Final Project Report

Group Name:- DSI_Lite-x-RCB
Members: Aman Kesharwani (2023102001) , Mohd Harish (2023102003) , Aaditya Bhatia (2023114012), Krish Agarwal (2023113017), Amit Sutradhar (2024702014), Ashvin Vinod (2022101015), Aadi Deshmukh (2022101018)

1. Problem Statement

Challenge: Manual child anthropometric measurements are time-consuming, error-prone, and require trained personnel. Rural healthcare providers lack resources for accurate growth monitoring.

Solution: AI-powered mobile application that:

- Captures child photos using smartphone camera
- Automatically measures height, head circumference, and wrist circumference
- Compares against WHO growth standards
- Provides malnutrition risk assessment
- Maintains digital health records

Impact: Enables early detection of malnutrition, reduces healthcare costs, and improves child health outcomes in resource-constrained settings.

2. Architecture

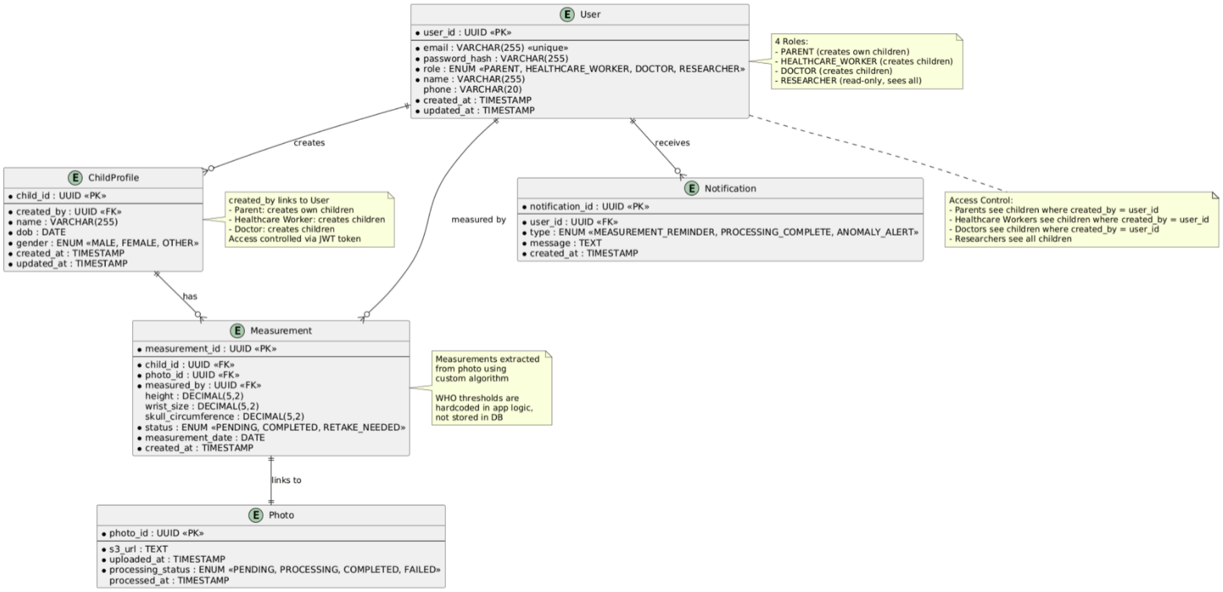
2.1 Technical Architecture

```
┌─────────────────────────────────────────────────────────────────
──┐
│                 CLIENT LAYER                            │
│  React Native (Expo) - iOS & Android & Web                 │
│  - Expo Router for navigation                           │
│  - Firebase Authentication                              │
│  - Image capture via expo-image-picker                    │
└────────────────────────────┬────────────────────────────────────
──┘
             │ REST API (HTTP/JSON)
             │
┌────────────────────────────▼───────────────────────────────────
───┐
│               API LAYER (Backend)                        │
│  Flask (Python 3.10) + CORS                               │
│  - /health - Health check endpoint                        │
│  - /predict - Image processing and prediction               │
│  - /predict-url - URL-based prediction                     │
└────────────────────────────┬────────────────────────────────────
──┘
             │
┌────────────────────────────▼───────────────────────────────────
───┐
│              ML/CV PIPELINE                             │
│  MediaPipe (Google) + OpenCV                              │
│  - Pose detection (33 landmarks)                          │
│  - Image preprocessing & calibration                       │
│  - Measurement calculation algorithms                       │
└────────────────────────────┬────────────────────────────────────
──┘
             │
┌────────────────────────────▼───────────────────────────────────
───┐
│               DATABASE LAYER                             │
│  Firebase Firestore (NoSQL)                               │
│ Collections: users, children, measurements, growth_data │
│ Firebase Storage: Image uploads │
└─────────────────────────────────────────────────────────────┘
```

2.2 ML Pipeline Details

```
Input Image (Child Photo)
    │
    ▼
┌─────────────────────────────┐
│ 1. Image Preprocessing      │
│ - Resize & normalize        │
│ - RGB conversion            │
│ - Quality validation        │
└──────────────┬──────────────┘
               │
               ▼
┌─────────────────────────────┐
│ 2. MediaPipe Pose Detection │
│ - Detect 33 body landmarks  │
│ - Extract key points:       │
│   * Head (nose, ears)       │
│   * Shoulders, hips, ankles │
│   * Wrists                  │
└──────────────┬──────────────┘
               │
               ▼
┌─────────────────────────────┐
│ 3. Calibration              │
│ - Calculate pixel-to-cm ratio │
│ - Use shoulder width reference │
│ - Apply scaling factor      │
└──────────────┬──────────────┘
               │
               ▼
┌─────────────────────────────┐
│ 4. Measurement Calculation  │
│ - Height: ankle to head top │
│ - Head circumference: ellipse │
│ - Wrist circumference: width │
└──────────────┬──────────────┘
               │
               ▼
┌─────────────────────────────┐
│ 5. WHO Standard Comparison  │
│ - Load age/gender growth chart │
│ - Calculate percentiles     │
│  - Classify nutritional status  │
└──────────────────────────┬──────────────────────────┘
             │
             ▼
Output: {height, head_circ, wrist_circ, status}
```

3. ER Diagram



E User
• user_id : UUID «PK»
• email : VARCHAR(255) «unique»
• password_hash : VARCHAR(255)
• role : ENUM «PARENT, HEALTHCARE_WORKER, DOCTOR, RESEARCHER»
• name : VARCHAR(255)
phone : VARCHAR(20)
• created_at : TIMESTAMP
• updated_at : TIMESTAMP
4 Roles:
- PARENT (creates own children)
- HEALTHCARE_WORKER (creates children)
- DOCTOR (creates children)
- RESEARCHER (read-only, sees all)
creates
receives
measured by
E ChildProfile
• child_id : UUID «PK»
• created_by : UUID «FK»
• name : VARCHAR(255)
• dob : DATE
• gender : ENUM «MALE, FEMALE, OTHER»
• created_at : TIMESTAMP
• updated_at : TIMESTAMP
created_by links to User
- Parent: creates own children
- Healthcare Worker: creates children
- Doctor: creates children
Access controlled via JWT token
E Notification
• notification_id : UUID «PK»
• user_id : UUID «FK»
• type : ENUM «MEASUREMENT_REMINDER, PROCESSING_COMPLETE, ANOMALY_ALERT»
• message : TEXT
• created_at : TIMESTAMP
Access Control:
- Parents see children where created_by = user_id
- Healthcare Workers see children where created_by = user_id
- Doctors see children where created_by = user_id
- Researchers see all children
has
E Measurement
• measurement_id : UUID «PK»
• child_id : UUID «FK»
• photo_id : UUID «FK»
• measured_by : UUID «FK»
height : DECIMAL(5,2)
wrist_size : DECIMAL(5,2)
skull_circumference : DECIMAL(5,2)
• status : ENUM «PENDING, COMPLETED, RETAKE_NEEDED»
• measurement_date : DATE
• created_at : TIMESTAMP
Measurements extracted
from photo using
custom algorithm
WHO thresholds are
hardcoded in app logic,
not stored in DB
links to
E Photo
• photo_id : UUID «PK»
• s3_url : TEXT
• uploaded_at : TIMESTAMP
• processing_status : ENUM «PENDING, PROCESSING, COMPLETED, FAILED»
processed_at : TIMESTAMP

Entity Relationship Diagram (High-Level)

Core Entities & Relationships

User (1) ──creates──> (N) ChildProfile
ChildProfile (1) ──has──> (N) Measurement
Measurement (1) ──links to──> (1) Photo
User (Healthcare Worker/Doctor) (1) ──creates──> (N) ChildProfile
User (1) ──receives──> (N) Notification
Key Entities:

- User: Stores parent, healthcare worker, doctor, researcher accounts
- ChildProfile: Child's basic information linked to creator (parent or healthcare worker)
- Measurement: Growth measurements (height, wrist size, skull circumference)
- Photo: Stores S3 URL and processing status

- Notification: User alerts and reminders

4. User Workflows

4.1 Parent/Guardian Flow

- START → App Launch
- │
- ├─ Not Authenticated
- │ └─► Login/Register Screen
- │   ├─ Enter email/password
- │   ├─ Select role: Parent
- │   └─► Registration Success → Dashboard
- │
- └─ Authenticated
-     └─► Dashboard (Home)
-         ├─ View Children List
-         ├─ View Recent Measurements
-         │
-         ├─► Add New Child
-         │   ├─ Enter name, DOB, gender
-         │   ├─ Optional: Upload photo
-         │   └─► Child Profile Created
-         │
-         ├─► Select Child → Child Details
-         │   ├─ View Growth Chart
-         │   ├─ View Measurement History
-         │   │
-         │   └─► Take New Measurement
-         │   ├─ Camera Permission Request
-         │   ├─ Capture/Upload Child Photo
-         │   ├─ Image Processing (Loading)
-         │   ├─ ML Prediction
-         │   └─► Results Screen
-         │   ├─ Height, Head Circ, Wrist Circ
-         │   ├─ WHO Percentile
-         │   ├─ Nutritional Status
-         │   ├─ Growth Chart Update
-         │   └─ [Save to History]

- │
- └─▶ Profile Settings
-    ├─ Edit Profile
-    ├─ Notification Settings
-    └─ Logout
-

4.2 Doctor Flow

- START → App Launch
-   │
-   └─▶ Doctor Login
-       └─▶ Doctor Dashboard
-           ├─ View All Patients (Children)
-           ├─ Search/Filter Patients
-           ├─ View Statistics Dashboard
-           │   ├─ Total patients
-           │   ├─ At-risk children
-           │   └─ Recent measurements
-           │
-           ├─▶ Add New Patient
-           │   ├─ Enter patient details
-           │   ├─ Assign parent/guardian
-           │   └─▶ Patient Created
-           │
-           ├─▶ Select Patient
-           │   ├─ View Complete Medical History
-           │   ├─ View Growth Trends (Charts)
-           │   ├─ View All Measurements
-           │   │
-           │   ├─▶ Conduct Examination
-           │   │   ├─ Take Measurement Photo
-           │   │   ├─ ML Processing
-           │   │   ├─ Review Results
-           │   │   ├─ Add Clinical Notes
-           │   │   └─▶ Save to EHR
-           │   │
-           │   └─▶ Generate Report
-           │   ├─ Select date range
-           │   ├─ Export PDF
-           │   └─ Share with parent
-           │

- └─► Settings
- ├─ Clinic Information
- ├─ Professional Profile
- └─ Logout

4.3 System Administrator Flow

- START → Admin Login
- │
- └─► Admin Dashboard
- ├─ System Statistics
- │ ├─ Total users
- │ ├─ Total measurements
- │ ├─ API usage metrics
- │ └─ Storage usage
- │
- ├─► User Management
- │ ├─ View all users
- │ ├─ Approve doctor accounts
- │ ├─ Suspend/activate accounts
- │ └─ Reset passwords
- │
- ├─► Data Management
- │ ├─ WHO growth data updates
- │ ├─ Backup database
- │ └─ Data export
- │
- ├─► ML Model Management
- │ ├─ View model version
- │ ├─ Update model parameters
- │ └─ Performance metrics
- │
- └─► System Logs
- ├─ API logs
- ├─ Error logs
- └─ Audit trail

5. APIs (Signatures, Status Codes, response, requests, etc)

5.1 Health Check

Endpoint: `GET /health`

Description: Check if API server is running

Response (200 OK):

```
{
  "status": "ok",
  "message": "Anthropometry API is running"
}
```

Status Codes:

- `200 OK` - Server is healthy
- `500 Internal Server Error` - Server is down

5.2 Predict Measurements (Base64)

Endpoint: `POST /predict`

Description: Process base64-encoded image and return anthropometric measurements

Request Body:

```
{
  "image": "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD..."
}
```

Success Response (200 OK):

```
{
  "success": true,
  "measurements": {
    "height_cm": 105.3,
    "head_circumference_cm": 48.7,
    "wrist_circumference_cm": 11.2,
    "pixel_per_cm": 3.547
  }
}
```

Error Response (400 Bad Request):

```
{
  "success": false,
  "error": "No person detected in image"
}
```

Status Codes:

- `200 OK` - Successfully processed image
- `400 Bad Request` - Invalid image or no person detected
- `500 Internal Server Error` - Server-side processing error

5.3 Predict Measurements (URL)

Endpoint: `POST /predict-url`

Description: Process image from URL and return measurements

Request Body:

```
{
  "url": "https://example.com/child-photo.jpg"
}
```

Success Response (200 OK):

```
{
  "success": true,
  "measurements": {
    "height_cm": 98.5,
    "head_circumference_cm": 47.2,
    "wrist_circumference_cm": 10.8,
    "pixel_per_cm": 3.421
  }
}
```

Error Response (400 Bad Request):

```
{
  "success": false,

  "error": "Failed to decode image from URL"

}
```

Status Codes:

- `200 OK` - Successfully processed image
- `400 Bad Request` - Invalid URL or image format
- `500 Internal Server Error` - Network or processing error

5.4 Firebase APIs (Frontend)

Used via Firebase SDK:

Authentication:

- `createUserWithEmailAndPassword()` - Register new user
- `signInWithEmailAndPassword()` - Login user
- `signOut()` - Logout user
- `onAuthStateChanged()` - Listen for auth changes

Firestore Database:

- `collection('users').doc(userId).set()` - Create user profile
- `collection('children').add()` - Add child
- `collection('measurements').add()` - Save measurement
- `collection('children').where('parentId', '==', userId).get()` - Query children

Storage:

- `ref(storage, 'images/...')` - Upload child photos
- `getDownloadURL()` - Get public image URL

6. Folder Structure

```
Dsi-Anthropometry-Project/
│
├── application/                 # Frontend (React Native)
│   ├── app/                     # Expo Router pages
│   │   ├── _layout.tsx          # Root layout
│   │   ├── home.tsx             # Home/Landing page
│   │   ├── LoginRegister.tsx    # Auth screen
│   │   └── (tabs)/              # Tab navigation
│   │       ├── _layout.tsx      # Tab layout
│   │       ├── dashboard.jsx    # Parent dashboard
│   │       ├── doctor_dashboard.jsx  # Doctor dashboard
│   │       ├── profile.jsx      # User profile
│   │       ├── anthroscan.jsx   # Camera/scan screen
│   │       └── results.jsx      # Results display
│   │
│   ├── components/              # Reusable UI components
│   ├── config/                  # Configuration
│   │   └── firebase.ts          # Firebase config
│   │
│   ├── services/                # API services
│   │   └── api.ts               # Backend API calls
│   │
│   ├── contexts/                # React contexts
│   │   └── AuthContext.tsx      # Authentication context
│   │
│   ├── assets/                  # Static assets
│   │
│   ├── ml_cv/                   # Backend (Python ML/CV)
│   │   ├── api_server.py        # Flask REST API
│   │   ├── child.py             # ML processing logic
│   │   ├── requirements.txt     # Python dependencies
│   │   └── Dockerfile           # Backend container
│   │
│   ├── package.json             # Node dependencies
│   ├── app.json                 # Expo config
│   └── Dockerfile               # Frontend container
│
├── docker-compose.yml           # Multi-container orchestration
│
├── docs/                        # Documentation
│   └── HACKATHON_REPORT.md      # This file
│
└── README.md                    # Project overview
```

7. Setup Instructions (Start to end How-to-run commands)

Prerequisites

- Docker Desktop (Mac/Windows) or Docker Engine (Linux)
- Node.js 18+ (for local development)
- Python 3.10+ (for local development)
- Git
- Expo Go app (for mobile testing)

7.1 Quick Start with Docker (Recommended)

Step 1: Clone Repository

```
git clone https://github.com/Mohd-Harish123/Dsi-Anthropometry-Project.git
cd Dsi-Anthropometry-Project
```

Step 2: Start All Containers

```
docker-compose up -d
```

Step 3: Verify Services

```
# Check container status
docker-compose ps

# Test backend
curl http://localhost:5001/health

# Test frontend
open http://localhost:8081
```

Step 4: Access Application

- **Web:** http://localhost:8081
- **Mobile:** http://localhost:19000 (scan QR code with Expo Go)
- **Backend API:** http://localhost:5001

Step 5: Stop Containers

```
docker-compose down
```

7.2 Testing the System

Test 1: Backend Health

Bash

```
curl http://localhost:5001/health
```

Expected: `{"status": "ok", "message": "Anthropometry API is running"}`

Test 2: ML Prediction

Bash

```
# Download sample image
curl -o test-child.jpg https://example.com/child-photo.jpg

# Convert to base64
base64 test-child.jpg > image.b64

# Send to API
curl -X POST http://localhost:5001/predict \
  -H "Content-Type: application/json" \
  -d '{"image": "'$(cat image.b64)'"}'
```

Test 3: Frontend Web

1. Open http://localhost:8081
2. Click "Get Started"
3. Register new account
4. Add child profile
5. Upload test image
6. View results

Test 4: Mobile (Android/iOS)

1. Install Expo Go app
2. Open http://localhost:19000
3. Scan QR code with Expo Go
4. App loads on mobile device
5. Test full workflow

8. Individual Contributions

Aadi Deshmukh, 2022101018 - Worked on the frontend template and documentation

Mohd Harish, 2023102003 – Implemented Firebase auth with role-based registration and worked on the frontend.

Aman kesharwani, 2023102001- Firebase integration, fixed issues with the previous non-functional UI, Additionally contributed to drafting project documentation, identifying stakeholders, and shaping the initial system requirements.

Ashvin Vinod Kaimal, 2022101015- worked on the ML model to predict the height and head and wrist circumference from the image.

Amit Sutradhar, 2024702014- Build wireframe for the UI
Aaditya Bhatia, 2023114012- Fixed the ML model, integrated the backend to the mobile app and dockerize the code

Krish Agarwal, 2023113017- Implemented the UI on the mobile app with role based features, documentation, integration of ML model to the app