Formative 1: Database & Prediction Pipeline - Report

Team Number: 15
Live Demo: https://pipeline-database.onrender.com/docs
GitHub Repository: https://github.com/reponseashimwe/ml_pipeline_database

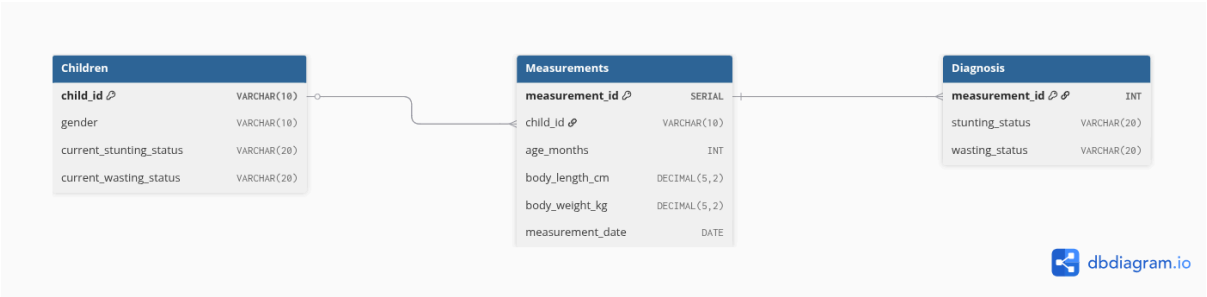
1. Problem Statement

Child malnutrition, specifically stunting, represents a significant global health challenge with long-term implications for child development and public health. Accurate and timely identification of malnutrition is crucial for effective intervention strategies. This project addresses the need for a robust data management and analysis system that can store child anthropometric data, facilitate CRUD operations, and integrate a machine learning model to predict stunting statuses (and wasting in future), thereby supporting data-driven decision-making for health practitioners.

2. Architecture

The project employs a multi-database backend architecture exposed via a FastAPI RESTful API, designed for scalability and flexibility.

- **API Layer:**
  - **FastAPI (Python):** Serves as the primary interface, handling all incoming requests for data management (CRUD)
  - **Uvicorn:** The ASGI server used to run the FastAPI application.
- **Database Layer:**
  - **MySQL (Relational Database):**
    - **Purpose:** Primary transactional data store for structured records.
    - **Hosting:** Deployed on Railway.



  - **MongoDB (NoSQL Document Database):**
    - **Purpose:** Offers flexible document storage, suitable for raw data ingestion, or potentially storing less structured information.
    - **Hosting:** MongoDB Atlas
- **Machine Learning Integration:**

    - **Random Forest Classifier:** A pre-trained model (from https://github.com/reponseashimwe/model_training_evaluation) integrated into the FastAPI application.
    - **Deployment:**
        - **Render.com:** The cloud platform used to deploy the FastAPI application, providing automated builds and continuous deployment from the GitHub repository.

Data Flow: CSV Dataset -> (Data Loading) -> MySQL & MongoDB -> (API Endpoints) -> FastAPI App -> (Prediction Endpoint) -> ML Model -> (API Response)

3. Model Used

The project integrates a pre-trained **Random Forest Classifier** model.

- **Model Source:** Link here
- **Purpose:** To predict stunting_status based on child anthropometric measurements (age_months, body_length_cm, body_weight_kg).

4. Preprocessing

Preprocessing is handled through a scikit-learn pipeline integrated into the ML prediction system:

Data Transformations:

- *Numerical Features*: StandardScaler for age_months, body_length_cm, body_weight_kg
- *Categorical Features*: OneHotEncoder for gender (Laki-laki/Perempuan)
- *Missing Values*: Median imputation for numerical data, constant fill for categorical
- *Feature Engineering*: Automated preprocessing pipeline that ensures consistency between training and prediction

5. Triggers and stored procedures

MySQL triggers are implemented to automate database actions, ensuring data integrity and efficient data management. These use stored procedures respectively.

Stored Procedures:

- **GenerateChildUniqueID()**
    - **Purpose**: Creates unique child identifiers in format YYYYMMDD-HHMMSS-XXXX. This ensures that every new child record receives a consistent, unique, and human-readable identifier without

requiring the application layer to explicitly generate it, improving data integrity and simplifying application logic.
    - **Implementation**: Uses current timestamp with MD5 hash suffix for uniqueness
- **SetGenderText**()
    - **Purpose**: Maps Indonesian gender values to English display text
    - **Mapping**: 'Laki-laki' → 'Male', 'Perempuan' → 'Female'

Triggers:

- **before_insert_children**
    - **Event:** BEFORE INSERT on the Children table.
    - **Reason:** Automates the generation of a unique child_id in a structured YYYYMMXDD-HHMMSS-XXXX format.
    - **Procedure**: GenerateChildUniqueID, SetGenderText
- **before_update_children**
    - **Event:** BEFORE UPDATE on the Children table.
    - **Reason:** Updates gender_text when gender value changes.
    - **Procedure**: SetGenderText

6. Requisites

To set up and run this project, the following requisites are needed:

- Python 3.9+
- pip (Python package installer)
- MySQL/MariaDB Server (e.g., Railway, SkySQL, or local instance)
- MongoDB Server (local or cloud instance like MongoDB Atlas)
- Virtual environment (recommended)

7. Contributions

This project was a collaborative effort with distinct responsibilities:

1. Omar Keita

- **Project Setup & Organization Structure:** Defined the initial project layout, directory structure, and established foundational development practices.
- **SQL Database Design:** Designed the relational schema for Children, Measurements, and Diagnosis tables, including column definitions, primary keys, and foreign key relationships.
- **Procedures and Triggers:** Designed and implemented the MySQL stored procedures (GenerateChildUniqueID) and triggers

- **Data Loading (SQL):** Implemented the logic for loading the initial dataset from CSV into the MySQL database, handling data transformation and insertion.
- **SQL DB Online Setup:** Configured and managed the MySQL database instance on Railway.

2. Reponse Ashimwe

- **SQL Application Development:** Developed the core application logic interacting with the SQL database.
- **Schema & Models:** Implemented the SQLAlchemy ORM models (models.py) corresponding to the SQL database schema and managed database synchronization
- **API Setup (FastAPI & Uvicorn):** Set up the FastAPI application, defined the main application instance, and configured Uvicorn for serving.
- **SQL CRUD Endpoints:** Implemented the RESTful API endpoints for Create, Read, Update, and Delete operations on children, measurements, and diagnosis records in the MySQL database.
- **Prediction Endpoint:** Developed the API endpoint responsible for retrieving data, performing necessary preprocessing, and calling the loaded Random Forest model to generate malnutrition predictions.
- **Deployment:** Managed the deployment of the FastAPI application on Render.com, including environment configuration and continuous deployment setup.
- **Documentation:** Application's technical documentation.

3. Rene Ntabana

- **MongoDB Setup & Collections:** Configured the MongoDB database instance and designed its collection structure for storing child records.
- **Data Initialization (MongoDB):** Implemented the process for loading initial data from the dataset into the MongoDB database.
- **MongoDB CRUD Applications:** Developed the application logic for Create, Read, Update, and Delete operations on MongoDB documents.
- **API Endpoints (MongoDB):** Created dedicated FastAPI endpoints to expose CRUD functionalities for the MongoDB database.