Bilkent University

Department of Computer Engineering

Senior Design Project

Project name: Upfix

Analysis Report

Meryem Efe, Hamza Pehlivan, Özge Yaşayan, Hazal Aksu, Rabia Nur Önal

Supervisor: Uğur Güdükbay

Jury Members: Can Alkan, Çiğdem Gündüz Demir

Contents

Analysis Report

Project Short-Name: Upfix

1. Introduction

Upfix is a tool for upscaling gameplay videos. The user can choose between the website and the desktop application, according to their needs.

The website is capable of upscaling a video provided via a URL from Twitch or YouTube, as well as a video that the user uploads to the system. The upscaling is done on the server, whereas the desktop application makes use of the resources of the user's computer, which is quite useful when the user does not have a reliable Internet connection. A video from the user's local file system can be upscaled on the desktop application.

Upfix uses artificial neural networks in order to upscale videos. A pre-trained neural network provides a way for users to enjoy watching gameplay without being affected by the speed of their Internet connection, which is the main advantage of using Upfix. Upfix aims to improve the video viewing experience for users by eliminating interruptions and increasing the quality of the videos.

Currently, there are video upscaling applications on the market, however, none of them are specialized for use in the gaming industry.

2. Proposed System

2.1. Overview

Upfix is a website and a desktop application for upscaling videos. It does not require making an account/logging in. The users can simply download the desktop app or provide a link to the website to start using the functionality of Upfix.

Both the website and the desktop application will provide a media player for the user to watch the upscaled videos.

The user can upload a video from their local files to the desktop application and wait for the neural network to upscale it. After the video is done being upscaled, the user can play the video and save it to their computer. The desktop application uses the resources of the user’s computer to upscale the videos.

Similarly, the user can provide a link to a video such as a YouTube or Twitch link, and wait for the neural network to upscale it. After the upscaling process, the user can play the video and download it to their computer. The web application uses the remote server’s resources to upscale the videos.

2.2. Functional Requirements

Desktop Application:

- The user can upload a video from local memory to be upscaled.
- The program will upscale the uploaded video using different algorithms with GPU or CPU offline.
- The user can watch the upscaled video.
- The user can save the upscaled video to local memory.

Web Application:

- The user can give a link to a video to be upscaled.
- The program will find the video from the given link.
- The program will upscale the video using different algorithms on an online server.
- The user can watch the upscaled video.
- The user can download the upscaled video.

2.3. Non-functional Requirements

2.3.1. Performance

- The user should be able to start watching the upscaled video in at most 1 minute.
- The response time for the website should not exceed 2 seconds.
- The response time for the desktop application should not exceed 2 seconds.

2.3.2. Usability

- The system should be easy enough to be used by anyone of all ages with basic computer skills.
- The system should come with an explanatory user's manual.

2.3.3. Scalability

- The desktop application will be able to upscale one video at a time.
- The web version should be able to serve at least 10 users at a time. The processing power of the server is to be increased in the future.

2.3.4. Extensibility

- The system should be designed in a way that will make it easy to integrate new video streaming platforms, new upscaling algorithms and support for more games in the future.

2.3.5. Security

- The system should not disclose the data of its users to third parties.

2.4. Pseudo Requirements

- The web application will be developed using Spring framework and React.JS, the desktop application will be developed using Java FX [1,2].

- The deep learning models and image processing parts will be written in Python 3.
- The version control of the project will be done via GitHub [3].
- IntelliJ IDEA Ultimate version will be used during the development.
- The web application will be available for internet users.
- The desktop application will be available on the project webpage for download and will not require internet connection after installation.
- Tensorflow 2.0 and Pytorch 1.6.0 libraries will be used to create artificial neural networks [4,5].
- Google Colab will be used to train the models efficiently [6].
- To perform image processing tasks, OpenCV 4 library will be used [7].
- An Nvidia graphics card for CUDA enabled processing will be needed when a user prefers inferring the data with GPU.
- The language of the system will be English.

2.5. System Models

2.5.1. Scenarios

2.5.1.1. Web Application Scenarios:

Scenario 1- Upload Video:

Use Case: Upload Video
Actors: End User, Server, Web Crawler
Entry Condition: User clicks the “Upload” button and gives a link to the application.
Exit Conditions: The video is successfully uploaded to the server.
Flow of Events:

1. User clicks on the “Upload” button and gives a link to upload a video to the server.
2. Web crawler finds the specified video from the internet.
3. Server obtains the video found by web crawler.

Alternative Flow of Events:

A. Download is cancelled by the user.
   a. User interrupts the uploading process.
   b. Web crawler is notified and it stops finding the video.
   c. The program waits for another input from the user.

B. Provided link cannot be found.
   a. User gives a link to the application.
   b. Web crawler cannot find the given link.
   c. Web crawler notifies the user about the error.
   d. The program goes back to the main page.

Scenario 2 - Upscale Video:

Use Case: Upscale Video
Actors: End User, Server
Entry Condition: The user clicks on the “Next” button to upscale a low resolution game video on the server.
Exit Conditions: The video is upscaled and returned to the user.
Flow of Events:

1. User selects the game type to be upscaled.
2. User selects the neural network type to be used.
3. The user initiates upscaling process by pressing the “Next” button.
4. Server upscales the video using parameters given by the user.
5. Server returns the upscaled version to the user.

Alternative Flow of Events:

A. User aborts the process.
   a. During upscaling, the user chooses to cancel the process.
   b. Server aborts the upscaling process.
   c. The program goes back to the main page.

Scenario 3 - Watch Video:

Use Case: Watch Video
Actors: End User
Entry Condition: User clicks the “Play” button.
Exit Conditions: Video ends or the user stops the video.
Flow of Events:

1. User clicks on an upscaled video to watch.
2. User watches the video until the end.

Alternative Flow of Events:

A. Video is aborted by the user.
   a. The user stops the video.
   b. The program waits for another input from the user.

Scenario 4 - Download Video:

Use Case: Download Video
Actors: End User, Server
Entry Condition: User clicks on the “Download” button to download an upscaled video.
Exit Conditions: Download finishes successfully or connection problem occurs.
Flow of Events:

1. User starts downloading an upscaled video.
2. Download finishes successfully.

Alternative Flow of Events:

A. Download is aborted by the user.
   a. The server stops sending the video.
   b. Program waits for another input from the user.
B. Internet connection is lost.
   a. The program notifies the user about internet connection issues.
   b. Download is aborted by the server.

2.5.1.2. Desktop Application Scenarios:

Scenario 1- Upload Video:

Use Case: Upload Video
Actors: End user, File Manager
Entry Condition: The user is in the main page. The user gives a path to low resolution video and clicks on the “Upload” button.
Exit Conditions: The video is successfully uploaded or the process is interrupted.
Flow of Events:

1. The user enters the path in which low resolution video is located.
2. File manager navigates the path and finds the specified video.
3. File manager uploads the video to the application.

Alternative Flow of Events:

A. The path does not exist.
    a. File manager searches the video, however, it cannot find the video.
    b. File manager notifies the user.
    c. The program waits for another input from the user.

Scenario 2- Upscale Video:

Use Case: Upscale Video
Actors: End User
Entry Condition: The video is uploaded successfully and the user clicks on the “Next” button.
Exit Condition: The video is upscaled and returned to the user.
Flow of Events:

1. User chooses the game type to be upscaled.
2. User selects the neural network type to be used.
3. User clicks on the “Next” button.
4. The program upscales the video using parameters given by the user.
5. The upscaled version is returned to the user.

Alternative Flow of Events:

A. Nvidia GPU cannot be found.
    a. The installation of the Nvidia GPU driver is incorrect or the user does not have an Nvidia GPU driver.
    b. The application aborts the upscaling process
    c. The application notifies the user about the error.
    d. The application returns to the main page.

B. User cancels the upscaling process.
    a. User aborts the upscaling process.
    b. The application is notified and it cancels the upscaling process.
    c. The application waits for another input from the user.

Scenario 3- Watch Video:

Use Case: Watch Video
Actors: End User
Entry Condition: User clicks the “Play” button.
Exit Conditions: Video ends or the user stops the video.
Flow of Events:

1. User clicks on an upscaled video to watch.
2. User watches the video until the end.

Alternative Flow of Events:

A. Video is aborted by the user.
   a. The user stops the video.
   b. The program waits for another input from the user.

Scenario 4- Save Video:

Use Case: Save Video
Actors: End User
Entry Condition: User clicks the “Save” button and chooses a place to save the video.
Exit Conditions: Video is saved successfully.
Flow of Events:

1. User chooses a path and saves the video.
2. The video is saved successfully.

Alternative Flow of Events:

A. Saving process is aborted by the user.
   a. User stops the process by clicking on the “Cancel” button.
   b. The program returns back to the main menu.

2.5.2. Use Case Models

2.5.2.1. Web Application Use Case Model

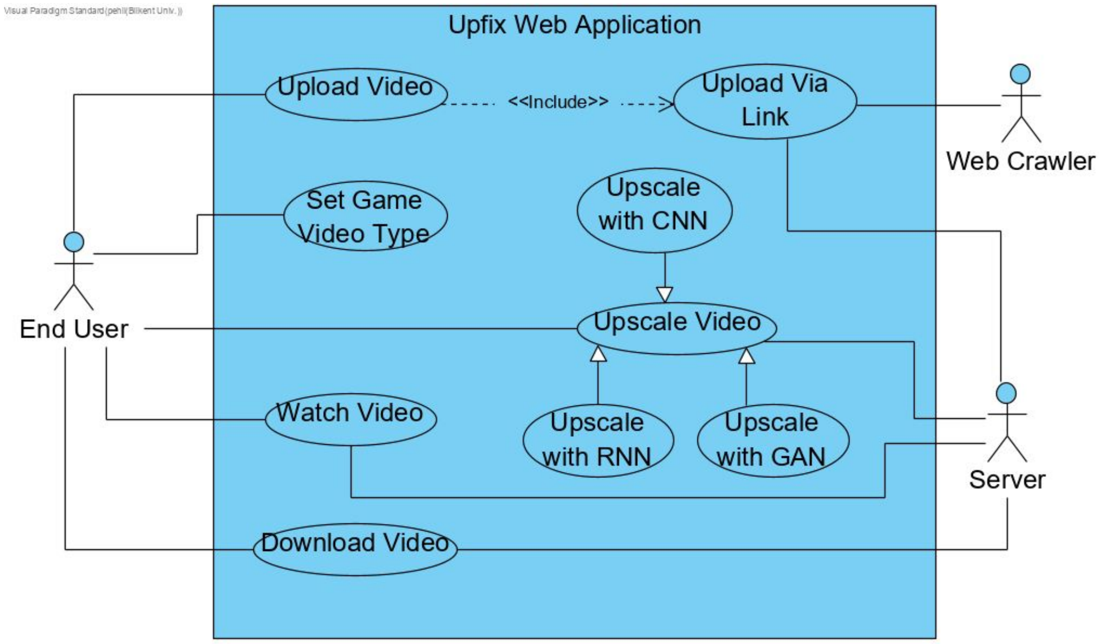


Figure 1: Use Case Diagram of Upfix Web Application

2.5.2.2. Desktop Application Use Case Model

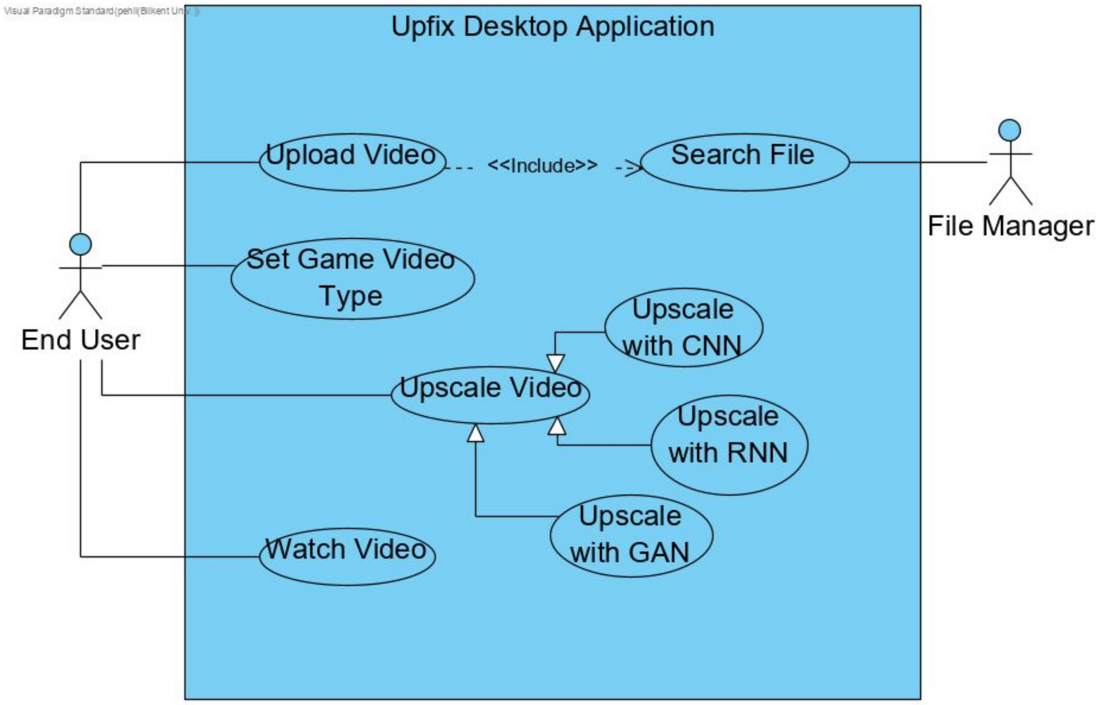


Figure 2: Use Case Diagram of Upfix Desktop Application

2.5.3. Object and Class Models

2.5.3.1. Web Application Class Diagram

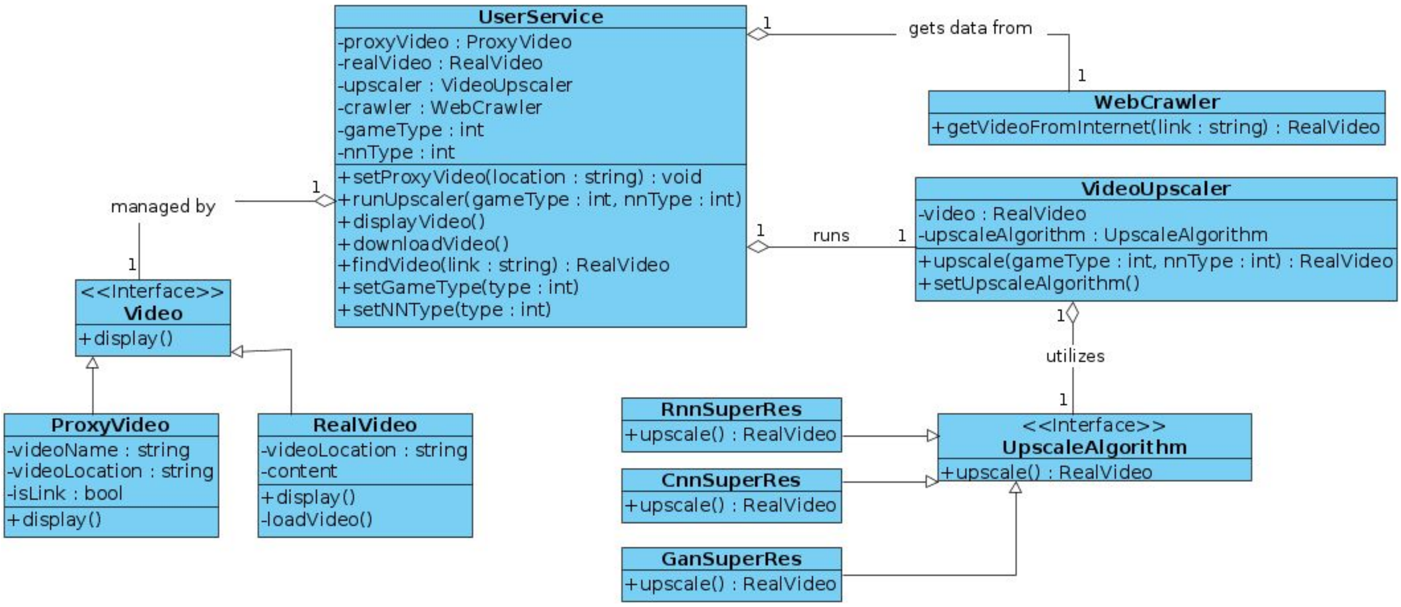


Figure 3: Class Diagram of Upfix Web Application

2.5.3.2. Desktop Application Class Diagram

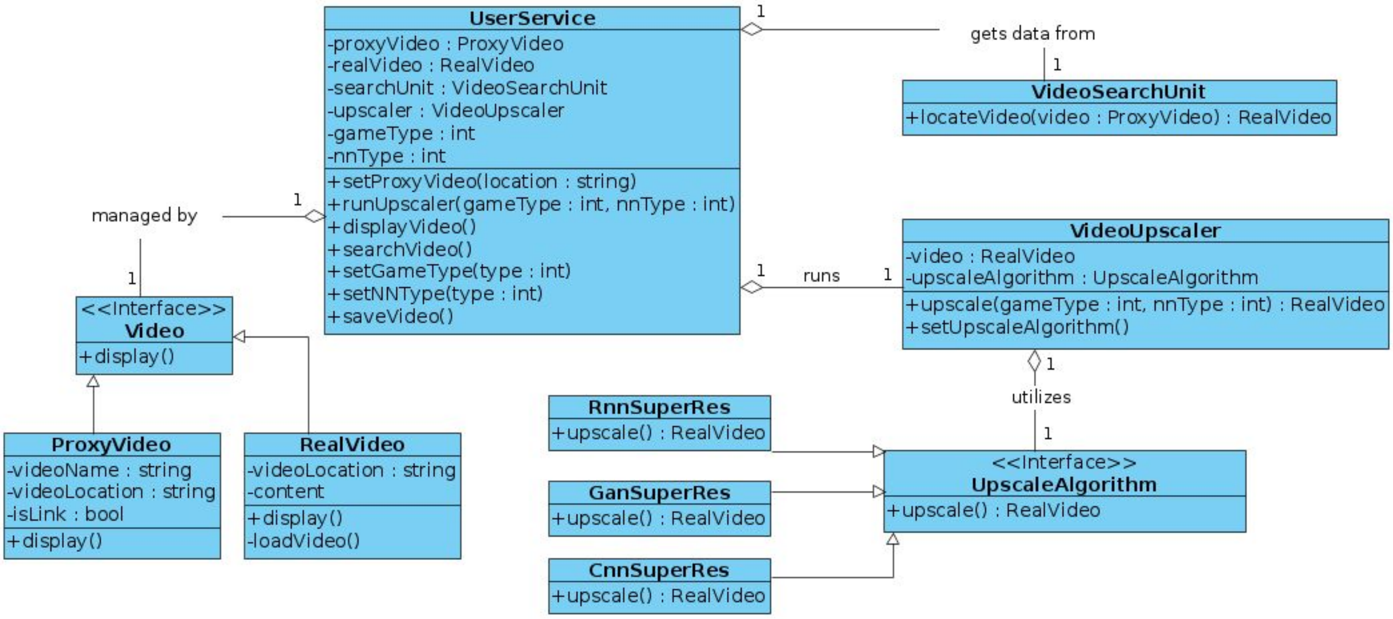


Figure 4: Class Diagram of Upfix Desktop Application

2.5.4. Dynamic Models

2.5.4.1. Sequence Diagrams

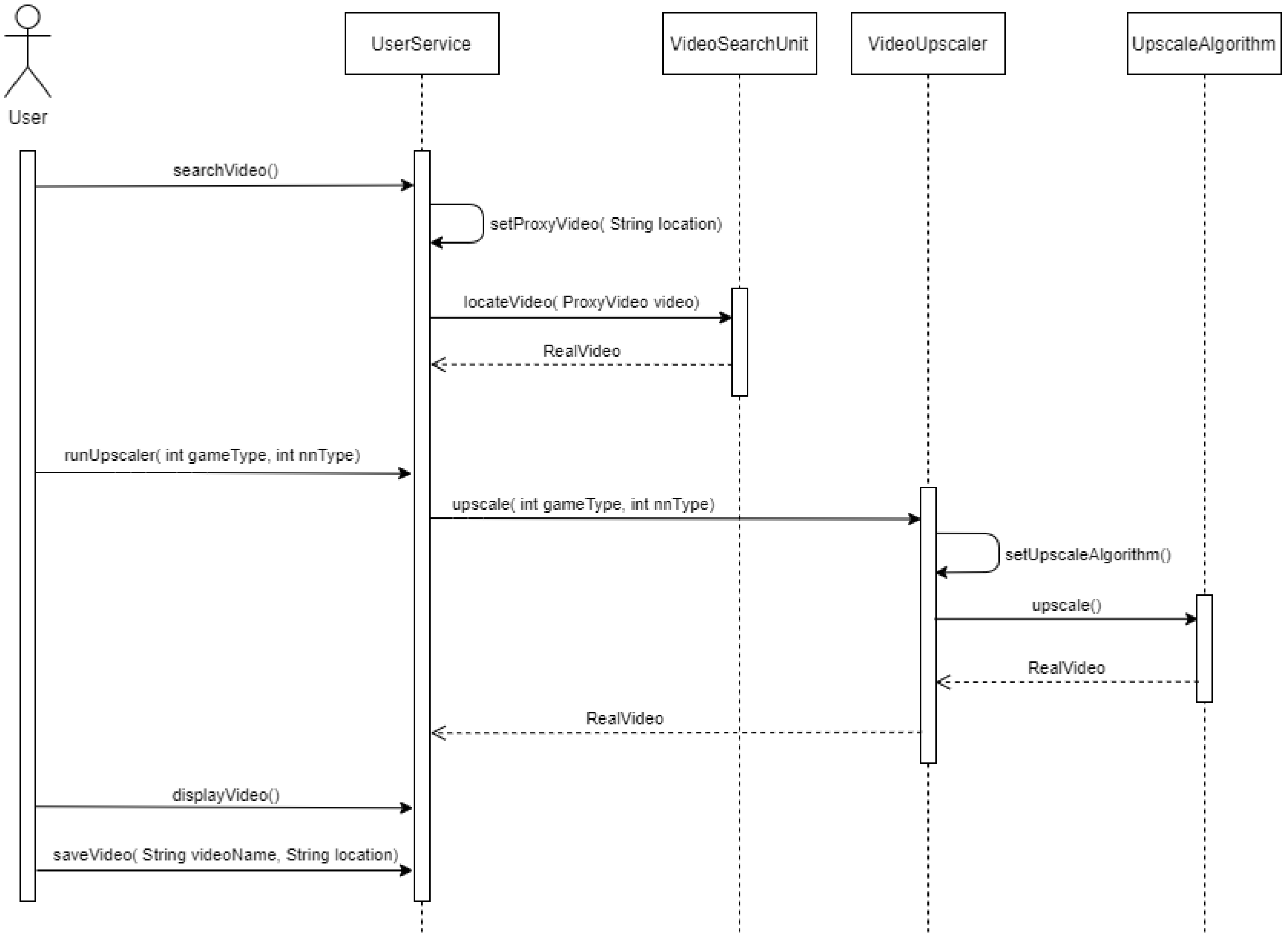

User
UserService
VideoSearchUnit
VideoUpscaler
UpscaleAlgorithm
searchVideo()
setProxyVideo(String location)
locateVideo(ProxyVideo video)
RealVideo
runUpscaler(int gameType, int nnType)
upscale(int gameType, int nnType)
setUpscaleAlgorithm()
upscale()
RealVideo
RealVideo
displayVideo()
saveVideo(String videoName, String location)

Figure 5: Sequence Diagram of Upfix Desktop Application

This diagram represents the entire process that users can carry out in the desktop application. The system does not require any authentication steps. Therefore, the process starts with uploading a video from local memory. VideoSearchUnit is the responsible class for converting a proxy video to a real video. After uploading the video, the users then can continue the upscaling process. In order to do that, users have to choose the game type of the video and the type of the neural network which they want to use. This is significant because a new model will be trained for each game separately. For that reason, the system should be informed about which model it should run beforehand. VideoUpscaler and UpscaleAlgorithm classes are responsible for running the proper model and upscaling the video. Additionally, after the upscaling process, users can watch the video and can save the video to local memory if they wish to do so.

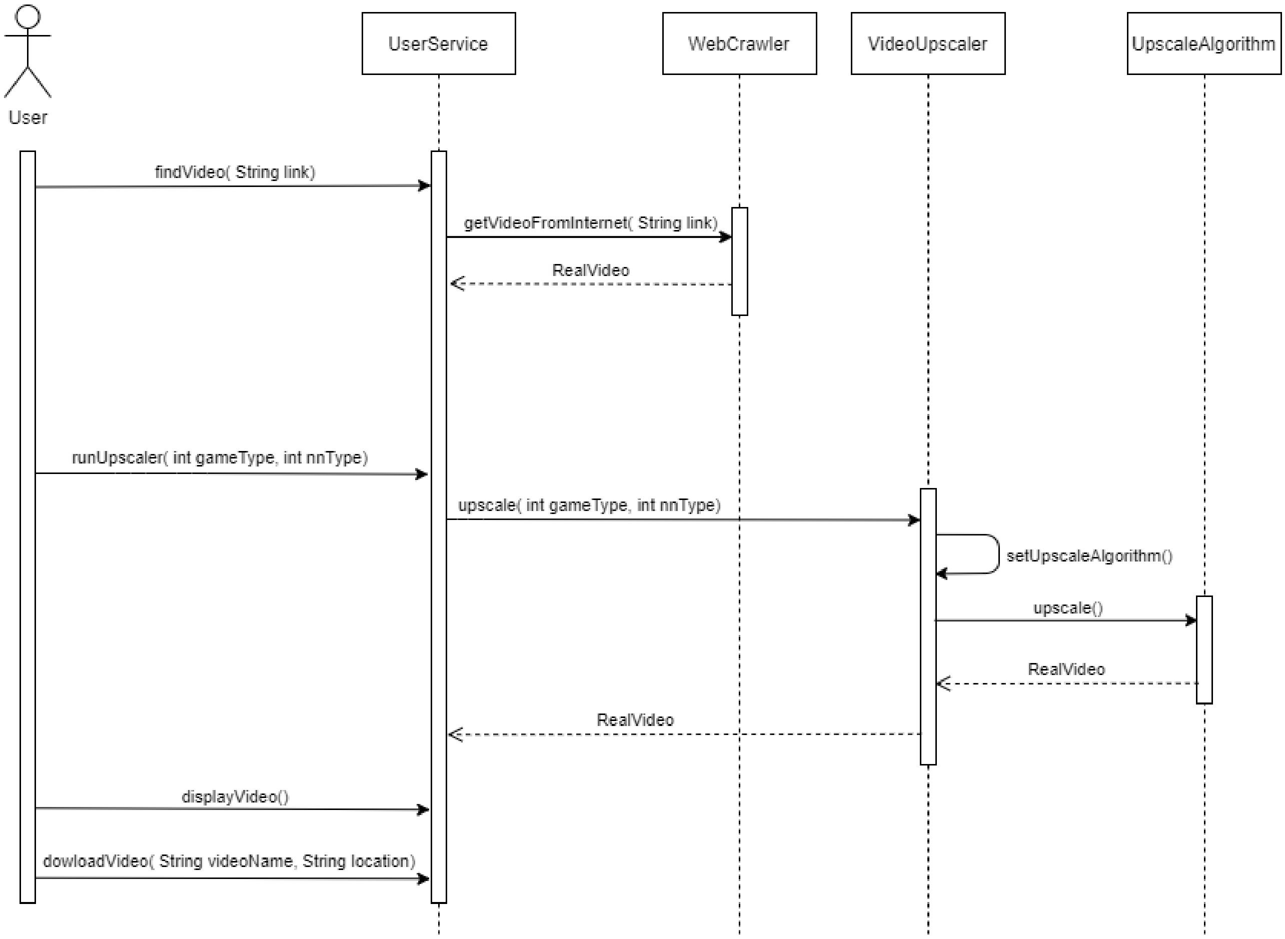

User
UserService
WebCrawler
VideoUpscaler
UpscaleAlgorithm
findVideo(String link)
getVideoFromInternet(String link)
RealVideo
runUpscaler(int gameType, int nnType)
upscale(int gameType, int nnType)
setUpscaleAlgorithm()
upscale()
RealVideo
RealVideo
displayVideo()
dowloadVideo(String videoName, String location)

Figure 6: Sequence Diagram of Upfix Web Application

This diagram represents the entire process that users can carry out in the web application.This diagram is very similar to the diagram in Figure 5 except for two differences. The first difference is the class which is responsible for the video uploading process. In the web application, WebCrawler class searches the video on the Internet and returns it to UserService. The second difference is that after the uploading process, users can download the upscaled video.

2.5.4.2. Activity Diagrams

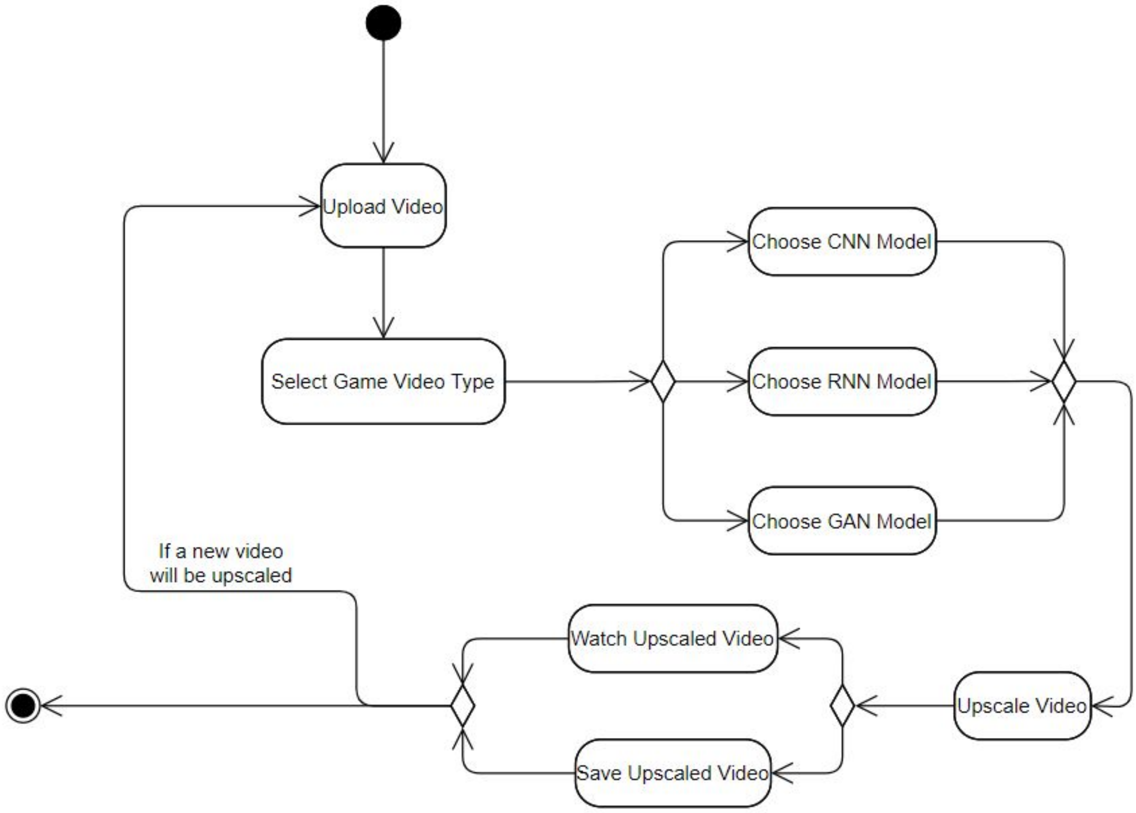


Figure 7: Activity Diagram of Upfix Desktop Application

When the desktop application is opened, the user is met with the main screen that prompts them to upload from their computer. After, the application asks the user to select the game type of which the uploaded video belongs to. In the next step, the application provides options for the user to select which neural network model should be used to upscale. After choosing the model, the application moves onto the upscaling step. The user can watch the upscaled video or save the video to local memory. If the user wants to upscale a new video, the process goes back to the beginning. Otherwise, the program is finished.

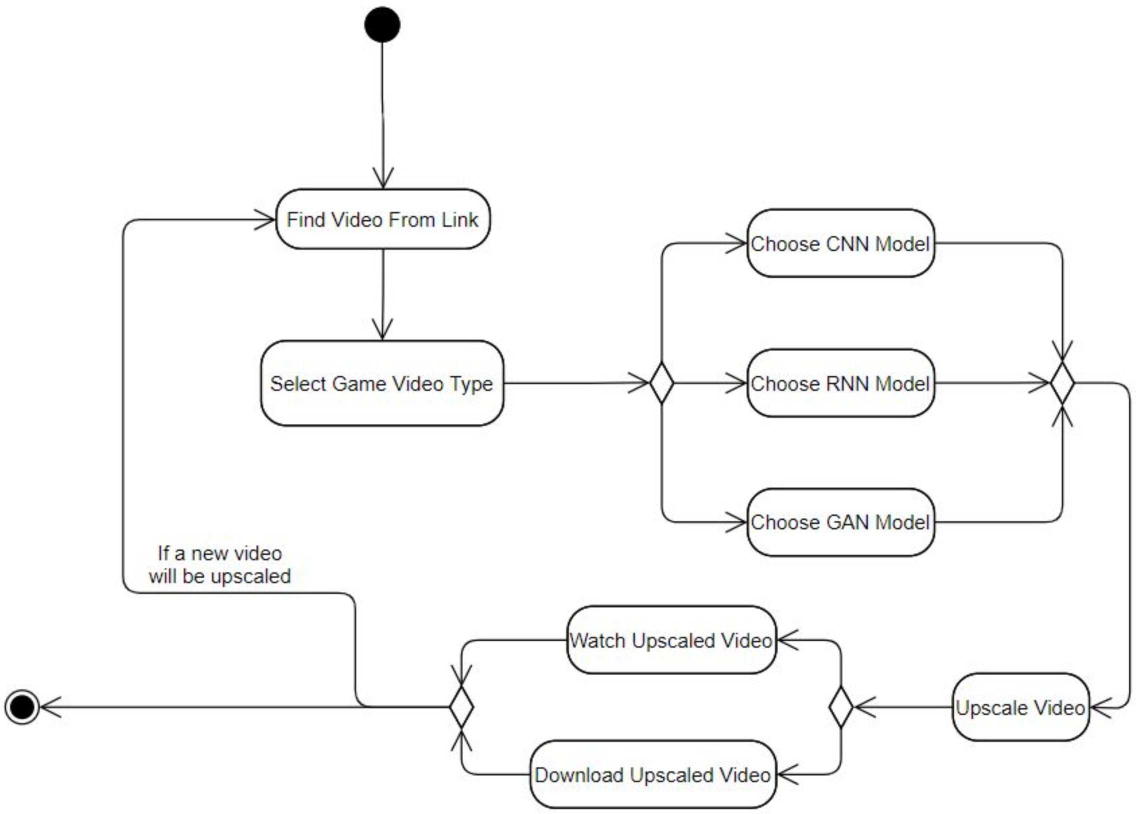
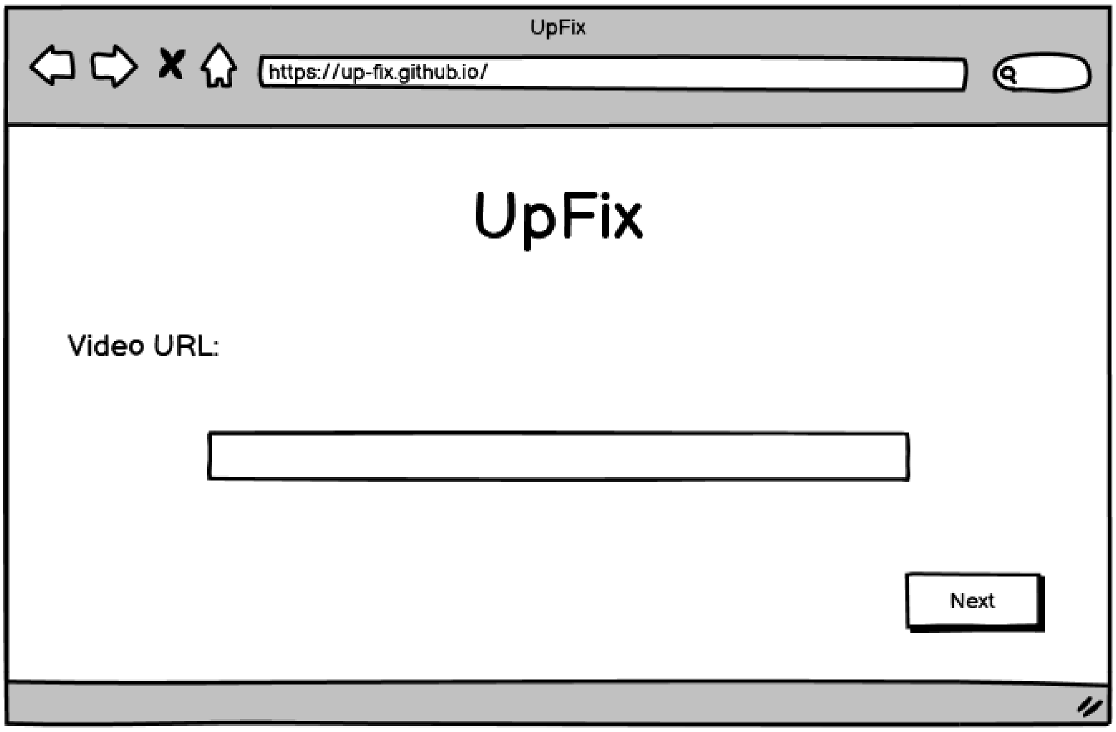


Figure 8: Activity Diagram of Upfix Web Application

Activity flow in the web application is very similar to the activity flow of the desktop application except for two differences. The first difference is that the application finds the video from the internet. The second difference is that after the uploading process, users can watch or download the upscaled video.

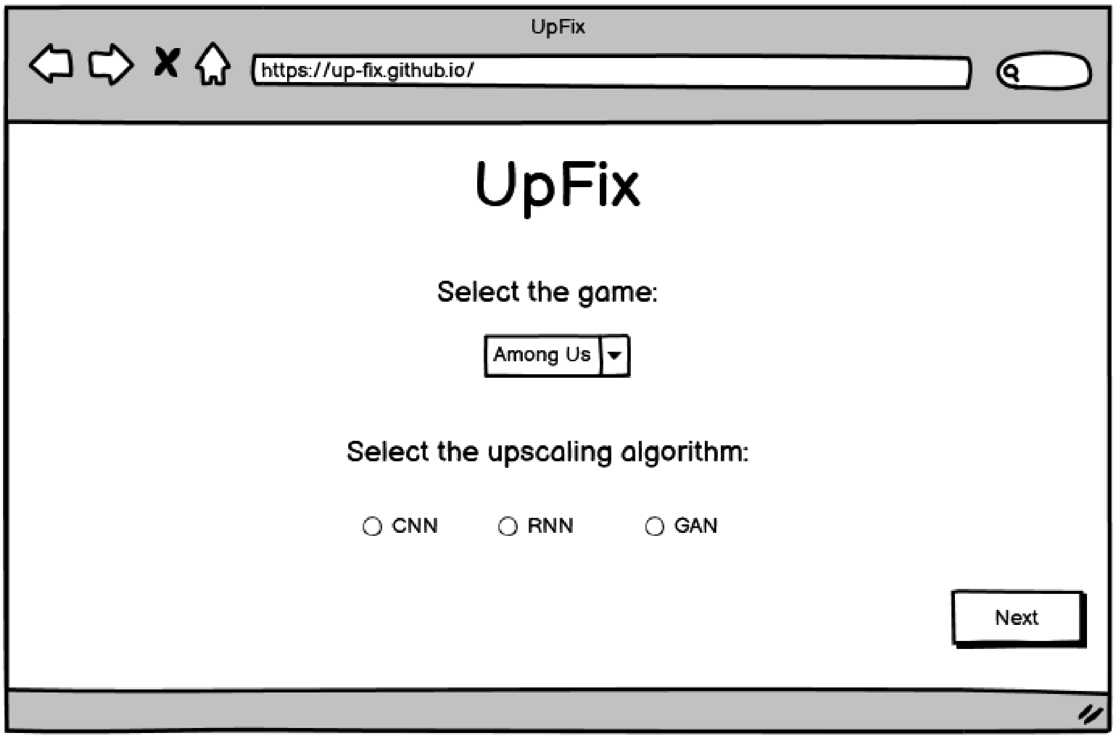
2.5.5. User Interface - Navigational Paths and Screen Mock-ups



https://up-fix.github.io/

Video URL:

Figure 9: Video upload page on the website



https://up-fix.github.io/

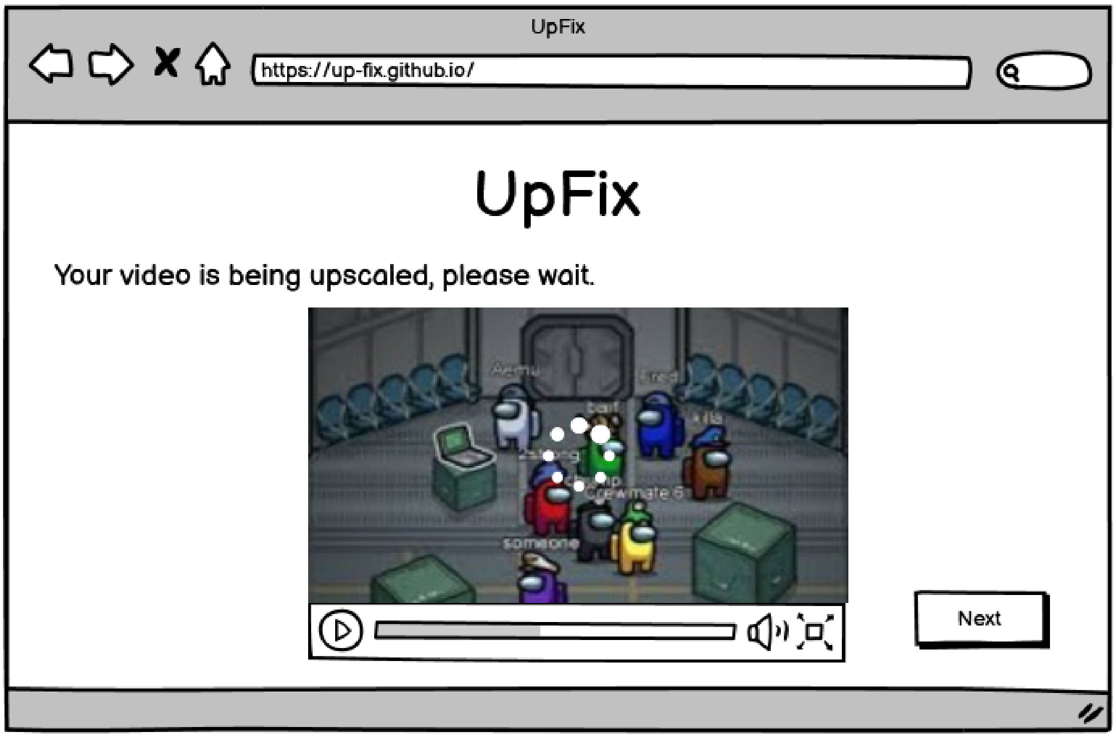
Select the game:

Among Us

Select the upscaling algorithm:

○ CNN ○ RNN ○ GAN

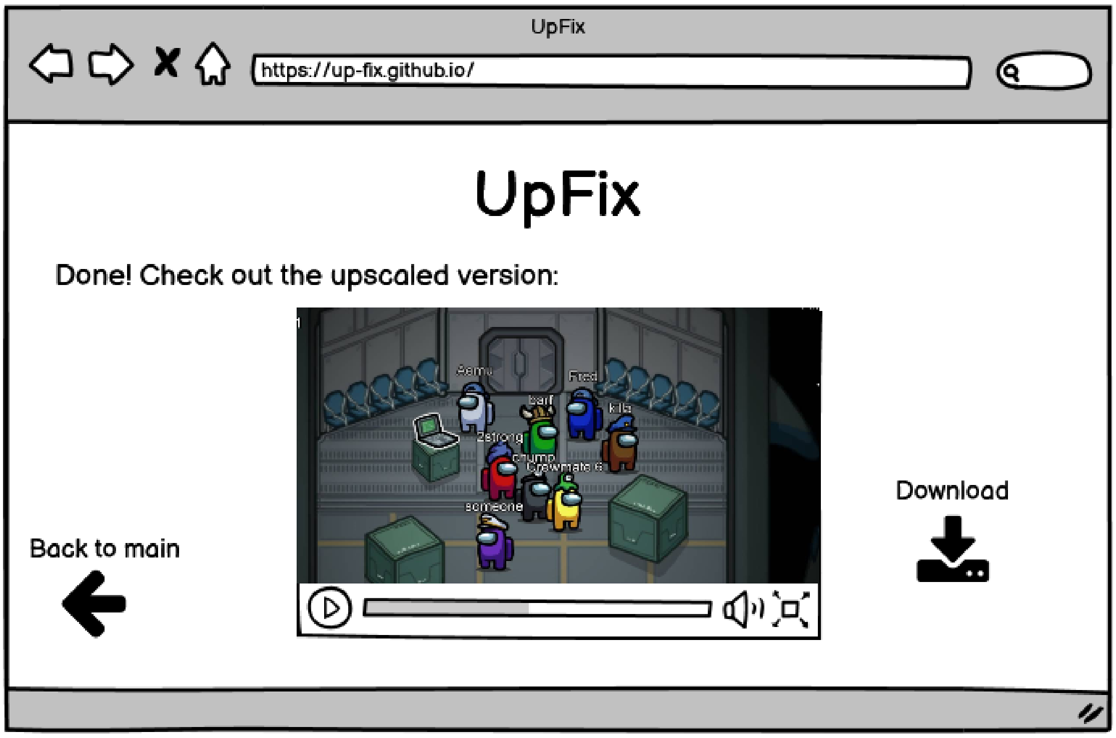
Figure 10: Selecting the game and the upscaling algorithm on the website



UpFix

Your video is being upscaled, please wait.

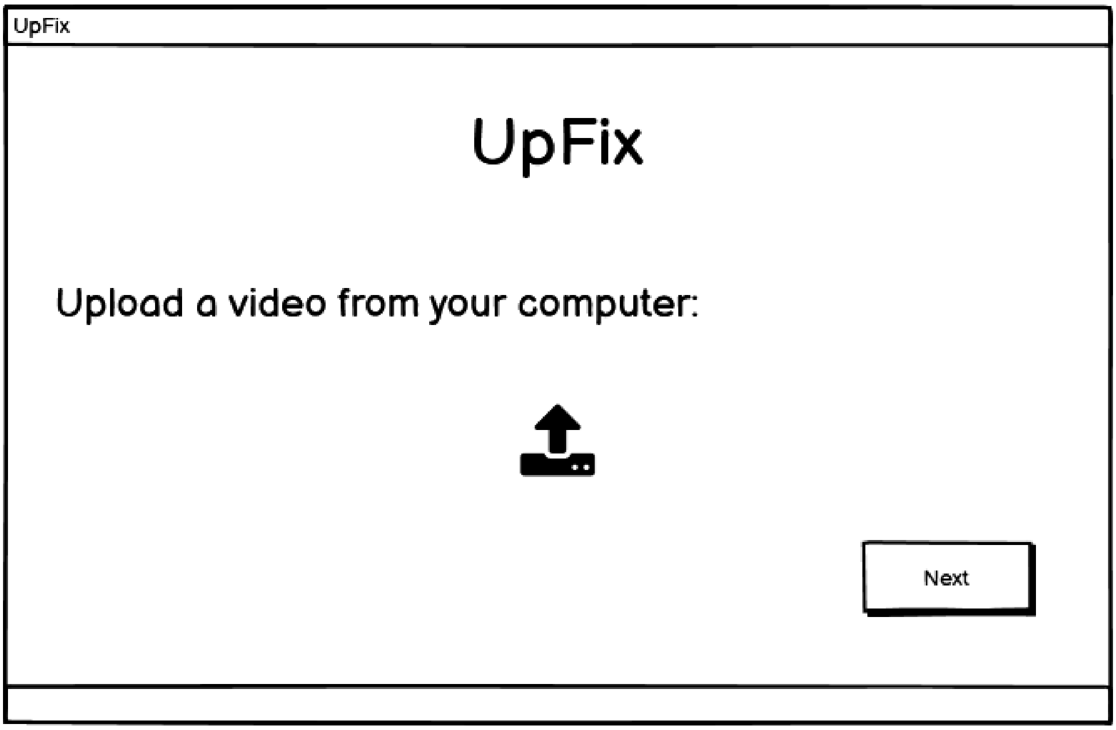
Figure 11: Loading page on the website



UpFix

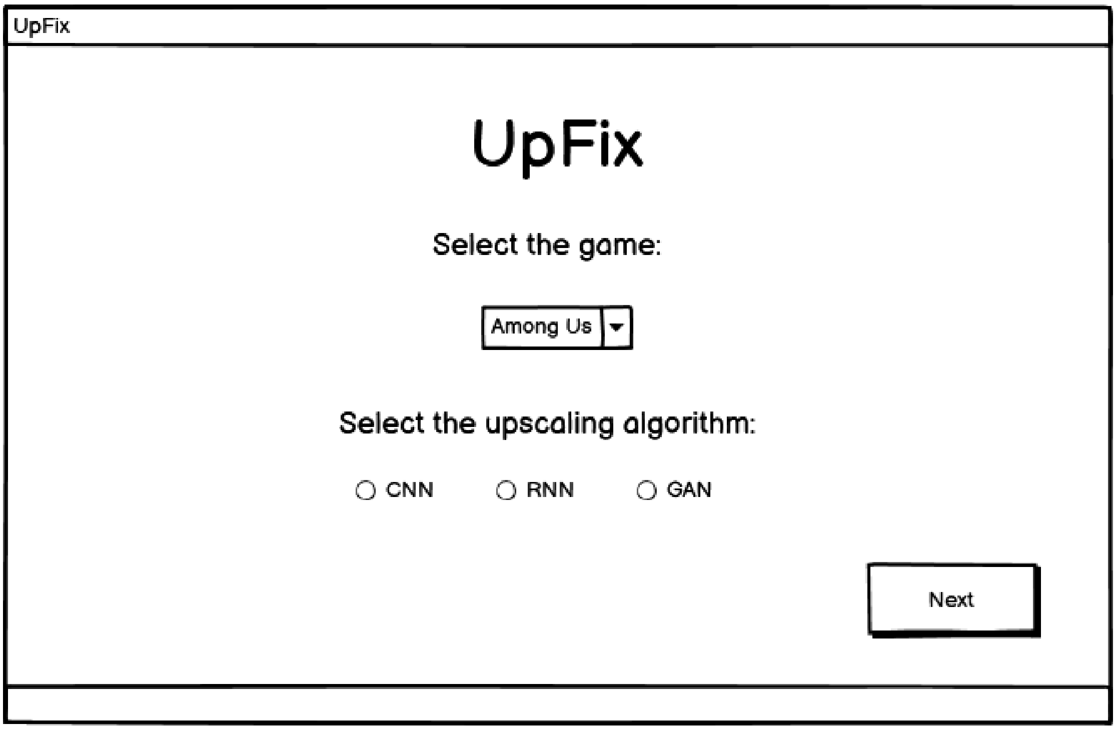
Done! Check out the upscaled version:

Figure 12: Download video page on the website



Upload a video from your computer:

Figure 13: Upload video page on the desktop app



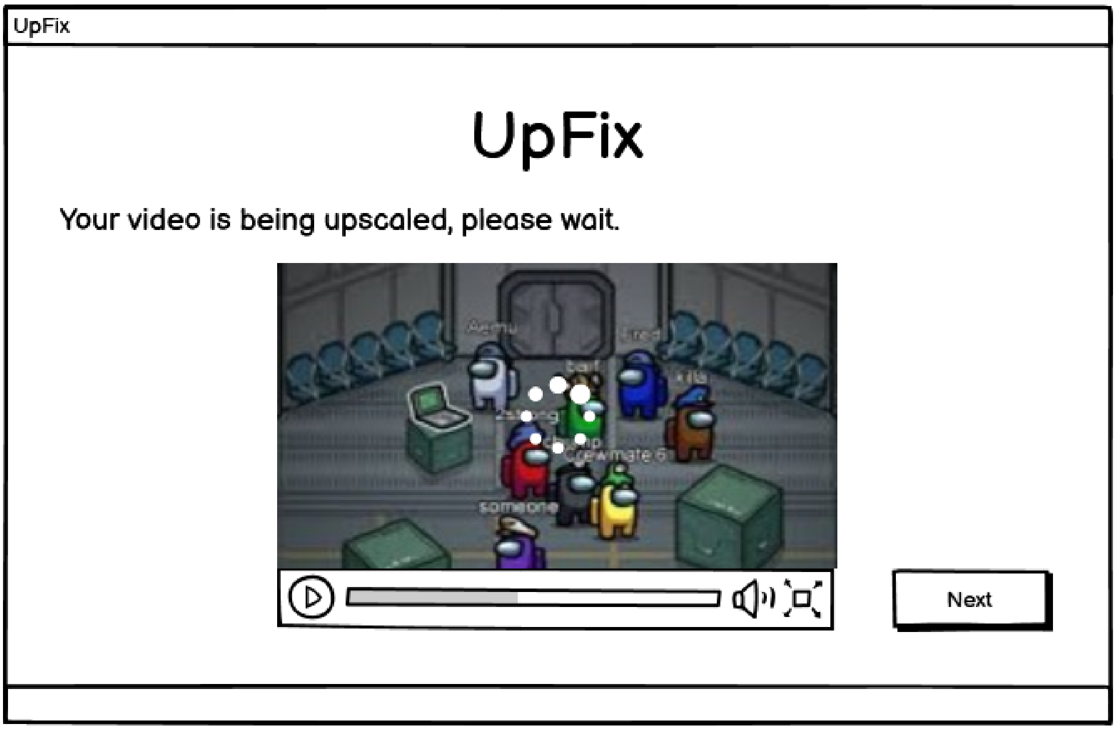
Select the game:

Among Us ▾

Select the upscaling algorithm:

○ CNN ○ RNN ○ GAN

Figure 14: Selecting the game and the upscaling algorithm on the desktop app

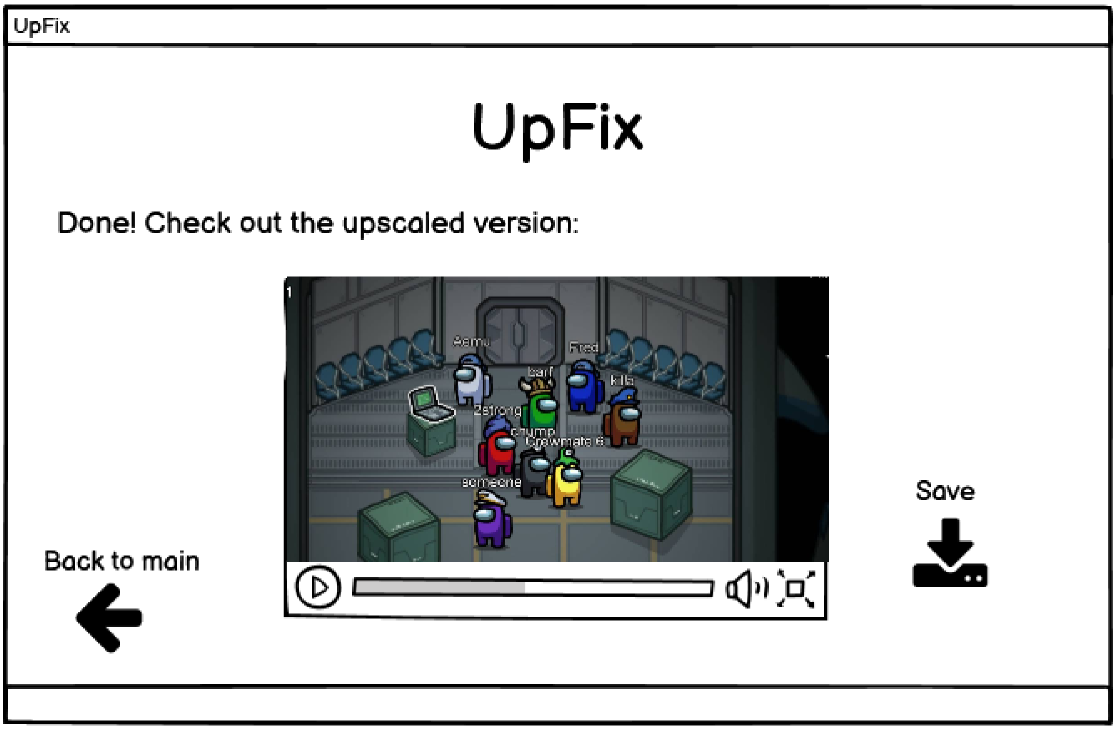


UpFix

UpFix

Your video is being upscaled, please wait.

Figure 15: Loading page on the desktop app



UpFix

UpFix

Done! Check out the upscaled version:

Figure 16: Video download page on the desktop app

3. Other Analysis Elements

3.1. Consideration of Various Factors in Engineering Design

Various factors that may affect the design and implementation of the application has been thought of and are described as follows:

3.1.1. Public Health

After evaluation, the current design in place for the application (both web and desktop) does not pose any danger to public health any more than other popular video platforms such as Twitch or YouTube.

3.1.2. Public Safety

Upfix does not hold any personal information about its users, nor does it sell to or share any information with third party applications. Since no information about users is kept or displayed, it is out of reach from malicious parties as well.

3.1.3. Public Welfare

Upfix does not affect public welfare, therefore it can not pose a threat to it.

3.1.4. Global, Cultural & Social Factors

Because Upfix's user experience consists only of interaction with the application itself rather than interaction with other users who use the application, it is not modified heavily by global, cultural, and social factors. The text that appears on the application's user interface will not contain any cultural insensitivity or hate-speech, and it will be inclusive to all races and genders. The application will have its user interface in English as it is the current lingua franca.

3.1.5. Environmental Factors

Upfix will be a software application, so it will not be affected by environmental factors.

3.1.6. Economic Factors

Upfix will be completely free of charge to use and download.

3.2. Risks and Alternatives

The main feature of Upfix is that it upscales the image quality of gaming videos. The application will do this via a neural network that is trained with game videos. In order to achieve this, a high number of such videos are needed to be fed into the network, and the biggest challenge in this is that there is no preset collection of videos (data); we will have to find and use the data by hand. The lack of enough data may then result in a biased model, which could potentially upscale given gaming videos wrongly. Thus, if the dataset found by hand is not sufficient, in order to find more data to feed into the network we may use web crawlers.

The web application will upscale videos on the server, which may cause delay while sending, processing and receiving data. This can be solved by using the desktop application instead, since it does not use servers.

The desktop application uses the resources of the user's own computer rather than a server which relies on the specifications of that specific computer, which may, in extreme cases, not be sufficient to upscale certain videos. The user will then have to use the web application instead, but in order to not face this problem often, an infographic with the minimum amount of specs needed in a computer to use the desktop application will be provided before users can choose to download the desktop application.

Table 1: Risks

	Likelihood	Effect on the project	Plan B
Risk of not collecting enough data	3	A biased model, incorrect upscaling of videos	Use web crawlers to access more data
Risk of delay in server (web application)	2	Potential customer dissatisfaction	Use desktop application instead
Risk of insufficient computer power (desktop application)	6	Potential customer dissatisfaction	Use web application instead

3.3. Project Plan

Table 2: Project Plan Overview

WP #	**Work Package Title**	**Leader**	**Members Involved**
WP 1	Analysis	Hazal Aksu	Everybody
WP 2	Design	Meryem Efe	Everybody
WP 3	Development of Application	Rabia Nur Önal	Özge Yaşayan Hazal Aksu
WP 4	Development of Models	Hamza Pehlivan	Meryem Efe
WP 5	Testing	Özge Yaşayan	Everybody

Table 3: Project Plan - WP 1

WP 1: Analysis			
Start Date: 12.10.2020		**End Date:** 21.11.2020	
Leader	Hazal Aksu	**Members Involved**	Hamza Pehlivan Meryem Efe Özge Yaşayan Rabi Nur Önal
Objectives: The aim of this work package is to determine the functional, non-functional and pseudo requirements of the Project as well as designing the system models and consider other analysis elements. The work package includes the concluding requirements, system models and other analysis elements of the project. At the end of the work package deadline, we are planning to present six deliverables which are the analysis report and the system models: Scenarios, Use Case Models, Dynamic Models and User Interface Mockups.			
Tasks: ● **Task 1.1 Analysis of Project:** Analyze the system that is to be developed. Propose the requirements of the system. ● **Task 1.2 System Models:** Come up with the system models. That includes scenarios, use case models, object and class models, dynamic models and user interface mockups. ● **Task 1.3 Further Analysis Elements:** Provide the remaining analysis elements such as risk and alternatives, Project plan etc. Address all relevant issues.			
Deliverables: ● **D1.1:** Analysis Report ● **D1.2:** Scenarios ● **D1.3:** Use Case Models ● **D1.4:** Object and Class Models ● **D1.5:** Dynamic Models ● **D1.6:** User Interface Mockups			

Table 4: Project Plan - WP 2

<table>
<tr><td colspan="4"><b>WP 2:</b> Design</td></tr>
<tr><td colspan="2"><b>Start Date:</b> 23.11.2020</td><td colspan="2"><b>End Date:</b> 07.02.2021</td></tr>
<tr><td><b>Leader</b></td><td>Meryem Efe</td><td><b>Members Involved</b></td><td>Hamza Pehlivan
Hazal Aksu
Özge Yaşayan
Rabi Nur Önal</td></tr>
<tr><td colspan="4"><b>Objectives:</b> The aim of this work package is to determine the requirements of the project and to make detailed design of the project. The work package includes interface design, high level design and low level design of the project. At the end of the work package deadline, we are planning to present two deliverables which are High Level and Low Level Design Reports.</td></tr>
<tr><td colspan="4"><b>Tasks:</b>
● <b>Task 2.1 Evaluation of Analysis:</b> Evaluate the analysis of the projects, make required regulations according to feedback
● <b>Task 2.2 User Interface Design:</b> Design the user interface of the application for the users
● <b>Task 2.3 High Level Design:</b> Define the project’s design goals, decompose the project into subsystems
● <b>Task 2.4 Low Level Design:</b> Explain the validity of the design principles in detail</td></tr>
<tr><td colspan="4"><b>Deliverables:</b>
● <b>D2.1:</b> High Level Design Report
● <b>D2.2:</b> Low Level Design Report</td></tr>
</table>

Table 5: Project Plan - WP 3

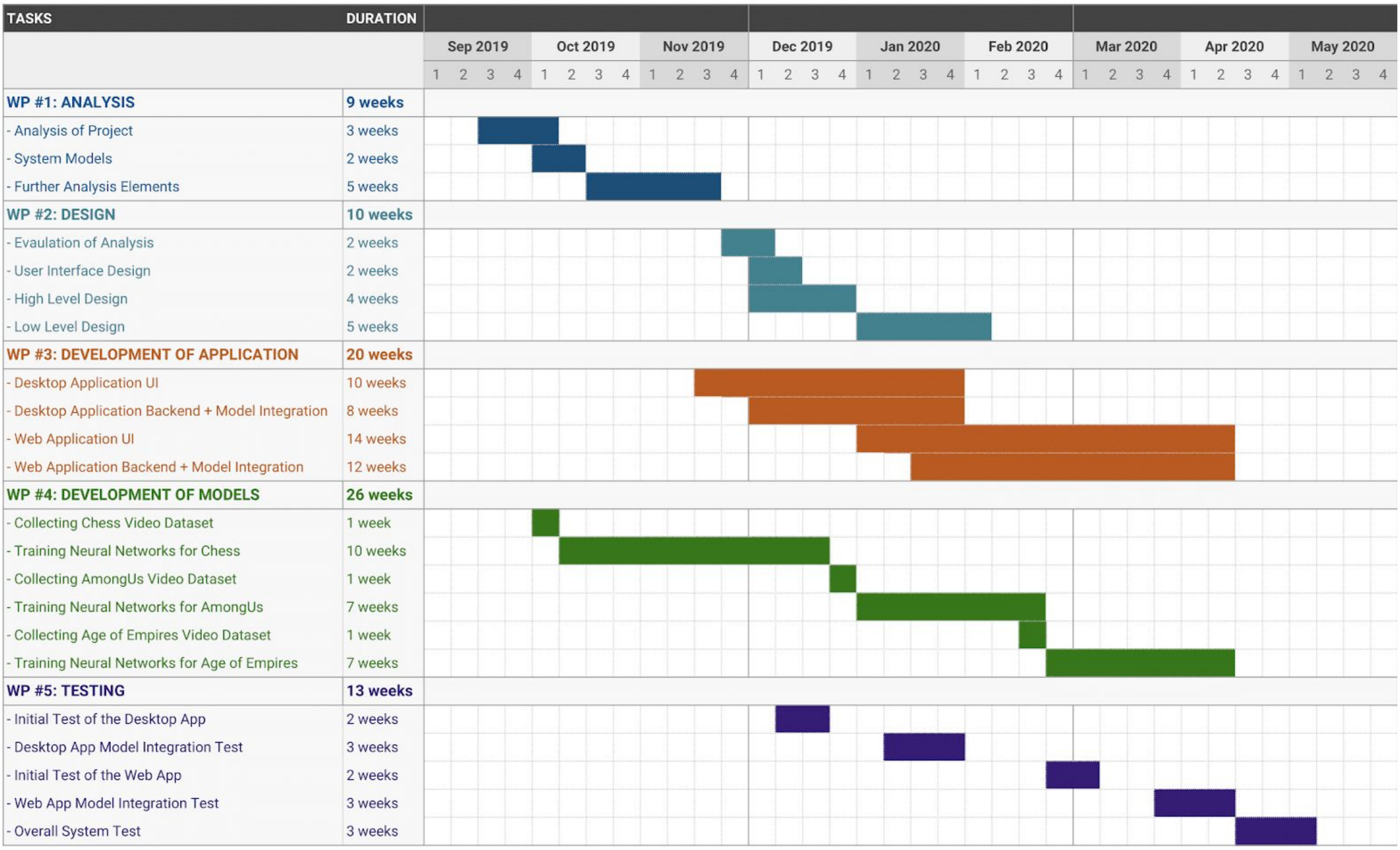
WP 3: Development of Application			
Start Date: 16.11.2020		**End Date:** 18.04.2021	
Leader	Rabia Nur Önal	**Members Involved**	Hazal Aksu Özge Yaşayan
Objectives: The aim of this work package is to implement the web and desktop versions of UpFix. The tasks to be completed in this work package are implementing the UI and backend of the desktop application and integrating the machine learning models as well as completing the implementation of the web application together with integration of models. By the deadline we plan to have an executable desktop application and the UpFix web application.			
Tasks: ● **Task 2.1 Desktop Application UI:** Implement the UI for the desktop application with JavaFX. ● **Task 2.2 Desktop Application Backend + Model Integration:** Implement the backend of the desktop application with Java and integrate the machine learning models. ● **Task 2.3 Web Application UI:** Implement the UI for the desktop application with ReactJS. ● **Task 2.4 Web Application Backend + Model Integration:** Implement the backend of the desktop application with Java Spring and integrate the machine learning models.			
Deliverables: ● **D2.1:** Desktop Application Executable ● **D2.2:** Web Application			

Table 6: Project Plan - WP 4

WP 4: Development of Models			
Start Date: 01.10.2020		**End Date:** 18.04.2021	
Leader	Hamza Pehlivan	**Members Involved**	Meryem Efe
Objectives: The aim of this work package is to provide necessary neural network models. The work package includes both data collection and training parts. There will be three deliverables for this work package.			
Tasks: **Task 4.1 Collecting Chess Video Dataset:** Collect high resolution chess videos. **Task 4.2 Training Neural Networks for Chess:** Train three neural networks for chess using the collected data. **Task 4.3 Collecting AmongUs Video Dataset:** Collect high resolution AmongUs videos. **Task 4.4 Training Neural Networks for AmongUs:** Train three neural networks for AmongUs using the collected data. **Task 4.5 Collecting Age of Empires Video Dataset:** Collect high resolution Age of Empires videos. **Task 4.6 Training Neural Networks for Age of Empires:** Train three neural networks for Age of Empires using the collected data.			
Deliverables: **D4.1:** *Chess Models* **D4.2:** *AmongUs Models* **D4.3:** *Age of Empires Models*			

Table 7: Project Plan - WP 5

WP 5: Testing			
Start Date: 07.12.2020		**End Date:** 09.05.2021	
Leader	Özge Yaşayan	**Members Involved**	Hazal Aksu Rabia Nur Önal Hamza Pehlivan Meryem Efe
Objectives: The aim of this work package is to ensure that all parts of the system behave as expected and provide the desired functionality as a whole with little to no inconvenience for the users. After this process is completed, the product will be ready to be deployed.			
Tasks: **Task 5.1 Initial Test of the Desktop App:** Test the basic UI elements of the desktop app, as well as uploading local videos & saving videos locally. **Task 5.2 Desktop App Model Integration Test:** Test whether the model works with the desktop app and how well it upscales various videos. **Task 5.3 Initial Test of the Web App:** Test the basic UI elements of the web app, as well as uploading videos to & downloading videos from the web server. **Task 5.4 Web App Model Integration Test:** Test whether the model works with the web app and how well it upscales various videos. **Task 5.5 Overall System Test:** Comprehensive testing of every part of the system, using videos of different games that differ in length as well as quality.			
Deliverables: **D5.1:** The latest versions of the web application and the desktop application will be delivered after this step.			



TASKS	DURATION
WP #1: ANALYSIS	**9 weeks**
- Analysis of Project	3 weeks
- System Models	2 weeks
- Further Analysis Elements	5 weeks
WP #2: DESIGN	**10 weeks**
- Evaulation of Analysis	2 weeks
- User Interface Design	2 weeks
- High Level Design	4 weeks
- Low Level Design	5 weeks
WP #3: DEVELOPMENT OF APPLICATION	**20 weeks**
- Desktop Application UI	10 weeks
- Desktop Application Backend + Model Integration	8 weeks
- Web Application UI	14 weeks
- Web Application Backend + Model Integration	12 weeks
WP #4: DEVELOPMENT OF MODELS	**26 weeks**
- Collecting Chess Video Dataset	1 week
- Training Neural Networks for Chess	10 weeks
- Collecting AmongUs Video Dataset	1 week
- Training Neural Networks for AmongUs	7 weeks
- Collecting Age of Empires Video Dataset	1 week
- Training Neural Networks for Age of Empires	7 weeks
WP #5: TESTING	**13 weeks**
- Initial Test of the Desktop App	2 weeks
- Desktop App Model Integration Test	3 weeks
- Initial Test of the Web App	2 weeks
- Web App Model Integration Test	3 weeks
- Overall System Test	3 weeks

Figure 17: Gantt Chart for the Project Schedule

3.4. Ensuring Proper Teamwork

In order to ensure proper teamwork, we have divided the project into work packages and we have chosen each team member as a leader of one work package. The leaders determined the tasks in the work page which they are responsible for. Additionally, we divided the tasks in equal workload. In this way, we could make a proper and equal work-share. We meet every monday in order to follow our progress and distribute new tasks.

Furthermore, we follow a collaborative approach. Even though every team member has a separate task, if one finishes their individual part, s/he will help the other team members to do their parts in order to ensure the equality of teamwork and ensure a collaborative work environment.

The project has a private GitHub repository to keep the progress of the project. Additionally, there are two separate folders in Google Drive for this purpose. One of them is used for the reports and tracking the weekly distribution of tasks. The other one includes Google Colab files to train neural network models. Through these repositories and files, we can track our progress with ease and they can be shared with the supervisors and the jury members.

3.5. Ethics and Professional Responsibilities

In order to not violate the privacy of our users, the application (both web and desktop) will not hold any personal information about its users nor sell such information to and/or share it with any third-party applications.

All data used by and in this project, including the intended neural network, will comply with KVKK and GDPR regulations to comply with and not violate any data privacy laws enforced by both Turkey and the European Union.

All sources, source libraries and third-party software used in this project will be open-source ones and credited accordingly in place.

3.6. Planning for New Knowledge and Learning Strategies

The project consists of two main parts which are the development of the web / desktop application and the training of neural network models. We divided our team into these two parts according to the team members' interests and experiences.

Since two of us (Hamza and Meryem) already have knowledge and experience in Deep Learning, they are responsible for the training of the neural network models. Even though they are familiar with CNN (Convolutional Neural Network) and RNN (Recurrent Neural Network), they need to learn GAN (Generative Adversarial Networks) which is the third model they will train. Therefore, they used online learning tools and started to follow the GAN course offered by deeplearning.ai in Coursera [8].

The other team members (Rabia, Özge and Hazal) are responsible for the development process of the application. Even though they have varying experiences in web and desktop applications with different tools, they all have to use the same tools for the project. Therefore, they decided to use Spring and React frameworks. In this part of the development process, they benefit mostly from online tutorials in YouTube and Udemy. Also, in general, we will use StackOverFlow when we face an error.

4. References

[1] “Spring”, Available: https://spring.io (accessed Nov. 21, 2020).

[2] “React – A JavaScript library for building user interfaces”, Available: https://reactjs.org/ (accessed Nov. 21, 2020).

[3] “Build software better, together”, Available: https://github.com/ (accessed Nov. 21, 2020).

[4] “TensorFlow”, Available: https://www.tensorflow.org/ (accessed Nov. 21, 2020).

[5] “PyTorch”, Available: https://www.pytorch.org/ (accessed Nov. 21, 2020).

[6] “Colaboratory'ye Hoş Geldiniz”, Available: https://colab.research.google.com/ (accessed Nov. 21, 2020).

[7] “OpenCV”, Available: https://opencv.org/ (accessed Nov. 21, 2020).

[8] “Generative Adversarial Networks (GANs),” Coursera. [Online]. Available: https://www.coursera.org/specializations/generative-adversarial-networks-gans. [Accessed: 21-Nov-2020].