JusticeAI Iteration Summary

Iteration 3

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 3 was where we laid down the infrastructure to get information from the user related to their claim. This includes a front-end where the user can input information by text and uploads files. We also started preliminary work to extract data from precedents so it can be used by our machine algorithms. During this sprint, we were able to complete **51** user story points.

The following is a list of user stories that were completed in iteration 3:

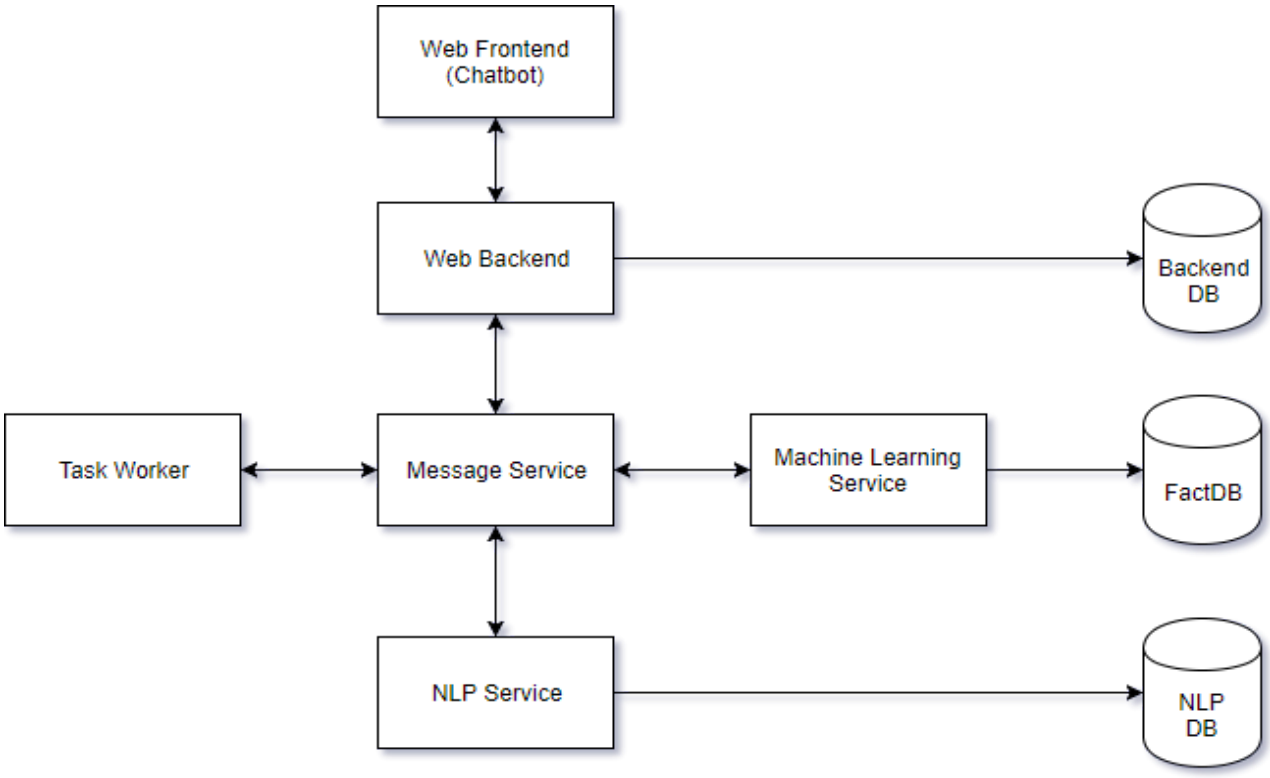
- #88 Evaluate ML requirements (8 points)
- #5 End User License Agreement (5 points)
- #15 Claim evidence gathering (13 points)
- #68 Be able to make the distinction between original case and rectified case (20 points)

- #66 Persist conversation text (5 points)

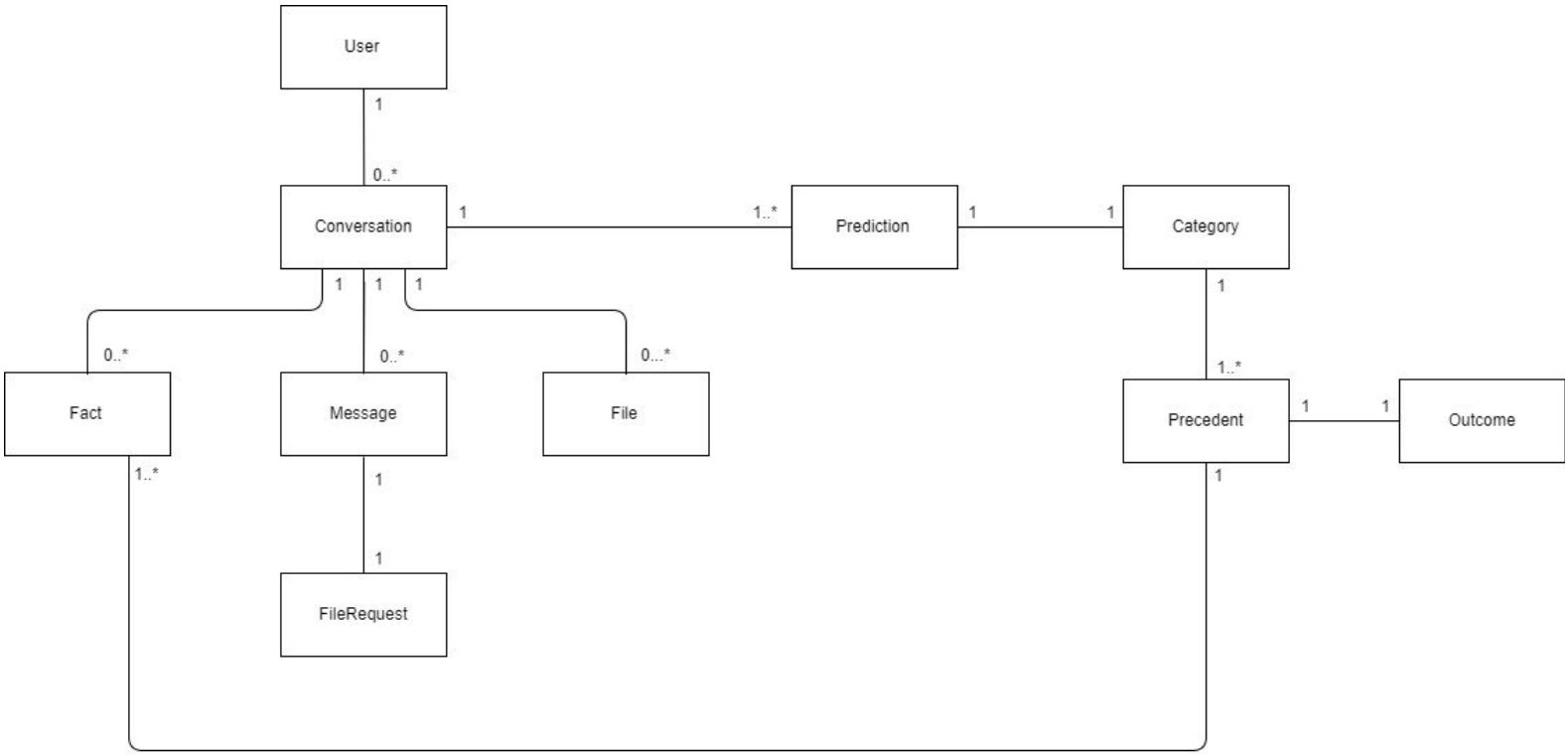
Due to advice from the professor and tutor, we will be limiting our stories to 8 points from now on, considering any story larger than 8 points an epic that has to be broken down into smaller stories. This means that our velocity may end up being lower in iteration 4 than the previous iterations, as the story points were previously determined on a scale set by the team. This does not however entail that less work was done.

Overall Architecture and Class Diagram

Architecture Diagram



Domain Model



Plan for Next Iteration

Iteration 4 (43 Points)

- Reprompt on ambiguous answer (8 points)
- As a user I would like to indicate my user type on a landing page (3 points)
- Better Classification for Tenant/Landlord Problem (8 points)
- Indicate to the user what the system understands (8 points)
- Outcome clustering (8 points)
- Fact clustering (8 points)

Technologies and Infrastructure

Overview

Component		
Tool	Language	Purpose
Web Frontend		
Bootstrap	HTML/CSS/JS	Front end styling and layout
Vue.js	Javascript	Minimal web application framework
Web Backend		
Flask	Python 3	Backend HTTP server
SQLAlchemy	Python 3	SQL ORM
Marshmallow SQLAlchemy	Python 3	SQLAlchemy model to JSON conversion
Message Service		
RabbitMQ	Python 3	Distributing messages between components
NLP Service		
NTLK	Python 3	Speech tagging, word tokenizer, text sentiments...
RASA	Python 3	Intent, entity extraction, classification, etc.
Machine Learning Service		
Keras	Python 3	Supervised Learning framework
Tensorflow	Python 3	Neural Network Machine Learning Framework

Scikit Learn	Python 3	Unsupervised Learning Framework
Task Worker		
Celery	Python 3	Performing background tasks
Database		
PostgreSQL	SQL	Data persistence

Web Frontend

Bootstrap

- Bootstrap is an open source front end framework developed by Twitter. It contains styling for various common web components, such as forms and inputs, as well as providing a convenient grid system that greatly facilitates web page styling and layout.
    - Alternatives: Foundation Framework, pure.css, skeleton
    - Reason Chosen:
        - Team member's past experiences
        - Industry standard

Vue.js

- Vue.js is an open source front end framework for building single page applications. It leverages component based architecture that allows for the creation of an interactive website. Its primary purpose will be to power the visible portion of the chatbot, displaying messages, sending messages to the server, and prompting the user for various interactions such as answering questions or providing files to use as evidence.
    - Alternatives: AngularJS, Angular 4, React
    - Reason Chosen:
        - Low learning curve
        - High performance
        - Small footprint and minimal API

Web Backend

Flask

- Flask is an open source HTTP backend web framework. It is a light framework, dubbed a 'micro-framework', that only supplies the most essential backend functionalities such as HTTP endpoint handling, form validation, and session handling amongst others. It is easily extendible and has supports a variety of middleware.
    - Alternatives: Django, web2py

    - Reason Chosen:
        - Our backend only requires JSON responses and requires no template generation. The alternatives contain too many features that would never be used.
        - Low learning curve and minimal API

SQLAlchemy

- SQLAlchemy is a python based database ORM that allows for the creation of database tables and relations defined in python classes. It greatly simplifies the task of development as it eliminates the need to write and maintain raw SQL queries. In addition, it supports schema migrations which are necessary for a project where models may change with as requirements are added or removed.
    - Alternatives: PeeWee, Django ORM
    - Reason Chosen:
        - Integrates well with flask
        - Established community and a great deal of well written documentation.

Marshmallow SQLAlchemy

- Marshmallow SQLAlchemy is a python library that allows SQLAlchemy models to be serialized into a JSON format. It serves as an extension of the marshmallow library, which allows for python objects to be serialized into JSON. Marshmallow SQLAlchemy removes the need to write custom serializer classes and instead allows a SQLAlchemy model to be retrieved directly from the database,and automatically serialized into a format that can be used by the front end (JSON).
    - Alternatives: Handwritten serialization classes
    - Reason Chosen:
        - Major convenience
        - Support for SQLAlchemy
        - Good documentation

Message Service

RabbitMQ

- RabbitMQ is a flexible, erlang-based, open source message broker with client implementations available in many popular languages (Python, Java, PHP…). It provides a message queue system that allows separate applications to publish messages and subscribe to numerous queues allowing for message consumption.
    - Alternatives: Celery, Amazon SQS, ZeroMQ
    - Reason Chosen:
        - Open Source
        - Multiple client implementations allow for flexibility in choosing programming languages for other system components

NLP Service

NLTK

- NLTK (Natural Language Toolkit) is a state-of-the-art suite of libraries providing comprehensive documentation with hands-on guides for Natural Language Processing with Python.
    - Alternatives: TextBlob, Stanford CoreNLP, spaCy, Gensim, Pattern, TreeTagger
    - Reason Chosen:
        - Apache license
        - Active community with abundant support
        - Customizable (ie. Allows new tokenizer models & compatible with machine learning libraries.)
        - Capable of supporting French tagging
        - Can be upgraded with add-ons (ie. Stanford libraries written in Java)

RASA

- RASA is a powerful natural language tool divided into two essential components handling natural language understanding (Rasa NLU) and multi-turn dialogues with machine learning (Rasa Core). Rasa NLU combines natural language processing and machine learning to perform intent classification and entity extraction from input text utilizing preconfigured processing pipelines composed of prewritten classifiers. Rasa core provides dialogue management (DM) and natural language generation (NLG), simplifying the task of determining what stage the conversation is at (DM) and being able to generate responses (NLG).
    - Alternatives: wit.ai, dialogflow.com
    - Reason Chosen:
        - Open source (Apache license)
        - Not a SaaS solution
        - Fits well with chatbot use case

Machine Learning Service

Keras

- Keras is a machine learning library that simplifies the use of other more complex machine learning frameworks
    - Reason Chosen:
        - Reduces the maintenance cost associated with our machine learning service
        - Makes machine learning code understandable

Tensorflow

- Tensorflow is a library that allows you to build neural networks, to be used for supervised machine learning

    - Reason Chosen:
        - Extensive documentation
        - Community support available for answering our questions

Scikit Learn

- Scikit learn is a library that simplifies vector computation. It simplifies the use of clustering and unsupervised learning algorithms
    - Reason chosen:
        - Simplifies vector manipulation
        - Standard tool for computation in the python language
        - Extensive documentation available

Task Worker

Celery

- Celery is an open source Python-based task queue. Its purpose is to perform potentially long tasks/jobs that are better done outside of the HTTP request/response cycle, in order to prevent delays from the user's point of view. It uses a message broker, such as RabbitMQ or Redis, to receive messages from components requiring a long task to performed, allowing it to continue execution while the task is performed by Celery.
    - Alternatives: Redis Queue, Taskmaster, Huey
    - Reason Chosen:
        - Most widely used python task queue.
        - Supports RabbitMQ as message broker.

Database

PostgreSQL

- PostgreSQL is an open source database management system that allows data persistence and retrieval via SQL queries. It is supported by SQLAlchemy, allowing it to be used in conjunction with an ORM in a Python project. In addition, it is known to be very efficient when working with complex data sets providing very quick join operations.
    - Alternatives: MySQL/MariaDB, MongoDB, Cassandra
    - Reason Chosen:
        - Relational database
        - Liberal open source license
        - Contains the bigserial data type, ensuring complex objects can be stored in the relational database. This will be useful for our natural language and machine learning components.
        - Traditional SQL syntax (Conforms to SQL standard)

Continuous Integration Processes and Naming/Coding Conventions

In order to maintain a high standard of code and release quality, our team has established strict development processes that are enforced by Travis CI, a continuous integration service which run on every commit push, pull request and merge.

Pull Request Process

Pull requests can be created after branching from the master branch or by forking the repository.

If you create any branch, the branch name must be in this format, or our continuous integration service will fail the build and you will not be able to merge:

<issue-number>/<short-description-of-issue>

Example: 87/store-user-session-data

In order for a pull request to be merged, it must have at least 2 approvals by reviewers that can validate your changes. Additionally, all continuous integration services must pass.

Git Commit Style

We will be following a similar approach to the Git commit guidelines.
Every commit in our repository must have a well-formatted message, or our continuous integration service will fail your build and you will not be able to merge:

[#<issue-number>] <very-short-summary-of-changes>
- Detailed bullet of important change 1
- Detailed bullet of important change n

Example:
[#12] Added user login feature
- Created user model
- Added OAuth library to our dependencies for authentication
- Created a login page

Naming and Coding Conventions

Our project will use strict continuous integration services to lint our code on every commit. We have chosen to abide by Google's style guide for all languages. This is enforced with the linters listed below during continuous integration.

- Python
  - Style Guide
    - Google Python Style Guide
  - Tools
    - pylint
    - pep8
      - autopep8
- HTML/CSS
  - Style Guide
    - Google HTML/CSS Style Guide
  - Tools
    - csslint
    - HTMLBeautify
- Javascript
  - Style Guide
    - Google JavaScript Style Guide
  - Tools
    - jshint
    - eslint

Unit Tests and Code Coverage

We expect every pull request to have unit tests. In order to ensure that unit tests are always being written, we are using CodeCov as a service which monitors for unit test code coverage. If the total code coverage falls below a certain preset percentage, or if that pull request introduces a negative diff greater than 1% in our code coverage, our CI build will fail and you will not be able to merge.

Sprint 3 Retrospective

What went well

- Communication on Slack
- Good learning pace for ML, NLP, and legal domain
- System deliverable done
- Have a Plan for ML Architecture
- Standups still consistent
- Research is good
- Team members helping each other
- Prepared for presentation

What went less well

- Didn't run Docker before merging, building dependencies
- Need more people to groom our backlog
    - Story splitting needs to occur
    - Defining user stories with more details (and extra parameters)
- Demo was not on prod
- Last minute work
- Lance only responsible for infrastructure of docker, need better repartition
- Not everyone is familiar with Docker
- UI should not only be Bruce
- Bus factor is one
- Not using Dev Docker

Measures to try out in Sprint 4 (Addressing pitfalls of Iteration 3)

- Groom the Backlog
    - Break down the stories into 13 point stories (below feature-level threshold)
    - Meetings on Thursday to refine backlog
- Refine CI/CD
    - Write a README about Travis and its imposed rules
    - Make docker cross platform, to prevent odd Docker bugs
    - Docker build should be tested in Travis
- Refine Agile Process
    - Merge to master often (Once a day or every 2 days) with complete modules
    - Smaller pull requests, to prevent last minute midnight code review

    - Scope our work to solving current sprint's problems
    - Add User Interface Mockups and Designs as user story comments
    - Comment on GitHub issues, not in Slack
- Prepare for Sprint Review
    - Meet at 3 PM. Have product owner come in for the Sprint review at 5 PM
    - Code Freeze Thursday at 6PM. Prevents last minute failures