HMS.iba.edu.pk

Hostel Management System

WORKFLOW DOCUMENTATION — Module 0: Authentication & Dashboard

Prepared By	Sagar Lekhraj (Group Leader) & Sudharth Kumar
Module	Module 0 — Authentication & Dashboard
Program	BSCS
Course	Web-Based Application Development
Version	1.0 — Final
Date	February 2026

Team

Sagar Lekhraj (Leader) — Module 1 & 2: Room Allocation & Mess Subscription
Sudharth Kumar — Module 3: Complaint & Feedback System
Module 0 (Auth & Dashboard) — Shared foundation for both modules

1. Document Overview

This document covers Module 0 of HMS.iba.edu.pk, the foundational layer that powers all other modules. It defines:

- Part A — Authentication Workflow: Student signup, login, session management, role-based redirection, and logout.
- Part B — Student Dashboard Workflow: Auth guard, dynamic data fetching from the database, layout rendering, and conditional UI logic.

This module acts as the entry gate. Every other module (Room Allocation, Mess Subscription, Complaint & Feedback) requires the student to have authenticated through this system.

Actor	Role in This Module
Student	Registers, logs in, accesses dashboard, navigates to modules.
Admin	Has separate login. Cannot access student dashboard. Redirected to /admin/dashboard.
HMS System	Validates credentials, hashes passwords, creates sessions, fetches and renders data.

Full System Flow

Open URL	Session check	Login / Signup	Validate	Role check	Dashboard	Load Data	Navigate Modules

PART A: Authentication Workflow (MVP)

Stage 0 **System Entry**
User opens system URL. System checks for an existing session.

When a user first visits hms.iba.edu.pk, the system immediately performs a session check:

- If a valid session exists → skip the landing page, redirect directly to Student Dashboard.
- If no session exists → display the landing page with Login and Signup options.
- If a logged-in user manually navigates to /login or /signup → redirect to dashboard.

Security Note: The landing page is fully public — no authentication required to view it. All other routes (/dashboard, /complaints, etc.) are protected.

Stage 1 **Signup Workflow**
New student creates an account and is auto-logged in.

1.1 Signup Form Fields

When the student clicks Signup, the system loads the registration form with the following fields:

Field	Required	Notes
`Full Name`	Yes	Student's full legal name.
`Program`	Yes	Dropdown: BSCS, BBA, BS Econ, MBA. Used by Room Allocation module.
`Email`	Yes	Must be valid format. Must not already exist in users table.
`Student ID (ERP)`	Yes	Must be unique. Stored in students table.
`Password`	Yes	Minimum 8 characters. Hashed with bcrypt before storage.
`Confirm Password`	Yes	Must match Password field.
`role`	Auto	Always set to "student" by system. Student cannot modify.
`room_number`	Auto	Set to NULL at registration. Updated when admin allocates room.
`hostel_block`	Auto	Set to NULL at registration.

1.2 Signup Validation

Before creating the account, the backend performs the following checks in order:

#	Rule	Applies To	Detail
1	**All required fields filled**	Frontend + Backend	Form will not submit if any required field is empty.
2	**Email format is valid**	Backend	Regex check — must match standard email pattern.
3	**Email not already registered**	Backend	SELECT from users table — duplicate email blocked.
4	**Student ID is unique**	Backend	SELECT from students table — duplicate ERP blocked.
5	**Password matches Confirm Password**	Backend	String comparison — exact match required.
6	**Password length >= 8 characters**	Backend	Minimum 8 characters enforced.

On Validation Failure: Form is NOT submitted. All entered values are retained. Inline error messages appear below each invalid field. A summary error banner is shown at the top of the form. Both email and student ID uniqueness errors are shown simultaneously if both fail.

1.3 Account Creation (Steps)

Step 1: Hash Password

- Use bcrypt to hash the password before any database write.
- The plain-text password is NEVER stored.

Step 2: Insert into users table

- id — auto-generated
- name — from form
- email — from form
- password — bcrypt hash
- role = "student"
- created_at — server timestamp

Step 3: Insert into students table

- user_id — FK from users.id
- student_id — ERP from form
- program — from form
- room_number = NULL
- hostel_block = NULL

Step 4: Create Session

- Store in session: user_id, role, name, student_id
- These values are immediately available for dashboard data fetch
- No second login required

Step 5: Redirect to Dashboard (Option A — MVP)

- MVP Choice: Auto-login after signup (Option A).
- Student is immediately redirected to /dashboard.
- Option B (future): Redirect to /login for manual verification.

1.4 Database Schema — users & students Tables

users Table

Column	Type	Required	Description
id	INT / UUID	Auto	Primary key, auto-generated
name	VARCHAR	Yes	Student full name from signup form
email	VARCHAR	Yes	Unique. Used as login identifier
password	VARCHAR	Yes	bcrypt hashed. Plain text never stored
role	ENUM	Auto	Values: 'student' or 'admin'. Always 'student' on signup
created_at	DATETIME	Auto	Timestamp of account creation

students Table

Column	Type	Required	Description
user_id	INT / FK	Auto	Foreign key referencing users.id
student_id	VARCHAR	Yes	ERP number. Must be unique
program	VARCHAR	Yes	e.g. BSCS, BBA. Referenced by room allocation module
room_number	VARCHAR	Auto	NULL until admin allocates. e.g. B-204
hostel_block	VARCHAR	Auto	NULL until allocated. e.g. Block B

Stage 2 **Login Workflow**
Existing user authenticates and is redirected by role.

2.1 Login Form Fields

When the user clicks Login, the system loads a simple two-field form:

- Email Address — checked against the users table
- Password — compared to the stored bcrypt hash using bcrypt.compare()

2.2 Login Validation Logic

Condition	Result	Response to User
Email not found in DB	Login fails	"Invalid credentials"
Email found, password mismatch	Login fails	"Invalid credentials"
Email + password match	Login succeeds	Create session, redirect by role

Security Principle: Both 'email not found' and 'wrong password' show the exact same error: "Invalid credentials". This prevents attackers from discovering which emails are registered in the system (email enumeration attack prevention).

2.3 Session Payload

On successful login, the system creates a session storing the following values:

Key	Example Value	Purpose
`user_id`	29325	Identifies the logged-in user for all DB queries
`role`	"student"	Controls route access and redirections
`name`	"Sagar Lekhraj"	Displayed in dashboard header and sidebar
`student_id`	"29325"	Used in complaints, mess, room queries without extra DB call

2.4 Role-Based Redirection

After a successful login, the system checks the role stored in the session:

role Value	Redirect To	Notes
`"student"`	`/dashboard`	Lands on Student Dashboard. Sees own data only.
`"admin"`	`/admin/dashboard`	Lands on Admin Panel. Can manage complaints, allocate rooms.

Stage 3 **Logout Workflow**
Student ends session and is redirected to login.

When the student clicks the Logout button in the dashboard topbar:

- The logout action is submitted as a POST request to /logout (not GET — to prevent accidental triggering by browser prefetch or link crawlers).
- The server destroys the session or invalidates the JWT token.
- The student is redirected to /login.
- If the student tries to go back to /dashboard after logout, the auth guard redirects them back to /login.

Implementation Note: Logout must be a POST request, not GET. A GET /logout URL can be triggered accidentally by browsers, security scanners, or link prefetchers, causing unintended session destruction.

2. Authentication Rules & Security

The following rules apply across the entire authentication system:

#	Rule	Detail
1	**Password hashing**	Passwords are hashed with bcrypt before storage. Plain-text passwords are never stored or transmitted.
2	**Session / JWT security**	Sessions store user_id, role, name, student_id. Tokens should have a reasonable expiry (e.g., 24 hours).
3	**Protected routes**	All routes except /login, /signup, and / require a valid session.
4	**Role enforcement**	Student cannot access /admin/* routes. Admin cannot access student-only pages without explicit permission.
5	**Login → Signup redirect**	If a logged-in student visits /login or /signup, they are redirected to /dashboard automatically.
6	**Duplicate prevention**	Email and Student ID uniqueness is enforced at the database level (UNIQUE constraints), not just application level.
7	**Generic error messages**	Login errors never specify whether the email or password was incorrect, to prevent enumeration attacks.
8	**Logout is POST**	Logout is a POST request to prevent accidental session destruction via link prefetching.

PART B: Student Dashboard Workflow (MVP)

Stage 0 — Dashboard Access — Auth Guard

Every request to /dashboard runs middleware to verify the session.

The dashboard is a protected route. Before any data is loaded or rendered, the system runs two mandatory checks:

#	Check	Pass	Fail
1	**Session exists and is valid**	Continue to check 2	Redirect to /login
2	**Session role = "student"**	Load dashboard	403 Forbidden or redirect admin

Precondition: Student must be authenticated and have an active session with role = "student".

Stage 1 — Dashboard Data Fetch

Four database queries run in parallel when the dashboard loads.

When the dashboard loads successfully, the system performs four database queries. These should be parallelised for performance rather than run sequentially:

#	Query Name	SQL / Logic	Used For
1	**Student Profile**	`SELECT * FROM students JOIN users ON students.user_id = users.id WHERE users.id = [session.user_id]`	Profile card — name, ERP, program, room, block
2	**Room Allocation Status**	`Read students.room_number — NULL = Not Allocated, else Allocated`	Room status card + module card CTA
3	**Mess Subscription Status**	`SELECT status FROM subscriptions WHERE student_id = [session.student_id] AND status = 'Active' LIMIT 1`	Mess status card + module card CTA
4	**Active Complaints Count**	`SELECT COUNT(*) FROM complaints WHERE student_id = [session.student_id] AND status IN ('Pending','In Progress')`	Complaints status card count

Loading UX: While queries are running, show skeleton loaders in place of profile card, status cards, and module cards. The sidebar and topbar are rendered immediately from session data (no DB query needed).

Stage 2 **Render Dashboard Layout**
Three sections are rendered using the fetched data.

The dashboard has three main sections. Each is data-driven:

2.1 Section 1 — Student Profile Card

Displayed prominently at the top of the dashboard:

- Avatar (initials), Full Name, ERP ID, Role chip
- Program, Batch, Term
- Email address
- Room Number — shown as allocated value (e.g. B-204) OR "Not Allocated" if NULL
- Hostel Block — shown as value (e.g. Block B) OR "N/A" if NULL

2.2 Section 2 — Status Overview Cards

Three small info cards displayed in a row, each dynamically populated:

Card	Condition	Display Value
Room Status	students.room_number IS NULL	Not Allocated (grey)
Room Status	students.room_number IS NOT NULL	Allocated (green) + room number
Mess Status	No Active subscription found	Not Subscribed (grey)
Mess Status	Active subscription exists	Subscribed (green) + plan name
Active Complaints	COUNT = 0	0 — No active complaints
Active Complaints	COUNT > 0	N open (amber) + breakdown

2.3 Section 3 — Module Navigation Cards

Three large clickable cards, each leading to a module. The CTA (call-to-action) text changes based on the student's data state:

Module	CTA if not started	CTA if active/allocated
Room Allocation	Apply for Room →	View Room Details →
Mess Subscription	Subscribe to Mess →	Manage Mess →
Complaint & Feedback	Raise Complaint →	Open Complaints → (with count badge)

Stage 3 **Conditional Dashboard Logic**

The dashboard adapts intelligently to each student's current state.

The dashboard is not a static page. It uses the fetched data to determine exactly what to show each student. The key conditional cases are:

Case A: Brand New Student (just registered)

- room_number = NULL → Show welcome banner. Room card shows "Not Allocated" with "Apply for Room" CTA.
- No mess subscription → Mess card shows "Not Subscribed" with "Subscribe" CTA.
- Zero complaints → Complaint count shows 0.
- Profile room/block fields show "Not Allocated" and "N/A".

Case B: Student with Room Allocated

- room_number = "B-204" → Room status card shows "Allocated" in green.
- Room Allocation module card CTA changes from "Apply" to "View Room Details".
- Profile card shows actual room number and block.
- Welcome banner is NOT shown.

Case C: Student with Active Mess Subscription

- Active mess subscription found → Mess card shows "Subscribed" in green + plan names.
- Mess module card CTA changes from "Subscribe" to "Manage Mess".

Case D: Student with Open Complaints

- COUNT > 0 → Complaints card shows the number in amber.
- Sub-label shows breakdown: e.g. "1 Pending · 1 In Progress".
- Complaint module card shows a badge with the open count.

Stage 4 — Navigation Flow

Dashboard acts as the central hub for all module navigation.

From the dashboard, students can navigate to any of the three modules by clicking the corresponding module card. The sidebar navigation also provides persistent access to all modules from any page.

Important: The dashboard does not navigate to admin routes. Clicking Logout (POST /logout) destroys the session and redirects to /login. The student cannot access another student's data — all queries are scoped to session.user_id.

3. Combined Validation & Business Rules

#	Rule	Applies To	Detail
1	**Term selection mandatory**	Room Application	Cannot submit application without selecting a hostel term.
2	**Password hashed**	Auth — Signup/Login	bcrypt used. Plain-text passwords never stored or logged.
3	**Email uniqueness enforced**	Auth — Signup	UNIQUE constraint at DB level. Application-level check also runs.
4	**Student ID uniqueness enforced**	Auth — Signup	UNIQUE constraint at DB level. Duplicate ERP blocked.
5	**Generic login error**	Auth — Login	Same message for both 'email not found' and 'wrong password'.
6	**Session-scoped queries**	Dashboard	All DB queries use session.user_id — students cannot see each other's data.
7	**Role-protected routes**	All modules	Students cannot access /admin/* routes. Enforced by middleware.
8	**Logout is POST**	Auth — Logout	Prevents accidental session destruction by link prefetching.
9	**Auto-redirect if logged in**	Auth — Login/Signup	Logged-in user visiting /login or /signup is redirected to /dashboard.
10	**Dashboard conditional CTAs**	Dashboard	Module card CTAs adapt based on room_number and subscription status.

4. GitHub & Conclusion

Repository	https://github.com/Sagarlekhraj-19/HMS.iba.edu.pk
Owner	Sagar Lekhraj
TAs	adeenaoop, Muh-Aqib-Shah
Strategy	Individual feature branches merged to main on completion.

Module 0 — Authentication and the Student Dashboard — establishes the secure, intelligent foundation upon which all three HMS modules operate. The authentication system provides clean role-based access control, password security through bcrypt hashing, and a frictionless signup-to-dashboard experience using Option A auto-login.

The Student Dashboard is not a static page but a dynamic data hub. By performing four targeted database queries on load and applying conditional logic to the UI, the dashboard gives every student a personalised view of their current hostel status — from room allocation to mess subscriptions to open complaints — and adapts its calls-to-action based on what they have or have not yet done. This makes the system feel intelligent and responsive without requiring any configuration from the student.

Together, these workflows replace static entry points with a fully digital, role-aware, data-driven access layer integrated into the HMS portal.

End of Document | Module 0: Authentication & Dashboard | HMS.iba.edu.pk | Version 1.0