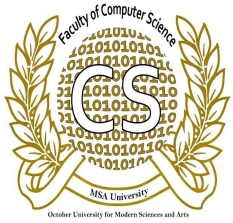

Software Design Document for project Forensics Linguistics and Authorship Attribution using Stylometry

Yousef Mohamed - 184367
Supervised by: Dr. Wael Gomaa

1 Introduction

1.1 Purpose

This software design document describes the architecture and system design of Forensics Linguistics and Authorship Attribution using Stylometry.

1.2 Scope

The project is intended to create a system for parsing and analysing text for the extraction of linguistic and stylometric features to be used in authorship attribution performed by linguistic analysts.

1.2.1 Goals

- Reduce analysis time.
- Ease exploration of new features.
- Authorship attribution.

1.2.2 Objectives

- Intuitive interface.
- Bulk analysis.
- Features management.
- Analysis modification.
- Statistical and machine learning methods for authorship attribution.

1.3 Overview

The system is meant to be used by linguistic professionals to extract features from text and receive a preliminary analysis on the whole corpus' author.

2 System Overview

The system will receive the corpus from the user in the form of a compressed file, and the features required for extraction, which will then be processed for the required output and analysed for a disputed text if necessary.

3 System Architecture

3.1 Architectural Design

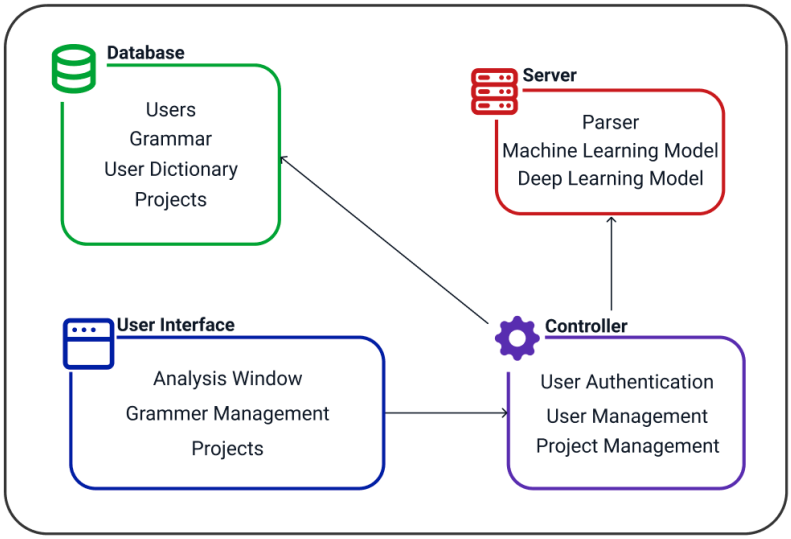

Figure 1: Architectural Design

3.2 Decomposition Description

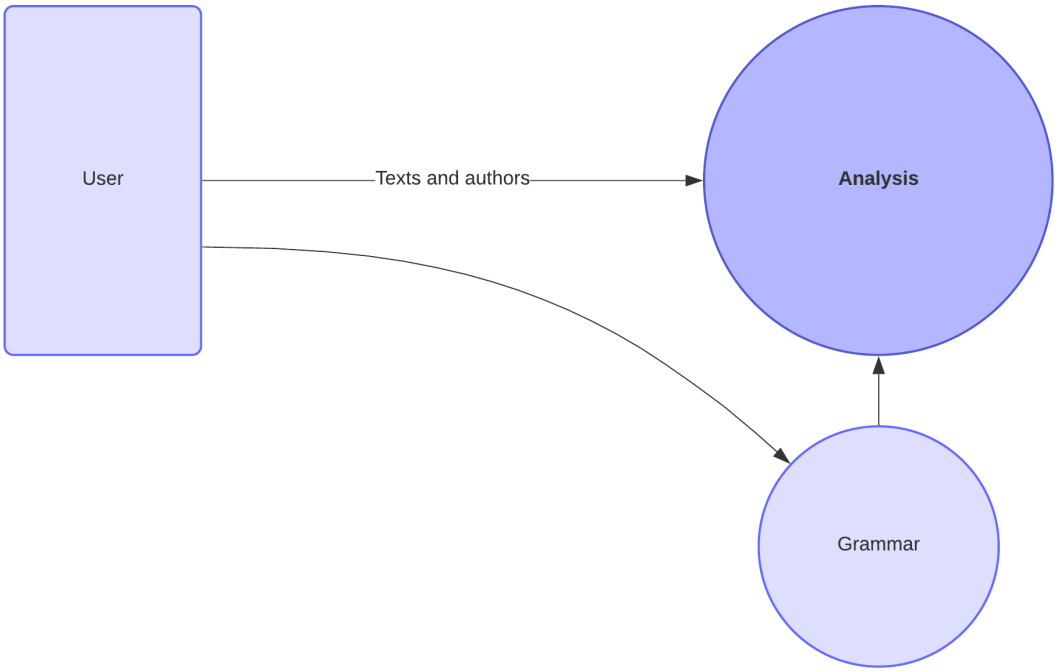

Figure 2: Data Flow Diagram

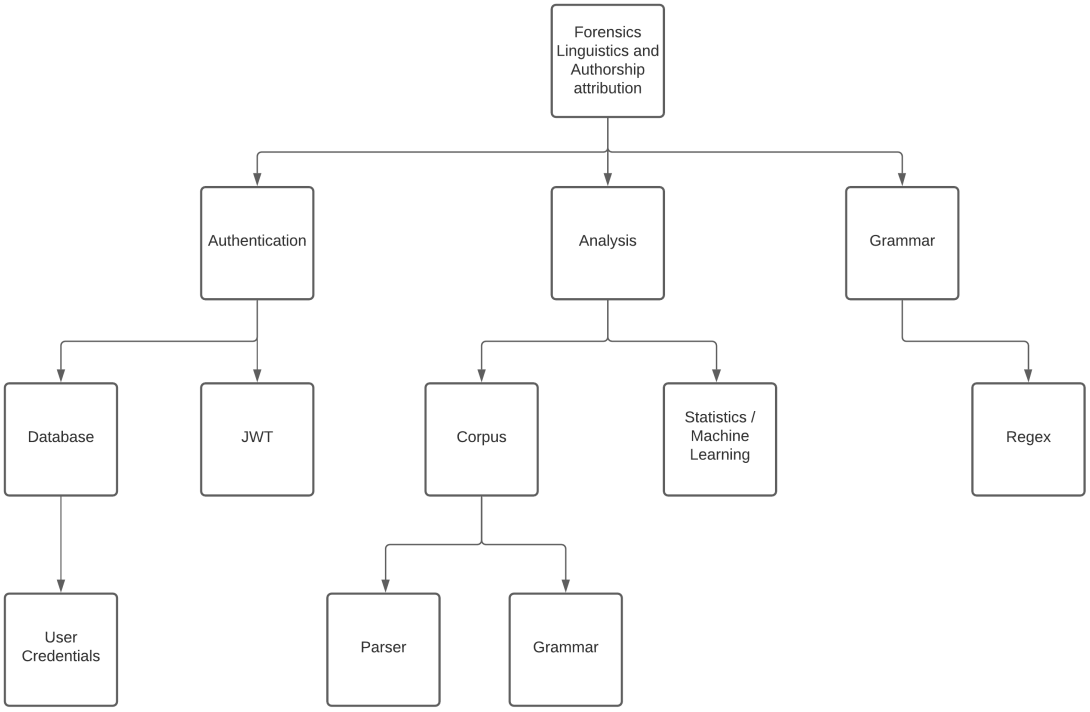

Figure 3: Functional Decomposition Diagram

3.3 Design Rationale

The principle behind the architecture used is separation of concerns, the main components of the system beside the user interface and the database are in the server and the controller. The server and the controller are two different entities and are deployed independently from each other. The reason for the separation is that they use different technologies with different architectures and different deployment needs. The server hosts the parser and the machine learning model of the system, which run on python and have a very specific task that may be modified but no extra features may be added. The controller on the other hand handles all the logistics of the system, by managing the users, grammar and projects by providing a graphQl API that is completely expandable.

4 Data Design

4.1 Data Description

1. Corpus
    - The corpus is uploaded as a compressed file containing folders of the texts, with the authors being the title of each folder. The file is extracted and turned into JSON format before parsing, then each element of the JSON object is parsed and returned as a parsed array that is saved in a noSQL database completely dedicated to storing the corpus and their analysis results.
2. Grammar
    - Grammar data is saved in a SQL database as a REGEX string that is imported during parsing of text, and editing of features. The grammar is created using a tool specifically made for this system.
3. Users
    - User data is saved in a SQL database. The user data includes their credentials, projects, and grammar.
4. Projects
    - Project data is split into multiple tables in a SQL database. The data includes collaborators, corpus id, and used features.

4.2 Data Dictionary

- createGrammar(featureName: string, regex: string)
- formatText(file: zip)
- parse(corpus: {[authorName]: string[][], disputed: string[][]}, grammar: string[])
- updateAnalysis(wordIdx: number, featureId: number)

- updateGrammar(featureId: number, regex: string)
- uploadText(file: zip)

5 Component Design

5.1 Authentication

```
username = input(username)
password = input(password)
userId = Select userId from database where username = username & password = passw
if userId
    response.send(userId, username, JWT) // JWT Token is auto generated
```

5.2 Analysis

```
corpus = input(corpus)
grammar = input(grammar)
formattedCorpus = format(corpus)
parsedCorpus = parse(formattedCorpus, grammar)
response.send(parsedCorpus)
```

5.3 Grammar

```
featureName = input(featureName)
regex = input(regex)
res.send(Insert Into database (featureName, regex), Values (featureName, regex))
```

6 Human Interface Design

6.1 Overview of User Interface

- Creating new grammar: The user will open the grammar tab and press on the 'New Feature' button, they will then be redirected to a page where they are presented with a tool that allows them to write the grammar which will be converted into regex for the parser.
- Editing grammar: On the grammar tab, the user will be presented with a list of all the features they created, which they can either delete or edit. If the user chooses to edit, they will be redirected to the grammar creation tool page, but with the existing feature characteristics already set for them to change.
- Creating a project: The user will be presented with a wizard involving a number of steps, they will first upload the corpus in a specified format which will be demonstrated on the wizard, then they will choose the needed features from the grammar they created, then all the data will be sent to the server and the user will be redirected to the workspace page where they can see the result of the analysis.

- Editing the analysis: The user can hover on any word in the workspace, set as a feature or not, and change its feature association as they please by selecting from the list of features they chose at the beginning of the project. The edit will also reflect on the statistical and the authorship attribution results accordingly.

6.2 Screen Images

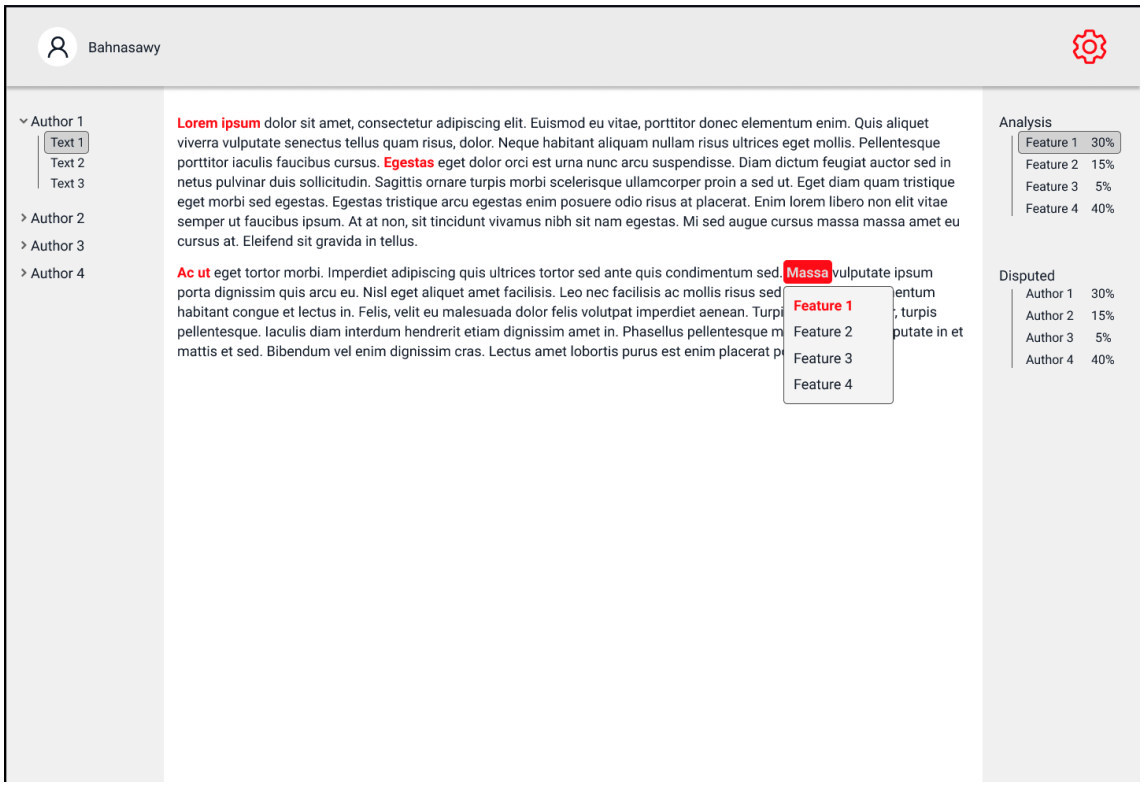

Figure 4: Screen 1

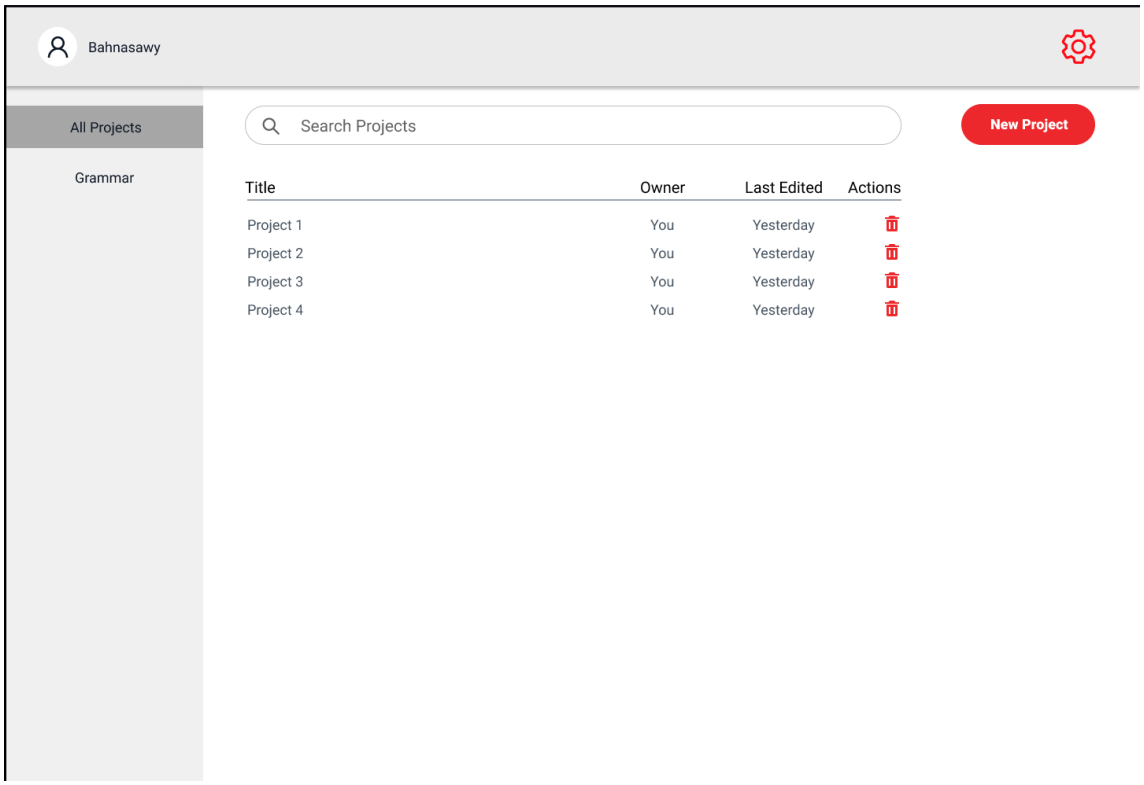

Figure 5: Screen 2

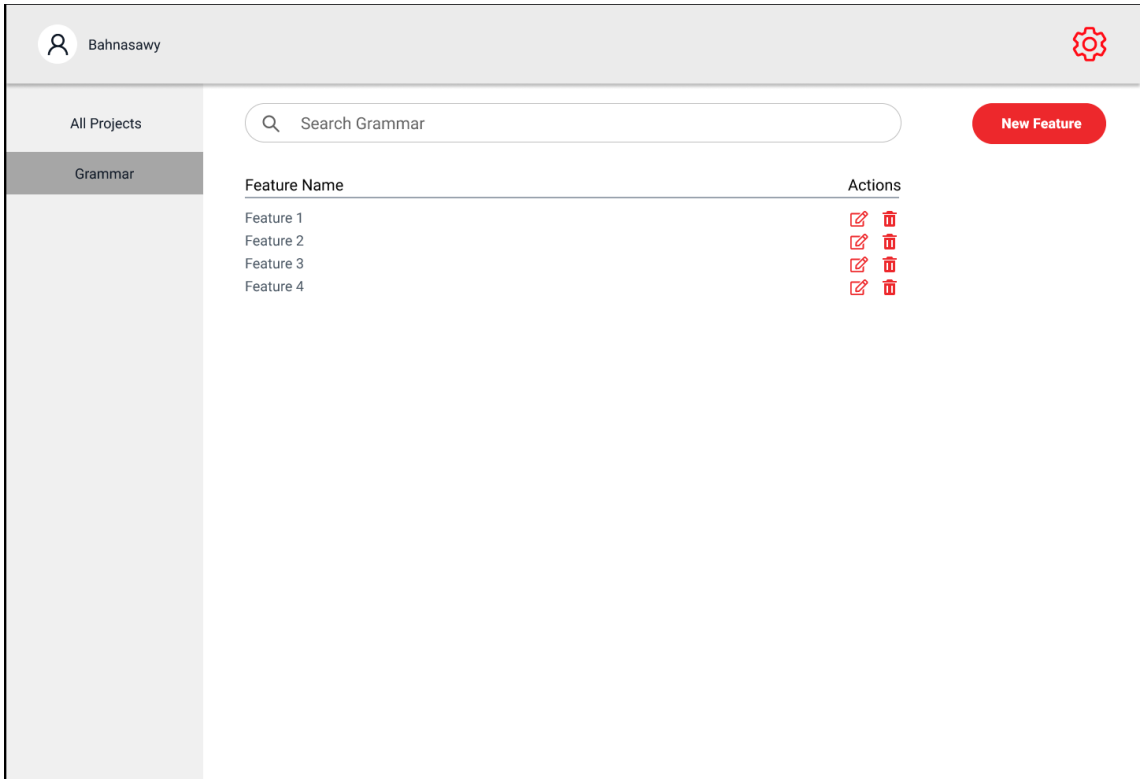

Figure 6: Screen 3

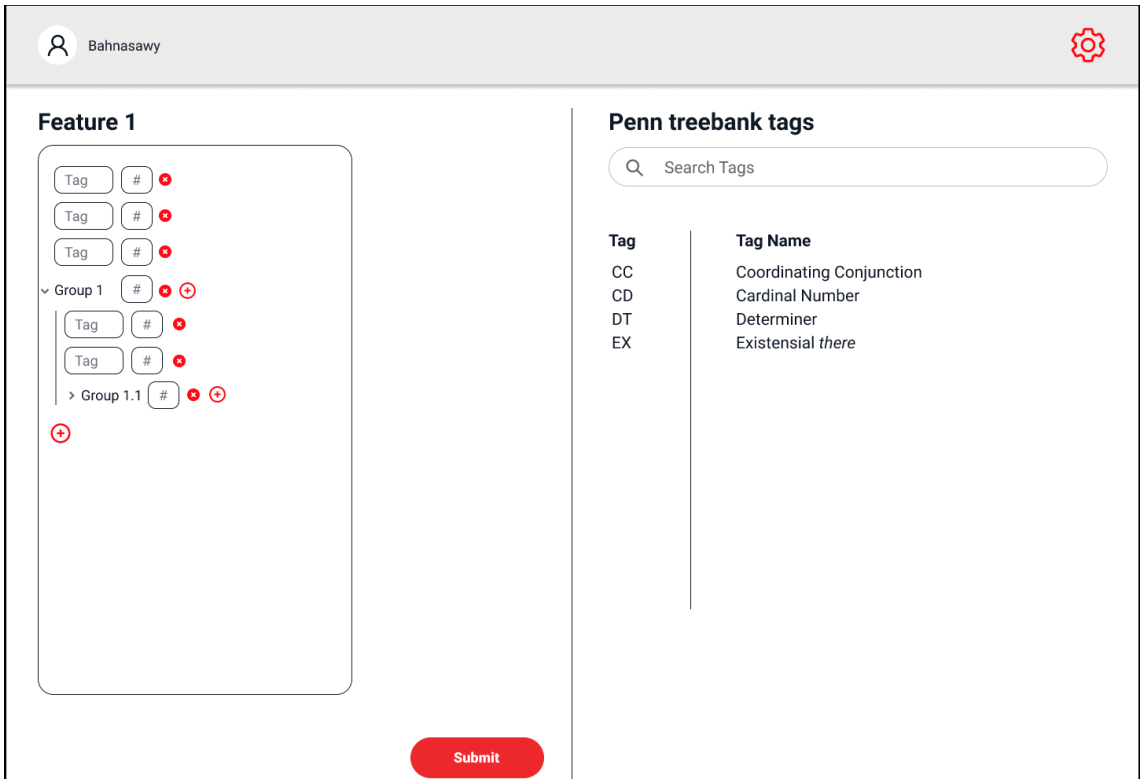

Figure 7: Screen 4

7 Requirements Matrix

System Component	Functional Requirement
Grammar	3, 4
Analysis	1, 2, 5, 6, 7, 8
Authentication	9, 10, 11