RealVision Team: Project Demand Analyzer (PDA) Sprint 1 Architecture Report

The Project Demand Analyzer (PDA) is a critical web application designed to eliminate guesswork in initial project planning for Ghanaian real estate firms. Its core purpose is to streamline the client quoting process by providing an **Instant Cost Tally** and forecasting market viability through a **Demand Feasibility Score (DFS)**.

I. Web Application Architecture and Major Pages

The front-end structure is built around a secure dashboard system, leveraging a clean, custom HTML and CSS interface. The current implemented pages reflect our commitment to security, core functionality, and data transparency.

A. Major Pages Implemented and Planned

Page Name	File Reference	Purpose and Functionality	Status (Sprint 1)
Dashboard	`index.html`	The main functional page. Users input material and quantity data for the **Instant Cost Tally** . It also displays the project summary and houses the future **Demand Insights** output (DFS).	**Functional Slice Implemented** **(TBD in video demo)**
Materials DB	`materials.html`	Serves as the administrative reference page. It displays the comprehensive list of raw materials, their **Unit Cost (GHS)**, and `Category` as retrieved directly from the database (our `GHA_PRICES` structure).	**Implemented (Database Display)**

Recent Analyses	`recent.html`	Provides historical context and persistence (F-4 Snapshot). This page will store and display records of past cost estimations and demand scores for review by Senior Management.	**Implemented (Static Mockup)**
Authentication	`login.html`, `signup.html`	Provides platform security and user access control for the application.	**Implemented (Layout only)**
Settings	`settings.html`	Allows users to manage application preferences (e.g., currency, theme) and view the PDA's mission statement.	**Implemented (Layout only)**

II. Functional Description of the PDA: Sprint 1 Focus

The PDA's overall functional architecture is composed of a multi-page **Web Application** (HTML/CSS/JavaScript) and an intelligent **Machine Learning Model** (Python back-end) which provides the predictive factor.

The **Dashboard (`index.html`)** is the functional core of this sprint. Its primary purpose is to enable the **Instant Cost Tally** . Users input quantities for key materials (like Cement and Blocks) via a simple form. The system then calculates the **Total Project Value** by cross-referencing these inputs with current unit prices pulled from the PHP data layer.

The second critical component is the **predictive logic** executed by the **Python-based machine learning algorithm** (as demonstrated in the Google Colab notebook). This intelligence layer, which uses **Linear Regression**, is trained on simulated market data to predict a **Demand Multiplier (DM)**. This DM is key for the final **Demand Feasibility Score (DFS) Generator**, which will be implemented in a subsequent sprint. This structured approach ensures our application is both a functional calculator and a predictive insight tool for AccraBuild Streamline.

III. Major PHP Functions and Backend Logic

The PHP layer is responsible for secure, dynamic interaction with the MySQL database. These functions underpin the retrieval, validation, and verification of project data.

1. **Validation:** Checks all user inputs (e.g., quantity, material selection) to ensure they are complete, correct, and in the right format (e.g., quantity is a positive number). This prevents calculation errors and ensures data integrity.
2. **Verification:** Confirms that the data submitted by the user matches valid records in the database. For F-1, this means confirming that a selected **material name exists** and has a **set `unit_cost`** before proceeding with the calculation.
3. **Display from Database:** Retrieves stored data (like the materials list for the `materials.html` page and the Dashboard dropdown) and formats it for dynamic display on the web pages. This ensures the pricing data shown is always current and centrally managed.

IV. Description of Frontend Choices (Architecture Requirement)

For our project so far, we have intentionally utilized **Vanilla HTML, CSS, and JavaScript**. This decision allows our team to focus intensely on core web technology fundamentals and architectural principles without relying on the abstractions of heavy external frameworks.

- **CSS and Layout:** We chose to build our layout using **custom CSS** (including Flexbox and CSS Grid principles) instead of a pre-built utility or library (like Bootstrap or Tailwind). This decision ensures **maximum performance, lightweight styling**, and provides the team with complete control over the application's unique, professional visual aesthetic and responsiveness across devices.
- **Interactivity (JavaScript):** We rely exclusively on **JavaScript** for form validation, asynchronous data fetching (AJAX/Fetch), and dynamic updates to the Dashboard. This ensures the team gains a deep, maintainable understanding of DOM manipulation and client-side logic, providing a robust foundation should the project later scale to incorporate a framework like React or Vue.

Additional Functionality

For our additional functionality, the Project demand analyzer (P.D.A.) doesn't just calculate cost, it also uses an AI model to predict how much demand a project will have. After the user enters materials and quantities, the system analyzes those inputs using a trained machine learning model. The AI compares the project to similar historical projects and estimates how attractive it will be in the market. This generates a Demand Multiplier that contributes to the final Demand Feasibility Score.

PROJECT DEMAND ANALYZER (PDA) – UPDATED SYSTEM ARCHITECTURE REPORT

1. Frontend Architecture

The PDA frontend is deliberately built with *Vanilla HTML, CSS, and JavaScript* to ensure simplicity, transparency, and full control over the interface. Key principles are:

- Lightweight pages for fast loading
- Clear separation between UI, data display, and backend calls
- No frameworks (e.g., React, Vue) to keep the technology accessible and the learning curve low

Main Pages

Page	File	Purpose	Status
Dashboard	index.html	Main workspace. Calculates project cost, shows summary, and will eventually display the Demand Feasibility Score (DFS).	Functional slice completed
Materials Database	materials.html	Shows all raw materials and their unit costs pulled directly from the central database.	Implemented
Recent Analyses	recent.html	Review area for previously saved estimates and demand scores.	Currently a static mockup
Login / Signup	login.html, signup.html	Controls access to the platform.	UI completed
Settings	settings.html	User preferences (theme, currency), profile info, mission statement.	UI completed

Frontend Responsibilities

- Collecting user inputs (materials, quantities, project attributes)
- Performing basic client-side validation
- Making asynchronous requests to the backend
- Displaying results from database queries and ML predictions

---

2. Backend Architecture (PHP Layer)

The backend uses a small collection of modular PHP scripts that each handle a specific responsibility. This separation keeps the system maintainable and improves debugging clarity.

Core Backend Files

File	Responsibility
db_connect.php	Initializes sessions, connects to MySQL, and provides an authentication check function for protected pages.
auth_handler.php	Handles login and signup form submissions. Validates users, hashes passwords, and manages sessions.
data_processor.php	Manages all dashboard-related calculations: retrieving material prices, calculating project totals, and saving results.
gpe_calculator_processor.php	Acts as the bridge between the dashboard and the external ML service. Prepares input data, sends the request, receives predictions, and stores everything in analysis history.
logout.php	Ends the user session and redirects to login.

Backend Responsibilities

- Validating all incoming data
- Retrieving pricing and materials data from the database

- Performing server-side calculations for accuracy and security
- Communicating with the machine learning service
- Recording completed analyses for reporting and traceability
- Handling all authentication, sessions, and access control

---

3. Machine Learning Service (Python/FastAPI)

The PDA includes a dedicated ML microservice responsible for generating the **Demand Multiplier (DM)** and contributing to the **Demand Feasibility Score (DFS)**.

Role of the ML Service

- Receives project attributes, materials, and quantities
- Runs the predictive model (trained separately in Colab)
- Returns a demand prediction used by the dashboard
- Allows the PHP application to remain simple and focused

Reason for a Separate Service

- Keeps the prediction logic independent
- Allows retraining without touching the core web app
- Keeps PHP clean by avoiding large data science libraries

4. Database Architecture

The database combines the minimal requirements for login with your extended property-material model.

Core Tables

Table	Purpose
Users	Holds user login details and roles. Required for authentication.
Materials	Stores material names, unit costs, units of measure, and categories.
Housing_Property	(From your ERD) Holds property attributes used in more advanced feasibility forecasting.
PropertyMaterial	Connects properties to materials for cost tracking, future expansions, or reporting.
AnalysisHistory	Stores past estimates, predicted demand multipliers, and overall DFS values.

Key Database Functions

- Acts as the single source of truth for pricing
- Stores historical analyses for management review
- Maintains secure and reliable user authentication data
- Supports the predictive model by supplying structured inputs

5. System Workflow

Step-by-Step Flow

1. **User logs in** through login.html (password hashing + session start).
2. **Dashboard loads**, retrieving the materials list for dropdowns.
3. User enters quantities and submits.

4. The backend retrieves unit prices and performs the **Instant Cost Tally**.
5. The backend then sends the calculation inputs to the **ML service**.
6. The ML service responds with a demand estimate.
7. The backend combines both results, saves them into **AnalysisHistory**, and returns them to the dashboard.
8. The front-end displays the cost and feasibility outcome to the user.

This flow ensures clear separation of responsibilities and avoids overloading any single layer.

6. Security Considerations

To ensure the PDA remains a secure workspace:

- All sessions are protected and verified before loading dashboard content
- Passwords are fully hashed before storage
- Inputs are sanitized to prevent injection
- Database access is controlled and standardized
- Login and signup pages do not expose system details