Bilkent University
Department of Computer Engineering

Senior Design Project

Project short-name: Prexcel

Final Report

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

Supervisor: Dr. Özcan Öztürk

Jury Members: Erhan Dolak and Tağmaç Topal

Final Report

May 06, 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

1. Introduction

In the current world of remote communication and meetings, it is paramount that the online workspace includes the remote presentation to go along with remote work. It is important then that during an online presentation, the presenter makes their point smoothly and the viewers absorb all of the necessary. Our application, named “Prexcel” aims to improve the efficacy of online presentations for its users. It aims to improve the general ability of presentation for the user, as well as helping them with analyzing the errors in their presentations as they practice or perform them.

Prexcel, as an online presentation assistant will record the presentations of the user, and analyze the transcript of said presentation as well as log the facial movements of the user. The application then detects the errors in the spoken parts of the presentation, to be later displayed as a transcript with the relevant portions being highlighted. This, along with the facial recognition of the presenter is then ranked and saved as a report on their user profile that they can refer back to and go over themselves afterwards.

In this report, we are going to discuss our final system considering its full life-cycle from requirements to implementation to testing. Then we are going to talk about other elements such as risks and alternatives, our project maintenance plan, how we managed our team-work environment, our ethical and professional responsibilities as well as the new knowledge and strategies we learned and followed. We will conclude with all the new knowledge we acquired while working on the project, as well as the possible future work that might commence on the project.

2. Requirements Details

Final Functional Requirements

User-Side Requirements

- The user should be able to upload the audio and video recordings of their presentations or speeches.
- The user should be able to start live recordings and record their presentations while giving them.
- The user should be able to receive live feedback based on their live presentations.
    - The user should see recommended words that are relevant to the presentation's context during the presentation.
    - The user should be able to see feedback about their pace of speech.
    - The user should be able to receive warning if they fail to look at the direction of presenters for a particular duration.
    - The user should be able to receive warning if their voice volume is above or below the normal threshold.
- After uploading or recording a presentation, the user should be able to receive system-generated reports containing the analysis of their presentations based on various metrics.
    - The report should contain the transcript of the speech that the user gave.
    - The report should contain the evaluation of the speech based on various metrics such as words per minute, repetitive words and their frequencies etc.
- The user should be able to view their reports on the system.
- The user should be able to create accounts using their username, email addresses and a password.
- The user should be able to sign in to its account with their login information.

- The user should be able to delete their account.

System-Side Requirements

- The system should be able to transcribe the user’s speech.
- The system should be able to conduct an analysis of the transcript based on the determined metrics.
- The system should be able to generate word recommendations relevant to the presentation’s context and flow of the sentences.
- The system should be able to analyze the visual input based on the user’s face orientation.
- The system should keep track of the registered users.
- The system should be able to retrieve and modify the user's presentation data as requested by the user.

Final Non-functional Requirements

Usability

- The user interface should be simple to use and user-friendly.
- The live feedback given by the system should not distract the presenter during the live presentation.
- The application should be usable by anyone who has an English level above A1.

Reliability

- The application should continue to function even if the voice input cannot be transcribed or some issue prevents the use of instant feedback from working within the appropriate time frame.

Performance

- The live feedback model should be able to make recommendations in at most 3 seconds after the presenter stops speaking.
- Processing of the audio files (when the user chooses to upload an audio file) should not take more than 5 seconds for a presentation of 10 minutes.
- Processing of the video files (when the user chooses to upload a video file) should not take more than 10 seconds for a presentation of 10 minutes.

Supportability

- Prexcel must support Windows, Linux, and Mac desktop operating systems.
- Any recording hardware (Microphone and camera etc.) that the users utilize should be supported by the application, assuming that the hardware components are compatible with the system on which the application is running on.

Security

- Users' processed data should be deleted right away after its usage.
- Users' password must be encrypted when it is stored in the database.

Scalability

- The application should work even when the user count exceeds projected counts, and the user experience should not suffer due to this fact.

Final Pseudo Requirements

- Prexcel will be a desktop application.
- GitHub will be used for version control, as well as tracking changes made across development.
- DeepSpeech [1] will be used to convert speech to text.
- OpenCV [2] will be used to implement the Facial Recognition Model.

- JavaScript [3] programming language will be used to program the user interface.
- Electron [4] and React [5] framework will be used for developing the applications UI.
- Python [6] programming language will be used to implement processes and machine learning models.
- There will be a database used for user accounts, utilising MySQL [7].

3. Final Architecture and Design Details

3.1 Three-Layered Architecture

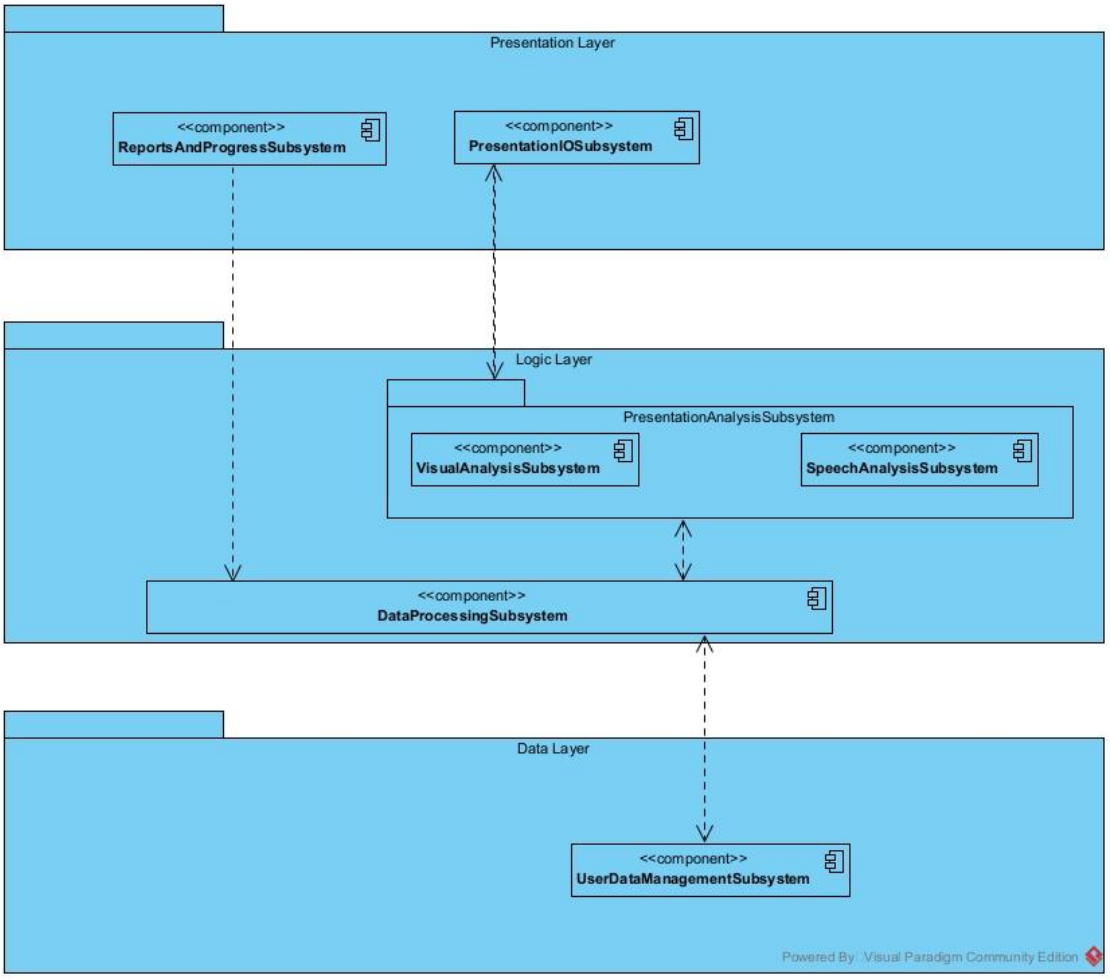

Figure 1. Three-Layered Architecture of Prexcel

As mentioned in the High-Level Design Report, Prexcel uses a three-tier architecture. This architecture enables the application to separate its different functionalities as much as possible

and reduce coupling between them. We have the three layers as follows:

- Presentation Layer components interact with the user and work with the logic layer. In other words, this layer contains front-end components that interact with backend processes in the logic layer. This layer includes the components in presentation recording and uploading screens, and also the report and transcript viewing screens.
- Logic Layer is the layer containing the modules that perform all backend operations. These modules process the data coming from the presentation layer and send them to the data layer. Some of those components include the speech and visual analysis classes, speech-to-text models, and next-word generation model.
- Data Layer contains the classes that communicate with the MySQL database on the AWS server. These classes are responsible for sending the processed data to the database for save operations, and also fetching the same data in order to show it to the user when requested.

3.2 Hardware-Software Mapping

Figure 2. Hardware-Software Mapping of Prexcel

As mentioned in the high-level design report, Prexcel will run on the user's computer. All frontend and backend processes will run on the user's PC. Prexcel uses the microphone and webcam of the user's computer. The only external server is the AWS database server that stores the presentation data. The connection to this external server is performed by the data layer classes.

3.3 Persistent Data Management

As mentioned earlier in the report, Prexcel stores its users' presentation and account details on an external MySQL database hosted on AWS Server. In this database, we have tables containing user account information, and also their presentation information such as the gaps in the presentation, word count and text transcript. More specifically, we have the following tables in the database.

- User: Stores account data. Contains the following fields:
    - username: Username of the user
    - password: Password of the user
    - mail_address: Mail address of the user
    - user_id: Unique ID number of the user
    - presentation_count: Number of presentations recorded for the user. Incremented when a new presentation is added for the user.

- Presentation: Stores all information regarding the user's presentations
    - presentation_id: Unique ID of the presentation
    - presentation_name: The name of the presentation which is specified by the user at the beginning of the presentation.

- transcript: The transcript of the presentation that is generated by speech-to-text and speech analysis models. Also contains the tags to denote which word falls into which category (Filler, repeated, dragged, etc.)
- user_id: The ID of the user that relates the user to the specific presentation. When a user wishes to view their presentations, the presentations are fetched using this key.
- wpm (word per minute): Number of words spoken in the presentation in a minute. Found by dividing the number of words to the duration of the presentation (in minutes).
- duration: Duration of the presentation in seconds.
- filler_ratio: The ratio of filler words in the presentation. Calculated by dividing the number of filler words by the number of total words.
- word_count: The number of words in the presentation.
- gap_ratio: The ratio of gaps during the presentation. Calculated by dividing the duration of gaps by the duration of the presentation.
- fd_score: Face detection score of the presentation. Calculated with the face detection data coming from the face detection class that uses the OpenCV model.
- grade: The calculated grade for the presentation.
- dragged_ratio: The ratio of the dragged words.
- repeated_ratio: The ratio of the repeated words.
- p_date: The date of which the presentation was recorded.

3.4.Object Design Trade-offs

3.4.1 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 that, in some cases, need to work simultaneously to give instant feedback. It is of great importance that these models are efficient for the app to be functional in real-time. 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.

3.4.2 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.

3.4.3 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.

4. Development/Implementation Details

4.1 Implementation Overview

After careful research and comparison, we decided to implement the frontend of Prexcel with React and Electron. Electron is a framework developed by GitHub [4] that enables us to build cross-platform apps with JavaScript, HTML, and CSS. For the backend processes and connections with the database, we used Python and Flask. React app is hosted using GitHub pages, and Electron app runs on the user's desktop, loads the React app from that URL and displays it to the user, as a desktop application. The Electron-React app uses the data coming from the backend processes by sending requests to the Flask app. The Flask

app contains specific endpoints, each of which is called by the Electron-React app to perform a different functionality (For example, performing a login or starting a presentation). For data storage, we used a MySQL database hosted on Amazon AWS Server. In order to manipulate this database, we implemented a database management class in Python by using the PyMySQL package.

For the word generation feature, we used Gensim's Word2Vec library and Keras's LSTM model, which will be explained in more detail in the next sections. For face recognition, we used OpenCV [2] which is an open-source computer vision library.

4.2 .Log-in and Sign-up

When a user attempts to log into the system, the entered username and password are compared with the records on the MySQL database in the AWS system by executing the function in the data management class. If the username and password match the existing records on the database, the user successfully logs in. Otherwise, the user receives an error message. When a user decides to sign-up for the system, the entered information is pushed to the database server and a new account is created with the given information. After that, the users can log into their newly created account.

4.3. Creating a presentation

4.3.1 Recording a live presentation

Prexcel has the functionality of recording a presentation and giving instant feedback to the user. During the recording, the user can receive feedback on their voice level, their facial orientation (whether their face can be detected or not), and can also see their presentation's transcript. Another feature of Prexcel is that it generates possible next words based on the user's speech in order to help them when they get stuck during the presentation. These

feedback mechanisms all work at the same time and they have their respective functions running on different threads to provide instant feedback. These features will be explained further in their respective subsections. After the user ends the presentation, a detailed report is generated and pushed to the database to be viewed later.

4.3.2 Uploading a pre-recorded presentation

In addition to the live recording functionality, Prexcel can also process pre-recorded presentations. The user needs to choose a file (.wav or .mp4) from their computer and the application can process that file and generate a detailed report. However, if the user uploads a '.wav' file, feedback on the facial orientation is not given in the report.

4.4 Facial recognition

During a live presentation, Prexcel can process the data coming from the camera feed and detect the presenter's face if they are looking towards the camera. We provide this functionality by using the functions from OpenCV. These functions analyze the camera input and notify the PresentationAssistant class that a face is detected. Furthermore, Prexcel can perform the same analysis for uploaded presentation videos (In 'mp4' format), again by using OpenCV's functions. In the live presentation screen, we give an immediate feedback on whether the presenter's face is detected, and we calculate a face detection score for both cases. After the analysis of presentation ends, this face detection score is calculated and pushed to the database to be displayed in the report.

4.4 Voice level feedback

In order to provide voice level feedback, pyaudio package is utilized. When the user is presenting a live presentation; the stream object passes the data to the RMS (Root Mean Square) function which in turn is transferred to the correct decibel amount by taking its log

and multiplying it with a constant value. If the decibel level of the presentation is lower or higher than a certain threshold, then the user is given a warning.

4.5 Converting Speech to Text

To convert audio input to text, we used DeepSpeech speech-to-text model developed by Mozilla. In our implementation, we can convert audio files, as well as live microphone input to text transcript. In order to process the live microphone feed, we needed to implement another class called VoiceDetectionManager that can detect the voice activity and notify the model to produce the transcript of the live speech.

4.6 Word generation

For word generation, we tested different machine learning models and decided to use Word2Vec and LSTM (Long Short-Term Memory) in order to generate possible next words for the presenter. Word2Vec is a word embedding method used to convert the meanings of the words into numerical vectors. This enables us to find new words by adding and subtracting these vectors. We used Gensim Word2Vec in our implementation. We chose Keras' LSTM model for its effectiveness on sequential inputs. We trained the model on the transcripts of TED Talks since they are accurate representations of how to give effective presentations. During the presentation, the program detects the last four meaningful words in the presenter's speech. Then, it converts them to vectors by using Word2Vec. After creating the vectors, these vectors are fed into the LSTM to produce a new word vector based on the input vectors. After that, Word2Vec detects the closest words to the output vector and lists the possible words that the vector can represent. After this process, the generated words are displayed on the screen.

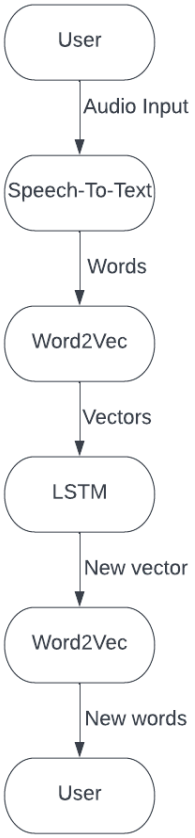

Figure 3. A diagram showing the next word generation process

4.7 Speech Analysis

To make a detailed analysis of the user's speech, we used the tokens returned by the DeepSpeech model with their timesteps. Therefore, we can determine whether there are gaps in the presentation, whether a word is filler or repeated too much, and also whether a word is dragged. More specifically, we have the following categories of words:

- Filler: The meaningless words that are used to fill the gaps in the speech. We have a file that contains all possible filler words and the speech analyzer checks which words in the transcript match one of these fillers. If a word matches an entry in the fillers file, it is tagged as filler.
- Repeated: If a word is repeated too much (above a certain threshold) in the text, it is tagged as filler.

- Dragged: If a word is spoken slowly, it is tagged as dragged. The analyzer decides this by looking at the timesteps between the first and the last characters of the word.

The speech analyzer class determines which word falls into which category and wraps these words with tags (For example, if a word is a filler, it will become ‘{filler}<word>{filler/}’) that will be visible in the transcript of the presentation. In addition to marking those words, we also produce statistics such as words per minute (wpm), the ratio of the gaps and fillers in the presentation, and the total number of words spoken. After the generation of the report containing the speech analysis, this information is pushed to the database.

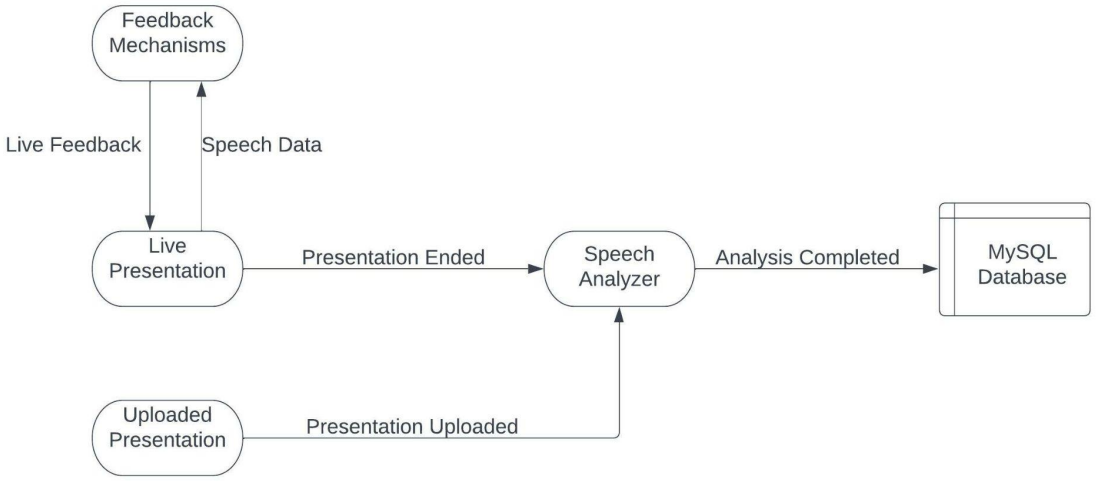

Figure 4. The general flow of creating a presentation

4.8 Viewing Past Presentations and Progress

A user’s presentations are stored in the MySQL database and are linked with the user’s ID. Therefore, the users can access their past presentations and view their reports. When they enter “My Presentations” screen, the user’s presentations are fetched from the database and shown to the user as a list. The user can also view the report of a specific presentation by choosing one from the list. After that, all information regarding the particular presentation can be viewed by the user such as the statistics and transcript. It is also possible to observe the

progress of the presentation scores on an auto-generated graph. This graph is generated by using the presentation scores of the user.

5. Testing Details

To test Prexcel, the following testing techniques are used.

5.1 Unit Testing

To test the backend functionality, unit testing for Python is utilized by comparing the expected results that the functions should output and the results that they actually do output. However, due to the nature of the application (the machine learning models give result on the given inputs, which change when, for instance, the face is shown in the video or not.) writing and executing these backend unit tests was a real challenge.

```
def test_face_detection_from_file(self):
    fd = FaceDetection()

    face_detection_flags = []
    test_file_name = "test.mp4"

    test_flag = False

    if fd.detect_face(test_file_name, face_detection_flags) > 0.0:
        test_flag = True


    self.assertTrue(test_flag)
```

Figure 5. Sample Python Unit Test file.

Above is an example Python Unit Test code, this piece of code in particular is for a successful face detection from file test. The “test.mp4” is a video in which the person in the

video is facing the camera directly throughout the video, which triggers the face detection field's test_flag.

5.2 Functional Testing

In order to fulfill the requirements mentioned in the Functional Requirements section, the listed requirements are cross-checked with the implemented functionalities. The purpose of this testing method is to ensure that all the promised functionality is present in the application. In order to test the full functionality of different user interface components, we have utilized React Testing Library, and wrote test cases accordingly. This way, even though the design of the user interface kept on changing; the test cases ensured that the full functionality remained within.

```
test('test for login page -- failed login', () => {
  render(<Login />);

  const username_field = screen.getByTestId("login_username_field");
  userEvent.type(username_field, "");

  const password_field = screen.getByTestId("login_password_field");
  userEvent.type(password_field, "");

  const login_button = screen.getByTestId("login_button_id");
  userEvent.click(login_button);

  const error_message = screen.getByTestId("login_error_message_id");
  expect(error_message.textContent).toStrictEqual("Incorrect password or username!");

});
```

Figure 6. Sample React Testing Library test case.

Above is an example React Testing Library code, this piece of code in particular test for a failed login operation. The test id's are added to the text fields, buttons, and the error messages. These test id's are used by calling the **`getByTestId()`**, which in turn returns the component which's id was passed. Within the code there are also statements which use

`userEvent, userEvent` is used to interact with the elements of the application (such as clicking or typing). Lastly, the way test cases are checked is by the `expect()` function of React Testing Library, when the error message gets displayed, the expect statement compares and returns the status of the test case (success or fail).

5.3 Integration Testing

Since there are many subsystems present in Prexcel, integration tests are done in order to ensure that there are no complications when said subsystems are interacting with each other. Since this testing technique applies the black box principle, there is no source code provided.

6. Maintenance Plan and Details

Two strategies have been decided on for the maintenance plans of Prexcel. Said plans are database and application maintenance. These plans also include additions to the project, such as implementing a DevOps pipeline.

6.1 Database Maintenance

Prexcel utilizes the AWS Cloud Database system to oversee the database and the tables that reside within it. Since the application revolves around having presentations and their analysis stored for given users, maintenance of the database is an absolute must. To maintain it, the developer team will continue to optimize the queries and back up the database.

6.2 Application Maintenance

So far, for the maintenance of the application between major releases, the maintenance of the bugs and issues that come up was handled using the Github Issues system. Considering it has held through in an efficient and effective way so far and also considering that the development team is familiar with Github in general (version control and Issues), in the future, the same system shall be used. When the application becomes public, if Github Issues becomes inadequate for the needs of the developers, then alternatives such as Jira would be considered. DevOps will also be implemented in order to ensure the continuous integration and continuous delivery. This will streamline the software updates and new creation.

7. Other Project Elements

7.1. Consideration of Various Factors in Engineering Design

In light of the requirements of the system, design goals considered for Prexcel are based on a variety of criteria including end-user criteria, performance criteria, and design criteria.

7.1.1 Consideration of End-User Factor and Related Design Goals

Design goals that stem from the consideration of the end-user factor for Prexcel is mainly usability and utility. Our system targets a wide array of audience, some of whom may use the system for business purposes in a variety of fields, and others may use it for educational purposes or even as an assistance in their language learning efforts. Therefore, first of all, it is of crucial importance that Prexcel has a high usability and users from different backgrounds can use the application with ease. We aim to achieve this with an intuitive GUI design, and carefully crafter tutorials on both how to give effective presentations as well as

how to use the app effectively. Second criteria in this part is utility. Prexcel must cater to the needs of its target audience. Therefore it includes many functionalities like exporting reports and transcripts to pdf, or an account system that allows users to access their past presentation data from any computer, both of which are fundamental features for both educational and for business use.

In conclusion, our top consideration for end-user factors are usability and utility.

7.1.2 Consideration of Performance Factor and Related Design Goals

Our consideration of performance factor and related design goals for Prexcel is mainly the reliability and the response time. As previously discussed, Prexcel has 2 main functionalities, one is the detailed end analysis and the other functionality is the live-feedback during presentations. For both of these functionalities, it is important that the app reliably predicts the words the user spoke (for speech to text conversion) and whether the user is looking at the screen/camera above a certain accuracy threshold, which is for the purposes of our app, 70% of the time for an average English speaker. In addition to this, especially 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. Therefore, whenever the user stops for more than 2 seconds, the app should make a word recommendation to the user in no longer than 1.5 seconds. Furthermore, the statistics presented to the user on the screen such as the volume of his/her voice or the WPM of the user, should be at most the statistics from 2 seconds before the moment the statistic is presented. In order to achieve the optimal reliability and response times, some performance trade-offs might be necessary and additional memory space may be used in order to enhance the performance in the other areas mentioned. Still, the total memory requirements of the programme should not exceed 2 GBs at any time, so that the app can also be used by people who do not have high-end computers.

In conclusion, our top consideration factors are reliability and response times. This might come at a trade-off of memory space.

7.1.3 Consideration of Design Factor and Related Design Goals

Our consideration of design factors and related design goals for Prexcel is robustness. 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 efforts. 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 conclusion, our top consideration of design factor is robustness.

7.1.4 Consideration of Maintenance Factor and Related Design Goals

Our consideration of maintenance factor for Prexcel is extensibility and modifiability. We especially wanted to enhance the system's extensibility and modifiability by designing the system as modular as possible. We aimed to design Prexcel in a way so that the "top-level" subsystems (which are divided into subsystems within themselves), the database, the graphical user interface, and the backend processes that make-up the app have as minimal coupling as possible and are modular. This allows for different teams working on these subsystems to work independently from each other and an easier division of labor as well as a higher maintainability. The classes should communicate with each other through communicating functions (similar to the facade design pattern) and as long as these

communicating classes/functions are not changed, the changes in one subsystem should not affect the others as much as possible. This design should render the app very extensible and modifiable.

In conclusion, our top consideration of maintenance factor is extensibility and modifiability.

7.2. Ethics and Professional Responsibilities

There is an array of professional responsibilities and ethical issues related to the development of Prexcel. A major part of these issues are rooted in the fact that the main functionality of Prexcel involves processing user's personal data. This issue is subject to various laws and regulations throughout the world and these regulations were an important factor to consider during our design of Prexcel. Kişisel Verileri Koruma Kanunu (KVKK) [8] in Turkey and General Data Protection Regulations (GDPR) [9] in the European Union are the most important regulatory laws to consider in this matter for our case.

In general, a big portion of these regulations are not relevant to our design of Prexcel, because these laws deal with the long-term storing and the processing of this data. They, however, still affected our design of Prexcel because we had to design our software in such a way that we would not have to deal with these laws. Because of this reason, for example, the database we use does not store the auditory input of the user, but rather the transcript of the auditory input. Because, if we stored the auditory input in an auditory form that contains the user's voice, we would have to conform to a very strict set of regulations set by these laws, and we would have to change our design & implementation. In general, we will present the user with an End User License Agreement (EULA) during the account creation process, which openly discloses how the user's personal data is used; which is that it is only stored temporarily, and is not stored after it is processed.

Therefore, Prexcel was developed especially with these ethical and professional responsibilities in consideration.

7.3. Judgements and Impacts to Various Contexts

Development of our project is affected by various factors like public health factors, cultural factors, global factors and social factors. Similarly, our project is to affect those areas as well.

With regards to public health, Prexcel is expected to have a significant positive impact, especially during the pandemic conditions. Prexcel is a tool that is optimized and geared towards online presentations, and in particular, it improves the users' online presentation abilities and enhances their online presentations with the live feedback features. The increased effectiveness of online presentations may decrease the need for face to face presentations, and help improve public health during the pandemic in this way.

With regards to cultural factors, again, Prexcel has significant positive effects. Today's working culture is swiftly shifting towards remote/online work. This is especially true for software development companies, which usually have a unique culture where working remotely is more accepted and is significantly more prevalent. Prexcel has the potential to significantly improve remote working culture by improving how well people can prepare for and deliver remote online presentations, which makes remote work more viable and desirable from the companies' perspective.

With regards to global factors, education is one of the most important global efforts. Prexcel is a great complimentary educational tool that could help students improve their presentation skills. Furthermore, the word recommendation feature of the live feedback feature of Prexcel

could be very beneficial for students whose native language is not English, during their presentations, if they are receiving an education in English.

These three areas, which are public health, culture (in particular remote/online working culture) and global factors and efforts (in particular education efforts) also significantly shaped our conceivement, analysis and design of our project. First of all, consideration of these 3 areas have shown us the target audience of our system, which are mainly business and educational users who make online presentations. This has led us to shape our analysis and design in a way that caters to this particular audience, by adding functionalities such as progress tracking, a key selling feature for both business and educational purposes, to increase its utility for these audiences.Moreover, the design goals we focused on such as the robustness and reliability are must to have qualities of a system for it to be able to confidently be used in a business environment for business purposes. Furthermore, since these audiences may come from a wide variety of backgrounds, it is crucial that the system has high usability and people from these different backgrounds can use the system intuitively and with ease, which are again, design goals we prioritized.

With regards to social factors, establishing effective communication and expressing one's ideas is extremely crucial. With this in mind, Prexcel enhances the user's ability to express themselves in a better way by giving them directions and statistics on how they can improve. Therefore, while giving online presentations the presenters are also going to develop themselves in their social relationships.

With regards to economic factors, effective communication and presentation skills are a very important skill in life that many people struggle with, and in general, many people spend a lot of money on various effective speech & presentation courses and resources. Prexcel is a tool that will help people improve those skills for free, saving many people a considerable

amount of money. In addition to this, the above mentioned effects on remote work also have a significant positive economic impact.

With regards to environmental factors, as also previously mentioned, Prexcel makes remote work more feasible. Making remote work more feasible would, in time, stop migration to big cities & allow people working in smaller towns & villages, and help stop the excess crowding in cities. This would mean less time spent commuting, consequently less pollution and a net positive impact on the environment.

The latter three factors, that are the social factors, economic factors and environmental factors, all enhance the importance of a system/application like Prexcel for its target audience, which just further cements significance of Prexcel's designs goals such as utility and usability, so that these target audiences can use the application for their needs effectively and easily.

Table 1: Factors that can affect analysis and design.

	Effect level	Effect
Public health	5/10	Prexcel helps the users improve their online presentation skills, as well as enhancing their online presentations by live feedback features. The increased effectiveness of online presentations may decrease the need for face to face presentations, and help improve public health during the pandemic.
Public safety	0/10	None.
Public welfare	0/10	None.
Global factors	7/10	Education is one of the most important global efforts. Prexcel could greatly help

		students improve their presentation skills as a complementary tool.
Cultural factors	4/10	Usually, software development companies have a unique culture where working remotely is more prevalent and accepted. Prexcel helps people develop better online presentation skills, making remote working more feasible and desirable, making a cultural impact in this aspect.
Social factors	7/10	Prexcel allows the users to explain their ideas on a deeper level, thus elevating their social skills.
Economical factors	9/10	Prexcel would help people develop their presentation skills without attending paid courses, helping many people save money. Furthermore, promoting remote work would also have a positive impact in this regard.
Environmental factors	5/10	Prexcel would have a positive impact on the environment, as it makes remote work more feasible, and makes people spend less time commuting. (Hence less pollution.)

7.4 Teamwork Details

7.4.1 Contributing and functioning effectively on the team

In general, we made a work division where each team member is assigned an equal amount of work. Instead of very large term work divisions, we assigned team members to embark on and learn particular knowledge areas, and then made smaller and more manageable work divisions for shorter terms. Making these work divisions (and consequently the meetings where the results of previous work division as well as the future work division) more

frequently, we increased the accountability and the dependability of each team member. Of course, sometimes during the semester, all team members were not available at some points due to the load of the curriculum. This however did not prevent us from doing effective teamwork as we communicated with each beforehand and planned ahead. Therefore we all did our best to function effectively on the team and contribute as best as we can.

Below, we have listed the areas each member has contributed majorly. However, it is important to note that all members have contributed to all aspects of the project, here, we are listing only the most important/major contributions of each member. For example, each member has contributed significantly in the frontend-backend integration of the project.

Can Kırşallıoba: Took a major responsibility in the backend development of the project, in particular the machine learning models. He also contributed to the design of the backend. Consequently, he took part in writing these areas in deliverable reports as well.

Can Kırımca: He also took a major responsibility in the backend development of the project, including the machine learning models and the database. He also took a major role in the design of the backend. He took part in writing and making diagrams of these areas in deliverable reports.

Burak Yiğit Uslu: He took chief responsibility in the development and configuration of the user interfaces and frontend as well as the development of the logic implemented in Javascript. He also contributed to the design in these areas. He also contributed to the reports in these areas as well as taking a major role in writing more different context-related analysis based parts (such as the “Consideration of Various Factors in Engineering” section and such) of the reports.

Alper Sarı: He also took large responsibilities in the backend development, but in non-ML areas including the data management, file management as well as the database. He contributed to the design in these areas as well. Finally, he took the tasks in various areas of the reports including the formalization and analysis of the requirements.

Barış Tiftik: He also took important responsibilities in the development of the user interfaces with Burak, in particular the area of data management in the frontend. He also contributed to the design of these frontend elements. Finally, he contributed to the reports in mostly editorial ways.

7.4.2 Helping creating a collaborative and inclusive environment

Communication is the important tool for creating a collaborative and inclusive environment. We paid great mind to making sure that lines of communication between members of the team were open and clear. It was important for us that we were aware of each other's shortcomings to be able to work together without any friction between team members. When conflicts arose, whether that conflict was about the end product or the deadlines, we settled those conflicts by taking all members' inputs into consideration.

Below are some of the ways each team member has collaborated with each other:

Can Kırşallıoba: Collaborated with Burak to better understand the design of the user interface, which led to developing the word recommendation module in a way that is more compatible with the user interface design. Also collaborated with Can Kırımca on the output of the DeepSpeech model. Collaborated and shared knowledge with the entire team to figure out the details of the integration of the backend processes with the frontend. Lastly, worked with Can Kırımca to design subsystem decomposition and services.

Can Kırımca: Collaborated with Can Kışallıoba for the research and initial implementation of word recommendation and speech-to-text models. Also worked with Can Kırşallıoba to design subsystem decomposition and services. Collaborated with Burak and Barış for the integration of frontend and backend parts of the project.

Burak Yiğit Uslu: Collaborated greatly with Barış for the implementation of the user interface. Collaborated with all team members for the integration of the frontend to backend. Collaborated with Can Kırşallıoba and Can Kırımca for the design and subsystem decomposition of the project for the parts that include UI elements and logic implemented in Javascript.

Alper Sarı: Collaborated closely on the integration of back end python scripts with the UI system as well as the ML scripts that are used for presentation analysis and live feedback. Closely worked with Burak Uslu, Barış Tiftik and Can Kırımca as well as Can Kırşallıoba on the aforementioned matters.

Barış Tiftik: Shared the knowledge and collaborated greatly with Burak Yiğit Uslu for the UI design and implementation. Worked together with him to create the necessary UI screens. Collaborated with the entire team members for the integration of the frontend to backend.

7.4.3 Taking lead role and sharing leadership on the team

The decisions that concern the end project were taken democratically. Throughout the development of the project, each team member have taken leadership roles in parts that they have strong knowledge and experience in (compared to other team members), and followed the leadership of other members or shared the leadership with them in areas where other members have strong experiences and knowledge. In general we shared the

leadership of these areas (or work packages) mostly equally so that every member of the team could hold some responsibility over parts of the project.

In particular, each member have taken lead roles and shared the leadership in the following way:

Can Kırşallıoba: Shared the leadership position for the Word Recommendation and Speech to Text modules with Can Kırımca. And started the implementation of the Word Recommendation module. Collaborated with Burak Yiğit Uslu on the connection aspect of the user interface. Also collaborated with Can Kırımca on how the DeepSpeech module gives its output (the format and the type of the data it gives is very important as it will be directly used for the input to the Word Recommendation model).

Can Kırımca: Shared the leadership position for the Word Recommendation and Speech-to-Text models with Can Kırşallıoba. Trained an LSTM network for word recommendations. Worked on the integration of frontend and backend processes. Also took the leadership position for the database integration of the project with Amazon AWS server. Worked with Alper Sarı to make detailed speech analysis and determine the logic of different presentation scores.

Burak Yiğit Uslu: He took the lead role in the development of user interface and logic implemented in Javascript. He delegated responsibilities to the team members for the parts that relate to the development of the UI or integration of the backend parts to the UI, and followed the leadership of the other team members when working in/contributing to the areas of the project that are under the leadership of other team members, mainly backend parts (especially when modifying them to integrate with the UI) and thus shared leadership for this process.

Alper Sarı: Took a lead role regarding the back end processes of gathering user input and its integration to both the UI of the application as well as the machine learning modules. It was deferred to Burak's leadership of the UI front where they worked to find out how it could be integrated and call the python scripts necessary for user audio and video input. On the other hand work with Can Kırımca was done with regards to how we would pass data between user input during live presentation and the ML models.

Barış Tiftik: Took lead role in the development of the data management of the frontend. Took initiative and autonomy for the development of parts of the UI screens. Followed the leadership of other team members for the general integration of the UI and in order to create a compatible UI.

7.4.4 Meeting objectives

In general, we can say the project has met its objectives. For all work packages, the functionalities/tasks that we want to implement are implemented. Of course, throughout the development process, some changes and modifications were made to some of the initial designs of the software architecture, user interface as well as the content of the work packages. However, in the end, all of the functionalities that were initially included in the requirements in the Prexcel are, right now, implemented in Prexcel.

The only exception to this is the functionality about importing reports to pdf. We decided to scrape off this feature because of a few reasons. Firstly, the scope of the JavaScript libraries that were used for formatting pdfs were very limited and did not allow us to format pdfs as we wanted, which made them aesthetically not as we desired. Secondly, and more importantly, we actually initially came up with that feature when we thought of the application as an application that might not have a database, in the very early stages of project specification. However, we decided to include a database in the application, while still in the

project specification stage. Looking retrospectively, exporting to a pdf function has lost its purpose. Currently, the user can access all of his/her presentations from anywhere by logging into his/her account and see the progress reports as he/she desires in the format we desire. Therefore, in the end, we decided to scrape off this exporting reports to pdf functionality. This minor functionality is the only functionality that we included in any of the prior documents and then decided not to implement.

Looking at the meeting the objectives criteria from a schedule perspective, we can say that in general the project has generally met the schedule we specified on our Gantt Chart in the analysis report. In general, initially, tasks moved slower than we planned, due to the burdensome learning processes we needed to develop effectively. After a while however, especially in the second semester, we moved faster than planned as we had gained more experience by then, and closed the gap.

7.5 New Knowledge Acquired and Applied

For the project, we had to acquire a lot of knowledge in a variety of different fields. In general, we lacked the knowledge in the areas of backend development with Python and Django, applications of machine learning models, frontend development using JavaScript, React and Electron, cloud databases (AWS) and verification and validation.

In general each team member has acquired significant knowledge in all of the areas listed above. However, for the effective use of the team’s time, we made the work division and each pair of members took the lead role of acquiring knowledge themselves in a particular area, and then helping others contribute to these parts by sharing their knowledge and leading others. In particular, Burak Yiğit Uslu and Barış Tiftik took the lead role in learning, applying and helping others contribute to frontend development with JavaScript, React and Electron. Similarly, Burak Yiğit Uslu and Can Kırşallıoba took lead role in verification and

validation, Can Kırımca and Can Kırşallıoba in machine learning applications, Can Kırımca and Alper Sarı in cloud databases. As stated above, each member has gained large knowledge in their respective fields, however, each member has also gained important knowledge in all of the areas listed above as well.

We applied many learning techniques and methodologies in order to acquire knowledge. Since the project was a multi semester effort, we learnt most of the materia in some of the areas by taking the relevant courses. Can Kırımca and Can Kırşallıoba have taken the introductory machine learning course for this purpose and similarly, Burak Yiğit Uslu and Can Kırşallıoba have taken software verification and validation courses. Furthermore, to learn the usage of tools such as Flask, React and Electron, various online courses were followed as well as their official documentation and tutorials were followed.

8. Conclusion and Future Work

Currently, extracting the executable file from the backend python packages does not work, which is a regret to us. This happened because we are using lots of imported packages and pyinstaller (the executable extractor that we have used) is known to have trouble with some of the imported packages that are used in the project. We plan to resolve this issue in the future. There are a few items that we plan to add to Prexcel in the future. First and perhaps the most important of all of these is maybe the support for other languages.The process for adding new languages to Prexcel is actually easy, DeepSpeech already has speech recognition libraries/models for most of the common languages, and almost all of our metrics are compatible with any language. In order to add a particular language, we only need to make a list of the “filler words” in that language, in addition to a language selection screen that is common for languages. Therefore we can add many languages to Prexcel as we like with relative ease. The reason we chose to not add languages other than English in this version of the application is that the speech recognition model for any language is quite large

on the disk, above 1 GB, which makes the app unnecessarily bulky. Therefore, as we only needed the English language version for the purposes of this assessment, the version we are turning in only includes the English language and works only for English for the convenience of evaluators. However, in the future, we will definitely add more languages to Prexcel.

Other than this, when one uses machine learning models, he/she can always try to improve the accuracy of the models and such. A second possible area of Prexcel that might be developed further in the future is its machine learning models.

8.1 Conclusion

In conclusion, we saw the change in our daily routines due to changing conditions in favor of online working and realized the necessity for an application that assists us and other people in this domain. This is how we realized the requirements for Prexcel. From there on we analyzed, designed and implemented our application based on the very real requirements of us and people around us. In the end we managed to develop an application that has a great and sleek user interface and design and uses cutting-edge machine learning algorithms to give feedback on and improve its users' presentations.

The journey has been quite difficult, but at the same time very enjoyable and informative for all of the team. All of us had to go out of our way to learn new technologies that we had not known before in order to build Prexcel as we had envisioned it. After all, we are all happy with the result.

9. Glossary

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

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

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: An open source computer vision library.

React: A Javascript framework developed by Facebook. It is used for creating 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.

Speech-To-Text: Conversion of auditory input to text format.

10. References

[1] DeepSpeech, https://deepspeech.readthedocs.io/en/r0.9/#, [Accessed: Oct. 8, 2021].

[2] OpenCV, https://opencv.org, [Accessed: Oct. 8, 2021].

[3] JavaScript, https://www.javascript.com/, [Accessed: Nov. 14, 2021].

[4] Electron, https://www.electronjs.org, [Accessed: Oct. 10, 2021].

[5] React, https://tr.reactjs.org/, [Accessed: Nov. 14, 2021].

[6] Python, https://www.python.org/, [Accessed: Nov. 14, 2021].

[7] MySQL, https://www.mysql.com/, [Accessed: Nov. 14, 2021].

[8] T.C. Cumhurbaşkanlığı Mevzuat Bilgi Sistemi, “Kişisel Verilerin Korunması Kanunu(KVKK)”, Apr. 7, 2016, https://www.mevzuat.gov.tr/MevzuatMetin/1.5.6698.pdf, [Accessed: Oct. 9, 2021].

[9] The European Parliament and the Council of the European Union, “General Data Protection Regulation”, Apr. 27, 2016, https://gdpr-info.eu/, [Accessed: Oct. 9, 2021].