CASE STUDY

DronaHQ

From tech debt to tech edge

How Purplle saved $200K+ and rebuilt 600+ internal panels with just two engineers

purplle.com

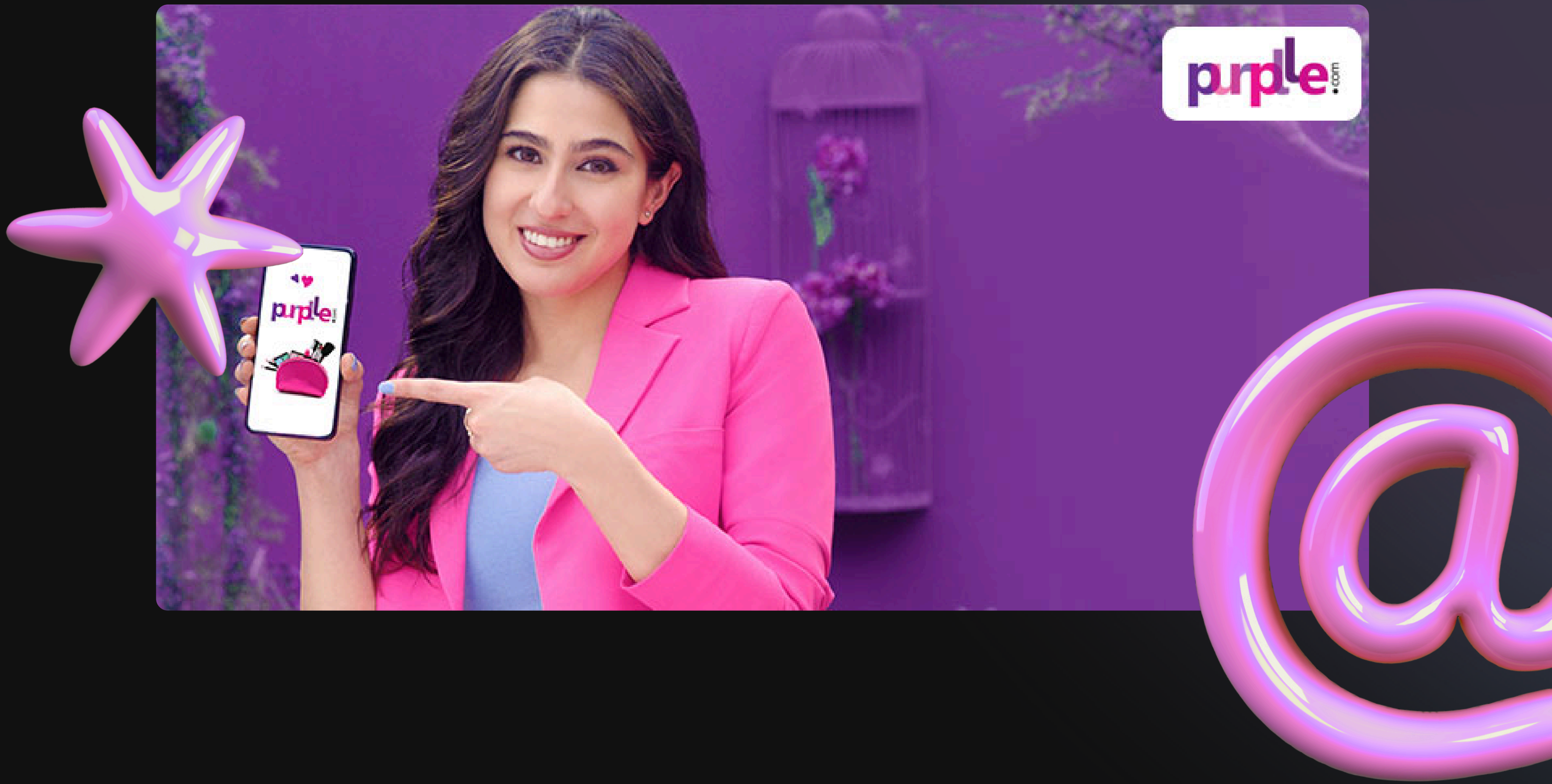
COMPANY SNAPSHOT

Company:	**Purplle.com**
Sector:	**Beauty & Personal Care**
Daily Orders:	**50,000–100,000 (2x during sales)**
SKU Count:	**60,000+**
Ops Panels Pre-Migration:	**600+**
Tech Stack (Pre):	**PHP, Angular, React, in-house scripts**
Platform Chosen:	**DronaHQ (self-hosted)**
Migration Timeline:	**Q4 2023 – Q3 2024 (phased)**

Purplle is one of India's **fastest-growing beauty marketplaces**, serving millions of users across tier 2 and tier 3 cities.

As **demand scaled** and digital touchpoints expanded, the team found itself **entangled in a web of fragmented internal tools**, most of which had been **created reactively** over the years to solve specific operational needs.

These **legacy panels** had grown from a few scripts into **hundreds of disconnected tools** stitched together by devs across different time periods and technologies.



1. The hidden cost of internal tools

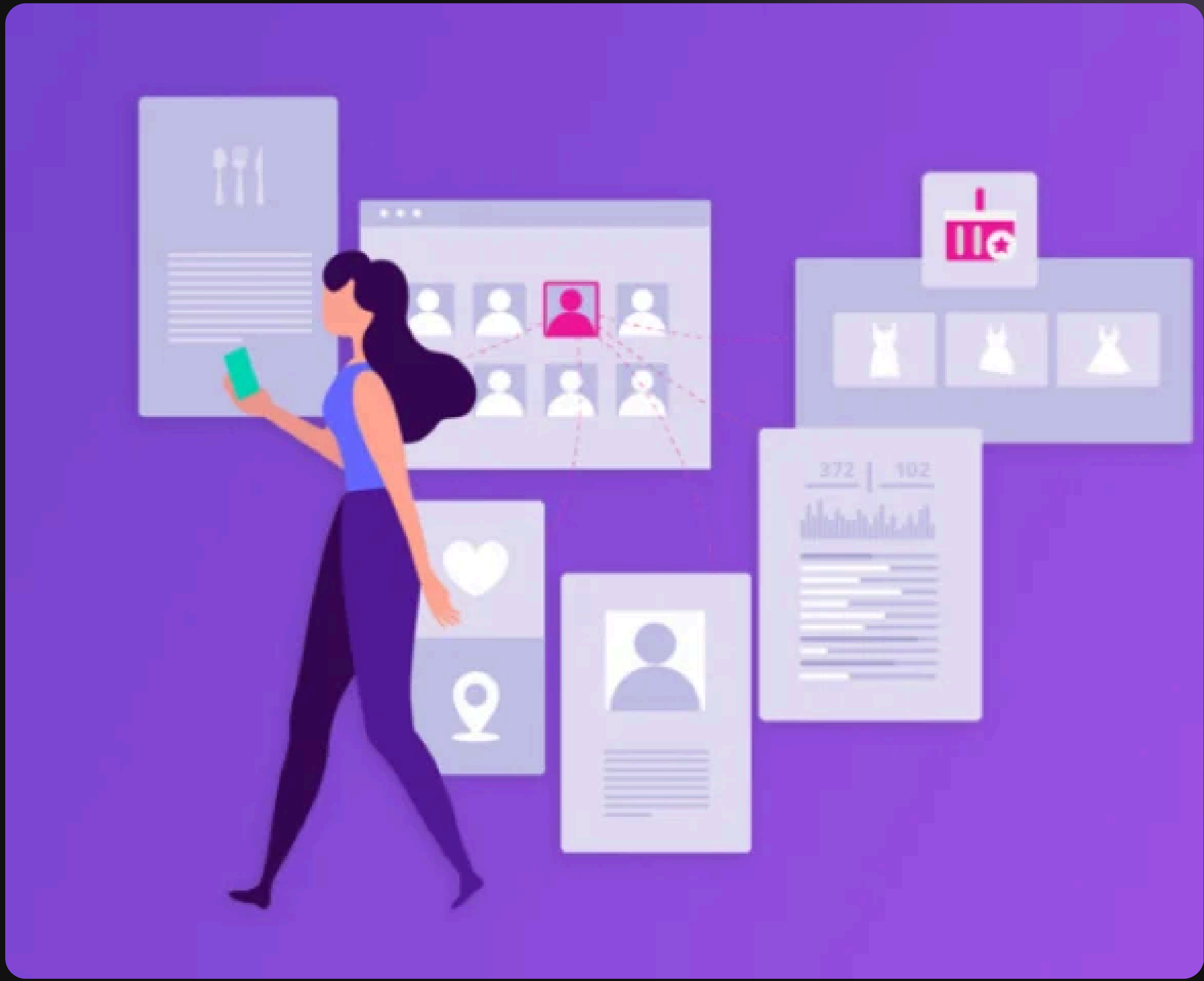
"Everyone talks about scale. No one talks about 600 panels quietly running in the background—until they break."

— Vivek Parihar, **VP Engineering**

Full conversation →

The **reality of scale** isn't just about user growth or GMV acceleration—it's about what happens behind the scenes when **core operations are run by systems no one wants to touch**.

Purplle had accumulated over 600 internal admin panels. Some were 8+ years old. **Some were still in use. Others were forgotten**, duplicated, or obsolete but continued to live in the system—until a crisis exposed them.



- Panels built in PHP, React, Angular, and ad hoc frameworks
- No shared codebase, no source-of-truth documentation
- Business teams relied on devs for every tweak
- Minor updates (e.g., adding a dropdown) could take 3–5 days
- Tech leadership struggled to even quantify what existed
- Dev hours were wasted maintaining systems with no future

This chaos created not just inefficiencies but real risk. **Upgrade a PHP version?** Break half a panel. **Train a new engineer?** Weeks of shadowing and still uncertainty. It became clear that to scale confidently, **the internal tooling strategy had to change.**

2. The low-code pivot

"We wanted to stop spending engineering hours on tasks like changing a textbox color. Low-code gave us a way to move fast without compromising on structure."

— Chintan Surelia, **Engineering Manager**

Full conversation →

The team needed a way to tame complexity without reinventing everything. The answer wasn’t to rebuild 600 tools.
It was to rethink the structure.

Why Low-Code?

- Free up backend engineers from UI maintenance
- Establish consistent design and access patterns
- Reduce turnaround from weeks to hours
- Enable rapid iteration for ops and business teams
- Consolidate scattered workflows onto a common platform

3. Core principles

Headless APIs handle logic and business rules

Admin panels become lightweight shells

Version control, RBAC, CI/CD enforced

Low-code is the default for all future internal tools

4. Migration blueprint

Phase	Description	Duration
0 – Foundation	REST/GraphQL gateway + middleware setup	3–4 months
1 – Audit	Identified 600+ panels, prioritized top 100	1 month
2 – Pilot	Formed 2-person pod; built first 60 panels	2 months
3 – Scale	Migrated 100+ panels with tight loops	4–5 months
4 – Ongoing	Default all new tools to low-code	Continuous

"Even a pin code toggle had its own panel. Migration needed a map."

During the audit, the team classified panels into must-have, good-to-have, and deprecated categories.

Some were rebuilt from scratch, others reimagined. Overlapping logic was abstracted into shared APIs. DronaHQ became the new surface layer.

The **two-member pod** worked closely with **backend owners** and business teams to streamline rollouts.

Because changes were no longer bottlenecked by frontend bandwidth, operations teams began requesting improvements more frequently.

5. Measurable impact

Manpower Efficiency

Metric	Pre-migration	Post-migration
Active builders	10–15 ad hoc devs (avg 12 FTE)	2 dedicated builders
Annual Effort	~50 man-months maintaining & enhancing 600 panels (≈ 10% annually)	24 man-months total (maintenance + new)
Man-months saved	26	

If a fully-loaded senior engineer costs $8,000/month:

Annual $ saved = 26 × $8,000 = $208,000

5. Measurable impact

Task	Pre-migration	Post-migration
Simple Panel	3–4 days	15–30 mins
Complex Panel	2–3 weeks	2–3 days
Rollbacks	Manual, risky	1-click

5. Measurable impact

- One central queue for all tooling requests
- Reusable connectors and components
- Shared UI patterns for better usability
- Audit trails, access controls, and environments in place
- Business teams could now think in terms of features, not blockers

“Standardization helped us prevent drift. Previously, we had Angular, React, and even old PHP scripts. A small change used to take days, sometimes a week. Now, a two-person team can manage requests that used to require 10 developers.”

— Vivek Parihar, **VP Engineering**

Full conversation →

6. What made it work

- **API-First**: All panels consumed clean REST or GraphQL endpoints
- **Middleware Design**: Panels were decoupled from backend constraints
- **Guardrails Built-In**: Developers worked within safe, repeatable systems
- **Central Ownership**: A focused pod meant faster delivery and fewer errors
- **Composable Components**: Reduced time spent on boilerplate UI and plumbing

These decisions enabled Purplle to move from reactive maintenance to proactive innovation.

What this means for engineering leaders

- Internal tools are business-critical. Treat them like products.
- Tech debt isn't just about code. It's about structure.
- Centralizing with low-code doesn't reduce flexibility—it amplifies it.
- You don't need 20 devs to manage 600 panels. You need the right foundation

"It's not just about saving time. It's about unlocking bandwidth for what really moves the business."

"Every team had a different way of doing things. We had to centralize development, add structure, and remove decision fatigue. With DronaHQ, we could focus on the logic and not worry about reinventing components or fixing broken UIs."

— Chintan Surelia, **Engineering Manager**

Full conversation →

Want to see what DronaHQ can do for your team?

Let’s talk about migrating your internal tools.

Schedule a walkthrough