Jan-Lucas Ott
CSC 415
Oct 18, 2016
Project Proposal

Web Oligos https://github.com/ottj3/weboligos

Option 1: This project will build on an existing software project.

Web application: This project will provide a web interface, written in Scala using the Play framework.

Web Oligos will provide a web service front-end for the oligo design software project, allowing users to access the oligo library design functionality via a web interface and API.

The original oligo design software was designed from a “back end” software point of view, and has many UX issues. Creating a web service “front end” for this software would greatly increase both usability and accessibility by masking many of the complications of the original design, such as the UI written in Swing (Java's outdated UI framework), the need for multiple libraries and languages to be installed and linked correctly (Python, Perl, and some intermediates). It would also allow extensibility such as distributing workloads (the underlying algorithm of the oligo designer is extremely thorough and complex, which means it can take a while), exposing a RESTful API, and allowing easy exports to third party web sites (blastn, idtdna tools) used by scientists in these fields.

The main algorithm that the web service will involve taking user requests, storing the details of the request, and then using a producer to create jobs, add them to a priority queue in a broker, which would then be accessed and consumed by worker agents running the oligo design software (with a slightly modified wrapper). The consumer would also be responsible for returning the results to the broker to be stored and returned to the user.
Although this is a simple algorithm, it is intended to integrate a wide scale of technologies from a UI in web languages (HTML, JS), a web server, broker, and workers in Scala, and a monolithic software in Java.

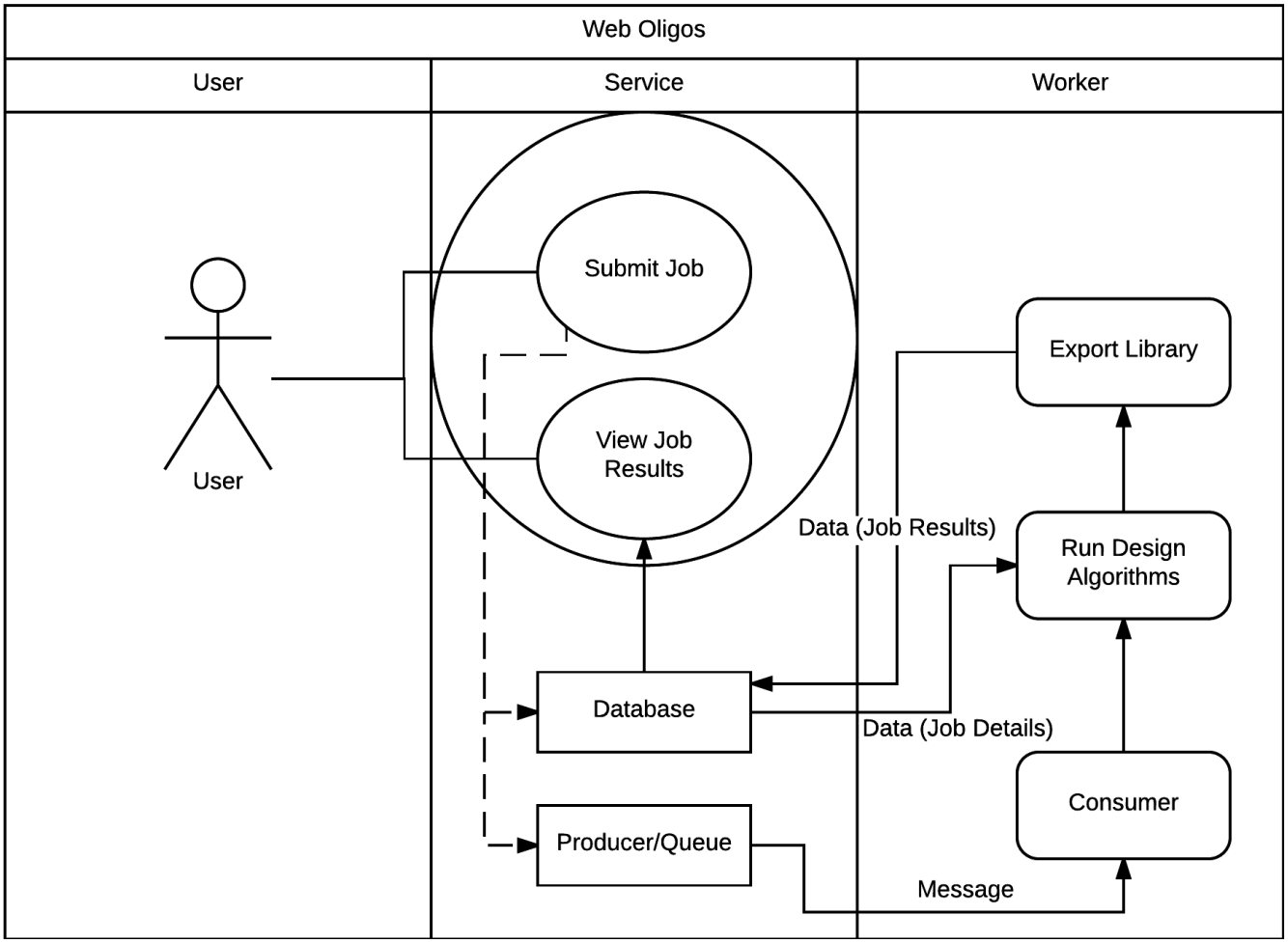
The core part will likely be the broker, which will be based around a priority queue, as described above. This queue will allow jobs to be completely in an extensible and scalable fashion given resources and worker agents. It is also important that the broker is able to store and requeue jobs from the database in the event of a failure.

This project will be a completely new undertaking for me. I have minimal experience in both web technologies and Scala. I expect to use many of the libraries available to me to reduce the learning curve. Scala and the Play framework have many tools and libraries which will help in implementing a broker with a message queue and a database, although the options are still being explored.

I believe learning Scala initially will consist of working through the getting started guide. When it comes to implementing specifics, I will be using resources like the official documentation and the top stackoverflow result on Google, as well as my knowledge of the underlying mechanisms of the JVM and Java.
I plan on structuring my project based on what I learn by looking at existing projects and example projects built on the Play framework or using other technologies that I will be using. The exact user

interface is not my priority, so the HTML/JS/CSS front end will likely be very plain and simple.

The following is a combined use case diagram via swim lanes which also shows how the various components will interact with each other.



Open Source License

There are a few widely used OSS licenses which are worth mentioning. Most of them follow a philosophy that allows reuse, modification, distribution, etc of the software provided that the copyright notice and license is not altered. The MIT and BSD licenses follow this pattern. The Apache and Mozilla Licenses also follow this philosophy, although go much more in depth as to what rights are protected, especially when it comes to patents or trademarks.
On the other end of the spectrum exists the GPL, which is a strict copyleft license that perpetuates itself to every project it touches. Any derivative work, library, or combined software must be re-licensed to the GPL after using GPL-licensed code.

I chose to license my work under the MIT license as it is very widely used and supported, and allows very permissive copying of the software, which I support.