Project Title and Overview

Title

Expression Tracker: Sentiment Analysis for Dyslexic Kids During Gameplay

Summary

This project involves developing a sentiment analysis system that monitors and evaluates the emotional states of dyslexic children while they play educational games. Using web camera feeds and game screenshots, the system captures images, analyzes emotions using a HuggingFace model, and provides detailed insights to therapists and game developers. Built using the MERN stack, it also includes role-based access for children, therapists, and administrators, ensuring security and usability.

Team Members

Team G161 :

SHIVANI MUDIGA - 23BD1A1239

KODIPYAKA SUJANA GUPTA - 23BD1A1231

GAMPA ANUPAMA - 23BD1A1216

BODDU JHANSI - 23BD1A1211

ANANYA GANJI - 23BD1A1206

SIRI ANUMANDLA - 23BD1A1258

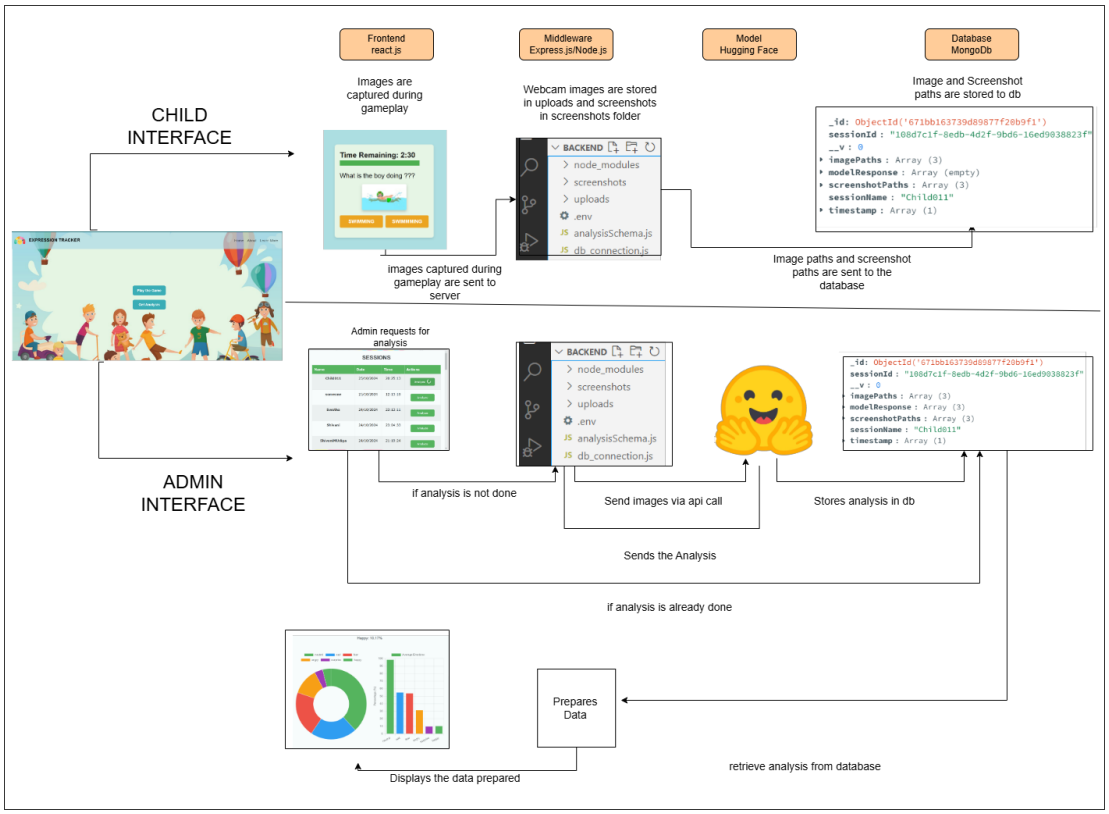
ANNAM MEGHAN - 23BD1A1268

Objectives

- **Problem Addressed**: Helps therapists and game developers understand emotional engagement and improve game design for dyslexic children.
- **Importance**: Provides insights into emotional states to optimize learning experiences and support children's mental well-being.

System Design

High-Level Architecture



- **Frontend**: React app capturing webcam images and displaying analysis.
- **Backend**: Node.js server managing file storage, API integration, and user roles.
- **Database**: MongoDB storing session data, images, and analysis results.
- **API**: HuggingFace for emotion detection.

Component Description

Frontend

- **Framework/Library**: React.js, Bootstrap for styling.

- **Key Components**:

  - Login/Registration pages: Role-based user authentication.
  - Game UI: Tracks gameplay and captures webcam images/screenshots.
  - Analysis Dashboard: Displays sentiment insights and screenshots.

Backend

- **Framework**: Express.js with Node.js.

- **Routes and Middleware**:

    - Image uploads and storage.
    - API integration with HuggingFace.
    - Role-based authentication and access control.

Database

- **Technology**: MongoDB (Atlas).

- **Schema Design**:

    - Collections: Sessions , adminschemas , games
    - Fields:
        - Sessions : session_id , sessionName , imagePaths(array) , screenshotPaths
        - (array) , modelResponse(array) , timestamps
        - Adminschemas: admin_name , phone_number , admin_email , role ,admin_professions, password(hashcodeed) , status , children_accounts (array).
        - Games: gameId , name , questions (array)

External API

- **HuggingFace API**: Emotion detection model.

- **Key Configurations**: Access tokens securely managed via `.env` files.

Technology Stack

- **Frontend**: React.js, Bootstrap.

- **Backend**: Node.js, Express.js.

- **Database**: MongoDB with Mongoose.

- **External API**: HuggingFace Transformers.

- **Other Tools**: Playwright (testing), GitHub (version control).

Development Workflow

Setup Instructions

How to clone the project.
Install dependencies.
Environment variable configuration (e.g., .env file setup for HuggingFace API key).

Development Process

1. Initial Planning and Goal Setting

Objective: The primary goal was to create a platform that could track and analyze children's emotional responses during gameplay to aid therapists, educators, and developers in understanding dyslexic children's learning behaviors.

Foundation Setup:

Technology Stack: MERN (MongoDB, Express.js, React, Node.js).

Initial features included image and screenshot capture and session ID generation during gameplay.

2. Core Feature Development

Game Design and Integration

Created a single game (quiz-based) to test the platform's functionalities.

Incorporated image capture, session ID generation, and emotion analysis models to track responses during gameplay.

Analysis Interface for Admins

Designed a basic admin interface that displayed session data for analysis.

Introduced the "Analyze" button to trigger model-based emotion analysis and store results in MongoDB.

3. Transition to Multi-Game Functionality

Game Selection and Session Management

With the introduction of multiple games, a game selection page was created.

Adjusted session ID generation to start only when a child selected a specific game and stopped upon game completion.

Storing Game Data

Added the game name to the session data stored in MongoDB to distinguish between multiple games during analysis.

4. Authentication and Role-Based Access

Secure Login System

Replaced the initial login/signup system with JWT-based authentication for secure access.

Role-based access control differentiated between:

Child: Access to games only.

Admin: Access to analysis and child management.

Super Admin: Full access, including admin approval.

Admin and Super Admin Features

Admins: Manage child profiles and view analysis results.

Super Admins: Approve admin registrations, manage games, and oversee the platform.

5. Improved Code Architecture

Reusable Hooks

Refactored the code to separate image capture, screenshot capture, and session generation into reusable hooks for integration into multiple games without disrupting existing functionalities.

Frontend and Backend Organization

Organized frontend components into styles, assets, and logical folders for better maintainability.

Separated backend routes and endpoints for admin and child functionalities, simplifying future updates.

Dynamic Game Content

Moved hardcoded questions and answers to MongoDB, enabling dynamic fetching during gameplay.

Allowed easy updates to game content without modifying the codebase.

6. Advanced Admin and Super Admin Features

Admin Enhancements

Added a child profile management system for admins, allowing the creation and editing of child accounts.

Super Admin Features

Introduced a super admin panel to manage games, approve new admins, and add child profiles.

Approval of new admins involved vetting sensitive data, ensuring only trusted individuals had access.

7. Model Development and Testing

Exploration of Models

Experimented with various models for emotion recognition:

Built model from scratch.

Tested existing models and libraries like DeepFace.

Fine-tuned Vision Transformers.

Final Model Choice

Selected the Vision Transformer model for its accuracy and suitability, leveraging the learning from other approaches.

8. Scalability and Final Enhancements

Data Storage Optimization: Streamlined MongoDB to handle both gameplay data and dynamic content.

UI Polishing: Enhanced the overall design with a modern aesthetic and intuitive navigation.

Git Workflow

Branching strategy (e.g., main, feature branches, etc.).
Pull requests and code reviews.

API Integration Details

HuggingFace API Details

- **Purpose**: Emotion analysis for images.
- **Endpoints Used**: /session/analyze/:sessionid
- **Request/Response Format**: Include sample payload and response if available.

.

Backend Integration

- Images sent to HuggingFace for analysis.
- Responses stored in MongoDB for visualization.

Frontend Integration

Data fetched from the backend and displayed using React hooks.

Challenges Faced

- **API Authentication**:
    - **Challenge**: Securing API keys and sensitive data during development and deployment.
    - **Solution**: Used `.env` files to store credentials and configured the application to load them securely with the `dotenv` library.
- **Asynchronous API Calls**:
    - **Challenge**: Managing multiple asynchronous requests for the HuggingFace model and handling potential failures or timeouts.
    - **Solution**: Utilized `async/await` for clean and manageable asynchronous calls and implemented robust error-handling mechanisms with retries.
- **CORS Issues**:
    - **Challenge**: Encountered cross-origin resource sharing (CORS) errors while connecting the frontend and backend.
    - **Solution**: Configured the backend to handle preflight requests by enabling CORS middleware and setting appropriate headers.
- **Webcam Image Dimensions**:
    - **Challenge**: Adjusting and maintaining consistent dimensions for the captured webcam images, especially when sending them to the model.
    - **Solution**: Dynamically resized the captured images using canvas elements on the frontend to standardize dimensions before encoding.
- **Base64 Encoding**:
    - **Challenge**: Converting the webcam images to Base64 format efficiently for model processing.
    - **Solution**: Used browser-based encoding functions to convert images into Base64 format and ensured compatibility with the backend API.

- **Model Processing Time**:
    - **Challenge**: The HuggingFace model occasionally took up to 20 seconds to analyze images, causing delays.
    - **Solution**: Implemented a delay mechanism with retries using a `for` loop and exponential backoff to handle such cases gracefully.

- **Handling Large Data Volumes**:
    - **Challenge**: Managing and storing a large volume of images and model responses in the database without performance degradation.
    - **Solution**: Optimized database indexing, used efficient schema designs, and set up periodic cleanup of temporary data.

- **Role-Based Access Control (RBAC)**:
    - **Challenge**: Restricting unauthorized access to certain pages and features based on user roles.
    - **Solution**: Implemented middleware on the backend to verify user roles and permissions for each API route and ensured frontend navigation respected role-based restrictions.

- **Integration of Webcam and Screenshot Functionality**:
    - **Challenge**: Capturing both webcam images and game screenshots simultaneously and linking them for analysis.
    - **Solution**: Used JavaScript's `MediaDevices` API for the webcam and game rendering frameworks for screenshot captures.

Future Scope

Suggestions for future enhancements

Personalized Analysis for Different Users:

Therapists:

- Display detailed game results, including:
    - The child's answer to each question.
    - The time taken to answer each question.
    - A breakdown of the child's emotional states during gameplay to assess engagement and learning outcomes.

- Provide insights on the child's progress over time and suggest potential areas of focus based on emotional trends.

Game Developers:

- Show metrics that help optimize game design, such as:
  - Average time taken by children to complete specific tasks or levels.
  - Emotional engagement levels at different points in the game.
  - Feedback on which game mechanics evoke the desired responses (e.g., joy, focus).
- Offer recommendations for enhancing user experience based on emotional and gameplay data.

Fine-Tune the Existing Model for Better Expression Tracking:

- Retrain the HuggingFace model using a more diverse dataset with facial expressions tailored for dyslexic children.
- Incorporate additional parameters like microexpressions, gaze detection, and context-based sentiment analysis for higher accuracy.
- Regularly update the model based on real-world data collected during gameplay to improve prediction and personalization.

Conclusion

This project demonstrates the potential of combining AI and web development to address educational and emotional challenges in dyslexic children. It also highlights the importance of designing accessible and secure systems for sensitive user data.

Appendices

Code Snippets

Include critical parts of the codebase with explanations.

References

Model reference:

https://huggingface.co/trpakov/vit-face-expression (we used)

https://huggingface.co/dima806/facial_emotions_image_detection

https://huggingface.co/motheecreator/vit-Facial-Expression-Recognition

Game references:

Login references:

GitHub use references:

How we started with the project:

API call references:

Glossary

Define technical terms or acronyms used in the document.