*Florida International University*

*School of Computing and Information Sciences*

CIS 4911 - Senior Capstone Project

Software Engineering Focus

Final Deliverable

LegalWise 2.0

<http://legalwise2ui.mybluemix.net/>

**Team Members:**

Fernando Gomez

Yang Zhang

**Product Owner**:

Jaime Borras

**Mentor**:

Jaime Borras

**Instructor**:

Masoud Sadjadi

The MIT License (MIT)

Copyright (c) *2016 <copyright holders>*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

***Abstract***

*LegalWise 2.0 is a web application that facilitates legal firms, lawyers, law associates, and law students to answer legal questions as well as find information about legal cases using artificial intelligence technologies. This application helps save time by filtering legal documents electronically. In this project, we are using modern technologies and cutting edge methodologies to achieve the expected outcome.*

Table of Contents

[Introduction 5](#_Toc450245183)

[Current System 5](#_Toc450245184)

[Purpose of New System 5](#_Toc450245185)

[User Stories 7](#_Toc450245186)

[Implemented User Stories 7](#_Toc450245187)

[Pending User Stories 17](#_Toc450245188)

[Project Plan 19](#_Toc450245189)

[Hardware and Software Resources 19](#_Toc450245190)

[Sprints Plan 21](#_Toc450245191)

[Sprint 1 21](#_Toc450245192)

[Sprint 2 22](#_Toc450245193)

[System Design 30](#_Toc450245194)

[Architectural Patterns 30](#_Toc450245195)

[System and Subsystem Decomposition 32](#_Toc450245196)

[Deployment Diagram 34](#_Toc450245197)

[Design Patterns 35](#_Toc450245198)

[System Validation 36](#_Toc450245199)

[Glossary 42](#_Toc450245200)

[Appendix 43](#_Toc450245201)

[Appendix A - UML Diagrams 43](#_Toc450245202)

[Static UML Diagrams 43](#_Toc450245203)

[Dynamic UML Diagrams 52](#_Toc450245204)

[Appendix B - User Interface Design 67](#_Toc450245205)

[Appendix C - Sprint Review Reports 75](#_Toc450245206)

[Appendix D - Sprint Retrospective Reports 79](#_Toc450245207)

[References 86](#_Toc450245208)

# Introduction

This document presents the design and architecture of LegalWise 2.0. There are five chapters which are divided into subsections. The first chapter is the introduction which describes the current system and the purpose of the new system. The second chapter shows our user stories, including both implemented user stories and pending user stories. The third chapter is the project plan which shows the hardware and software resources and our sprint planning. The fourth chapter describes the system design, including architectural patterns, system and subsystem decomposition, deployment diagram and design patterns. The fifth chapter shows the system validation which are the test cases for the system and subsystems.

## Current System

Our application expands on the current system: LegalWise 1.0. This system was built to address the same problems. It was trained to answer simple general questions. This system allows the user to register, log in, search for anything within the **application domain**: which is a partially wiki-based general indexed domain for answers with percentage of accuracy.

LegalWise is intended for legal firms, or users working in the legal field. The main purpose is to allow legal representatives to get answers for their legal questions and download legal cases. The current system does not facilitate this to the legal industry, therefore, the main task of the new system is to completely shift the focus to the legal field.

## Purpose of New System

Lawyers, law students, and legal associates use previous legal cases’ outcomes to prepare for cases they are currently working on. To find a legal case that relates to a current case, they have to manually read each of the documents and determine whether the document is useful or not. This process is very inefficient and time consuming. Before LegalWise, there was no system that help in the document search process. This is the main problem our application intents to tackle.

LegalWise 2.0 will solve many of the problems detected in the current system. The new system uses artificial intelligence technologies on real legal cases and meaningful training data to answer legal questions. Besides the log in, log out functions, this new system has a lot more to offer.

LegalWise 2.0 supports three types of users. The *regular registered user* is allowed to ask legal questions and get recommended answers. Furthermore, *regular users* will be able to download legal cases related to the questions they ask. On the other hand, *Administrators* are allowed to upload new legal cases for indexing, view users in the system and control user access, and upload new training data. which are question and answer pairs to help improve the system.

# User Stories

This section shows all the user stories that were ellicitated and linked to the LegalWise 2.0 project. These user stories were approved by Mr. Jaime Borras, the owner of this project. The previous system was based in a QA server, hosted in a virtual machine. This introduced a number a maintenance issues, and in this version, the underlying technology had to be changed and most of the user stories had to be recreated. As a result, our user stories are planned as follows:

## Implemented User Stories

**User Story # 848 - Search for a keyword**

As a registered user I would like to search a keyword, so that I could get my search results listed with the keyword highlighted.

Use Case:

This use case let the registered user search a keyword and then see the returned results from the system.

Actors:

Registered user.

Pre-Conditions:

1. Must be a registered user.
2. Must be logged in.

Description:

1. Use case begins when the registered user go to the search page and enter a keyword and click “search.”
2. After the system receives this search request, it parses the request to the Solr server to look for the word from indexed documents.
3. Then the Solr server returns the results to page.
4. Use case ends when the highlighted results display on page.

**User Story # 859 - Ask a general question**

As a registered user I would like to ask a general question so that I could get recommend answers to my question.

Use Case:

This use case let the registered user ask a general question and then get the ranked results returned from the system.

Actors:

Registered user.

Pre-Conditions:

1. Must be a registered user.
2. Must be logged in.

Description:

1. Use case begins when the registered user go to the search page and enter a general question and click “search.”
2. After the system receives this search request, it parses the query to Bluemix Retrieve and Rank Service.
3. Then the Retrieve and Rank Service returns the ranked results to page.
4. Use case ends when the ranked results display on page.

**User Story # 880 - Use legal cases as answers to a legal question**

As a registered user I would like to ask a legal question, so that I could get the most related legal cases as answers to my question from the ranked results by the system.

Use Case:

This use case let the registered user ask a legal question and then get the ranked results returned from the system.

Actors:

Registered user.

Pre-Conditions:

1. Must be a registered user.
2. Must be logged in.

Description:

1. Use case begins when the registered user go to the search page and enter a legal question and click “search.”
2. After the system receives this search request, it parses the query to the updated Bluemix Retrieve and Rank Service.
3. Then the Retrieve and Rank Service returns the ranked results to page.
4. Use case ends when the ranked results display on page.

**User Story # 892 - Train the ranker with updated training data**

As an administrator I would like to collect legal question and answer pairs as the updated training data, so that I could use these data to retrain the ranker to improve the accuracy of the system.

Use Case:

This use case let the administrator to add more training data and retrain the ranker on Retrieve and Rank to improve the accuracy of the system.

Actors:

Administrator.

Pre-Conditions:

1. Must be an administrator.
2. Must be logged in as administrator.

Description:

1. Use case begins when the administrator have the training data ready and issue a train command from the command line window.
2. In less than 5 minutes, the training process for the ranker will be finished with a message showing the ranker’s new ID.
3. Administrator then enter the retrain command.
4. The ranker will be retrained and return a status message to the command line window.
5. Use case ends when the ranker’s status is shown as “ready.”

**User Story # 903 - Ask a legal question**

As a registered user I would like to ask a legal question so that I could have the improved system returned me the ranked answers of the legal cases instead of the whole legal case document as my answers.

Use Case:

This use case let the registered user ask a legal question and then get the head notes part of the legal cases as the ranked results returned from the system.

Actors:

Registered user.

Pre-Conditions:

1. Must be a registered user.
2. Must be logged in.

Description:

1. Use case begins when the registered user go to the search page and enter a legal question and click “search.”
2. After the system receives this search request, it parses the query to the updated Bluemix Retrieve and Rank Service.
3. Then the Retrieve and Rank Service returns the ranked results to page.
4. Use case ends when the ranked results display on page.

**User Story # 915 - Get more accurate answers to legal questions**

As a registered user I would like to ask a legal question, so that the improved system would recommend me with more accurate answers.

Use Case:

This use case let the registered user ask a legal question and get the recommended answers with more accuracy from the system compare to the user story 903.

Actors:

Registered user.

Pre-Conditions:

1. Must be a registered user.
2. Must be logged in.

Description:

1. Use case begins when the registered user go to the search page and enter a legal question and click “search.”
2. After the system receives this search request, it parses the query to the updated Bluemix Retrieve and Rank Service.
3. Then the Retrieve and Rank Service returns the ranked results to page.
4. Use case ends when the ranked results display on page.

**User Story # 867 - Browse registered users**

As an administrator, I would like to see a list of currently registered users.

Use Case:

This use case provides an interface that allows administrators to see all users that are currently registered in the system.

Actors:

1. Administrator.

Pre-Conditions:

1. User must be authenticated.
2. User must have administrative privileges.

Acceptance Criteria:

1. User must see a tab title Users in the main tabs.
2. User must see a list of users after clicking the Users tab.

Description:

1. Use case begins when an administrator clicks on the Users tab and is redirected to the Users page.
2. After the page loads, a request is sent to the server requesting the list of users.
3. The server responds with a list of users.
4. Use case ends when the list of users is shown to the administrator.

**User Story # 857 - Update user status**

As an administrator, I would like to activate, inactivate, lock, unlock currently registered users.

Use Case:

This use case provides an interface so administrators can control the access and permissions of regular users. It allows administrators to grant and revoke access from regular users.

Actors:

1. Administrator.

Pre-Conditions:

1. User must be authenticated.
2. User must have administrative privileges.

Acceptance Criteria:

1. Administrator must be in the Users page.
2. Administrator should see a menu for each user.
3. Administrator should be able to lock/inactivate users using the menu.
4. Administrator should be able to unlock/activate users using the menu.

Description:

1. Use case begins when an administrator clicks one of the following menu options next to an user row: Lock, Unlock, Inactivate, Activate.
2. A request is sent to the server with the user Id and the instructions.
3. The server updates the user and returns a confirmation message.
4. Use case ends when the confirmation message is shown to the administrator.

**User Story # 855 - Browse legal documents**

As an administrator, I would like to view a list of currently available documents.

Use Case:

This use case allows administrators to see all documents currently in the database.

Actors:

1. Administrator.

Pre-Conditions:

1. User must be authenticated.
2. User must have administrative privileges.

Acceptance Criteria:

1. The tab menu must show a link to the Documents page.
2. When the link is clicked, the user must be redirected to the Documents page, and a list of currently available document must be shown.
3. If the amount of documents shown is large enough so it takes a space larger that the available screen area to show all the documents, 50 documents should be loaded, and when the user scrolls down, more documents are loaded and appended to the end of the lists.

Description:

1. Use case begins when an administrator clicks on the Documents tab and navigates to the Documents page.
2. A request is automatically sent to the server requesting documents.
3. The server sends the first 50 legal documents in the database, ordered by their upload date.
4. Use case ends when the documents are shown to the user.

**User Story # 851 - Download legal documents**

As a registered user/administrator I would like to download the original PDF file of a legal document.

Use Case:

The use case allows all users in the system to download a PDF version of a legal document.

Actors:

1. Administrator
2. Regular User

Pre-Conditions:

1. User must be authenticated.
2. User must have either performed a search that returned results from legal documents, or user must be an administrator while in the Documents page.
3. User must click the Download button next to a legal document or a result row.

Acceptance Criteria:

1. User must see a DOWNLOAD link in each search result, legal answer, or legal document row.
2. When this link is clicked, the download of the related legal document must automatically start.
3. If file cannot be built and a document cannot be downloaded, the user must see a detailed error message.

Description:

1. Use case begins when the user click on the download button next to a legal document or search result row.
2. A request is sent to the system with the Id of the document.
3. The systems puts the document bytes in the payload of the response.
4. Use case ends when, depending on the user’s browser, the document starts downloading or is opened in a browser tab.

**User Story # 843 - Search legal cases**

As a regular user/administrator, I would like to search for legal documents using a textual search criteria.

Use Case:

This use case allows all users in the system to search one or more legal case by entering some text criteria.

Actors:

1. Administrator
2. Regular User

Pre-Conditions:

1. User must be authenticated.

Acceptance Criteria:

1. User/Admin should see a search box in the Search page.
2. User/Admin should be able to enter some text into the search box.
3. Upon pressing enter, User/Admin should see the textual matches found, meaning, the total number of results, and a page with ten of those results.
4. Users should be able to differentiate between QA pairs, headnotes, and documents (labels).

Description:

1. Use case begins when a regular user enter some text in the Search box, while in the Search page, and presses ENTER.
2. A search request is sent to the system containing the text entered by the user.
3. The user searches the index and returns a list of results.
4. User case ends when the results are shown to the user.

**User Story # 838 - Upload legal document for indexing and ranking**

As an administrator, I want to be able to upload a legal document into the LegalWise 2.0 database so it is available for search.

Use Case:

This use case provides an interface that allows administrators to upload legal cases into the index, ranker, and database. These documents will be immediately made available for searches.

Actors:

1. Administrator

Pre-Conditions:

1. User must be authenticated.
2. User must have administrative privileges.

Acceptance Criteria:

1. Administrators should see a link to the Documents page in the global tabs.
2. In the Documents page, administrators should be able to drag and drop PDF files.
3. Administrators should receive a confirmation message when the upload process is finished.
4. Regular users and administrator should now be able to search and download the uploaded documents.

Description:

1. Use case begins when an administrator navigates to the Documents page.
2. User drops a PDF file into the labelled section.
3. The system notifies the user when the upload process is finished.
4. The use case ends when the confirmation message is shown to the user.

**User Story # 910 - Submit QA Pairs for indexing**

As an administrator, I would like to be able to add manually created question-answer pairs to the indexer and ranker, so they can be included in search results.

Use Case:

This use case provides an interface for Administrators to add question-answer pairs, so they are processed and made available for future searches.

Actors:

1. Administrator

Pre-Conditions:

1. User must be authenticated.
2. User must have administrative privileges.

Acceptance Criteria:

1. Administrator should see a link to the Questions page in the global tabs.
2. In the Questions page, administrator should be able to enter as much as 200 QA pairs.
3. Administrator should be able to add/remove/edit QA pairs before submitting.
4. Administrator should be able to submit the QA pairs, which should be properly included in the index and ranker.
5. Regular users and administrators should now be able to ask any of the added questions and see the answer in the results.

Description

1. Use case begins when an administrator navigates to the Questions page.
2. User adds one or more question using the form provided, and clicks Submit.
3. A click on the Submit button sends a request to the server containing all questions added by the user.
4. The server receives and processes the request and sends a confirmation message.
5. The use case ends when the confirmation message is shown to the user.

## Pending User Stories

**User Story # 850 - Choose frequently asked questions**

As a registered user I would like to choose from the top asked questions list so that I could see the answers easily.

**User Story # 856 - View/Edit User Profile**

As a registered user, I would like to view/edit my profile.

**User Story # 858 - Make a payment**

As a registered user, I would like to make a payment to the system so that I could keep my subscription active.

**User Story # 849 - Reset password**

As a registered user I would like to reset my password once I lost it, so that I could log in again.

# 

# Project Plan

This section describes our project plan, including the hardware and software resources, as well as sprint planning.

The initial plan was to continue working on top of LegalWise 1.0. But after sprint 1, we failed to fix the errors that were left in the first version. There were issues with the virtual machine, used for the QA server, and the application hosted in BlueMix was not responding. These were very critical problems that would have taken us three full sprints to solve. Our decision was to completely redesign the application.

Team members, in compliance with the project owner, decided to use Java as the language for the source code. The virtual machine concept was removed, and the application was decided to be entirely hosted on the BlueMix PaaS server.

To attack the project at hand, we divided the workload into two main parts. QA, as in question-answer handling, and UI/UX/Admin, which is the layer in charge of the user interface/user experience and administrative tasks, in terms of development. As there were two resources available, each resource was assigned a part.

## Hardware and Software Resources

The hardware and software resources used for LegalWise 2.0 are listed as follows:

Hardware

* No hardware resources are part of this solutions, as it is fully hosted in the cloud.

Software

* **Java.** The system was developed using Java version 1.7.
* **Apache Maven**. This is the building profile of our system.
* **HTML, CSS, Javascript**. Front end typical components. This application contains a considerable deal of front end logic.
* **Python**. Used to convert scripts into Watson’s (IBM) expected format, initial indexing and training for Retrieve and Rank.
* **IBM Bluemix cloud server**. The full application is hosted in BlueMix, which is a Platform-as-as-Service provider from *IBM*.
* **Watson Retrieve and Rank Service**. This service, build on top of Solr, aids in the indexing process.
* **Watson Document Conversion Service**. This service aids in the document conversion service.
* **Eclipse EE.** IDE used for the whole development process.
* **ElephantSQL (PostgreSQL)**. The persistent data uses PostgreSQL technologies.

## Sprints Plan

### Sprint 1

(01/16/2016 - 01/29/2016)

**User Story # 839 - Learn and install last version**

***Tasks***

* Learn Node.js structure and characteristics.
* Install requirements.

***Acceptance Criteria***

* None.

***Modeling***

No diagram available for this user story.

**User Story # 836 - Fix test failures in version 1**

***Tasks***

* Read the code, documentation, videos, specifications, and feature list of version 1.0
* Recursively, attempt to fix the code
* Re-run the installation again

***Acceptance Criteria***

* LegalWise 1.0 should be running in a local environment as well as in the BlueMix server.

***Modeling***

No diagram available for this user story.

### Sprint 2

(01/30/2016 - 02/12/2016)

**User Story # 848 - Search for a keyword**

***Tasks***

* Set up Solr server.
* Upload file to Solr for indexing.
* Write an HTML page for searching.

***Acceptance Criteria***

* The registered users should be able to search for a keyword.
* The system should return part of the document indexed with the keyword.

***Modeling***

* Figure Y-1
* Figure Y-7

**User Story # 842 - Redesign UI**

***Tasks***

* Add markup for new design (CSS, HTML).
* Show design to project and obtain approval from project owner.
* Design new Icon for app.
* Design new look and feel for login page.

***Acceptance Criteria***

* Design, look and feel, and flow must be approved by project owner
* The user must be logged in.

***Modeling***

* See Appendix B.

**User Story # 838 - Upload documents into database**

***Tasks***

* Create Documents page.
* Create script for drag and drop.
* Create Search Page.
* Create table structure in QA database.
* Process and submit dragged elements.

***Acceptance Criteria***

* Administrators should be able to upload documents into the database.

***Modeling***

* Figure F-4
* Figure F-12

***Sprint 3***

(02/13/2016 - 02/26/2016)

**User Story # 859 - Ask a general question**

***Tasks***

* Setup Retrieve and Rank Service.
* Follow the documentation to prepare for Retrieve and Rank.
* Index sample data via Retrieve and Rank.
* Train the ranker and retrieve answers.
* Retrank the results.

***Acceptance Criteria***

* The registered users should be able to ask a general question within the index and training data domain.
* The system should return with ranking results to this question.
* This first result should be the most relevant answer to this question.

***Modeling***

* Figure Y-2
* Figure Y-8

**User Story # 857 - Update User Status**

***Tasks***

* Create documentation.
* Add frontend support for lock/unlock, activate/inactivate.
* Login should fail for locked users.
* Add backend functionality to lock/unlock and activate/inactivate and user.

***Acceptance Criteria***

* Administrator should be able to lock/unlock and intivate/deactivate user.
* After lock/inactivate, target user should not be able to login.
* After unlock/activate, target user should be able to login.

***Modeling***

* Figure F-2
* Figure F-10

U**ser Story # 867 - View registered users**

***Tasks***

* Create documentation.
* Add frontend support to show a list of users.
* Add backend support to obtain list of users.
* Create Users page.

***Acceptance Criteria***

* Administrators should be able to see a list of currently registered users when the Users page loads.

***Modeling***

* Figure F-6
* Figure F-13

***Sprint 4***

(02/27/2016 - 03/11/2016)

**User Story # 892 - Train the ranker with updated training data**

***Tasks***

* Collect legal questions and answers pairs.
* Retrain the Ranker

***Acceptance Criteria***

* The updated training data should include at least 49 question and answer pairs with weights.
* The Retrieve and Rank should be trained with the updated training data.
* The status of the ranker should be “Ready.”

***Modeling***

* Figure Y-3
* Figure Y-9

**User Story # 880 - Ask a Legal question**

***Tasks***

* Try to see if Retrieve and Rank works with PDF for indexing.
* Working on converting PDF to JSON.
* Manually generate training data.
* Train the ranker.
* Testing

***Acceptance Criteria***

* The registered users should be able to ask a legal question within the indexed and trained legal domain.
* The search result should be the legal case document related to this question.
* The first search result should be the most related answers to this question.

***Modeling***

* Figure Y-4
* Figure Y-10

**User Story # 855 - View uploaded documents**

***Tasks***

* Create JSON response with retrieved documents.
* Create database access support to retrieve documents.
* Create service to fulfill a list-documents request.

***Acceptance Criteria***

* The tab menu must show a link to the Documents page.
* When the link is clicked, the user must be redirected to the Documents page, and a list of currently available document must be shown.

***Modeling***

* **Figure F-11**
* **Figure F-1**

**User Story # 851 - Download original PDF**

***Tasks***

* Add proper HREF to Download link.
* Format response for file download.
* Create method in connector to retrieve file bytes and info by fileId.
* Create service to fulfill download requests.

***Acceptance Criteria***

* User must see a DOWNLOAD link in each search result, legal answer, or legal document row.
* When this link is clicked, the download of the related legal document must automatically start.

***Modeling***

* Figure F-1
* Figure F-12

***Sprint 5***

(03/19/2016 - 04/01/2016)

**User Story # 903 - Use head notes as legal answers**

***Tasks***

* Python script to convert PDF cases to txt files
* Extract the head notes of legal PDF
* Manually collect training data.
* Modify the format of the training data.
* Re-index Solr with new JSON file
* Retrain the ranker.
* Testing

***Acceptance Criteria***

* The answer to the legal question should be the head notes part of the legal case document instead of the whole document.
* The first answer should be the most related one to this legal question.

***Modeling***

* Figure 5
* Figure 11

**User Story # 843 - Search document**

***Tasks***

* Process search results.
* Create Search service.
* Integrate and consume Solr indexed documents.
* Create search listen route.
* Create proposal about Bluemix addon integration.

***Acceptance Criteria***

* After typing text in the search box, user should see a list of matching results, or a message indicating that no results were found.

***Modeling***

* Figure F-14
* Figure F-5

***Sprint 6***

(04/02/2016 - 04/15/2016)

**User Story # 915 - Get more accurate answers to legal questions**

***Tasks***

* Manually create more training data by reading cases.
* Retrain the Ranker.
* Testing.

***Acceptance Criteria***

* The training data should be more than 49 legal question and answer pairs.
* The ranker should be retrained with the updated training data.
* The search result might be more for a single legal question.

***Modeling***

* Figure Y-6
* Figure Y-12

**User Story # 843 - Add QA pairs to index and ranker**

***Tasks***

* Add method to RetrieveAndRankWrapper to submit a list of QA pairs to the Retrieve and Rank service.
* Create dispatcher service to add QA pairs.
* Create Questions page.

***Acceptance Criteria***

* Administrators should see a link to the Questions page in the global tabs.
* In the Questions page, administrator should be able to enter as much as 200 QA pairs.
* Administrators should be able to submit the QA pairs, which should be properly included in the index and ranker.
* Regular users and administrator should now be able to ask any of the added questions and see the answer in the results.

***Modeling***

* Figure F-15
* Figure F-3

# 

# 

# System Design

This section shows the system design for the LegalWise 2.0 application, including the architectural patterns, system and subsystem decomposition, deployment diagram, and design patterns.

## Architectural Patterns

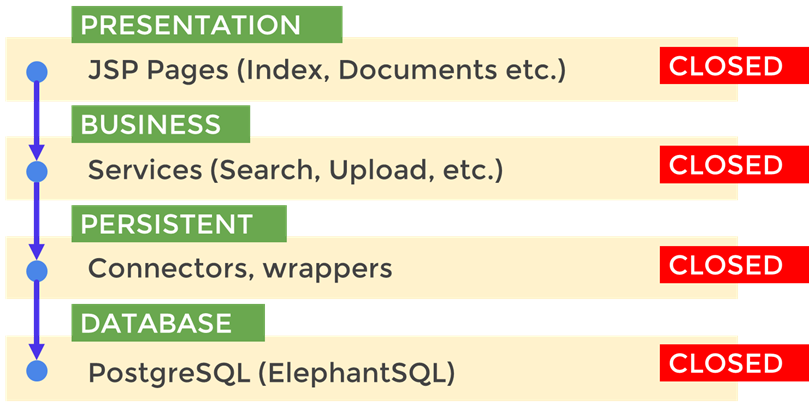


Figure 4.1 - System Architecture

As shown above, LegalWise 2.0 was built using a Layered Architecture, strictly broken down into 4 layers.

* The **Presentation** layer contains all UI/UX components, mainly JSP pages.
* The **Business** layer includes the logic of all services offered by the application.
* The **Persistent** layer contains data access components such as connectors and wrappers.
* The **Database** layer encapsulates the PostgresSQL database that store all persistent information of the application.

Note that all layers in the system are closed, meaning that a layer can only access the layer right next to it. For example, a component in the Presentation layer cannot access a feature of the Persistent layer directly, it has to be through the Business layer.

As imposed by this pattern, the JSP pages in the application are not linked to a controller. They access the server through services. The UI/UX is totally separated from any server logic, as no rendering is done using Java variables, and server logic has no information on the state of the Presentation layer. This pattern was chosen to create a robust, independent backend logic that can also serve as a REST api.

## System and Subsystem Decomposition

Package.png

Figure 4.2 - Package diagram

The above diagram shows the system and subsystem decomposition of our project. A detailed description of each subsystem is as follows:

**Servlets [package]**

This package contain the Java servlets that will listen for requests. As of now, our system contains only one dispatcher servlet, which handles all incoming requests. Both POST and GET requests will be analyzed by this dispatcher, and handed to the specialized subsystem accordingly.

**Dispatch [package]**

This package contains members that are more specialized in handling specific requests. The objects in this package are described as follows:

* **File Dispatcher**: handles requests asking for public files such as CSS and Javascript files, as well as images and related features.
* **Page Dispatcher**: handles requests asking for JSP pages. As all incoming traffic is handled by the main dispatcher, this object is in charge for preparing and returning web pages.
* **Service Dispatcher**: handles the fulfilment of all application services. Output and input of data, as well as business logic is handled specifically through web services. This object is in charge of selecting and executing the right service.

**Security [package]**

This package handles all security features such as authentication.

**Services [package]**

This package contains all services provided by the application. Each service performs a specific task. These services are invoked by the Service Dispatcher. They perform a specified task on a request, and generate a response. The services available as of now are listed as follows:

1. **Download**: fulfills a request to download a document.
2. **List Documents**: returns a list of documents currently available in the system.
3. **List Users**: returns a list of users currently active in the system.
4. **Search**: fulfill a search performed by a user. This is the main feature of our application. This is the object in charge of returning answers to legal question and documents that match a textual criteria.
5. **Update User**: fulfills a lock/unlock-inactivate/activate request.
6. **QA Addition**: fulfills a request to add QA pairs to the index.

**Data [package]**

This package contains all members that in some way communicate directly with the database. As of now, there is only one member: Connector, which is in charge of all DML operations, as well as retrieving information from the database. The information retrieved by this member is wrapped and in no way allows a trace back to the database.

**Wrapper [package]**

This package contain objects that represent tables in the database. These are the members used by connectors to return wrapped information.

**QA [package]**

This package contain objects that handle document conversion and indexing. The objects in this package are also in charge of invoking external systems such as **Retrieve and Rank** and **Document Conversion**.

## Deployment Diagram

System deployment.png

Figure 4.3 - Deployment Diagram

LegalWise 2.0 is a web application built and hosted on IBM Bluemix™ (PaaS) and written in Java, using Liberty profile as runtime. Apache Maven is used as building profile. The Watson Retrieve and Rank and Document Conversion services are used to aid in the indexing and document conversion processes. ElephantSQL is used as the database hosting service, which leverages the PostgesSQL DBMS as a service. All this services are integrated in the Bluemix platform.

## Design Patterns

LegalWise 2.0 uses the following design patterns.

**Factory Method** - creational

This pattern is used to handle the instantiation of services. The Service Dispatcher defers to a helper method the decision about which service to instantiate. As they all implement the same interface, the interface definition is the one used while invoking methods.

**Singleton** - creational

The object that handles the system global settings needs no more that one instance as settings are the same and should be synchronized. This is a singleton class, with only one instance across the board.

**Front controller** - structural

As ours is a web application, we have only one entry point, which is the Dispatcher servlet, a centralized point for handling all kinds of requests.

**Mediator** - behavioral

The interaction of most object in our system is controlled by a moderator. This keeps the objects loosely coupled as the interaction between them is minimal. As an example, service dispatchers can only reference services; they cannot directly access connectors or wrappers.

# 

# 

# System Validation

This section shows the system, subsystem and the unit test cases of each related user story. The subsystem test cases are for the question and answer (QA) subsystem to verify if the Retrieve and Rank was indexed, trained properly, and if it works well communicating with front end page. The unit test cases are for the front end part of our system.

**User Story # 848 - Search for a keyword**

System Tests

* SysTest\_1 - Search a unique text to see if returns 1 result highlighted.
* SysTest\_2 - Search a text that may occur multiple times in all documents to see if results are highlighted.
* SysTest\_3 - Search a text which is not in the documents to see if nothing found.

Subsystem Tests

* QATest\_1 - Check if batch index pdf files works.
* QATest\_2 - Check if highlight input text works.

**User Story # 859 - Ask a general question**

System Tests

* SysTest\_4 - Ask a related question to check if related answers returned with ranking.
* SysTest\_5 - Ask the related question in another way to check if the same answers will return.
* SysTest\_6 - Ask an unrelated question to check if the system returns “0” result.

Subsystem Tests

* QATest\_4 - Check if the script could train the ranker.
* QATest\_5 - Check if the Retrieve and Rank could retrieve related answers.
* QATest\_6 - Check if input a non related question would result in retrieving no answers.

**User Story # 880 - Ask a legal question**

System Tests

* SysTest\_7 - Ask a related legal question to check if related answers returned with ranking.
* SysTest\_8 - Ask an unrelated question to check if returns “0.”

Subsystem Tests

* QATest\_6 - Check if answers a legal question.
* QATest\_7 - Check if no answer for unrelated question.

**User Story # 892 - Ask a legal question**

System Tests

* SysTest\_9 - Ask a related legal question to check if related answers returned with ranking.
* SysTest\_10 - Ask an unrelated question to check if returns “0.”

Subsystem Tests

* QATest\_8 - Check if the ranker is trained with updated data.
* QATest\_9 - Check if the ranker is re-trained and ready for searching.

**User Story # 903 - Use head notes as legal answers**

System Tests

* SysTest\_11 - Ask a legal question to the improved QA system to see if the answers are from the head notes.
* SysTest\_12 - Ask a not related question to the improved QA system to check if it returns nothing.

Subsystem Tests

* QATest\_10 - Check if the improved QA system returns the head notes as the answer to a related legal question.
* QATest\_11 - Check if the improved QA system works well for unrelated questions.

**User Story # 915 - Get more accurate answers to legal questions**

System Tests

* SysTest\_13 - Ask the same legal question as of user story 903 to the more accurate system to see if there are more results.
* SysTest\_14 - Ask another legal question “What is the right of appeal?” to this improved system.
* SysTest\_15 - Ask a not related question to improved QA system.

Subsystem Tests

* QATest\_12 - Check if the improved QA system return a more accurate answer compare to the previous one.
* QATest\_13 - Check if the improved QA system return a more accurate answer.
* QATest\_14 - Check if the improved QA system work for unrelated questions.

**User Story # 838 - Upload documents into database**

Unit Test

* **UploadServiceTest.test\_execute\_success**: Test if the upload service can process an upload request and successfully create a record.
* **UploadServiceTest.test\_execute\_failure**: Test if the upload service can process an upload request and fail upon an invalid request.
* **AuthorizationTest.test\_isValidSession\_True**: Test if the Authorization class properly properly recognizes a valid session.
* **AuthorizationTest.test\_isValidSession\_False**: Test if the Authorization class properly properly recognizes an invalid session.

Integration Test

* **UploadServiceTest.test\_sucess\_integration**: Test that after the service is deployed, the related services work as expected.

**User Story # 857 - Update User Status**

Unit Test

* **UpdateUserServiceTest.test\_execute\_lock\_succcess**: Test that the service properly accepts a lock request, and the user in question is properly locked afterwards.
* **UpdateUserServiceTest.test\_execute\_inactivate\_succcess**: Test that the service properly accepts a inactivate request, and the user in question is properly inactive afterwards.
* **UpdateUserServiceTest.test\_execute\_lock\_failure**: Test that the service properly fails after an invalid lock request.
* **UpdateUserServiceTest.test\_execute\_inactivate\_failure**: Test that the service properly fails after an invalid inactivate request.

Integration Test

* **UpdateUserServiceTest.test\_sucess\_integration**: Test that after the service is deployed, the related services work as expected.

**User Story # 867 - View registered users**

Unit Test

* **ListUsersServiceTest.test\_execute\_succcess**: Test if the list users service can obtain a list of users and return that list as a JSON string.
* **ListUsersServiceTest.test\_execute\_nofailure**: Test if the list users service still performs a valid search and returns the proper JSON even if URL parameters have an invalid format.

Integration Test

* **ListDocuments.test\_sucess\_integration**: Test that after the service is deployed, the related services work as expected.

**User Story # 855 - View uploaded documents**

Unit Test

* **ListUsersServiceTest.testListDocumentsServiceSucceedsIfValidRequest**: Test if service (ListDocumentsService) fulfils a lis-documents request properly.
* **ListDocuments.testListDocumentsServiceFailsIfInvalidMethod**: Test if service (ListDocumentsService) fails if accessed through a POST request.

Integration Test

* **ListDocuments.test\_sucess\_integration**: Test that after the service is deployed, the related services work as expected.

**User Story # 851 - Download original PDF**

Unit Test

* **DownloadService.textIfCanDownload**: Test if service (DownloadService) fulfils a download request properly.
* **DownloadServoce.testDownloadServiceFailsIfInvalidMethod**: Test if service (DownloadService) fails if accessed through a POST request.

Integration Test

* **DownloadService.test\_sucess\_integration**: Test that after the service is deployed, the related services work as expected.

**User Story # 843 - Search document**

Unit Test

* **SearchUserServiceTest.test\_execute\_search\_succcess**: Test that the service properly accepts a search request, and the search is fulfilled properly.
* **SearchUserServiceTest.test\_execute\_search\_error**: Test that the service properly accepts an invalid search request, and properly returns information about the error.

Integration Test

* **SearchUserServiceTest.test\_sucess\_integration**: Test that after the service is deployed, the related services work as expected.

**User Story # 910- Add QA pairs to index and ranker**

Unit Test

* **QaAdditionService.test\_execute\_succcess**: Test that the service properly accepts an Add-Qa-Pair request, and each question is indexed and ranked.
* **QaAdditionService.test\_execute\_fail**: Test that the service properly rejects an Add-Qa-Pair request, with an invalid body.

Integration Test

**QaAdditionService.test\_integration\_succcess**: Test that the service properly copes with the search mechanism.

# Glossary

**QA subsystem**: Question and Answer subsystem.

**Solr**: Solr is a standalone enterprise search server with a REST-like API. You put documents in it (called "indexing") via JSON, XML, CSV or binary over HTTP. You query it via HTTP GET and receive JSON, XML, CSV or binary results.

**IBM Bluemix**: Cloud platform to accelerate innovation on both sides of the firewall. It is mainly a Platform As A Service provider that supports many programming languages and technologies.

**Watson Retrieve and Rank**: The IBM Watson™ Retrieve and Rank service combines two information retrieval components in a single service: the power of Apache Solr and a sophisticated machine learning capability. This combination provides users with more relevant results by automatically re-ranking them by using these machine learning algorithms.

**Watson Document Conversion**: The IBM Watson™ Document Conversion Service converts a single HTML, PDF, or Microsoft Word™ document. The input document is transformed into normalized HTML, plain text, or a set of JSON-formatted Answer units that can be used with other Watson services, like the Watson Retrieve and Rank Service.

**ElephantSQL**: ElephantSQL is a PostgreSQL database hosting service. It exposes the PostgreSQL engine as a web service.

# Appendix

## Appendix A - UML Diagrams

### Static UML Diagrams

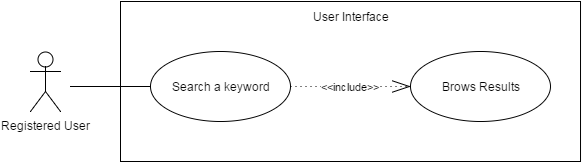


Figure Y-1 - Use Case Diagram 1

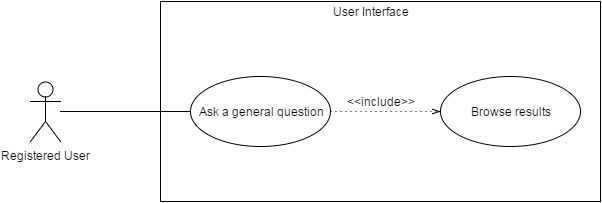


Figure Y-2 - Use Case Diagram 2

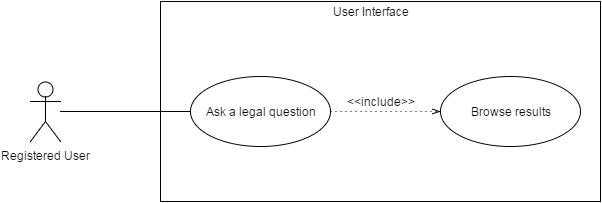


Figure Y-3 - Use Case Diagram 3

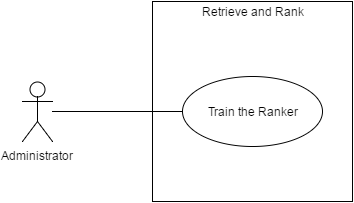


Figure Y-4 - Use Case Diagram 4

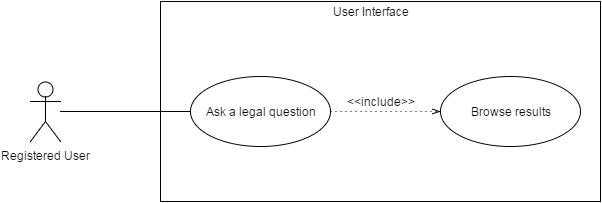


Figure Y-5 - Use Case Diagram 5

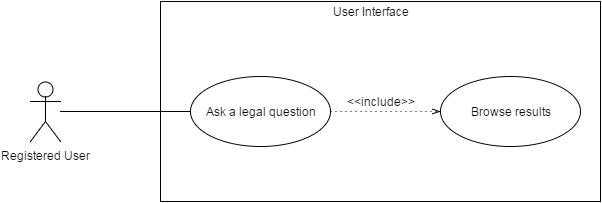


Figure Y-6 - Use Case Diagram 6

User Story #838 User.png

Figure F-1 - Use Case Diagram 7

User Story #857 User.png

Figure F-2 - Use Case Diagram 8

User Story #910 User.png

Figure F-3 - Use Case Diagram 9

User Story #855 User.png

Figure F-4 - Use Case Diagram 10

User Story #843 User.png

Figure F-5 - Use Case Diagram 11

User Story #867 User.png

Figure F-6 - Use Case Diagram 12

User Story #851 User.png

Figure F-7 - Use Case Diagram 12

### 

### 

### Dynamic UML Diagrams



Figure Y-7 - Sequence Diagram 1



Figure Y-8 - Sequence Diagram 2



Figure Y-9 - Sequence Diagram 3



Figure Y-10 - Sequence Diagram 4



Figure Y-11 - Sequence Diagram 5



Figure Y-12 - Sequence Diagram 6

User Story #838 Sequence.png

Figure F-9 - Sequence Diagram 8

User Story #857 Sequence.png

Figure F-10 - Sequence Diagram 9

User Story #855 Sequence.png

Figure F-11 - Sequence Diagram 10

User Story #851 Sequence.png

Figure F-12 - Sequence Diagram 11

User Story #867 Sequence.png

Figure F-13 - Sequence Diagram 12

User Story #843 Sequence.png

Figure F-14 - Sequence Diagram 13

User Story #910 Sequence.png

Figure F-15 - Sequence Diagram 14

## 

## 

LegalWise 2.0 Use Cases.png

Figure F-16 - LegalWise 2.0 Use Case diagram: FULL

## 

## 

## Appendix B - User Interface Design

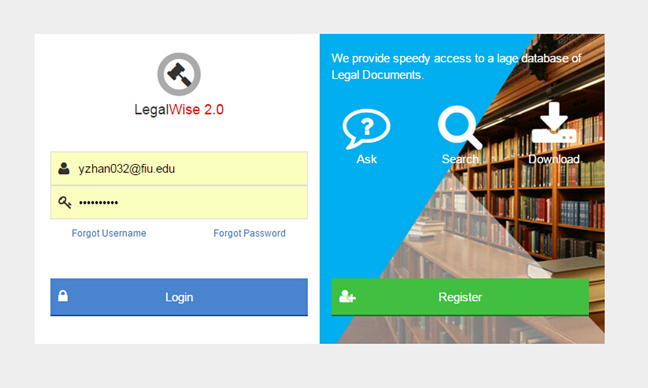


Figure B1 - Login Page

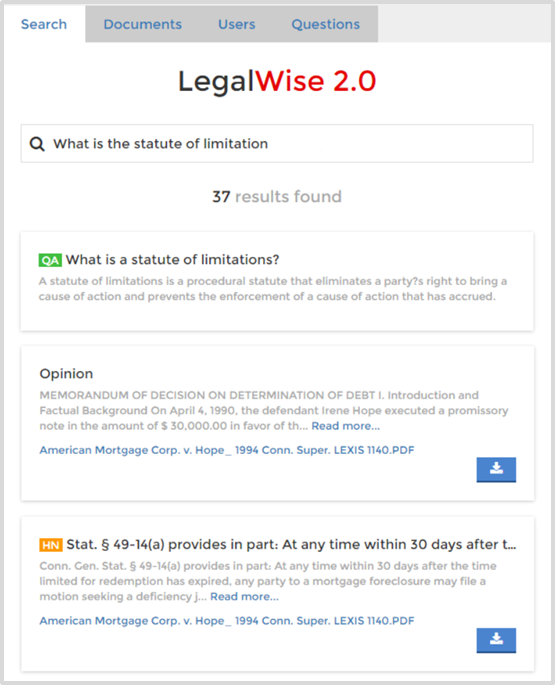


Figure B2 - Search Page

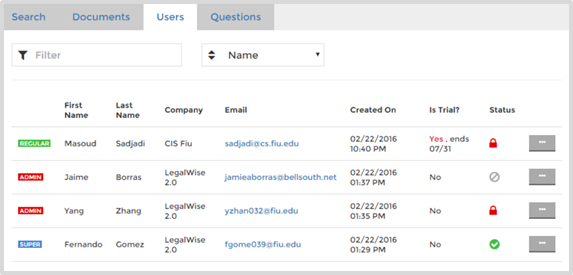


Figure B3 - User Page

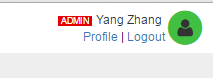


Figure B4 - Logout

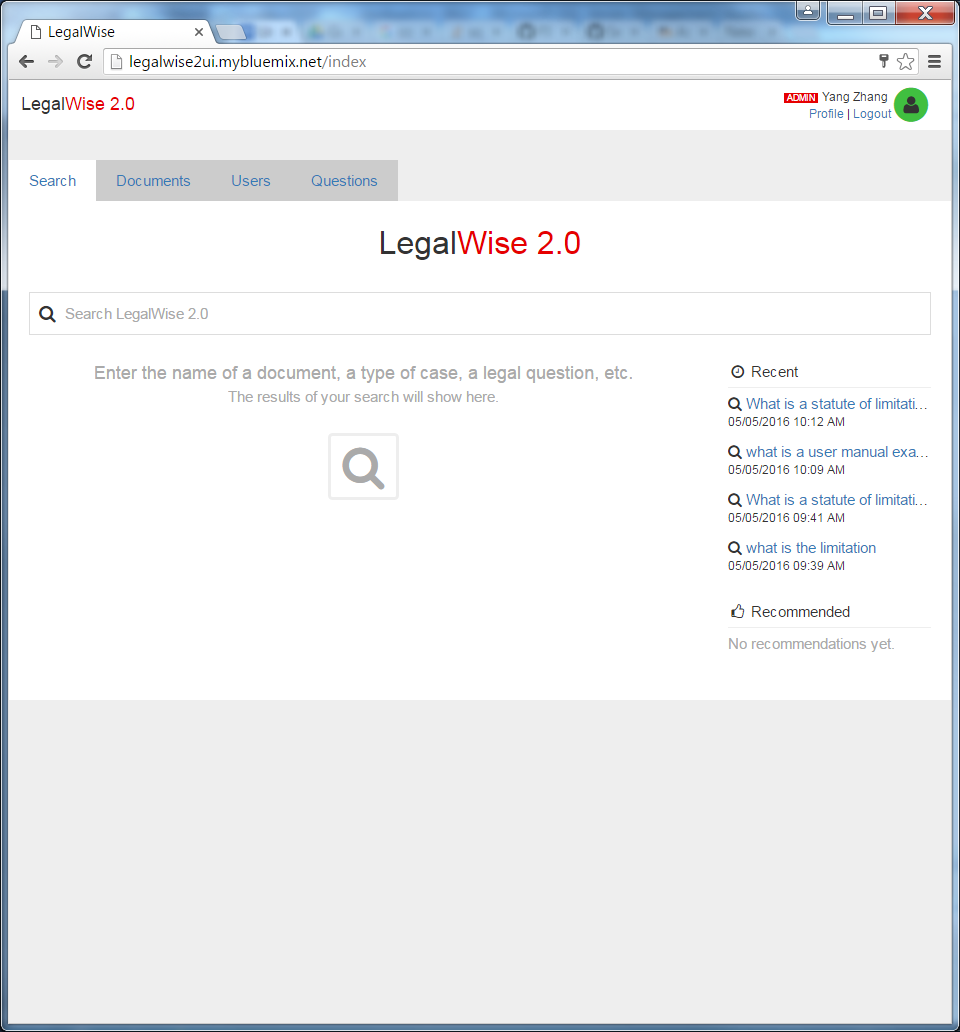


Figure B5 - Recent Search History

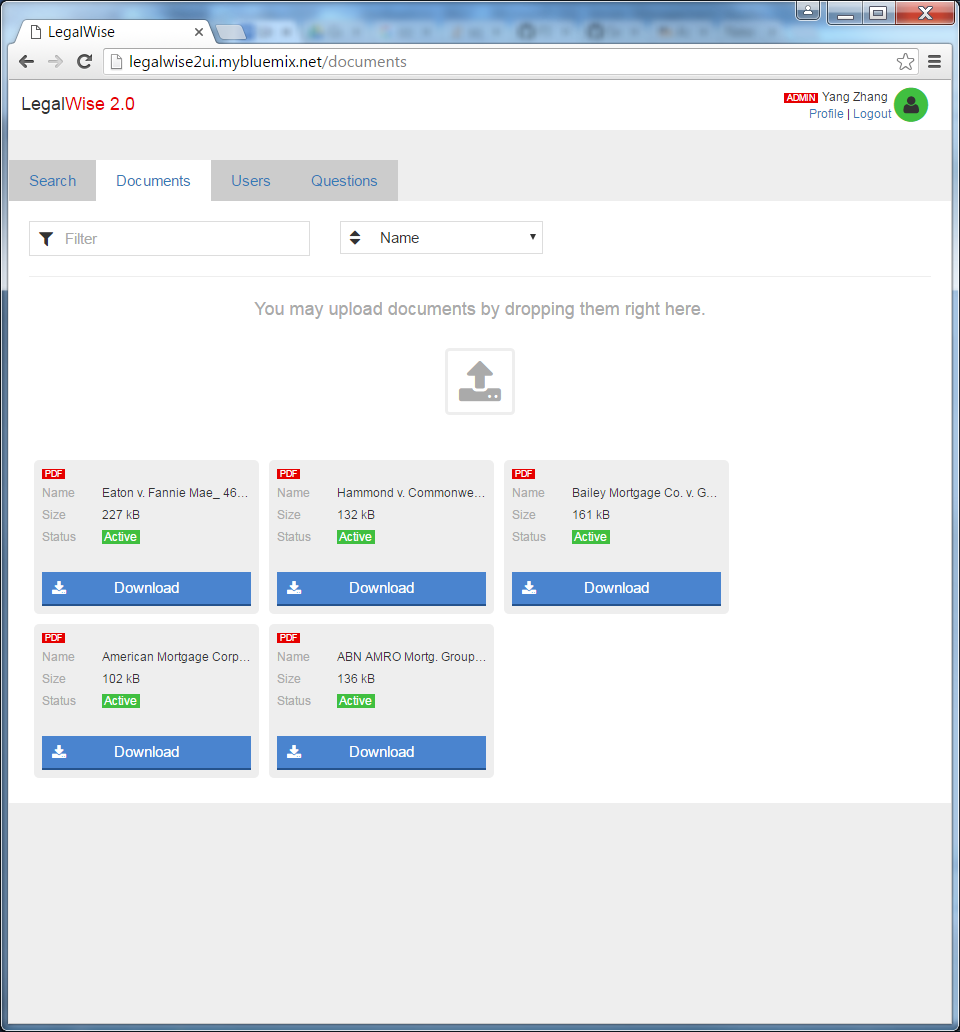


Figure B6 - Download/Upload Document Page

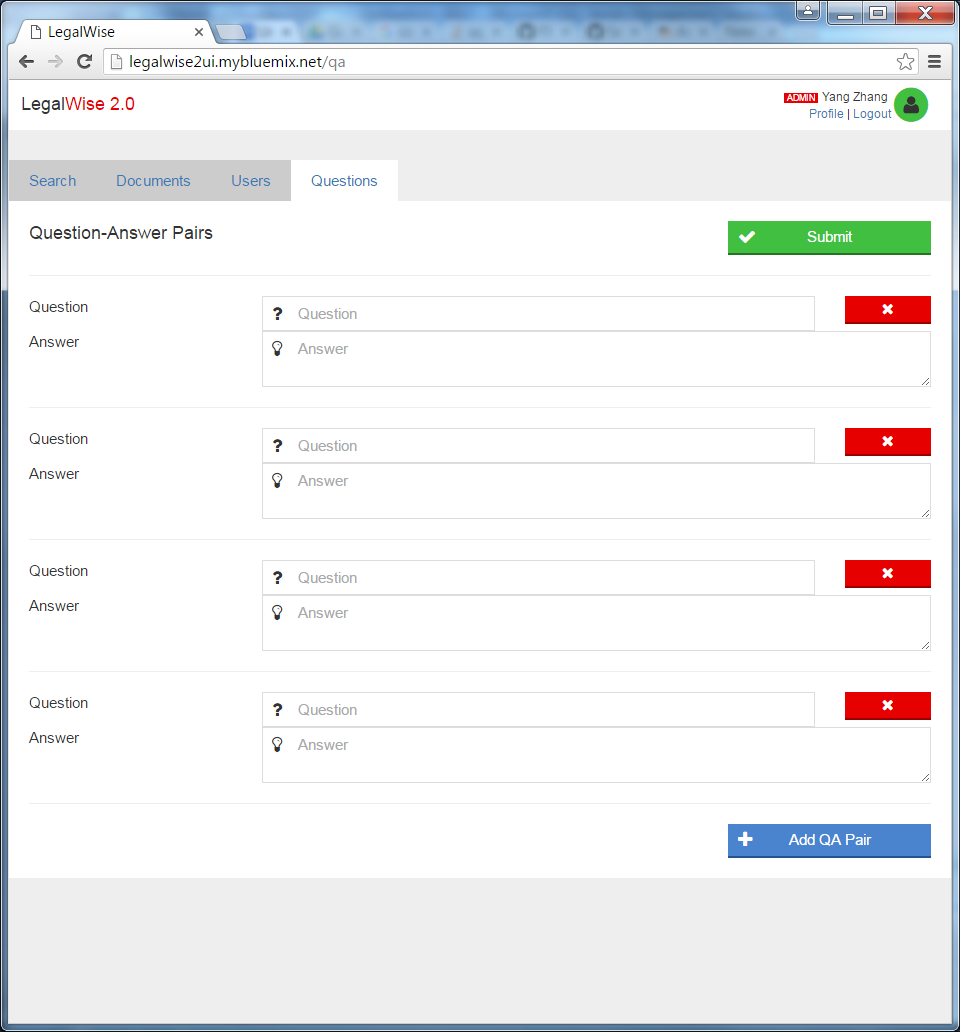


Figure B7 - Add question-answer Pairs Page

## Appendix C - Sprint Review Reports

**Sprint 1 Report**

**Date:** 01/29/2016

**Attendees:** Yang Zhang, Jaime Borras

**Discussed Topics:**

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

User Story #836

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

None.

**Sprint 2 Report**

**Date:** 02/11/2016

**Attendees:** Fernando Gomez, Yang Zhang, Jaime Borras

**Discussed Topics:**

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

User Story #842

User Story #848

User Story #838

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

None.

**Sprint 3 Report**

**Date:** 02/26/2016

**Attendees:** Fernando Gomez, Yang Zhang, Jaime Borras

**Discussed Topics:**

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

User Story #857

User Story #859

User Story #867

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

None.

**Sprint 4 Report**

**Date:** 03/18/2016

**Attendees:** Fernando Gomez, Yang Zhang, Jaime Borras

**Discussed Topics:**

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

User Story #880 (continue working on training data & train the model)

User Story #855

User Story #851

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

User Story #843

**Sprint 5 Report**

**Date:** 04/01/2016

**Attendees:** Fernando Gomez, Yang Zhang, Jaime Borras

**Discussed Topics:**

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

User Story #843

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

User Story #903

**Sprint 6 Report**

**Date:** 04/15/2016

**Attendees:** Fernando Gomez, Yang Zhang, Jaime Borras

**Discussed Topics:**

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

User Story #903

User Story #915

User Story #910

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

None

**Sprint 7 Report**

**Date:** 04/29/2016

**Attendees:** Fernando Gomez, Yang Zhang, Jaime Borras

**Discussed Topics:**

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

No new user stories for sprint 7

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

None

## 

## 

## Appendix D - Sprint Retrospective Reports

**Sprint 1 Retrospective**

**Date:** 01/29/2016

**Attendees:** Fernando Gomez, Yang Zhang

**Discussed Topics:**

**What went wrong?**

* We underestimated the time to needed to get version 1 of the application working in both local and remote environments. Specifically, we had issues running a local version of the Websphere server to run Java Liberty apps. We estimated 4 hours, and ended up using 10.
* We miss calculated the severity of the issues present in version 1. Basically, we separated 6 hours for setup, and ended up using 20.

**What went right?**

* We established a healthy communication with our project owner. We were able to understand the objective and scope of the project, and we agreed on how to include the remaining features.
* We design a new UI which was very well received by the project owner.
* Team communication was established successfully.

**How to address the issues in the next sprint?**

* **How to improve the process?**

Before assigning points, we need to improve the research process. We should not carry on with an estimate until we are completely familiar with all of the technologies involved in the process.

* **How to improve the product?**

- Our app is currently dependent on the CIS server, running the QA system from a virtual machine. We need to move it to the cloud (IBM Bluemix) and host everything in a totally backup and cloud-based environment.

- As this is an app for users who might not be computer savvy, we need to increase the priority of the user experience (UI-UX), and ensure each visual feature is mobile ready.

**Sprint 2 Retrospective**

**Date:** 02/12/2016

**Attendees:** Fernando Gomez, Yang Zhang

**Discussed Topics:**

**What went wrong?**

* We spend 100% of our resources coding and testing the user stories and left the engineering of the project a bit unattended. This is a problem of prioritization.
* We fell behind in the documentation.

**What went right?**

* We were able to create a working new design which was very well-liked by the project owner.
* We were able to implement two very important user stories that were left in the backlog of version 1. Document upload and highlight searching are currently working.
* We got the Solr server to work locally, and have a very promising lead on how to use for indexing database records in Bluemix PostgreSQL.

**How to address the issues in the next sprint?**

* **How to improve the process?**

- From now on, the software engineering side of the project has the highest of the priorities. User stories must be completely designed and planned before coding starts.

- Documentation must be updated along with the code. Team members cannot move to the next task/story before the documentation of the current one is completely finished and reviewed.

**How to improve the product?**

* Investigate alternatives to VM hosted Solr.

**Sprint 3 Retrospective**

**Date:** 02/26/2016

**Attendees:** Fernando Gomez, Yang Zhang

**Discussed Topics:**

**What went wrong?**

* We were not able to test the IBM Watson Services Bluemix on real documents. The test data was manually created. This was due to the fact that the time it took to get the service working exceeded the time previously allotted.

**What went right?**

* We were able to finish all user stories related to user management.
* The session manager is now using the actual database for identity provision, and the concepts of ‘locked user“ and “inactive user” were inserted and properly implemented and tested.
* The project owner was very fond of the UI/UX created for the users page.
* We inserted the concept of “profiles”, which is basically a field in the user table that says what type of access does each user have.
* Administrative pages (documents, users) were properly restricted so regular users do not have access to them.
* IBM Retrieve and Rank is on and working for sample data and can ask questions and then get back related ranked results.

**How to address the issues in the next sprint?**

* **How to improve the process?**

We need to focus on the search mechanism which it is the heart of the application. For this reason, both resources in the team should plan the next 2 sprints ahead of time and make sure all stories related to the search mechanism are finished by then. All other sides of the application shall have a lower priority and shall be placed on hold until the search feature is functional.

* **How to improve the product?**

The following Bluemix addons should be deeply reviewed and a proposal should be constructed, to be evaluated by Dr. Finalyson. These features will help us get rid of the QA VM.

**Sprint 4 Retrospective**

**Date:** 03/18/2016

**Attendees:** Fernando Gomez, Yang Zhang

**Discussed Topics:**

**What went wrong?**

* The format of training data accepted by Retrieve and Rank is different than the one we were suggested using by Dr. Finlayson. We need to figure out what we should do with the training data. Also we need to create more legal data.
* What went right?
* Presentation practiced and got feedback.
* Converted legal PDFs to JSON format which is required by Retrieve and Rank.
* Can ask a legal question now.
* The project owner was very fond of the UI/UX created for the users page.

**How to address the issues in the next sprint?**

* **How to improve the process?**

We need to focus on the collection of legal training data and train the ranker for the ask legal questions mechanism which it is the heart of the application. We will find a way for the format of the training data first and then collect as many legal training data as possible to train the ranker to make sure the accuracy of our system.

* **How to improve the product?**

The training data are to be evaluated by Dr. Finalyson. This would help us improve our accuracy of the system.

**Sprint 5 Retrospective**

**Date:** 04/01/2016

**Attendees:** Fernando Gomez, Yang Zhang

**Discussed Topics:**

**What went wrong?**

* One of the resources had a serious family emergency and had to miss a week of work.

**What went right?**

* We established communication with IBM and were able to fix the issues regarding Watson SDK.
* We successfully extracted legal q&a pairs.
* We agreed on a question extraction pattern and algorithm.
* We develop the full search.

**How to address the issues in the next sprint?**

* **How to improve the process?**

- Create an accurate ground truth table.

- Parse the head notes and turn them into answer units.

* **How to improve the product?**

Buffer the frequent questions and show last responses fast. The perform the search and modify the result accordingly.

**Sprint 6 Retrospective**

**Date:** 04/15/2016

**Attendees:** Fernando Gomez, Yang Zhang

**Discussed Topics:**

**What went wrong?**

Need to convert our training data to a new format for the new mechanism .

**What went right?**

* We successfully extracted legal q&a pairs.
* We have more training data.
* We agreed on a new question extraction pattern and algorithm.
* The system now has more accuracy to answer questions.

**How to address the issues in the next sprint?**

* **How to improve the process?**

- Improve the accurate ground truth table.

- Convert the training data to the new format.

* **How to improve the product?**

Retrain the ranker and it will provide more accuracy.

**Sprint 7 Retrospective**

**Date:** 04/29/2016

**Attendees:** Fernando Gomez, Yang Zhang

**Discussed Topics:**

**What went wrong?**

Need to wait for the Bluemix profession’s solution to the ranker’s problem caused by its updates.

**What went right?**

* We created new format of the training data.
* We have more training data.
* The system now has more accuracy to answer questions.
* The system now allow admin to add QA pairs via web form.

**How to address the issues in the next semester?**

* **How to improve the process?**

- Improve the accurate ground truth table.

- Accomplish the user stories in product backlog.

* **How to improve the product?**

Collect as many as possible training data.

# References

1. IBM BlueMix - platform as a service <http://www.ibm.com/cloud-computing/bluemix/>
2. Watson Retrieve and Rank Service - used for basic search and ranking: <http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/retrieve-rank.html>
3. Watson Document Conversion Service - used for convert PDF files to JSON files for indexing: <http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/document-conversion.html>
4. ElephantSQL - Used to expose PostgreSQL as a service. <https://www.elephantsql.com/>
5. Apache Maven - java building profile used in our application. <https://maven.apache.org/>