Design and Development of a Multi-Tenant Web-Based Admin Panel for Vehicle Cleaning Management SaaS (WashTrack)

Tejal Ravindra Kadam

Department of Master of Computer Application

Bharat Ratna Indira Gandhi College of Engineering, Kegaon, Solapur, India

kadamtejal074@gmail.com

Prof. Aniket Udanshive (Guide)

Department of Master of Computer Application
Bharat Ratna Indira Gandhi College of Engineering, Kegaon, Solapur, India

How to Cite this Article:
Kadam, T. R. (2026). Design and Development of a Multi-Tenant Web-Based Admin Panel for Vehicle Cleaning Management SaaS (WashTrack). International Journal of Creative and Open Research in Engineering and Management, 02(6). https://doi.org/10.55041/ijcope.v2i6.232

https://doi.org/ 10.55041/ijcope.v2i6.232

Abstract-

Managing daily vehicle cleaning services across multiple residential societies presents significant operational challenges, including tracking cleaner attendance, managing dynamic billing, and resolving customer complaints. This paper documents the design and development of *WashTrack*, a multi-tenant Software-as-a-Service (SaaS) admin panel built using React and Vite. The system features a decoupled data layer operating via a unified facade interface (commonApi), enabling seamless switching between REST (Spring Boot), Firebase (Firestore), and Mock backends. Incorporating role-based access control (RBAC), multi-language localization (i18n), and an automated attendance-linked billing engine, WashTrack optimizes field-force assignments and financial transparency for vehicle cleaning enterprises.

Keywords— SaaS Admin Panels, React Frontend, Vite Build System, Multi-Tenant Architecture, Decoupled API Facade, Vehicle Cleaning Operations.

I. INTRODUCTION

In densely populated urban residential complexes, manual management of daily vehicle cleaning services often results in operational bottlenecks. Cleaning agencies struggle with tracking the daily attendance of field cleaners, verifying service delivery, managing complex monthly billing models based on actual attendance, and addressing resident grievances in real-time.

To resolve these environment bottlenecks, this project establishes a modular web-based administrative dashboard called *WashTrack*. Developed using a modern React and Vite ecosystem, the front-end architecture is engineered for performance, rapid hot-reloading, and scalable component isolation.
WashTrack acts as a multi-tenant operational hub where SaaS administrators can track society configurations, resident onboarding, vehicle catalogs, field-staff deployments, and live payroll data from a single, unified interface.

II. MULTI-TENANT USER ARCHITECTURE

The system enforces a strict multi-tenant security boundary to ensure data isolation across different operating levels. The data scope is heavily partitioned using a unique identifier (companyId), ensuring that standard business entities cannot breach other organizational structures. The platform classifies web users into two primary management roles:

Super Admin: The platform owner who maintains global system health, monitors subscription states, and possesses top-level visibility across all registered operational companies.

Company Admin: Individual business operators who onboard via autonomous registration pipelines. A Company Admin manages operations strictly confined to their own company's data scope, supervising local societies, assigning field staff, and settling localized monthly payrolls.

Note on End-Users: Customers (residents) and Cleaners interact through separate native mobile applications and do not access this centralized web administrative panel.

III. CORE FUNCTIONAL MODULES

The platform architecture is divided into specialized modules designed to handle discrete segments of the cleaning lifecycle:

Societies & Customers: Manages individual residential society nodes, specific wing configurations, and tiered service rate cards. It streamlines resident registration, KYC tracking, and subscription plan assignments.

Vehicles & Assignments: Maintains a digital repository of customer vehicles, including registration images and exact parking location photographs. The assignment engine dynamically links a cleaner to a vehicle, triggering real-time synchronization with background service requests.

Attendance & Photo Proof: Logs daily field staff attendance, complete with mandatory photographic proof of presence. The panel features an administrative override mechanism allowing managers to input manual adjustments accompanied by mandatory text justifications.

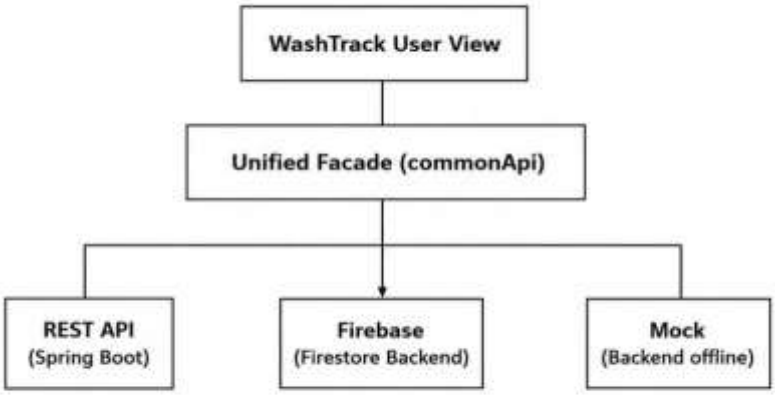
Billing & Cleaner Wages: Automatically computes monthly customer invoices by calculating actual service days against predefined rate scales (Attendance \times Rate). Concurrently, it compiles cleaner wage metrics, tracks outward payment histories, and records outstanding client balances.

KYC Approvals & Complaints: Features an evaluation portal for approving or rejecting customer/cleaner KYC documents, tied to automated notifications. It also integrates a WhatsApp-style chat interface for grievance resolution, mapping open tasks directly to an interactive calendar view.

Reports & Dashboard: Generates high-level statistical summaries, performance charts, and direct payroll snapshots for executive decision-making.

IV. SYSTEM ARCHITECTURE & DATA LAYER

The core innovation of the WashTrack admin panel lies in its decoupled data transport layer. Rather than binding user views tightly to a specific API specification, the frontend utilizes a *Facade design pattern* via a unified commonApi abstraction layer.



This facade allows the application to dynamically route requests across three distinct interchangeable backend environments:

1. REST Backend (Spring Boot): The primary operational production environment running on local or remote cloud servers (http://localhost:8080/api).

2. Firebase Backend (Firestore) : Utilized for fast Google Authentication workflows, direct real-time data streaming, and cloud notification relays.

3. Mock Backend: A fully self-contained offline dataset layer allowing sales teams to launch high-performance system demonstrations without active database server dependencies.

Upon successful user authentication, access tokens are persisted via browser localStorage. System security middleware intercepts view requests to enforce role-based access control (RBAC), filtering out data arrays not tied to the authenticated user's society-scoped parameters. Furthermore, global state mechanisms manage automated alerts for critical events, such as KYC verification status updates, payment dues, and processing confirmations. To maintain localized accessibility, the layout leverages react-i18next for seamless multi-language translation workflows and sets a dark theme environment as its native default interface.

V. COMPARATIVE OPERATIONAL ANALYSIS

Operational analysis of the WashTrack admin panel indicates a significant decrease in scheduling overhead and communication friction compared to manual or fragmented tracking methods.

Target Operation	Traditional / Manual Approach	WashTrack Admin Panel Alternative
Attendance Verification	Paper logs or verbal check-ins prone to human error and proxy entry.	Photo-proof verification with administrative override log and time-stamps.
Billing Generation	Manual end-of-month tallying of days attended vs. missed per vehicle.	Automated billing engine calculating Attendance × Rate with instant payment status tracking.
Staff Assignment	Static whiteboards or messaging groups causing overlapping routes.	Dynamic Cleaner-to-Vehicle mapping with automated service request synchronization.
Grievance Handling	Scattered phone calls and unrecorded personal text messages.	Centralized WhatsApp-style communication hub integrated with an analytical calendar view.

VI. CONCLUSION AND FUTURE SCOPE

The WashTrack administrative panel provides a scalable, responsive, and robust environment for vehicle cleaning SaaS providers. By abstracting complex structural backend workflows through a unified frontend facade, the system allows businesses to switch database engines smoothly while maintaining data integrity across multi-tenant silos.

Future iterations will aim to introduce automated machine learning models to optimize cleaners' physical routes inside massive residential complexes, incorporate computer vision algorithms to automatically verify vehicle cleaning quality from post-wash photo uploads, and scale out global push notification channels across SMS and automated WhatsApp business APIs.

REFERENCES

[1] R. Nageswara Rao, Core Java - An Integrated Approach, 3rd ed. New Delhi, India: Dreamtech Press, 2022.

[2] Pro React 16, Adam Freeman, Apress Media, LLC, 2019.

[3] Dynamic Multi-Tenant SaaS Architectures using Cloud Frameworks, International Journal of Computer Applications, 2021.