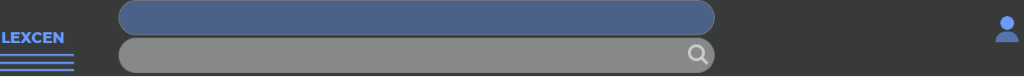
**6.0 Application Header**

Runs across the top of the page and appears on **every page** of Lexcen. The purpose is to house features common across all users and functions and efficiently and effectively convey exactly where they are and what is available.

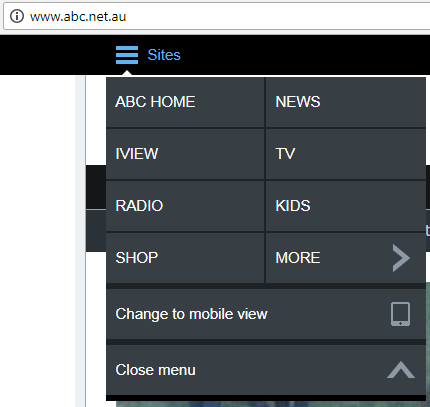
* Menu toggle (**III**) - To be placed top left and show/hide the navigation menu.
* Lexen Logo - To be placed immediately on the right of **"III",** image provided in the banding file -
* Search – A two-line search box. The first line is used for search input, whereas the second displays scope of the workspace. Centre aligned.
* User – A user avatar, with the name of the user in text below it. Clicking on it allows you to logout and redirects you to the lexcen website.

6.1 UI Wireframes

Header



Menu style/functionality to be based on <http://www.abc.net.au/>

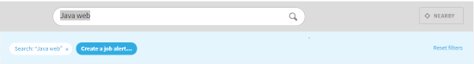


User Avatar – We need to agree on if we want to use an icon or just a circle with the first initials from name like cisco.

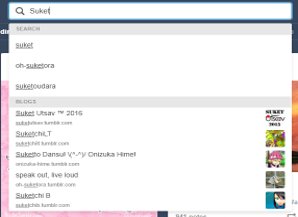
Logo – Provided in the separate file. We need to use the “CYAN” coloured half cloud version. As below.



Search – example of two lines



Ordering of results – Lexcen DB First, then web



5.2 User Story Flow Chart (TBA)

5.3 Screen Logic

**Menu**

Level 1



Level 2



Each of these items would either open a new link outside the application in a new tab or load the body of the application with a new page.

Menu data should be populated using a Backend API call, could just read from a menu Json object.

**Search**

A context aware search mechanism that allows users to first search and then reduce the effort required to complete the action by collating and prepopulating as much information as possible.

**Sub Elements**

1. Type forward – Displays recommended items in a list of 4 from Laxen and4 from the WEB (google to start with and we will add others in the future).

Type forward search will show the following

- Existing Customers - Customer Name - {SolutionID} Solution Description. The results will Hyperlink to the "Workspace" loading the solution.

- If no customers are found - it will display top three results from the GooglePlacesAPI - BuisnessName - business address. Clicking on these will take you to a new page for reviewing the data collected for customer/site creation. We should consider using the dashboard list view to populate the search results \*Future usecase\*.

2. Scope/Tags - Immediately under/above the search input box would be the Search Scope box. This would display the current filters being applied to the search. Along with the ability to reset all filters and start from scratch. When you are working on a "Solution" these tags would be automatically populated with Tag1. Customer Tag2. Solution description being worked on. Tag1. Customer would be locked and not allowed to be when in the context of a workspace. \*Future usecase\*.

**User Avatar**

Always - Display first letter in Caps for the users first name.

On mouseover – it will we need view see the basic details of the user (Full name, email, Application role and give the user the option to sign out)

On sign-out redirect the user to the Lexcen.com

5.4 data

Searchdatalexcen.csv -- list of records we could use to filter for searching customers already in lexcen.

Searchdataweb.csv -- list of records we could use to filter for searching customer info from google.

User list – Just use the three of us for now and a default application role of “Pre-sales eng”

**Acceptance criteria**

Verify non functional requirements are met including:

* Browser support: Chrome, IE, Explorer, Safari and Firefox
* Desktop, iPAD and Android tablets should be supported
* Screen resolutions of (TBD) result in a usable interface and the modal pop up is displayed correctly
* The interface does not present any significant lag after clicking the “Add” icon before displaying the modal pop up

The following software design guidelines are met:

* The list of display actions in the modal popup may be extended by changes in the backend without requiring a change of client side code
* Client should leverage capability of AWS AppSync SDK
* The client – backend exchange is restrained to a singular GraphQL query noting responses maybe cached for reuse within the client side code and maybe pre-populated at the creation of the call flow (for developer to decide)

The following regression testing items are noted, ensuring no bugs or undesired change in behaviour are introduced:

* None