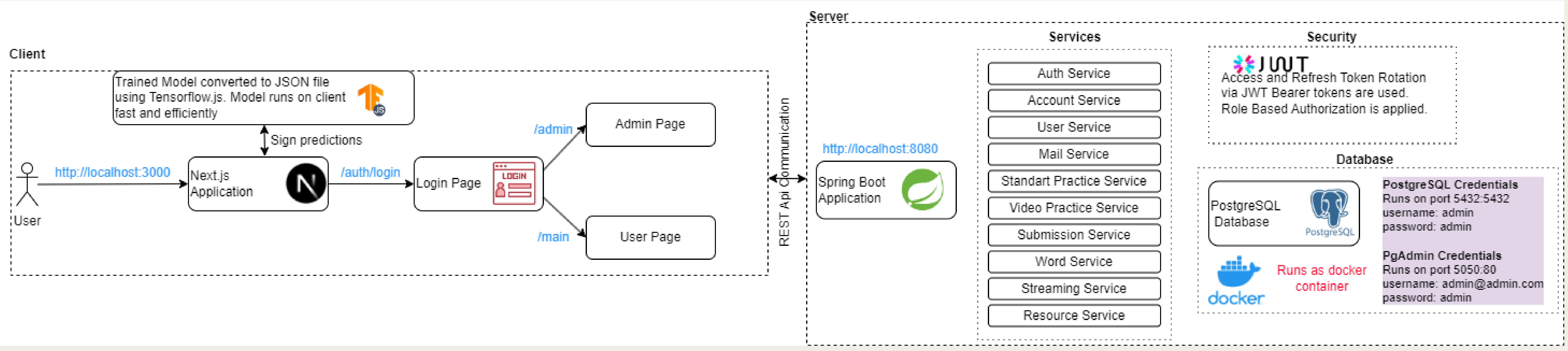
DuoSignLanguage: Turkish Sign Language Education Platform

Numan Kafadar [1] Yunus Emre Terzi[1] Mustafa Çağrı Korkmaz [1] Hacer Yalım Keleş [2]

[1]Project Developer [2]Supervisor

Introduction

DuoSignLanguage is a web application designed to help users learn Turkish Sign Language interactively through an intuitive interface. It utilizes a trained Keras model running on the client side to detect and interpret users' sign gestures.



Client

Trained Model converted to JSON file using Tensorflow.js. Model runs on client fast and efficiently

Sign predictions

User

http://localhost:3000

Next.js Application

/auth/login

Login Page

/admin

Admin Page

/main

User Page

REST Api Communication

Server

http://localhost:8080

Spring Boot Application

Services

Auth Service

Account Service

User Service

Mail Service

Standart Practice Service

Video Practice Service

Submission Service

Word Service

Streaming Service

Resource Service

Security

JWT

Access and Refresh Token Rotation via JWT Bearer tokens are used.
Role Based Authorization is applied.

Database

PostgreSQL Database

Runs as docker container

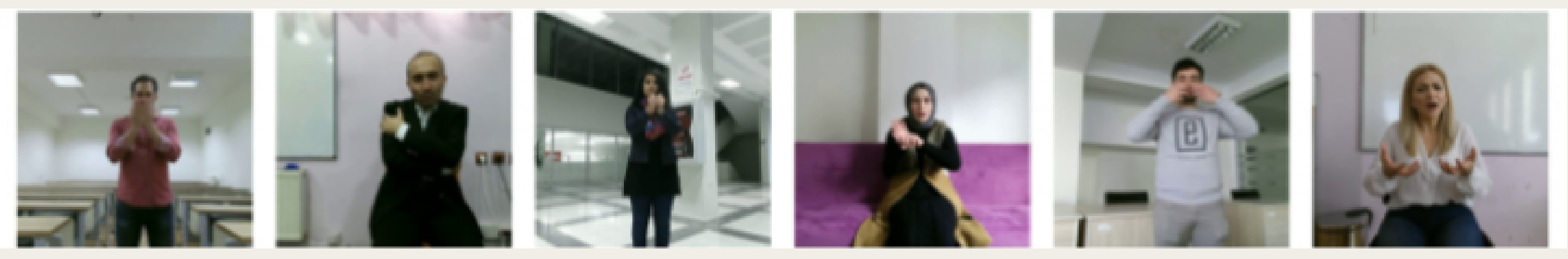
PostgreSQL Credentials
Runs on port 5432:5432
username: admin
password: admin

PgAdmin Credentials
Runs on port 5050:80
username: admin@admin.com
password: admin

System Overview

Dataset

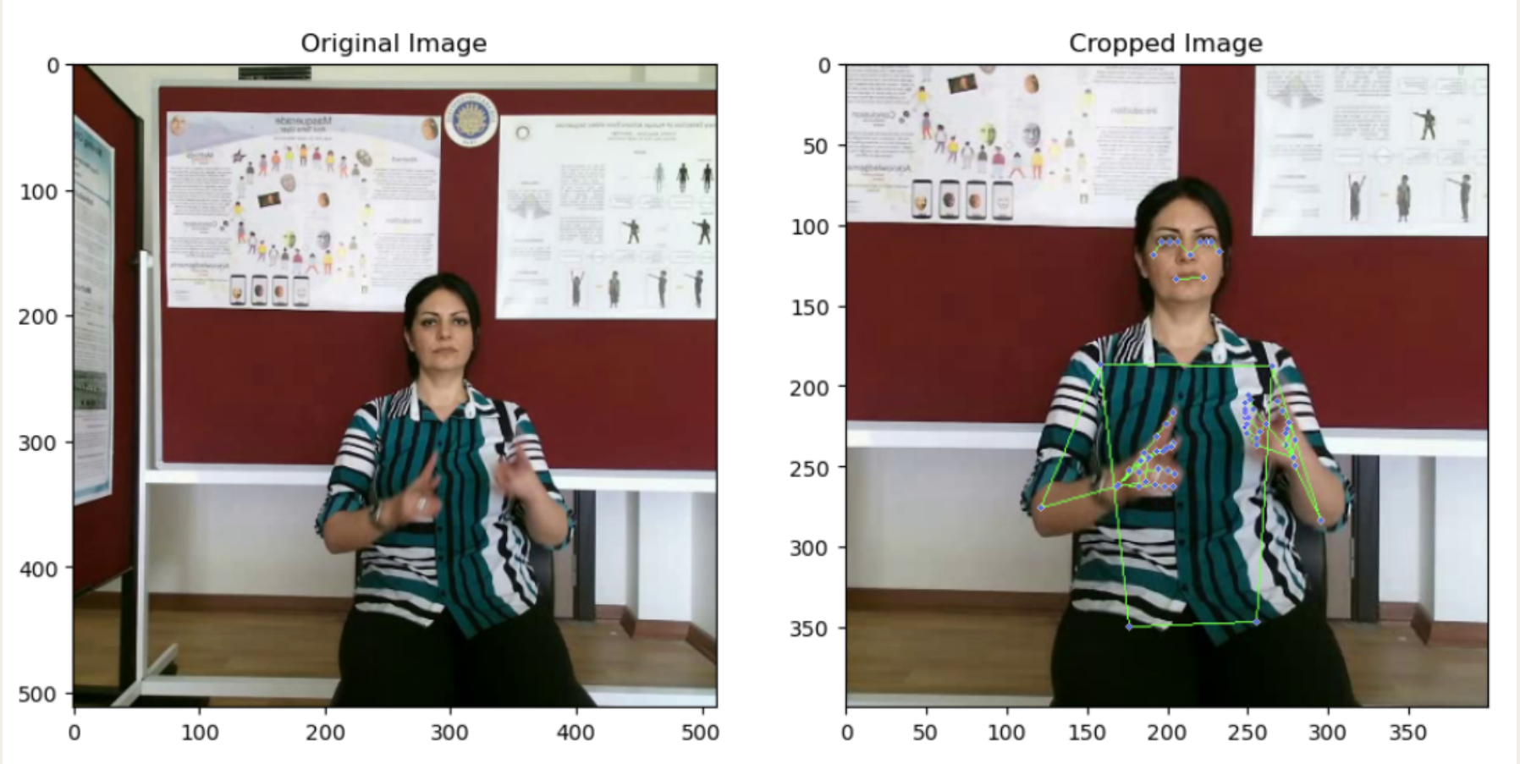
In this project, the Ankara University Turkish Sign Language Dataset (AUTSL) is utilized. This large-scale, multimodal dataset includes 226 isolated Turkish sign videos performed by 43 different signers, totaling 38,336 video samples.



Dataset

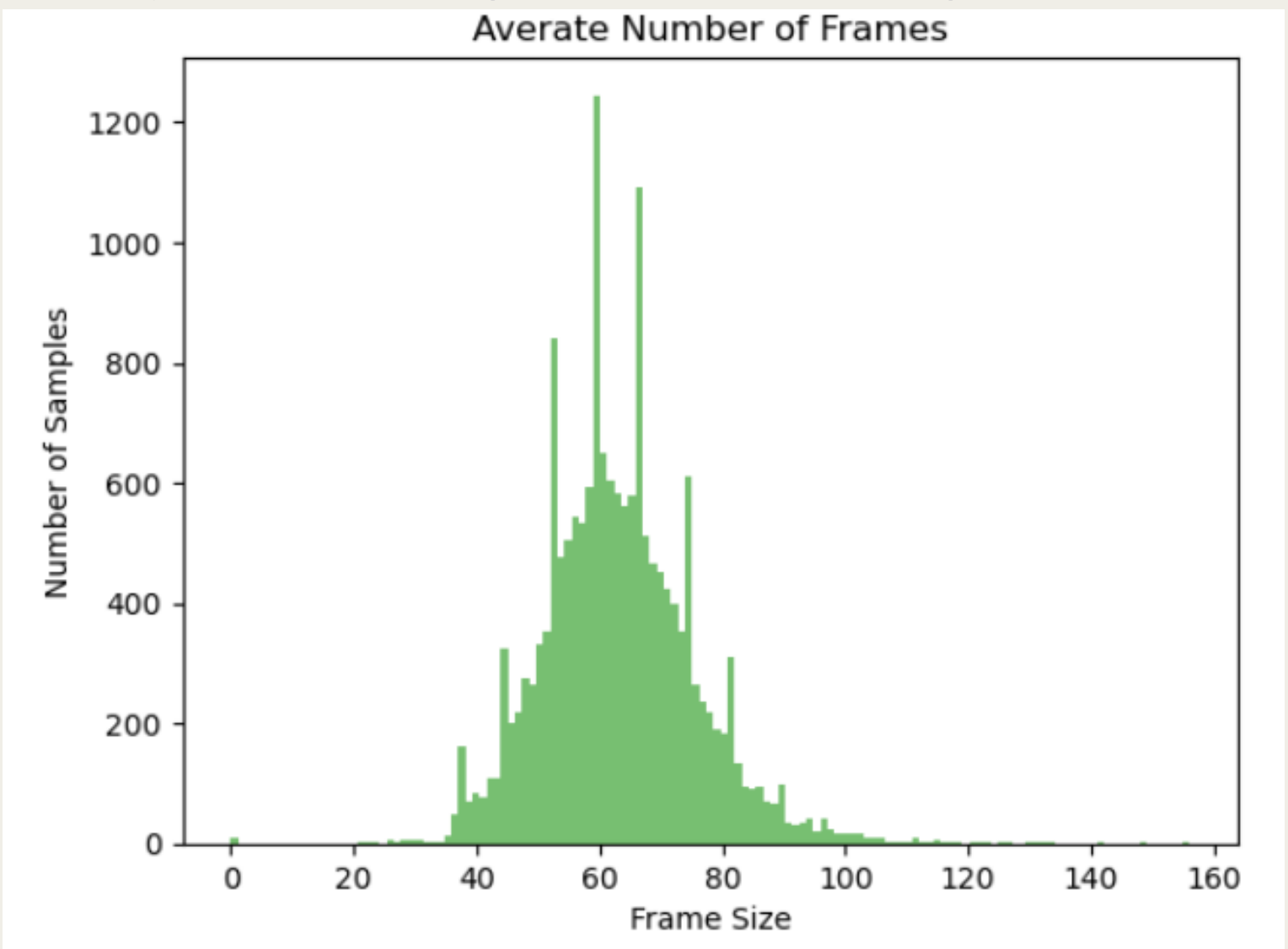
Data Preprocessing

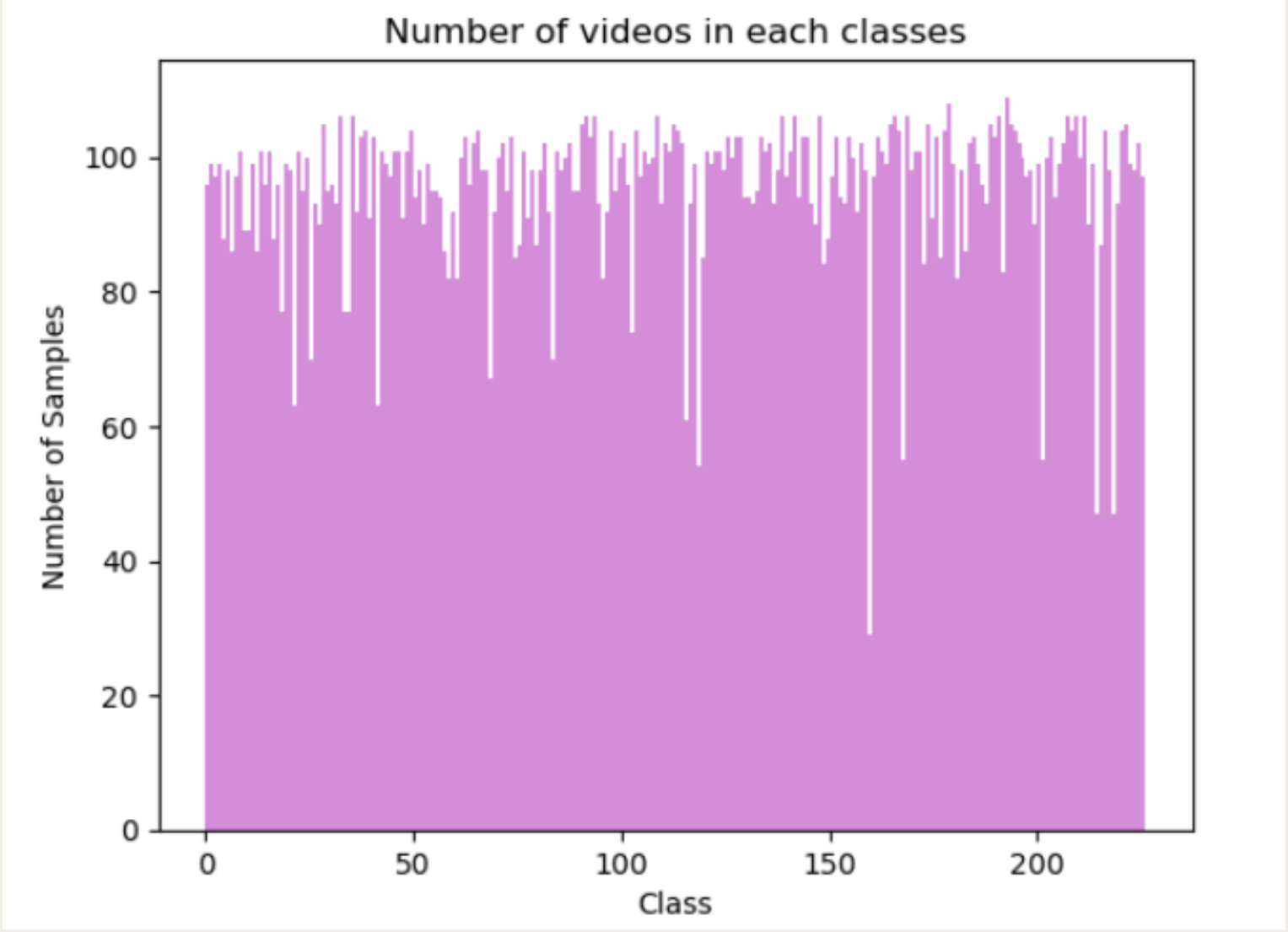
- **Standing Signer Elimination:** Only videos of sitting signers are used.
- **Cropping Videos:** A 400x400 crop is applied to all videos in the dataset for better prediction accuracy.
- **MediaPipe Keypoint Detection:** Instead of using image frames directly, pose and left-right hand landmarks are extracted from each frame using the Mediapipe Holistic Model, utilizing only the x and y coordinates.



Crop and Keypoint Detection

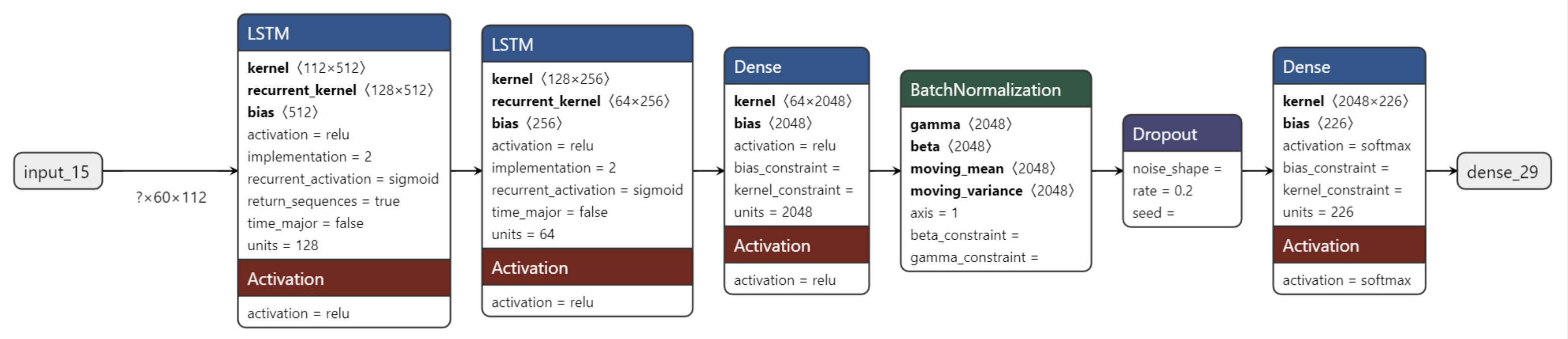
- **Frame Number Equalization:** To ensure consistency, all videos are resized to 60 frames by evenly eliminating and duplicating consecutive frames.





Model Methodology and Training

LSTM architectures are employed in conjunction with dense layers to construct the model. Regularization methods, including Dropout, BatchNormalization, and L2 regularizers, are implemented to enhance accuracy.



input_15

?×60×112

LSTM
kernel ⟨112×512⟩
recurrent_kernel ⟨128×512⟩
bias ⟨512⟩
activation = relu
implementation = 2
recurrent_activation = sigmoid
return_sequences = true
time_major = false
units = 128
Activation
activation = relu

LSTM
kernel ⟨128×256⟩
recurrent_kernel ⟨64×256⟩
bias ⟨256⟩
activation = relu
implementation = 2
recurrent_activation = sigmoid
time_major = false
units = 64
Activation
activation = relu

Dense
kernel ⟨64×2048⟩
bias ⟨2048⟩
activation = relu
bias_constraint =
kernel_constraint =
units = 2048
Activation
activation = relu

BatchNormalization
gamma ⟨2048⟩
beta ⟨2048⟩
moving_mean ⟨2048⟩
moving_variance ⟨2048⟩
axis = 1
beta_constraint =
gamma_constraint =

Dropout
noise_shape =
rate = 0.2
seed =

Dense
kernel ⟨2048×226⟩
bias ⟨226⟩
activation = softmax
bias_constraint =
kernel_constraint =
units = 226
Activation
activation = softmax

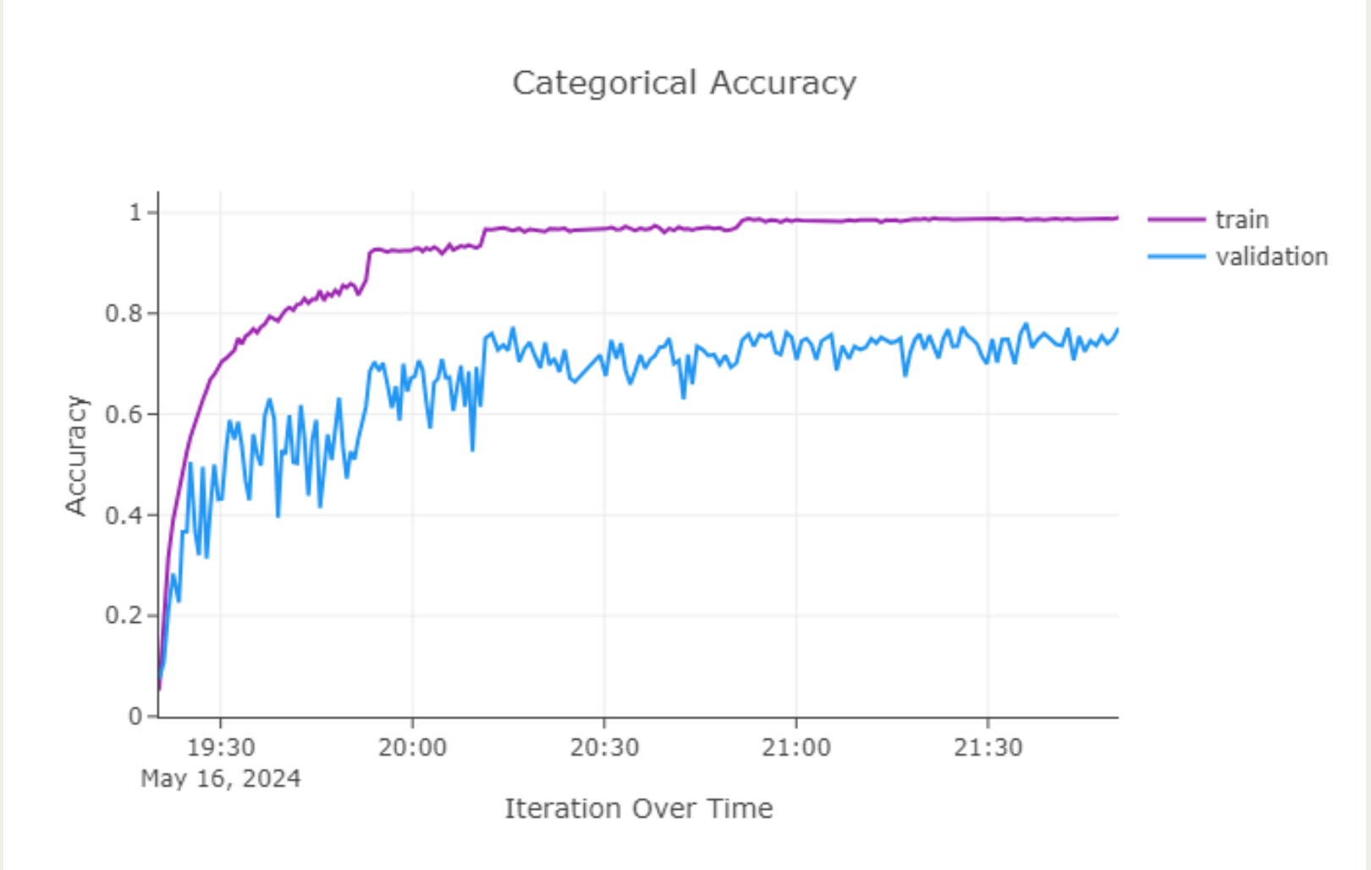
dense_29

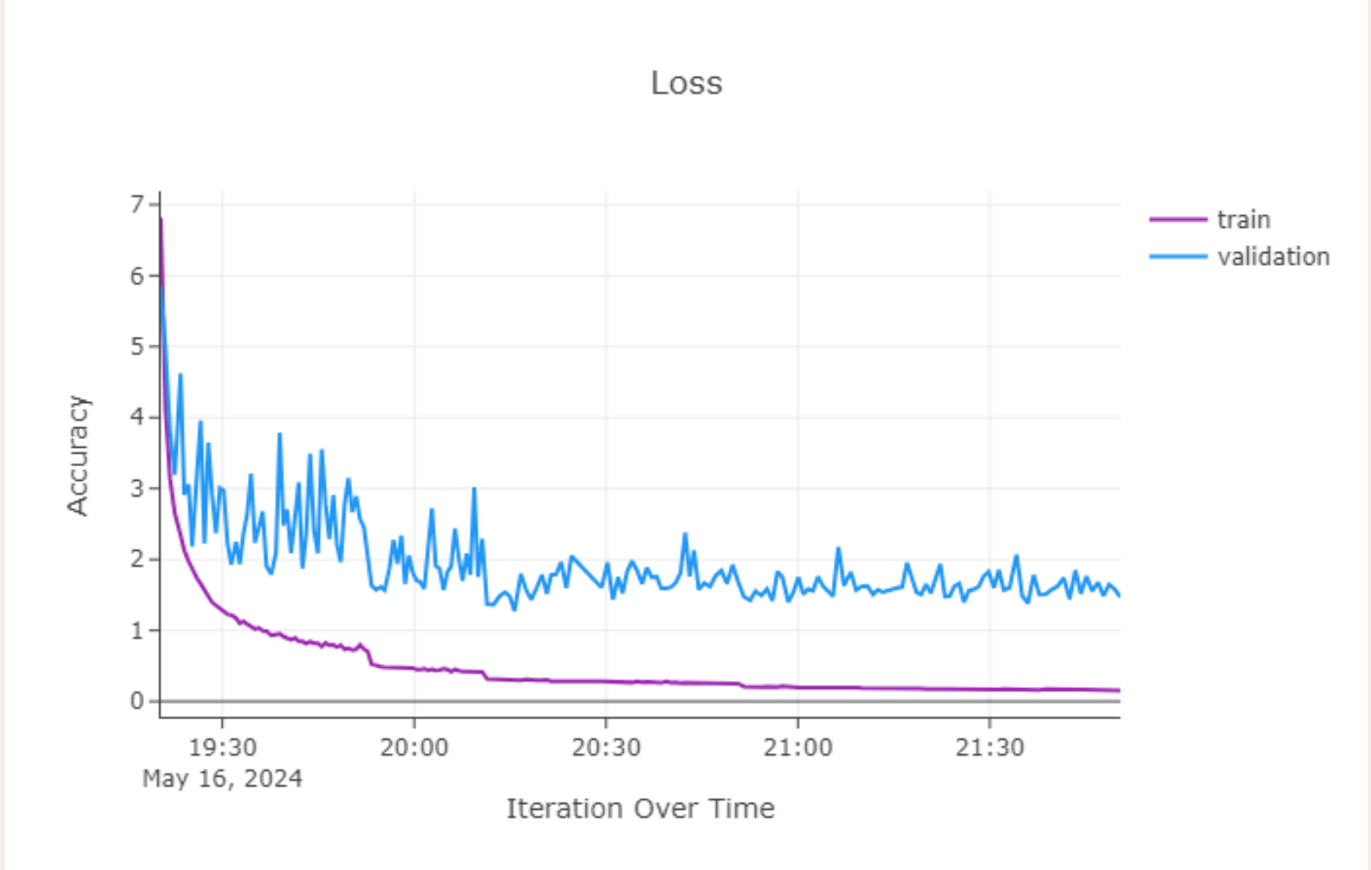
Model Overview

The model is trained using the Adam optimizer with Categorical Cross Entropy loss function. Dynamic learning rate scheduling is implemented based on validation loss. Training is conducted over 200 epochs with a batch size of 64.

Results

The model achieved an accuracy of 77.3% on the test dataset, indicating a strong performance. Furthermore, when tested on real-time videos, the model demonstrated acceptable performance.





Backend Server and Database

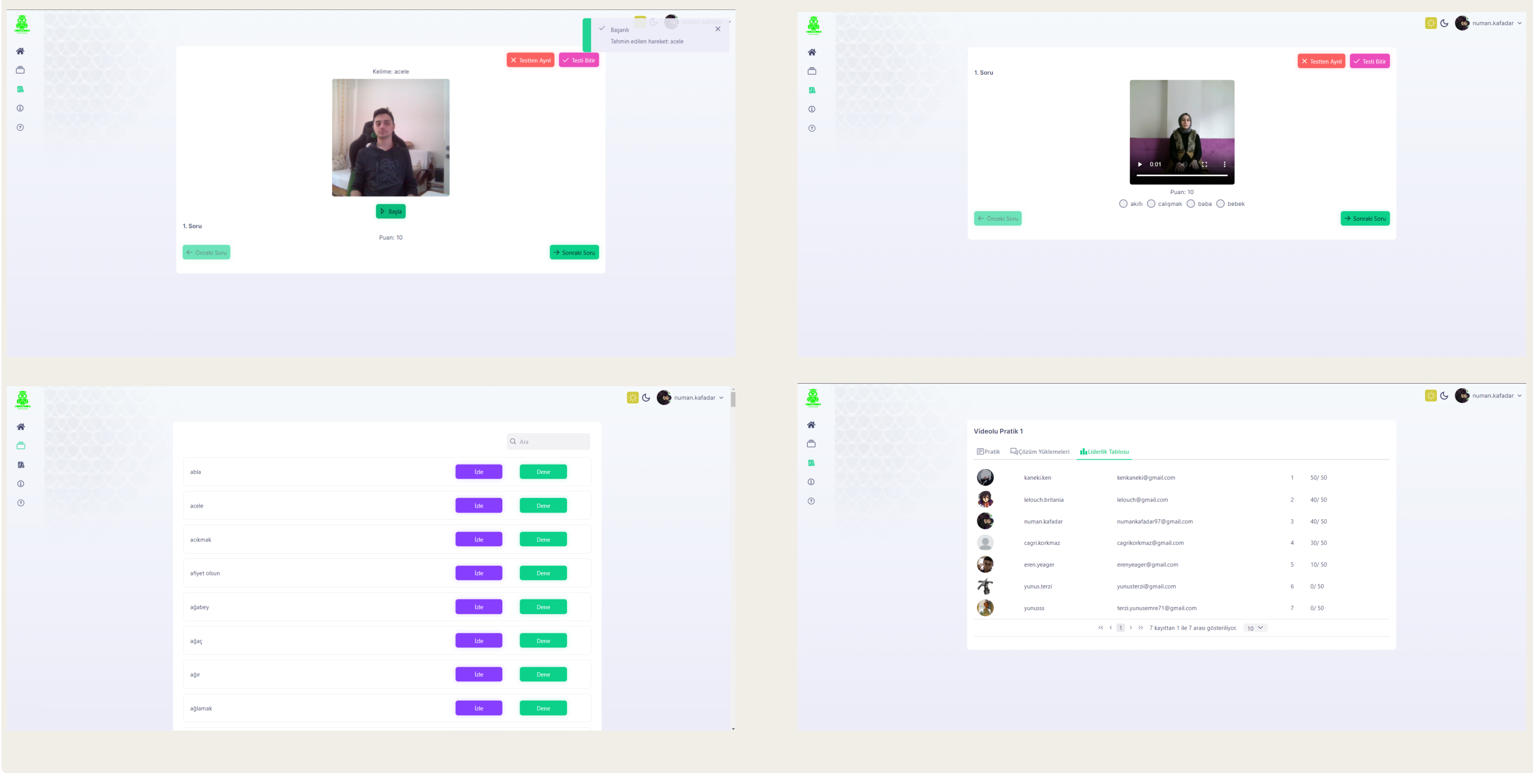
The server is developed using Java Spring Framework. It is connected to PostgreSQL Database.

- **Security:** JWT access and refresh token rotation are implemented using Spring Security. Role-based authorization dynamically secures data resources. Additionally, all passwords are stored encrypted in the database for enhanced security.
- **Data Storage:** Hibernate ORM is utilized for reading from and writing to SQL tables, as well as executing SQL commands. It manages storage of refresh tokens, user information, practices, submissions, resource videos, and more within the database using ORM principles.
- **Services and Controllers:** REST API controllers handle user interactions by responding to requests. Secure resources require a JWT Bearer token in the HTTP header for data retrieval. Services act as intermediary layers between controllers and the database, processing domain-related requests.

Client Application

The client web application is built using the Next.js Framework. It includes standard features such as login, register, password recovery via email, and account information update. Depending on their roles, users are directed to either admin or main pages upon login.

- **Admin Pages:** From these pages, administrators have the ability to manage the system, including tasks such as creating, updating, and deleting users, practices, and other relevant entities.
- **Main Pages:** Users can participate in solving practices, view their submissions, access leaderboards and dictionary.



Future Work

Currently, the model faces challenges in detecting similar signs, and its performance can be enhanced further. The system currently supports detection of 226 signs and is designed to be extendable. Based on user feedback, additional features such as different types of practices can be integrated into the application.

Desing Project Representation June 07, 2024 Hacettepe University, Computer Engineering Department