PROJECT SEMESTER REPORT

On

Cashify Web and Console Features

Submitted by

Vartika Gautam
Roll No : 102103397

Under the Guidance of

Dr. Saif Nalband
Assistant Professor, CSED, TIET
(Faculty Mentor)

Mr. Hemchand Yadav
Engineering Manager
(Industrial Mentor)

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Submitted to the
Computer Science & Engineering Department
Thapar Institute of Engineering & Technology, Patiala

In Partial Fulfilment of the Requirements for the Degree of
Bachelor of Engineering in Computer Engineering
at
Thapar Institute of Engineering & Technology, Patiala June 2025

Cashify Web and Console Features

by Vartika Gautam

Place of work: Cashify

Submitted to the Computer Science & Engineering Department, Thapar Institute of Engineering & Technology

June 2025

In Partial Fulfilment of the Requirements for the Degree of Bachelor of Engineering in Computer Engineering

Abstract

During my internship at Cashify, I worked on multiple projects involving the development and optimization of front-end workflows and backend-integrated dashboards.

My primary focus areas included the implementation of the Incentive Redemption Flow, where I contributed to building a secure KYC process, integrated ERP workflows, and developed dynamic dashboards for finance validation and redemption tracking. I also worked on the Tier-Pincode Override Panel, which introduced a configurable pricing override mechanism based on region, source, and product line to replace rigid, hardcoded tier logic.

Additionally, I contributed to the Centralized UI Testbed—a platform to validate reusable components and ensure consistency across pods. Throughout the internship, I strengthened my technical proficiency in React.js, TypeScript, React Native, and REST API integration while collaborating with cross-functional teams to deliver scalable, business-aligned solutions.

This report highlights the work done across different modules and their impact on improving operational efficiency, pricing accuracy, and UI standardization.

Author : Vartika Gautam

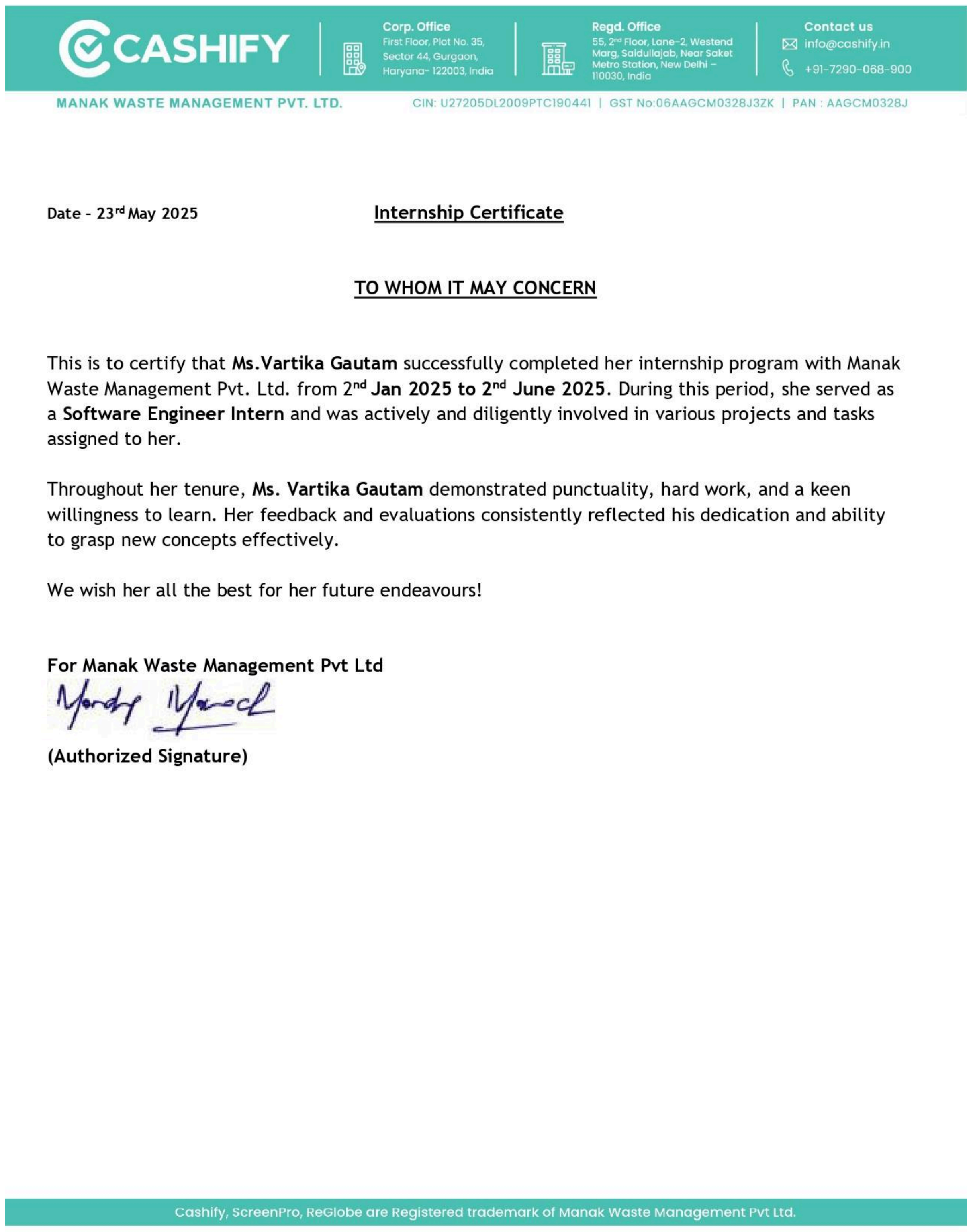
Saif Nalband

Certified by
Dr. Saif Nalband
Associate Professor, CSED,
TIET (Faculty mentor)

Hemchand

Mr. Hemchand Yadav
Engineering Manager
(Industrial Mentor)

CERTIFICATE (PROJECT SEMESTER TRAINING) FROM THE COMPANY OR THE ORGANISATION



CASHIFY

Corp. Office
First Floor, Plot No. 35,
Sector 44, Gurgaon,
Haryana- 122003, India

Regd. Office
55, 2nd Floor, Lane-2, Westend
Marg, Saidullajab, Near Saket
Metro Station, New Delhi –
110030, India

Contact us
info@cashify.in
+91-7290-068-900

MANAK WASTE MANAGEMENT PVT. LTD. CIN: U27205DL2009PTC190441 | GST No:06AAGCM0328J3ZK | PAN : AAGCM0328J

Date – 23rd May 2025 **Internship Certificate**

TO WHOM IT MAY CONCERN

This is to certify that **Ms.Vartika Gautam** successfully completed her internship program with Manak Waste Management Pvt. Ltd. from 2nd **Jan 2025 to** 2nd **June 2025**. During this period, she served as a **Software Engineer Intern** and was actively and diligently involved in various projects and tasks assigned to her.

Throughout her tenure, **Ms. Vartika Gautam** demonstrated punctuality, hard work, and a keen willingness to learn. Her feedback and evaluations consistently reflected his dedication and ability to grasp new concepts effectively.

We wish her all the best for her future endeavours!

For Manak Waste Management Pvt Ltd

[illegible]

(Authorized Signature)

Cashify, ScreenPro, ReGlobe are Registered trademark of Manak Waste Management Pvt Ltd.

Table of Contents

Table of Figures

1. Company Profile

CASHIFY

Figure 1: Company Logo

1.1 Historical Background

Cashify was founded in 2013 by Mandeep Manocha, Nakul Kumar, and Amit Sethi with the vision of creating a structured and trustworthy platform for the resale and recycling of used electronics, especially smartphones. At the time, India's second-hand mobile market was largely unorganized, lacking transparency, fair pricing, and standardized processes. Cashify emerged to bridge this gap by offering a convenient, secure, and technology-driven solution for users to sell, repair, recycle, or upgrade their devices.

Initially launched under the name ReGlobe, the company rebranded as Cashify to better reflect its core value proposition: instantly providing cash in exchange for old gadgets. The platform leveraged AI-based pricing algorithms and doorstep pickup services, which revolutionized how people disposed of their electronics.

Over the years, Cashify expanded its services beyond smartphones to include laptops, tablets, smartwatches, and even refurbished phone sales. It also ventured into B2B partnerships with major e-commerce and OEM players like Apple, Amazon, Samsung, Xiaomi, Oppo, and OnePlus to support device exchange and trade-in programs.

Cashify has raised multiple rounds of funding from investors like Blume Ventures, Bessemer Venture Partners, and Olympus Capital and has grown into a household name in India’s recommerce space. Today, it operates both online and through a network of offline retail stores, aiming to build a circular economy by promoting sustainable electronics usage and minimizing e-waste.

1.2 Services Offered

Cashify is a leading recommerce platform that offers a wide range of services focused on the buyback, repair, resale, and recycling of electronic devices. With its tech-driven ecosystem, Cashify caters to both individual consumers and business partners through the following key services:

1. Sell Old Devices

Cashify's core service allows users to sell their old smartphones, laptops, tablets, smartwatches, and more in just a few simple steps. The platform offers:

- Instant price quotes using an AI-powered pricing algorithm
- Free doorstep pickup
- Instant payment via UPI, bank transfer, or cash

2. Buy Refurbished Phones

Cashify sells certified refurbished smartphones at affordable prices. These devices go through a thorough quality check and come with:

- Up to 6-month warranty
- 7-day replacement policy
- Multiple payment options, including EMI

3. Repair Services

Cashify provides mobile repair services at the customer’s doorstep or at authorized service centers. Key highlights include:

- Screen and battery replacement
- Water damage and charging port repairs
- Use of genuine spare parts
- Service warranty post-repair

4. Exchange Offers

Cashify partners with top brands and e-commerce platforms to power exchange programs, allowing users to trade in their old devices for discounts on new ones. Brands include:

- Apple, Samsung, Xiaomi, OnePlus, Vivo, Oppo, and Amazon

5. Mobile Accessories

Cashify sells smartphone accessories such as chargers, cables, and screen guards, all quality-tested and compatible with major brands.

6. Recycle E-Waste

To support environmental sustainability, Cashify enables users to recycle electronic waste responsibly, ensuring safe disposal through government-authorized recycling units.

7. Enterprise Solutions

Cashify provides B2B services for corporates, OEMs, and e-commerce players, including

- Bulk device procurement or liquidation
- Buyback and exchange APIs
- KYC and ERP-integrated incentive flows
- Dashboard and reporting tools for internal use

8. Retail Stores

Cashify has also established a growing network of offline retail stores across India where customers can:

- Sell or buy devices in person
- Get their phones repaired
- Purchase refurbished devices and accessories

1.3 Financial Performance

Cashify has demonstrated strong financial growth, with revenue crossing ₹500 crore in FY 2022-23, driven by its core services like device buyback, refurbished sales, and repair services. The refurbished smartphone segment has been a major contributor due to increasing demand for affordable, warranty-backed devices.

Though operating on thin margins, the company is improving profitability through in-house refurbishment, retail expansion, and value-added services. With over $90 million in funding from investors like Bessemer Venture Partners and Blume Ventures, Cashify was valued at around $250 million during its 2022 Series E round.

Its growing offline presence (170+ stores) and strong B2B partnerships with brands like Apple, Amazon, and Xiaomi are key revenue drivers. As demand for sustainable tech grows, Cashify is well-positioned for continued financial success.

1.4 Leadership

Cashify was co-founded in 2013 by Mandeep Manocha (CEO), Nakul Kumar (COO), and Amit Sethi (CTO). Under their leadership, the company has evolved from a startup addressing the unorganized second-hand electronics market to a leading player in India's recommerce ecosystem.

- Mandeep Manocha, the CEO, brings a strong vision for building a circular economy and leads overall business strategy, fundraising, and partnerships.

- Nakul Kumar, as COO, has played a key role in scaling operations, expanding offline retail, and ensuring service consistency across verticals.

- Amit Sethi, the CTO, drives technology innovation, overseeing platform development, backend infrastructure, and data-driven automation.

1.5 Customer Base

Cashify serves a diverse and rapidly growing customer base across India. Its primary customers include

- Individual Consumers: People looking to sell, buy, repair, or recycle their smartphones and other electronic devices. The platform is especially popular among price-conscious buyers seeking affordable, warranty-backed refurbished phones.

- B2B Clients: Cashify partners with major OEMs, e-commerce platforms, and retailers like Apple, Amazon, Xiaomi, Samsung, and OnePlus to power trade-in programs, exchange offers, and device procurement.

- Corporate Clients: Businesses that rely on Cashify for bulk device sales, secure data disposal, and incentive program management through ERP-integrated solutions.

- Retail Footfall Customers: With over 170+ offline stores, Cashify also attracts walk-in customers in tier 1, 2, and 3 cities, strengthening its physical presence and brand trust.

1.6 My Team and Work

During my internship at Cashify, I was part of the Central Pod, a cross-functional unit that collaborates with multiple product managers across various verticals of the organization. This unique setup allowed me to contribute to diverse projects aligned with different business functions, enhancing my understanding of end-to-end product development.

While the broader team managed interactions with multiple departments, my primary role was that of a frontend developer. I worked extensively on building and refining user interfaces, ensuring seamless user experiences across internal dashboards and customer-facing modules. My work spanned multiple projects, allowing me to engage closely with both technical and product teams and contribute to the rapid development and iteration of Cashify's platform.

2. Introduction

During my internship at the company, in addition to completing mandatory training on topics such as full-stack development, personal account management, and general company operations, I worked on numerous projects and gained knowledge of new full-stack industry-standard technologies through various knowledge transfer sessions and online Udemy courses.

The projects used numerous cutting-edge technologies, like React.js, Next.js, TypeScript, React Native, and Redux.

These are the tasks that I worked on during this internship.

2.1 Incentive Program Phase 2 - ERP Integrated Redemption Flow

An Enterprise Resource Planning (ERP) system is a centralized platform that integrates and manages core business processes such as finance, procurement, human resources, supply chain, and vendor management. The ERP system ensures that every approved user is uniquely identified, and their redemption transactions are logged with traceability, thereby reducing the risk of fraud and maintaining transparent financial operations [1].

In the context of Cashify's Incentive Program Phase 2, the ERP system plays a critical role in ensuring compliance by registering verified users as vendors once their KYC is approved. This integration enables accurate tracking, streamlined incentive payouts, and financial audit readiness.

Key Features

1. KYC Verification Flow: Users upload Aadhaar, PAN, and bank details, validated via third-party APIs.

2. ERP Vendor Creation: KYC-approved users are automatically registered as vendors in the ERP system.

3. Wallet Activation: Incentive wallets are created only after successful KYC, enabling point accumulation.

4. Redemption with TDS Compliance: Users can redeem points with visibility into TDS deductions and payout timelines.

5. Finance Dashboards: Admin panels for KYC approvals, redemption tracking, and financial reports.

6. CSV Support: The finance team can manage redemptions using CSV upload/download for UTR updates.

7. Email Notifications: Users are informed of approval/rejection with re-upload options.

8. Audit Logging: All actions are logged for compliance and audit purposes.

9. User-Friendly Interface: Smooth Switch App experience for KYC and redemption tracking.

Benefits

1. Ensures tax and ERP compliance for incentive payouts.

2. Provides a secure and streamlined KYC and redemption process.

3. Enhances transparency for both users and the finance team.

4. Reduces manual effort through automation and dashboard tools.

5. Improves user trust and engagement via timely communication and clear tracking.

The main objective of the *Incentive Program Phase 2* at Cashify is to establish a secure, compliant, and end-to-end automated system for managing user incentive redemptions. This includes integrating KYC verification, ERP-based vendor creation, and wallet activation to ensure only verified users can redeem their accumulated points. The system is designed to meet taxation and financial compliance requirements while providing real-time visibility, reducing manual effort, and enhancing transparency for both users and the finance team through dedicated dashboards and reports.

2.2 Centralized UI Testbed

At Cashify, diverse product pods develop and maintain different features of the platform. This results in inconsistencies in the user interface due to the absence of a centralized system to view, validate, and align UI components. During my internship, I contributed to building a centralized UI testbed—a modular platform to standardize, preview, and isolate component behavior across pods, ensuring visual consistency, efficient testing, and design compliance throughout the system.

Key Features

1. Standardized UI Library Showcase: Buttons, inputs, typography, radio buttons, and chips in multiple states (hover, active, disabled).

2. Theme Support: Toggle between light and dark modes to test responsiveness and contrast compliance.

3. Live Style Variable Testing: Edit `--primary-color`, `--font-size-base`, etc., and see real-time impact.

4. Pod-Specific Modules: Add pod-specific flows (e.g., price cards, dropdowns) while aligning with the global design system.

5. Versioned Component Library: Maintain changelogs and usage logs, and attach Figma design links.

6. Responsive Testing Toggle: Simulate desktop, tablet, and mobile breakpoints for layout integrity checks.

Benefits

1. Ensures consistent design across all product teams

2. Reduces rework and visual bugs

3. Speeds up QA validation and design sign-off

4. Encourages reusable and modular component development

5. Enhances user experience with uniformity

We followed a modular design approach using React.js and Tailwind CSS, integrating a real-time sandbox environment where pods could test component variations in isolation. The system also supported design documentation linking and responsive testing with theme previews.

2.3 TIER Override Panel

Decoupling pricing logic from geography or tier hardcoding was central to the Tier-Pincode Override Panel. Traditionally, pricing systems relied on static tier assignments based on geography—for example, a pincode like `121002` would always be tied to `Tier 1`, regardless of business-specific scenarios. This approach was rigid. requiring backend code changes or database updates for any exceptions, making the system hard to scale and maintain. To address this, we introduced a dynamic override layer that allowed mapping a specific combination of source, pincode, and product line to an alternate tier—without altering the original tier logic. This mapping could be managed externally via CSV uploads, empowering business teams to configure changes without developer intervention. During pricing computation, the system first checks for a matching override entry and applies it if found; otherwise, it falls back to the default configuration. This not only removed the dependency on hardcoded geographic tiers but also made the pricing system more flexible, context-aware, and business-friendly. [2]

Key Features

1. **Dynamic Tier Mapping**
   Enables mapping of existing pricing tiers to specific combinations of source, pincode, and product line, allowing granular control over regional and partner-specific pricing.

2. **Override Precedence Logic**
   Implements a priority-based resolution where the system first checks for override entries before applying default tier pricing, ensuring accurate and context-aware pricing.

3. **CSV Upload Support**
   Allows business users to add or remove mappings in bulk using CSV files. This

eliminates the need for manual database updates or code-level changes.

4. **Filter & Search Functionality**
   Provides filters for tier, source, and product line to help users quickly find and manage existing mappings.

5. **Sample & Template Download**
   Offers downloadable sample CSV templates to guide users on the correct data format, reducing upload errors and increasing usability.

6. **Configurable Without Code Deployment**
   All mappings are handled at the configuration level, so no engineering deployment is required for changes—enabling faster turnaround for pricing updates.

7. **UI Designed with Zeplin Integration**
   User-friendly interface built based on detailed Zeplin designs, ensuring consistency with the product's design language.

8. **Validation Mechanism**
   Ensures data integrity with checks for duplicates, invalid tier entries, or malformed pincode formats before accepting the uploaded CSV.

9. **Backward Compatibility**
   The system ensures existing tier-based pricing continues to work seamlessly for entries without overrides, maintaining stability across all flows.

Benefits

1. Enhanced Pricing Flexibility: Allows customized pricing per source, region, and product line without modifying core logic.

2. Business Autonomy: Empowers non-tech teams to manage pricing overrides through CSV uploads, reducing developer dependency.

3. Faster Execution: Enables quick rollout of pricing changes without code deployments or release cycles.

4. Improved Accuracy: Ensures the correct tier is applied contextually, reducing pricing mismatches and enhancing customer experience.

5. Scalable Solution: Designed to handle a growing number of override cases without affecting existing logic or performance.

6. Risk Mitigation: Keeps default pricing logic intact, applying overrides only when configured—ensuring backward compatibility.

3. Background

The internship at Cashify provided an opportunity to learn and adapt to the tech stack actively used within the company, including both open-source frameworks and internally developed tools that were specific to the organization’s ecosystem.

Some of these technologies had limited external documentation and community support, which required focused learning through internal knowledge transfer sessions, peer collaborations, and practical exposure.
Given the application's peculiarity and specificity, getting accustomed to these tech stacks was crucial before taking on live projects and providing noticeable support to the team.

Understanding the unique business workflows and Cashify’s codebase structure was essential before contributing to ongoing development initiatives. Once familiar with the tools, frameworks, and architectural standards followed by the central pod and other product teams, I began working on live projects and production codebases, delivering meaningful contributions that directly impacted both internal operations and end-user experiences

3.1 Incentive Program Phase 2 – ERP Integrated Redemption Flow

The team faced considerable challenges in streamlining the incentive redemption process, particularly in integrating secure KYC validation with ERP workflows. These challenges involved overcoming asynchronous document verification issues, handling third-party API dependencies, and ensuring data consistency across multiple systems. Solving these problems required backend architectural improvements, coordinated process flows, and robust error-handling mechanisms to ensure system reliability, data integrity, and financial compliance.

3.1.1 Unstructured KYC Process

Previously, users could create and use incentive wallets without completing mandatory KYC verification. This lack of validation posed serious compliance risks and opened up the system to potential fraud, identity misrepresentation, or duplicate entries. There was no mechanism in place to validate either user identity or their bank account details.

3.1.2 Manual Bank Entry During Redemption

Every time a user initiated a redemption, they were required to manually enter their bank details. This repeated manual step introduced frequent errors, delayed payouts, and increased the risk of fraudulent or mismatched transfers due to incorrect data entry.

3.1.3 Lack of Real-Time Dashboards

There was no centralized dashboard for the finance team to track KYC status, monitor redemption activity, or view incentive transactions. This lack of visibility made reconciliation highly time-consuming and error-prone, especially during audits or when processing large volumes of transactions.

3.1.4 No TDS or UTR Tracking

The earlier system did not support tracking of Tax Deducted at Source (TDS) or UTR (Unique Transaction Reference) numbers associated with redemption payments. This resulted in gaps in financial records and made it difficult for the finance team to verify payment statuses, ensure statutory compliance, or generate audit-ready reports.

3.2 Tier-Pincode Override Panel—Context-Aware Pricing Flow

The team encountered significant challenges in enabling granular control over pricing logic across different regions and partner sources. The existing system applied a uniform tier-based pricing structure determined solely by the user's pincode, which lacked flexibility and did not support business-driven exceptions. Addressing this limitation required designing a configurable system that could override the default logic based on real-time combinations of source, product line, and geography, all while maintaining system integrity and backward compatibility.

3.2.1. Rigid Geography-Based Pricing

Previously, all customer-facing prices were tied to a fixed tier based on pincode, with no provision to account for source-specific pricing strategies. This rigidity made it difficult to accommodate special business rules for specific partners like Samsung or for targeted campaigns.

3.2.2. No Override Mechanism

There was no interface or backend mechanism to map a different tier to a specific source and region. Any such exception required code-level changes and redeployment, delaying execution and increasing developer dependency.

3.2.3. Absence of a Configurable Admin Tool

Business and operations teams had no tool to configure pricing overrides dynamically. This led to misaligned pricing in production and prevented timely reactions to market or partner-specific requirements.

3.2.4. Risk of Mispricing and Manual Errors

Due to the lack of mapping controls, incorrect pricing was often surfaced to customers. Manual intervention was needed to fix issues, leading to operational inefficiencies and loss of business opportunities.

3.3 Centralized UI Testbed—Unified Visual Consistency Flow

The team encountered substantial challenges in maintaining a consistent user interface across various product pods at Cashify. Each pod independently developed its UI components, leading to fragmented implementations and inconsistent visual elements across the application. The absence of a centralized system to preview, validate, and manage shared UI components hindered collaboration, delayed testing cycles, and resulted in visual regressions that negatively impacted user experience. To address these issues, a robust, modular testbed platform was envisioned to bring design standardization, improve development efficiency, and enhance visual quality across the board.

3.3.1. Fragmented Design Language Across Pods

Different teams implemented UI components like buttons, input fields, and cards with varying styles and behaviors due to the absence of a common reference. This led to a broken and non-cohesive visual experience on the platform.

3.3.2. No Visual Preview Before Integration

There was no pre-integration testing environment where developers could preview how global tokens (like font sizes and colors) would impact other modules. This often resulted in visual side effects that went undetected until late in the release cycle.

3.3.3. Lack of a Shared Component Library

Without a versioned and accessible UI component library, teams often recreated existing components instead of reusing them, increasing duplication, effort, and technical debt.

3.3.4. Manual QA and Design Review Bottlenecks

QA and design teams had to manually inspect every screen and flow for consistency and bugs, leading to longer testing cycles and higher chances of missing visual discrepancies.

4. Methodology

Throughout the internship, I learned a lot of tech stacks and frameworks, including internal company and open-source frameworks. For the open-source frameworks, I have learned them from YouTube, and for the internal frameworks, I have read company blogs and hands-on exercises to learn about frameworks. For the initial three months of the internship, I learned various tech stacks and made demo apps with those tech stacks to get familiar with them.

4.1 Incentive Redemption Flow – ERP Integrated Phase 2

4.1.1 KYC Flow Implementation

The first major task involved implementing a secure and structured KYC journey within the Cashify Switch App. I helped define the UI/UX for KYC screens and implemented form validation logic for fields like PAN number, bank account, and IFSC code. I integrated APIs that securely submitted user documents for verification.

4.1.2 Wallet Creation and Vendor Mapping

Post-KYC, I worked on the logic required to create wallets only for verified users and initiate vendor creation within the ERP system. I ensured proper request-response validation with backend services and added error handling to deal with failed vendor creation attempts.

4.1.3 Redemption Request Workflow

I helped design the form where users could initiate wallet redemption requests. This involved linking verified bank account details to avoid re-entry and building a confirmation interface to minimize fraud or input mistakes. The data flow was connected to backend services that checked redemption eligibility and updated ERP records.

4.1.4 Finance Dashboard Setup

To support internal teams, I contributed to the front end of the Finance Reconciliation Dashboard, which displayed the KYC status, redemption history, and failure reports. I integrated filters and export capabilities to simplify audit processes.

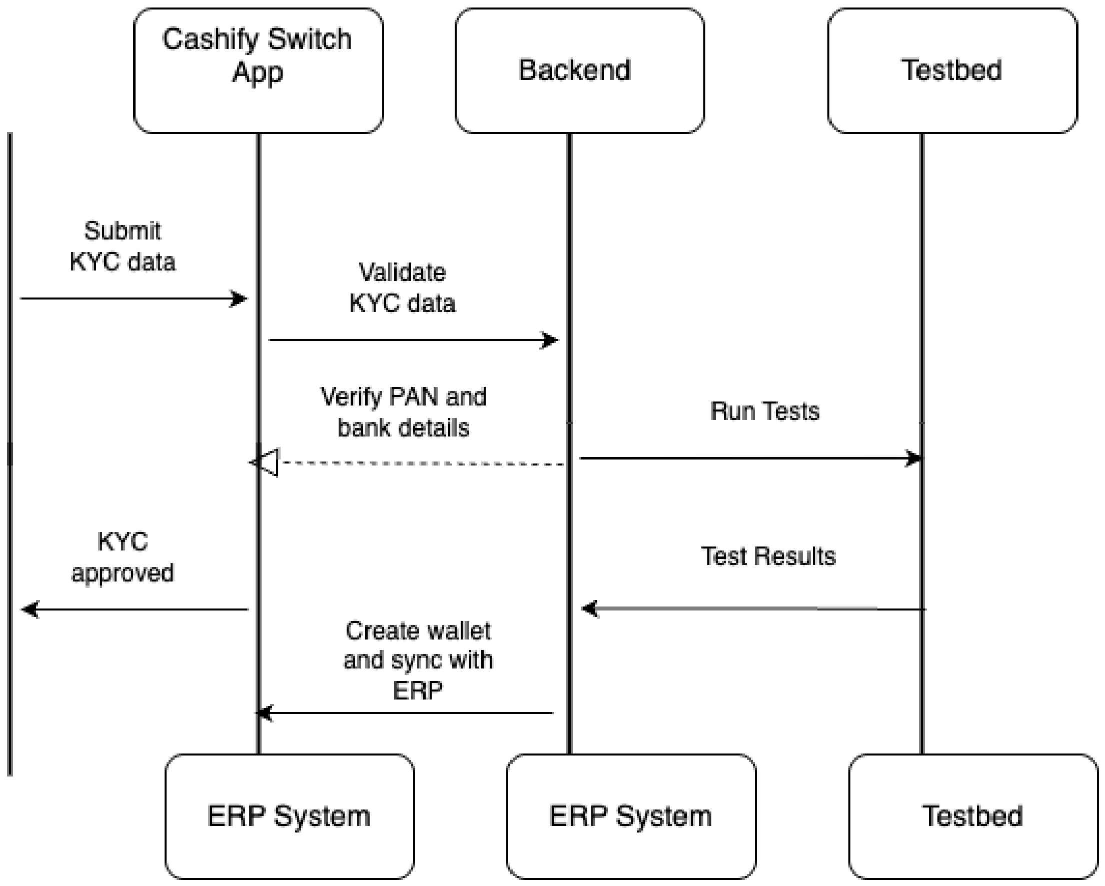


Figure 2: Sequence Diagram

4.2 Centralized UI Testbed – Unified Visual Consistency Flow

4.2.1 Component Library Foundation

The first phase involved creating a baseline set of reusable UI components (buttons, inputs, chips, etc.) using React and Tailwind CSS, structured within a shared component library. These components were created in a modular manner and showcased in various states—default, hover, active, and disabled.

4.2.2 Design Token Integration

Design tokens like --primary-color, --font-size-base, and --border-radius-sm were added to the root config and made dynamic using CSS variables. I enabled live editing of these tokens within the testbed to visualize theme shifts in real time.

4.2.3 Testbed Preview Implementation

I implemented a live preview environment similar to Storybook, where components were rendered in isolation. This environment helped verify their behavior across multiple states and themes. I also integrated device simulation toggles (mobile, tablet, desktop) for responsive validation.

4.2.4 Pod-Specific Module Showcase

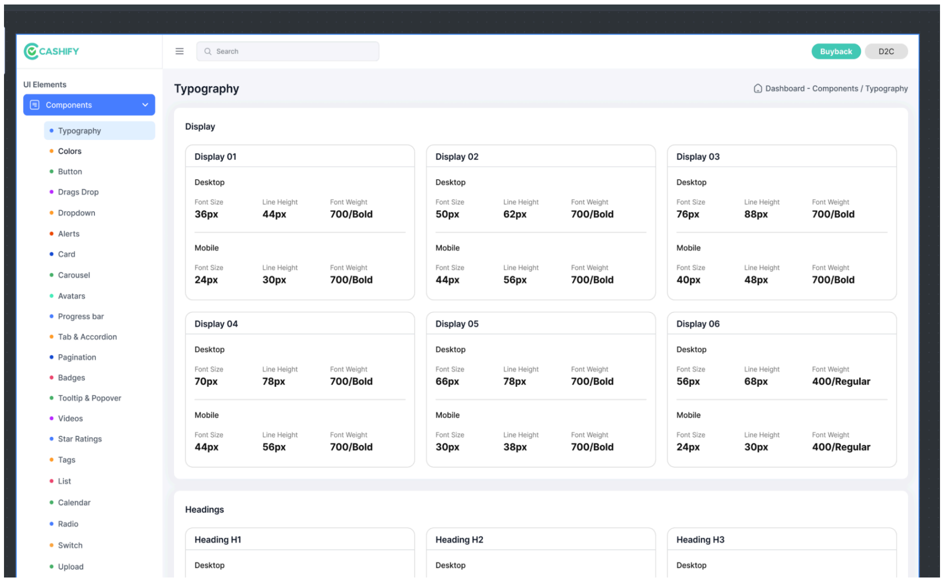
To support customization, I worked on a structure that allowed each product pod to contribute their UI flows, such as pricing cards or calendar selectors. These components followed the shared design principles while allowing flexibility in structure.

4.2.5 Versioning and Documentation

I helped establish a basic versioning system and added documentation links (e.g., Figma files and implementation notes) for each component. This encouraged collaboration across pods and maintained a single source of truth.

UI Features of Testbed

Typography



Typography

Display

		Font Size	Line Height	Font Weight
Display 01	Desktop	36px	44px	700/Bold
	Mobile	24px	30px	700/Bold
Display 02	Desktop	50px	62px	700/Bold
	Mobile	44px	56px	700/Bold
Display 03	Desktop	76px	88px	700/Bold
	Mobile	40px	48px	700/Bold
Display 04	Desktop	70px	78px	700/Bold
	Mobile	44px	56px	700/Bold
Display 05	Desktop	66px	78px	700/Bold
	Mobile	30px	38px	700/Bold
Display 06	Desktop	56px	68px	400/Regular
	Mobile	24px	30px	400/Regular

Headings

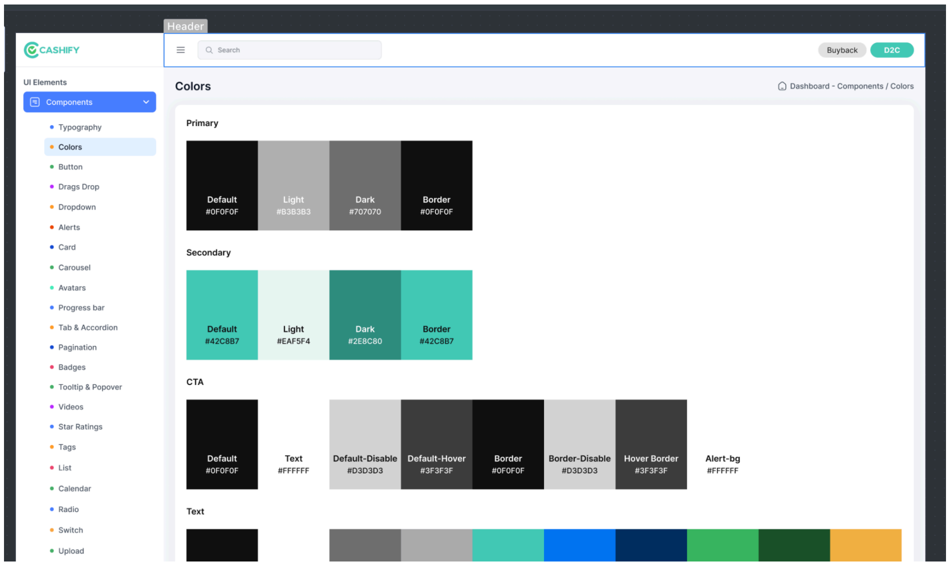
Heading H1 — Desktop

Heading H2 — Desktop

Heading H3 — Desktop

Figure 3: Typography Card

Buttons



Colors

Primary

Default	Light	Dark	Border
#0F0F0F	#B3B3B3	#707070	#0F0F0F

Secondary

Default	Light	Dark	Border
#42C8B7	#EAF5F4	#2E8C80	#42C8B7

CTA

Default	Text	Default-Disable	Default-Hover	Border	Border-Disable	Hover Border	Alert-bg
#0F0F0F	#FFFFFF	#D3D3D3	#3F3F3F	#0F0F0F	#D3D3D3	#3F3F3F	#FFFFFF

Text

Figure 4: Button Card

And Similar for Alerts and Badges etc..

4.3 Tier-Pincode Override Panel – Context-Aware Pricing Flow

4.3.1 CSV Upload & Mapping Interface

I developed the UI panel that allowed business users to upload CSV files containing combinations of source, pincode, tier, and product line. I implemented real-time validations to prevent duplicate mappings and incorrect tier values.

4.3.2 Manual Override Panel

In addition to bulk uploads, I designed a manual override form where a user could map one source-tier-product combination to multiple pincodes at once. This involved setting up dependent dropdowns and ensuring only valid tiers appeared for selected product lines.

4.3.3 Preview & Filter Controls

I added dynamic filter controls to search for existing mappings based on pincode, tier, source, and product line. I also implemented a preview panel to allow users to confirm override changes before saving them to the backend.

4.3.4 Backend Integration & Error Handling

To connect the UI with the backend services, I implemented API calls using Axios and added standardized error messages for common issues such as 409 conflicts (duplicate entries) or 500 internal errors. I also ensured that updates were reflected in real-time after each successful save.

5. Tools and Technologies Used

5.1 React

Figure 5: React Logo

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used as a base in the development of single-page applications (SPAs), mobile applications, or even server-rendered applications when paired with frameworks like Next.js [3]. The library encourages the creation of reusable and composable components that present dynamic data and enhance user interactivity over time.

React abstracts away the complexities of directly manipulating the Document Object Model (DOM), offering a simpler programming model that boosts performance. Through its virtual DOM [4], React minimizes costly updates to the actual DOM by first updating a virtual representation in memory, only reflecting changes to the real DOM when necessary.

One of the core features of React is JSX (JavaScript XML), a syntax extension that allows developers to write HTML-like elements within JavaScript. This makes it easy to visualize the structure of a component and its UI without losing the power of JavaScript logic. React's component-based architecture is another key feature, allowing developers to break down complex UIs into smaller, reusable parts. [5]

5.2 Next.js

Figure 6: Next.js Logo

Next.js is a popular open-source JavaScript framework built on top of React, designed for building fast, scalable web applications. It enables developers to create both server-rendered and statically generated websites with ease. Next.js supports automatic code splitting, which helps optimize performance by loading only the necessary code for each page. It also offers server-side rendering (SSR) and static site generation (SSG), making it a versatile framework for building dynamic applications that require SEO optimization and fast load times.

One of Next.js's standout features is its built-in routing system, where each file in the pages directory automatically becomes a route, simplifying the setup of page navigation. It also supports API routes, allowing developers to create backend endpoints within the same application without the need for a separate server. This makes it particularly convenient for full-stack development.

Next.js also comes with powerful optimizations like image optimization and automatic static optimization, reducing the need for manual configuration and allowing for faster page rendering. It has strong support for TypeScript, CSS Modules, and other modern web development features, making it highly flexible and easy to integrate into any project.

Overall, Next.js simplifies the process of developing React-based applications by offering features like SSR, SSG, API routes, and automatic performance optimizations out of the box, making it an ideal choice for building production-ready applications [6].

5.3 Typescript

Figure 7: Typescript Logo

TypeScript is a statically typed, object-oriented programming language developed by Microsoft. As a superset of JavaScript, it enables developers to write robust and maintainable code by incorporating optional static type checking. This feature is particularly beneficial for large-scale JavaScript applications, as it helps prevent type-related errors and enhances code quality.

TypeScript's key features include static type checking, which ensures that variables, parameters, and function return values are correctly typed. It also supports classes, modules, and interfaces, allowing developers to write true object-oriented code. The language is designed to be compatible with ES6 features [7], making it easy for developers familiar with ES6 syntax to transition to TypeScript.

TypeScript is not directly run on the browser; instead, it needs to be compiled into JavaScript using the TypeScript Compiler (TSC). This compilation process makes it easy to integrate TypeScript code into existing JavaScript projects. With its ability to enhance the development experience, TypeScript has gained significant traction in the developer community, making it an attractive choice for developers seeking to improve their coding experience.

TypeScript's ability to catch errors during development, before runtime, significantly reduces the potential for bugs in production environments. This leads to higher-quality code and better maintainability, especially for large projects with multiple developers. Additionally, TypeScript's powerful type inference system makes it easier for developers to understand and work with complex codebases, improving overall productivity. The language also integrates seamlessly with popular JavaScript libraries and frameworks, further increasing its adoption across various development environments.

5.4 Tailwind CSS

Figure 7: Tailwind CSS Logo

Tailwind CSS is a utility-first CSS framework that provides low-level utility classes to build custom designs quickly and efficiently. Unlike traditional CSS frameworks, which offer predefined components like buttons and navigation bars, Tailwind enables developers to create their own designs directly within the HTML using utility classes. These utility classes cover a wide range of design properties, such as spacing, colors, typography, and borders, allowing for fine-grained control over every aspect of the layout without the need for writing custom CSS.

One of the key benefits of Tailwind CSS is its flexibility and scalability. Developers can easily configure the framework to fit their specific project requirements by adjusting settings in the configuration file, such as colors, font sizes, breakpoints, and more. This customization allows for a highly tailored design system while still leveraging the utility-first approach that promotes consistency across the application. As a result, developers can achieve unique designs without worrying about clashing CSS styles or repetitive code.

Tailwind also promotes a more maintainable approach to CSS by reducing the need for writing custom class names and selectors. Since utility classes are small and reusable, developers can avoid large, bloated CSS files and instead use the classes directly in the markup, which leads to cleaner, more efficient code. Additionally, the framework includes a powerful purge feature, which removes unused styles in production builds, further optimizing the performance of the website or application.

One of the standout features of Tailwind CSS is its strong community support and ecosystem. There are numerous plugins, extensions, and integrations available that can enhance the functionality of Tailwind, such as form controls, animations, typography utilities, and more. Tailwind also integrates smoothly with modern front-end frameworks like React, Vue, and Angular, making it a popular choice for both traditional websites and single-page applications. With its growing popularity, Tailwind has become an essential tool for developers who want to create responsive, customizable, and performance-optimized user interfaces.

5.5 Redux

Figure 8: Redux Logo

Redux is a state management library for JavaScript applications, commonly used with React. It provides a predictable way to manage application state by using a single, immutable store that holds the entire state. The state can only be modified by dispatching actions, which are processed by pure functions called reducers. This structure makes data flow clear and helps maintain a consistent state across the application [8].

The core concept of Redux is the store, which holds the application state, and actions, which describe changes to that state. Actions are processed by reducers, pure functions that determine how the state should change..

Redux also supports middleware, which allows developers to add custom logic between dispatching actions and updating the state. Middleware like Redux Thunk or Redux Saga is useful for handling asynchronous actions and side effects, such as API calls. This gives developers more flexibility in managing complex application workflows, like handling real-time data or performing background tasks.

In summary, Redux offers a predictable and maintainable way to manage state in JavaScript applications, especially in large, complex projects. Its structure, combined with tools like middleware, makes it a powerful solution for managing application state effectively. As the ecosystem grows, Redux continues to be a popular choice for developers seeking a robust and scalable state management solution.

5.6 React Native

Figure 9: React Native Logo

React Native is an open-source framework developed by Facebook for building mobile applications using JavaScript and React. It allows developers to create cross-platform apps for iOS and Android with a single codebase, significantly reducing development time and costs.[9] By combining the flexibility of React with native mobile components, React Native provides a way to develop high-performance applications that deliver a native-like experience.

One of the key features of React Native is its ability to render UI components as native components, rather than using webviews like traditional hybrid apps. This results in better performance and smoother user experiences. Additionally, React Native supports hot-reloading, allowing developers to instantly see changes in the app without rebuilding, which speeds up the development process.

React Native communicates between JavaScript and native code through a bridge, enabling developers to access platform-specific APIs or write custom native modules in languages like Java, Swift, or Objective-C when needed. This flexibility makes it easy to incorporate platform-specific features while maintaining a unified codebase.

With a rich ecosystem of third-party libraries and strong community support, React Native has become a popular choice for building mobile apps. It is used by major companies like Facebook, Instagram, and Airbnb, proving its scalability and reliability for large-scale applications. React Native offers a balance between performance, efficiency, and development speed, making it an ideal framework for mobile app development.

5.7 Git

Figure 10: Git Logo

Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows (thousands of parallel branches running on different systems). It is very helpful in version control, as it is often used with GitHub. [10]

5.8 Jira

Figure 11: Jira Logo

Jira is a popular project management and issue-tracking tool developed by Atlassian. It is widely used by software development teams to plan, track, and manage agile projects. Jira provides a centralized platform for managing tasks, bugs, user stories, and other project-related work, allowing teams to collaborate effectively and stay on top of project timelines.

Jira supports various project management methodologies, including Agile, Scrum, and Kanban. It allows teams to create customizable workflows, assign tasks, and track progress with boards, sprints, and burndown charts. The tool also integrates with other Atlassian products like Confluence and Bitbucket, as well as third-party tools, providing a comprehensive solution for managing software development projects.

5.9 Postman

Figure 12: Postman Logo

Postman is a popular API testing and development tool that simplifies the process of building, testing, and managing APIs. It is widely used by developers and QA engineers to ensure that APIs work as intended before they are integrated into applications. Postman allows users to send HTTP requests, inspect responses, and automate testing workflows, making it an essential tool for API development and testing.

One of the key features of Postman is its user-friendly interface, which allows developers to quickly compose and send requests without writing complex code. Users can define request methods (GET, POST, PUT, DELETE, etc.), add parameters, headers, and authentication details, and view detailed responses, including status codes, response times, and data. Postman also supports multiple formats for request and response bodies, including JSON, XML, and form data.

Postman provides powerful testing capabilities with its built-in scripting environment. Developers can write test scripts in JavaScript to validate API responses, ensuring that the data meets expected conditions. It also allows for the automation of repetitive tasks, such as running a series of tests or executing requests in a collection. This makes it ideal for continuous integration (CI) workflows and for teams working in agile environments.

Additionally, Postman offers collaboration features that allow teams to share collections, environments, and test scripts. This is particularly useful for teams working on large projects, where consistency in testing and API management is crucial. With support for versioning, monitoring, and integrating with third-party services, Postman has become a go-to tool for API development, testing, and documentation.

5.10 Zeplin

Figure 13: Zeplin Logo

Zeplin is a design collaboration tool that helps streamline the handoff process between designers and developers. It allows designers to upload their designs, which are then automatically converted into design specifications, assets, and code snippets for developers. Zeplin ensures that both design and development teams are on the same page by providing a shared platform for viewing and commenting on design files, making collaboration more efficient and reducing errors in the implementation phase.

6. Observations and Findings

After accomplishing the required tasks mentioned above, the team made the following observations: Compared to what the team had anticipated, the actual changes to the code showed more promising results.

These observations are listed task-wise for clarity purposes, showing clear improvements from the previous implementations and how they helped the team.

6.1 Incentive Redemption Flow

1. PAN and Penny Drop verification automated the KYC process and reduced manual validation time.
2. Email notifications improved user clarity regarding KYC status and next steps.
3. Centralized logging helped in tracking failed validations and resolving issues efficiently.
4. Automated ERP vendor creation minimized manual data entry and reduced errors.

5. Wallets were only created post-KYC approval, ensuring compliance and preventing misuse.
6. User states (pending, approved) were clearly handled in the system, improving process control.
7. Redemption flow with TDS display helped users understand deductions and reduced support queries.
8. Redemption details, including UTR numbers and payable amounts, were captured accurately.
9. CSV-based UTR uploads simplified batch payment processing for the finance team.
10. Real-time dashboards provided better visibility into KYC, redemption, and wallet status.
11. Downloadable reports improved financial reconciliation and audit readiness.
12. Manual actions like document rejection and UTR updates were made easier via the console panel.

6.2 Centralized UI Testbed

1. Enabled seamless visual validation and testing of UI components across pods and teams.
2. Reduced front-end release bugs by catching inconsistencies early in development.
3. Improved collaboration between product managers and developers by offering a live preview environment.
4. Accelerated UI review cycles and improved turnaround time for design and QA feedback.
5. Ensured standardization of UI components, typography, and interaction patterns across modules.
6. Acted as a reference point for implementing and validating reusable components.

6.3 Tier-Pincode Override Panel

1. Allowed dynamic mapping of product pricing based on tier, pincode, source, and product line combinations.

2. Eliminated the limitation of a static pricing structure by enabling granular control at the regional level.

3. Simplified data entry through CSV upload and bulk mapping functionality, improving operational efficiency.

4. Reduced pricing errors and ensured better alignment with localized market strategies.

5. Enabled faster updates and overrides without code changes, enhancing agility for the business team.

6. Provided download and update functionality, which made bulk data handling more transparent and manageable.

7. Limitations

7.1 Incentive Redemption Flow

1. KYC validation depends on third-party APIs, causing potential delays or failures.

2. KYC is processed via cron jobs, not in real-time, which slows user feedback.

3. Partial KYC failures require users to re-upload all documents.

4. ERP downtime or API errors can block vendor creation and wallet activation.

5. No retry mechanism exists for failed wallet creation due to ERP issues.

6. Hardcoded TDS rates require manual backend changes for tax updates.

7. UTR updates rely on manual CSV uploads, delaying real-time redemption status.

8. Redemption requests can’t be edited once submitted, reducing flexibility.

9. Finance dashboard performance may slow with large data volumes.

10. Limited customization and role-based access in reporting tools.

11. UI Testbed supports only visual validation, not functional or responsive testing.

12. Component previews in Testbed can become outdated without manual syncing.

13. The Tier-Pincode Panel lacks conflict checks and rollback options for incorrect uploads.

14. Bulk upload errors in tier mapping lack detailed validation feedback.

7.2 Centralized UI Testbed

1. Only supports visual validation; does not cover functional or automated testing.

2. Requires manual updates to stay in sync with the latest component changes.

3. Does not support real-time responsiveness checks across different screen sizes.

4. Limited integration with external QA/testing tools for end-to-end validation.
5. Lacks role-based access or version control, which can cause inconsistencies in shared environments.

7.3 Tier-Pincode Override Panel

1. No built-in validation to detect conflicts in tier mappings for the same pincode.

2. Bulk upload errors lack detailed feedback, making troubleshooting difficult.

3. No rollback or undo functionality for incorrect or accidental uploads.

4. Manual CSV handling can introduce human errors if not properly reviewed.

5. Does not support versioning or change history for audit or recovery purposes.

8. Conclusions and Future Work

8.1 Incentive Redemption Flow

The successful implementation of Incentive Program Phase-2 brought significant improvements in the incentive redemption process. By introducing a structured KYC flow, automated ERP vendor creation, and wallet activation only upon compliance approval, the project ensured enhanced security, transparency, and adherence to financial regulations. Additionally, the integration of TDS handling, UTR tracking, and finance dashboards streamlined internal workflows and improved user satisfaction.

Future Work

- Enable real-time KYC validation instead of daily cron jobs for faster user onboarding.
- Integrate automated payment gateways to eliminate reliance on manual UTR uploads.
- Implement retry and rollback mechanisms for ERP sync and wallet creation failures.
- Introduce version control and audit trails for Tier-Pincode overrides.
- Enhance dashboard performance with pagination and advanced filters.
- Expand UI Testbed to support functional and responsive testing.

8.2 Centralized UI Testbed

Conclusions

The Centralized UI Testbed proved to be a valuable tool for improving front-end development efficiency across pods. It enabled developers and designers to preview and validate reusable components in isolation, ensuring visual consistency and reducing UI-related bugs before deployment. It also facilitated smoother collaboration between tech and product teams by serving as a live reference for shared components.

Future Work

- Extend the testbed to support functional and integration testing alongside visual validation.
- Add responsive testing capabilities for different screen sizes and devices.
- Implement auto-sync with the component library to ensure previews are always up to date.
- Introduce role-based access and version tracking for better collaboration and auditability.
- Integrate with CI/CD pipelines to automatically test new component changes.

8.3 Tier-Pincode Override Panel

Conclusions

The Tier-Pincode Override Panel significantly enhanced the pricing system's flexibility and responsiveness. By decoupling pricing logic from hardcoded geographic tiers, it allowed business teams to apply context-aware pricing based on source, product line, and region. The CSV-based configuration enabled faster rollouts of pricing changes without requiring backend deployments, reducing dependency on developers. It also improved pricing accuracy and consistency across diverse business cases, supporting better alignment with partner requirements.

Future Work

- Enable in-app edit and delete functionality for override entries to reduce reliance on CSV files.
- Introduce bulk preview and validation before upload to prevent errors.
- Implement role-based access control to allow granular permissions for pricing teams.
- Integrate audit logs to track changes made in override mappings.

9. Bibliography/References

[1] Monk, Ellen F., and Bret J. Wagner. *Concepts in Enterprise Resource Planning*. Cengage Learning, 2012.

[2] Bertini, M., & Koenigsberg, O. (2023). Dynamic pricing: Definition, implications for managers, and future research directions. *Journal of Retailing*, 99(4), 580–593.

[3] React, "*Getting started—React documentation,*" Meta, 2024. [Online].

[4] JavaTpoint, "React Features," JavaTpoint.com, [Online]. Available: https://www.javatpoint.com/react-features.

[5] Procoders, "Advantages of Using ReactJS," Procoders.tech, [Online]. Available: https://procoders.tech/blog/advantages-of-using-reactjs/.

[6] Vercel, "*Next.js documentation,*" 2024. [Online].

[7] TypeScript, "TypeScript Documentation," TypeScriptlang.org, [Online]. Available: https://www.typescriptlang.org/docs/.

[8] Redux State Management System—A Comprehensive Review" by Krutika Patil (2022)

[9] Evaluating the Performance of React Native and .NET MAUI for Cross-Platform Mobile Development: A Comparative Study" by Stiv Abdullwahed and Xuan Dung Tran (2023)

[10] "An Overview of Git" by Gayatri Makrand Ghodke and Trupti Chavan (2024)

10. Appendix A: Evaluation Form for Peer Review

Name of the student: (to be reviewed)	Vartika Gautam	**Roll no. of the student:**	102103397
Title of the project:	Cashify Web and Console Features		
Name of the company:	Cashify		
Project report (Tick the appropriate)	Excellent. ✓	Good	Average
Project poster (Tick the appropriate)	Excellent ✓	Good	Average
Project video (Tick the appropriate)	Excellent ✓	Good	Average
Rate the work done	0 – 10 points	*(Provide rating here) →*	**5**
Give marks to the student on the basis of the overall performance	0 -5 marks	*(Provide marks here) →*	**5**

Abstract of the project (max. 100 words):
During my internship at Cashify, I contributed to enhancing both web and console-based application features. I gained hands-on experience in full-stack development, system integration, and workflow optimization. The project deepened my understanding of real-world application architecture and agile collaboration. It significantly improved my technical, debugging, and problem-solving skills.

Mention three strengths of the work done:

1. **Robust System Integration**: Seamlessly connected frontend features with backend workflows, ensuring smooth and reliable operations across modules.
2. **Scalable Architecture**: Contributed to building features that are easily maintainable and scalable for future enhancements.
3. **User-Centric Design**: Prioritized usability and compliance, resulting in intuitive interfaces and efficient validation mechanisms.

Provide some useful recommendations (It may be some improvements, some suggestions to further raise the quality of the project):

1. Integrate unit, integration, and end-to-end tests to improve code reliability and minimize manual QA efforts.
2. Provide detailed technical documentation and onboarding guides to help new developers quickly understand system architecture and workflows.
3. Structure the UI using reusable and isolated components to enhance maintainability and ensure design consistency across the platform.

Name of the **evaluator** student:	**Kartikye Chaddha**	Roll no. of the **evaluator** student:	102103708
Signature of the **Evaluator** student:	Kartikye		