Progress Report 4

Work Date / Hours Log

Date	Number of Hours	Description of Work Done
Nov 5	3	Established the backend foundation for the billing system. Defined TypeScript types for BillingKpis, PlanBucket, and FailedPayment, and added environment variables for enterprise pricing and webhook secrets. Implemented Stripe helper functions (calculateMRR, getActiveSubscribers, calculateChurnRate, getFailedPayments). Verified all TypeScript compilation.
Nov 5	4	Implemented core billing API endpoints under /admin/billing, including metrics, plan-distribution, and failed-payments. Each endpoint retrieves data from Stripe and Supabase, calculates MRR, ARR, ARPU, and churn rate, and applies short-term caching for performance. Added validation for environment variables (STRIPE_PRICE_ENTERPRISE, STRIPE_WEBHOOK_SECRET) and ensured TypeScript build success.
Nov 5	3.5	Integrated Stripe webhooks to handle subscription and invoice events for cache invalidation. Created webhook router at /webhooks/stripe, verified Stripe signatures, and implemented event handlers to invalidate cached billing metrics and failed payment data upon subscription or payment updates. Added logging and mounted webhook before body parser to ensure correct signature validation.
Nov 6	3	Connected the frontend billing module with live Stripe data. Added billing API functions (fetchBillingMetrics, fetchPlanDistribution, fetchFailedPayments) and replaced mock data with real API calls. Integrated React Query for caching and data fetching in Billing.tsx, removed mock imports, and ensured real-time synchronization between frontend and backend.
Nov 7	1	Online meeting with Gluu CEO Zane, and a short meeting with group members after the meeting with Zane.

Description of Work Done

Over this development cycle, I focused on completing the end-to-end billing integration for the Gluu Admin Dashboard. The work began with establishing the backend foundation, including defining TypeScript billing types, implementing Stripe helper functions for MRR, churn rate, and failed payments, and introducing an in-memory caching layer with TTL. Next, I developed three key admin endpoints—/admin/billing/metrics, /admin/billing/plan-distribution, and /admin/billing/failed-payments—each returning real-time metrics from Stripe and Supabase with caching and pagination support. I then integrated Stripe webhooks to handle subscription and invoice events, ensuring automatic cache invalidation and verified webhook signatures. On the frontend, I replaced all mock billing data with live API calls, using React Query for real-time updates and proper cache durations.

Repo Check-in of Implementation Completed

The files/folders I have checked into the repository so far include:

- Implementation/backend/src/ (branch feat/billing-real-data)
- Implementation/backend/src/lib/cache.ts (branch feat/billing-real-data)
- Implementation/backend/src/types/billing.ts (branch feat/billing-real-data)
- Implementation/backend/src/env.ts (branch feat/billing-real-data)
- Implementation/backend/src/routes/admin.ts (branch feat/billing-real-data)
- Implementation/backend/src/routes/webhooks.ts (branch feat/billing-real-data)
- Implementation/backend/src/server.ts (branch feat/billing-real-data)
- Implementation/frontend/src/lib/adminApi.ts (branch feat/billing-real-data)
- Implementation/frontend/src/pages/Analytics/Billing.tsx (branch feat/billing-real-data)
- Implementation/frontend/src/services/billing.ts (branch feat/billing-real-data)