JusticeAI Iteration Summary

Iteration 5

Team Members

Name	Student ID	GitHub ID
Lance Lafontaine	26349188	lancelafontaine
Arek Manoukian	21710389	arekmano
Sylvain Czyzewski	27066333	vynny
Mihai Damaschin	27177895	mihaiqc
Samuel Campbell	26457959	samuel-campbell
Taimoor Rana	26436110	taimoorrana1
Zhipeng Cai	21346482	choitwao

Project Summary

JusticeAI (ProceZeus) is a web chat bot that aims to facilitate access to judicial proceedings involving specific domains of law. Users will have the ability to converse with the chatbot, describing in detail the situation for which they wish to pursue litigation. The system, which will leverage the power of machine learning and natural language processing, will guide the user through a process wherein they'll be prompted with a series of questions relating to their potential case allowing the system to ultimately determine, based on provincial jurisprudence, whether the user has a valid case worth pursuing in the judicial system. Alternatively, the system may also suggest remedies in lieu of legal action if it is deemed unlikely to be in the user's best interest.

Velocity

Iteration 5 was where delivered tangible predictions to users of our system. The primary focus of this sprint was to deliver a system which can predict, based on user input, whether or not the lease must be terminated. During this sprint, we were able to complete **32** user story points.

The following is a list of user stories that were completed in iteration 5:

- #46 - Indicate the understood response from the user's answer (8 points)
- #183 - Claim: Lease Termination (8 points)
- #151 - DEV STORY: Create Machine-Readable Representations of Precedents (5 points)
- #153 - Understand user facts and translate to values (8 points)
- #156 - Dev Story: Address Technical Debt (3 points)

The following story was functionally completed, but removed from active iterations and labeled as blocked until user input data by beta testers is obtained.

- #65 - Reprompt on ambiguous answer (8 points)

Plan for Next Iteration

Iteration 6 (26 Points)

- DEV STORY: Lint Fix for Iteration 6 (1 points)
- Claim: Retaking the rental (8 points)
- Ask only fact questions that are relevant to lease resiliation (3 points)
- Accept user feedback within system (3 points)
- Remove previously understood fact (3 points)
- See previously understood facts (5 points)
- Remove English precedents from the dataset (3 points)

Noteworthy Achievements

This iteration was the first time all of our services (a web client, our backend middleware, our NLP service and our ML service) communicated through REST APIs and operated as a functional system, returning a predicted result based on user input.

Additionally, this iteration's focus was to incorporate our work on defining facts into a benchmarked machine learning model in the context of predicting whether a user's lease termination is justified. With the narrowing of our fact state space, our support vector machine implementation was able to achieve an F1 score of approximately 95% for this specific outcome.

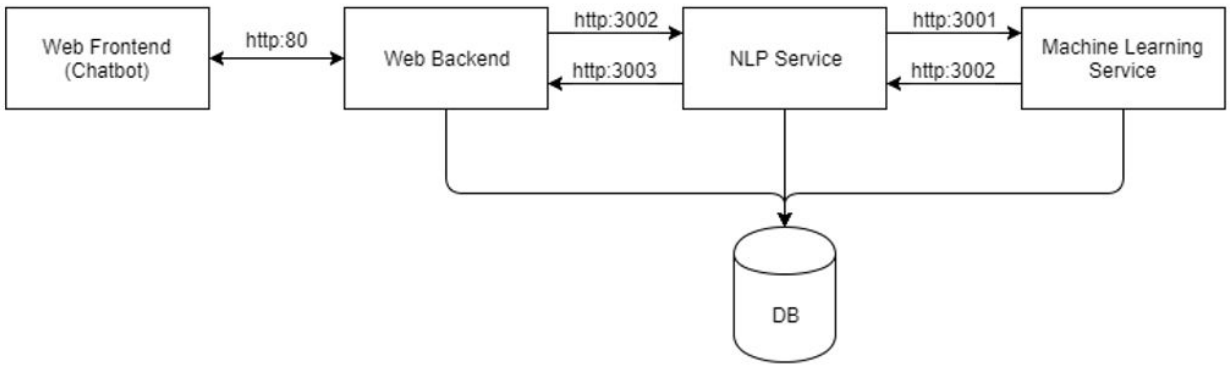
Finally, toward the end of the iteration, our team presented the project's progress and demonstrated the system's capabilities to interested stakeholders from the CyberJustice Lab and the Montreal Institute for Learning Algorithms (MILA). The presentation persuaded the lab in allocating further resources to our project.

These issues (#153 and #183) along with their linked pull requests explores the details regarding these achievements.

Technologies, Architecture and Library Changes

The only noteworthy change to the technologies and libraries used for our project was the removal of our specification for a message queue service. We previously stated that we intended to create this message queue service using the RabbitMQ and Celery technologies. However, we have deemed it unnecessary given the pipeline of processing steps that are required for each request to our system.

The high-level architecture diagram we have submitted for Release 1 (shown below) is now a more accurate description of our system, which features a layered approach as opposed to a centralized model which featured the message queue.



Continuous Integration Processes and Naming/Coding Changes

One of the issues brought up in the previous iteration retrospective was that some accumulated technical debt was slowing development speed. In order to correct this, we opened this issue and addressed some of our technical concerns. The primary tasks that were resolved this iteration concerned Docker build time reduction and external dependency management.

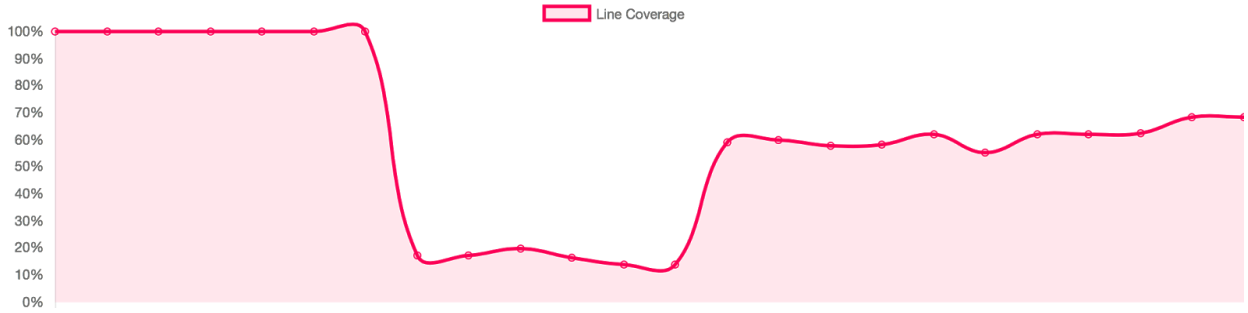
One of the action items from the Iteration 4 retrospective was to change our build configuration to allow Travis CI to pass a build even if there were linting errors. This decision was made as development speed was being hindered by build failures caused by trivial whitespace issues. This iteration resolved this issue by simply printing the code style diffs to the build log instead. We've also begun creating simple development stories for every future iteration to ensure that the code style standards are being followed. This iteration, this PR was merged to make all necessary code style modifications.

Unit Tests and Code Coverage

To keep track of our Code coverage, we use the free **CodeCov** service. Every time a new build is completed successfully on Travis CI, the resulting line coverage is uploaded. At the

time of this writing, our line coverage is at 68%, with a total of 56 unit tests. The test breakdown per service is shown below.

Service	Number of unit tests	Line coverage %
ML Service	22	82.51 %
NLP Service	7	36.29%
Web Client	21	95.89 %
Backend Service	6	39.76 %
Total	56	68%



Line coverage % over the time of JusticeAI.

We have maintained and even slightly increased our coverage from our previous sprint

Iteration 5 Retrospective

What went well

- Efficient Teamwork between teammates of subteams assigned to user stories
- Efficient communication between subteams for different user stories
- Development speed was on point
- First implementation of all services working together successful with much less issues than expected

What went less well

- Always merge branches which deliver core functionality first, and code refactoring afterwards

- Whoever makes a binary should be responsible for uploading to CJL and making sure our init script downloads it.
- Parsing data was done individually. As a result we obtained duplicate work.
- Lack of domain knowledge for the cases we are investigating (need legal aid that was still not supplied to us)
- More explicit documentation of facts implemented in the code are required
- Lack of documentation concerning binary files which lead only a part of the team understanding the purpose of their existence
- Prevention of parallelization of work on one task was not executed properly despite being warned against in previous meetings
- Commenting on a PR should constitute a nonsensical reply such as “NO”. Clarification is required. (Outlier incident)

Measures to try out in Iteration 6

- Need team meeting to cleanup facts together to form a cohesive base of understanding of what should be considered a fact
- Facts should be found as a team or by a particular expert subteam that consults the legal advisors
- Ask Product Owner for domain expertise resources (legal aid)
- Create user stories per iteration for linting our code for style violations
- Attempt to reduce manual intervention in the code implementation