# **Contact information for Official Representative:**

**Name:** Abdullah Yusuf

**Email:** Abdullah.yusuf@precise-soft.com

**Team Name:** Precise Soft

# **Names of additional team members:**

**Name:** Hang Wang

**Name:** Bohan Niu

**Name:** Ruoyan Chen

**Name:** Revati Rajane

**Name:** Charlie Shou

**Name:** Yusheng Hu

**Name:** Miranda Liu

**Name:** Balasundaram Avudai Nayagam

# **Introduction to Team:**

Our team consists of innovative thought leaders and engineers from Precise Software Solutions based in Rockville, Maryland. We are excited for the opportunity to compete in this competition and are confident in our solution. If allowed to further build on this project we will develop a complete end to end solution that can be deployed in any environment.

A close up of a logo

Description automatically generated

SBA 8(a) | GSA IT 70 | CIO-SP3SB | CMMI Level 3

# **Executive Summary of Solution:**

Our solution utilizes a modern state-of-the-art ML model and an intuitive UI to provide an easy to use but powerful EULA analysis system that is designed with the business users in mind.

A user simply selects the EULA they want to analyze, and the tool sends the pdf or word document provided to the backend which parses the clauses and then semantically analyses them individually within the context of the surrounding clauses before outputting the parsed clause, prediction, and confidence to the be displayed on the UI. Then the user may download this report at any given time.

The user may also decide to disagree with the analysis of any given clause and provide an alternate analysis. In doing so, not only can the user save their edits to the report and download the new version, but in saving their edits this information is automatically sent to the backend so that we can retrain our model and improve its accuracy in making predictions. This way, there are no manual steps involved and no human-in-the-middle is required. The process of analysis and retraining is completely automated and happens in real time.

This creates a continuous learning model that is automatically updated for future analysis, and results in a truly automated CL (Continuous Learning)/CI (Continuous Improvement)/CD (Continuous Deployment)/ system.

In designing our solution this way, we were able to make our development efficient while not sacrificing any power in our solution. By executing our processes in real time our solution does not store any data and does not contain a database eliminating many security concerns. In this way we were able to develop the entirety of our solution in 2 weeks.

Try it out yourself here:   
<http://precise-gsa-challenge.s3.us-east-2.amazonaws.com/index.html#/>

# **EULA Analysis Architecture:**

## **Technology Scope:**

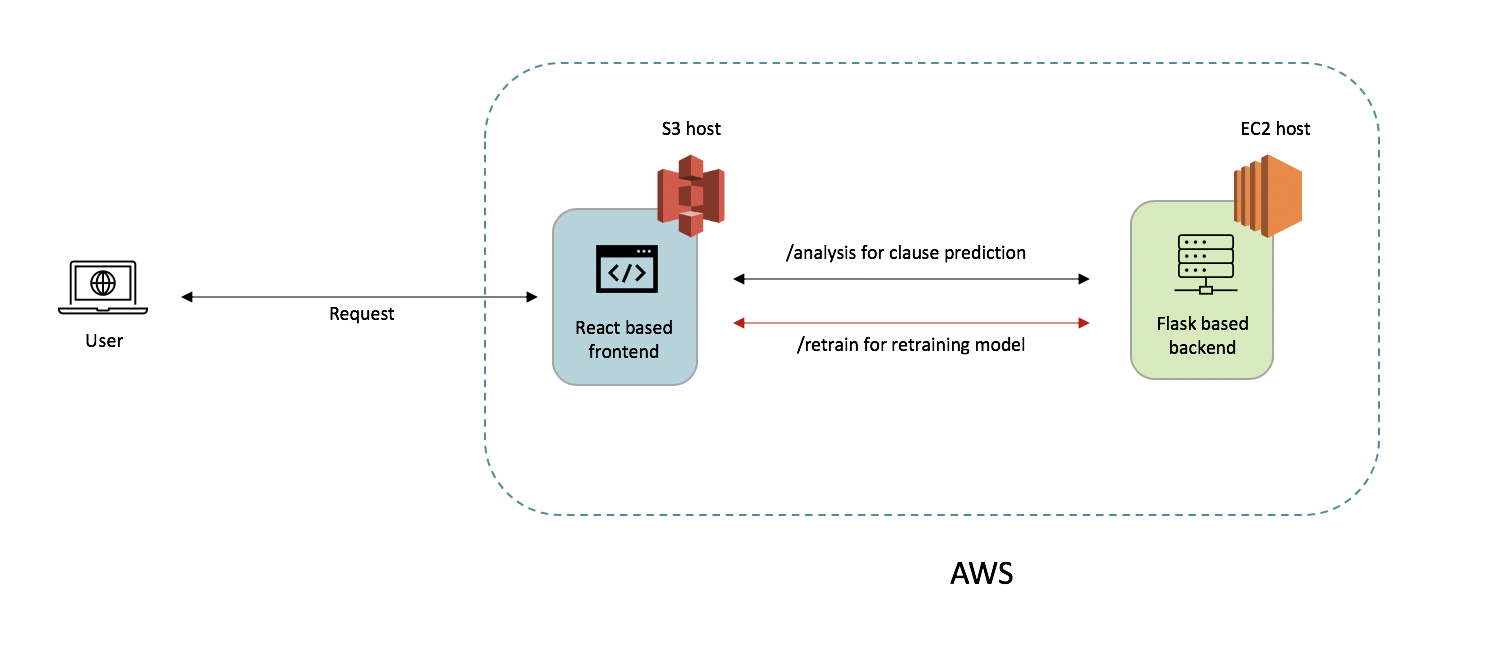


Fig 1: Application Architecture

Our application uses:

* Javascript Material UI React front end hosted on AWS S3
* Python Flask server deployed in AWS EC2.
* REGEX for text extraction
* XLNet AI/ML NLP Model for text analysis

## **The User Interface – Material UI and React**

Our application exposes a webpage for the user to interact with—all of the user’s interaction with the application happen here. We have designed the front-end part of the application using React.js library and have used Material UI for styling the components.

**React.js (React):**

* React is an open-source JavaScript library for building user interfaces. React can be used as a base in the development of single-page applications.
* Reason for using React.js is because of its component-based approach and single page rendering concept.
* Since we are dealing with displaying hundreds of components, to reduce the load in the rendering process, single page application works better and faster by rendering just the component that got changed.

**Material UI:**

* Material-UI is an open-source project that features styling React components with Google's Material Design.
* Material UI is very easy to use and has a good community support for React Component styling.

## **Text Extraction:**

In order to extract the terms out of files uploaded to the website, we’ve built two separate logic for word file (.docx format) and pdf file (.pdf format). When the user uploads the file to the website, the website will first check whether the uploaded file is a pdf file or a word file, and then it will feed the file into separate code.

**PDF file:**

If the selected file is in .pdf format, we use an open source python library called pdftotext (<https://github.com/jalan/pdftotext>). This library will extract all the text, line by line, from the pdf file and append them into a string. For extracting the terms clauses from this string. we process this list of string by regular expression. Regular expression is a pattern in which the rules for matching text are written in the form of metacharacters, quantifiers, or plain text. For this term extraction case, we know that all the terms will start with a bullet point. In sample\_eula\_1 and sample\_eula\_2, we have found four styles of bullet points.

* numeric bullet point: 1.
* character bullet point: a.
* roman letter bullet point: iv.
* parentheses style bullet point: (a). or (1).

The bullet point indicates the end of the previous term clause and the start of a new term clause. Our code will record the regular expression of these four types of bullet points. Whenever the code reads a regular expression of a bullet point, the code will end the previous term, start a new term, and put everything after the bullet point into the new term until it reads another bullet point.

**Word File:**

If the selected file is in .docx format, we use another open source python library called docx (<https://github.com/dolanmiu/docx>). This library is much more powerful than pdftotext. Apart from extracting the text from the word file, it can also keep the text file in the same format as the word file. For extracting the term clauses, we only need to extract the paragraphs from the docx file, since the terms are separated by paragraphs.

## **Application of Artificial Intelligence/Machine Learning (AI/ML)**

To train a language model from scratch and to have it achieve good performance on a certain task requires at least millions of text corpuses. Due to the limited amount of data available for this project, we decided to use transfer learning to tackle this problem: applying a pre-trained model to solve a new problem. We have selected transfer learning models that represent state-of-the-art in NLP filed, and further train and fine tune them with GSA’s EULA clauses classification dataset.

Traditionally, NLP tasks, such as text classification, utilizes pre-trained models such as word2vec, fasttext, GloVe, and Bi-LSTM, while LSTM based deep neural network models often outperform the others. Since 2018, the birth of BERT model led a wave of new transformers that far outperform any of the previously mentioned models and keep setting records for almost all the NLP benchmark tests. In this project, we chose to work with the transformers that were proven to be outstanding text classifiers including BERT, ALBERT, XLNet, and T5.

In the training process, we split the dataset into training/test sets with a ratio of 80:20. Models were created using HuggingFace transformers library. The models were fine-tuned by experimenting different learning rates and number of epochs for training, and select the best model primarily based on the F1 score for each training iteration. Due to the limitation of computing power, training batch size were limited to a range of 3 to 8. For BERT, ALBERT, and XLNet, models started overfitting when trained for more than 6 epochs, and learning rate at 3\*e-5 does yielded the best testing result.

Overall, XLNet outperformed BERT and ALBERT by a significant margin when it comes to F1 score. T5 model had the problem of producing invalid result such as letters or numbers other than 0 and 1 when using inference mode to predict the labels of new data. This is because T5 is a text to text model that encode the input sequence and decode it to an output sequence, which in our case were strings “0” and “1”, instead of treating the labels as actual classes. Although if given a much larger training dataset T5 model shouldn’t have this problem, as of right now it is not suitable for our use case. Based on above comparison, we went on to use XLNet as our classifier to identify EULA clauses.

|  |  |  |
| --- | --- | --- |
|  | 0(Predicted) | 1(Predicted) |
| 0(Actual) | 1183 | 102 |
| 1(Actual) | 84 | 207 |

Table 1. Confusion Matrix of XLNet’s prediction on test dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | TPR | FPR | TNR | FNR | F1 score |
| BERT | 57.4% | 42.6% | 93.3% | 6.7% | 61.40% |
| ALBERT | 57.0% | 43.0% | 93.5% | 6.5% | 61.34% |
| XLNet | 71.1% | 28.9% | 92.1% | 7.9% | 66.99% |

## Table 2. Accuracy comparison of BERT, ALBERT, and XLNet

This result was expected as Google’s XLNet has achieved SOTA in many NLP tasks including text classification. In our application, XLNet trained on GSA’s training data serves as the backend algorithm to classify input EULAs as either acceptable or problematic and then return the result to the user. The user has the choice to review the returned result and change the label of the clauses if they found some predictions not accurate. The edited clauses will be sent back to the XLNet model for re-training, so the algorithm’s performance can improve over time.

## **Functionality**

There are two primary types functions the user may execute using our application:

### Upload EULA to receive analysis report

As a User you are able to access the home page of the application and upload a EULA document in either PDF or DOCX formats.

A screenshot of a social media post

Description automatically generated

Fig 2a: Upload EULA Screenshot

This document is then sent to the server where the text is parsed into individual clauses using REGEX expressions.  
  
These clauses are then passed into the XLNet model where they are assigned a prediction on their accessibility and a score representing the probability for a successful prediction.  
  
All of this information is then returned to the Ui and displayed to the User. Here the User may use the search bar to find the clause they are curious about by keyword for a smooth and efficient experience.

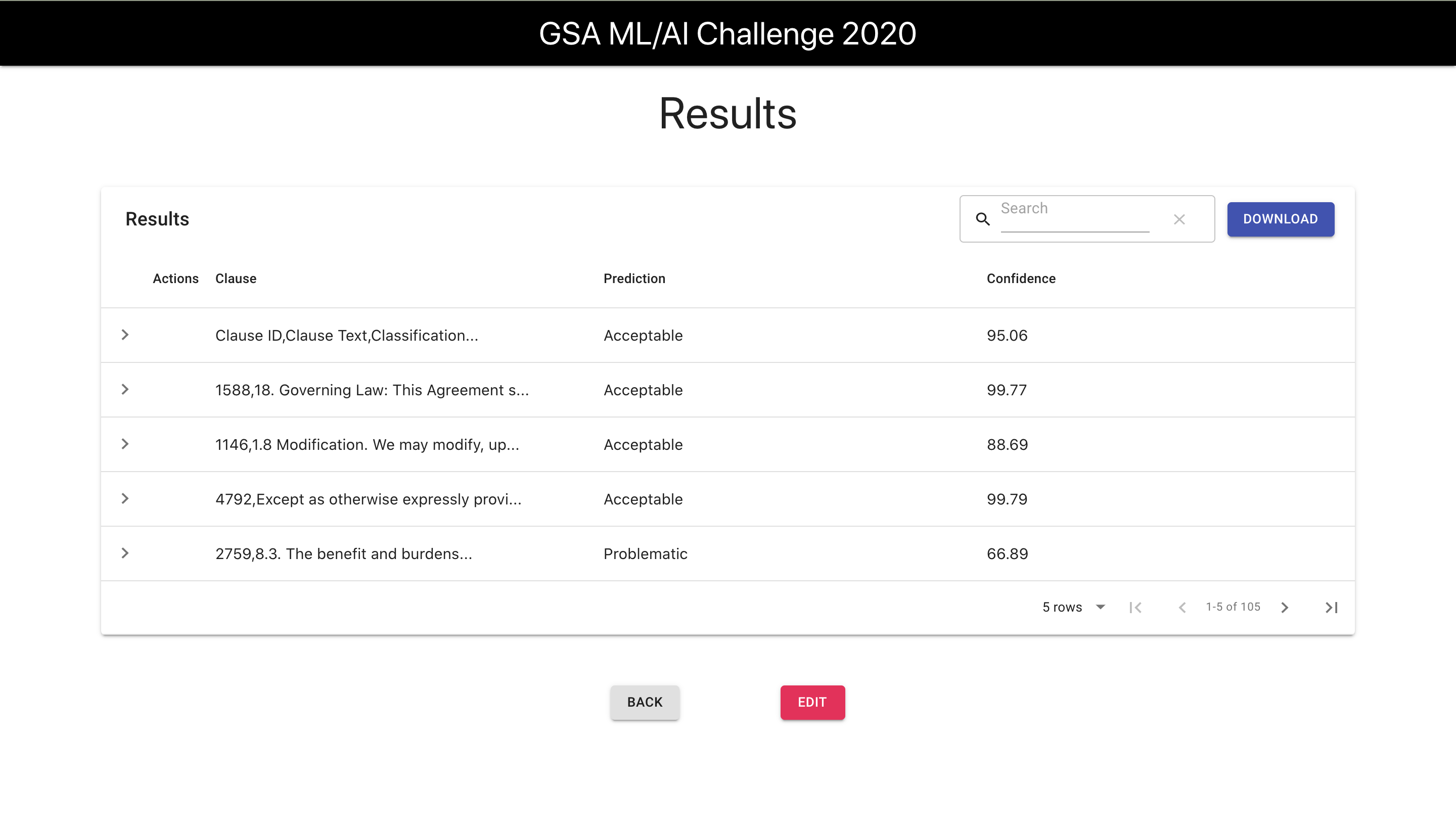


Fig 2b: EULA Report Screenshot  
  
The User may also download the report as a CSV or a PDF. So that it may be shared easily.

### Editing the analysis report

Once the User has the EULA analysis report they may choose to disagree with some of the predictions of the models and may edit the report by changing the prediction and saving their changes.

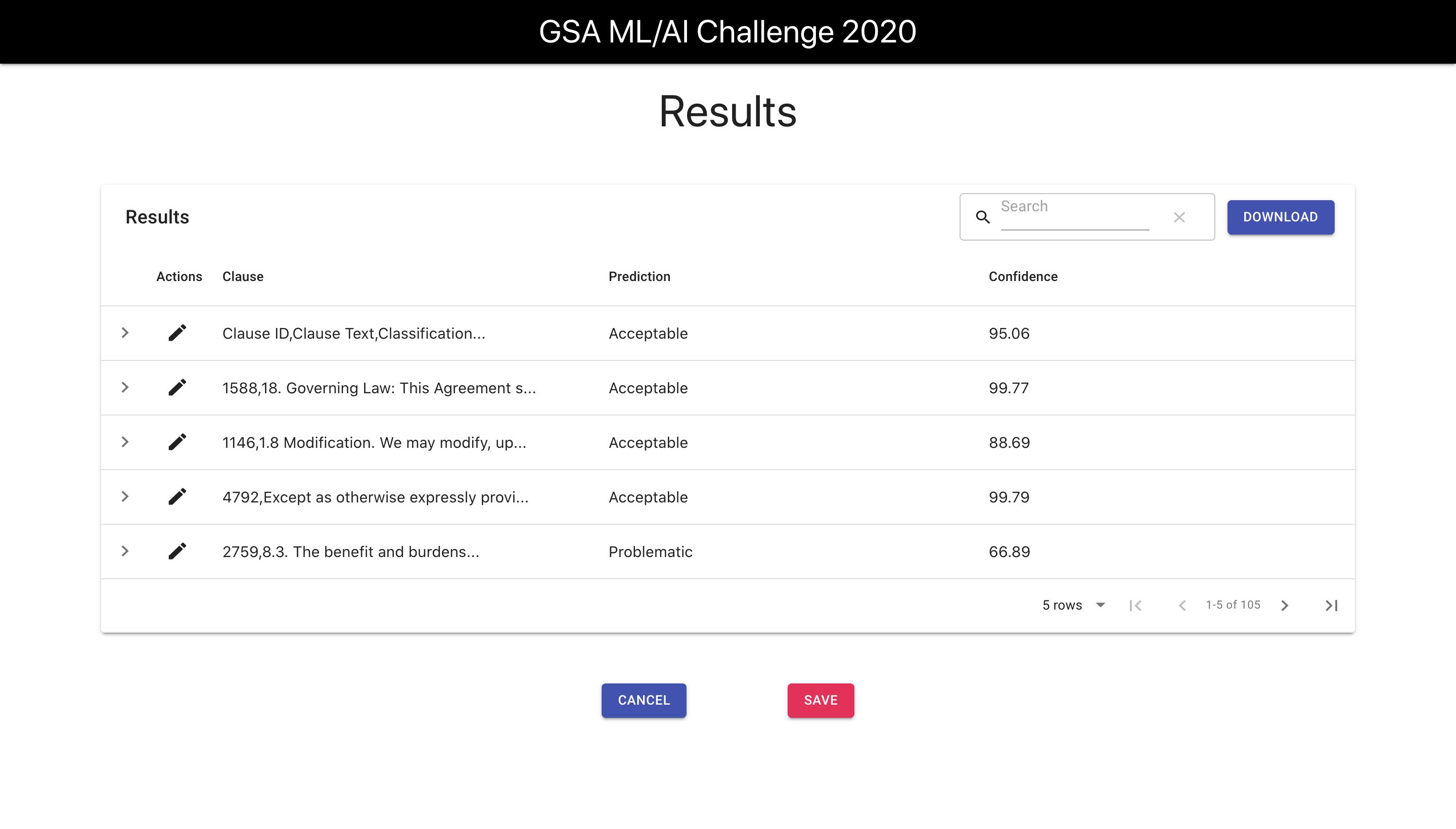


Fig 3a: Edit Mode Report Screenshot

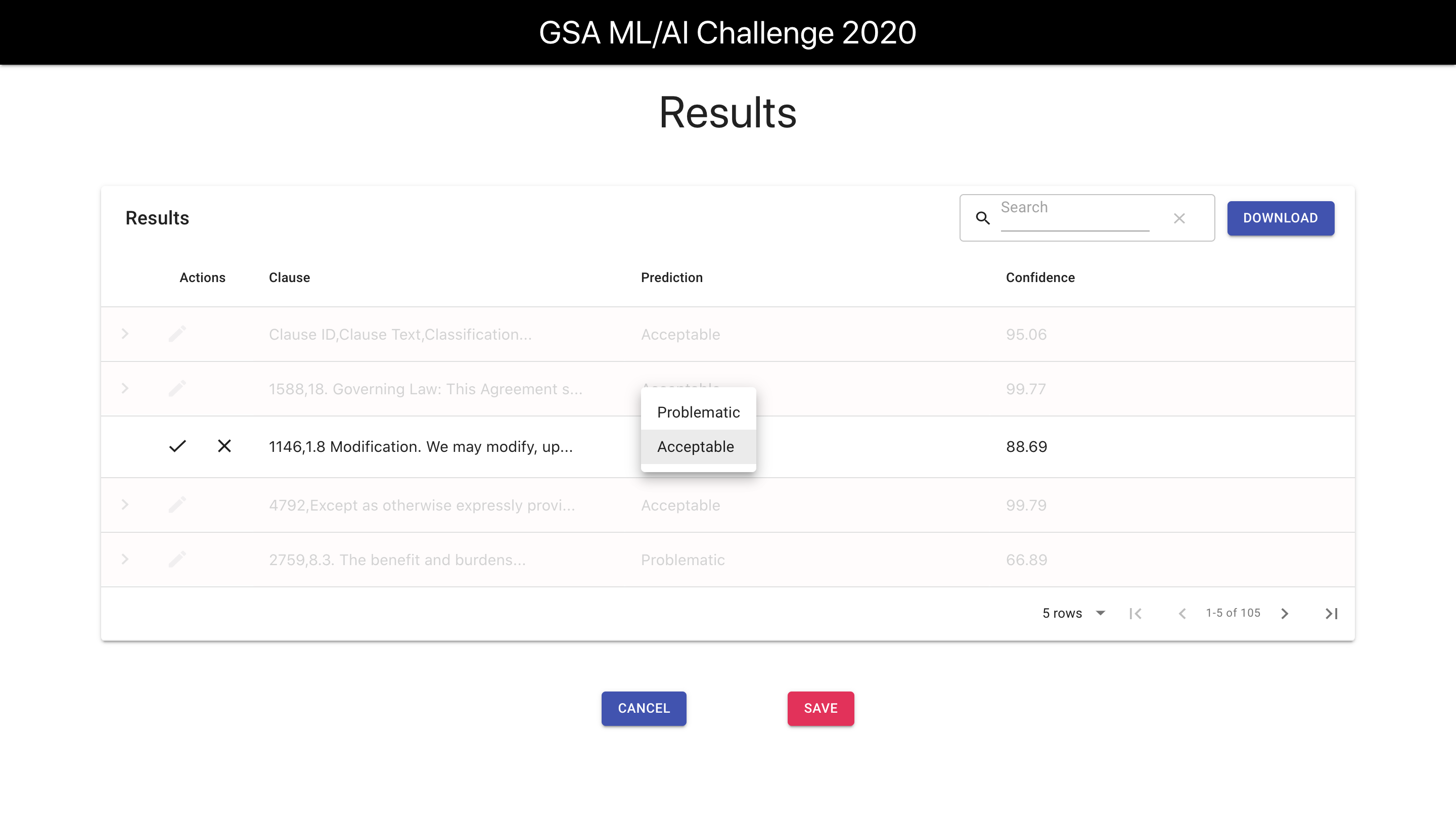


Fig 3b: User editing a clauses prediction screenshot  
  
When a user saves their input these changes are passed back to the server so that it may retrain the XLNet model immediately. The new retrained model then replaces the old one. This sets up the application for improved results going forward by making it more familiar with the data.  
  
The User may then download the edited report as a CSV or PDF.