HMS.iba.edu.pk

Hostel Management System

WORKFLOW DOCUMENTATION — Admin Panel: Modules A - D

Prepared By	Sagar Lekhraj (Group Leader), Sudharth Kumar
Module Scope	Admin Panel - Modules A (Students), B (Rooms), C (Mess), D (Complaints)
Program	BSCS
Course	Web-Based Application Development
Version	1.0 — Final
Date	February 2026

1. Document Overview

This document defines the complete Admin Panel workflow for HMS.iba.edu.pk. It covers four operational domains that give administrators full control over the hostel management system:

Module	Domain	Admin Responsibilities
A	**Student Management**	View students, view detail profiles, manage room allocation.
B	**Room Management**	Create room inventory, allocate rooms to students, deallocate.
C	**Mess Subscription**	View all subscriptions, mark payment status (Paid/Unpaid).
D	**Complaint Management**	Review complaints, update status, and add remarks. Status must follow defined transitions.

The admin panel operates as the operational engine behind the student-facing HMS portal. Every student action — applying for a room, subscribing to mess, raising a complaint — generates a task that the admin must act on through this panel. This document defines each workflow, data schema, business rule, and permission boundary precisely.

Admin Authentication & Dashboard

2. Admin Authentication

Admin authentication follows the same session-based mechanism as student login, but with a different role check and redirect target.

Stage 0 **Login Flow**
Admin submits credentials. System validates and redirects to Admin Dashboard.

Step 1: Submit Credentials

- Admin navigates to hms.iba.edu.pk/admin/login (separate from student login page).
- Enters Email and Password.
- System queries users table: SELECT * FROM users WHERE email = [input].
- If not found OR bcrypt.compare(password, hash) fails → show generic error: "Invalid credentials".

Step 2: Role Check

- If credentials are valid, system checks session: role = "admin".
- If role = "student" → deny access, show 403 error. Admin credentials only.
- If role = "admin" → proceed.

Step 3: Create Session & Redirect

- Session stores: user_id, role = "admin", admin_name.
- Redirect to /admin/dashboard.
- If admin is already logged in and visits /admin/login → redirect to /admin/dashboard automatically.

Route Protection: ALL admin routes (/admin/*) require a valid session with role = "admin". Middleware runs on every request. Students cannot access admin routes — any attempt returns 403 Forbidden. Admin cannot access student routes (/dashboard, /complaints/new, etc.).

3. Admin Dashboard — Control Center

When the admin logs in, the dashboard loads seven live data cards drawn from the database. This gives immediate operational visibility without navigating into individual modules.

#	Card	DB Query	Purpose
1	**Total Students**	`SELECT COUNT(*) FROM students`	Total registered student accounts.

#	Card	DB Query	Purpose
2	**Allocated Rooms**	`SELECT COUNT(*) FROM students WHERE room_number IS NOT NULL`	How many students currently have a room.
3	**Available Rooms**	`SELECT COUNT(*) FROM rooms WHERE status = 'available' AND occupied_count < capacity`	Rooms with open capacity right now.
4	**Pending Applications**	`SELECT COUNT(*) FROM room_applications WHERE status = 'Pending'`	Applications awaiting admin action — most actionable number.
5	**Active Complaints**	`SELECT COUNT(*) FROM complaints WHERE status IN ('Pending','In Progress')`	Open complaints requiring admin attention.
6	**Active Mess Subscriptions**	`SELECT COUNT(*) FROM subscriptions WHERE status = 'Active'`	Currently running meal plan subscriptions.
7	**Unpaid Subscriptions**	`SELECT COUNT(*) FROM subscriptions WHERE payment_status = 'Unpaid' AND status = 'Active'`	Students with outstanding mess dues.

Admin sidebar navigation provides persistent access to all modules:

Sidebar Item	Route
Dashboard	`/admin/dashboard`
Students	`/admin/students`
Rooms	`/admin/rooms`
Mess Management	`/admin/mess`
Complaints	`/admin/complaints`
Logout	`POST /admin/logout`

Module A — Student Management

4. Module A: Student Management Workflow

Admin manages the student roster — viewing all registered students and drilling into individual student profiles. Room allocation is performed from here and from the Rooms module.

Stage A-1 | **View All Students**
Admin sees a full table of every registered student with key status indicators.

The students list is the admin's primary working surface. It surfaces all students with their current room and account status.

Name	Student ID	Email	Room	Status	Actions
Sagar Lekhraj	29325	s.lekhraj@...	B-204	Active	View
Sudharth Kumar	26925	s.kumar@...	—	Active	View
Ahmed Raza	28901	a.raza@...	A-101	Active	View

- Filter and search: by name, Student ID, room status (Allocated / Not Allocated).
- Table is paginated for large cohorts.
- "View" opens the Student Detail page (Stage A-2).
- Account deactivation is NOT included in MVP — requires a defined cascade plan for room, mess, and complaints.

Stage A-2 | **Student Detail View**
Admin sees the complete profile of one student from an admin perspective.

The Student Detail page is the admin's full view of one student's HMS record. It aggregates data from multiple tables.

Section	Data Shown
Personal Info	Full Name, ERP / Student ID, Email, Program, Batch, Account created date.
Room Status	Current room number and block (or 'Not Allocated'). Full application history: App ID, term, status, allocated room, date applied.
Mess Subscription	All subscriptions: Order #, meal types, dates, total, payment status. Current active subscription highlighted.
Complaint History	All complaints by this student: ID, category, date, status, admin remark. Sorted newest first.

Section	Data Shown
Admin Actions	Allocate Room button (if no current room for active term). Deallocate Room button (if room currently allocated, admin-only).

Design Note: The Student Detail page is read-only except for the Allocate / Deallocate Room actions. Admin cannot edit student personal information or passwords from this view.

Module B — Room Management & Allocation

5. Module B: Room Management & Allocation Workflow

Room Management is the most operationally critical module. It controls the physical hostel inventory and the assignment of students to specific rooms. All allocation logic is enforced at the backend — the UI cannot bypass capacity or duplicate-assignment rules.

5.1 The rooms Table — Schema

A rooms table must be created and seeded before any allocation can occur. MVP can use static pre-seeded data.

Column	Type	Required	Description
`id`	`INT`	Auto	Primary key, auto-generated.
`block`	`VARCHAR`	Yes	Hostel block identifier. e.g. 'A', 'B', 'C'.
`room_number`	`VARCHAR`	Yes	Room label. e.g. '101', '204'. Unique within block.
`capacity`	`INT`	Yes	Maximum number of students this room can hold.
`occupied_count`	`INT`	Auto	Current number of students assigned. Starts at 0.
`status`	`ENUM`	Auto	Values: 'available', 'full', 'maintenance'. Derived from occupied_count vs capacity, or set manually.
`term`	`VARCHAR`	Yes	Hostel term this room applies to. e.g. 'Spring 2026'. Allows same room across terms.

Stage B-1 **Room Inventory View**
Admin sees all rooms with capacity and occupancy status.

The room inventory table gives the admin a full overview of the hostel estate:

Block	Room No.	Capacity	Occupied	Status	Actions
A	`101`	2	1	Available	View
A	`102`	3	3	Full	View
B	`204`	2	2	Full	View
C	`301`	2	0	Available	View

- Admin can also add a new room from this view (Add Room form).
- Admin can set a room's status to 'maintenance' manually, which removes it from the allocation pool.

Stage B-2 Allocate Room to Student

Admin assigns an available room to a student who has a pending application.

Allocation can be initiated from two entry points: the Student Detail page, or directly from a pending room application in the Applications queue.

Step 1: Admin selects a student with a Pending application

- Admin navigates to Students → selects a student → clicks 'Allocate Room'.
- OR: Admin navigates to Room Applications queue → finds a Pending application → clicks 'Allocate'.

Step 2: System shows available rooms

- Query: SELECT * FROM rooms WHERE status = 'available' AND occupied_count < capacity AND term = [application.term].
- Only rooms with open capacity and matching term are shown.
- Rooms with status = 'full' or status = 'maintenance' are excluded from the list.

Step 3: Admin selects a room and confirms

- Admin picks a room from the available list and clicks 'Confirm Allocation'.
- System performs TWO backend checks before writing (even if UI looked valid):
- Check A: occupied_count < capacity for the selected room.
- Check B: student does not already have a room_number for this term.
- If either check fails → reject with error. No partial writes.

Step 4: System writes allocation

- UPDATE students SET room_number = [room], hostel_block = [block] WHERE user_id = [student_id].
- UPDATE rooms SET occupied_count = occupied_count + 1 WHERE id = [room_id].
- If occupied_count now = capacity → UPDATE rooms SET status = 'full'.
- UPDATE room_applications SET status = 'Allocated', allocated_room = [room] WHERE id = [app_id].
- Student sees updated room on next dashboard load — no notification system needed in MVP.

Critical Rule: Both checks (capacity and one-room-per-student-per-term) MUST be enforced at the backend, not just in the UI. If a user bypasses the UI and hits the API directly, the backend must still block the allocation.

Stage B-3 Deallocate Room

Admin removes a student from their current room. Requires a reason.

Deallocation is an admin-only action available from the Student Detail page. It is intentionally friction-heavy to prevent accidents.

Step 1: Admin initiates deallocation

- Admin opens Student Detail page → clicks 'Deallocate Room'.
- System shows a confirmation modal with a mandatory reason field (text input).
- Admin must enter a reason before confirming (e.g. 'Student withdrew from hostel').
- Blank reason field = form will not submit.

Step 2: System writes deallocation

- UPDATE students SET room_number = NULL, hostel_block = NULL WHERE user_id = [student_id].
- UPDATE rooms SET occupied_count = occupied_count - 1 WHERE id = [room_id].
- If room was 'full' → UPDATE rooms SET status = 'available'.
- Log the deallocation with reason and admin_id for audit (store in a deallocations log table or as a note on the room_applications record).

MVP Note: Deallocation does NOT automatically cancel the student's mess subscription. Admin must handle mess separately if needed. This keeps the systems decoupled and avoids cascade errors.

Module C — Mess Subscription Management

6. Module C: Mess Subscription Management Workflow

In MVP, students subscribe to mess plans directly and their subscription is auto-activated. The admin's role in this module is purely operational: viewing all subscriptions and managing payment status. No approval step is required.

Design Decision: Approval workflow for mess subscriptions has been deliberately excluded from MVP. Auto-activation on student submission reduces friction and eliminates admin bottlenecks. If a student has a billing dispute, admin can update payment status manually.

Stage C-1 — View All Subscriptions

Admin sees all mess subscriptions across all students, filterable by status and payment state.

Student	Meal Plans	Duration	Total (PKR)	Status	Payment
Sagar Lekhraj	Sehri + Dinner	18 Feb–10 Mar	11,970	Active	Unpaid
Ahmed Raza	Dinner only	20 Feb–15 Mar	5,280	Active	Paid
Sara Khan	Iftar + Dinner	18 Feb–01 Mar	4,760	Expired	Paid

- Filter options: by Subscription Status (Active / Expired / Cancelled) and Payment Status (Paid / Unpaid).
- Table shows Total Amount (PKR) — admin needs to know the outstanding amount, not just who has not paid.
- Admin can click any row to see full subscription details.

Stage C-2 — Mark Payment Status

Admin manually marks a subscription as Paid or Unpaid. No payment gateway in MVP.

When a student pays (e.g. cash, bank transfer), the admin manually updates the payment status in the system:

Step 1: Admin finds the subscription
- Admin filters the subscriptions table by Payment Status = 'Unpaid'.
- Identifies the relevant student and order.

Step 2: Admin updates payment status
- Admin clicks 'Mark as Paid' on the subscription record.
- System shows a confirmation prompt (to prevent accidental clicks).

- On confirm: UPDATE subscriptions SET payment_status = 'Paid' WHERE id = [order_id].
- The change is reflected immediately in the subscriptions table and on the admin dashboard 'Unpaid' count.

6.1 Subscriptions Table — payment_status Field

The existing subscriptions table from Module 2 needs one additional field for admin management:

Column	Type	Required	Description
`order_id`	`UUID`	Auto	Unique order number, auto-generated.
`student_id`	`INT / FK`	Auto	FK from students table via session.
`meal_types`	`VARCHAR[]`	Yes	Array: Sehri, Iftar, Dinner.
`total_amount`	`DECIMAL`	Auto	sum(rates/day) x total_days.
`start_date`	`DATE`	Yes	Student-selected subscription start.
`end_date`	`DATE`	Yes	Student-selected subscription end.
`status`	`ENUM`	Auto	Active | Expired | Cancelled.
`payment_status`	`ENUM`	Auto	Unpaid (default) | Paid. Admin updates this.
`subscribed_on`	`DATETIME`	Auto	Timestamp when order was placed.

Module D — Complaint Management

7. Module D: Complaint Management Workflow

Complaint management is the most time-sensitive admin function. Students submit complaints and expect visible progress. Admin must review, update status with a defined progression, and provide remarks. All rules below are enforced at the backend.

Stage D-1 View All Complaints

Admin sees all complaints across all students, filterable by status and category.

ID	Student	Category	Room	Date	Status	Action
`#CMP-024`	Sudharth K.	Plumbing	`B-204`	Feb 12	Pending	Review
`#CMP-021`	Sudharth K.	Electrical	`B-204`	Feb 08	In Progress	Review
`#CMP-015`	Sagar L.	Cleanliness	`A-101`	Jan 28	Resolved	View

- Filter pills: All | Pending | In Progress | Resolved | Rejected.
- Filter by category: Plumbing | Electrical | Cleanliness | Mess | Security | Other.
- Sorted by date — newest first by default.
- 'Review' action opens the Complaint Detail page (Stage D-2).

Stage D-2 Review Complaint Detail

Admin reads full complaint including student info, room, description, and any attachment.

The complaint detail page gives the admin everything needed to understand and act on a complaint:

Section	Content
Complaint Info	Complaint ID, Category, Date Submitted, Last Updated, Current Status.
Student Info	Name, ERP, Room Number, Block — pulled from students table.
Description	Full complaint text as entered by student.
Attachment	Image preview (if uploaded). Click to view full size.
Previous Remark	Admin remark from last status update (if any). Shown in crimson box.
Admin Actions	Status update dropdown + remark text field + Submit button. Remark is REQUIRED when rejecting.

Stage D-3

Update Complaint Status

Admin changes the status and adds a remark. Status must follow defined transitions.

Allowed Status Transitions

Complaint status follows a strict linear progression. Random jumps are blocked at the backend:

From Status		To Status	Remark Required?	Notes
Pending	→	**In Progress**	Optional	Work has begun on the issue.
In Progress	→	**Resolved**	Optional	Issue fully resolved.
Pending	→	**Rejected**	**REQUIRED**	Complaint invalid / outside scope. Remark explains why.
Resolved / Rejected	→	**Any Status**	**BLOCKED**	Status cannot go backwards or be changed after terminal states.

Step 1: Admin selects new status and enters remark

- Admin opens complaint detail, selects new status from dropdown.
- Dropdown only shows valid next states based on current status (invalid transitions are not shown).
- If status = Rejected → remark field becomes required (cannot submit without it).
- For other transitions → remark is optional but strongly encouraged.

Step 2: System validates transition and writes update

- Backend re-validates the transition (even if UI already filtered options).
- If invalid transition attempted via API → 400 Bad Request.
- UPDATE complaints SET status = [new], admin_remark = [text], updated_at = NOW(), updated_by = [admin_id] WHERE id = [complaint_id].
- Student sees updated status and remark on their next view of the complaint.

No Deletion: Complaint records cannot be deleted — not even by admin. Any DELETE /complaints/:id request returns 403 Forbidden at the route level. This ensures a full audit trail. If a complaint is spam or invalid, the correct action is to Reject it with a remark.

7.1 Updated Complaints Table Schema

The complaints table from Module 3 requires two additional columns for admin management:

Column	Type	Required	Description
`complaint_id`	`UUID`	Auto	Unique complaint ID, auto-generated.
`student_id`	`INT / FK`	Auto	FK from students table.
`room_number`	`VARCHAR`	Auto	Pulled from student record at time of submission.
`category`	`ENUM`	Yes	Plumbing | Electrical | Cleanliness | Mess | Security | Other.
`description`	`TEXT`	Yes	Full complaint text. Minimum 1 character.
`attachment`	`VARCHAR`	No	File path to uploaded image. JPEG/PNG, max 2MB.
`status`	`ENUM`	Auto	Pending | In Progress | Resolved | Rejected. Default: Pending.
`admin_remark`	`TEXT`	No	Admin's response. Required when status = Rejected.
`updated_by`	`INT / FK`	Auto	FK to users.id of the admin who last updated status.
`created_at`	`DATETIME`	Auto	Timestamp of submission.
`updated_at`	`DATETIME`	Auto	Timestamp of last admin action.

8. Permission Boundaries

The following table defines exactly what admin can and cannot do, enforced at the backend route level:

Action	Admin	Notes
Create / add rooms to inventory	✓ Allowed	Rooms module.
Allocate room to student	✓ Allowed	With capacity + duplicate checks.
Deallocate room (with reason)	✓ Allowed	Confirmation + mandatory reason field.
Update mess payment status (Unpaid → Paid)	✓ Allowed	Manual toggle. No gateway needed.
Update complaint status (valid transitions only)	✓ Allowed	Backend validates transition.
Add / update admin remark on complaint	✓ Allowed	Required on Reject.
View student profiles and history	✓ Allowed	Read-only view.
Delete complaint records	✗ Blocked	403 at route level. Use Reject instead.
Modify student passwords	✗ Blocked	Not in MVP scope. Requires reset flow.
Access student routes (/dashboard, /complaints/new)	✗ Blocked	403 Forbidden. Role middleware.
Deactivate student account	✗ Blocked	Not MVP. Cascade plan required first.
Skip complaint status transitions	✗ Blocked	Backend validates every transition.

9. Critical System Rules

The following rules must be enforced at the database and backend level — not just in the UI. If someone bypasses the interface and calls the API directly, these rules must still hold.

#	Rule	Module	Detail
1	**Room capacity cannot exceed limit**	Room Allocation	Backend checks occupied_count < capacity before every allocation. UI cannot bypass this.
2	**One student = one room per term**	Room Allocation	Backend checks for existing room_number per term before allocating. Duplicate allocation is blocked.

#	Rule	Module	Detail
3	**Complaint status follows defined transitions**	Complaint Management	Pending→In Progress→Resolved or Pending→Rejected only. All others return 400 Bad Request.
4	**Admin remark required on Rejection**	Complaint Management	If new status = Rejected and remark is blank, backend rejects with 422 Unprocessable Entity.
5	**Complaints cannot be deleted**	Complaint Management	DELETE /complaints/:id returns 403 Forbidden regardless of role.
6	**Deallocation requires mandatory reason**	Room Management	Backend validates reason field is non-empty before processing deallocation.
7	**Role enforcement on all admin routes**	All Admin Modules	Every /admin/* route requires session with role = 'admin'. Middleware runs before handler.
8	**payment_status only admin-updatable**	Mess Management	Students cannot change payment_status. Only admin can toggle Paid/Unpaid.
9	**updated_by recorded on every status change**	Complaint Management	Every complaint status update stores the admin_id in updated_by for audit trail.
10	**Soft-delete pattern only**	All modules	No hard deletes on any core record in MVP. Use status fields to mark records inactive.

10. Conclusion

Repository	https://github.com/Sagarlekhraj-19/HMS.iba.edu.pk
Owner	Sagar Lekhraj
TAs	adeenaoop, Muh-Aqib-Shah

The Admin Panel is not a secondary concern — it is the operational core of the Hostel Management System. Every student action generates a responsibility for the admin: a room application to review, a complaint to resolve, a payment to record. Without a well-designed admin layer, the student-facing modules are incomplete.

Module A (Student Management) gives admin a complete view of every student and their hostel lifecycle. Module B (Room Management) enforces physical capacity constraints with dual backend validation that cannot be bypassed. Module C (Mess Management) keeps billing transparent with a simple Paid/Unpaid toggle, deliberately avoiding payment gateway complexity in MVP. Module D (Complaint Management) enforces a strict status progression with mandatory remarks on rejection, ensuring students always receive a meaningful response and creating a full audit trail.

Together, these four modules give the admin complete visibility and control over the hostel — cleanly, logically, and in a way that is directly expandable post-MVP.

End of Document | Admin Workflow Documentation | HMS.iba.edu.pk | Version 1.0