Bilkent University
Department of Computer Engineering

Senior Design Project

Project short-name: Prexcel

Low-Level Design Report

Burak Yiğit Uslu, Can Kırımca, Can Kırşallıoba, Alper Sarı, Barış Tiftik

Supervisor: Dr. Özcan Öztürk

Jury Members: Erhan Dolak and Tağmaç Topal

Low-Level Design Report

February 28, 2022

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

Low-Level Design Report

Project Short-Name: Prexcel

1. Introduction

Prexcel is a presentation assistance application that helps its users improve their presentation abilities and their presentations. The app accomplishes this by giving feedback to users' presentations and by providing live feedback to the user when presenting. It is geared towards online presentations in particular.

This report expands on the design initially presented in the High-Level Design Report, and discusses object design trade-offs, interface documentation guidelines and engineering standards at Section 1, Packages at section 2 and Class Interfaces at Section 3 respectively.

1.1. Object Design Trade-Offs

Throughout the design and development of Prexcel, a series of trade-offs between the design goals were made in order to optimize the application for its purpose. This section discusses these design trade-offs, case by case.

1.1.1. Efficiency vs Portability

Efficiency is a top priority design goal for Prexcel. Prexcel processes users' speech and facial orientation using machine learning algorithms and libraries, which do not have 100% accuracy. It is crucial that the machine learning models that are being used are efficient enough (efficient in the sense that they have high accuracy) that the app is functional. This sometimes requires us (for the development) and the users to use particular versions of certain software, that are required to run certain libraries or certain versions of those libraries which yield the highest accuracy for our machine learning algorithms. This reduces the portability of the app.

In brief, a design trade-off is made between the efficiency and the portability of Prexcel in favor of efficiency.

1.1.2. Robustness vs. Cost

Robustness is a top priority design goal for Prexcel. This is because the users will use the app to make presentations, which may be sometimes lengthy. If the app crashes halfway during a presentation trial, the user would lose the presentation data and

would have to start the presentation from the beginning in order to get the detailed end analysis for the presentation. It is important that the app does not waste the user's time and effort. Furthermore, if the app crashes during an actual live presentation (where it might be used for its live-feedback purposes) the crash might throw off the user and make the user unnecessarily excited, which might affect his/her presentation in a negative way. Therefore it is of crucial importance that the app is robust and bugs that can crash the app must not occur. In order to prevent these fatal crashes, the system must be tested extensively and exhaustively, which drives up the cost of development.

Therefore, as robustness is crucial to the app because of the aforementioned reasons, a design trade-off is made between robustness and cost, in favor of robustness.

1.1.3. Speed vs Memory Space

One of the two main features of the Prexcel is the live-feedback functionality. And for the live-feedback functionality, response time is of critical importance. If the app cannot make recommendations to the user when necessary in a timely manner, live feedback functionality loses its purpose. The definition of "timeliness" in this respect is discussed in *"High-Level Design Report, Section 1.2.2, Performance Criteria"* in detail, however, in essence, the application must make speedy recommendations to users during the live feedback. In order to achieve the optimal response times, some object design trade-offs are necessary and additional memory space is used in order to enhance the performance in this area.

Therefore, for the functionality of the live-feedback feature, a design trade-off is made between speed and memory space in favor of speed.

1.2. Interface Documentation Guidelines

All class and package names are written in PascalCase. The naming convention for the function names changes depending on the package it is implemented in. Functions and variables implemented in JavaScript have camelCase naming conventions, and those implemented in Python have snake_case conventions. In general, the naming of variables and functions follows the conventions and rules of the languages they were implemented in. Constants are named in all capital letter snake cases in both languages.

Naming conventions within this and every other document regarding Prexcel follow the naming conventions of the code, and the class names and such are presented as described above.

The Python code is documented in the Pydoc format, as can be seen in figure 1 below.

```python
def example_function(argument1, argument2):
    """ Summary of the function

    Args:
        argument1 (type): description
        argument2 (type): description

    Returns:
        (type): Return value
    """
    pass

```

Figure 1. *Example Pydoc function documentation.*

The JavaScript code is documented in the JSDoc format, as can be seen in figure 2 below.

```javascript
/**
 * Multiple lines of JSDoc text are written here,
 * wrapped normally.
 * @param {number} arg A number to do something to.
 */
function exampleFunction(arg) {

}

```

Figure 2. *Example JSDoc function documentation.*

1.3. Engineering Standards

Prexcel processes users' personal auditory and visual data, which is a very critical process that is subject to a wide array of regulations from different governing bodies. All of our implementation and handling of users' personal auditory and visual data, as well as our implementation of user accounts, comply with the GDPR and KVKK [1] [2] regulations and directives.

Since Prexcel handles such sensitive data, it is implemented to be very robust and secure, and therefore complies with all the safety and security recommendations from the official React guide [3].

1.4. Definitions, Acronyms, and Abbreviations

AWS: Amazon Web Services. The database for the system is hosted on AWS. [4]
GDPR: General Data Protection Regulation
GUI: Graphical user interface.
KVKK: Republic of Turkey's Law Regarding The Protection of Personal Information
Model: Model within the context of this project refers to machine learning models, which is fundamentally a program that is trained to recognize patterns. Within the context of this project, machine learning models will be used for converting users speech to text, to recommend words to the user based on the flow of the sentence, and to decide whether the user is looking at the camera (for the face contact grade).
OpenCV: Open source computer vision library [5].
Prexcel: Prexcel is the name of our system, the multifunctional presentation assistant desktop application.
Progress-Tracking: Within the context of Prexcel, progress-tracking refers to the comparison of grades between the selected presentations and consequently an overview of the users presenting ability's improvement.
Speech-To-Text: Conversion of auditory input to text format.
UI: User interface.

2. Packages

2.1 Internal Packages

2.1.1 SpeechToText Package

The classes in this package are responsible for capturing the live audio input from the user and feeding it into the DeepSpeech [6] model to get a transcript of the user presentation. These classes also convert the uploaded audio data into text. Regardless of the type of transcription (live or not), the output of these classes is used to perform a detailed analysis of the presentation. If the presentation is live, the output is also sent to the presentation assistant classes in real-time to provide live feedback to the user. Other than providing a transcript of the speech, these

classes provide the time-related data of the speech such as the start and end time of each word in the transcript.

2.1.2 FacialRecognition Package

This package consists of the class that handles the face detection functionality during the live presentations. It analyses the continuous video feed and when it cannot detect the presenter's face, it sends a flag to the necessary UI components. This package will also feed information to the PresentationAssistant package for the statistics that will be placed in the generated report.

2.1.3 PresentationAnalysis Package

This package contains the classes that perform the detailed analysis of the presentation. This analysis contains both the grammatical analysis of the presentation’s transcript such as the amount of the filler words and also the analysis of the presenter’s speech such as the duration of the gaps in the presentation. These classes work with the output of the classes in the SpeechToText package to produce meaningful reports for the user.

2.1.4 PresentationAssistant Package

This package contains the classes that are responsible for giving live feedback to the user. These classes analyze the textual data (the sequence of words) to give word recommendations, and also the voice level and facial orientation of the presenter to display warnings during the presentation.

2.1.5 UserDataManagement Package

This package’s classes manage the storage and retrieval of the user’s data by connecting the MySQL database hosted on AWS Server and executing queries to manipulate the presentation reports and transcripts.

2.1.6 UserInterface Package

This package contains the classes that constitute the user interface such as menus, recording and uploading screens, and also the reports and graphs of presentations that the user can view.

2.2 External Packages

2.2.1 OpenCV

OpenCV is an open-source library used for computer vision and image processing. In this project, this library is used to perform face detection in order to provide feedback about the presenter's facial orientation. This package is used by the classes in the FacialRecognition package mentioned in section 2.1.

2.2.2 DeepSpeech

DeepSpeech [6] is an open-source speech-to-text engine that can transcribe text using its pre-trained model. In this project, DeepSpeech [6] is used to get the transcript of the presentation and timestamp of the words in order to execute a detailed speech analysis regarding the presentation. This package is imported and used by the SpeechToText package mentioned in section 2.2.

3. Class Interfaces

3.1 PresentationAnalysis Subsystem Service

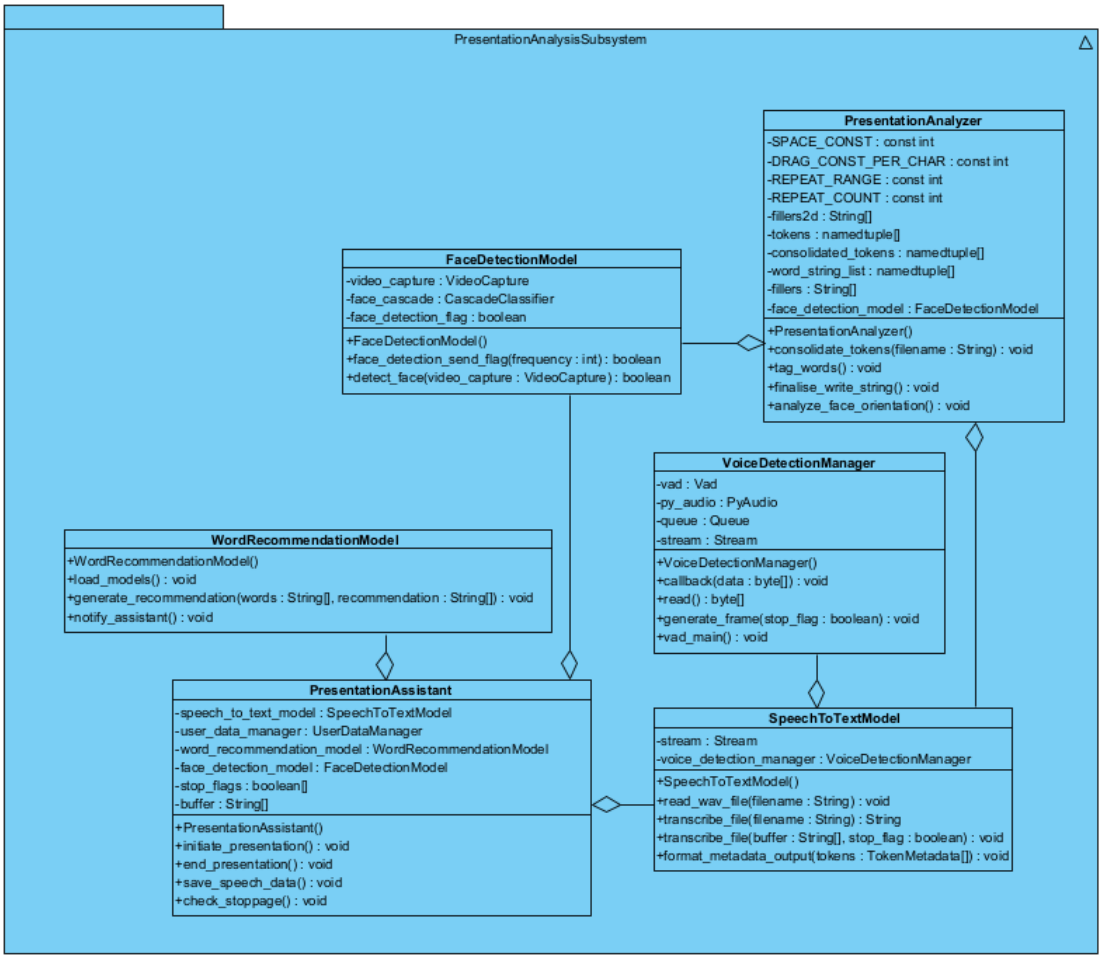

Figure 3. Presentation Analysis Subsystem of Prexcel

3.1.1 VoiceDetectionManager

Attributes
-Vad vad: Voice Activity Detection object created using webrtcvad library.
-PyAudio py_audio: PyAudio object created for capturing microphone input.
-Queue queue: Queue data structure holding the frames as they are being received via the microphone.
-Stream stream: Stream object called on py_audio attribute.
Methods
-void callback(byte[] data): Puts the passed data into the queue object.

+byte[] read(): Returns the next element in the queue.
+void generate_frame(boolean stop_flag): This generator function yields the frames by calling the read() method until the program is terminated (by setting the stop_flag True).
+void vad_main(): Provides the main functionality of the class. By using the previously mentioned attributes and methods, collects the user's voice and returns it frame by frame.

3.1.2 SpeechToTextModel

Attributes
-Stream stream: The stream object used to transcribe the voice input
-Model model: The external speech-to-text model used by DeepSpeech [6] functions.
-VoiceDetectionManager voice_detection_manager: The VoiceDetectionManager object that is used to capture live input from the user's microphone.
Methods
+void read_wav_file(String filename): Reads the .wav file uploaded by the user and makes it ready for speech-to-text conversion.
+String transcribe_file(String filename): Transcribes the .wav file by using the DeepSpeech [6] model.
+void transcribe_live(String[] buffer, boolean stop_flag): Converts live speech to text by using the VoiceDetectionManager object and appends it to the buffer passed into it.
+void format_metadata_output(TokenMetadata[] tokens): After the DeepSpeech [6] model converts the audio input to tokens, this function formats the metadata (the timestamp of each character) and saves it into a temporary text file.

3.1.3 WordRecommendationModel

Methods
+void load_models(): Loads the trained models from the directory.
+void generate_recommendation(String[] words, String[]

recommendations): By analyzing the words in the "words" array, generates the word recommendations and adds them to the "recommendations" array.
+void notify_assistant(): Notifies the PresentationAssistant module that a recommendation is available.

3.1.4 FaceDetectionModel

Attributes
-VideoCapture video_capture: The stream object used to capture the video input
-CascadeClassifier face_cascade: Cascade classifier object from OpenCV
-boolean face_detection_flag: Boolean flag that is True if the face is detected and False otherwise.
Methods
+boolean face_detection_send_flag(int frequency): Sends the face_detection_flag to the PresentationAssistant.
+boolean detect_face(VideoCapture video_capture): Detects the presenter's face and creates the required flags to pass it onto the face_detection_send_flag() function.

3.1.5 PresentationAssistant

Attributes
-SpeechToTextModel speech_to_text_model: SpeechToTextModel object that performs the transcription.
-UserDataManager user_data_manager: UserDataManager object that saves user data after the presentation.
-WordRecommendationModel word_recommendation_model: WordRecommendationModel object that generates the next word prediction based

on the most recently used words of the presenter.
-FaceDetectionModel face_detection_model: FaceDetectionModel object that checks the facial orientation of the user during the presentation.
-boolean[] stop_flags: The array holding the stop flags of each component (speech-to-text model, word recommendation mode, etc.). Since the different models work simultaneously during a presentation, a stop flag is necessary to terminate each thread at the end of a presentation.
-String[] buffer: The array of words that are updated by the speech-to-text manager as the presentation proceeds. These words are used to generate next-word predictions.
Methods
+void initiate_presentation(): Starts the presentation by creating threads for each functionality and running them. Sets the stop flags to False.
+void end_presentation(): Ends the presentation by setting each stop flag to True. This terminates the threads that are running simultaneously.
+void save_speech_data(): Saves the presentation data (transcripts and metadata) after the presentation. Uses user_data_manager object to store the output in the database.
+void check_stoppage(): Checks if the user is silent for a period. If that is the case, sends a signal to the recommendation model to ask for recommendations.

3.1.6 PresentationAnalyzer

Attributes
-const int SPACE_CONST: The minimum amount of time-steps that must be between words for it to be considered extraneous space.

-const int DRAG_CONST_PER_CHAR: The maximum amount of time-steps that a character in a word can have on average before it is considered to be a "dragged" word, as in it has been spoken too slowly.
-const int REPEAT_RANGE: The number of words forward that will be looked up when searching for repetitions of a word.
-const int REPEAT_COUNT: The maximum amount of repetitions that can be within the repeat range of a word before they are tagged as repeats.
-string[] fillers2d: Converted 2d list version of the filler phrases where each word is a separate index in an inner list for every phrase.
-namedtuple[] tokens: List of "token" tuples that denote every letter and their time-step, read in from a file.
-namedtuple[] consolidated_tokens: List of "word" tuples that are created by consolidating tokens that have the text, the timestamps, and the tags necessary for analysis.
-namedtuple[] word_string_list: The list of words and their attributes that have been turned into a string list, this is written to a file for possible later use.
-string[] fillers: The filler phrases read from a file to be used in tagging such phrases in the final result string
Methods
+void consolidate_tokens(String filename): Reads the token data from the specified file, and writes it as "word" tuples with empty tags into "consolidated_tokens".
+void tag_words(): Searches through the list of word tuples and applies tags to them depending on their time-step values or the amount of repetition they present
+void finalise_write_string(): Iterates through the "consolidated_tokens" list and finalizes it into a singular string object where the tagged words or phrases are encased in markup tags, and then writes to an output file.

3.2 UserDataManagement Subsystem Service

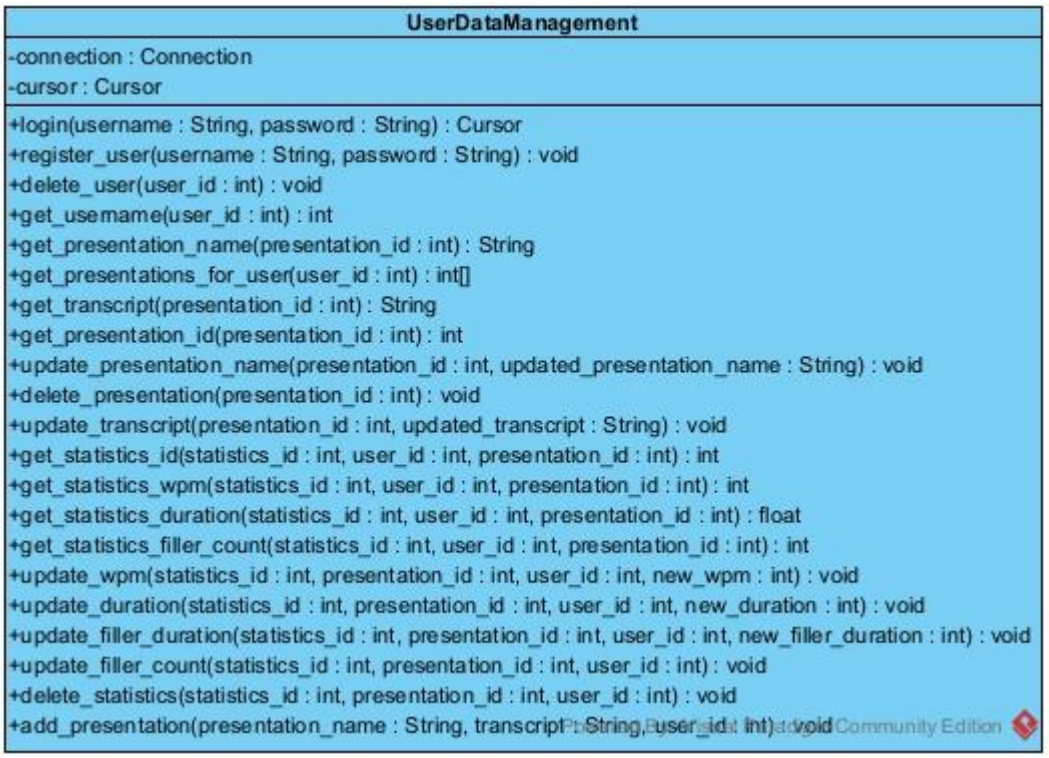

Figure 4. User Data Management Subsystem of Prexcel

Attributes
-Connection connection: Connection with the database
-Cursor cursor: The commands that are affecting the database go through the cursor
Methods
+Cursor login(String username, String password): Method for logging in a username and password
+void register_user(String username, String password, String mail_adress): Method for registering a user with username, password, mail_adress
+void delete_user(int user_id): Method for deleting a user with its user id
+int get_username(int user_id): Method for getting a username with user id
+String get_presentation_name(int presentation_id): Method for getting presentation name with presentation id
+int [] get_presentations_for_user(int user_id): Method for getting all the

presentation a user has with its user id
+String get_transcript(int presentation_id): Method for getting transcript with presentation_id
+int get_presentation_id(int presentation_id): Method for getting presentation id
+void delete_presentation(int presentation_id): Method for deleting presentation with presentation id
+void update_presentation_name(int presentation_id, String updated_presentation_name): Method for updating presentation name with presentation id
+void update_transcript(int presentation_id, String updated_transcript): Method for updating transcript with presentation id
+int get_statistics_id(int statistics_id, int user_id, int presentation_id): Method for getting the statistics with user id and presentation id
+int get_statistics_wpm(int statistics_id, int user_id, int presentation_id): Method for getting wpm statistics with user id and presentation id
+float get_statistics_duration(int statistics_id, int user_id, int presentation_id): Method for getting duration statistics with user id and presentation id
+float get_statistics_filler_duration(int statistics_id, int user_id, int presentation_id): Method for getting filler duration statistics with user id and presentation id
+int get_statistics_filler_count(int statistics_id, int user_id, int presentation_id): Method for getting filler count statistics with user id and presentation id
+void update_wpm(int statistics_id, int presentation_id, int user_id, int new_wpm): Method for updating wpm with statistics id, user id, presentation id
+void update_duration(int statistics_id, int presentation_id, int user_id, int new_duration): Method for updating duration with statistics id, user id, presentation id

+void update_filler_duration(int statistics_id, int presentation_id, int user_id, int new_filler_duration): Method for updating filler duration with statistics id, user id, presentation id
+void update_filler_count(int statistics_id, int presentation_id, int user_id, int new_filler_count): Method for updating filler count with statistics id, user id, presentation id
+void delete_statistic(int statistics_id, int presentation_id, int user_id): Method for deleting statistics with statistics id, user id, presentation id
+void add_presentation(String presentation_name, String transcript, int user_id): Method for adding presentation with presentation name, transcript, user id

3.3 UserInterface Subsystem Service

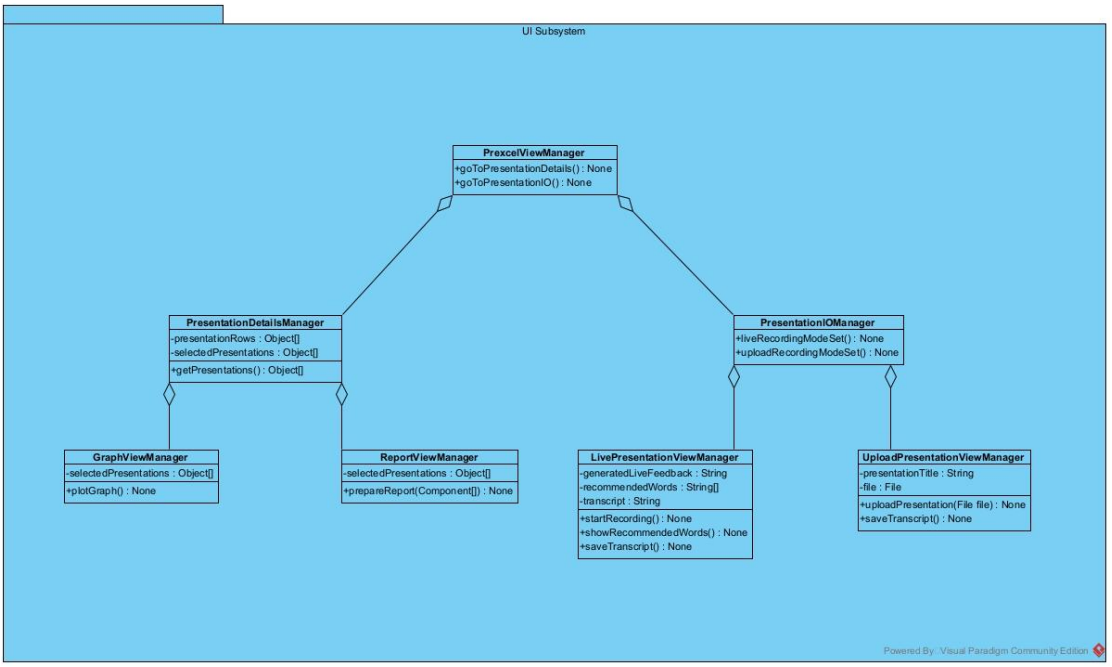

Figure 5. User Interface Subsystem of Prexcel

3.3.1 PrexcelViewManager

Methods
+void goToPresentationDetails(): Goes to Presentation Details Screen
+void goToPresentationIO(): Goes to Presentation IO Screen

3.3.2 PresentationDetailsManager

Attributes
-Object[] presentationRows: Rows of presentation objects
-Object [] selectedPresentations: Rows of selected presentation objects
Methods
+Object [] getPresentations(): Retrieves the selected portion of the presentations

3.3.3 GraphViewManager

Attributes
-Object [] selectedPresentations: List of selected presentation objects
Methods
+void plotGraph(): Plots the graph based on the selected presentations

3.3.4 ReportViewManager

Attributes
-Object [] selectedPresentations: List of selected presentation objects
Methods
+void prepareReport(Component []): Takes the components of the report and merges them

3.3.5 PresentationIOManager

Methods
+void liveRecordingModeSet(): Sets the application mode to live recording mode
+void uploadRecordingModeSet(): Sets the application mode to upload recording mode

3.3.6 LivePresentationViewManager

Attributes
-String generatedLiveFeedback: Live feedback that is generated by the Presentation Assistant
-String [] recommendedWords: Live feedback that is generated by the Word Recommendation algorithm
-String transcript: The transcript of the presentation
Methods
+void startRecording(): Starts the recording of the live presentation
+void showRecommendedWords(): Shows the list of recommended words
+void saveTranscript(): Saves the currently shown transcript to the database

3.3.7 UploadPresentationViewManager

Attributes
-String presentationTitle: Title of the presentation
-File file: File to be uploaded
Methods
+void uploadPresentation(): Uploads the presentation to be analyzed

3.4 DataProcessing Subsystem Service

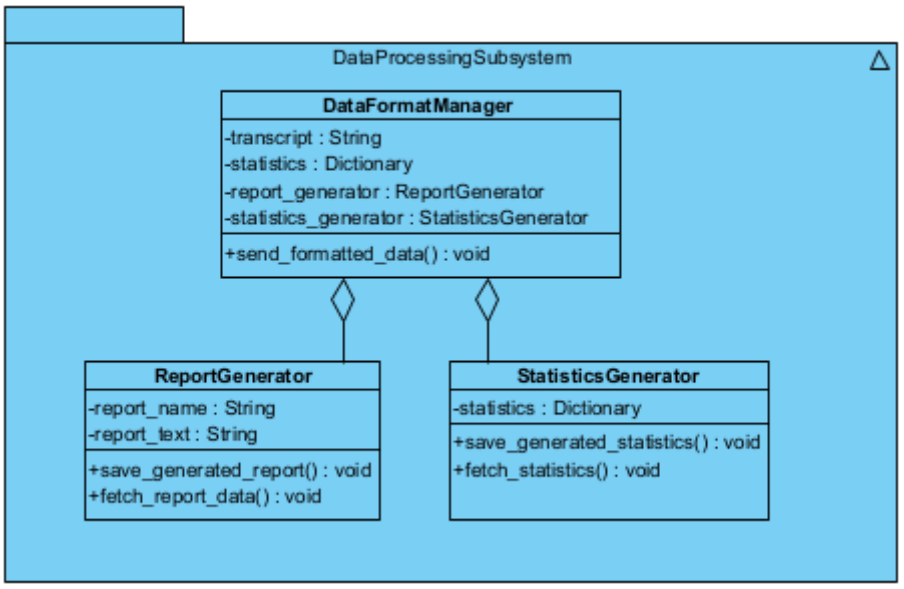

Figure 6. Data Processing Subsystem of Prexcel

3.4.1 DataFormatManager

Attributes
-String transcript: Transcript that is generated from speech-to-text algorithm.
-Dictionary statistics: Statistics of the presentations.
-ReportGenerator report_generator: Merges all the components of the report.
-StatisticsGenerator statistics_generator: Merges the statistics of the presentation.
Methods
+void send_formatted_data(): Send the formatted data to be converted to report and statistics by using the ReportGenerator and StatisticsGenerator object.

3.4.2 ReportGenerator

Attributes
-String report_name: Name of the report.
-String report_text: The transcript of the report.
Methods
+void save_generated_report(): Saves the report generated by the generator.
+void fetch_report_data(): Retrieves the report data from the database to be sent to the frontend processes.

3.4.3 StatisticsGenerator

Attributes
-Dictionary statistics: Statistics of the presentations.
Methods
+void save_generated_statistics(): Saves the generated statistics.
+void fetch_statistics(): Retrieves the statistics from the database to be sent to the frontend processes.

Glossary

DeepSpeech: An open-source machine learning algorithm developed by Mozilla that is used for converting speech to text.

Electron: A Javascript library that is used for creating web-based apps on the desktop platform.

MySQL: An open-source relational database management system.

OpenCV: An open-source computer vision library.

React: A Javascript framework developed by Facebook. It is used for creating web apps for browsers or mobile platforms (using React native). Within the context of the project, it will be used alongside Electron to create a desktop app.

References

[1] “What is GDPR, the EU's new Data Protection Law?,” *GDPR.eu*, 13-Feb-2019. [Online]. Available: https://gdpr.eu/what-is-gdpr/. [Accessed: 27-Feb-2022].

[2] “Kişisel Verileri Koruma Kurumu: KVKK: Personal Data Protection Law,” *KVKK*. [Online]. Available: https://www.kvkk.gov.tr/Icerik/6649/Personal-Data-Protection-Law. [Accessed: 27-Feb-2022].

[3] “Getting started,” *React*. [Online]. Available: https://reactjs.org/docs/getting-started.html. [Accessed: 27-Feb-2022].

[4] AWS. https://aws.amazon.com/, “Amazon Web Services”, [Accessed: Dec. 15, 2021].

[5] OpenCV, https://opencv.org, “OpenCV”, [Accessed: Dec. 9, 2021].

[6] DeepSpeech, https://deepspeech.readthedocs.io/en/r0.9/#, [Accessed: Dec. 9, 2021].