*Florida International University*

*School of Computing and Information Sciences*

Software Engineering Focus

Final Deliverable

Project Title:

LegalWise 3.0

[http://legalwise3.mybluemix.net/](http://legalwise2ui.mybluemix.net/)

**Team Members:**

Michel Angelo Roger

Renato Almeida

**Product Owner(s)**:

Jaime Borras

**Mentor(s)**:

Jaime Borras

**Instructor**:

Masoud Sadjadi

The MIT License (MIT)

Copyright (c) *2016 Florida International University*

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 3.0 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 6](#_Toc468236984)

[Current System 6](#_Toc468236985)

[Purpose of New System 6](#_Toc468236986)

[User Stories 8](#_Toc468236987)

[Implemented User Stories 8](#_Toc468236988)

[Pending User Stories 18](#_Toc468236989)

[Project Plan 20](#_Toc468236990)

[Hardware and Software Resources 20](#_Toc468236991)

[Sprints Plan 22](#_Toc468236992)

[Sprint 1 22](#_Toc468236993)

[Sprint 2 23](#_Toc468236994)

[Sprint 3 23](#_Toc468236995)

[Sprint 4 24](#_Toc468236996)

[Sprint 5 25](#_Toc468236997)

[Sprint 6 27](#_Toc468236998)

[Sprint 7 28](#_Toc468236999)

[System Design 28](#_Toc468237000)

[Architectural Patterns 29](#_Toc468237001)

[System and Subsystem Decomposition 30](#_Toc468237002)

[Deployment Diagram 33](#_Toc468237003)

[Design Patterns 33](#_Toc468237004)

[System Validation 35](#_Toc468237005)

[Glossary 38](#_Toc468237006)

[Appendix 39](#_Toc468237007)

[Appendix A - UML Diagrams 39](#_Toc468237008)

[Static UML Diagrams 39](#_Toc468237009)

[Dynamic UML Diagrams 41](#_Toc468237010)

[Appendix B - User Interface Design 45](#_Toc468237011)

[Appendix C - Sprint Review Reports 52](#_Toc468237012)

[Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents 57](#_Toc468237013)

[References 57](#_Toc468237014)

Introduction

This document presents the design and architecture of LegalWise 3.0. 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 2.0. 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 3.0 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, forgot password and register, this new system has a lot more to offer. This new release is a good attempt to make functional improvements and enhancements to the current version. There will be an improvement in the ranking and search accuracy. The response time search return will definitely improve. The system will make an enhancement to train the system to be smart in they way the results are retrieved. Moreover, the training system will be easy to use due to a different storage layer and access to the document repository. Also analyze further enhancement possibilities such as highlight the results that match the query and different phrase referring to a context.

LegalWise 3.0 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 elicited and linked to the LegalWise 3.0 project. These user stories were approved by Mr. Jaime Borras, the owner of this project. The previous system was based in Bluemix PaaS. The legalwise2.0 implements a wonderful UI, but need some enhancements on the retrieve and raking as well as several other features were missing in order to have a complete prove of concept. As a result, our user stories are planned as follows:

Implemented User Stories

**User Story # 932 - Environment setup and learning about the project**

As a developer, I would like to get together with the team and the project owner in order to introduce and meet each other. The purpose of this initial encounter was to understand and learn the environment setup and read documentation of previous release.

*Use Case*:

This use case lets the developer understand and learn the environment through the reading of documentation from previous releases in order to setup the system

*Actors:*

Developer.

*Pre-Conditions:*

1. Must be an administrator user on Bluemix.

*Description*:

1. Setting the development environment, Git, eclipse, database

2. Getting access to GitHub, IBM Bluemix.

3. Learning about the actual configuration

4. Learning about actual and future technologies that need to be in the project, Watson, Solr.

5. Reading and learning about the project and the goals of it.

6. Review previous release documents.

**User Story # 933 - Installation/ Configuration of tools and learning project material**

As an developer user, I would like to meet with team members and the group owner to understand the requirements and goals for this project. Installation and configuration of tools to setup development environment. Review previous project material and start learning the required technologies in order to complete this project.

*Use Case*:

This use case lets the administrator user meet with team member and group owner to understand requirements, review material, required technologies and goals for this project. Learning about the tools necessary for installation and configurations, in order to setup the development environment of the system.

*Actors:*

Developer user.

*Pre-Conditions:*

1. Must be an administrator user.

*Description*:

1. Understand and review project material of previous versions.

2. Investigate and learn new material and technologies to optimize the application.

3. Configuration and installation of necessary tools to run code. (Eclipse, Apache, Maven, Git, etc)

4. Obtain access to GitHub and IBM Blue-mix

5. Learning about the process of the environment set up.

6. Investigate about artificial technologies. Learn about Retrieve and Ranking and Solr to optimize application.

**User Story # 962 - Setup of IBM Bluemix 3.0**

As an administrator and developer, I need to have a development cloud environment similar to production on IBM, in order to test the new release version.

*Use Case*:

This use case lets the administrator user setup development cloud environment similar to production on IBM, in order to test the new release version in the system.

*Actors:*

Developer user.

*Pre-Conditions:*

1. Must be an administrator user.

2. Must have credentials

*Description*:

1. Creating user on the database side in order to be able to access the application

2. Set password and credentials of the users

3. Setup Git Repository in BlueMix for code development and peer programing

4. Brings R&R from previous version to 3.0

5. Brings Document conversion from previous version to 3.0

6. Brings SQL database from previous version to 3.0

7. Brings Solr Repository from previous version to 3.0

8. Create new java liberty application on Bluemix through cloud foundry

9. Learn the framework and the deployments method

10. Set environmental variables for old services

11. Give access to all the team members to the application

**User Story # 936 Search and Ranking Accuracy**

As a user, I need the capability to search in the repository of documents based on a question, so I be able to get information of a case in the best efficient and easy manner.

*Acceptance Criteria*

* more than 90% of accuracy
* Search ½ sec max for returning

*Use Case # Improve Retrieve and Rank*

This use case let the user to enter a question to the system in order to get an accurate answer sort it by most to less relevant to the question.

*Actors:*

1. Users

*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 Retrieval and Ranking(R&R) of Watson IBM to look for a answer.
3. Then the Solr server returns the results to the R&R.
4. R&R sort and rank the results in order to give the most accurate at first
5. Use case ends when the results display on page.

**User Story # 849 Reset Password**

As a registered user I want to be able to reset my password. So that if I forget my login password, I can reset it and gain access to the site.

*Acceptance Criteria*

* Must be a registered user.
* Invalid login results in error message on login page
* If user clicks “reset password”, is asked for e-mail address to send password reset link

*Use Case # Reset Password*

This use case lets the registered user to reset password. So that if registered user forgets login password, the system can reset it and gain access to the site.

*Actors:*

1. Registered user

*Pre-Conditions:*

1. Must be a registered user.

Description:

1. User clicks on forgot password button
2. User enters their email address (username) and submits verification link
3. The server generates a hash code and sends email to user address with the an embedded link
4. User clicks on link (or pastes it into browser window)
5. User enters new password which is submitted along with the given hash code to the database
6. Database validates the hash code and password and matches it up to the User
7. Password is hashed and saved to User account
8. User is sent to the login page again.

**User Story # 956 Taming Text**

As a developer, I need to be able to create question and answer for the already indexes documents in order to train the system to be smart in the way the results are retrieved.

*Acceptance Criteria*

* Improving the retrieval process in 95%
* Highlight the results that match the query

*Use Case #Taming Text*

This use case let the user to enter a question to the system in order to get an accurate answer sort it by most to less relevant to the question and the system need to handle in an smart way the question.

*Actors:*

1. 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 Retrieval and Ranking(R&R) of Watson IBM to look for a answer.
3. Then the Solr server returns the results to the R&R.
4. R&R sort and rank the results with the Machine learning algorithm previously train
5. Use case ends when the highlighted results display on page.

**User Story # 957 Continuation of Reset Password**

As a registered user I want to be able to reset my password. So that if I forget my login password, I can reset it and gain access to the site.

*Acceptance Criteria*

* Must be a registered user.
* Invalid login results in error message on login page
* If user clicks “reset password”, is asked for e-mail address to send password reset link

*Use Case # Reset Password*

This use case lets the registered user to reset password. So that if registered user forgets login password, the system can reset it and gain access to the site.

*Actors:*

1. Registered user

*Pre-Conditions:*

1. Must be a registered user.

*Description:*

1. User clicks on forgot password button
2. User enters their email address (username) and submits verification link
3. The server generates a hash code and sends email to user address with the an embedded link
4. User clicks on link (or pastes it into browser window)
5. User enters new password which is submitted along with the given hash code to the database
6. Database validates the hash code and password and matches it up to the User
7. Password is hashed and saved to User account
8. User is sent to the login page again.

**User Story # 974 Choose a persistent layer for easy training of the system**

As a developer, I need a flexible database schema that allow me to make changes on the project easily and fast in order to help me on the taming of the text labor.

*Acceptance Criteria*

* Schema less 100%
* Easy to make changes on the project, just modified the java class not further development need it.
* Solr workload save on the database for future training

*Use Case #001 Saving documents by parts on the Database*

This use case let the user to upload a document on the application which needs to be stored on the database in different pieces such as byte file, plain text per sections and metadata.

*Actors:*

1. Administrator users

*Pre-Conditions:*

1. Must be a registered user.
2. Must be logged in.
3. Must be an Administrator

*Description:*

1. Use case begins when the registered user go to the Documents page and drops a document on the drooping area.
2. After the system receives this upload request, it parses the request to the Document conversion in order to get the document per section as plain text.
3. Then the Document conversion return the result to the DAO layer
4. The DAO layer gets the document to be storage as metadata, binary file and plain text.
5. Use case ends when a successful file is shown on the document page back to the user.

**User Story # 975 Register new user**

As a user I want to be able to register into the application. So that if I provide the information required I could gain access to the site.

*Acceptance Criteria*

* Must be a user not stored in the system.
* Registered user will not be able to duplicate itself in the system
* Must provide a valid email address

*Use Case # Register*

This use case lets the user to register into the application. So that if I provide the information required, the system could gain me access to the site.

*Actors:*

1. User

*Pre-Conditions:*

1. Must be a user not stored in the system.

*Description:*

1. User clicks on Register button
2. User enters required fields and submits it to the system
3. System executes a validation process to see if email is valid and verify is not duplicated in the database.
4. If all conditions successful, user is registered in database.
5. If conditions are not satisfied, system return with an error
6. User is sent to back to the login page

**User Story #982 Analyze further enhancement possibility**

As a user, I would like to see if the system can be more intelligent in order to get better results in my search

*Acceptance Criteria*

* More intelligence response based on the input search, i.e. different phrase referring to a context

*Use Case #001 Different ways to refer to the same text store on the database*

This use case let the user to submit a particular query to the system and the system should be able to identify the correct part of the test

*Actors:*

1. Users

*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 Retrieval and Ranking(R&R) of Watson IBM to look for a answer.
3. Then the Solr server returns the results to the R&R.
4. R&R sort and rank the results with the Machine learning algorithm previously train

**User Story #986 MongoDB enhancement**

As an admin of the system, I need to have better control of the register users in the system in order to be able to quickly see user status on the system.

*Acceptance Criteria*

* Search queries for users need to return with 99% of accuracy

*Use Case #001 Search for a particular user by name*

This use case let the admin to search for a particular user in by name as an example of the filter

*Actors:*

1. Users

*Pre-Conditions:*

1. Must be a registered user.
2. Must be logged in.
3. Must be an Administrator

*Description:*

1. Use case begins when the registered admin user go to the user page and enter a name on the filter box.
2. After the system receives this search request, it parses the request to the DAO layer
3. Then the MongoDB system parser the query.
4. MongoDB return the results to the user.

**User Story #996 Creating documentation for future releases**

As a developer, I have to create the documentation of this release for future reference

*Acceptance Criteria*

* Final Project Documentation
* Poster
* Manual and Instruction
* Video of installation
* Github migration

*Use Case #001 Files ready for submission*

This use case let the future developer to have a complete set of documentation in order to start working on future releases.

*Pre-Conditions:*

1. All the documentation should be on GDrive, mingle and github.

*Description:*

1. Need to finish all the required documentation

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 # 937 - Load more foreclosure cases**

* As a administrator user I would like to load more PDF documents for the foreclosure cases.

**User Story # 939 - Offer opinion about previous cases related to the search**

* As a administrator user I would like to create a case based on the search that the user put on the system. Provide a Pro & Con arguments about the cases related to the question.

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 2.0. But after sprint 1, we failed to obtain access to the Bluemix organization where the application is hosted This became a problem since we need to recreate all the hosting services in another organization in order to continue the work. We decide to create a new organization but link the services to the previous version in this case we didn’t has to recreate all the services and maintain the same data repositories.

To attack the project at hand, we divided the workload into two main parts. Fix and enhance the machine learning services, as in question-answer handling. Finish the UI features missing such as register and reset password for a user of the application. Change the under layer rational database for a NoSQL based in order to increase agility and time to market as well as easy the process of training of the system. As there were two resources available, each resource was assigned a part.

Hardware and Software Resources

The hardware and software resources used for LegalWise 3.0 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 1.8.
* **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.
* **mLab (MongoDB)**. The persistent data uses MongoDB technologies.

## 

Sprints Plan

Sprint 1

(8/29/2016 - 9/9/2016)

**User Story # 932 - Environment setup & learning about the project**

***Tasks***

● Environment setup.

● Reading project material.

***Acceptance Criteria***

● Meet with everyone on the project

● Finish environment setup

● Read project documentation

***Modeling***

No diagram available for this user story.  
  
**User Story # 933 - Installation/ Configuration of tools and learning project material**

***Tasks***

● Review previous documents and crucial information

● Development setup  
  
***Acceptance Criteria***

● First meeting as a group with product owner.

● Installation and configuration of necessary tools to set up development environment.

● Review, investigate, and learn project material in to complete the application.

***Modeling***

No diagram available for this user story.

Sprint 2

(9/12/2016 - 9/23/2016)

**User Story # 962 – Setting up IBM Blue Mix 3.0**

***Tasks***

● Create new user for the application

● Learning about Solr, schema design

● Git setup for team members

● Bring old application services

● IBM organization setup  
  
***Acceptance Criteria***

● Working environment with previous repositories

***Modeling***

No diagram available for this user story.

Sprint 3

(9/26/2016 - 10/7/2016)

**User Story # 849 - Reset password**

***Tasks***

● Handling Reset password.

● Apply email services.

● Review verification hash and service method

● Send html email

***Acceptance Criteria***

● Must be a registered user.

● Invalid login results in error message on login page

● If user clicks “reset password”, is asked for e-mail address to send password reset link

***Modeling***

- Figure – A3

- Figure – A4

**User Story # 936 - Search and Ranking Accuracy**

***Tasks***

● Document conversion

● Learning about Watson R&R training

● Taming text

● Solr setup

● App integration with Solr and R&R

***Acceptance Criteria***

● More than 90% of accuracy

● Search ½ sec max for returning

***Modeling***

- Figure - A1

- Figure - A2

Sprint 4

(10/10/2016 - 10/21/2016)

**User Story # 956 – Taming text**

***Tasks***

● Documentation.

● Creating ranking and Training system

● Integration testing

● Unit testing

● Development of the feature

● Solr configuration changes

● Research about Solr highlight capability

***Acceptance Criteria***

● Improving the retrieval process in 95%

● Highlight the results that match the query

***Modeling***

- Figure - A1

- Figure - A2

**User Story # 957 –Reset Password Continued**

***Tasks***

● Verify registered user

● Development of feature

● Testing Integration

● Documentation

● Understand the mail sender service and password resource controller

● Develop solution

***Acceptance Criteria***

● Must be a registered user.

● Invalid login results in error message on login page

● If user clicks "reset password", is asked for email address to send password reset link

***Modeling***

- Figure – A3

- Figure – A4

Sprint 5

(10/24/2016 - 11/4/2016)

**User Story** [#974](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/legalwise/cards/974) **Choose a persistent layer for easy training of the system**

***Tasks***

● Documentation

● [Integration testing](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/legalwise/cards/980)

● [Unit testing](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/legalwise/cards/979)

● [Make change on the objects that interact with the persistance layer](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/legalwise/cards/978)

● [Make change on the Persistence layer object](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/legalwise/cards/977)

● [Setup MongoDB](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/legalwise/cards/976)

● [Research about different schema less technologies](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/legalwise/cards/975)

***Acceptance Criteria***

● Schema less 100%

● Easy to make changes on the project, just modified the java class not further development need it.

● Solr workload save on the database for future training

***Modeling***

- Figure – A5

- Figure – A6

**User Story # 989 -Register a new user**

***Tasks***

● Validation stored in DB

● Development of feature

● Testing Integration

● Documentation

● Validation process in the database

● Develop solution

***Acceptance Criteria***

● Must be a user not stored in the system.

● Registered user will not be able to duplicate itself in the system

● Must provide a valid email address

***Modeling***

***Modeling***

- Figure – A9

- Figure – A10

Sprint 6

(11/7/2016 - 11/18/2016)

**User Story #** [982](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/legalwise/cards/982) **Analyze further enhancement possibility**

***Tasks***

● Documentation

● Implements paraphrasing

● Research about different solution

***Acceptance Criteria***

● More intelligence response based on the input search, ie different phrase referring to a context

***Modeling***

- Figure – A5

- Figure – A6

**User Story** [#986](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/legalwise/cards/986) **MongoDB enhancement**

***Tasks***

● Documentation

● Implementation and testing

***Acceptance Criteria***

● Search queries for users need to return with 99% of accuracy

***Modeling***

***Modeling***

- Figure – A7

- Figure – A8

Sprint 7

(11/28/2016- 12/2/2016)

**User Story #996 Creating documentation for future releases**

***Tasks***

● GitHub migation

● Presentation slides

● Poster

● Video

● Final Project Documentation

***Acceptance Criteria***

* Final Project Documentation
* Poster
* Manual and Instruction
* Video of installation
* Github migration

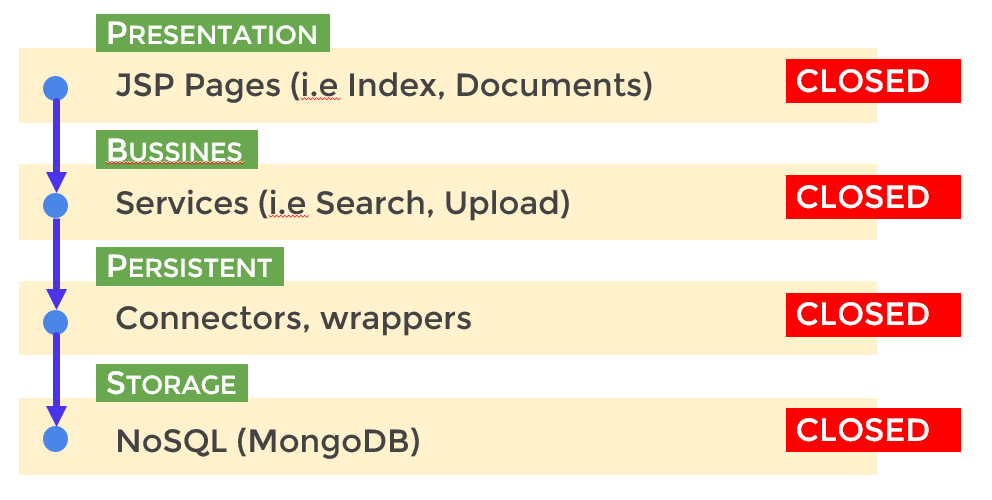
***Modeling***

No diagram available for this user story.

System Design

This section shows the system design for the LegalWise 3.0 application, including the architectural patterns, system and subsystem decomposition, deployment diagram, and design patterns.

Architectural Patterns



## 

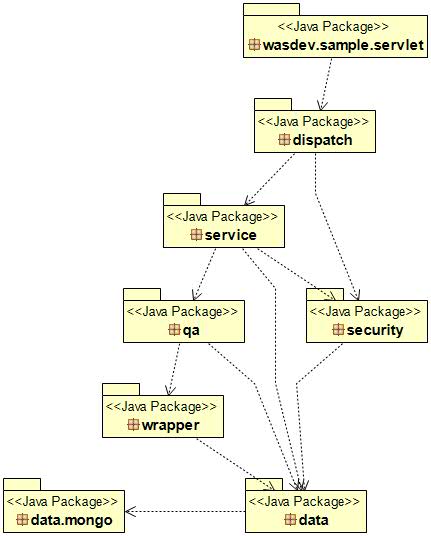
As shown above, LegalWise 3.0 was built using a Layered Architecture, strictly broken down into 4 layers.

* 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 **Storage** layer encapsulates the MongoDB 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



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, user password and create.

**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.
7. **Password:** fulfills a request to reset password.

**Data [package]**

This package contains all members that in some way communicate directly with the database. This is a factory that can create connectors of two types SQL or NoSQL 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.

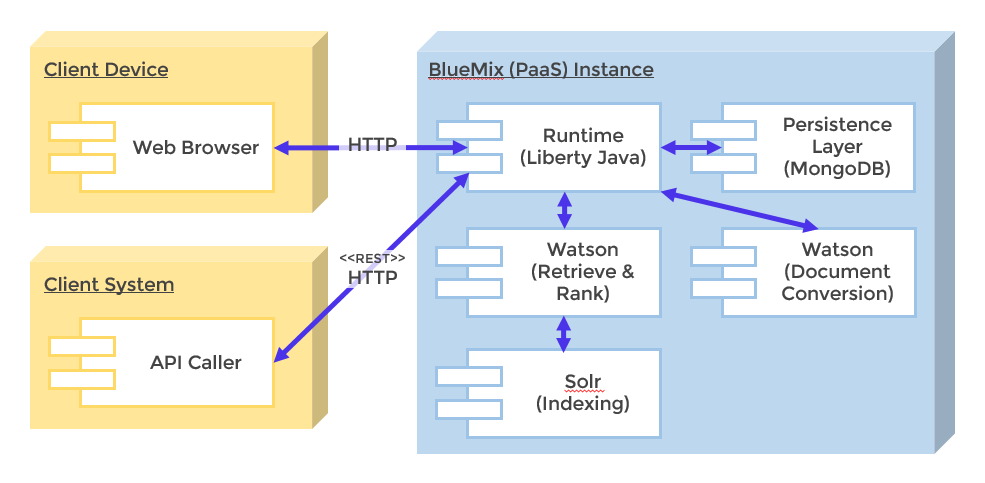
**Wrapper [package]**

This package contain objects that are used to persist the information on the storage layer.

**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



LegalWise 3.0 is a web application built and hosted on IBM BlueMix™ (PaaS) and written in Java, using Liberty profile as runtime. Apache Maven is used to manage dependencies and as building profile. The Watson Retrieve and Rank and Document Conversion services are used to aid in the retrieval and document conversion processes. Solr system is used to stored and index the unstructured data for the retrieve a ranking service. MongoDB is used as the NoSQL database hosting service for the app persistence layer. Most of this services are provided by the BlueMix platform, except the MongoDB which is hosted by mLab.

## Design Patterns

LegalWise 3.0 uses the following design patterns.

**Factory Method** - creational

This pattern is used to handle the instantiation of services. The Service Dispatcher and the Persistence layer connector defers to a helper method the decision about which service or connector 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 and the connections needs no more that one instance as settings are the same and should be synchronized, and connections need to has a centralize place to manage in order to reuse the open ones and avoid leaks of the same. Those are 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 integration and the unit test cases of each related user story. The integration 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.

**User Story # 962 – Setting up IBM Blue Mix 3.0**

Integration Test

* IT\_1 – SQL database has communication with the application.
* IT\_2 – Document Conversion has communication with the application.
* IT\_3 – Retrieve & Ranking has communication with the application.

**User Story # 936 - Search and Ranking Accuracy**

Unit Test

* UT\_01- Ranking and search priority are working as expected, 90% of effectivity.

Integration Test

* IT\_01- Test that the service properly accepts a search request, and the search is fulfilled efficiently.

**User Story # 956 – Taming text**

Unit Test

* UT\_01- Ranking and highlight text is reflected on the user page.

Integration Test

* IT\_01- Verify that user can execute a search and receive highlighted text.

**User Story # 957 –Reset Password Continued**

Unit Test

* UT\_01- Password is stored on the database and the application get a response back.

Integration Test

* IT\_01- Let the user reset password, get the email and gain access to the site.

**User Story** [#974](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/legalwise/cards/974) **Choose a persistent layer for easy training of the system**

Unit Test

* UT\_01- Load a document on MongoDB.
* UT\_02- Verified that the document can be retrieved.

Integration Test

* IT\_01- Load a document on MongoDB from the application.
* IT\_02- Verified that the user can download the document storage on the Data Layer.

**User Story # 989 -Register a new user**

Unit Test

* UT\_01- User is created on the database and can access the application.

Integration Test

* IT\_01- User can be created from the application and have access as a regular user to the application.

**User Story #** [982](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/legalwise/cards/982) **Analyze further enhancement possibility**

Unit Test

* UT\_01- Verify that user can execute a search and receive the expected text.

**User Story** [#986](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/legalwise/cards/986) **MongoDB enhancement**

Unit Test

* UT\_01- Verify that user can execute a search and receive the expected user.

# 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:** TheIBM 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.

**mLab:** Hosting service that provides MongoDB database infrastructure as a service.

Appendix

Appendix A - UML Diagrams

Static UML Diagrams

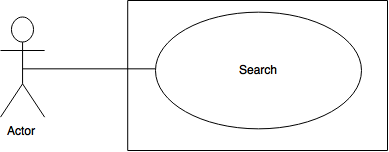
[](https://drive.draw.io/#G0Bzm9fyazWvO_ZmZzYXZXRjN1MUE)

Figure A-1

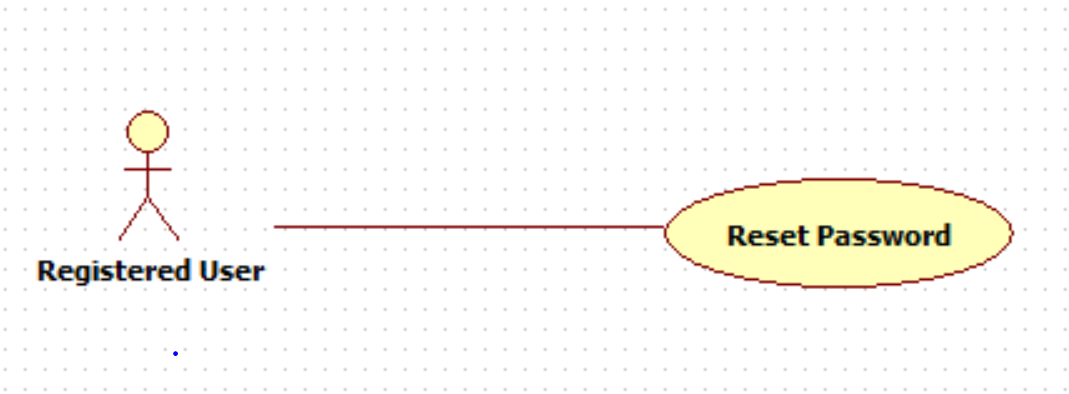


Figure A-3

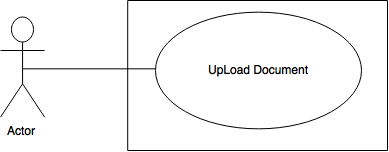
[](https://drive.draw.io/#G0Bzm9fyazWvO_UDNKVG5kc3B6M1U)

Figure A- 5

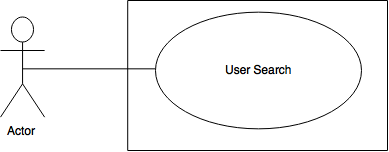
[](https://drive.draw.io/#G0Bzm9fyazWvO_VGQ3QmtHaHpYdXc)

Figure A- 7

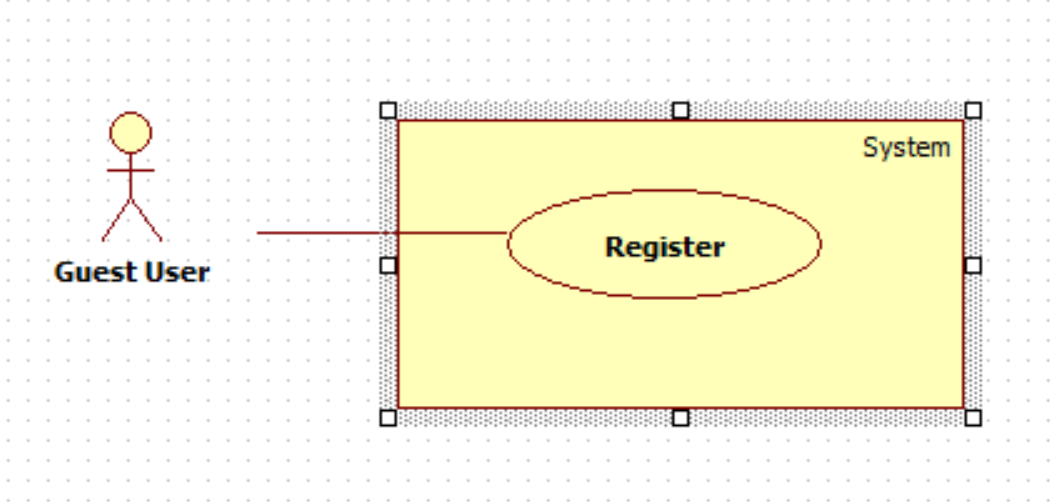


Figure A- 9

Dynamic UML Diagrams

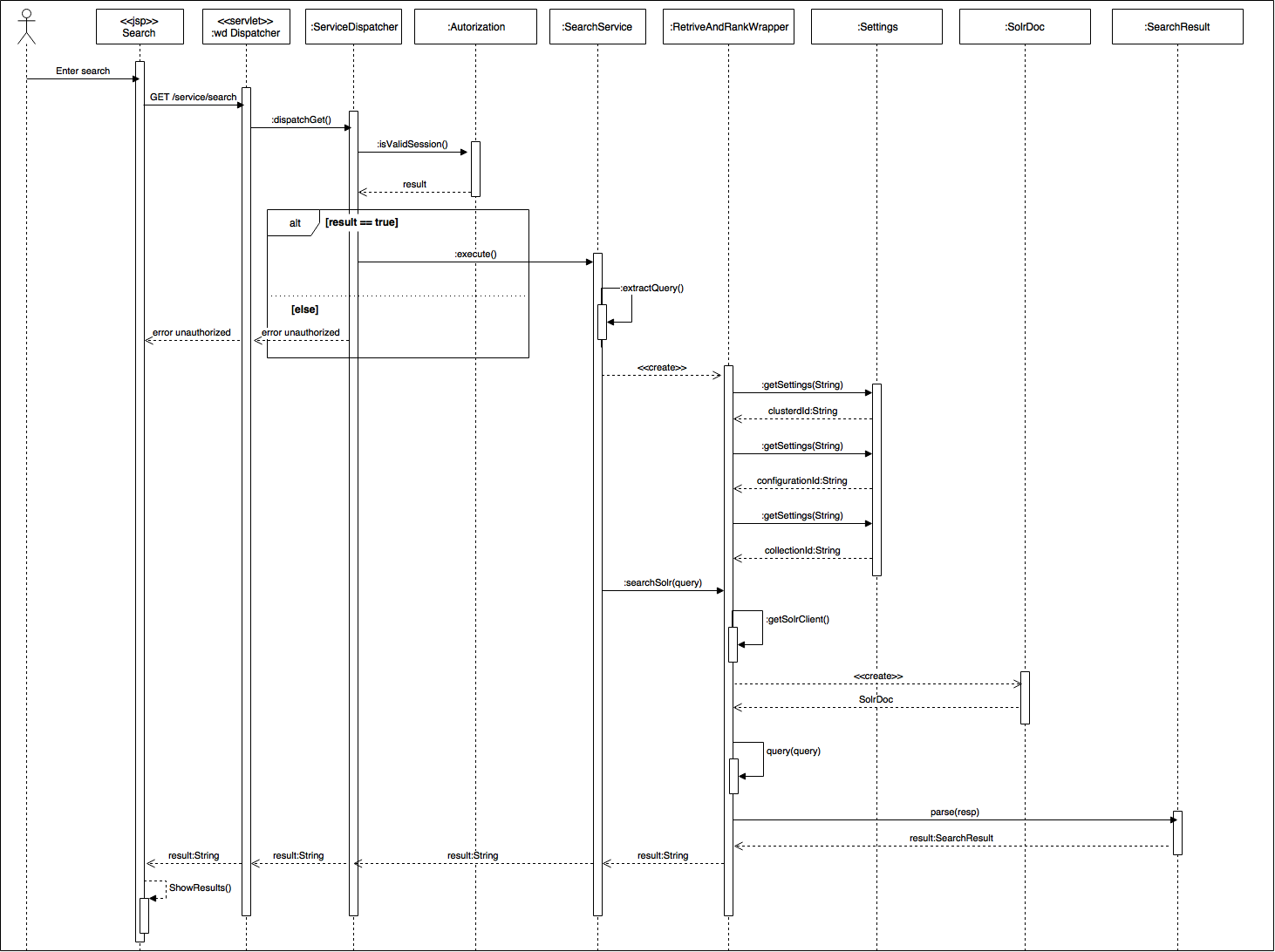
[](https://drive.draw.io/#G0Bzm9fyazWvO_NlFZMlRIaGloUlE)

Figure A-2

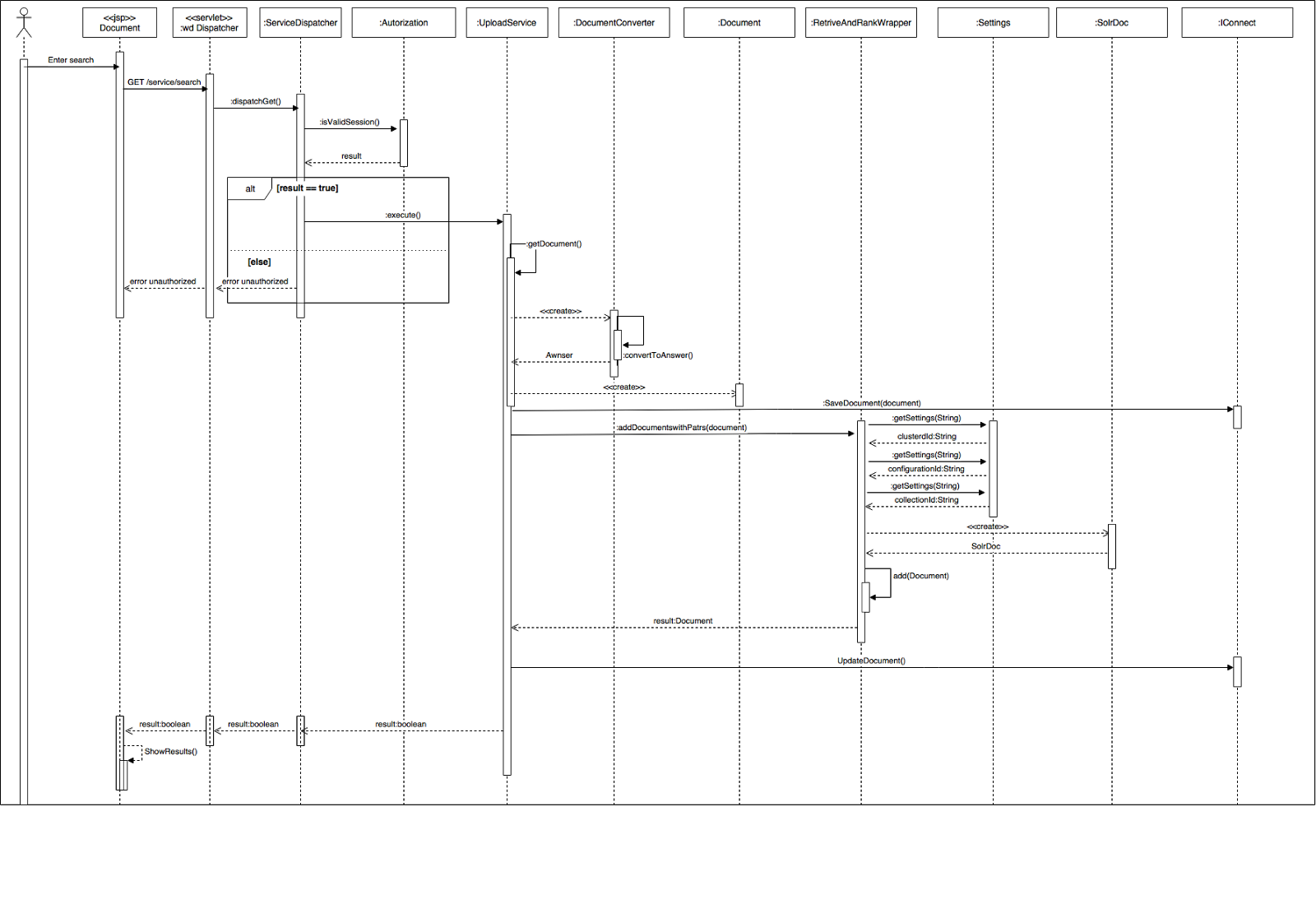
[](https://drive.draw.io/#G0Bzm9fyazWvO_VFVQdTh6S0NyWEE)

Figure A- 4

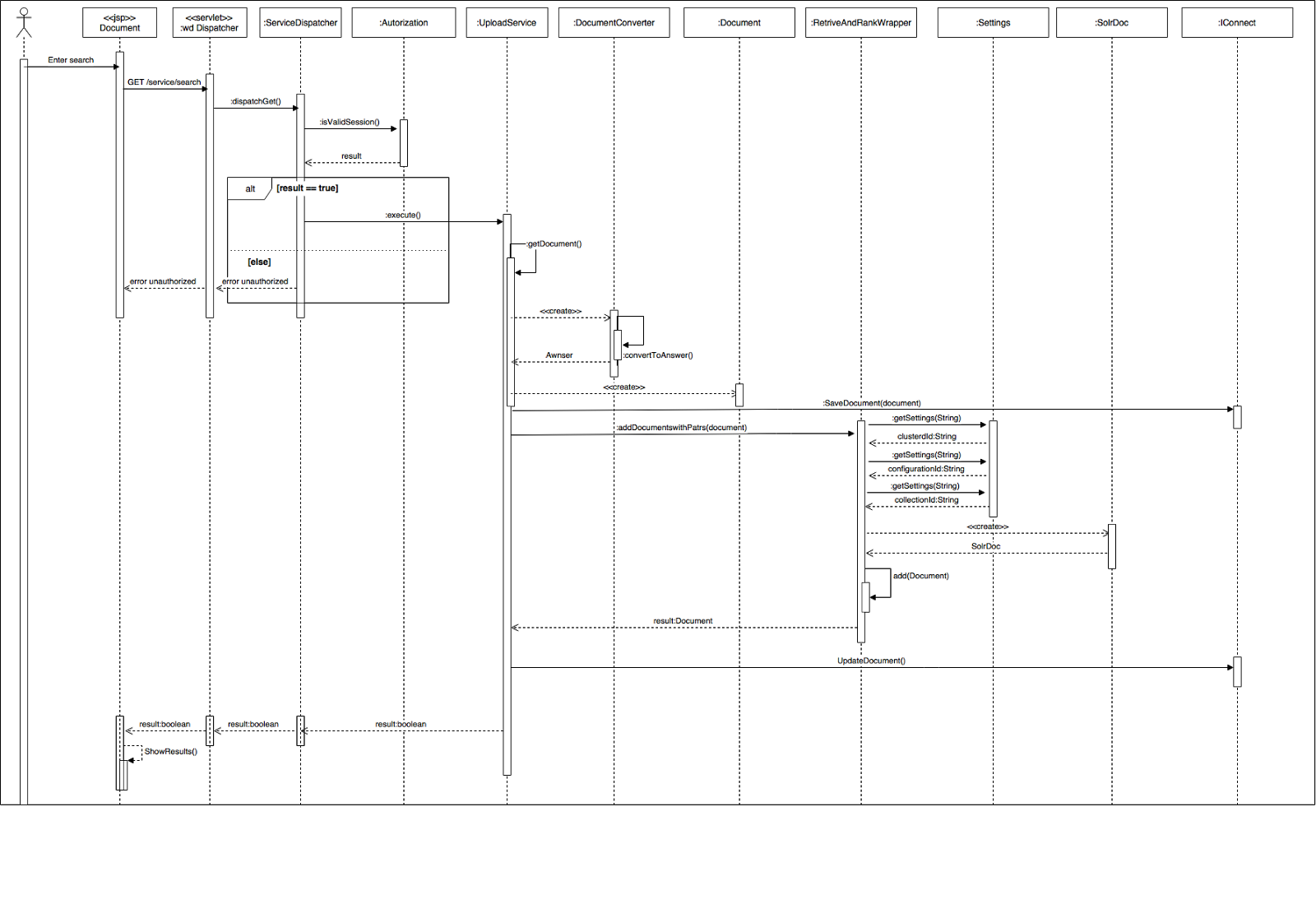
[](https://drive.draw.io/#G0Bzm9fyazWvO_VFVQdTh6S0NyWEE)

Figure A- 6

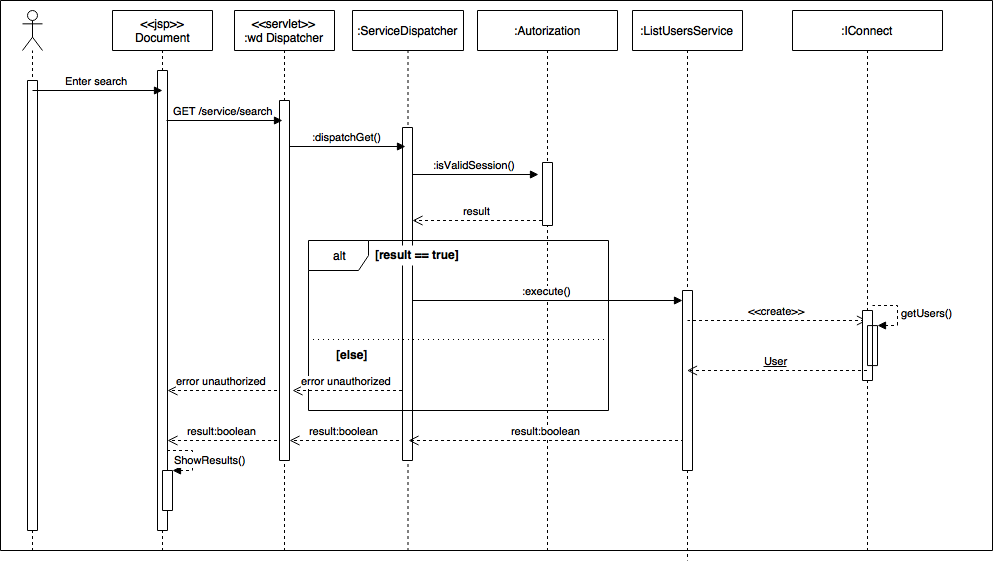
**[](https://drive.draw.io/#G0Bzm9fyazWvO_akU1OUdRaXhPcUk)**

Figure A- 8

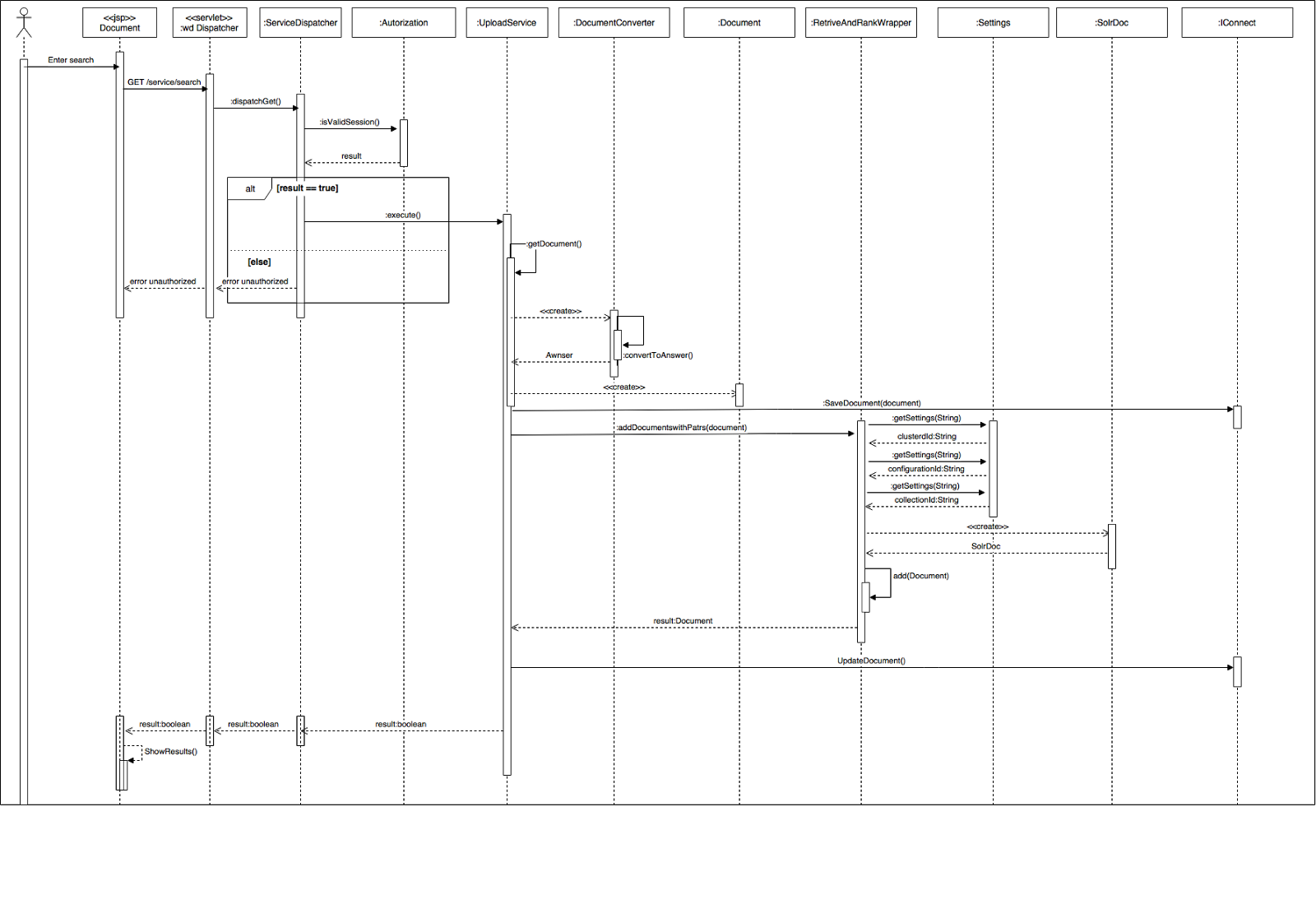
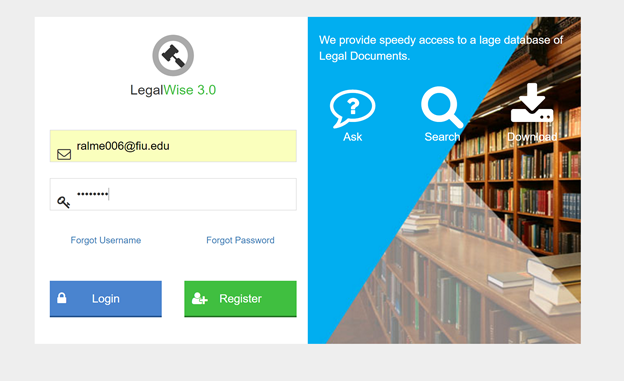
[](https://drive.draw.io/#G0Bzm9fyazWvO_VFVQdTh6S0NyWEE)

Figure A- 10

Appendix B - User Interface Design

 Figure B1 - Login Page

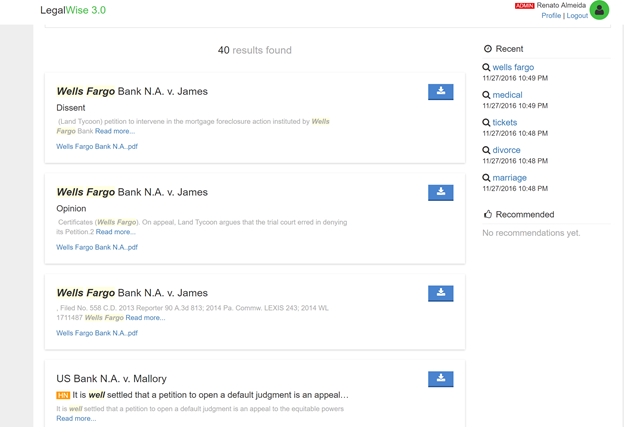


Figure B2 - Search Page

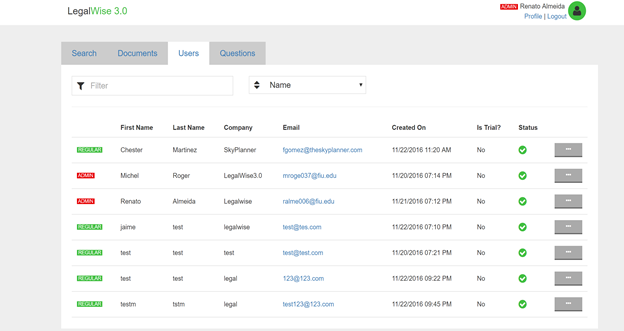


Figure B3 - User Page

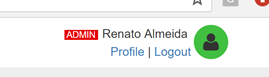


Figure B4 - Logout

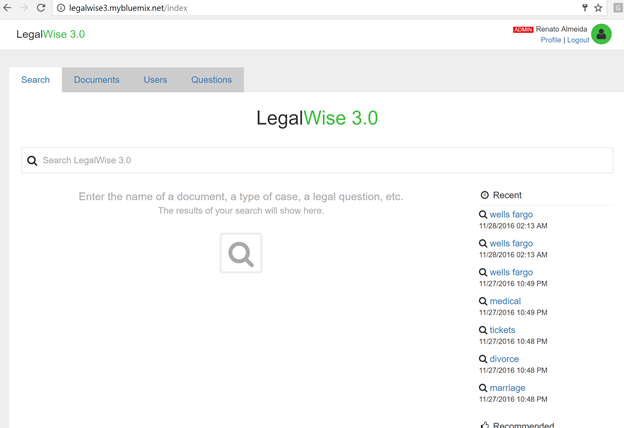


Figure B5 - Recent Search History

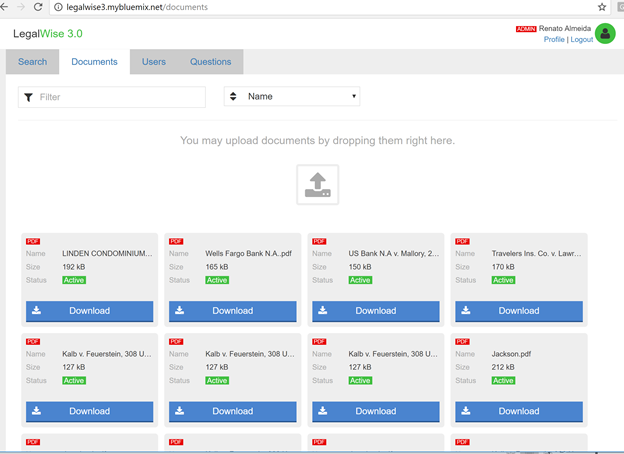


Figure B6 - Download/Upload Document Page

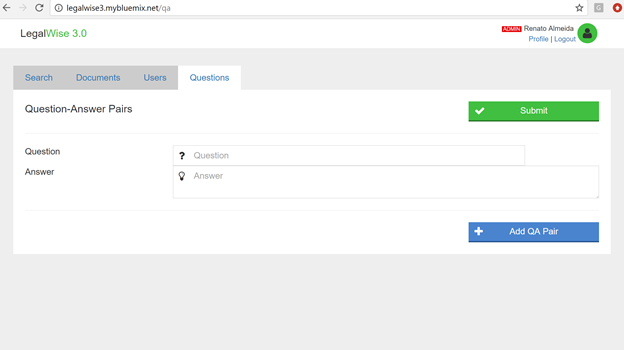


Figure B7 - Add question-answer Pairs Page

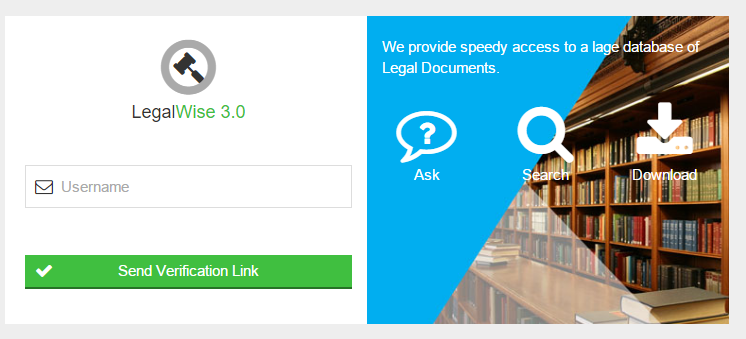


Figure B8 – Reset password email sent

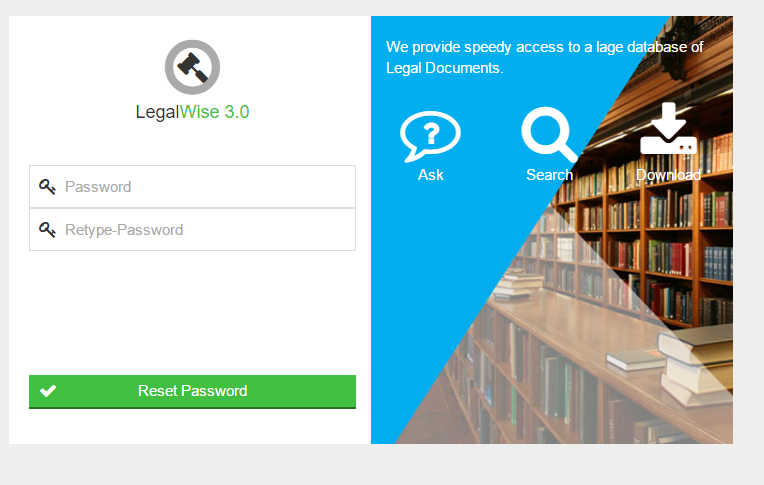


Figure B9 – Reset password

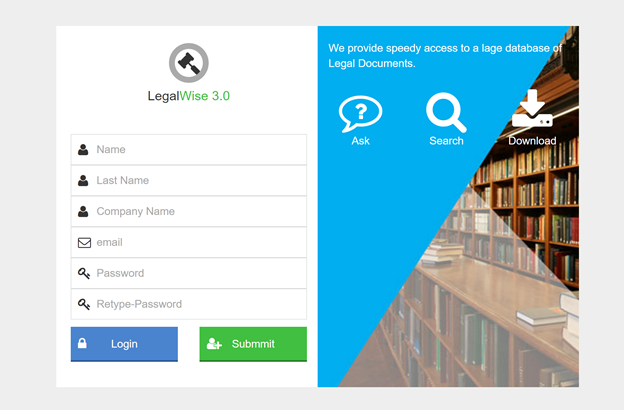


Figure B10 – Register new user

Appendix C - Sprint Review Reports

**Sprint 1 Retrospective**

Attendees: Renato Almeida, Roger Michel

Start time: 19:30

End time: 20:10

What went wrong?

* Did we do a good job estimating our team's velocity?
  + Preliminary task does not give you an idea of how long it will take each task
* Did we do a good job estimating the points (time required) for each user story?
  + No we did not because it was our first sprint
* Did each team member work as scheduled?
  + Yes, indeed.

What went right?

* Each team was able to complete main tasks. For example, installation and reviewing previous documents

How to address the issues in the next sprint?

* How to improve the process?
  + Better communication as a team
* How to improve the product?
  + Getting a better understanding of the requirements

**Sprint 2 Retrospective**

Attendees: Renato Almeida, Roger Michel

Start time: 19:30

End time:20:10

What went wrong?

* Did we do a good job estimating our team's velocity?
  + Yes, we expected that the max number of point would be 20 at max 25.
* Did we do a good job estimating the points (time required) for each user story?
  + Yes, we just get one user story for this spring due to this was huge and high priority for the project
* Did each team member work as scheduled?
  + Yes, indeed.

What went right?

* Each team was able to complete main tasks. For example, Bluemix application setup and services running.

How to address the issues in the next sprint?

* How to improve the process?
  + Schedule of the task in a chronological order
* How to improve the product?
  + Getting a better understanding of the requirements

**Sprint 3 Retrospective**

Attendees: Renato Almeida, Roger Michel

Start time: 19:30

End time:20:10

What went wrong?

* Did we do a good job estimating our team's velocity?
  + We planned as best as we can but the hurricane did not allows of to accomplish the planned work due to storm assignment at work
* Did we do a good job estimating the points (time required) for each user story?
  + Yes, we did.
* Did each team member work as scheduled?
  + We worked as best as the time availability allow us due to the storm assignments at work

What went right?

* We complete most of our planned stories

How to address the issues in the next sprint?

* How to improve the process?
  + We need to have more time talking preparing our spring planning
* How to improve the product?
  + We have been thinking in a couple of solutions like implementing MongoDB as a storage solution instead of relational database

**Sprint 4 Retrospective**

Attendees: Renato Almeida, Roger Michel

Start time: 19:35

End time:20:10

What went wrong?

* Did we do a good job estimating our team's velocity?
  + Yes we did a good estimation based on the team knowledge of the system and technologies in used
* Did we do a good job estimating the points (time required) for each user story?
  + Yes, went a little down the number of points, 3-6, but it was able to be done.
* Did each team member work as scheduled?
  + We work at the cadence that each team member possible do it, due to the different technologies involved on the project and the acquisition of the knowledge of them.

What went right?

* We complete most of our planned stories

How to address the issues in the next sprint?

* How to improve the process?
  + We need to have more working hours, specially for learning technologies used on the project
* How to improve the product?
  + We have been thinking in a couple of solutions like implementing MongoDB as a storage solution instead of relational database

**Sprint 5 Retrospective**

Attendees: Renato Almeida, Roger Michel

Start time: 19:35

End time:20:10

What went wrong?

* Did we do a good job estimating our team's velocity?
  + Yes we plan based on the experience of the previous spring velocity and better knowledge of the team member abilities
* Did we do a good job estimating the points (time required) for each user story?
  + Yes, it was a fair estimation of the amount of hour required
* Did each team member work as scheduled?
  + We work at the cadence that each team member possible can do it, due to the different technologies involved on the project and the acquisition of the knowledge of them.

What went right?

* We complete most of our planned stories

How to address the issues in the next sprint?

* How to improve the process?
  + We need to have more working hours, specially for learning technologies used on the project.
* How to improve the product?
  + Continue working on the product backlog

**Sprint 6 Retrospective**

Attendees: Renato Almeida, Roger Michel

Start time: 19:35

End time:20:10

What went wrong?

* Did we do a good job estimating our team's velocity?
  + Yes we plan based on team member skills
* Did we do a good job estimating the points (time required) for each user story?
  + Yes, it was a fair estimation of the amount of hour required
* Did each team member work as scheduled?
  + We work at the cadence that each team member possible can do it, due to the different technologies involved on the project and the acquisition of the knowledge of them.

What went right?

* We complete most of our planned stories

How to address the issues in the next sprint?

* How to improve the process?
  + Team member need to learn quick the technologies used on the project
* How to improve the product?
  + Continue working on the product backlog

Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents

This is on separate documents specified by github layout.

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. mlab - Used to expose MongoDB as a service. <https://mlab.com/>
5. Apache Maven - java building profile used in our application: <https://maven.apache.org/>