ZhadigerAI: AI Services for Kazakh Language

Aidana Baglanova, Zilola Babakhojayeva, Nail Fakhrutdinov, Ruslan Kalimzhanov, Umit Azirakhmet
Advisor Prof. Adnan Yazici
Department of Computer Science
Nazarbayev University

***Abstract*—This project addresses the underrepresentation of the Kazakh language in modern AI services by designing and deploying a comprehensive web platform offering 11 AI services tailored for the Kazakh language, including ext-to-Speech, Speech-to-Text, Question Answering, Automatic Summarization, Machine Translation, Named Entity Recognition, Optical Character Recognition, Image Captioning, Large Language Model, KazCLIP, and VideoCLIP. The system features a React frontend and Spring Boot backend with secure user authentication and efficient resource management. For models serving we utilized Triton Inference Server framework. The platform follows the full cycle of designing, implementing, and evaluating a computing-based solution.**

***Index Terms*—Large Language Models, Text-to-Speech, Speech-to-Text, Question Answering, Automatic Summarization, Machine Translation, Named Entity Recognition, Optical Character Recognition, Image Captioning, KazClip, VideoCLIP**

I. Introduction

A. Overview of the project

THE rapid development of Large Language Models (LLMs) has introduced significant challenges for underrepresented languages like Kazakh. These languages risk being excluded from the digital world, which can lead to the erosion of cultural identity, limited access to modern technology, and slower local progress.

To address these issues, our team aims to create an accessible solution that enables the effective use of AI services in the Kazakh language. Our goal is to empower Kazakh speakers to benefit from modern technologies in communication, education, and content creation—without requiring a technical background.

Our project is a website that offers **11 AI-powered services** specifically designed for the Kazakh language. These services include:

- Text-to-Speech (TTS)
- Speech-to-Text (STT)
- Question Answering (QA)
- Automatic Summarization (AS)
- Machine Translation (MT)
- Named Entity Recognition (NER)
- Optical Character Recognition (OCR)
- Image Captioning (IC)
- Large Language Model
- KazCLIP
- VideoCLIP

The platform features a user-friendly and intuitive interface that allows users to interact with these tools easily. Additionally, it includes search history and multimodal search functionalities to enhance the overall user experience.

B. Objectives and goals

The project's technical objectives focus on building a robust and efficient AI platform that supports the Kazakh language and addresses a wide range of linguistic and technological challenges.

In the field of **Natural Language Processing (NLP)**, the project aims to implement tools such as NER, a Kazakh-specific LLM for coherent text generation, QA, AS, and MT.

Within the domain of **Computer Vision (CV)** and **Multimodal AI**, the platform includes tools such as OCR for digitizing printed and handwritten Kazakh text. It also integrates IC, KazCLIP, and VideoCLIP to enable image and video based content understanding.

To ensure performance and usability, the platform is optimized for high speed, low latency, and minimal resource consumption.

A key objective is to design an intuitive, user-friendly interface that abstracts away technical complexity while addressing technological and memory constraints. Users will interact with AI tools through simplified input-output mechanisms, and session data will be securely stored in a database to preserve data integrity and user privacy. Additionally, the platform includes a built-in search function and history tracking, allowing users to revisit previous interactions and efficiently access past results across all services.

II. Background/Related Work/Literature review

A. TTS and STT

In recent years, the field of NLP has seen considerable progress in addressing low-resource languages like Kazakh, with several projects contributing valuable datasets and models for enhancing AI capabilities in this language. For TTS and STT applications, KazakhTTS [1] offers a high-quality dataset, enabling the development of accessible voice-assisted technologies and other speech-based applications for Kazakh speakers. The authors utilize Tacotron 2 model, achieving 4.535 ± 0.049 and 4.144 ± 0.063 Mean opinion score (MOS) scores on F1 and M1 speakers. In total dataset contains over 90 hours of recordings, and 3 female and 2 male speakers.

The Kazakh Speech Corpus [2] contains approximately 332 hours of transcribed audio with over 153,000 utterances, spoken by individuals from various regions, age groups, and genders. It was reviewed by native Kazakh speakers to ensure

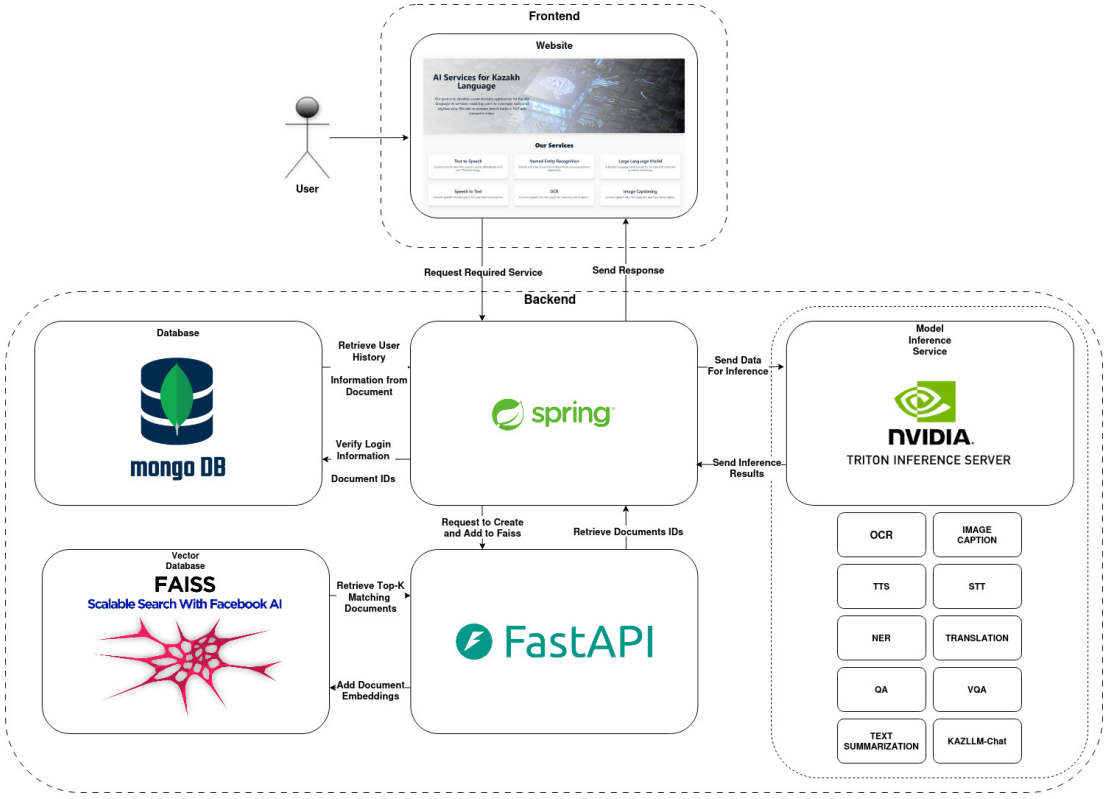

Fig. 1. System Design diagram.

high quality. The experimental results show that the audio and transcripts perform well, with a 2.8% character error rate and 8.7% word error rate (WER) on the test set. To support reproducibility and ease of use, an ESPnet recipe for speech recognition models was released.

B. QA, AS, and LLM

Key contributions also include KazQAD [3], an open-domain question-answering dataset tailored for Kazakh, which has facilitated the development of virtual assistants and other interactive applications. The dataset contains pairs of question and answers from the following categories: natural questions, Unified National Testing, and Kazakh Wikipedia. Additionally, a large portion of natural questions were translated from English NQ dataset using Yandex Translate API. In total dataset contains more than 60,000 question-answer pairs.

Moreover, the integration of LLMs, such as Irbis-7b [4] and LLaMA-1.9B [5] on Hugging Face, further extends Kazakh NLP capabilities by supporting diverse applications in text generation, translation, and other essential language tasks.

Key contributions in Kazakh automatic summarization include a hybrid method combining extractive and abstractive techniques. One notable work proposed using TF-IDF to identify key sentences, followed by sentence simplification via transfer learning with Seq2Seq and transformer models trained on a translated version of the Simple English Wikipedia corpus. The final Kazakh summarization model achieved a BLEU score of 7 and a SARI score of 36, demonstrating the feasibility of neural summarization despite limited resources [6].

C. MT

For the translation between different languages, KazParC [7] dataset provides textual sources for Kazakh, English, Russian, and Turkish. These sources contain wide range of different topics, such as literary works, language learning materials, TED talks, governmental and regulatory legal documents, publications and many more. In total dataset contains more than 370,000 parallel sentences. Using Tilmash (fine-tuned NLLB model [8]), authors achieved 0.21 and 0.32 BLEU scores on English to Kazakh and Kazakh to English translations, respectively. Another work [9] focuses on the effects of morphological structures of Kazakh language for machine translation. They evaluate their results on Tatoeba Translation Challenge [10].

D. NER

Additionally, KazNERD [11] provides a foundational dataset for NER task, aiding in the accurate extraction and classification of entities within Kazakh texts, which is essential for tasks like information retrieval and content summarization. These resources collectively aim to preserve Kazakh language and culture in digital spaces and support linguistic inclusivity. The dataset contains more than 112,000 sentences for 25 different classes. The authors were able to achieve F-1 score of 97.99% using BERT model [12].

E. OCR

OCR systems have seen significant improvements with deep learning techniques, especially CNNs and RNNs, which

enhance character and font recognition. Popular models like Tesseract, Easy OCR, and Paddle OCR use these techniques for higher accuracy and faster processing. i2OCR and MMOCR also leverage deep learning for competitive performance, while TradOCR focuses on accuracy despite slower processing times [13].

Algorithm	Accuracy	Precision	Recall	F1-Score
Easy OCR	0.6953	0.6931	0.7018	0.6942
Tesseract	0.6868	0.6847	0.6934	0.6858
Keras OCR	0.6784	0.6764	0.6850	0.6774
i2OCR	0.7037	0.7014	0.7103	0.7026
Paddle OCR	0.7136	0.7111	0.7201	0.7123
MMOCR	0.6643	0.6625	0.6709	0.6634
TradOCR	0.5982	0.5972	0.6048	0.5977

TABLE I
PERFORMANCE METRICS OF OCR MODELS.

F. IC

Furthermore, recent advancements in computer vision have led to the development of image captioning models for the Kazakh language, supporting tasks such as visual search, accessibility, and image-based question answering. One notable work introduces a Kazakh IC model based on ExpansionNet v2, trained on the COCO dataset with over 600,000 image-caption pairs translated into Kazakh. The model achieved 21.9 BLEU-4, 21.8 METEOR, and 81.3 CIDEr scores, demonstrating its potential for integration into accessibility applications, such as generating spoken image descriptions in Kazakh through text-to-speech systems. [14].

G. CLIP

Contrastive Language-Image Pre-training (CLIP) [15] is a method of training a pair of models from different modalities to align thier outputs in a vector space. The contrastive objective is used to train both visual and text models. The model is trained on large dataset of images and thier textual descriptions in order to link their representations. Because of good generalization the model is able to connect unseen images with textual descriptions. This ability is called Zero-Shot Learning. CLIP models achieves 76.2% accuracy on ImageNet dataset [16], which is similar to the performance of ResNet-50 [17], despite having no training on this dataset. Another work [18] has similar objective but tries to adapt this model for Azerbaijani language. The authors use ResNet50 and MultiBERT for image retrieval task using textual description. On MSCOCO dataset [19] they achieve 0.8 MAP score.

H. VideoCLIP

In the field of multimodal video understanding, recent work includes VideoCLIP [20], a contrastive pretraining method for zero-shot video-text retrieval. The model was trained on millions of video-caption pairs and evaluated on standard benchmarks such as MSR-VTT and YouCook2, achieving Recall@1 of 23.4% and 13.4%, respectively, in video retrieval tasks. Without requiring task-specific fine-tuning, VideoCLIP outperformed several supervised baselines, making it a strong candidate for low-resource language applications like Kazakh.

III. PROJECT APPROACH

A. Requirements and Features

1) Functional Requirements: The platform's design emphasizes accessibility, featuring a streamlined interface to facilitate intuitive interactions. Users can select an AI service, upload the desired input, and receive real-time outputs, creating a smooth and efficient workflow. To ensure secure access, user authentication is required, allowing only verified users to utilize the platform's services. User data privacy is prioritized through the secure storage of session data, safeguarding personal information and maintaining a record of request histories. This secure session data storage also supports a search feature, enabling users to retrieve previous interactions and continue their work as needed.

The platform offers a comprehensive suite of AI services, all tailored specifically for applications in the Kazakh language. Users can upload various types of input, such as text, audio, or images, through an intuitive interface, which then directs the input to the appropriate AI model for processing. Results are delivered back to users, ensuring a cohesive and user-friendly experience that requires no specialized knowledge to operate effectively. This structure of functional requirements ensures that the platform remains accessible, reliable, and specifically tailored to meet the needs of Kazakh-speaking users across various applications.

2) Non-functional Requirements:: To ensure a high-quality user experience, the platform is designed for fast response times and efficient handling of multiple simultaneous requests. As the user base grows, the system must be able to scale effectively, particularly for resource-intensive tasks, to maintain performance standards. Security is a top priority, with strict measures including secure authentication, session management, and data encryption applied both during data transmission and while at rest, protecting sensitive user information at all times. Additionally, to maximize efficiency, the platform optimizes GPU and CPU usage for AI model processing, ensuring resources are used effectively without unnecessary consumption.

IV. PROJECT EXECUTION

A. Front-End

The developed web application comprises several key components, including a landing page, a dashboard, and dedicated pages for various services. It also includes login/registration pages to manage user authentication. The front-end of the application was implemented using the React framework, chosen for its modular architecture and efficient rendering capabilities. Considering the state management, we decided to use Context API, which is suitable for user sessions. This approach is suitable for the scope of our project, given its simplicity and ease of use. Other solutions for state management, like Redux or MobX, are considered too complex and redundant for the complexity of our project. The interface was developed in two languages: English and Kazakh.

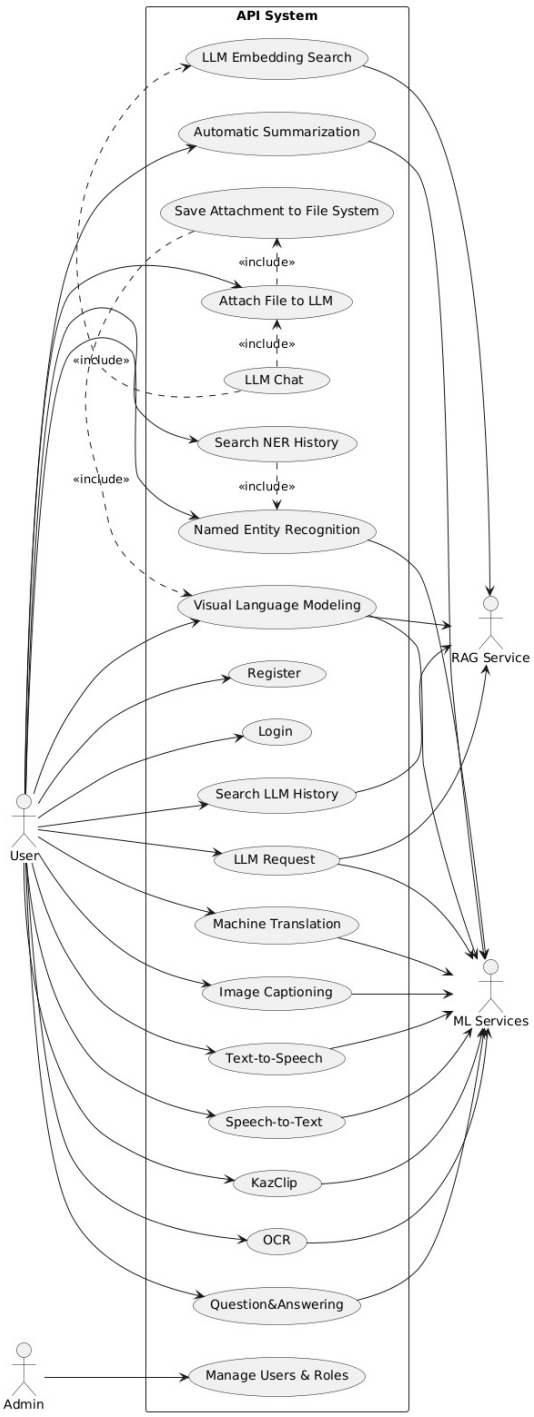

Fig. 2. Use case diagram

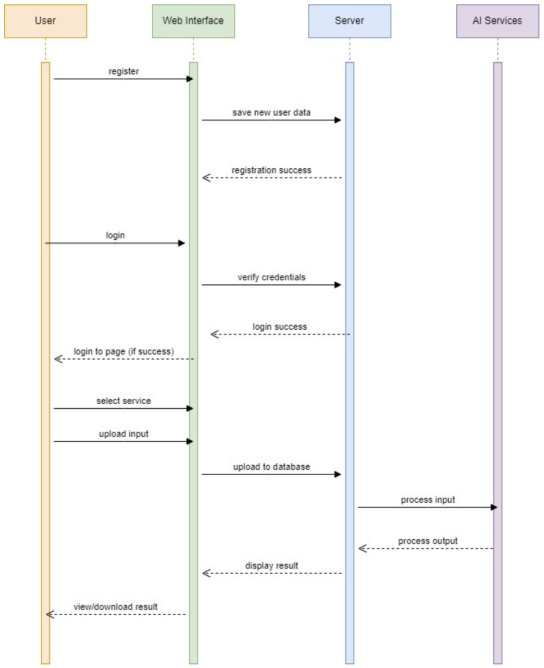

Fig. 3. UML sequence diagram

1) Libraries and Tools:

- Tailwind CSS: Used for utility-first CSS styling, enabling rapid and consistent user interface development.
- Axios: A library for handling HTTP requests, providing a straightforward way to interact with APIs.
- React Router: Employed to manage navigation and routing within the application.
- Formik: Implemented to streamline form handling and validation, ensuring a user-friendly and robust approach for managing input fields across the application.
- Yup: Integrated alongside Formik to provide schema-based validation for form inputs.
- Ant Design (AntD): Utilized as a comprehensive component library to build polished and consistent user interfaces.
- SockJS: Used to establish a WebSocket-like connection between the client and server, enabling real-time communication in environments where native WebSockets may not be fully supported.
- i18next: Integrated to support internationalization (i18n), allowing the application to serve users in multiple languages through a flexible and scalable translation system.

2) Implemented Pages: The following pages have been developed:

- Homepage: Serves as the landing page, providing an overview of the application and its features.
- TTS: A page dedicated to converting text into speech, that has required fields for text insertion and audio playing.
- STT: Facilitates the transcription of spoken language into text. Has a field for recording the voice and displaying the transcription.
- OCR: Extracts text from images for further processing. Has fields for uploading the documents and displaying the extracted text.
- IC: Generates captions for uploaded images. Has fields for uploading an image in different formats and displaying the captions.
- NER: Identifies and categorizes entities within text. Has two fields: for pasting the text and displaying the same text but with color-coded identified entities.
- LLM: Provides access to AI-driven language models for advanced text analysis or generation. This service consists

of a chat UI where the user can start a chat or choose from previous ones to create a query. The page also allows searching through chats by a keyword and displays the top 5 matching chats.
- KazClip: Service is dedicated to creating a query (description of the desired image) and retrieving five images corresponding to the user's description.
- Text Summarization: This service allows users to paste the text and retrieve the summary of that text.
- Question Answering: This page consists of the fields for text and a question regarding that text. The result of the query is the answer to the users question.
- Machine Translation: This service allows users to translate the text in several languages (English, Russian, Turkish, and Kazakh).
- VideoClip: This service allows users to make a query (description of the desired video) and receive three videos that correspond to that query.
- Login/Registration: Handles user authentication and registration securely, with forms managed using Formik, validated using Yup, and styled with components from Ant Design.

B. Backend

The project is based on the distributed monolith. We have the main project on Java Springboot, which as serves as the gateway API; while ML services are responsible for processing the information on other servers. The decision for this architecture is reasoned by the single point of logging and control over the database systems.

We have implemented logic for the authorization system with Spring Security, where we limit the access for API based on the user's role. The choice of database system was left on the MongoDB. MongoDB is a NoSQL database that uses a flexible, document-oriented data model. This flexibility allows us to store and manage various types of data without the constraints of a fixed schema. Since our platform handles diverse data types like text, audio, and images, MongoDB's ability to store JSON-like documents is particularly beneficial.

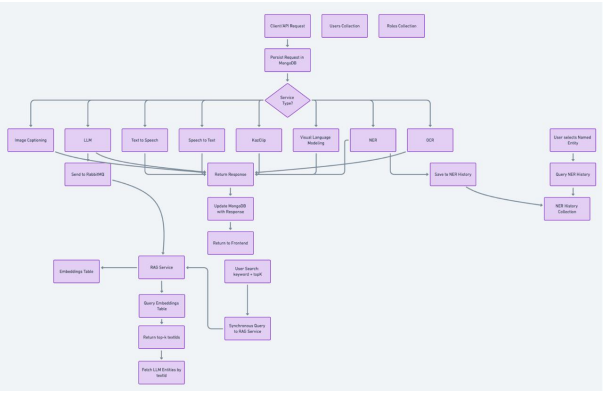

Fig. 4. Data-flow chart

Also developed the software solution for our project. We decided to use the NoSQL database due to its flexibility and to adapt to changing requirements quickly. For each service, we have separate document collection in our database with the logging information including time taken for the processing, and presence of errors. In addition, we collect the information about users for authorization purposes. The user document stores the data regarding their personal data, and authorization information. However, the logic of the authorization is happening with the Spring Security library to retrieve the information from the token and determine the access rights of the user. Another reason for the authorization process is making the requests more personal. It will be possible in the future to check the history of the requests. Also, our services take serious computational resources; that is why it would be reasonable in the future to connect payment system.

Our application has to work around with files a lot. That is why, we have integrated consistent pattern to work with file system. Storing all the data directly in the database is extremely resource-consuming and may degrade performance over the time as the number of files will grow. To alleviate the stress on the application, we store only metadata of the files in our database to retrieve it quickly from the separate package of actual file storage. That is an optimal way to balance between performance and complexity.

The processing of LLM request is done in 2 phases. Once the user submits the question for the LLM service, we have to give the response in a synchronous way using websockets fetching the response word by word. We have decided to make this decision because it makes our communication non-blocking; so, the user does not have to wait until the whole answer is ready. In addition, we had to implement search of user requests by semantics. It was a bottleneck at the beginning since the processing of request-response into embeddings takes considerable amount of time, deteriorating user experience. Our team came up with the solution to utilize message brokers, namely RabbitMQ. It turned out to be a very reasonable solution because the user does not need to get an instant response from our RAG service transforming and storing embeddings. We send to our message queue request-response pair for processing and, in case of search of messages within the LLM chats, we can obtain preprocessed results with ranking really fast.

C. MLOps

In order to make all models to work in the same development environment, we have selected Anaconda [21], package manager for Python. For neural network development, evaluation, optimization, and experiments, we have selected the PyTorch [22] framework. Together with ONNXRuntime [23] and TensorRT [24], which are used for the deployment of the models. ONNXRuntime is a high-performance inference engine for models that can be used on different hardware. TensorRT is an SDK made by NVIDIA specifically designed for efficient inference on NVIDIA GPUs. In addition, models have been deployed using the NVIDIA Triton Inference Server framework [25], to provide a RESTful API. Moreover, FastAPI [26] framework has been utilized to develop a microservice for RAG, which is used for the search system. Furthermore, the

models and their dependencies were packaged in a Docker container to ensure consistent deployment across different environments. Our current deployment setup leverages an 11GB NVIDIA RTX2080Ti GPU with 64GB of RAM. Due to the significant constraint in the GPU memory, we were forced to run most of the models on the CPU, therefore, their response time is not optimal but sufficient.

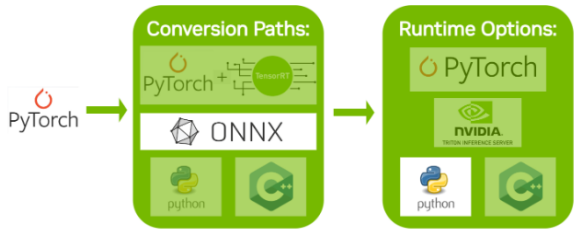

Fig. 5. Models export pipeline. Taken from [27]

V. Model Selection and Development

A. TTS

The first model developed for our platform was TTS. This task was prioritized due to its broad applicability across various fields. TTS technology can be used for facilitating social projects, particularly to support accessibility for visually impaired individuals by enabling screen readers and other assistive tools. It also has potential uses in language education, interactive voice response systems, and voice-enabled virtual assistants. For development, we collaborated with ISSAI to obtain a dataset comprising audio recordings of two male and three female voices, along with corresponding transcripts.

```
dataset/
├── male/
│   ├── speaker1/
│   │   ├── wav/
│   │   │   ├── file1.wav
│   │   │   ├── file2.wav
│   │   │   └── ...
│   │   ├── transcript.txt
│   └── ...
├── female/
│   ├── speaker2/
│   │   ├── wav/
│   │   │   ├── file1.wav
│   │   │   ├── file2.wav
│   │   │   └── ...
│   │   ├── transcript.txt
│   ├── speaker3/
```

Fig. 6. Dataset structure for text-to-sppech

We then conducted a comprehensive review of state-of-the-art TTS models using research literature and resources like Papers with Code. Based on this research, we selected models such as Tacotron2 [28], VTTS [29], HiFi-GAN [30], and FastSpeech [31]. These models vary in architecture, where FastSpeech use sequence-to-sequence framework to convert text to spectrograms, followed by vocoders like HiFi-GAN to generate natural-sounding audio. The fine tunning of this models was hard, as dataset are very large and we didn't had enough computational resources. However Tacotron2 had a hands on documentation on fine tunning on new language.

Model	Audio Quality MOS	Year
Tacotron2	3.52	2020
VITS	4.43	2022
HiFi-GAN	4.34	2020
FastSpeech	3.84	2019

TABLE II
Audio Quality MOS scores for selected TTS models.

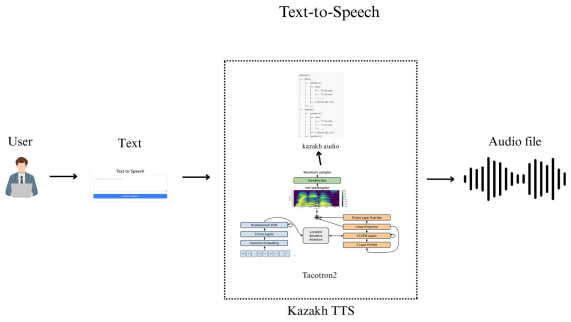

Fig. 7. Text-to-Speech model architecrute

B. STT

The next model in development is STT which is useful across various applications, such as transcribing audio for accessibility, enabling hands-free operation of devices, and supporting content creation through dictation. Most available models on the internet are paid; however, since we aim to provide an accessible service, we focused on open-source solutions. For this task, we selected the Transformers based model, fine-tuned with a Kazakh dataset by ISSAI [2].

Name	WER (%)	Total Words	Insertions	Deletions	Substitutions
valid	9.60	35275	451	496	2441
test	8.32	35884	482	334	2171

TABLE III
WER Results for Different Experiments

C. QA, AS, and LLM

The QA and AS functionalities in our system are powered by ISSAI's KAZLLM custom Kazakh large language model built on the LLaMA architecture. We utilized the quantized 4-bit 8B version of the model (checkpoints_llama8b_031224_18900-Q4_K_M.gguf), optimized for deployment in resource-constrained environments. These capabilities are served via a Triton Inference Server wrapper, where user inputs (text and task specification) are passed to the model along with a structured prompt in Kazakh. Depending on the selected task the model either provides a concise summary or answers a user query based on the provided context. The model is integrated using llama-cpp for

inference and supports streaming responses for low-latency output. This implementation allows us to offer high-quality, language-specific generative tasks with minimal overhead. Overall, we have tested two models: KazLLM and Sherkala. Their detailed performance comparison can be seen on Table IV. In the future we plan to integrate Sherkala model on our website as well.

D. MT

The machine translation functionality was developed using ISSAI's Tilmash model. Because of that our service supports four languages: Kazakh, English, Turkish, and Russian. The model performs as state of the art model even when compared to tools like Google translate and Yandex translate. Detailed results can be seen on Tables V and VI.

Language	Translator	BLEU	ChrF
English	Google Translate	**0.32**	**0.63**
	Yandex Translate	0.29	0.61
	Tilmash	**0.32**	**0.63**
Russian	Google Translate	0.26	0.59
	Yandex Translate	0.26	**0.60**
	Tilmash	**0.27**	**0.60**
Turkish	Google Translate	**0.21**	**0.58**
	Yandex Translate	0.13	0.52
	Tilmash	0.16	0.55

TABLE V
TRANSLATION FROM KAZAKH

Language	Translator	BLEU	ChrF
English	Google Translate	**0.27**	**0.63**
	Yandex Translate	0.18	0.58
	Tilmash	0.21	0.60
Russian	Google Translate	**0.21**	**0.60**
	Yandex Translate	0.20	**0.60**
	Tilmash	0.20	**0.60**
Turkish	Google Translate	**0.17**	**0.56**
	Yandex Translate	0.13	0.53
	Tilmash	0.15	0.55

TABLE VI
TRANSLATION INTO KAZAKH

E. NER

Next, we developed a NER model. NER is critical for extracting essential information and classifying main elements within text, such as names, organizations, dates, and locations, which can then support search and retrieval tasks. NER models are commonly constructed by adding a classification layer to a LLM backbone.

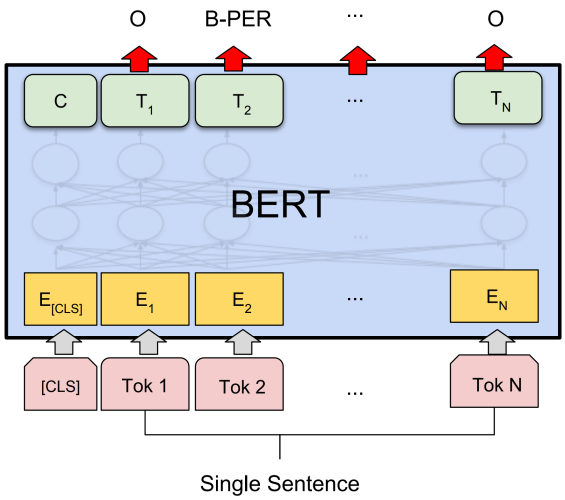

Fig. 8. NER model structure. [32]

For this project, we acquired a Kazakh-language NER dataset from ISSAI, which was originally translated from English. The models we selected for evaluation were a fine-tuned BERT and a BERT+span model, both tailored to handle the nuances of Kazakh. Performance was measured using the F1 score, with the fine-tuned BERT achieving 96.1% and BERT+span reaching 95.41%.

Model	F1 Score (%)
Fine-tuned BERT	96.10
BERT+span	95.41

TABLE VII
COMPARISON OF NER MODELS.

F. OCR

The OCR model is essential for recognizing and digitizing text from images. OCR technology enables users to convert scanned paper documents into digital format, extract text from photographs for translation, digitize invoices, process documents for searchable digital archives, and even read information from identification documents. In reviewing OCR models, we examined research papers that frequently mentioned PyTesseract [33], PaddleOCR [34], SuryaOCR [35] and EasyOCR [36] as effective tools. These models offer support for Kazakh, their current accuracy varies and presented in next table. The EasyOCR and PaddleOCR were not fine tuned because of outdated trainning code which raised a lot of issues, the PyTesseract did not outperform SuryaOCR barely acheiving 80%. We used SuryaOCR model for our solution as for it doesn't require additional fine-tunning.

Model	Accuracy (%)	Processing Speed (Rank)
PaddleOCR	72.9	2
EasyOCR	46.3	3
PyTesseract	18.7	1
SuryaOCR	92.7	4

TABLE VIII
PERFORMANCE COMPARISON OF OCR MODELS FOR KAZAKH TEXT.

Model	AVG	Knowledge			Commonsense Reasoning						Misinfo. & Bias			
		KazMMU	MMLU	Belebele	HS	PIQA	BoolQA	SIQA	ARC	OBQA	NIS	COPA	T-QA	CS-Pairs
LLaMA3.1-KazLLM-1.0 (8B)	43.7	37.0	31.5	**27.8**	46.0	62.8	**69.8**	**44.7**	35.5	34.2	**32.0**	50.4	**50.9**	45.0
Sherkala (8B)	**45.7**	**51.6**	**37.7**	25.9	**53.1**	**68.1**	66.9	42.2	**38.1**	**37.0**	18.0	**51.0**	50.3	**54.3**

TABLE IV
PERFORMANCE COMPARISON OF KAZLLM AND SHERKALA MODELS

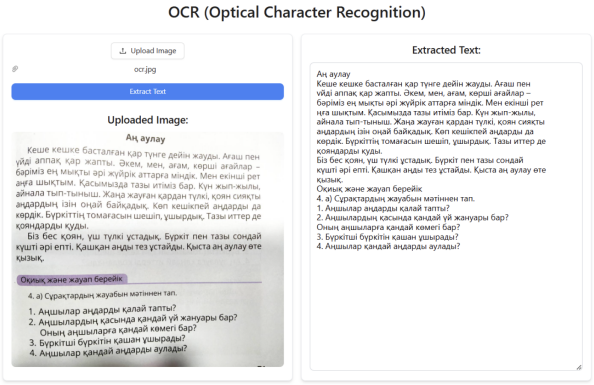

Fig. 9. Website view of OCR

Also, the document intelligence pipeline was implemented by combining OCR with LLM, this enables users to upload several documents and images and ask questions on them.

From a technical perspective, if the uploaded files are in PDF format, we first convert each page into individual images. These images are processed through the OCR model to extract raw text. The extracted text is then split into manageable batches, and embeddings are precomputed and stored.

When a user submits a query, it is also embedded and compared against the stored document embeddings using semantic similarity. The most relevant text segments are selected, and the LLM is used to generate a human-readable answer based on the retrieved content.

G. IC

For the Image Captioning (IC) module, we adopted the pretrained ExpansionNet-v2 model for Kazakh-language captioning [14]. To enable its integration into our platform, we converted the model to ONNX format and deployed it using the Triton Inference Server framework. This setup allows efficient and scalable real-time caption generation from user-uploaded images. Given the model's strong performance on captioning tasks, we selected it without further fine-tuning, ensuring compatibility with our system's resource constraints while maintaining high output quality.

Model	BLEU-4	METEOR	CIDEr
ExpansionNet v2 (en)	41.0	30.2	139.6
ExpansionNet v2 (en-kk)	20.0	20.8	71.8
ExpansionNet v2 (kk)	21.9	21.8	81.3

TABLE IX
EVALUATION RESULTS OF EXPANSIONNET V2 ON KAZAKH AND ENGLISH CAPTIONS USING THE COCO DATASET.

Its accuracy on the COCO "Karpathy" test split was measured at 21.9 BLEU-4, 21.8 METEOR, and 81.3 CIDEr, making it a strong choice for our real-time captioning pipeline.

H. CLIP

Developed functionality allows users to retrieve images from the database using textual description on the Kazakh language. The model has been trained on the same dataset that was used for the image captioning task. During the testing of the model, we found that due to the fact that significant part of the machine translated captions were not accurate, the model performs not good with the average MAP of 0.67 on test dataset. We plan to utilize OpenAI API in order to improve the quality of the image retrieval on Kazakh language.

I. VideoCLIP

The VideoCLIP module in our platform provides text query video retrieval by leveraging the pretrained CLIP (ViT-B/32) model. We extract one frame per second from each video and compute visual embeddings using CLIP's image encoder. These embeddings, along with frame metadata, are stored for similarity search.

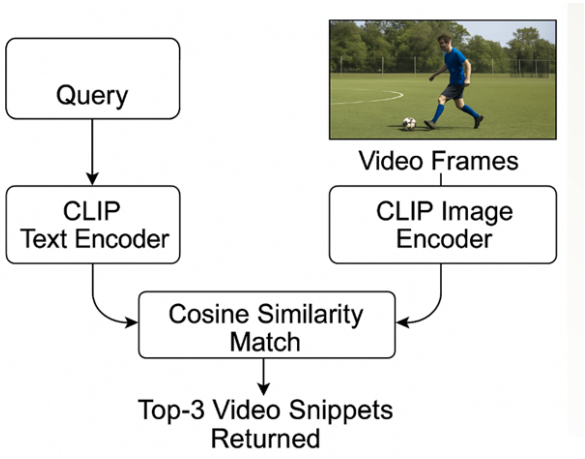

Fig. 10. CLIP-based video retrieval pipeline.

When users enter a text query, the input is encoded via CLIP's text encoder. Cosine similarity is computed between the query and stored frame embeddings, and the top-3 most relevant frames are returned.

Although we did not fine-tune the model, CLIP showed strong zero-shot performance on benchmark datasets, as reported in the original VideoCLIP paper [20].

This setup enabled fast prototyping without task-specific training while still achieving meaningful retrieval results for Kazakh-language queries.

Dataset	Recall@1	Recall@5	Recall@10
MSR-VTT	23.4%	50.6%	65.4%
YouCook2	13.4%	35.2%	49.7%

TABLE X
RETRIEVAL PERFORMANCE OF VIDEOCLIP ON STANDARD VIDEO-TEXT BENCHMARKS.

J. RAG

The search functionality was implemented using a RAG approach, as it enables semantic search rather than relying on traditional lexical matching. The development pipeline works as follows: embeddings of the chat history are precomputed and stored in a database. When a user submits a query in the search tab, the query is converted into an embedding and compared against the stored embeddings using cosine similarity. The system retrieves the top 5 most relevant past chats based on similarity scores. Our research indicates that retrieving the top 5 results significantly improves the accuracy and relevance of the final AI-generated response.

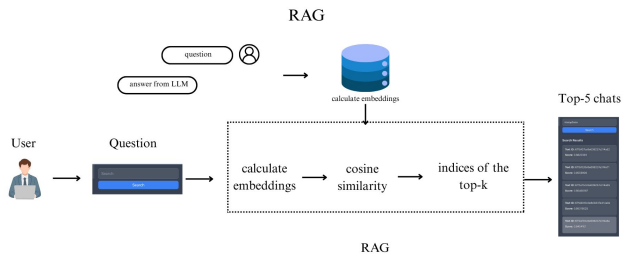

Fig. 11. RAG pipeline

We used the "intfloat/multilingual-e5-large-instruct" embedding model, which supports Kazakh among other languages, ensuring accurate representation and retrieval for Kazakh-language queries.

To evaluate the performance of the RAG-based retrieval system, we used the *Nazarbayev University Regulations* document. The evaluation was conducted across three languages: Kazakh, Russian, and English and accuracy indicates whether at least one of the top-k documents retrieved was relevant to the query. The results are shown below:

Top-k	Accuracy
1	0.76
2	0.86
3	0.90
4	0.90
5	0.94

TABLE XI
TOP-k RETRIEVAL ACCURACY FOR RAG SYSTEM

VI. RESPONSIBILITIES

Machine learning part was a responsibility of Aidana Baglanova, Umit Azirakhmet and Nail Fakhrutdinov. The models firstly were divided among us for a research part but their implementation required a collaboration.

Backend part was a responsibility of Ruslan Kalimzhanov. Frontend part was a responsibility of Zilola Babakhojayeva.

VII. EVALUATION

Our primary evaluation focus is response time, as delivering fast and smooth results is essential for a good user experience. Currently, we operate with a single 11GB GPU, which imposes significant constraints. For instance, the OCR model alone requires around 6GB of memory during inference, making it impractical to run all models on the GPU simultaneously. As a result, we strategically allocate models between CPU and GPU based on performance needs. Some tasks may require slightly more time when queued or executed on the CPU, but the system remains responsive and usable.

Model Name	Response Time (s)	Resource
Text-to-Speech (TTS)	113	CPU
Speech-to-Text (STT)	7.88	CPU
Question Answering (QA)	18.24	CPU
Automatic Summarization (AS)	16.46	CPU
Machine Translation (MT)	3.41	GPU
Named Entity Recognition (NER)	3.00	CPU
Optical Character Recognition (OCR)	30.00 / 1.67	CPU / GPU
Image Captioning (IC)	3.55	CPU
Large Language Model (LLM)	19.31	GPU
KazCLIP	2.82	CPU
VideoCLIP	28.95	GPU

TABLE XII
EVALUATION OF MODEL RESPONSE TIME AND RESOURCE USAGE

VIII. ANALYSIS OF OUR WORK

The development of the project followed an agile approach, where each AI model went through a cycle of research, development, testing, and integration. In some cases, fine-tuning was applied to improve performance. Once a model was ready, it was deployed and maintained in both the backend and frontend environments.

As the number of models increased, we encountered limitations in our initial framework. This prompted a design decision to migrate to a more efficient and scalable framework called Triton. Our focus then shifted to integrating all models within the new architecture and optimizing specific components simultaneously. This transition was challenging but ultimately allowed us to improve GPU and CPU utilization, ensuring that the platform remained scalable and capable of handling increased workloads.

While some models required more experimentation and debugging than anticipated, we addressed these challenges by iterating quickly, benchmarking different tools, and adjusting our pipeline where necessary.

The main difficulties in this project arise during the preparation of models for development. Particularly, conversion of models from Torch format to optimized TensorRT inference engine. This process starts from capturing computational graph of the model into ONNX format, which is then compiled into TensorRT engine. During the conversion to ONNX format, the model architecture need a little tweaking, especially when

model has dynamic input shapes or dynamic branches during the inference. This requires manually setting dynamic shapes in ONNX graph. Once the model is converted to ONNX, we need to build execution engine from that. Moreover, TensorRT engines only work in the environment in which they were built. Also, since different models have different dependencies and conflicting libraries, it is also beneficial to convert them into ONNX format, which allows us to remove all the dependencies used by the models.

Our initial plans were successfully implemented. We created a platform for AI services for Kazakh language. These services include TTS, STT, QA, LLM, AS, MT, NER, OCR, IC, and KazClip. As a result we have a fully deployed platform allowing multiple users to interact with it and use it to automate tasks and digitize data. To further develop our system, our next goals included enabling the implemented services to operate in two languages: Kazakh and English. So, implemented services accept queries in both languages. Additionally, to create a more complex system, we integrated the developed models into pipelines. These pipelines are used in the OCR-based document QA process and LLM chats. In addition to that, to broaden the scope of the project, the VideoClip service was added. This service became a part of our multimedia retrieval feature, where the pipelines were also used.

IX. Conclusion and possible future work

This project successfully resulted in the development and deployment of a comprehensive web-based platform offering 11 AI services specifically designed for the Kazakh language. The system is optimized for performance, supports real-time user interaction, and ensures secure handling of user data. By providing accessible AI solutions for Kazakh speakers, the platform contributes to narrowing the digital divide and fostering the inclusion of low-resource languages in modern technological ecosystems.

Subsequent development efforts will focus on enhancing model accuracy through the collection of higher-quality Kazakh-language datasets and task-specific fine-tuning. Further improvements will include the integration of user personalization mechanisms, multilingual support to extend the platform's applicability, and the exploration of additional processing pipelines to support more complex and domain-specific use cases.

References

[1] S. Mussakhojayeva, A. Janaliyeva, A. Mirzakhmetov, Y. Khassanov, and H. A. Varol, "KazakhTTS: An Open-Source Kazakh Text-to-Speech Synthesis Dataset," in *Proc. Interspeech 2021*, 2021, pp. 2786–2790.

[2] Y. Khassanov, S. Mussakhojayeva, A. Mirzakhmetov, A. Adiyev, M. Nurpeiissov, and H. A. Varol, "A crowdsourced open-source kazakh speech corpus and initial speech recognition baseline," *arXiv preprint arXiv:2009.10334*, 2020. [Online]. Available: https://arxiv.org/pdf/2009.10334

[3] R. Yeshpanov, P. Efimov, L. Boytsov, A. Shalkarbayuli, and P. Braslavski, "KazQAD: Kazakh open-domain question answering dataset," in *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti, and N. Xue, Eds. Torino, Italia: ELRA and ICCL, May 2024, pp. 9645–9656. [Online]. Available: https://aclanthology.org/2024.lrec-main.843

[4] "IrbisAI/Irbis-7b-v0.1 · Hugging Face — huggingface.co," https://huggingface.co/IrbisAI/Irbis-7b-v0.1.

[5] K. Nurgali, "llama-1.9b-kaz," 2024. [Online]. Available: https://huggingface.co/nur-dev/llama-1.9B-kaz

[6] T. Zhabayev and U. Tukeyev, "Development of technology for summarization of kazakh text," *(IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 12, no. 9, pp. 111–116, 2021. [Online]. Available: https://thesai.org/Downloads/Volume12No9/Paper_13-Development_of_Technology_for_Summarization_of_Kazakh_Text.pdf

[7] R. Yeshpanov, A. Polonskaya, and H. A. Varol, "Kazparc: Kazakh parallel corpus for machine translation," 2024.

[8] N. Team, M. R. Costa-jussà, J. Cross, O. Çelebi, M. Elbayad, K. Heafield, K. Heffernan, E. Kalbassi, J. Lam, D. Licht, J. Maillard, A. Sun, S. Wang, G. Wenzek, A. Youngblood, B. Akula, L. Barrault, G. M. Gonzalez, P. Hansanti, J. Hoffman, S. Jarrett, K. R. Sadagopan, D. Rowe, S. Spruit, C. Tran, P. Andrews, N. F. Ayan, S. Bhosale, S. Edunov, A. Fan, C. Gao, V. Goswami, F. Guzmán, P. Koehn, A. Mourachko, C. Ropers, S. Saleem, H. Schwenk, and J. Wang, "No language left behind: Scaling human-centered machine translation," 2022. [Online]. Available: https://arxiv.org/abs/2207.04672

[9] E. Bekbulatov and A. Kartbayev, "A study of certain morphological structures of kazakh and their impact on the machine translation quality," in *2014 IEEE 8th International Conference on Application of Information and Communication Technologies (AICT)*, 2014, pp. 1–5.

[10] J. Tiedemann, "The Tatoeba Translation Challenge – Realistic data sets for low resource and multilingual MT," in *Proceedings of the Fifth Conference on Machine Translation*. Online: Association for Computational Linguistics, Nov. 2020, pp. 1174–1182. [Online]. Available: https://www.aclweb.org/anthology/2020.wmt-1.139

[11] R. Yeshpanov, Y. Khassanov, and H. A. Varol, "KazNERD: Kazakh named entity recognition dataset," in *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, June 2022, pp. 417–426. [Online]. Available: https://aclanthology.org/2022.lrec-1.44

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[13] S. Francis and M. Sangeetha, "A comparison study on optical character recognition models in mathematical equations and in any language," *Results in Control and Optimization*, vol. 18, p. 100532, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666720725000189

[14] S. N. Batyr Arystanbekov, Askat Kuzdeuov and H. A. Varol, "Image captioning for visually impaired and blind: A recipe for lower-resourced languages," 2023.

[15] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021. [Online]. Available: https://arxiv.org/abs/2103.00020

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[18] A. Asgarov and S. Rustamov, "Lowclip: Adapting the clip model architecture for low-resource languages in multimodal image retrieval task," 2024. [Online]. Available: https://arxiv.org/abs/2408.13909

[19] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015. [Online]. Available: https://arxiv.org/abs/1405.0312

[20] H. Xu, G. Ghosh, P.-Y. Huang, D. Okhonko, A. Aghajanyan, F. Metze, L. Zettlemoyer, and C. Feichtenhofer, "Videoclip: Contrastive pre-training for zero-shot video-text understanding," 2021. [Online]. Available: https://arxiv.org/abs/2109.14084

[21] "Anaconda software distribution," 2020. [Online]. Available: https://docs.anaconda.com/

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019. [Online]. Available: https://arxiv.org/abs/1912.01703

[23] O. R. developers, "Onnx runtime," https://onnxruntime.ai/, 2021.

[24] "GitHub - NVIDIA/TensorRT: NVIDIA® TensorRT™ is an SDK for high-performance deep learning inference on NVIDIA GPUs. This

repository contains the open source components of TensorRT. — github.com," https://github.com/NVIDIA/TensorRT.
[25] "GitHub - triton-inference-server/server: The Triton Inference Server provides an optimized cloud and edge inferencing solution. — github.com," https://github.com/triton-inference-server/server, [Accessed 24-01-2025].
[26] "GitHub - fastapi/fastapi: FastAPI framework, high performance, easy to learn, fast to code, ready for production — github.com," https://github.com/fastapi/fastapi, [Accessed 14-02-2025].
[27] "Quick Start Guide :: NVIDIA Deep Learning TensorRT Documentation — docs.nvidia.com," https://docs.nvidia.com/deeplearning/tensorrt/quick-start-guide/index.html.
[28] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," 2018. [Online]. Available: https://arxiv.org/abs/1712.05884
[29] Y. Nakano, T. Saeki, S. Takamichi, K. Sudoh, and H. Saruwatari, "vtts: visual-text to speech," 2022. [Online]. Available: https://arxiv.org/abs/2203.14725
[30] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," 2020. [Online]. Available: https://arxiv.org/abs/2010.05646
[31] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech: Fast, robust and controllable text to speech," 2019. [Online]. Available: https://arxiv.org/abs/1905.09263
[32] Walidamamou, "Mastering named entity recognition with bert in 2024," Apr 2024. [Online]. Available: https://ubiai.tools/mastering-named-entity-recognition-with-bert/
[33] "GitHub - h/pytesseract: Python-tesseract is an optical character recognition (OCR) tool for python — github.com," https://github.com/h/pytesseract.
[34] "GitHub - PaddlePaddle/PaddleOCR: Awesome multilingual OCR toolkits based on PaddlePaddle (practical ultra lightweight OCR system, support 80+ languages recognition, provide data annotation and synthesis tools, support training and deployment among server, mobile, embedded and IoT devices) — github.com," https://github.com/PaddlePaddle/PaddleOCR/tree/main.
[35] "GitHub - VikParuchuri/surya: OCR, layout analysis, reading order, table recognition in 90+ languages — github.com," https://github.com/VikParuchuri/surya.
[36] "GitHub - JaidedAI/EasyOCR: Ready-to-use OCR with 80+ supported languages and all popular writing scripts including Latin, Chinese, Arabic, Devanagari, Cyrillic and etc. — github.com," https://github.com/JaidedAI/EasyOCR.