Forensics Linguistics and Authorship Attribution using stylometry

by

Yousef Mohamed Abdullah - 184367

A dissertation submitted in partial fulfillment of the
requirements for the degree of
Bachelor of Computer Science(SE programme)

in the

Faculty of Computer Science
of the
October University for Modern Sciences and Arts (MSA), EGYPT

Graduation Projects advisor:
Dr. Wael Gomaa

(July 2021)

Abstract

Linguistic professionals have been manually analyzing and extracting sylometric features from texts for a long time, which is a tedious and time-consuming process. Previous software have attempted to aid linguists to search for said features, however most of them are either expensive or were counter-intuitive to use. This project aims to fix these issues by offering an easy to use interface with a simple but powerful set of tools(Regular Expression, NLP), using Obama's presidential speeches for validation, in order to achieve the desired objectives with high accuracy.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

In written text, each person has a unique style of writing similar to a fingerprint, and in order to identify that writing style linguists have devised methods and researched features that allows those styles to be differentiated and identified from one another.

1.1.1 Background

Stylometry is the statistical methods used to analyze and differentiate between different literary styles between one author and another. A stylometric analysis is be conducted using NLP techniques and libraries due to the large and extensive tool set and previous research and applications achieved.

1.1.2 Motivation

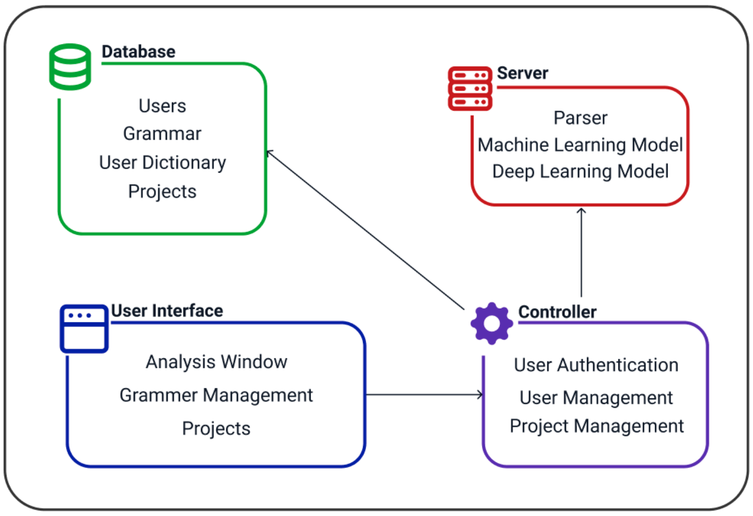
The process of identifying a writing's features is a difficult task that involves excessive manual labor and time analyzing each sentence and clause several times to extract all of the features.

1.1.3 Problem Definitions

The main problems are:

- Converting the stylometric features to machine readable form.
- Applying those features with NLP techniques for analysis.
- Providing a functional interface that allows for complete utilization of the system and its features.

1.2 Project Description



1.2.1 Scope

The scope of the project lies is limited to NLP, text analysis, and statistical operations.

1.2.2 Project Overview

- Document and project storage.
- Analysis server.
- User interface.

Chapter 2

Background

2.1 Project Context

The project is intended to be used by linguistics professionals to extract stylometric features and analyze written text to differentiate between authors.

- NLP: Natural Language Processing is a subfield of linguistics that is concerned with providing means for a machine to comprehend and analyze human language.
- Regular Expression: It is a simple tool used to find patterns in written text.
- Literary style: It is an approach to writing that is unique to each writer through the use of different vocabulary, grammar, techniques, etc..

Chapter 3

Specification - (SRS)

3.1 Introduction

3.1.1 Purpose of this document

The purpose of this document is to define the requirements to be met by the system to be classified as successful and functional.

3.1.2 Scope of this document

The system is wholly engineered and developed by me, with the supervision of Dr. Wael Gomaa. The users are the professors of MSA University's Faculty of Languages.

3.1.3 Overview

The system will take a collection of texts, organized by author, to be analyzed by the system through a set of stylometric and linguistic features, and will return the extracted features embedded in the text with a statistical report of the analysis. The system will also require a selection of predefined grammar sets defined by the user and/or the community.

3.1.4 Business Context

The MSA University Faculty of Languages is the organization supporting and contributing to the project.

3.2 General Description

3.2.1 Product Functions

1. Create new project.
2. Upload texts.
3. Select from predefined grammar sets.
4. Create new grammar sets.
5. Extract stylistic and linguistic features.
6. Statistical analysis.
7. Highlight the extracted features.
8. Modify analysis results.
9. Login
10. Logout
11. Signup

3.2.2 User Characteristics

The users will be linguistic professionals that can verify the output of the system, and are able to evaluate and verify the results of the analysis.

3.2.3 User Problem Statement

The main problem the system is facing is the time consuming labor involved in extracting those linguistic features and their analysis.

3.2.4 User Objectives

- Identify features of the text.
- Modify the analysis results.
- Identify the disputed text's author.
- Create custom grammar for new features.

3.3 Functional Requirements

Table 3.1: Create New Project

Function Name	Create New Project
Description	The user will create a new project that will host the documents of the same context to be analyzed.
Critical	This requirement is the opening point of the system, so the system cannot work without it.
Technical issues	None.
Risks	None.
Dependencies with other requirements	None.
Precondition	Starting state.
Post-Condition	Awaiting documents to be uploaded.

Table 3.2: Upload Texts

Function Name	Upload Texts
Description	The user will upload a collection of texts with each group of texts put in a separate folder with the label of the folder being the name of the author, and the disputed text if available.
Critical	This requirement is the opening point of the project, so the system cannot work without it.
Technical issues	Detecting the user uploaded the texts in the proper format.
Risks	None.
Dependencies with other requirements	’Create New Project’
Precondition	A new project is created.
Post-Condition	Awaiting the features to be explored.

Table 3.3: Select from predefined grammar sets

Function Name	Select from predefined grammar sets
Description	The user will select a number of grammar sets that the text will be analyzed for.
Critical	System can’t advance without it.
Technical issues	None.
Risks	None.
Dependencies with other requirements	None.
Precondition	Texts have been uploaded.
Post-Condition	Awaiting analysis results.

Table 3.4: Create new grammar sets

Function Name	Create new grammar sets
Description	The user will be presented with an interface that will allow them to define their own grammar to apply on the corpus. The generated grammar will also be saved to the user's account.
Critical	Optional.
Technical issues	Ensuring that the grammar is of proper and valid format.
Risks	None.
Dependencies with other requirements	None.
Precondition	Grammar sets have been selected.
Post-Condition	Awaiting analysis of the corpus.

Table 3.5: Extract stylistic and linguistic features

Function Name	Extract stylistic and linguistic features
Description	The system will receive the user's input and proceed to make the required analysis and return the result.
Critical	Critical, system can't function without it.
Technical issues	None.
Risks	None.
Dependencies with other requirements	'Create New Project', and 'Upload texts', 'Select from predefined grammar sets'
Precondition	Grammar is selected.
Post-Condition	Feature extraction is complete and returned to the user.

Table 3.6: Statistical analysis

Function Name	Statistical analysis.
Description	The system will apply statistical analysis to the extracted features received from the parser.
Critical	Critical, system can't function without it.
Technical issues	None.
Risks	None.
Dependencies with other requirements	'Extract stylistic and linguistic features'
Precondition	Features are extracted.
Post-Condition	Statistical analysis is applied and ready for modification.

Table 3.7: Highlight the extracted features

Function Name	Highlight the extracted features.
Description	The system will highlight the extracted features in the work space section of the project and offer methods of editing of the results.
Critical	Not critical.
Technical issues	None.
Risks	None.
Dependencies with other requirements	'Extract stylistic and linguistic features'
Precondition	Features are extracted.
Post-Condition	System awaiting features modification.

Table 3.8: Modify analysis results

Function Name	Modify analysis results.
Description	The user can hover over a word in the work space (highlighted or not) and edit its feature association.
Critical	Not critical.
Technical issues	None.
Risks	None.
Dependencies with other requirements	'Highlight extracted features'
Precondition	Features are highlighted in the work space.
Post-Condition	Modifications are applied to the work space and the statistical analysis.

Table 3.9: Login

Function Name	Login.
Description	User is logged in to system using their credentials.
Critical	Critical.
Technical issues	None.
Risks	None.
Dependencies with other requirements	None.
Precondition	User has no access to the system.
Post-Condition	User is greeted with the home screen.

Table 3.10: Logout

Function Name	Logout.
Description	User is logged out of the system and is greeted with the homepage.
Critical	Critical.
Technical issues	None.
Risks	None.
Dependencies with other requirements	None.
Precondition	User has access to the system.
Post-Condition	User is greeted with the Login screen.

Table 3.11: Sign-up

Function Name	Sign-up.
Description	User creates a new account using their credentials.
Critical	Critical.
Technical issues	None.
Risks	None.
Dependencies with other requirements	None.
Precondition	User has no account registered in the system.
Post-Condition	User account is registered and they are greeted with the system's homepage.

3.4 Interface Requirements

3.4.1 User Interfaces

3.4.1.1 GUI

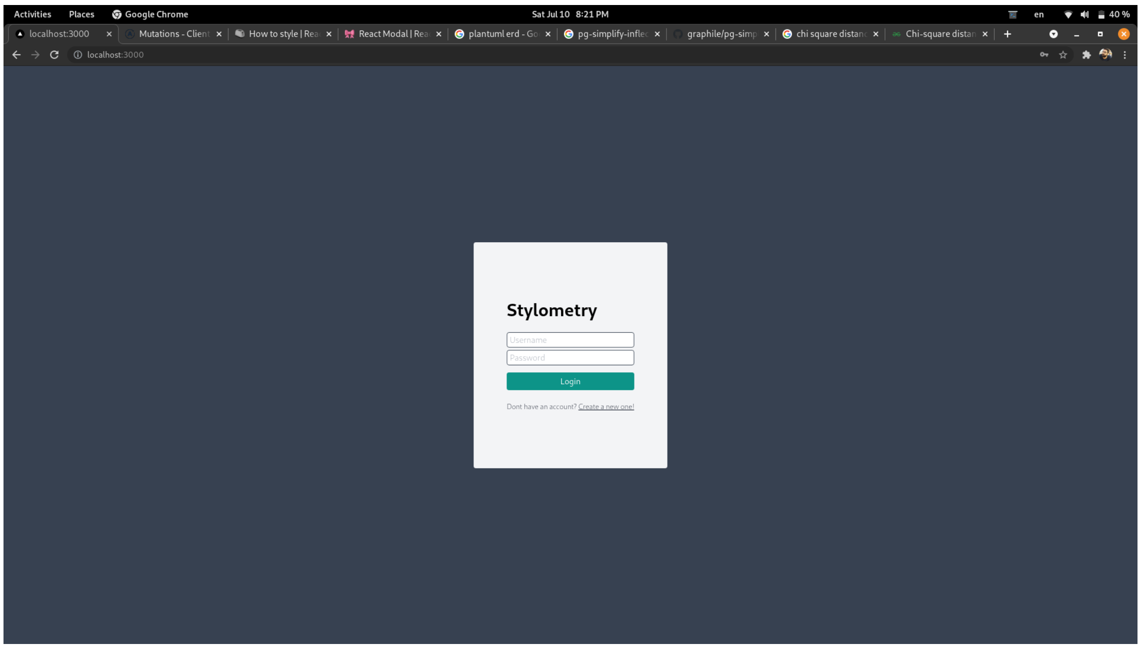
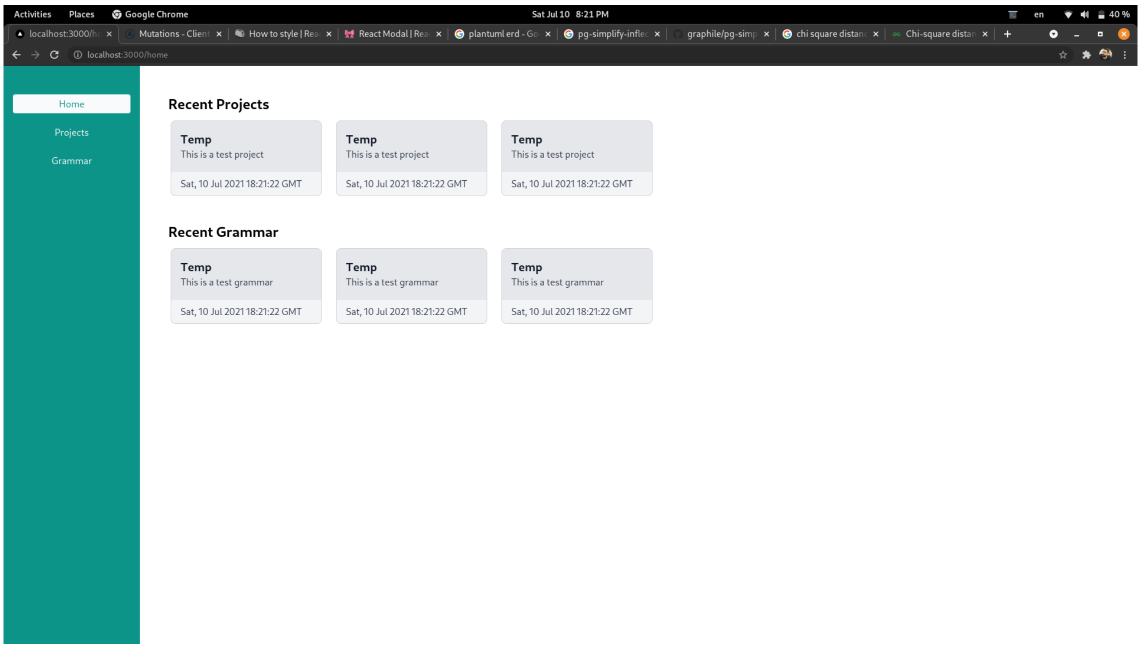


Figure 3.1: Login



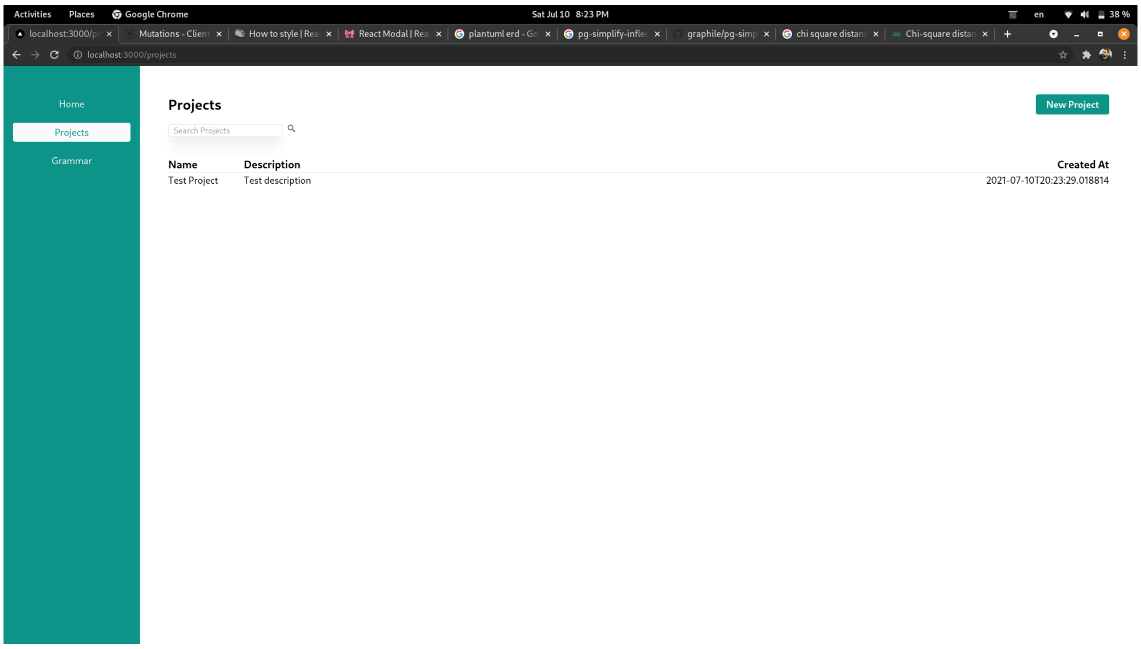
Recent Projects

Temp	Temp	Temp
This is a test project	This is a test project	This is a test project
Sat, 10 Jul 2021 18:21:22 GMT	Sat, 10 Jul 2021 18:21:22 GMT	Sat, 10 Jul 2021 18:21:22 GMT

Recent Grammar

Temp	Temp	Temp
This is a test grammar	This is a test grammar	This is a test grammar
Sat, 10 Jul 2021 18:21:22 GMT	Sat, 10 Jul 2021 18:21:22 GMT	Sat, 10 Jul 2021 18:21:22 GMT

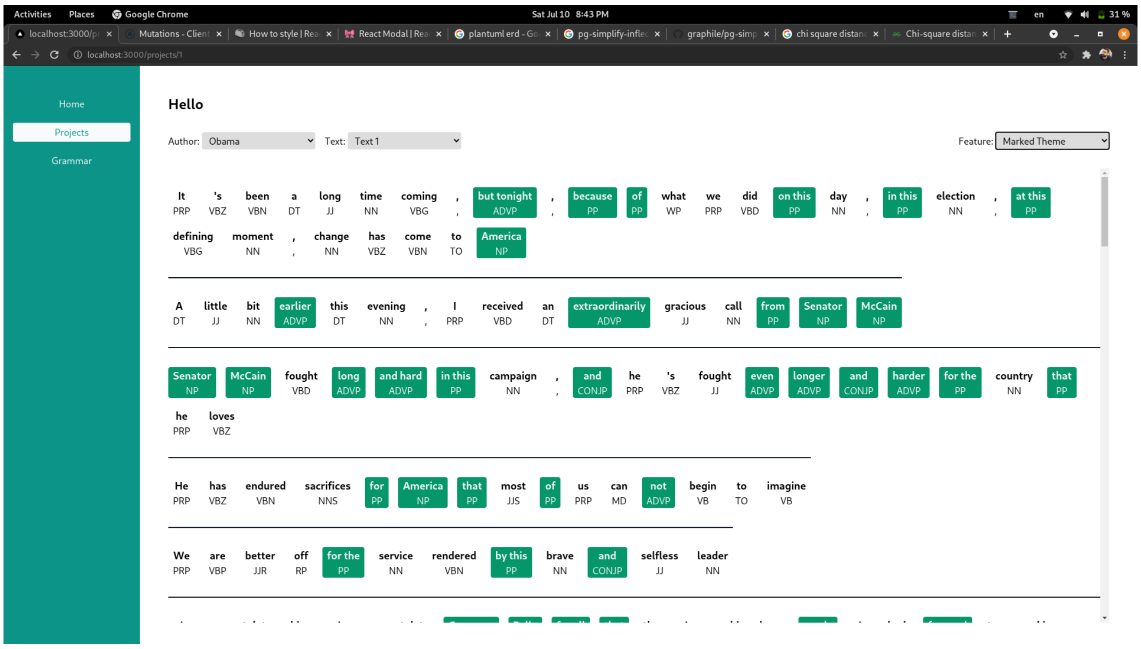
Figure 3.2: Home



Projects

Name	Description	Created At
Test Project	Test description	2021-07-10T20:23:29.018814

Figure 3.3: Projects



Hello

Author: Obama Text: Text 1 Feature: Marked Theme

It	's	been	a	long	time	coming	,	but tonight	,	because	of	what	we	did	on this	day	,	in this	election	,	at this
PRP	VBZ	VBN	DT	JJ	NN	VBG	,	ADVP	,	PP	PP	WP	PRP	VBD	PP	NN	,	PP	NN	,	PP

defining	moment	,	change	has	come	to	America
VBG	NN	,	NN	VBZ	VBN	TO	NP

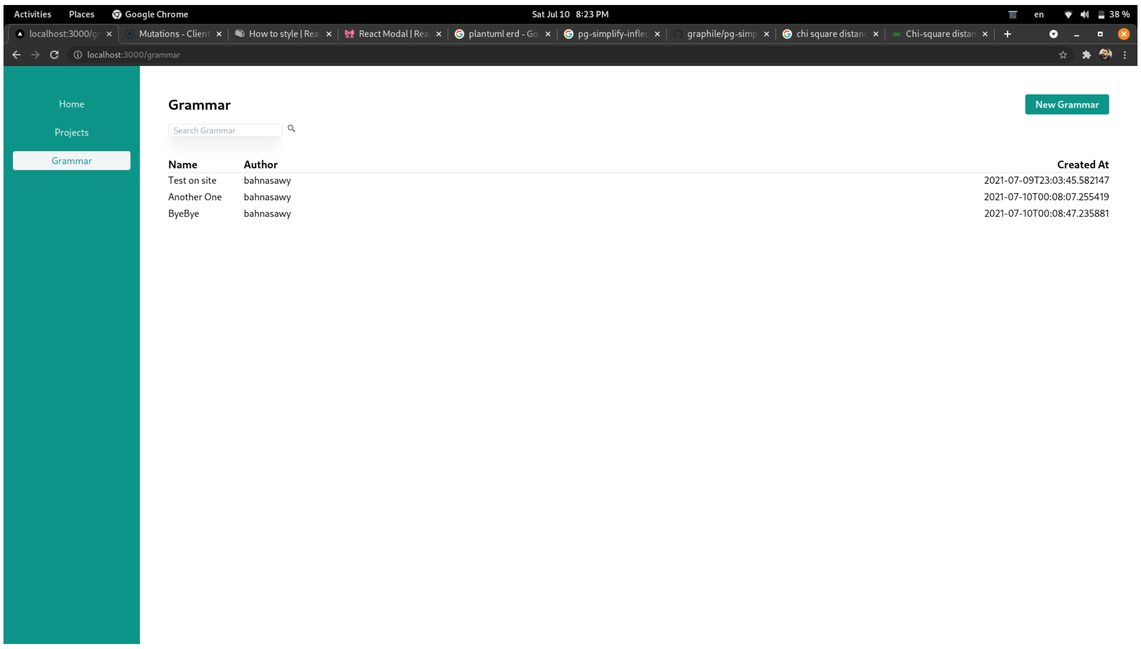
A	little	bit	earlier	this	evening	,	I	received	an	extraordinarily	gracious	call	from	Senator	McCain
DT	JJ	NN	ADVP	DT	NN	,	PRP	VBD	DT	ADVP	JJ	NN	PP	NP	NP

Senator	McCain	fought	long	and hard	in this	campaign	,	and	he	's	fought	even	longer	and	harder	for the	country	that	he	loves
NP	NP	VBD	ADVP	ADVP	PP	NN	,	CONJP	PRP	VBZ	JJ	ADVP	ADVP	CONJP	ADVP	PP	NN	PP	PRP	VBZ

He	has	endured	sacrifices	for	America	that	most	of	us	can	not	begin	to	imagine
PRP	VBZ	VBN	NNS	PP	NP	PP	JJS	PP	PRP	MD	ADVP	VB	TO	VB

We	are	better	off	for the	service	rendered	by this	brave	and	selfless	leader
PRP	VBP	JJR	RP	PP	NN	VBN	PP	NN	CONJP	JJ	NN

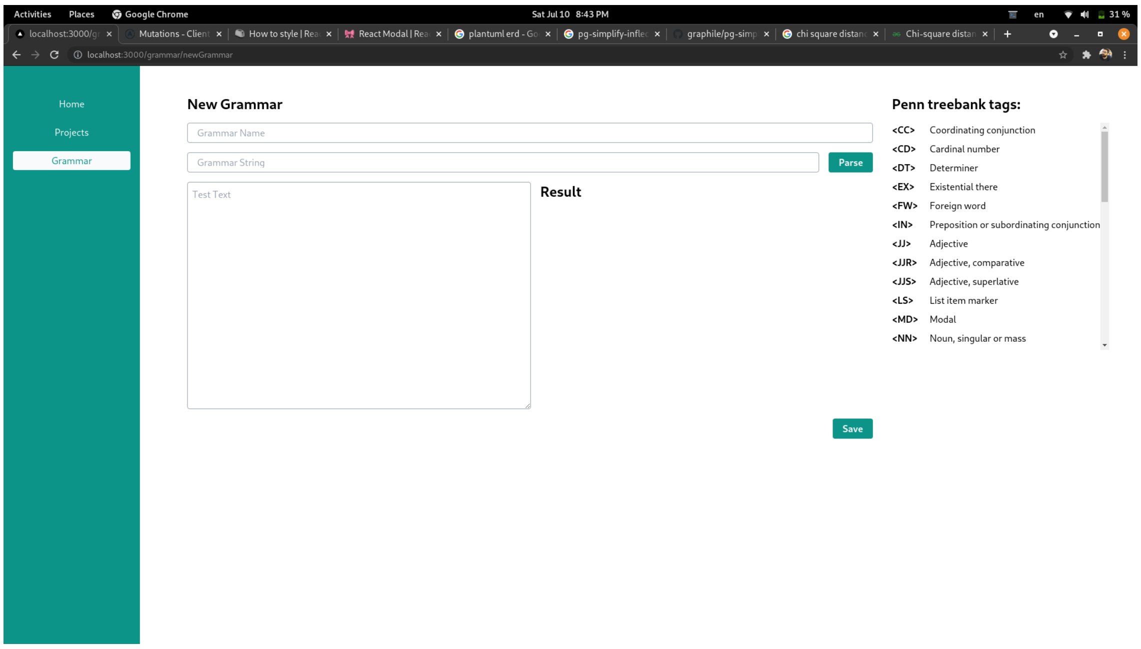
Figure 3.4: Project



Grammar

Name	Author	Created At
Test on site	bahnasawy	2021-07-09T23:03:45.582147
Another One	bahnasawy	2021-07-10T00:08:07.255419
ByeBye	bahnasawy	2021-07-10T00:08:47.235881

Figure 3.5: Grammars



<CC>	Coordinating conjunction
<CD>	Cardinal number
<DT>	Determiner
<EX>	Existential there
<FW>	Foreign word
<IN>	Preposition or subordinating conjunction
<JJ>	Adjective
<JJR>	Adjective, comparative
<JJS>	Adjective, superlative
<LS>	List item marker
<MD>	Modal
<NN>	Noun, singular or mass

Figure 3.6: Grammar

3.5 Design Constraints

No constraints provided.

3.6 Other non-functional attributes

3.6.1 Security

The system will use JWT (JSON Web Token) for user authentication.

3.6.2 Reliability

The system will be deployed as web services on AWS which offers a 99.99% availability guarantee.

3.6.3 Maintainability

The system will be utilizing the micro-services architecture ensuring smaller and more maintainable code.

3.6.4 Portability

The system will be web based, so it is completely portable.

3.6.5 Extensible

The system is using the micro-services architecture, so it is easily extensible.

3.6.6 Re-usability

The system is using the micro-service architecture, and utilizing the functional programming paradigm, so any bit of code can be reused.

3.7 Operational Scenarios

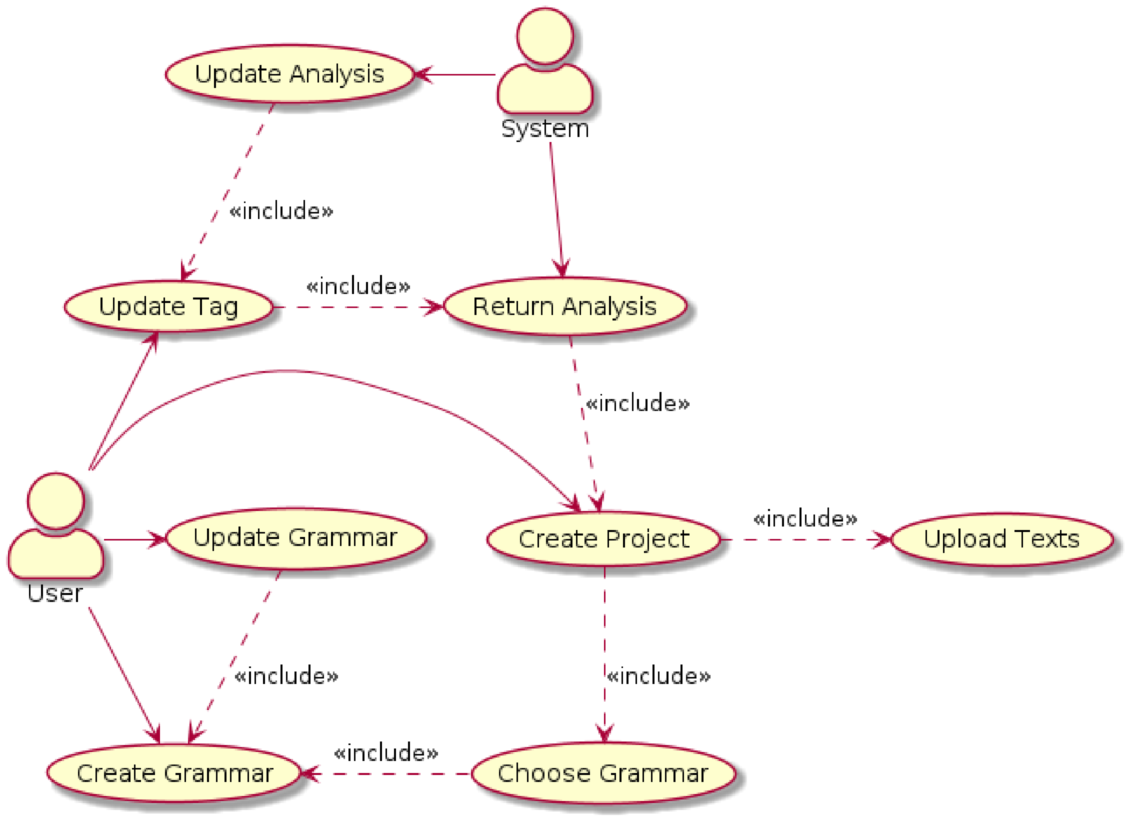


Figure 3.7: Use case Diagram

3.8 Preliminary Schedule Adjusted

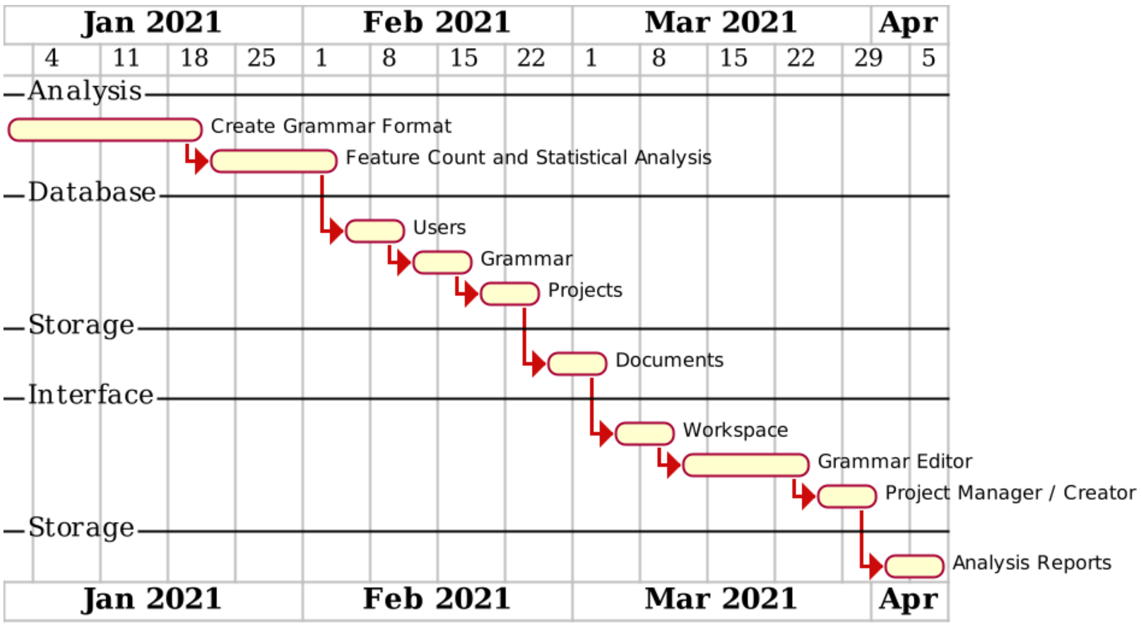


Figure 3.8: Gantt Chart

Chapter 4

Design

4.1 Introduction

4.1.1 Purpose

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

4.1.2 Scope

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

4.1.2.1 Goals

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

4.1.2.2 Objectives

- Intuitive interface.

- Bulk analysis.
- Features management.
- Analysis modification.
- Statistical and methods for authorship attribution.

4.1.3 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.

4.2 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 analyzed for a disputed text if necessary.

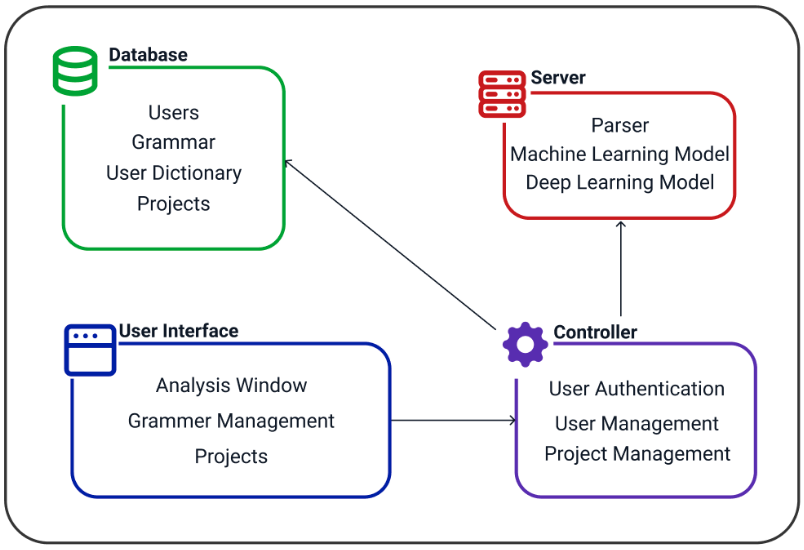


Figure 4.1: Architectural Design

4.3 System Architecture

4.3.1 Architectural Design

4.3.2 Decomposition Description

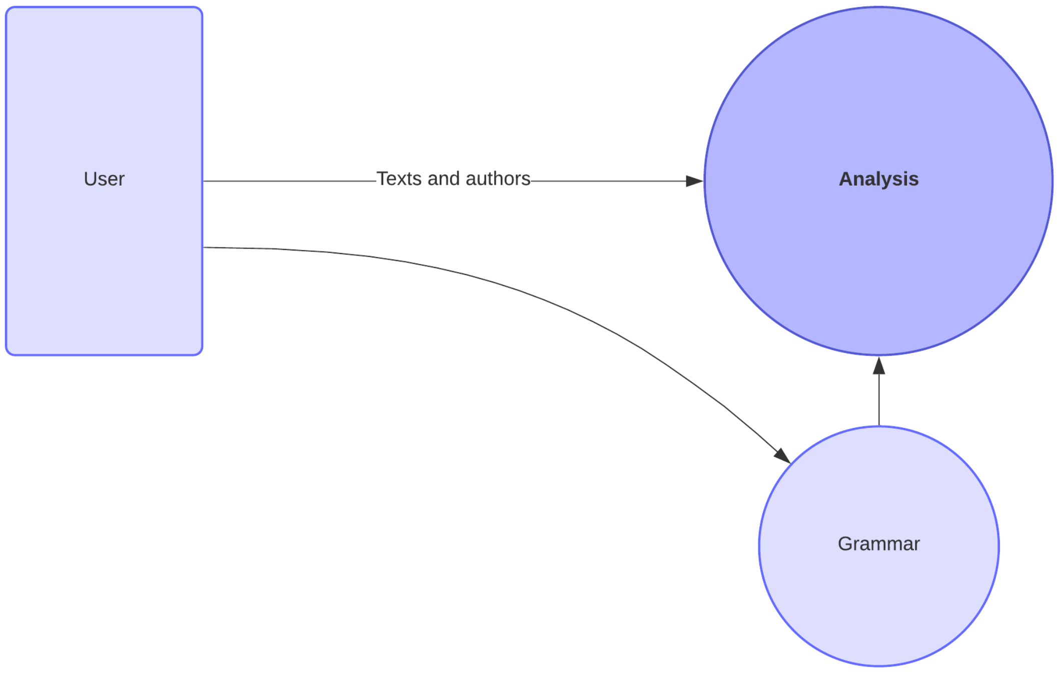


Figure 4.2: Data Flow Diagram

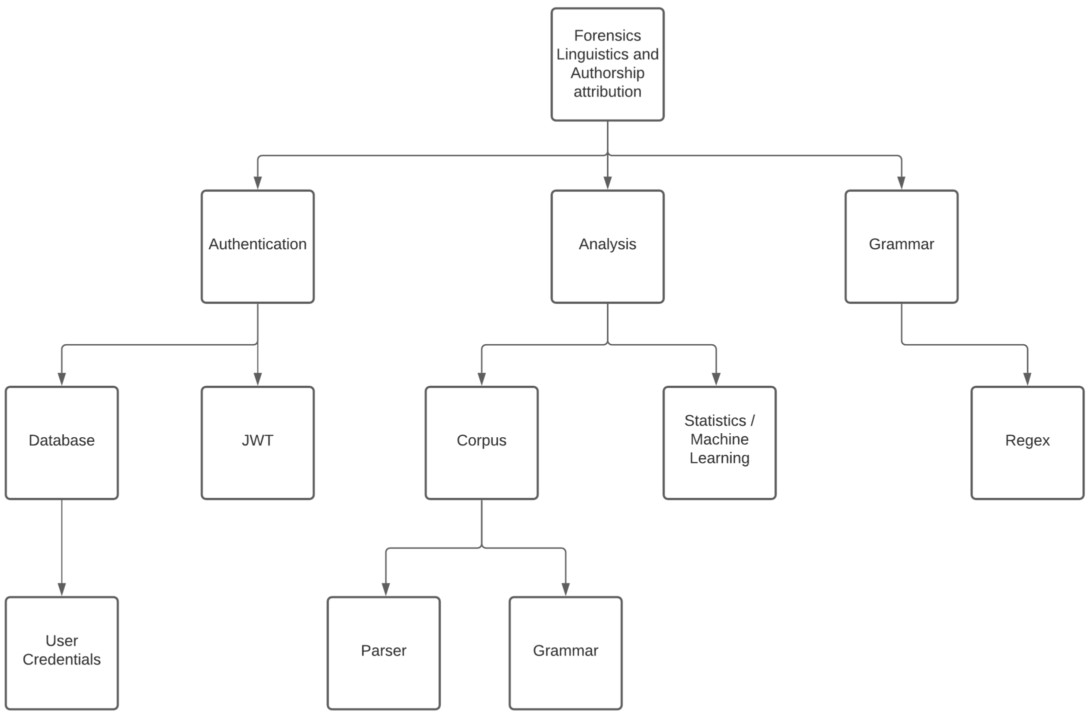


Figure 4.3: Functional Decomposition Diagram

4.3.3 Design Rationale

The principle behind the architecture used is separation of concerns, the main component of the system besides the user interface and the database is the server. The server, user interface, and database are completely separate entities that are deployed independently. The reason for the separation is that they use different technologies with different architectures and different deployment needs. The server hosts the text parser, which runs on python and has a very specific task that may be modified but no extra features may be added. The database exposes a graphQL API to query and manipulate the user data. The interface is the coordinator of all the processes of the system.

4.4 Data Design

4.4.1 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.

4.4.2 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)

4.5 Component Design

4.5.1 Authentication

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

4.5.2 Analysis

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

4.5.3 Grammar

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

4.6 Human Interface Design

4.6.1 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, or use one from the community, 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.

4.6.2 Screen Images

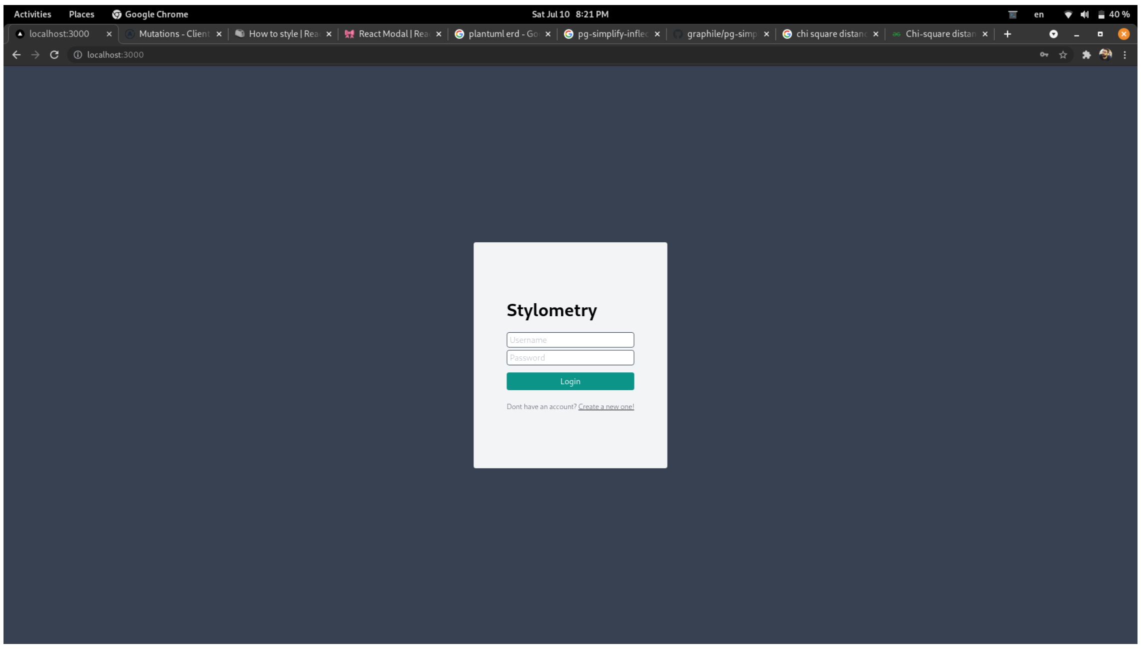
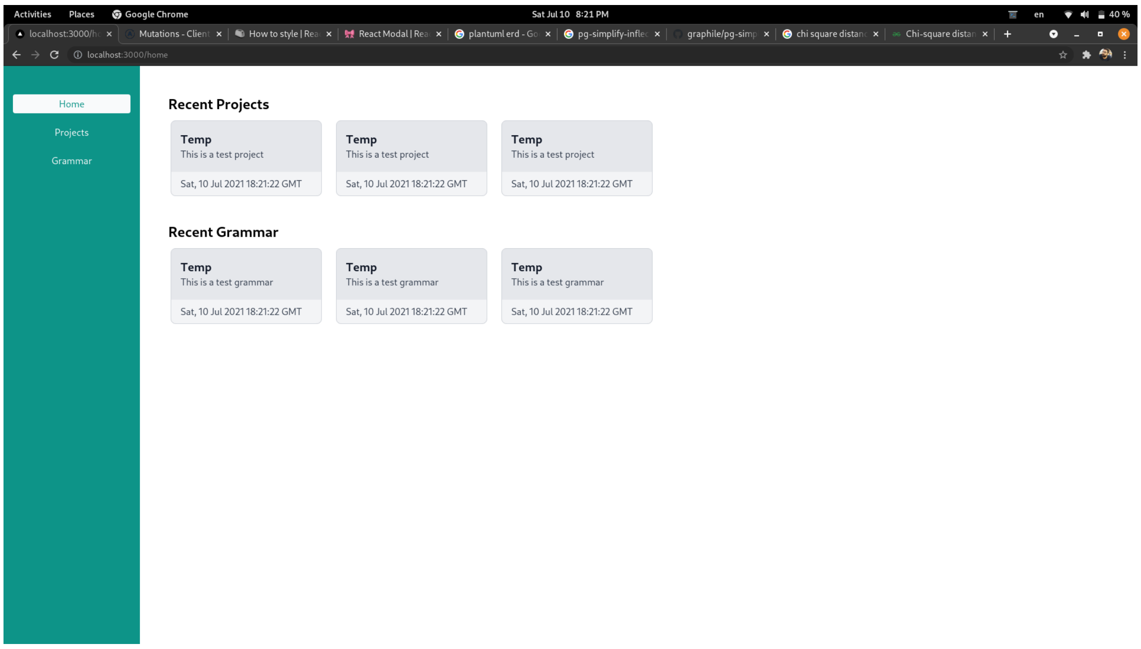


Figure 4.4: Login



Recent Projects

Temp
This is a test project
Sat, 10 Jul 2021 18:21:22 GMT

Temp
This is a test project
Sat, 10 Jul 2021 18:21:22 GMT

Temp
This is a test project
Sat, 10 Jul 2021 18:21:22 GMT

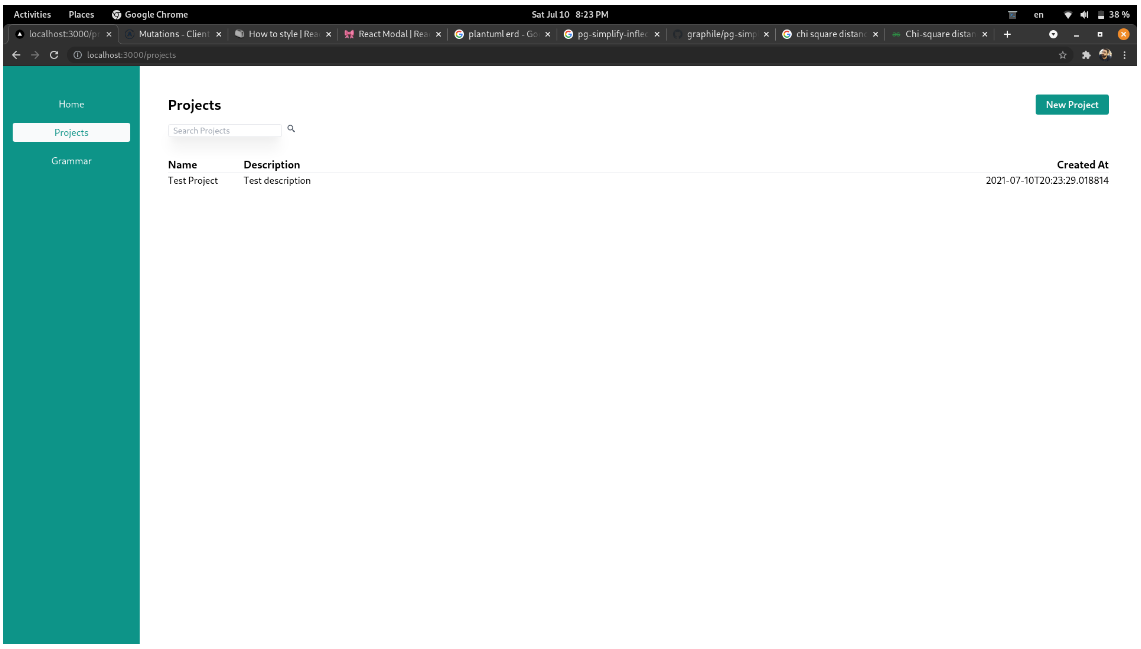
Recent Grammar

Temp
This is a test grammar
Sat, 10 Jul 2021 18:21:22 GMT

Temp
This is a test grammar
Sat, 10 Jul 2021 18:21:22 GMT

Temp
This is a test grammar
Sat, 10 Jul 2021 18:21:22 GMT

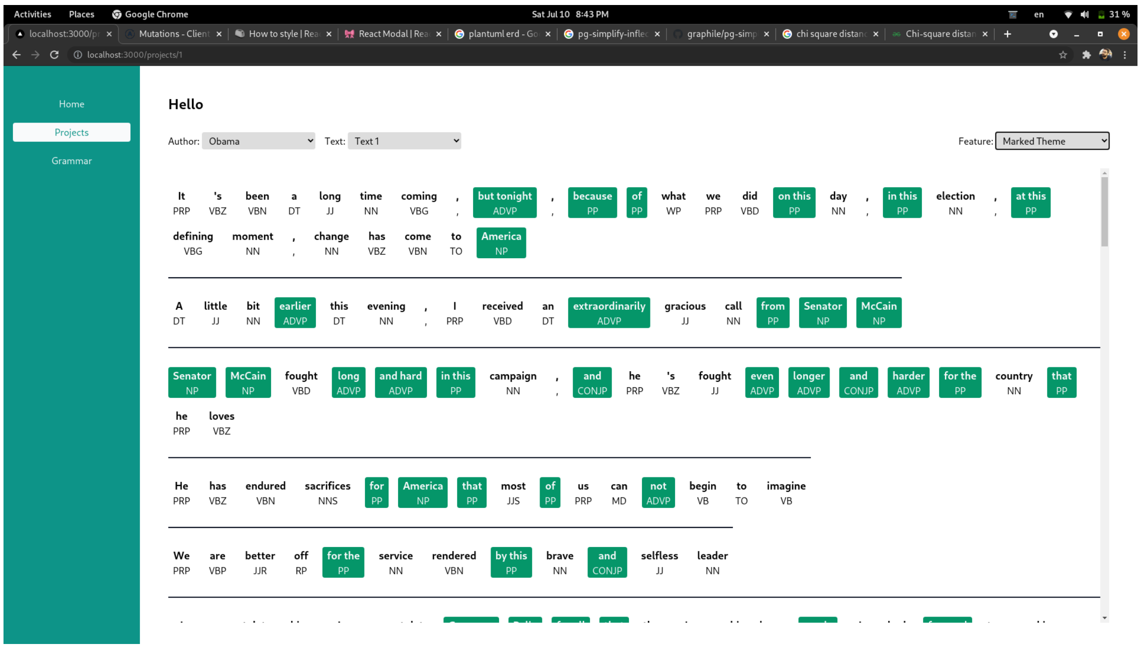
Figure 4.5: Home



Projects

Name	Description	Created At
Test Project	Test description	2021-07-10T20:23:29.018814

Figure 4.6: Projects


Activities Places Google Chrome Sat Jul 10 8:43 PM

localhost:3000/projects/1

Home
Projects
Grammar

Hello

Author: Obama Text: Text 1 Feature: Marked Theme

It PRP | 's VBZ | been VBN | a DT | long JJ | time NN | coming VBG | , , | but tonight ADVP | , , | because PP | of PP | what WP | we PRP | did VBD | on this PP | day NN | , , | in this PP | election NN | , , | at this PP | defining VBG | moment NN | , , | change NN | has VBZ | come VBN | to TO | America NP

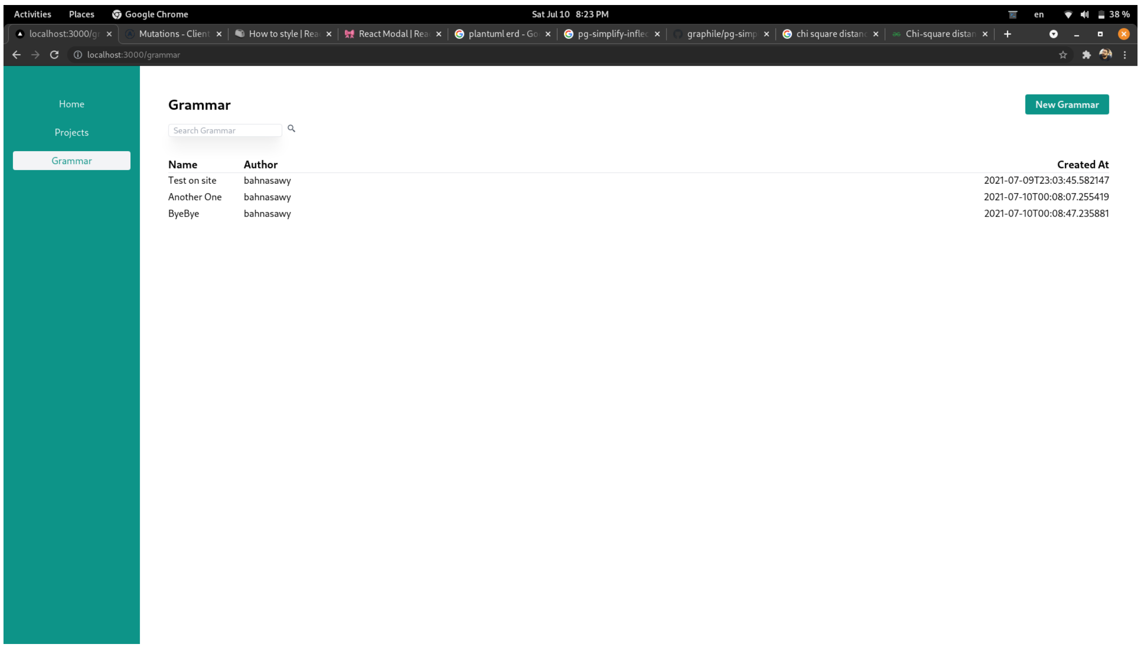
A DT | little JJ | bit NN | earlier ADVP | this DT | evening NN | , , | I PRP | received VBD | an DT | extraordinarily ADVP | gracious JJ | call NN | from PP | Senator NP | McCain NP

Senator NP | McCain NP | fought VBD | long ADVP | and hard ADVP | in this PP | campaign NN | , , | and CONJP | he PRP | 's VBZ | fought JJ | even ADVP | longer ADVP | and CONJP | harder ADVP | for the PP | country NN | that PP | he PRP | loves VBZ

He PRP | has VBZ | endured VBN | sacrifices NNS | for PP | America NP | that PP | most JJS | of PP | us PRP | can MD | not ADVP | begin VB | to TO | imagine VB

We PRP | are VBP | better JJR | off RP | for the PP | service NN | rendered VBN | by this PP | brave NN | and CONJP | selfless JJ | leader NN

Figure 4.7: Project


Activities Places Google Chrome Sat Jul 10 8:23 PM

localhost:3000/grammar

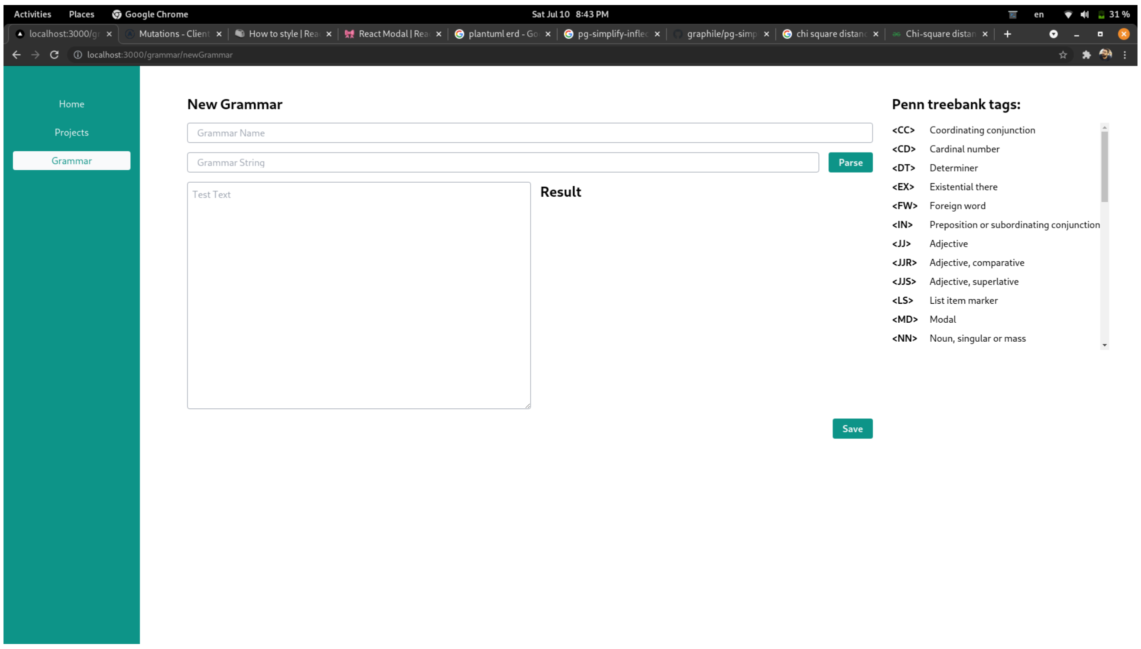
Home
Projects
Grammar

Grammar New Grammar

Search Grammar

Name	Author	Created At
Test on site	bahnasawy	2021-07-09T23:03:45.582147
Another One	bahnasawy	2021-07-10T00:08:07.255419
ByeBye	bahnasawy	2021-07-10T00:08:47.235881

Figure 4.8: Grammars



New Grammar

Result

Penn treebank tags:

<CC>	Coordinating conjunction
<CD>	Cardinal number
<DT>	Determiner
<EX>	Existential there
<FW>	Foreign word
<IN>	Preposition or subordinating conjunction
<JJ>	Adjective
<JJR>	Adjective, comparative
<JJS>	Adjective, superlative
<LS>	List item marker
<MD>	Modal
<NN>	Noun, singular or mass

Figure 4.9: Grammar

4.7 Requirements Matrix

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

Chapter 5

Implementation

5.1 Database

The database is accessed through a serverless function running a library called PostGraphile. The library introspects the database and creates a graphQL API to communicate with the database.

5.1.1 Code

```
const express = require("serverless-express/express");
const { postgraphile } = require("postgraphile");
var cors = require("cors");

const { SCHEMA, PG_DEFAULT_ROLE, POSTGRES_CONNECTION } = process.env;

const app = express();
app.use(cors());
app.use(
        postgraphile("postgres://postgres:@localhost:5432/grad", "public", {
                dynamicJson: true,
                graphqlRoute: "/",
                extendedErrors: ["hint", "detail", "errcode"],
                legacyRelations: "omit",
                pgDefaultRole: PG_DEFAULT_ROLE,
                subscriptions: true,
                setofFunctionsContainNulls: false,
                ignoreRBAC: false,
                appendPlugins: [
                        require("@graphile-contrib/pg-simplify-inflector"),
                        require("postgraphile-plugin-connection-filter"),
                        require("postgraphile-plugin-many-create-update-delete")
                ],
                retryOnInitFail: true,
                graphileBuildOptions: {
                        connectionFilterRelations: true,
                },
                graphiql: true,
                graphiqlRoute: "/graphiql",
        })
);

app.listen(8000, () => console.log(‘Server running on port 8000‘));
```

5.2 Server / Parser

The parser server is meant to parse the text according to the selected grammar and returns the result in JSON format.

5.2.1 Problems

The system uses the nltk tokenizer, which has an extremely high success rate, but with large enough texts defects start to show and those errors may cause the parser have false results, positive and/or negative.

5.2.2 Code

```
tokens = nltk.tokenize.word_tokenize(text)

sentences = []
temp = []
for token in tokens:
    if token == "." or token == "?" or token == "!":
        sentences.append(temp)
        temp = []
        else:
            temp.append(token)

pos = nltk.pos_tag_sents(sentences)


# Parsing
results = []
for feature in features:
    temp = []
    for sent in pos:
        temp.append(nltk.RegexpParser(feature["grammar"]).parse(sent))
    results.append(temp)


# Clean Up and Separation of features results
featuresTags = {}
for idx, feature in enumerate(features):
    sents = []
    for sent in results[idx]:
        temp = []
        for tag in sent:
            if type(tag) is tree.Tree:
                leafArr = []
                for leaf in tag.leaves():
                    leafArr.append(leaf[0])
                temp.append([leafArr ,tag.label(), True])
            else:
                temp.append([[tag[0]],tag[1], False])
        sents.append(temp)
    featuresTags[features[idx]["name"]] = sents
```

Chapter 6

Results and Evaluation

6.1 Results

6.1.1 Parser

The parser produces accurate results with a degree of certainty that matches the tokenizer accuracy. It only fails if given an incorrect Regex pattern, but it is an unavoidable limitation so no action was taken to fix it.

6.1.2 Analysis

The system can successfully measure the probability of similarity between authors and the disputed text, but the accuracy requires rigorous testing since it needs a significantly large data set. The tests will be conducted by collecting authors from similar genres and separating them into classes, taking a test text as a control, then they will be passed through the system and the outputs will be logged and studied to calculate the system's accuracy.

6.1.3 User interface

The user interface achieves its purpose, but is in need of clarifications, and user experience improvements since at this stage it requires exploration and/or trial and error to operate.

Sections of the interface were tested for validity by using invalid data formats or removing them entirely. The only failed test is the corpus loader as it needs to be able to reject poorly formatted corpus.

The only missing feature is the analysis editor as it needed a structural reformation of the analysis format that would've taken too much time that was not available.

Chapter 7

Conclusions and Future work

7.1 Conclusion

The project's goal is to provide an easy to use tool that requires minimal training that parses text for stylometric features and provide analysis if authorship attribution is needed. The system achieves this by using nltk and regular expression. While the system achieves its purpose, it is heavily dependent on the user submitting valid inputs and formats, with little no protection from mistakes.

7.2 Future Work

The parser could be vastly improved by introducing machine learning techniques for more accurate and more consistent results. The UI/UX also needs an overhaul to accommodate for inexperienced/new users. The system could also be improved by adding support for more file formats, as it only supports plain text files that are heavily reliant on the user to be formatted correctly. []

Please be sure to put here any materials you have collected during your graduation project.

Bibliography