Product Requirement Document (PRD) — Bit8 Class System

Prepared from: provided system-instruction PDF
Language: English
Target stack: MERN (MongoDB, Express, React, Node) — implementor may adapt, but architecture and APIs must follow the specification below.

1. Purpose & Scope

Purpose: Regenerate and implement a complete Bit8 class management system that supports student and teacher administration, groups, courses, finance, reports, and user roles. The system must follow modern web app standards (MERN-ready), be responsive, accessible, and visually consistent with the color and UI rules specified.

Scope: Frontend UI, backend API, data model, PDF export, authentication for roles (Admin / Finance / User), reporting, and full validation rules described below. This PRD covers functional & non-functional requirements, UI/UX rules, data model, constraints, acceptance criteria, and implementation notes.

2. Product Vision & Goals

- Provide a single coherent admin panel for managing students, teachers, committees, groups, courses, hobbies, and finances.
- Ensure data integrity (no duplicated student assignments per group/subject, correct student → data matching).
- Deliver professional PDF exports and Excel export for teachers.
- Create an attractive, modern, responsive UI using the specified color system and visual rules.
- Make the product implementation-ready (APIs, DB schema, UI components, validation rules).

3. Roles & Permissions

- **Admin** — Full CRUD on students, teachers, users, committees, leaders, groups, courses; manage finances; export reports; create semesters; assign subjects/groups.

- **Finance** — Access finance module & reports; open payments; view semester financial data; generate finance reports.
- **User** — Student-level limited access (view own profile/hobbies/groups); login via 4-digit PIN; basic interactions.

Access control rules: Role-based endpoints and UI controls. All write operations require Admin or Finance roles appropriately.

4. High-level System Architecture (Guidance)

- **Frontend:** React (component-based), responsive design system, global theme variables (colors, fonts). Use modern state management (Redux / Context) for admin state.
- **Backend:** Node + Express APIs, RESTful endpoints (or GraphQL optionally). Authentication via JWT with role claims.
- **Database:** MongoDB collections keyed by stable IDs (students, teachers, users, groups, courses, semesters, finance).
- **Export utilities:** Server-side or headless renderer for professional PDF generation (e.g., Puppeteer, wkhtmltopdf, or PDF libraries); Excel export using libraries (xlsx).

5. Functional Requirements (Detailed by module)

5.1 Dashboard (global)

Description: Central navigation + summary widgets (counts for students, teachers, active groups, pending payments).
UI: Sidebar with round gradient buttons (use provided colors). Top bar advanced search (global search).
Advanced global search behavior:

- Case-insensitive
- Supports full name, partial name, and nickname/hint matching
- Fuzzy matching (typo-tolerant): e.g., “Abdirashiid” should match “Abdirashid” (use a fuzzy search algorithm like Levenshtein or Elasticsearch fuzzy query)
- Exact-match precedence: exact queries should return exact results first
  Acceptance criteria:

- Global search returns correct results for full, partial, and nickname hints; exact matches ranked higher.

5.2 Students Module

UI / Table: Columns: No (auto) | id | name | gender | phone | location | action (update, delete)
Functionalities:

- Search box (supports case-insensitive, partial, hints/nicknames)
- Registration (student create form)
- Edit & Delete (action buttons)
- PDF export (A4 professional layout for current table view or selected records)
- Per-row action column must include a Copy-Phone action (copy to clipboard)
  Business rules:
- Student ID unique
- Data integrity: student rows cannot be assigned another student's data
- Group assignment rule: a student can belong to at most one group per subject (enforced)
  Acceptance criteria:
- Registration validates required fields and saves correct data
- Copy phone works from action column without opening popup
- PDF export formatted per PDF spec (see section 9)

5.3 Teachers Module

UI / Table: Columns: id | full name | subject | phone | status (active/inactive) | semester | actions (update/delete)
Functionalities:

- Search box
- Export to Excel (current view)
- New teacher create form (modal)

- Status toggle (active/inactive)
- Actions update/delete (with confirmation)
  Acceptance criteria:
- Excel exports have correct headers and accurate row data.

5.4 Users Module

Functionalities:

- Create user accounts with role assignment
- 4-digit login button flow for student users (PIN)
- Role / access level controls
  Student user flow:
- Input 4-digit code → authenticate → show student name at the top
  Acceptance criteria:
- 4-digit authentication works for student users; role enforcement applied correctly.

5.5 Committee Module

UI / Features:

- New member button (create modal)
- Show committee members with photos in shadowed boxes (editable: edit/delete)
  Acceptance criteria:
- Photo upload/resizing, edit/delete works.

5.6 Leaders Module

UI / Features:

- Add leader button
- Leader cards with image, name, group; edit/delete from image box
  Acceptance criteria:

- Leaders list displays correctly and allows CRUD.

5.7 Courses Module

Workflow:

1. Add “semester” via a small rounded popup (title + metadata).
2. Once semester is created, it appears as a button.
3. When clicking a semester button, show “Add subjects” button that allows adding subjects assigned to that semester.
4. Subjects can then have groups attached.
   Acceptance criteria:

- Semester creation and subject assignment work; subjects list read by Finance module for payments.

5.8 Groups Module

Features:

- New group / subject button → create groups for a subject
- One student can be assigned to only one group per subject. No duplicate assignment allowed.
- Group delete
- Remove students from group
- Generate group PDF (subject / group / members)
  Validation rules:
- Enforce unique membership per subject
- Prevent duplicate entries (same student in same group twice)
  Acceptance criteria:
- Group assignment enforces uniqueness; group PDF shows members correctly.

5.9 Student Hobbies Module

UI / Table: student name | contact phone | hobbies/interests | actions
Functionalities:

- Add my hobbies (UI flow as provided in images)
- Edit / delete hobbies
  Acceptance criteria:
- Hobbies can be edited per student; data visible in table and searchable.

5.10 Finance & Reports

Sidebar: Finance & Reports
Functionalities:

- Semesters list (pulled from Courses module)
- Manage finance for selected semester: list subjects, create “Open payment” per subject
- Report generation: choose a semester → generate report (per PDF specs)
  Report behavior:
- Reports view must match provided images and be printable/exportable
  Acceptance criteria:
- Finance flow reads semesters/subjects from Courses; payment records per subject; reports generate correctly.

6. Data Model (Suggested collections / tables)

Students

- _id (ObjectId)
- studentId (string, unique)
- fullName (string)
- gender (enum)
- phone (string)
- school (string)
- location (string)

- hobbies (array of strings)
- groups (array of { subjectId, groupId })
- createdAt, updatedAt

Teachers

- _id, teacherId, fullName, subjects (array), phone, status, semester, timestamps

Users

- _id, username, role, pin4 (hashed), linkedStudentId (optional)

Courses / Semesters / Subjects

- semesters collection: _id, title, year, subjects[]
- subjects: _id, title, semesterId, groups[]

Groups

- _id, subjectId, title, members[] (studentId), timestamps

Finance

- transactions, payments, subjects payments config

Integrity rules to enforce in DB / application layer:

- Unique studentId, teacherId
- Group membership constraint: unique on (studentId, subjectId)
- Referential integrity: deleting a subject/group must optionally cascade or block when members exist (explicit remove flow)

7. UI / UX Requirements & Visual System

Colors (mandatory):

- Primary: #34832A
- Secondary: #705E8B
- Accent: #EDDB7A
  Typography & Alignment:

- Use a clean web font (e.g., Inter, Roboto) for UI; PDF uses Times New Roman as specified.
- Inputs, labels and table columns must be aligned and consistent.
  Components & Interactions:
- Splash screen: advanced animation, logo placeholder, text “WELCOME TO BIT8 PLATFORM”
- Sidebar with rounded gradient buttons (use gradient of Primary → Secondary), drop shadow and subtle glow.
- Buttons: some “liquid transparent” style options allowed (background with alpha), but text must remain readable.
- Popups: rounded modals for semester/subject creation.
- Action column: copy phone button (hover visual effect, copy to clipboard on click).
- All page layouts must be responsive and accessible (WCAG AA recommended).
  Design tokens: Define root CSS variables for colors, radii, shadows, font-sizes.

8. PDF & Export Requirements (SPECIFIC)

General formatting (required exactly):

- Font: **Times New Roman**
- Header font size: **14pt**
- Body font size: **12pt**
- Alignment: **justify** for body text; date present in header
- Header: logo placeholder at top-left (justify), document title/class name near logo
- Footer: signature line with provided image (“Kaaba-galasska line”) and space for signature
- Page size: **A4**
- All PDF outputs (students list, group members, reports) must follow the same visual style and colors (use Primary & Secondary as decorative shapes)

- Decorative shapes permitted but must not break times new roman requirement for main text

PDF content rules:

- For students lists: columns No | Name | ID | Phone (or No | Name | Phone | ID as requested)
- Date in header; signature and role in footer
- Correct alignment and spacing to be professionally acceptable

Export mechanics:

- Exports generated server-side for consistency (Puppeteer, headless Chrome, or server PDF lib)
- Provide a replaceable logo mechanism (configurable file path or upload within admin)

9. Non-Functional Requirements

- **Responsiveness:** Mobile / Tablet / Desktop. Use breakpoints and fluid grids.
- **Performance:** Dashboard payloads < 500 ms for basic summary queries; paginate large tables.
- **Security:** Role-based access control; secure PIN storage (hashed); JWT for sessions; input validation and sanitization.
- **Scalability:** Design collections and APIs to handle thousands of student records.
- **Reliability:** Backups for DB, transactional integrity for finance operations.
- **Maintainability:** Modular code, documented API contract, environment config.
- **Accessibility:** Keyboard navigable, semantic HTML, alt tags for images, visible focus states.
- **Internationalization-ready:** Texts must be externalized (i18n) for later translation (Somali/English).

10. Constraints & Assumptions

- **Constraint:** PDF formatting mandates Times New Roman and precise header/footer formatting.

- **Assumption:** Images referenced (committee/leader photos, signature) will be supplied as uploads and stored in a media folder/CDN.
- **Constraint:** Use MERN conventions — collections and REST API endpoints expected.
- **Assumption:** Exact UI mockups (images) will be provided by requestor for pixel-accurate areas (e.g., group card design). If not provided, implement according to the description.

11. Edge Cases & Validation Rules

- Prevent adding a student to more than one group per subject; show user-friendly error message if attempted.
- Prevent duplicate student entries (by studentId or phone).
- On delete actions (group/student/teacher) require confirmation; provide cascade options or block if dependent records exist.
- When generating PDFs or Excel exports, validate the data; fail gracefully with an error showing missing required fields.

12. API Suggestions (minimum endpoints)

- GET /api/students — list with filters, pagination, fuzzy search
- POST /api/students — create
- PUT /api/students/:id — update
- DELETE /api/students/:id — delete
- GET /api/teachers, POST /api/teachers, etc.
- GET /api/semesters, POST /api/semesters, POST /api/semesters/:id/subjects
- POST /api/subjects/:id/groups — create group
- POST /api/groups/:id/members — add student to group (validate uniqueness)
- GET /api/reports/{type} — PDF generation
- GET /api/export/teachers — Excel

Include clear request/response contracts in implementation docs.

13. Acceptance Criteria (What “Done” Means)

1. All modules implemented according to functional sections above.
2. Global and module-level searches are case-insensitive and support hint/nickname matching and fuzzy typos.
3. Student group membership constraint enforced server-side and client-side.
4. Copy phone action works in the Action column (no popup dependency).
5. PDF exports follow the exact formatting rules (Times New Roman, sizes, header/footer, logo placeholder).
6. Excel export for teachers works correctly.
7. All pages responsive and visually consistent with the color system and buttons as described.
8. Authentication and role-based access enforced.
9. Documentation: README, API docs, data model diagrams, deployment steps delivered.

14. Deliverables

- Source code (frontend + backend), with environment setup instructions.
- Database schema / sample seed data.
- API documentation (OpenAPI/Swagger preferred).
- UI design tokens (CSS variables) and Sketch/Figma or style-guide snippets.
- PDF generation templates.
- Test cases (unit + integration) and QA checklist.
- Deployment instructions (production-ready build scripts).

15. Implementation Notes & Developer Hints

- Use a theme file with CSS variables for the 3 colors, radii, shadows and gradients; reuse across components.

- Implement fuzzy search using a library (Fuse.js on frontend or MongoDB text + regex or Elasticsearch for backend).
- Validate unique membership with a compound index: { studentId: 1, subjectId: 1 } (or application-level check).
- For PDF: render server-side HTML with CSS that uses Times New Roman, then convert to PDF to ensure pixel-perfect layout.
- Use clipboard API for copy phone actions and ensure accessible labels for screen readers.
- Provide feature flags for enabling/disabling modules during rollout.

16. Risks & Mitigation

- **Risk:** Fuzzy search returns false positives.
  Mitigation: tune threshold; prioritize exact matches; show confidence or allow search filters.
- **Risk:** PDF rendering differences across environments.
  Mitigation: use headless Chrome (Puppeteer) to render HTML → PDF consistently.
- **Risk:** Group assignment race condition (two admins assigning same student concurrently).
  Mitigation: server-side transaction/atomic check before write.

17. Next Steps (recommended)

1. Review and approve PRD and UI color & typography choices.
2. Provide UI mockups / screenshots referenced in the PDF (if any).
3. Provide sample images: logo, signature image, photos for committee/leaders.
4. Approve data model or provide adjustments.
5. Proceed to build sprint plan: estimate tasks for Students, Teachers, Courses, Groups, Finance, and PDF generation.

Appendix — Quick checklists for each major module

- **Students:** CRUD, search, PDF, registration, copy phone in Action column, group assignment constraints.
- **Teachers:** CRUD, search, Excel export, status toggle, semester link.
- **Courses / Semesters:** create semester (rounded popup), attach subjects, subject-level groups.
- **Groups:** create/delete, add/remove members, uniqueness enforcement, group PDF.
- **Finance:** semester-driven payments, open payment UI per subject, reports.
- **PDF:** Times New Roman, header 14, body 12, date in header, signature/footer, logo placeholder top-left.

Note : You are the system developer responsible for implementing and maintaining the platform.

If the person requesting the system (the client / product owner) provides new instructions such as:

- **"I want the system to be built using MERN stack."**
- **"Change this specific part of the system."**
- **"Modify this module."**
- **"Adjust this feature in a specific way."**

You must:

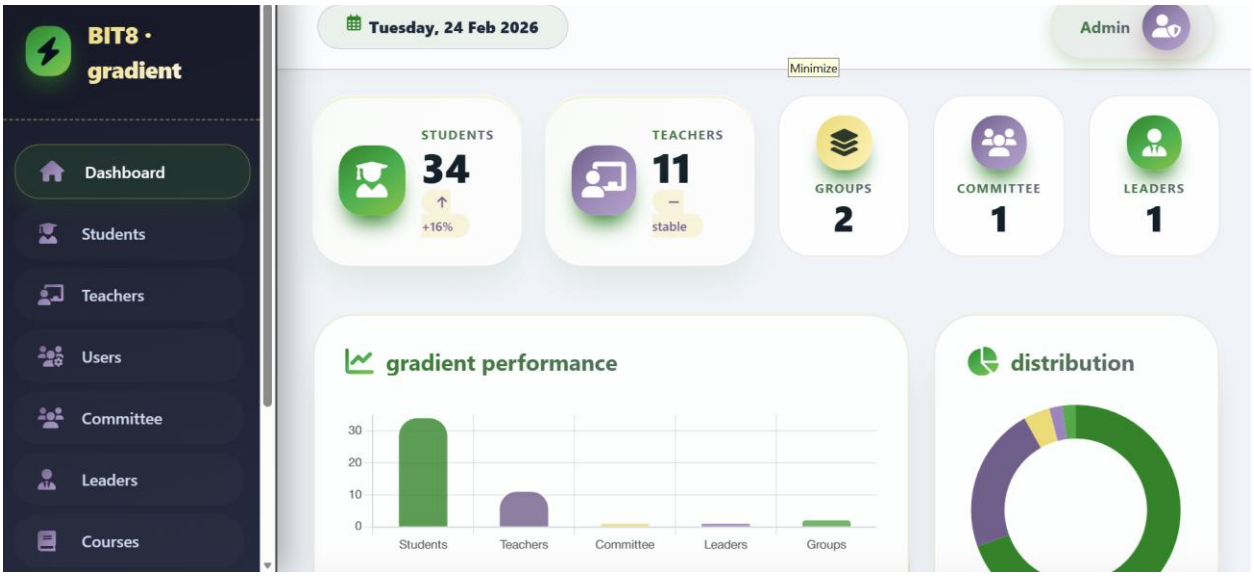
1. **Fully understand the request.**
2. **Follow the requested technical stack (e.g., MERN) exactly as specified.**
3. **Implement the requested changes without ignoring or altering the core system requirements.**
4. **Modify only the specified parts unless additional changes are logically required.**
5. **Preserve system integrity, structure, and data consistency.**
6. **Ensure that all updates remain aligned with the overall architecture and design standards.**

The client's request takes priority, and the system must be updated accordingly while maintaining professional development standards.

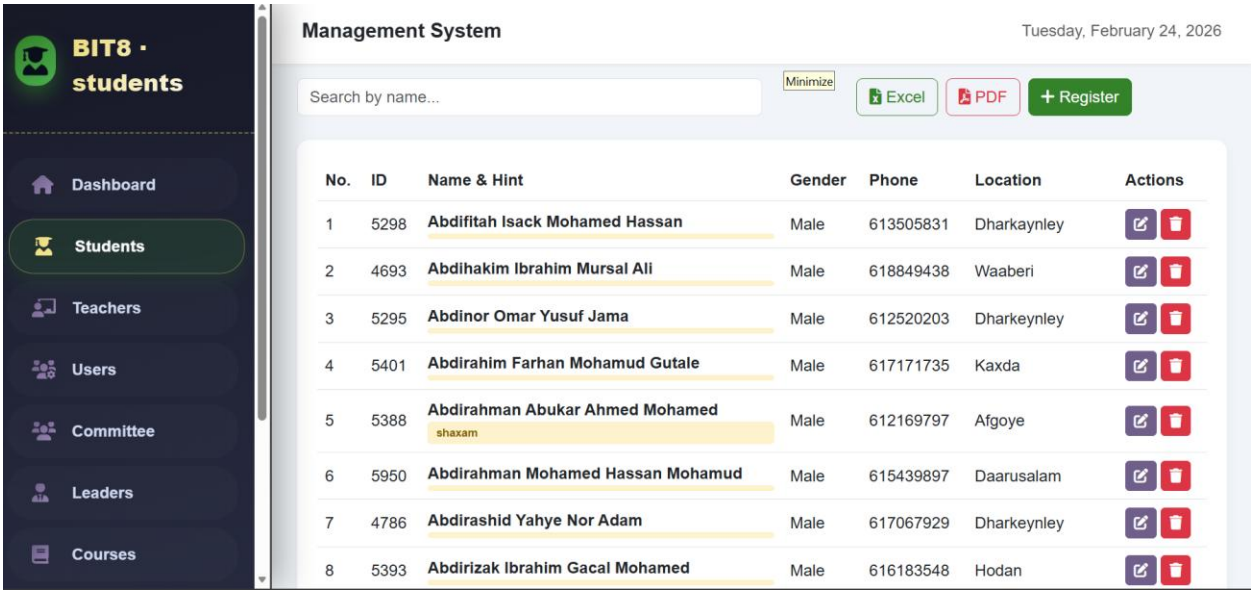
Images

Admin Section Images

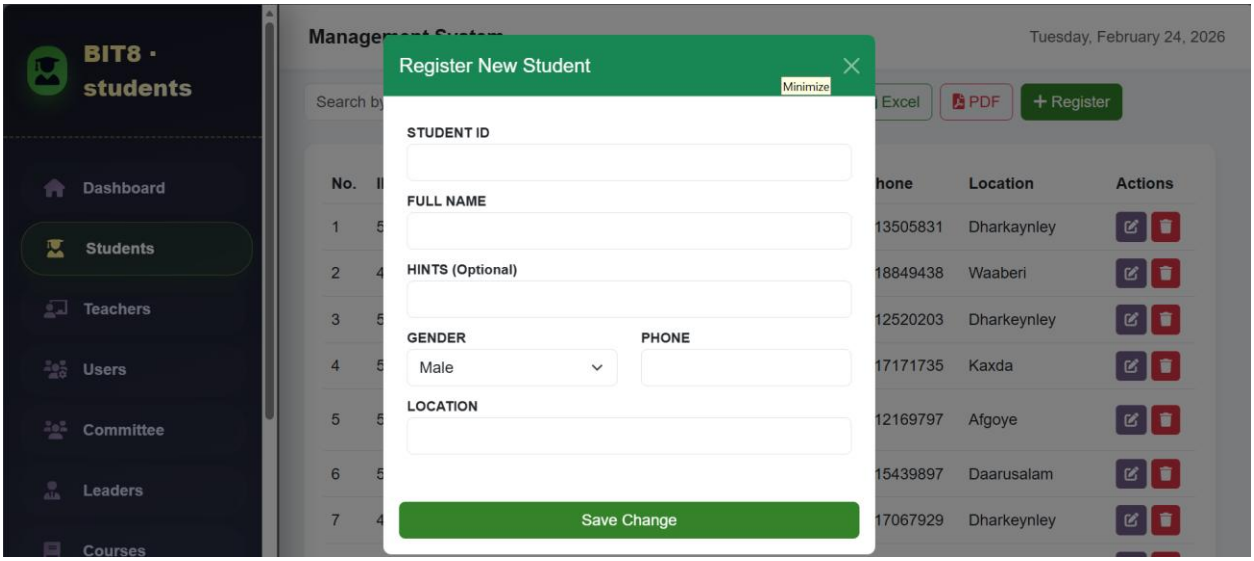
Dashboard



Students

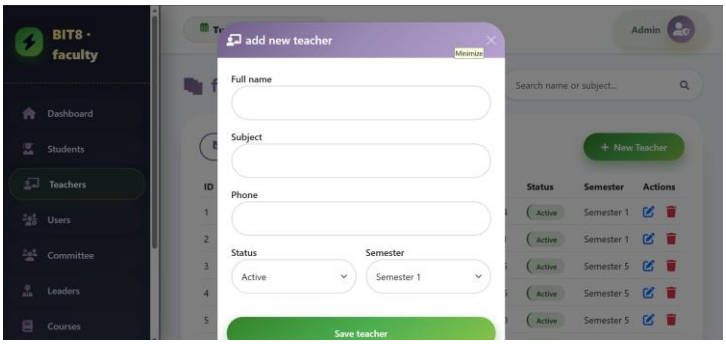


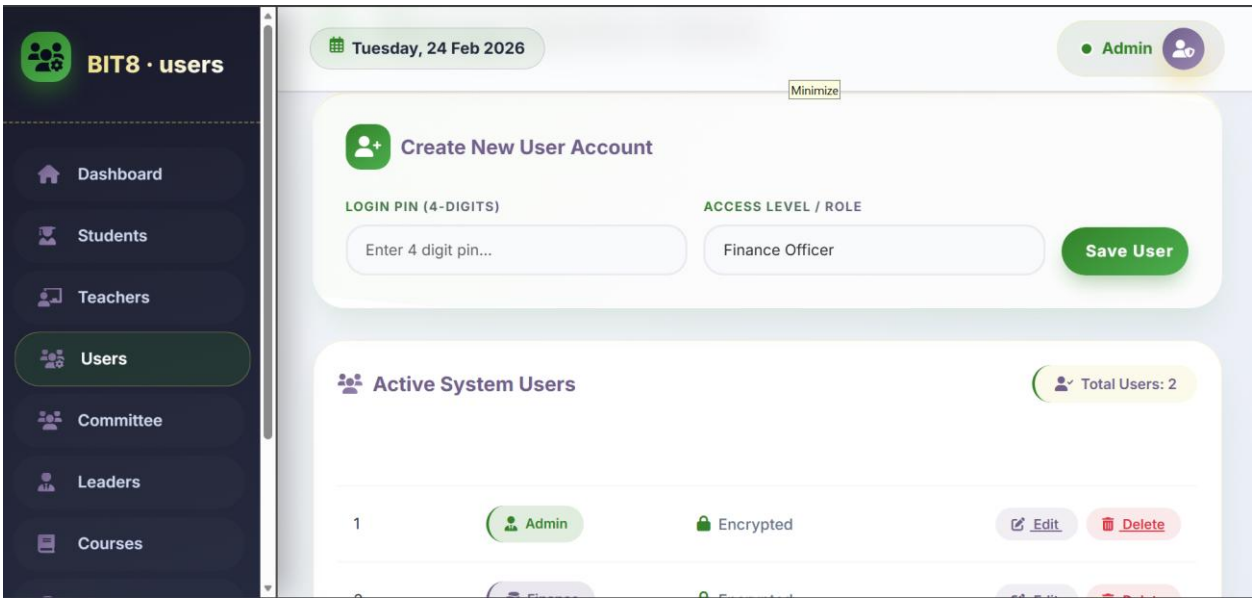
Student – register



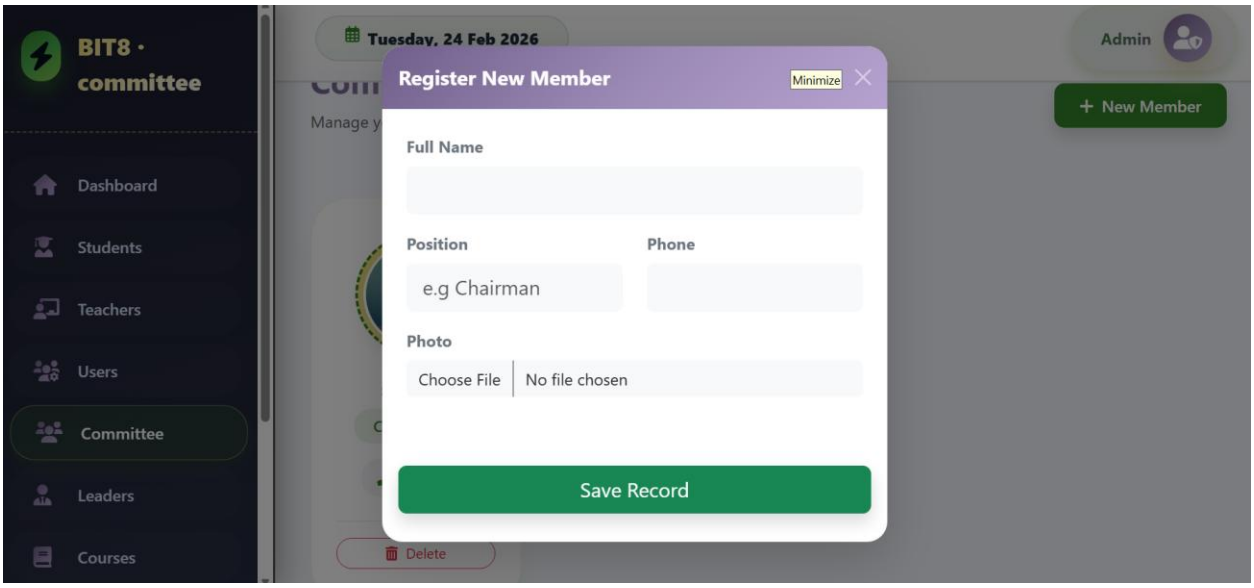
Teachers section

Users section

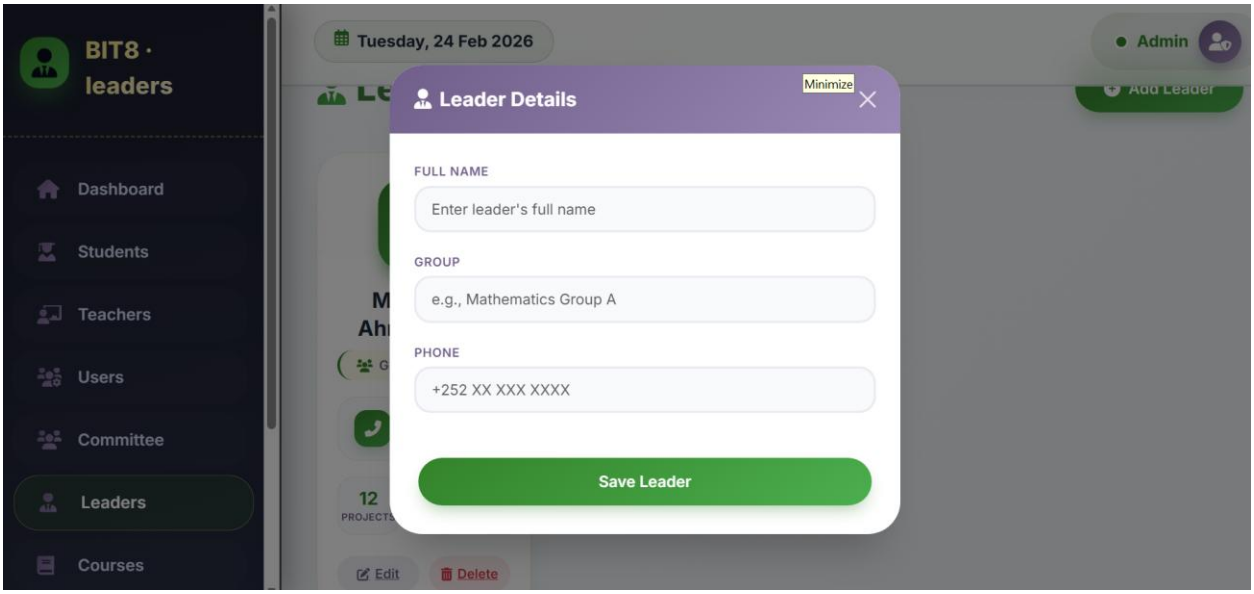




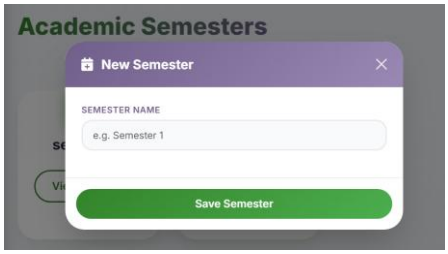
Committee section



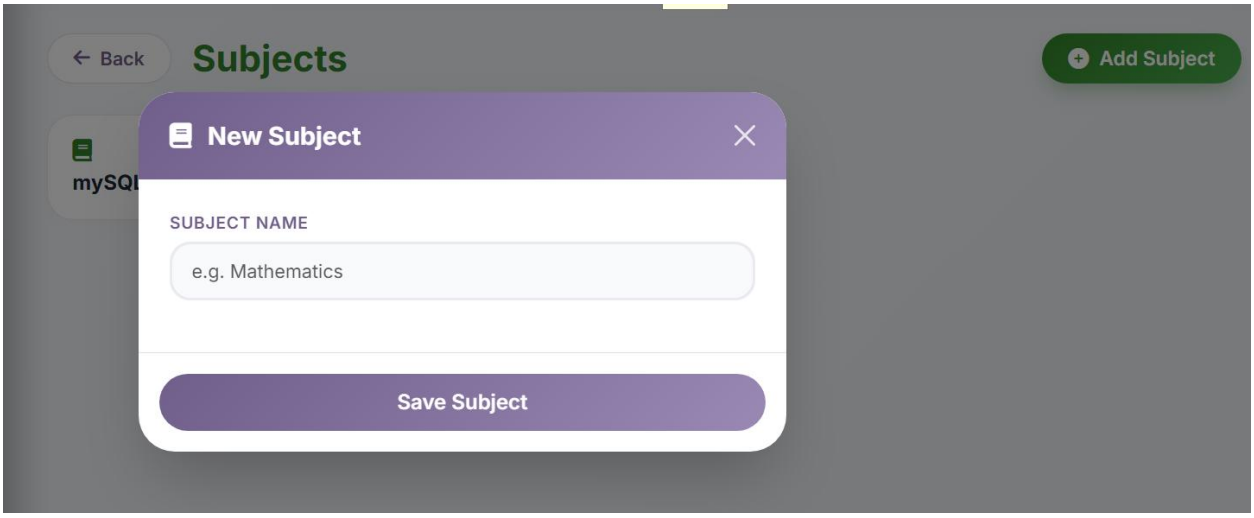
Leaders section



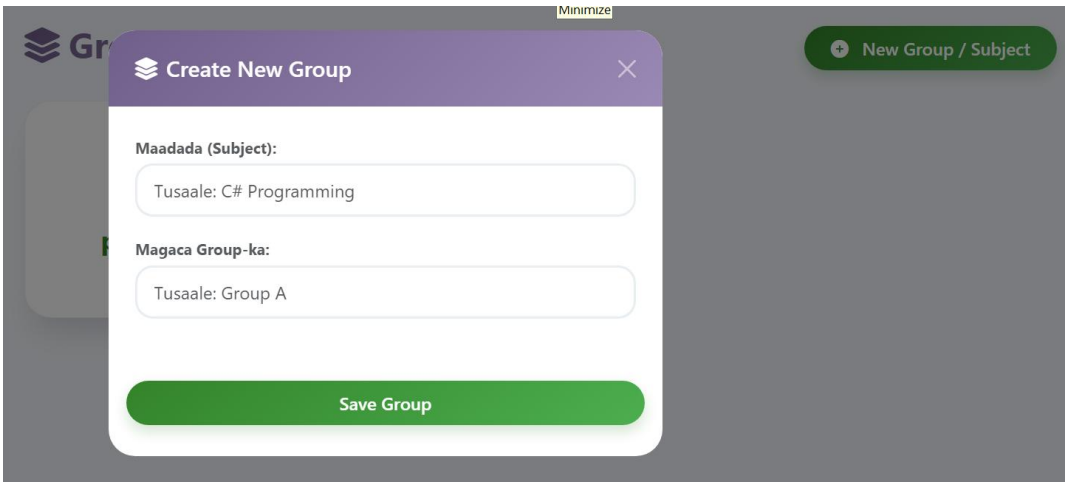
Courses section



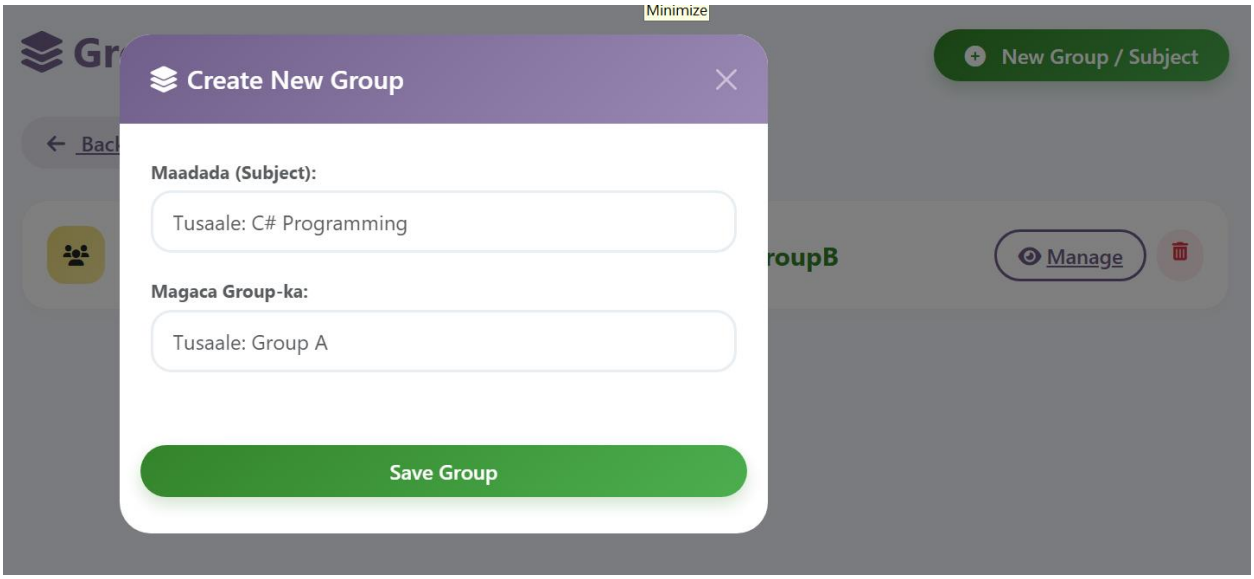
Afer click view subject and click add subject button, do this



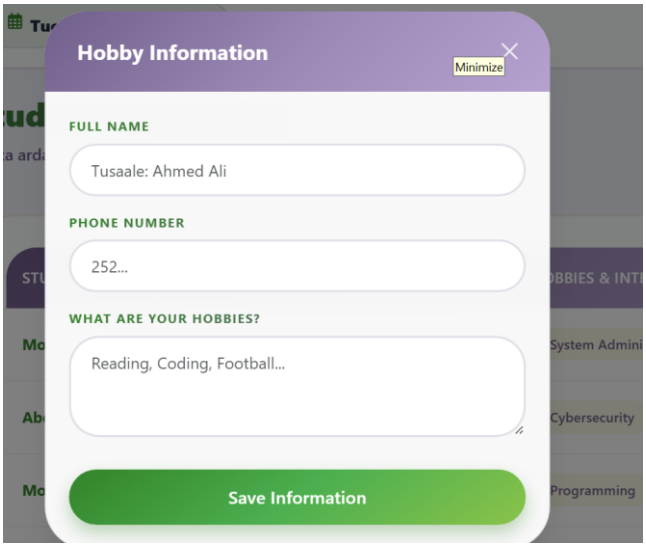
Groups section

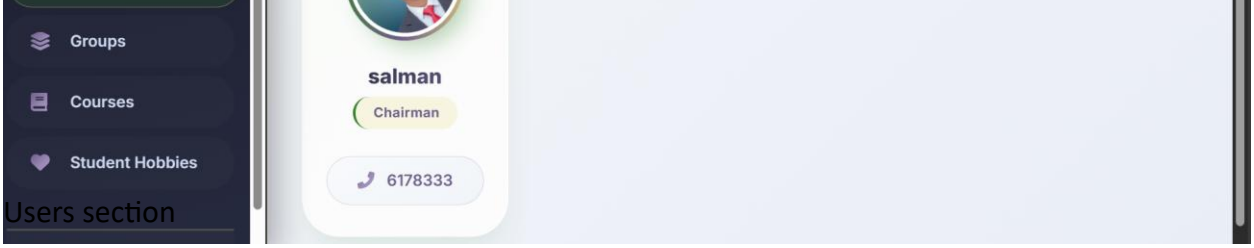


After click subject and click new group button

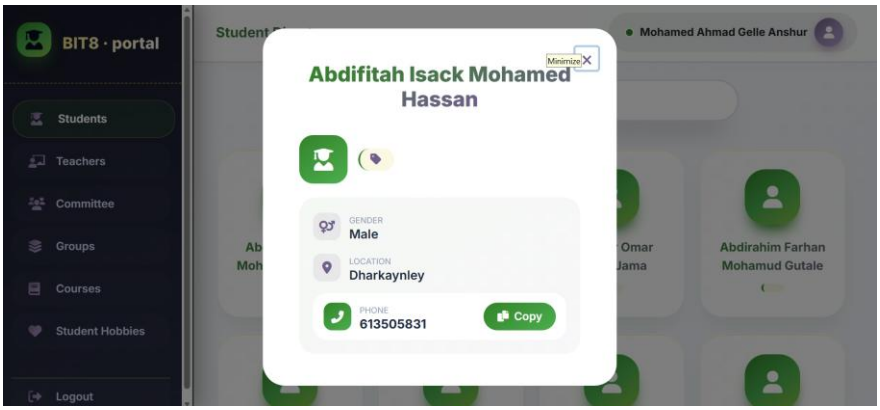


Student hobbies





Users section

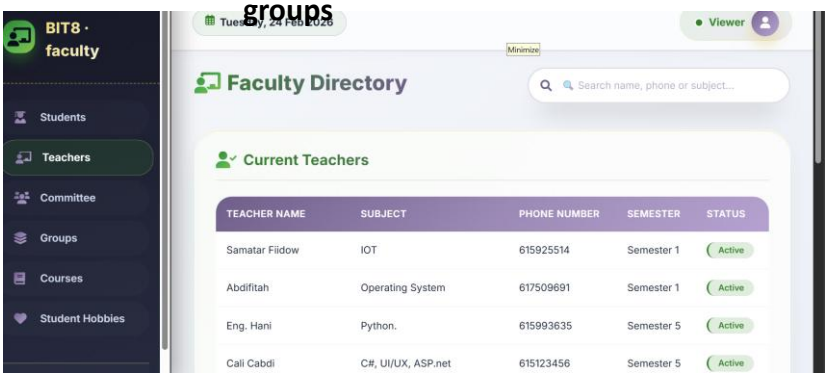


Teachers section

-
-

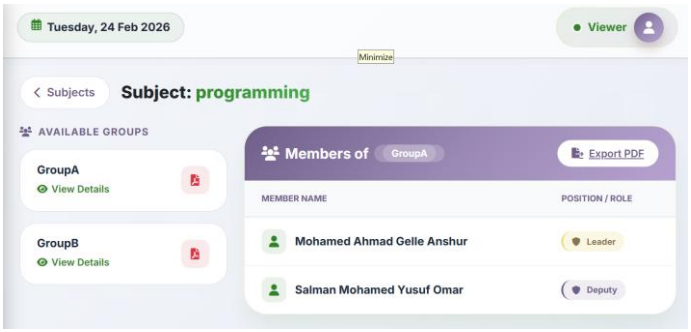
Committee section

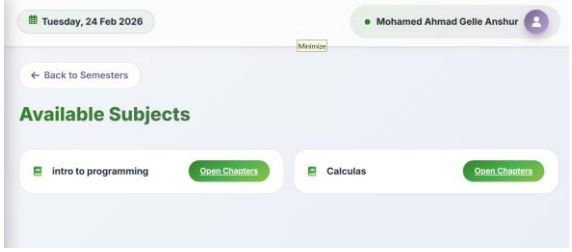
groups



-

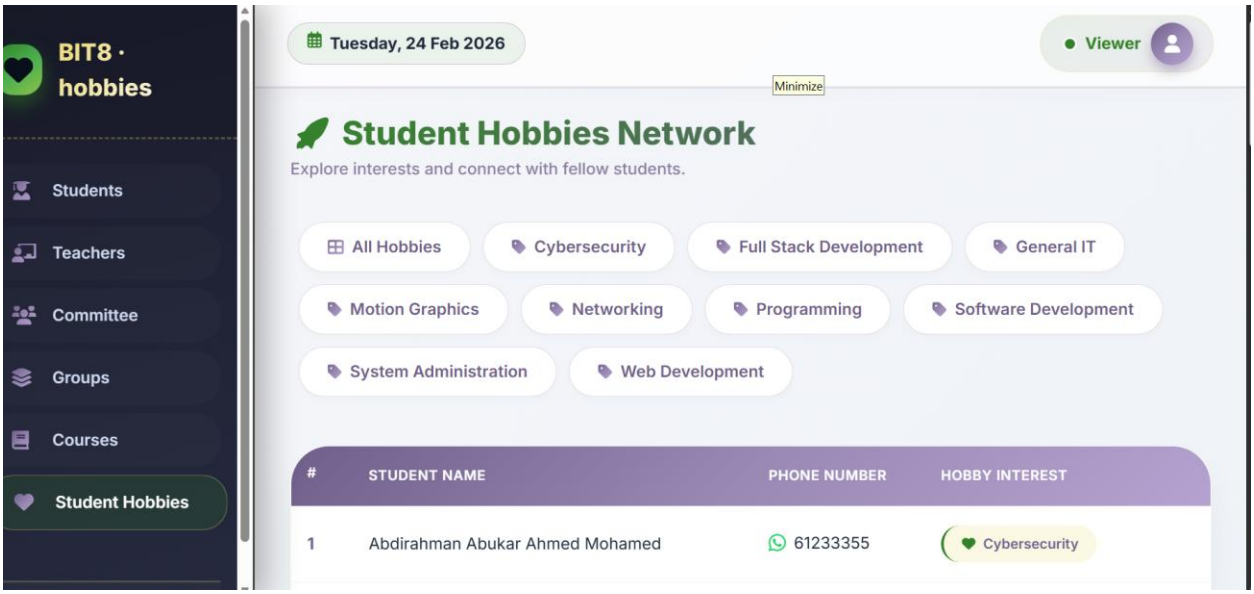
Courses section



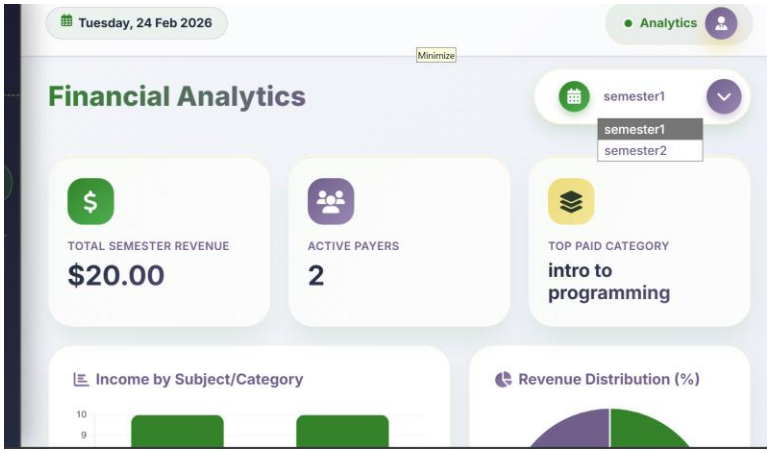


❖ **Hobies**

Finance section

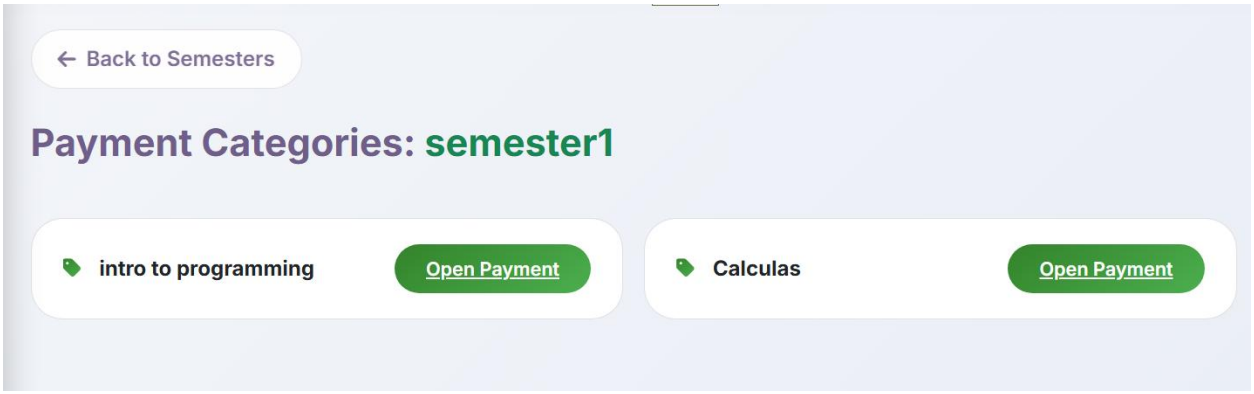


Reports section

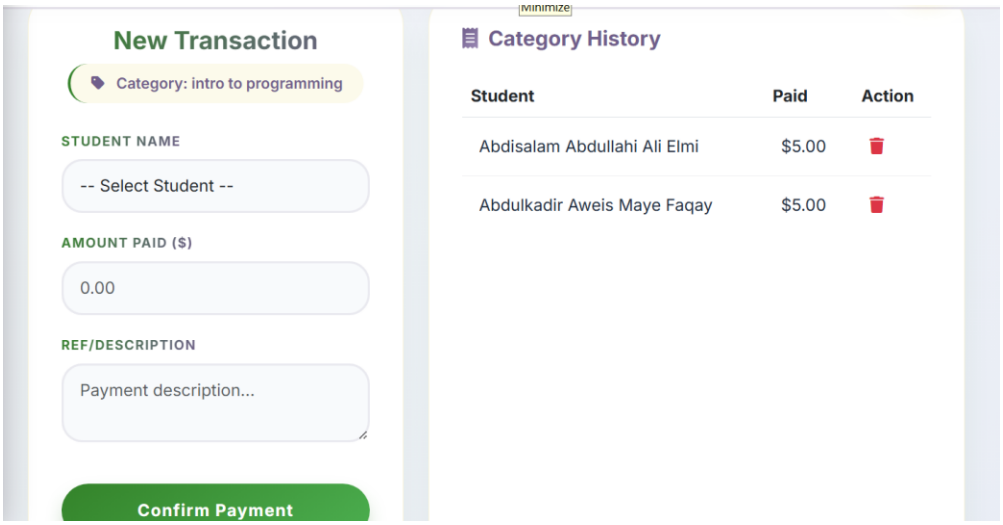


Finance section

After click manage finance



After click open payment



Css must follow

```
:root {
    --primary: #34832A;
    --primary-dark: #265f1e;
    --primary-light: #4CAF50;
    --secondary: #705E8B;
    --secondary-dark: #5a4b72;
    --secondary-light: #9B89B5;
    --accent: #EDDB7A;
    --accent-dark: #e5d15c;
    --accent-light: #f5e79a;

    --grad-1: linear-gradient(145deg, #34832A, #4CAF50);
    --grad-2: linear-gradient(135deg, #705E8B, #9B89B5);
    --grad-3: linear-gradient(125deg, #EDDB7A, #F5E79A);

    --grad-card-1: linear-gradient(115deg, rgba(52,131,42,0.1), rgba(114,94,139,0.05));

    --dark-bg: #0f1422;
    --glass-bg: rgba(255,255,255,0.85);
    --shadow-float: 0 30px 40px -20px rgba(0,0,0,0.25);
    --shadow-elegant: 0 15px 30px -10px rgba(52,131,42,0.15);
    --transition-bounce: all 0.3s cubic-bezier(0.34, 1.56, 0.64, 1);
    --transition-smooth: all 0.3s ease;
}
```