Integrated Logging Framework for the conversational AI product

To do a proof of concept and a demonstration of an integrated logging framework. This document describes the features that such a logging framework should have.

The broad level requirements of this development is -
A. Allow end-to-end logging, tracing and auditing of all requests served by the product, all its constituent services and components.
B. Configurability of the logging through a Web based UI
C. Tools that can help serve as debugging tool
D. Reporting for longer term system performance analysis.
E. Be independent of deployment concerns such as cloud provider (AWS/Azure) or whether docker is used and which persistence is used (Database or ElasticSearch).

Major Use Cases

To add a level of detail to the above aims, here is a list of features that the integrated logging framework should provide -

A. Basic

1. Logging should be possible from any specific layer or service within the entire product. The product has services written in Ruby on Rails, Java, Python as well as a UI involving HTML/Javascript. So the framework should have interfaces in each such language/environments and be generally cross technology compliant.
2. Logging content can take various formats. Examples are – Simple log messages varying number of fields, audit logs, exception stack traces and context traces - should be accepted by the logging framework.
3. Framework should help identify a request that may be user or system generated. A unique request id that can be generated and assigned to each request can help isolate a request and will serve as the key to trace across services.
4. Individual services may be replicated, auto scaled on-demand in cloud. The logging should be able to capture information from such spawned services.

B. Configurability

1. Configurability required in increasing or decreasing logging granularity by well defined log severity levels, log message formats.
2. Evaluate offering live logging versus file based asynchronous harvesting. Live logging will push logs to the database directly and can therefore require system resources while file based logs show up after a time lag that is needed for harvesting and push to persistence. Configurability here means we should be able to switch between live logging and file based async logging.
3. Configuration updates in logging such as those in previous items 1 and 2 should not require any system restarts.
4. If a logging mechanism has specific dependencies like particular databases or other third part software then plugins should be available for interfacing so that the product itself or its deployment does not suffer an impact.

C. Aid Debugging

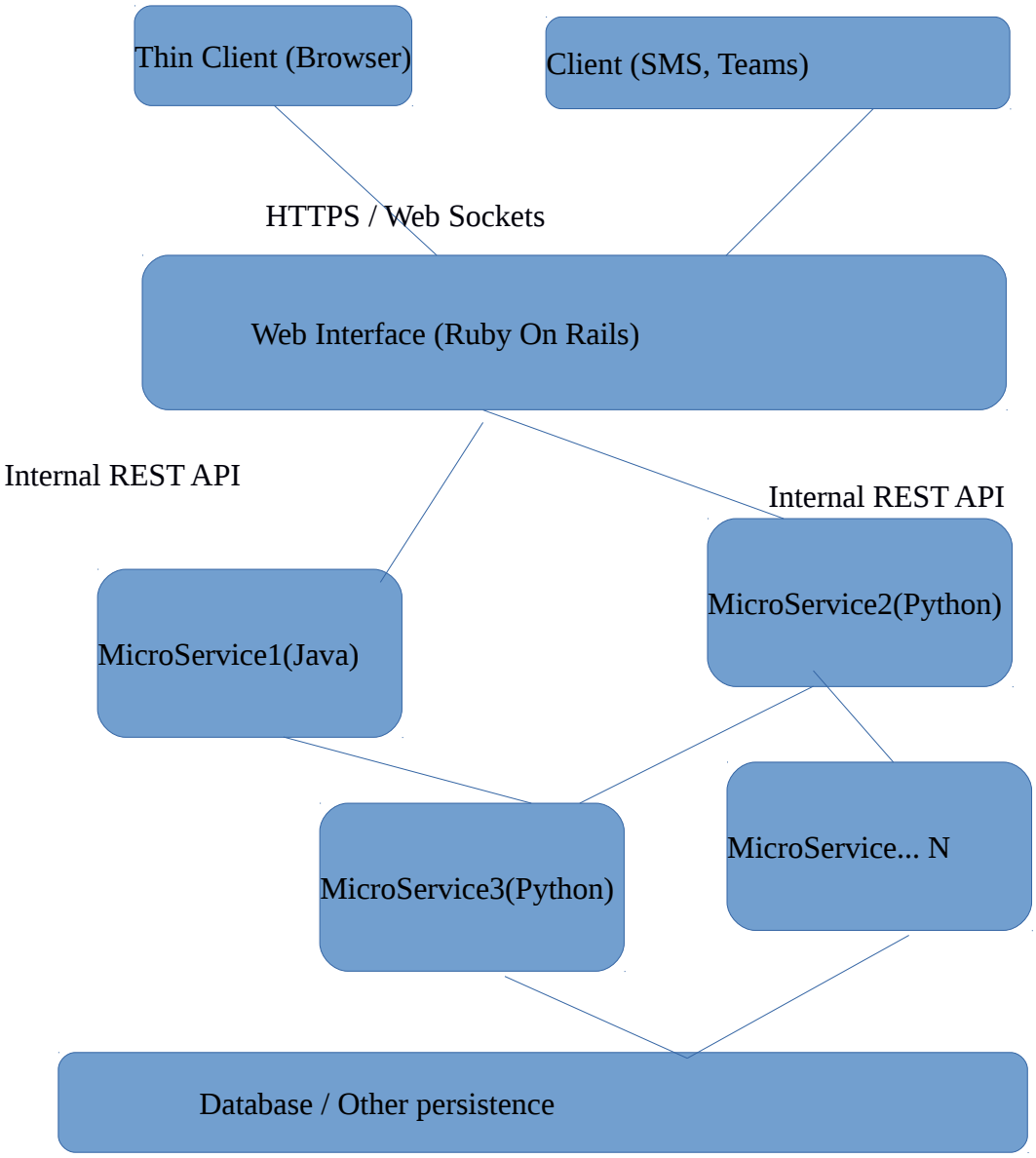
1. When an end user observes a failure or even a success, they should have access to a request id. That request id, when provided should help trace the end to end action of the users request. Such end to end information will obviously start at client end and in most cases involve the web layer and multiple services. This should help debugging reduce the need to replicate exact user commands as far as possible.
2. This information can be aggregated and presented in a UI to further help debugging.

D. Reporting

1. Create reports that are based on aggregating past data over certain time periods.
2. Provide filtering mechanisms based on time periods, log severity levels etc.
3. Visualization of data, particularly, error trends over time periods etc.
4. Provide all such reporting over a web interface that may be accessible in the internet under authentication.

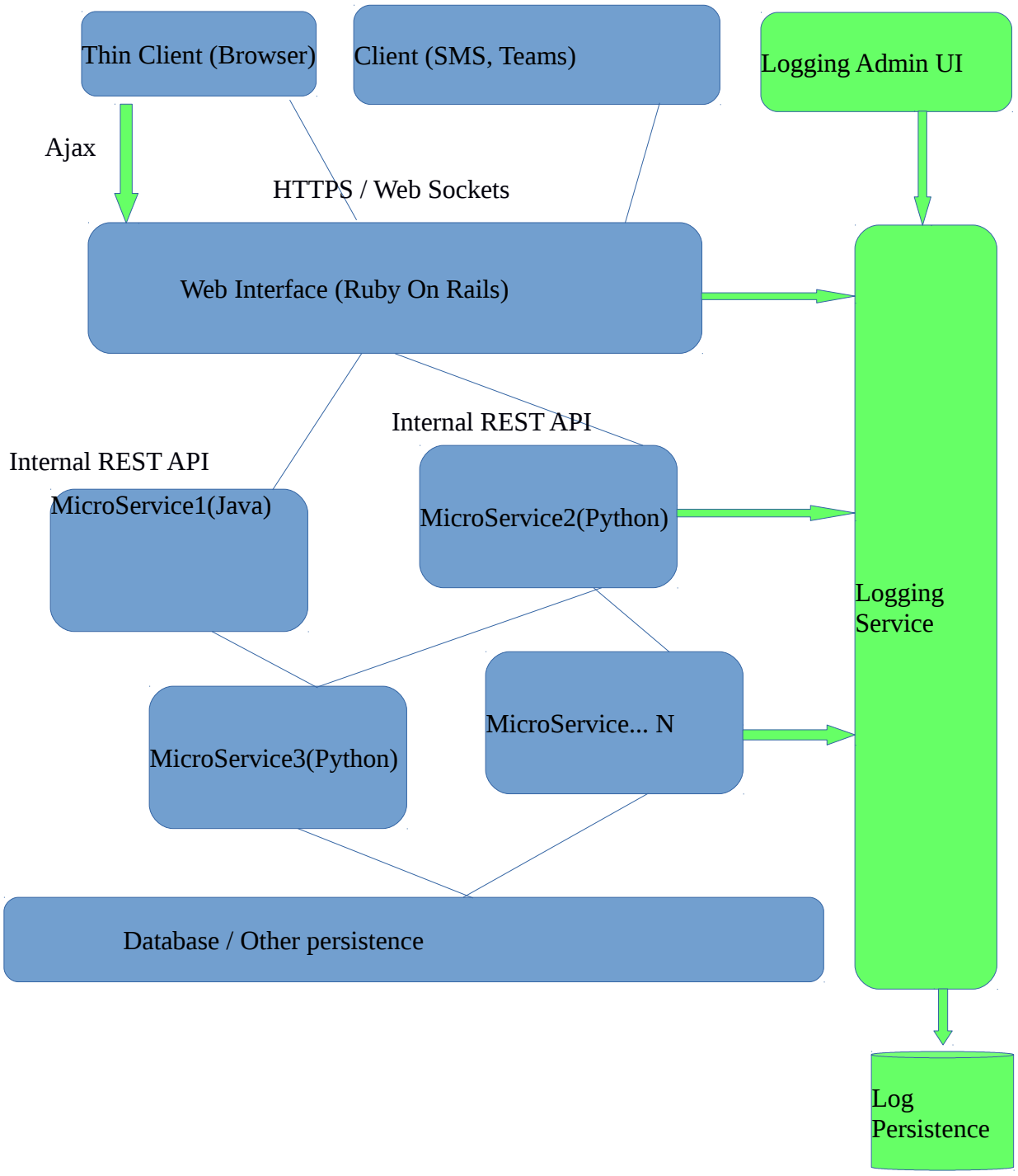
Existing Product Architecture

The following diagram describes the services and their dependencies as far as logging framework is concerned.
The lines between services describe possible remote invocations.



Updated Architecture Proposed

The following diagram only describes the services and their interconnections as far as logging framework is concerned.
New invocations described by Block arrows. New services in green color
Note that this diagram introduces a new logging service but that is not necessarily the only possible design. Another possible design is presented on the next page.



Updated Architecture Proposed [Alternative 1]

The following diagram only describes the services and their interconnections as far as logging framework is concerned.
New invocations described by Block arrows. New services in green color
Note that this diagram is not necessarily the only possible design.