EXPRESSION TRACKER :

SENTIMENT ANALYSIS FOR DYSLEXIC KIDS DURING GAMEPLAY

- This project revolutionizes learning and therapy for children with learning disabilities, focusing on dyslexia.
- By leveraging emotion recognition and game-based learning, it provides innovative solutions for education, therapy, and emotional understanding.
- With **dyslexia affecting 5–10% of the global population and 20% of school-age children struggling with learning disabilities**, this platform offers a transformative approach to improving education and therapy worldwide.

GROUP G161-

23BD1A1206 G.ANANYA

23BD1A1211 B.JHANSI

23BD1A1216 G.ANUPAMA

23BD1A1231 K.SUJANA GUPTA

23BD1A1239 M.SHIVANI

23BD1A1258 A.SIRI

23BD1A1268 A.MEGHANA

Business Case

Business-to-Business (B2B)

The primary audience includes:

- **Educational Institutions**: Implementing personalized tools for special education needs **(Educators)**
- **Therapy Centers**: Evaluating emotional responses and progress in therapy. **(Therapists)**
- **EdTech Companies**: Licensing the emotion recognition API for platform enhancement. **(Game Developers)**

Partnerships and Expansion

The platform supports collaboration with universities, NGOs, and government bodies to refine and expand its reach. It plans to:

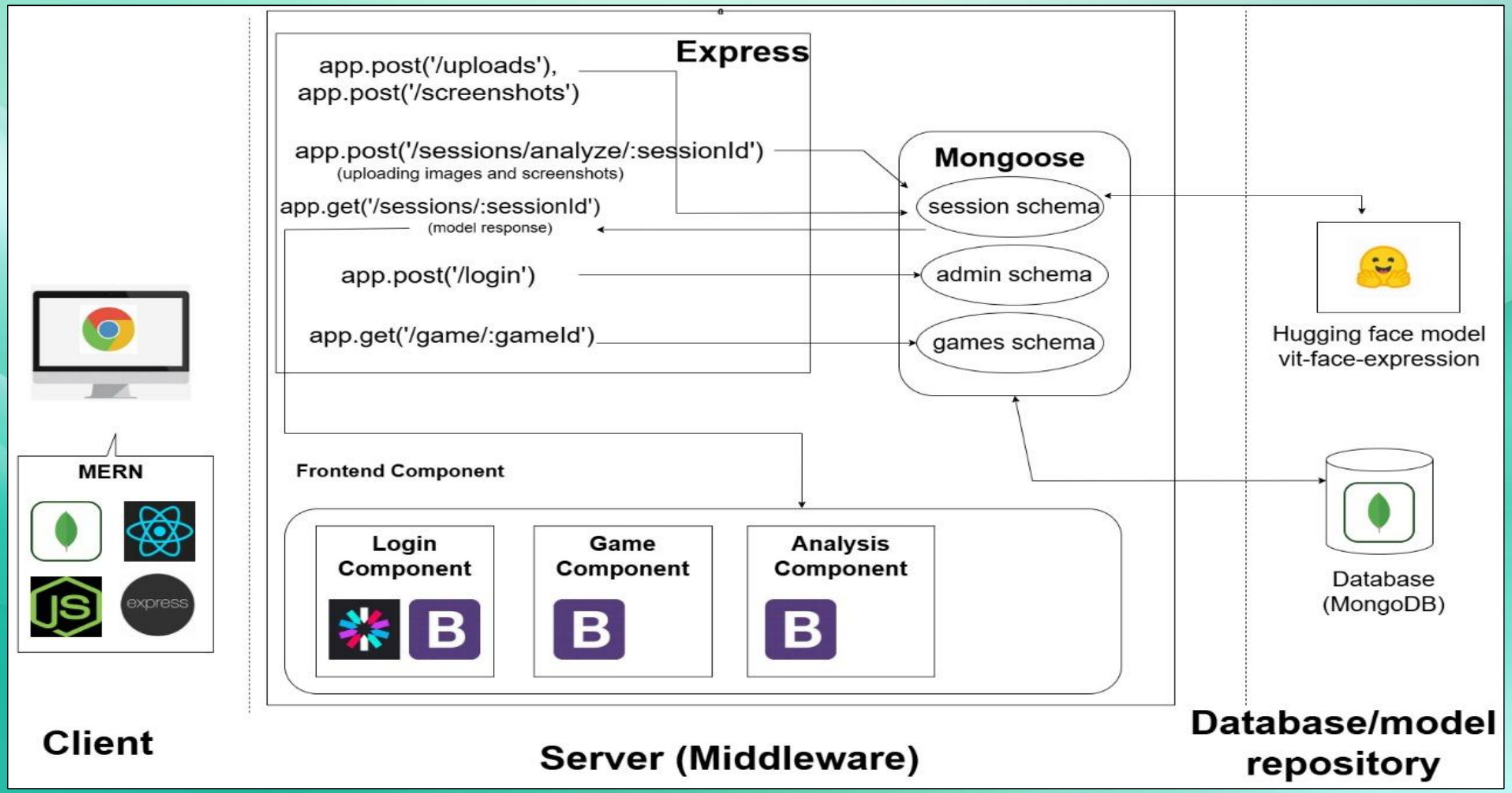
- **Localize Content**: Multi-language support for global audiences.
- **Increase Accessibility**: Mobile and tablet apps for families and educators.
- **Enhance Features**: Advanced analytics for tracking and addressing learning challenges.

Long-Term Goals

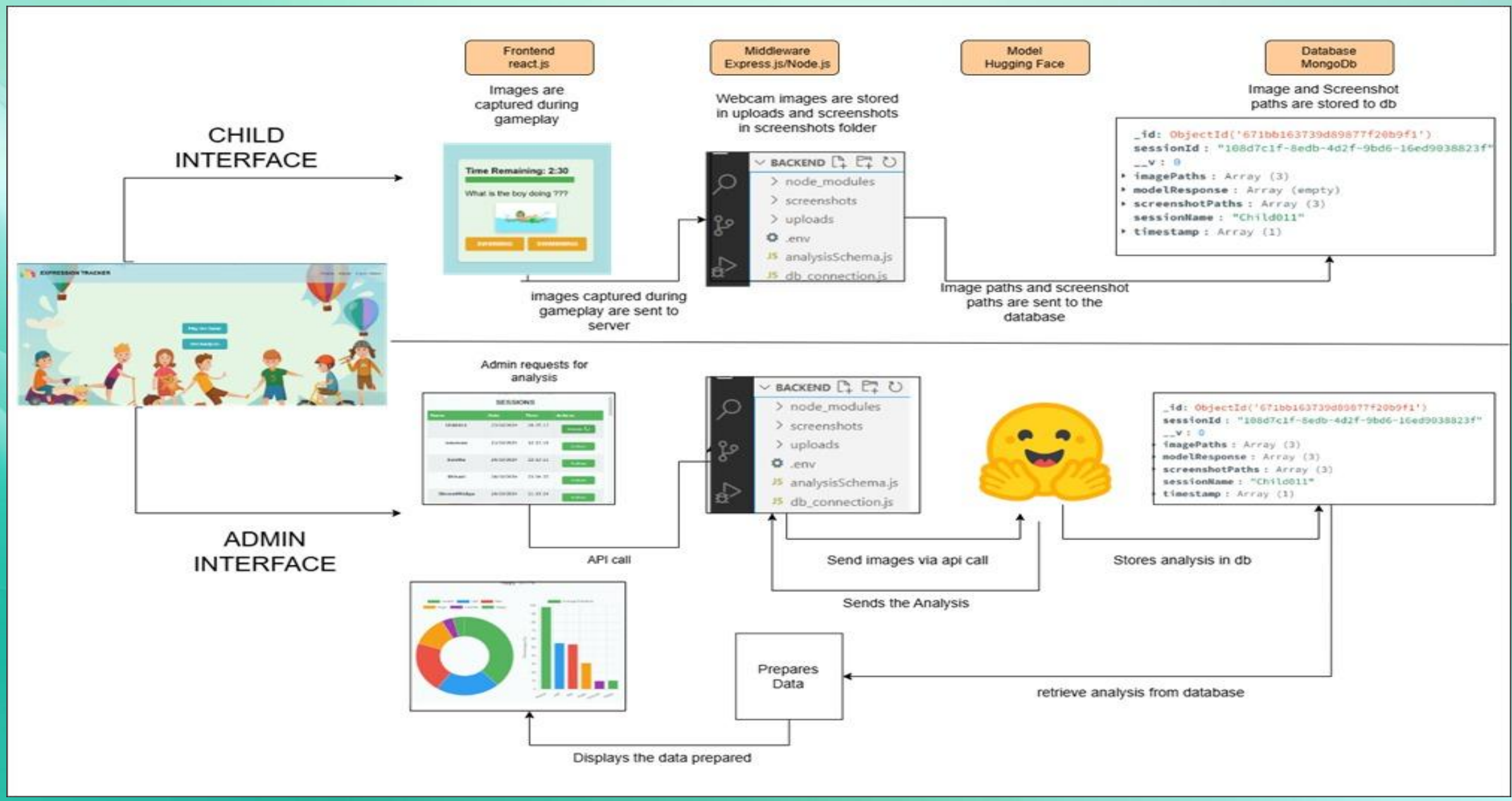
Future development includes:

- Expanding tools for other learning disabilities (ADHD, autism).
- Partnering with organizations to serve underserved populations.
- Advancing AI-driven learning tools through research collaborations.

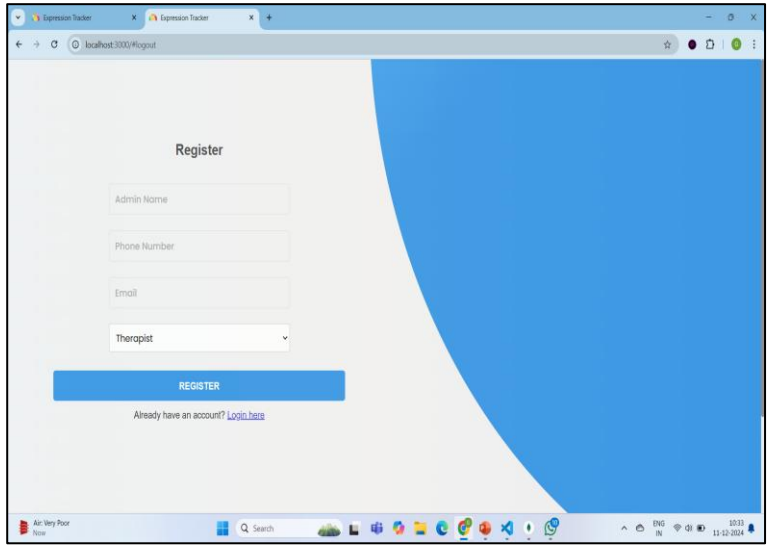
Architecture Diagram

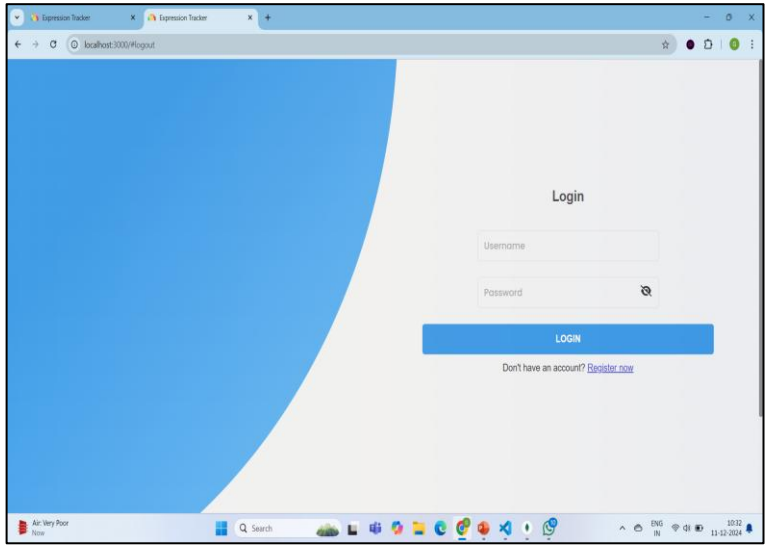


Workflow

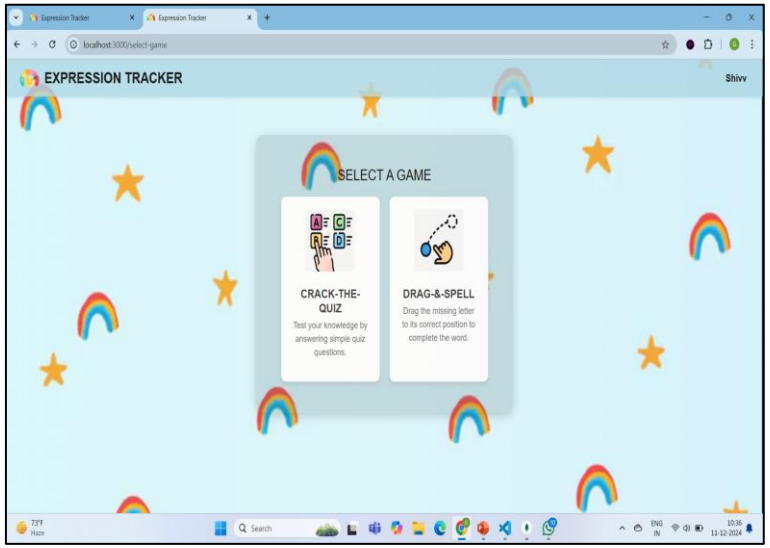

Frontend
react.js
Middleware
Express.js/Node.js
Model
Hugging Face
Database
MongoDb
CHILD
INTERFACE
Images are
captured during
gameplay
Webcam images are stored
in uploads and screenshots
in screenshots folder
Image and Screenshot
paths are stored to db
Time Remaining: 2:30
What is the boy doing ???
BACKEND
node_modules
screenshots
uploads
.env
analysisSchema.js
db_connection.js
_id: ObjectId('671bb163739d89877f20b9f1')
sessionId : "108d7c1f-8edb-4d2f-9bd6-16ed9038823f"
__v : 0
imagePaths : Array (3)
modelResponse : Array (empty)
screenshotPaths : Array (3)
sessionName : "Child011"
timestamp : Array (1)
images captured during
gameplay are sent to
server
Image paths and screenshot
paths are sent to the
database
Admin requests for
analysis
SESSIONS
ADMIN
INTERFACE
API call
Send images via api call
Stores analysis in db
Sends the Analysis
Prepares
Data
retrieve analysis from database
Displays the data prepared

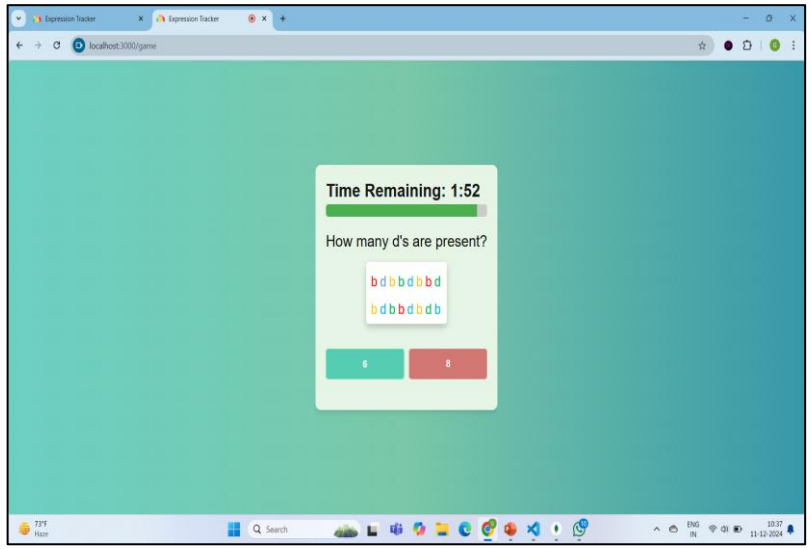
LOGIN & REGISTER PAGES

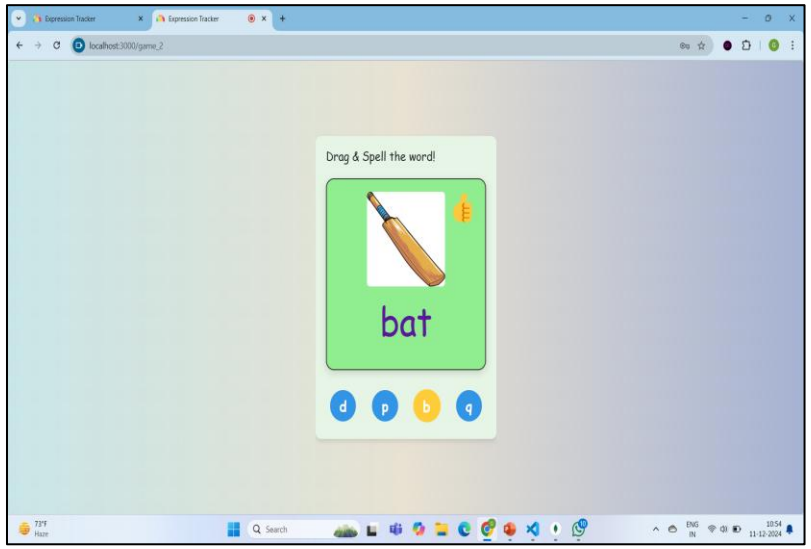




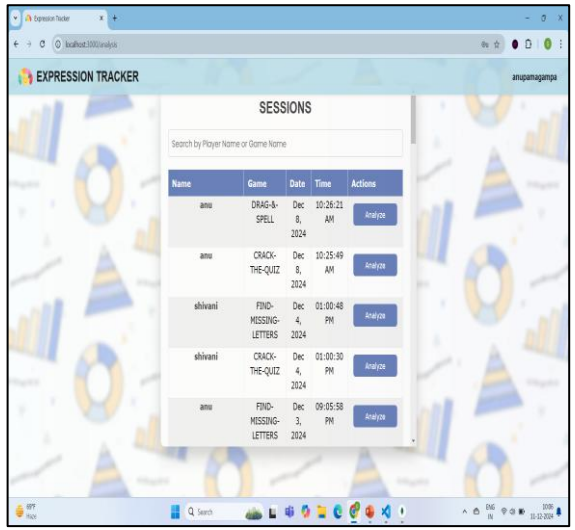
CHILD PAGES

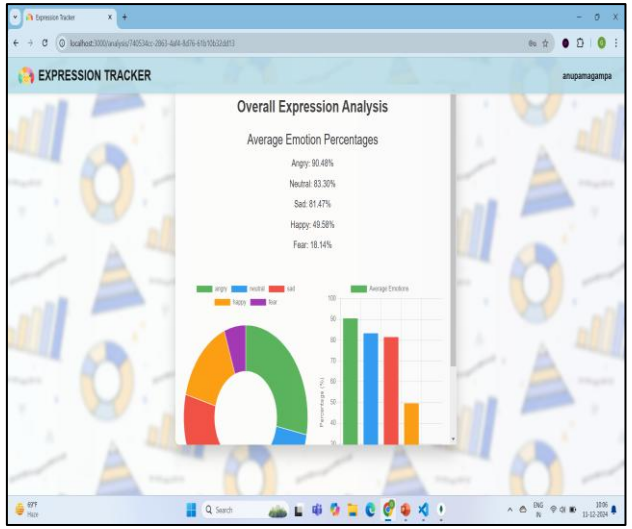


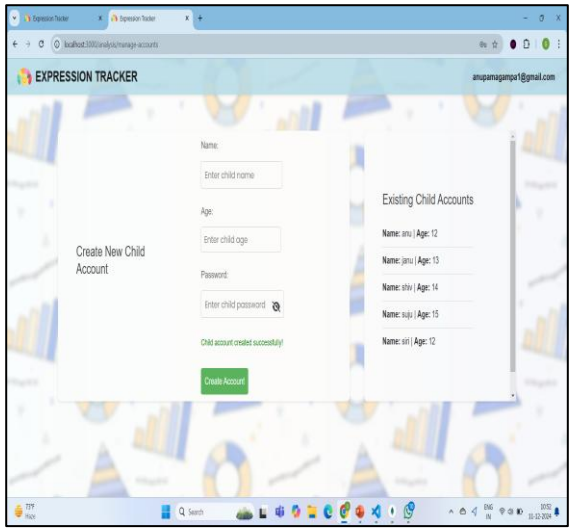


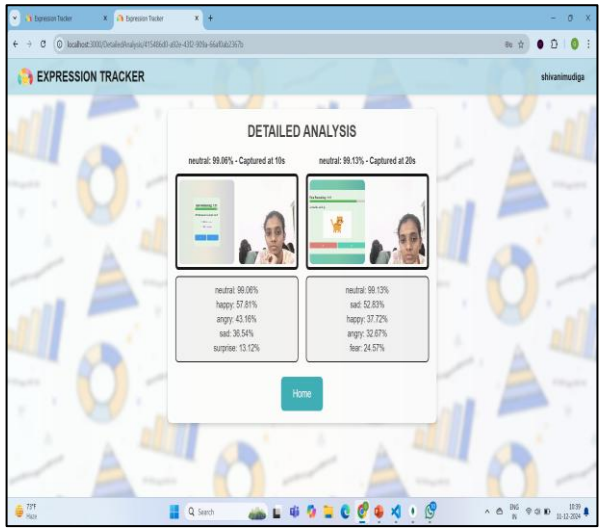


ADMIN PAGES

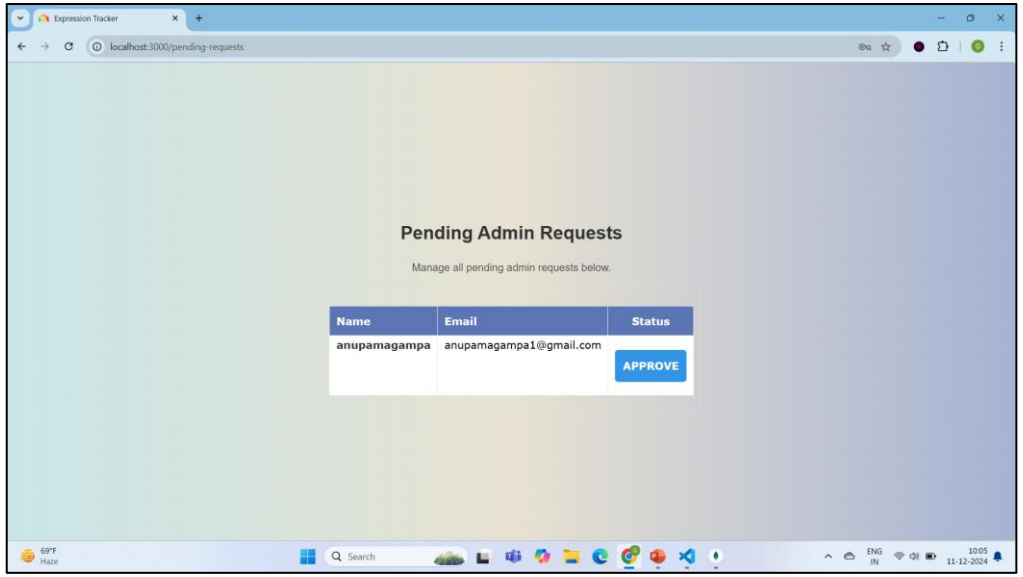


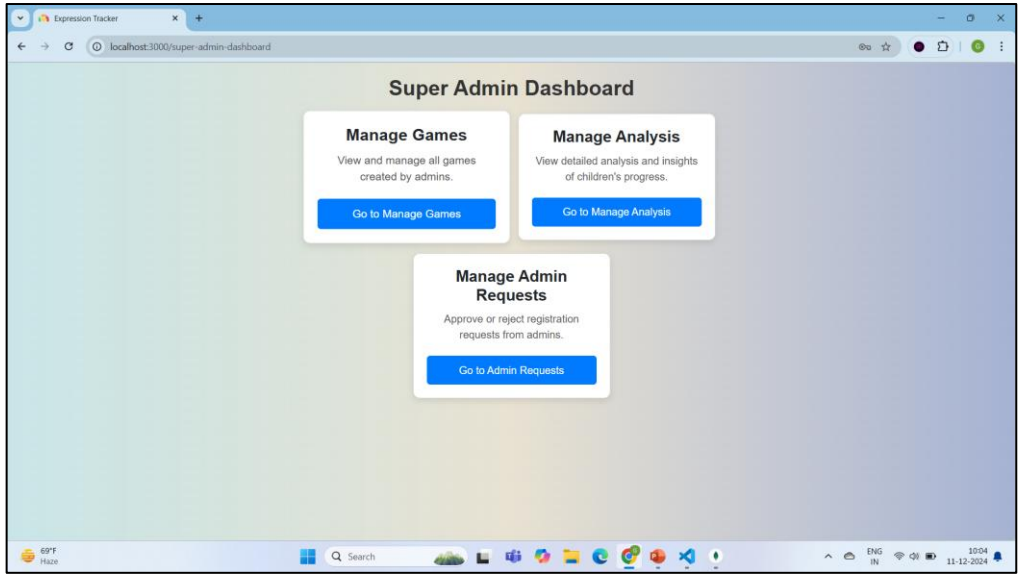






SUPERADMIN PAGES





Code Snippets

src > hooks > JS useCapture.js > [e] useCapture > [e] captureImage

```
5   const useCapture = (videoRef) => {
9     const captureImage = (newSessionId, sessionName, gameName) => {
11      const video = videoRef.current;
12      const canvas = canvasRef.current;
13
14      if (!canvas) {
15        console.error("Canvas is not available.");
16        return;
17      }
18
19      const context = canvas.getContext("2d");
20      if (!context) {
21        console.error("Canvas context is not available.");
22        return;
23      }
24
25      context.drawImage(video, 0, 0, canvas.width, canvas.height);
26
27      canvas.toBlob((blob) => {
28        if (!blob) {
29          console.error("Failed to create Blob from canvas.");
30          return;
31        }
32
33        const formData = new FormData();
34        formData.append("image", blob, "capture.png");
35        formData.append("newSessionId", newSessionId);
36        formData.append("sessionName", sessionName);
37        formData.append("gameName", gameName);
38
39        console.log("In useCapture:", newSessionId, sessionName, gameName);
40        axios
41          .post("http://localhost:5000/child/uploads", formData)
42          .then((response) => console.log("Image uploaded:", response.data))
43          .catch((error) => console.error("Error uploading image:", error));
44      });
45    };
```

Storing captured images

ontrollers > JS admin_helper.js > [e] getSessionMedia

```
6   const MODEL_URL =
7     "https://api-inference.huggingface.co/models/trpakov/vit-face-expression";
8
9   // Helper function to get session media
10  const getSessionMedia = async (sessionId) => {
11    try {
12      // Fetch the session by sessionId from MongoDB
13      const session = await Session.findOne({ sessionId: sessionId });
14      if (!session) {
15        return { error: "Session not found" };
16      }
17      console.log(
18        "this are the image paths in getSessionMedia function",
19        session.imagePaths
20      );
21
22      // Return imagePaths (add screenshotPaths if necessary)
23      return {
24        imagePaths: session.imagePaths || [],
25      };
26    } catch (error) {
27      console.error("Error fetching media:", error);
28      return { error: "Failed to fetch media" };
29    }
30  };
31
32  // Helper function to send the image to Hugging Face model
33  async function sendImageToModel(imageBuffer, retries = 5, delay = 5000) {
34    // Convert the image buffer to base64 encoding
35    console.log("SendImageToModel function called ");
36    const base64Image = imageBuffer.toString("base64");
37
38    for (let i = 0; i < retries; i++) {
39      try {
40        console.log("this is the token", process.env.HUGGING_FACE_API_KEY);
41
```

Sending images to model

src > components > JS PendingRequests.js > [e] PendingRequests > useEffect() callback > [e] getAdminStatus > [e] filteredAdmi

```
6   const PendingRequests = () => {
10    useEffect(() => {
11      const getAdminStatus = async () => {
12        try {
13          const response = await axios.get(
14            `${process.env.REACT_APP_BACKEND_URL}/auth/getAllStatus`
15          );
16          const filteredAdmins = response.data.admin_info?.filter(
17            (admin) => admin.admin_email !== "superadmin@example.com"
18          ) || [];
19
20          setAdmins(filteredAdmins || []);
21
22        } catch (error) {
23          console.log("Failed to fetch details: ", error);
24        }
25      };
26      getAdminStatus();
27    }, []);
28
29    // Handle approval action
30    const handleApproval = async (adminId) => {
31      const req_body = { status: "Approved" };
32      try {
33        await axios.patch(
34          `${process.env.REACT_APP_BACKEND_URL}/auth/updateAdminStatus/${adminId}`,
35          req_body
36        );
37        setAdmins((prevAdmins) =>
38          prevAdmins.map((admin) =>
39            admin._id === adminId ? { ...admin, status: "Approved" } : admin
40          )
41        );
42      } catch (error) {
43        console.log("Failed to approve admin: ", error);
44      }
45    };
```

Authorization of admins