Search through your notes/files

**Environment**

Considering we have less UI Components to use and more backend logic, It is preferable to look into the frameworks best suited for backend logic. But we would still need to research into Frontend frameworks.

*Possible Options for Frontend:*

1. **Angular.js**
   1. Can be used to develop mobile & web apps. Mobile apps with Ionic
   2. It is a full fledged framework for sw development including testing with no need of additional libraries.
   3. Learning curve is medium as it is a huge library.
   4. Performance is affected as the app becomes more complex & dynamic but recent updates have improved performance data. Even the app size is lesser than the react app.
   5. App structure is fixed & has MVC structure
   6. It goes well along with MEAN stack (Mongo, Express, Angular, Node)
2. **React.js**
   1. Can be used to develop mobile & web apps. Mobile apps with React native
   2. It is a UI only framework providing components but needs libraries for external support like Redux for state management, stack navigation for routing.
   3. Learning curve is low due to it’s minimalistic nature.
   4. Performance is good due to light-weight & virtual DOM structure
   5. App structure can be flexible & provides developer with freedom to experiment

*Possible Options for Backend:*

1. **Express.js** 
   1. Built on top of node.js to provide middleware support to help manage our servers & routes.
   2. It’s robust API allows users to configure routes to send/receive requests between the front-end and the database (acting as a HTTP server framework).
   3. A good advantage with express is how it supports a lot of other packages and other template engines such as Pug, Mustache, EJS and a lot more.
   4. Focused on browsers making templating & rendering easy. Supports MVC.
   5. Low learning curve, lots of resources available online
2. **Next.js** 
   1. Built upon front-end library React.js
   2. It provides Server Side Rendering
   3. It also provides the same option as React “Build once, Run everywhere”. It can run on Web, Mobile & Desktop.
   4. It offers automatic code splitting and filesystem-based routing.
   5. It has built-in CSS support called CSS-In-JS
   6. Also has Webpack Configurations to Continuous Deployment
   7. <https://levelup.gitconnected.com/a-simple-next-js-frontend-for-a-multilingual-website-ae31a17387e2>

*Factors to choose the framework:*

* Complexity of frontend - *Low*
* Complexity of backend - *High*
* Speed of the application (backend/frontend side) - *Backend fast*
* Learning curve - *Low*
* Flexibility & resources available (like npm packages, libraries)

Database:

**Searching Algorithm/Library**

***Elastic Search*** as indexing and searching library

*Benefits*:

* Search analytics engine, distributed & horizontally scalable
* Seamless Integration, accessed over HTTP through the RestFul API
* Enables full-text searches which indexes every line in the document

*Available as:*

NPM Package: <https://www.npmjs.com/package/elasticsearch>

*Resources:*

1. <https://www.elastic.co/guide/en/elasticsearch/client/javascript-api/current/introduction.html>
2. <https://www.compose.com/articles/getting-started-with-elasticsearch-and-node/>