NeuroDetective - project document

Michał Korwek, Paulina Kulczyk, Jan Pogłód

1. Introduction

1.1. Motivation

According to the World Health Organization (2022), approximately **one in eight people globally live with a mental disorder**. A significant proportion remain undiagnosed, particularly in low-income settings or environments where stigma around mental health is prevalent.

Several psychological conditions—such as **ADHD, ADD, autism spectrum disorders, anxiety, and depression**—are frequently overlooked in children. Their symptoms are often misinterpreted as behavioral problems or lack of effort, rather than signs of a mental health challenge. Without early identification and tailored support, these children may struggle academically and socially, which can affect their long-term development and life outcomes (Barbaresi et al. (2007); Lord et al., 2020).

Our tool aims to **support the early detection of such conditions, focusing primarily on children**. Early identification and appropriate guidance can significantly improve long-term outcomes, enabling individuals to thrive and reach their full potential—often comparable to those without such diagnoses (CDC, 2024).

1.2. Project Overview

NeuroDetective is a digital platform designed to **detect ADD/ADHD, autism spectrum, anxiety and depression symptoms in kindergarten and elementary school children** using a combination of **machine learning algorithms**, **mobile games**, and **teacher-filled questionnaires**. The app is used under teacher and parental supervision and captures behavioral and cognitive data during gameplay.

The project is built using **Google Cloud Platform**, leveraging services like **Cloud SQL**, **Cloud Run**, **Artifact Registry**, and **Terraform** for full infrastructure-as-code deployment (see section 7). It consists of two main user-facing components: a **mobile application** (section 6), which children interact with daily through cognitive and emotional games, and a **web application** (section 5), which is used by parents and teachers to monitor progress and input survey data. The backend exposes a well-documented **REST API** (section 3), with strong data modeling and relational integrity in PostgreSQL (section 4). Data flows securely between the mobile app, backend, and storage using authenticated API endpoints. Storage is optimized using both **SQL for persistent records** and **Redis for high-speed lookups**.

Table of Contents

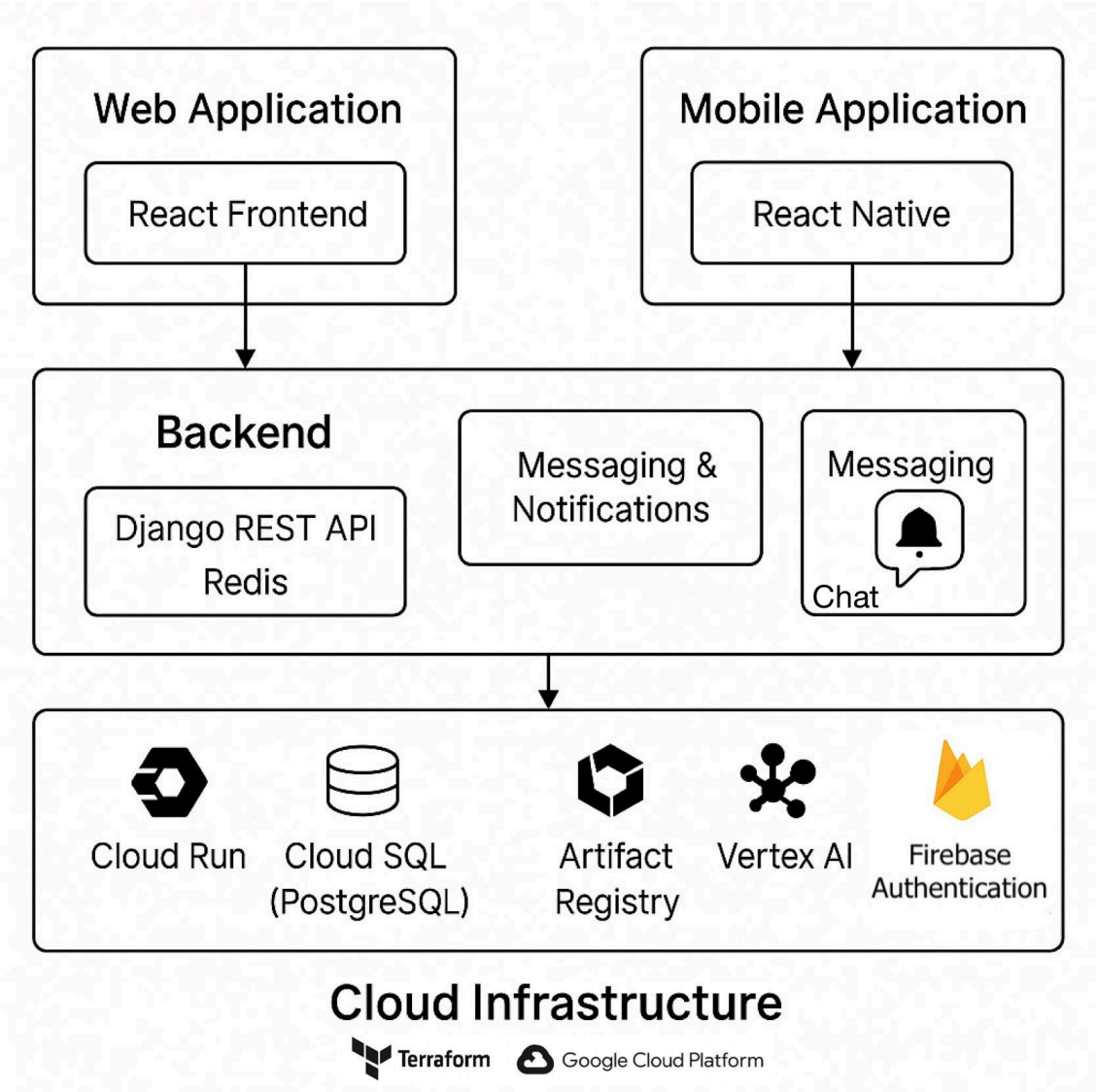
1. Introduction 1
    1.1. Motivation 1
    1.2. Project Overview 1
2. Technology Stack 3
3. Component Interactions 4
        3.1 Web Application (React Frontend) 4
        3.2 Mobile Application (React Native) 4
        3.3 Backend Services (Django + REST API) 4
        3.4 Messaging and Notifications (Firebase Cloud Messaging) 4
        3.5 Cloud Infrastructure (Google Cloud Platform) 4
        3.6 AI Integration (Vertex AI) 5
4. Storage Characteristics 5
    4.1 Primary Storage: Cloud SQL (PostgreSQL) 5
        4.1.1 Database Schema Overview 5
            a) Teachers 5
            b) Parents 5
            c) Students 5
            d) Game Results 6
            e) Questionnaire Results 6
        4.1.2 Relational Structure (Foreign Key Relationships) 6
        4.1.3 Characteristics Summary 6
    4.2 Secondary Storage: Redis (In-Memory) 8
        4.2.1. Characteristics Summary 8
5. Web Application Interface 8
    5.1 Navigation Structure 8
6. Tablet / Mobile Application 9
    6.1 Functionality 9
        a) Game 1: Shapes Recognition 9
        b) Game 2: Emotions 10
        c) Game 3: Writing / Drawing 10
7. Terraform Infrastructure 10
    7.1 Core Terraform Modules 10
8. SLA / SLO / SLI 11
9. Testing & Validation 11

2. Technology Stack

The system leverages a modern, cloud-native architecture, combining robust backend services, interactive frontends, and scalable infrastructure. The technology stack includes:

- **Frontend**: React (Web), React Native (Mobile)
- **Backend**: Django with Django REST Framework
- **Authentication**: Firebase Authentication
- **Infrastructure**: Terraform, Docker, Cloud Run
- **Databases**: Google Cloud SQL (PostgreSQL), Redis (In-memory)
- **AI Services**: Vertex AI
- **DevOps**: Artifact Registry, CI/CD pipelines
- **Messaging & Event Handling**: Firebase Cloud Messaging (FCM)

Infrastructure provisioning is declaratively managed via **Terraform**, enabling reproducible deployments and secure resource configuration on **Google Cloud Platform (GCP)**.



3. Component Interactions

3.1 Web Application (React Frontend)

The web frontend is built with React and interfaces with the Django backend via a RESTful API.

- **Authentication**: Firebase Authentication handles login and user sessions. Tokens are passed in HTTP headers for secure backend access.
- **API Requests**: The frontend consumes REST endpoints for users, assessments, feedback reports, and file uploads.
- **User Interaction**: Teachers and parents use the web interface to view reports, submit evaluations, and receive system notifications.

3.2 Mobile Application (React Native)

Designed for child users, the mobile app delivers gamified interactions and captures behavioral data.

- **Game Interaction**: Children play cognitive and emotional games, and the results are recorded locally and sent to the backend via RESTful endpoints.
- **Authentication**: Firebase Authentication is used for identity management tailored to underage users.
- **Data Submission**: Structured results from games and assessments are transmitted to Cloud SQL through the backend.

3.3 Backend Services (Django + REST API)

The backend is built with Django and Django REST Framework, exposing secure REST endpoints for all client apps.

- **Business Logic**: Handles user roles, test processing, data validation, and access control.
- **Database Interaction**: Communicates with Cloud SQL for persistent data storage, including test results, survey records, and profile data.
- **Security**: Authentication tokens are verified against Firebase. Cross-Origin Resource Sharing (CORS) is enforced to control frontend access.

3.4 Messaging and Notifications (Firebase Cloud Messaging)

Firebase Cloud Messaging is used to propagate **event-based notifications from the frontend/mobile clients to the backend**.

- **Use Case**: Upon specific user actions (e.g., game completion or survey submission), tokens are sent to the backend to trigger downstream processing such as logging, model retraining, or notifications to educators.

3.5 Cloud Infrastructure (Google Cloud Platform)

- **Cloud Run**: Hosts the Django backend in a containerized, serverless environment, auto-scaling based on traffic volume.
- **Cloud SQL**: Acts as the central relational database, storing application state and domain-specific data.
- **Artifact Registry**: Stores Docker images of the backend for continuous delivery and version control.
- **Redis**: Used as an optional caching layer for fast access to ephemeral data such as teacher-class mappings or recent queries.

3.6 AI Integration (Vertex AI)

Vertex AI is integrated to provide predictive analytics and model-based feedback for student performance.

- **Data Ingestion**: Receives anonymized student and test data from Cloud SQL.
- **Inference**: Returns insights and alerts about potential learning challenges or emotional patterns.
- **Retraining**: The system is capable of model retraining based on updated game and assessment data.

4. Storage Characteristics

4.1 Primary Storage: Cloud SQL (PostgreSQL)

The core application data is stored in a managed PostgreSQL instance hosted on **Google Cloud SQL**. This relational database system supports **structured data**, **strong consistency**, and **ACID-compliant transactions**, making it suitable for managing entities such as students, parents, test results, and interactions.

4.1.1 Database Schema Overview

The PostgreSQL schema includes:

a) **Teachers**

Stores personal and professional information about teachers:

- Fields: name, surname, education level, subject, age, gender, class, school name.
- Primary key: id_teacher.

b) **Parents**

Stores contact and demographic info:

- Fields: name, surname, occupation, contact number, email.
- Primary key: id_parent.

c) **Students**

Holds student-specific data:

- Linked to parents and teachers via foreign keys (id_parent, class).
- Fields include grades, behavior average, teacher notes.
- Primary key: id_student.

d) **Game Results**

Performance metrics from game-based testing:

- Linked to students and games.
- Stores multiple performance metrics and completion time.
- Primary key: id_result_games.

e) **Questionnaire Results**

Quantitative feedback and observations:

- Contains answers to 5 predefined questions.
- Linked to both student and teacher IDs.
- Primary key: id_result_quests.

4.1.2 Relational Structure (Foreign Key Relationships)

- **Reference parents_students:** parents.parent_id → students.parent_id
  One-to-many – One parent can be linked to multiple students.

- **Reference teachers_students:** teachers.class → students.class
  Many-to-one – Many students can belong to the same class, assigned to one teacher.

- **Reference students_game_results:** game_results.student_id → students.student_id
  One-to-many – Each student can have multiple game results.

- **Reference questionnaire_results_students:** questionnaire_results.student_id → students.student_id
  Many-to-one – Each questionnaire result is linked to one student.

- **Reference questionnaire_results_teachers:** questionnaire_results.teacher_id → teachers.teacher_id
  Many-to-one – Each questionnaire result is linked to one teacher.

4.1.3 Characteristics Summary

Attribute	Description
Storage Type	Structured SQL
Database Engine	PostgreSQL (Cloud SQL)
Data Model	Relational with foreign key constraints
Consistency Model	Strong consistency (default for PostgreSQL)
Access Pattern	Read & Write
Estimated Volume	Medium — thousands of rows per table
Write Frequency	Moderate (on data entry, testing, survey completion)
Read Frequency	High for analytics and dashboards
Backup Strategy	Automated daily backups (Terraform managed)
Security	IAM, SSL connections, encryption at rest (GCP defaults)

4.2 Secondary Storage: Redis (In-Memory)

For fast access and low-latency operations, an **in-memory Redis** instance is used. Redis serves primarily as a **cache** and **temporary data store**, particularly useful for mapping class-teacher relationships, session tokens, and real-time data where persistence is not critical.

4.2.1. Characteristics Summary

Attribute	Description
Use Case	Rapid access to teacher-class mapping
Storage Type	Key-Value (in-memory)
Data Volume	Low (few hundred entries)
Access Pattern	High-frequency reads, occasional writes
Persistence	Optional (can be disabled for performance)
Consistency	Eventual / best-effort in Redis context
Tooling	Managed Redis instance (MemoryStore)

5. Web Application Interface

The primary interface for end-users is the web application, developed using React. Access is gated based on authentication and authorization via Firebase Authentication.

5.1 Navigation Structure

Before login, the top navigation bar includes:

- Home
- About Us
- PsychoSphere
- Login / Register

After successful login, options are adjusted dynamically based on the user's role:

- Parent: Child
- Child: My Humor, Games, Chat
- Teacher: Class, Chat

Additionally, once logged in, the Login/Register button is replaced with Logout, adhering to intuitive UX standards.

6. Tablet / Mobile Application

The mobile app is developed in **React Native with Expo**, allowing rapid cross-platform deployment to both **iOS (App Store)** and **Android (Google Play)**.

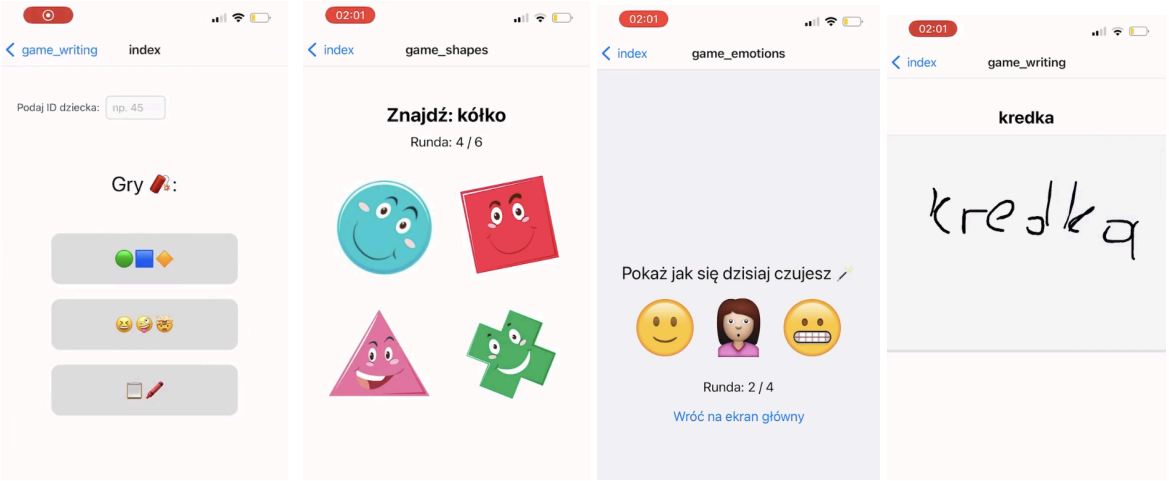
6.1 Functionality

The app is designed specifically for children and features:

- A login entry where parents input a child's ID.
- A menu screen enabling the child to access three interactive games.
- Automatic transmission of gameplay data to **Cloud SQL** for analytics and performance tracking.

The app communicates with the backend using secure RESTful API calls. Firebase Authentication manages user sessions on mobile as well.

This app is designed for **children**, who use it daily to play a set of games. The entry point is a simple menu screen (see first left screenshot below), where a **parent enters the child's ID**, after which the child can proceed to play one of three available games:



---

a) Game 1: Shapes Recognition

- The game contains **6 rounds**.
- In each round, the child must select the correct shape from multiple colorful options (e.g., "Find: Circle").
- This game is designed to **assess the child's focus and attention**, as supported by educational research.
- After playing, the following data is sent to the backend via API:

  { "student_id": X, "correct": N1, "incorrect": N2, "time_spent": T }

b) Game 2: Emotions

- This game has **4 rounds**.
- In each round, the child selects one emoji that best represents their **current emotional state**.
- Emotions are categorized into **Happy, Sad, and Angry**.
- This activity helps track emotional trends and social-emotional development over time.
- After completion, the app sends:

  { "student_id": X, "happy": N1, "sad": N2, "angry": N3, "time_spent": T }

c) Game 3: Writing / Drawing

- A word appears at the top of the screen (e.g., "crayon"), and the child is instructed to **write it using their finger or stylus**.
- The game evaluates both **spelling accuracy** and **writing behavior**, including stroke pressure and shape.
- In future versions, this game will incorporate **emotional journaling** as part of behavioral analysis.

- The app sends:

  { "student_id": X, "image": (.png file), "time_spent": T }

Later, all the data is saved to a game_results file, and based on it, using machine learning methods, the cases in which children may have a higher chance of having adhd are classified.

7. Terraform Infrastructure

Infrastructure provisioning and configuration management is fully automated using Terraform, following Infrastructure as Code (IaC) principles. The following modules and resources are defined:

7.1 Core Terraform Modules

- Cloud Run (Django backend)
  Serverless compute platform that scales automatically based on demand.
- Cloud SQL (PostgreSQL)
  Managed relational database service with high availability and regular backups.
- Artifact Registry
  Stores and versions Docker images of the backend. Images are referenced during CI/CD pipelines and deployed to Cloud Run or GKE.
- IAM Roles and Service Accounts
  Access control policies that ensure secure and limited resource access per service.
- GCS — Frontend Hosting
  Static frontend hosting for the React web app via a GCS bucket configured for static website delivery. Files are built with Vite and deployed using *gsutil rsync*.

8. SLA / SLO / SLI

To ensure quality of service, we define performance commitments and objectives.

Service Commitment:
If the service is unavailable beyond the agreed threshold, clients will receive a 10% discount.

Service	Service Level Agreement (SLA)	Service Level Objective (SLO)	Service Level Indicator (SLI)
Backend API	98% uptime per 7-day rolling period	99.9% daily availability	% of successful API request responses
Web Application	95% overall availability .	99% daily page load success	% of web pages loaded and fully functional

Mobile Application	95% daily uptime	98% game data submission success	% of gameplay data submitted to backend successfully
Authentication System	99% uptime	99.9% login success rate	% of completed login attempts via Firebase Auth
Messaging & Alerts	95% timely delivery (≤ 5s)	98% success rate	% of Firebase Cloud Messages delivered within 5 seconds
Data Storage (PostgreSQL)	99.5% uptime	99.9% successful read/write ops	% of database transactions completed successfully

9. Testing & Validation

Types of Testing:

- Unit tests (Django)
- Integration tests (API + DB)
- Frontend E2E tests (Cypress, etc.)