Midway Report
on
Cashify Web and Console Feature

Faculty Mentor:
Mr. Saif Nalband

Submitted by:
Vartika Gautam
(102103397)

Industrial Mentor:
Mr. Hemchand Yadav

Computer Science and Engineering Department
Thapar Institute of Engineering & Technology, Patiala

1. Introduction/Background:

During the course of my association with Cashify, I have worked extensively on enhancing the **Cashify web platform, mobile app, and internal console systems**. My work spanned across a wide range of technologies, including **React, Next.js, TypeScript, Tailwind CSS and React Native** I focused on building high-performance, scalable systems that support real-time adaptability without requiring redeployment. This experience enabled me to contribute effectively to both frontend and backend development, ensuring a seamless and efficient experience for users and internal teams alike.

2. Problem Statement/Domain:

Incentive Redemption Flow—Phase 1

1. **Lack of Seamless Integration Across Systems**
   The KYC-to-wallet flow had multiple dependencies across ERP, finance, and app systems, creating challenges in ensuring smooth data synchronization and compliance across all platforms. This often led to data discrepancies and inefficiencies in the KYC process.

2. **Complex User Interface Design for Compliance**
   Designing a user-friendly KYC UI that also adheres to regulatory compliance requirements proved challenging. Ensuring clarity in error and success states while managing document uploads and maintaining a simple user experience created additional complexity.

3. **Manual Handling of KYC Verification**
   The verification process for PAN and bank details required manual intervention or was vulnerable to failures due to the integration of third-party services like document services and PennyDrop. This led to delays and potential issues with user validation, rejection handling, and retries.

Incentive Redemption Flow—Phase 2

1. **Delayed KYC Processing and Integration with Backend**
   The integration of the "Complete KYC" form with React and TypeScript, along with third-party services, led to delays in the KYC approval process. Ensuring seamless submission, validation, and document upload with proper previews and error handling was complex and time-consuming.

2. **Inefficient Error Handling and Retry Mechanisms**
   The backend logic for KYC approval, wallet creation, and ERP synchronization was prone to errors due to inconsistent error handling and retry mechanisms. These failures required manual intervention, reducing overall system efficiency and increasing operational workload.

3. **Lack of Visibility for Finance Team**
   While the backend logic for syncing data with ERP was in place, the lack of a user-friendly dashboard for the finance team delayed validation and approval processes. Without a dedicated dashboard, tracking and managing user status and redemption reports became cumbersome.

Testbed Implementation

1. **Inconsistent UI and Feature Validation Across Systems**
   The absence of a unified testbed environment created difficulties in ensuring that UI changes were consistently validated across various systems before being implemented. This led to fragmented testing and missed visual discrepancies that impacted the final user experience.

2. **Lack of Real-Time Feedback for UI Changes**
   The testbed lacked real-time integration with other systems, resulting in a delayed feedback loop for UI changes. This impacted the ability to preview global style updates or ensure that components displayed correctly before pushing them live.

3. **Manual Validation and Inefficiencies in Testing**
   The manual inspection of UI components in the Testbed led to inefficiencies in testing and validation, often resulting in UI bugs being discovered late in the development cycle, increasing the time to fix issues and pushing back delivery timelines.

3. Techniques/Tools/Technologies Used:

1. **Next.js & TypeScript**
   Next.js and TypeScript were used to build the front end of the KYC journey, ensuring efficient rendering of UI components and type safety across the application. React's component-based architecture facilitated modular development, while TypeScript provided type-checking and helped prevent errors during the development process.

2. **API Integration**
   APIs were designed and integrated for KYC document submission, PAN verification, and bank validation. RESTful APIs were used for smooth communication between the front end and the back end, ensuring the validation processes were accurate and efficient.

3. **Third-Party Services**

   - **PAN Verification**: Integrated third-party PAN verification services to validate the authenticity of PAN details provided by users.

   - **PennyDrop**: Used for bank account validation, ensuring that the provided bank details were accurate and valid for wallet creation.

4. **ERP Integration**
   The project involved integrating the KYC process with the ERP system, ensuring that vendor creation and user data were pushed correctly after successful KYC completion. This integration ensured synchronization between the app, finance, and ERP systems.

5. **Email Automation**
   Automated email notifications were implemented for user communication, including approval, rejection, and resubmission messages, to keep users informed throughout the KYC and wallet activation process.

6. **Backend Technologies**
   Server-side logic and database management were handled to process KYC approval, wallet creation, and ERP synchronization. This involved designing the backend systems to handle document uploads, manage user data, and ensure smooth system integration.

7. **Testbed Environment**
   A **testbed** environment was set up to validate UI and backend integrations. The testbed provided a space to run simulations and tests, ensuring that the system worked as expected before deployment.

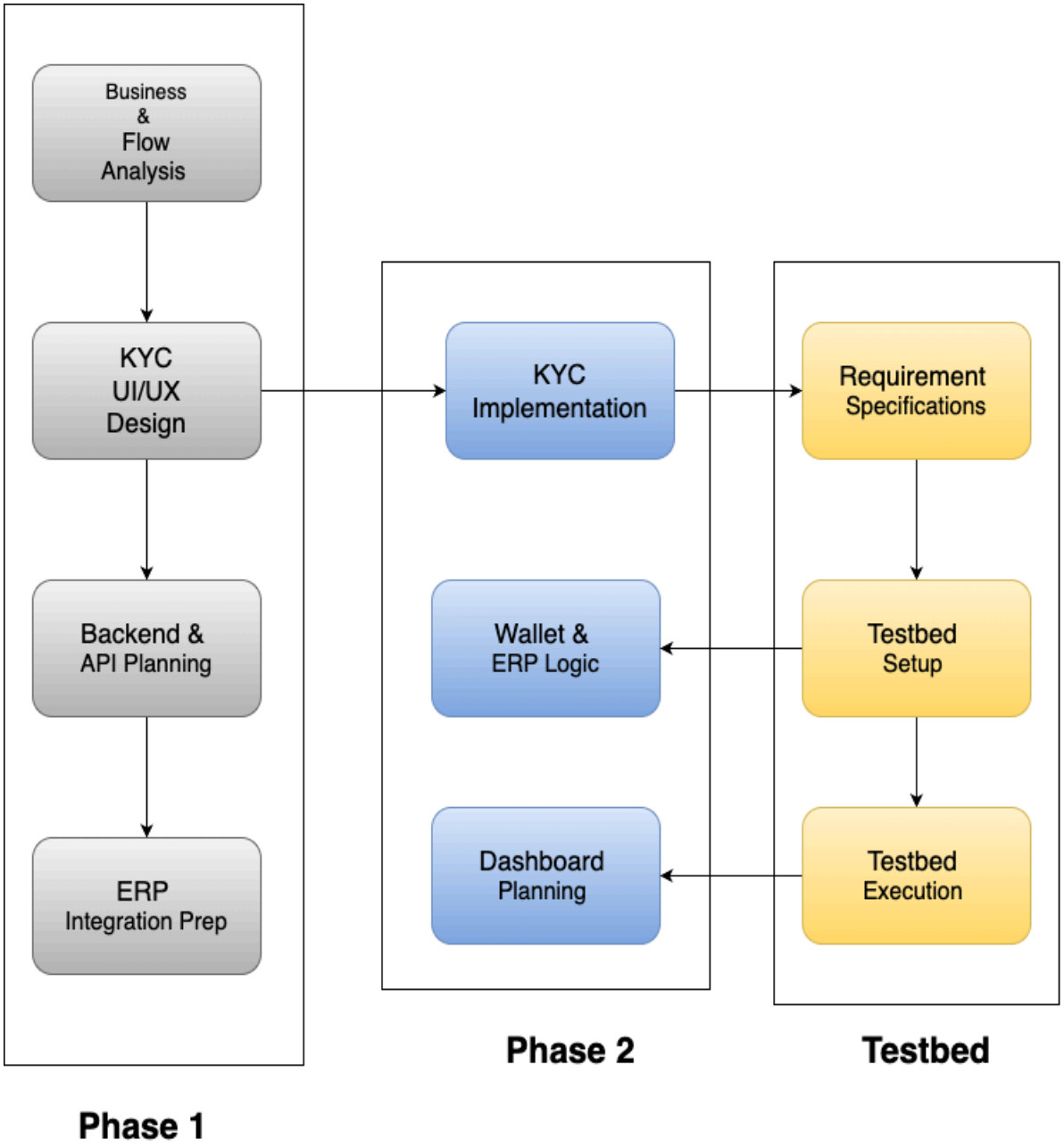
8. **Version Control (Git)**
   Git was used for version control to manage the development process, ensure collaborative work, and track changes made to the codebase during the project's lifecycle.

9. **CI/CD Tools**
   Continuous integration and continuous deployment tools were employed to automate testing and deployment, ensuring seamless integration and fast delivery of new features and fixes.

4. Block Diagram/Architecture/Methodology:

The incentive phase program follows an iterative and collaborative approach, integrating key processes such as KYC implementation, wallet activation, and ERP synchronization. It involves careful planning, including business flow analysis, UI/UX design, and backend development. Throughout the phases, third-party integrations and testing (via a testbed environment) ensure seamless system performance, with real-time feedback and validation being crucial to ensuring the accuracy and efficiency of the process. Each phase focuses on building and refining specific components, ensuring a comprehensive and systematic approach to achieving the project's objectives.

Block Diagram for Incentive Phase Program :



1. **Business & Flow Analysis**
   The process begins with the analysis of the entire KYC-to-wallet flow, identifying the dependencies and interactions across various systems like ERP, finance, and app systems.

2. **KYC UI/UX Design**
   After the analysis, the design of the user interface (UI) for the KYC process is completed, including aspects like document upload sections and user-friendly error/success states.

3. **Backend & API Planning**
   At this stage, the backend and API are planned to support document uploads, PAN and bank verifications, and user validation processes.

4. **KYC Implementation**
   This step involves the actual development of the KYC form, including validation of fields and document uploads. Third-party integrations for PAN and bank validation are also implemented here.

5. **Third-Party Integrations**
   The third-party integrations, like PAN verification services and PennyDrop for bank validation, are set up for KYC to function smoothly.

6. **Wallet & ERP Logic**
   After the KYC process is complete, the system moves on to implementing logic for wallet creation and ERP integration to sync user data with the ERP system.

7. **Dashboard Planning**
   This step focuses on creating dashboards for monitoring KYC approval and redemption statuses, providing real-time insights to finance teams.

8. **Testbed**
   A **testbed** environment is set up to test the various integrations and processes. This includes requirement specification, test setup, and execution, ensuring everything functions smoothly before deployment.

9. **Communication & Docs**
   This step involves the creation of emails for users, including approval, rejection, and resubmission notifications, to ensure seamless communication with customers.

Sequence Diagram of the Incentive Phase :