The idea was to first understand the scenario and time scope the implementation.

So, I went through by solving the following questions and empathizing the needs of the user.

* Look for a simple and clear design for the user and is intuitive and friendly
* Which APIs can be utilized .
* Understand the api documentation so that its possible to make a REST call from the application. (This takes a bit of time as the documentation and the process varies per engine).
* Create an interceptor for making api calls and a service that can be utilized to integrate the API.
* Start with a single api integration and handle the results mapping for google API.
* Logging the whole search object that be utilized for history and later in future can be utilized to handle the cache response mechanisim.
* Show the relevant information based on user search , i.e to show content and the history

After the api was integrated and have basic structure in place, and bindings done. Thought about the integration of Bing search api.

* Every browser has its own response , So have mapped the response so that every engine finally has a single view. This way the application is scalable to different search engines and can actually be an integrated search web application.
* Once the positive scenarios were handled , added the edge cases and error case handling mechanism to clearly indicate the use with the feedback.
* Added the pagination code to handle the next and previous as beta version to showcase the possibility we can have as future.
* Refactored the UI design after the final changes.

Later worked on setting up the test structure and dependencies

* Added the unit test cases and e2e test cases that cover the basic idea of doing the functional testing.

Some Backlog items—

Have the logging in place that can be integrated with a backend / session /localStorage to handle the cached responses based on the search term.

The logging object is design in a way to identify the “Search engine” , “Search query” and “ResultData” that can be enriched and made smart.