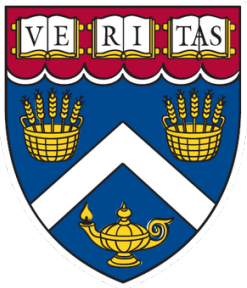
Harvard Extension School



Product Requirements and Specifications

PharmaDB

A Drug Label and Patent Explorer

May 13, 2021

CSCI-599 Section 2 Software Engineering Capstone

Teaching staff: Dr. Peter Vaughan Henstock, Roman Burdakov

Customer: Dr. Tom Marman

Anthony Mancini, Brett Bloethner, Krithika Sundararajan, Michael Thornton, Terry Chau

Table of Contents

1. Abstract

New prescription drugs frequently deliver a significant and beneficial impact on society. They lead to longer life, shorter hospital stays, fewer days of lost work, and lower overall medical prices (Lichtenberg). However, creating new drugs can cost billions of dollars (DiMasi et al.). These costs are recouped through sales once a drug is released to the market. Drug companies rely on patents to protect the Intellectual Property (IP) related to the new drug. Patents give the patent holder exclusive control of the IP preventing a competitor from making a copy of the drug. This gives the patent holder an opportunity to sell their new drug without the competition of a copy for a period of several years. Without this protection, companies would be discouraged from investing in novel drugs.

In order to understand patent strategies being used and maximize future strategies, companies study what patents are associated with drugs and how those patents affected the drug labels. Researching the relationship between drugs, associated patents, and their drug labels is difficult. The information is spread across disparate systems with inconsistent formats with limited availability of historical data. Analyzing this information is time consuming and labor-intensive.

Our system finds changes in drug labels then identifies patents that appear to be associated with that change in text. If a patent is found to be related to a text change, the patent is also available for reading. To accomplish this, we pull data from the Orange Book, drug labels, and US patents on a scheduled cadence. The data is then processed to identify links between patents and drug labels. The information is presented in a web browser where the full label text can be viewed with changes over versions of the label notated. The data from our system is available for export to allow its use in other systems where additional analysis not anticipated by this system can be performed.

We believe this will save the analysts research time and allow them to focus more on the analysis of the data. In addition, the system's ability to identify links between patents and drug labels will point the analysts to areas of the data that might have been overlooked.

1.1 Introduction

Researching, developing, and gaining approval for a novel prescription drug is expensive. It costs, on average, over $2.8 billion per new drug and the costs have been going up at 8.5% per annum over general inflation (DiMasi et al.). The use of patents to defend the intellectual property for a new drug is considered to be essential in that it gives a company a period of time to offer the drug exclusively thus recouping the upfront costs (Congressional Research Service). Patent holders may extend the normal exclusivity period by filing auxiliary patents on a drug as additional uses and applications are discovered.

Drug companies are always trying to optimize patent strategy (Marman). To fully understand strategies used in the past and craft the best strategy for the future, companies need to look at more than just the patent. Information from three sources is needed: the Orange Book, FDA Drug Labels, and Patents.

- The Orange Book is published by the US Food and Drug Administration (FDA). It contains all the US approved drugs, their related patents, and exclusivity (U.S. Food and Drug Administration). The Orange Book is updated with monthly cumulative supplements.

- Approved drugs are included in the FDA's Drug Label Database. A drug label has the information needed to instruct healthcare professionals in prescribing the drug appropriately (Kremzner and Osborne).
- Patents are registered with the US Patent Office.

The information from these three sources can be used to see when patents were associated with a drug and how those patents affected the drug label. By analyzing changes over time to the patents linked to a drug along with modifications in its corresponding Drug Label, it is possible to detect the past strategies that were employed and deduce potential future strategies.

1.2 Literature Review

We built a novel, open-source tool for mapping changes in drug labels approved by the FDA, to claims in patents registered with the USPTO. The primary source of this information is the Orange Book, which is a well-known publication that identifies drugs approved by the FDA on the basis of the Federal Food, Drug, and Cosmetic Act, and patents related to the drugs. However, this does not provide any further details on the individual claims in the patents and what portions of drug labels they cover. Thus, much of our work revolved around understanding the structure of the patents as well as the existing tools and techniques used to mine data from them, which could be extended to our use case.

Lens, Questel-Orbit and PatSeer are some of the industry standard tools used to search through various attributes of patents. Google Patent Search is another popular tool that is free to use. Orbit and PatSeer are backed by semantic search capabilities. Aside from that, PatSeer and Google Patent Search also provide a 'Chemical Lookup' feature to search patent claims for variants of drug compounds. None of these tools were created to link patents to drug labels and are unable, as they stand, to do so.

Similarly, tools such as The DailyMed, Drug Central and PharmGKB have been used to search through and study the drug labels. The sites provide access to the data on drug labels. However, they do not offer any analytics or insights into the drug labels. The only patent information made available by these resources is just the same patent list already found in the Orange Book.

Our research did not find any specialised tools to explore the evolution of drug labels and patents together, beyond some basic keyword based searches. The problem of analyzing the label information covered by patents, however, can be generalized into a class of Natural Language Processing (NLP), and text mining approaches used in either domains and elsewhere can inform us of the existing advancements and pitfalls.

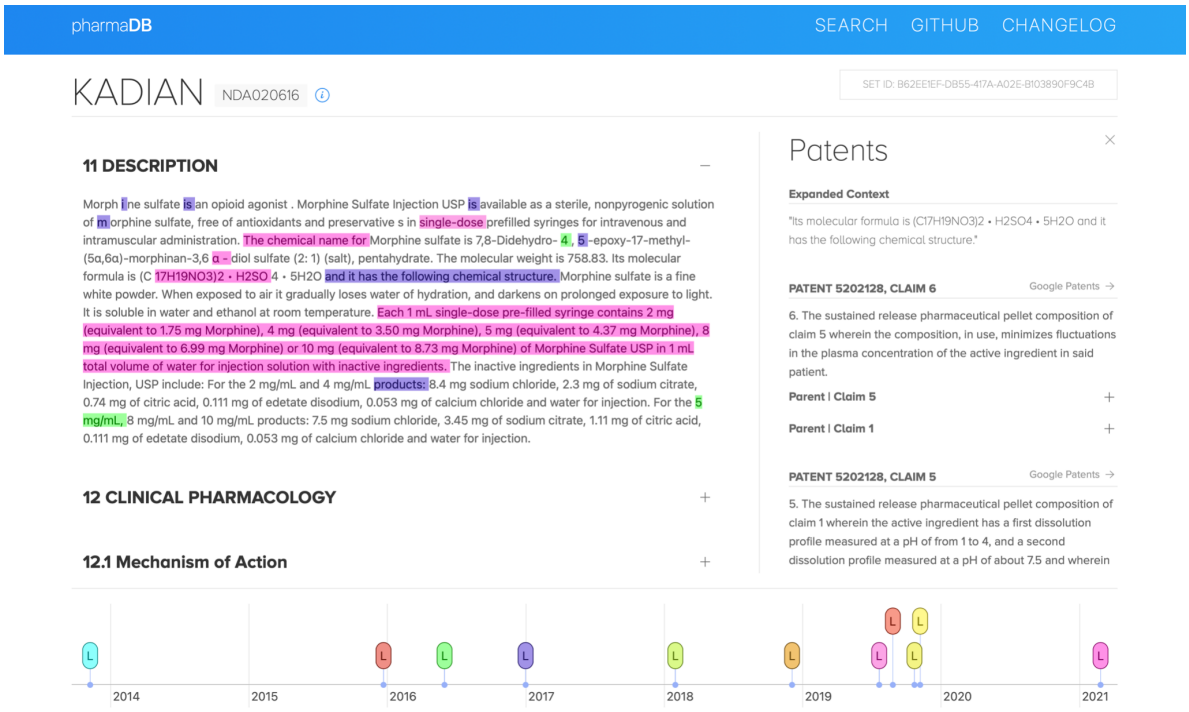
1.3 Data and Method

In light of the drawbacks of the tools discussed above, we proposed PharmaDB, a new pharmaceutical information resource to understand changes in drug labels over time, and to map those changes to claims across multiple patents. Chief among our goals is to provide an easy to consume web interface for identifying drug label history, and for studying how changes to drug labels over time are related to the patent filings for the drug.

We are able to correlate patents with approved drugs by way of the Patent and Exclusivity Tables from the *Approved Drug Products with Therapeutic Equivalence Evaluations* publication. This publication is commonly referred to as the Orange Book. The Patent and Exclusivity Table provides a means for mapping drugs, as defined by a National Drug Application (NDA) number, to corresponding United States patents. By combining past Orange Book datasets, such as the dataset digitized by Professor Heidi Williams of the National Bureau of Economic Research[1], with more modern Orange Book datasets, which can be retrieved directly from the FDA.gov, or by means of the Wayback Machine to the FDA.gov site, PharmaDB is able to relate any drug label associated with an NDA number to a patent, from the inception of the Patent and Exclusivity Tables in 1985 to the present day.

The patent files, used by PharmaDB, are retrieved from the bulk data archives of the United States Patent and Trademark Office. The drug labels are retrieved using a combination of application programming interface calls to the openFDA API[2], and the DailyMed API[3]. In total, PharmaDB archives about 21,000 drug labels and about 8,700 corresponding patents.

To identify changes in drug labeling, PharmaDB provides a time-based visualization of FDA approval dates, drug label publication dates, and corresponding patent issuance dates for a queried drug. The time-based visualization includes a built-in diffing tool that colors additions or subtractions between two drug labels at consecutive publication dates.



[1] https://www.nber.org/research/data/orange-book-patent-and-exclusivity-data-1985-2016

[2] https://open.fda.gov/apis/

[3] https://dailymed.nlm.nih.gov/dailymed/app-support-web-services.cfm

Figure 1 PharmaDB web application screenshot

Arguably, the collation of dates, and the diffing of drug label texts provide a minor convenience for a drug researcher. PharmaDB's true *tour de force,* however, lies in the way in which it meaningfully maps changes between drug labels with the patent claims that recite those changes. We achieve our mapping using Natural Language Processing approaches.

In this approach, we first take the delta between two drug labels at consecutive publication dates to find sections of the labels that have changed. Drug labels are composed of many standardized sections. After consultation with a patent search specialist, we identified three sections of interest, which are 'Indications and Usage', 'Dosage Form and Strengths', and 'Description'. These sections are selected because they likely contain patent-eligible subject matter. Patent-eligible subject matter includes (i) an addition of a new indication or other condition of use; (ii) an addition of a new strength; (iii) a minor change to the product formulation; or a change to the crystalline structure of the active ingredient.

From these deltas, we identify new additions to the labels over time. Additions in the diff, as opposed to subtractions, are of particular interest, since new patents are issued for novel and non-obvious subject matter that has not previously been disclosed. Additions therefore should have a higher likelihood of correlating with new patent claims, since they both introduce new drug features.

To perform the correlation, the performance of semantic similarity comparison engines were compared, and the best option selected. NLP engines considered for the task include: scikit-learn's TfidfVectorizer[4], the ScispaCy[5] model with the spaCy library, and various models based on transformer networks using the sentence_transformer library. As the sentence_transformer package is especially modular, and can utilize most models uploaded to the HuggingFace model repository, many tests with the sentence_transfomer package focused on high performing models as identified by a spreadsheet[6] maintained by the sentence_transformer team. Transformer models considered include stsb-distilroberta-base-v2, stsb-roberta-base-v2, stsb-mpnet-base-v2, and Longformer (Beltagy et al, 2020). The performance of transformer models particularly adapted to the biomedical domain, such as SciBERT (Beltagy et al, 2019), and BioBERT (Lee et al. 2019), were also observed.

These modules were all considered using a cosine similarity approach. It is understood that other measures of similarity, like Jaccard or Manhattan distances, can be used for semantic similarity comparisons. However, our focus was narrowed due to time constraints, and processing power. Cosine similarity is a common approach, and is conveniently packaged with most NLP libraries.

To study the efficacy of the various NLP engines, we sampled two distributions using random sampling for each model. One of these distributions measured similarity scores when the label additions are matched to claims from related patents. The other distribution measured similarity scores when label

[4] scikit-learn TfidfVectorizer

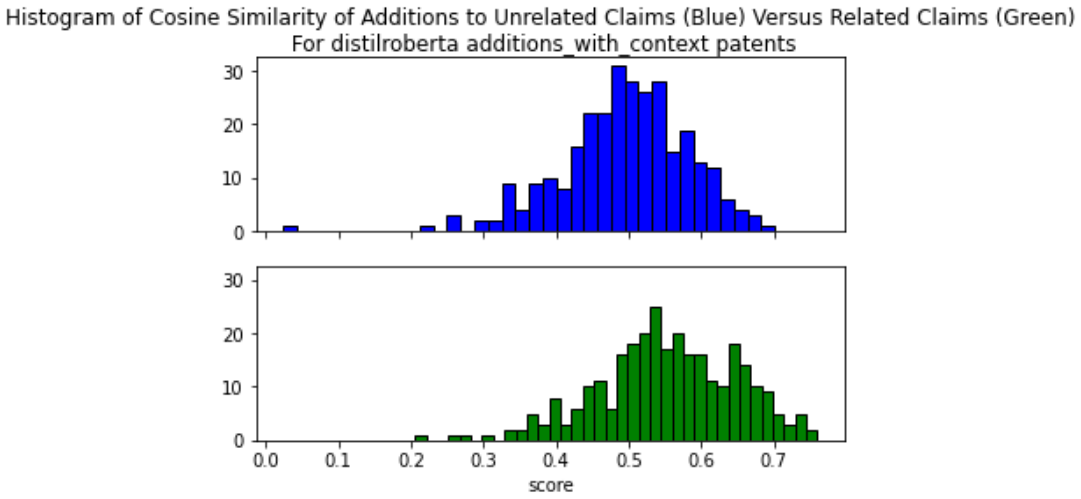
[5] ScispaCy

[6] SentenceTransformer Pretrained Models

additions are matched to claims from unrelated patents (hereafter referred to as the 'noise' distribution). Again, we know which label additions are related to which patents by way of the Orange Books Patent and Exclusivity Tables. The patent claim with a similarity score having the highest value is selected as the patent claim which best matches with the label addition.

From our collection of drug labels, one-third of those labels were chosen at random. We then dropped all labels with missing related patent claims, and all labels without any additions. The dropped labels do not add to our understanding of the similarity scores, since they cannot be compared. In the end, we had 295 drug labels that we used as part of our study.

As part of the study, our goal was to determine the model that has the distribution with the greatest standard deviation on the right-hand side from that model's noise distribution. We reasoned that the model with the greatest standard deviation would likely be a good candidate for the similarity comparison module in our software, since that model is furthest away from the 'noise' model, which was designed to perform particularly poor.




```
mean noise: 0.49429224932004184
std_dev noise: 0.08914535609158014
mean related: 0.5527198500920639
std_dev related: 0.09542776364072823
z score: 0.6554194557481904
```

Figure 2. Histograms for two distributions, one in which the additions are matched with unrelated claims, and another in which additions are matched with related claims, for a single model

Figure 2. Illustrates one of the set of distributions generated as part of the study to select an optimal model. The top histogram in blue represents the *noise* distribution and the lower histogram in green represents the distribution when the additions are matched to related patent claims. In this particular example, we see that the mean of the distribution in green is .655 standard deviations away from the

mean of the *noise* distribution in blue. We want to select a model that maximizes the standard deviation between the means of the noise distribution and of the related distribution.

Furthermore, it is noted that the distribution in figure 2 is for *'addition_with_context'* instead *'just_additions'*. A diff between two documents is constituted by additions, subtractions or no-changes. For this natural language processing step, we are only concerned with matching additions to the patent claims. However, since additions may be one or two words, we decided to optionally take the context around the addition to test if it improves results. Context, in this case, refers to the entire sentence in which the addition lies. From the table below, it can be seen that *'addition_with_context'* generally performs better than just additions in separating the related distribution from the noise distribution.

`Patents_longhand` is similar to `additions_with_context` in that dependent claims for patents can be written in long-hand form (without dependencies) or short-hand form (with dependencies). For example, the preamble of a dependent claim may begin with: "The widget of claim 1, …". This is equivalent to a claim which includes all the subject matter of claim 1 with the subject matter of the dependent claims. Since additional context may improve NLP results, we also tested whether patents in longhand form outperform patents in shorthand form. From the table below however, it is shown that the longhand form of the patent does not appear to improve the results of stsb-distilroberta-base-v2 or stsb-mpnet-base-v2 with patent claims written in shorthand form.

Summary statistics of distributions tested for all models are shown below. Full results of the analysis is available on github.[7]

							Hypothesis Test for Difference of Sample Means		
Model Name	**Size (n)**	**Mean Noise**	**Std Dev Noise**	**Mean Related**	**Std Dev Related**	**Z-Score for `Mean Related` From Noise Dist.**	**Difference in Mean Related To Mean Noise**	**Std Dev of Difference**	**1.65*Std Dev of Difference [1.65 is the critical z-value for 95%]**
stsb-distilroberta-base-v2 additions_with_context patents_shorthand	295	0.4943	0.0891	0.5527	0.0954	0.6554	0.0584	0.0076	0.0125

[7] Jupyter/Colab notebook with test results

stsb-distilroberta-base-v2 just_additions patents_shorthand	295	0.3653	0.1016	0.3910	0.1251	0.3117	0.0257	0.0094	0.0155
stsb-distilroberta-base-v2 additions_with_context patents_longhand	295	0.5012	0.0938	0.5518	0.1033	0.5332	0.0506	0.0081	0.0134
stsb-roberta-base-v2 additions_with_context patents_shorthand	295	0.4923	0.0897	0.5540	0.0976	0.6888	0.0617	0.0077	0.0127
scibert_scivocab_uncased additions_with_context patents_shorthand	295	0.8200	0.0425	0.8365	0.0531	0.3878	0.0165	0.0040	0.0065
biobert-v1.1 additions_with_context patents_shorthand	295	0.8938	0.0249	0.9041	0.0306	0.4158	0.0103	0.0023	0.0038
longformer-base-4096 additions_with_context patents_shorthand	295	0.9742	0.0128	0.9749	0.0151	0.0604	0.0008	0.0012	0.0019
spacy_en_core_sci_lg additions_with_context patents_shorthand lemmatize stopword removal punctuation removal	295	0.5411	0.1218	0.6340	0.1399	**0.7626**	0.0929	0.0108	0.0178
tf-idf additions_with_context patents_shorthand lemmatize stopword removal punctuation removal	295	0.0718	0.0606	0.1444	0.1106	**1.1978**	0.0726	0.0073	0.0121
stsb-mpnet-base-v2 additions_with_context patents_shorthand	295	0.4317	0.0895	0.5029	0.1089	**0.7950**	0.0712	0.0082	0.0135

stsb-mpnet-base-v2 additions_with_context patents_shorthand lemmatize stopword removal punctuation removal	295	0.4504	0.0891	0.5165	0.1073	0.7408	0.0660	0.0081	0.0134
stsb-mpnet-base-v2 additions_with_context patents_shorthand	295	0.4432	0.0855	0.5134	0.1001	0.8203	0.0702	0.0077	0.0126
stsb-mpnet-base-v2 additions_with_context patents_shorthand	295	0.4355	0.0907	0.5019	0.1073	0.7330	0.0665	0.0082	0.0135
stsb-mpnet-base-v2 additions_with_context patents_shorthand	295	0.4361	0.0889	0.5140	0.1041	0.8757	0.0778	0.0080	0.0132
stsb-mpnet-base-v2 additions_with_context patents_shorthand	295	0.4316	0.0933	0.5036	0.1050	0.7712	0.0720	0.0082	0.0135
stsb-mpnet-base-v2 additions_with_context patents_longhand	295	0.4457	0.1064	0.5101	0.1048	0.6050	0.0644	0.0087	0.0143
stsb-mpnet-base-v2 additions_with_context patents_longhand_trunc_start	295	0.4465	0.1049	0.5104	0.1044	0.7287	0.0639	0.0086	0.0142

The results in gray from the table above ran with different random samples from the results in white, which all shared the same random sample, but different models.

From the table, we see that hypothesis tests are performed on each model to see if the differences in the mean of the noise distribution and the means of the related distributions are significant. That is, the null hypothesis is that the difference between mean of the related distribution and mean noise is zero, and the alternative hypothesis is that the difference between mean of the related distribution and mean of the noise distribution noise is greater than zero. For each model, we calculated the difference between the mean of the related distribution and the mean of the noise distribution. We then calculated the standard deviation of the difference of the sample means between the related and noise distributions. Finally, we multiplied the standard deviation of the difference of the sample means between the related and noise distributions by 1.65 to get the deviation at the 5% significance level, where we can reject the null hypothesis. Since the difference between mean related to mean noise is greater than 1.65 * standard deviation for all models, with the exception of Longformer, we can reject the null hypothesis for all models except for Longformer, and conclude that there is a significant

difference between the mean of the related distribution and the mean of the noise distribution for all models.

GIven that there is such a difference, the next consideration is to pick the model with the highest z-score between the mean of the related distribution and the mean of the noise distribution. Again, the higher the z-score, the less likely the model's results resemble noise. The three highest z-scores for models tested on the same sample of additions is in bold in the table above

TF-IDF which has the highest z-score is disregarded, since its distribution is not normal, and its means for both the noise and the related distribution are near 0. The embeddings for TF-IDF are determined based on word frequency counts, so it is understandable that when the set of terms used in the addition do not match the set of terms used in the claims, TF-IDF would naturally produce a low similarity score. The results indicate that TF-IDF generally produces poor matches for our sample dataset, since it's low means indicates that it does not find many patent claims with word sets similar to word sets for the label additions.

Thus given these results, we are left with the choice of the stsb-mpnet-base-v2 model or the ScispaCy model as our preferred NLP semantic similarity model. These two results have the second and third highest remaining z-scores between the mean of the noise and the mean of the related distribution for the same random sample of drug labels, respectively. Stsb-mpnet-base-v2 is chosen as the NLP engine for this project, since it is reasonably fast and performs equally well with or without preprocessing. See `lemmatization`, `stopword removal` and `punctuation removal` for stsb-mpnet-base-v2 in the table above. Only models with these terms were run with these preprocessing steps. From the table above, we see that having these preprocessing steps turned on or off does not significantly affect stsb-mpnet-base-v2.

Stsb-mpnet-base-v2 is the sentence_transformer build of MPNet, which is designed to leverage full positional information of a sentence during pretraining and fine-tuning phases of the model development (Song et al. 2020). Song is able to show that MPNet is able to outperform previous state of the art pre-trained methods under the same model settings.

Using the selected NLP engine to map addition to claims, PharmaDB then stores the results of the mapping in a document database, so that it can be retrieved and displayed by the front end. PharmaDB preprocesses the similarity measurements, and caches the results in the database to reduce loading times, and improve the user experience.

By way of an extract-transform-load (ETL) pipeline that is scheduled using cron jobs, datasets used by PharmaDB are updated with the issuance of new editions of the Orange Book, or their supplements. A new Orange Book edition comes out yearly and its supplements are released on a monthly basis. The ETL pipeline will fetch labels or patents referenced by the new edition of the Orange Book or its supplements, perform the corresponding machining learning to map sections of the drug label to patent claims, and store the resulting calculations along with the retrieved texts in the document database. This automated data capture ensures that our system is up-to-date.

1.4 Conclusion

In this project, we built a system that uses data pipelines to automate pulling data for the Orange Book, Drug Labels, and US Patents and then analyzes this data to find links between the addition of patents and changes to a drug label. Due to the automated nature of PharmaDB's text mining approach, an end user can quickly ascertain the changes to a drug label overtime, and the corresponding patent claims that recite those changes. This information gives an analyst insight into the patent strategy being used to protect a drug and helps to craft future strategies.

1.5 References

Beltagy, Iz, et al. "SciBERT: A Pretrained Language Model for Scientific Text." 2019. arvix, https://arxiv.org/abs/1903.10676. Accessed 6 May 2021.

Beltagy, Iz. "Longformer: The Long-Document Transformer" 2020. arxiv,https://arxiv.org/abs/2004.05150. Accessed 6 May 2021.

Congressional Research Service. "Drug Pricing and Pharmaceutical Patenting Practices." *fas.gov*, 11 February 2020, https://crsreports.congress.gov/product/pdf/R/R46221. Accessed 28 February 2021.

DiMasi, Joseph A., et al. "Innovation in the pharmaceutical industry: New estimates of R&D costs." *Journal of Health Economics*, 2016, pp. 20-33.

Gupta, Himanshu. "Patent protection strategies." *Journal of pharmacy & bioallied sciences*, vol. 2, no. 1, 2010, pp. 2-7.

Kremzner, Mary E., and Steven F. Osborne. "An Introduction to the Improved FDA Prescription Drug Labeling." https://www.fda.gov/files/about%20fda/published/Prescription-Drug-Labeling-Course-Slides.pdf. Accessed 28 February 2021.

Lee, Jinhyuk, and Wonjin Yoon. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining" 2019. arxiv, https://arxiv.org/abs/1901.08746. Accessed 6 May 2021.

Lichtenberg, F. R. (2001). Are the benefits of newer drugs worth their cost? evidence from the 1996 MEPS. Health Affairs, 20(5), 241-51. doi:http://dx.doi.org.ezp-prod1.hul.harvard.edu/10.1377/hlthaff.20.5.241

Song,Kaitao, et. al.. "MPNet: Masked and Permuted Pre-training for Language Understanding" 2020. arxiv,https://arxiv.org/abs/2004.09297. Accessed 6 May 2021.

Tostrup, Emil, and Sani Mesic. "Massive Patent Data Mining." *Master's Thesis at Lund University*, 2019.

U.S. Food and Drug Administration. "Approved Drug Products with Therapeutic Equivalence Evaluations." *fda.org*, 12 February 2021, https://www.fda.gov/drugs/drug-approvals-and-

databases/approved-drug-products-therapeutic-equivalence-evaluations-orange-book. Accessed 28 February 2021.

Marman, Thomas. "Re: Questions about Drug Labels & Patent Claims." REceived by Michael Thornton, 20 May, 2021. Email interview.

2. Journal Publication

The below journals are our top choices for publishing our research paper. We think all of them are a good fit for publishing our research since they have a related scope, contain high quality publications, and are readily accessible for publication.

- International Journal of Information Management Data Insights – This is a journal dedicated to applying data science techniques to the field of information management in order to draw actionable insights through the use of business data. The application of data science techniques that this journal focuses on is primarily applied towards data in corporate, government, educational, and social contexts. We think this journal would be a good place to publish our work as our research is focused on the patent space (which is both within a corporate and government scope), and applies data science techniques (in the form of Natural Language Processing techniques) to the patent space. Additionally, this journal has **waived it's Article Publishing Charge until September 31, 2021**[8], so this journal is our first choice for publication of our research.
- World Patent Information – This journal has a focus purely on the intellectual property space, including patents, trademarks, trade secrets, copyrighted material, and other forms of intellectual property, and is interested in articles that research meta data relating to IPRs such as legal status data for IPR, classifications and bibliographic data[9]. Since our research involves applying Natural Language Processing techniques to the drug patent and label space for improved classification and linking of patent information, we think that our article is very likely to get approved for this journal. However, this journal has an Article Publishing Charge of **USD 1900**[10].
- The Journal of High Technology Management Research – This journal's primary focus is on the management of emerging technologies. Given that our research uses technologies that are still emerging (notably Natural Language Processing techniques and libraries), and given the broad scope of this particular journal, we think that our research would be accepted into this journal. However, this journal is our third choice for publishing our research as the journal's scope doesn't focus on patents or Natural Language Processing. Additionally, this journal also has an Article Publishing Charge of **USD 2100**[11].

[8] https://www.journals.elsevier.com/international-journal-of-information-management-data-insights/

[9] https://www.journals.elsevier.com/world-patent-information/

[10] https://www.elsevier.com/about/policies/pricing

[11] https://www.elsevier.com/about/policies/pricing

3. Changes Since Milestone 1

Since Milestone 1, there have been a few additional requirements and architectural changes. Some of these changes resulted in simplification of the software development, while others add complications.

3.1 Requirement Changes

Per the discussion with the client on March 21, we came to the agreement that patent submission dates, which needs to be gathered from PDF scans of old Orange Books and its supplements, can be shifted to a "Could Have" requirement. See Requirement 13 of the appendix. While the display of the patent submission date is informative, it is not vital to the minimum viable product, since the user is able to view the patent claims directly from a drug label section, without necessarily going through a timeline.

In the review of milestone 1, it was suggested by the teaching staff that patent payment schedules and active state of the patent may be included in the software solution so that a user can determine these values without further lookups. This requirement was added under a "Could Have" category. See Requirement 18 in the appendix.

In the meetings dated March 21 and March 26, it was proposed that changes between multiple consecutively published labels should be presented in a single view. This requirement was added as Requirement 3.2 in the appendix. This requirement presents additional complications for diffs, since the team originally envisioned that diffs can only occur between two labels.

3.2 Architectural Changes

We had originally planned to use a graph database as our primary database, with an auxiliary document database to store additional information. The patent claims have an inherent tree-like structure that is easily represented in a graph DB. Such a DB would also permit us to gather insights about the data as well as assist in querying nested relationships (such as NDA numbers to drug label sections and patent claims) with ease. However, due to time constraints and the learning curve for team members that are new to graph database paradigm, we collectively decided to simplify the application architecture by removing the graph database. We will store corresponding data linking the labels to the patents within the document database.

4. Application Design

4.1 Design Process and Core Ideas

The user experience and user interface design was an iterative process that occurred over a series of meetings with Dr Tom Marman where potential designs and user flows were reviewed and either modified or rejected. These meetings gave the team valuable insight into the sort of design models that

could be incorporated into the application in order to most effectively address the project requirements. These insights and their impact on the applications design are listed below.

- **All of the data the user is looking for falls under one NDA number**
    - In order to determine which NDA number the user is looking for, we provide an initial search page similar to what the user might see in the favorite search engine. This should help make the user feel familiar with the application's entry point and provide the application with the correct NDA number for its subsequent API queries.
- **Most of the insights gained from the application come in the context of labels**
    - All of the information the user gains form the core requirements of the app come in the form of a drug labels relationship to either another label or a patent. As a result, the app has a label-centric view where a label dominates the UI and relevant design elements are overlaid to indicate where interesting relationships exist.
- **Labels change over time and these changes are valuable**
    - Labels are living documents that frequently receive updates and can be considered time series data. An intuitive reflection of this data over time would be valuable and could be provided in the form of a timeline or some other time related construct.

4.2 Design & Wireframes

The application design ultimately took shape as a two view based application where one view was a simple search engine and the second view was a label viewer. This label viewer contains multiple viewing modes which enable the user to view labels in various contexts including a label and its patent claims as well as a label diff and its patent claims. The decision to overlay data onto the in-view label allowed for displaying patent and label diff data simultaneously as opposed to providing a separate view for label information and a separate view for patent information like some of our earlier proposed designs had done.

Search view (App Entrypoint)

The search feature is the first view the user is presented with once they open the web app (Appendix Figure A1 and Figure A2). A single prominent search bar and placeholder text is meant to direct the user through the first step in the drug label searching process which begins with simply finding the drug that the sought after labels belong to. This view is meant to look and feel similar to today's most popular search engines. This way the user naturally knows the goal on this view is to find a correct result for their query.

Search view with results

Search results are shown instantly after the user types a fourth character in the search bar (Appendix Figure A3 and Figure A4). This saves from having to direct the user to a separate results page but requires a careful balance between autocomplete and the detail provided in the autocomplete search results. Oftentimes the user needs more than just matches to their exact query in order to determine which drug result is what they're looking for. The incorporation of detail-rich tiles helps accomplish this by showing drug specs such as brand names, generic names and active ingredients just to name a few.

Drug view entrypoint

The drug view is the second of the two views that make up the web app (Appendix Figure A5 and Figure A6). A header sits at the top of the page and provides standard navigation information such as which drug the user is currently viewing, the drugs NDA number and other convenient resources such as icons that open modals for viewing additional drug metadata as well as images of the drug if they're available. The timeline at the bottom of the page relays high level information to the user regarding at what point in time the most label related activity occurred. Markers on the timeline can be clicked on in order to view label content, diffs and details.

Drug view label diff view mode

The drug view contains multiple viewing modes. One of these modes lets the user view a single label's diff against the closest prior label on the timeline (Appendix Figure A7 and Figure A8). In this viewing mode the user can see all the label additions and deletions between the two labels as well as which additions are covered under what patent claims. The user also has the option of clicking on the additions in order to switch to the next viewing mode which includes both the label diff and the patent claims related to the selected addition. The timeline provides a visual and color coded indicator that shows when the particular label in view was published.

Drug view label diff mode with patent window

Viewing both a drug label's diff and a diff additions patent claims is done by clicking one of the green highlighted additions displayed on the label text. This opens a panel to the side of the label that shows the relevant patent and patent claims as well as related parent claims (Appendix Figure A9 and A10). Color coded indicators also appear on the timeline in order to give a high level idea of when the label was published.

Drug view historical addition mode

The drug view with historical label additions (Appendix Figure A11 and Figure A12) is nearly identical to the drug view with label diff with the exception of it including highlighted text to show which portions of text have been added at what time and from which label. In this view, the highlighting color of the text matches the color of the timeline indicator for the particular label, that way the user can determine when that particular addition was published.

Drug view historical additions mode with patent window

The drug view with historical label additions and patent window (Appendix Figure A13 and Figure A14) is nearly identical to the drug view with label diff and patent window with the exception of it including highlighted text to show which portions of text have been added at what time and from which label. In this view, the highlighting color of the text matches the color of the timeline indicator for the particular label. The user is able to select any addition that might have an associated patent claim in order to view those claims, regardless of which label the claim came from.

5. Application Architecture

The PharmaDB application uses a Service Oriented Architecture design[12] and consists of four components:

- The **ETL (Extract, Transform, Load)[13] Pipeline** component is responsible for ingesting, processing and storing information that's not readily available using free public facing APIs from pharmaceutical and patent industry vendors.
- The **Web Application API** (Figure 9) is responsible for bridging the gap between the stored data from the ETL pipeline and the client side web application.
- The **Web Application UI** component, together with the API, is responsible for creating a user experience that enables our customer to easily visualize drug data and insights gained through the ETL pipeline processing algorithms.

- A set of **RESTful[14] and Archival Web Resources** that are hosted by US government agencies and provided free of charge.

[12] https://www.ibm.com/cloud/learn/soa

[13] https://www.ibm.com/cloud/learn/etl

[14] https://developer.ibm.com/technologies/web-development/articles/ws-restful

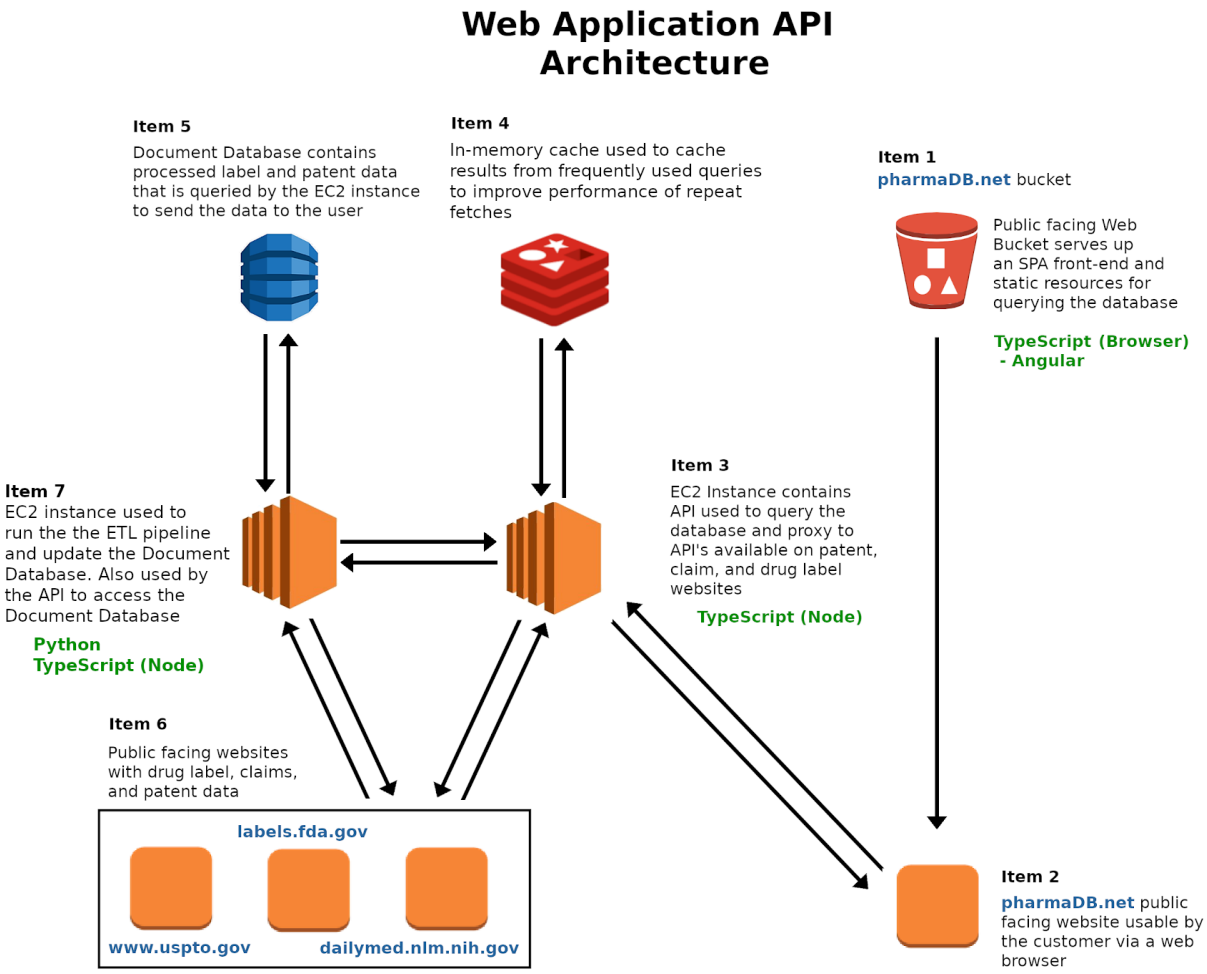


FIgure 3. AWS application architecture. AWS cloud resources used to run pharmaDB. Resources are indicated by logos, and connections between resources are indicated by arrows.

5.1 Web Application

5.1.1 UI

The client facing user interface is a single page application (SPA) built using Angular and includes two main views. The first view is responsible for the drug search feature (Figure 9) while the second view is responsible for viewing drug labels and patents (Figure 10). Visualization of differences between versions of drug labels is done using Google's diff-match-patch[15] and happens primarily as part of the ETL pipeline process. Diffs are returned from the Web Application API after they're fetched from MongoDB where the ETL pipeline persists its diffing, parsing and scoring results. Visualisation over time of label publication

[15] https://github.com/google/diff-match-patch

dates is done using the open source vis-timeline[16] library, a component of the community powered vis.js[17] browser based visualisation project. The timeline includes color coordinated icons by which the color associated with a label icon on the timeline is also the color used to highlight that label's particular additions in the historical diff viewing mode.

The component based architecture of Angular makes development simple by allowing a logical separation of the core pieces that make up each view. The drug view's timeline is a perfect example of this since it's structured as its own decoupled component and is drop-in replaceable by anything that shares a similar interface. Unfortunately, the design patterns used in Angular applications don't lend themselves well to being represented via UML. These components can be thought of as modular mini angular applications and are easily distinguishable in the below wireframe diagrams as opposed to UML.

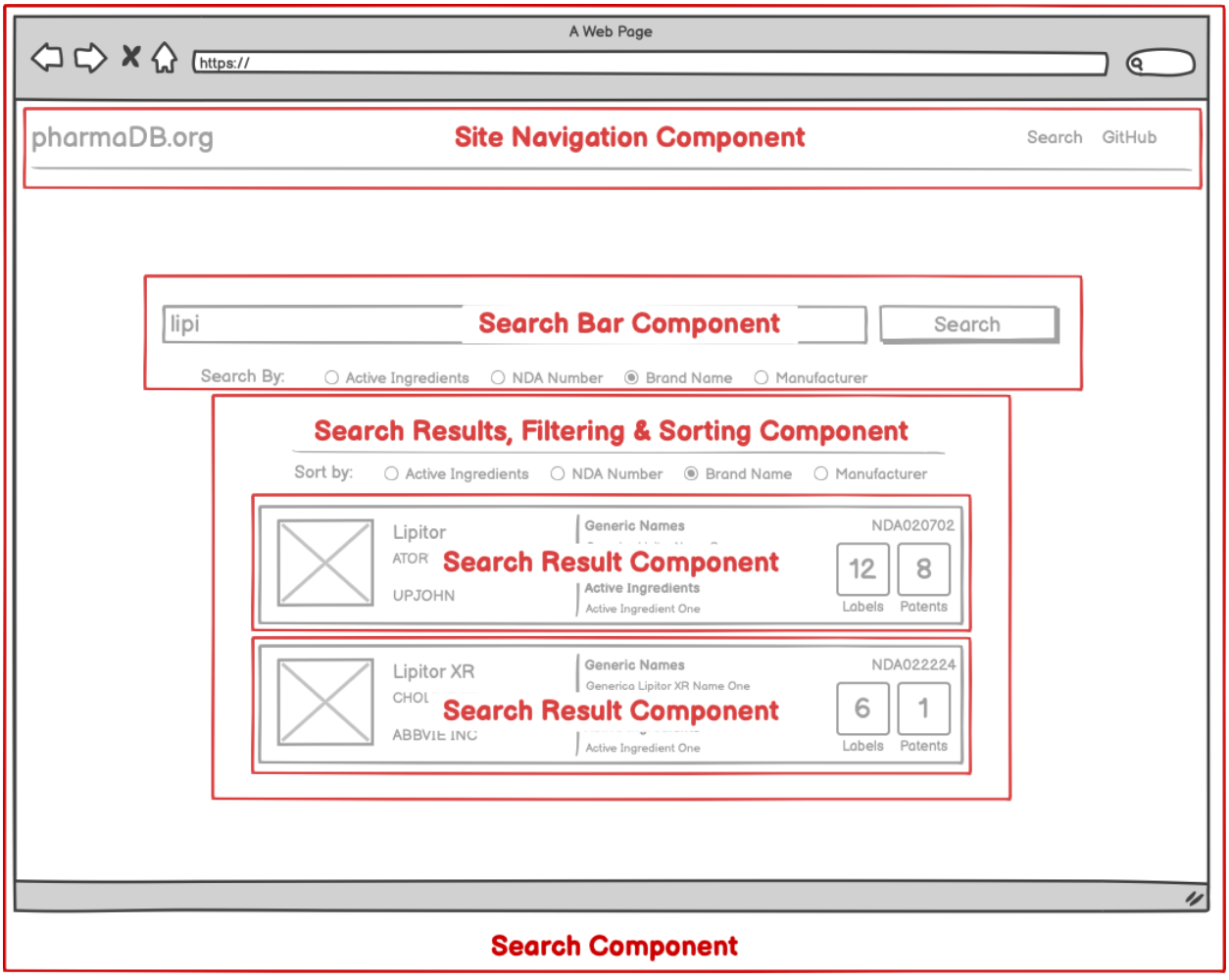


Figure 4. Angular component outline for the search view.

[16] https://github.com/visjs/vis-timeline

[17] https://visjs.org

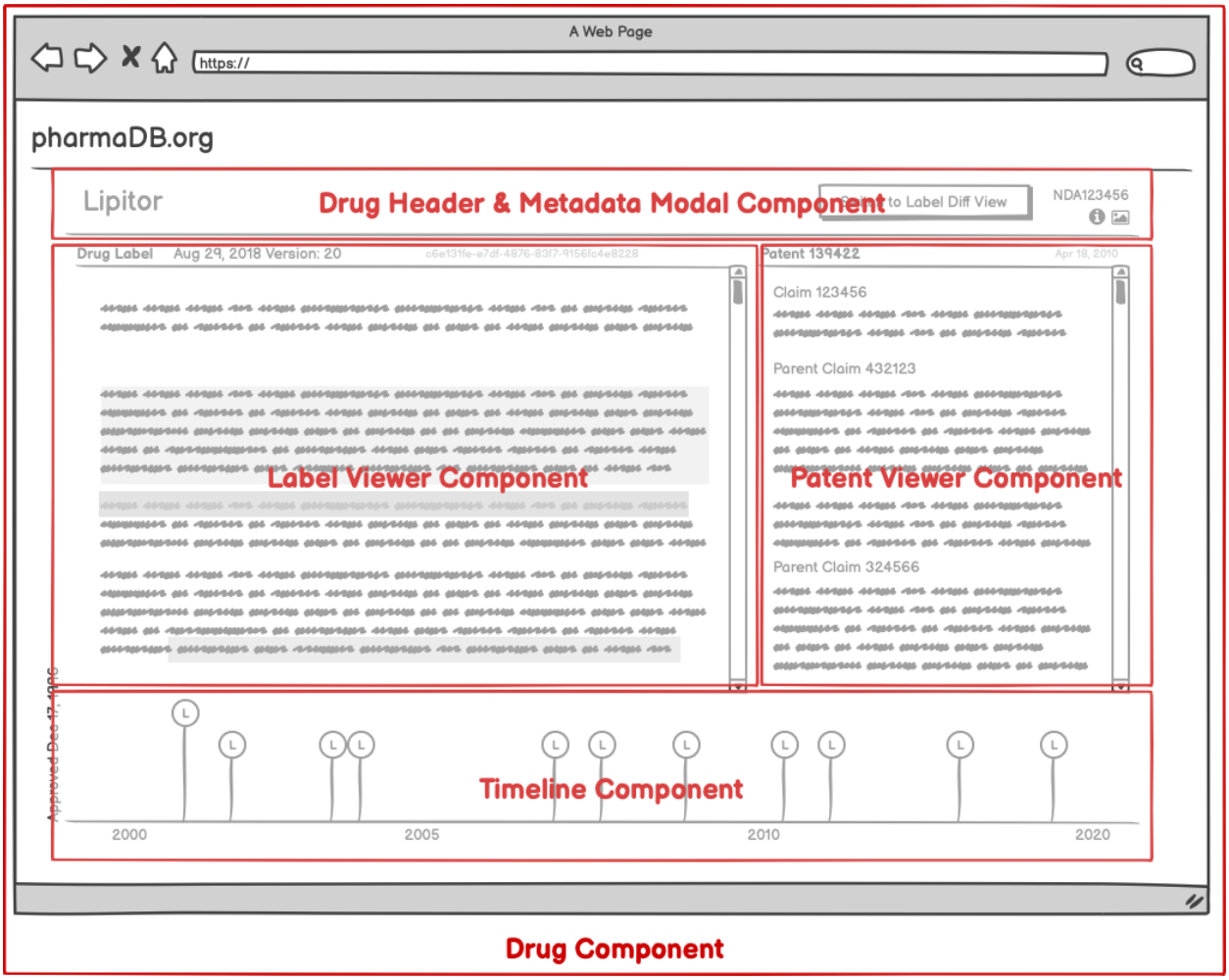


Figure 5. Angular component outline for the drug view.

Since the single page application does not require server rendering, the optimized and packed distributions of the web application can easily be hosted in an AWS S3 Bucket (Figure 8, Item 1) or any other similar web service[18]. Similar to other web applications built with the Angular framework, deployment of updates is a fairly easy process. Once a new version of the application is transpiled and bundled for release, the compiled application and assets can be pushed from a local repository into the project's S3 bucket and later on, through automated CI/CD.

5.2 API

The Web Application API for our single page application will be hosted on a standard AWS Elastic Cloud Compute (EC2) instance (Figure 8, Item 3). The SPA will be connected to our backend infrastructure through the EC2 instance's RESTful API. The API will be written using TypeScript and will run on the Node.JS runtime. Besides connecting to our backend infrastructure, the EC2 instance will leverage a local memory based cache to cache calls that require data from APIs available on the FDA and DailyMed websites (Figure 8, Item 4). This will allow for quicker searches in the future when the user searches for

[18] https://dzone.com/articles/deploying-an-angular-app-to-aws-s3

the same patent or label multiple times. The API's core functionality comes from the incorporation of the "builder" code pattern as well as a standard implementation of the ExpressJS API library which is used in the implementation of the only two endpoints the API provides. A UML class diagram of the APIs design can be found in the appendix (Appendix Figure A15).

5.3 Data

5.3.1 Data Sources

There are 3 main types of information that PharmaDB would require:

- Drug metadata and labels
- Patents linked to the drugs
- Detailed Patent information

These data are covered by 3 different sources (Figure 11) and the information on the data and endpoints relevant to the ETL pipeline are explained below.

The Orange Book

This identifies drug products approved by the FDA and the related patent and exclusivity information. The latest patent and drug info are updated once a month[19].

Historical Orange Book data can be obtained from the Wayback machine archives and other sources such as the NBER Research dataset. Where this data is insufficient, the Orange Book PDFs are currently the only publicly available dataset spanning a wide time range, and would require significant development effort to parse and extract the information, and as such, may be considered to be out of scope for this project.

Drug Labels

Per the historical Orange Book NBER data mentioned in the previous section, there are ~2,620 unique NDA numbers for the drugs, between 1985 and 2016. Drug Labels and Application numbers share a many-to-many relationship, and there are about 3,180 unique drug labels covered by the various NDA numbers in the Orange Book. Each label may have multiple versions and in all, there are about 21,000 label versions of interest.

Bulk updates to the drug labels are released on the DailyMed website once a month[20], in XML format. Historical labels are downloaded via an API which requires the DailyMed data's internal Set ID / SPL ID. The OpenFDA API makes it possible to correlate the NDA application number with the drug's Set ID / SPL ID, for efficient use of the DailyMed API.

[19] The latest patent and drug info may be found here

[20] The monthly drug label updates can be found here

Patents

Weekly patent archives can be downloaded from the USPTO website[21]. New data is added every Tuesday.

The Orange Book lists about 8,700 unique patents from 1985.

One of the main challenges with the patent data is extracting the claim information, along with the dependent claims. Furthermore, older patent data released by the USPTO are stored in a different format than their newer XML archives. The USPTO claims research dataset have been used partially, to address these challenges for the patent data upto 2014.

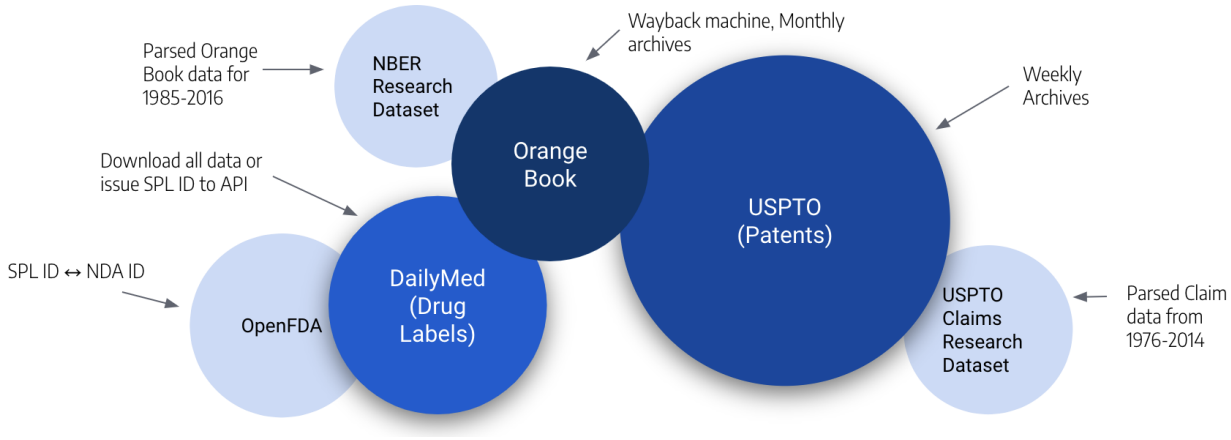


Figure 6. Data Sources. Available data sources used in the pharmaDB project.

5.3.2 ETL Pipeline and Databases

An ETL pipeline (Figure 12, 13) is set up for periodic data collection of the Orange Book data, drug labels and patents associated through the Orange Book. This pipeline will run in monthly intervals, to coincide with the frequency of issuance of new Orange Books, which are generally issued yearly, or its supplements, which are published monthly. The data would be parsed and stored in a MongoDB document database hosted on the cloud (Figure 8, Item 5).

Pipeline Components

The pipeline has been developed as modules (Figure 12) with minimal inter-dependencies. The top-level module is a pipeline orchestrator that invokes individual services that aid in the processing of data from different sources. Each service writes the result of its execution into the database which may be consumed by a different service.

[21] The weekly patent archives can be downloaded from https://bulkdata.uspto.gov/data/patent/grant/redbook/fulltext/<year>

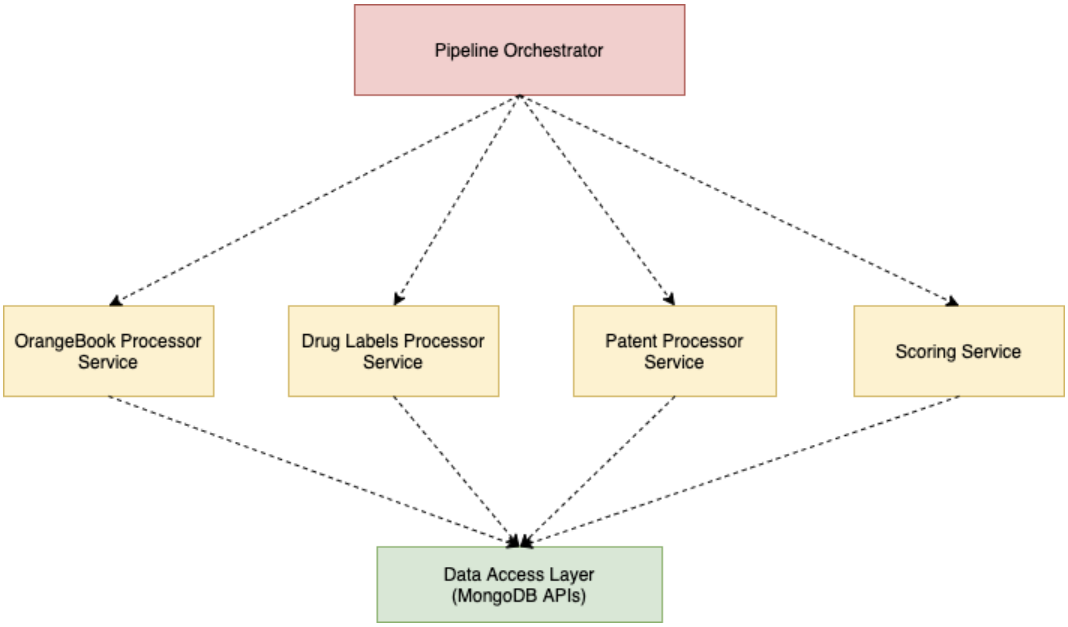


Figure 7. ETL Pipeline Components. A simplified component diagram representing the various modules in the ETL pipeline.

Pipeline Steps

The sequence of actions in the ETL pipeline can be logically visualized as a DAG (Directed Acyclic Graph) of tasks, as shown in Figure 13 below.

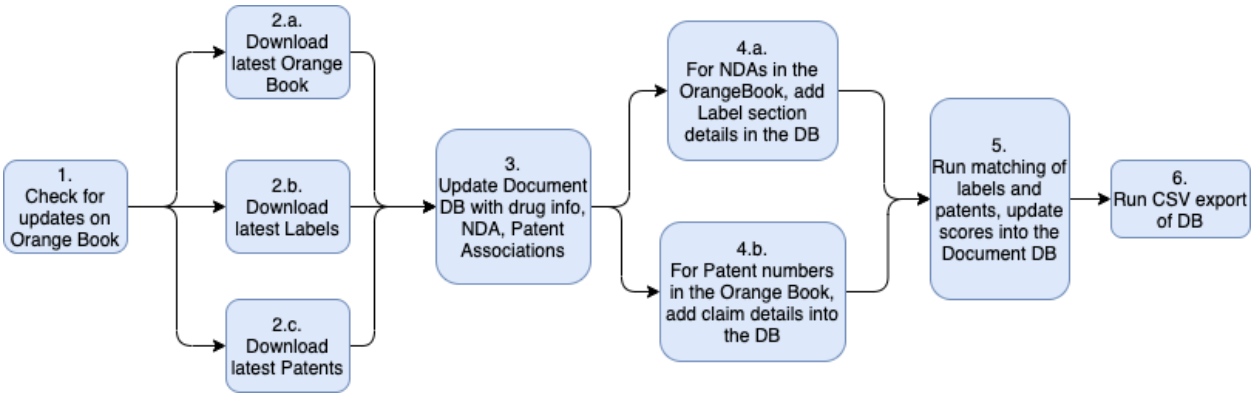


Figure 8. ETL Pipeline Overview. A broad overview of all the steps in the ETL pipeline backend process for pharmaDB.

As it can be seen, the first steps (1, 2.a and 3) in the pipeline collect the latest supplement of the Orange Book, parse and store its contents into the DB. It is expected that this step would only yield a handful of new NDA numbers and patents (as updates in a given month are minimal). However, there are no convenient APIs made available by the respective data sources to query the labels and the patents using just the information found in the Orange Book (namely, the NDA numbers and patent numbers).

This problem may be tackled in one of two ways.

1. Collect and save (initially and periodically) all the drug labels and patents published by the sources so that data related to a new association in the Orange Book would already be present in the database. However, this would require storing a large amount of data.
2. Collect and save (initially and periodically) some properties of the labels and patents that may be useful in accessing the APIs exposed by the data sources.

So as to not increase storage, the second approach is preferred. The following two maps have been created in the DB during the initial database seeding (bulk data import), which are also kept updated when the ETL pipeline is run.

- A map of the Drug NDA Application number to the DailyMed set ID – Label data can be downloaded from DailyMed using a given set ID
- A map of the Patent ID to the USPTO weekly bulk data file URL – A single patent archive file may be downloaded from the USPTO website

Steps 2.b and 2.c in the pipeline help update this mapping while 4.a and 4.b use this mapping to download the required labels and patents and save them to the DB.

The last step (5) in the pipeline matches the drug labels and patent claims, using a combination of *rule-based processing* and *machine learning*. The Machine Learning phase calculates the similarity between two pieces of text, namely the patent claim and the label sections, and is described in further detail below. The 'rules' assist in pre-processing of the data, handling special scenarios such as:

- Chaining dependent patent claims
- Grouping of drug label data extracted from tables
- Processing of numbers and measurements (eg: K vs Kelvin, 5000 vs five thousand, etc.)

In the final step 6, the data in the DB is exported as a CSV file, zipped and hosted using a simple server on the ETL pipeline's instance, for the web app to download upon user request.

5.3.3 Machine Learning

For performing Natural Language Processing operations on our patent additions and label data, we are relying on diff-match-patch to pick out the additions between consecutive labels, and the sentence_transformer library to perform the similarity comparison. The stsb-mpnet-base-v2 model was selected based on our study of its performance compared with other models and engines. Please refer to *6.2. Natural Language Processing Engine Selection* for further discussion on the NLP selection process.

5.3.4 Data Model

The MongoDB database will contain the following collections, which may be readily exported as JSON.

- **Labels** – Indexed by the NDA number and DailyMed Set ID. Each label document in a given set id is stored as an individual document in the collection and the field SPL version can be used to identify the version. The label data in this collection is limited to those NDA numbers that appear in the Orange Book.

```
{
    application_numbers: ["",], // Link to the Orange Book
    nda_to_patent: [
        // Association of each application_number to patents
        {
            application_number: "",
            patents: ["",]
        }
    ]
    setid: ..., // Association to DailyMed
    spl_id: ..., // The unique SPL id of the version
    spl_version: ..., // The numerical sequencing
    published_date: ..., // The publication date of label
    name: ..., // The brand name
    generic_name: ..., // The generic name
    active_ingredient: ..., // The active ingredient
    sections: [
        // Holds the sections of interest
        {
            name: "INDICATIONS AND USAGE",
            text: "",
        },
    ],
    // identifiers for previous and next label
    previous_label_published_date: ...,
    previous_label_spl_id: ...,
    previous_label_spl_version: ...,
    next_label_published_date: ...,
    next_label_spl_id: ...,
    next_label_spl_version: ...,
    diff_against_previous_label: [
        // Sections from above are broken down into diffs
        {
            name: 'INDICATIONS AND USAGE',
            text: [
                    [
                      1,     // -1 means subtraction
                             // 0 means no change
                             // 1 means addition
                      '...', // text segment
                      '0',   // reference to `additions` key
                    ],

            ],
            parent: ... // parent name if this is a subsection
        },
    ]
    additions: {
        '0': {        // key from diff_against_previous_label
            expanded_content: '...',  // context around
                                      // section text
            scores: [
                {
                    patent_number: 'XXXXXXXX'
```

- **Patents** – Indexed by the Patent number, contains the claims text and other metadata. The patent data in this collection is limited to only the numbers appearing in the Orange Book.

```
{
    patent_number: ...,
    claims: {
        "CLM-00001": "",
        "CLM-00002": "",
        ...
    },
    published_date: ...
}
```

- **Pipeline** – Each document stores metadata about each pipeline run.

```
{
    timestamp: … // The last successful run of the pipeline
}
```

- **OrangeBook** – Each document holds an NDA ↔ patent association.

```
{
    nda: ..., // NDA application number
    patent_num: ..., // Patent number
    created_at: ..., // The timestamp at which the mapping is
                     // added to MongoDB, used by the pipeline
}
```

- **LabelHelper** – Each document stores an NDA number and a list of DailyMed Label SetIDs associated with it. This collection is used and updated by the ETL pipeline and the data covers all associations encountered in the DailyMed website. Documents are indexed by the set ID.

```
{
    nda: "", // NDA application number
    set_ids: "" // A single DailyMed SetIDs linked to the NDA
}
```

- **PatentHelper** – Each document stores limited data on the patents, i.e., the patent number, its published date and the USPTO weekly archive file URL containing its data. This collection is used and updated by the ETL pipeline and the data covers all patents published by the USPTO.

```
{
    patent_number: "",
    published_date: ...,
    uspto_archive_url: ""
}
```

5.4 Deployment and Deliverables

The application will have three main deliverables.

1. A live public website deployed in AWS infrastructure that can be used to search for and display drug patent, label, and claim data, as well as differentials and correlations between the data. This website will be live for 1 month following the completion of the course, with the option to migrate the application to the customer to continue hosting it.
2. A series of public code repositories complete with documentation – these repositories will live indefinitely and will include the entirety of all of the code in every component and script of the application[22]. Markdown README style documentation will be included to help future developers understand the application interfaces and get the app setup in their own cloud environment. Code comments and annotations will be included in the application code in order to help developers extend the application functionality to fit their individual use case, if need be.
3. A data repository that contains all of the pre- and post-processed data needed to run the application. This could be delivered privately over a file transfer service. However, we would prefer to provide this data publicly and will do so, provided we can find a data repository service that can store the entirety of our large dataset indefinitely and free of charge.

One of our goals for this project is to make it accessible to other developers and potential users. With this in mind, we've planned to keep our application architecture and deliverables as agnostic as possible to the cloud platform, by providing a majority of the functionality and value via code based features, as opposed to building around any one cloud provider's product offerings. However, in situations where a cloud provider emphasis can't be avoided, our documentation will focus on an AWS implementation since they seem to be the most popular choice today.

6. Testing Results

6.1 Unit Tests

Unit tests and regression tests are being added to every individual code repository such that all major functionality written for the project would be exercised. This includes the backend APIs, data processing services and the Machine Learning models. The exceptions to this are the infrastructure set up code

[22] pharmaDB's official GitHub Organization

(using Terraform) and the Angular UI, and the latter is covered by other tests that are described in the following section.

6.2 Web App User Interface Testing

The Web App is tested with a combination of manual and automated tests. Manual tests are ad hoc to evaluate usability. The automated tests are implemented using Selenium.

6.2.1 Automated tests

Test	Description
Validate UI elements	Verifies expect UI elements are found on the search page. Includes 4 individual tests.
Search by criteria	Tests searching by name, NDA number, and manufacturer. Functionality is verified by counting the number of results returned. Incl 3 individual tests.
View Drug details	Check that the expected drug details are found when navigating to a label detail screen from search results. Includes 4 individual tests.
Select drug labels	Select two labels from the timeline and checks the expected number of differences. Includes individual tests.

6.3 Web App API Testing

All of the web app API endpoints are tested via the popular API development platform called Postman. With Postman, the API endpoints are called with a pre-programmed URL and query params and the HTTP response is validated against the pre-programmed expected response, also known as an assertion. The Postman tests and related Postman collection have been exported and are included in the PharamDB API github repository.

6.4 Natural Language Processing Engine Tests and Selection

Extensive tests were performed as part of the natural language processing engine selection process. As these tests were already discussed in section 1. They will not be repeated here.

6.5 Data Aggregation Testing

Data aggregation testing focused primarily on the data conversion process. Data from the USPTO is collected in 3 different formats depending on the year the bulk data was compiled by the USPTO. All 3

different patient data structures are converted into a single unified format, and tests are run on a few sample data in each of the 3 formats to ensure parsing was successful and that a common format was created. A total of 18 unit tests are run (6 unit tests per USPTO patent format), and the expected converted data is checked against predefined actual formatted data to ensure the conversion process was successful.

7. Risks & Constraints

7.1 Data Risks

The following are some potential risks concerning the data aggregated and processed by our system.

- **Sourcing Historical Orange Book Data** – A key value add of our solution is the incorporation of historical OrangeBook entries and historical drug labels. Historical OrangeBook data is not readily available and while we've found sources for this data, we cannot guarantee its accuracy. Accurate historical drug labels are available from a trusted government agency but only in formats that aren't easily ingestible programmatically. This leaves some concern for the reliability of our historical drug labels feature and whether or not the updating process can be easily executed or automated.
- **Performance of Similarity matching** – We would need to evaluate various approaches to meaningfully correlate the label and claims data. The performance of popular NLP algorithms on our dataset, at this point, is unknown and may need several iterations to achieve a high level of staBioBERTtistical accuracy.
- **Validating Similarity matching** – There is a lack of existing datasets or baselines to evaluate the performance of our solution for matching the claim text and drug label data. This would require us to create a test dataset (perhaps with the help of the client), and this carries two risks. Firstly, due to the time limitation, we may only be able to compile a small dataset. And secondly, since this dataset would not be validated widely and by multiple experts, there is a possibility that it carries biases. This makes it difficult to quantify the results of our ML model.
- **Dependence on public APIs** – Much of our application's data and functionality relies heavily on the fact that federal agencies such as the FDA, NLM, DailyMed and USPTO have pushed for the democratization of data by implementing free public facing APIs for anyone to incorporate into their applications. We're hoping those agencies continue these initiatives. Without the free offerings and continued stewardship of the APIs and data provided by these institutions, the data required for our app to properly function may come at a cost or simply cease to exist in any usable format.

7.2 Security Risks

The application has no concept of user accounts or user data beyond ephemeral event based data regarding which HTTP requests were made from which clients and when. Additionally, incorporation of HTTP encrypted over TLS (HTTPS) between the application servers and the client machine should provide an effective defense against third parties monitoring request data such as request bodies and query parameters.

In regards to the CIA triad principle[23] of confidentiality, our application's lack of persistent storage of search queries and metadata, and encryption of drug search queries by users through transport, ensures a strong level of confidentiality of user data. Additionally, if confidentiality is of highest concern, any temporary caching of search queries can be completely disabled, reducing the chance of metadata leaks. Finally, the open source nature of the project allows users to copy the code and run the project in any locality they desire so that they can comply with all applicable data protection laws.

In order to ensure the application's data integrity and overall availability, we aim to implement AWS' best practices in security by putting sensitive components within a secured VPC while the Web Application API acts as a bastion host between critical components and the outside world[24]. With this implementation, only developers who are granted access to the VPC will be able to modify data in storage or change sensitive components of the application.

Another potential security risk to consider is costly traffic spikes from unwelcome visitors to our website or API. While this doesn't affect the confidentiality or integrity of the application, it could have an effect on AWS costs thus impacting availability. Incorporating a web application firewall and filtering guests by IP address may be an effective solution for us to block unwanted visitors while the project is under development.

After delivery of the application, cybersecurity will be largely up to anyone who clones the code base and runs the application in their own cloud. We may suggest best practices similar to the design in the previous paragraphs, but the application will only be as secure as the next development team is willing to make their environment. We've also considered the OWASP Top 10 Web Application Security Risks[25] when creating the application.

OWASP Security Risk	**Comment**
Injection	No part of incoming requests run in an executable cmd style
Broken Authentication	Authentication is not a requirement of the application
Sensitive Data Exposure	Sensitive data is encrypted in transit and not stored
XML External Entities (XXE)	Not applicable in the NodeJS express env
Broken Access Control	No user restrictions exist, all users are treated the same
Security Misconfiguration	Continuously patched AWS services are used as well as the r up to date versions of libraries
Cross-Site Scripting XSS	XSS is restricted by our API to only our subdomains

[23] https://www.f5.com/labs/articles/education/what-is-the-cia-triad

[24] https://blog.osgcorp.com/cloud/aws/setup-vpc-web-app/

[25] https://owasp.org/www-project-top-ten/

Insecure Deserialization	Deserialization is provided by ExpressJS and there is no destructive operations in the API, only GET requests
Known Vulnerabilities	NPM is used to automatically scan packages for deprecated versions that may contain known vulnerabilities
Insufficient Logging & Monitorin	our solution doesn't provide an specific guidance on how th host might prefer to log and monitor the app in their own pr

7.3 Cost Constraints

We estimate that storage costs are likely to be the largest cost consideration. The patent grant data collected includes weekly snapshots of the new patent grants in compressed zip folders published by the USPTO. With an estimated average size per zip file of 100 MB, and an estimated 3500 zip files (one for each week, starting back from 1985), the total expected size of the patent data is approximately 180 GB. However, costs can be reduced significantly by using infrequently accessed cloud storage resources to store bulk data files, such as Amazon S3 Glacier storage, and by keeping only the most useful data to the users (linked to the Orange Book) in the MongoDB database. With glacial storage, the total costs are less than $1 per month for keeping the bulk data files. Additionally, by trimming down the data to only a subset of the information and patents needed (only a small subset of patents are drug patents, and we estimate this to be around 10,000 - 20,000 patents), we estimate that the total amount of storage space in the MongoDB database will be only a few gigabytes. As a result, we don't think costs are a limited factor, but need to be managed in order to ensure they remain low for any user to be able to run the project.

8. Tools and Processes

8.1 Tools

8.1.1 Communication

Communication among the team members is primarily done on **Slack**. Where external parties are involved (such as the course instructors or the client), communication may also occur on **Canvas** or by **emails**.

8.1.2 Project Management

Our team has identified a collection of tools for use at different stages of the project, documented in the **Appendix** section, as a part of the team contract (jump to SDLC tools).

8.2 Team Strategy and Dynamics

We adopted an Agile software development model, running through iterations of the SDLC stages. Some aspects of our working are detailed below.

8.2.1 Roles

All members of the team were involved in all phases of the project. However, roles and responsibilities were divided to utilize their current skill set (jump to Skill Strength Identification), with room for picking up new skills (jump to Skill Development Identification). As an example, Terry Chau, who is familiar with the patent space, took the lead on the client interactions and requirements gathering. Similarly, Brett Bloethner, who is proficient in web development, took the lead on architecting the Web Application.

8.2.2 Decision Making

This is documented in the team contract in the **Appendix** section (jump to Decision Making).

8.2.3 Client Interactions

The team met with the client once every 1-2 weeks by Zoom calls, to gather requirements. Additionally, we also communicated offline, by emails, for questions and clarifications.

9. Reflection

This section examines the time and effort expended on the project retrospectively and identifies some areas for improvement.

9.1 Meeting the Requirements

All 'Must Have' requirements agreed upon with the client have been implemented. The following is a summary of the features.

1. Ability to search for a drug by various attributes
2. Ability to view a single drug label or a single patent linked to it in the Orange Book
3. Ability to visualize the changes in the consecutive versions of a drug label
4. Matching of changes in the drug labels to the specific claim text in the linked patents, and their display in the app
5. Bulk data exports of the label section, patent claims and the scores from their matching
6. An ETL pipeline that automatically updates the data in the DB every month

9.1.1 Changes from the Requirements Specification

The requirements for a few features evolved during the course of the project, based on the user feedback and/or the development team's concerns.

1. **Granularity of the Drug Label to Patent Claims matching**
   In the initial design, the drug labels were being matched to claims in the linked patents, one section at a time. The app would be able to report the label changes covered by a patent claim by comparing the matched claims for different versions of the label section. However, during the subsequent interactions with the clients it was determined that it would be a lot more beneficial to be able to match the claims on the individual diffs in the label section, across two consecutive versions. This entailed a major revamp of the data structures and the matching algorithm that the team managed to incorporate.

2. **Bulk Data Exports**
   The original requirement was to produce the results in csv. Instead, the team had proposed to provide a JSON export of the MongoDB database. On the one hand, this enabled the team to avoid investing in a csv export feature and instead focus more on other critical features. At the same time, this solution is also more appropriate since the data structure contains nested elements, which when flattened into rows (as in a csv), may make them unwieldy or lead to redundant information across multiple rows. Ultimately, however, the team has also added the csv export functionality, as the last feature developed.

9.1.2 Design Decisions with Minimal Changes

Since the inception of the project, the technology and workflows surrounding the user facing app have undergone little to no design change.

- The app is backed by TypeScript and AngularJs, chosen based on familiarity with the frameworks. This enabled the team members to implement the requirements quickly.
- The app directly invokes the DailyMed, USPTO and OpenFDA APIs where applicable, to minimize the load on the app's main DB. At the same time, this has helped the data collection and storage effort to only focus on procuring and saving the critical data and disregard other metadata, which the app can fetch from the data sources' APIs on the fly.

9.1.3 Design Decisions with Significant Changes

Major design changes were seen with the data collection, storage and Machine Learning workflows, owing to the following reasons.

- The deprecation of a previously accessible patent collection API forced the team to find alternatives for collecting the incremental patent data as a part of the monthly update (described in the ETL Pipeline and Databases section).
- While the original design for the data model included a graph database as the primary database and MongoDB as the secondary database, the graph DB was dropped for simplifying the development and maintenance effort.
- The change to the granularity of the label section to patent claim matching (as described in the earlier section) necessitated a change to the data model and matching algorithms.

9.2 Estimates

The team has largely managed to deliver the features according to the original estimates. However, some aspects of the project required more time to refine and consequently, the integration of the components was delayed. These include:

- Bulk data collection and processing – Changing data formats over time and attempts to reconcile data from multiple sources (eg: the Orange Book data has been compiled from over three different websites) took several iterations to get right.
- Changes to the matching algorithm and the generally poor performance of the original modelling approaches used resulted in the team having to look for ensembling techniques and other innovative ways to implement a reasonable solution. This cost the team significant time and effort.
- Iterative changes to the user interface, although not unusual, were difficult to account for in time estimates. User interviews led to numerous iterations in the user interface and application functionality which required the team to revisit previously completed tasks in order to optimize the user experience. The landing page and time-based viz tasks were especially prone to these sorts of delays.
- Experimentation with front end libraries also led to delays specifically in the time-based viz task. Outsourcing the timeline code to a third part library saved immense amounts of time but also required more experimentation than we originally thought as we worked with multiple libraries to find the one that best fit our use case.

Owing to these circumstances, the integration of the data related work had to be moved by 3-4 weeks, as well as the integration with the web application, with no room for implementing the 'Should Have' and 'Could Have' requirements.

The original estimates may be found under the Appendix section on Timeline and Capacity Planning. Below is a comparison of estimated vs actual time spent on some of the prominent features.

Task	**Estimated (Man Weeks)**	**Actual (Man Weeks)**
Bulk Data Collection and Processing (Data)	4.5	9
ETL Pipeline	6	3
Scoring Algorithm (Data)	4.5	6
Label Diff (Data; originally planned for Web App)	0	4
Export Feature (Data; original planned for Web App)	0	1
Landing Page (Web App)	2.5	3.5
Time-Based Viz (Web App)	3	4
Label Diff (Web App)	1.5	3

Label to Patent Mapping (Web App)	2	3
CSV export (Web App)	1	.25
AWS Infrastructure (Web App)	1	2
Integration and Testing (Web App)	1	2

9.3 Lessons Learnt

We have three key take-aways from this project.

1. **Early Data Collection and EDA**
   Besides ML modelling decisions, data can also drive the design of its overall architecture. While we originally envisioned the entire solution to be built based on a small dataset that could easily be translated to the full dataset when the data collection was complete, obtaining the historical data brought about some surprises in the form of missing data and inconsistent formats. In hindsight, we would have liked to prioritize this over the other Engineering tasks such as infrastructure set up.

2. **Tighter User Feedback Loop**
   Demonstrating the progress to the user regularly has helped gather valuable feedback quickly. Both in terms of the user experience and the overall solution, our customer had specific requirements that we were able to understand better with every interaction and demonstration of the project, thus averting surprises in the last minute.

3. **Over-Communication is Key**
   As the team is spread across multiple time zones and it isn't always possible to find a common meeting time, much of the communication occurred on Slack. In addition, the team had been proactively documenting important design decisions and other knowledge in Confluence, Pull Request messages, etc. which helped bridge the gap on many occasions.

10. Appendix: Requirements

The functional requirements for this application were elicited through interviews with our customer, Dr. Marman. His insights and feedback were critical in helping us identify exactly which features were needed in order to deliver an application that helps drug researchers find and cross reference information about drug labels and patents.

Our goal was to focus primarily on what Dr. Marman and the project group identified as the key value adding features while still considering other features and improvements for implementation at a later

date. We used the MoSCoW prioritization method to categorize project requirements into four priority based groups.

- **Mo | Must Have:** These requirements are critical to the project's success. If any of these requirements are not completed or reprioritized within the project's timeframe then the project is thought to have not met the customers expectations. This is also known as the “minimum usable subset” or requirements and can also be thought of as the requirements of the project as it stands as a “minimum viable product.”
- **S | Should Have:** Should have requirements are oftentimes just as valuable as must have requirements but they don't face the same sense of urgency. Since this project has a short and definite timeline, we have few requirements that fall into the should have category.
- **Co | Could Have:** Could have requirements are features that are desirable but not critical and will only be implemented if resources and time allows.
- **W | Will Not Have:** Will not have requirements are requirements that were determined by stakeholders to not be valuable enough to be prioritized during planning. They’re recognized as potential features but are specifically not to be worked on.

High level requirements were converted into epics, each of which falls into one of the four MoSCoW categories. Each epic includes user stories as well as functional and nonfunctional requirements along with “t-shirt” style estimation. MoSCoW prioritization helps the project team decide what features to work on first while t-shirt size style estimation helps the team determine how long each feature may take to implement. *Each of these requirements, their prioritization and estimation has been reviewed and approved by our customer on 3/2/2021.*

10.1 “Must Have” Requirements

10.1.1 Functional Requirements

	Requirement	**Estimate**
1	**Epic:** Search Bar/Landing Page	

1.1	As a drug researcher, I want to be able to search for a drug in the Orange using the drug's brand name, active ingredient, NDA number, or manufa name, so that I can narrow the scope of my research to the particular dr **Given** that the search bar is empty **When** an active ingredient is entered into the search bar **Or** a proprietary drug name is entered into the search bar **Or** an NDA number is entered into the search bar **Or** a drug manufacturer name is entered into the search bar **Then** a list of drugs having the respective query component is displayed **And** a drug can be selected from the list of drugs to narrow the scope of research to a drug, as defined by an NDA number.	M
2	**Epic:** Time-based Visualization for a Drug from the Orange Book	
2.1	As a drug researcher, I want to view a time-based visualization of drug la various publication dates, for a single drug, as defined by an NDA numbe that I can tell if there are changes to the labeling for the drug over time. **Given** a list of drugs is displayed after a search query **When** a drug entry is selected **Then** a time-based visualization of drug labels, at various publication dat the drug entry, as defined by an NDA number, is shown.	M
2.2	As a drug researcher, I want to view a time-based visualization of patents various dates for a drug, as defined by an NDA number, so that I can dete if a drug is covered by intellectual property rights, and study the patent h for a particular drug over time. **Given** a list of drugs is displayed after a search query **When** a drug entry is selected **Then** a time-based visualization of patent numbers, at corresponding pa issuance dates, for US patents related to the drug by way of the Orange I shown.	M
2.3	As a drug researcher, I want to view a single label (at a publication date a version of my choosing), so that I can read the usage details of the drug. **Given** that a drug may have one or more labels **When** a single label is selected **Then** content of the selected label is shown.	S
2.4	As a drug researcher, I want to see the FDA approval date for the drug in based visualization, so that it can be used as a reference point, since the Orange Book submission dates always come after the FDA approval date **Given** a list of drugs is displayed after a search query **When** a drug entry is selected **Then** the FDA approval date for that drug is shown on the time based visualization.	L

3	**Epic:** Mining Diff Between Drug Labels for a Drug from the Orange Book	
3.1	As a drug researcher, I want to view a diff of two drug labels (each at a publication date and/or version of my choosing) for a drug, as defined by NDA number, that way I can compare how the usage or features of the d may have changed over time. **Given** that there are more than one label for a particular drug, defined b NDA number **When** two labels for the drug are selected **Then** the before and after differences between the two labels, on a secti section basis, will be shown **And** links to the two labels will be provided.	L
3.2	As a drug researcher, I want to see the changes between multiple drug la with consecutive publication dates in a single view. **Given** that there are more than two labels for a particular drug, as define an NDA number, each published at different publication dates, **When** two non-consecutive drug labels are selected **Then** the changes from each label to the next between the two non-consecutive labels are displayed.	L
4	**Epic:** Mining Delta Between Labels to Patent Claim Mapping for Drugs from the Orange	
4.1	As a drug researcher, I want the patent claim that best recites the delta between two drug labels of a drug, as defined by a single NDA number, identified, so that I can understand which patent claim best covers the fe in the delta of the drug labels. **Given** that a drug, as defined by an NDA number, has multiple labels **And** multiple labels have deltas therebetween **And** each delta has a before state and an after state **When** a delta is selected **Then** a patent claim, from the list of patents associated with the NDA nu for the drug, which best describes the after state of the delta, is shown.	L
5	**Epic:** Bulk Data Collection	

5.1	As a drug researcher, I want to be able to work with historical data from Orange Books, past Orange Book supplements, past drug label datasets a past patent datasets, so that I can observe changes to the drug labels, ar changes to the patent scope, over time. **Given** that a drug is listed on the FDA's Orange Book Data File Page, or hi copies of that page from the Wayback Machine, or by nber.org's Orange Patent Data from 1985-2016 page **When** the drug is queried **Then** the query results should include that drug **And** corresponding drug labels should be made available **And** corresponding patent information should be made available.	L
6	**Epic:** ETL Pipeline	
6.1	As a drug researcher, I want to work with the most up-to-date Orange Bo drug label datasets from the FDA, and the most up-to-date patent datase from the USPTO, so that I can observe changes to the drug labels, and ch to the patent scope, over time. **Given** that there is an update to the Orange Book **When** the update occurs **Then** the running system should pull in the data from the latest Orange B its supplement, and corresponding patent dataset and drug label datase	M
7	**Epic:** Button to Export Data to CSV	
7.1	As a drug researcher, I want the ability to download the bulk data, which includes all the metadata for the timelines and for the comparisons, in ta form, at any future point in time, so that I can review its content. **Given** that the bulk data for the timelines and for the comparisons grows time **When** an export button is pressed or a script is run **Then** metadata for the timelines, metadata for the deltas between labels metadata for the mapping of the deltas to patent claims will be exported tabular form.	M

10.1.2 Nonfunctional Requirements

	Requirement	**Estimate**
8	**Epic:** Responsiveness	

8.1	As a drug researcher, I want to work with a product that is responsive, so my research is not impeded. **Given** that the search bar is empty **When** a query is entered into the search bar **Then** the query results should return in a reasonable time (under 5 secor **Given** that query results are shown **When** there is interaction with drug label or patent content **Then** the response time should be reasonably quick (under 5 seconds).	N/A
9	**Epic:** Recoverability	
9.1	As a drug researcher, I want a recoverable copy of the codebase, so that continue to use the product after development has ended. **Given** that development of this product is complete **When** the development ends **Then** a copy of the entire code base, corresponding documentation, and instructions for hosting the code base, will be supplied to the customer **And** the code base, corresponding documentation, and instructions for h the code base will be released under an open source license.	S

10.3 “Should Have” Requirements

10.3.1 Functional Requirements

	Requirement	**Estimate**
10	**Epic:** Button to Export Data to Csv	
10.1	As a drug researcher, I want to export hit results (including the metadata the timelines and for the comparisons) to tabular form, so that I may use output to augment my research. **Given** that a list of drugs is displayed after a search query **And** a drug entry is selected **When** an export button is pressed **Then** metadata for the timelines, metadata for the deltas between label metadata for the mapping of the deltas to patent claims will be exported tabular form.	M

10.3.2 Nonfunctional Requirements

11	**Epic:** Portability	
11.1	As a drug researcher, I want to transfer ownership of hosting the final pr from the development team to another host of my choosing, so that I m continue to use the product after development has ended. **Given** that AWS permits transfer of hosting ownership **When** the development ends **Then** hosting ownership will be transferred to the customer.	N/A

10.4 “Could Have” Requirements

10.4.1 Functional Requirements

	Requirement	**Estimate**
12	**Epic:** Search Bar/Landing Page	
12.1	As a drug researcher, I want to be able to search for a drug in the Orange by way of a patent number, so that I can narrow the scope of my researc the particular drug covered by the patent. **Given** that the search bar is empty **When** a patent number is entered into the search bar **Then** a list of drugs, having NDA numbers from Orange Book, related to t patent, is displayed.	S
13	**Epic:** Time-based Visualization for a Drug from the Orange Book	

13.1	As a drug researcher, I want to see submission dates for all patents in the Orange Book, even for the case in which clean text/csv versions of submi dates are unavailable and the submission dates must be pulled from pdf versions of the Orange Book or its supplements, so that I may consider submission dates for litigation purposes. **Given** a list of drugs is displayed after a search query **When** a drug entry is selected **Then** submission dates for any patent in the Orange Book, across the ent history of the Orange Book, are also viewable.	M
14	**Epic:** FDA Submission Paperwork for a Drug Label	
14.1	As a drug researcher, I want to see a link to the FDA submission paperwo drug label, when available, so that I may glean useful dates and other tid information from that paperwork. **Given** that a drug may have one or more labels **When** a single label is selected **Then** FDA submission paperwork is shown.	S
15	**Epic**: Exportable Table or Gnatt Chart of Drug's Appearance and Disappea from Orange Book	
15.1	As a drug researcher, I want the ability to export to a table, or view on a (chart, information on the lifespan of drugs, including when the drugs firs appeared and when the drugs went off patent and disappeared from the Orange Book, ideally in a sorted or sortable manner by company and dise area, so that I holistically understand the history of drug development. **Given** that drugs appear and disappear from the Orange Book **When** an script to export to a table or a button to produce a Gnatt chart triggered **Then** information on the lifespan of drugs, including when the drugs first appeared and when the drugs went off patent and disappeared from the Orange Book, ideally in a sorted or sortable manner by company and dise area, is produced.	L
16	**Epic:** Conglomerate Analysis of Drug Labels and Patents	
16.1	As a drug researcher, I want to know about drug label and patent change time, with respect to the corpus of all drug labels and patents, and how t those changes are, so that I can observe trends. (Acceptance criteria unclear at the moment. Further study is needed on t type of changes that are typical across multiple patents and multiple labe multiple drugs over time.)	XL
17	**Epic:** Data and Feature Set Exploration	

17.1	As a drug researcher, provided I have a new readily available tool that inc a unique and comprehensive data set of drug metadata as well as time s labels, patents, and orange book entries, I want to know what additional insights I can gain from analyzing this data beyond the scope of the relati between patent claims and labels. (Acceptance criteria unclear at the moment. This story may include a ser compelling use cases that are addressed by the application but are not ir scope of our customers overall intended use of the application. This may include demos or a documentation related to those additional use cases provided they can be executed through functionality in the final delivera	L
18	**Epic:** Patent Payment Schedules and Active State	
18.1	As a drug researcher, I want to see patent payment schedules and if a pa active for a patent from the Orange Book. **Given** that a patent is referenced in the Orange Book **When** information about the patent is shown **Then** payment schedules for the patent and the active state of the paten also displayed.	M

10.5 “Will Not Have” Requirements

10.5.1 Functional Requirements

	Requirement	**Estimate**
19	**Epic:** Bulk Data Collection	
19.1	As a drug researcher, I want to see the information about the exclusivity and exclusivity_date from the exclusivity.txt file within the compressed from the FDA’s Orange Book Data File Page.	M

11. Appendix: Timeline and Capacity Planning

This section examines the **“Must Have” Functional User Requirements** (our MVP) from the point of view of the implementation and lists down the tasks and the corresponding estimates. This would also include those items that are not captured directly by any of the user requirements but are required internally for more efficient development cycles (such as setting up test datasets).

Since we would like to allow for a month for integration testing and development of the must have nonfunctional requirements and other lower priority user requirements, we would aim to target completion of our MVP around the **15th of April**. With this in mind, the effort is broadly divided under Data and Web Application categories and per the demands of the tasks, the team (of 5 members) is planned to be split into 3 and 2, respectively.

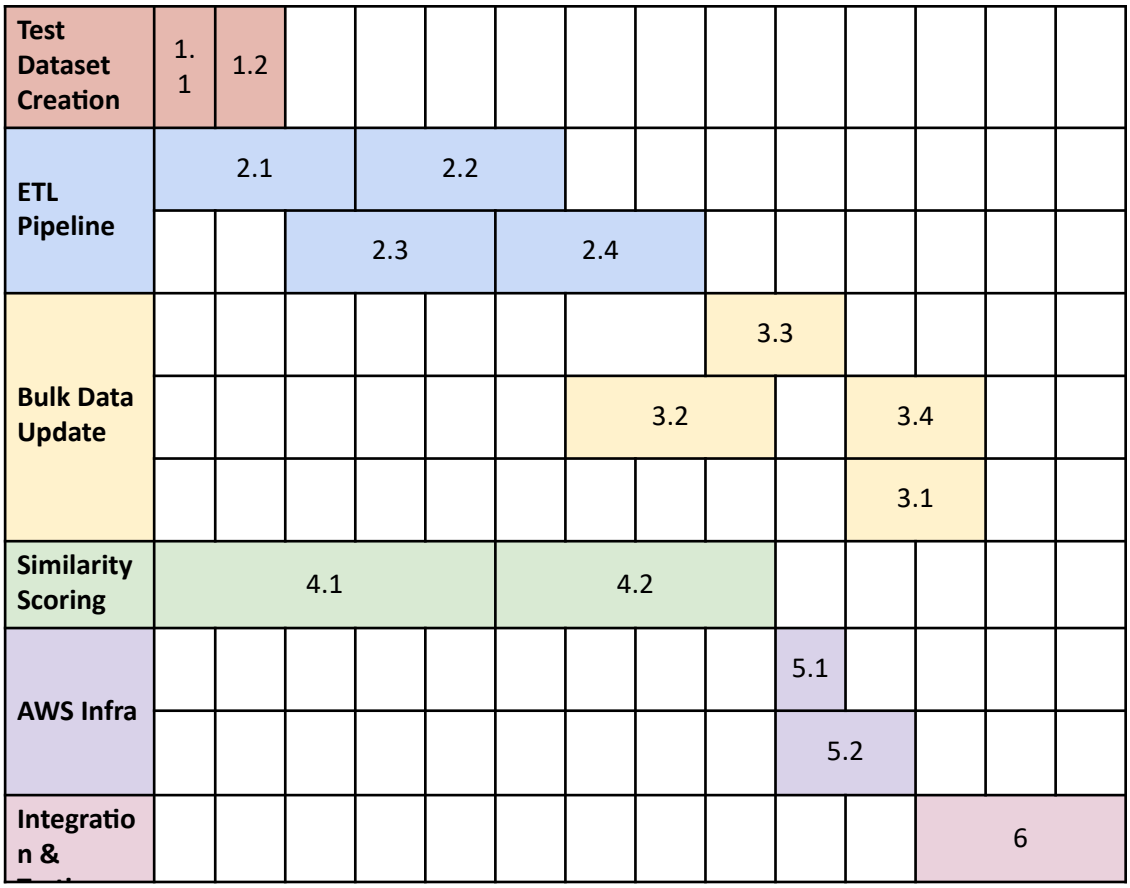
11.1 Data

	Task	**Estimate**
1	Test Dataset Creation	
1.1	Create local Docker set up for graph DB and document DB	S
1.2	Put together a test dataset	S
2	ETL Pipeline Creation	
2.1	Prepare scripts for the Orange Book	S
2.2	Prepare scripts for the Drug Labels	M
2.3	Prepare scripts for the Patents	S
2.4	Integration of all steps	M
3	Bulk Data Collection	
3.1	Orange Book	M
3.2	Drug Labels	M
3.3k	Patents	M
3.4	Integration testing	M
4	Similarity Scoring	
4.1	Iterate over various approaches and models	L
4.2	Pipeline scripts and bulk data processing	M
5	AWS Infra	
5.1	Set up DB infra on AWS	S
5.2	Set up scheduled pipeline on AWS	M
6	Integration and Testing	L

11.1.1 Timeline

Capacity: 3 Developers

	Week of Mar 8	**Week of Mar 15**	**Week of Mar 22**	**Week of Mar 29**	**Week of Apr 5**	**Week of April 12**	**Week of April 19-26**


Test Dataset Creation
1.1
1.2
ETL Pipeline
2.1
2.2
2.3
2.4
Bulk Data Update
3.3
3.2
3.4
3.1
Similarity Scoring
4.1
4.2
AWS Infra
5.1
5.2
Integration &
6

11.2 Web Application

	Task	Estimate
1	Search Bar/Landing Page	
1.1	Implement Drugs@FDA API call proxy and cache (Web App API)	M
1.2	Implement home/search bar view (Web App)	M
2	Time-based Visualization for a Drug from the Orange Book	
2.1	Implement patent text, label text and ETL related API calls	M
2.2	Create front-end view for visualizing time series patents and labels	M
3	Diff Between Two Drug Labels for a Drug from the Orange Book	
3.1	Create strategy to efficiently fetch patent label subsections (Web App AP	S
3.2	Implement view for showing label diff (Web App)	M
4	Delta Between Label to Patent Mapping for Drugs from the Orange Book	
4.1	Implement view to show time series ETL data for label/patent correlation	L
5	Button to Export Data to CSV	
5.1	Add UI Button, Implement Web App API	M
6	AWS Infra	
6.1	Set up API server on EC2	S
6.2	Set up Frontend S3 Bucket	S
7	Integration and Testing	M

11.2.1 Timeline

Capacity: 2 Developers

	Week of Mar 8	Week of Mar 15	Week of Mar 22	Week of Mar 29	Week of Apr 5	Week of April 12	Week of April 19-26
Landing Page	1.1	1.2					
Time-based Viz	2.1	2.2					
Label Diff			3.1	3.2			
Label to Patent Mapping					4.1		
CSV Export					5.1		
AWS Infra						6.1 6.2	
Integration & Testing							7

Note: After integrating the web application with the data pipeline, we will focus on additional Requirements outside of the "Must Have" group of requirements. The priority of all requirements are noted above, but we will consult with the client before proceeding further.

12. Appendix: Team Contract

12.1 Project Vision

The team intends to work towards achieving the client objectives and requirements by delivering a solution that improves the client's workflow to analyze changes in US Drug Labels and match them to claims in the US Patents, for both historical and new drug label and patent information.

12.2 Expectations

Expection	Example
Communication	Check project communication channels (email, Slack, Canvas, Piazza, etc. keep up to date with ongoing information and changes in client requirem
Communication	Share updates on project milestones as progress is made towards those milestones

Communication	Share feedback with the team to improve collaboration and to meet clien objectives.
Communication	Meet as needed to achieve client objectives (likely once a week, or more as needed).
Communication	Coordinate communications between the team and the client to ensure t client objectives are being met.
Leadership	Ensure the team is on track to meeting client deadlines.
Accountability	The developer covers all aspects of SDLC (or ensures that it has been / wi done) for every component implemented
Accountability	Ensure the team's deliverables meet the expected functional and non-functional client requirements through reviews of code, architectural des and client objectives.
Accountability	Should issues arise after a component is delivered to the client, work wit client and the team to ensure the necessary steps are taken to rework th component to meet client expectations.
Team Work	Decisions will be made together, prioritizing the collective efficiency and the team with the aim of meeting client objectives in an efficient manner

12.3 Processes and Tools

12.3.1 System Development Life cycle (SDLC)

The following steps will be taken iteratively, for each feature required by the client.

Phase	Workflow	Tools
Planning	● Client interactions to gather domain knowledge at a high level, their requirements ● Document the information gathered from the cli ● Decide major milestones for the client	Confluence, GitHub Projects Kanban board, Slack, Zoom
Defining	● Create the Software Requirements Specification document, detailing the functional and non-functional requirements ○ Include priorities for the features ○ Define a timeline for each client feature	Google Doc
Designing	● Create the Design Document Specification ● Circulate the document among the team for revi and future reference by the team	Google Doc, Balsamiq

Building	● Develop the functionality in a shared repository owned by the team ● Use PRs for reviewing and getting approval from peers prior to code merges ● Provide in-repository markdown documentation the source code and additional notes as needed ensure easier review by the team members and maintainability	GitHub, GitHub Actions
Testing	● Unit and integration tests should be performed t ensure that the system will meet client expectat and needs	Google Sheets, Confluence, Code based testing
Deployment & Maintenance	● Automate Deployment of the component ● Deploy ● User Acceptance Testing	GitHub Actions, Jenkins

12.3.2 Communication

Internal communication will primarily be done on Slack. Other communication may also occur by email or on Canvas, when project instructors or external stakeholders are involved. Lastly, relevant team members can also communicate on other project tools, as appropriate (such as GitHub and Confluence).

12.3.3 Decision Making

Decisions will be made based on group decision making, a collective understanding of the requirements, and technological merit for solving the task at hand at each step of the project.

12.4 Skill Strength Identification

This section lists the primary hard and soft skills that each member brings.

Name	Skills
Anthony Mancini	Languages: Python, TypeScript, JavaScript, Java, C#, C++ Backend: Django, Flask, Express, Rails, ASP.NET, AWS (some Azure and GCP) Frontend: React, Vue, Preact, Svelte Databases: Postgres, MySQL, MariaDB, MongoDB, Redis, AWS DynamoDB, A Neptune Big Data: AWS EMR, Hadoop, Apache Spark Data Science: TensorFlow, PySpark, Pandas, Numpy, Jupyter Notebook, Matp
Brett Bloethner	Languages: Typescript, JS, Java, Python Backend: Typescript, JS, Python, Express, Rails, AWS, DigitalOcean Databases: MongoDB, ElasticSearch, Postgres, DynamoDB, blockchain

Krithika Sundararajan	Languages: C++, Java (low), Go, Python Backend: Django, Flask, Go Frontend: React Databases: Postgres, MongoDB, Redis, Neo4J, Cassandra Big Data: Spark, Hadoop (low) Infra / Tooling: Shell scripting, Docker, AWS, GCP Data Science: Python, R (low), Tensorflow
Michael Thornton	Languages: Swift, Kotlin, Java, C#, Objective-C Backend: Very little in the last 10 years Frontend: Native iOS and Android apps Database: mysql, sqlite, MSSQL Server
Terry Chau	Languages: Python, JavaScript, Java, C++, R, matlab Backend: Django, Flask, Express Frontend: React, Angular Databases: MongoDB, SQLite Big Data: Hadoop Data Science: Pandas, Numpy, Jupyter Notebook, Matplotlib, pytorch, tenso (low)

12.5 Skill Development Identification

This section describes the skills that each member would like to develop through the course of this project.

Name	Skills
Anthony Mancin	Understanding of the drug patent space and improved familiarity with the t stack and tools used by other team members.
Brett Bloethner	Further develop my skills building APIs in Typescript and get a better unders of how data processing works between non standardized datasets.
Krithika Sundararajan	Text Mining, Familiarity with Java
Michael Thornton	Web development
Terry Chau	Further develop my skills in natural language processing and project manag

12.6 Team Member Availability

Most members are located in the US and thus, the scheduling will be done in EST. The general availability is 6-10 PM on Monday - Thursday. Timings are more flexible on the weekends and will be coordinated as required.

13. Appendix: Images and Diagrams

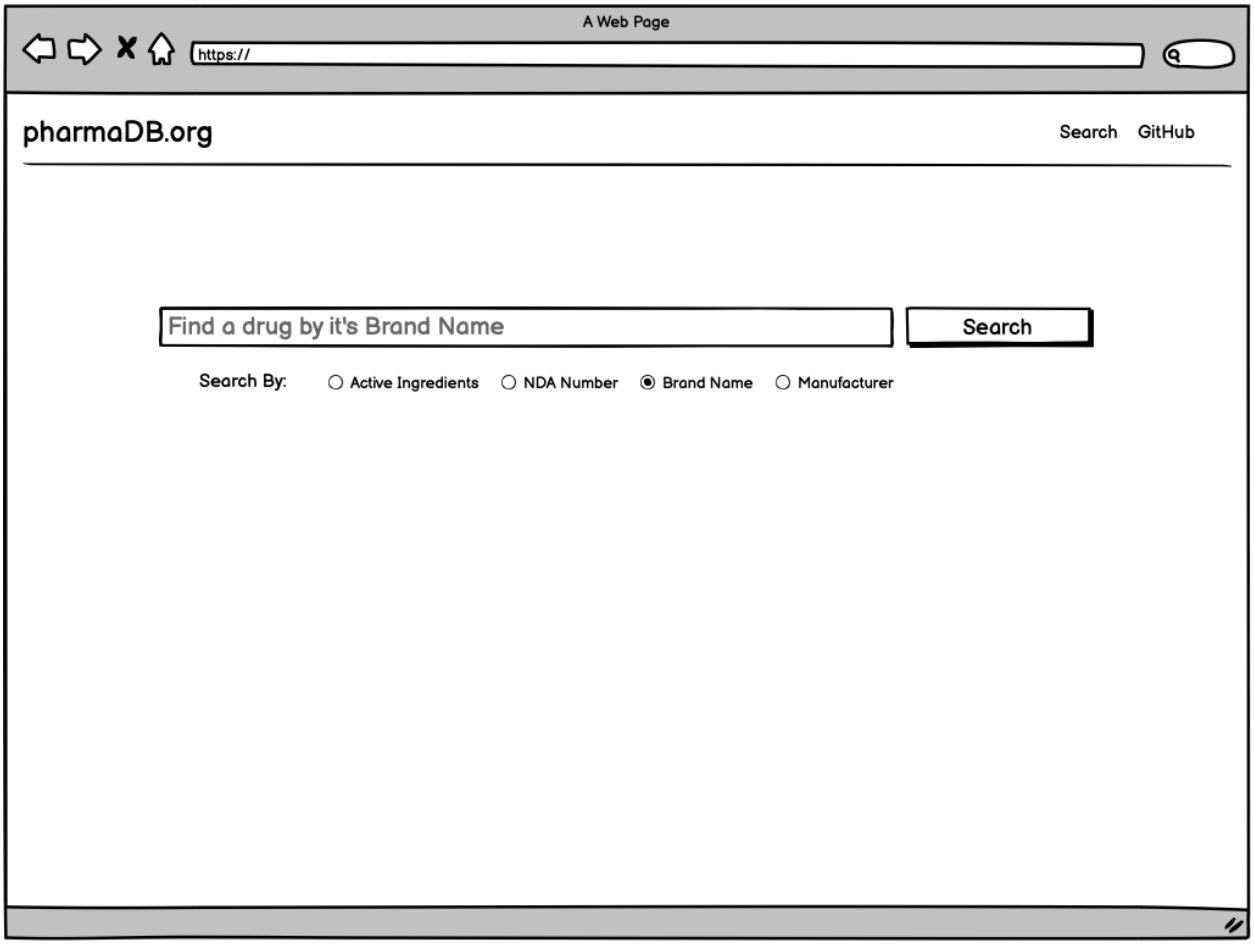


Figure A1. Search View (App Entrypoint) Wireframe. A simple search bar dominates the first page and makes it apparent the user flow begins with a query.

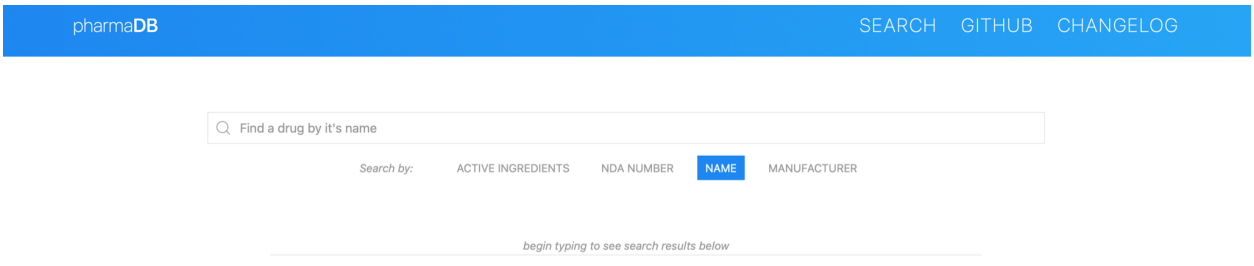
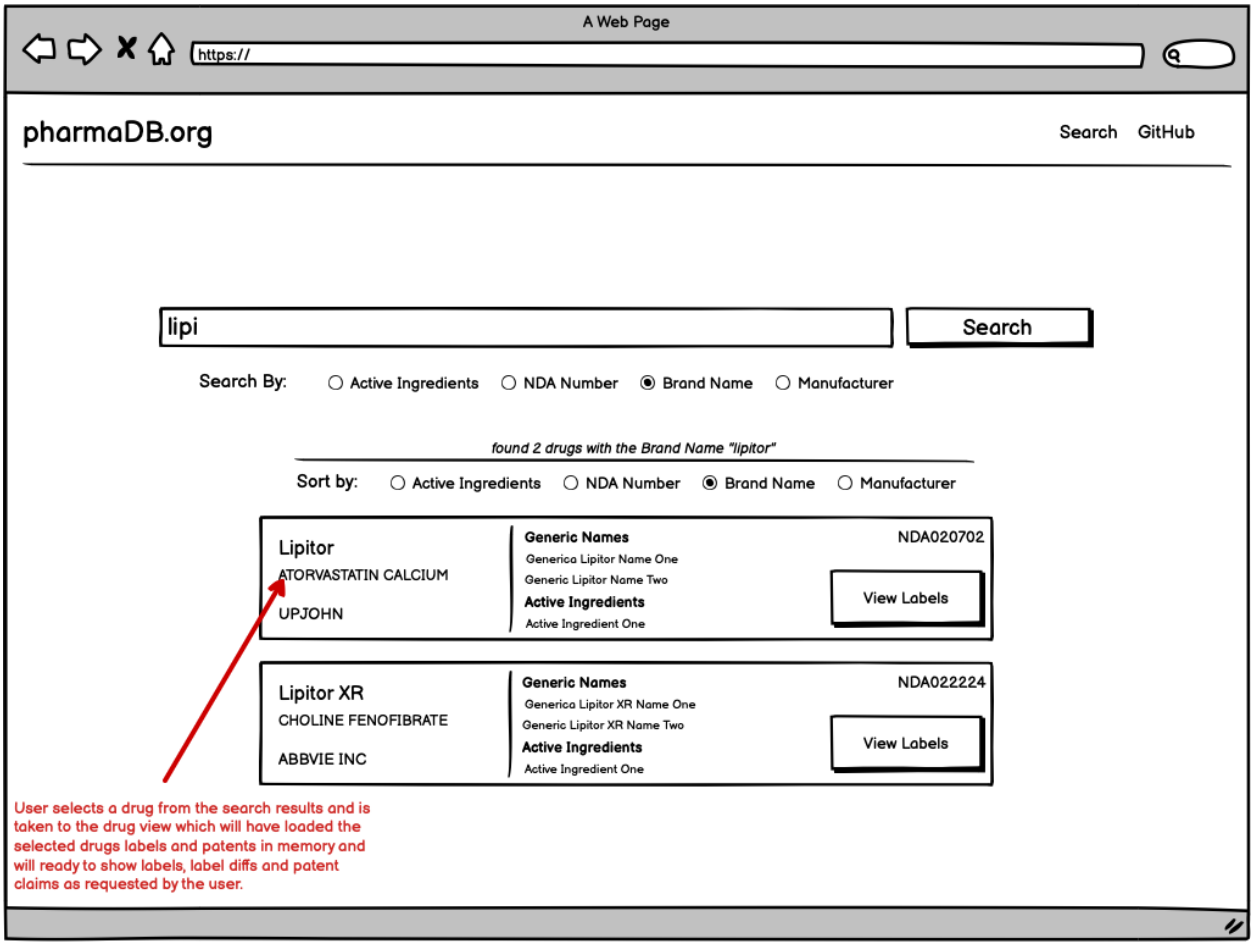


Figure A2. Search View (App Entrypoint). Search View (App Entrypoint). A simple search bar dominates the first page and makes it apparent the user flow begins with a query.



Search View w/ Results

Figure A3. Search View With Results Wireframe. A red arrow points to search results shown after a user searches 'lipi'

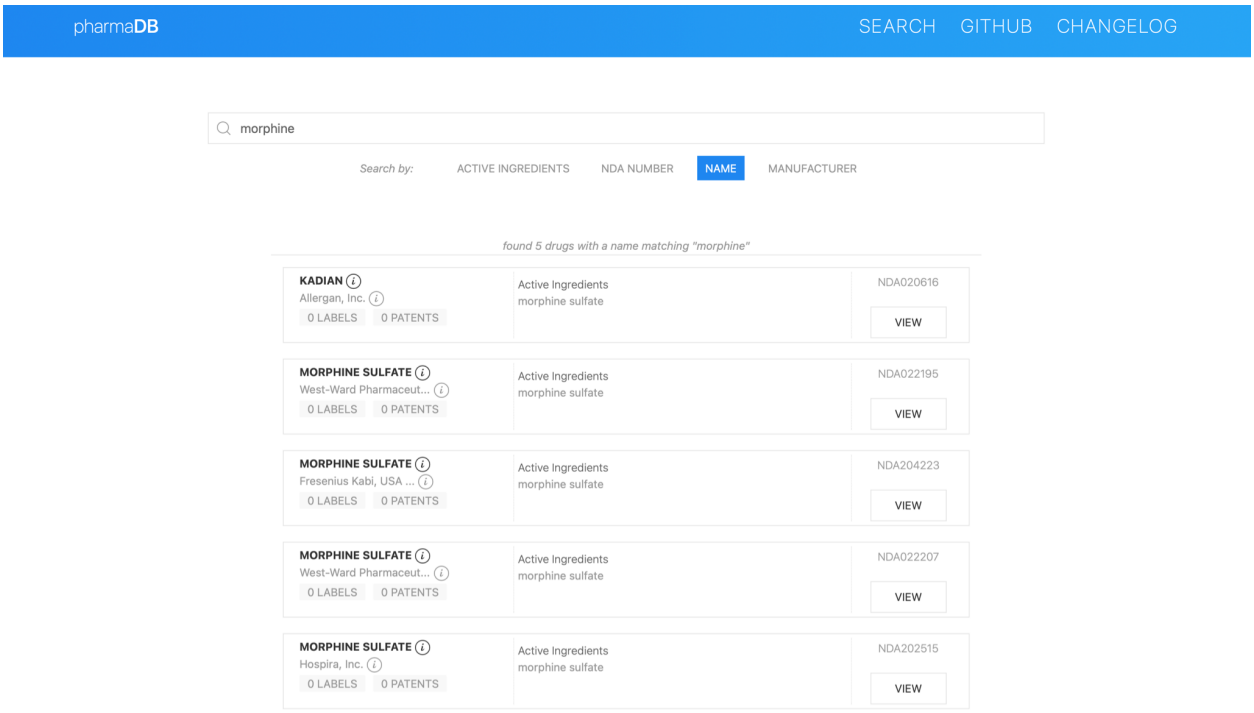


Figure A4. Search View With Results. Results shown on the search screen after a user searches 'morphine'

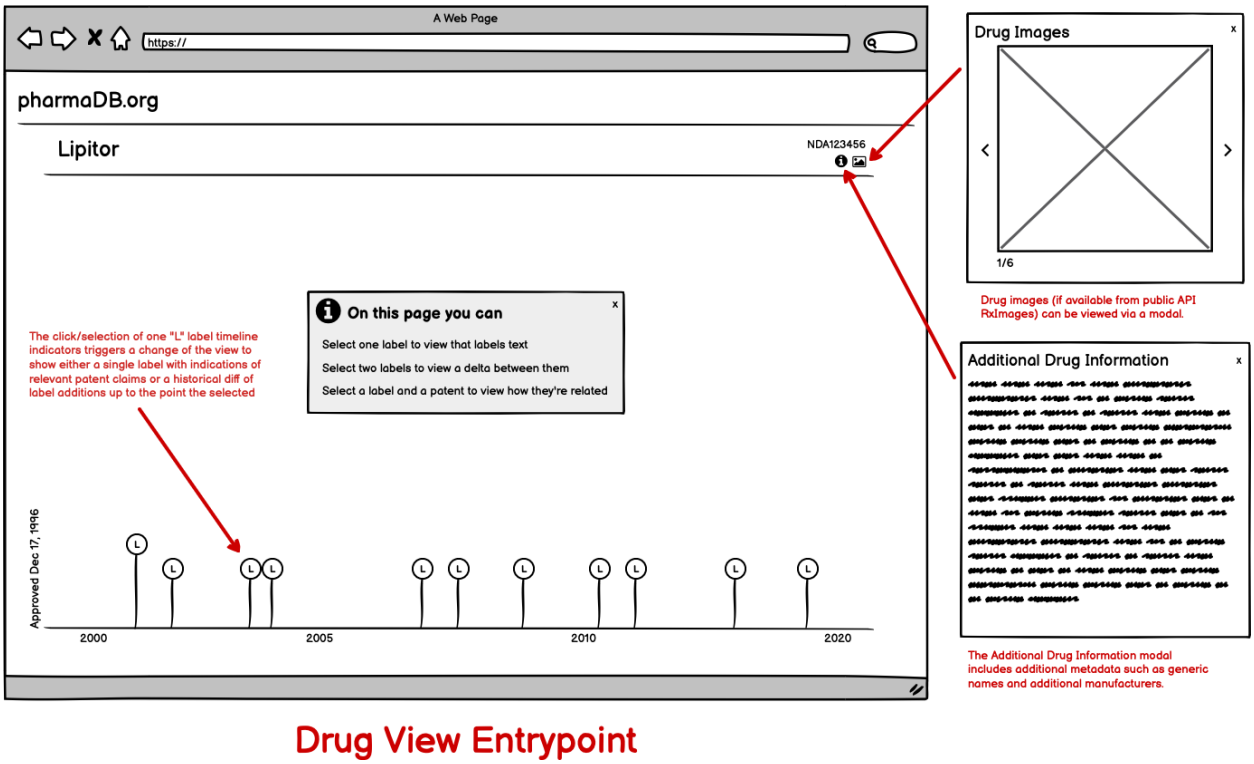
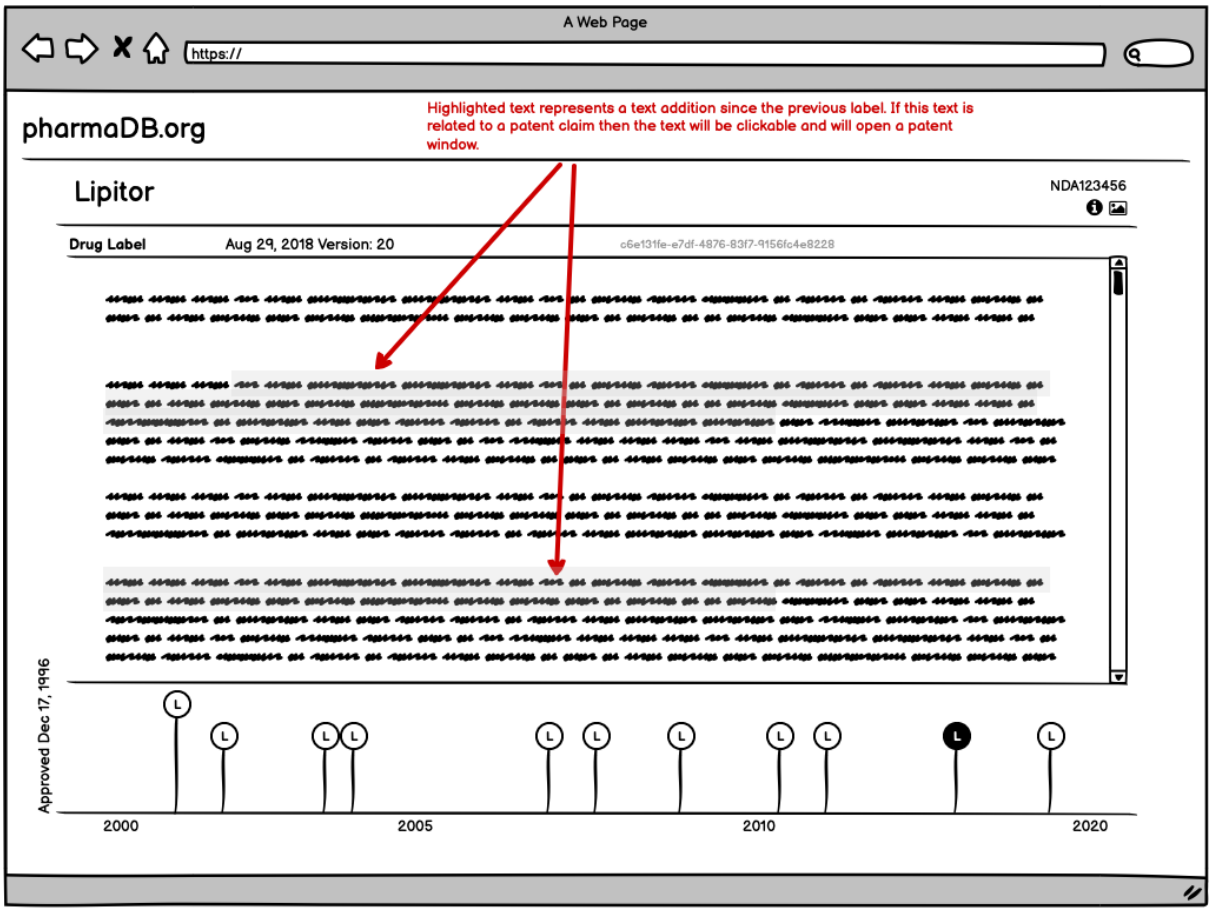


Figure A5. Drug View Entrypoint Wireframe. Red arrows point to the timeline as well as icons which trigger modals that show additional drug metadata

Figure A6. Drug View Entrypoint. Timeline is visible with multicolor indicators representing drug labels.



Drug View Label Diff View Mode

Figure A7. Drug View With Label Diff View Mode Wireframe. Red arrows point to a highlighted text that indicates an addition between the selected label and the previous label. Some of these additions may be related to patent claims.

Figure A8. Drug View With Label Diff View Mode. Highlighted text indicates an addition or deletion between the selected label and the previous label. Some of these additions may be related to patent claims.

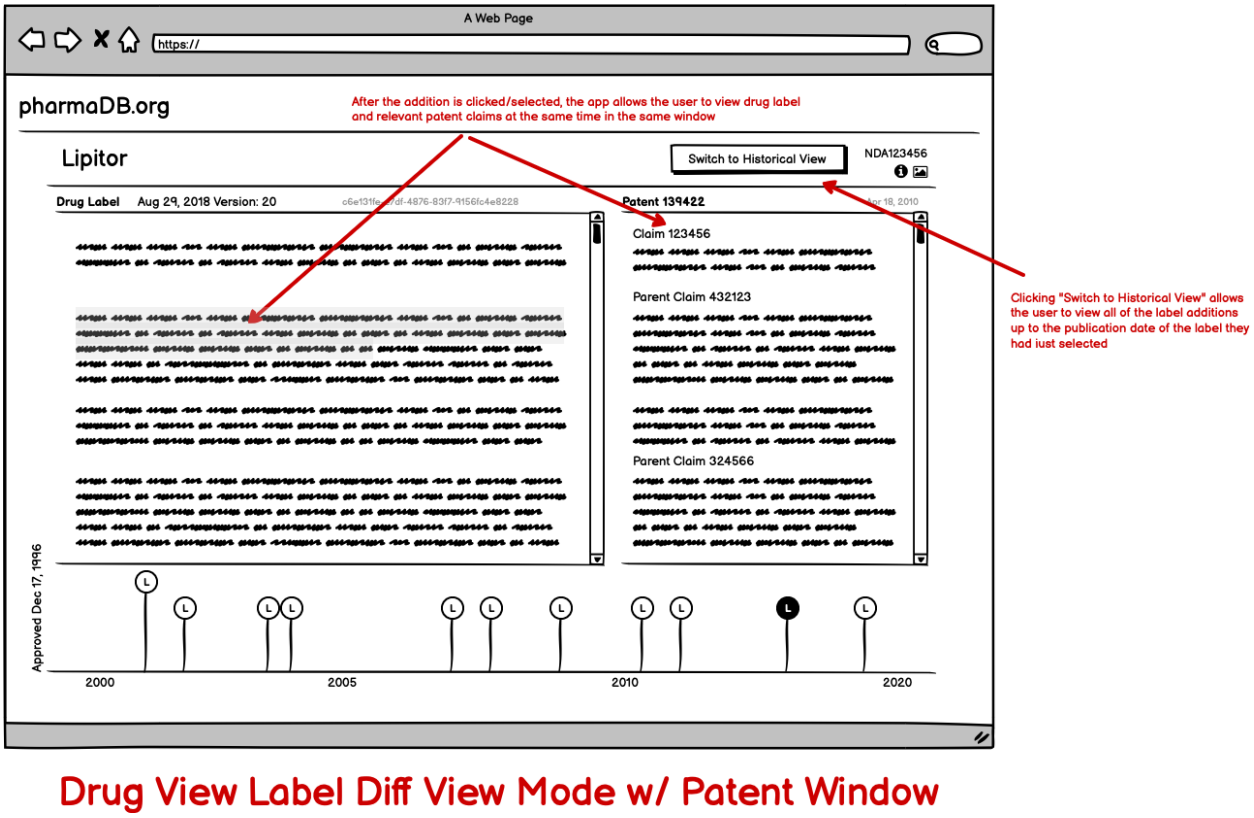


Figure A9. Drug View Label Diff View Mode with Patent Window Wireframe. Red arrows point to identifiers showing the link between a label diffs added text and a patent claim while a third arrow points to a button that lets the user switch to a drug label historical additions viewing mode.

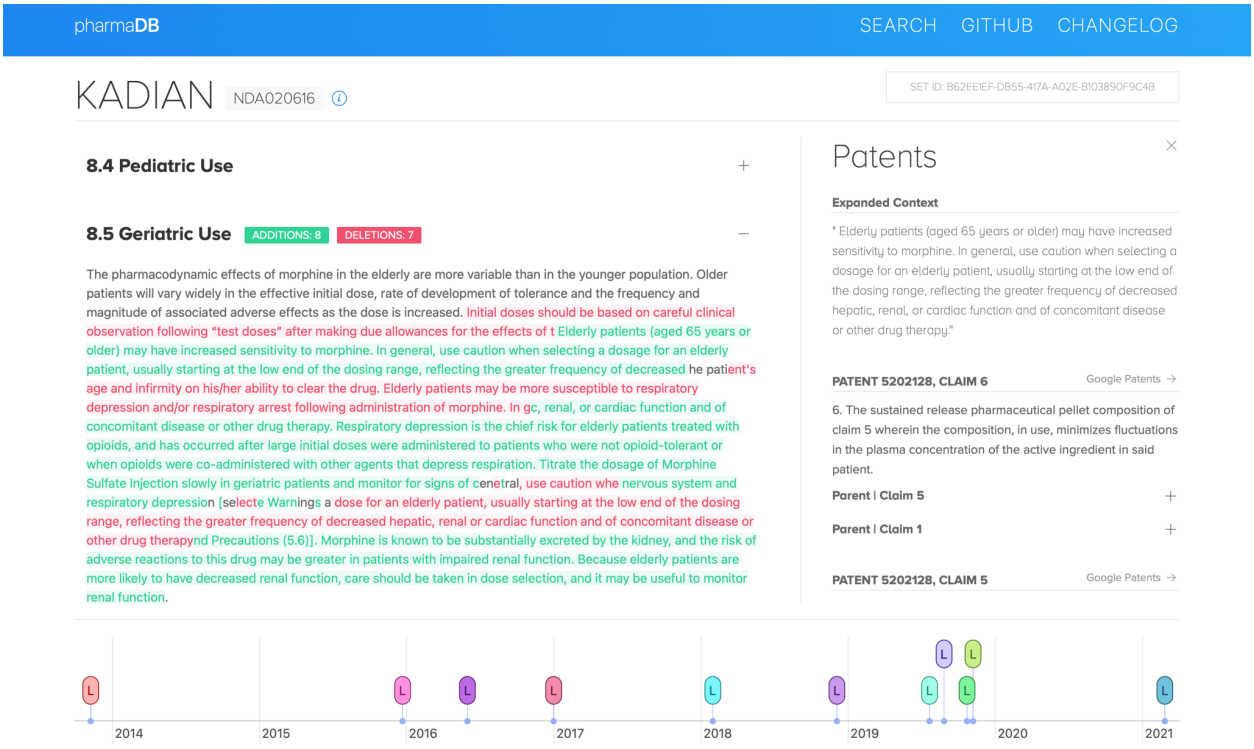
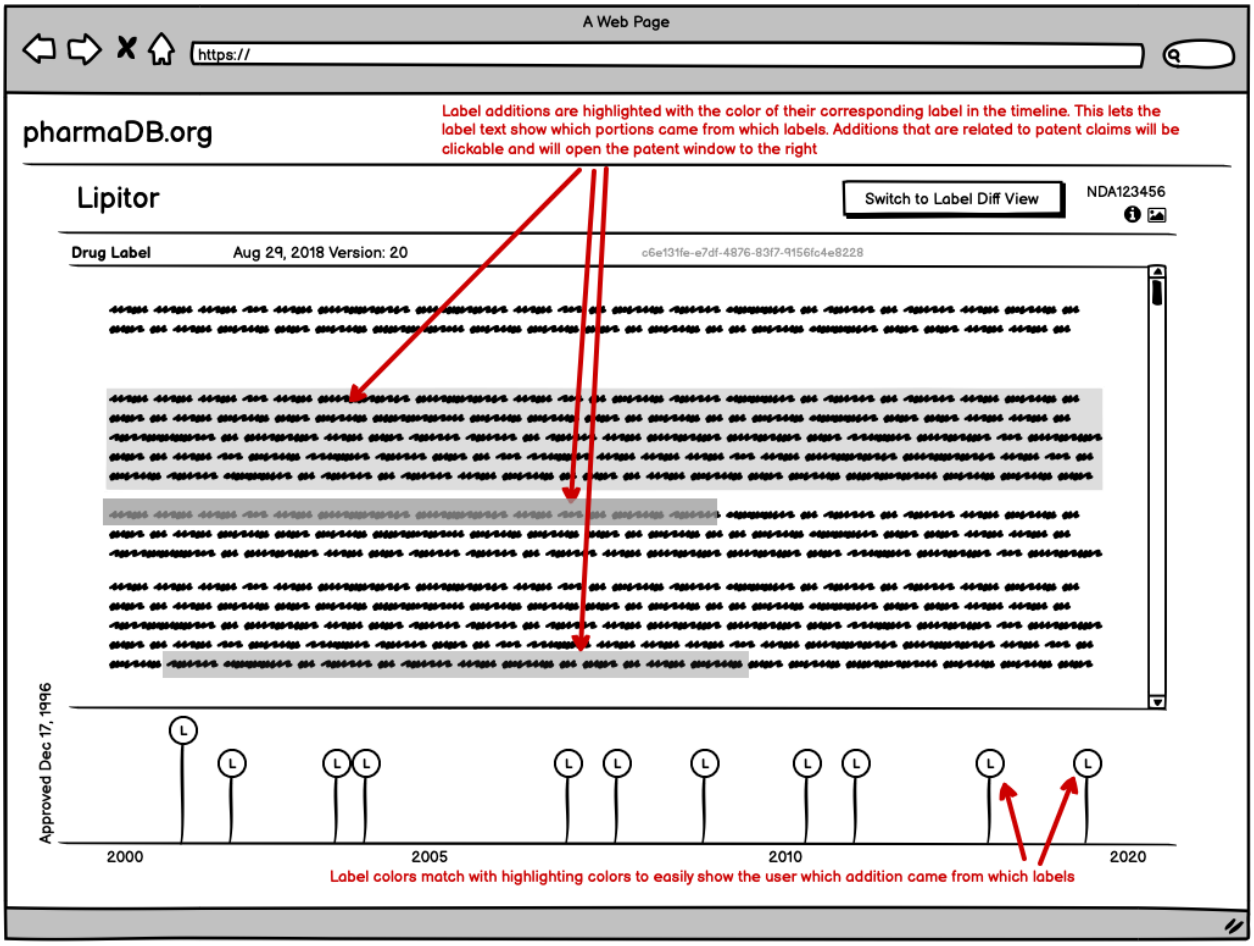


Figure A10. Drug View Label Diff View Mode with Patent Window. Both the label diff and a patent claim associated with one of the diffs additions are shown in the same window.



Drug View Historical Additions Mode

Figure A11. Drug View WIth Historical Additions Wireframe. Label text is highlighted in different colors to represent a text addition from a particular label on the timeline. Red arrows on the timeline point to highlighted addition text as well as label timeline indicators.

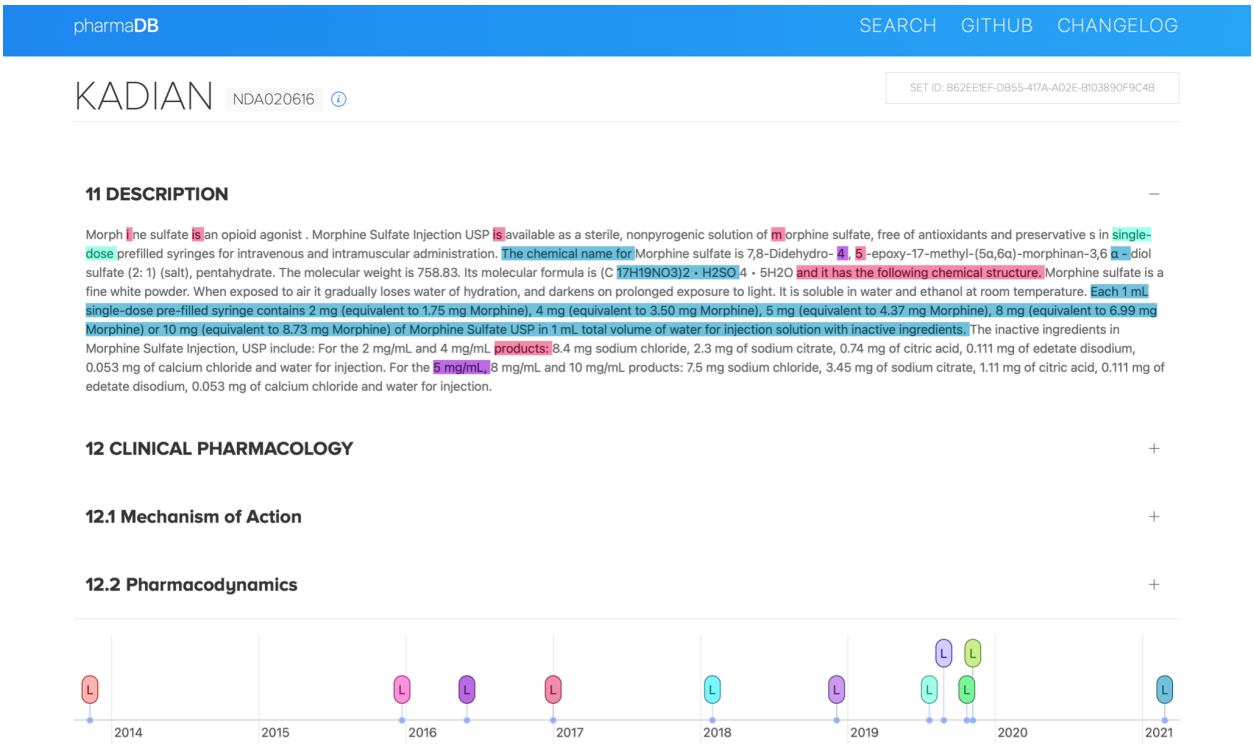


Figure A12. Drug View WIth Historical Additions. Label text additions are represented by highlighting of the color of the label indicator they came from in the timeline.

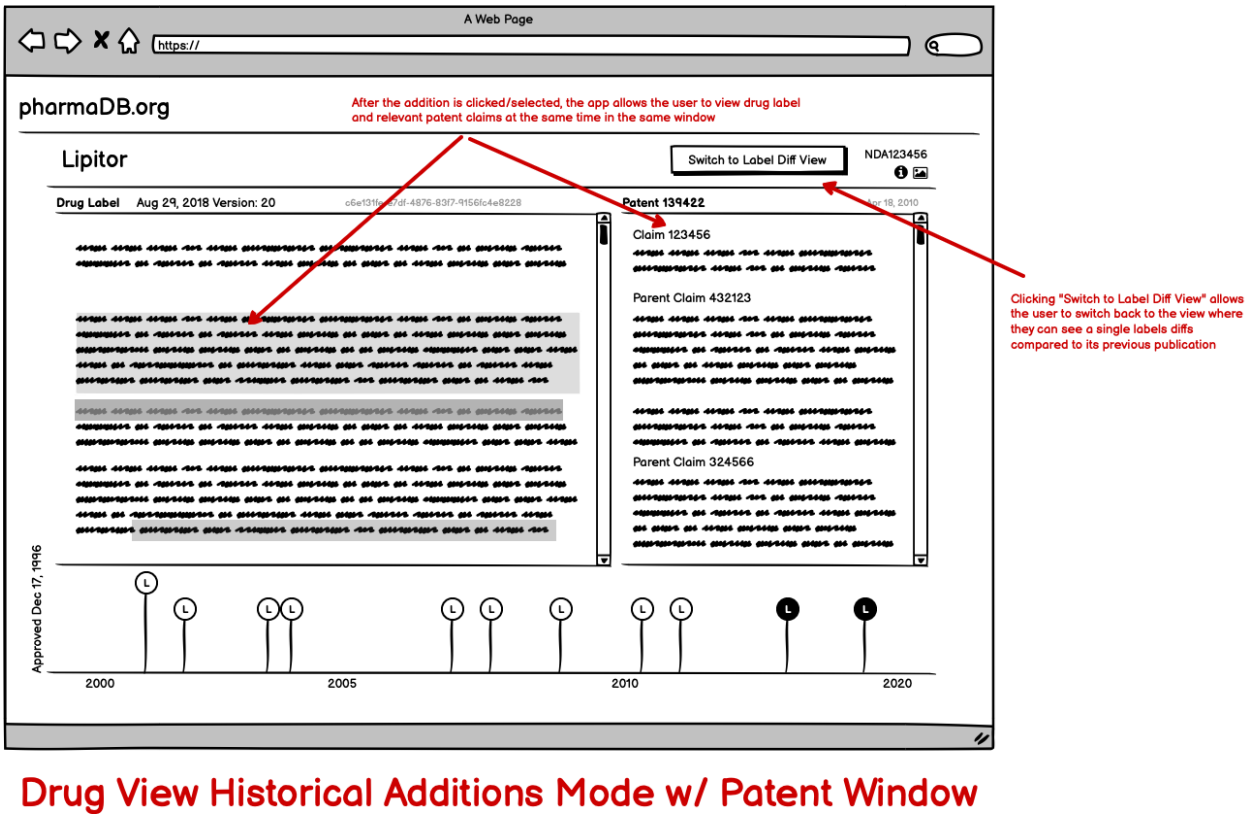


FIgure A13. Drug View Historical Additions Mode with Patent Window Wireframe. Red arrows point to the addition and the additions related patent claims as well as the button that lets the user switch back to the label diff view.

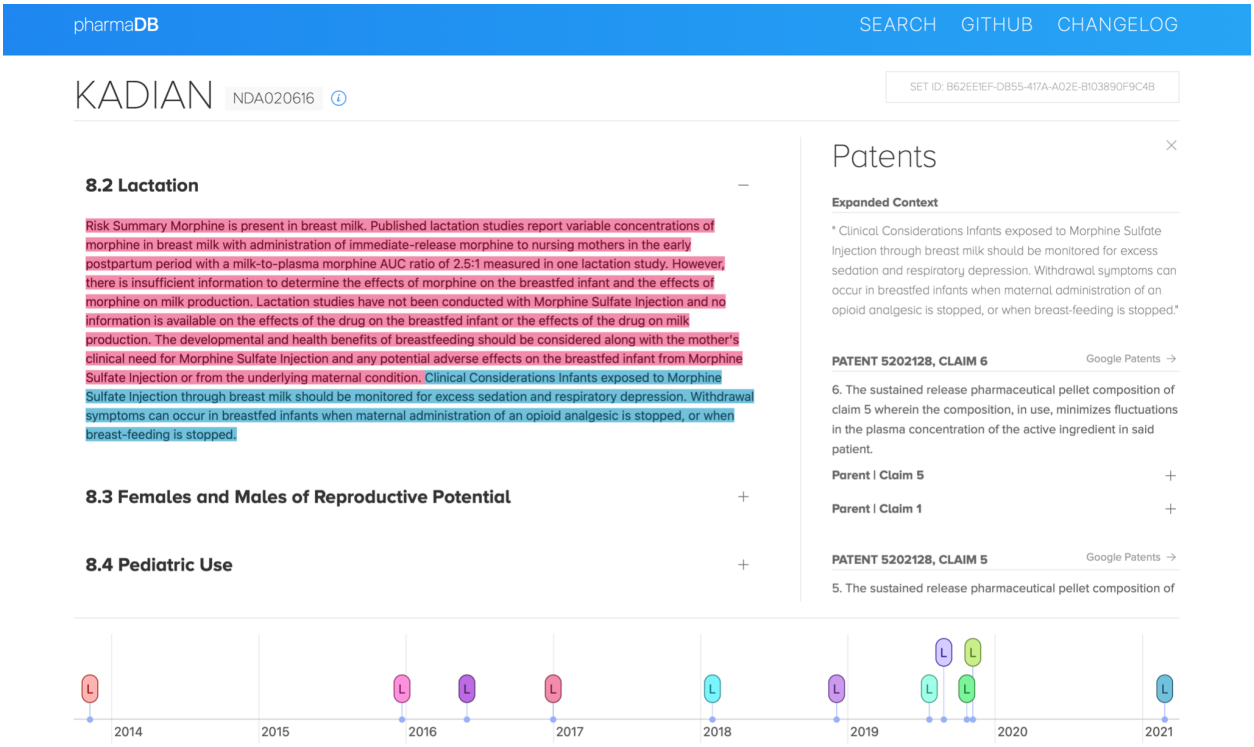


FIgure A14. Drug View Historical Additions Mode with Patent Window. Highlighted additions that are related to patent claims can be clicked on, revealing a panel that shows information about the specific patent claim.

FIgure A14. Web API UML Diagram (https://github.com/pharmaDB/pharmadb_api/blob/main/umlDiagram.png)

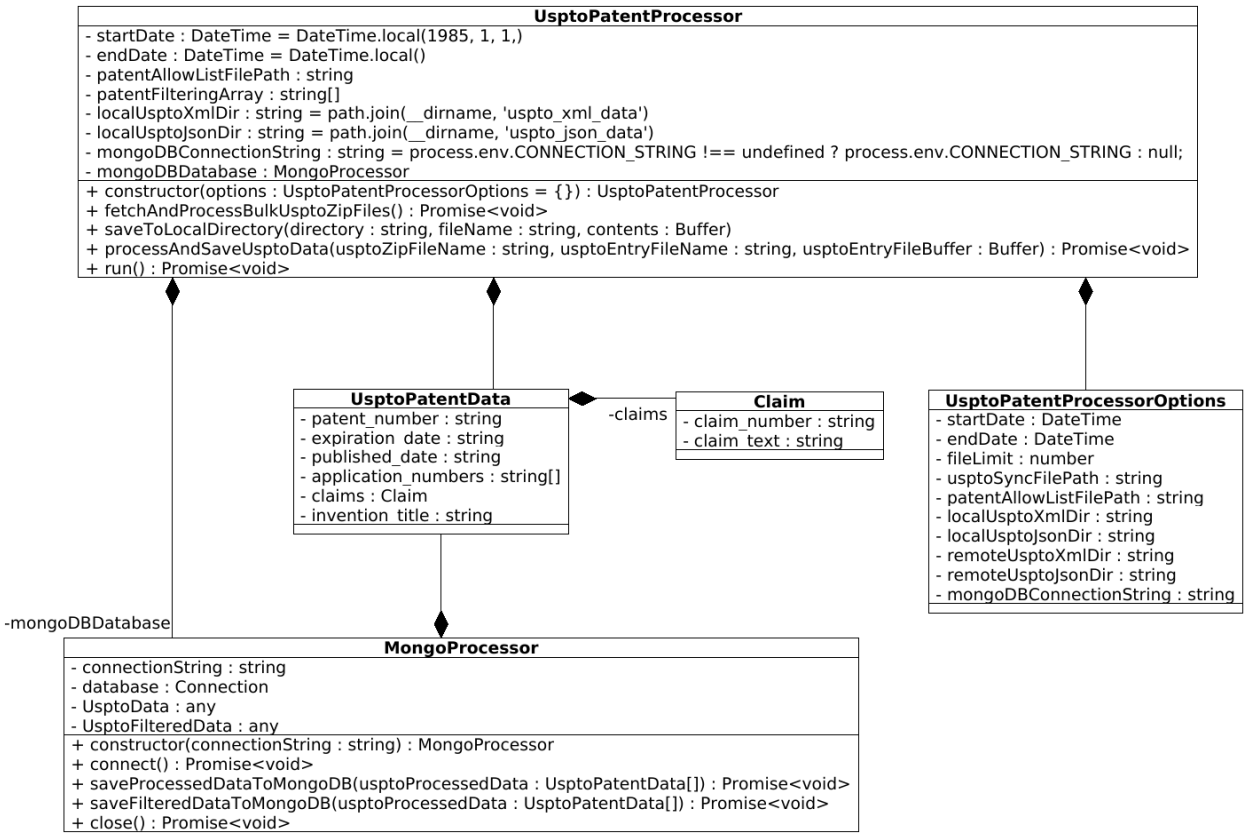


FIgure A15. Patent Data Aggregation UML Diagram

14. Appendix: Working Version and Code

14.1 Introduction

The working version of the code is available at www.pharmadb.org.

Below are the links to various repositories, along with instructions on how to get started on the codebase. Each repository contains instructions on how to run each individual piece of the pharmaDB project locally. Instructions on how to run the whole pharmaDB project on cloud-hosted infrastructure are available in the organization repository at https://github.com/pharmaDB/PharmaDB-README.

14.2 AWS/Terraform

https://github.com/pharmaDB/database_infrastructure

This repository contains the source code for deploying the back-end database infrastructure to the major cloud platform providers. The source code includes terraform configuration code that will deploy 2

Ubuntu 20.04 virtual servers to the AWS cloud (Azure and GCP are available as well with minor modifications to the terraform configuration code), as well as bash scripts that will be run on the deployed virtual servers to install and host a MongoDB database. This repository requires the AWS CLI 2+ and Terraform CLI 0.14+ in order to run the code.

14.3 Web Application

14.3.1 Web Application API

The Web Application API code can be found on GitHub at the below link. In the GitHubs README you'll find directions on how to start the API and what the API endpoints are. The code is extensively commented and so should be easy to understand. One thing to note is that the caching/Memcached feature is still under development.

https://github.com/pharmaDB/pharmadb_api

14.3.2 Web Application UI

The Web Application UI code can be found on GitHub at the below link. You can also find directions on how to start the Web App UI and how to access it via that GitHub repo README. The Web Application, by default, is set to use the https://api.pharmadb.org API. There is no need to have the Web Application API running locally in order to also run the Web Application UI locally. The code is extensively commented and so should be easy to understand. I would actually consider the code to be over-commented, but that was done intentionally so that the code would be easier to pick up for any future development by outside developers.

https://github.com/pharmaDB/capstone_spa

14.4 Data Collection and Processing

The Data Collection and Processing repositories contain scripts that collect all relevant patents, labels, and Orange Book data, as well as populate a MongoDB database. Currently, the MongoDB database consists of patent and label collections along with diffs between the labels, and the collation between each diff to respective patent claims.

14.4.1 Pipeline Orchestration

https://github.com/pharmaDB/etl_pipeline

This repo contains a docker-compose set up for a local MongoDB along with Mongo Express viewer to interact with the DB conveniently. The repo also combines the Patent ETL, Drug Label ETL and ML steps, by including each of the other data related repos (described in sections below) as sub-modules, and adding additional functionality required to collect and process updates in any of the data domains. A main script acts as the orchestrator, running through the sequence of steps required for the pipeline. The

main script can be scheduled as a cron job that runs on a monthly frequency. More information can be found in the repo's README file.

14.4.2 Patent Bulk Data and ETL

https://github.com/pharmaDB/uspto_bulk_file_processor_v3

This repository contains the source code for extracting, transforming, and loading bulk data from the United States Patent and Trademark Office (USPTO) into a local directory, remote directory, and/or a MongoDB database. The source code contains data processors that will convert the different USPTO patent data formats into a single unified format, and can process bulk data files from 1971 and onward. This repository requires NodeJS 10+ and NPM 5+ in order to run the code.

14.4.3 Label Bulk Data and ETL

https://github.com/pharmaDB/dailymed_data_processor

This repo contains the source code for downloading, processing and loading the Drug Label data from DailyMed, to the configured MongoDB instance, under a collection called 'labels'. Detailed instructions for running the code may be found in the repo's README file.

The scripts have been designed such that they can be run in 2 modes:

- **Bulk Data mode –** This is useful to download the entire index from DailyMed, process each version of each drug label, and save to MongoDB those drug labels that have an association to an 'NDA*****' number. This is very time consuming, but does not require any other data to fetch all drug labels that have an association to the Orange Book.
- **ETL Pipeline mode –** In this mode, a list of 'Set IDs' would be supplied in a file and the entire history of labels for these IDs would be run through the processing steps. It is expected that the Set ID list would come from the pipeline step that downloads the monthly incremental data from DailyMed, and would be minimal.

Both these modes are described under Usage Examples in the README.

14.4.4 Diffing, Mapping and Scoring

https://github.com/pharmaDB/scoring_data_processor

This package maps changes in drug label to patent claims using the Orange Book Table's of Exclusivity. This package also includes scripts to generate a csv output of all entries in the labels collection within the MongoDB database and to host that csv output as a zip folder. All instructions for setup, running this package, and running unit tests are included with the main README.md file visible with the link above.

This package first adds all previous and next labels key-value pairs to a label with a common set-id with other labels. It also creates the `nda_to_patent` entry in the label document. It then stores all diffs of `sections` against a previous label having the same set-id for a label with common NDA within

`diff_against_previous_label`. Additions are notes as `1`, subtractions as `-1`, and no changes as `0` in the `text` field of `diff_against_previous_label`.

All additions from `diff_against_previous_label` are collected in `additions`. If the additions are not full sentences, this package will rebuild the sentence around each addition and store this information as `expanded_content` within each `additions` . These additions are then compared against claims using the `sentence_transformer` library. The output of the comparisons are then stored back into the labels underneath the `additions` and additionally next to each `text` addition in `diff_against_previous_label` per the request of the front end developers.

This module is also able to pull new Orange Books zipped files from FDA.gov, and outputs lists of NDA numbers or patent numbers from the collected Orange Books that are currently missing in the database. The Orange Book data is used mainly for the mapping of additions to claims.

This module also includes a feature to export the database as a .csv file. The .csv file is visible in `assets/hosted_folder/`. The hosted_folder can be hosted on the same server running this `scoring_data_processor` package using the `server.py` script.

Finally, this module includes all Colab analysis results in the `analysis/` folder.

14.4.5 Data Analysis

https://github.com/pharmaDB/data_analysis

Scripts and datasets used in the EDA phase of the project as well as some helper scripts for collecting the bulk data may be found in the Data Analysis repository. The README file under ds_notebook/ contains instructions for setting up the Jupyter Notebook, hosted from a docker container, with the necessary dependencies.