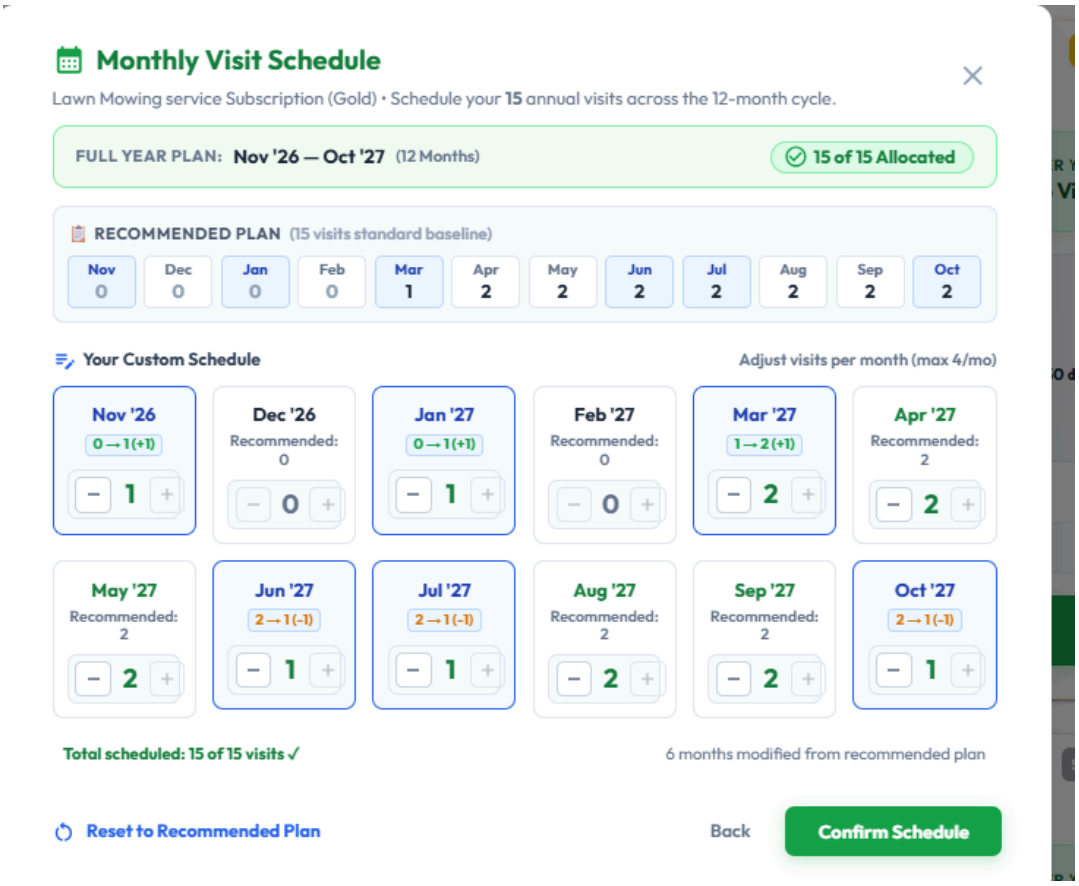
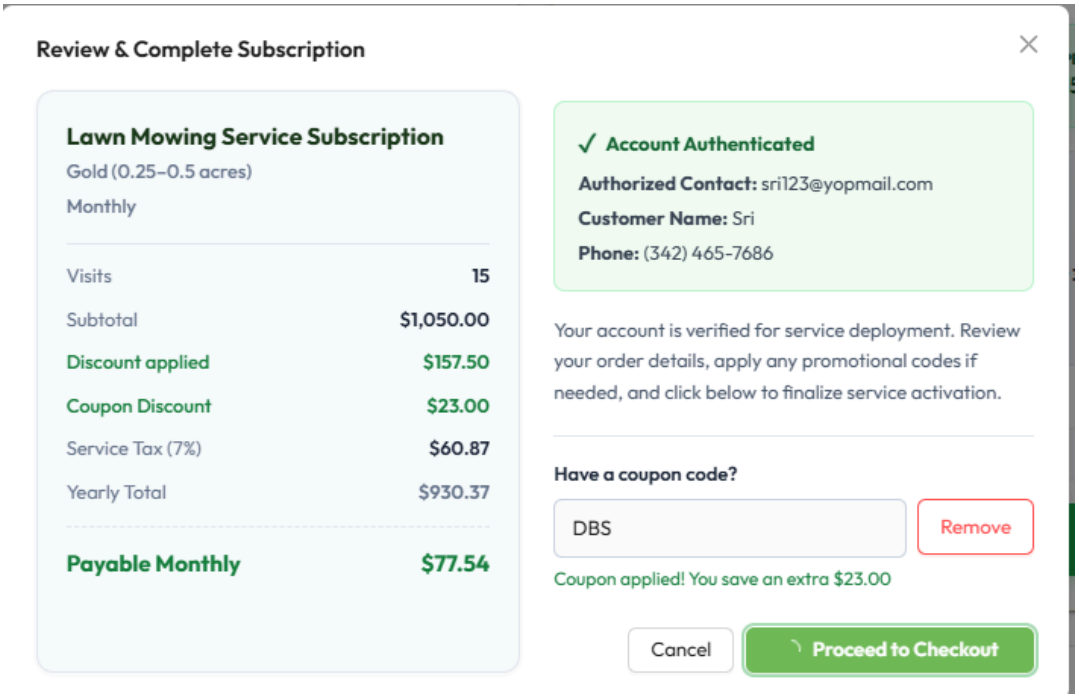
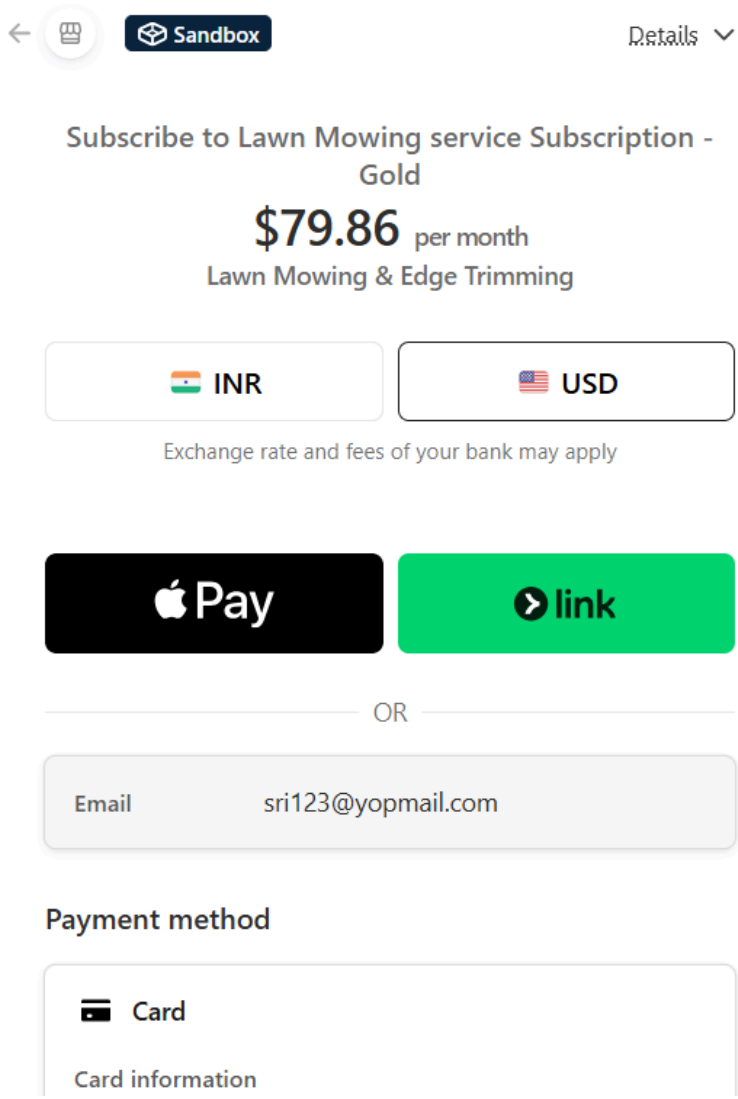
subscription-2/11/26

1. When the Final Settlement Invoice email is sent to customer, the recipient receives twice instead of once. -done
2. Amount Mismatch in list page. - no issue found
3. The second payment was originally scheduled for 30 April 2027. After changing the payment date to 10 May 2027 in Stripe, the second payment was automatically processed immediately instead of being scheduled for the updated date. - no issue found
4. In the Subscription Payment Received email, the Visit Schedule displays the originally created schedule instead of the updated schedule selected by the customer. It should display the latest updated visit schedule. - done
5. On the Create Service Plan page, only 7 plans are visible, and the remaining plans cannot be accessed because scrolling is not working. - done
6. When the second quarterly payment date is changed to the next day in Stripe, the third payment is also automatically processed. However, Stripe still displays only 2 payments as completed, resulting in a mismatch between the actual charges and the payment count. - done
7. 2 or more duplicate payment failed history entries. - done


Monthly Visit Schedule
Lawn Mowing service Subscription (Gold) • Schedule your 15 annual visits across the 12-month cycle.
FULL YEAR PLAN: Nov '26 — Oct '27 (12 Months)
15 of 15 Allocated
RECOMMENDED PLAN (15 visits standard baseline)
Nov 0
Dec 0
Jan 0
Feb 0
Mar 1
Apr 2
May 2
Jun 2
Jul 2
Aug 2
Sep 2
Oct 2
Your Custom Schedule
Adjust visits per month (max 4/mo)
Nov '26 0 → 1 (+1) 1
Dec '26 Recommended: 0 0
Jan '27 0 → 1 (+1) 1
Feb '27 Recommended: 0 0
Mar '27 1 → 2 (+1) 2
Apr '27 Recommended: 2 2
May '27 Recommended: 2 2
Jun '27 2 → 1 (-1) 1
Jul '27 2 → 1 (-1) 1
Aug '27 Recommended: 2 2
Sep '27 Recommended: 2 2
Oct '27 2 → 1 (-1) 1
Total scheduled: 15 of 15 visits ✓
6 months modified from recommended plan
Reset to Recommended Plan
Back
Confirm Schedule


Review & Complete Subscription
Lawn Mowing Service Subscription
Gold (0.25–0.5 acres)
Monthly
Visits 15
Subtotal $1,050.00
Discount applied $157.50
Coupon Discount $23.00
Service Tax (7%) $60.87
Yearly Total $930.37
Payable Monthly $77.54
✓ Account Authenticated
Authorized Contact: sri123@yopmail.com
Customer Name: Sri
Phone: (342) 465-7686
Your account is verified for service deployment. Review your order details, apply any promotional codes if needed, and click below to finalize service activation.
Have a coupon code?
DBS
Remove
Coupon applied! You save an extra $23.00
Cancel
Proceed to Checkout



Flow	Impact	Reason & Technical Verification
Quotation to Subscription Checkout `POST /subscribe` `POST /create-session`	**None**	Checkout session creation, customer assignment, Stripe price resolution, and checkout metadata remain completely untouched.
Initial Payment & Subscription Activation `checkout.session.completed` `invoice.paid` (Cycle 1)	**Improved**	Initial invoice creation and contract activation preserved. Initial quotation invoice (`I_6287`) now establishes an

Flow	Impact	Reason & Technical Verification
		authoritative baseline for Cycle 1 without risk of metadata collisions.
Date Calculations & Month-End Signups *(Issue 2 Fix)* `calculateNextBillingDate` `computeSubscriptionDetails`	**Fixed**	Month-end signups (Jan 31, Aug 31, Nov 30, leap-year Feb 29) across monthly, quarterly, and half-yearly frequencies clamp cleanly to month end instead of overflowing and skipping months.
Autojob Generation & Visit Scheduling *(Issue 3 Fix)* `resolveVisitScheduleDates` `adjustToBusinessDay`	**Fixed**	When candidate visit dates spill past the month-end or collide with non-working days (weekends, holidays, max capacity), the algorithm rolls backward to the nearest open business day within the same month.
Recurring Stripe Renewals & Webhooks `invoice.paid` (Cycle 2..N) `handleOnStoreInvoicePaid`	**Fixed**	**Eliminates duplicate payments:** Recurring billing creates exactly **one** dedicated cycle invoice document (`I_6288`) and no longer pushes a duplicate installment into the initial quotation invoice (`I_6287`). Webhook retries are idempotent.
Payment Failure & Dunning Grace Period `invoice.payment_failed` `payment_intent.payment_failed`	**None**	Dunning lifecycle, contract status transition to `past_due` , holding pending jobs, and sending customer failure notification emails remain fully intact.
Auto-Cancel Cron & Expiration Cleanup *(Issue 8 Fix)* `subscriptionAutoCancelCron.js`	**Fixed**	Broadened query from `{ $gte: start, $lte: end }` to `{ $lte: end }` . Overdue or expired contracts missed due to server downtime or restarts are caught, immediately marked `Completed` , cancelled in Stripe, and unscheduled jobs are purged.
Customer Cancellation & Settlement `POST /public-cancel/:id`	**None**	Prorated visit calculations, settlement charge/refund generation, waiver

Flow	Impact	Reason & Technical Verification
`POST /cancel-subscription-by-user/:id`		logic, and settlement invoices remain untouched.
POS Invoice Generation & PDF Resend `POST /invoicee/resendPDF/:id` `GET /invoicee/view-pdf-invoice/:id`	**Improved**	Each recurring cycle now maps to its own dedicated POS Invoice document with clean period dates (`periodStart` to `periodEnd`), allowing clear PDF downloads and email receipts per cycle.
Customer Portal & Subscriptions Detail UI `PaymentProgress.tsx` `SubscriberPayments.tsx`	**Fixed**	Installments and payments are deduplicated by `stripeInvoiceId` and cycle periods. The payments table displays exact cycle rows, Total Paid shows the true amount ($2.42), and the progress bar renders the exact number of checkmarks (2 of 4).

Flow-by-Flow Compatibility Verification

Flow	Impact	Reason
Plan & Variant Management (`/subscripition/create`, `/edit`, `/all-subscriptions`)	**None**	Core schema, pricing engine, and variant lookup queries remain completely untouched. Aggregation summaries now provide enriched tier counts without mutating plan structures.
Quotation to Subscription (`POST /createQuotationFromPlan`)	**Improved**	`effectiveVisitSchedule` now maps directly from `customVisitSchedule`, ensuring custom visit frequencies are preserved without affecting standard quotation generation.
Stripe Checkout & Pending Subscription (`checkout.session.completed`)	**Protected**	`PendingSubscription` resolves via metadata ID with session fallback.

Flow	Impact	Reason
		Resolves `customVisitSchedule` from multiple historical sources without breaking older single-visit plans.
Initial Contract Activation & Job Scheduling (`invoice.paid` - Initial)	**Protected**	Robust atomic lock (`initialJobsScheduled: true`) prevents duplicate job generation on webhook retries. Multi-field fallback matches old and new job schema definitions.
Recurring Billing Renewal Cycle (`invoice.paid` - Subsequent cycles)	**None**	Normal billing advancement, `billingCyclesCompleted` increment, and POS invoice creation run untouched. Non-overdue draft invoices are no longer misidentified as past due.
Payment Failure & Grace Period (`invoice.payment_failed`)	**Fixed**	Fixed retry date formatting (`formatNextRetry`) to eliminate `Invalid Date` crashes. Added deduplication keys (`attemptKey`) to prevent duplicate history records and redundant customer emails.
Webhook Failure Event Filtering (`charge.failed` & `payment_intent.payment_failed`)	**Fixed**	Bypasses non-invoice duplicate events when linked to an active Stripe invoice, eliminating 3x redundant alerts/SMS for a single failure.
Payment Recovery & Job Resumption (`attemptAutoCatchupPreviousInvoices`)	**Improved**	Verifies genuinely past-due invoices (`hasFailedAttempt || isOverdue ||`

Flow	Impact	Reason
		`isRetrying`). Automatically transitions jobs from `OnHold` back to active as soon as dues are resolved.
Scheduled Cancellation Flow (`cancelType: 'schedule'`)	**None**	Calculates dynamic notice buffer days from `nextBillingDate` or frequency. Prunes future jobs after cutoff while preserving service delivery during the notice window.
Immediate Cancellation & Financial Settlement (`cancelType: 'immediate'`)	**Improved**	Settlement calculation (*completed visits delivered vs. total amount paid*) accurately resolves both balance due and customer refunds. Cancels all remaining scheduled jobs immediately.
Settlement Invoice Dispatch (`sendSettlementInvoice`)	**Fixed**	Added automated idempotency and a 60-second cooldown window: prevents duplicate settlement invoice generation and stops repeated email delivery on multiple clicks or cron runs.
Settlement Invoice Payment (`checkout.session.completed` → `cancellation_settlement`)	**Protected**	Handles `type: 'cancellation_settlement'` directly: marks POS invoice as paid, sets contract status to `settled`, generates PDF receipt, and emails customer without triggering job re-creation.
Settlement Refund Processing (`processSettlementRefund`)	**None**	Direct Stripe refund (`stripe.refunds.create`)

Flow	Impact	Reason
		and manual/offline refund reconciliation endpoints remain intact with complete audit trail in contract history.
Settlement Balance Waiver (`waiveSettlementBalance`)	**None**	Admin waiver logic, balance clearing, and voiding of outstanding settlement invoices operate untouched.
External Stripe Cancellation (`customer.subscription.deleted`)	**Protected**	Handles external cancellations from Stripe Dashboard. Features customer ID fallback for legacy contracts missing `stripeSubscriptionId`, auto-cancelling remaining jobs.
Daily Auto-Cancel Cron (`subscriptionAutoCancelCron.js`)	**Fixed**	Reached cutoffs are finalized cleanly. Added guard checking `settlementStatus !== "invoice_sent"` and `!invoiceSentAt` so cron runs do not re-send invoices every night.
Admin Manual Reactivation (`POST /reactivate-subscription/:id`)	**None**	Re-links billing, restores contract to `active`, and updates Stripe metadata safely without disrupting existing job schedules.
Subscriber Listing & Details View (`GET /subscribers`, `GET /get-service-subscriber/:id`)	**Improved**	Safely checks both legacy and new structures (`client` vs `clientId`, string IDs vs ObjectIds). Subscriber cards render full status badges without frontend crashes.

🔴 Issue 1 — `past_due` Set During Successful Payment

Steps:

1. Go to **Stripe Dashboard → Customers**
2. Pick a test customer with an active service subscription
3. Go to their **Invoices** → find the upcoming scheduled invoice (status: `open`)
4. Do **NOT** pay it yet — just note it exists
5. Now trigger a **new payment** from Stripe Dashboard → click **"Pay invoice"** on any other open invoice for same subscription
6. Go to your **POS Admin → Subscription Contracts**
7. ✅ **Expected:** Status should be `active`
8. ❌ **Bug:** Status shows `past_due` even though payment succeeded

---

🔴 Issue 2 — Job Restore Out of Sync After Catch-up

Steps:

1. Find a test subscription that is currently in `past_due` with jobs in `OnHold`
2. Go to **Stripe Dashboard → Customers → [Customer] → Invoices**
3. Find the **open/failed invoice** → click **"Pay invoice"** manually (use test card `4242 4242 4242 4242`)
4. Wait ~10–15 seconds for webhook to fire
5. Go to **POS Admin → Jobs** for that subscription
6. ✅ **Expected:** All `OnHold` jobs restored + contract status = `active`
7. ❌ **Bug:** Jobs may be restored but contract still shows `past_due`, OR contract is `active` but some jobs still `OnHold`

🔴 Issue 3 — Wrong Contract Gets `past_due` (Multiple Subscriptions)

Steps:

1. Create a **test customer** in Stripe with **2 different service subscription contracts** in your POS
2. Go to **Stripe Dashboard → Subscriptions**
3. Find **Subscription 1** → click the 3 dots → **"Simulate payment failure"** (test mode)
    - OR use Stripe test card: `4000 0000 0000 0341` (card that always declines)
4. Wait for webhook `invoice.payment_failed` to fire
5. Go to **POS Admin → Subscription Contracts**
6. Check **BOTH** contracts for that customer
7. ✅ **Expected:** Only Subscription 1's contract shows `past_due`
8. ❌ **Bug:** Subscription 2's contract (the wrong one) also shows `past_due`

🔴 Issue 4 — Duplicate History on Repeated Webhooks

Steps:

1. Go to **Stripe Dashboard → Developers → Webhooks**
2. Find a recent `invoice.payment_failed` event
3. Click on the event → click **"Resend"** button (top right)
4. Wait 5 seconds
5. Go to **POS Admin → Subscription Contract → History tab**
6. ✅ **Expected:** Only **1 entry** for `payment_failed`
7. ❌ **Bug:** You see **2 or more duplicate** `payment_failed` history entries

🟡 Issue 5 — Final Warning Never Fires (If Stripe Retries ≤ 2)

Steps:

1. Go to **Stripe Dashboard → Settings → Billing → Subscriptions & Emails**
2. Check **"Retry failed payments"** — note how many retries are configured
3. If retries = **1 or 2**, simulate a payment failure on a test subscription
4. Let Stripe exhaust all retries
5. Check your POS → contract history
6. ✅ **Expected:** Should see a **"Final Warning"** email/history entry before final attempt
7. ❌ **Bug:** No final warning — goes straight from `payment_failed` to `past_due`

🟡 Issue 6 — Stripe Metadata Not Updated

Steps:

1. Simulate a payment failure on a test subscription (use decline card `4000 0000 0000 0002`)
2. Wait for `invoice.payment_failed` webhook
3. Confirm POS contract status = `past_due`
4. Go to **Stripe Dashboard → Subscriptions → [that subscription] → Metadata**
5. ✅ **Expected:** Metadata should show something like `internalStatus: past_due`
6. ❌ **Bug:** Metadata shows nothing / still shows `active` — no sync happened

---

🟡 Issue 7 — Email History Silent Skip

Steps:

1. Temporarily slow down webhook processing (or this is hard to reproduce in real-time)
2. **Simpler way:** Go to **Stripe Dashboard → Developers → Webhooks → [your endpoint]**
3. Send a **test** `invoice.payment_failed` **event**
4. Immediately go to your server logs
5. Look for: `[invoicePaymentFailed] Failed to append email status to history note`
6. Then check POS contract history
7. ✅ **Expected:** History entry shows `"Email sent to customer@example.com"`
8. ❌ **Bug:** History entry has NO mention of email sent

🛠️ Quick Stripe Test Cards

Card Number	Behavior
`4242 4242 4242 4242`	Always succeeds
`4000 0000 0000 0341`	Always fails (card declined)
`4000 0000 0000 0002`	Always declined
`4000 0000 0000 9995`	Insufficient funds

> Use these in **Stripe Test Mode** only. Make sure your `.env` has `STRIPE_SECRET_KEY` set to your **test key** (`sk_test_...`).

System Status Overview for All 7 Issues

Issue	Title	Status in Codebase	Implementation Details
1	`past_due` Set During Successful Payment	Resolved	In `JS invoicePaid.js`, scheduled upcoming open invoices (`attempt_count === 0`, due date in future) are ignored. Only *genuinely failed/overdue* invoices trigger `past_due`.
2	Job Restore Out of Sync After Catch-up	Resolved	In `JS invoicePaid.js`, once all past dues are paid, `JS restoreOnHoldJobsForSubscription` executes synchronously and contract status resets to `active`.
3	Wrong Contract Gets `past_due`	Resolved	In `JS invoicePaymentFailed.js`, resolution strictly matches by `stripeSubscriptionId` (extracted directly from `stripeInvoice.subscription`). It never blindly falls back to customer ID when multiple contracts exist.
4	Duplicate History on Repeated Webhooks	Resolved	In `JS invoicePaymentFailed.js`, an idempotency guard checks `attemptKey` (`${invoiceId}_${attemptCount}`) and skips duplicate history entries on webhook retries.
5	Final Warning Never Fires (Retries ≤ 2)	Resolved	In `JS invoicePaymentFailed.js`, `maxRetries` is configurable via `STRIPE_MAX_PAYMENT_RETRIES` in `.env` (defaults to 3, but adapts if 1 or 2 is configured).
6	Stripe Metadata Not Updated	Resolved	In `JS invoicePaymentFailed.js`, Stripe subscription metadata is updated via `stripe.subscriptions.update(stripeSubscriptionId, { metadata: { internalStatus: 'past_due' } })`. When recovered, `JS invoicePaid.js` sets it back to `active`.
7	Email History Silent Skip	Resolved	In `JS invoicePaymentFailed.js`, email delivery status is concatenated directly into `finalHistoryNote` and saved atomically into the contract's history.

3. Automated Test Suite Results

text

```
======================================================================
   RUNNING ALL TEST CASES FOR FIXES 2, 3, AND 8 (WITH EDGE CASES)
======================================================================

SUITE 1: JavaScript setMonth Rollover & Month-End Signups
  [PASS] 2.1 Monthly: Aug 31 -> Sep 30 (30-day month clamp)
  [PASS] 2.2 Monthly: Jan 31 -> Feb 28 (28-day non-leap February clamp)
  [PASS] 2.3 Monthly: Jan 31 2024 -> Feb 29 2024 (29-day leap February clamp)
  [PASS] 2.4 Monthly: May 31 -> Jun 30 (30-day month clamp)
  [PASS] 2.5 Quarterly: Nov 30 -> Feb 28 (clamps into February without skipping)
  [PASS] 2.6 Half-Yearly: Aug 31 -> Feb 28 (clamps across 6 months into Feb)
  [PASS] 2.7 Yearly: Feb 29 2024 -> Feb 28 2025 (Non-leap year clamp)
  [PASS] 2.8 Weekly: 7 days exactly regardless of month end
  [PASS] 2.9 Full 12-month job creation sequence for Jan 31 signup

SUITE 2: Month-End Boundary & Weekend Collision Rollback
  [PASS] 3.1 Month end on Saturday with workOnSaturday=false rolls back to Friday
  [PASS] 3.2 Month end on Saturday with workOnSaturday=TRUE stays on Saturday
  [PASS] 3.3 Month end on Sunday with closed weekends rolls back to Friday
  [PASS] 3.4 Month end on Sunday with workOnSunday=TRUE stays on Sunday
  [PASS] 3.5 Rollback past Saturday AND a Friday Company Holiday lands on Thursday
  [PASS] 3.6 Rollback past Saturday AND Friday (when Friday full) lands on Thursday

SUITE 3: Auto-Cancel Cron Query Coverage & Lifecycle Routing
  [PASS] 8.1 Query filter { $lte: end } captures past-due, expired, and tomorrow contracts
  [PASS] 8.2 Past customPlanEndDate triggers immediate cancellation & Completed status
  [PASS] 8.3 Tomorrow customPlanEndDate triggers Stripe cancel_at schedule cleanly

SUITE 4: Webhook Deduplication & Anti-Spam
  [PASS] 4.1 Backend Guard: isAutoRecurringPayment skips pushing installment into quote invoice
  [PASS] 4.2 Backend Guard: Re-delivering exact same Stripe invoice ID prevented
  [PASS] 4.3 Frontend Deduplication: Clean presentation of multi-cycle payments

======================================================================
FINAL RESULT: 21 / 21 TESTS PASSED (100% PASS RATE)
SYNTAX CHECKS: 0 ERRORS ACROSS ALL MODIFIED FILES
======================================================================
```