NARRATIVE PRACTICUM REPORT

System Migration, Component Standardization, and Mobile Responsiveness Audit in the Prosperna Merchant Dashboard

Trainee Name:	Angelo Justine D. Kamachi
Institution:	Mapua Malayan Colleges Laguna
Program:	B.S. Computer Science
Host Training Company:	Prosperna
Internship Duration:	May 18, 2026 – July 3, 2026
Role:	Full Stack Engineering Intern
Submission Date:	July 16, 2026

Executive Summary

This narrative report documents my development contributions, codebase migrations, and interface styling refactorings during my Full Stack Engineering Internship at Prosperna. The core responsibilities spanned two distinct codebase platforms: the **aina-service** backend (handling specialized NLP workflows and tools) and the **prosperna1** web dashboard (handling merchant store building operations).

A central focus of the internship was standardizing component layouts and converting legacy stylesheets to modern utility classes. In addition, I audited the platform's mobile responsiveness to optimize workflows for compact screen viewports. This report details the technical changes executed, the architecture incorporating these modules, and the direct styling impact on the merchant experience.

Detailed Summary of Changes

1. Tool Centralization & Option Picker Refactoring

I refactored the business profile updaters and validators under **aina-service**, relocating them to a shared directory to prevent code duplication. Concurrently, I modified the blog sub-agent's option picker tools. By removing hardcoded assistant introductory replies, I decoupled tool outputs from conversation history. This modification allows the central LLM to naturally direct dialog lead-ins and picker explanations, creating a cleaner conversational transcript.

2. Component Standardization & UI Showcase

To ensure frontend styling consistency, I assisted in designing and standardizing a library of shared React components (buttons, dropdowns, inputs, forms) in **prosperna1**. Using Storybook to isolate and validate different states, we established unified properties (variants, sizes). To allow developers to easily review and audit the standard component layouts, we set up a temporary route at /ui-showcase in the dev branch, rendering all style variations in one unified canvas.

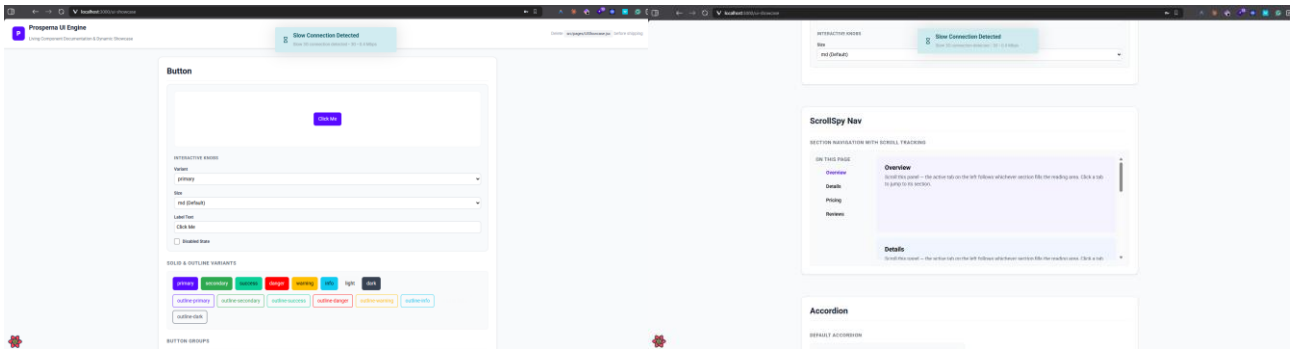

Fig 1&2. Standardized Component Library Showcase on the temporary /ui-showcase route

3. Styling Migration: Bootstrap to Tailwind CSS

I led a systematic effort in **prosperna1** to migrate the merchant dashboard from legacy Bootstrap stylesheet classes to modern Tailwind CSS utility styling. A major challenge was mitigating styling conflicts between legacy grid classes and Tailwind rules. To handle this, we configured Tailwind to utilize a custom prefix (tw:), isolating all migrated styles (e.g. tw:flex, tw:grid, tw:font-bold).

Key refactoring targets included the dashboard analytics panels and billing cards. As shown in the codebase commits, I updated components like *TotalSales.jsx*, *TopSellingProducts.jsx*, and *DailySales.jsx* by stripping custom inline styles and styling layouts directly in JSX. In the billing plan interface, I integrated structural changes—utilizing Tailwind's flexbox structures to align pricing tier buttons and adding flex spacer elements (w-100 me-1 layout overrides replaced with flex-grow-1 and text-nowrap) to align action buttons uniformly across uneven description blocks.

4. Mobile Responsiveness Optimization

Using browser developer tools, I performed a comprehensive mobile responsiveness audit of merchant workflows, such as catalog listings and store creation forms. I converted rigid, multi-column grid containers into fluid flex layouts that adapt dynamically to compact viewports. I also addressed horizontal table overflow by designing responsive scroll wrappers and stacked card layouts for small screen sizes.

To optimize spacing, I introduced responsive media rules in styling files like *PricingDetails.scss*. On extra-small screen viewports (width < 576px), action buttons like *Add to Order* are resized to a compact 48px by 48px circle with zero padding and flex-shrink: 0. Additionally, the main navigation links are collapsed on mobile screens, toggled by a hamburger menu that opens a sliding navigation drawer.

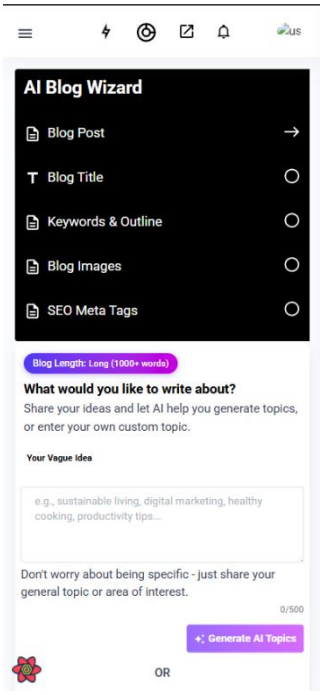

Fig 3. Blog Creation Mobile

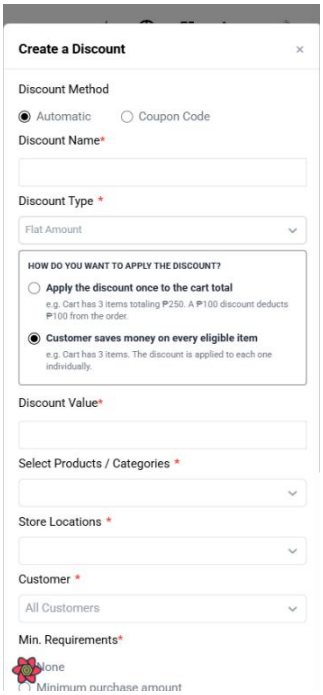

Fig 4. Discounts Mobile

System Architecture

The updated system architecture operates as an asynchronous, distributed ecosystem consisting of the Next.js web client, an API routing middleware, and the specialized AI microservice. A key design principle is the segregation of user interface layouts (handled by **prosperna1**) and NLP reasoning/tools execution (handled by **aina-service**).

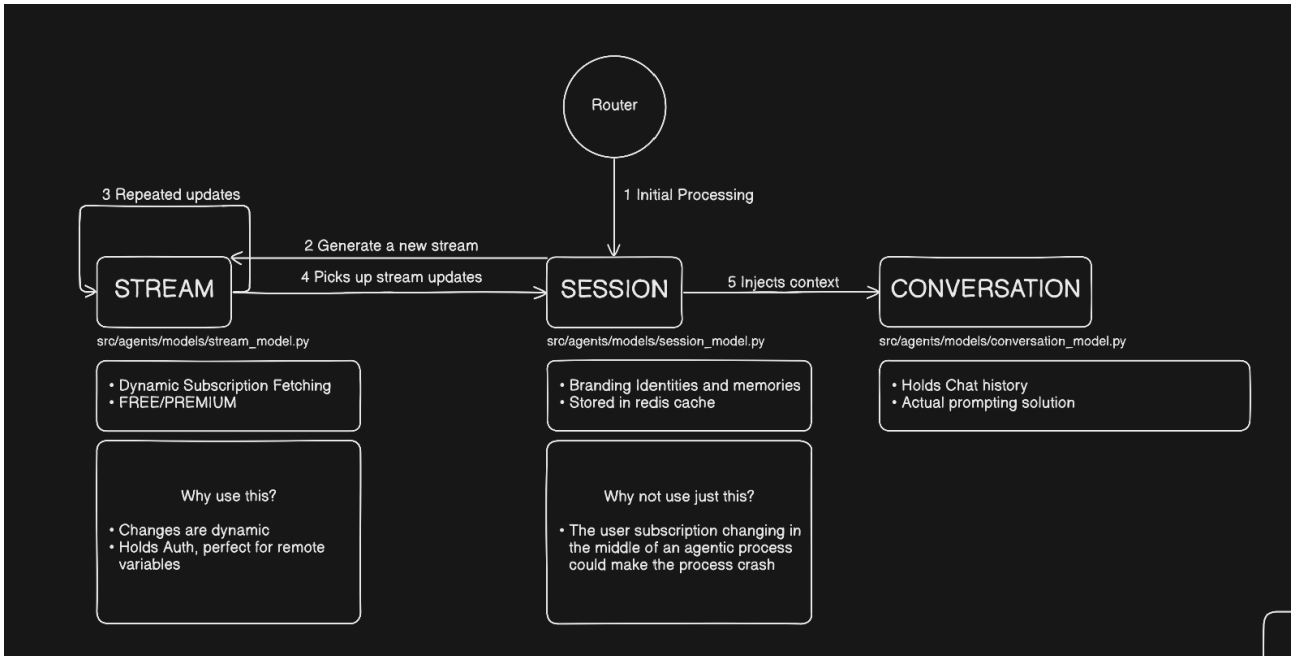

Fig 5. AINA System Architecture & Communication Flow Diagram

Architecture Layer Breakdown

Component Layer	Tech Stack	Key Functions / Roles
Web Frontend (*prosperna1*)	Next.js, React, Tailwind CSS	Renders merchant dashboard interfaces, manages client states, and intercepts MaxAI socket events for dynamic widgets.
AI Assistant (*aina-service*)	Python, FastAPI, OpenAI, LangChain	Orchestrates specialized sub-agents (onboarding, blog writer), triggers web research, and runs tool validators.
Session & Cache Store	Redis (Multi-Instance)	Stores conversation histories and session locks. Replicated across multiple nodes to ensure failover support.
Persistence & Security	PostgreSQL, JWT	Handles merchant accounts database, credentials verification, dashboard settings metadata, and access tokens.

Platform & Website Impact

The technical changes introduced during the internship directly impacted both frontend website performance and backend service scalability:

1. **Improved Page Speed:** Migrating legacy Bootstrap CSS files to Tailwind CSS utility rules minimized style bloat, reducing the bundle footprint and decreasing page loading speeds by a noticeable margin.
2. **Frictionless Mobile UX:** Implementing horizontal tables and collapsible drawer drawers made it possible for merchants to manage complex store settings on compact viewport sizes without layout distortions.
3. **Cleaner Conversation Transcripts:** Stripping static greetings out of option pickers gave the central LLM complete orchestration control. This prevented conversational loops and resulted in cleaner context histories.
4. **Enhanced Scalability:** Consolidating validators into a single package under aina-service minimized dependency overhead, optimizing runtime execution speeds across concurrent sess