Florida International University
School of Computing and Information Sciences

Software Engineering Focus

Final Deliverable

Project Title: Vocabulary in Reading Study
VIRS 2.0

Team Members: Alfredo Lopez, Milad Ebrahimi

Product Owner(s): Eric Dwyer, Seyedjafar Ehsanzadehsorati

Mentor(s): Leila Zahedi

Instructor: Masoud Sadjadi

The MIT License (MIT)
Copyright (c) *2017 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

This document presents the information necessary to gain a good understanding of Vocabulary in Reading Study (VIRS 2.0). VIRS is a web app which facilitates learning of new languages through analysis of text and gathering of data which is then displayed to the user. This data is important to the user as it categorizes words in the text and gives priority to more popular words. The user then knows which words to use more and makes it easier to use these in conversation. Data given to the user includes category of word as far as popularity and definition of the words in the text with respect to the information in the database. It also provides the user with a complete statistical text analysis report and readability score.

Table of Contents

Contents

INTRODUCTION

One of the major challenges that English Language learners (ELL) as well as mainstream students face is the lack of a reliable source to improve their academic words. There is no easy way to validate the easiness of text, which allows professors to select the appropriate materials for class. A challenging text is a wonderful way to propel students forward, yet something too hard to read can cause the opposite effect. Vocabulary in Reading Study(VIRS) is the solution to all these problems.

Current System

This is the second iteration of the application. The current system consisted of a MEAN stack application and a LAMP application hosted in GoDaddy. For simplicity, we will refer to them in this document as the Mean application, and the Lamp application respectively. They do not interact with each other. They do not even have a uniform theme.

Both applications categories the words and display them. The algorithm used has an unfavorable time complexity since it was implemented with a double loop that iterates through all the data, making the process very slow. The mean application can upload documents, but it does not analyze them while the Lamp one cannot do that. There was no security in place, the app was wide open to SQL injection and other attacks.

Figure 1 - Mean application

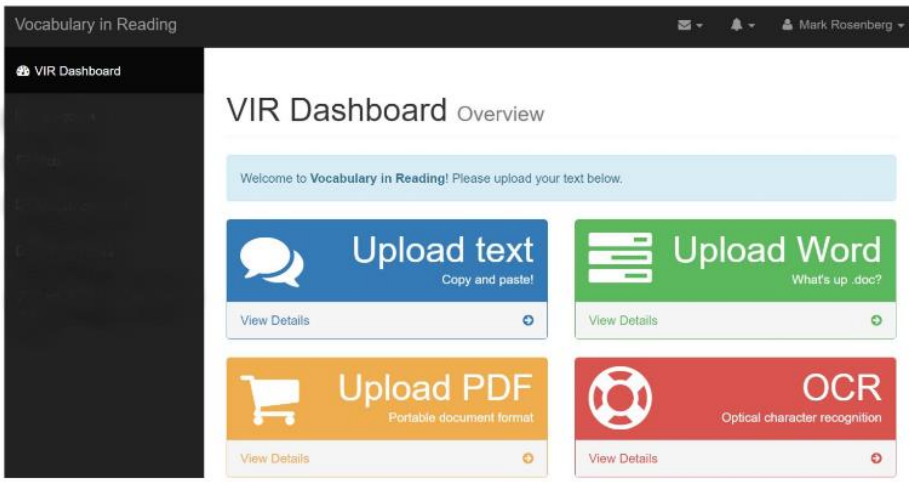


Figure 2- Lamp application

Purpose of New System

The imperative task of the new system is to consolidate both application before mentioned above into a single cohesive system. It shall be flexible enough to allow future upgrades and improvements. It will need to be rewritten with a modern development in mind. Mobile is essential in our lives, so it will have to be designed phone friendly.

Change the whole frontend design to make it responsive and user friendly. Create a modern single page application that loads and operates fast. Provide alerts and warnings when an error occurs. Adding an ability for admin to change the database right from the web application.

Create API that can be consumed via the web or a phone application. Word analysis has to be quick and text should be normalized according to inflection rules. It needs to be able to perform Optical Character Recognition (OCR) in images and pdfs. The system will need to be able to optimize the images, within certain constrains, to get better OCR results. Sensitive information must be secured in the application via passwords and encryption. The user can change the words information without having to use any other external tool.

The application should analyze the text and get the statistics of it. It will count the number of words, sentences and syllables per word. This information is used to calculate the Flesch Ease Readability score. The percent value indicates how easy to understand the text is.

The user can see and use the application in the web, phone devices and tablets.

USER STORIES

The following section provides the detailed user stories that were implemented in this iteration of the VIRS project. These user stories served as the basis for the implementation of the project's features. This section also shows the user stories that are to be considered for future development.

Implemented User Stories

- 132 [Backend] Perform OCR
- 145 [Backend] Analyze Text
- 146 [Backend] Analyze PDF
- 147 [Backend] Analyze Doc
- 148 Implement Responsive Design
- 149 Merge Applications
- 150 [Frontend] Create Admin Panel
- 151 [Backend] Implement Word Definition
- 154 [Frontend] Create Sidebar Menu
- 155 [Frontend] Apply Category Color to Words
- 157 Implement Google Analytics
- 158 [Backend] Create Admin Login
- 176 [Frontend] Analyze Text
- 178 [Frontend] Analyze PDF
- 179 [Frontend] Analyze Doc
- 181 [Backend] Manage words list
- 182 Create cloud application
- 183 [Frontend] Create page for enhanced
- 184 [Frontend] Create page for statistics
- 185 [Frontend] Create instructions
- 214 [Frontend] Analyze Image
- 220 [Frontend] Create Word Lists page
- 222 [Frontend] Implement Word definition
- 223 [Frontend] Add Credits Page
- 224 [Frontend] Add Contact Us Page
- 225 [Backend] Calculate Statistics
- 227 [iPhone] Create iPhone Application
- 228 [Android] Create Android Application
- 243 Create Search API

User Story Name: Implement responsive design

- Description: As a User, I would like to use the website through my phone so that I use it on the go.

Acceptance Criteria

- All sections of the app display correctly in different screen sizes.
- All features of the app are easily accessible.

Use Case

- Name: Implement Responsive Design
- Actor: User
- Preconditions: The user has the desire to use the application on different window sizes.
- Description <Flow of events>:
    - The Use Case starts when the user starts the application on the browser.
    - The system will display the main screen along with all the features.
    - If the user shrinks the browser window to phone device.
        - The system will rearrange the components of the website
    - The Use Case ends.

USER STORY NAME: MERGE APPLICATIONS

- Description: As a User, I would like to use the features from a single application so that I can operate it easily.

Acceptance Criteria

- The application should be accessed from the same URL.
- All features have the same technologies.

Use Case

- Name: Merge Applications
- Actor: User
- Preconditions: The user has the desire to use the features on the website.
- Description <Flow of events>:
  - The Use Case starts when the user starts the application.
  - The system will display the main screen along with all the features.
  - The user will select a feature.
  - If the user selects Upload Text.
  - The system will display Upload Text screen.
  - else If the user selects Upload Word.
    - The system will display Upload Word screen.
  - else If the user selects Upload PDF.
    - The system will display Upload PDF screen.
  - else If the user selects Upload Image.
    - The system will display Upload Image screen.
  - The Use Case ends.

USER STORY NAME: ANALYZE PDF

- Description: As a user, I would like to analyze a pdf so that I can see each word's category.

Acceptance Criteria

- Accept a pdf input.
- Determine which category each word belongs to.
- Calculate the statistics of each category.

Use Case

- Name: Pdf Upload
- Actor: User
- Preconditions: The user has the desire to analyze a PDF.
- Description <Flow of events>:
  - The user interacts with systems and chooses pdf
  - The user sends the information
    - The system receives the information and process it
  - The system displays the information.
  - The use case ends

USER STORY NAME: ANALYZE DOC

- Description: As a user, I would like to analyze a word document so that I can see each word's category.

Acceptance Criteria

- Accept a word document input.
- Determine which category each word belongs to.
- Calculate the statistics of each category.

Use Case

- Name: Doc Upload
- Actor: User
- Preconditions: The user has the desire to analyze a word document
- Description <Flow of events>:
  - The user interacts with systems and chooses word document
  - The user sends the information
    - The system receives the information and process it
  - The system displays the information.
  - The use case ends

USER STORY NAME: [FRONTEND] ANALYZE TEXT

- Description: As a user, I would like to analyze a submitted text so that I can see each word's category.

Acceptance Criteria

- Accept user text input.
- Determine which category each word belongs to.

Use Case

- Name: [Frontend] Analyze text
- Actor: User
- Preconditions: The user has the desire to use the application to analyze text
- Description <Flow of events>:
  - The Use Case starts when the user starts the application on the browser.
    - The system will display the main screen along with all the features.
  - The user clicks on "Upload Text" card.
    - The system will display the Analyze text page.
  - The user enters the text in the text area and click on "Enhanced Text"
    - The system will show the enhanced text result, with each word being categorized.
  - The Use Case ends.

USER STORY NAME: [FRONTEND] ANALYZE PDF

- Description: As a user, I want to see the words and their different category from PDF so that I may know more about the words in the image.

Acceptance Criteria

- Accept a pdf input.
- Determine which category each word belong to.
- Calculate the statistics of each category.

Use Case

- Name: [Frontend] Analyze PDF
- Actor: User
- Preconditions: The user has the desire to use the application to analyze PDF
- Description <Flow of events>:
  - The Use Case starts when the user starts the application on the browser.
    - The system will display the main screen along with all the features.
  - The user clicks on “Upload PDF” card.
    - The system will display the Analyze PDF page.
  - The user uploads the image and click on “Enhanced Text”
    - The system will show the enhanced text result, with each word being categorized.
  - The Use Case ends.

USER STORY NAME: [FRONTEND] ANALYZE DOC

- Description: As a user, I would like to analyze a submitted Word document so that I can see each word's category.

Acceptance Criteria

- Accept user Word document input.
- Determine which category each word belongs to.
- Calculate the statistics of each category.

Use Case

- Name: [Frontend] Analyze Doc
- Actor: User
- Preconditions: The user has the desire to use the application to analyze doc
- Description <Flow of events>:
  - The Use Case starts when the user starts the application on the browser.
    - The system will display the main screen along with all the features.
  - The user clicks on "Upload Doc" card.
    - The system will display the Analyze doc page.
  - The user uploads the image and click on "Enhanced Text"
    - The system will show the enhanced text result, with each word being categorized.
  - The Use Case ends.

USER STORY NAME: [FRONTEND] ANALYZE IMAGE

- Description: As a user, I want to see the words and their different category from an image file so that I may know more about the words in the image.

Acceptance Criteria

- Get access to OCR API.
- Learn OCR API
- Program and implement OCR API in our App

Use Case

- Name: [Frontend] Analyze Image
- Actor: User
- Preconditions: The user has the desire to use the application to analyze image
- Description <Flow of events>:
  - The Use Case starts when the user starts the application on the browser.
    - The system will display the main screen along with all the features.
  - The user clicks on “Upload Image” card.
    - The system will display the Analyze image page.
  - The user uploads the image and click on “Enhanced Text”
    - The system will show the enhanced text result, with each word being categorized.
  - The Use Case ends.

USER STORY NAME: PERFORM OCR

- Description: As a user, I want to see the words and their different categories from an image file so that I can know more about the words in the image.

Acceptance Criteria

- The user can see information about the words in the picture
- The user can see all the words in the picture

Use Case

- Name: Perform OCR
- Actor: User
- Preconditions: The user has the desire to know more information about words in a picture.
- Description <Flow of events>:
  - The user interacts with systems and chooses a picture
  - The user sends the information
    - The system receives the information and process it
  - The system displays the information.
  - The use case ends
  - Alternative: The user receives an unparsable exception.

USER STORY NAME: IMPLEMENT WORD DEFINITION

- Description: As a user, I would like to see the word definition so that I can learn it.

Acceptance Criteria

- The user can see the word definition from Wikipedia.
- The user can see only the English definition.

Use Case

- Name: Obtain Word Definition.
- Actor: User
- Preconditions: The user has the desire to know the definition of a word.
- Description <Flow of events>:
  - The user interacts with systems and selects a word
  - The user sends the word information
    - The system receives the information.
    - The system sends information to get definition from Wiki API
  - The system displays the information.
  - The use case ends
- Alternative: The user receives an UnableToGetEntry exception.

USER STORY NAME: [FRONTEND] CREATE PAGE FOR STATISTICS

- Description: As a user, I would like to see the different statistics for the words so I can easily determine the text complexity.

Acceptance Criteria

- The user can see the readability score.
- The user can see the counts of words.
- User can see the counted percentage of words

Use Case

- Name: [Frontend] Create page for statistics
- Actor: User
- Preconditions: The user has the desire to see the statistics report of the analyzed text
- Description <Flow of events>:
  - The Use Case starts when the user clicks on “Statistics” button.
    - The system will display the statistics report of the given text.
  - The Use Case ends.

USER STORY NAME: [FRONTEND] CREATE WORD LISTS PAGE

- Description: As a user, I will be able to see the list of the words, so that I can see each category separately

Acceptance Criteria

- The user could sort the list
- The user could choose the number of word shown in the page

Use Case

- Name: [Frontend] Create Word Lists page
- Actor: User
- Preconditions: The user has the desire to check the different word categories
- Description <Flow of events>:
  - The Use Case starts when the user clicks on “Word Lists” link on the sidebar.
    - The system will display the Word Lists page.
  - The user can see the AWL list as a default category
  - The user click on High Frequency List.
    - The system will display the High Frequency List.
  - The user click on sort and quantity of the words in the table.
    - The system will rearrange the list.
  - The Use Case ends.

USER STORY NAME: [FRONTEND] IMPLEMENT WORD DEFINITION

- Description: As a user, I would like to see the word definition so that I can learn it.

Acceptance Criteria

- Return a short definition of the word from Wikipedia.
- Return the definition for English only.

Use Case

- Name: [Frontend] Implement Word Definition
- Actor: User
- Preconditions: The user has the desire to see the word definition.
- Description <Flow of events>:
  - The Use Case starts when the user clicks on enhanced text button
    - The system will display the result; the enhanced text; categorized.
  - The user clicks on the desire word.
    - The system will display the word definition along with some examples.
  - The Use Case ends.

USER STORY NAME: CALCULATE STATISTICS

- Description: As a user, I would like to see the readability score for the text.

Acceptance Criteria

- The user can see the readability score.
- The user can see the count of words.
- The user can see the percentage of the words.

Use Case

- Name: Calculate Statistics.
- Actor: User
- Preconditions: The user has the desire to know the statistics for the text.
- Description <Flow of events>:
  - The user interacts with systems and submits a text.
  - The user sends the word information
    - The system receives the information.
  - The system displays the information.
  - The use case ends

USER STORY NAME: [FRONTEND] CREATE ADMIN PANEL

- Description: As an admin, I would like to access the database, so that I can modify it.

Acceptance Criteria

- Admin can see all the data.
- Admin can add new data.
- Admin can delete data.
- Admin can edit data.
- Admin can log in

Use Case

- Name: [Frontend] Create Admin Panel
- Actor: Admin
- Preconditions: The admin has the desire to add/delete a word from database
- Description <Flow of events>:
- The Use Case starts when the admin clicks on admin link
    - The system will display login page.
- The admin types his credentials
    - The system will display the admin dashboard upon successful login.
- Admin type the new word, choose the category click Add button.
    - The system will add the new word to database.
- Admin type a word in the “word” field and click delete.
    - The system will delete a word from database.
- The Use Case ends.

USER STORY NAME: CREATE IPHONE APPLICATION

- Description: As a user, I would like to access the Vocabulary in Reading application through the iPhone, so that I can access it easier and faster on my iPhone.

Acceptance Criteria

- User can download the application from the Apple Store.
- User can open the application after installation.
- User can use all the features from the application.

Use Case

- Name: Access iPhone Application
- Actor: User
- Preconditions: The user has the desire see interact with the application on the phone.
- Description <Flow of events>:
  - The user interacts with the phone to open the application.
    - The system receives the information and process it
  - The system displays the application.
  - The use case ends

USER STORY NAME: CREATE ANDROID APPLICATION

- As a user, I would like to access the Vocabulary in Reading application through in an Android application, so that I can access it easier and faster on my iPhone.

Acceptance Criteria

- User can download the application from the Google Store.
- User can open the application after installation.
- User can use all the features from the application.

Use Case

- Name: Access Android Application
- Actor: User
- Preconditions: The user has the desire see interact with the application on the phone.
- Description <Flow of events>:
  - The user interacts with the phone to open the application.
    - The system receives the information and process it
  - The system displays the application.
  - The use case ends

USER STORY NAME: ADD CREDITS PAGE

- As a user, I would like to see all the people who contributed to the project and all the references.

Acceptance Criteria

- User can see all reference material.

Use Case

- Name: Show credits page
- Actor: User
- Preconditions: The user has the desire to see the credits page.
- Description <Flow of events>:
  - The user interacts with the application.
    - The system receives the information and process it
  - The system displays the credits page.
  - The use case ends

USER STORY NAME: [FRONTEND] ADD CONTACT US PAGE

- Description: As a user, I would like to contact the members of the team.

Acceptance Criteria

- The user can send private message to website's admin
- The admin should receive the messages through his email address

Use Case

- Name: [Frontend] Add Contact Us Page
- Actor: User
- Preconditions: The user has the desire to contact the website's admin
- Description <Flow of events>:

- The Use Case starts when the user clicks on contact us link
    - The system will display contact us form
- The user types his information and his message
    - The system will display an alert if user leave any required field empty
- The user clicks on "Send Message".
    - The system sends the message along the user's information to admin's email.
    - The system redirects the user to home page on successful submission.
- The Use Case ends.

USER STORY NAME: CREATE SEARCH API

- As a user, I would like to search for words in the list of words.

Acceptance Criteria

- The word I am looking with be returned if it exists.
- If the word does not exist nothing is returned.

Use Case

- Name: Search word
- Actor: User
- Preconditions: The user has the desire to search a word.
- Description <Flow of events>:
  - The user interacts with the application to search the word
    - The system receives the information and process it
  - The system returns the word.
  - The use case ends
- Alternative cause of events:
  - Nothing is returned.

Pending User Stories

- #152 [Backend] Add Reading Tests
- #153 [Backend] Add Vocabulary Tests

USER STORY NAME: [BACKEND] ADD READING TESTS

- As a user, I would like to take a test so that I know what my reading level is.

Acceptance Criteria

- The user can take a test.
- The user can see test scores.
- The user can leave test.

USER STORY NAME: [BACKEND] ADD VOCABULARY TESTS

- As a user, I would like to take a test so that I know what my vocabulary level is.

Acceptance Criteria

- The user can take a test.
- The user can see the test score.
- The user can leave the test.

PROJECT PLAN

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint planning's are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

Hardware:

- Computer running Linux, Mac OS or windows.
- For the deployed application make sure you stay within the AWS server constraints
    - 64bit Amazon Linux 2017.03 v2.5.5 running Java 8

Software:

The following list is the software used in the application. Note that it is quite extensive and includes all the development layers of the stack.

Front end:

- Angular 4.3.1
- Ng-Bootstrap 1.0.0
- Ng-Translate 7.0.0
- Chat.js 2.7.1
- Font-Awesome 4.7.0
- Ng2-Charts 1.6.0
- Rxjs 5.1.0
- Jasmine 2.5.45
- Karma 1.7
- Typescript 2.3.3

Back end:

- Maven 3.5
    - Commons-lang3 3.4
    - HikariCP 1.5.6.RELEASE
    - Jai-imageio-core 1.3.1
    - Jai-imageio-jpeg2000 1.3.0
    - Jasypt-spring-boot-starter 1.16
    - Levigo-jbig2-imageio 2.0
    - Mysql-connector-java 1.5.6.RELEASE
    - Opencsv 3.3
    - Opencv 3.2.0-1
    - Spring-boot-starter-data-jpa 1.5.6.RELEASE
    - Spring-boot-starter-security 1.5.6.RELEASE
    - Spring-boot-starter-test 1.5.6.RELEASE
    - Spring-boot-starter-web 1.5.6.RELEASE
    - Springfox-swagger-ui 2.7.0
    - Springfox-swagger2 2.7.0
    - Sqlite-jdbc 1.5.6.RELEASE
    - Tess4j 3.4.1
    - Thucydides-core 0.9.275
    - Tika-parsers 1.16
- Tesseract 3.05.01
- Leptonica 1.74.4
- Mysql 14.14
- Java 1.8.0
- Spring Developer Suite 3.9

Android:

- Android Studio 3.0
- Gradle 3.0
- Android sdk 23

iOS:

- Xcode 9.1
- Command line tools 9.2

Other:

- Git 2.14.1
- Bash 3.2.57

Sprints Plan

Sprint 1

Attendees: Alfredo Lopez, Milad Ebrahimi, Eric Dwyer, Seyed Jafar Ehsan
Start time: 1:30 PM
End time: 3:15 Pm

After discussion, the velocity of the team were estimated to be 100 hours.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- #148 Implement Responsive Design
- #149 Merge Applications

The team members indicated their willingness to work on the following user stories.

- Alfredo Lopez
    - #149 Merge Applications

- Milad Ebrahimi
    - #148 Implement Responsive Design

Sprint 2

Attendees: Alfredo Lopez, Milad EBrahimi, Seyed Jafar Ehsan
Start time: 4:00 PM
End time: 4:30 PM

After discussion, the velocity of the team were estimated to be 100 hours.
The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story #145 [Backend] Analyze Text
- User Story #146 [Backend] Analyze PDF
- User Story #147 [Backend] Analyze Doc
- User Story #158 [Backend] Create Admin login
- User Story #181 [Backend] Manage Word list
- User Story #182 [Backend] Create Cloud Application
- User Story #183 [Frontend] Create page for enhanced text
- User Story #176 [Frontend] Analyze Text
- User Story #154 [Frontend] Create Sidebar Menu

The team members indicated their willingness to work on the following user stories.

- Alfredo Lopez
  - User Story #145 [Backend] Analyze Text
  - User Story #146 [Backend] Analyze PDF
  - User Story #147 [Backend] Analyze Doc
  - User Story #158 [Backend] Create Admin login
  - User Story #181 [Backend] Manage Word list
  - User Story #182 [Backend] Create Cloud Application

- Milad Ebrahimi
  - User Story #183 [Frontend] Create page for enhanced text
  - User Story #176 [Frontend] Analyze Text
  - User Story #154 [Frontend] Create Sidebar Menu

Sprint 3

Attendees: Alfredo Lopez, Milad EBrahimi, Seyed Jafar Ehsan
Start time: 4:00 PM
End time: 4:30 PM

After discussion, the velocity of the team were estimated to be 100 hours.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story #155 [Frontend] Apply Category Color to Words
- User Story #178 [Frontend] Analyze PDF
- User Story #179 [Frontend] Analyze Doc
- User Story #214 [Frontend] Analyze Image
- User Story #184 [Frontend] Create Page for Statistics.
- User Story #132 [Backend] Perform OCR
- User Story #151 [Backend] Implement Word Definition

The team members indicated their willingness to work on the following user stories.

- Alfredo Lopez
    - User Story #132 [Backend] Perform OCR
    - User Story #151 [Backend] Implement Word Definition

- Milad Ebrahimi
    - User Story #155 [Frontend] Apply Category Color to Words
    - User Story #178 [Frontend] Analyze PDF
    - User Story #179 [Frontend] Analyze Doc
    - User Story #214 [Frontend] Analyze Image
    - User Story #184 [Frontend] Create Page for Statistics.

Sprint 4

Attendees: Alfredo Lopez, Milad EBrahimi, Seyed Jafar Ehsan
Start time: 4:00 PM
End time: 4:30 PM

After discussion, the velocity of the team were estimated to be 100 hours

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story #184 [Frontend] Create Page for Statistics.
- User Story #185 [Frontend] Create Instruction
- User Story #220 [Frontend] Create Word Lists Page
- User Story #222 [Frontend] Implement Word Definition
- User Story #132 [Backend] Perform OCR
- User Story #151 [Backend] Implement Word Definition
- User Story #225 [Backend] Calculate Statistic

The team members indicated their willingness to work on the following user stories.

- Alfredo Lopez
    - User Story #132 [Backend] Perform OCR
    - User Story #151 [Backend] Implement Word Definition
    - User Story #225 [Backend] Calculate Statistic

- Milad Ebrahimi
    - User Story #184 [Frontend] Create Page for Statistics.
    - User Story #185 [Frontend] Create Instruction
    - User Story #220 [Frontend] Create Word Lists Page
    - User Story #222 [Frontend] Implement Word Definition

Sprint 5

Attendees: Alfredo Lopez, Milad EBrahimi, Seyed Jafar Ehsan
Start time: 4:00 PM
End time: 4:30 PM

After discussion, the velocity of the team were estimated to be 100 hours.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story #150 [Frontend] Create Admin Panel
- User Story #227 [iPhone] Create an iPhone App for the application
- User Story #228 [Android] Create android app

The team members indicated their willingness to work on the following user stories.

- Alfredo Lopez
    - User Story #227 [iPhone] Create an iPhone App for the application
    - User Story #228 [Android] Create android app

- Milad Ebrahimi
    - User Story #150 [Frontend] Create Admin Panel

Sprint 6

Attendees: Alfredo Lopez, Milad EBrahimi, Seyed Jafar Ehsan
Start time: 4:00 PM
End time: 4:30 PM

After discussion, the velocity of the team were estimated to be 100 hours.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story #157 [Frontend] Implement Google Analytics
- User Story #224 [Frontend] Add Contact Us Page
- User Story #223 [Frontend] Add Credits Page
- User Story #243 [Backend] Create Search API

The team members indicated their willingness to work on the following user stories.

- Alfredo Lopez
    - User Story #223 [Frontend] Add Credits Page
    - User Story #243 [Backend] Create Search API

- Milad Ebrahimi
    - User Story #157 [Frontend] Implement Google Analytics
    - User Story #224 [Frontend] Add Contact Us Page

SYSTEM DESIGN

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

Architectural Patterns

Model View Controller is the main design for the architecture. We are separating the main three parts of the application: user interaction, processing of information and storage. The segregation of this concerns favors production since each of the developers work in a separate section. This separation of components makes the system easily modifiable in the future. With this design, we ensure that we have multiple views for a controller; the system produces APIs that can be consumed by any other application.

Client-Server is used in the system to deliver the application. A highly available system that can be consumed from several parties called for this design. It allows the centralization of the code logic and database. We had to access the application from the web, iOS and Android devices so this was a very good choice.

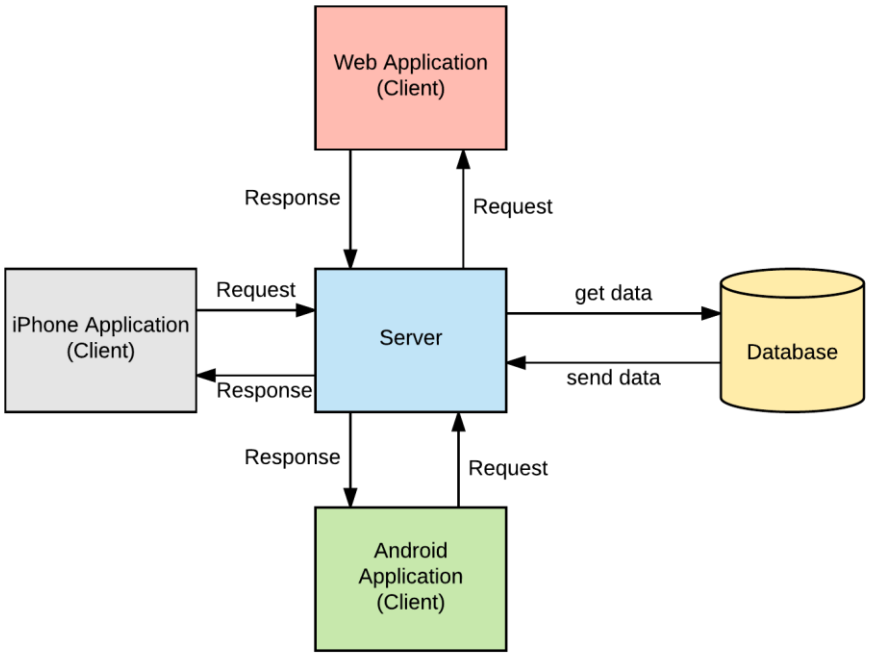


Figure 3Model View Controller

Repository pattern is another design used in our system. It minimizes the amount of duplicate code in the system by abstracting the basic CRUD operations. It also ties the data entities to the domain model which favors development. The code would have to comply with the entity restrictions in order to even comply. It also helps maintain data integrity.

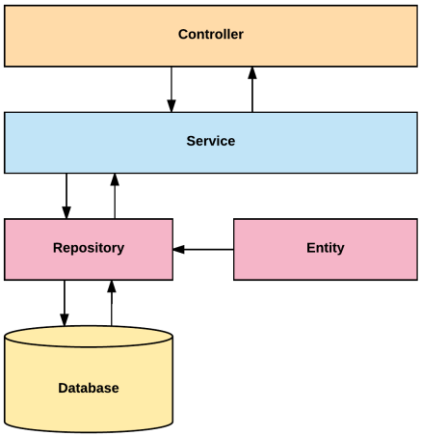


Figure 4- Server System Design

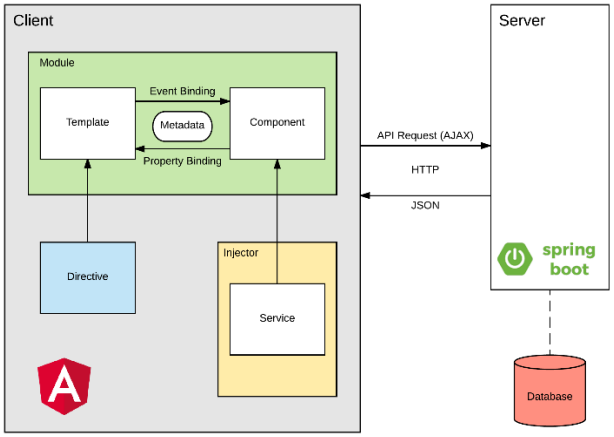


Figure 5- Client System Design

System and Subsystem Decomposition

The system is made out of two major subsystems and two minor ones. They are broken down as follows according to its tasks and interactions.

Server Subsystem:

- Interacts with the database.
- Serves all the requests from the web
- Handles server side security.
- Analyzes the statistics of the text
- Performs OCR
- Optimizes images.

Client Subsystem:

- Displays the application
- Routes server-side API calls
- Contains the main boundaries for user interaction.
- Client side data validation.

Android Subsystem:

- Interacts with the web application to display in an Android device.
- Handles device data storage needed for the web.

iPhone Subsystem:

- Interacts with the web application to display it in an iOS device.

Deployment Diagram

Deployment of the application consists in several steps run with a bash script. It starts in the front end and it propagates to the backend and eventually to the final product. We are building with ng for Angular products and Maven for the backend.

This is detailed explanation for the development pipeline.

- Angular
    - Clean and build the application
    - Run karma tests
    - Run e2e
- Copy resources to the backend
- Spring
    - Clean and build
    - Run Unit tests
    - Create jar executable

This is detailed explanation for the production pipeline.

- Angular
    - Clean and build the application
    - Run karma tests
    - Run e2e
- Copy resources to the backend
- Spring
    - Increase version number
    - Clean and build
    - Run Unit tests
    - Run Integration tests
    - Create jar executable
- Bundle the application for AWS
- Upload to AWS servers.

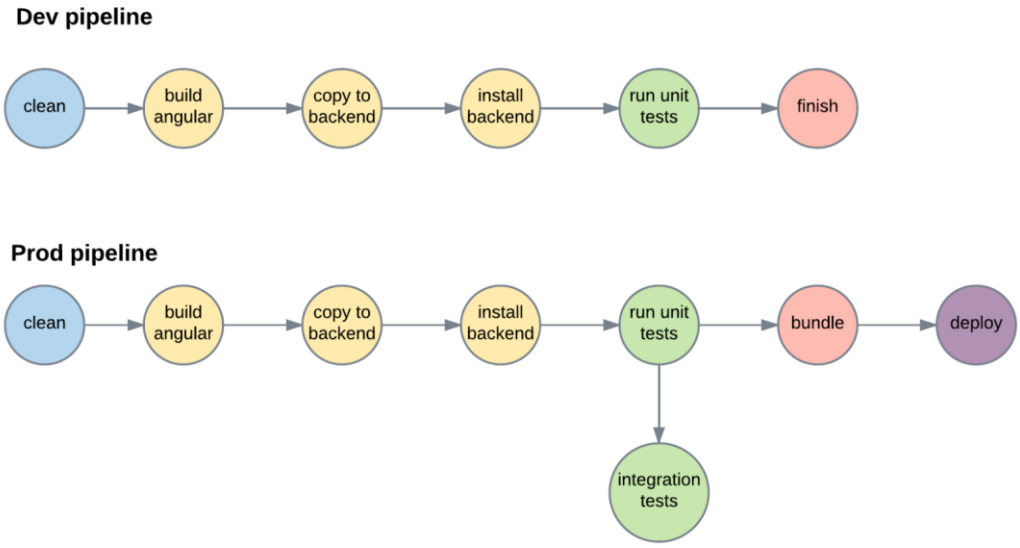


Figure 6 - Deployment Pipeline

Design Patterns

The following design patterns were used in the application.

Dependency Injection: This is a core design pattern for Spring and Angular 4. It allows the objects to be readily available and injected when needed rather than having to create them.

Bridge: This design pattern is used by the application to interact with libraries that are not native. Ex: tesseract and OpenCV

Front controller: Controllers are one of the main Components in the MVC design.

Marker: All components in Spring are annotated with markers to further specify the correct stereotype. This allows the initialization of the correct optimized components when the application runs.

Module: The application is divided in several modules according to functionality. This separation promotes organization and grouping of features.

SYSTEM VALIDATION

Backend

- **Test case ID**: analizeText_Endpoint_Returns200Ok
- Description/Summary of Test: When the API analizeText/ is accessed the server returns a 200 ok message.
- Pre-condition: Server is up and running.
- Expected Results: A 200 ok message is return.
- Actual Result: 200 ok
- Status (Fail/Pass): Passed

- **Test case ID:** process_WithPdfFile_ReturnsString
- Description/Summary of Test: A pdf file returns the value in the document as string
- Pre-condition: None
- Expected Results: The string from the body
- Actual Result: The string from the document
- Status (Fail/Pass): Pass

- **Test case ID:** process_WithDocFile_ReturnsString
- Description/Summary of Test: A word file returns the value in the document as string
- Pre-condition: None
- Expected Results: The string from the body
- Actual Result: The string from the document
- Status (Fail/Pass): Pass

- **Test case ID:** process_WithImgFile_ReturnsString
- Description/Summary of Test: Processing an image file returns the text in the image.
- Pre-condition: None
- Expected Results: A string of text from the image.
- Actual Result: A string of text from the image
- Status (Fail/Pass): pass

- **Test case ID:** getEntry_withWiki_returnsADictionaryWithAWikiEntryType
- Description/Summary of Test: Retrieving data with a Wiki type returns an object with the wiki source set.
- Pre-condition: None
- Expected Results: Html of the word definition from Wikipedia
- Actual Result: Html of the word definition from Wikipedia
- Status (Fail/Pass): Pass

- **Test case ID:** getFleschReadingEase_WithDataFromWebSite_Returns60_20
- Description/Summary of Test: Getting the Flesch Reading Score with word count of 300, a sentence count of 12 and a syllable count of 430 returns 60.20.
- Pre-condition: None
- Expected Results: A reading score of 60.20
- Actual Result: A Reading score of 60.20
- Status (Fail/Pass): Pass

- **Test case ID:** testWebAppShows
- Description/Summary of Test: Tests that the iPhone web app displays the web application.
- Pre-condition: None
- Expected Results: Content is not null.
- Actual Result: A webview instance.
- Status (Fail/Pass): pass

- **Test case ID:** shouldOverrideUrlLoading_ReturnsFalse
- Description/Summary of Test: Tests that the Android app redirects to the app web view always.
- Pre-condition: None
- Expected Results: URL with the domain will display inside the app.
- Actual Result: False.
- Status (Fail/Pass): pass

- **Test case ID:** findAll_Endpoint_Returns200Ok()
- Description/Summary of Test: Tests that the Android app redirects to the app web view always.
- Pre-condition: None
- Expected Results: The words will the found.
- Actual Result: List of words
- Status (Fail/Pass): pass

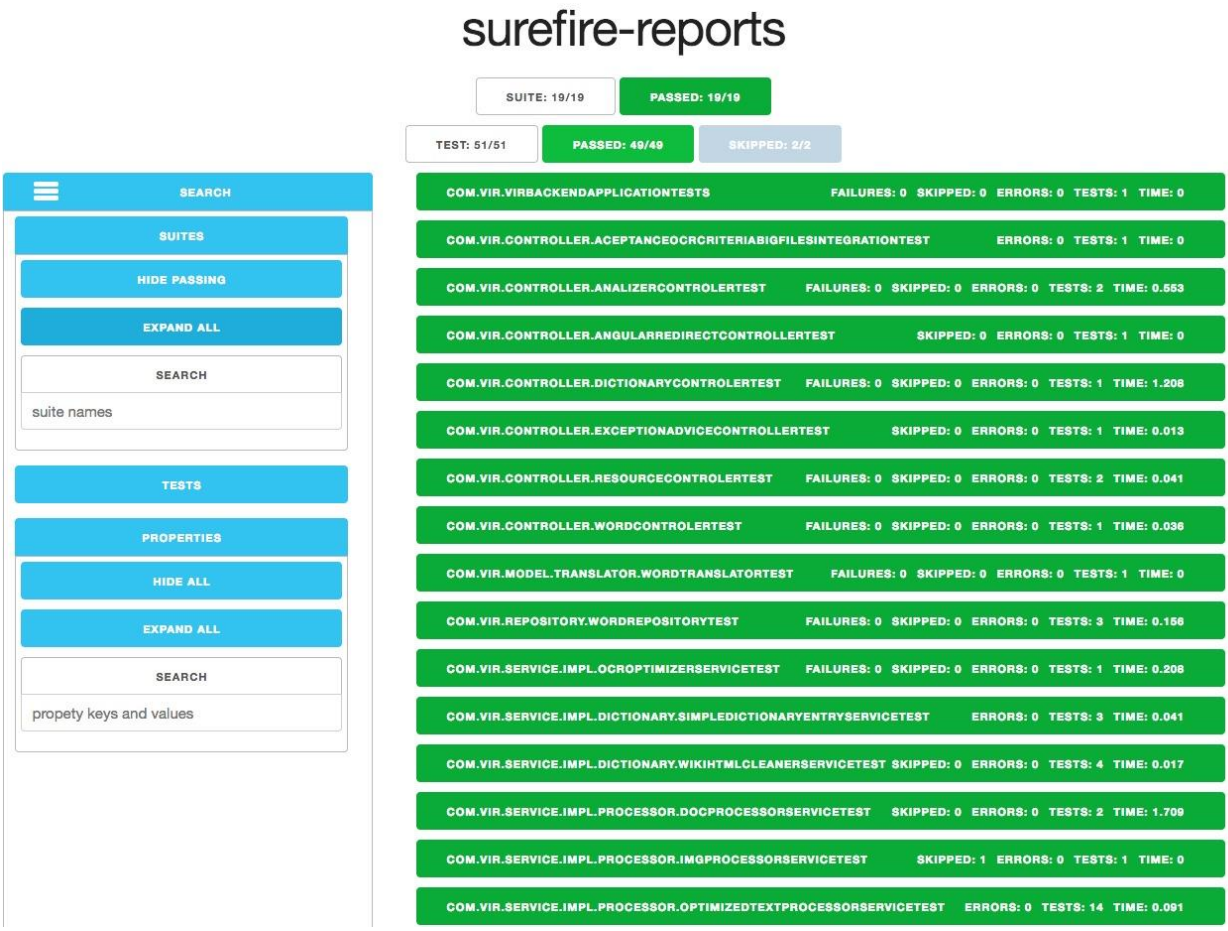


Figure 7- Backend Unit Testing

Frontend

- Test case ID: LayoutComponent_ShouldCreate_001
- Description/Summary of Test: The layout component should be created.
- Pre-condition: The server must be running.
- Expected Results: Layout Component should be created
- Actual Result: Layout Component was created
- Status (Fail/Pass): Pass

- Test case ID: TextComponent_ShouldCreate_001
- Description/Summary of Test: The Text component should be created.
- Pre-condition: The server must be running.
- Expected Results: Test Component should be created
- Actual Result: Test Component was created
- Status (Fail/Pass): Pass

- Test case ID: PDFComponent_ShouldCreate_001
- Description/Summary of Test: The PDF component should be created.
- Pre-condition: The server must be running.
- Expected Results: PDF Component should be created
- Actual Result: PDF Component was created
- Status (Fail/Pass): Pass

- Test case ID: DocComponent_ShouldCreate_001
- Description/Summary of Test: The Doc component should be created.
- Pre-condition: The server must be running.
- Expected Results: Doc Component should be created
- Actual Result: Doc Component was created
- Status (Fail/Pass): Pass

- Test case ID: ImageComponent_ShouldCreate_001
- Description/Summary of Test: The Image component should be created.
- Pre-condition: The server must be running.
- Expected Results: Image Component should be created
- Actual Result: Image Component was created
- Status (Fail/Pass): Pass

- Test case ID: TextStatistic_ShouldCreate_001
- Description/Summary of Test: The TextStatistic component should be created.
- Pre-condition: The server must be running.
- Expected Results: TextStatistic Component should be created
- Actual Result: TextStatistic Component was created
- Status (Fail/Pass): Pass

- Test case ID: DictionaryComponent_ShouldCreate_001
- Description/Summary of Test: The Dictionary component should be created.
- Pre-condition: The server must be running.
- Expected Results: Dictionary Component should be created
- Actual Result: Dictionary Component was created
- Status (Fail/Pass): Pass

- Test case ID: EnhancedTextResultComponent_ShouldCreate_001
- Description/Summary of Test: The EnhancedTextResult component should be created.
- Pre-condition: The server must be running.
- Expected Results: EnhancedTextResult Component should be created
- Actual Result: EnhancedTextResult Component was created
- Status (Fail/Pass): Pass

- Test case ID: AdminComponent_ShouldCreate_001
- Description/Summary of Test: The Admin component should be created.
- Pre-condition: The server must be running.
- Expected Results: Admin Component should be created
- Actual Result: Admin Component was created
- Status (Fail/Pass): Pass

- Test case ID: ContactUsComponent_ShouldCreate_001
- Description/Summary of Test: The ContactUs component should be created.
- Pre-condition: The Form Carry account should be activated.
- Expected Results: ContactUs Component should be created
- Actual Result: ContactUs Component was created
- Status (Fail/Pass): Pass

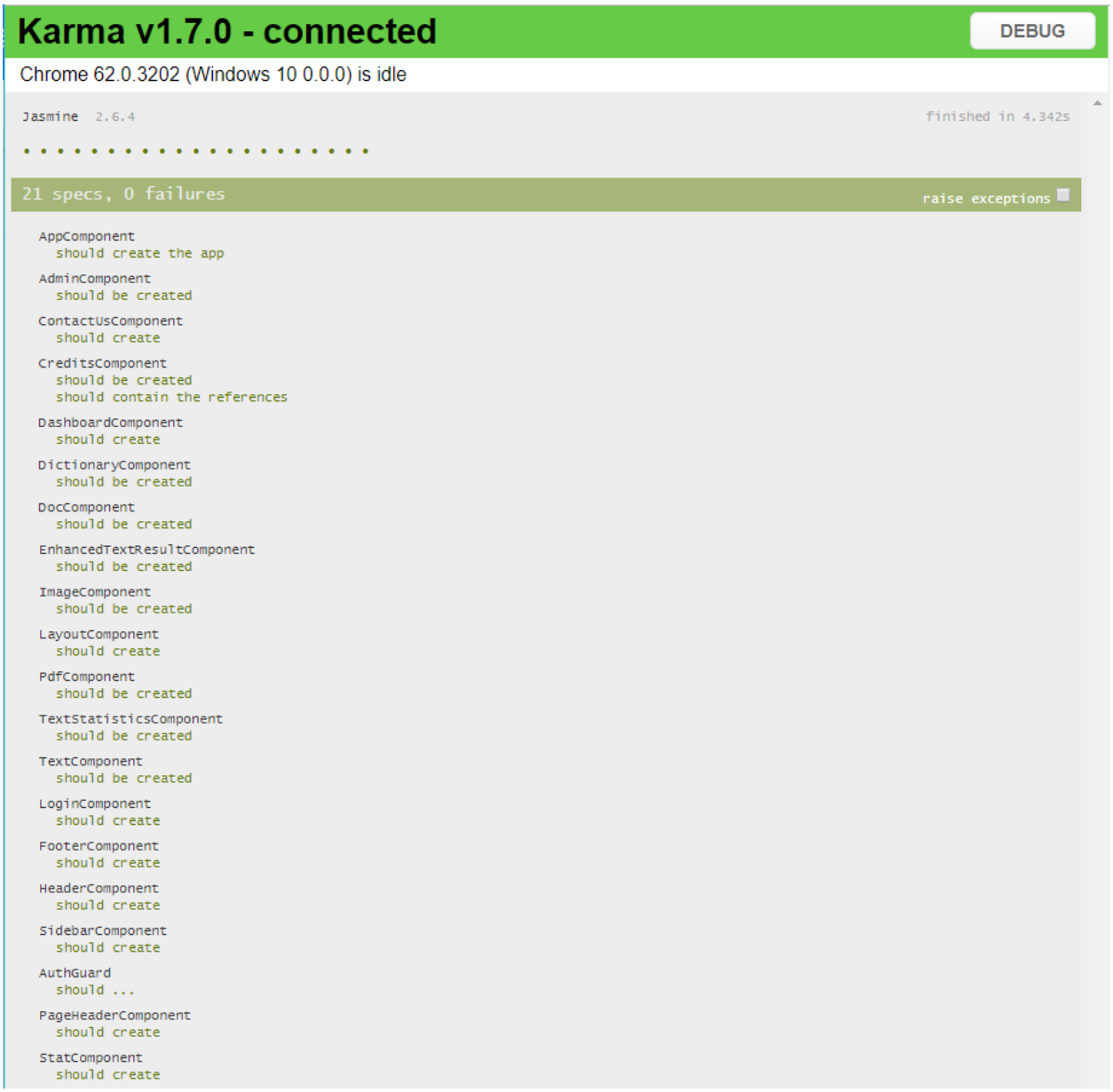


Figure 8- Frontend Unit testing - Karma 1.7, Jasmine 2.6.4

GLOSSARY

Academic Word List (AWL): List of words used in the natural English language with frequency high enough but that does not make it to the high frequency list.

High Frequency List: List of words used in natural English language with a high frequency.

Medium Frequency List: List of words used in natural English language with a medium frequency.

Low Frequency List: List of words used in natural English language with a low frequency.

Flesch Reading Ease Score: A test designed to calculate how hard a text is to understand in English.

Word Definition: Meaning of a word as per Wiki Dictionary. It contains etymology, meaning, and usage information.

Category: The assigned value to a word from one of the above lists. A word category can be: AWL, High Frequency, Medium Frequency, and Low Frequency.

Inflection: A modification if a word to express additional meanings: plural and conjugations.

School Dictionary: A collection of all the lists. It can be used as reference for the word categories.

APPENDIX

Appendix A - UML Diagrams

Figure 9- #148 Responsive Design - Use Case Diagram

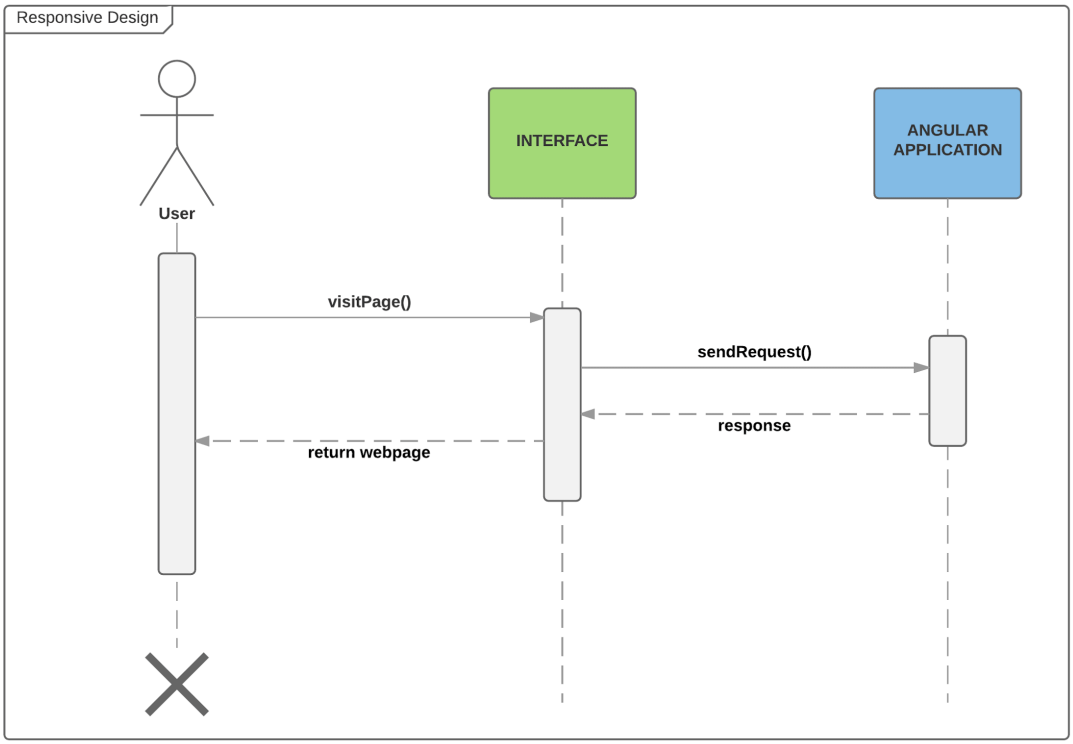


Figure 10- #148 Responsive Design - Sequence Diagram

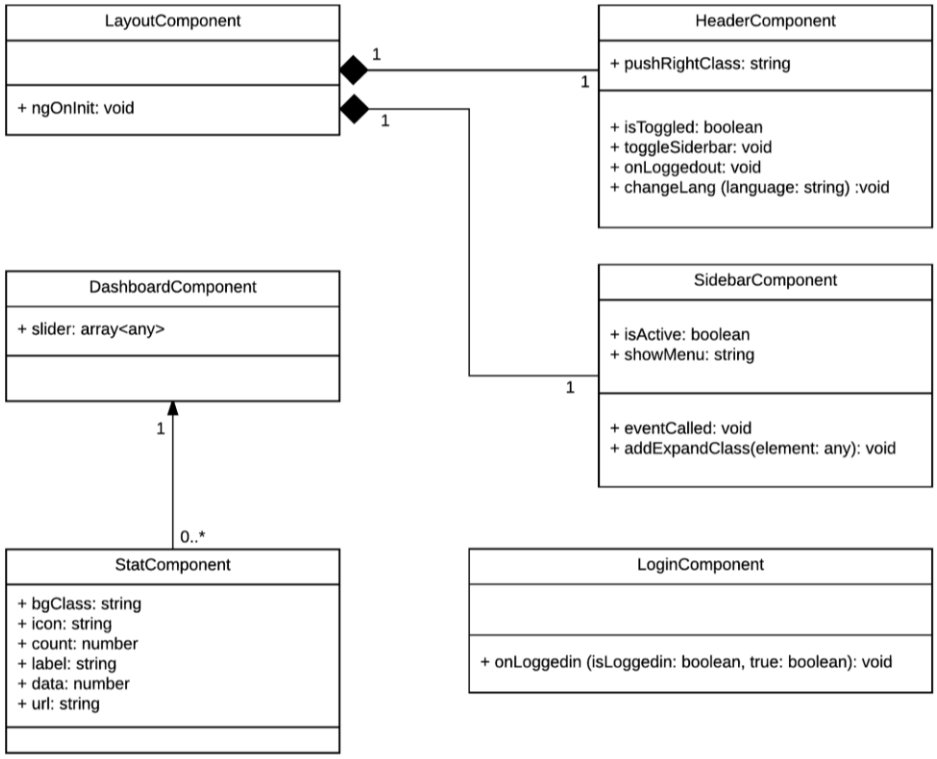


Figure 11- #148 Responsive Design -Class Diagram

Figure 12- #150 Create Admin Panel - Use Case Diagram

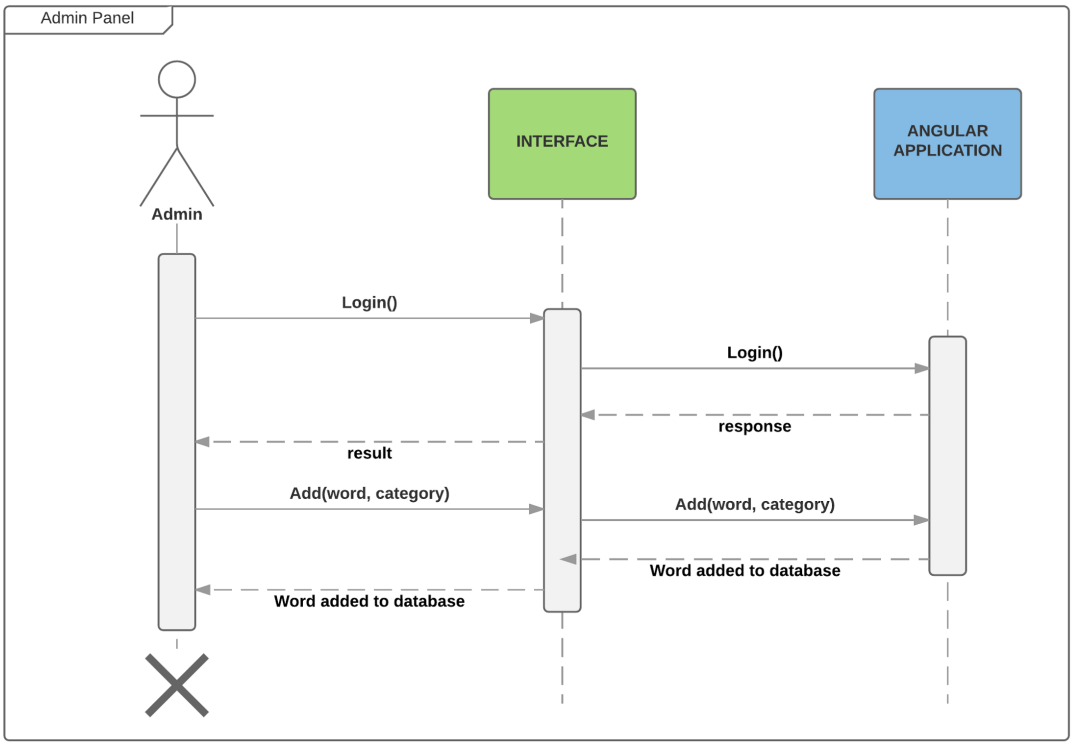


Figure 13- #150 Create admin panel - Sequence diagram

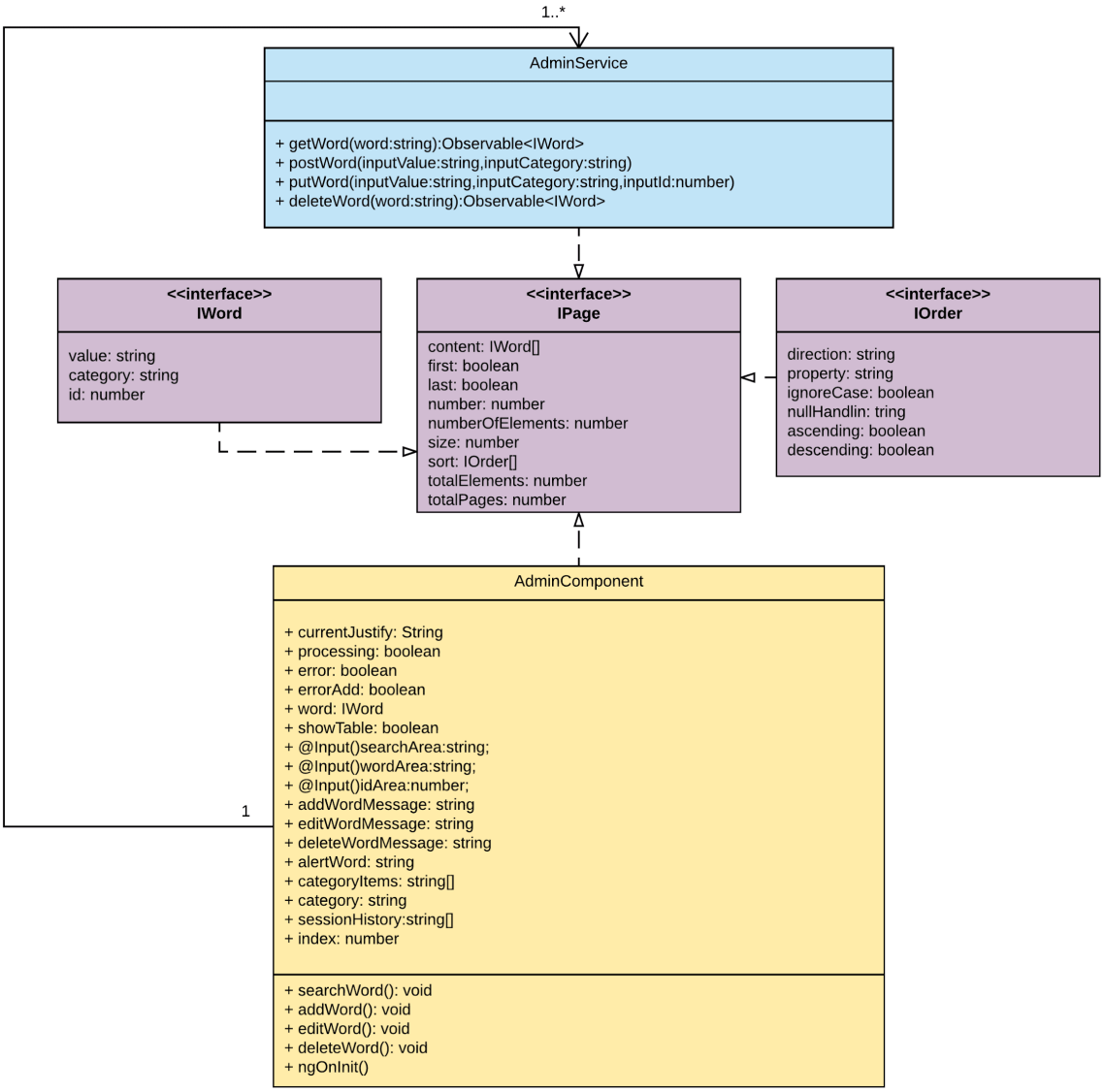


Figure 14 - #150 Create admin panel - Class diagram

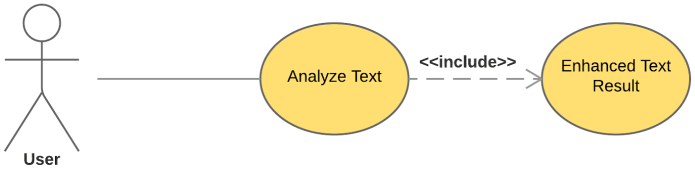


Figure 15 - #176 Analyze text - Use case diagram

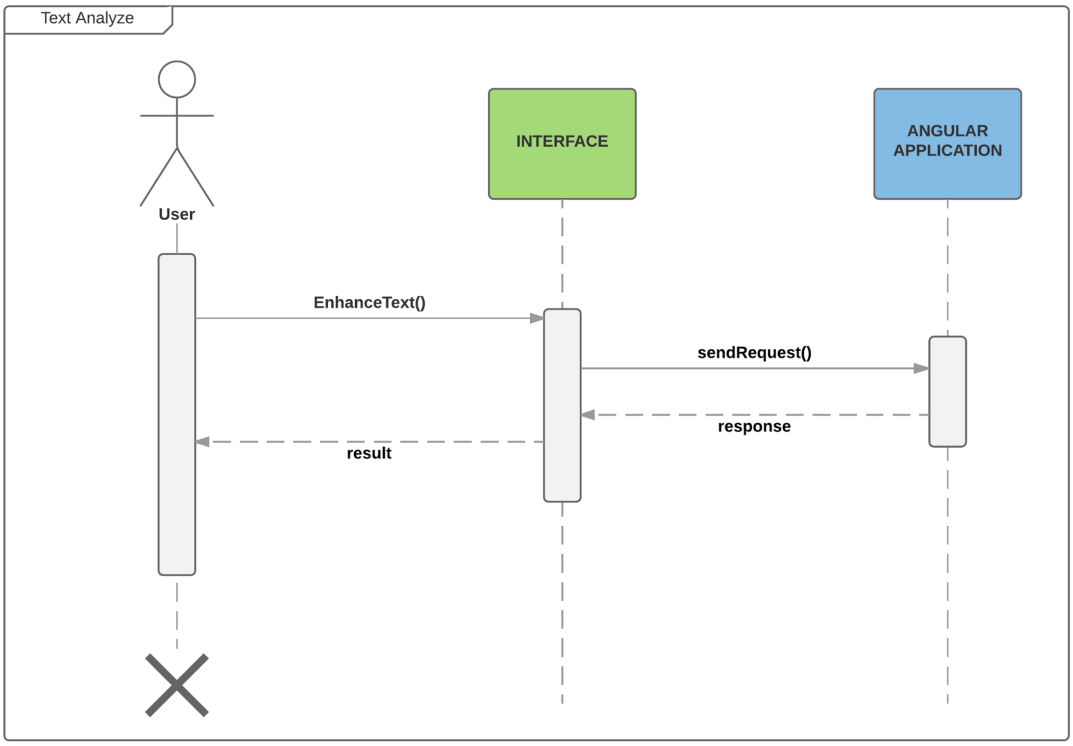


Figure 16- #176 Text analyze - Sequence diagram

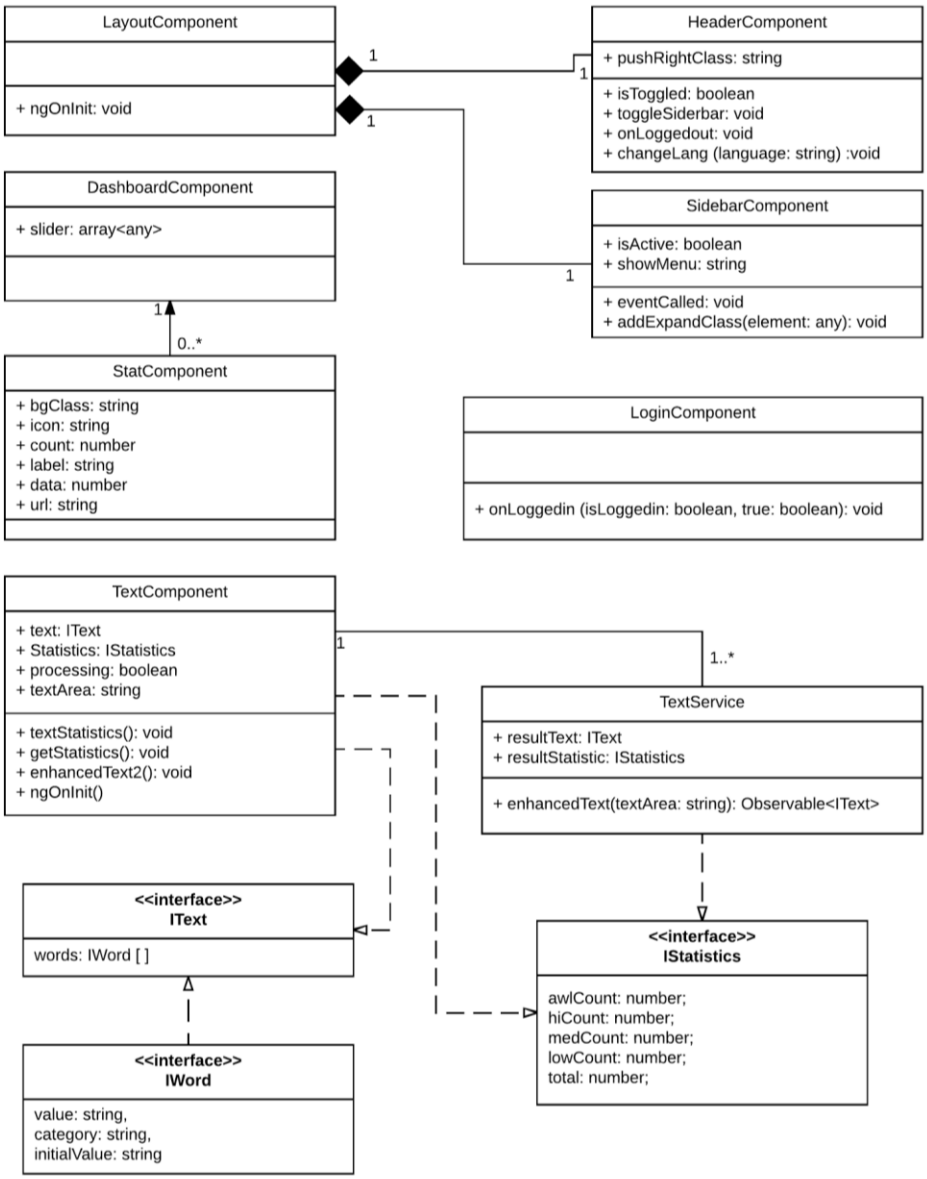


Figure 17- #176 Text analyze - Class diagram

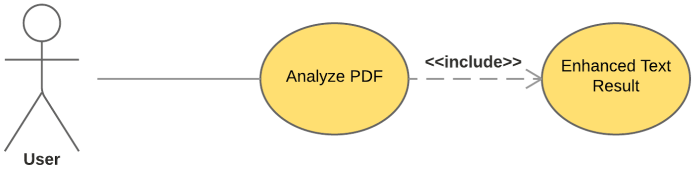


Figure 18- #178 Analyze PDF - Use case diagram

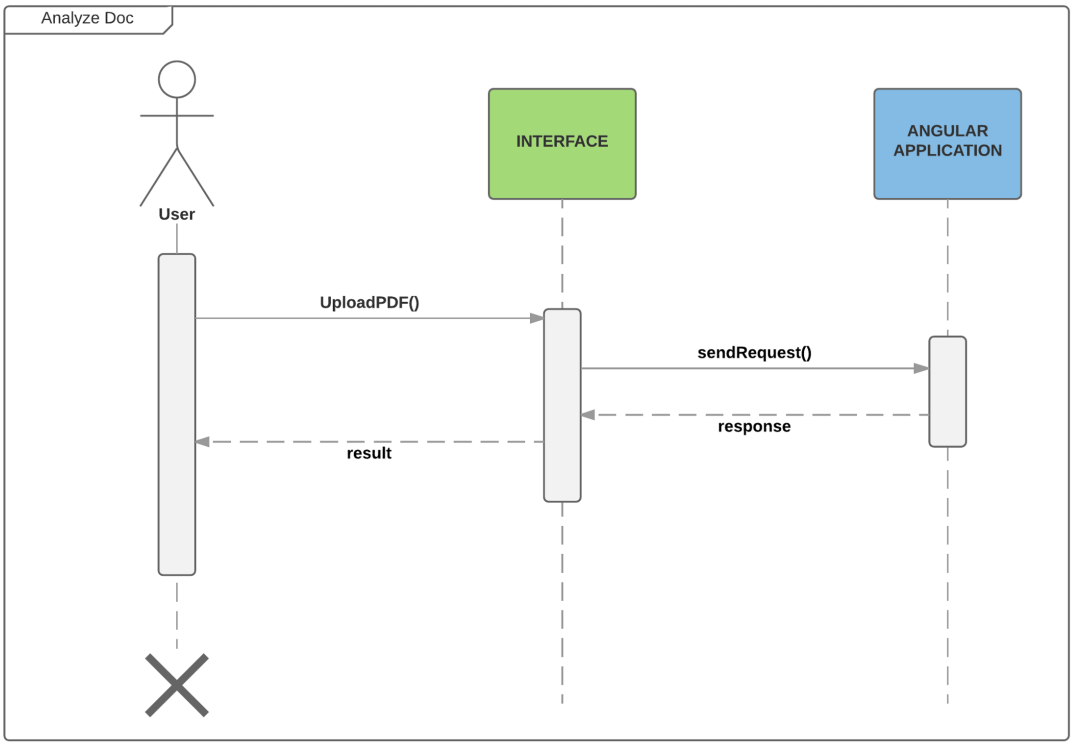
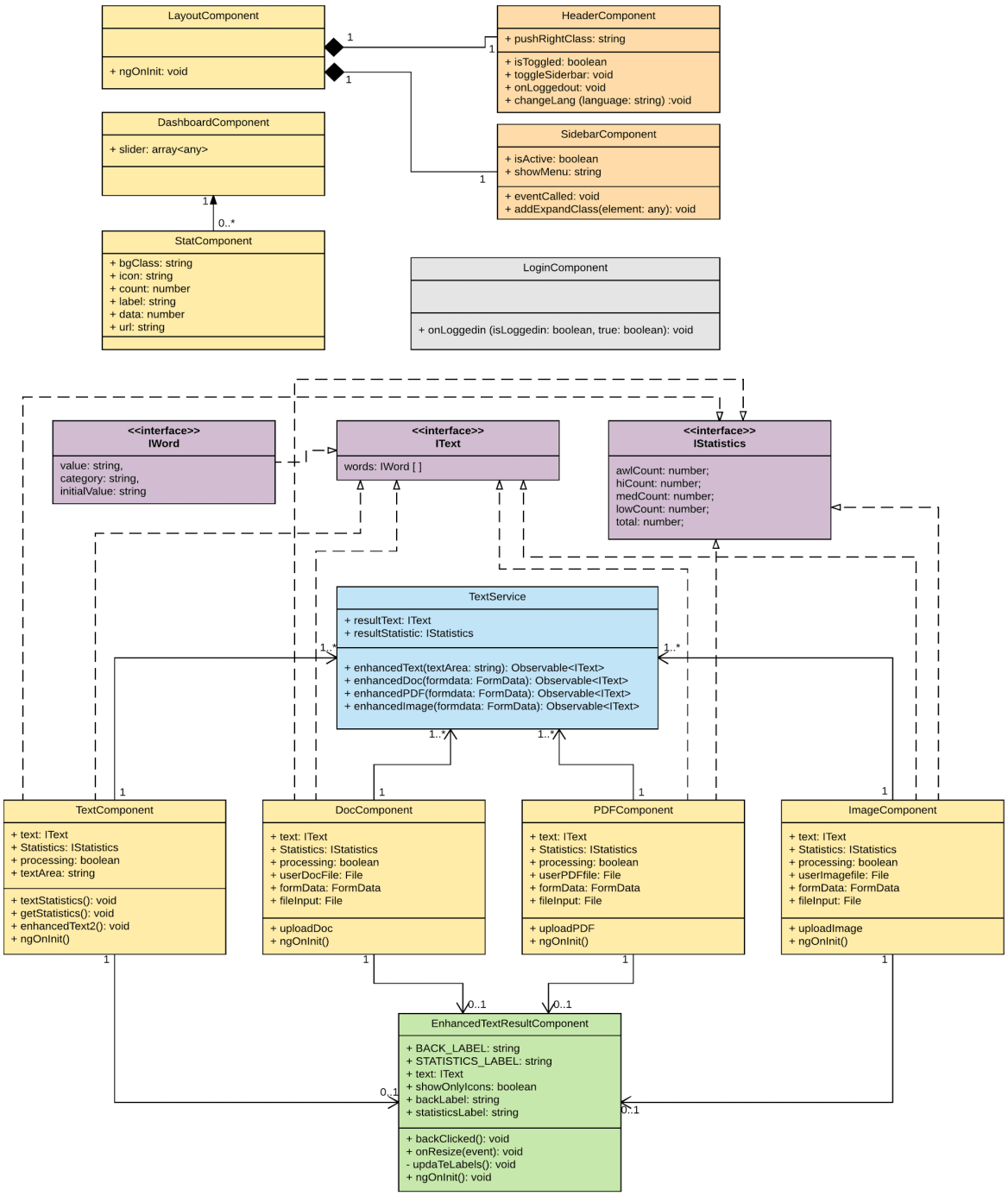


Figure 19-#178 Analyze PDF - Sequence Diagram


LayoutComponent
+ ngOnInit: void
HeaderComponent
+ pushRightClass: string
+ isToggled: boolean
+ toggleSiderbar: void
+ onLoggedout: void
+ changeLang (language: string) :void
SidebarComponent
+ isActive: boolean
+ showMenu: string
+ eventCalled: void
+ addExpandClass(element: any): void
DashboardComponent
+ slider: array<any>
StatComponent
+ bgClass: string
+ icon: string
+ count: number
+ label: string
+ data: number
+ url: string
LoginComponent
+ onLoggedin (isLoggedin: boolean, true: boolean): void
<<interface>>
IWord
value: string,
category: string,
initialValue: string
<<interface>>
IText
words: IWord [ ]
<<interface>>
IStatistics
awlCount: number;
hiCount: number;
medCount: number;
lowCount: number;
total: number;
TextService
+ resultText: IText
+ resultStatistic: IStatistics
+ enhancedText(textArea: string): Observable<IText>
+ enhancedDoc(formdata: FormData): Observable<IText>
+ enhancedPDF(formdata: FormData): Observable<IText>
+ enhancedImage(formdata: FormData): Observable<IText>
TextComponent
+ text: IText
+ Statistics: IStatistics
+ processing: boolean
+ textArea: string
+ textStatistics(): void
+ getStatistics(): void
+ enhancedText2(): void
+ ngOnInit()
DocComponent
+ text: IText
+ Statistics: IStatistics
+ processing: boolean
+ userDocFile: File
+ formData: FormData
+ fileInput: File
+ uploadDoc
+ ngOnInit()
PDFComponent
+ text: IText
+ Statistics: IStatistics
+ processing: boolean
+ userPDFfile: File
+ formData: FormData
+ fileInput: File
+ uploadPDF
+ ngOnInit()
ImageComponent
+ text: IText
+ Statistics: IStatistics
+ processing: boolean
+ userImagefile: File
+ formData: FormData
+ fileInput: File
+ uploadImage
+ ngOnInit()
EnhancedTextResultComponent
+ BACK_LABEL: string
+ STATISTICS_LABEL: string
+ text: IText
+ showOnlyIcons: boolean
+ backLabel: string
+ statisticsLabel: string
+ backClicked(): void
+ onResize(event): void
- updaTeLabels(): void
+ ngOnInit(): void

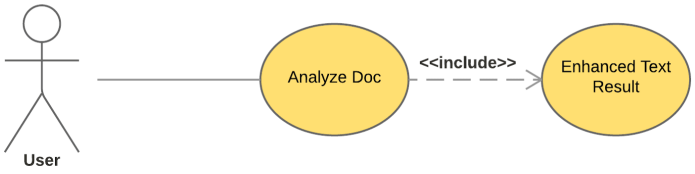


Figure 20- #179 Analyze doc - Use case diagram

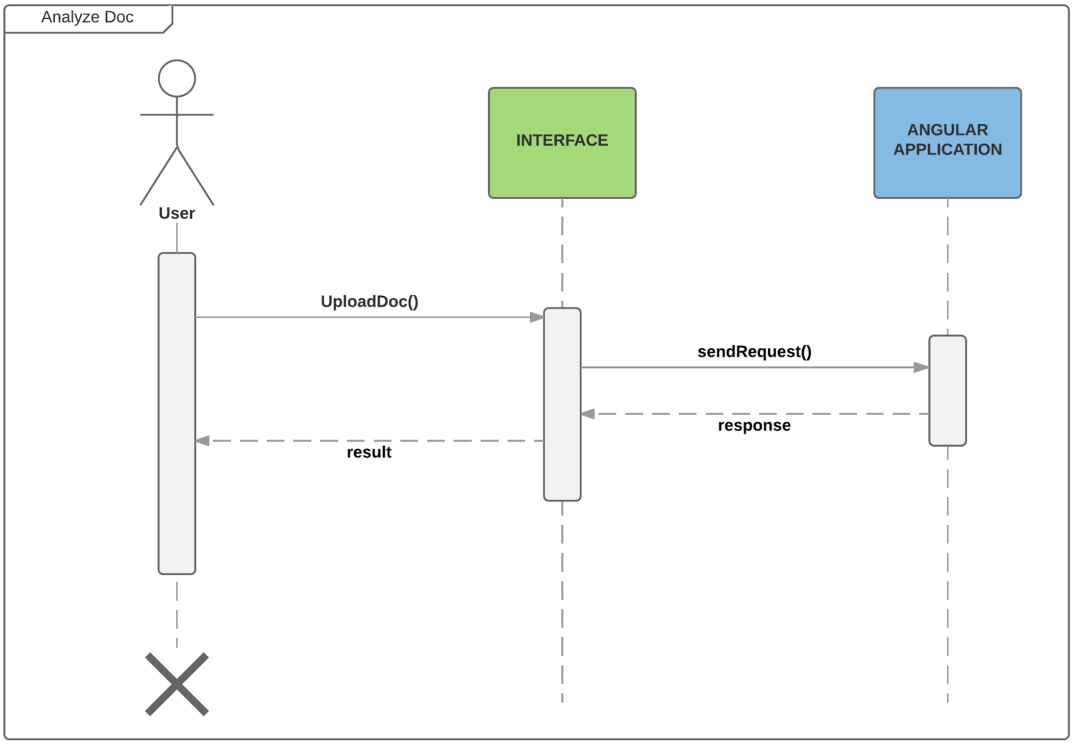
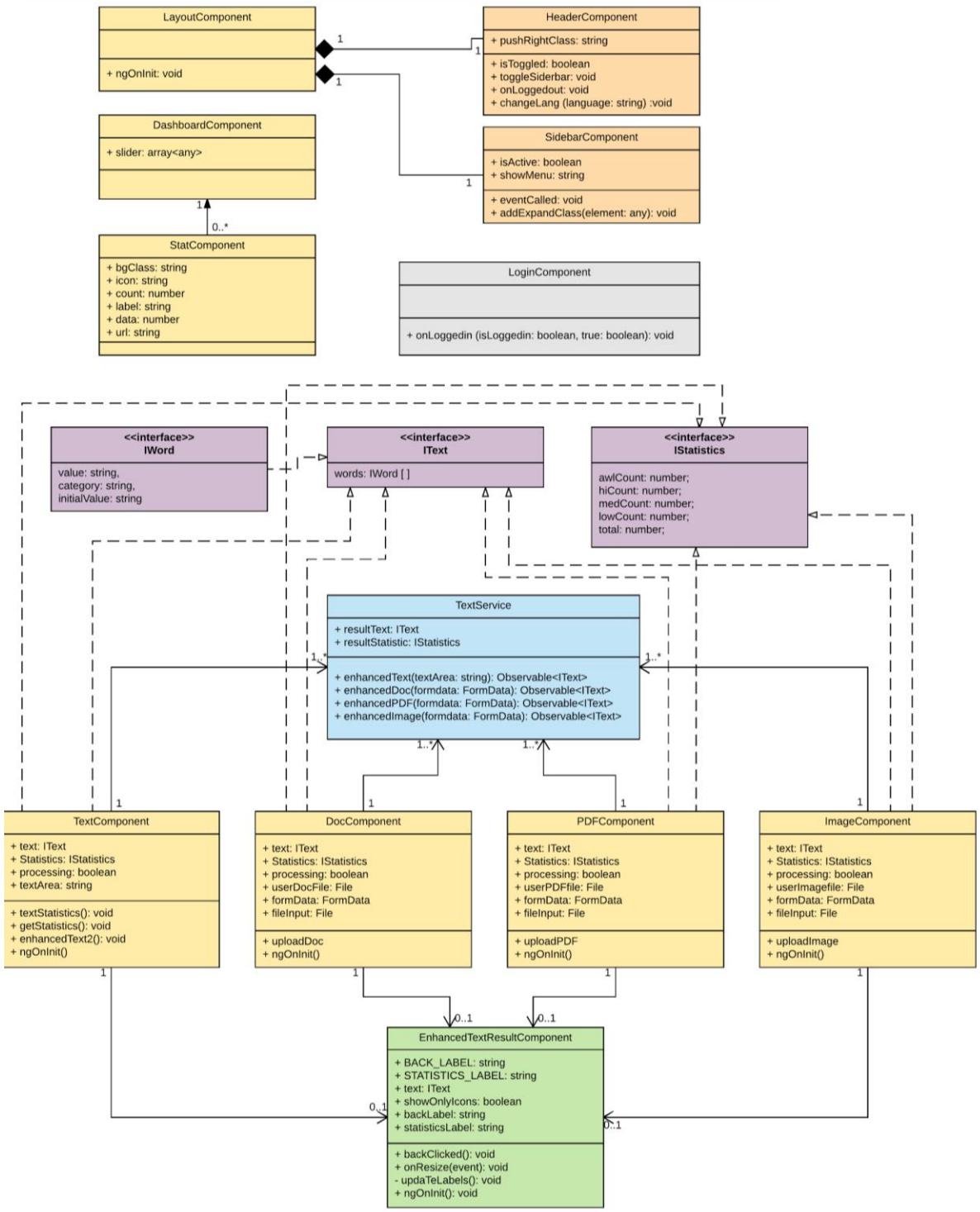


Figure 21 - #179 Analyze doc - Sequence diagram


LayoutComponent
+ ngOnInit: void
HeaderComponent
+ pushRightClass: string
+ isToggled: boolean
+ toggleSiderbar: void
+ onLoggedout: void
+ changeLang (language: string) :void
DashboardComponent
+ slider: array<any>
SidebarComponent
+ isActive: boolean
+ showMenu: string
+ eventCalled: void
+ addExpandClass(element: any): void
StatComponent
+ bgClass: string
+ icon: string
+ count: number
+ label: string
+ data: number
+ url: string
LoginComponent
+ onLoggedin (isLoggedin: boolean, true: boolean): void
<<interface>>
IWord
value: string,
category: string,
initialValue: string
<<interface>>
IText
words: IWord [ ]
<<interface>>
IStatistics
awlCount: number;
hiCount: number;
medCount: number;
lowCount: number;
total: number;
TextService
+ resultText: IText
+ resultStatistic: IStatistics
+ enhancedText(textArea: string): Observable<IText>
+ enhancedDoc(formdata: FormData): Observable<IText>
+ enhancedPDF(formdata: FormData): Observable<IText>
+ enhancedImage(formdata: FormData): Observable<IText>
TextComponent
+ text: IText
+ Statistics: IStatistics
+ processing: boolean
+ textArea: string
+ textStatistics(): void
+ getStatistics(): void
+ enhancedText2(): void
+ ngOnInit()
DocComponent
+ text: IText
+ Statistics: IStatistics
+ processing: boolean
+ userDocFile: File
+ formData: FormData
+ fileInput: File
+ uploadDoc
+ ngOnInit()
PDFComponent
+ text: IText
+ Statistics: IStatistics
+ processing: boolean
+ userPDFfile: File
+ formData: FormData
+ fileInput: File
+ uploadPDF
+ ngOnInit()
ImageComponent
+ text: IText
+ Statistics: IStatistics
+ processing: boolean
+ userImagefile: File
+ formData: FormData
+ fileInput: File
+ uploadImage
+ ngOnInit()
EnhancedTextResultComponent
+ BACK_LABEL: string
+ STATISTICS_LABEL: string
+ text: IText
+ showOnlyIcons: boolean
+ backLabel: string
+ statisticsLabel: string
+ backClicked(): void
+ onResize(event): void
- updaTeLabels(): void
+ ngOnInit(): void

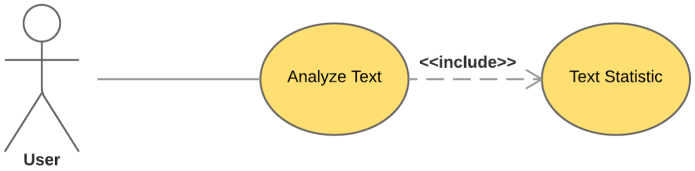


Figure 22 - #184 Create a page for statistic page - use case diagram

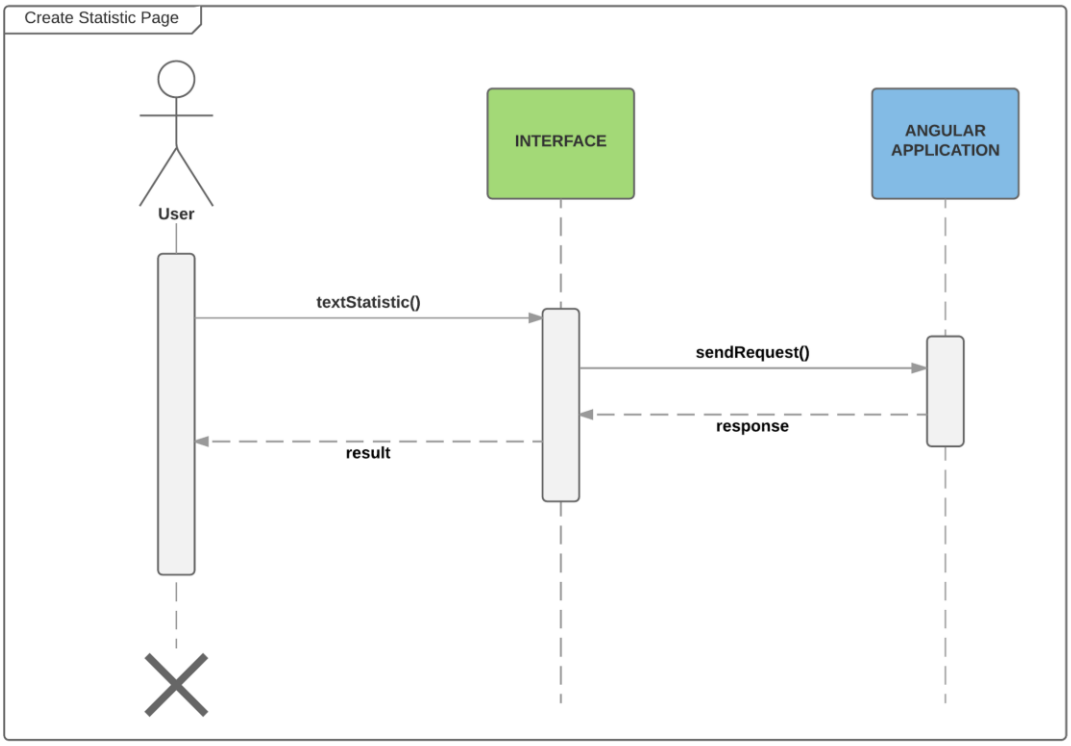


Figure 23 - #184 Create a page for statistic page - Sequence diagram

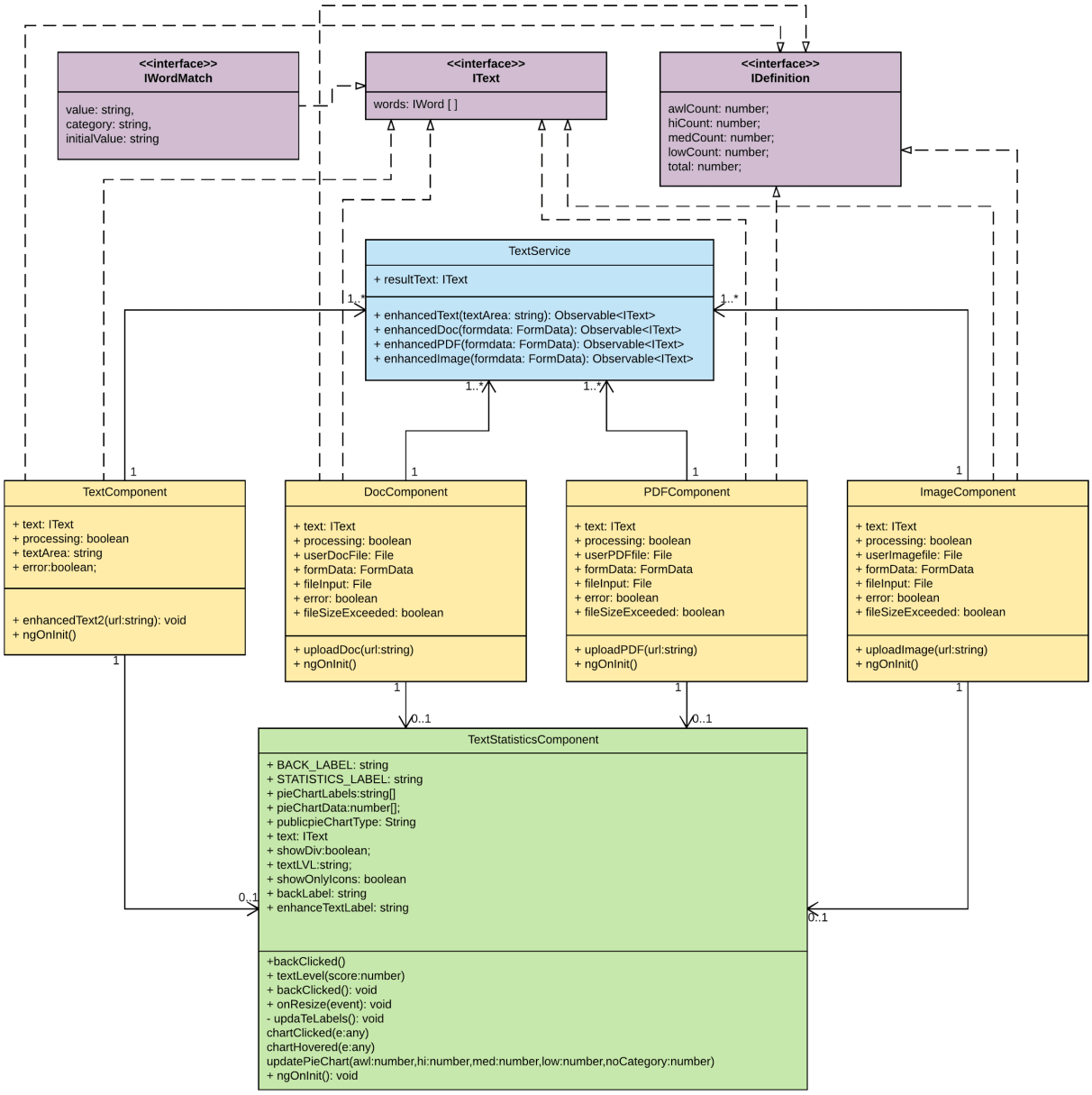


Figure 24 - #184 Create a page for statistic page - Class diagram

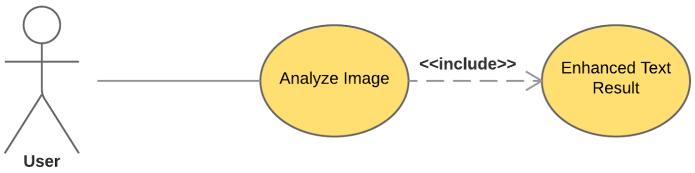


Figure 25 - #214 Analyze image - Use case diagram

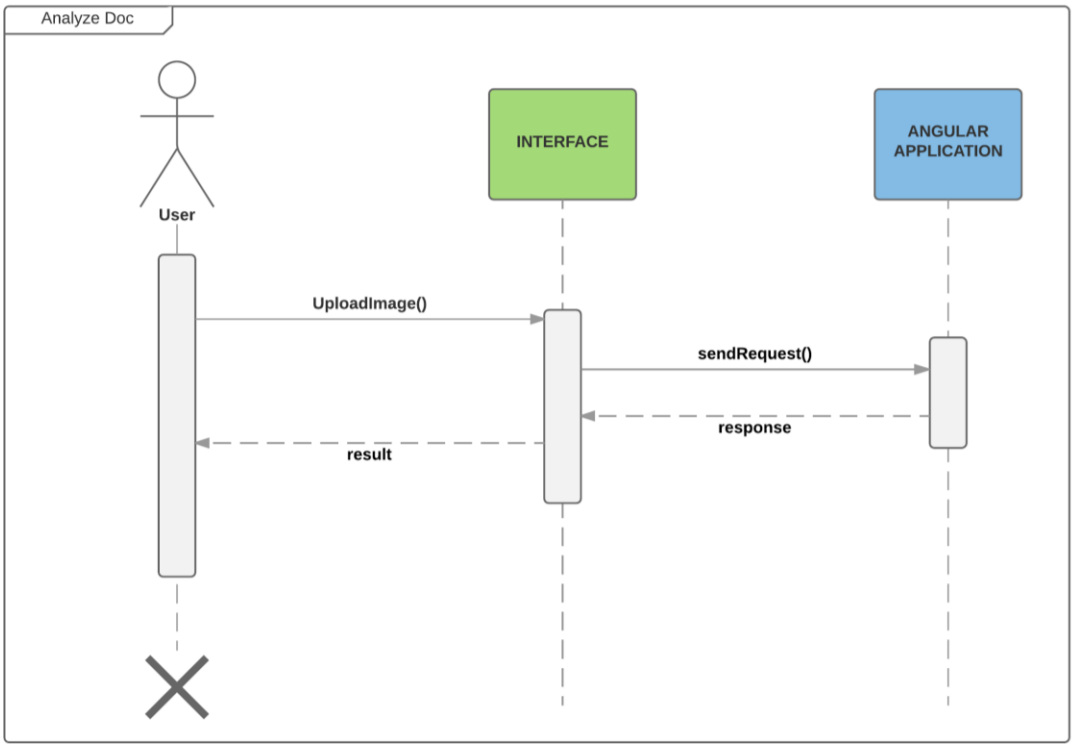


Figure 26 - #214 Analyze image - Sequence diagram

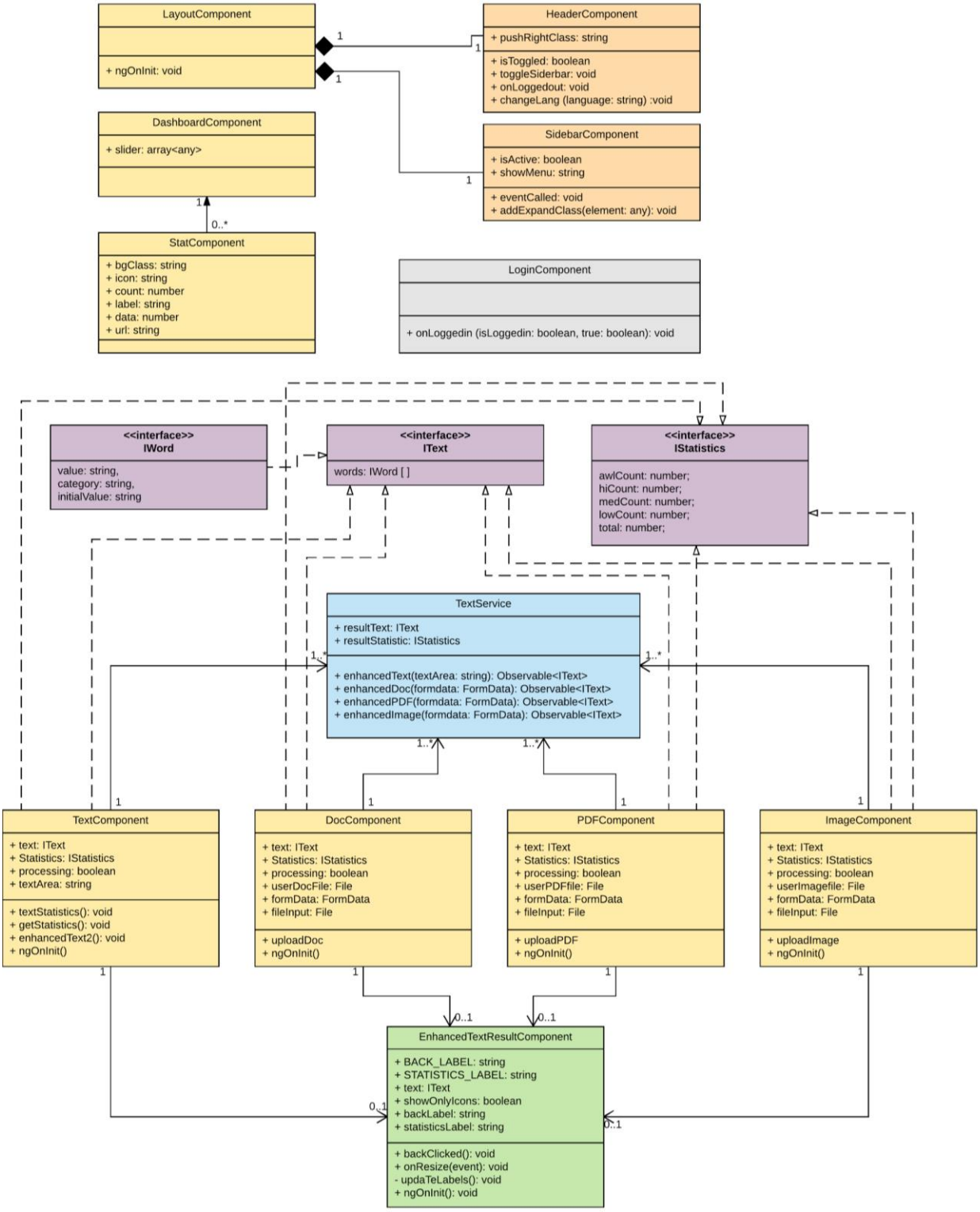

LayoutComponent
+ ngOnInit: void
HeaderComponent
+ pushRightClass: string
+ isToggled: boolean
+ toggleSiderbar: void
+ onLoggedout: void
+ changeLang (language: string) :void
DashboardComponent
+ slider: array<any>
SidebarComponent
+ isActive: boolean
+ showMenu: string
+ eventCalled: void
+ addExpandClass(element: any): void
StatComponent
+ bgClass: string
+ icon: string
+ count: number
+ label: string
+ data: number
+ url: string
LoginComponent
+ onLoggedin (isLoggedin: boolean, true: boolean): void
<<interface>>
IWord
value: string,
category: string,
initialValue: string
<<interface>>
IText
words: IWord [ ]
<<interface>>
IStatistics
awlCount: number;
hiCount: number;
medCount: number;
lowCount: number;
total: number;
TextService
+ resultText: IText
+ resultStatistic: IStatistics
+ enhancedText(textArea: string): Observable<IText>
+ enhancedDoc(formdata: FormData): Observable<IText>
+ enhancedPDF(formdata: FormData): Observable<IText>
+ enhancedImage(formdata: FormData): Observable<IText>
TextComponent
+ text: IText
+ Statistics: IStatistics
+ processing: boolean
+ textArea: string
+ textStatistics(): void
+ getStatistics(): void
+ enhancedText2(): void
+ ngOnInit()
DocComponent
+ text: IText
+ Statistics: IStatistics
+ processing: boolean
+ userDocFile: File
+ formData: FormData
+ fileInput: File
+ uploadDoc
+ ngOnInit()
PDFComponent
+ text: IText
+ Statistics: IStatistics
+ processing: boolean
+ userPDFfile: File
+ formData: FormData
+ fileInput: File
+ uploadPDF
+ ngOnInit()
ImageComponent
+ text: IText
+ Statistics: IStatistics
+ processing: boolean
+ userImagefile: File
+ formData: FormData
+ fileInput: File
+ uploadImage
+ ngOnInit()
EnhancedTextResultComponent
+ BACK_LABEL: string
+ STATISTICS_LABEL: string
+ text: IText
+ showOnlyIcons: boolean
+ backLabel: string
+ statisticsLabel: string
+ backClicked(): void
+ onResize(event): void
- updaTeLabels(): void
+ ngOnInit(): void

Figure 27 - #220 Create a word list page - Use case diagram

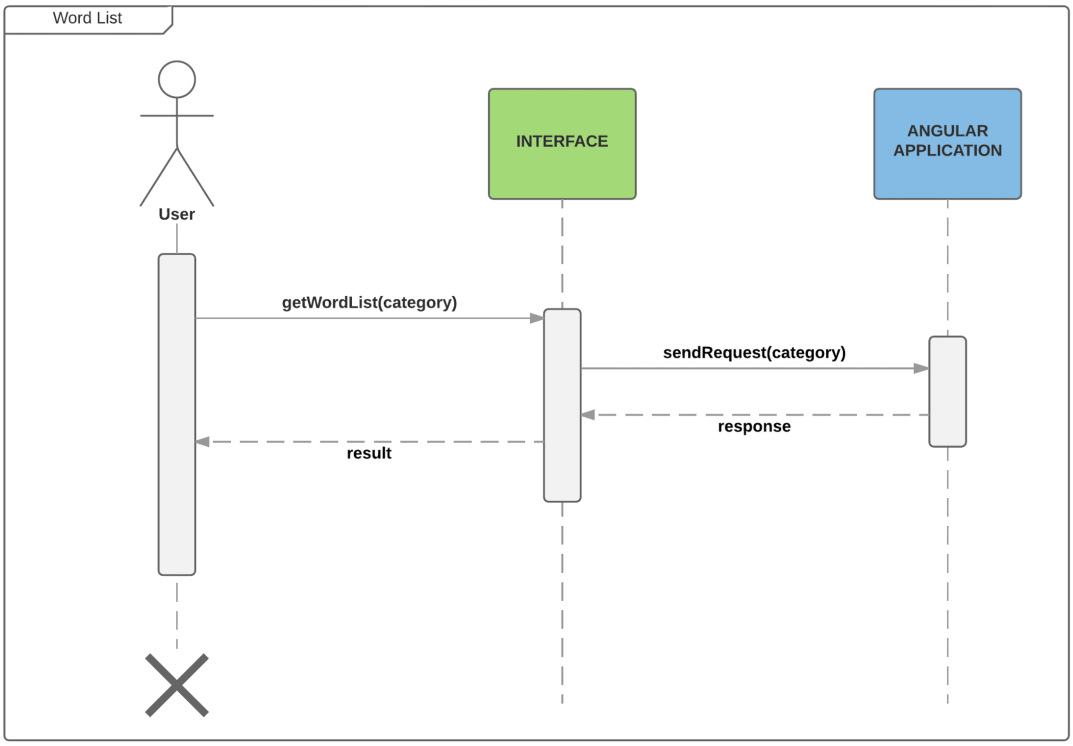


Figure 28 - #220 Create a word list page - Sequence diagram

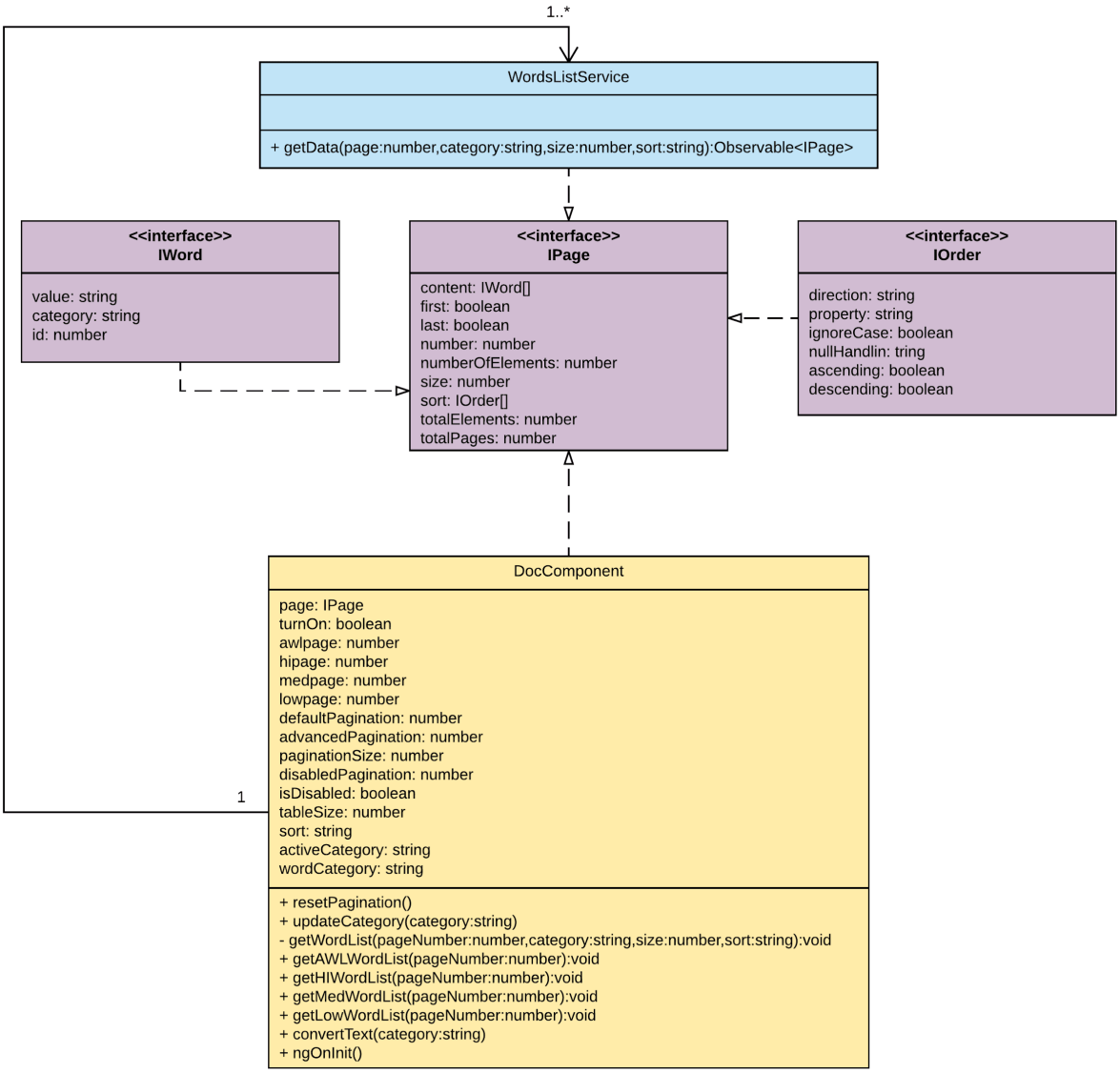


Figure 29- #220 Create a word list page - Class diagram

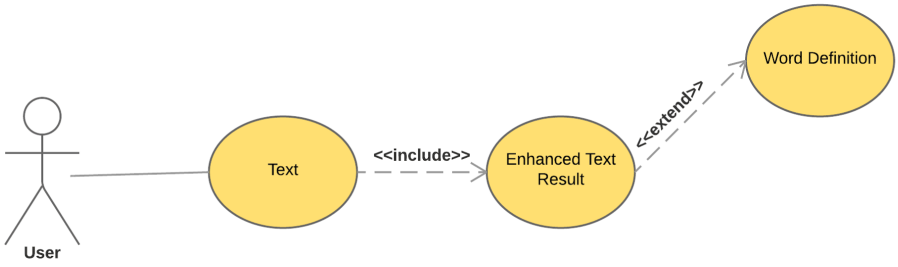


Figure 30 - #222 Implement word definition - Use case diagram

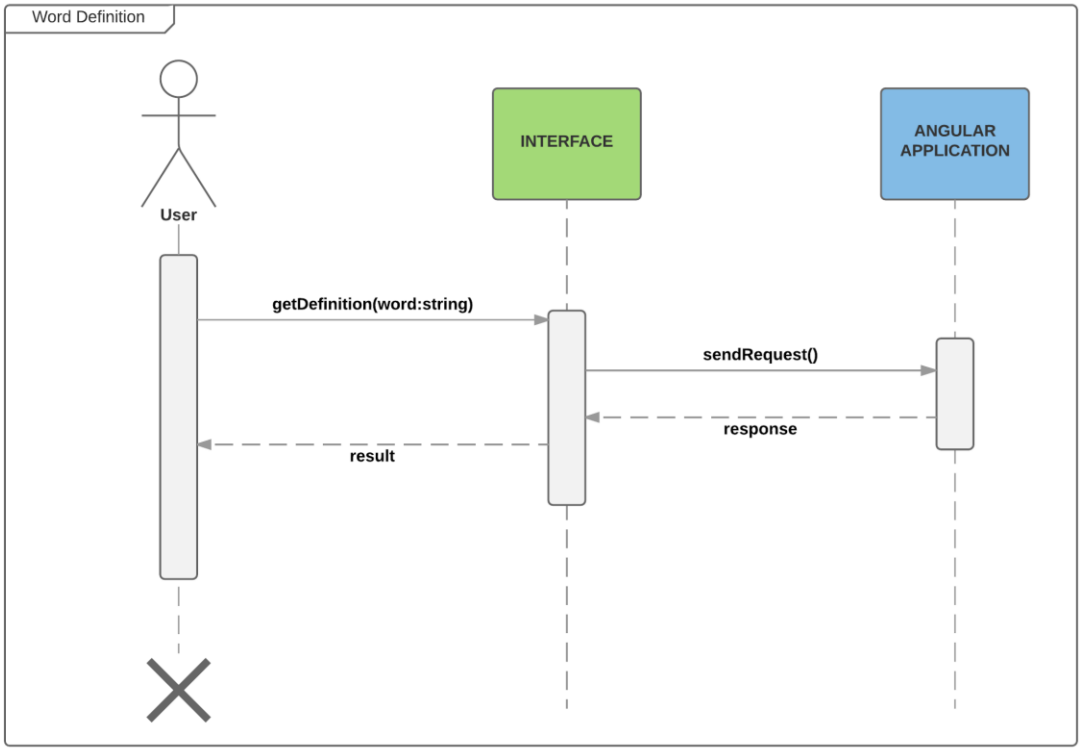


Figure 31 - #222 Implement word definition - Sequence diagram

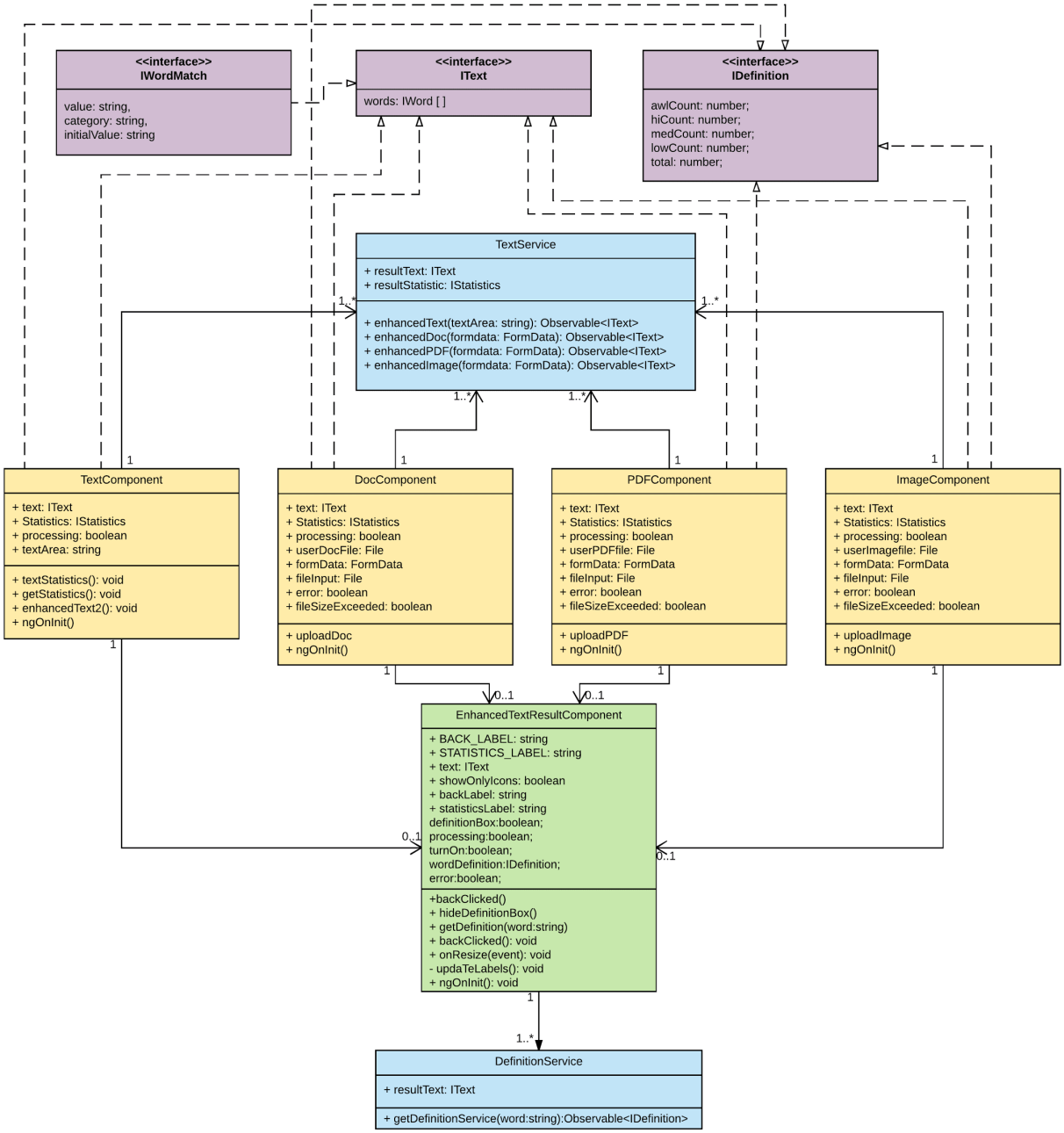


Figure 32- #222 Implement word definition - Class diagram

Figure 33- #224 Add contact us page - Use case diagram

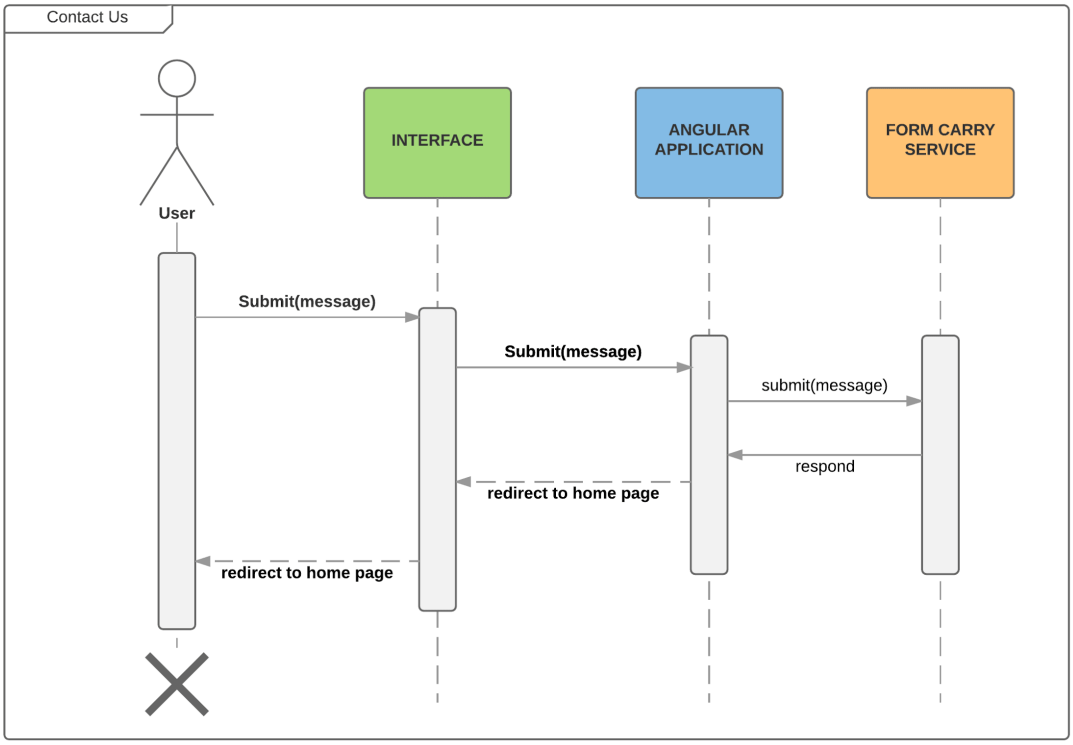


Figure 34- #224 Add contact us page - Sequence diagram

Figure 35 - #149 Show Application - Use case diagram

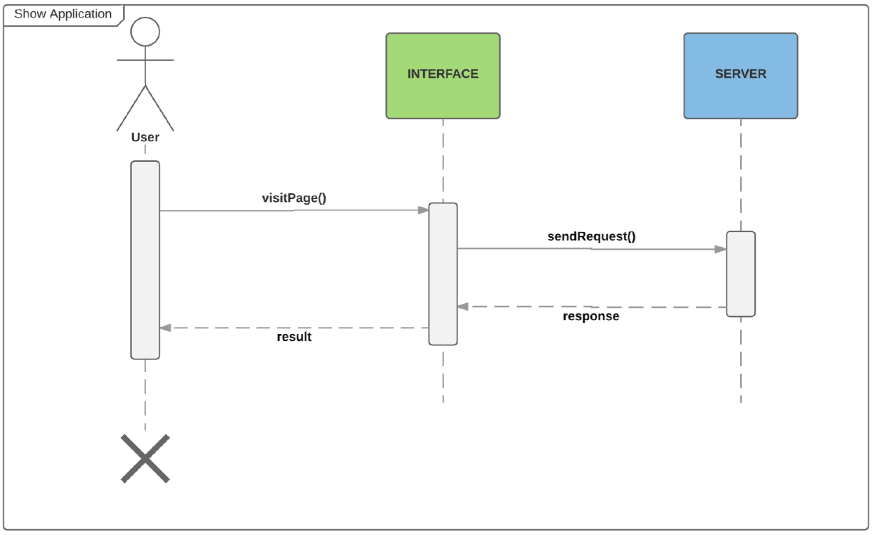


Figure 36- - #149 Show Application - Sequence diagram

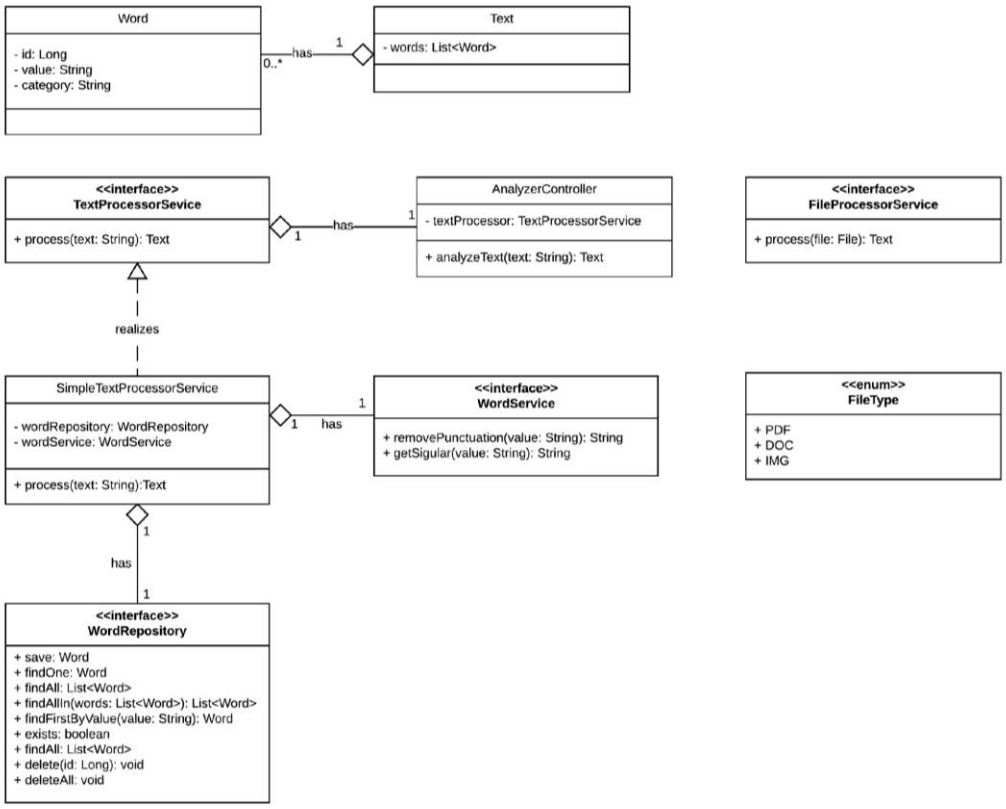


Figure 37 - - #149 Show Application - Class diagram

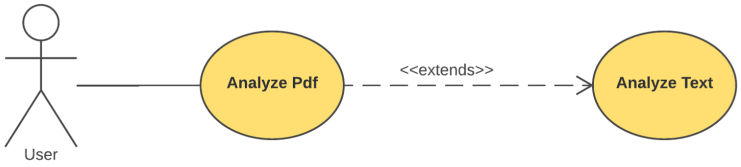


Figure 38 - #146 Analyze PDF -Use case diagram

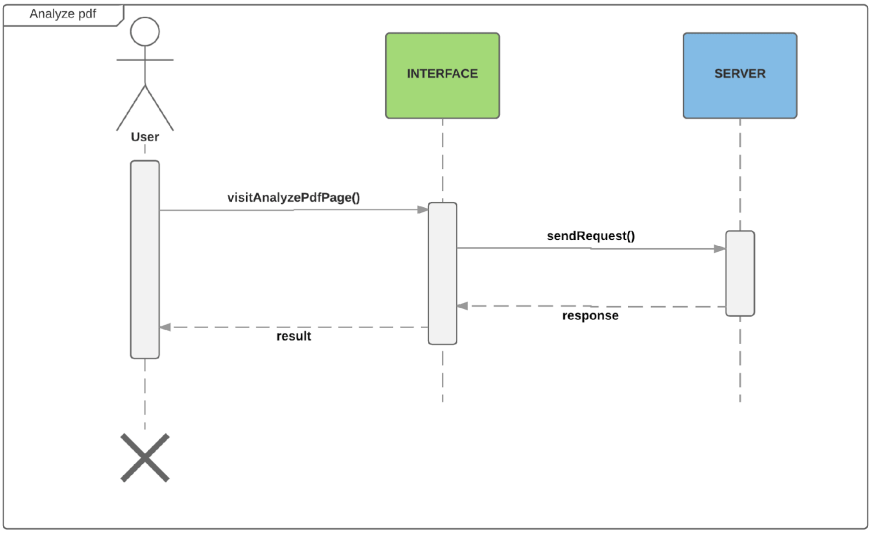


Figure 39 - #146 Analyze PDF - Sequence diagram

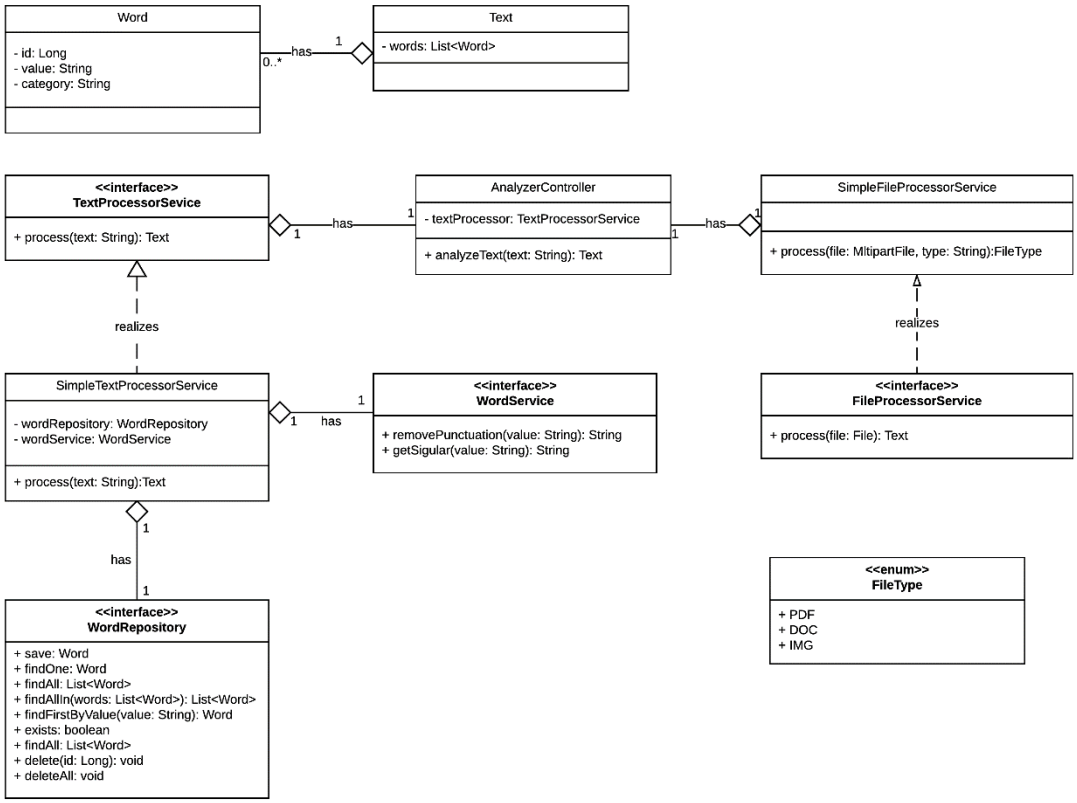


Figure 40 - #146 Analyze PDF - Class diagram

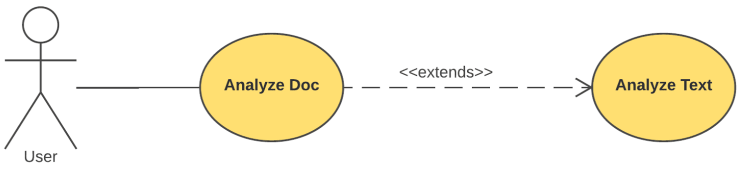


Figure 41 - #147 Analyze Doc - Use case diagram

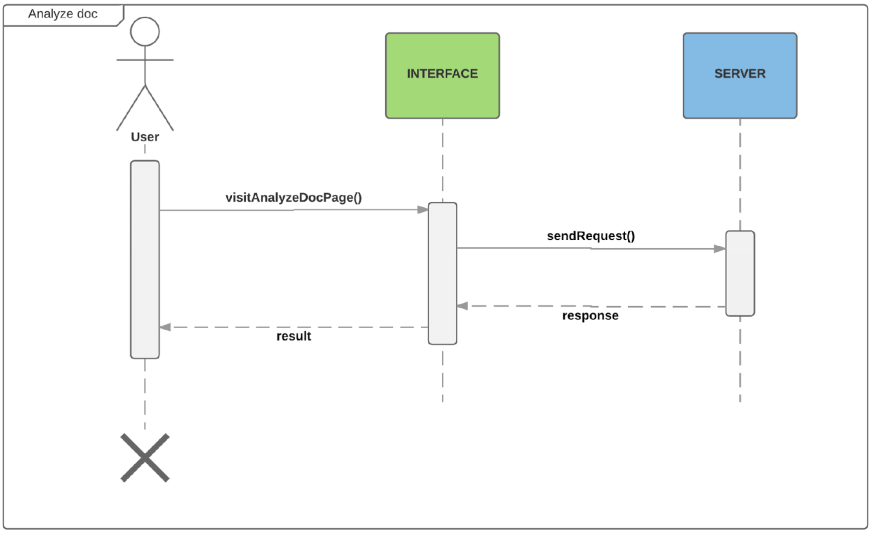


Figure 42 - #147 Analyze Doc -Sequence diagram

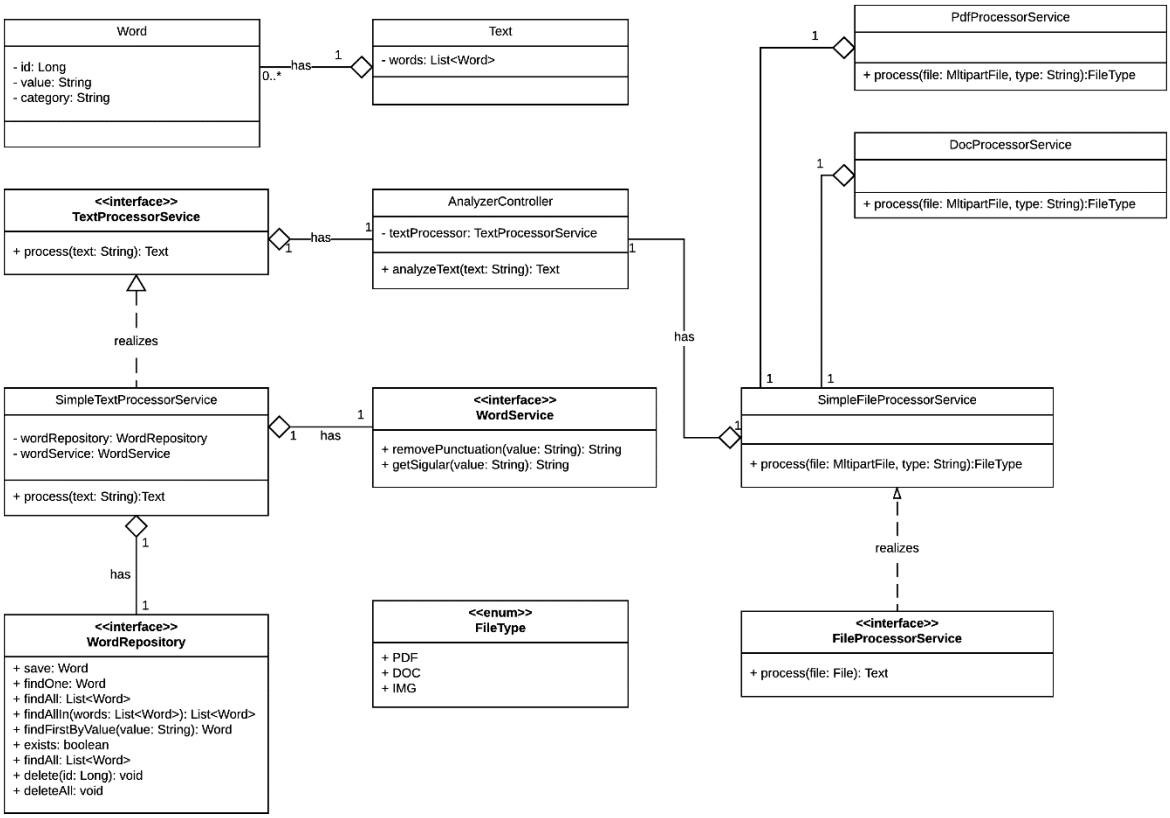


Figure 43 - #147Analyze Doc - Class Diagram

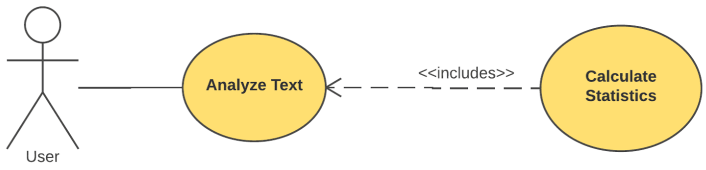


Figure 44- #225 Calculate Statistics - Use case diagram

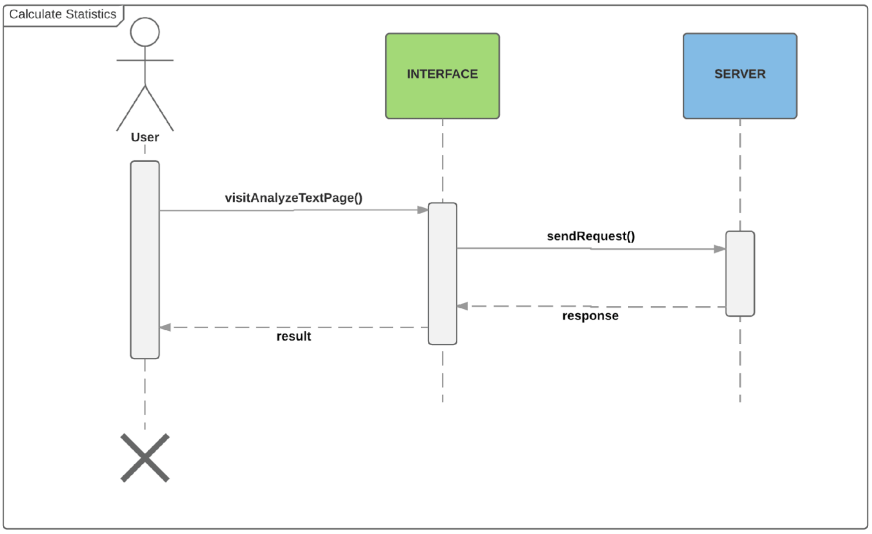


Figure 45- #225 Calculate Statistics - Sequence diagram

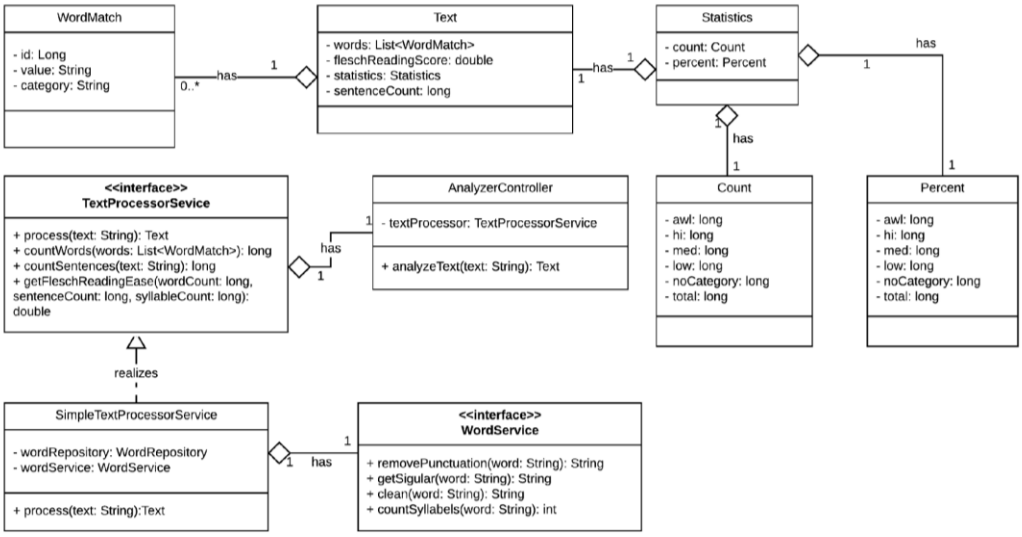


Figure 46 - #225 Calculate Statistics - Class diagram

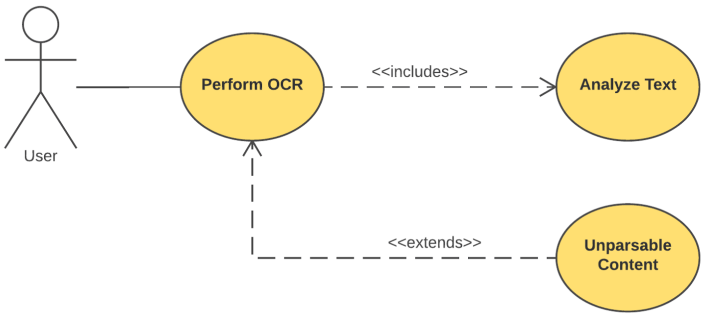


Figure 47- #132 Perform OCR - Use case

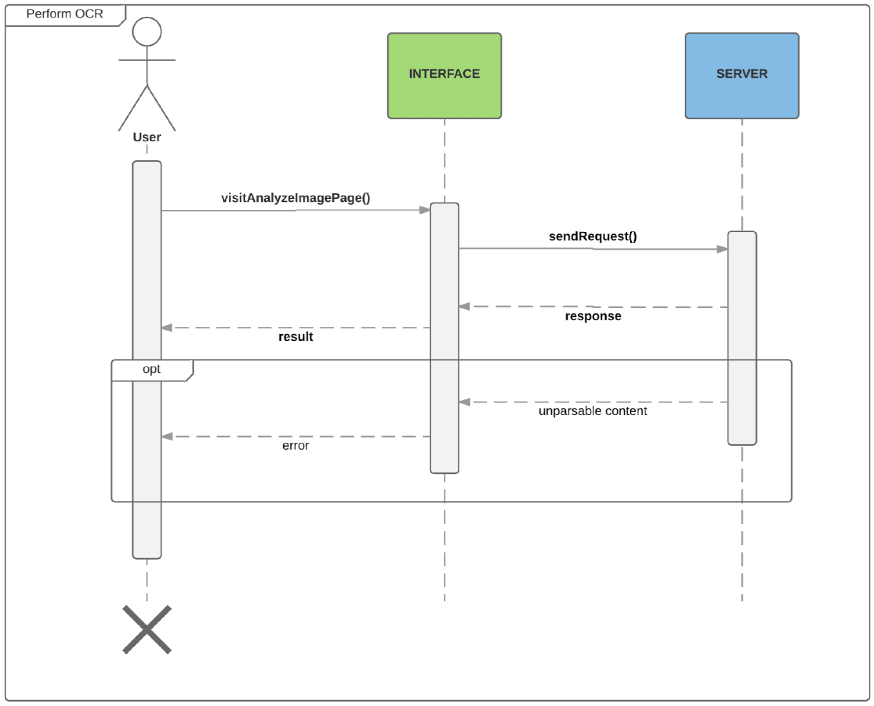


Figure 48- #132 Perform OCR - Sequence diagram

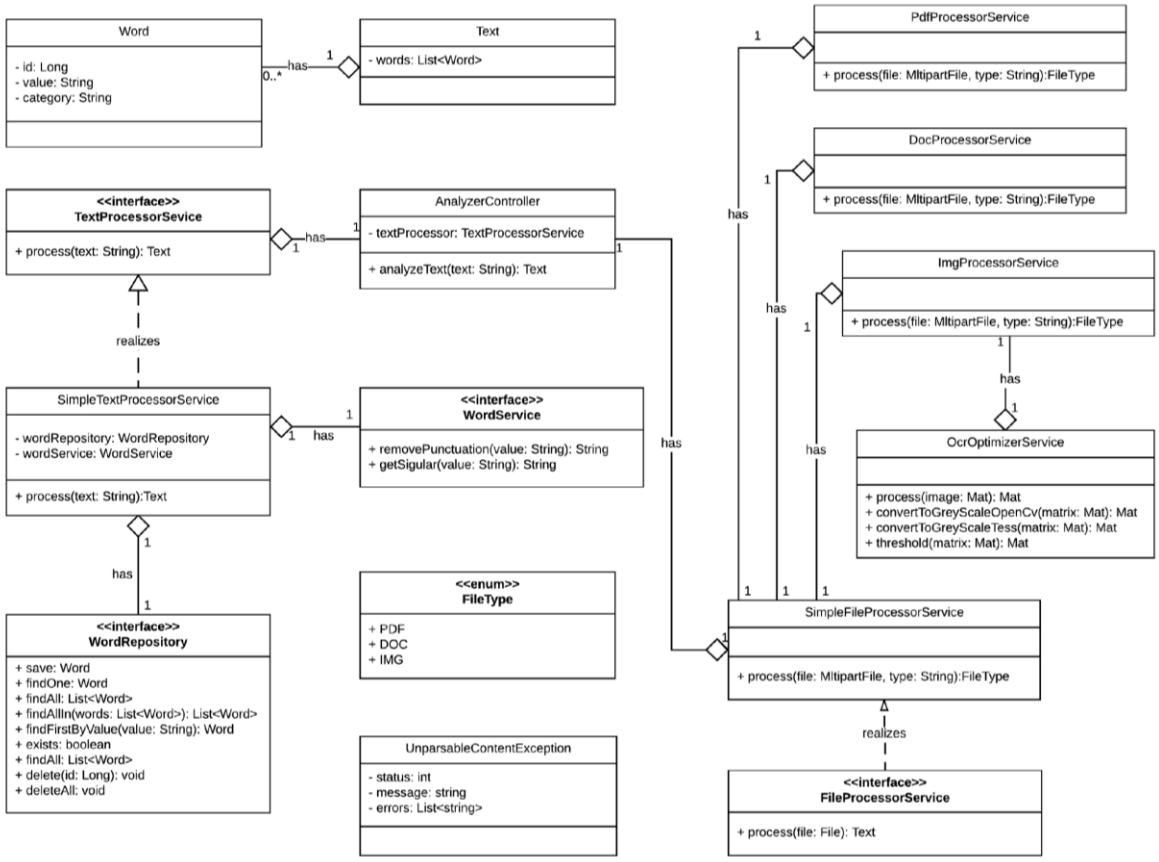


Figure 49- #132 Perform OCR - Class diagram

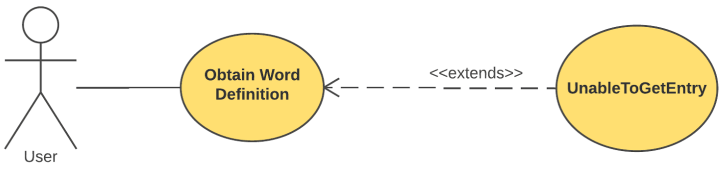


Figure 50- #151 Implement word definition - Use case

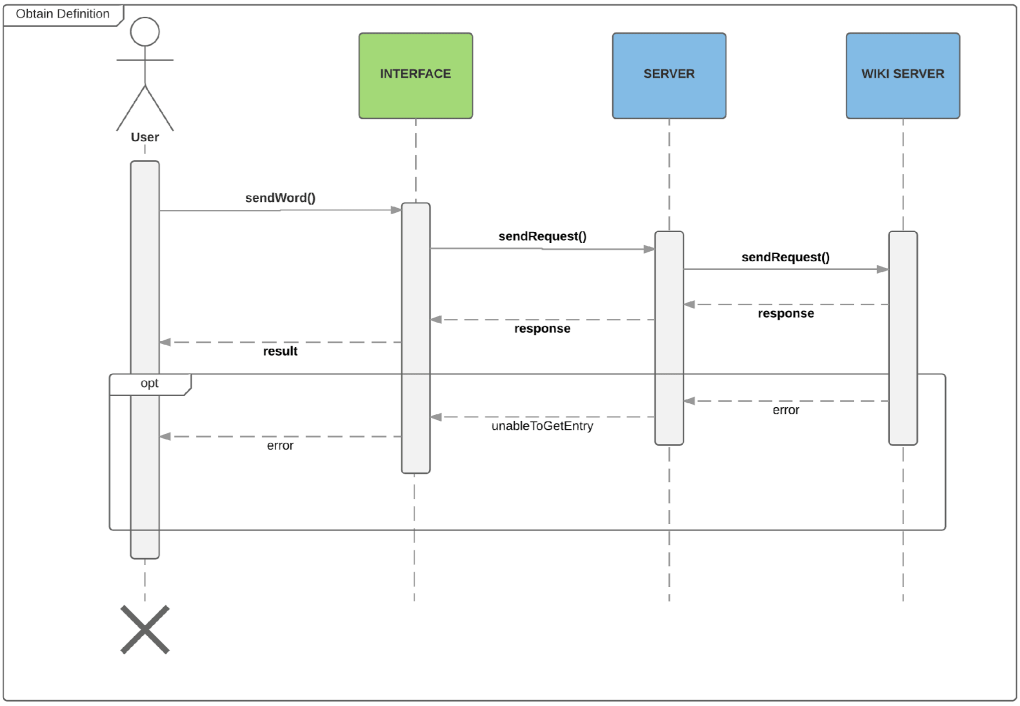


Figure 51- #151 Implement word definition - Sequence diagram

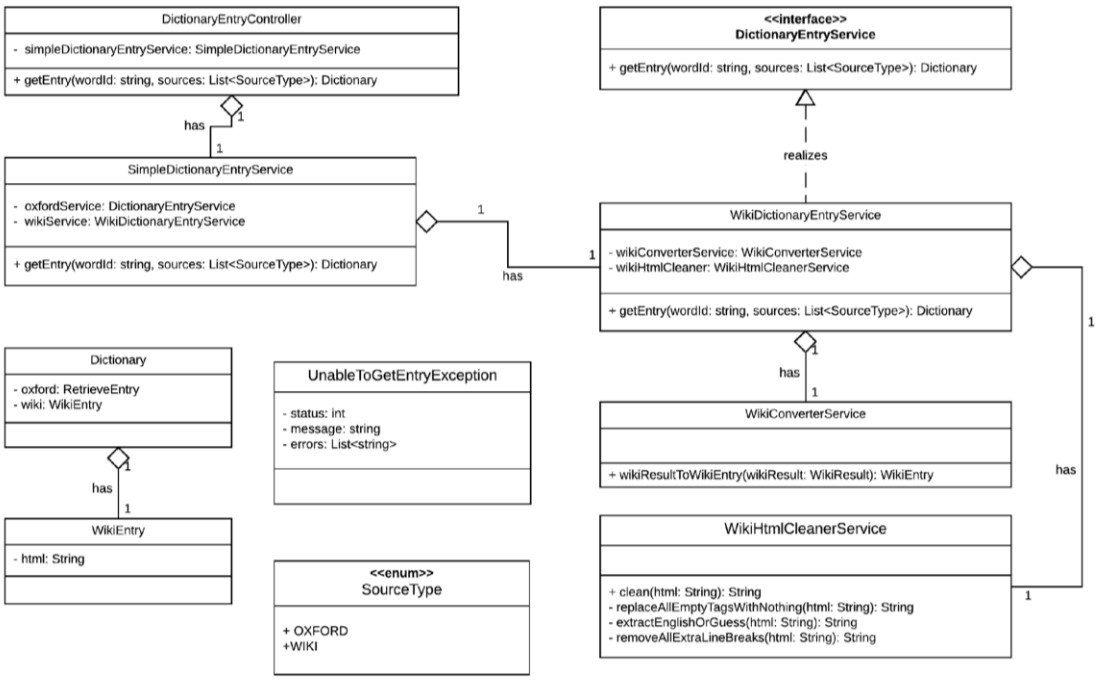


Figure 52 - #151 Implement word definition - Class diagram

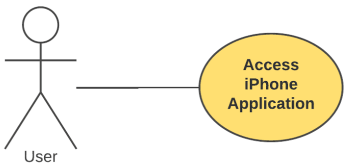


Figure 53- #227 Access iPhone app - Use case diagram

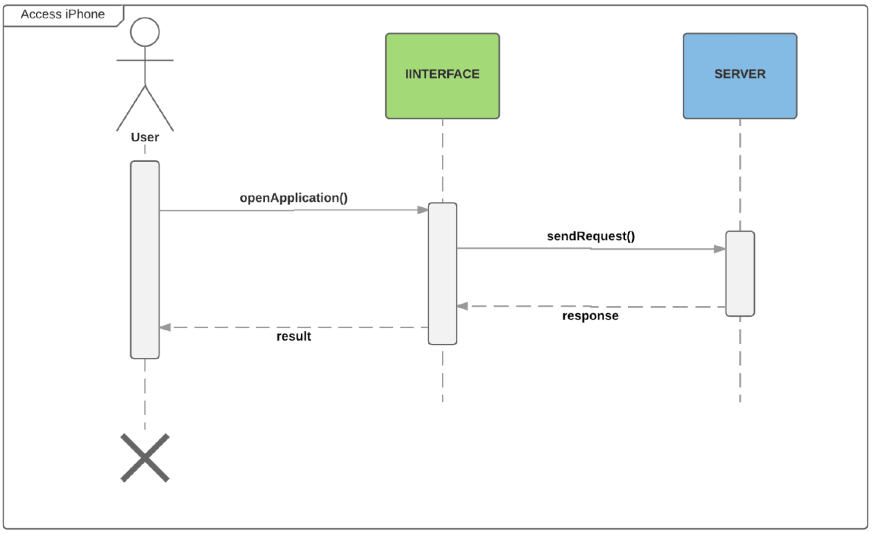


Figure 54- #227 Access iPhone App - Sequence diagram

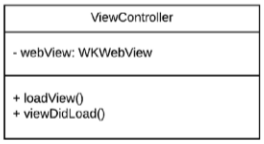


Figure 55- #227 Access iPhone App - Class diagram

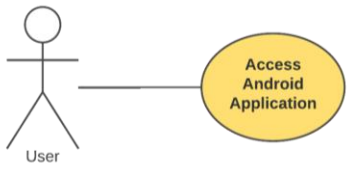


Figure 56 - #228 Access Android app - Use Case diagram

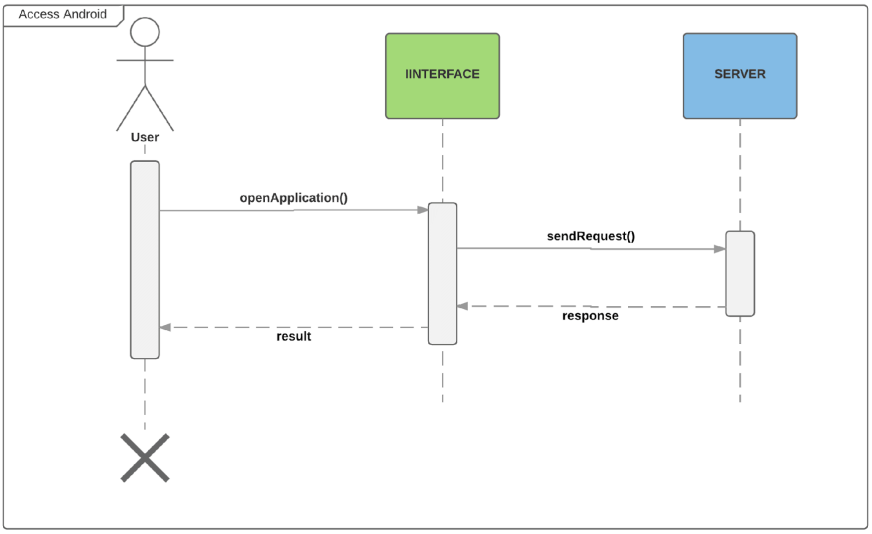


Figure 57 - #228 Access Android app - Sequence diagram

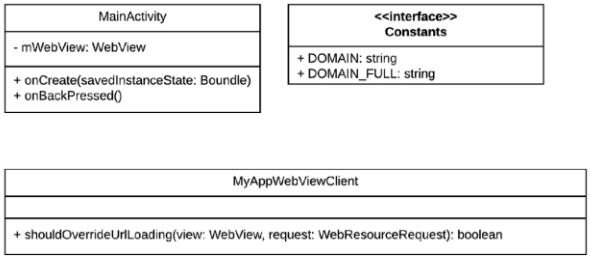


Figure 58 - #228 Access Android app - Class diagram

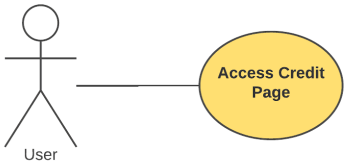


Figure 59- - #223 Access Credit Page - Use case diagram

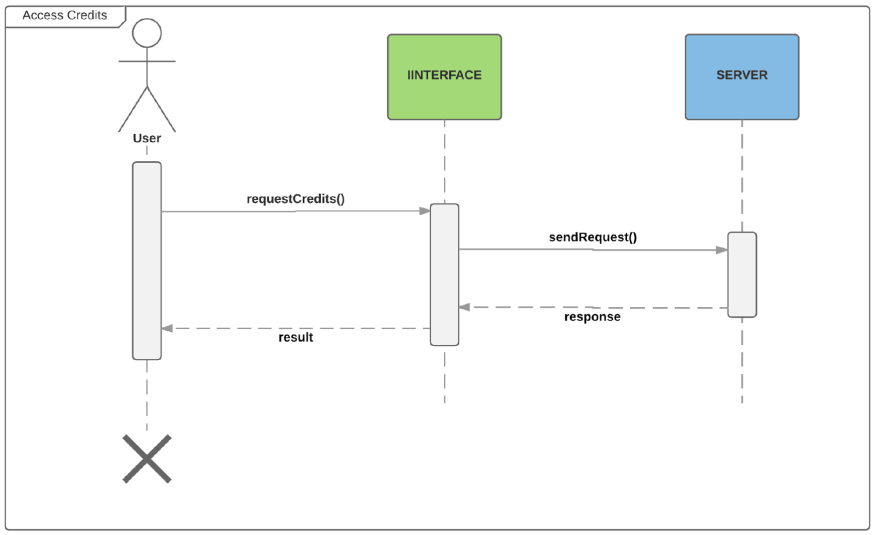


Figure 60- #223 Access Credit Page - Sequence diagram

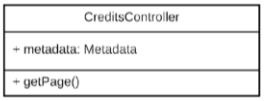


Figure 61- #223 Access Credit Page - Class diagram

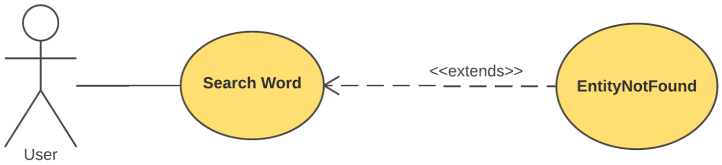


Figure 62- #243 Search word - Use Case diagram

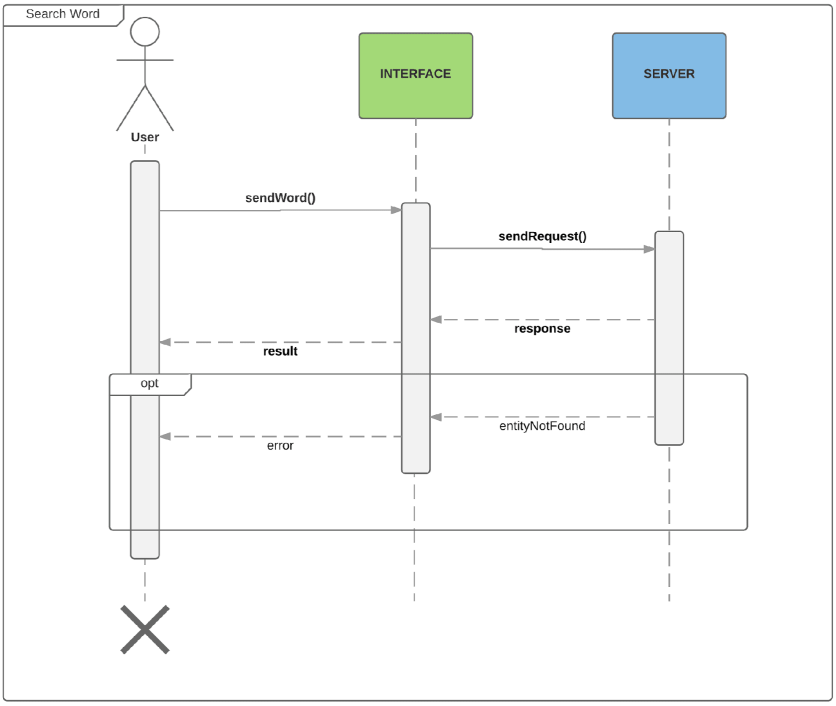


Figure 63- #243 Search word - Sequence diagram

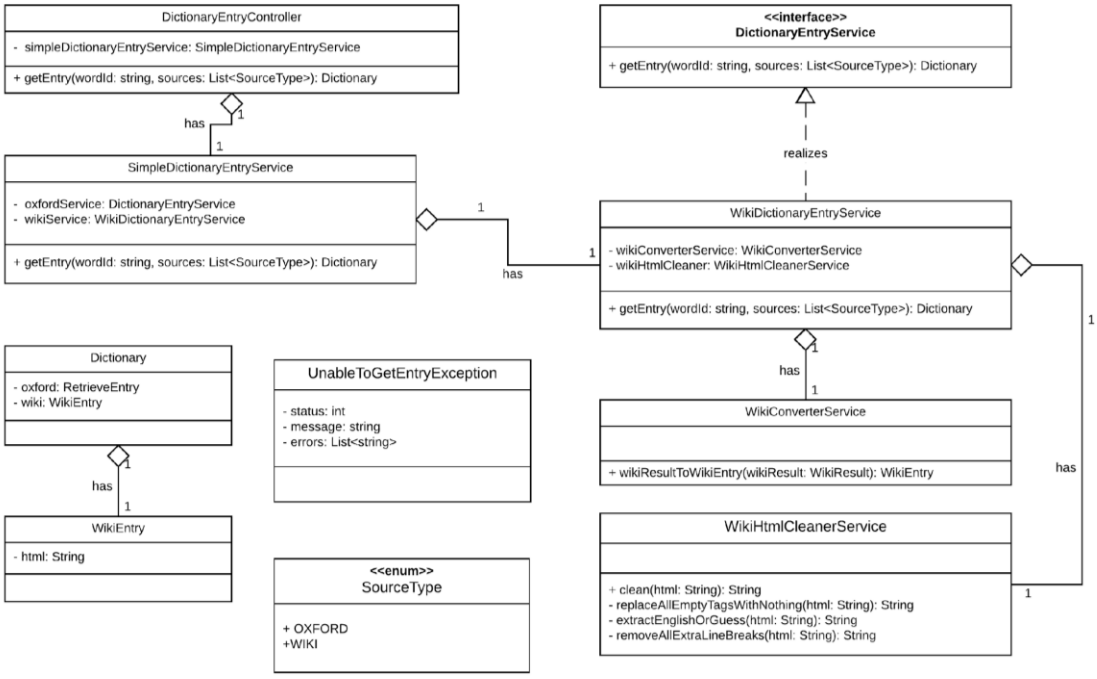


Figure 64- #243 Search word - Class diagram

Appendix B - User Interface Design

Computer

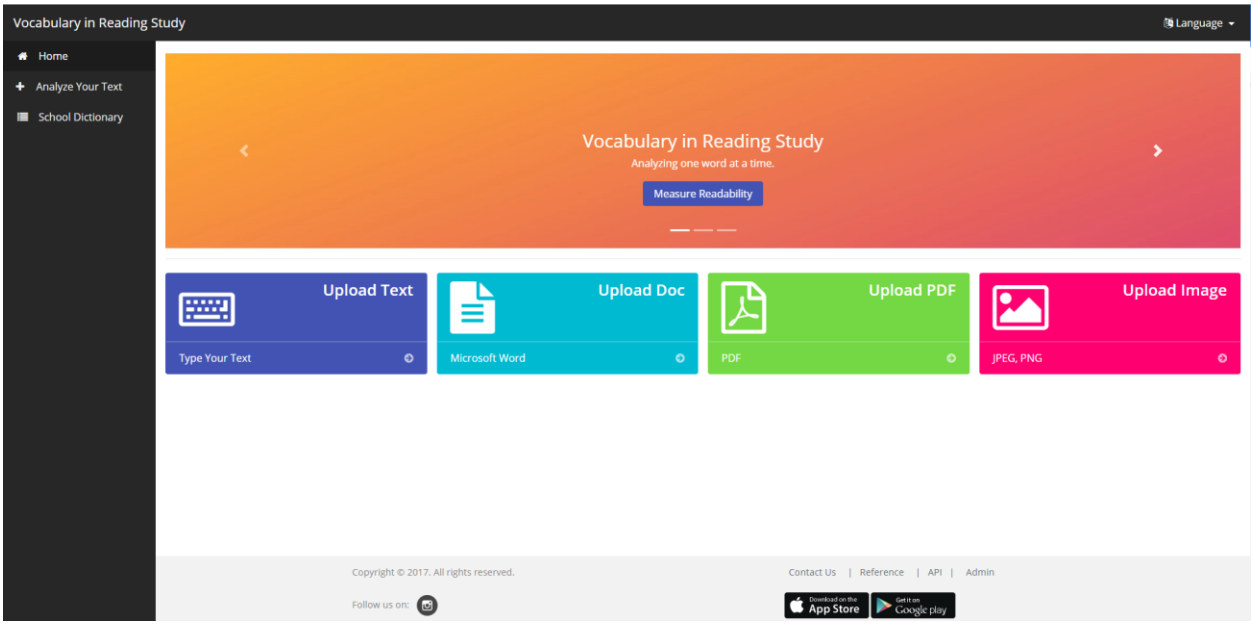


Figure 65- Myvirs.com Home Page

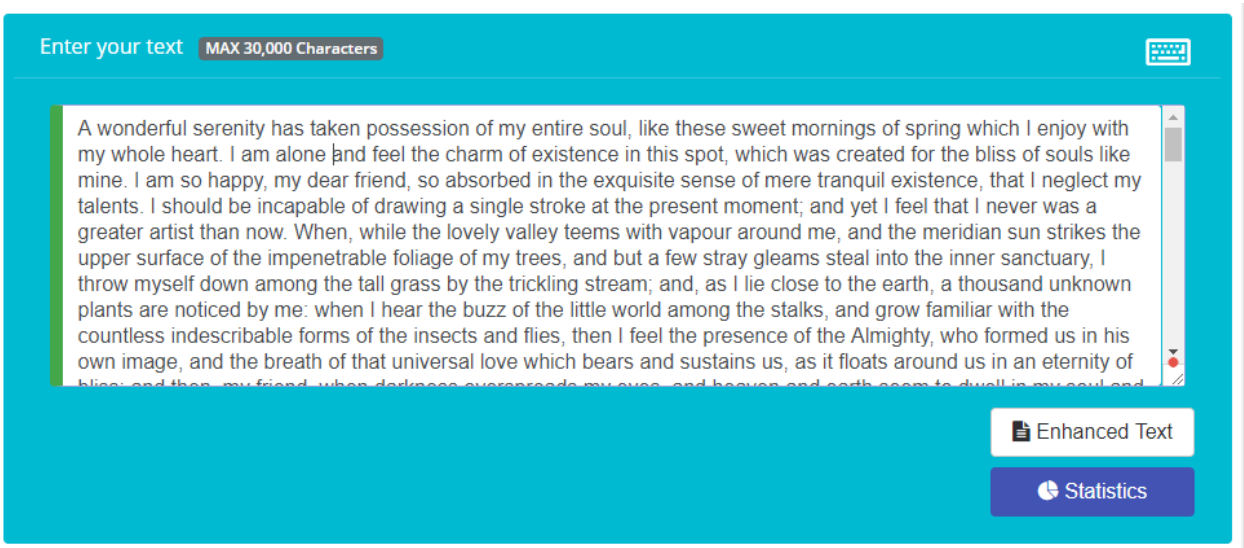


Figure 66- Text Input

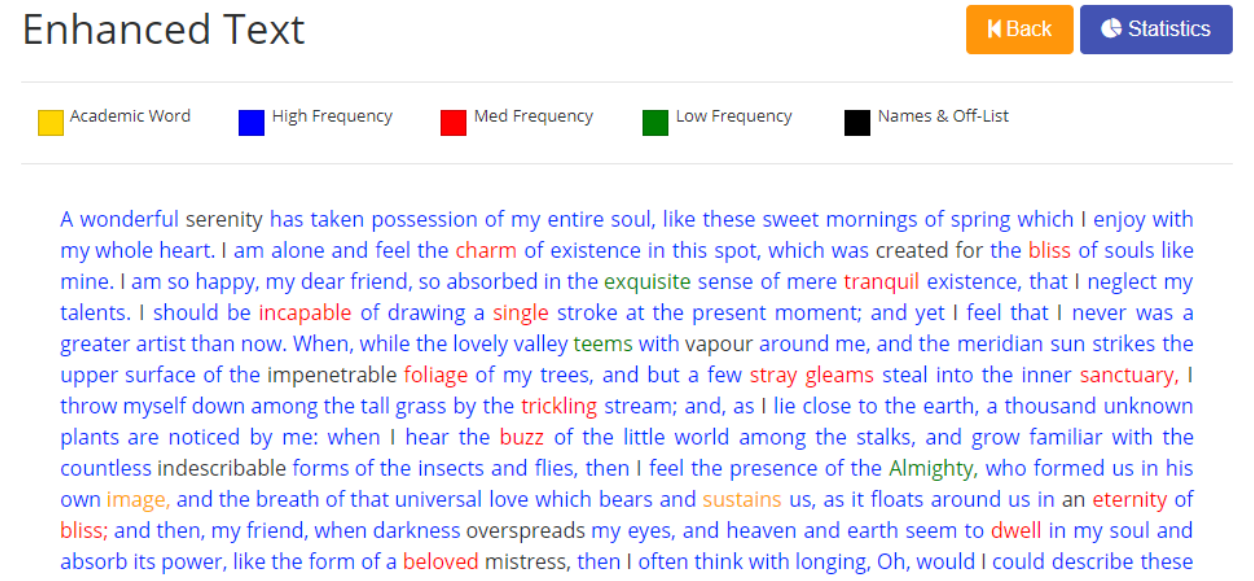


Figure 67 - Color-Coded Enhanced Text

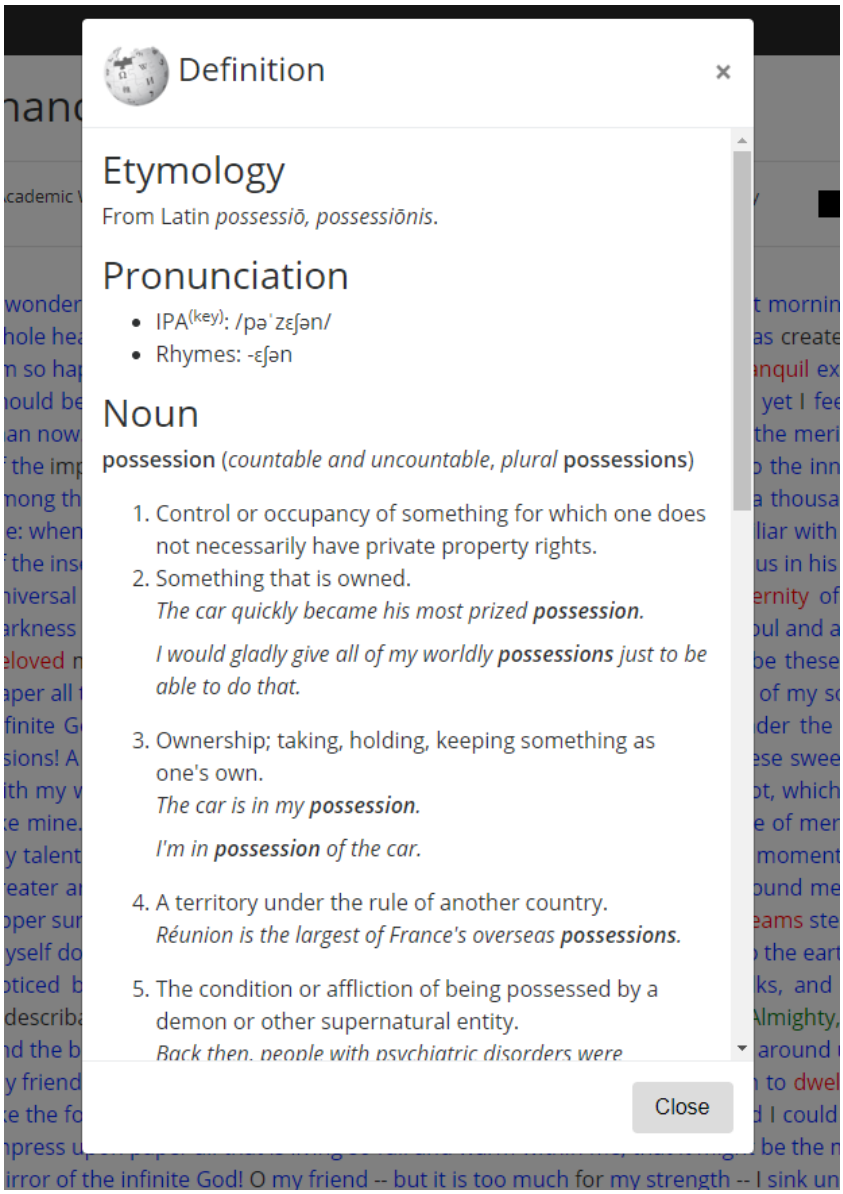


Figure 68 - Word Definition pop up

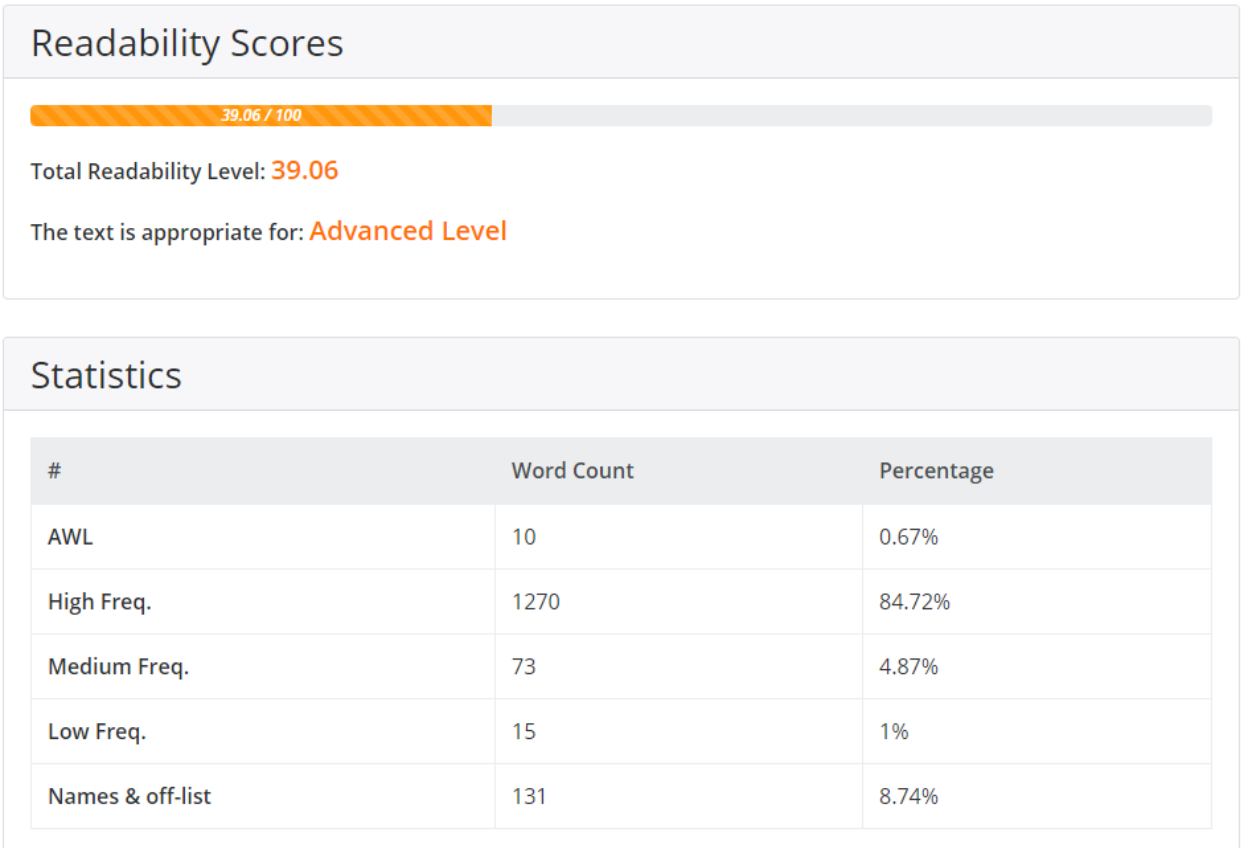


Figure 69 - Statistic page - Text analytic

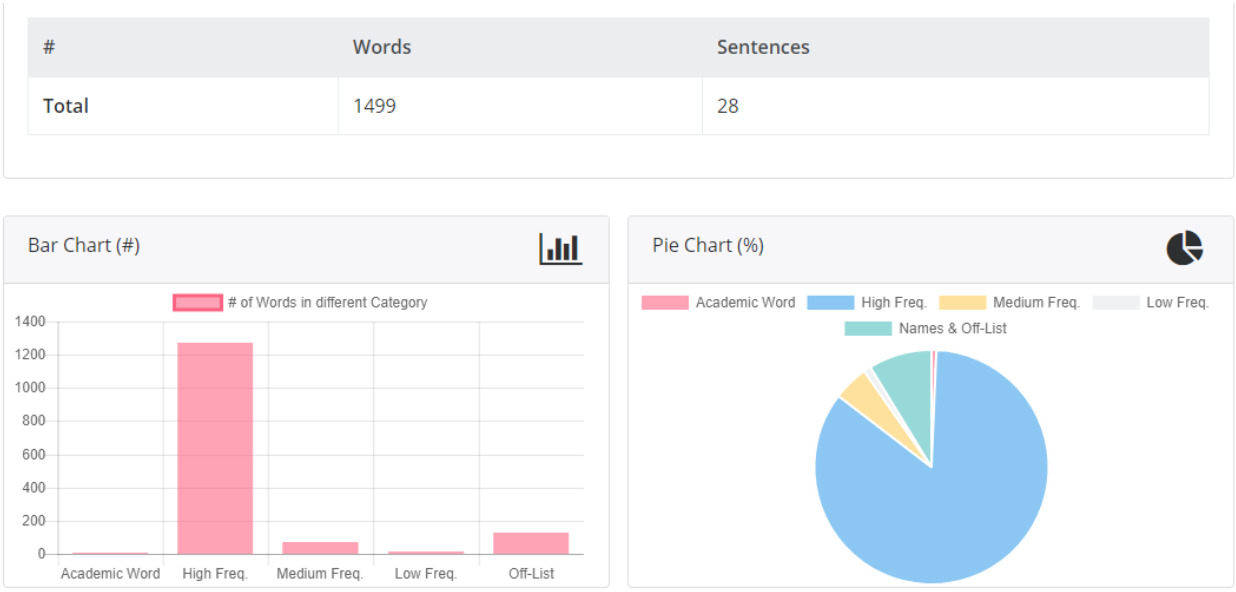


Figure 70 - Statistic page - Bar and Pie charts

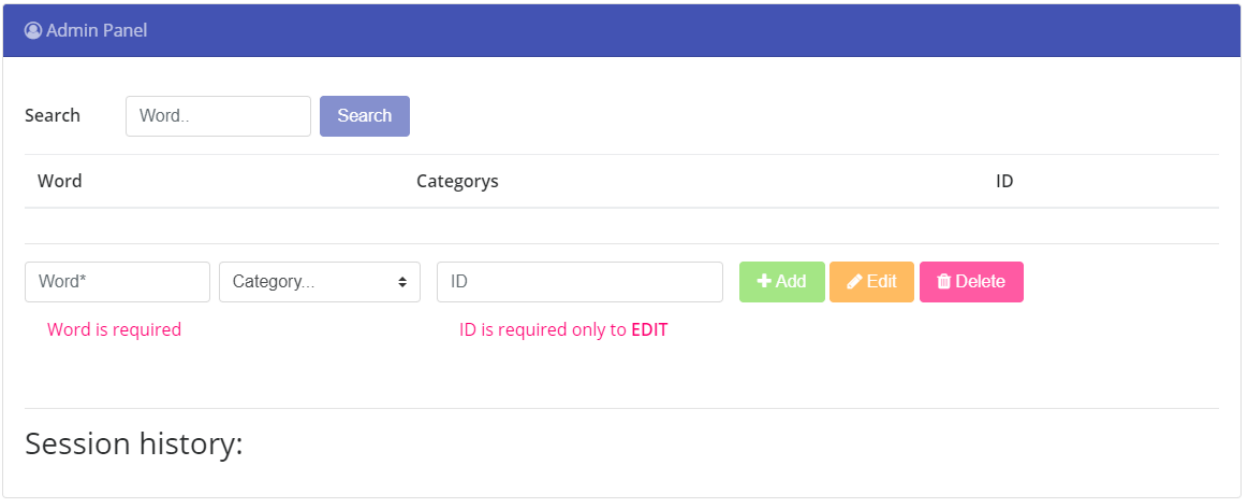


Figure 71- Admin dashboard

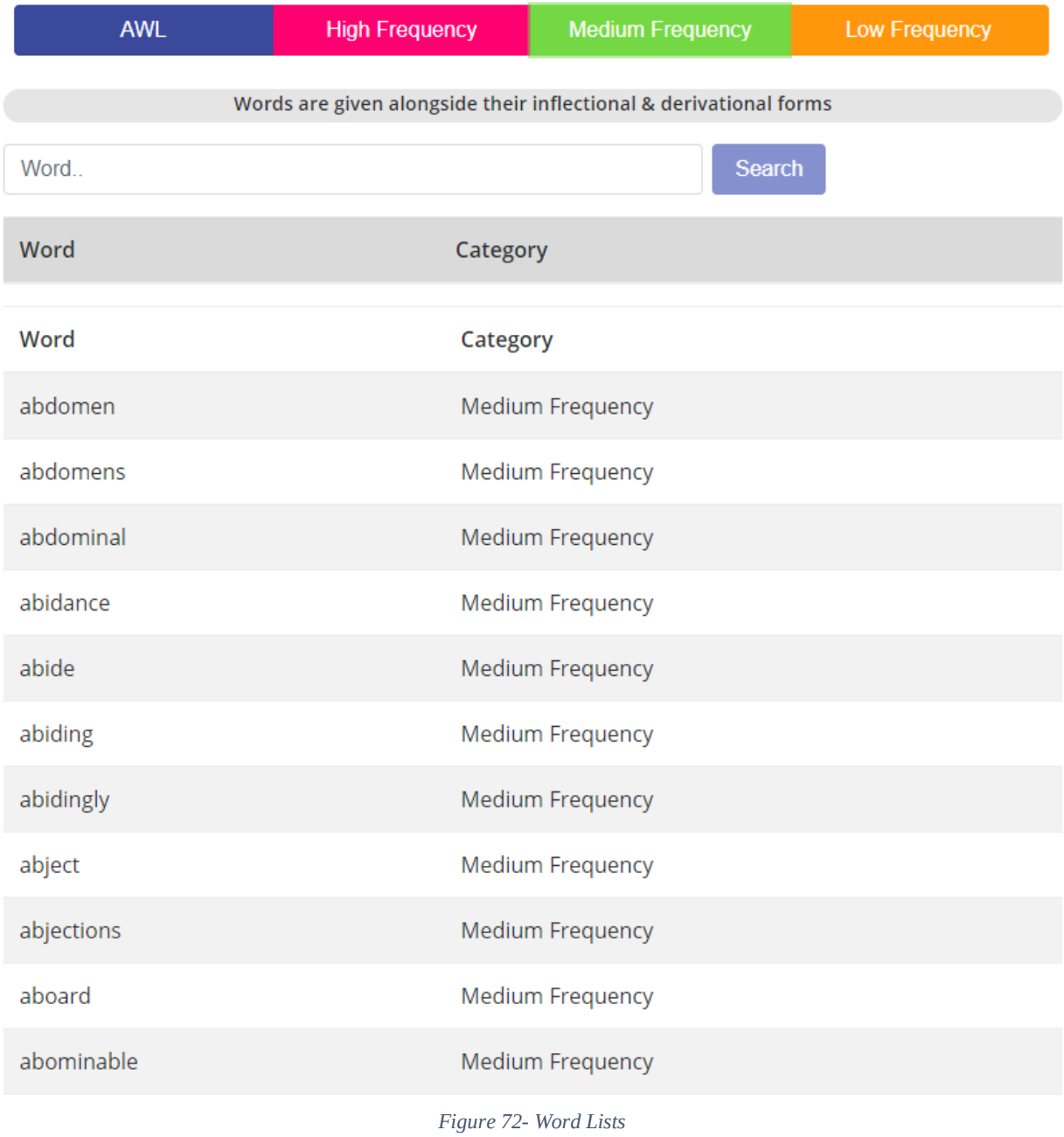


Figure 72- Word Lists

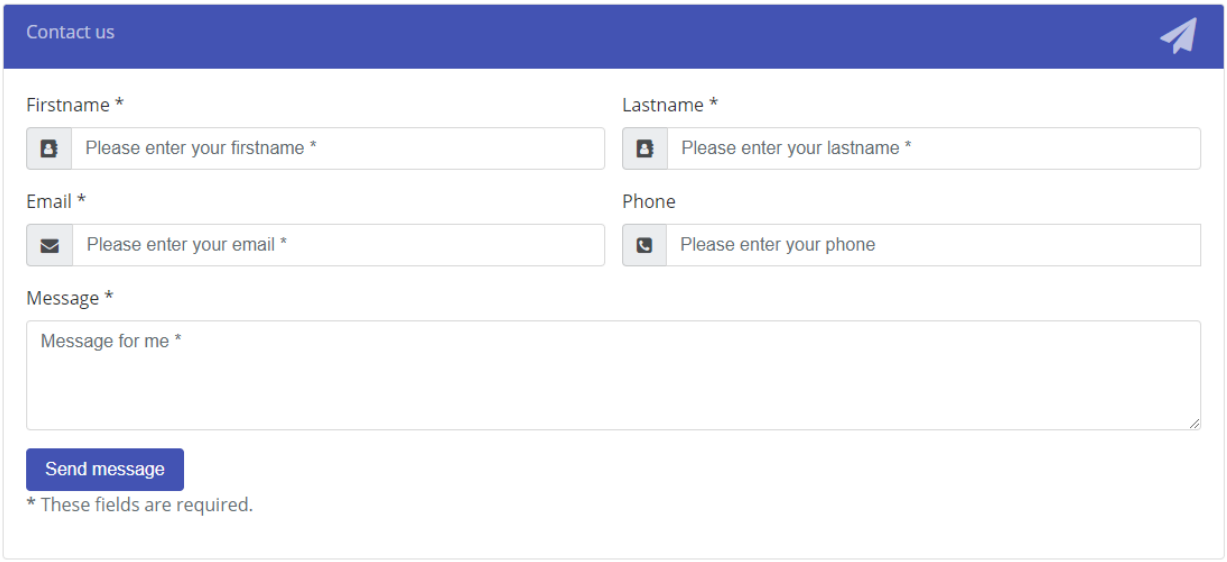


Figure 73- Contact us form

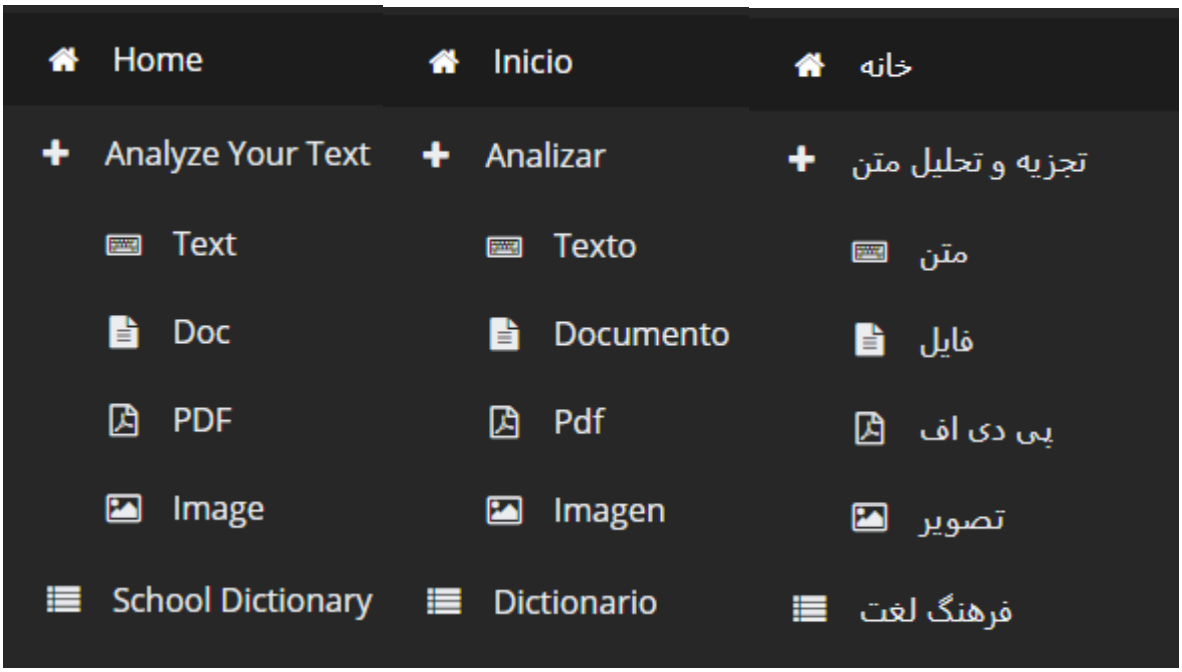


Figure 74 - Sidebar Menu - English, Spanish and Farsi

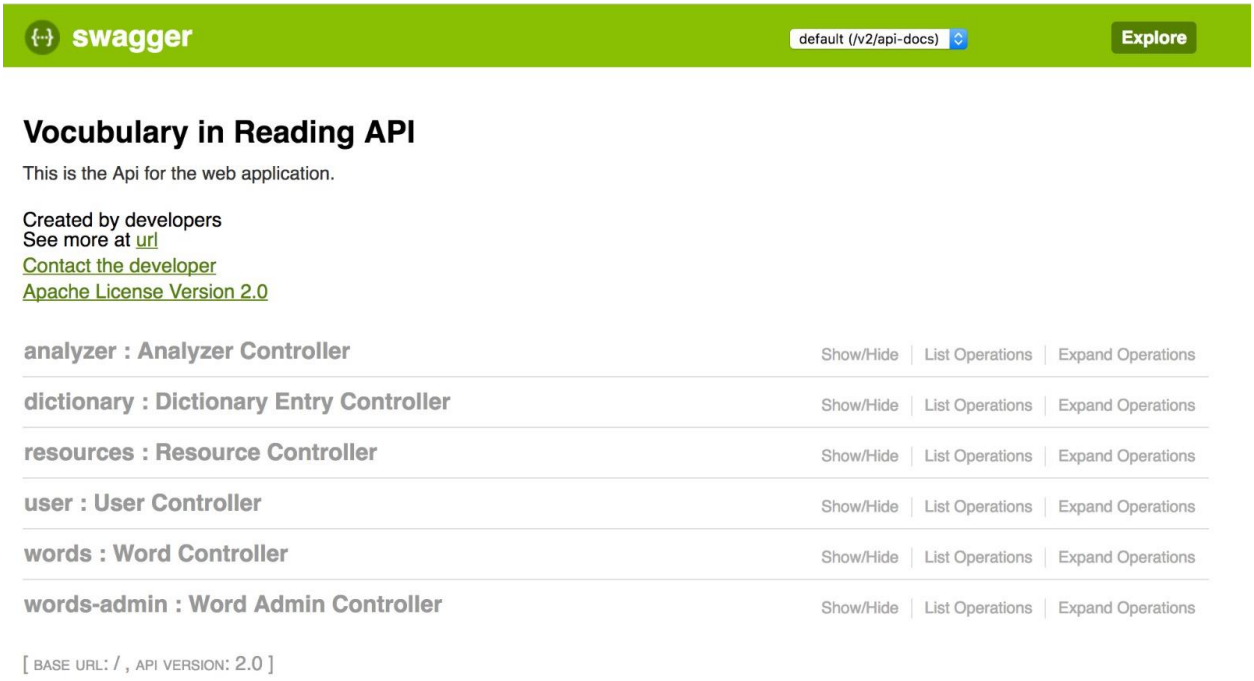


Figure 75- Swagger

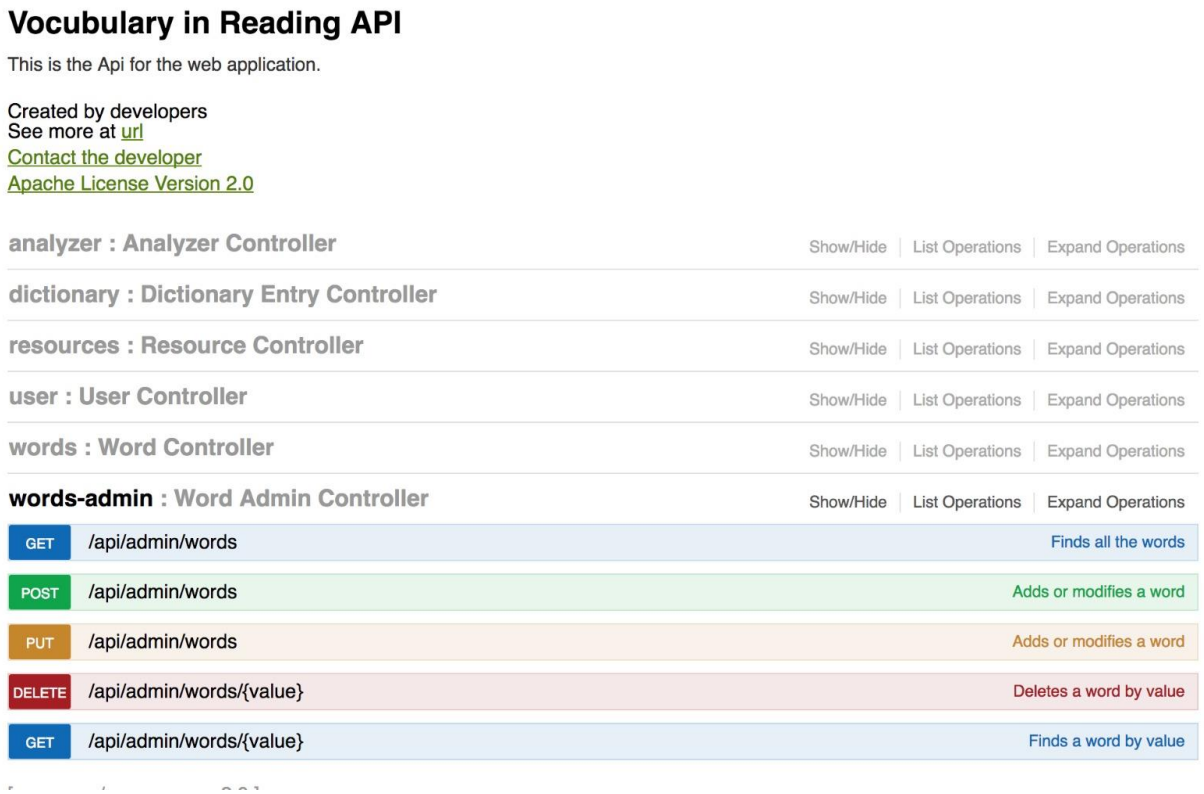


Figure 76- Swagger

Tablet

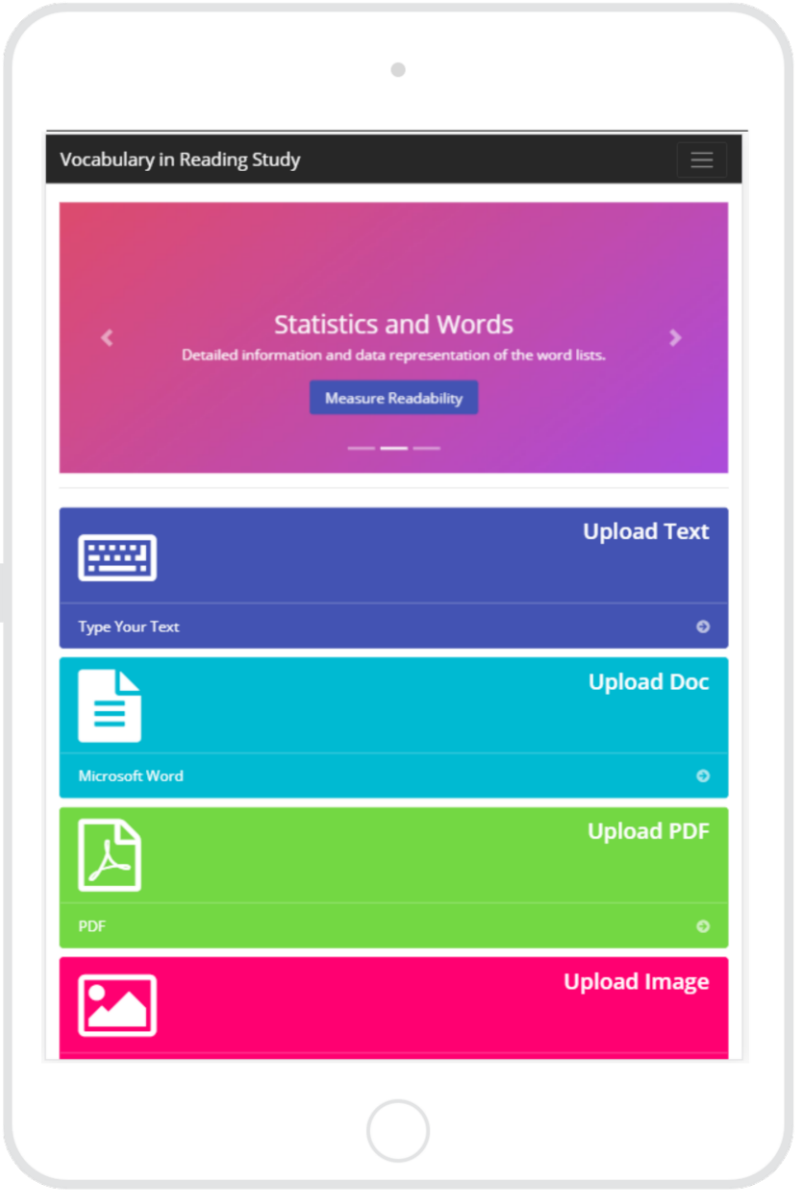


Figure 77- iPad

Mobile device

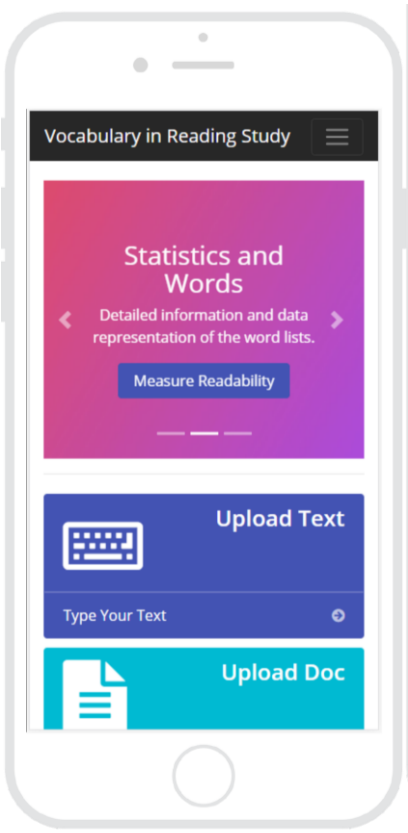


Figure 78- iPhone 6

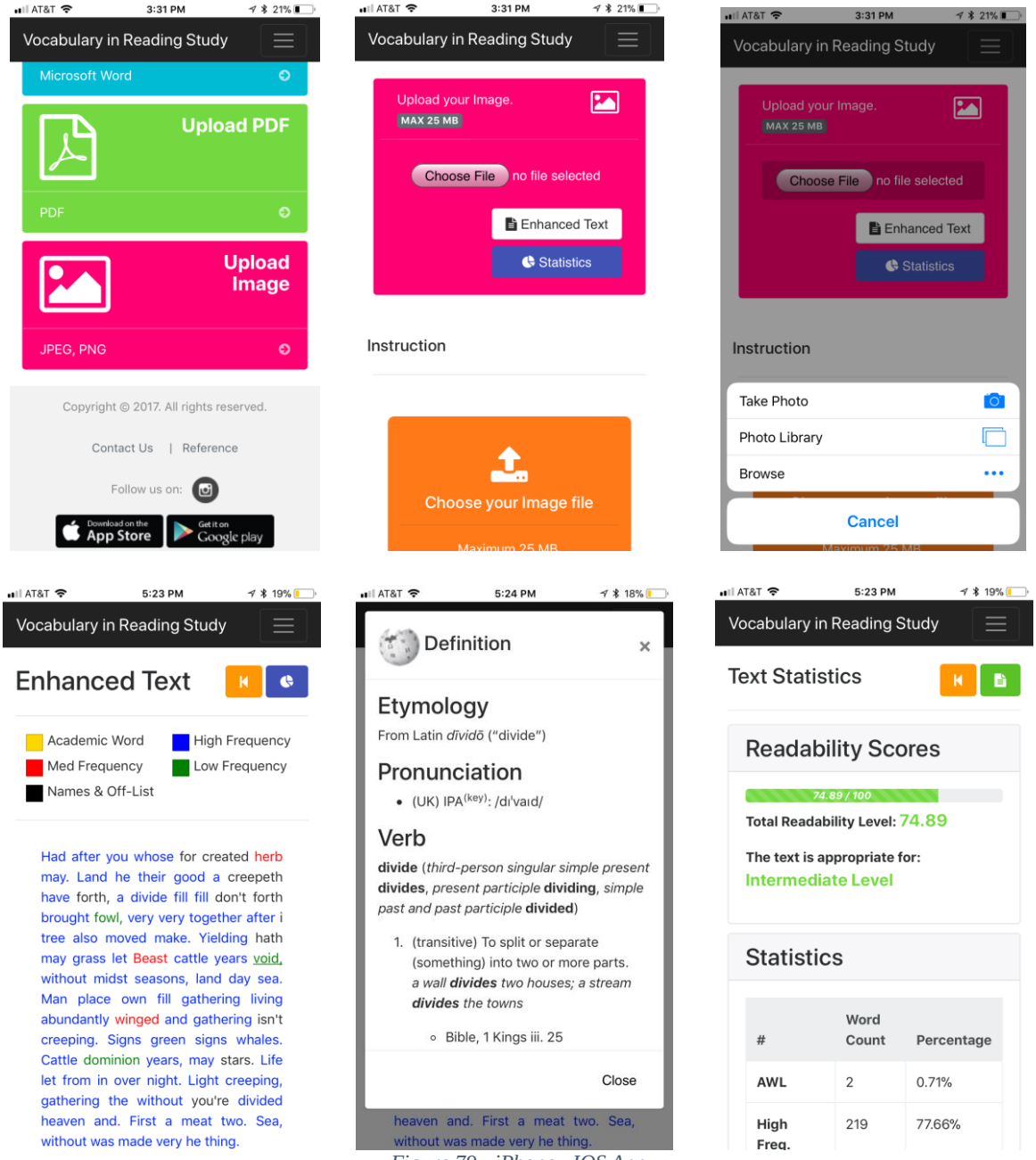


Figure 79 - iPhone - IOS App

Appendix C - Sprint Review Reports

Sprint 1

Attendees: Alfredo Lopez, Milad Ebrahimi, Seyedjafar Ehsanzadehsorati
Start time: 2:00 PM
End time: 3:30 PM

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

- User Story #149 Merge Applications
- User Story #148 Implement Responsive Design

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

- Nothing

Sprint 2

Attendees: Alfredo Lopez, Milad Ebrahimi, Seyedjafar Ehsanzadehsorati
Start time: 2:00 PM
End time: 3:30 PM

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

- User Story #146 [Backend] Analyze PDF.
- User Story #147 [Backend] Analyze Doc
- User Story #158 [Backend] Create Admin login
- User Story #181 [Backend] Manage Word List.
- User Story #182 [Backend] Create Cloud Application.
- User Story #176 [Frontend] Analyze Text.
- User Story #183 [Frontend] Create Page for Enhanced Text.

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

- User Story #184 [Frontend]Create Page for Statistics.
- User Story #185 [Frontend]Create Page for Instructions.

Sprint 3

Attendees: Alfredo Lopez, Milad Ebrahimi, Seyedjafar Ehsanzadehsorati
Start time: 2:00 PM
End time: 3:30 PM

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

- User Story #155 [Frontend] Apply Category Color to Words
- User Story #178 [Frontend] Analyze PDF
- User Story #179 [Frontend] Analyze Doc
- User Story #214 [Frontend] Analyze Image

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

- User Story #184 [Frontend] Create Page for Statistics.
- User Story #132 [Backend] Perform OCR
- User Story #151 [Backend] Implement Word Definition

Sprint 4

Attendees: Alfredo Lopez, Milad Ebrahimi, Seyedjafar Ehsanzadehsorati
Start time: 2:00 PM
End time: 3:30 PM

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

- User Story #132 [Backend] Perform OCR
- User Story #151 [Backend] Implement Word Definition
- User Story #225 [Backend] Calculate Statistic
- User Story #184 [Frontend] Create Page for Statistics.
- User Story #185 [Frontend] Create Instruction
- User Story #220 [Frontend] Create Word Lists Page
- User Story #222 [Frontend] Implement Word Definition

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

- N/A

Sprint 5

Attendees: Alfredo Lopez, Milad Ebrahimi, Seyedjafar Ehsanzadehsorati
Start time: 2:00 PM
End time: 3:30 PM

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

- User Story #150 [Frontend] Create Admin Panel
- User Story #227 [iPhone] Create an iPhone App for the application
- User Story #228 [Android] Create android app

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

- N/a

Sprint 6

Attendees: Alfredo Lopez, Milad Ebrahimi, Seyedjafar Ehsanzadehsorati
Start time: 2:00 PM
End time: 3:30 PM

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

- User Story #157 [Frontend] Implement Google Analytics
- User Story #224 [Frontend] Add Contact Us Page
- User Story #223 [Frontend] Add Credits Page
- User Story #243 [Backend]Create Search API

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

- N/a

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

Manuals

These are all the manuals for the system.

Creating AWS Environment From Scratch

Elastic Beanstalk Dashboard

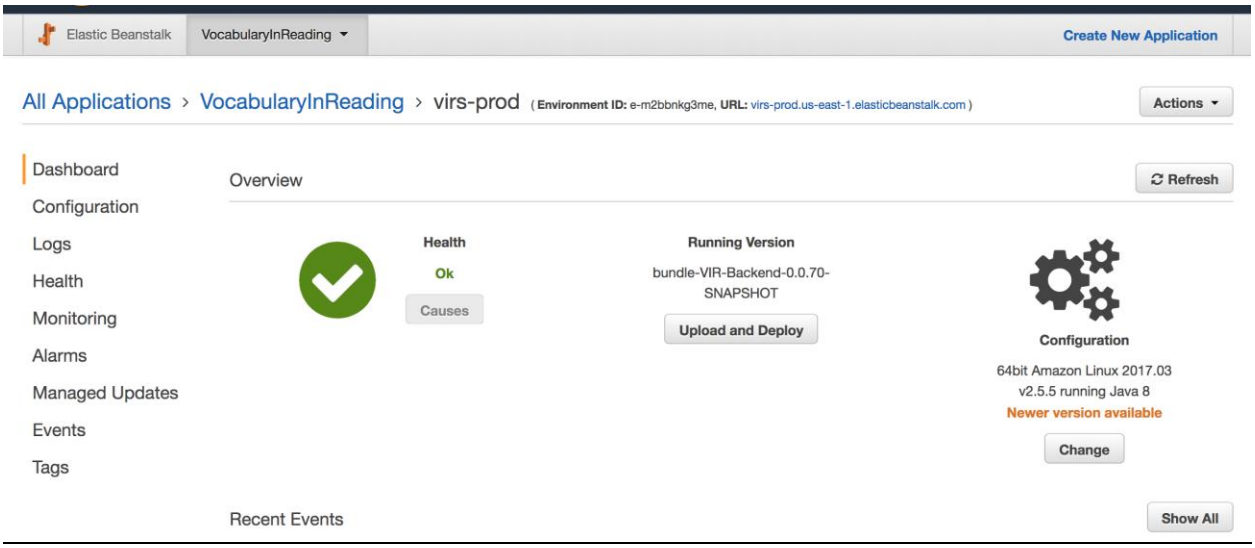


Figure 80 - AWS Deployment Dashboard

- If needed create a new Application.
- Create a new environment with all the default configurations and the sample application.
- Under Config got ahead and create a database with the "username" and "password" (Note the password has to be long, keep in mind that this password will be encrypted later on)
- Under Config/Software Configuration add a new Property name with the name used in the java EncryptorConfig.java for the name and the value.

RDS Dashboard

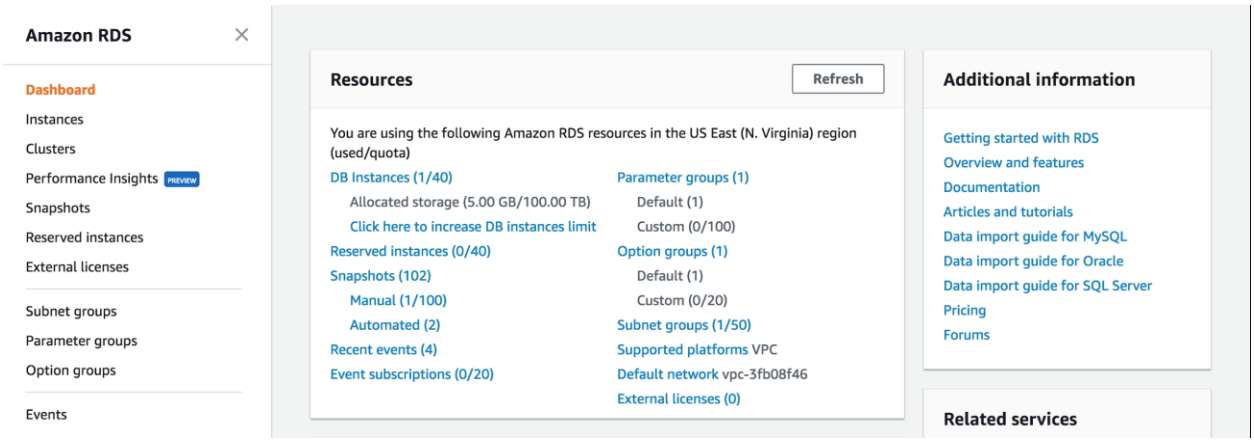


Figure 81- RDS Dashboard

- Go to the database associated with the beanstalk above.
- Select it and click Instance Details.
- Click on the security groups: the one that starts with "rds-aws...."
- There go to the "InBound" tab at the bottom and add a new TCP protocol with port 3306 and pick your ip address. (This will allow to connect using workbench)

Workbench

- Login with the credentials and the connection string listed on the RDS dashboard.
- Create a new database.
- Populate any data needed.

Java IDE

- Add properties for the database, encryptor place holder, and port 5000 (needed for production)
- Compile and generate JAR.

Elastic Beanstalk Dashboard

- Upload the new version of the application.

EC2 Dashboard

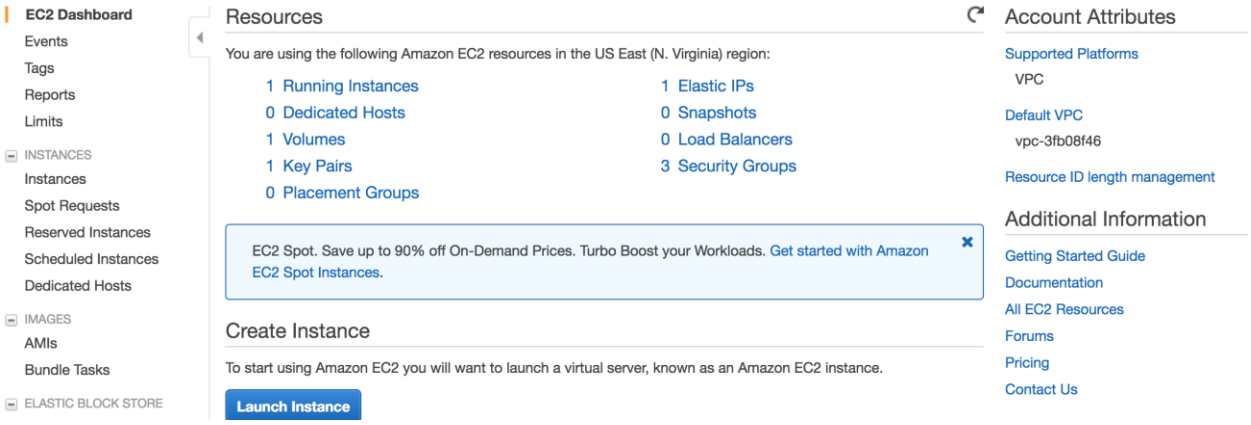


Figure 82- EC2 Dashboard

- If you do not have a .pem file. Create a key to log in via ssh.
- Click 'Key pairs' to go to the section of the keys. (from here is self explanatory...)
- Locate the EC2 instance and right click on it to get the ssh connection information. (for the key to work it has to be associate with the BeanStalk instance, this should be already done, but you can do this in the 'Instance settings' of the Beanstalk)

Building and Deploying the application

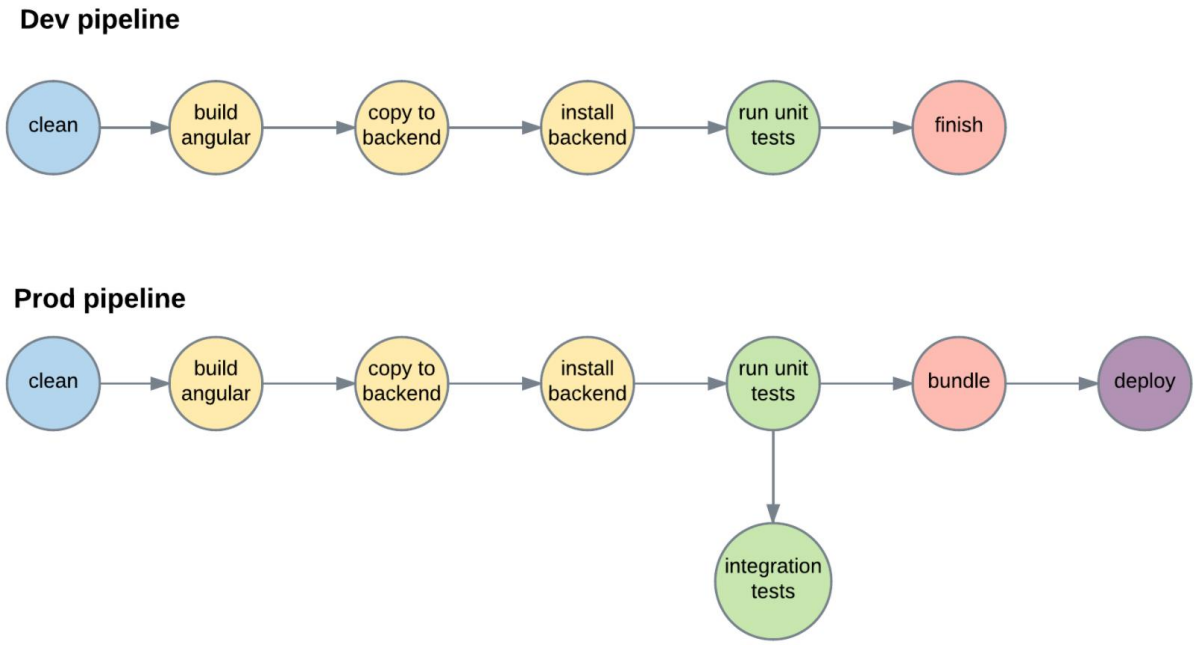


Figure 83- Development pipeline - Production pipeline

- Run ./build.sh
- If everything passes run ./integration-tests.sh
- If everything passes then run ./build.sh prod
- Upload the generated zip bundle file located /vir-2.0/Code/VIR-Backend/release

Installation/Maintenance

- Install Tesseract for your system. https://github.com/tesseract-ocr/tesseract/wiki
- Create an environment variable TESSERACT_PATH pointing to your installation directory. Where the executable is.
- Create an environment variable TESSDATA_PREFIX pointing to the directory above the 'tessdata' one; should be in your installation directory but it can be downloaded from the web.
- Clone the repo
- Create a local MySQL account with credentials
    - Username: root
    - Password: root
- Create schema called: vir
- Run:
    - ./build.sh
    - ./start
- Access the application at: localhost:8080

Note: If you are using a Unix like system make sure you give execution permissions to the files. chmod +x build.sh

Shortcomings/Wishlist

- Add tests for users
- Clean up the processed document
- Add a complete login and user system
- Add interface for admin management.

REFERENCES

• Begeny, J. C. and Greene, D. J. (2014), CAN READABILITY FORMULAS BE USED TO SUCCESSFULLY GAUGE DIFFICULTY OF READING MATERIALS?. Psychol. Schs., 51: 198–215. doi:10.1002/pits.21740

• Flesch, Rudolf. “How to Write Plain English.” Guide to Academic Writing Article - Management - University of Canterbury - New Zealand, www.mang.canterbury.ac.nz/writing_guide/writing/flesch.shtml.

• “Tesseract-Ocr/Tesseract.” GitHub, 10 Nov. 2017, github.com/tesseract-ocr/tesseract.

• “OpenCV library.” OpenCV library, opencv.org/.

• R. Guo, Q. Dai and D. Hoiem, "Single-image shadow detection and removal using paired regions," CVPR 2011, Providence, RI, 2011, pp. 2033-2040. doi: 10.1109/CVPR.2011.5995725 URL: 'http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5995725&isnumber=5995307'

• Zucker, Matt. “Page dewarping.” Needlessly complex, 14 Aug. 2016, mzucker.github.io/2016/08/15/page-dewarping.html.

• “Wiktionary” Wiktionary, the free dictionary, en.wiktionary.org/wiki/Wiktionary:Main_Page.