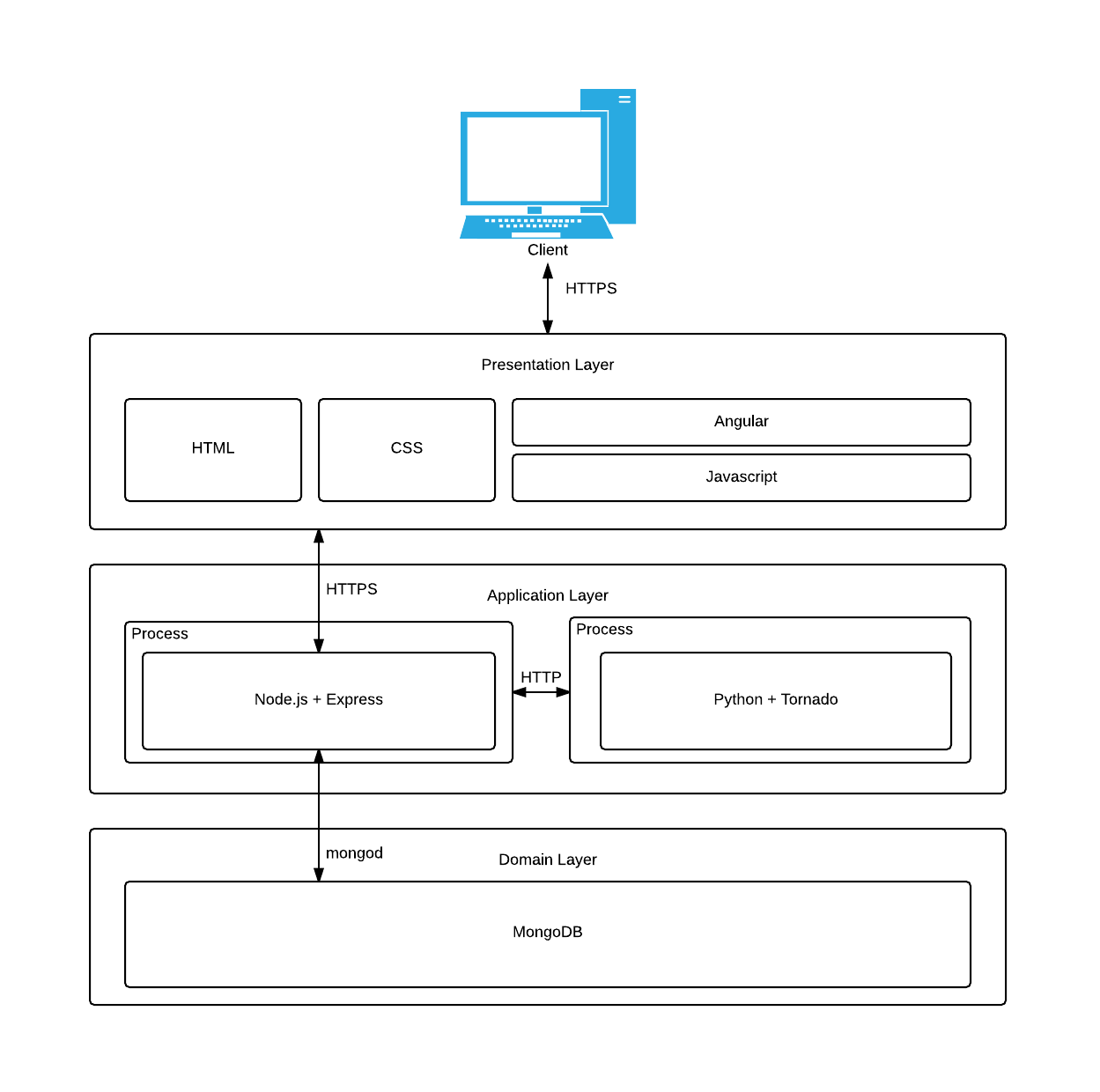
**Architecture Document**

Peter Mikitsh, Kristen Mills, Mike Wideman, Dan Lavoie

Team Rigatoni, Project Linguini

Table of Contents

1. Architecture Diagram
   1. Overview
   2. Diagram
2. Components and Functions
3. Entity-Relationship Diagram
4. Sequence Diagrams
5. Design Rationales
6. Architecture Diagram
   1. Overview  
        
      The system is broken into three major layers: presentation, application, and domain. The presentation layer is the system’s user interface and user experience, and provides a visualization of the data layer. The presentation utilizes Bootstrap, Twitter’s front-end development framework for styling user interface elements and providing a flexible grid system to adapt gracefully to a variety of viewports (desktop, tablet, mobile, etc.). Angular.js provides client-side templating of data and simulated application routing. The application layer provides authentication, authorization, and communicates with processes that perform domain-specific tasks, such as part-of-speech tagging. The application also communicates with the Data Layer, responsible for persisting user credentials, user-uploaded corpora, and analyses.
   2. Diagram



1. Communication Protocol

Communication between the two components of the application layer will be achieved via HTTP. There is an established JSON payload structure to ensure consistency.

Outgoing from Express to Webserver:

{  
 "transactionId": "A unique string associated on per-request basis.",  
 "operation": "The analysis technique to be applied.",  
 "library": "The analytics engine ",  
 "data": ["An array of the id's to the documents containing the data to analyze."]  
}

Incoming from Webserver to Express:

{  
 "transactionId": "A unique string associated on per-request basis.",  
 "operation": "The analysis technique to be applied.",  
 "library": "The analytics engine ",  
 "results": {}  
}

The results object format is dependent on the operation and library specified.

1. Scalability

Because of the computing power required for NLP calculations, scalability is a key concern of our architecture. We will be utilizing several strategies in order to maximize performance and scalability.

3.1 Separation of Concerns

By separating the two components of the application layer, we can more easily provide additional computing power for running analyses. The Node.js and Express component is separate from the Python component. They communicate via JSON over HTTP as described above.

3.2 Usability

The user interface will be designed under the presumption that not all analysis will be completed immediately. As such, users will receive visual feedback as to the state of their pending analyses, and will continue to be able to use the system in other ways while analyses are pending.

3.3 Caching of Common Analyses

Because the system is targeted at educational environments, and because there are predefined corpora included with the system, there is a high potential for repeated analysis requests for the same analysis on the same text. In order to avoid rerunning calculations, the system will provide a mechanism for caching results of analyses for some small length of time.

3.4 Parallel Execution and Chunking of Analyses

Since many analyses involve commutative operations, which will return the same data regardless of the order in which they are run or how large of a chunk of data is being analyzed, being able to work on smaller pieces of text will allow the system to work more efficiently under load. The system will provide a mechanism to break large texts into smaller chunks for distributed processing when possible, using a MapReduce concept and the Pool multiprocessing class.