

[7.0 Appendices 21](#_Toc38319460)

[7.1 Requirements traceability matrix 21](#_Toc38319461)

[7.2 Packaging and installation issues 21](#_Toc38319462)

[7.3 Design metrics to be used 21](#_Toc38319463)

[7.4 Supplementary information 21](#_Toc38319464)

# 1.0 Introduction

## 1.1 Goals and objectives

LiveSearch is a software that searches for results of other sites as users are typing. The overall aim of this project is to provide a template that gives a functional search feature that can be implemented and expanded upon as the user desires.

The goal is to being an enhanced version of a basic search tool that provides extra features while maintaining its simplicity. These features include:

* Live Searching: Each keystroke renders the most relevant results. Instead of pressing search constantly, the user is able to type more or less in order to see their intended results
* Subjects: The subjects are displayed to the user, so that results that are strictly part of that subject are the only ones that are shown
* All Results: The user is able to see all the results in the database with one button

## 1.2 Statement of scope

The efficiency and accuracy of the live searching is the main software requirement for this project, and it will be based off data that is stored in the MySQL database. The live search tool will search based off name of the article, the article description, or the subject of the article. The subjects will also be presented to the user so that they can easily see all subjects available, and the articles that are specific to the certain subject. The deliverables of this project will be a website that contains sample data and is able to be searched by the three areas mentioned.

## 1.3 Software context

The software will be used by anyone who wants to use a template for live searching and expand and personalize it. All the code will be available on GitHub and can expand on the code that is given to them. Users are able to either implement the given features or customize to their liking for each aspect.

## 1.4 Major constraints

* Reachability of the database: Implementation is limited to using MySQL database
* The program was only tested to work on Google Chrome and Mozilla Firefox
  + The program cannot be used with Internet Explorer
* Limited time: Project had the time frame of the semester

# 2.0 Data design

## 2.1 Internal software data structure

* HomePage – Main site that holds all the components
* SearchPage – Component that holds the search bar, the subjects, and all the results.
  + Contains fetch function for returning results
* Routes/index – Index for the back end functionality of the program
  + Establishes MySQL connection and listens for a GET request
* NavigationBar: Displays the top Navigation Bar on every page

## 2.2 Global data structure

* Live Searching – handleOnInputChange(): Function used to detect the press or deletion of a key in the search bar. Sets the state of the current query, and starts loading, and calls fetchSearchResults().
  + fetchSearchResults(): calls fetch from the back-end and sets the state of the search results
* Subjects - handleOnInputChangeSubject(): Function used to detect the current subject that is selected. Sets the state of the current query, and starts loading, and calls fetchSearchResults().
  + fetchSearchResultsSubject(): calls fetch from the back-end and sets the state of the search results

## 2.3 Temporary data structure

* There are no temporary files or structures that are being used by the program itself.

## 2.4 Database description

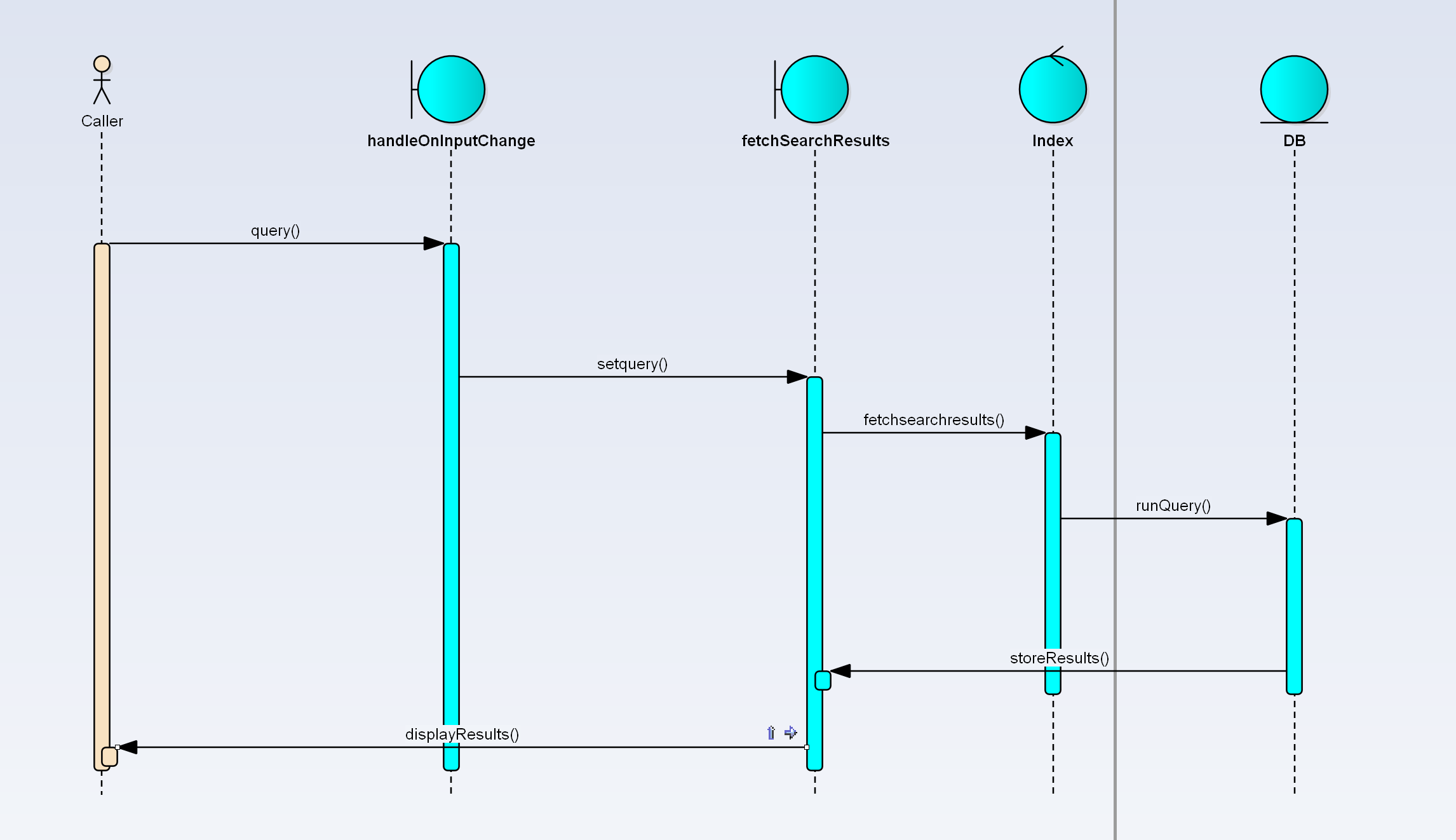
The project will be using a MySQL database to store all of its information. The database includes the id, title, description, subject, and link to the actual article stored.

# 3.0 Architectural and component-level design

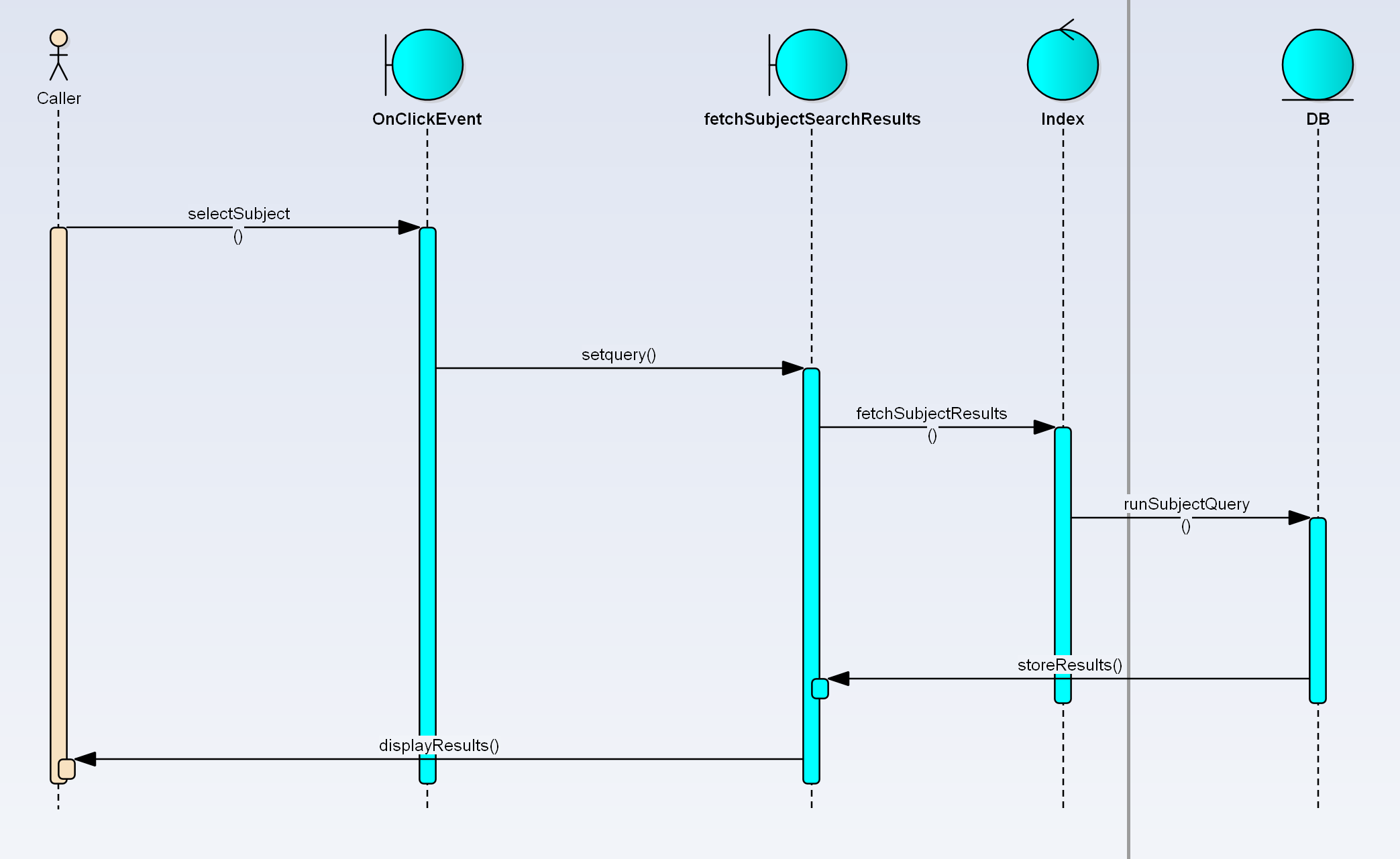
A description of the program architecture is presented.

## 3.1 Program Structure

**Live Search**

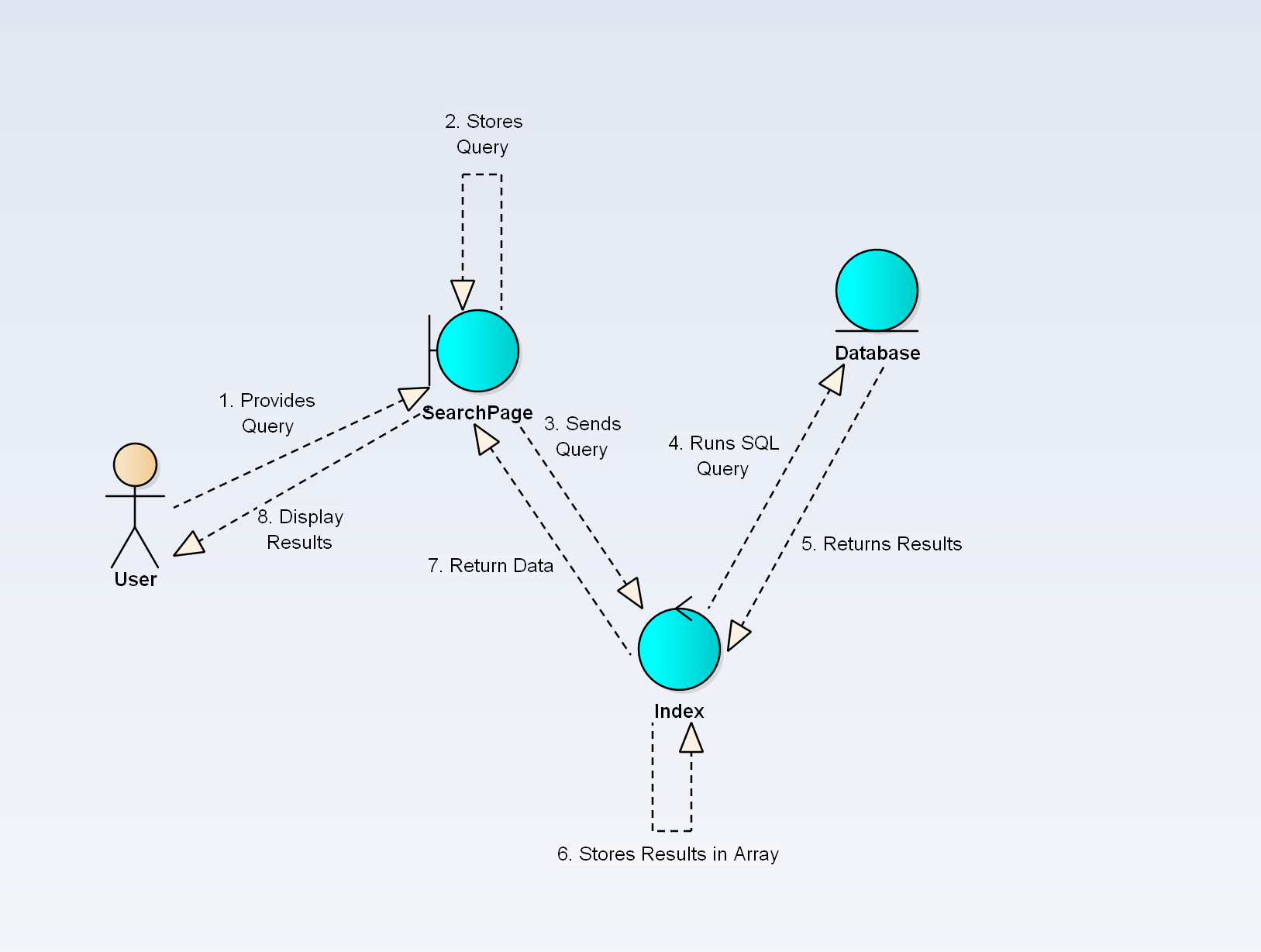


**Figure 3.0 Sequence Diagram 1 – Sequence Diagram for User searching for a result**

**Subject Search** 

**Figure 3.1 Sequence Diagram 2 – User Searching for a specific subject**

### 3.1.1 Architecture diagram



**Figure 3.4 Communication Diagram - Represents the architecture of the system and shows the interaction’s actors and system objects have with one another.**

### 3.1.2 Alternatives

The only functionality that can have an alternative is the method in which the data is received, where Axios can be used instead of the fetch() function.

## 3.2 Description for Components

A detailed description of each software component contained within the architecture is presented. Section 3.2 is repeated for each of n components.

### 3.2.1.1 Processing narrative (PSPEC) for component Homepage

A processing narrative for component n is presented.

Provides the main page in which all other components and their functionalities are displayed in

### 3.2.2.1 Component Homepage interface description.

The Homepage contains and displays the navigation bar, search bar, subjects and the results

### 3.2.1.2 Processing narrative (PSPEC) for component SearchPage

A user will use the search bar and the given subjects in order to perform a search.

### 3.2.2.2 Component SearchPage interface description.

The user will interface directly with the search page through subjects and the search bar

### 3.2.3.2.1 Sub-Component SearchPage.LiveSearch processing detail

A detailed algorithmic description for each sub-component within the component n is presented. Section 3.2.3 is repeated for each of the m sub-components of component n.

### 3.2.3.2.1.1 Interface description

Subcomponent LiveSearch will be the provided search bar that is available to the user on the Homepage.

### 

### 3.2.3.2.1.2 Algorithmic model (e.g., PDL)

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.13.0/css/all.css"  
 integrity="sha384-Bfad6CLCknfcloXFOyFnlgtENryhrpZCe29RTifKEixXQZ38WheV+i/6YWSzkz3V"  
 crossOrigin="anonymous"/>  
{/\*heading\*/}  
<h2 className="heading">Live Search</h2>  
<label className="search-label" htmlFor="search-input">  
 <input  
 type="text"  
 name="query"  
 value={query}  
 id="search-input"  
 placeholder="Search"  
 onChange={this.handleOnInputChange}  
 />

### 3.2.3.2.1.3 Restrictions/limitations

The user must be on a modern browser. The user can only get results for data that is stored in the MySQL database.

### 3.2.3.2.1.4 Local data structures

The LiveSearch will have the functions handleOnInputChange() and fetchSearchResults() in order to fetch the results

### 3.2.3.2.1.5 Performance issues

Internet speed may play a role in the speed in which results are retrieved.

### 3.2.3.2.1.6 Design constraints

There are no design constraints for the LiveSearch sub-component.

### 3.2.3.2.2 Sub-Component SearchPage.Subjects processing detail

A detailed algorithmic description for each sub-component within the component n is presented. Section 3.2.3 is repeated for each of the m sub-components of component n.

### 3.2.3.2.2.1 Interface description

Subcomponent subjects is a set of buttons that the user can click in order to retrieve the results for the specific subjects.

### 3.2.3.2.2.2 Algorithmic model (e.g., PDL)

<div className="searchSubject">  
 <br />  
 <button type = "radio" id="entertainment" value = "Entertainment" onClick={this.handleOnInputChangeSubject}>Entertainment</button>  
 <button type = "radio" id="evolution" value = "Evolution" onClick={this.handleOnInputChangeSubject}>Evolution</button>  
 <button type = "radio" id="food" value = "Food" onClick={this.handleOnInputChangeSubject}>Food</button>  
 <button type = "radio" id="health" value = "Health" onClick={this.handleOnInputChangeSubject}>Health</button>  
 <button type = "radio" id="sports" value = "Sports" onClick={this.handleOnInputChangeSubject}>Sports</button>  
 <button type = "radio" id="all" value = "All" onClick={this.handleOnClickEvent}>All Results</button>  
  
  
</div>

### 3.2.3.2.2.3 Restrictions/limitations

User must be on a modern browser.

### 3.2.3.2.2.4 Local data structures

The Subjects sub-component will have handleOnClickEvent() and fetchSubjectSearchResults() in order to process and display the data.

### 3.2.3.2.2.5 Performance issues

There are currently no known performance issues

### 3.2.3.2.2.6 Design constraints

There are currently no known design constraints

## 3.3 Software Interface Description

### 3.3.1 External machine interfaces

* Input devices
  + Mouse and keyboard
* Network connections
  + Wi-Fi/LAN

### 3.3.2 External system interfaces

* Database
  + MySQL
* Browser
  + Chrome, Firefox

### 3.3.3 Human interface

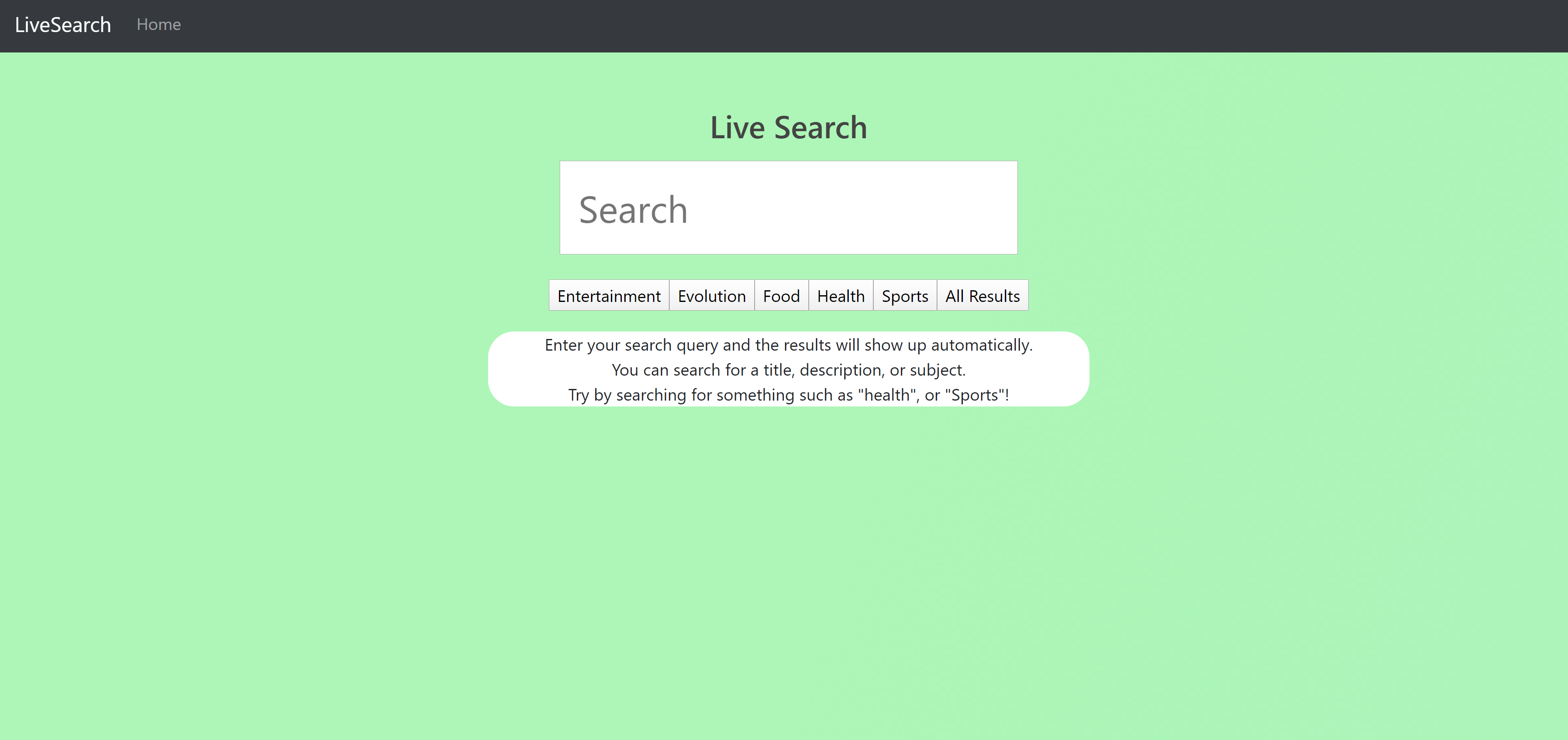
The interface will consist of a webpage that allows the user to view search results.

# 4.0 User interface design

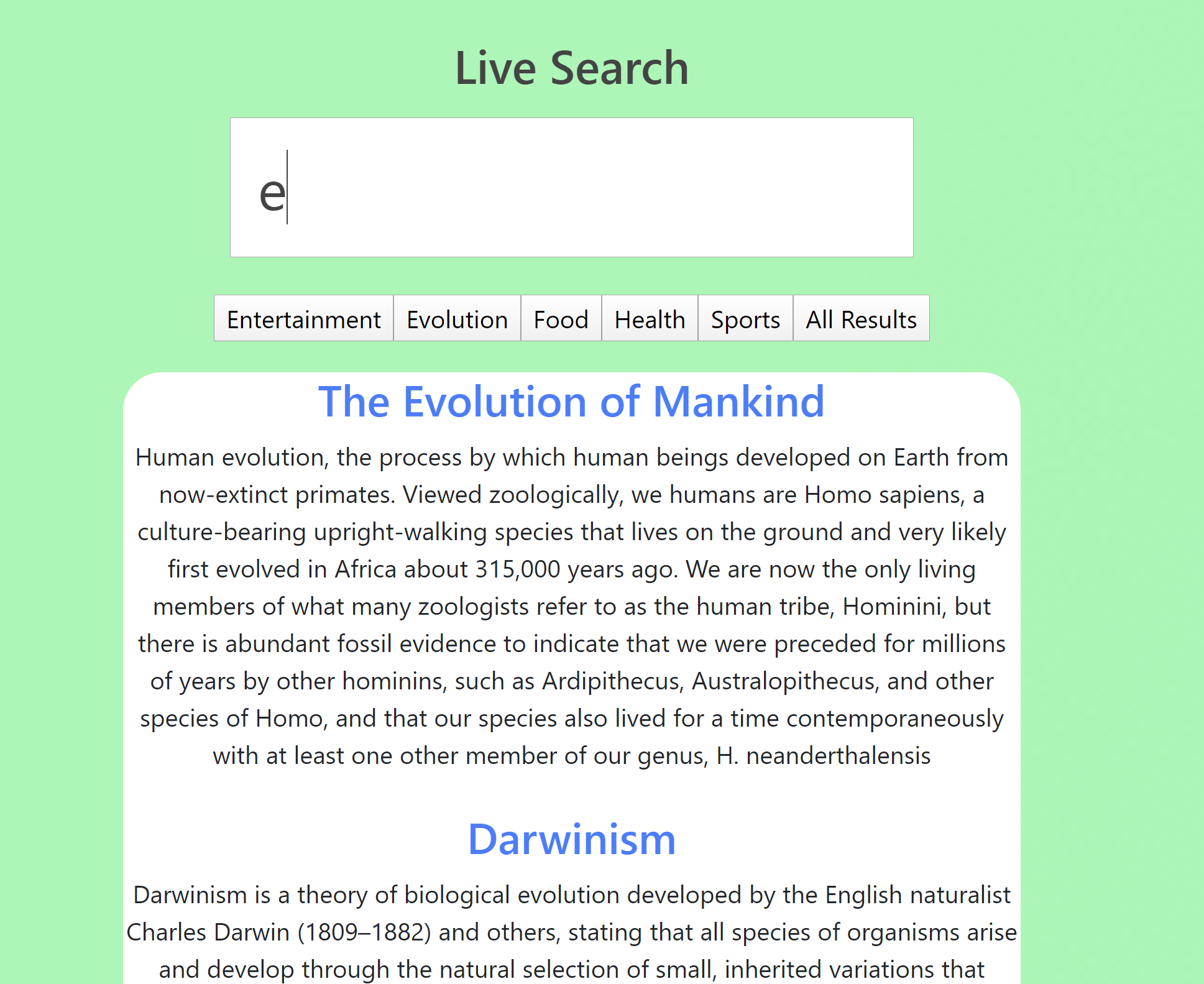
## 4.1 Description of the user interface

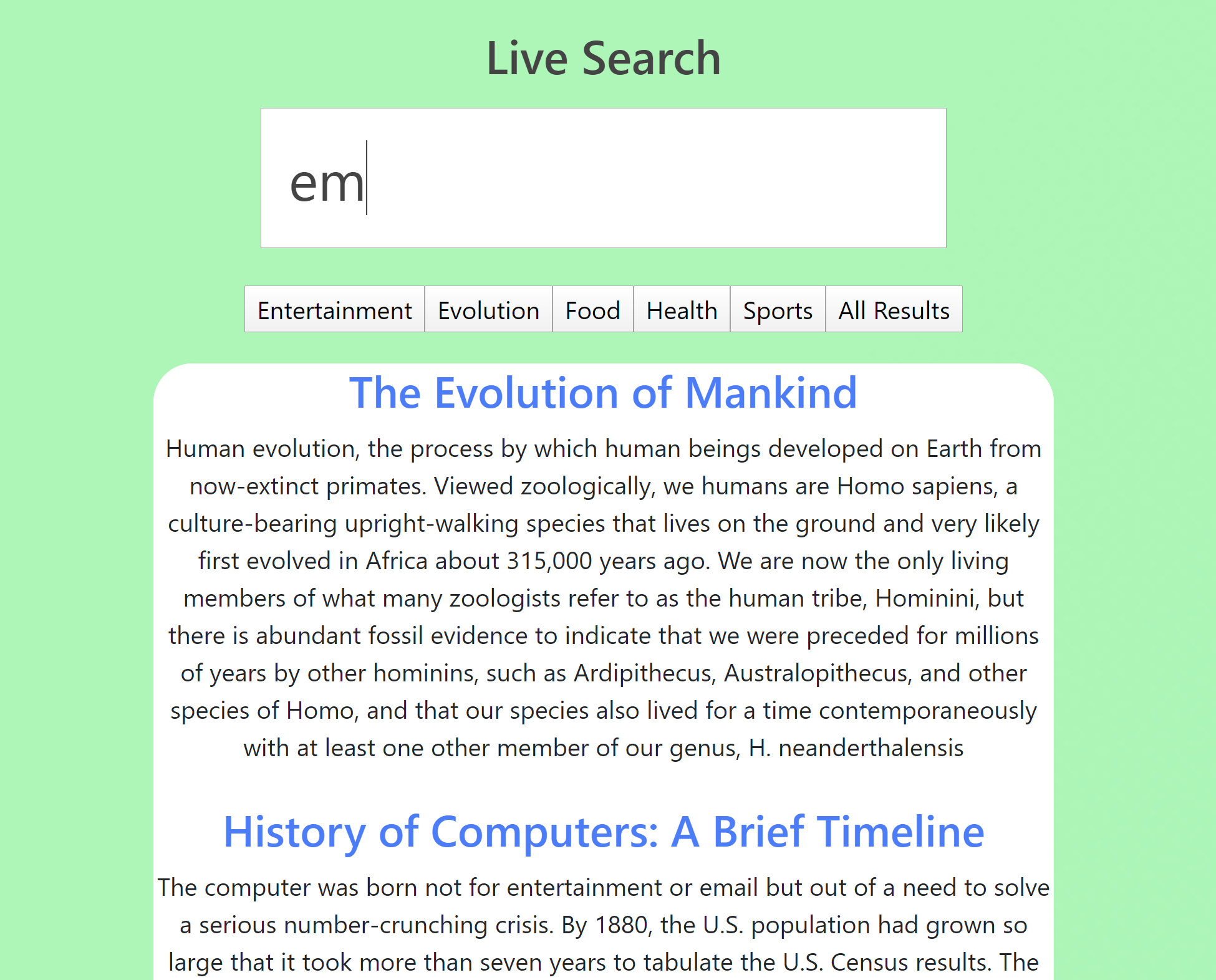
### 4.1.1 Screen images

Below are the screenshots of the user interface design. More information on each of these UI components can be found in **section 4.1.2**.

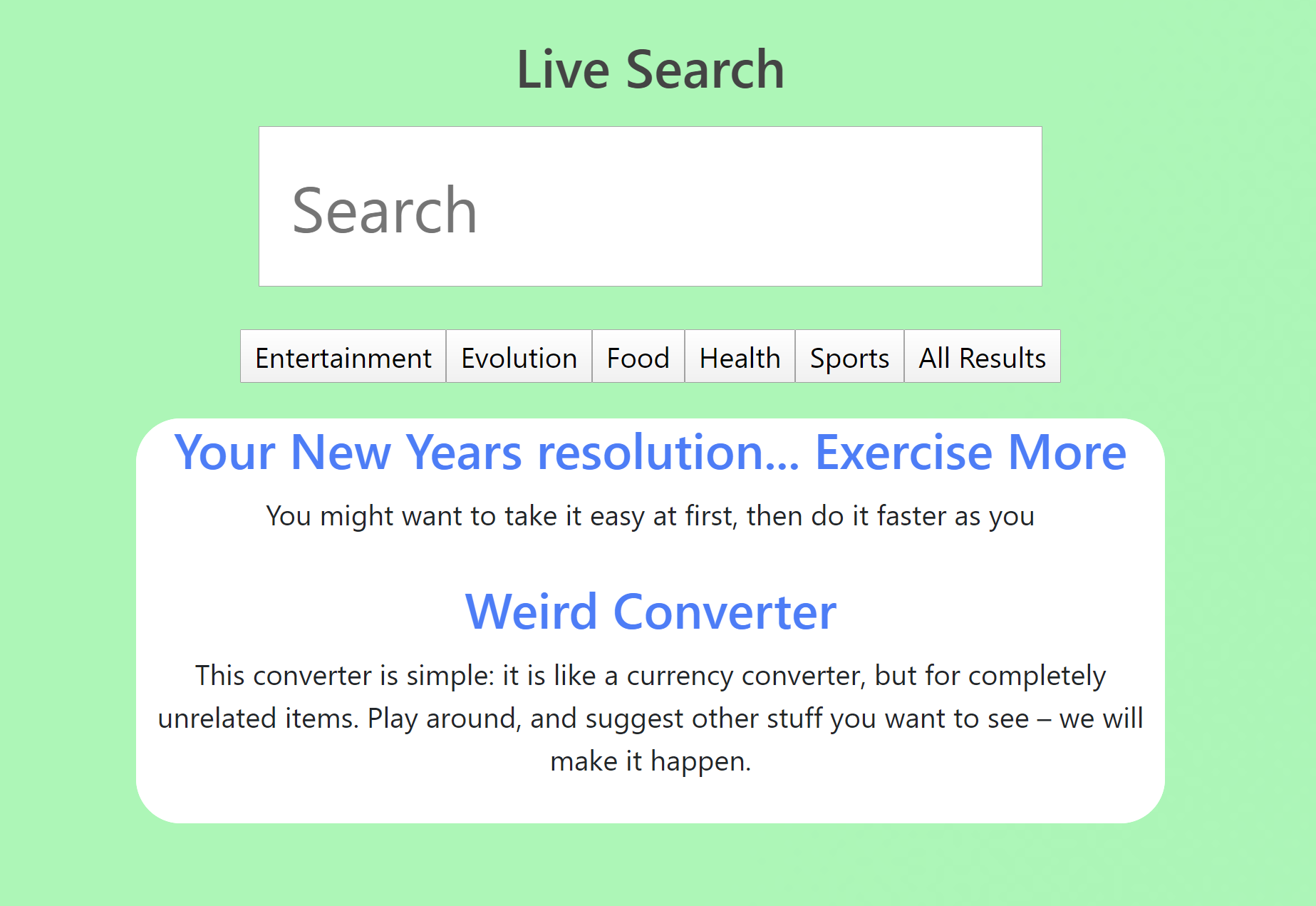


**Figure 4.0 Main page**





**Figure 4.1 LiveSearch – Functionality according to what is typed**





**Figure 4.2 Search the subject of entertainment vs searching all results for entertainment**

### 4.1.2 Objects and actions

Listed below are the different UI elements found in the software:

|  |  |  |
| --- | --- | --- |
| UI Object (Parent) | UI Object (Child) | Description |
| Homepage | SearchPage | Displays the Search bar and subjects |
| NavigationBar | Displays the top navigation bar |
| Searchbar | Live Search | Tools used to modify the landscape. |
| Subjects | Allows for the user to move an object. |
|  | Allows for user to select an already placed object. |
| Removes object from landscape. |
| Rotates the object clockwise in the landscape. |
| Rotates the object counterclockwise in the landscape. |
| Lists all objects in the landscape. |
| Removed all objects from the landscape. |
| All Results | Return all results in the database. |
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|

## 4.2 Interface design rules

* The interface must be simplistic.
* The interface must return results on the same page
* Interface must consist of clear instructions of what results will show

## 4.3 Components available

* Live Search is shown for the User
* Subject is shown for the User
* Results are shown after search

## 4.4 UIDS description

* UI Design Elements
  + The requirements for the User Interface was given through research and developer abilities

# 5.0 Restrictions, limitations, and constraints

* The project is dependent on a reliable internet connection.
* The project must be used on a modern browser. Currently it is only tested on Chrome, Firefox, and Internet Explorer. Internet Explorer has been shown to not work.

# 6.0 Testing Issues

## 6.1 Classes of tests

|  |  |  |
| --- | --- | --- |
| **Test Class** | **Description** | **Inputs** |
| **Homepage** | Testing the homepage to ensure that it displays the navigation bar, search page, subjects and results properly. | * <SearchPage/> * <NavigationBar/> |
| **Search Page** | Testing the functionalities of the search page feature. Ensure that the correct results are displayed, the results are returned in a timely manner, and the ability to search by subjects. | * fetchAllResults() * fetchSearchResults(query) * fetchSearchResultsSubject() * handleOnInputChange() * handleOnInputChangeSubject * handleOnClickEvent() |
| **Index** | Ensuring that the index communicates with the database properly. The port is also open, receiving queries and returning results properly. | * app.get('/results') * app.get(‘/subject’) * app.get(‘/all’) * mysql.createConnection() |

## 6.2 Expected software response

|  |  |
| --- | --- |
| **Test Class** | **Expected Result** |
| **Homepage** | Page displays Navigation Menu, Search bar |
| **Search Page** | User searches through search bar or subjects, and results are displayed below. |
| **Index** | Back end takes the query, sends the correct SQL query and stores the intended results |

## 6.3 Performance bounds The application should take well less than a second to process every keystroke and result showing up.

## 6.4 Identification of critical components

The critical components to LiveSearch include the live searching and subjects functions. Live search is the basis of the whole project, and subjects allow the user to sort their results.

# 7.0 Appendices

## 7.1 Requirements traceability matrix

## 7.2 Packaging and installation issues No packaging or installation is required

## 7.3 Design metrics to be used The time to query is the main metric that will be used

## 7.4 Supplementary information

* Pressman, R. S., & Maxim, B. R., PhD. (2015). Software engineering: A practitioner's approach (8th ed.). New York: McGraw-Hill.
* Pressman & Associates, Inc., Adaptable Process Model Document Templates: System Specification. (2006). Retrieved June, 2019, from <http://www.rspa.com/docs/Systemspec.html>