|  |
| --- |
| Brighleaf Software Architecture Document |

**DOCUMENT VERSION 0.1 Draft**

**11-Feb-2019**

**DOCUMENT HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Document Revision Description** | **Document Author** |
| 11-Feb-2019 | 0.1 Draft | First Draft of the Software Architecture Document | Shama Tungare |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**APPROVALS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Approval Date** | **Approved Version** | **Approver Role** | **Approver** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Introduction 4](#_Toc2181076)

[1.1 Purpose 4](#_Toc2181077)

[1.2 Overview 5](#_Toc2181078)

[2 Architecture Goals and Constraints 8](#_Toc2181079)

[3 Architectural Representation 9](#_Toc2181080)

[3.1 Architectural design pattern 9](#_Toc2181081)

[3.2 Sample Login architecture 11](#_Toc2181082)

[3.3 Services for Text Extraction Engine 12](#_Toc2181083)

[3.4 Interaction between services 13](#_Toc2181084)

[3.5 Security between services 13](#_Toc2181085)

[3.6 Spring Session Management 13](#_Toc2181086)

[4 Software Development Methodology 15](#_Toc2181087)

[4.1 Scrum Process 15](#_Toc2181088)

[5 Coding Best Practices 15](#_Toc2181089)

[5.1 Progress Reporting 16](#_Toc2181090)

[5.2 Error Logging 16](#_Toc2181091)

[5.3 Error Display to user 16](#_Toc2181092)

# Introduction

This introduction provides an overview of the entire ***Software Architecture Document* for the Conservation Planning Support System**. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of the **system**.

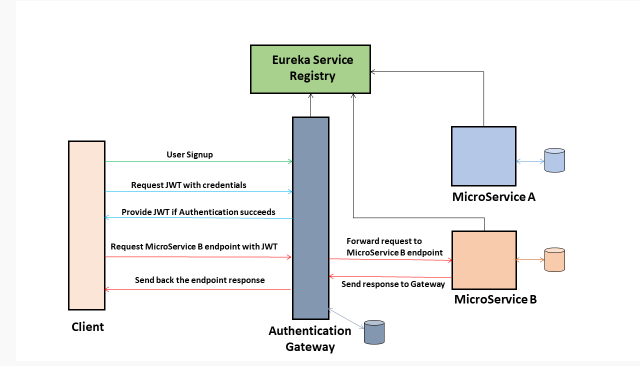
## Purpose

This document provides an architectural overview of the Text Extraction Engine (TEE).

The primary purpose of the Text Extraction Engine is to define rules for extraction and extract attributes according to rules.

This document is intended to capture and convey the significant architectural decisions which have been made in designing and building the system. It is a way by which the systems’ architect and others involved in the project can better understand the problems to be solved and how it will be represented with this system.

The following diagram shows the overall architecture diagram of the Text Extraction Engine.



## Overview

The main modules of Text Extraction Engine are Company Management, User Management, Rule Creation, Execution, Quality Assurance Assignment, QA/Verification and Exporting Results.

Company Management – Super admin can add company, edit company and delete company. When a new company is created, super admin has to add at least one Company Admin to make the company “Active”. Company Admin can then add users with various permissions to the company. When super admin deletes a company, all the transactions for that company shall be deleted. When company is created, super admin has to select the attributes being extracted for the company, levels of QA required by the company and whether that company requires MFA.

User Management – Company admin can add users with various permissions and assign them to a company. He can edit user and delete user.

Rule Creation – User with “Rule Writer” role can create rules. For rule creation, PDF file can be viewed in “pdfviewer” in the left pane of the screen and rule writer can copy text from pdf and paste in the rule creation window. He can add regular expressions to extract entities. Clauses shall be extracted as paragraphs.

Initially entire document shall be divided into sentences and paragraphs. Paragraph splitting shall depend upon the gap between two lines and start of the next line. There are two types of paragraphs: either the first line of the paragraph starts with some indentation or there is gap between last line of previous paragraph and first line of the next paragraph. Based on these co-ordinates, paragraphs shall be determined and used for clause extraction.

If regular expression is given, and the attribute type is “clause”, then matching string shall be the start of the paragraph but if the attribute type is “entity”, then the regular expression shall be executed on the sentences to match. Regular expression will have higher priority over the “before” and “after” attribute text.

PDF Viewer

PDF File Name

Rule

Text before attribute

Text before attribute

Text before attribute

Text before attribute

Text before attribute

And

Or

Or

Or

Or

And

And

And

Text after attribute

Text after attribute

Text after attribute

Text after attribute

Text after attribute

And

Or

Or

Or

Or

And

And

And

Attribute

RegEx

RegEx

Execution – User can select either a single file or multiple files and a rule set to apply on. Every rule from the rule set shall be applied on the selected document(s).

Quality Assurance Assignment – The person who executes the transaction automatically becomes level one QA person. Once he has verified all the attributes, he assigns next level QA person. That person shall receive a mail telling him transaction number for which he has been assigned as a QA person. Once he has verified all the attributes, he assigns next level QA person and so on till all the levels of QA are exhausted.

QA Assignment

QA - 1

QA - 2

Action

QA - 3

Transaction ID

QA / Verification – Extracted attributes shall be shown to the user with text highlighted on the PDF file displayed in “pdfviewer”. If user feels that the extracted text is not correct, he/she can modify the text. Once the user is satisfied with the extraction, he can verify the attribute. Once all the attributes are verified, the document goes for next level of QA. Next level QA person shall be informed about the QA via email. Every change made to the attribute data at every level shall be stored in MongoDB along with user name of QA person.

Exporting Results to Excel – Once all the QA levels are done and all the attributes are verified by the last QA person, the results can be exported to excel sheet. ( Or results can be exported to excel at any level.)

# Architecture Goals and Constraints

The Text Extraction Engine architecture has been designed with the following objectives in mind:

1. To facilitate the rule writing process for the users, to make the rule writing simple and easy to use
2. To extract entities as efficiently and correctly as possible
3. To facilitate company and user configurations
4. To display results along with the document to facilitate the verification method
5. To export the results to excel

The major design and implementation constraints for the system are:

1. Simplicity
2. Flexibility

Each micro-service is an entity in itself and is a self-content unit which does not depend on any other service for its core functionality. It may request other services to do some work for it but will have its own DAO.

# Architectural Representation

## Architectural design pattern

The design pattern used to create the Text Extraction Engine is MVC (Model View Controller) design pattern. The MVC design pattern clearly separates the web application’s behavior, presentation and control. MVC supports rapid and parallel development. With MVC, one programmer can work on the view while other can work on the controller to create business logic of the web application. The modularity of this design pattern allows for easier code reuse, more centralized control, and easier code modification.

The presentation, or view, of the Text Extraction Engine will be implemented in Angular 7. Angular-7 is a client-side technology that provides a powerful way of achieving really effective things in a way that embraces and extends HTML, CSS and JavaScript. Angular-7 is considered to be the most popular framework with essential features.

For backend processing, Spring Boot with Microservices will be used. Using micro-services, Text Extraction Engine will be split into number of microservices which are small, independently versioned, and scalable customer-focused services with specific business goals. They can communicate with each other over standard protocols with well-defined interfaces. Microservices provide a more decentralized approach to building software, they also allow each service to be deployed, rebuilt, redeployed and managed independently.

JSON Web Token (JWT) is an encrypted token which defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA. By encrypting JSONs, the data will be hidden from other parties and only those which are intended to use it will verify the integrity of the claims contained within it. Hence the information which is going over the network, will be encrypted and tamper proof.

8761

Browser

Angular Application

Spring Security

Microservice 1

Spring Boot

Mysql

Microservice 1

Spring Boot

Mysql

Microservice 1

Spring Boot

Mysql

8080

Eureka

Spring Cloud Config

## Sample Login architecture

Login Page

Login clicked

Forgot password clicked

Show user the list of companies

Only one company, select that company

Get all companies

Get all roles

Spring security authentication

Send link to email

Get email

MFA?

Select from list

Send OTP

Authenticate

One company?

Show tabs according to roles

## Services for Text Extraction Engine

Eureka Service Registry

Text Extraction Engine

Login Service

Company Service

User Service

Annotation Service

Execute Service

Results Service

Reports Service

Chunker Service

Document Type Service

Attribute Service

OTP Service

MySQL Service

MongoDB Service

AUTHORIZATION  
  
GATEWAY

Email Service

## Interaction between services

Attribute Service

Login Service

Execute Service

Document Type Service

MySQL Service

MongoDB Service

User Service

Company Service

Annotation Service

Email Service

OTP Service

Result Service

Report Service

## Security between services

Since Micro-service architecture will be used for development, there is a need to protect unauthorized access to micro-services. OAuth should be used to authorize the access. OAuth does not share password data but instead uses authorization tokens to prove an identity between consumers and provider. OAuth is an authentication protocol that allows approval of one application interacting with another without giving away password.

## Spring Session Management

* Spring Session keeps application logic and session management logic separate.
* Spring Session keeps information in the database
* Since session data is stored in the database, even if the application crashes, user session data is not lost and the application can pick up user session from the database and start from that point.
* It is easy to switch between session storage. Just by changing the configuration, we can switch from using JDBC to Redis.

In order to save session objects to MySQL database, no code has to be written. Spring Boot provides this functionality out of the box by specifying the following configuration property

*spring.session.store-type=jdbc*

and providing MySQL properties in the *application.properties* file.

*spring.datasource.driver-class-name=com.mysql.jdbc.Driver*

*spring.datasource.url=jdbc:mysql://localhost/springSession?createDatabaseIfNotExist=true&autoReconnect=true*

*spring.datasource.username=*

*spring.datasource.password=*

*spring.datasource.initialization-mode=always*

Browser 1

Browser 2

Browser 3

Session1

Session 3

Session 2

Server

Data Store

# Software Development Methodology

Agile methodology will be followed for software development. Scrum which is a subset of Agile will be followed. Creation of scrum project, creating sprints and managing them will be done in JIRA.

## Scrum Process

Daily scrum meeting

Scrum

Product Backlog

Prioritize product features

Sprint Backlog

Assign features to sprint

Expanded features

Sprint Cycle

Product Increment

JIRA shall be used to as a tool to handle the development using Agile methodology and GitHub shall be used for source control. “Process Document” shall be followed for all the development related processes.

# Coding Best Practices

* Development, test and staging environments must be set up to function with the lowest possible privilege so that production will also work with lowest possible privileges
* Database access should be through parameterized queries to allow all table access to be revoked (i.e. select, drop, update, insert, etc) using a low privilege database account. This account should not hold any SQL roles above "user" (or similar)
* When the application is satisfied that a user is authenticated, associate the session ID with the authentication tokens, flags or state.

Web Service, like other distributed applications, require protection at multiple levels:

* Sensitive Data should never be passed in the URL. Never use username, password or session token in a URL, these values should be passed to Web Service via the POST method.
* The server needs to be confident who it is talking to and what the clients are entitled to.
* The clients need to know that they are talking to the right server, and not a phishing site
* System message logs should contain sufficient information to reliably reconstruct the   
  chain of events and track those back to the authenticated callers

Since Micro-service architecture will be used for development, there is a need to protect unauthorized access to micro-services. OAuth should be used to authorize the access. OAuth does not share password data but instead uses authorization tokens to prove an identity between consumers and provider. OAuth is an authentication protocol that allows approval of one application interacting with another without giving away password.

Session management strategy –

* [Session ID](https://searchsoftwarequality.techtarget.com/definition/session-ID) should be stored in a secure [cookie](https://searchsoftwarequality.techtarget.com/definition/cookie)
* Session ID should be encrypted
* Cookie should expire when the session is invalidated or the browser is closed
* Application should force the browser to be closed when the user logs out or session is invalidated.

## Progress Reporting

Progress bar will display the progress of execution. If batch processing is selected, for every file, every operation being performed will be shown to the user so that the user does not get confused about the execution.

## Error Logging

Errors will be logged in the log files for further analysis. While coding, a care shall be taken that every log is meaningful and shall lead to trace back the error. Along with errors, some DEBUG or INFO logs also shall be added to give more informative picture of the execution of the application.

## Error Display to user

Any error shall take user to a standard error page where user will be informed of the error.