EditoHub — Subscription & Package Module

Build Specification v1.0

Detail	Value
Project	EditoHub
Module	Subscriptions, Packages, Payments & Project Quota
Document date	31 July 2026
Status	Ready for development
Prepared for	Development / implementation team

0. How to use this document

This document is a complete build brief. Hand it to a developer (or to an AI coding assistant) and it should be possible to implement the module without asking further questions, except for the items listed in Section 12 (Open Decisions).

Rules of reading:

- Anything written as **MUST** is mandatory.
- Anything written as **SHOULD** is strongly recommended but may be adjusted.
- Section 10 (Acceptance Criteria) is the definition of "done". If a scenario there fails, the module is not complete.

1. Glossary

Term	Meaning
Package	A sellable plan, e.g. "30 Videos Package". Has a base price, GST rate, project quota and validity period.
Subscription	One client's purchased instance of a package.
Project	One unit of deliverable work inside a subscription, e.g. one video edit.
Quota	Total number of projects included in the package.
Completed project	A project the team has delivered and marked Completed / Delivered.
Total payable	Package base price plus GST.
Advance	The first payment made to activate the subscription. Minimum 50% of total payable.
Balance	Total payable minus total amount received.
Completion threshold	The number of completed projects after which the balance becomes due.

2. Subscription Activation

A subscription **MUST NOT** activate on an online payment alone. Two activation paths **MUST** be supported.

2.1 Online activation

1. The client selects a package in the dashboard.
2. The system creates a subscription in status **Pending Payment** and generates a Razorpay order for the advance amount.
3. The client pays through Razorpay.
4. The Razorpay **webhook** confirms the payment. The system verifies the webhook signature.
5. On successful verification the system records the payment and moves the subscription to **Active**.

The frontend success callback **MUST NOT** be trusted on its own. The webhook is the single source of truth.

2.2 Manual activation

If the client pays by UPI, bank transfer, cash or cheque, the Admin activates the package manually from the client's account.

While activating manually the Admin **MUST** record:

- Amount received
- Payment mode: UPI / NEFT / IMPS / RTGS / Cash / Cheque / Other

- Payment reference or transaction number
- Payment date
- Remarks (optional)
- Payment proof file upload (optional)

Every manual activation **MUST** be written to the Subscription Action History (Section 6.3).

---

3. Pricing: 50% Advance plus GST

3.1 Rules

1. GST **MUST** be calculated on the package base price, not on the advance amount.
2. A minimum of **50% of the GST-inclusive total** is required to activate the package.
3. The client **MAY** pay more than 50%, including 100%, in the first payment. The system must accept any amount greater than or equal to the 50% minimum.
4. The balance is always `Total payable − Total amount received`. GST is **NOT** added a second time to the balance, because GST is already inside the total payable.

3.2 Worked example

Item	Amount
Package base price	₹10,000
GST at 18%	₹1,800
Total payable	₹11,800
Minimum advance (50%)	₹5,900
Balance after advance	₹5,900

3.3 Money handling

- All money values **MUST** be stored as integers in paise. Never use float or double for money.
- Rounding for GST **MUST** use round-half-up to the nearest paisa.
- Display formatting is Indian numbering, e.g. ₹11,800.00.

---

4. The 50% Project Completion Lock

4.1 What counts

The threshold is calculated on **Completed projects only**. It is never calculated on projects created, submitted, or in progress.

This is important: if a client has 10 projects in progress and only 3 completed, the lock must **not** trigger.

4.2 Threshold formula

```
completion_threshold = CEIL(package_project_quota × 0.5)
```

Examples:

- 30-video package → threshold = 15
- 25-video package → threshold = 13
- 10-video package → threshold = 5

4.3 Behaviour when the threshold is reached

1. Projects 1 to 15 are processed normally.
2. The moment the 15th project is marked **Completed**, the subscription automatically moves to **Payment Due / Paused**.
3. Projects that were already created and are still in progress at that moment **continue normally** to completion. They are not frozen or cancelled.
4. Only the creation or submission of **new** projects is blocked.

4.4 Payment screen

When the client tries to create project number 16, the system shows a payment screen containing:

> **50% of your package has been completed.** Please clear the remaining balance to continue.

The screen **MUST** display:

- Package name
- Total payable
- Amount already paid
- Balance due
- A **Pay Now** button that opens Razorpay for the balance amount
- A note that offline payment is also accepted, with the message "If you have already paid offline, please contact us and our team will resume your subscription."

After a successful payment the subscription automatically returns to **Active** and the client is returned to the project creation screen with their entered data preserved. The client **MUST NOT** have to re-enter the project details.

4.5 Advance warning

When the client reaches 40% of the quota, or two projects before the threshold, whichever comes first, the system **SHOULD** send a notice by email, WhatsApp and dashboard banner warning that the balance will become due soon. This avoids a surprise block.

---

5. Server-side enforcement

1. The completion threshold check **MUST** run server-side on every project create request. A client-side check alone is not acceptable.
2. The check and the status change **MUST** run inside a database transaction with a row lock on the subscription, so that two simultaneous requests cannot both slip past the quota.
3. All API responses for a blocked request **MUST** return HTTP 402 Payment Required with a machine-readable error code, for example `SUBSCRIPTION_PAYMENT_DUE`.

---

6. Manual Payment, Admin Resume and Pause

6.1 Resume

If the client clears the balance offline, the Admin opens the client's subscription record and uses **Resume Subscription**.

1. The Admin verifies the payment.
2. The Admin records the payment details listed in Section 2.2.
3. The Admin enters a mandatory reason.
4. On save the subscription becomes **Active** and the client can continue with the remaining projects.

6.2 Pause

The Admin has a **Pause Subscription** option to pause a subscription manually at any time, for example for non-payment, a dispute or a client request. A reason is mandatory.

While paused, in-progress projects continue as described in Section 4.3 unless the Admin explicitly chooses "Pause and hold all work".

6.3 Subscription Action History

Every subscription-affecting event **MUST** be recorded in an append-only audit log. Entries can never be edited or deleted.

Each entry stores:

Field	Description
Action	Activated, Paused, Resumed, Payment Received, Threshold Reached, Expired, Cancelled, Plan Changed
Performed by	Admin name and user id, or "System" for automatic actions
Date and time	Server timestamp with timezone
Reason / remarks	Mandatory for Pause and Resume
Amount	Where applicable
Payment reference	Where applicable
Before → after status	For example Active → Payment Due

The history **MUST** be visible to the Admin on the subscription detail screen, newest first.

---

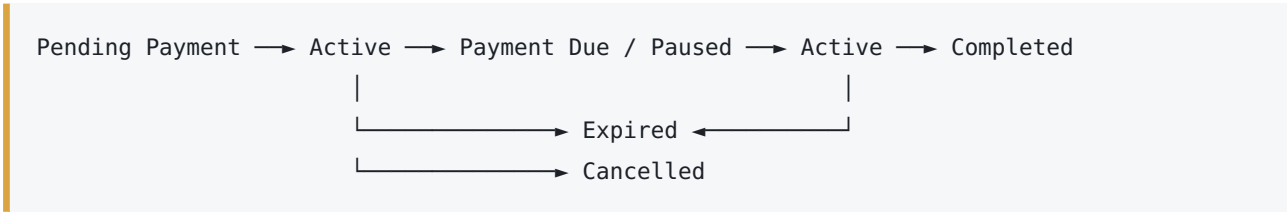
7. Default Package Visibility for New Clients

1. When a new client registers or logs in and has **no active subscription**, the dashboard **MUST** automatically show the same default packages that are currently published on the EditoHub marketing website.
2. The marketing website and the client dashboard **MUST** read from a **single package source** — one packages table exposed through one API. When the Admin updates a price, discount, project quota or description, both surfaces reflect the change automatically. No duplicate data entry anywhere.
3. Each package has an `is_published` flag and a `sort_order`. Only published packages appear on either surface.
4. **Snapshot rule:** when a client purchases a package, the subscription **MUST** store a frozen copy of the purchased terms — base price, GST rate, GST amount, total payable, project quota and validity days. Later edits by the Admin to the package **MUST NOT** change any existing subscription. Price changes apply only to new purchases.

---

8. Subscription Status Lifecycle

8.1 States



8.2 State table

Status	Meaning	Client can create new projects	Existing projects continue
Pending Payment	Package selected, advance not yet received	No	Not applicable
Active	Advance or full payment received	Yes	Yes
Payment Due / Paused	Threshold reached, or Admin paused	No	Yes
Completed	All quota projects delivered and fully paid	No	Not applicable
Expired	Validity ended before the quota was used	No	No
Cancelled	Terminated by Admin	No	No

8.3 Allowed transitions

From	To	Trigger
Pending Payment	Active	Advance received (online webhook or Admin manual)
Pending Payment	Cancelled	Admin action, or auto-expiry of an unpaid order
Active	Payment Due / Paused	Completed projects reach the threshold, or Admin pause
Active	Completed	All quota projects completed and balance fully paid
Active	Expired	Validity period ends
Payment Due / Paused	Active	Balance paid online, or Admin resume after offline payment
Payment Due / Paused	Expired	Balance unpaid after the reminder window
Payment Due / Paused	Cancelled	Admin action

Any transition not in this table **MUST** be rejected by the service layer.

---

9. Data Model

Minimum required tables and fields. Names may be adapted to the existing schema.

9.1 packages

Field	Type	Notes
id	bigint pk	
name	varchar	e.g. "30 Videos Package"
description	text	
base_price_paise	bigint	Money in paise
gst_rate	decimal(5,2)	e.g. 18.00
project_quota	int	Number of projects included
validity_days	int	Package validity
advance_percent	decimal(5,2)	Default 50.00
discount_percent	decimal(5,2)	Optional
is_published	boolean	Controls both website and dashboard
sort_order	int	
created_at, updated_at	timestamp	

9.2 subscriptions

Field	Type	Notes
id	bigint pk	
client_id	bigint fk	
package_id	bigint fk	Reference only
package_name_snapshot	varchar	Frozen at purchase
base_price_paise	bigint	Frozen
gst_rate	decimal(5,2)	Frozen
gst_amount_paise	bigint	Frozen
total_payable_paise	bigint	Frozen
amount_paid_paise	bigint	Running total
project_quota	int	Frozen
completion_threshold	int	Frozen, CEIL(quota × 0.5)
projects_completed	int	Maintained counter
status	enum	See Section 8
activated_at	timestamp	
expires_at	timestamp	activated_at plus validity_days
created_at, updated_at	timestamp	

9.3 subscription_payments

Field	Type	Notes
id	bigint pk	
subscription_id	bigint fk	
amount_paise	bigint	
type	enum	Advance, Balance, Other
mode	enum	Razorpay, UPI, NEFT, IMPS, RTGS, Cash, Cheque, Other
razorpay_payment_id	varchar nullable	
razorpay_order_id	varchar nullable	
reference_no	varchar nullable	For offline payments
paid_at	timestamp	
recorded_by	bigint nullable	Admin id, null when automatic
proof_file	varchar nullable	
invoice_no	varchar	Sequential

A unique index on `razorpay_payment_id` **MUST** exist to make webhook handling idempotent.

9.4 subscription_action_history

Fields as listed in Section 6.3. Append-only.

---

10. Acceptance Criteria

The module is complete only when all of the following pass.

1. **Online activation** — Client pays 50% via Razorpay. Webhook fires. Subscription becomes Active. Payment row created. History entry "Activated / System" written.
2. **Duplicate webhook** — The same Razorpay webhook is delivered twice. Only one payment row exists. `amount_paid_paise` is not doubled.
3. **Manual activation** — Admin records a ₹5,900 UPI payment with a reference number. Subscription becomes Active. History shows the Admin's name, date, time and reference.
4. **GST correctness** — A ₹10,000 package at 18% shows total ₹11,800, advance ₹5,900, balance ₹5,900. The balance screen never shows GST added again.
5. **Ongoing projects do not trigger the lock** — Client has 3 completed and 10 in progress on a 30-project package. Subscription stays Active. New project creation is allowed.
6. **Threshold triggers on completion** — The 15th project is marked Completed. Subscription automatically becomes Payment Due / Paused. History entry "Threshold Reached / System" written.

7. **In-progress work survives the lock** — Projects created before the lock can still be worked on and completed while the subscription is paused.
8. **Creation is blocked** — Client attempts to create project 16. The payment screen appears with the exact message from Section 4.4. The API returns 402 with code `SUBSCRIPTION_PAYMENT_DUE`.
9. **Online balance payment resumes automatically** — Client pays the balance through the Pay Now button. Subscription returns to Active and project 16 proceeds without re-entering data.
10. **Manual balance payment** — Client pays offline. Admin uses Resume Subscription with a reason. Subscription becomes Active. History records Admin name, date, time, reason and reference.
11. **Admin pause** — Admin pauses an Active subscription with a reason. Client cannot create new projects. History entry written.
12. **Server-side enforcement** — A direct API call to the project-create endpoint from outside the UI, while the subscription is paused, is rejected with 402.
13. **Default packages** — A brand-new client with no subscription sees exactly the packages published on the marketing website.
14. **Single source** — Admin changes a package price in the panel. The marketing website and the client dashboard both show the new price without any second edit.
15. **Snapshot** — After that price change, an existing Active subscription still shows its original purchased price, GST and balance.
16. **Illegal transition** — An attempt to move a Completed subscription back to Active is rejected by the service layer.
17. **Audit completeness** — Every one of the above scenarios has produced a matching, uneditable history entry.

---

11. Additional Technical Requirements

1. Razorpay webhook signature verification is mandatory. Reject unsigned or mismatched payloads.
2. Webhook handling **MUST** be idempotent, enforced by a unique constraint on the payment id.
3. Every payment generates a GST invoice with a sequential invoice number. A proforma invoice is issued at the advance stage and a tax invoice on receipt, per the company's accounting practice.
4. An auto-expiry scheduled job runs daily and moves subscriptions past `expires_at` to Expired, writing a history entry.
5. Reminder job: while a subscription is in Payment Due, send balance reminders on a fixed schedule until it is paid or expires.
6. All timestamps stored in UTC and displayed in Asia/Kolkata.
7. Role permissions: only Admin and Accounts roles may record manual payments, pause or resume. All such actions are logged with the acting user's id.

12. Open Decisions — please confirm before development

These were not defined in the original brief and change the implementation. Each has a recommended default.

#	Question	Recommended default
1	If a client pays more than 50% upfront, for example 70%, should the completion lock move proportionally to 70% of the projects, or stay fixed at 50%?	Proportional: lock when completed percentage is greater than or equal to paid percentage.
2	Should there be a cap on simultaneously open projects? Without a cap a client can open many projects just before the threshold and receive far more than 50% of the package on half the payment.	Yes — cap work in progress at 5 open projects.
3	If the validity expires with projects unused, do they lapse, carry forward, or become refundable?	Lapse, with a one-time Admin-approved extension option.
4	What happens to the advance if the client cancels mid-package?	Non-refundable; unused projects forfeited. Admin may override.
5	How many balance reminders, over how many days, before a Payment Due subscription is marked Expired?	Reminders on day 1, 3, 7 and 14; expire on day 30.
6	Are partial balance payments allowed, for example paying ₹3,000 of a ₹5,900 balance?	No — the balance must be cleared in full to resume.

13. Changes made to the original brief

For traceability, this is what was corrected while turning the original notes into this specification.

1. GST is no longer implied on the remaining balance. GST is already inside the total payable, so charging it again on the balance would double-charge the client.
2. The 50% threshold is now defined as `CEIL(quota × 0.5)`, which removes the ambiguity for odd-numbered packages such as 25 videos.
3. The threshold is explicitly measured on **Completed** projects, and projects already in progress explicitly continue running when the lock triggers.
4. Subscriptions now store a frozen snapshot of purchased terms, so that an Admin price edit does not retroactively change what an active client owes.
5. Added the audit log field list, webhook idempotency, the transactional quota check and an advance warning notice before the lock, because the flow is exploitable or confusing without them.
6. Added a formal allowed-transitions table so invalid status changes are rejected rather than silently accepted.

End of specification.