



BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

**SUPPORT TOOLS FOR VERIFYING HUMAN ABILITY
TO DETECT DEEPFAKES**

PODPŮRNÉ NÁSTROJE PRO OVĚŘOVÁNÍ LIDSKÉ SCHOPNOSTI ROZPOZNÁVAT DEEPFAKES

BACHELOR'S THESIS

BAKALÁŘSKÁ PRÁCE

AUTHOR

AUTOR PRÁCE

PATRIK POTANČOK

SUPERVISOR

VEDOUČÍ PRÁCE

Ing. ANTON FIRIC

BRNO 2024

Bachelor's Thesis Assignment



153356

Institut: Department of Intelligent Systems (DITS)
Student: **Potančok Patrik**
Programme: Information Technology
Title: **Support Tools for Verifying Human Ability to Detect Deepfakes**
Category: Security
Academic year: 2023/24

Assignment:

1. Learn the procedures and technologies for creating web applications.
2. Learn about creating user interfaces for web applications. Focus on best practices and user experience.
3. Design a custom web application usable for research on the human ability to recognize synthetic speech. The application must allow you to insert recordings, set the order in which recordings are played, return to a running experiment, select the number of recordings for a single step of the experiment, and support multiple languages.
4. Implement the proposed application using selected web development technologies.
5. Test the functionality and user-friendliness of the application. Evaluate the achievement of the desired implementation goals.

Literature:

- WELLING, Luke; THOMSON, Laura. *PHP and MySQL Web development*. Sams Publishing, 2003.
- CASTRO, Elizabeth; HYSLOP, Bruce. *HTML5 a CSS3*. Computer Press, 2015.
- SOTNIK, Svitlana; MANAKOV, Volodymyr; LYASHENKO, Vyacheslav. *Overview: PHP and MySQL Features for Creating Modern Web Projects*. 2023.

Requirements for the semestral defence:
1 - 3

Detailed formal requirements can be found at <https://www.fit.vut.cz/study/theses/>

Supervisor: **Firc Anton, Ing.**
Head of Department: Hanáček Petr, doc. Dr. Ing.
Beginning of work: 1.11.2023
Submission deadline: 9.5.2024
Approval date: 6.11.2023

Abstract

The aim of this thesis is to create a web application using PHP and MySQL, that will test the human ability to detect deepfake recordings while collecting their data like date of birth, native language, proficiency in other languages, how many times and how long did they listen to a recording and a number of correct answers. This application includes management of the recordings and the users and the ability to export user data in CSV format. The application was implemented using Laravel, Vue.js and MySQL.

Abstrakt

Cielom tejto práce je vytvoriť webovú aplikáciu s použitím PHP a MySQL, ktorá bude testovať schopnosť ľudí rozpoznať deepfake nahrávky a pritom zbierať ich údaje ako dátum narodenia, rodný jazyk, znalosti ďalších jazykov, koľkokrát a ako dlho počúvali nahrávku a počet správnych odpovedí. Aplikácia zahŕňa správu nahrávok a používateľov, a možnosť exportu údajov používateľov vo formáte CSV. Aplikácia bola implementovaná s použitím Laravel, Vue.js a MySQL.

Keywords

deepfake, synthetic speech, web application, experiment, Laravel, user interface, data analysis, PHP, MySQL

Klíčová slova

deepfake, syntetická reč, webová aplikácia, experiment, Laravel, užívateľské rozhranie, analýza dát, PHP, MySQL

Reference

POTANČOK, Patrik. *Support Tools For Verifying Human Ability To Detect Deepfakes*. Brno, 2024. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Anton Firc

Rozšířený abstrakt

Deepfake videá a zvukové nahrávky sa rýchlo rozvíjajú a začínajú mať vplyv na naše každodenné životy. Stávajú sa nerozoznatelnými od pravdy, a preto predstavujú hrozbu pre demokraciu a ľudskú spoločnosť. Schopnosť rozlišovať medzi tým, čo je skutočné a čo falošné, sa v dnešnom svete stáva takmer nevyhnutnosťou pre fungujúcu spoločnosť. Každý deň nás bombardujú tony informácií, čo robí takmer nemožné overiť každý kúsok informácie. Deepfake toto zhoršuje tým, že ľudia cítia, že informáciám nemožno veriť. Navyše ľudia môžu považovať pravdivú informáciu za falošnú, jednoducho preto, že jej nechcú uveriť. Inými slovami, najväčšia hrozba nie je, že ľudia budú oklamaní deepfakes, ale že budú považovať všetko za podvod.

Pokroky v strojovom učení a umelej inteligencii, najmä vo vizuálnej analýze a spracovaní reči, ulahčili vytvorenie veľmi realistických falšovaných videí a zvukových nahrávok, ktoré boli zneužitú ľuďmi so zlými úmyslami. Preto je nevyhnutné informovať a vzdelávať ľudí o deepfakes. Veľa ľudí dnes stále nepozná slovo deepfake. To vyvoláva otázku: Ktoré skupiny ľudí sú najviac ovplyvnené a ohrozené? Na odpoveď na túto otázku budeme potrebovať experimenty, z ktorých budeme môcť získať údaje na určenie tejto skupiny.

Cieľom tejto práce je vytvoriť webovú aplikáciu s použitím PHP a MySQL, ktorá bude testovať schopnosť ľudí rozpoznať deepfake nahrávky a pritom zbierať ich údaje ako dátum narodenia, rodný jazyk, znalosti ďalších jazykov, koľkokrát a ako dlho počúvali nahrávku a počet správnych odpovedí, čo pomôže vytvoriť podrobný obraz o tom, kto je najviac náchylný na technológie deepfake. Aplikácia zahŕňa správu nahrávok a užívateľov, a poskytuje možnosť exportovania dát vo formáte CSV. Táto aplikácia nielen umožní používateľom trénovať rozpoznávanie deepfakes, ale tiež poskytne cenné informácie pre budúci výskum v tejto oblasti. Neskôr môže byť pridaná do väčšieho projektu na vytvorenie portálu pre vzdelávanie ľudí o deepfakes.

Implementácia aplikácie prebiehala s využitím technológií Laravel, Vue.js a MySQL, pričom dôraz bol kladený na vytvorenie užívateľsky prívetivej aplikácie, ktoré bude ponúkať jednoduchú navigáciu a interakciu s aplikáciou. Pomocou Vue.js a ovládačov sa v užívateľskom rozhraní zobrazujú zvukové nahrávky a užívatelia musia rozhodnúť či sa jedná o skutočnú nahrávku alebo syntetickú. Všetky užívateľské interakcie sú asynchrónne spracované na serveri pomocou Laravel, čo zaisťuje rýchlu odozvu aplikácie. Backend aplikácie je navrhnutý tak, aby efektívne spracovával získané dáta a zabezpečoval ich konsistenciu a bezpečnosť. Kontroléry Laravel spravujú logiku aplikácie týkajúcu sa správy používateľov, správy nahrávok a experimentov. V rámci testovania boli používatelia požiadaní, aby vykonali sériu úloh a poskytli spätnú väzbu na používateľskú prívetivosť, intuitívnosť a celkovú funkčnosť aplikácie. Okrem užívateľského testovania sa realizovalo aj testovanie funkčnosti aplikácie pomocou nástrojov Insomnia, ktorá dokáže testovať ovládače, a DataGripu, pomocou ktorého prebiehala analýza databázy. Tieto testy overovali správnu funkcionálnosť všetkých komponentov systému, ako aj správnosť a bezpečnosť uložených dát.

Výsledkom práce je plne funkčná webová aplikácia, ktorá umožňuje používateľom overiť svoje schopnosti rozpoznávania deepfake nahrávok a zároveň poskytuje nástroje pre administrátorov na správu experimentov a analýzu získaných dát. Práca poukazuje na potrebu ďalšieho výskumu v oblasti detekcie deepfake materiálov a zdokonalenia technológií na ich rozpoznávanie. Hlavné príspevky tejto práce sú:

- **Rozvoj povedomia o deepfake:** Zvyšovanie verejného povedomia a pripravenosti proti deepfake prostredníctvom interaktívnej webovej platformy, ktorá vzdeláva používateľov v identifikácii syntetických nahrávok.

- **Získavanie dát:** Zbieraním údajov od používateľov počas interakcie s deepfakes poskytuje aplikácia prehľady o demografických zraniteľnostiach a odolnosti voči technológiám deepfake.
- **Škálovateľnosť pre vzdelávanie:** Návrh a nasadenie systému, ktorý možno rozšíriť alebo upraviť pre širšie vzdelávacie účely nad rámec detekcie deepfake.

Support Tools For Verifying Human Ability To Detect Deepfakes

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Mr. Ing. Anton Firc. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Patrik Potančok
May 6, 2024

Acknowledgements

I would like to sincerely thank my supervisor, Ing. Anton Firc, for his help, guidance and advice throughout this project. I am also thankful to the whole Security@FIT research group, which also helped me throughout the process of creating this thesis. Lastly, my thanks go to my family who supported me every step of the way.

Contents

1	Introduction	5
2	What are Deepfakes?	7
2.1	Creation of deepfakes	7
2.1.1	Architectures	7
2.1.2	Visual deepfakes	9
2.1.3	Voice deepfakes	11
2.2	Examples of Deepfakes	12
2.3	Impacts of deepfakes on society	16
2.3.1	Positive Impacts	16
2.3.2	Negative Impacts	17
2.4	How to detect deepfakes?	19
2.4.1	Naturally	19
2.4.2	Using computers	21
3	Web Development	23
3.1	Front-end	23
3.1.1	HTML	23
3.1.2	CSS	24
3.1.3	JavaScript	24
3.1.4	Best Practices for Creating a User Interface	25
3.2	Back-end	27
3.2.1	PHP	27
3.2.2	MySQL	27
3.3	Frameworks	28
3.3.1	Server-side Frameworks	28
3.3.2	Client-side Frameworks	28
3.3.3	Choosing Laravel and Vue.js for Project Development	29
3.4	Development Process of Web Applications	29
4	App Design	33
4.1	Requirements Analysis	33
4.2	Architecture Design	33
4.2.1	Client-Server Architecture	33
4.3	User Interface	35
4.3.1	User part	36
4.3.2	Admin part	41
4.4	Storing data	44

4.4.1	User table	45
4.4.2	User Progress Table	45
4.4.3	Recordings Table	46
4.4.4	Experiment setting table	47
5	Implementation	48
5.1	Front-end Development with Vue.js	48
5.1.1	Front-end Routing with Vue.js	49
5.1.2	Employed Libraries and Frameworks	50
5.1.3	Use of Best Practices for User Interface	51
5.2	Back-end Development with Laravel	53
5.2.1	Controllers	54
5.2.2	Application programming interface	56
5.2.3	Models	57
5.2.4	Migrations	57
5.3	Implementation Output	58
6	Testing	61
6.1	Database and controllers testing	61
6.1.1	Insomnia	61
6.1.2	Database testing with DataGrip	61
6.2	User-based front-end testing	62
6.2.1	Evaluation of user testing	63
6.3	Test experiment	64
7	Conclusion	67
	Bibliography	68

List of Figures

1.1	A chart of the percentage of people who knew what a deepfake is. From: ([1].)	5
2.1	A simplified example of how ED networks create deepfakes. From: ([32]) . .	8
2.2	Simplified architecture of Generative Adversarial Networks. From: ([9]) . .	8
2.3	Architecture of convolutional neural network. From: ([41])	9
2.4	Architecture of recurrent neural network. From: ([13])	10
2.5	An example of the face swapping process. From: ([38])	10
2.6	Illustration of the overall architecture of the model ICface. From: ([50]) . .	11
2.7	The interconnected modules of a text-to-speech system. From: ([61])	12
2.8	The basic structure of voice conversion modules. From: ([46])	12
2.9	David Beckham deepfake in the Zero Malaria Britain campaign. From: ([60])	13
2.10	Comparison between the original VFX and Shamook’s fan-made deepfake in The Mandalorian. From: ([44])	13
2.11	Obama deepfake by Jordan Peele, demonstrating the realistic synthesis of speech and expressions. From: ([10])	14
2.12	Satirical deepfake of former President Donald Trump, showcasing the use of deepfakes in political satire. From: ([15])	14
2.13	A screenshot from the Deepfake Roundtable by Collider. From: ([16]) . . .	15
2.14	Before and after of the Tom Cruise Deepfake by Deep Cruise on TikTok. From: ([52])	15
2.15	A deepfake of Queen Elizabeth II. From: ([12])	16
2.16	A deepfake of President Zelensky compared to his real photo. From: ([51]) .	19
2.17	An example of how we can spot a deepfake. From: ([47])	20
2.18	An overview of computer-based deepfake detection methods. Adapted from: ([39])	22
3.1	Comparison of requests per second of Laravel, Symfony and CodeIgniter. From: [31]	29
3.2	Comparison of the response time of Laravel, Symfony and CodeIgniter. From: [31]	30
3.3	Comparison of memory usage of Laravel, Symfony and CodeIgniter. From: [31]	30
3.4	Comparison of the number of files of Laravel, Symfony and CodeIgniter. From: [31]	31
3.5	Web Development Life Cycle. From: ([21])	32
4.1	A use case diagram for this application	34
4.2	Two-Tier Architecture	35
4.3	Three-Tier Architecture	35

4.4	The initial screen design prompts the user to begin the deepfake detection experiment.	36
4.5	The user input form for collecting demographic information before starting the experiment.	37
4.6	The screen for user login using the token.	37
4.7	Consent form presented to users, requiring agreement before participating in the experiment.	38
4.8	Experiment interface where the user determines if a recording is a deepfake or real.	38
4.9	Double experiment interface where the user selects the real recording. . . .	39
4.10	Practice mode interface	39
4.11	End of the experiment interface.	40
4.12	Interface for the user account with options to request results or account deletion.	40
4.13	Options for the user to continue, start a new experiment or edit their data.	41
4.14	Interface for admin login.	41
4.15	Admin panel for managing user data and experiment records.	42
4.16	Interface for the admin to add new recordings to the experiment database. .	43
4.17	Interface for the admin to add a new recording to the experiment database.	43
4.18	Multiple recordings upload page.	44
4.19	An ER diagram for this application	44
5.1	Screenshot of the home screen of the application.	52
5.2	Implemented experiment screen with a single recording.	52
5.3	Implemented admin screen.	53
5.4	Upload screen from the application.	53
5.5	Communication between the components of the application.	60
6.1	The simplified output of the test experiment showing the incorrect and correct answers for the most problematic recordings of the test experiment. . .	65
6.2	This graph showcases the percentage of incorrect answers for different Czech language proficiencies.	65
6.3	A graph that shows which recordings were the most replayed.	66

Chapter 1

Introduction

Deepfakes are rapidly emerging, and they are starting to impact our day-to-day lives. They are becoming indistinguishable from the truth, and they therefore pose a moderate threat to democracy, public discourse, and human society [56]. The importance of being able to differentiate between what is real and what is fake in today’s world is becoming almost a necessity for a functional society. We are bombarded with tons of information every day, which makes it almost impossible to fact-check every piece of information. Deepfakes make this situation even more dire by making people feel that information cannot be trusted. Furthermore, people may dismiss a piece of genuine information as fake, simply because they have become entrenched in the notion that anything they do not want to believe is fake. In other words, the greatest threat is not that people may be fooled by deepfakes but that they will regard everything as deception [56].

Advances in machine learning and artificial intelligence, especially in computer vision and speech processing, have made it much easier to create very realistic fake videos and audio clips, which have been misused by people with bad intentions. Because of this, it is necessary to inform and educate people about deepfakes. From this 2022 chart 1.1, it can be seen that not even a third of the respondents knew what a deepfake is. This raises the question: Which groups of people are most affected? To answer this, we will need experiments from which we can gather data to determine this group.

The goal of this work is to develop a web application that is easily accessible to the general public and that will enable us to identify key groups. The application will collect

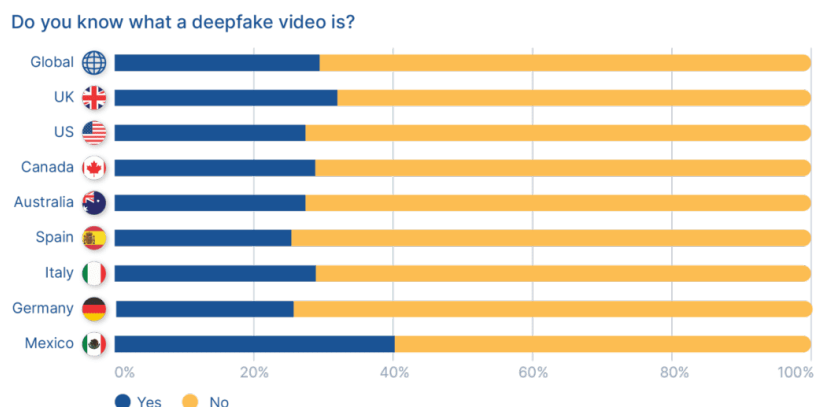


Figure 1.1: A chart of the percentage of people who knew what a deepfake is. From: ([1].)

anonymous user data, such as age, occupation, and nationality, to help create a detailed picture of who is most vulnerable to deepfake technologies. This application will not only allow users to train in recognising deepfakes but will also provide valuable information for future research in this area. It can later be added to a larger project to create a portal for educating people about deepfakes.

As mentioned, the application must be easy to use and support multiple languages to gather as much data as possible (large font, automatic redirection). It will also be possible to return to an unfinished experiment. It must allow the insertion and setting of the order of recordings to create the same testing environment for users. Finally, downloading data in an easily processable format, such as CSV. The implementation will use PHP and MySQL, along with the Laravel framework. The main contributions of this thesis can be summarised as follows:

- **Advancing Deepfake Awareness:** Elevating public understanding and readiness against deepfakes through an interactive web platform that educates users on the identification of digital falsifications.
- **Data-Driven Insights:** By collecting user data during the interaction with deepfake content, the project provides insights into demographic vulnerabilities and resistance to deepfake technologies.
- **Scalable Educational Framework:** The design and deployment of a system that can be expanded or modified for broader educational purposes beyond deepfake detection.

This thesis is structured first to explore what deepfakes are, how they are created, their impacts on society from positive to negative impacts and lastly, how can they be detected either naturally or by using computers, in Chapter 2. Chapter 3 delves into various tools used for website development, programming languages, frameworks, and processes of developing a website. Next in Chapter 4, we move on to the designing of the application, which is the output of this project, using the best practices to create an easy-to-use user interface prototype design was created and is shown in this chapter. This chapter also contains a requirement analysis, which produced a use-case diagram, it also contains the ER diagram and tables related to the database. Chapter 5 talks about front-end and back-end development using the Laravel and Vue.js frameworks and shows the output of the implementation. Lastly, Chapter 6 talks about tools that were used to test the application programming interface and the database, user-based testing, and a test experiment.

Chapter 2

What are Deepfakes?

As defined by Mirsky et al. [35], deepfakes are “Believable media generated by a deep neural network.”

The word deepfake itself is a combination of two words, “deep learning” and “fake”. Deepfakes are AI-generated media which involve swapping the face or voice of a target person with a different person in a video or a photo, to make the target person say or do things, they otherwise have never done or said. Deepfakes come in many forms such as videos, voice recordings, photos, or any other type of manipulation of video or audio. Deepfakes rely on neural networks which have analysed a lot of data sets to learn to mimic a person’s expression, mannerisms, voice, and inflexions. With computational power increasing, it is a simple task to create a deepfake on your own in a relatively short time. Thanks to their easiness of production, they can be seen in our day-to-day lives. What was once reserved for people with good computer skills and knowledge can be accessed by the masses. Thankfully deepfakes are mainly produced for comedic purposes, with exceptions to political propaganda and porn [29].

2.1 Creation of deepfakes

Nowadays, the creation of deepfakes has gained popularity due to being accessible to a wide range of users, from professionals to novices, due to user-friendly applications. These applications are mostly developed using deep learning techniques, further discussed in this section.

2.1.1 Architectures

- **Encoder-Decoder network (ED):** ED consists of at least two networks, an *encoder* and a *decoder*. The encoder compresses the input data, like an image or voice, into a compact form, known as the latent space [35]. This latent space contains all the required information for the data to be reconstructed using a decoder. For example, if there was a face on the input, it would have information about whether the eyes are opened or closed, expression and so on. Then the decoder reconstructs the data from the compressed form [22]. In face swapping, you typically need two sets of Encoder-Decoder pairs. Each pair is trained on different data sets and then the outputs of the encoders are swapped between decoders. This swapping allows the network to reconstruct a face with the expressions or features of another face as seen in Figure 2.1.

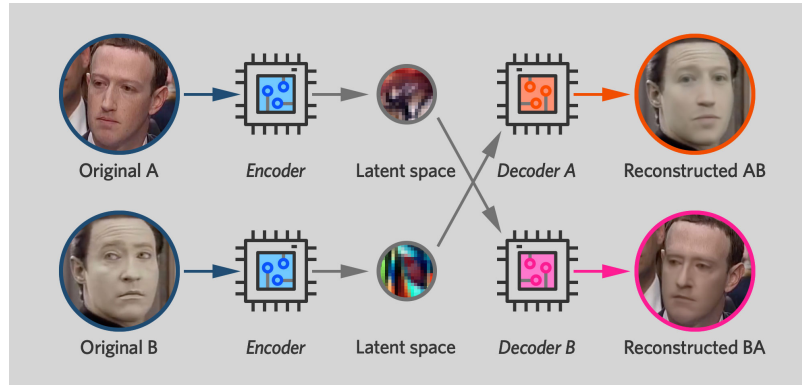


Figure 2.1: A simplified example of how ED networks create deepfakes. From: ([32])

- Generative Adversarial Network (GAN):** GAN consists of two neural networks that are working against each other, the generator and a discriminator as seen in Figure 2.2. The generator is trying to create as realistic images or data as possible so that they are indistinguishable from actual, authentic data [35]. It does this by learning from a dataset of real images and attempting to create new images that closely resemble the real ones. The discriminator evaluates each image to distinguish a real image from a dataset and a generated one by the generator. This creates a competitive dynamic between the generator and the discriminator pushing the generator to create images and videos that are as realistic as possible.

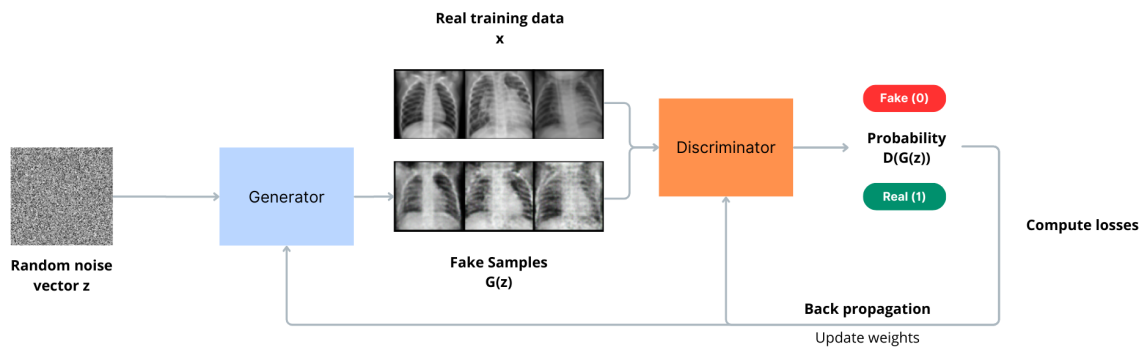


Figure 2.2: Simplified architecture of Generative Adversarial Networks. From: ([9])

- Convolutional neural network (CNN):** The difference between a fully connected network, where each neuron is connected to every neuron in the previous and the following layers, and CNNs is that CNNs learn patterns hierarchically, making them efficient for image processing [35]. CNNs use layers like convolutional, pooling, and upsampling to process and understand images [35], as seen in Figure 2.3.

- Convolutional Layers:** These layers apply a set of learnable filters to the input images. Each filter detects different features, such as edges or textures, at various locations in the input [23].
- Pooling Layers:** Pooling reduces the spatial dimensions of the input volume for the next convolutional layer. This downsampling process not only reduces

computational load but also helps the network to focus on the most important features [23].

3. **Upsampling Layers:** In some CNN models, particularly those involved in image generation or segmentation. Upsampling layers are used to increase the spatial dimensions of the output, enabling the network to maintain or recover detailed information [23]. In the context of deepfakes, CNNs are used to analyse and manipulate facial features in images and videos.

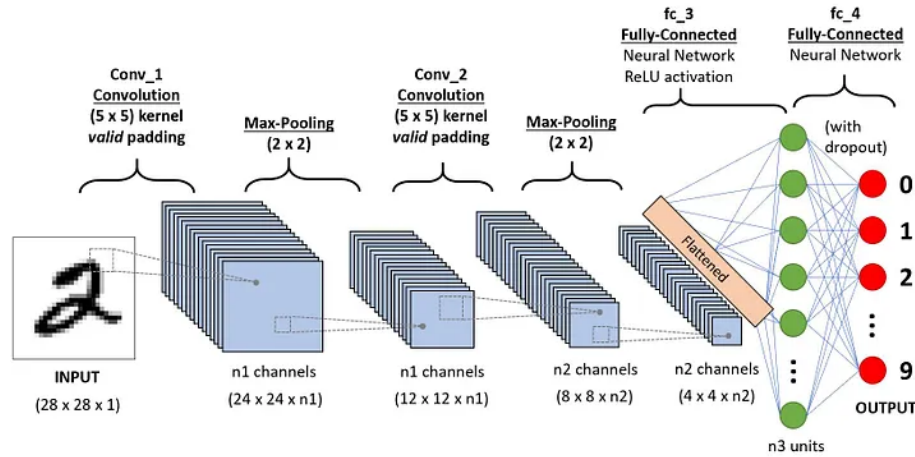


Figure 2.3: Architecture of convolutional neural network. From: ([41])

- **Recurrent neural network (RNN):** RNN is a network whose neurons send feedback signals to each other illustrated in Figure 2.4. It is designed to handle sequential and variable-length data. In deepfake creation, RNN is usually used to create a voice deepfake [35].

2.1.2 Visual deepfakes

Visual deepfake technology generally falls into two primary categories: face swapping and facial reenactment.

- **Face swapping:** Face swapping involves replacing the face of one person in a video with the face of another person, maintaining the original person’s expressions, head movements, and lighting conditions [37], as seen in Figure 2.5. This is achieved through deep learning models that learn the facial features of both the source and target faces to perform the swap convincingly. A popular example of this technology is DeepFaceLab, a tool that has been widely used for creating deepfake videos. The process of face swapping has several stages, including data collection, training the model on the collected data, and refining the output to enhance realism. For example, as described by Ivan Petrov et. al. [37], the process of face swapping in DeepFaceLab involves three key stages:

1. **Extraction:** This initial step focuses on preparing source and destination faces through detection, alignment, and segmentation to isolate faces from any obstructions.

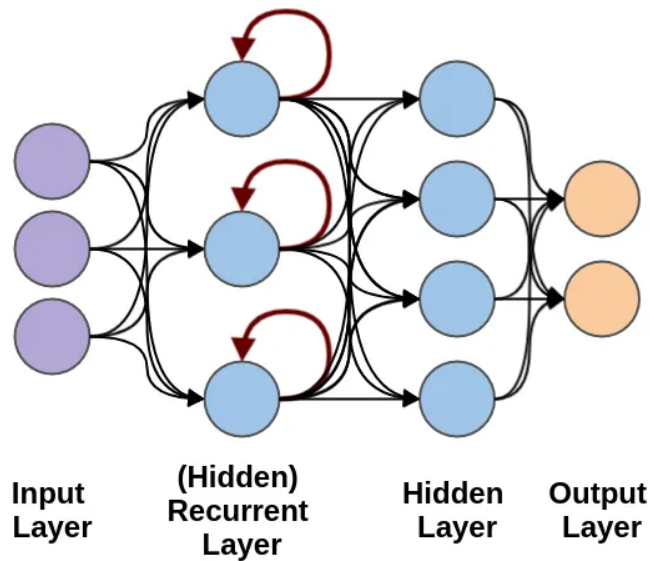


Figure 2.4: Architecture of recurrent neural network. From: ([13])

2. **Training:** This phase uses deep learning to teach the model how to convincingly swap faces, using structures designed to handle differences in lighting and expressions for photorealistic results.
3. **Conversion:** The final step applies the trained model to perform the swap and includes post-processing like blending and colour correction for seamless integration and enhanced realism.

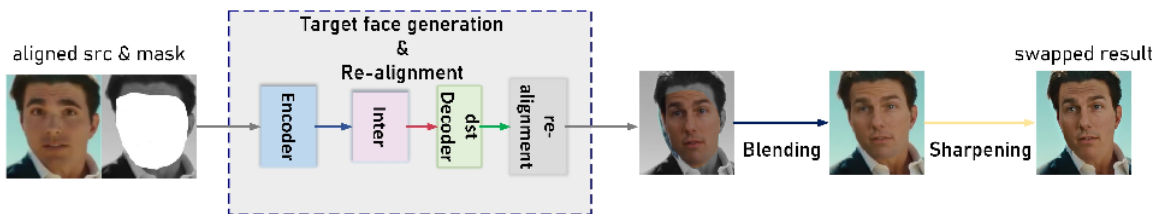


Figure 2.5: An example of the face swapping process. From: ([38])

- **Facial reenactment:** Facial reenactment, also known as face synthesis, involves the generation or alteration of facial expressions in a video, making it possible to control the movements and expressions of the target face [58]. This technology enables the creation of videos where individuals appear to say or do things they never actually did. Key techniques in facial reenactment include the use of Generative Adversarial Networks (GANs) and Recurrent Neural Networks (RNNs) to achieve high levels of realism in the generated expressions [58], an example of this architecture can be seen in Figure 2.6.

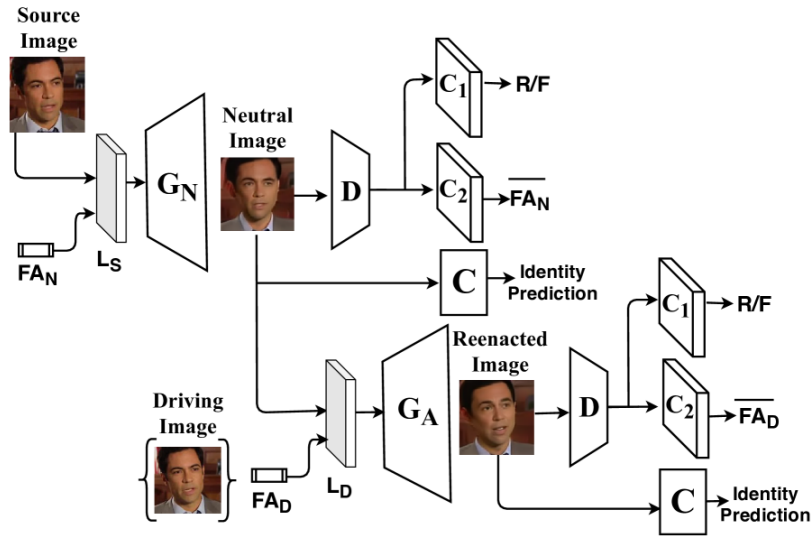


Figure 2.6: Illustration of the overall architecture of the model ICface. From: ([50])

2.1.3 Voice deepfakes

The synthesis of voice deepfakes represents a significant challenge in the field of artificial intelligence, with implications for security, privacy, and media integrity [53]. Human voice synthesis encompasses two general categories: text-to-speech (TTS) and voice conversion (VC).

- **Text-to-speech (TTS):** TTS refers to the artificial transformation of text to audio. The goal of a good TTS is to read a text, considering the naturalness and expressiveness of the voice. A computer system designed for this task is known as a speech synthesizer and can be implemented through either software or hardware platforms [6]. TTS transform text input into audio using the target voice typically with the help of three components: A text analysis module that converts input text into linguistic features, a spectrum decoder (also known as an acoustic module) which generates the acoustic features and a vocoder, which synthesises the actual speech waveform from the acoustic features generated by the spectrum decoder [49], as seen in Figure 2.7. Recent advancements, such as neural TTS models, have significantly improved the quality, making the synthesised speech nearly indistinguishable from genuine human speech.
- **Voice conversions (VC):** The goal of voice conversions is to modify the source speaker’s voice to sound like someone else’s voice while preserving the linguistic content. In other words, it focuses on changing non-linguistic information, which typically includes aspects such as the speaker’s identity, accent, or pronunciation. A typical VC pipeline includes analysis, mapping and reconstruction modules as illustrated in Figure 2.8. The speech analyser breaks down the speech signals of a source speaker into distinct features that represent supra-segmental (like intonation, stress and rhythm) and segmental (like individual phonemes and articulations) information. The mapping module is often the focus of voice conversion research and is pivotal to the conversion process. It takes the analysed features and changes them to resemble

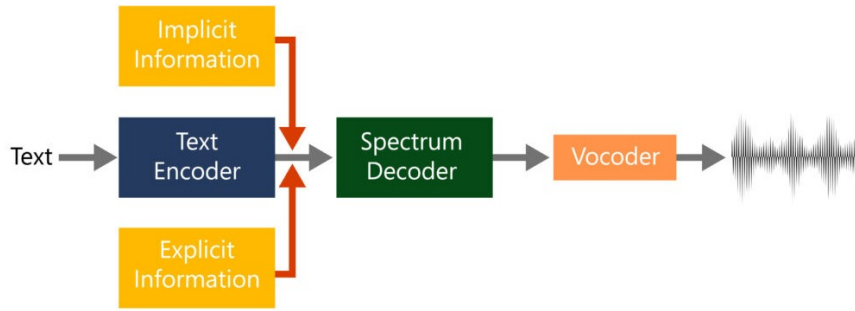


Figure 2.7: The interconnected modules of a text-to-speech system. From: ([61])

the target speaker. Finally, the reconstruction module re-synthesises time-domain speech signals [46].

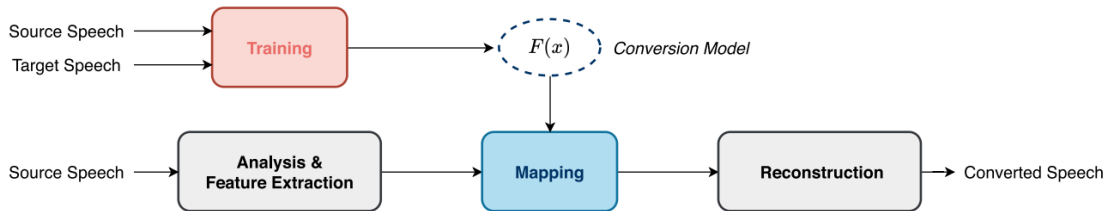


Figure 2.8: The basic structure of voice conversion modules. From: ([46])

2.2 Examples of Deepfakes

The advent of deepfake technology has led to the creation of compelling and sometimes concerning media. Below are several examples demonstrating the wide range of applications and implications of deepfakes.

- **David Beckham:** Utilised in an anti-malaria campaign, deepfake technology allowed Beckham to speak nine different languages, as seen in Figure 2.9, demonstrating its potential to spread public health messages globally.

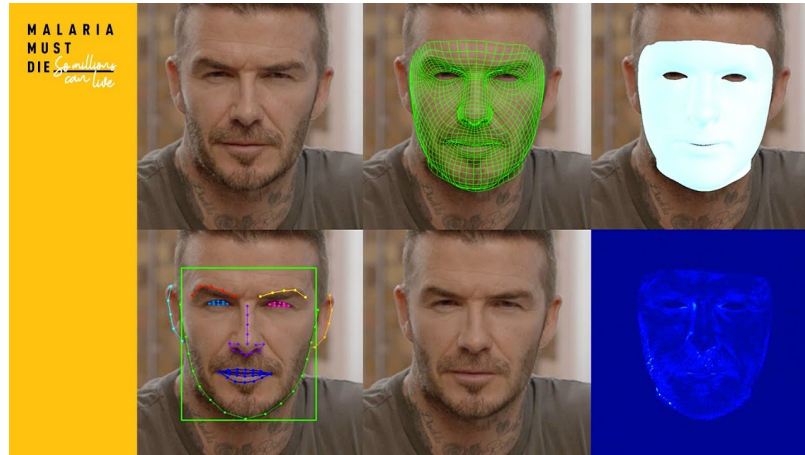


Figure 2.9: David Beckham deepfake in the Zero Malaria Britain campaign. From: ([60])

- Star Wars Fan-Made Deepfakes:** A fan and a deepfake creator Shamook has created deepfakes to improve or reimagine scenes in the Star Wars series The Mandalorian, seen in Figure 2.10, showcasing how technology can enhance visual effects in film making. He was later employed by the Industrial Light & Magic studio, which worked on the series Star Wars and many other movies and series.



Figure 2.10: Comparison between the original VFX and Shamook’s fan-made deepfake in The Mandalorian. From: ([44])

- Barack Obama:** Jordan Peele created a deepfake of Barack Obama, using it to deliver a public service announcement about the dangers of fake news. This deepfake, which synthesises speech and mannerisms similar to the real Obama, is highlighted in Figure 2.11, illustrating the potential for misinformation.



Figure 2.11: Obama deepfake by Jordan Peele, demonstrating the realistic synthesis of speech and expressions. From: ([10])

- **Donald Trump:** Deepfake videos of Donald Trump have been used for both satire and political commentary. They illustrate the potential of deepfake technology in influencing public opinion or creating humorous content. A notable example of such a deepfake is presented in Figure 2.12. These deepfakes have been widely shared across social media platforms, reflecting the ease with which such content can spread.



Figure 2.12: Satirical deepfake of former President Donald Trump, showcasing the use of deepfakes in political satire. From: ([15])

- **Deepfake Roundtable:** A project by Collider, featuring a roundtable discussion among various deceased or aged actors, created using deepfake technology is depicted in Figure 2.13.



Figure 2.13: A screenshot from the Deepfake Roundtable by Collider. From: ([16])

- **Tom Cruise TikTok Deepfakes:** Viral TikTok videos by a Belgian artist, realistically portraying Tom Cruise, as shown in Figure 2.14, demonstrating the sophistication of deepfake technology.



Figure 2.14: Before and after of the Tom Cruise Deepfake by Deep Cruise on TikTok. From: ([52])

- **Queen Elizabeth’s Alternative Christmas Message:** Channel 4’s deepfake of Queen Elizabeth II, aims to illustrate the power of deepfakes and the need for viewer vigilance, as seen in Figure 2.15.



Figure 2.15: A deepfake of Queen Elizabeth II. From: ([12])

- **Deepfake in Music Videos:** The Weeknd’s and Drake’s voices were synthesised by a ghostwriter to create lyrics for his song, which got quite popular before Drake’s label had it pulled [5]. A part of it is still available online here ¹ on the CBC News YouTube channel. Drake and other artists have been deep faked multiple times and it is an ongoing issue whether it is legal or not to synthesise and use someone’s voice [5].

2.3 Impacts of deepfakes on society

It is important to recognise both the positive and the negative impacts of deepfakes. While deepfakes have positive impacts on fields like entertainment, healthcare, education, and communication, they also pose a societal and ethical threat [59]. This chapter will talk about the spectrum of deepfake influence, and their potential to transform industries and personal experiences positively, while also acknowledging the risks and threats they present to personal security, privacy, and the media in our modern age. From medical advancements to posing threats to democracy, the use cases of deepfakes are diverse, making understanding their role in shaping our society’s future important.

2.3.1 Positive Impacts

Deepfakes have a large number of positive impacts on many industries, including movies, educational media and digital communications, games and entertainment, social media and healthcare, material science, and various business fields, such as fashion [56]. This chapter will discuss what those positives are and the future implications of deepfakes.

Medicine

As stated by Westerlund et al. [56] deepfakes can benefit people who lost their voice due to disease by creating an artificial voice for them that sounds similar or the same as their original voice. This is very helpful for people without voice and people around them, to get accustomed to their disability, both mentally and physically. Deepfakes can help people with the loss of their loved ones by creating something of a copy and bringing their loved ones “back to life”, aiding them with their grieving. They also can help people with

¹https://youtu.be/C0ok3Y01L5w?si=azMm3_CSSx0fdSA6

Alzheimer's disease, who often forget the changed faces of their loved ones or even their own. Deepfake faces could be introduced to help these people to interact with others or themselves. Deepfakes can also help provide a deepfake dispatcher or 911 operator, in a way it could stay calm and give advice to people in need. And lastly, as a lifeline operator, which could help people with mental problems, and help them by feeling they can talk to someone.

Spreading messages

Deepfakes can have a positive impact in spreading true and well-meant messages. As mentioned before, David Beckham, during an anti-malaria appeal as seen in Figure 2.9, spoke nine languages at once, showing how deepfakes can broaden the reach of public messages [34].

Modelling and fashion

Did you ever wonder how would a certain outfit look on you? With the help of deepfakes, this could be possible. This could in return save money for you and for the company itself, by reducing the cost of mail and packaging. It would also save you a considerable amount of time, by not having to leave to buy clothes. Another positive for the fashion industry in modelling is that the models would no longer have to re-shoot certain photos for clothes. Companies also often do a re-shoot for clothes that do not sell that well, which could also reduce costs. According to Zara, they could cut as much as 75% of the cost associated with photo shoots [20].

Movie production

Deepfakes could swap faces and voices, which has benefited movie production. It has already been used to revive dead actors, for example in Star Wars, both in Rogue One to make the actress Carrie Fisher look young and in Star Wars: The Rise of the Skywalker, after she sadly passed away [59]. Deepfakes can also be used for dubbing. It is already used for dubbing advertisements and messages from celebrities [33]. Deepfakes could be in the future used for updating episodes of movies without the need to re-shoot them [26].

Education

In the field of education, we can use them to create deepfakes of freedom fighters, scientists, doctors and any person in history. Then a fake video with them talking about their life or the ability to communicate with them can be helpful to students. Interactive ways of learning in this way can prove to be more effective [59].

2.3.2 Negative Impacts

Deepfakes are like a flame, that in some instances can be useful, but in others can be extremely dangerous. It is crucial to find a balance between useful and dangerous traits of deepfakes. This section will talk about the possible negatives of deepfakes, some of which could happen now, and some of which will be possible with the unstoppable improvements made in the field. Of course, there are more kinds of illegal or harmful usages of deepfakes than illustrated here, but they all boil down to swapping faces or voices to harm a person.

Spreading misinformation

Deepfakes are getting increasingly complex, and the imperfections are getting harder to spot. Also, the technology is becoming more popular, and it is generally easier to create a deepfake. It is very difficult to distinguish between a real and a fake video, and it gets even harder with a fake voice. People start spreading fake news with fake facts on social media and the fake video makes it even more believable [59].

Identity theft

Identity theft is a common cybercrime in developed countries. It involves theft and forgery of persons' details, like social security numbers, passports, etc. Then the attacker can perform various purchases, book tickets, or engage in unlawful activities on the victim's behalf. Deepfakes in this case can extend the severity of the attack by providing the attacker with the victim's face and voice. Then the attacker could create video calls, and post photos and videos on social networks, which will be very difficult to spot. Because the model's training data is much more accessible than the victim's data, deepfakes can help the attacker to expand their scope of attack if they need to [45].

Personal attacks

Deepfake was created for entertainment purposes, but nowadays communities of people create all kinds of immoral imagery, ranging from pornography to personal attacks. Celebrities are affected greatly by deepfake pornography. Users online want to bring their dreams to life so they deepfake a celebrity's face on an adult actress or actor [59]. Personal attacks on individuals are rare, typically being of a humorous nature. However, there are exceptions, such as when a cheerleader's mother created a deepfake video of another cheerleader smoking an electric cigarette which led to the cheerleader being removed from the team and her daughter being able to get into the team [54].

Threat to democracy

Deepfakes can be easily spread and can be used to misinform the public knowingly or unknowingly for political advantage [8]. If this technology is used against a politician or an important public figure it could damage their reputation in favour of the other political party or candidates [59]. One example of this would be a deepfake of Ukrainian President Zelensky, asking Ukrainian troops to lay down weapons and surrender [43].

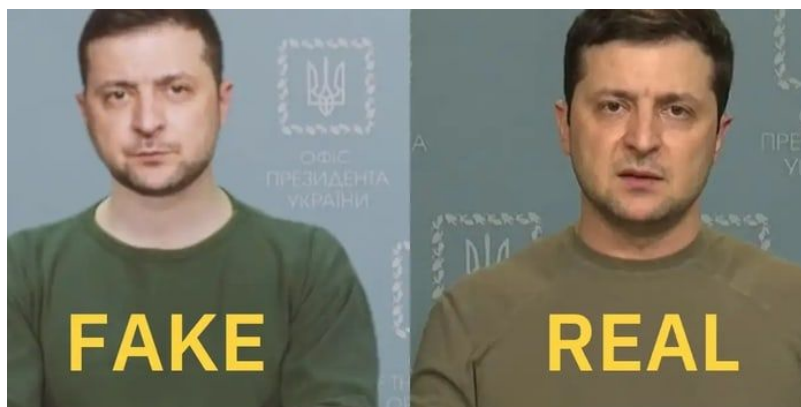


Figure 2.16: A deepfake of President Zelensky compared to his real photo. From: ([51])

2.4 How to detect deepfakes?

Detecting a deepfake is one of the fundamental skills for the near future. As the technology progresses, we probably won't be able to spot a deepfake by ourselves. This is where technology comes in, to help us look for the small details and imperfections. This section will go over general tips on how to spot a deepfake and the technology that can do it in our stead.

2.4.1 Naturally

Spotting a deepfake can be challenging, yet several inconsistencies and abnormalities can often aid us in spotting a deepfake. This section outlines practical methods that can help in distinguishing real videos and images from their deepfake counterparts. By observing details closely on deepfake images, we can leverage innate discrepancies that are overlooked by algorithms generating the deepfakes, as seen in Figure 2.17. Each method listed provides an approach to assess and question the authenticity of digital media.

- **Compare to real videos and images of the person:** This is the simplest way to check if something is a deepfake, is to compare a real video or a photo to the deepfake and look for abnormalities.
- **Unnatural eye movement:** A common red flag with video deepfakes is weird eye movement, such as an absence or too much blinking. Try to also focus on how the eyes are moving. It is difficult to replicate correct eye movement because eyes usually follow the person they're talking to [48].
- **Unnatural facial expressions:** When something does not look right about the face. For example, it is common that deepfakes have no or too many wrinkles while moving their lips or raising their eyebrows. Some facial expressions may even be looped or be the same as on a different deepfake [48].
- **Awkward facial-feature positioning:** If someone's face is pointing in a different direction than their nose, you should be sceptical about the video's authenticity [48].

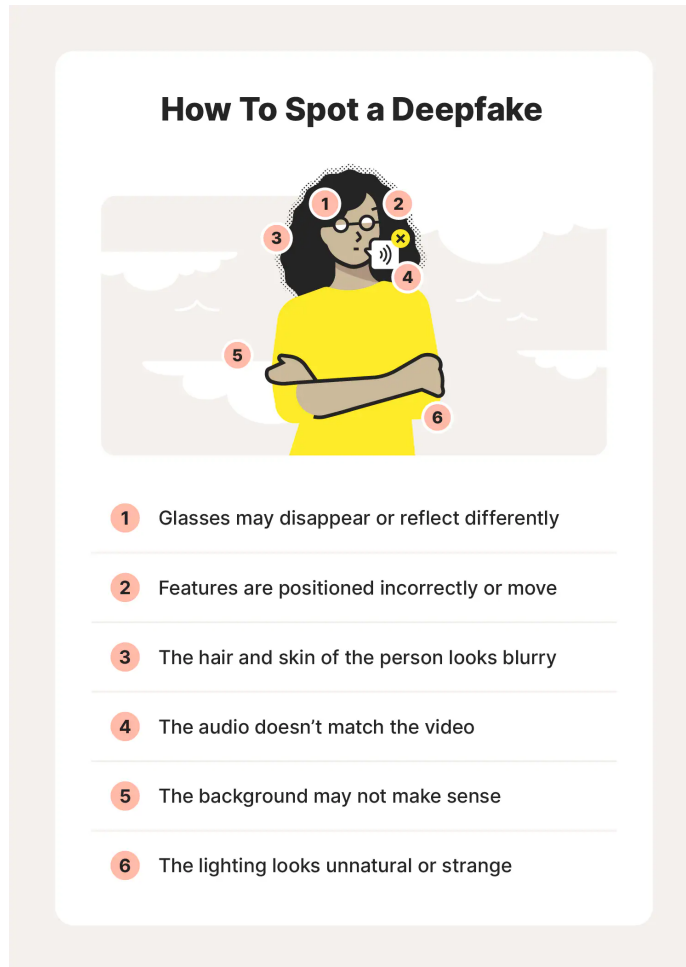


Figure 2.17: An example of how we can spot a deepfake. From: ([47])

- **Hair that does not look real:** Creating hair that looks realistic is difficult so you usually won't see a deepfake with frizzy or flyaway hair [48]. This goes for facial hair too. Deepfakes sometimes add or remove moustaches, sideburns, or beards [24].
- **Unnatural colouring:** Abnormal skin tone, discolouration, weird shadows or lighting are all signs that the imagery is fake [48].
- **Inconsistent audio and noise:** Creators of deepfakes usually spend more time on the imagery than the audio. Look for unsynced audio with lip movement, robotic-sounding voices, strange pronunciations, or even a lack of audio [48].
- **Glasses:** Deepfakes may fail to fully represent the natural physics of lighting, so look out for too much or too little glare. Parts of glasses might even disappear [24].
- **Unnatural body movement:** Look out for jerky or distorted movement, or even lack of movement [48].
- **Blurred or misaligned background:** The background might be too blurry or unnatural and usually won't move or have a shadow of the person [48].

2.4.2 Using computers

The challenge of detecting deepfakes is an indefinitely evolving field that calls for a multi-faceted approach as seen in Figure 2.18. While machine learning and forensic analysis have proven effective, the sophistication of deepfake technology means that no single method is foolproof. Techniques for identifying deepfakes range from artefact-specific methods that target inconsistencies in blending boundaries, environment context, and forensic details, to temporal approaches that exploit anomalies in behaviour, physiological signals like heart rates or blinking patterns, and coherency across video frames. Instead of focusing on a specific artefact, some authors train deep neural networks to work as a generic classifier and let the network decide which features to analyse [35].

Machine Learning-Based Methods

Machine learning uses traditional algorithms like Decision Trees, Random Forests, and Extremely Randomised Trees, providing explainability and manageability in decision-making. These algorithms are useful for explaining the reasoning behind decisions in a way that people can easily understand, which is very important for dealing with deepfakes [39].

Deep Learning-Based Methods

Deep learning extends the capabilities of machine learning by providing more complex models, that can be used to target artefacts and inconsistencies in the pipeline of generating deepfakes. Methods range from GAN simulators replicating and detecting collective GAN-image artefacts to networks that extract standard features from RGB data and physiological measurements like heartbeat detection. The development of specialised network architectures, including attention mechanisms and capsule networks, further enhances the accuracy and efficiency of deepfake detection models, achieving significant success rates [39].

Statistical Measurements-Based Methods

Statistical methods provide another angle for deepfake detection, focusing on the analysis of unique noise patterns and cross-correlation scores between original and suspected data. Techniques like the examination of photo response non-uniformity (PRNU) patterns offer a statistical basis for distinguishing between genuine and manipulated content [39].

Blockchain-Based Methods

The advent of blockchain technology introduces a novel approach to verifying the authenticity and provenance of digital content. By leveraging blockchain's decentralised and secure nature, this method provides a means to trace the origin of videos and images conclusively [39].

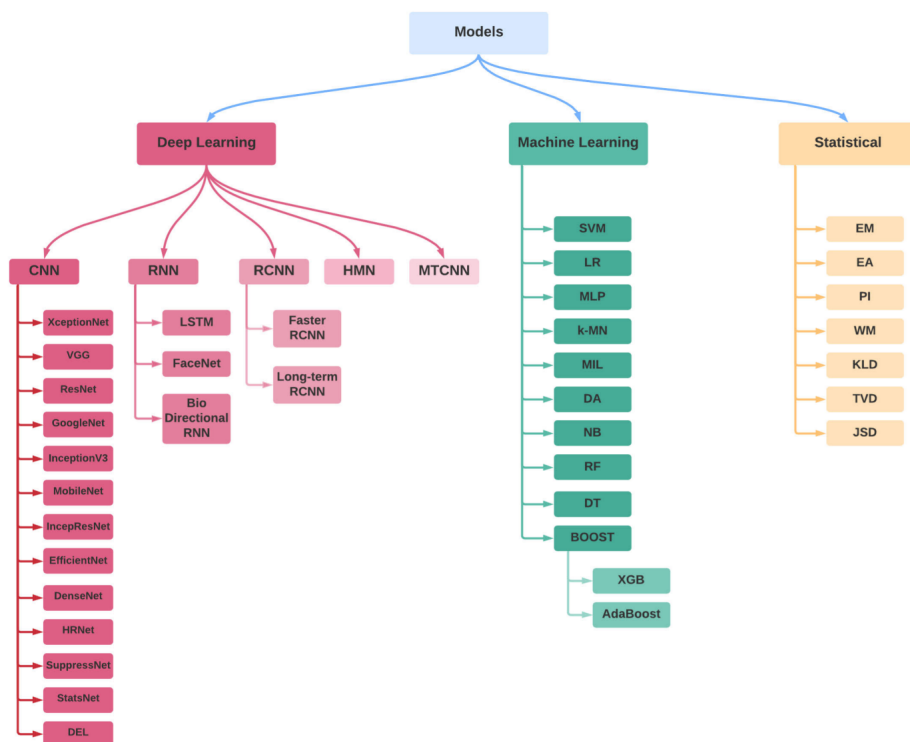


Figure 2.18: An overview of computer-based deepfake detection methods. Adapted from: ([39])

Chapter 3

Web Development

As the capabilities of deepfakes continue to advance, people need to stay informed. Web technologies can be used to create a robust platform that will test critical thinking and help create an understanding of such sophisticated AI-generated content. Web applications are easily accessible to all with a connection to the internet and a computer, which will help with collecting as much important data as possible.

This chapter then creates an overview of technologies used for development like front-end and back-end technologies and how to create effective, user-friendly applications. Starting with HTML and CSS, the core technologies responsible for structuring and styling [11]. Later we will move on to server-side scripting language PHP and SQL for database management. These technologies are crucial for creating a complex web application, that is dynamic and can store data. This part is not only about the technologies themselves but how they can be used together to create a comprehensive user experience.

3.1 Front-end

Front-end development, often referred to as client-side, is responsible for showing data to the user [11]. It is crucial to make a web application visually engaging but also intuitive and accessible, enhancing the user's interaction with the technology. This section of the chapter will focus on how HTML (HyperText Markup Language) and CSS (Cascading Style Sheets) serve as the foundational elements of web design and how together with JavaScript they can create a simple front-end-only application. For more complex applications, a back-end is needed for data storage and retrieval.

3.1.1 HTML

HyperText Markup Language or HTML is not a programming language in the traditional sense, but rather a markup language that defines the structure of the web [57]. Unlike traditional programming languages, HTML focuses on defining the layout and structure of web pages through the use of elements [11]. Elements are typically composed of a starting tag, content, and an ending tag. Elements tell the browser how to display the content. A standard HTML document should include a definition of the document type `<!DOCTYPE html>`, which indicates the HTML version for browser compatibility. Following this, the HTML tag `<html>` with optional language attribute `<lang="language">`, which encapsulates the entire content. The content is then divided into a head and a body. Head tag for title and metadata of the website and body for the content itself [11]. Language

attribute is not mandatory but it is a good practice to define the document's language for accessibility purposes. Below is an example of a basic HTML document structure.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <title>Title</title>
</head>

<body>
  Content
</body>

</html>
```

This structure serves as a backbone for web pages, with proper nesting and maintaining of the integrity of the HTML structure a more complex website can be created.

3.1.2 CSS

Cascading Style Sheets or CSS help us define the style of the web page [11]. CSS file is an ordinary text document, which consists of one or more rules, which define how HTML elements should appear on the web. Beyond basic text formatting, such as changing font sizes, colours, and styles, CSS offers a wide range of capabilities [11]. It controls layout properties, including margins, borders, padding, and positioning, which are crucial for designing responsive and aesthetically pleasing web interfaces. CSS also supports advanced features like animations, transitions, and even grid or flexbox layouts, providing immense creative flexibility. CSS is also crucial when it comes to the ability to view a web page on plethora types of devices, like phones, tablets and desktops. It formats the web page to look great and be usable on these types of devices.

3.1.3 JavaScript

JavaScript (JS) is a versatile, high-level programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. JS provides us with tools to make web pages feel alive by making them interactive. Initially, it was only used for client-side scripting, but now it is also a popular choice for server-side programming through environments like Node.js [28].

Key attributes of JavaScript include:

- JavaScript can manipulate webpage content, and styles, and respond to user events.
- It can perform network requests to servers, and handle files, cookies, and local storage.
- Modern JavaScript can even interact with the device hardware, such as cameras and microphones, with user permission.

The language's asynchronous capabilities, particularly with the advent of Promises and Async/Await syntax, have simplified the handling of operations that may take an indefinite

amount of time, such as fetching data from a remote server [28]. In terms of limitations, JavaScript in a browser environment operates within a secure sandbox, limiting access to the user's file system and restricting the execution of potentially harmful operations. This security model ensures the web remains a safe environment for users [28].

3.1.4 Best Practices for Creating a User Interface

To gain as much data as possible from the experiment the user interface (UI) needs to be intuitive and easy to use. Creating a user interface is naturally an iterative process, It is impossible to just visualise the design for a large software system well enough to make it comprehensive and understand all of its implementations [18]. Therefore, designers often start with a simple sketch or a mock-up of an application. To help us with the creation of user interfaces, best practices and methods were created. While not everyone can create an innovative interface, these best practices are tried and tested methods whose correct application can lead to a comprehensive UI. Ruiz et. al. [25] performed a systematic literature review to identify the most relevant authors in the functional design domain and extracted 257 principles, which were then unified and derived into a core selection of 36 principles, which can be used to improve the UI functional design. The list below is the most cited 36 principles from a study created by Ruiz et. al. [25] with their explanation, these principles are used for the design of the deepfake experiment web application.

- **Offer informative feedback:** Ensure that the system provides feedback that is clear and informative about what actions have been taken and what results have been achieved.
- **Strive for consistency:** Maintain uniformity in the user interface to prevent confusion, with consistent commands, colours, layouts, and terminologies throughout the system.
- **Prevent errors:** Design the interface in a way that minimises the chances of user errors and makes it easy to recover from them when they do occur.
- **Minimise user's memory load:** Reduce the amount of information users need to remember by keeping options visible and using intuitive interface elements.
- **Simple and natural dialogue:** Keep interactions straightforward and natural, mimicking human dialogue to simplify user understanding and response.
- **Know the user:** Design interfaces with a deep understanding of the user's background, skills, and needs.
- **Actions should be reversible:** Allow users to undo actions to alleviate the fear of making irreversible mistakes, enhancing their comfort and confidence.
- **Make things visible:** Ensure all needed options and materials are visible without cluttering the interface, making the system intuitive.
- **Give the user control:** Empower users by allowing them to control their interactions with the system, providing flexibility and adaptability.
- **Structure the user's interface:** Organise the interface logically and functionally to enhance usability and learnability.

- **Iterative design to remove usability problems:** Use iterative design processes to identify and eliminate usability issues, improving the user experience over time.
- **Provide shortcuts:** Offer experienced users faster ways to navigate or perform tasks, which can speed up the interaction for frequent users without confusing novice users.
- **Good error messages:** Provide clear, informative, and non-technical error messages that guide users towards solving problems.
- **Speak the user's language:** Use language that is familiar to the user, avoiding technical jargon, to make information easy to understand and act upon.
- **Use real-world metaphors (transfer):** Use metaphors that draw on users' real-world experiences to make digital interactions more relatable and understandable.
- **Understand the tasks:** Design with a clear understanding of the tasks users will perform, ensuring the interface facilitates these tasks effectively.
- **Empirical measurement:** Validate design decisions with empirical data from user testing, ensuring that the interface meets actual user needs and behaviours.
- **Allow users to change focus:** Design interfaces that accommodate changes in user focus, allowing for flexibility in interactions.
- **Provide visual cues:** Use visual elements that guide users through their interactions with the interface, such as buttons that look clickable or progress bars that show completion status.
- **Provide clearly marked exits:** Make it easy for users to exit any given state without having to go through an extended process, enhancing usability and comfort.
- **Help users recognise, diagnose and recover from errors:** Design systems that not only prevent errors but also make it easy for users to recover from them when they occur.
- **Accommodate users with different skill levels:** Ensure the interface is accessible and usable by people with varying levels of technical skill.
- **Provide a good conceptual model of the system:** Offer users a clear and intuitive understanding of how the system works, making interaction easier.
- **Flexibility and efficiency of use:** Design interfaces that cater to both inexperienced and experienced users, allowing them to customise or adapt procedures to meet their needs.
- **Recognition rather than recall:** Design interfaces that minimise the need for users to recall information from memory by making information available when needed.
- **Reuse:** Utilise existing design solutions where appropriate to save time and ensure consistency.
- **Allow users to customise the interface:** Provide options that let users tailor the system to their needs, enhancing personal relevance and ease of use.

- **Integrated design:** Ensure all components of the interface work together seamlessly to provide a coherent user experience.
- **Encourage exploration:** Design systems that invite users to explore different functionalities without the risk of making serious errors.
- **Display inertia:** Ensure that the interface behaves in predictable ways that are consistent with user inputs and expectations.
- **Design dialogues to yield closure:** Structure dialogues with users so they have a clear beginning, middle, and end, providing a sense of completion.
- **Cater to universal usability:** Design interfaces that are accessible and usable by as wide a range of people as possible, regardless of age or ability.
- **Support internal locus of control:** Empower users by designing interfaces that make them feel they are in control of the technology rather than being controlled by it.
- **Constraints:** Limit what users can do in the interface to prevent errors and streamline the interaction process.
- **Help and documentation:** Provide easily accessible help and documentation that supports users in understanding and using the system.

3.2 Back-end

3.2.1 PHP

PHP, which is an acronym for “PHP Hypertext Preprocessor”, is a server-side scripting language that is specifically designed for web development. PHP’s scripting code can be embedded within HTML, and it executes on the server side, generating HTML or other outputs that are then sent to the client’s browser. This process allows for the creation of dynamic content that updates each time a web page is visited [55]. PHP’s support for both functional and object-oriented programming paradigms, along with its extensive built-in and external functionalities, make it an adaptable and powerful language for web development. Its ability to seamlessly integrate with HTML and other web technologies, combined with its vast community support, ensures that PHP remains a popular choice for developers around the globe.

3.2.2 MySQL

MySQL is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL), the most popular language for managing and manipulating databases. As an RDBMS, MySQL facilitates the efficient storage, retrieval, searching, and sorting of data. This makes it an essential component in the architecture of many web applications. MySQL controls access to the database to ensure that multiple people can work with it concurrently. Hence, MySQL is a multi-user, multi-threaded server [55]. MySQL is founded on the SQL language and enhances it with a range of built-in functions. The interaction with the database is carried out using SQL queries.

3.3 Frameworks

In the early stages of the web development era, all of the applications were programmed by hand, which led to a lot of errors and work. Frameworks mitigate this by providing tools that help build a website thus avoiding bugs and conserving time [17]. Frameworks can be divided into client-side and server-side. Client-side frameworks are responsible for implementing the user interface and improving them by adding animations. Server-side frameworks offer the ability to implement more complex, secure and complete web applications faster by providing a convenient file structure [7]. Frameworks increase the productivity of developers by being modular, consisting of a set of libraries, tools and conventions, which makes it easier to focus on functionality rather than spending hours writing a piece of code [7]. Frameworks usually adopt the model-view-controller (MVC) architecture. Frameworks also provide a comprehensive Application Programming Interface (API) that includes classes and methods to streamline routine tasks such as processing requests and generating responses.

3.3.1 Server-side Frameworks

PHP frameworks, coupled with MySQL databases, represent a powerful combination for server-side development. Popular PHP frameworks include [42]:

- **Laravel:** Laravel¹ is known for its elegant syntax trying to make the development process more pleasing without sacrificing functionality [14]. Laravel simplifies tasks like routing, sessions, caching, and authentication. It can be combined with Vue.js easily, making it an ideal choice for full-stack development. Laravel's features include a modular packaging system with dependency management, ease of access to relational databases through Eloquent ORM, and its orientation towards syntactic sugar [14]. It is great for smaller applications because of its performance efficiency [14].
- **Symfony:** Symfony² is one of the older PHP frameworks. This framework is recognised for its performance and flexibility. It is a go-to framework for high-performance web development projects [14]. Symfony is a heavyweight framework, meaning that not a lot of third-party packages have to be installed.
- **CodeIgniter:** CodeIgniter³ is a light-weight yet powerful frameworks [14]. It is easy to install, which makes it accessible to many developers. CodeIgniter focuses on a small footprint, elegance and simplicity. It does not have a built-in scaffolding tool, so models, views and controllers have to be all programmed by hand.

3.3.2 Client-side Frameworks

Using front-end frameworks is essential for crafting interactive and responsive user interfaces in client-side development. These frameworks not only make it easier to develop a website but also provide a structured approach to application design. Some of the most popular frameworks are [40]:

¹<https://laravel.com/>

²<https://symfony.com/>

³<https://codeigniter.com/>

- **Vue.js:** Vue.js⁴ is one of the most optimised frameworks for developing both multi-page (MPA) and single-page (SPA) applications. It is also very flexible and easy to use. Vue.js is one of the most popular and modern front-end frameworks, released in only 2016 and is one of the most starred repositories on Git Hub. Vue.js also supports server-side rendering, is easy to set up, and has low overhead and robust state management [27].
- **React.js:** React.js⁵ can be used for both MPA and SPA development but it is less suitable for SPA than Vue.js, due to its reliance on third-party packages for routing and state management [27].
- **Angular:** Angular⁶ is a platform and framework for building single-page client applications using HTML and TypeScript. Angular is not suitable for MPA due to its complexity, size and optimisation needs, but it is great for SPA thanks to its comprehensive structure [27].

3.3.3 Choosing Laravel and Vue.js for Project Development

For this project, Laravel was chosen together with Vue.js. While Laravel is one of the larger frameworks it provides great security. It is one of the fastest and most popular PHP frameworks as seen in these Figures 3.1, 3.2, 3.3 and 3.4 comparing Laravel, Symfony and CodeIgniter. The combination with Vue.js was done for several reasons their great compatibility, both frameworks focus on elegant code, and both frameworks are fast and good for small projects.

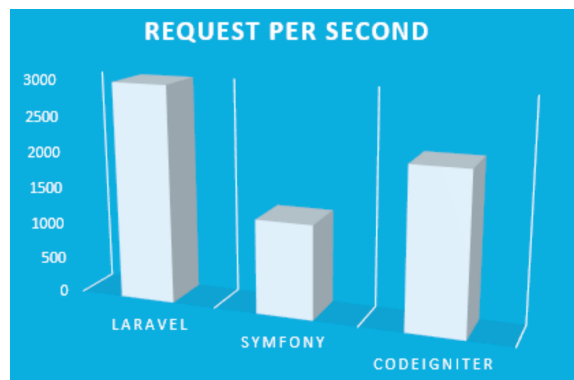


Figure 3.1: Comparison of requests per second of Laravel, Symfony and CodeIgniter. From: [31]

3.4 Development Process of Web Applications

The process of developing a successful web application according to French et al. [21] consists of 8 steps as seen in Figure 3.5. All these steps will be described below. The deepfake experiment application was implemented using this development process. The process is

⁴<https://vuejs.org/>

⁵<https://react.dev/>

⁶<https://angular.io/>

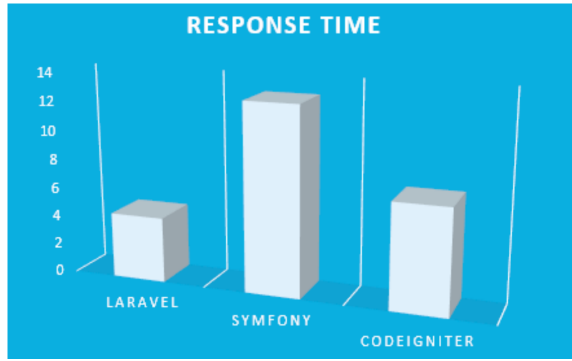


Figure 3.2: Comparison of the response time of Laravel, Symfony and CodeIgniter. From: [31]

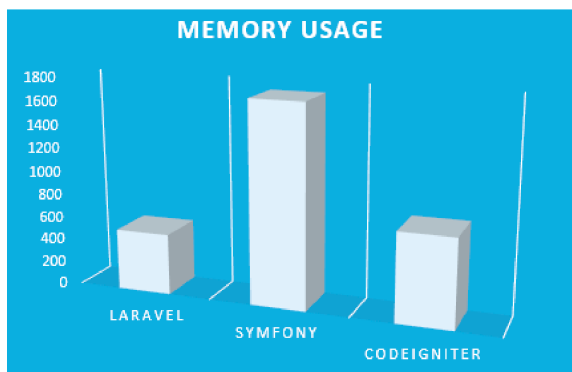


Figure 3.3: Comparison of memory usage of Laravel, Symfony and CodeIgniter. From: [31]

similar to a waterfall approach, however, by incorporating prototyping methodology, the process became iterative to better adapt to the changes needed to be done to create a well-working application.

Information Gathering (Graphical)

The first phase is the information gathering for designing a website. The design of the website is important because if it is not appealing to the user they might not want to use it again and the customer will be less likely to purchase the product [21]. The analyst is required to gather the information that will help designers create an effective layout of the application and determine pages that will be included. How information will be presented and navigation through this information is also discussed in this step.

Analysis (Graphical)

The second phase is analysing the gathered information and documenting the needs. This documentation then includes colour schemes, logos and other graphical elements [21]. The analyst will also outline the site map showing how the users will navigate through the website. The graphic designers then create an image of what the website should look like, which the programmers will use to develop the website.

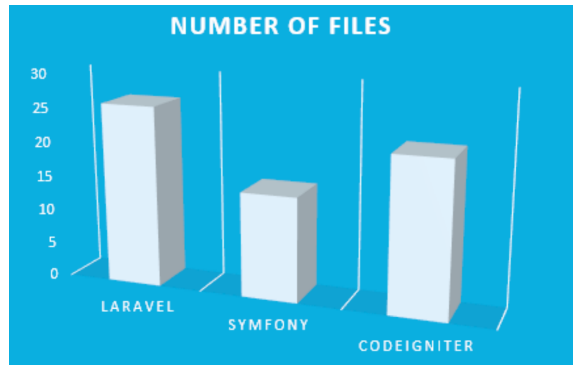


Figure 3.4: Comparison of the number of files of Laravel, Symfony and CodeIgniter. From: [31]

Graphical Design

In this step, the prototype of the website is created. The only functionality available at this stage is navigation through the website [21]. This is also where the templates are created that will be later used in the actual system.

Information Gathering (Functional)

In phase 2 the functional development is done. During this phase, the analyst meets with users and collects requirements for the functionality [21]. For this, a requirement analysis is needed together with a use case diagram.

Analysis (Functional)

The functional analysis phase is where the analyst creates the Entity Relationship Diagram (ERD) and Data Flow Diagram (DFD) needed for the functionality of the website and to break down different components into smaller fragments [21].

Functional Design

In this phase, the developers start creating a prototype for each component of the website. The main focus is on developing the website's functionalities. Developers closely collaborate with the users to pinpoint the essential components for successful implementation. While the users can have a clear vision of their requirements they might not fully grasp all the possibilities within the web development [21]. As features are added to the website the users must test them and give feedback to the developer. After the component has been successfully implemented and tested the developer moves on to the next component and repeats the process. This creates an iterative process and increases the development speed as the modules can be developed separately.

Implementation

The next step is developing a prototype on a test server. This allows the users to work with the developers until the prototype is ready for deployment on the production server [21].

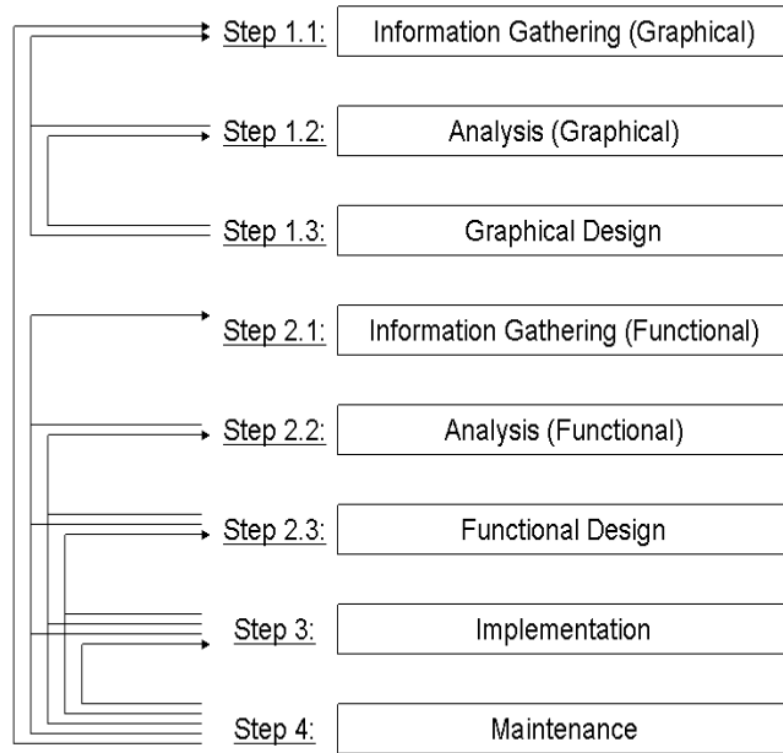


Figure 3.5: Web Development Life Cycle. From: ([21])

Once the components are ready all files of the web page and the database are moved to the production server.

Maintenance

Maintenance might include modifications, updating style, fixing bugs or anything that the user might want to change [21].

Chapter 4

App Design

4.1 Requirements Analysis

This web application is designed to research the human ability to recognise synthetic speech. Therefore, the application should allow administrators to add recordings, even multiple at once, remove recordings, edit recordings, download user data, remove user data, and remove user profiles. It is important to have the ability to remove and add recordings to have a comprehensive dataset, that always keeps up with the current trends in deepfake technology. Administrators can set the order of these recordings, ensuring a controlled environment for the experiments. For the users participating in the experiments, the interface should be easy to navigate and use. Users will have the ability to change the language of the website and watch the tutorial which further makes the application more accessible. Multi-language support is crucial to gaining as much data as possible, helping researchers gain an understanding of how people from different countries and backgrounds interact with synthetic speech. One of the main features is the ability for a user to return to an ongoing experiment. This functionality is crucial if there are many recordings in the experiment and the user cannot finish the experiment in one go. This information was then used to create the use case diagram, depicted in Figure 4.1, which is the product of the requirements analysis. It shows the interactions between the system, users, and administrators, clarifying the roles and functionalities. Creating a use case diagram can help with the designing of the website and that all the uses are taken into consideration.

4.2 Architecture Design

This chapter is focused on two main architectures used, Two-Tier and Three-Tier Client-Server architectures, as they are the most used and also the most relevant for this thesis.

4.2.1 Client-Server Architecture

The client-server architecture is a fundamental design model that segregates the system into two primary components: the client, which initiates requests for services, and the server, which processes those requests and serves the responses.

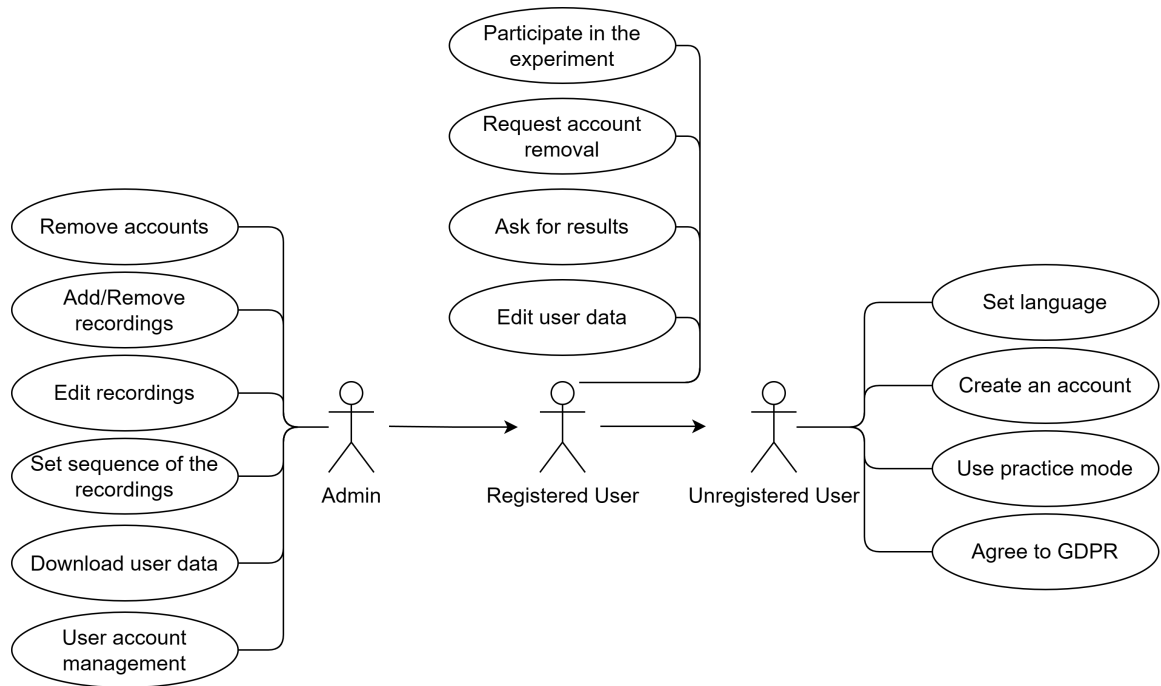


Figure 4.1: A use case diagram for this application

Two-Tier Architecture

The two-tier architecture application is separated into two parts, application and database. The application part handles both the presentation and application layer [30]. It is also known as a “thick client” because the users will run applications on their PC, which connects through a network to the server [36]. The client device sends the request to the server and the server processes the request and, sends back the request data to the client system, which is then displayed as seen in this Figure 4.2.

Three-Tier Architecture

Three-tier architecture introduces an intermediary layer between the client (presentation layer) and the server (data layer), known as the application layer. The client system handles the presentation logic only, the server manages the application layer and the database server then manages the database layer [30] as seen in Figure 4.3.

- **Presentation Layer:** This layer is responsible for the user interface and user experience. It shows the users the functionalities in a graphical manner, these functionalities then can be performed, and the results of these operations will then be displayed to the user. The main focus here is on accessibility and visuals [30].
- **Application Layer:** Application or business layer is responsible for all the logic and connecting the presentation layer with the data layer. All of the calculations, instructions and the manipulation of data are done here [19]. When a user creates a request it is routed through the application layer, and then the application requests the data from the database. After manipulating the data the data are sent back to the presentation layer.

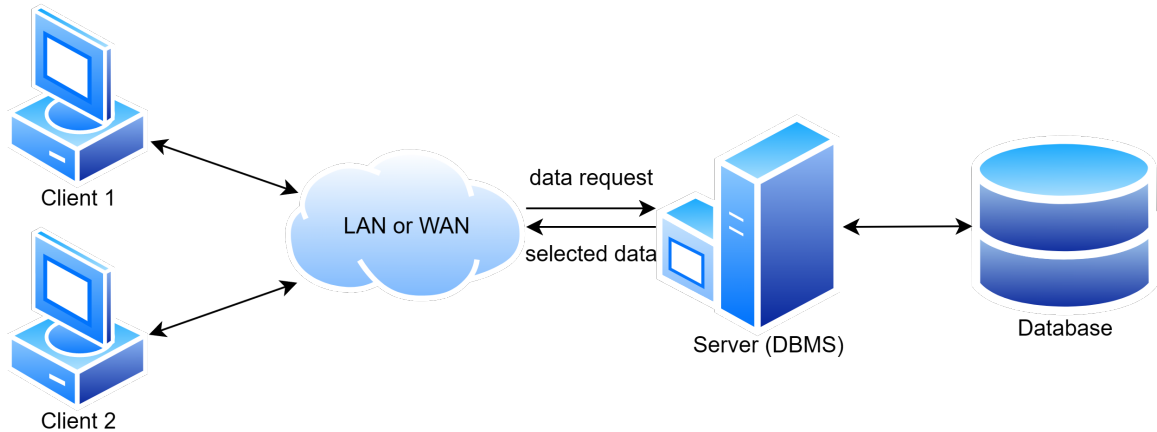


Figure 4.2: Two-Tier Architecture

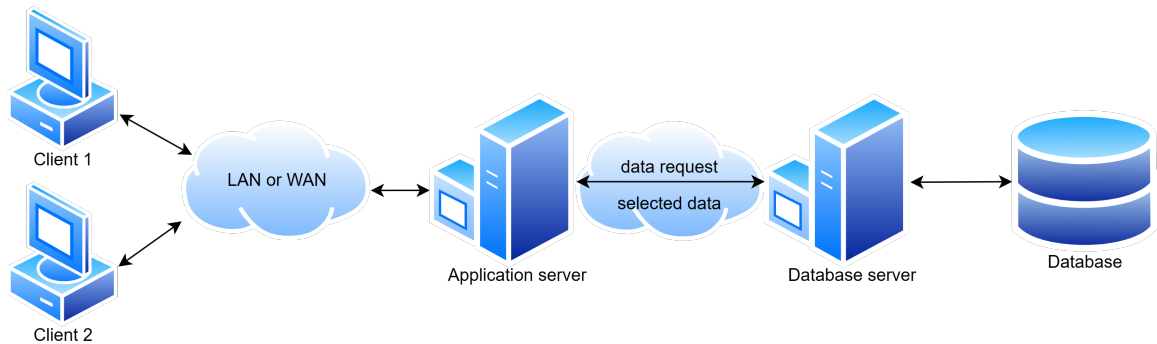


Figure 4.3: Three-Tier Architecture

- **Data Layer:** Also known as the database layer, the data layer is where data is stored, retrieved, and managed. The data is not altered by the database layer but itself but rather through requests from the application layer [19]. It provides the application layer with a means to perform CRUD (Create, Read, Update, Delete) operations. This layer focuses on data integrity, privacy, and efficient data storage and retrieval.

Since the application will be implemented using Laravel framework, the architecture will be a three-tier architecture. This architecture has many benefits like logical blocks being separated, scalability and that the user will only need a browser and not a “thick client”.

4.3 User Interface

The user interface plays an important role in the success of the experiment. It needs to be user-friendly, allowing for easy navigation, confirmation of user actions, and responsiveness, adhering to the best practices for UI design mentioned in Section 3.1.4. Therefore, a simple colour palette has been chosen with colours that are easy to differentiate from the background. When creating a user interface for this web application it was important to focus on the best practices for front-end design. All pages have informative feedback for users to know if something is not right and visual cues of what needs to be done. Consistency

across the pages was also achieved as shown in the figures below. All important buttons were made easily visible with bolder colours. The design of the web page was simplified as much as possible so that users did not make as many mistakes and was organised logically and functionally. The pictures below show the prototype of the user interface before it was implemented.

4.3.1 User part

This section shows the user part of the experiment and explains the decisions behind the design.

Initial screen

The first screen of the experiment has a simple design. It displays the name of the experiment and a brief description of the experiment. There are two large buttons, one to start the experiment and the other to start practice mode. Additionally, there is a drop-down menu that allows the user to select the preferred language, and a button to log in as seen in Figure 4.4.

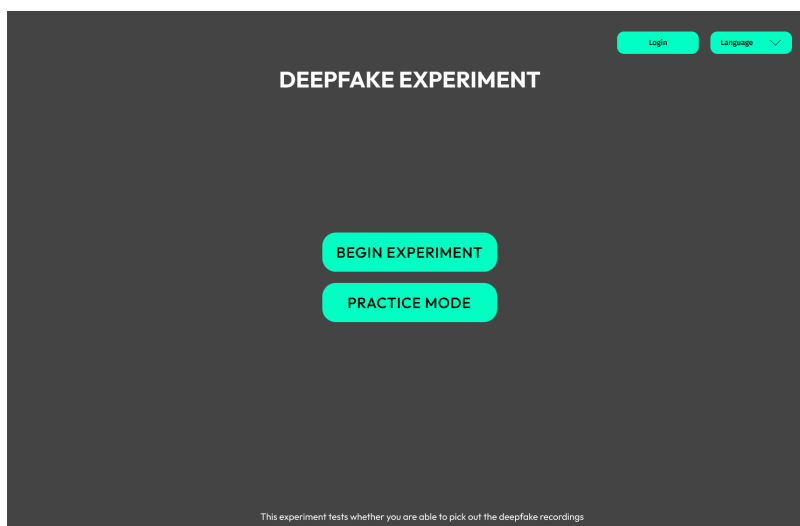


Figure 4.4: The initial screen design prompts the user to begin the deepfake detection experiment.

User data form screen

To obtain valuable data from this experiment, we require some information from the users. This information includes their age, gender, occupation, native language, and proficiency levels in other languages. Once users have submitted their data for the first time, they will receive a token that they can use to resume the experiment at a later time, shown in Figure 4.5.

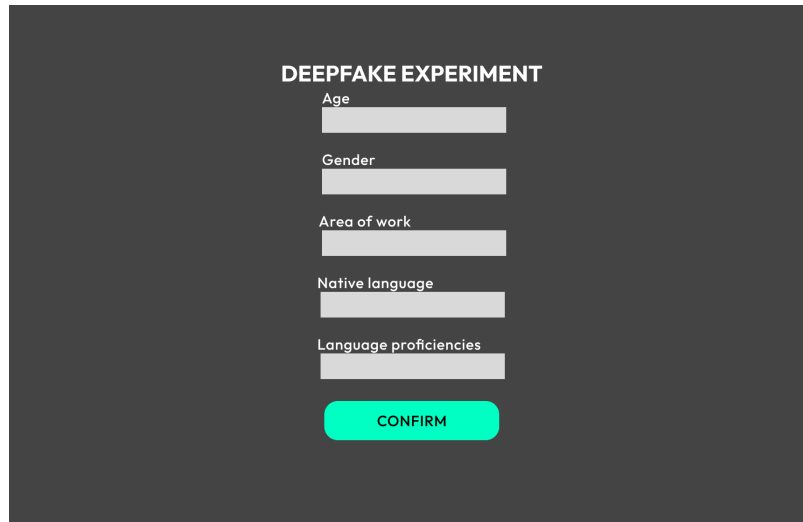


Figure 4.5: The user input form for collecting demographic information before starting the experiment.

User login

The user logs in using a randomly generated token as seen in Figure 4.6, created during their first data entry.

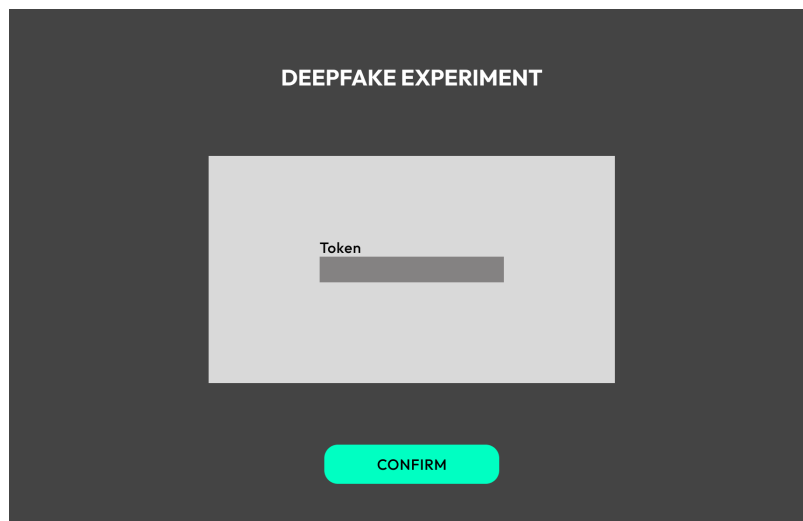


Figure 4.6: The screen for user login using the token.

Consent screen

For all applications that deal with personal information, a consent form must be provided to the user. This form should clearly state who has access to the data and the purposes for which it will be used. Users have the option not to accept the form, but they cannot participate in the experiment. The consent screen is shown in Figure 4.7.

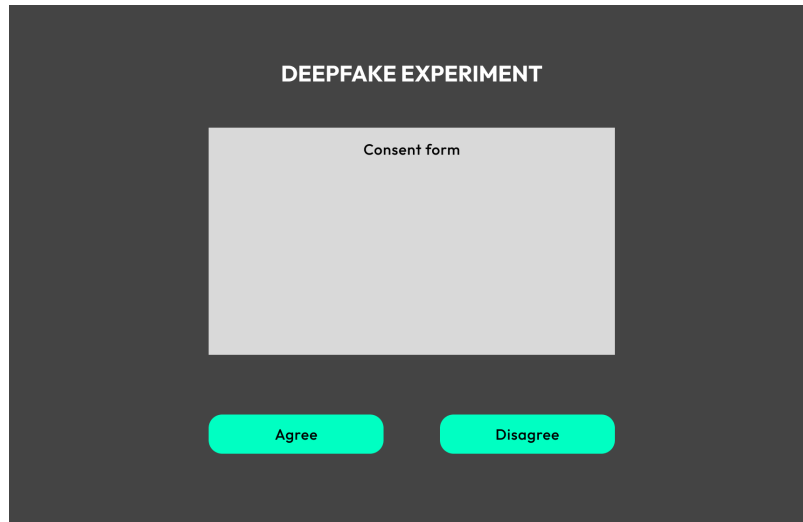


Figure 4.7: Consent form presented to users, requiring agreement before participating in the experiment.

Experiment screen

The experimental screen is designed, containing a recording and buttons for the user's response. A progress bar has also been included at the bottom to help track their progress during the experiment, shown in Figure 4.8, which gives users feedback on their actions and shows how many steps they have left. The double experimental screen is similar to the first one but with the addition of a second recording. Here, the user will be asked to decide which of the two recordings is real, shown in Figure 4.9. In both versions of the experiment, the user must select the device, which was used to listen to the recordings.

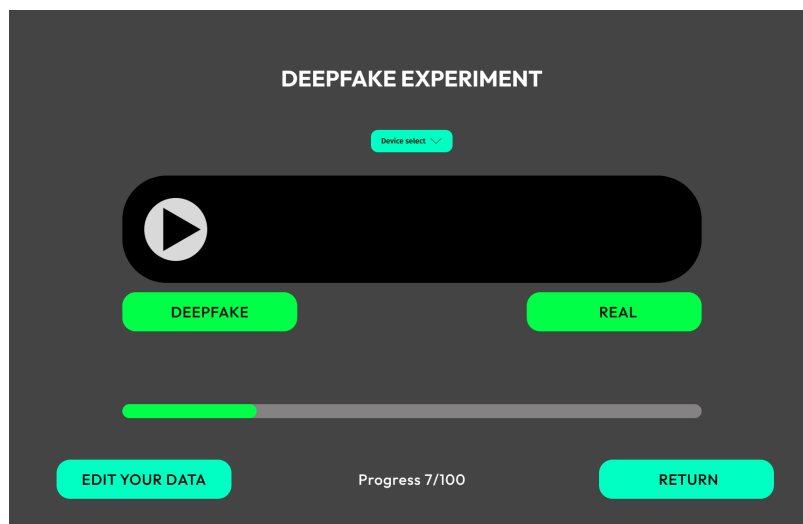


Figure 4.8: Experiment interface where the user determines if a recording is a deepfake or real.

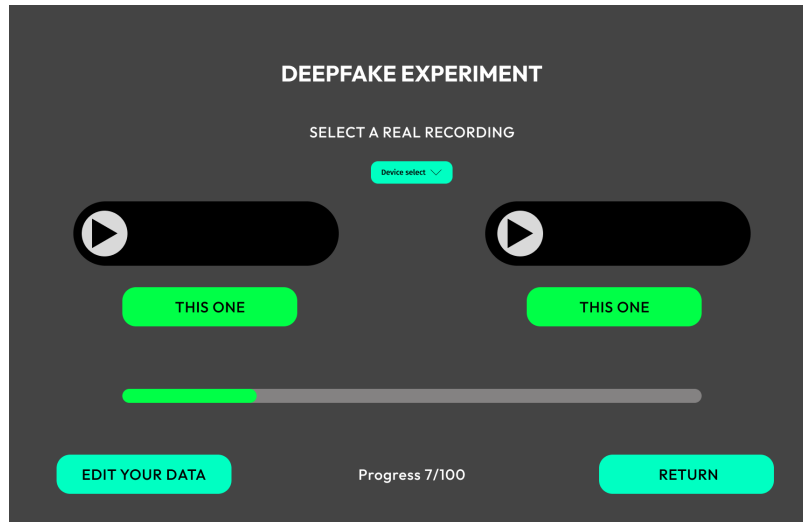


Figure 4.9: Double experiment interface where the user selects the real recording.

Practice mode screen

Practice mode is very similar to the normal experiment, except it gives the user feedback right away, the user does not have to be logged in and does not have to select the listening device as seen in Figure 4.10.

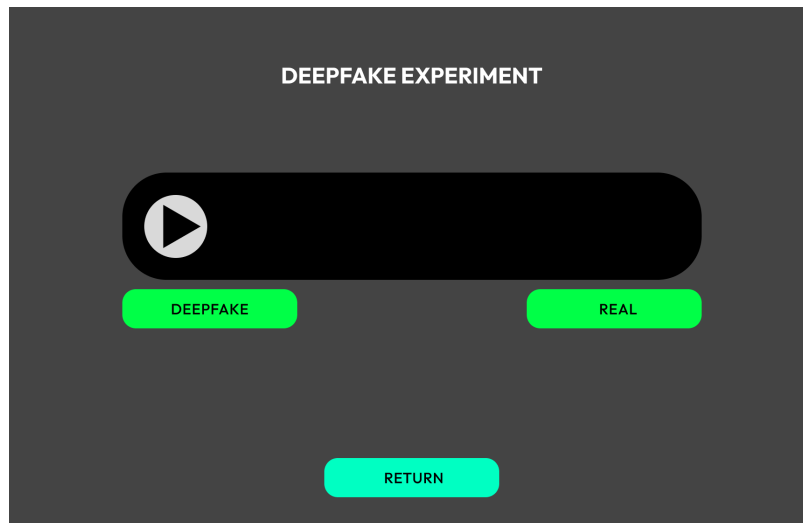


Figure 4.10: Practice mode interface

Experiment end screen

On the experiment end screen, shown in Figure 4.11, users have the option to start a new experiment, request results via email, or begin practice mode.

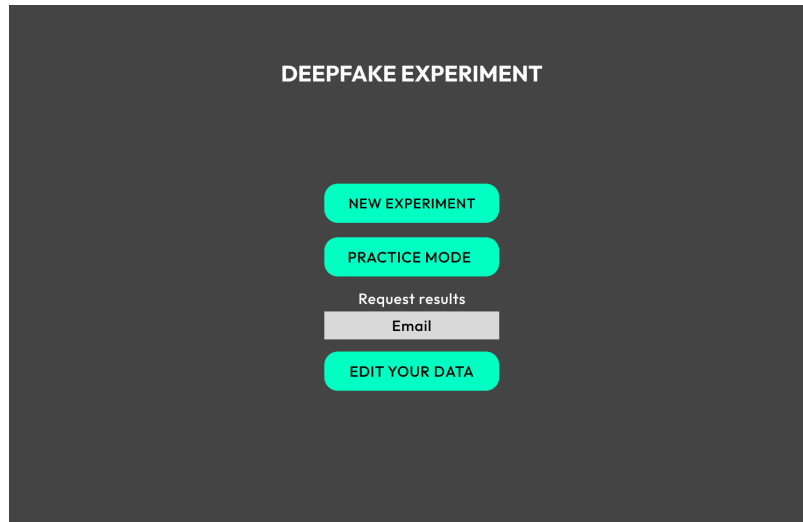


Figure 4.11: End of the experiment interface.

User account screen

The user account screen has been designed so that it allows users to request experiment results or delete their accounts. The layout is clear and straightforward, making it easy for users to navigate through options and take necessary actions related to their data, shown in Figure 4.12.

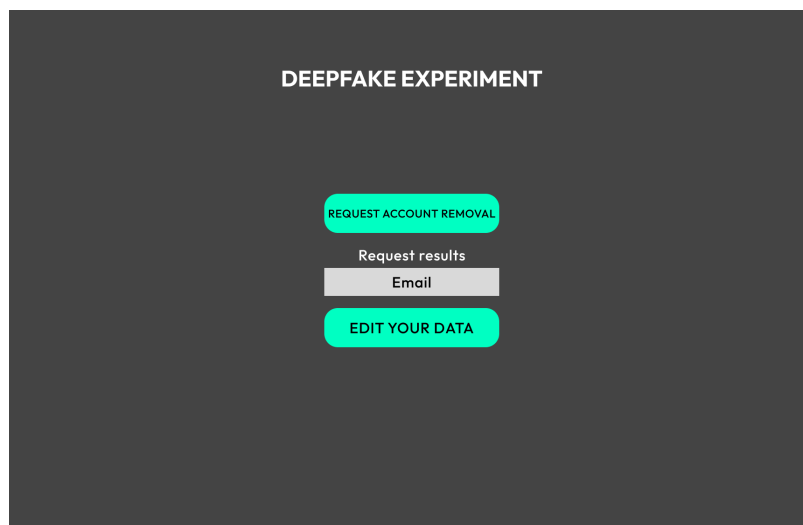


Figure 4.12: Interface for the user account with options to request results or account deletion.

Logged in user screen

In this screen, users can either continue their existing experiment from where they left off, start the practice mode session or edit their data, as shown in Figure 4.13.

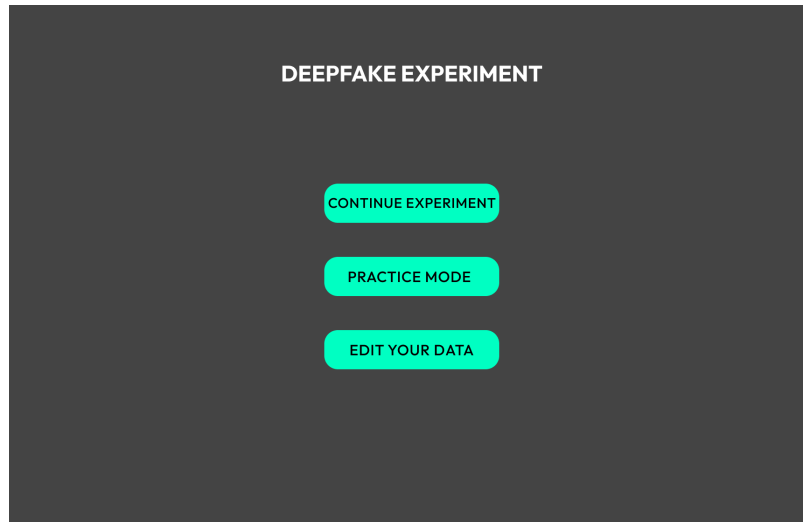


Figure 4.13: Options for the user to continue, start a new experiment or edit their data.

4.3.2 Admin part

The admin section should be designed to be efficient and user-friendly so researchers can focus on their experiments without getting lost in the app.

Admin login

The login interface for administrators has been designed to be both secure and easy to use. When logging in administrators must provide both the username and the password, making the login for administrators more secure, as seen in [Figure 4.14](#).

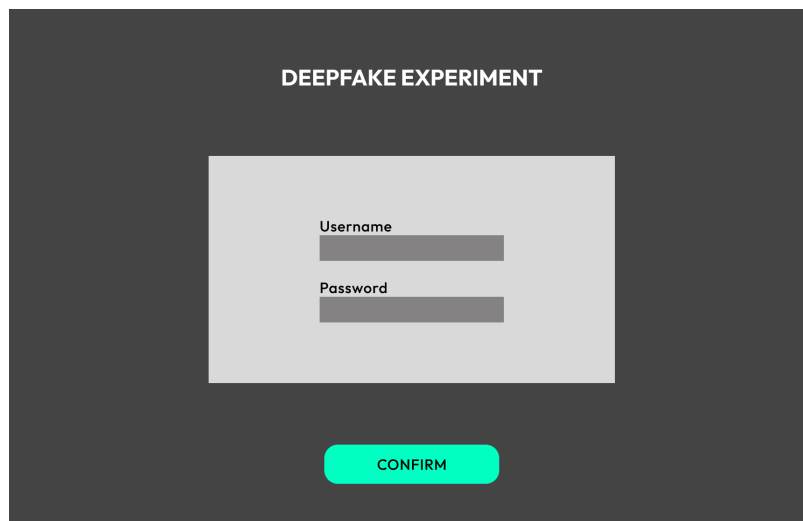


Figure 4.14: Interface for admin login.

Admin initial screen

The admin gets a choice of downloading user data, wiping user data or editing recordings. The get user data button can download data right away in CSV form or any other suitable form as seen in Figure 4.15. Wiping user data has been added as an easy mechanism to get rid of all the data and also in terms of GDPR if the user decides they no longer want to provide their data. The admin can also view a table of all users and filter them using filters.

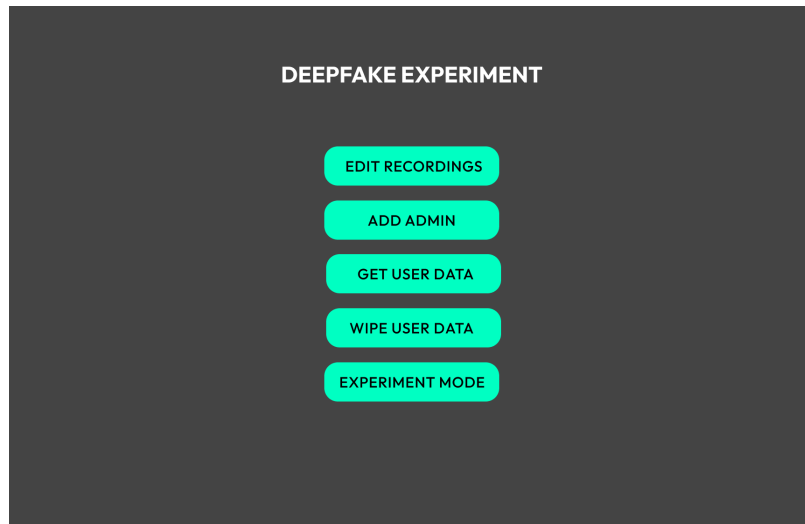


Figure 4.15: Admin panel for managing user data and experiment records.

Admin edit recording screen

On the edit recording screen, the admin can view all the recordings and perform various actions on them, such as changing their order, deleting multiple recordings simultaneously, and uploading multiple recordings at once seen in Figure 4.16. The admin can rearrange the order of the recordings by dragging them to a different position within the screen, changing their order in the experiment.

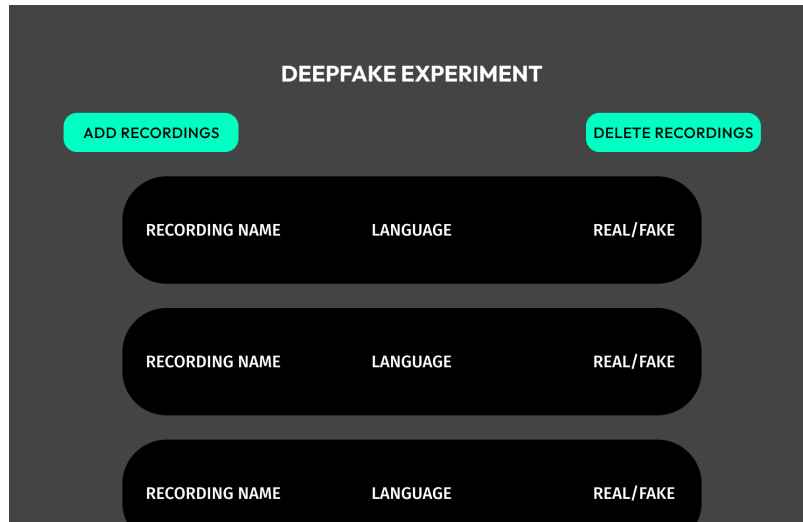


Figure 4.16: Interface for the admin to add new recordings to the experiment database.

Admin upload recording screen

On the admin upload screen, there is a typical file input field along with a confirmation button to upload the recording. The admin has the option to set the name of the recording, its language, and whether it is synthetic or not, see Figure 4.17. Additionally, the admin can upload numerous recordings simultaneously by uploading a file that contains a text file, which sets the parameters of the recordings and the recordings themselves, as seen in Figure 4.18.

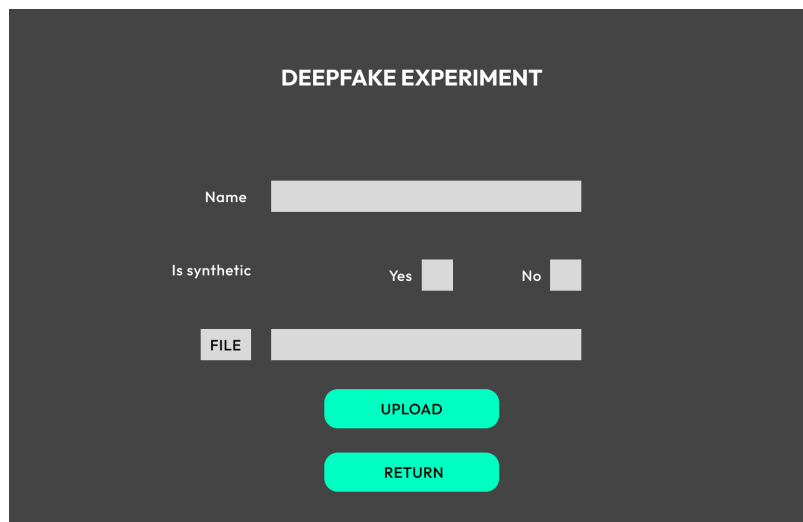


Figure 4.17: Interface for the admin to add a new recording to the experiment database.

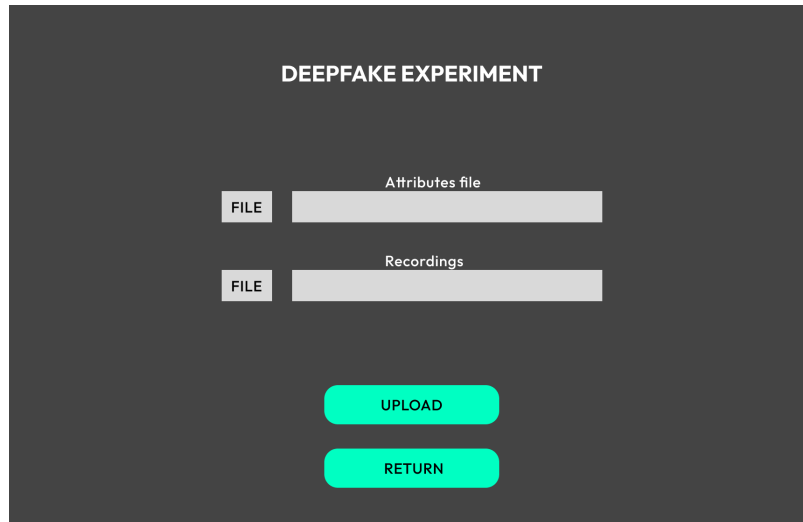


Figure 4.18: Multiple recordings upload page.

4.4 Storing data

Data in web applications can be stored in cookies, browser local storage and databases. This application stores data mainly in the database, session ID is stored in the session and active components for example language settings are stored in the browser's local storage. For database management, the MySQL RDBMS was chosen. In total, there are four tables in the database as shown in Figure 4.19.

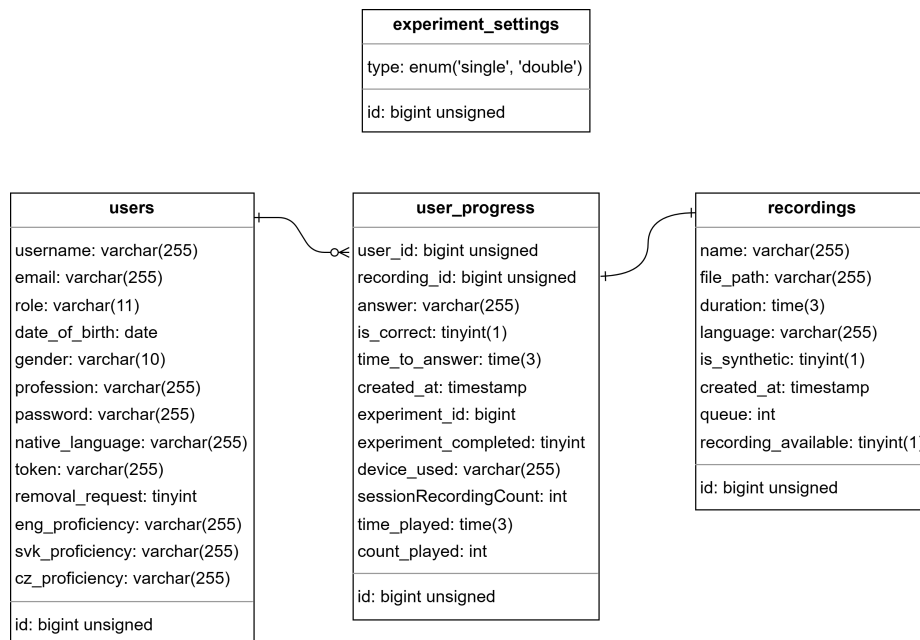


Figure 4.19: An ER diagram for this application

4.4.1 User table

In the user table, we need to store important data for the experiment to be usable for research, such as date of birth, gender, field of work, native language and proficiency in other languages. This information should provide a better view of which group of people have the hardest time detecting audio deepfakes. As seen in the user's Table 4.1.

User		
Attribute name	Type	Description
user_id	bigint	
username	varchar(255)	Unique username for admins.
token	varchar(255)	Unique identifier for users
role	varchar(11)	The role can be either admin or user. Used for granting access to actions and pages.
password	varchar(255)	Passwords are used for admin login.
date_of_birth	date	Date of birth of the user.
gender	varchar(10)	Gender of the user.
profession	varchar(255)	Area of work where the given user operates.
native_language	varchar(255)	Native language of the user.
removal_request	tinyint	Used for distinguishing which users want their account removed.
eng_proficiency	varchar(255)	Proficiency in English language
svk_proficiency	varchar(255)	Proficiency in Slovak language
cz_proficiency	varchar(255)	Proficiency in Czech language
email	varchar(255)	Users email where the results will be sent

Table 4.1: User Table

4.4.2 User Progress Table

The purpose of this table is to save the progress of the user so that if they leave the website, their progress is not lost. This table contains all the vital information about the user and the recording it pertains to. It records the ID of the user and the recording, the user's answer and the correct answer. The experiment ID is the ID of the user's current progress so that the user can have multiple experiments. The primary goal of the experiment is for users to discern whether each recording is genuine (real) or fabricated (synthetic). You can find this information in the user's progress in Table 4.2.

User Progress		
Attribute name	Type	Description
recording_id	int	Foreign key to link progress with recording.
user_id	int	Foreign key to connect the user with their progress.
answer	bool	User's answer is saved so that if they leave it will stay saved until they return to the experiment.
created_at	timestamp	Marks when the answer was created.
is_correct	bool	Saves information about the correctness of the user answer.
experiment_id	bigint(20)	Stores experiment id, experiment ID increments by one when a new experiment is created.
experiment_completed	bool	Marks completed or ended experiments.
device_used	varchar(255)	Stores what device did the user use to listen to the recordings.
sessionRecordingCount	int	Stores the number of recordings the user has answered in the current experiment.
time_to_answer	time(3)	How long did it take for the user to answer.
time_played	time(3)	Stores for how long did the user listen to the recording.
count_played	int	Stores how many times the user listened to the recording.

Table 4.2: User Progress Table

4.4.3 Recordings Table

The table named “recordings” contains important information about all the recordings such as their unique identification number, file path, language, and whether they are deepfakes or not as seen in Table 4.3. Since the recordings are stored on the server, the file path is required to locate them.

Recordings		
Attribute name	Type	Description
recording_id	int	
name	varchar(255)	Stores name of the recording.
file_path	varchar(255)	Stores path to the file and file name of the recording.
duration	time(3)	Stores the length of the recording.
language	varchar(255)	This part of the table stores information about the language used in the recording.
created_at	date	Date and time when the recording was uploaded.
is_synthetic	bool	Says whether the recording is a deepfake or not.
queue	int	Determines the order of the recordings in the experiment.
recording_available	tinyint(1)	Makes the recording available or unavailable in the experiment.

Table 4.3: Recordings Table

4.4.4 Experiment setting table

This table is used to store the setting of the experiment. The experiment can be done in the single or double version, determining how many recordings there are per step, either one or two recordings at once that need to be listened to and chosen which one is real, as seen in Table 4.4.

Experiment_settings		
Attribute name	Type	Description
id	int	
type	enum(single,double)	Stores the type of the current experiment.

Table 4.4: Experiment Settings Table

Chapter 5

Implementation

The system operates on a three-tier architecture, segregating the presentation layer, application layer, and database layer. Users interact with the application through the presentation layer, which runs directly in the browser. This chapter outlines the implementation of the proposed solution created in the design phase. The web application was implemented using Laravel 10.10 and Vue.js 3.2.36, integrating them to create a user-friendly and responsive interface utilising the best practices.

5.1 Front-end Development with Vue.js

The client side is implemented using the framework Vue.js written in JavaScript and HTML. The code is separated into three main parts [2]:

- **<script setup>**: This section uses the Composition API, which provides a flexible way to compose component logic. Composition API in script setup allows for better organisation and reuse of the code using JavaScript's capabilities. It houses the definition of reactive state, which is used to make primitive data types (like strings, numbers, booleans) reactive, computed properties, which automatically update their value based on changes to other data and methods used in the component.
- **<template>**: The template section defines the component's HTML structure, which creates the user interface. It uses declarative rendering to bind the component's data and methods to the view. It uses directives like **v-for** for rendering lists, **v-if** for conditional rendering, and **v-bind** for binding attributes.
- **<style scoped>**: Styling is used for defining the structure of CSS, which gives the application its looks, creating a visually appealing interface. It enables scoped styles that do not leak to other views and do not affect other parts of the application. It allows the creation of a unique look to every view, segregating styles like for example for lists of users and lists of recordings, which have different parameters and sizes to accommodate the data.

In this project, almost all Vue.js files are structured like this except the `Layout.vue` page, which serves as the root layout container for the entire application. This page imports all the global styles from `app.css`, which apply to every page of the application. It also integrates the `Header.vue` and `Footer.vue`, so that they are present on every page of the application. Lastly, it uses `<router-view>` which renders the child component based on the route inside of the parent component [3].

The Vue.js framework offers a modular approach to web application development, where the application is composed of many self-contained components. In this project, the Vue files were separated into different folders, depending if they were shared or not, which enhances maintainability and clarity within the project.

- **Pages:** This directory hosts the Vue components that represent full pages within the application, corresponding to different routes or endpoints. For instance, `Home.vue` serves as the landing page, while `About.vue` displays information about the application. Other components like `Admin.vue`, `Experiment.vue`, and `EditUserData.vue` represent distinct functional sections of the application, each encapsulating the logic and presentation for that specific part of the user experience.
- **Shared:** Common components reused across different application parts are placed within the `Shared` directory. These include layout components such as `Header.vue` and `Footer.vue`, which define sections of the UI that remain consistent throughout the application. The `Layout.vue` file contains the overarching layout used by other page components. Additionally, modal or dialogue components for example `NewRecordingModal.vue` that can be invoked from various points in the application are also found here.

5.1.1 Front-end Routing with Vue.js

Routing in Vue.js is an essential part of the user experience and the front-end architecture, allowing transition between components that represent pages or views of the application. Vue Router is used to define the navigation and the behaviour of the application. It interprets the browser's URL and maps the corresponding Vue components, allowing for a multi-page user experience within a single-page application framework. In this application, the Vue Router is configured with multiple routes that control which component should be displayed based on the current URL. Here is an example of how the routes are structured in this application:

- **Home Route:** The root path `"/` is associated with the `Home` component, which serves as the landing page of the application.
- **Dynamic Routes:** Paths like `"/add_recording` and `"/edit_user_data` dynamically load the respective components `Recording` and `EditUserData`.
- **Admin Routes:** Secured routes such as `"/admin` ensure that only authenticated users with administrative privileges can access the `Admin` component.

Each route is defined as an object with at least two properties: the path and the component to render. The Vue Router instance is then created by passing the array of route objects and configuring it to use the browser's history API, enabling clean URLs without hash symbols.

The router instance manages the rendering of components within the `<router-view>` element, which is used in the main layout and renders the pages, depending on the URL. Additionally, `<router-link>` elements are used in the application's navigation menu, enabling users to click links to navigate to different parts of the application without a page reload. In summary, Vue Router provides the tools for navigation in this Vue.js application, offering a user-friendly and efficient way to manage single-page applications.

5.1.2 Employed Libraries and Frameworks

The development of the client side uses several libraries and frameworks to improve the user interface, and functionality and streamline the development. Each library provides a comprehensive toolset for building a modern web application.

- **Axios:** Axios¹ is a tool that helps apps to talk to servers, sending and receiving data without waiting through asynchronous communication. It simplifies the process of sending HTTP requests and handling responses. In this application, Axios is used for operations like user authentication, data retrieval, and other interactions, like updating email, with the server-side API.
 - A `.vue` file makes a call to Axios with a specific API endpoint.
 - Axios sends the request to the server and waits for a response.
 - Once the data is returned, Axios updates the component’s state, triggering updates in the template if necessary.
- **Vue Router:** Another way of routing in Vue.js is used in this application is Vue Router², which also enables the mapping of URLs to components. It also uses the Router.js file in this application, but instead of being used in the HTML, it is used in JavaScript to for example redirect unauthenticated users, using `router.push('/')`; command. This improves the safety as the admin or the experiment page cannot be accessed via URLs and the user has to be authenticated to access these pages.
 - User requests a route (e.g., `’/home’`).
 - Vue Router fetches and renders the `Home.vue` component, providing a seamless navigation experience without reloading the page.
- **Pinia:** Pinia³ is used for reactive management of the data organising and sharing data across different components, replacing Vuex with a more flexible and less verbose API. In this application, Pinia is used to manage user data such as authentication and user preferences, across the client-side application. It is defined in the `userStore.js` file.
 - A component dispatches an action to fetch data using Axios.
 - Once Axios retrieves the data, the action commits a mutation to update the state in Pinia.
 - Components that are connected to this state will reactively update to reflect the new state.
- **Drag and Drop with Vue Draggable:** Vue Draggable⁴ is a library that adds a Vue component that provides the ability to create sortable lists and grids using native HTML5 drag-and-drop API. Its integration into the application allows administrators to intuitively reorder elements in a list through a drag-and-drop interface. In this application, Vue Draggable is used for reordering the recordings and changing their

¹<https://axios-http.com/>

²<https://router.vuejs.org/>

³<https://pinia.vuejs.org/>

⁴<https://github.com/SortableJS/Vue.Draggable>

queue parameter in the database, for the experiment. Administrators can click and drag the recording to the desired position within the list and the application will asynchronously update the order accordingly.

- A draggable component (from Vue Draggable) is included in the `.vue` file.
 - Users interact with the component, dragging items within a list.
 - Vue Draggable updates the list’s order in the component’s data, which can then be sent to the server via Axios or used locally.
- **Internationalisation with Vue I18n:** To support multiple languages the I18n⁵ library was utilised. It enables the application to switch languages seamlessly without page refresh. All translations are initialised in the `app.js` file and are later used in Vue.js files, either directly in the HTML or in the JavaScript sections.
 - The user selects a language.
 - Vue.js updates all the localised strings in the application accordingly.
 - Components re-render with the updated language settings.

5.1.3 Use of Best Practices for User Interface

Vue.js together with all the libraries mentioned create an easy-to-use and navigate interface, where importance was given to the best practices to create a front-end application. Here is a breakdown of how the best practices are used in this application. The application is responsive and offers users feedback for their actions with the help of visual indicators or messages. Every interaction within the application, from navigating between pages to submitting data forms, is managed through Vue components. Axios is utilised to send data asynchronously, ensuring that the application remains responsive and provides feedback to the user’s actions such as form submission or authentication request without a page reload. Consistency is achieved through `Layout.vue` and `app.css`, as mentioned before `Layout.vue` serves as the root layout container and also imports all styles from `app.css`. Users are prevented from making errors through validations on the front-end and displaying errors on the page. Multiple asynchronous tools are used for this together, Vue.js adds conditional rendering that is used in all forms in the code and displays errors if something is not filed in, together with Axios errors can be shown after any incorrect data is inputted to the database asynchronously without the need to refresh the page. The user’s memory load is minimised by always having the header and the footer visible so that they do not have to remember how to go to a different page, it also provides shortcuts to the user and makes the usage more intuitive. The interface is customisable thanks to I18n which can change the language of the application in an instant, all translations are stored in `app.js`. In summary, this application adheres to recognised UI design principles such as providing informative feedback, preventing errors, minimising cognitive load, and ensuring consistency and customizability. The output of front-end implementation can be seen in these Figures 5.1, 5.2, 5.3 and 5.4, which show the implemented user interface, showing the most important parts of the application. These screenshots showcase consistency, which is one of the best practices for user interface.

⁵<https://vue-i18n.intlify.dev/>

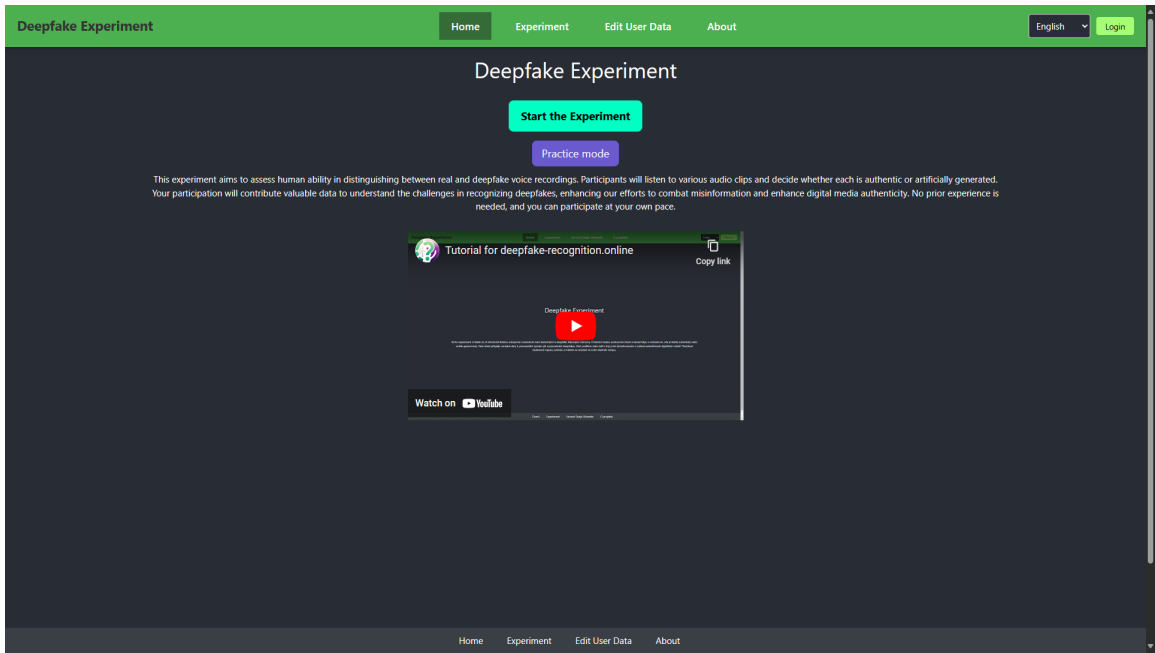


Figure 5.1: Screenshot of the home screen of the application.

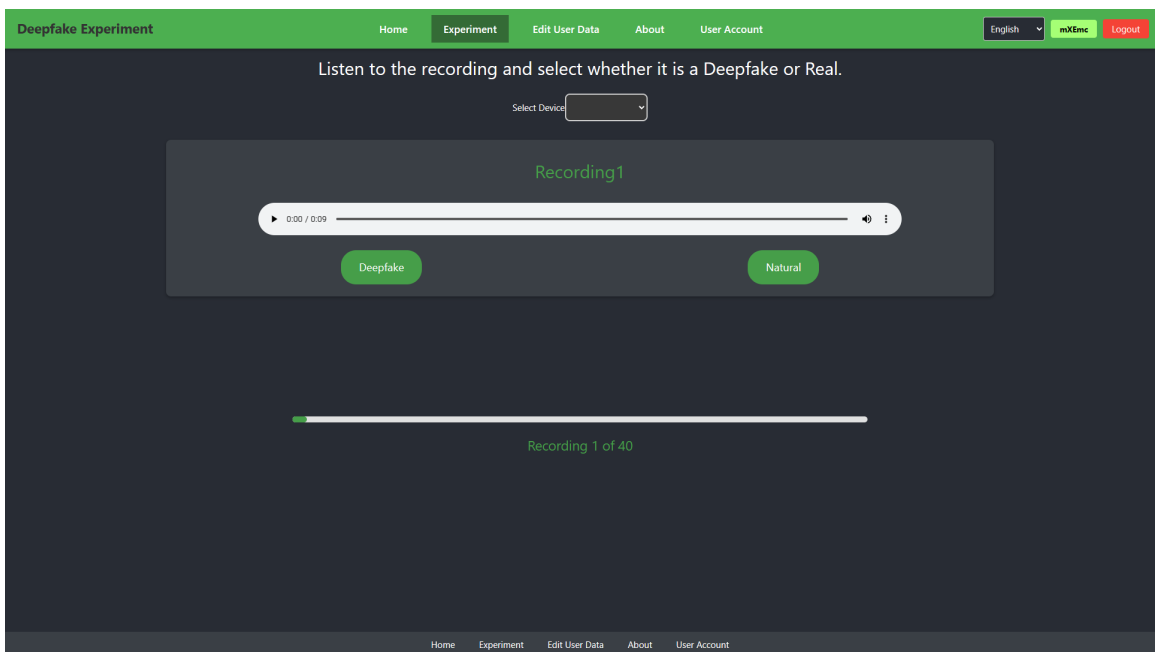


Figure 5.2: Implemented experiment screen with a single recording.

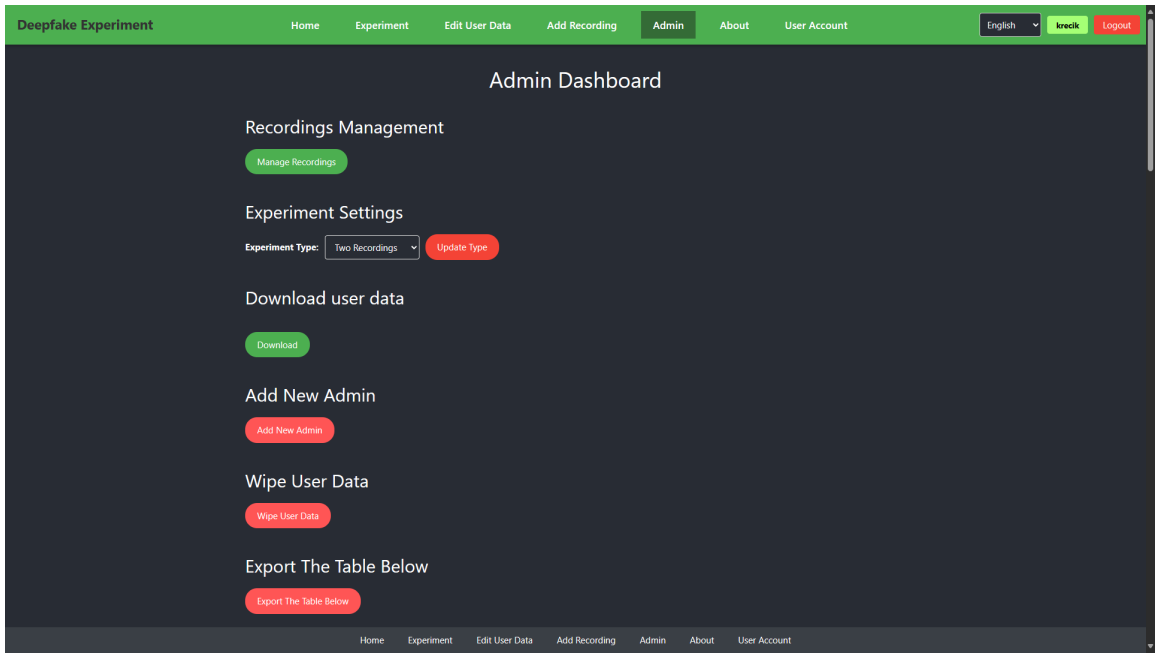


Figure 5.3: Implemented admin screen.

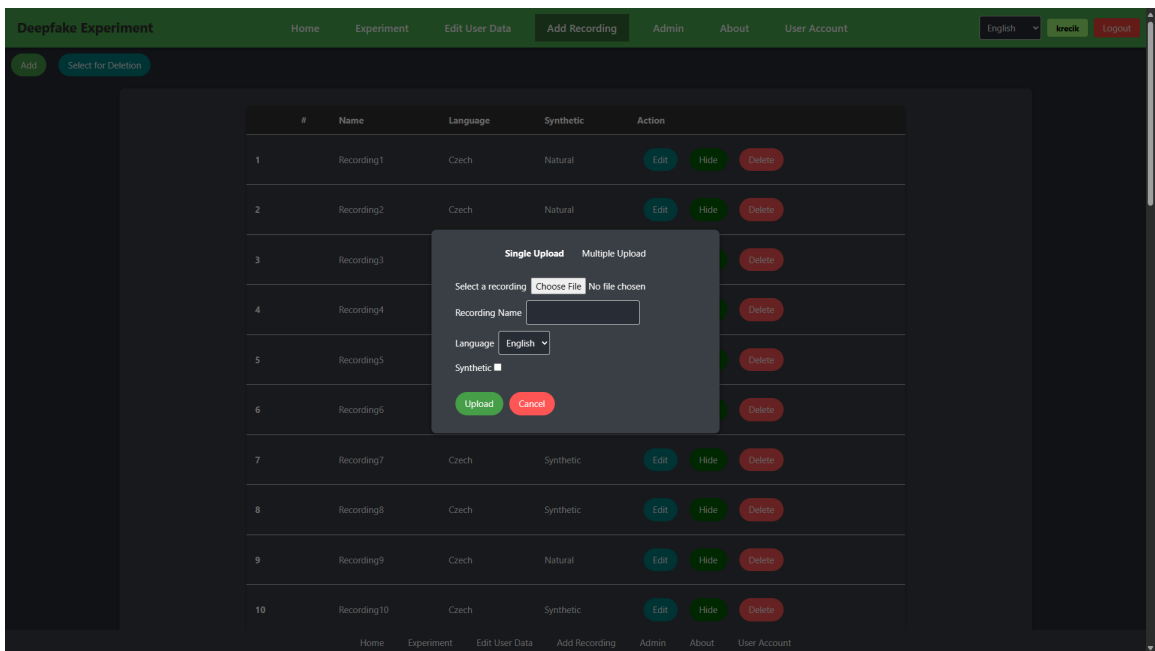


Figure 5.4: Upload screen from the application.

5.2 Back-end Development with Laravel

This section will discuss parts of the Laravel back-end that create this application together. The code is divided into **Controllers**, **Models** and **Migrations**, which are essential for structuring a robust and scalable web application.

5.2.1 Controllers

Controllers in Laravel are tasked with handling the business logic associated with various models and are responsible for making queries to the database.

Controllers connect the application front-end and the business logic on the server side. These controllers handle the user requests and process the necessary logic, they communicate with the database through models and return the response, typically in the form of a view or JSON data. Controllers are implemented within the `app/Http/Controllers` directory, controllers in Laravel follow a rule that each one should handle just one job. This helps to keep the code clean, easy to understand and manageable by segregating different aspects of the application logic. Another file in the `app/Http/Controllers` is `helperFunctions`, which is not a controller and was implemented so that code does not have to be repeated. It consists of functions used multiple times in the controllers, for example, to create a JSON error response. These are all the controllers implemented in this application:

Experiment controller: The main purpose of this controller is to manage the settings of the experiment, which is to update settings in the database, get settings from the database and complete experiments. The settings can be either single or double, meaning whether there is one recording per step of the experiment.

- `completeExperiment($experimentId)`: Marks an experiment as complete for a user, updating the experiment's status in the database.
- `updateSettings(Request $request)`: Updates the settings of the experiment in the database, depending on the previous setting.
- `getSettings(Request $request)`: Get the settings of the experiment, before the experiment starts the settings are sent to the front-end and depending on the value the corresponding number of recordings is loaded.

Recordings controller: These methods are for managing and loading recordings from the database.

- `index()`: Lists all of the recordings and is used in the add recording page, where recordings are listed and can be managed.
- `store(Request $request)` and `storeMultiple(Request $request)`: This application can either store one recording or multiple at once in the database.
- `update(Request $request)`: Recordings can be updated, and their name language and whether they are synthetic or not can be changed thanks to this method.
- `destroy($id)` and `bulkDestroy(Request $request)`: Similarly to store and store multiple methods, multiple recordings can be removed at once using this method. Even all recordings can be removed at once which makes the work of an admin easier.
- `toggleRecordingAvailability($id, Request $request)`: As opposed to removing the recording, the recording can be made unavailable, meaning that it won't show up in the experiment. This is done because if the recording is deleted, the user progress is deleted with it so no data can be extracted.

- `updateQueue(Request $request)`: This method is used together with the `vueDraggable` library, where by dragging a recording to a different position its attribute `queue` is changed in the database. This attribute is responsible for the order of the recordings in the experiment.
- `nextRecording($experimentId, $currentQueue)`: Fetches the next recording in an experiment sequence, depending on its queue number and whether it was already used in the experiment.
- `getRandomRecording()`: Retrieves a random recording from the database which is used in the practice mode, where the user does not have to be logged in and no data is stored. The practice mode is endless, but the recordings will repeat.
- `maxPairs()`: Calculates the maximal number of pairs of real and synthetic recordings. This is used in the double experiment, where there are two recordings at once and one of them is real and one is synthetic. Thanks to this the user can see a progress bar at the bottom.
- `totalRecordings()`: Gets the number of total recordings and is used in the single mode of the experiment to show the user a progress bar at the bottom.
- `checkRecording($id, $userGuess)`: Checks the user's guess to the actual answer which is stored in the database, and returns a value which is then stored in the user's progress table.

User controller: Manages user accounts and their data.

- simple get methods: `getUserName`, `getRole()`, `getUserId(Request $request)` are used in this application to retrieve data from the database, like admin username or token so that it can be displayed for the user when they are logged in. Getting role is used in all pages so that only data that is accessible to the user is displayed, for example, users do not have the option to see the administrative menus when they do not have the correct role. User ID is mainly used in the experiment page to create and store user progress under the user ID.
- `requestAccountRemoval(Request $request)`: Users can request their accounts to be removed. This will change the attribute `request_removal` in the user table to true and will show up on the admin page next to the user's ID.
- `update(Request $request)`: Used for updating user's information. It is also used if the user is creating their account and after they fill out their details the method generates a token which is later used to log in to the application the user is also automatically logged in after the first time. Administrators can also update their user account, but in addition, they can change their username, which users do not have.
- `updateEmail(Request $request)`: Used for updating email.
- `wipeUserData(Request $request)`: Admins can wipe all user data at once. which is a needed feature in this kind of project because of GDPR.
- `exportUserData()`: Admins can use this method to export all user data in a CSV format as seen in Table 5.1.

- `add(Request $request)`: This method is used for adding more administrators, only other already created administrators can do this action. Administrators that create new accounts also create their username and password, password is hashed and then stored in the database.
- `getUserData()`: This is used to retrieve user data from the database for the edit page and is inputted into the form so that users and administrators can see what data they inputted when creating an account rather than seeing an empty form.
- `logout(Request $request)`: Used for logging out, both for administrators and users.
- `login(Request $request)` and `loginUser(Request $request)`: Both methods are for logging in, the first one is for the administrators and works differently because administrators have a username and a password, rather than just a token like users.
- `index(Request $request)`: It is used in the administration page and lists all users in a table, the table has filters and can be downloaded, the downloaded table is in CVS format and the content depends on the filters. The table download is done through JavaScript.
- `deleteUser($userId)`: This method is used for administrators, they can remove accounts and all data about the removed user.

User progress controller: Tracks and manages the progress of users in experiments.

- `getSessionRecordingCount($userId)`: This method counts how many recordings has the user interacted with, which is also used in the progress bar.
- `store(Request $request)` and `storeDouble(Request $request)`: Saves user responses for single or paired recordings, together with `user_id`, `recording_id`, `experiment_id`, `time_to_answer`, `device_used`, `count_plated` and `time_played`.
- `loadExperiment()`: Loads the experiment depending on whether the user has an unfinished experiment or not. If the user indeed has an unfinished experiment, the method looks for the maximal queue number in the unfinished experiment using `experiment_id` so that the user can continue where they left off. Otherwise, a new experiment is started.
- `hasUnfinishedExperiment()`: This method is used on the home page of the application and returns true if the user has an unfinished experiment. If the method returns true the button is changed from start the experiment to continue the experiment.

5.2.2 Application programming interface

In a Laravel application, the web routes file `web.php`, defines the application programming interface (API) endpoints. Each endpoint is associated with a specific controller action, enabling the front-end to interact with the back-end through HTTP requests. The routes file connects HTTP verbs such as GET, POST and so on to the corresponding controller method.

The following Code 5.1 is an example of how the API routes are defined within the `web.php` file. The routes enable functionalities of the application, such as managing recordings, user authentication, and handling user progress within experiments. These routes can

either send a request to the controller method or directly respond to the request as seen in Code 5.1 in `/check-auth` route. Other API routes are paired with a controller and a specific method within that controller. For example, the route for getting all recordings is handled by the `index` method within the `RecordingsController` class.

```
Route::get("/recordings", [RecordingsController::class, 'index']);
Route::get('/check-auth', function () {
    return response()->json(['isAuthenticated' => Auth::check()]);
});
```

Listing 5.1: Controllers implementation example.

5.2.3 Models

Models in Laravel serve as the data layer within the MVC architecture, offering a way of communication with the database through an object-oriented interface. Each model represents a table in the database and provides operations such as data retrieval, insertion, updating, and deletion through console commands. In Laravel these models can be created through a command `php artisan make:model Recordings`. Models also define relationships between tables like foreign keys or one-to-one, one-to-many and many-to-many. Laravel uses Eloquent, an Object-Relational Mapping (ORM), to handle database interactions in controllers, for example, `$newRecording->duration = $duration;`. Eloquent has features like active record implementations, relationships, and mass assignment protection.

The application includes several models `ExperimentSetting`, `Recordings`, `User`, and `UserProgress`, which encapsulate the business logic for the database tables and define relationships between the tables. Below are the models used in this application and their purpose:

- **ExperimentSetting Model:** Manages the settings for experiments, with attributes like `type` for specifying the experiment's type. This model simplifies the manipulation of experiment settings in the database.
- **Recordings Model:** Contains details about recordings, such as their name, file path, language, synthetic status, availability, queue position, and duration. It also defines a `hasMany` relationship with `UserProgress`, connecting a recording with multiple user progress entries.
- **User Model:** Inherits from `Authenticatable` for user authentication functionalities. It includes attributes like date of birth, gender, profession, and language proficiency. The model defines a `hasMany` relationship to `UserProgress` to track the experiments' progress by the user.
- **UserProgress Model:** Represents a user's progress in an experiment, tracking answers, correctness, time taken to respond, and the play count of recordings. It establishes `belongsTo` relationships with both `Recordings` and `User`, linking progress records to specific recordings and users.

5.2.4 Migrations

Migrations in Laravel are used to make database changes. Migrations can be used for creating new tables, removing tables, adding, editing or removing columns. Migrations

ensure that database schema changes are consistent. Creating a migration in Laravel is straightforward with the Artisan CLI. For instance, to create a table to track user progress in an experiment, you can generate a migration file with this command `php artisan make:migration create_user_progress_table`. This command generates a new migration file in the `database/migrations` directory. Here is what a migration file might look like for creating a `'user__progress'` table.

```
return new class extends Migration {
public function up()
{
Schema::create('user_progress', function (Blueprint $table) {
    $table->id();
    $table->foreignId('user_id')->constrained()->onDelete('cascade');
    $table->foreignId('recording_id')->constrained()->onDelete('cascade');
    $table->string('answer');
    ...
});
}

/**
 * Reverse the migrations.
 */
public function down()
{
    Schema::table('user_progress', function (Blueprint $table) {
        $table->dropColumn('sessionRecordingCount');
    });
}
}
```

This migration defines a new table named `'user__progress'` with several columns to store users' progress through an experiment. It includes foreign keys to the `'users'` and `'recordings'` tables, creating relationships between these entities. The `'up'` method describes how to build the table, and the `'down'` method defines how to remove it if the migration is rolled back. To apply the migration and update the database schema, run command `php artisan migrate`. In this application database was created using migrations, so that application is portable if needed because when creating an application the database can be recreated using `php artisan migrate` command. This makes work with the database easier because no direct MySQL scripts have to be written.

5.3 Implementation Output

The output of the implementation is a fully functional and accessible web application, available at <http://deepfake-recognition.online/>, that allows users to participate in experiments involving listening to recordings and judging their authenticity, contributing to research. The application collects data as seen in Table 5.1. It also provides an admin interface for managing experiments, users, recording organisation, and data. The application design was made accessible using the best practices for UI design, It is built to handle

multiple users and manages data integrity with a robust backend, which also manages the user sessions. This Figure 5.5 illustrates the integration of the front-end and back-end of this application. It shows how user actions trigger data flows through the system. The diagram showcases the modularity and the three-tier architecture.

Table 5.1: CSV Data Column Headers

Column Name	Description
Identifier	Unique identifier for each user
Date_of_Birth	Date of birth of the user
Gender	Gender of the user
Native_Language	Native language of the user
English_Proficiency	Proficiency level in English
Slovak_Proficiency	Proficiency level in Slovak
Czech_Proficiency	Proficiency level in Czech
Field_of_Work	Field of work of the user
Experiment_ID	Identifier for the experiment
Recording_ID	Identifier for the recording
Recording_name	Name of the recording
Recording_Language	Language of the recording
Recording_Length (seconds)	Length of the recording in seconds
User_Answers	Answers given by the user
Time_to_Answer	Time taken to answer
Is_Correct	Correctness of the user's answer
Play_Count	Number of times the recording was played
Approx._Times_Played	Approximate number of times played
Time_Played (seconds)	Total time the recording was played
Device_Used	Device used for listening to the recording
Created_At	Date and time when the answer was created

Below is the description of the Figure 5.5:

1. **User Interface (UI):** This is where the user interaction happens, for example filling out forms or clicking buttons, which sends an HTTP request. The UI then updates dynamically and provides feedback based on responses received from the server. This creates a responsive user interface.
2. **Vue Router:** Manages the navigation by interpreting the URLs and directing the Vue.js to render the correct component based on this URL. The URL is based on user actions, like clicking a redirecting button which is linked to a different page.
3. **Axios:** Handles asynchronous HTTP requests to the server, without the need to reload a page. It sends requests and receives responses from the controllers through the API needed for user authentication, data retrieval and posting data to the server.
4. **API:** Interprets the requests from Axios routing them to the appropriate method from the controllers. Serves as a middleware between the front-end and the back-end. API also standardises the response from the controllers to ensure consistent data format.

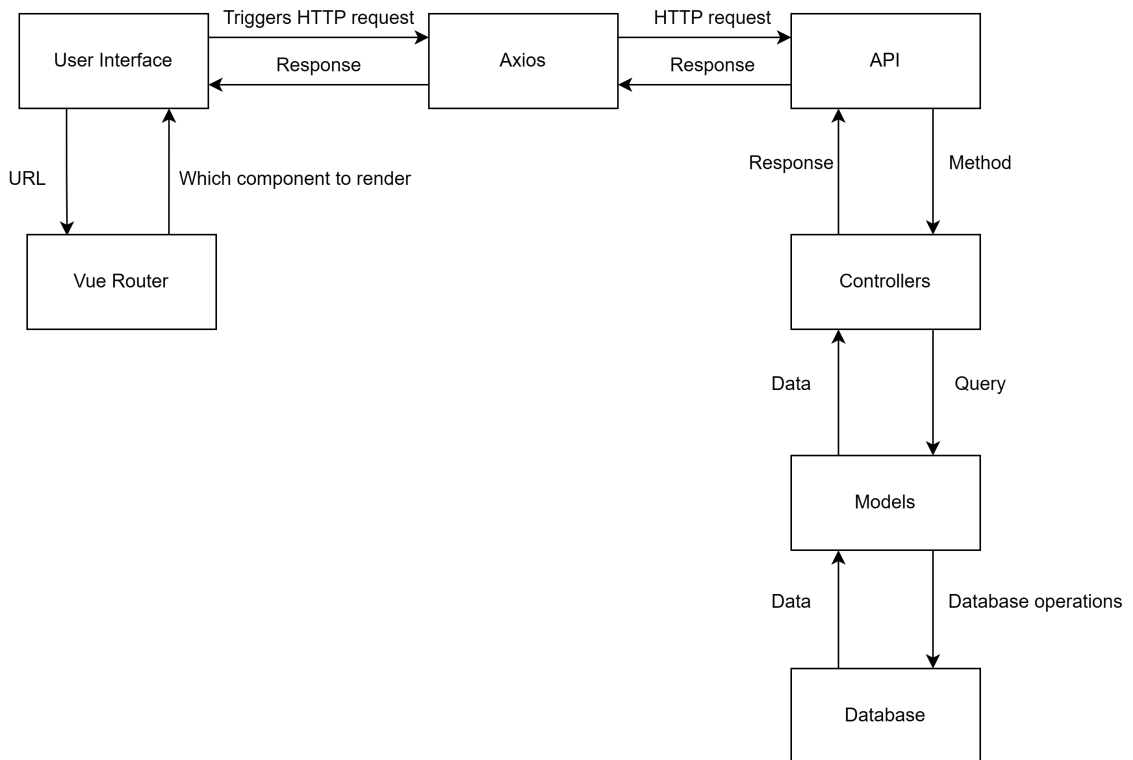


Figure 5.5: Communication between the components of the application.

5. **Controllers:** They are located on the server-side and are responsible for processing incoming requests. They perform actions requested by the front-end like querying or updating the database.
6. **Models:** These are responsible for direct interactions with the database, handling data retrieval, updates and other data manipulations. Models also enforce business logic related to data handling.
7. **Database:** Serves as a data storage of the application. Models interact with the database to fetch, update, insert and delete data at the request of the controller.

Chapter 6

Testing

Testing is an integral part of application development, not only to meet all the requirements for functionality but also to make the user experience better by getting feedback on the looks and expected behaviour of the application. This application was tested by tools, users and by demo run of the experiment. This chapter talks about the tools used for testing, evaluation of testing, user feedback, test experiment results and changes done after testing.

6.1 Database and controllers testing

Database and controller testing was done using two main tools: Insomnia for testing the controllers and DataGrip for database testing.

6.1.1 Insomnia

Insomnia¹ is a tool for testing APIs [4] without needing a front-end. It was used to simulate HTTP requests and view their response. Each endpoint in the controllers was tested for correct response. This included testing GET, POST, PUT and DELETE methods to ensure that the controllers handled all CRUD operations correctly. Controllers were also tested for handling input data correctly, checking if all the input rules worked.

6.1.2 Database testing with DataGrip

DataGrip² is an application that can be connected to a database and view the tables and their data in an easy-to-read format.

- **Data Integrity Checks:** Datagrip was used to run queries to verify the integrity of the constraints, such as foreign keys. This ensured that the database enforced data consistency automatically.
- **Viewing and Analysing Data:** Regular inspection of tables and relationships helped to see if the data was stored correctly. For example, date or time can be problematic and sometimes be stored in a different format than intended by accident.

¹<https://insomnia.rest/>

²<https://www.jetbrains.com/datagrip/?var=light>

6.2 User-based front-end testing

User-based testing is one of the steps to a successful application because it allows one to find mistakes in the design and the application back-end. Users might respond differently to the design than the developer because they are unfamiliar with the layout, which can be used to enhance the layout if the users are not pleased with the current state. The testing on this application was done after it was implemented and it was necessary to know if users knew how to work with the application. User testing involved five people from various age groups, aged 22, 52, 56, 21 and 26, to ensure accessibility. All test users were from Slovakia and knew the Czech language at least at the B1 level. The test users knew the English language at various levels, the older test subjects aged 52 and 56 knew English at A2 and A1 levels while the younger subjects aged 22, 26 and 21 knew English at C2, C1 and B2 levels. All participants used their own devices at home, which also tested the portability to different screen formats and browsers. The questions below were given to the user test group with no other instructions so that we get a realistic output of how difficult it was to perform these actions in the web application and their feedback was used to refine the user interface.

- **Task: Create an account**

- *Question: How would you rate the account creation process? Were the steps clear and understandable?*
- **Answers:** The feedback to this part was mostly positive, users found it easy and intuitive to create an account. The problems found in this part were that for some screens the GDPR pop-up that appears before creating an account was too big on some screens and users were not able to press the close button. Another problem was found while creating an account and filling in user data, the data was accidentally removed when the privacy policy button was pressed.
- **Fix:** There were some problems while creating an account and filling in user data as they were accidentally removed when the privacy policy button was pressed, which was easily fixed using `Pinia` to store data while the user can view the GDPR page and when they return the data will remain there. Another problem was for some screens the GDPR pop-up that appears before creating an account was too big, which was fixed by changing the CSS for the GDPR pop-up so that it would be scrollable.

- **Task: Perform the experiment**

- *Question: How would you rate your experience with the experiment? Were the instructions during the experiment clear?*
- **Answers:** All users responded positively to this part and no problem was found. The progress bar was praised for showing the progress at the bottom and the tutorial video for being informative while being relatively short.

- **Task: Change language**

- *Question: Were there any parts of the application that did not change the language? How would you rate the overall process of changing the language?*
- **Answers:** Again the response was positive to this part all users found it intuitive and easy to change the language. No problems with the translations were found.

- **Task: Request account removal**
 - *Question: How would you rate the process of requesting account deletion?*
 - **Answers:** This task was also completed successfully by all users. They had no problem finding the button on the user menu page.
- **Task: Request user data**
 - *Question: How would you rate the process of requesting user data?*
 - **Answers:** The response to this part was mostly positive. The older test users thought that after filling out the email the results would be sent right away after the experiment completion. Otherwise filling out the email was no problem for any test user.
 - **Fix:** Requesting the user data was a problematic part as some of the test users thought the data would be sent to them right away and not after the whole run of the experiment, which was easily fixed by adding more descriptive text about what was going to happen.
- **Task: Change user data (e.g., date of birth, native language, language proficiencies)**
 - *Question: How would you rate the process of editing your data? Was it easy to change your personal information?*
 - **Answers:** Editing data was also highly rated as users described it as simple to edit data because the process and the looks were similar to when creating a user profile. There was only one problem found with the ability to change the date of birth into the future.
 - **Fix:** The ability to change the date of birth into the future was fixed by adding a maximal date, which would update every day automatically to the current date, as a parameter to the HTML calendar.
- **Task: Log in and log out**
 - *Question: What were your experiences with logging in and out? Did you encounter any issues?*
 - **Answers:** This task got mixed responses as logging in was easy but logging out was not. The logout button was hidden in a drop-down menu which was done for aesthetic reasons, but it was harder to find.
 - **Fix:** The log-out button was changed as it was previously in a drop-down menu under the username, which was changed to a red log-out button in the right corner.

6.2.1 Evaluation of user testing

The response from the user-driven tests was positive, there were a couple of changes made to the layout, the size of the font and the colours to make the application as user-friendly as possible, based on the response from the test. Otherwise, creating an account for the users was easy as there was a tutorial video in case the test users did not know what to do. There were no problems while performing the experiment, as the experiment screen

is reactive and gives a warning if no device is selected. The test users responded to the application as a whole positively, even with small errors which were easily fixed. From the administrative point of view, there were no significant front-end problems. Still, there was a problem with the server which allowed for only 20 recordings per upload, which was fixed directly by changing the server settings. Otherwise, the application performed as expected. This user test proved to be valuable as some things went unnoticed while developing the application.

6.3 Test experiment

This section displays data collected from the application’s test experiment, showcasing various use cases. The application outputs data in the CSV format with headers, seen in Table 5.1. The test experiment was done on the same group of people as the user interface testing. There were 40 recordings of which 20 were deepfakes. The main purpose is to showcase what kind of data can be gathered from this application and also to test if the output is in usable format. This output can then be used to gather information as shown in the examples below. These data were handpicked to show the ability to gather data and were made easier to read than the actual data from which we can gather much more information.

Examples of the usage of the data

We can see what recordings were the most problematic in Figure 6.1, this could help us find what kind of software can produce the most realistic recording or the type of recording that is difficult to detect by listening. There is a natural recording called Recording7, which is the only one of the most commonly misidentified recordings that is not a deepfake. It provides valuable data that can help us interpret the problematic areas of detecting a deepfake. The recordings that are frequently misidentified might be used to test the robustness of deepfake detection algorithms. The software that can correctly identify these challenging samples would be considered more advanced. The graph was simplified to be readable in an A4 format and only the top 7 recordings with the most incorrect answers are shown out of the 40 recordings.

Another thing we can observe is the effect of language proficiencies on the performance of the users. This Figure 6.2 shows which proficiency level performed the best in the test experiment, where all the recordings were in the Czech language. This can help us identify the most problematic language, what level of the said language must the user know to perform the best and the effect of mastering a language. By comparing performance across different levels of language proficiency, linguists can gain insights into which aspects of a foreign language are more susceptible to confusion in the context of deepfake detection. This could have broader implications for language learning and automated language assessment tools.

Another interesting thing we can observe is which recordings were the most replayed in Figure 6.3. This can show us which recordings were problematic or needed to be replayed to detect whether the recording was a deepfake or natural recording. Recordings that are replayed often might suggest a higher cognitive load or interest. This can be an important factor in user experience research, indicating parts of content that require more attention or that engage users to a greater extent.

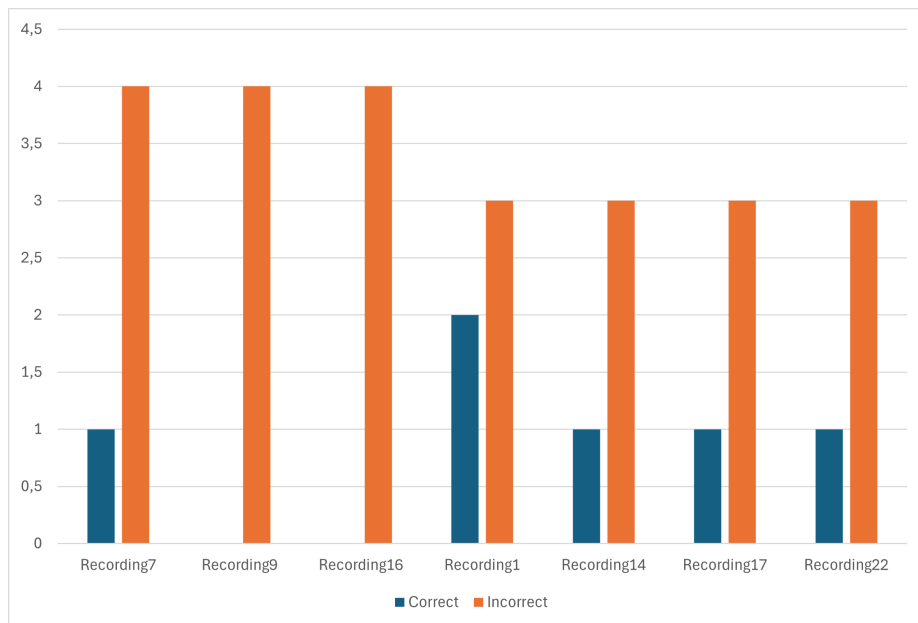


Figure 6.1: The simplified output of the test experiment showing the incorrect and correct answers for the most problematic recordings of the test experiment.

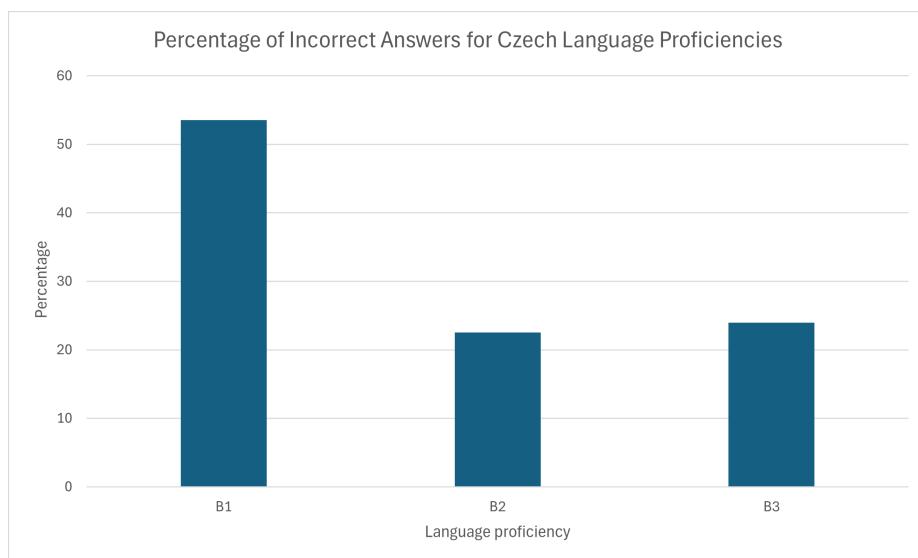


Figure 6.2: This graph showcases the percentage of incorrect answers for different Czech language proficiencies.

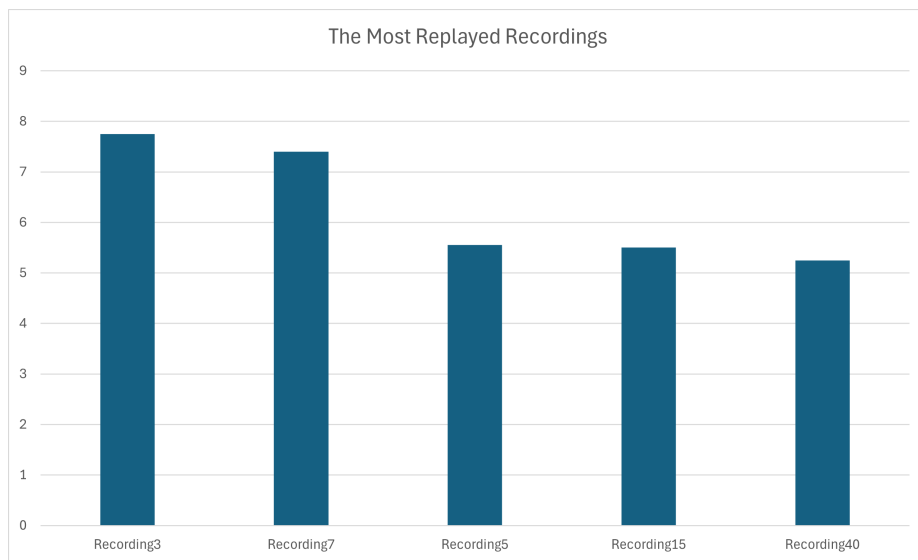


Figure 6.3: A graph that shows which recordings were the most replayed.

Chapter 7

Conclusion

The purpose of this thesis was to create a support tool for verifying the human ability to detect deepfakes using PHP and MySQL. The application provides user management, recording manipulation, and data export in CSV format. The main feature is that the application creates an interface for the users to partake in experiments.

The development process was described together with the choice of tools and the reason for their usage. Laravel's MVC architecture ensures a clear separation of logic and presentation, which is crucial for maintaining a large codebase. Vue.js was chosen for its reactive binding and component-based architecture.

The combination of Laravel and Vue.js frameworks provides a user-friendly and robust environment that manages the back-end processes and front-end interactions. The application was tested throughout the development process and at the end by users, showing that the application has the required functionality and visuals. Users were able to use the application easily and intuitively. The user interface is responsive providing the users with the necessary feedback to prevent errors and misunderstandings. Not only the goals of this project were achieved but the development also provided a learning experience in web development using modern technologies.

Future enhancements could focus on adding even more languages, a different way of adding recordings with deepfake detection, eliminating the need for an attributes file and optimisation of the application for multiple devices.

In conclusion, this thesis successfully demonstrated the capability to create a practical tool for data gathering, which could be helpful for research and can be extended further with additional improvements. Lastly, the experience gained from creating a complex three-tier application is invaluable and will assist in future development.

Bibliography

- [1] *How To Protect Against Deepfakes – Statistics and Solutions*. Visited 20.1.2024. Available at: <https://www.iproov.com/blog/deepfakes-statistics-solutions-biometric-protection>.
- [2] *Introduction*. Visited 20.11.2023. Available at: <https://vuejs.org/guide/introduction>.
- [3] *Nested Routes*. Visited 05.11.2023. Available at: <https://router.vuejs.org/guide/essentials/nested-routes>.
- [4] *Send Your First Request*. Visited 02.04.2024. Available at: <https://docs.insomnia.rest/insomnia/send-your-first-request>.
- [5] *Viral ‘Drake’ and ‘The Weeknd’ deepfake shocks industry experts*. Visited 20.12.2023. Available at: <https://www.cbc.ca/player/play/video/1.6817678>.
- [6] AMEZAGA, N. and HAJEK, J. Availability of Voice Deepfake Technology and Its Impact for Good and Evil. In: *Proceedings of the 23rd Annual Conference on Information Technology Education*. New York, NY, USA: Association for Computing Machinery, 2022, p. 23–28. SIGITE ’22. DOI: 10.1145/3537674.3554742. ISBN 9781450393911. Available at: <https://doi.org/10.1145/3537674.3554742>.
- [7] BENMOUSSA, K., LAAZIRI, M., KHOULJI, S., LARBI, K. M. and EL YAMAMI, A. A new model for the selection of web development frameworks: application to PHP frameworks. *International Journal of Electrical and Computer Engineering*. IAES Institute of Advanced Engineering and Science. 2019, vol. 9, no. 1, p. 695.
- [8] BUO, S. A. The Emerging Threats of Deepfake Attacks and Countermeasures. *CoRR*. 2020, abs/2012.07989. Available at: <https://arxiv.org/abs/2012.07989>.
- [9] BUSQUET, M. *Understanding Image Generation: Beginner’s Guide to Generative Adversarial Networks (GAN)*. Visited 20.12.2024. Available at: <https://blog.ovhcloud.com/understanding-image-generation-beginner-guide-generative-adversarial-networks-gan/>.
- [10] BUZZFEEDVIDEO. *You Won’t Believe What Obama Says In This Video!* [YouTube]. 2023. Available at: <https://www.youtube.com/watch?v=cQ54GDm1eL0>.
- [11] CASTRO, E. and HYSLOP, B. *HTML5 a CSS3: názorný průvodce tvorbou WWW stránek*. 7th ed. Brno: computer press, 2022. ISBN 978-80-251-5045-0.
- [12] CHANNEL 4 ENTERTAINMENT. *Deepfake Queen: 2020 Alternative Christmas Message* [YouTube]. 25. December 2020. Available at: <https://www.youtube.com/watch?v=IvY-Abd2FFM>.

- [13] CHATURVEDI, N. *Recurrent Neural Network with Keras*. Visited 02.01.2024. Available at: <https://medium.datadriveninvestor.com/recurrent-neural-network-with-keras-b5b5f6fe5187>.
- [14] CHAVAN, P. R. and PAWAR, S. Comparison Study Between Performance of Laravel and Other PHP Frameworks. *International Journal of Research in Engineering, Science and Management*. Oct. 2021, vol. 4, no. 10, p. 27–29. Available at: <https://journal.ijresm.com/index.php/ijresm/article/view/1420>.
- [15] CLARK, J. *Americans warned to ‘beware a flood of fake Trump mugshots’ powered by AI in advance of arraignment* [Fox News]. 4. April 2023. Available at: <https://www.foxnews.com/media/americans-warned-beware-flood-fake-trump-mugshots-powered-ai-advance-arraignment>.
- [16] COLLIDER LADIES NIGHT. *Deepfake Roundtable: Cruise, Downey Jr., Lucas & More - The Streaming Wars / Above the Line* [YouTube]. 11. November 2019. Available at: https://www.youtube.com/watch?v=l_6Tumd8EQI.
- [17] CURIE, D. H., JAISON, J., YADAV, J. and FIONA, J. R. Analysis on Web Frameworks. *Journal of Physics: Conference Series*. IOP Publishing. nov 2019, vol. 1362, no. 1, p. 012114. DOI: 10.1088/1742-6596/1362/1/012114. Available at: <https://dx.doi.org/10.1088/1742-6596/1362/1/012114>.
- [18] DEY, P., SINHA, B. R., AMIN, M. and BADKOOBEHI, H. Best Practices for Improving User Interface Design. *International Journal of Software Engineering & Applications*. september 2019, vol. 10, p. 71–83. DOI: 10.5121/ijsea.2019.10505.
- [19] DHALI, S. *Web application development with .NET : 3-tier architecture*. 2012.
- [20] DIETMAR, J. GANs and deepfakes could revolutionize the fashion industry. *Forbes*; <https://www.forbes.com/sites/forbestechcouncil/2019/05/21/gans-and-deepfakes-could-revolutionize-the-fashion-industry>. 2019.
- [21] FRENCH, A. M. Web development life cycle: a new methodology for developing web applications. *Journal of Internet Banking and Commerce*. Citeseer. 2011, vol. 16, no. 2, p. 1.
- [22] GAUR, L. *Deepfakes: Creation, detection, and impact*. 1st ed. CRC Press, 2023. ISBN 9781032139234.
- [23] GÉRON, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. 2nd ed. „ O’Reilly Media, Inc.“, 2022. ISBN 9781492032649.
- [24] GROH, M. *Detect DeepFakes: How to counteract misinformation created by AI* [MIT Media Lab]. Available at: <https://www.media.mit.edu/projects/detect-fakes/overview/>.
- [25] JENNY RUIZ, E. S. and SNOECK, M. Unifying Functional User Interface Design Principles. *International Journal of Human–Computer Interaction*. Taylor & Francis. 2021, vol. 37, no. 1, p. 47–67. DOI: 10.1080/10447318.2020.1805876. Available at: <https://doi.org/10.1080/10447318.2020.1805876>.

- [26] KADHEM, N. Efficiency Evaluation of Popular Deepfake Methods Using Convolution Neural Network. *Al-Nahrain Journal of Science*. Altaei, Mohammed. 2023, vol. 26, no. 3, p. 44–50.
- [27] KALUŽA, M., TROSKOT, K. and VUKELIĆ, B. Comparison of front-end frameworks for web applications development. *Zbornik Veleučilišta u Rijeci*. Veleučilište u Rijeci. 2018, vol. 6, no. 1, p. 261–282.
- [28] KANTOR, I. *The Modern JavaScript Tutorial* [online]. [cit. 2023-12-06]. Available at: <https://javascript.info/>.
- [29] KIETZMANN, J., LEE, L. W., MCCARTHY, I. P. and KIETZMANN, T. C. Deepfakes: Trick or treat? *Business Horizons*. 2020, vol. 63, no. 2, p. 135–146. DOI: <https://doi.org/10.1016/j.bushor.2019.11.006>. ISSN 0007-6813. ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING. Available at: <https://www.sciencedirect.com/science/article/pii/S0007681319301600>.
- [30] KUMAR, S. A Review on Client-Server based applications and research opportunity. *International Journal of Recent Scientific Research*. 2019, vol. 10, no. 7, p. 33857–3386.
- [31] LAAZIRI, M., BENMOUSSA, K., KHOULJI, S. and KERKEB, M. L. A Comparative study of PHP frameworks performance. *Procedia Manufacturing*. 2019, vol. 32, p. 864–871. DOI: <https://doi.org/10.1016/j.promfg.2019.02.295>. ISSN 2351-9789. 12th International Conference Interdisciplinarity in Engineering, INTER-ENG 2018, 4–5 October 2018, Tirgu Mures, Romania. Available at: <https://www.sciencedirect.com/science/article/pii/S2351978919303312>.
- [32] LAWSON, A. *How I created a deepfake of Mark Zuckerberg and Star Trek's Data*. Visited 2.11.2023. Available at: <https://arstechnica.com/science/2019/12/how-i-created-a-deepfake-of-mark-zuckerberg-and-star-treks-data/3/>.
- [33] LU, D. Dubbing with deepfakes. *New Scientist*. october 2019, vol. 244, p. 8. ISSN 0262-4079.
- [34] MARTIN, K. *Ethics of Data and Analytics: Concepts and Cases*. 1st ed. CRC Press, 2022. ISBN 9781000566260. Available at: <https://books.google.sk/books?id=E51kEAAAQBAJ>.
- [35] MIRSKY, Y. and LEE, W. The Creation and Detection of Deepfakes: A Survey. *ACM Comput. Surv.* New York, NY, USA: Association for Computing Machinery. january 2021, vol. 54, no. 1, p. 1–41. DOI: 10.1145/3425780. ISSN 0360-0300. Available at: <https://doi.org/10.1145/3425780>.
- [36] OLUWATOSIN, H. S. Client-server model. *IOSR Journal of Computer Engineering*. IOSR Journals. 2014, vol. 16, no. 1, p. 67–71.
- [37] PEROV, I., GAO, D., CHERVONIY, N., LIU, K., MARANGONDA, S. et al. DeepFaceLab: A simple, flexible and extensible face swapping framework. *CoRR*. 2020, abs/2005.05535. Available at: <https://arxiv.org/abs/2005.05535>.

- [38] PETROV, I., GAO, D., CHERVONIY, N., LIU, K., MARANGONDA, S. et al. DeepFaceLab: A simple, flexible and extensible face swapping framework. *ArXiv*. 2020, abs/2005.05535. Available at: <https://api.semanticscholar.org/CorpusID:218595801>.
- [39] RANA, M. S., NOBI, M. N., MURALI, B. and SUNG, A. H. Deepfake Detection: A Systematic Literature Review. *IEEE Access*. 2022, vol. 10, p. 25494–25513. DOI: 10.1109/ACCESS.2022.3154404.
- [40] RETZIUS, S. and SUNDHOLM, E. *Development of an evaluation model for client-side JavaScript Frameworks*. 2022.
- [41] SAHA, S. *A Comprehensive Guide to Convolutional Neural Networks: The ELI5 Way*. Visited 26.01.2024. Available at: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.
- [42] SAMRA, J. *Comparing performance of plain PHP and four of its popular frameworks*. 2015.
- [43] SERAFIN, T. Ukraine’s President Zelensky Takes the Russia/Ukraine War Viral. *Orbis*. 2022, vol. 66, no. 4, p. 460–476. DOI: <https://doi.org/10.1016/j.orbis.2022.08.002>. ISSN 0030-4387. Available at: <https://www.sciencedirect.com/science/article/pii/S0030438722000424>.
- [44] SHAMOOK. *The Mandalorian Luke Skywalker Deepfake* [YouTube]. 2023. Available at: <https://www.youtube.com/watch?v=wrHXA2cSpNU>.
- [45] SHTEFANIUK, Y., OPIRSKYI, I., IVANCHENKO, I. and PINDEL, K. DEEPFAKE–NEW TECHNOLOGY FOR IMPERSONATION IN CYBERATTACKS.
- [46] SISMAN, B., YAMAGISHI, J., KING, S. and LI, H. An Overview of Voice Conversion and Its Challenges: From Statistical Modeling to Deep Learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. Institute of Electrical and Electronics Engineers (IEEE). 2021, vol. 29, p. 132–157. DOI: 10.1109/TASLP.2020.3038524. Available at: <http://dx.doi.org/10.1109/TASLP.2020.3038524>.
- [47] STOUFFER, C. *What are deepfakes? How they work and how to spot them* [Norton Blog]. 1. November 2023. Available at: <https://us.norton.com/blog/emerging-threats/what-are-deepfakes>.
- [48] STOUFFER, C. *What are deepfakes? How they work and how to spot them* [Norton Blog]. Nov 2023. Available at: <https://us.norton.com/blog/emerging-threats/what-are-deepfakes>.
- [49] SUN, C., JIA, S., HOU, S. and LYU, S. AI-Synthesized Voice Detection Using Neural Vocoder Artifacts. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2023, p. 904–912.
- [50] TRIPATHY, S., KANNALA, J. and RAHTU, E. *ICface: Interpretable and Controllable Face Reenactment Using GANs* [arXiv:1904.01909v1]. 2019. Available at: <https://ar5iv.labs.arxiv.org/html/1904.01909v1>.

- [51] VERD, N. *Can You Trust Your Eyes Anymore? – The Rise of Synthetic Reality* [LinkedIn]. 9. May 2023. Available at: <https://www.linkedin.com/pulse/can-you-trust-your-eyes-anymore-rise-synthetic-reality-nicky-verd>.
- [52] VINCENT, J. *Tom Cruise deepfake TikTok videos highlight AI impersonator problems* [The Verge]. 5. March 2021. Available at: <https://www.theverge.com/2021/3/5/22314980/tom-cruise-deepfake-tiktok-videos-ai-impersonator-chris-ume-miles-fisher>.
- [53] WANG, R., JUEFEI XU, F., HUANG, Y., GUO, Q., XIE, X. et al. DeepSonar: Towards Effective and Robust Detection of AI-Synthesized Fake Voices. In: *Proceedings of the 28th ACM International Conference on Multimedia*. New York, NY, USA: Association for Computing Machinery, 2020, p. 1207–1216. MM '20. DOI: 10.1145/3394171.3413716. ISBN 9781450379885. Available at: <https://doi.org/10.1145/3394171.3413716>.
- [54] WANG, T. and CHOW, K. P. A Lightweight Reliably Quantified Deepfake Detection Approach. 2022.
- [55] WELLING, L. and THOMSON, L. *PHP and MySQL Web development*. Sams Publishing, 2003. ISBN 0-672-31784-2.
- [56] WESTERLUND, M. The Emergence of Deepfake Technology: A Review. *Technology Innovation Management Review*. november 2019, vol. 9, no. 11, p. 39–52. ISSN 19270321. Copyright - © 2019. This work is published under <https://creativecommons.org/licenses/by/4.0> (the“License”). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2023-11-12. Available at: <https://www.proquest.com/scholarly-journals/emergence-deepfake-technology-review/docview/2329154005/se-2>.
- [57] WILLARD, W. *HTML A Beginner’s Guide*. 2003.
- [58] WU, X., ZHANG, Q., WU, Y., WANG, H., LI, S. et al. F³A-GAN: Facial Flow for Face Animation With Generative Adversarial Networks. *IEEE Transactions on Image Processing*. 2021, vol. 30, p. 8658–8670. DOI: 10.1109/TIP.2021.3112059.
- [59] YADAV, D. Deepfake: A Survey on Facial Forgery Technique Using Generative Adversarial Network. In: *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*. Salmani, Sakina. Madurai, India: IEEE, May 2019, p. 852–857. DOI: 10.1109/ICCS45141.2019.9065881. ISBN 978-1-5386-8113-8.
- [60] ZERO MALARIA BRITAIN. *How we made David Beckham speak 9 languages* [YouTube]. 2023. Available at: https://www.youtube.com/watch?v=CF_e0kMCW2o.
- [61] ZHAO, S. *Azure AI milestone: New Neural Text-to-Speech models more closely mirror natural speech* [Microsoft Research Blog]. 17. December 2021. Available at: <https://www.microsoft.com/en-us/research/blog/azure-ai-milestone-new-neural-text-to-speech-models-more-closely-mirror-natural-speech/>.