**Application Architecture**

Our web app will be deployed on the Carolina CloudApps OpenShift platform.

The backend components will include a MongoDB database and a webserver, as well as Go processes for querying the database through a RESTful API and for periodically polling updates to the Scopus citation database.

The webpage for the search tool will use the React framework, along with some JS scripts and SASS/CSS style sheets from the Gillings website.

**Diagram

Description automatically generated**

**Decision 1: Integration with sph.unc.edu**

*In order to ensure that our search tool is functional, accessible from the existing Gilling’s website,  and feels seamlessly integrated we decided to build the page as an external site using the school’s stylesheets and a subdomain of sph.unc.edu.*

**Problem** We want our search tool to be accessible from the existing website, with relatively seamless integration. It should “feel” like a component of the existing site.

**Constraints** The current site is done in WordPress and uses a customized version of the UNC theme. Customizations are primarily graphical, so adding scripts and interactive HTML directly to the site isn’t really an option.

**Options** We considered either embedding the search tool in an iframe or linking to it as an external site. The iframe route limits a lot of the features we can use, and could be harder for the site’s maintainers. On the other hand, building an external site puts the onus on us to make sure the navigation and styles are easy to keep up to date with the school’s website.

**Rationale** We’ve provisionally decided to build the page as an external site. We can use the school’s stylesheets and a subdomain of sph.unc.edu to give the appearance of integration without having to add functionality to the WordPress theme. In terms of implementation, the main part of our web app will stay the same regardless of whether it gets embedded in the WordPress site, so we can change this later if needed.

**Decision 2: Hosting service**

*In order to host our web app to provide ease of access for the school we decided to use Carolina CloudApps.*

**Problem** We need our web app to be hosted on a server, long-term and in a way that leaves the school in control of future upkeep.

**Constraints** Ideally, we want to use infrastructure already available to the school, in order to avoid unnecessary costs and keep things under the control of our clients. At the same time, we would like the hosting service to be flexible enough for us to choose our server and database technologies.

**Options** Our best options were to host our project either on the Gillings SGPH web server or on the Carolina CloudApps platform. The Gillings web server is fully under control of the school, and would mean cooperating with the school’s operators to deploy our web app. Their server mainly uses PHP with MariaDB. The CloudApps platform is still within the domain of UNC, and uses OpenShift to host web applications. OpenShift can accommodate a variety of server and database architectures, and can be synchronized with our GitHub repository.

**Rationale** We went with CloudApps for our hosting because it seems to allow more flexibility and portability in our development process. It should still be free to our clients at Gillings, but should not require direct maintenance by their web site operators.

**Decision 3: Database**

*In order to be able to query, check, and clean citations from Scopus we decided to use MongoDB due to both familiarity and flexibility.*

**Problem** Our app needs to be able to query the citations obtained from Scopus, but also provide some way to manually check and clean these results. Any modifications or filters set by an administrator should be stored and reflected in future queries. In addition, results should be limited to current faculty only, which will need to be updated over time.

**Constraints** The publications relevant to current faculty may potentially include research from earlier in their careers, so “new” entries in the database may be added either when new papers are published or when new faculty members are hired. A list of faculty IDs and department affiliations is already maintained by the school and should serve as the basis for which faculty to include in search results.

**Options** The databases available through CloudApps were PostgreSQL, MariaDB, or MongoDB. PostgreSQL seems like the more flexible of the two tabular SQL databases, whereas MongoDB is a document-based NoSQL database.

**Rationale** We decided to use MongoDB in part because of our familiarity with it, but also because we expect it to be more flexible for adding or restructuring our data to optimize queries based on user searches.

**Decision 4: Server program**

*In order to connect our database to our search tool webpage we decided to use “Go” to create a process for Scopus polling and a process for serving the website and handling queries. Both processes would then interact with the database.*

**Problem** The server program needs to connect our database to our search tool webpage. It should also connect to the Scopus API to get regular updates for our database.

**Constraints** The OpenShift platform supports several major languages for the server implementation. Among these, we should choose something that will be easy to maintain in the future, but also easy for us to develop within the short timeframe we have this semester.

**Options** The options we felt most comfortable with based on prior experience were Node.js, Python, or Go. Concerning Node.js and Python our group had similar experiences from previous classes and were fairly comfortable using it. Experience with Go however ranged from experienced to inexperienced yet we considered it a valuable tool to learn and knew it could provide us with the needed functionality.

**Rationale** We chose Go as we considered the learning experience to be very valuable and felt comfortable working with it despite our varying ranges of experience. We decided to use Go to create one process for Scopus polling and one process for serving the website and handling queries.  Both of these processes would then be able to  interact with the database.

**Decision 5: Administrator authentication**

*In order to  provide access to administrative features with authentication we decided to use UNC Shibboleth to verify the user information.*

**Problem** Our app needs to provide restricted access to administration features and needs some login authentication tool in order to verify the user.

**Constraints** It should be something that would be able to be used by every employee at Gillings and should be able to secure the imperative information that should only be accessed by verified users.

**Options** We considered using something like Firebase in order to create our own verification system or use the UNC Shibboleth system. By making our own verification system we would have control on how certain aspects of the verification system would be used but this could cause issues with single sign on and having the employees at Gillings make accounts. UNC Shibboleth on the other hand was already available with UNC CloudApps and allows for the use of UNC single sign on. We would however have to make sure we have permission to use this tool and have it work as intended.

**Rationale** We decided to use UNC Shibboleth since it’s already available with CloudApps and would keep in line with everyone at Gillings as they all already have a UNC single sign on account. We are fairly certain we would be able to utilize this tool, as CloudApps was recommended by Gillings, and if we need assistance with troubleshooting we would be able to ask the associates of our client.

**Decision 6: Frontend framework**

*In order to establish our user interface to be intuitive and editable  we decided to use React as our frontend framework.*

**Problem** Our app needs to be able to be viewed on multiple internet browsers with a consistent user interface that is easy to use and able to be altered in the future by Gillings.

**Constraints** We want to use something that would make the design of our UI easy to understand and preferably easy to program. With the use of this tool we should be able to construct a UI that is intuitive for the user as well as intuitive for those who may make adjustments to it in the future.

**Options** We considered using React or WordPress for our front end framework. We knew react was a reliable tool for making front end frameworks although not everyone had experience using it including those at Gillings. WordPress on the other hand is what Gillings had been using in the past for its front end yet we were skeptical of the limitations it might come with the functionality that we wished to accomplish.

**Rationale** We decided to use React in order to construct the front end framework of our project. We understand that React is a reliable tool to create front end interfaces as well as something that would be beneficial to learn in preparation for the future. The team was willing to learn React for this project and decided that it was not worth the risk using WordPress if all of our desired functionality would not be possible through this tool. Although those at Gillings may not have the experience in React, they do have HTML and CSS experience and the read.me that our team produces should be able to bridge their knowledge with the code we produce.