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.