System Documentation

for

ABC System

Version <X.X>

Tutorial Section: *<place your Tutorial section here>*

Group No.: *<place your group number here>*

<name>	**<student #>**
<name>	**<student #>**
<name>	**<student #>**
<name>	**<student #>**
<name>	**<student #>**

Date: **<place the date of submission here>**

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
SRS in Part 1(as Ver 1.0) SDS in Part 2(as Ver 2.0.X) *System Documentation in Part 3 (as Ver 3.0) Draft Type and Number	Full Name	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	00/00/00

1 Project Management

1.1 Team Members

<*TO DO: List down the team members and their assigned actor/processes*>

Name	Actor
LAW JUN FENG	RESEARCHER
LIM JIAN FENG	REVIEWER
SURAJ	HOD

1.2 Problem statement

<TO DO: State the problem as a short, clear explanation of an issue or challenge that sums up what you want to change.>

1.3 Project Plan

<TO DO: Place the project Gantt Chart>

2 System Overview

<Describe your personal responsibilities, tasks, and modules developed.>

2.1 Description

<*Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, will be effective.*
TO DO: Describe the major processes to be performed by the system and the actors involved in each process.>

The Research Grant Management System (RGMS) automates the grant lifecycle through three distinct functional groups: Application, Assessment, and Administration. The system streamlines the flow of data from the initial research proposal to the final funding decision and financial tracking.

Actors	Major Processes
Researcher (Law Jun Feng)	**1. Proposal Submission & Management:** Allows researchers to draft, edit, and submit grant applications, including uploading necessary PDF documentation. It also handles version control to track changes. **2. Post-Award Tracking:** Enables successful applicants to monitor their fund utilization (burn rate) and submit mandatory milestone progress reports to the department.
Reviewer (Lim Jian Feng)	**1. Merit-Based Assessment:** Facilitates the qualitative and quantitative evaluation of proposals. Reviewers can access assigned documents and input scores based on defined rubrics. **2. Evaluation Workflow Management:** Automates the "Review" lifecycle, managing the transition of a proposal from "Pending Review" to "Complete" and dispatching it for final decision-making.
Head of Department (HOD) (Suraj A/L Prakash)	**1. Strategic Decision Making:** Provides the authority to approve or reject grants based on reviewer feedback and department goals. **2. Financial Governance:** Monitors the department's total liquidity, tracks real-time expenditure against the allocated budget, and visualizes key performance indicators (KPIs) through the dashboard.

2.2 Tasks

Team Member	Tasks
Law Jun Feng	**1. Database & Model Design:** Designed the `Proposal` and `ProgressReport` entities, including handling complex file paths for document storage. **2. Form Validation Logic:** Implemented Django forms to enforce business rules, such as preventing submission if the requested budget exceeds the maximum cap. **3. Frontend State Management:** Developed the logic for dynamic status badges (e.g., "Under Review," "Approved") that update automatically based on database changes.
Lim Jian Feng	**1. Relational Mapping:** Defined the `Evaluation` model with foreign keys to link specific `Proposals` to specific `Users`, ensuring data integrity. **2. Workflow Automation:** Created the backend logic that transitions a proposal's status to "Review Completed" immediately after a score is submitted. **3. UI/UX Development:** Designed the Reviewer Dashboard to provide a clear, distraction-free interface for reading proposals side-by-side with the scoring form.
Suraj A/L Prakash	**1. Financial Logic Implementation:** Coded the `BudgetManager` logic that deducts grant amounts from the master department fund only upon formal approval. **2. Analytics Integration:**

	Integrated `Chart.js` with Django to render real-time graphs for "Budget vs. Expenditure" and "Research Output." **3. Security & Locking:** Implemented the state freezing logic that prevents any further edits to a proposal once the final decision (Approve/Reject) is made.

2.3 Modules Developed

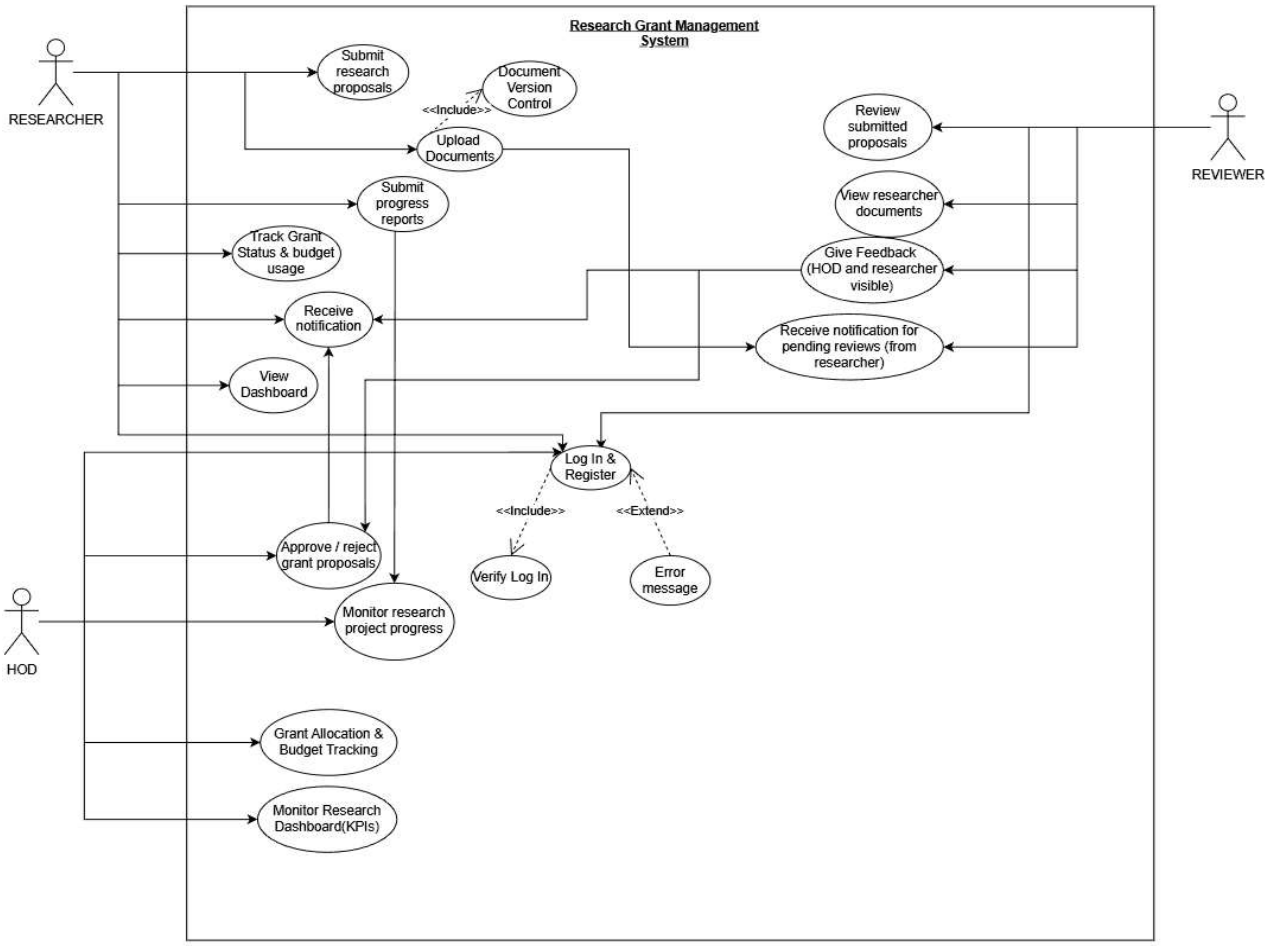
<*List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.*
TO DO: Provide a short list of some major assumptions that might significantly affect your design. For example, you can assume that your client will have 1, 2 or at most 50 Automated Banking Machines. Every number has a significant effect on the design of your system. >

Team member	Modules Developed	Assumptions
Law Jun Feng	**1. Proposal Manager:** Handles CRUD operations, file uploads, and versioning. **2. Grant Analytics Engine:** Calculates personal budget burn rates for researchers. **3. Notification Service:** Generates system alerts for deadlines and status updates.	**1. Software Dependency:** The PDF generation feature relies on the **xhtml2pdf** library; if incompatible, award letters cannot be generated. **2. Connectivity:** It is assumed the user has a stable internet to upload large PDF proposal files. **3. Browser Support:** The system assumes a modern browser (Chrome/Edge) with PDF viewing capabilities. submission.
Lim Jian Feng	**1. Evaluation Manager:** Processes scoring inputs, validates ranges (1-10), and saves feedback. **2. Assignment Engine:**	**1. Data Integrity:** Relies on **SQLite/PostgreSQL** foreign key constraints to prevent orphan evaluations.

	Filters database queries to display only proposals assigned to the current user. **3. HOD Dispatcher:** Triggers the status update to notify the HOD upon review completion.	**2. Reviewer Competency:** Assumes assigned Reviewers are subject matter experts and need no system guidance on *how* to grade. **3. Conflict of Interest:** Assumes administrative checks for conflicts are performed *before* assignment.
Suraj A/L Prakash	**1. Approval Manager:** Executes the final status update and triggers the award letter generation. **2. Financial Tracker:** Monitors total department funds and triggers "Critical Budget" alerts. **3. KPI Engine:** Aggregates system-wide data for visual reporting on the dashboard. **4. Dashboard Manager:** Aggregates system-wide data (budgets, pending approvals) to render the main HOD control panel.	**1. Third-Party Component:** The dashboard relies on **Chart.js** via CDN; charts will not render if the external service is down. **2. Budget Accuracy:** Assumes initial Department Budget data entered by admin is accurate. **3. Single Currency:** Assumes all transactions are in MYR without real-time exchange rate conversion.

2.4 Use Case Diagram

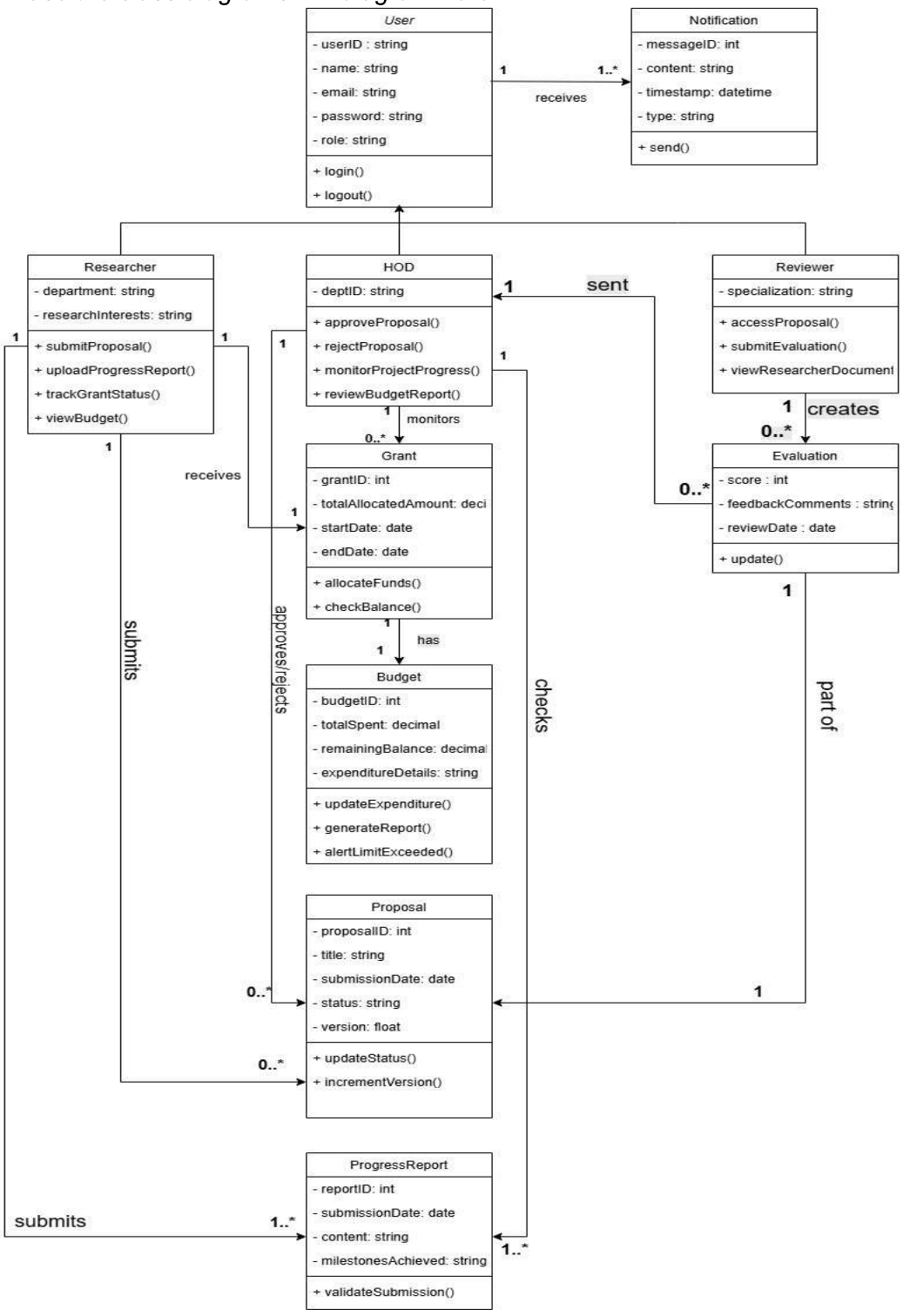
<TO DO: Place the use case diagram here.>



3 Requirements

3.1 Class Diagrams / ERD

<TO DO: Place the class diagram / ER diagram here.>



3.2 State Diagrams

<TO DO: Place the state diagram here.>

4 Design

4.1 Data Dictionary

<TO DO: Describe the data dictionary and place the table with the details here>

User Entity

Field Name	Data Type	Length	PK/FK	Description
userID	Integer	10	PK	Unique identifier for the user.
name	String	100	-	The title of the research project.
email	String	100	-	Academic or professional email address.
password	String	255	-	Encrypted login credentials
role	String	20	-	Identifies if the user is a Researcher, Reviewer, or HOD.

Proposal Entity

Field Name	Data Type	Length	PK/FK	Description
proposalID	Integer	-	PK	Unique identifier for the proposal.
title	String	255		The title of the research project.
submissionDate	Date			Date the proposal was submitted.
status	String	50		Tracks state (e.g., Pending, Under Review, Approved)
version	Float			Tracks document iterations for version control.

Grant Entity

Field Name	Data Type	Length	PK/FK	Description
grantID	Integer	-	PK	Unique identifier for the grant
totalAllocatedAmount	Decimal	12,2		The total funding amount authorized.

startDate	Date			Start date of the funding period.
endDate	Date	50		End date of the funding period.
proposalID	Integer		FK	Links the grant back to the original proposal.

Budget Entity

Field Name	Data Type	Length	PK/FK	Description
budgetID	Integer	-	PK	Unique identifier for the budget.
totalSpent	Decimal	12,2	-	The cumulative amount spent so far.
remainingBalance	Decimal	12,2	-	Calculated balance (Allocated - Spent).
expenditureDetails	String	1000	-	Text log of project expenses.
grantID	Integer	-	FK	Links the budget to the specific Grant

ProgressReport Entity

Field Name	Data Type	Length	PK/FK	Description
reportID	Integer	-	PK	Unique identifier for the report.
submissionDate	Decimal	-	-	The date the report was filed.
content	Decimal	1000	-	Summary of recent project progress.
milestonesAchieved	String	500	-	Specific goals reached during this period.
proposalID	Integer	-	FK	Links the report to the parent proposal.

Researcher Entity

Field Name	Data Type	Length	PK/FK	Description
userID	Integer	10	PK/FK	Inherited primary key from User.
department	string	100	-	The faculty or department the researcher belongs to

researchinterests	string	255	-	Specific fields of study for the researcher.

Reviewer Entity

Field Name	Data Type	Length	PK/FK	Description
userID	Integer	10	PK/FK	Inherited primary key from User.
specialization	string	100	-	The reviewer's area of expert knowledge.

HOD Entity

Field Name	Data Type	Length	PK/FK	Description
userID	Integer	10	PK/FK	Inherited primary key from User.
deptID	string	100	-	Unique identifier for the department they lead.

Evaluation Entity

Field Name	Data Type	Length	PK/FK	Description
score	Integer	100	PK	Numerical rating assigned by the Reviewer.
feedbackComments	string	1000	-	Qualitative remarks regarding the proposal.
reviewDate	date	1000	-	The date the evaluation was submitted.

Notification Entity

Field Name	Data Type	Length	PK/FK	Description
messageID	Integer	-	PK	Unique identifier for the notification.
content	string	500	-	The text body of the notification message.
timestamp	datetime	-	-	The exact time the notification was

				sent.
type	string	20	-	Category of alert (e.g., Email, System Alert).

4.2 Software Architecture

<TO DO: Describe the software architecture and place the architecture diagram here.>

<TO DO: Describe the separation of the system into subsystems and how the subsystems are assigned to team members.>

Subsystem	Team members
Subsystem 1 (Researcher Management)	Law Jun Feng
Subsystem 2 (Reviewer Management)	Lim Jian Feng
Subsystem 3 (Hod Administration)	Suraj A/L Prakash

4.2.1 Subsystem 1

<TO DO: Describe the subsystem and place the architecture diagram here.>
This **subsystem** encapsulates all workflows initiated by the **Researcher** actor. It is designed to handle the **"Entry"** and **"Monitoring"** phases of the grant lifecycle. The subsystem follows the application's three-tier structure, dividing responsibilities between the User Interface (UI) and the underlying Logic Modules.
Key Functional Modules:

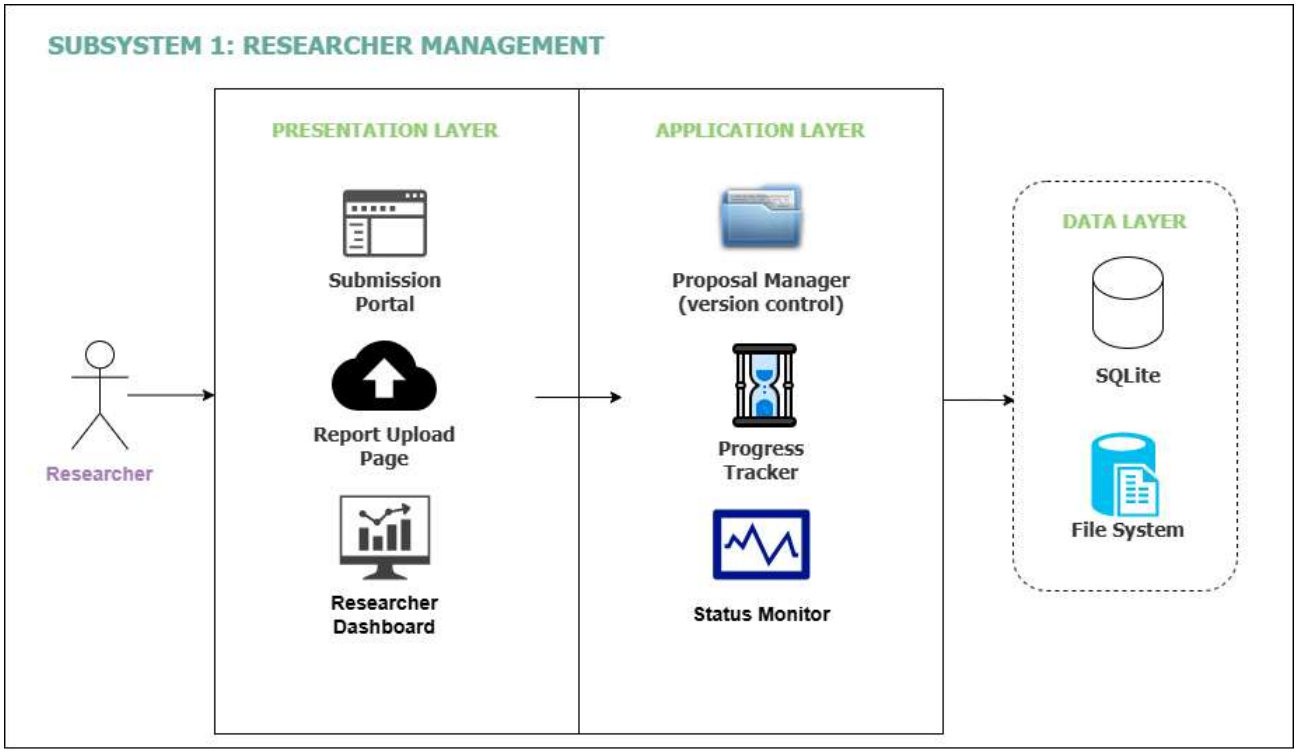
- **1. Proposal Manager (Logic):** This is the core input module for the subsystem. It processes data received from the **Submission Portal UI**. Its primary responsibility is to handle the creation of new grant proposals and manage **Document Version Control**. When a researcher updates a proposal, this manager checks existing records in the database and automatically increments the version number (e.g., from 1.0 to 1.1) to ensure data integrity is preserved.

- **2. Progress Tracker (Logic):** This module manages the post-award phase. It interfaces with the **Report Upload Page** to allow **researchers** to submit milestone updates. It validates that the report is attached to an active grant before saving the **ProgressReport** entity to the **database.**

- **3. Status Monitor (Logic):** This module powers the **Researcher Dashboard**. It is a read-only retrieval engine that aggregates data from multiple sources to provide a "single pane of glass" view. It queries the **Notification System** for unread alerts

and the **Budget** table to calculate and display real-time financial metrics (Total Allocated vs. Total Spent).

Data Flow: Interaction begins when the **Researcher** inputs data via the **Presentation Layer**. The specific Logic Module (e.g., Proposal Manager) validates this input and commits it to the **Data Layer**. Conversely, when the **Researcher** requests to view their status, the **Status Monitor** retrieves the relevant records from the **Database** and passes them back to the **UI** for display.



4.2.2 Subsystem 2

<TO DO: Describe the subsystem and place the architecture diagram here.>

This **subsystem** encapsulates the specialized workflows performed by th**e Reviewer actor** within the Research Grant Management System. It is responsible for the systematic assessment of submitted proposals and the hand-off of recommendations to the Head of Department (HOD). Like the Researcher's subsystem, this follows a three-tier architecture to separate evaluation interfaces from backend logic.

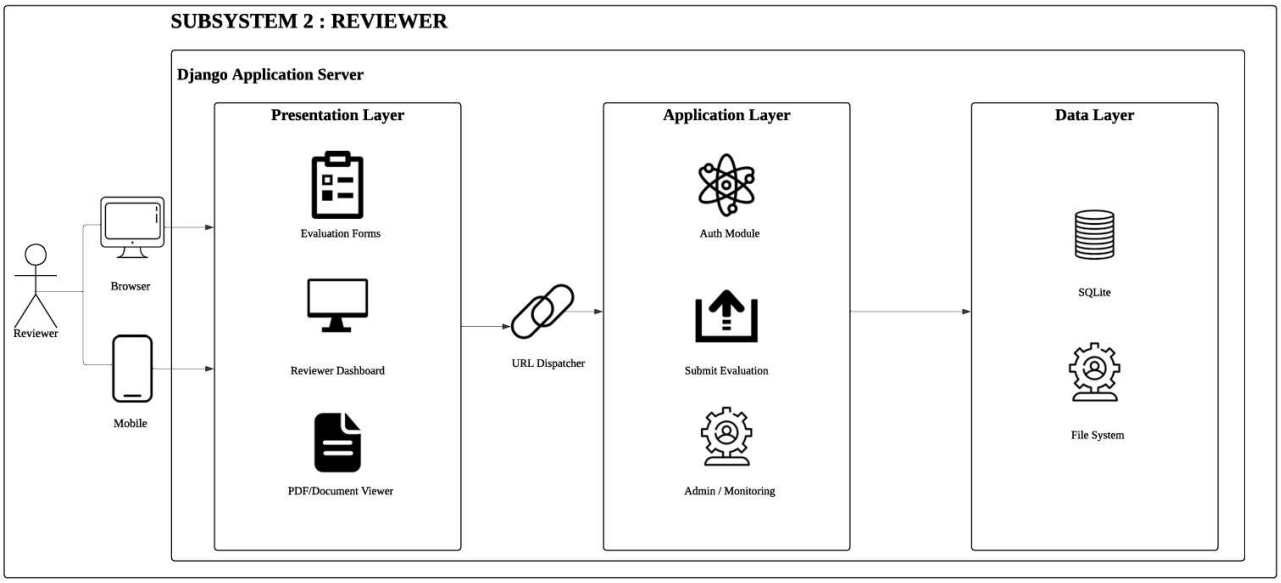
Key Functional Modules:

1. **Evaluation Manager (Logic)**: This is the core engine for the review process. It processes inputs from the Evaluation Forms in the Presentation Layer. Its primary

responsibility is to store quantitative scores and qualitative feedback in the Database. It ensures that an evaluation is "locked" once submitted to maintain the integrity of the review.

2. **Assignment Engine (Logic)**: This module powers the Reviewer Dashboard. It filters the Database to display only the specific grant proposals assigned to the logged-in Reviewer. It works closely with the Auth Module to verify that a user has the correct "Reviewer" permissions before granting access to sensitive proposal documents.
3. **HOD Dispatcher (Logic):** This module handles the "Submit Evaluation" workflow. Once a Reviewer completes their assessment, the Dispatcher updates the proposal's status (e.g., from "Under Review" to "Review Completed"). It packages the reviewer's recommendation and triggers a notification to the HOD's dashboard, ensuring a seamless transition between the two roles.

Data Flow: The workflow begins when the Reviewer accesses the Reviewer Dashboard via a Browser or Mobile device. The Assignment Engine retrieves pending proposals from the Data Layer and displays them. When the Reviewer submits an assessment via the Evaluation Forms, the Evaluation Manager validates the data and saves it to the Database. Finally, the HOD Dispatcher updates the proposal status, making the evaluation results visible to the HOD for final decision-making.



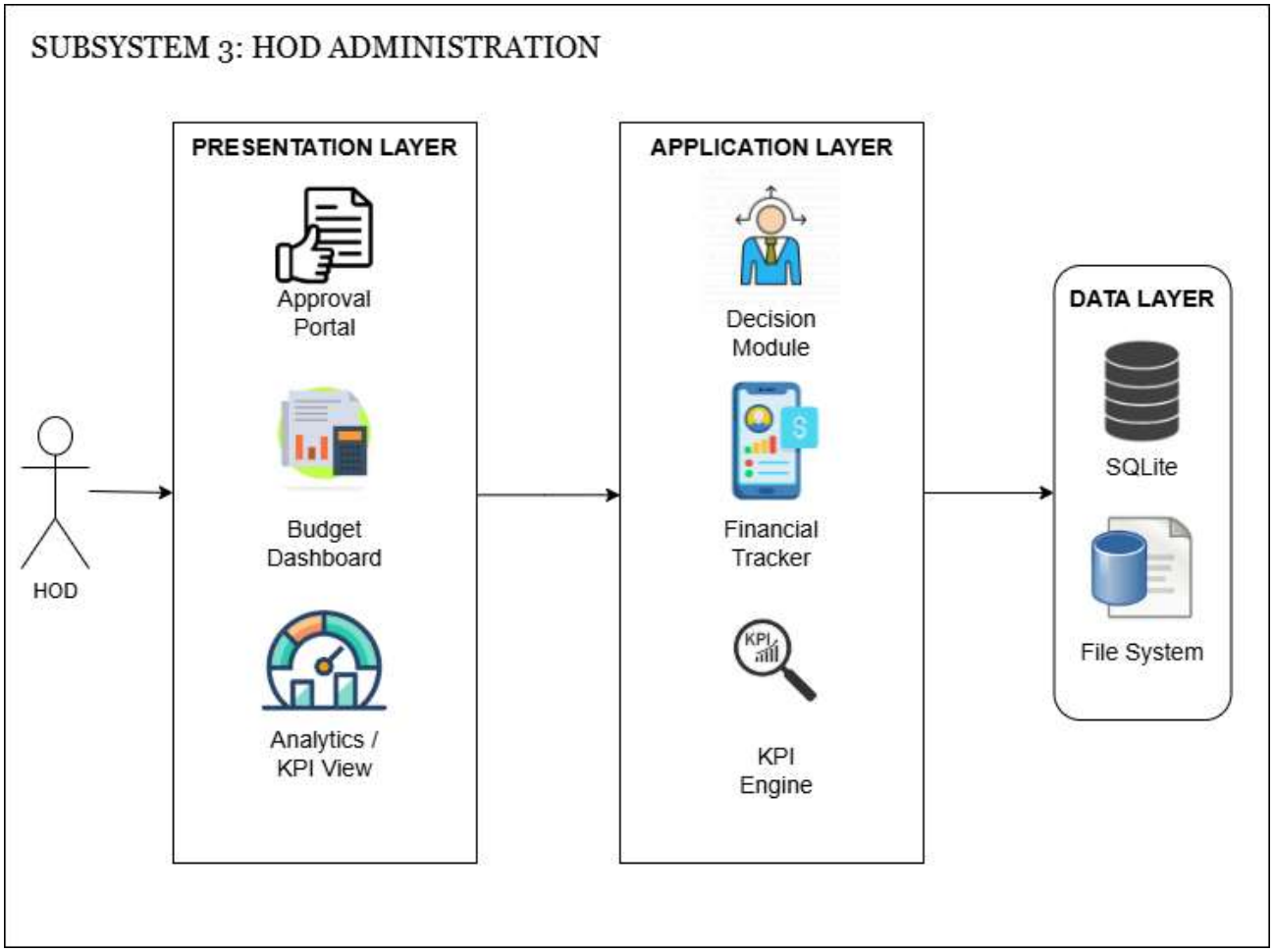
4.2.3 Subsystem 3

This **subsystem** encapsulates all workflows initiated by the **HOD (Head of Department)** actor. It is designed to handle the **"Governance"**, **"Financial Oversight"**, and **"Strategic Monitoring"** phases of the grant lifecycle. The subsystem follows the application's three-tier structure, dividing responsibilities between the administrative User Interface (UI) and the underlying Logic Modules.

Key Functional Modules:

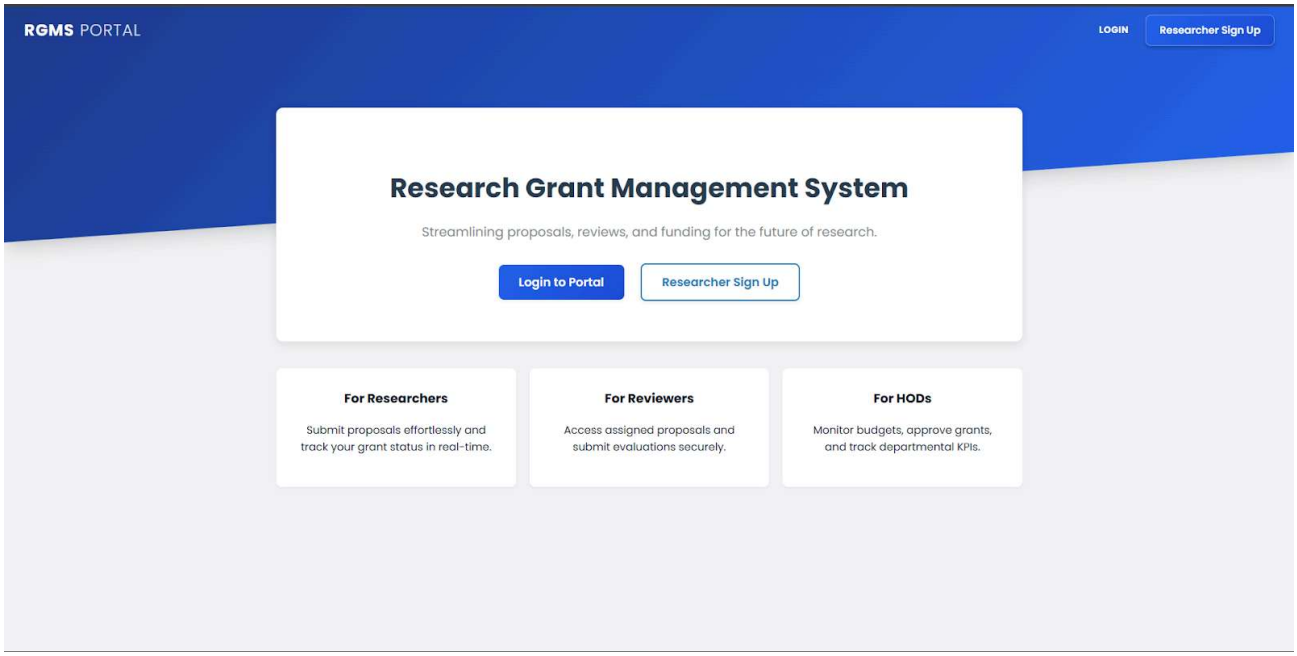
1. **Decision Module (Logic):** This is the primary control module for grant initiation. It processes requests from the **Approval Portal UI**. Its primary responsibility is to execute the "Approve/Reject" logic for new proposals. When an HOD approves a proposal, this module triggers the creation of the **Grant** entity and ensures the initial funding status is strictly validated against available department resources.
2. **Financial Tracker (Logic):** This module manages the ongoing fiscal health of active grants. It interfaces with the **Budget Dashboard** to allow the HOD to adjust allocations and monitor spending. It contains the specific business rule for the **"90% Threshold Alert"**, automatically flagging projects that are nearing budget exhaustion and preventing unauthorized overspending before committing updates to the **Database**.
3. **KPI & Analytics Engine (Logic):** This module powers the **Department Analytics Dashboard**. It is a high-level aggregation engine that queries multiple tables (Proposals, Progress Reports, Budgets) to calculate performance metrics. It handles the data processing for "Burn Rates" and "Completion Percentages" and manages the **Export** function to generate PDF/Excel reports for external stakeholders.

Data Flow: Interaction begins when the **HOD** inputs a decision or request via the **Presentation Layer**. The specific Logic Module (e.g., Decision Module) validates this input (checking funds or permissions) and commits the transaction to the **Data Layer**. Conversely, when the HOD requests an analytics report, the **KPI Engine** aggregates the relevant records from the **Database**, processes the metrics, and passes them back to the **UI** for visualization.

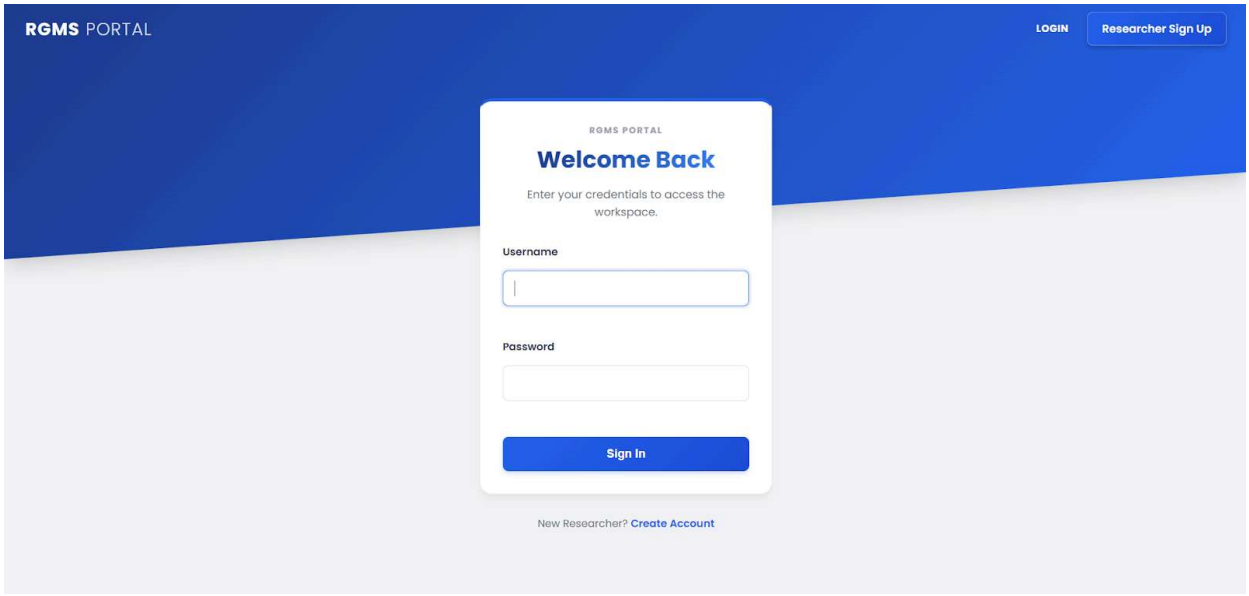


4.3 Main Screens

<TO DO: Describe the main screens of the system and place the screen designs here.>
The image below shows the **landing page** all users will first land on. From here users can either navigate to the login **portal** or **sign up** as **researcher.**



The image below shows the **login page** for all users. Users who already have an account can login from here while users who have not signed up, can proceed to **sign up.**



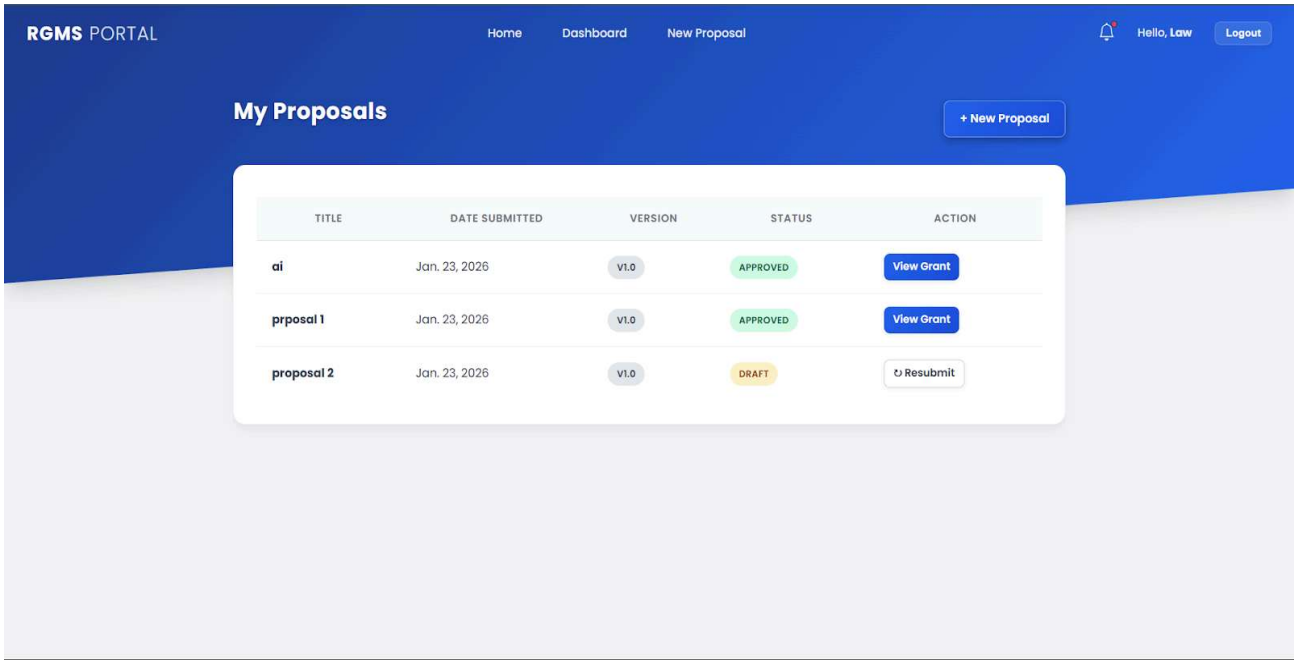
4.4 Subsystem 1 Screens: Researcher Subsystem

<TO DO: Describe the screens of subsystem 1 and place the screen designs here.>

4.4.1 Researcher Dashboard

Researcher Dashboard The primary landing page that aggregates the Researcher's activities. The interface features a prominent Hero section with a skewed geometric blue background.

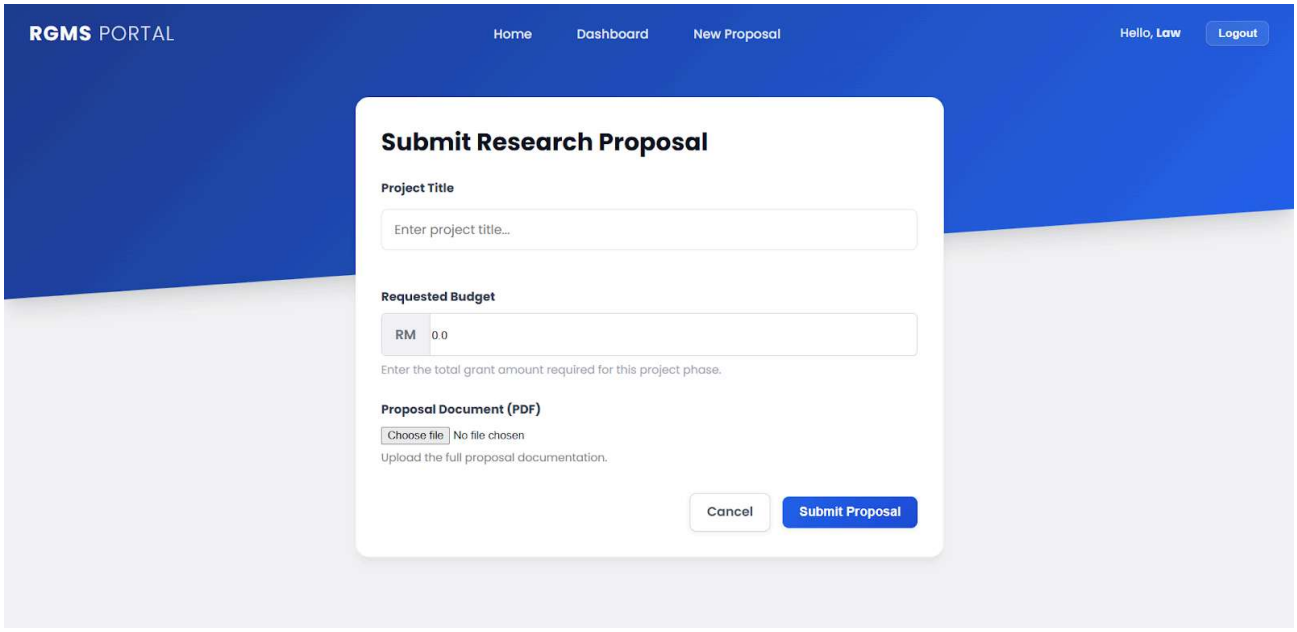
- **KPI Grid:** Uses a 4-column CSS grid (.info-grid) to display high-level metrics (e.g., Active Grants, Pending Proposals).
- **Summary Table:** A clean, spaced table layout listing recent proposals with color-coded status badges (.badge-success for Approved, .badge-warning for Pending).



4.4.2 Submit Proposal or Resubmit Proposal

Submit Proposal Form A standardized data-entry interface designed for minimal friction.

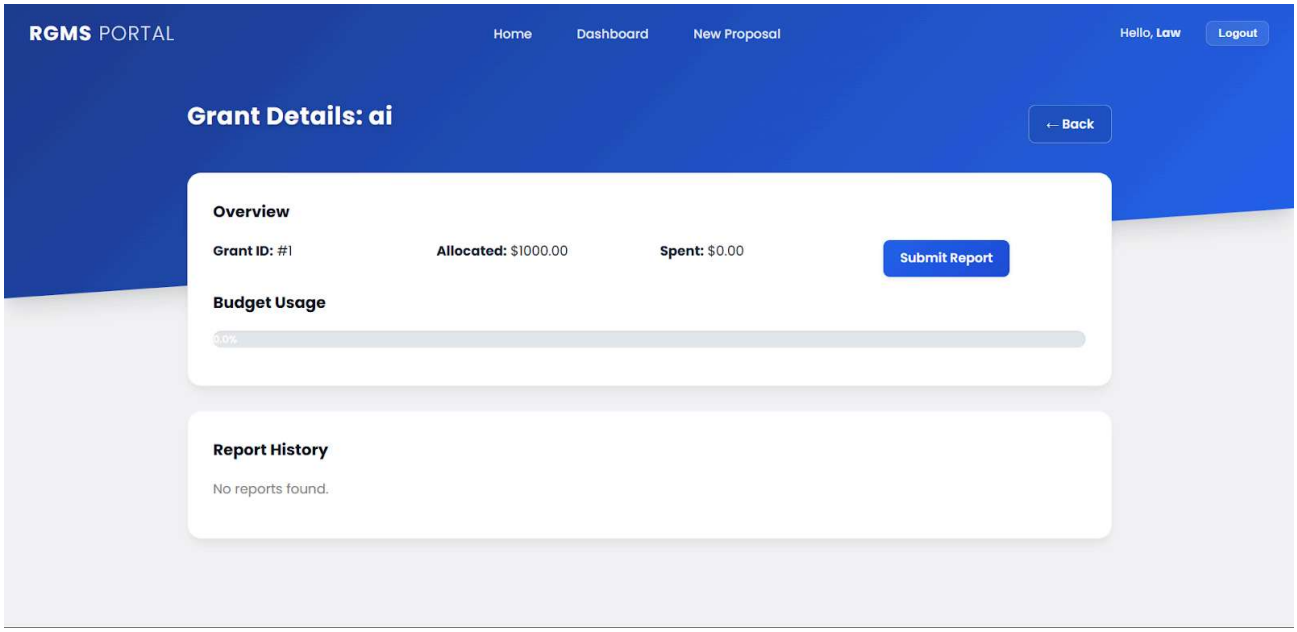
- **Form Layout**: Clean form groups with subtle focus-state borders that guide the user's attention.
- **Document Upload:** Includes a dedicated file input field for the Research Proposal PDF.
- **Action Prompts**: A prominent, gradient-styled Primary Button confirms the submission.



4.4.3 Grant Details & Financial View

Grant Details & Financial View A read-only information hub for an approved grant. This interface uses the custom .card component to organize complex data.

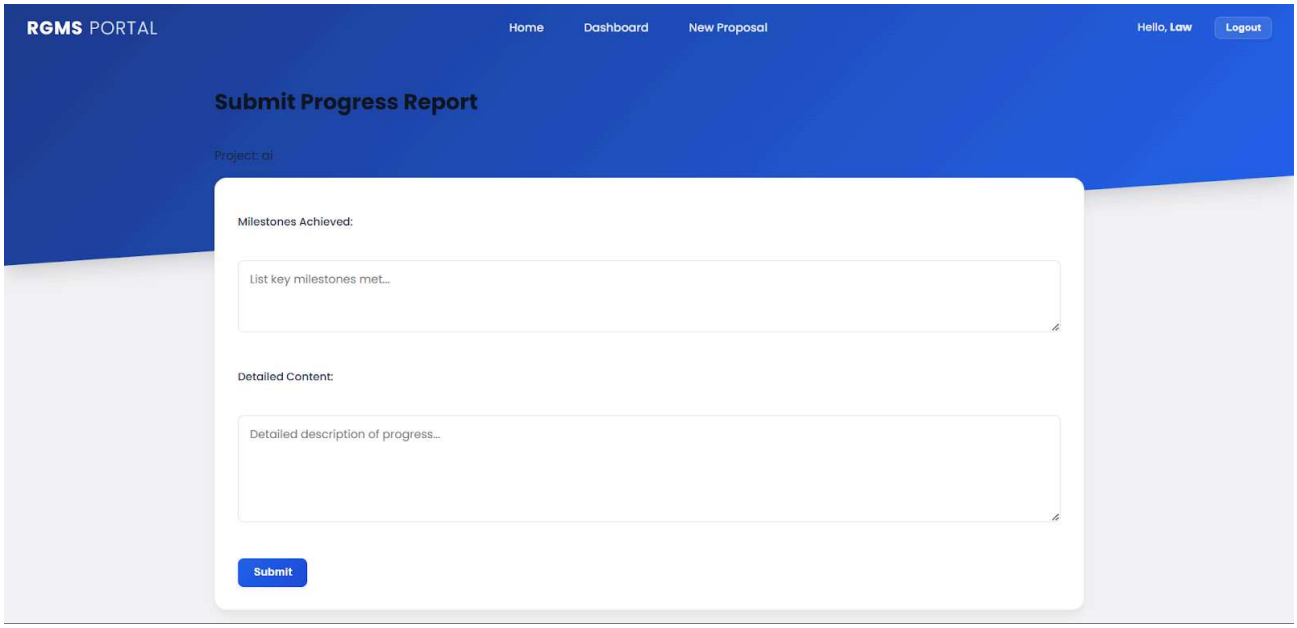
- **Budget Tracker:** Integrates native HTML5/CSS progress bars (.progress-fill) that dynamically display budget usage. Bars change color (Green, Yellow, Red) based on expenditure thresholds.
- **Expenditure Log:** A tabular view detailing date, item description, and receipt proofs for individual expenses.



4.4.4 Submit Progress Report

Submit Progress Report Form An interface tailored for qualitative data collection, allowing the researcher to update the department on their research milestones.

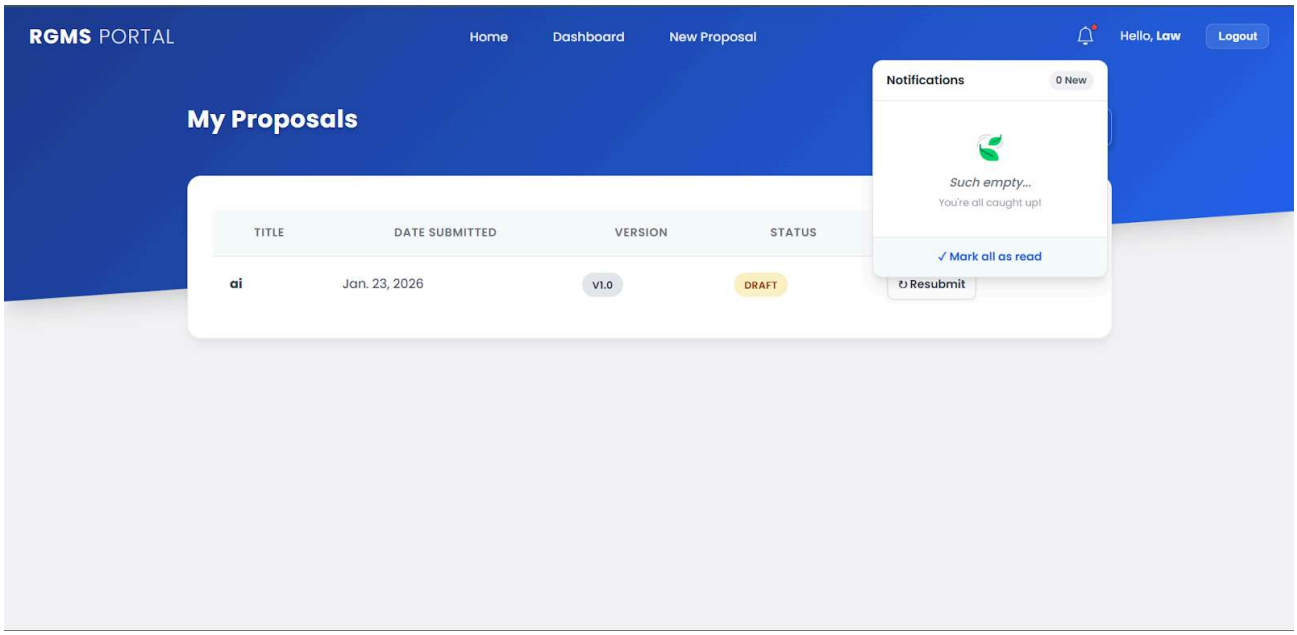
- **Narrative Input:** Features large, resizable text areas (.custom-textarea) for detailed milestone descriptions.
- **Context Preservation:** Displays the Grant Title and current timeline at the top of the card so the researcher retains project context while writing.



4.4.5 Notification System

Notification System Integrated directly into the navigation bar, the Notification Bell acts as the central hub for grant updates (e.g., "Proposal Approved", "Report Reviewed").

- **Visual Cue:** A custom red indicator dot (notif-badge) alerts the researcher to unread system messages.
- **Interactive Dropdown:** Clicking the bell reveals a floating, card-styled menu powered by pure CSS animations.
- **Empty State:** When no notifications are present, the system displays an illustrated "Such empty... " state, reducing visual clutter and confirming to the user that they are caught up.



4.5 Subsystem 2 Screens: Reviewer Subsystem

<TO DO: Describe the screens of subsystem 2 and place the screen designs here.>

Interface Description: Reviewer Dashboard

Screen Name: Reviewer Dashboard

User Role: Reviewer

Purpose: To provide a streamlined list of all research proposals assigned to the reviewer for evaluation.

Key Interface Components:

1. **Hero Header:** A blue-themed header featuring the "RGMS Portal" branding and a personalized welcome message (e.g., "Welcome, ljf") to confirm the user session.

2. **Proposal Evaluation Table:**

   Data Columns: Displays essential metadata including Proposal ID, Title, and Researcher Name.

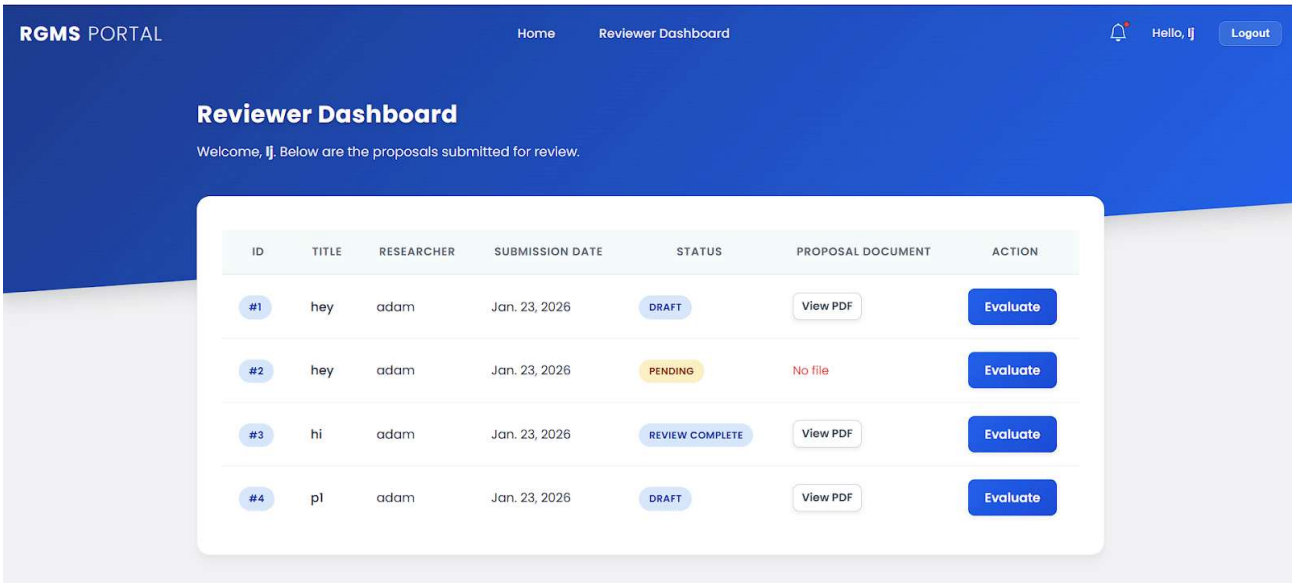
   Submission Tracking: Shows the Submission Date (e.g., Jan 23, 2026) and current Status (e.g., "Draft") to help reviewers prioritize their workload.

3. **Action Hub:**

   View PDF: A primary blue button that allows the reviewer to open and read the proposal document without leaving the application environment.

   Evaluate: A secondary action button that triggers the transition to the formal assessment form

4. **Empty State Logic:** When no tasks are assigned, the table displays a "No proposals available for review" message, preventing UI clutter.



Interface Description: Proposal Evaluation Form

Screen Name: Evaluate Proposal Screen

User Role: Reviewer

Purpose: To capture the reviewer's quantitative score and qualitative feedback for a specific research submission.

Key Interface Components:

1. **Contextual Header:** The screen title dynamically updates to include the proposal title (e.g., "Evaluate Proposal: hey"), ensuring the reviewer knows exactly which project they are scoring.

2. **Researcher Attribution:** Clearly displays the "Submitted by" field to maintain transparency during the review process.

3. **Assessment Inputs:**

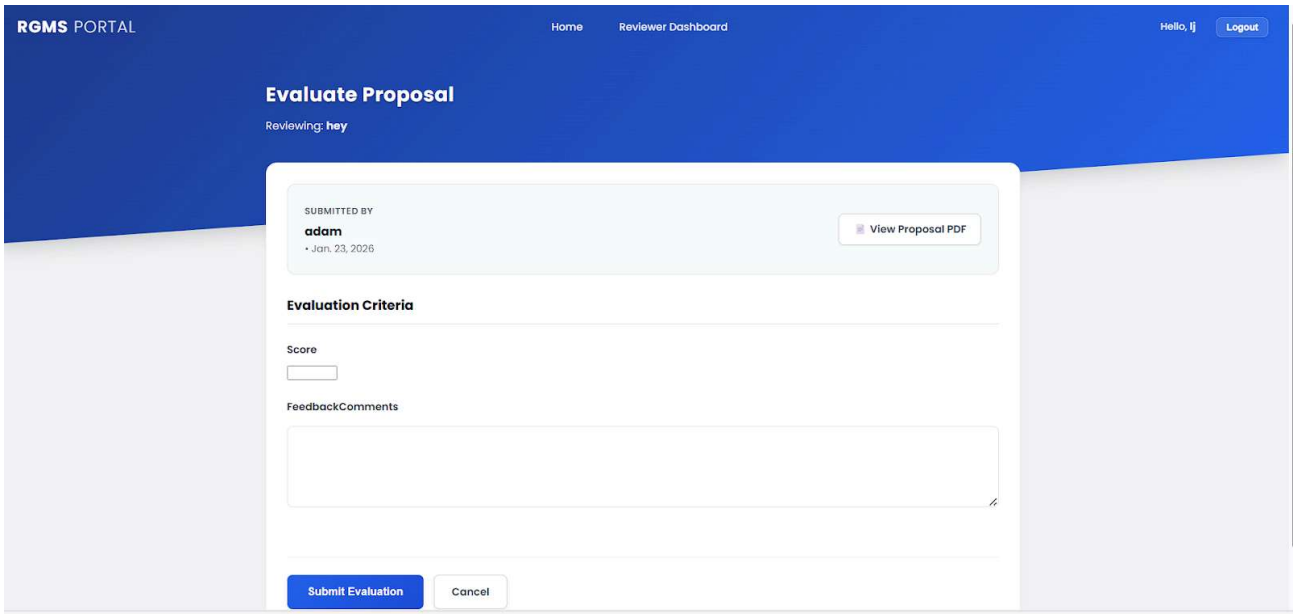
   Score Field: A dedicated numeric input box for the reviewer to provide a quantitative rating based on institutional rubrics.

   Feedback/Comments Box: A large, expandable textarea labeled "FeedbackComments" designed for detailed peer-review notes, critiques, and suggestions for the researcher.

4. **Form Submission Controls:**

   Submit Evaluation: A high-visibility blue button to finalize the review and commit the data to the system.

   Cancel: A neutral button that allows the user to exit the form and return to the dashboard without saving changes, serving as a safety mechanism against accidental edits.



4.6 Subsystem 3 Screens: Screens 9 (HOD)

4.6.1 Interface Description: HOD Management Dashboard

Screen Name: HOD Dashboard

User Role: Head of Department (HOD)

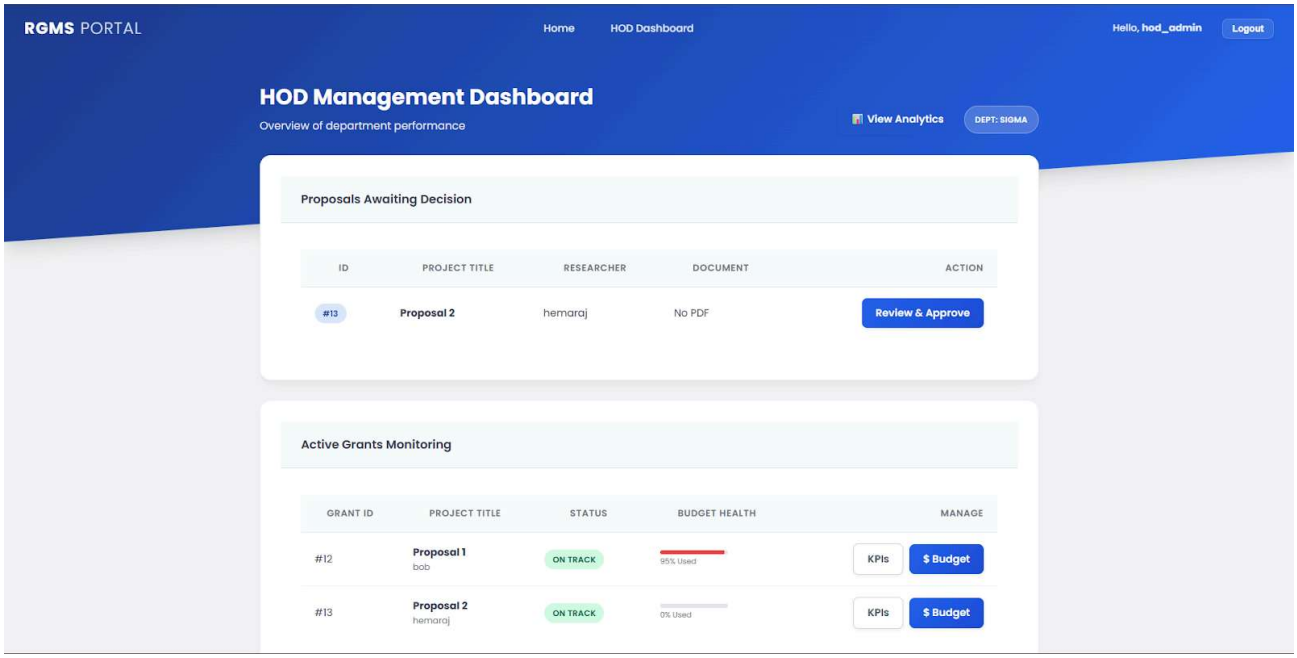
Purpose: To provide a centralized overview of departmental research activities, allowing the HOD to manage incoming proposals and monitor the financial and operational health of active grants.

Key Interface Components:

1. **Hero Header & Global Controls:**
   - **Department Identity:** The top section clearly identifies the user's jurisdiction with a specific badge (e.g., "DEPT: SIGMA"), ensuring the HOD knows which department's data is being viewed.

- **Analytics Access:** A prominent **"View Analytics"** button is positioned in the header, providing single-click access to the graphical reports (Budget Utilization Doughnut Chart & Success Rate Bar Chart).
- **Navigation:** Consistent top-bar navigation allows quick switching between Home and the Dashboard, with a user profile and logout option.

2. **Pending Proposals Section (Action Queue):**
   - This card displays a tabular list of **"Proposals Awaiting Decision"**.
   - **Data Columns:** Displays critical info including Proposal ID , Project Title , Researcher Name , and Document availability.
   - **Primary Interaction:** The **"Review & Approve"** button directs the HOD to the detailed approval form for that specific proposal.
3. **Active Grants Monitoring (Project Oversight):**
   - A comprehensive table tracking **"Active Grants Monitoring"** for projects that have already been approved.
   - **Status Indicators:** Uses color-coded badges (e.g., Green for "ON TRACK") to give an immediate visual status of project health.
   - **Visual Budget Health:** Features a dynamic **Progress Bar** under the "Budget Health" column.
     - **Logic:** The bar visualizes the percentage of funds used. It automatically changes color to **Red** (Critical) when usage exceeds 90% (as seen in the "95% Used" example), alerting the HOD to potential overspending without needing to click into the details.
   - **Management Actions:** Provides two distinct action buttons for every grant:
     - **KPIs:** To review project milestones and timelines.
     - **$ Budget:** To view the financial ledger and perform fund top-ups



4.6.2 Interface Description: Approve and Grant Allocation

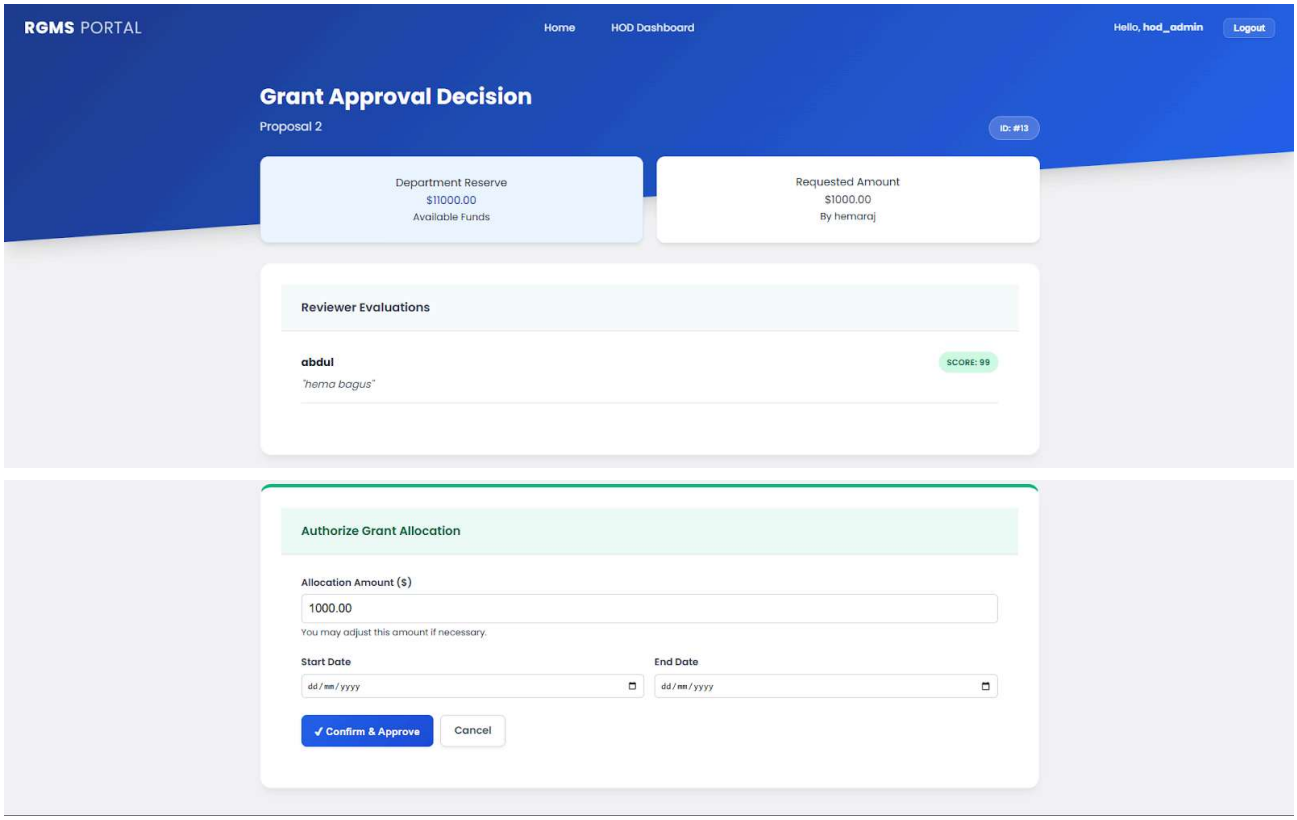
Screen Name: Grant Approval Decision

User Role: Head of Department (HOD)

Purpose: To enable the HOD to review reviewer evaluations, assess budget feasibility, and formally authorize the allocation of funds for a specific research proposal.

Key Interface Components:

1. **Financial Context Header (Top Grid):**
    - **Purpose:** To provide immediate financial awareness before a decision is made.
    - **Component:** Two prominent "KPI Cards" displayed side-by-side.
        - **Department Reserve:** Displays the total available funds (e.g., "$11,000.00") in the department's account.
        - **Requested Amount:** Displays the specific amount requested by the researcher (e.g., "$1,000.00").
    - **Design Logic:** Placing these figures adjacent to each other allows the HOD to instantly verify if the department can afford the grant without navigating to a separate budget page.
2. **Evaluation Summary (Middle Section):**
    - **Section Title:** "Reviewer Evaluations".
    - **Content:** Aggregates qualitative and quantitative feedback from the review phase.
    - **Data Points:** Displays the reviewer's username (e.g., "abdul"), their specific comments (e.g., "hema bagus"), and the assigned **Score Badge** (e.g., "SCORE: 99").
    - **Visual Cue:** High scores are highlighted with green badges to quickly signal quality proposals.
3. **Authorization Form (Bottom/Action Section):**
    - **Section Title:** "Authorize Grant Allocation".
    - **Visual Distinction:** This card features a distinct **Green Top Border** to visually signify that this is the "Action Zone" for final approval.
    - **Editable Allocation:** The "Allocation Amount ($)" field is pre-filled with the requested amount ($1000.00) but remains editable, giving the HOD the flexibility to approve partial funding if necessary.
    - **Timeline Controls:** Mandatory "Start Date" and "End Date" pickers to define the grant's active duration.
    - **Primary Action:** A **"Confirm & Approve"** button that commits the transaction and creates the grant record.



4.6.3 Interface Description: Project KPI & Monitoring Review

Screen Name: Project Monitoring Detail (project_monitoring_detail.html)

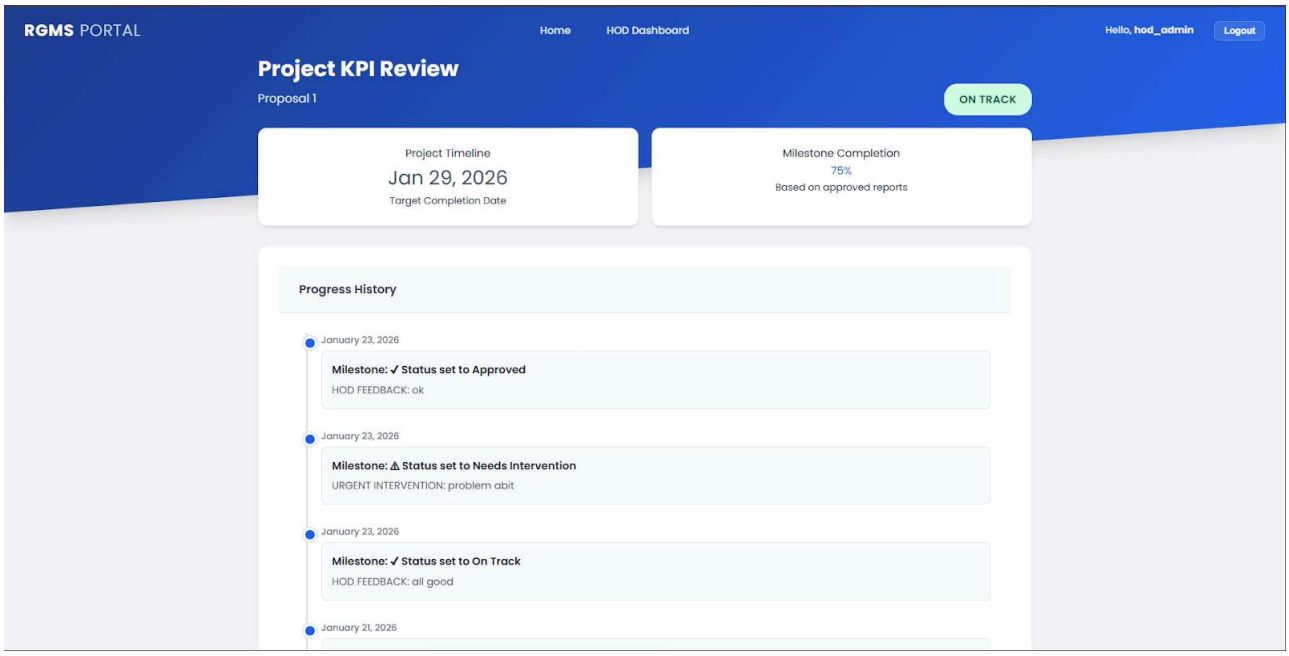
User Role: Head of Department (HOD)

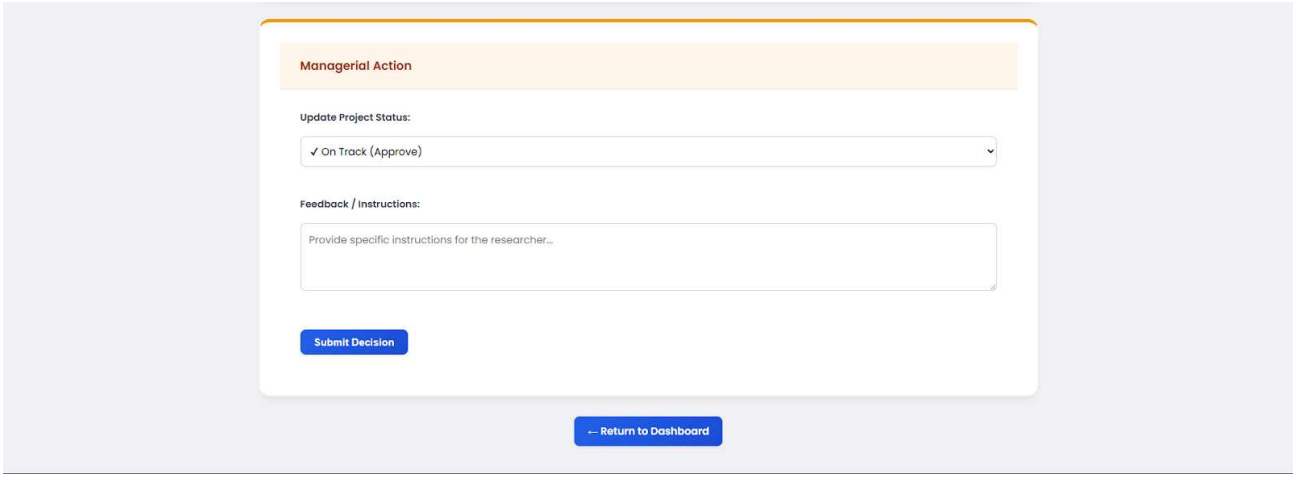
Purpose: To provide a detailed historical and real-time view of a specific grant's progress, allowing the HOD to review milestone achievements and issue managerial directives.

Key Interface Components:

1. **Project Status Header:**
    - **Project Identity:** Clearly displays the project title (e.g., "Proposal 1") and the current status badge (e.g., "ON TRACK").
    - **Visual Context:** Uses the standard HOD blue hero header for consistency, ensuring the user knows they are in a management view.
2. **Performance Metrics (KPI Grid):**
    - **Timeline Tracking:** Displays the "Project Timeline" with the target completion date (e.g., "Jan 29, 2026"), helping the HOD assess if the researcher is behind schedule.
    - **Completion Metric:** A large numeric indicator for "Milestone Completion" (e.g., "75%"). This provides an instant quantitative measure of progress based on approved reports.
3. **Progress History (Vertical Timeline):**
    - **Component:** A chronological, vertical timeline on the left side of the screen.
    - **Data Points:** Each entry acts as a historical log, displaying:

        - **Date:** When the report/action occurred (e.g., "January 23, 2026").
        - **Event:** The specific milestone or status change (e.g., "Milestone: Status set to Needs Intervention").
        - **Feedback/Context:** Displays the content of the report or the HOD's previous feedback (e.g., "URGENT INTERVENTION: problem abit").
    - **Design Logic:** This timeline format allows the HOD to trace the "story" of the project—seeing how it went from "On Track" to "Needs Intervention" and back again.
4. **Managerial Action Panel (Intervention Form):**
    - **Purpose:** To give the HOD direct control over the project's lifecycle.
    - **Visual Distinction:** This card uses an **Orange/Amber Header** ("Managerial Action") to differentiate it from the informational cards, signaling that this is a control panel.
    - **Status Control:** A dropdown menu allows the HOD to force a status update (e.g., "✓ On Track (Approve)" or "⚠ Needs Intervention").
    - **Feedback Loop:** A text area for "Feedback/Instructions" enables the HOD to send specific directives directly to the researcher.
    - **Primary Action:** The "Submit Decision" button commits these changes to the system.





4.6.4 Interface Description: Project KPI & Monitoring Review

Screen Name: Budget Detail View

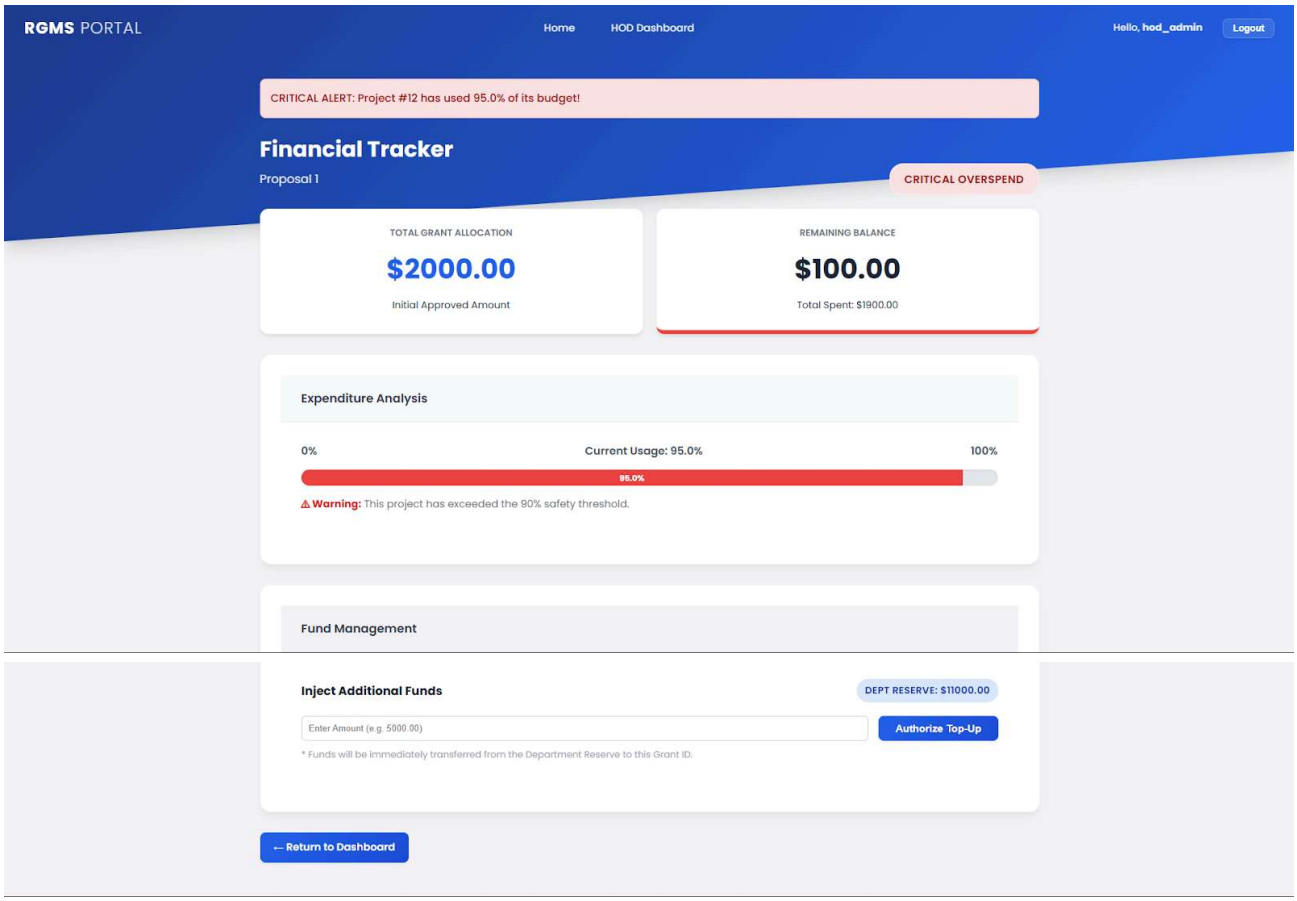
User Role: Head of Department (HOD)

Purpose: To provide granular oversight of a specific grant's financial health, alert the HOD to critical overspending, and enable emergency funding interventions (top-ups).

Key Interface Components:

1. **Critical Alert System:**
   - **Banner Alert:** If a project exceeds the 90% spending threshold, a prominent **Red Alert Banner** appears at the top ("CRITICAL ALERT: Project #12 has used 95.0% of its budget!").
   - **Visual Status:** A "CRITICAL OVERSPEND" badge in the header instantly signals that this project is in a danger zone.
2. **Financial KPI Cards (Grid Layout):**
   - **Total Grant Allocation:** Displays the original approved amount (e.g., "$2000.00").
   - **Remaining Balance:** Displays the live funds left (e.g., "$100.00").
   - **Visual Logic:** The "Remaining Balance" card dynamically applies a **Red Bottom Border** (class: border-critical) when funds are low, reinforcing the urgency visually.
3. **Expenditure Analysis (Progress Tracking):**
   - **Component:** A visual progress bar tracking "Current Usage" against a 0% - 100% scale.
   - **Logic:** The bar turns **Solid Red** when usage exceeds 90% (e.g., "95.0%").
   - **Diagnostic Text:** A warning message below the bar explicitly states the rule violation: "⚠ Warning: This project has exceeded the 90% safety threshold".
4. **Fund Management Panel (Action Zone):**
   - **Context:** Displays the "Dept Reserve" balance (e.g., "$11,000.00") so the HOD knows how much money is available to give.

- **Input Control:** A precise numeric input field ("Enter Amount") allows the HOD to specify the exact top-up value.
- **Primary Action:** The **"Authorize Top-Up"** button executes the transfer immediately.
- **Disclaimer:** Clear instructional text confirms that funds will be deducted from the Department Reserve.



4.6.5 Interface Description: Department Analytics & Performance Reports

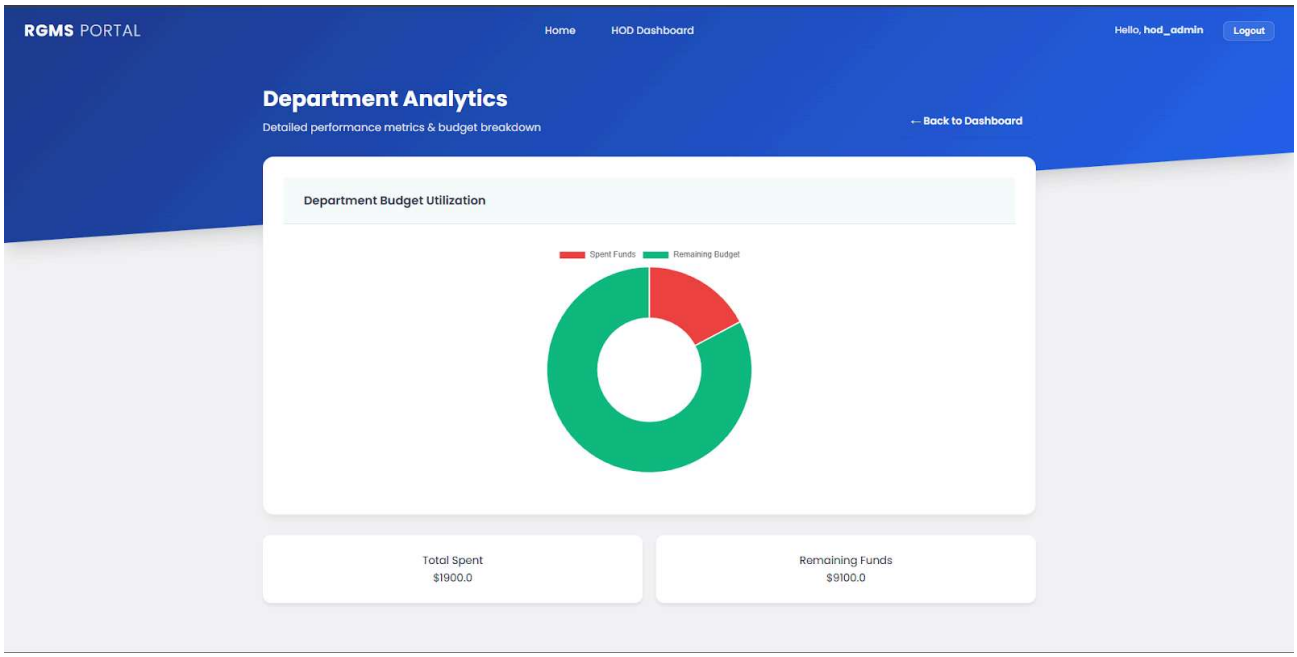
Screen Name: Department Analytics View

User Role: Head of Department (HOD)

Purpose: To provide high-level visual intelligence regarding the department's financial efficiency and proposal success rates, allowing for data-driven strategic planning.

Key Interface Components:

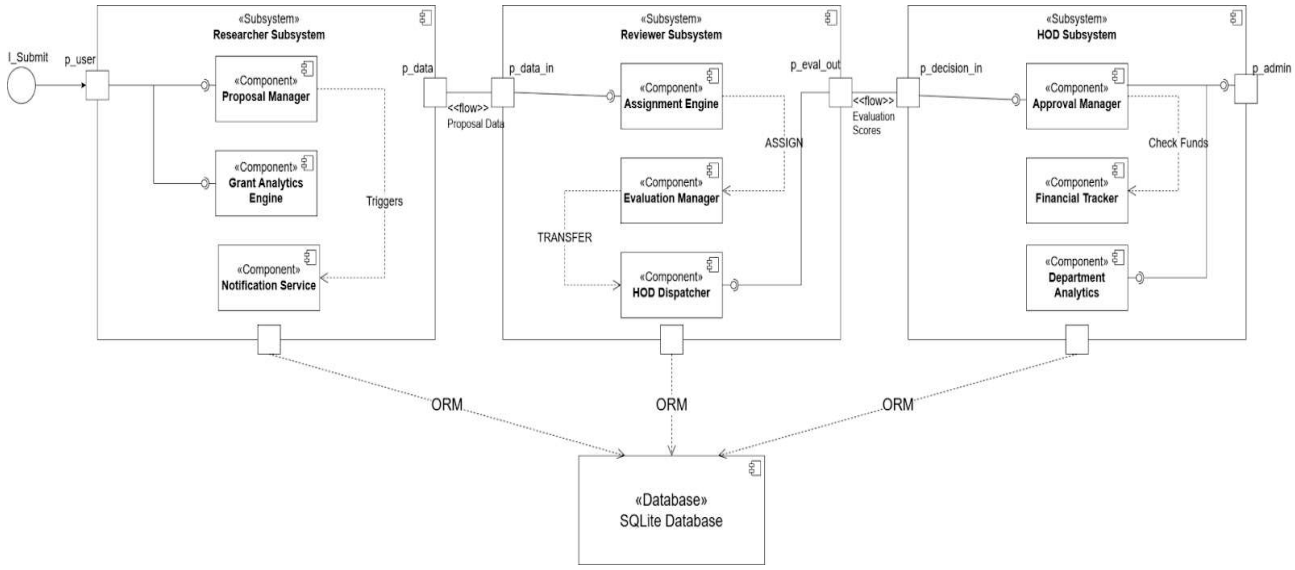
1. **Navigational Header:**
    - **Context:** The header clearly labels the page "Department Analytics" with a subtitle "Detailed performance metrics & budget breakdown," setting the expectation for statistical content.
    - **Back Navigation:** A prominent "← Back to Dashboard" button allows the HOD to quickly return to the main operational view after reviewing the data.
2. **Visual Analytics (Charts Section):**
    - **Budget Utilization Chart:** A Doughnut Chart visualizing the ratio of **"Spent Funds" (Red)** vs. **"Remaining Budget" (Green)**.
        - **Design Rationale:** The use of red/green color coding provides an immediate "traffic light" assessment of financial health. A large red segment warns of budget depletion at a glance.
    - **Proposal Success Metrics:** (Implicit in layout/code, though partially cut off in this specific crop) A Bar Chart comparing approved versus rejected proposals to track department research output quality.
3. **Summary KPI Cards (Bottom Grid):**
    - **Total Spent:** A dedicated card displaying the aggregate expenditure across all active grants (e.g., "$1900.0"). The text is color-coded **Red** to signify money flowing out.
    - **Remaining Funds:** A card displaying the available balance (e.g., "$9100.0"). The text is color-coded **Green** to signify available resources.



4.6 Main Components

<TO DO: Describe the main components (modules, classes, packages, etc.) and the table with the components and related subsystems here.>

Component Diagram



1. Description of Main Components

The Research Grant Management System (RGMS) is structured into three distinct subsystems (Packages) that communicate via a shared repository architecture (SQLite). Each subsystem encapsulates specific business logic modules.

A. Researcher Subsystem

This package handles the initiation of the grant lifecycle.

- **Proposal Manager:** The core logic module responsible for the creation, editing, and submission of grant proposals. It handles input validation and ensures data integrity before persistence.
- **Grant Analytics Engine:** A read-only module that queries the database to visualize the status of submitted proposals and track milestones for the researcher.
- **Notification Service:** An internal utility that triggers status alerts (e.g., "Proposal Received") to the user upon successful submission.

B. Reviewer Subsystem

This package serves as the quality assurance layer.

- **Assignment Engine:** A logic module that retrieves unassigned proposals and links them to available reviewers based on workload or expertise.

- **Evaluation Manager:** Handles the input of quantitative scores and qualitative feedback. It implements the grading logic and calculates the final average score.
- **HOD Dispatcher:** A gateway component that packages completed evaluations and flags them as "Ready for Decision," effectively transferring control to the HOD subsystem.

C. HOD Subsystem

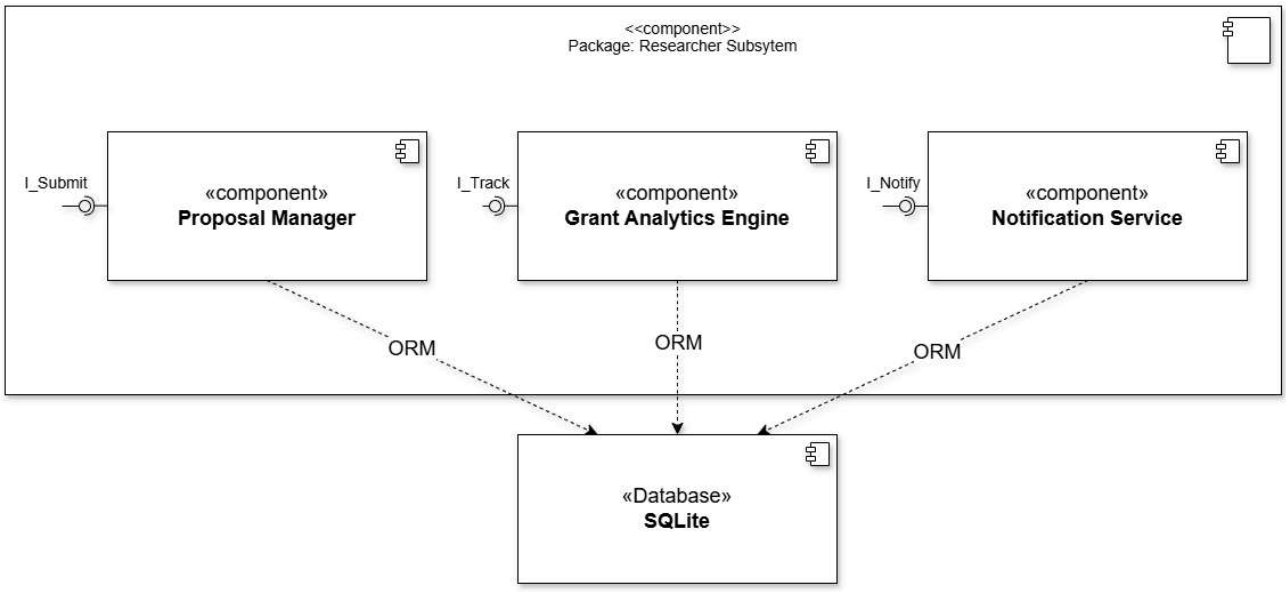
This package handles high-level decision-making and financial governance.

- **Approval Manager:** The decision engine that allows the HOD to accept or reject proposals based on the aggregated scores provided by the Reviewer subsystem.
- **Financial Tracker:** A risk-management module that monitors department funds. It enforces the "90% Budget Alert" rule and processes fund top-ups.
- **Department Analytics:** A reporting module that aggregates data to generate visual charts (Doughnut/Bar charts) regarding budget utilization and success rates.
- **Monitoring Engine:** Tracks the operational progress of active grants (e.g., Timeline adherence) after approval.

4.7 Component 1 (Subsystem 1: Researcher)

<TO DO: Describe the component and place the diagram here. There should be algorithm, pseudocode, flowchart, activity diagram to support the processing in the component.>

4.7.1 Component Diagram



4.7.2 Main Components

Component Name	***Type***	***Description***	***Use Cases Covered***
Proposal Manager	*Function Module*	*Manages the submission lifecycle. It handles form validation, file uploads (PDFs), and enforces **Version Control** (e.g., v1.0 -> v1.1) to prevent duplicate entries.*	*UC 1 (Upload Proposals)*
Grant Analytics Engine	*Function Module*	*Tracks the financial health of active grants. It calculates the **Utilization Percentage** (Spent / Allocated) and triggers visual alerts (Red/Green) based on the 90% threshold.*	*UC 2 (Track Grant Status)*
Notification Service	*Function Module*	*Acts as the communication bridge. It listens for system events (Approvals/Deadlines) to generate alerts and manages the **Read/Unread** state for the dashboard badge.*	*UC 3 (Receive Notifications)*

4.7.3 Component 1: Proposal Manager

Description: The Proposal Manager is responsible for handling the submission lifecycle. It validates user inputs, manages file uploads (PDFs), and critically, **enforces version control**. It detects if a title already exists and automatically increments the version number (e.g., v1.0 —> v1.1) instead of creating duplicates.

Pseudocode:

```
ALGORITHM submit_proposal(user, title, file, amount):
   1. VALIDATE form inputs (check if title is empty, file is PDF)
   2. IF form is invalid:
      RETURN error_message

   3. SEARCH Database for existing proposals by this 'user' and 'title'

   4. IF existing_proposal FOUND:
      a. FETCH the highest version number (max_version)
      b. SET new_version = max_version + 0.1
      c. SET status = "Pending"
   5. ELSE:
      a. SET new_version = 1.0
      b. SET status = "Draft"

   6. CREATE new Proposal record:
      - Version = new_version
      - File = uploaded_file
      - Status = status

   7. SAVE to Database
   8. TRIGGER Notification to Reviewer
   9. REDIRECT to Dashboard
END ALGORITHM
```

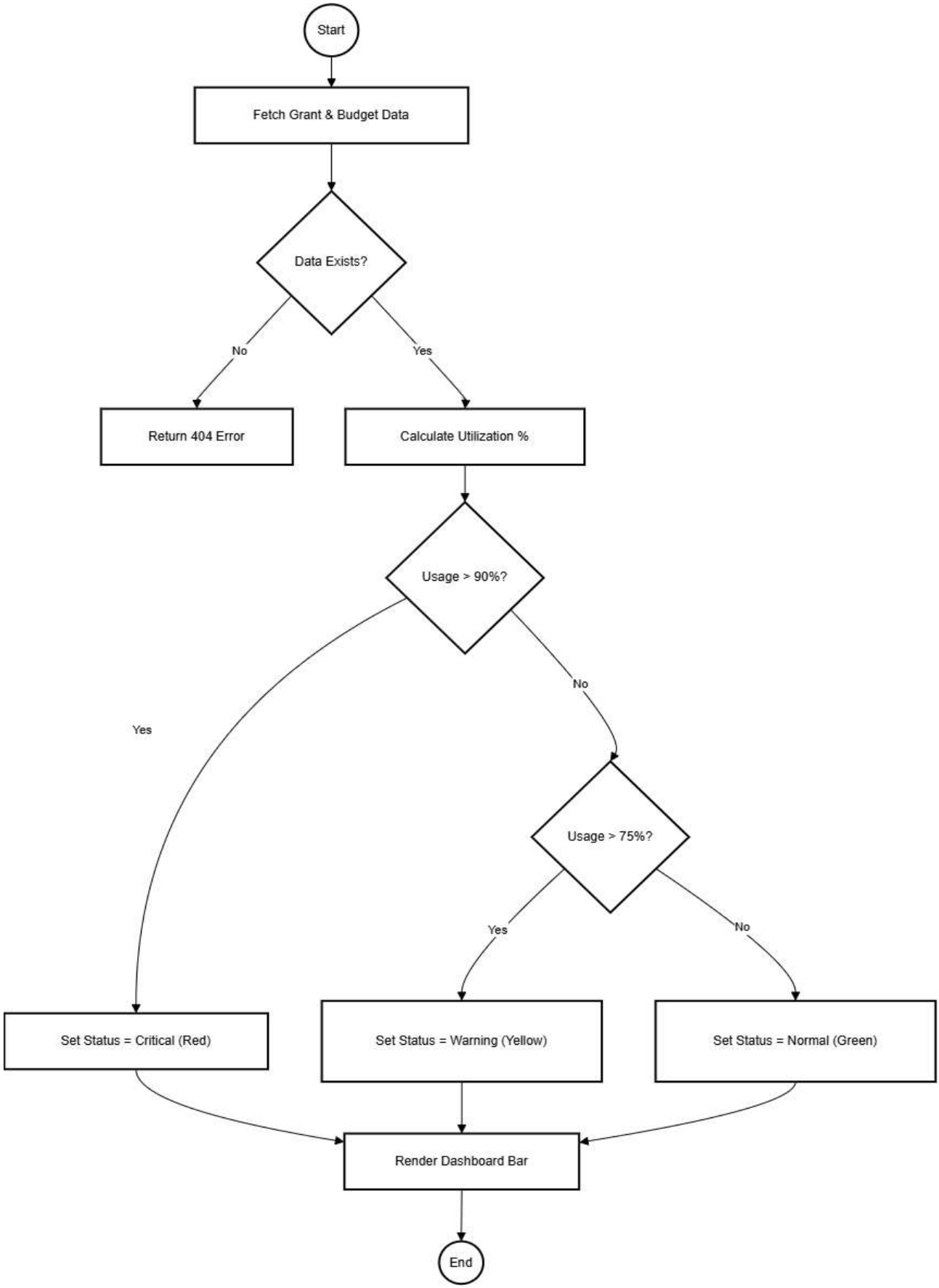

Start
User enters Title & Amount
User Uploads PDF
Is Form Valid?
No
Yes
Display Error Message
Title Exists?
Yes
No
Get Max Version
Set Version = Max + 0.1
Set Version = 1.0
Save to Database
Trigger Reviewer Alert
Redirect to Dashboard
End

4.7.4 Component 2: Grant Analytics Engine

Description: The Grant Analytics Engine is responsible for visualizing the financial health of a project. It fetches the raw financial data (Total Allocated vs. Total Spent), calculates the **Utilization Percentage**, and determines the **Health State** (Normal vs. Critical). It dynamically updates the UI elements (Green vs. Red progress bars) based on these calculations.

Pseudocode:

```
ALGORITHM track_budget(grant_id):
    1. FETCH Grant object WHERE id = grant_id
    2. FETCH Budget object associated with Grant

    3. IF Grant or Budget not found:
       RETURN 404_Error

    4. CALCULATE Utilization:
       percentage = (Budget.totalSpent / Grant.totalAllocated) * 100

    5. DETERMINE Health State:
       IF percentage >= 90:
          SET alert_level = "CRITICAL"
          SET color = "Red"
       ELSE IF percentage >= 75:
          SET alert_level = "WARNING"
          SET color = "Yellow"
       ELSE:
          SET alert_level = "NORMAL"
          SET color = "Green"

    6. RENDER "grant_detail.html" with:
       - usage_percent
       - alert_level
       - progress_bar_color
END ALGORITHM
```

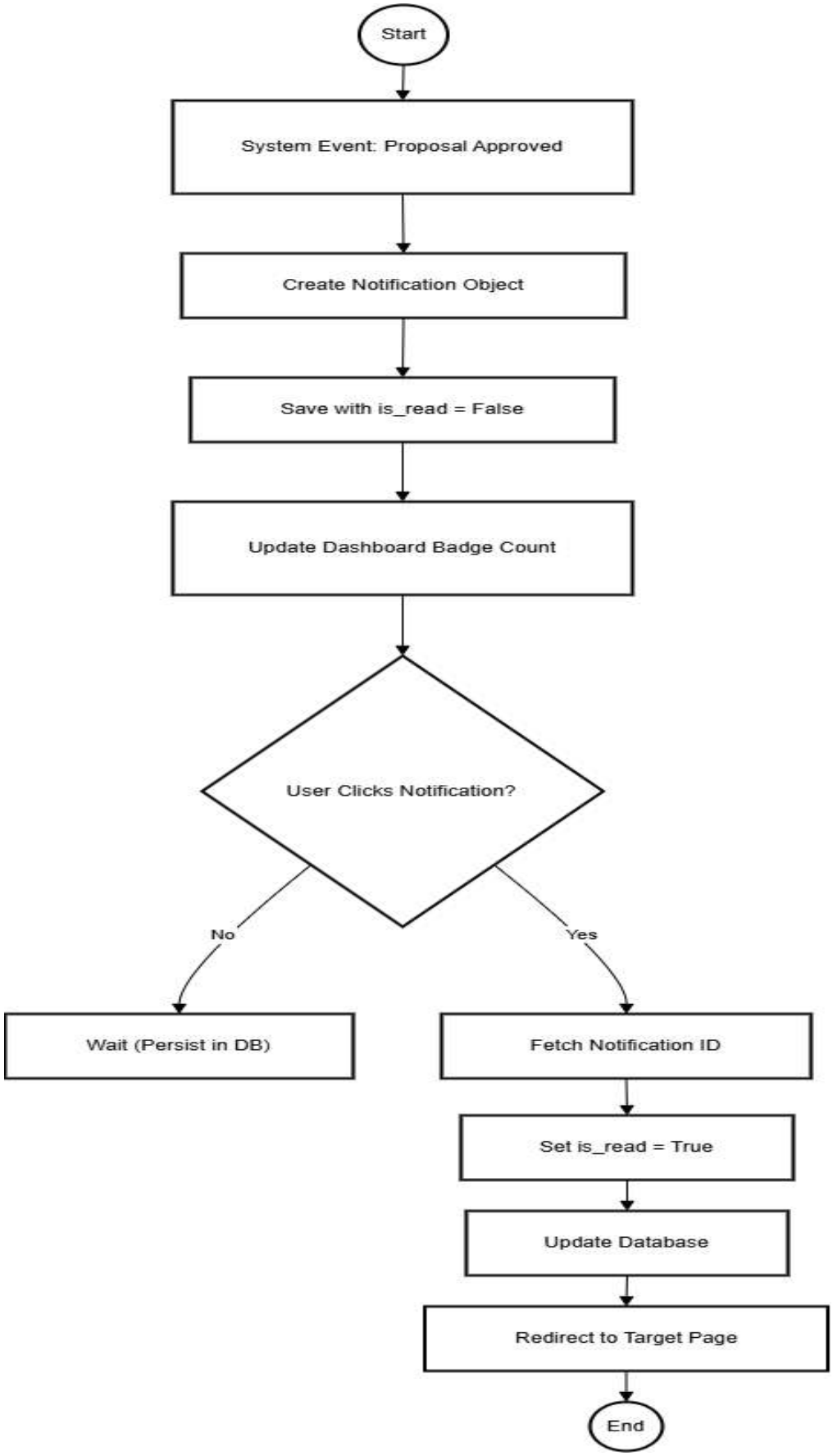

Start
Fetch Grant & Budget Data
Data Exists?
No
Yes
Return 404 Error
Calculate Utilization %
Usage > 90%?
Yes
No
Usage > 75%?
Yes
No
Set Status = Critical (Red)
Set Status = Warning (Yellow)
Set Status = Normal (Green)
Render Dashboard Bar
End

4.7.5 Component 3: Notification Service

Description: The Notification Service acts as the communication bridge. It has two main responsibilities: **Listening** for system events (Signals) to generate alerts, and **Managing** the read/unread state when a user interacts with them. It ensures the "Badge Count" on the dashboard is always accurate.

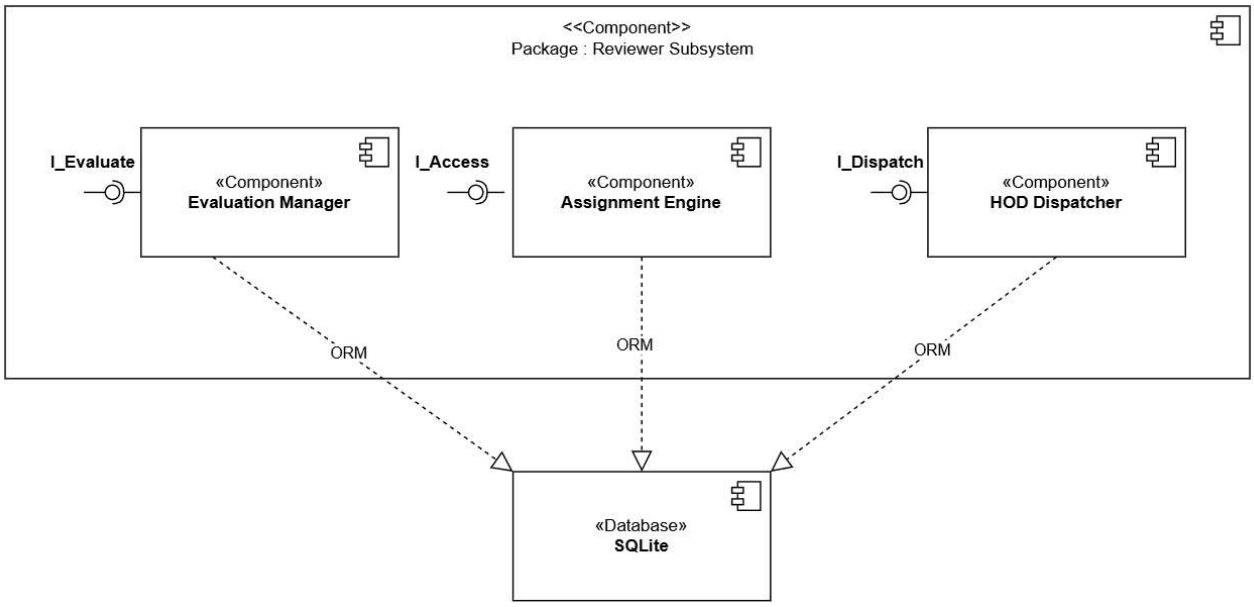
Pseudocode:

```
ALGORITHM mark_as_read(notification_id):
    1. RECEIVE request when user clicks "Bell Icon" or "Message"

    2. FETCH Notification object WHERE id = notification_id

    3. UPDATE object:
       is_read = True
       read_at = Current_Timestamp

    4. SAVE changes to Database

    5. DECREMENT User's Badge Count (Global Context)

    6. EXTRACT target_url from Notification (e.g., link to specific Grant)

    7. REDIRECT User to target_url
END ALGORITHM
```


Start
System Event: Proposal Approved
Create Notification Object
Save with is_read = False
Update Dashboard Badge Count
User Clicks Notification?
No
Yes
Wait (Persist in DB)
Fetch Notification ID
Set is_read = True
Update Database
Redirect to Target Page
End

4.7 Component 2 (Subsystem 2: Reviewer)

4.7.1 Component diagram



4.7.2 Main Components

The Reviewer subsystem is architected into three distinct functional modules located within the grants application. These components handle the assessment lifecycle of submitted proposals, from initial assignment to the systematic evaluation and final hand-off of recommendations to the Head of Department (HOD).

Component Name	Type	Description	Use Cases Covered
Evaluation Manager	Function Module	Processes qualitative and quantitative feedback. It ensures evaluations are saved securely and "locked" once submitted to maintain integrity.	UC 4 (Evaluation/Scoring Proposals)
Assignment Engine	Function Module	Get the global database to present reviewers with their specific assigned proposals. It verifies permissions via the Auth Module before granting access to documents	UC 4 (View researcher documents)
HOD Dispatcher	Function Module	Handles the "Submit	UC 4 (Give Feedback)

		Evaluation" workflow by updating proposal statuses and triggering automated notifications to the HOD.	

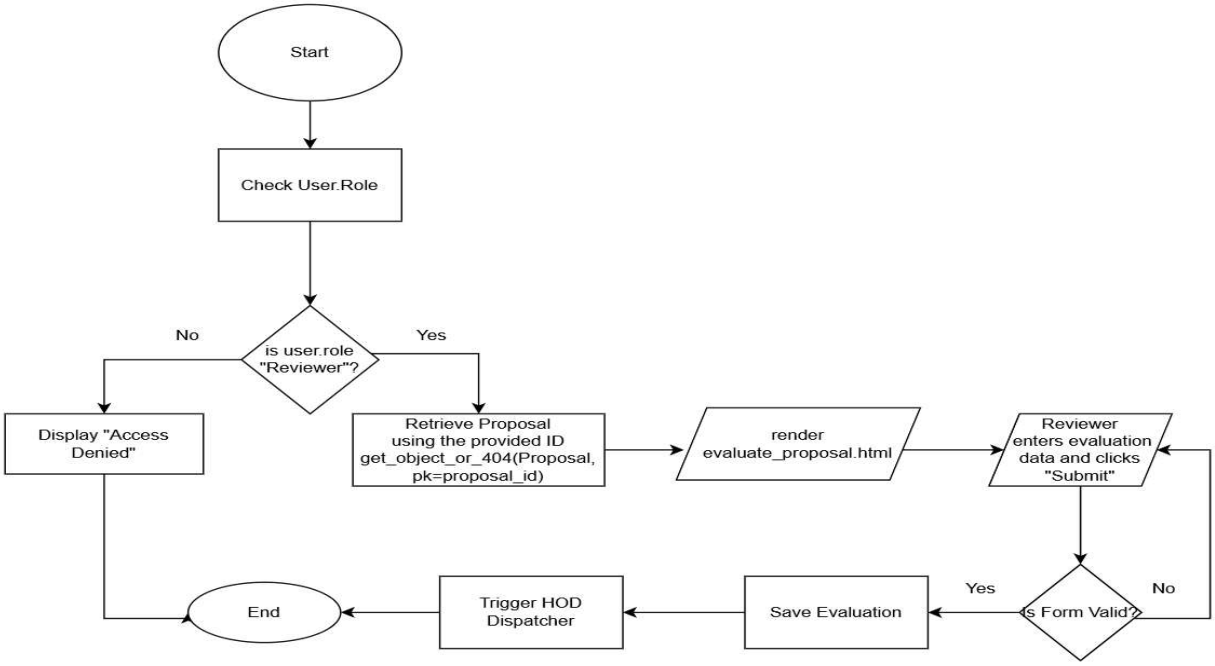
4.7.3 Component 1: Evaluation Manager

Description: This manager is responsible for capturing the reviewer's assessment. It validates that the score and comments meet system constraints before committing the data to the database.

Pseudocode:

ALGORITHM evaluate_proposal(user, proposal_id, score, comments):
1. VERIFY user.role equals "Reviewer"
   ELSE RETURN "Access Denied"
2. RETRIEVE Proposal record from Database using proposal_id
3. VALIDATE form inputs (e.g., score is integer, comments are present)
4. IF form is valid:
   a. CREATE new Evaluation record linked to Proposal and Reviewer
   b. SAVE Evaluation to Database
   c. CALL HOD Dispatcher to update status
5. ELSE:
   a. RENDER evaluation form with validation errors

END ALGORITHM



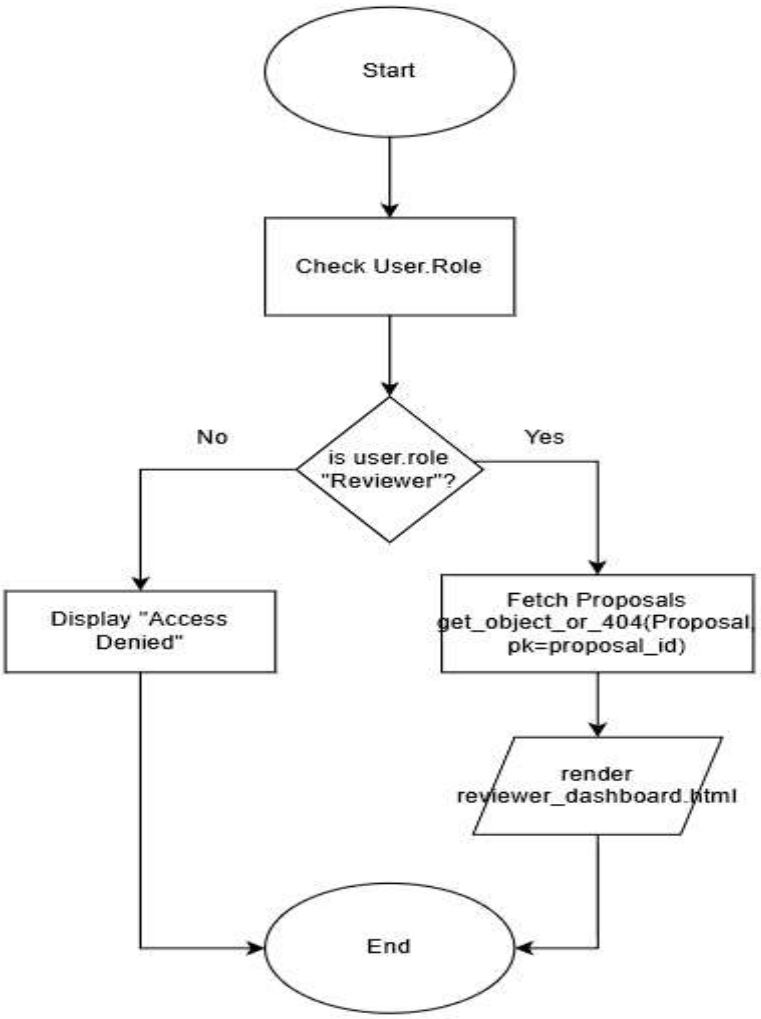
4.7.4 Component 2: Assignment Engine

Description: This module serves as the primary engine for the Reviewer Dashboard. It aggregates and displays all proposals that are ready for review, filtering versions to ensure only valid submissions are assessed.

Pseudocode:

ALGORITHM load_reviewer_dashboard(user):
1. VERIFY user.role equals "Reviewer"
2. RETRIEVE list of all Proposals from Database
3. RENDER "reviewer_dashboard.html" with the filtered proposal list

END ALGORITHM



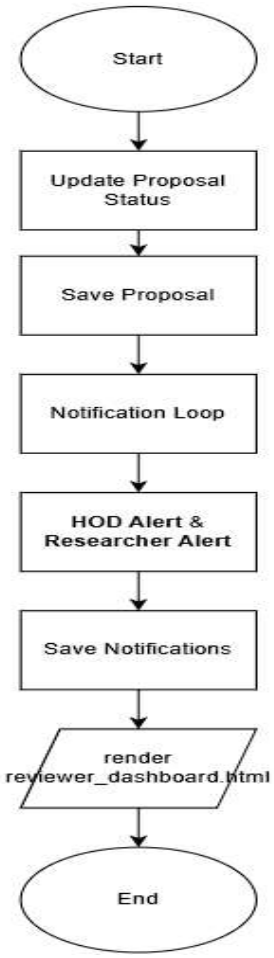
4.7.5 Component 3: HOD Dispatcher

Description: The HOD Dispatcher manages the workflow transition once an evaluation is submitted. It bridges the gap between the Reviewer and HOD roles by updating the proposal state and notifying stakeholders.

Pseudocode:

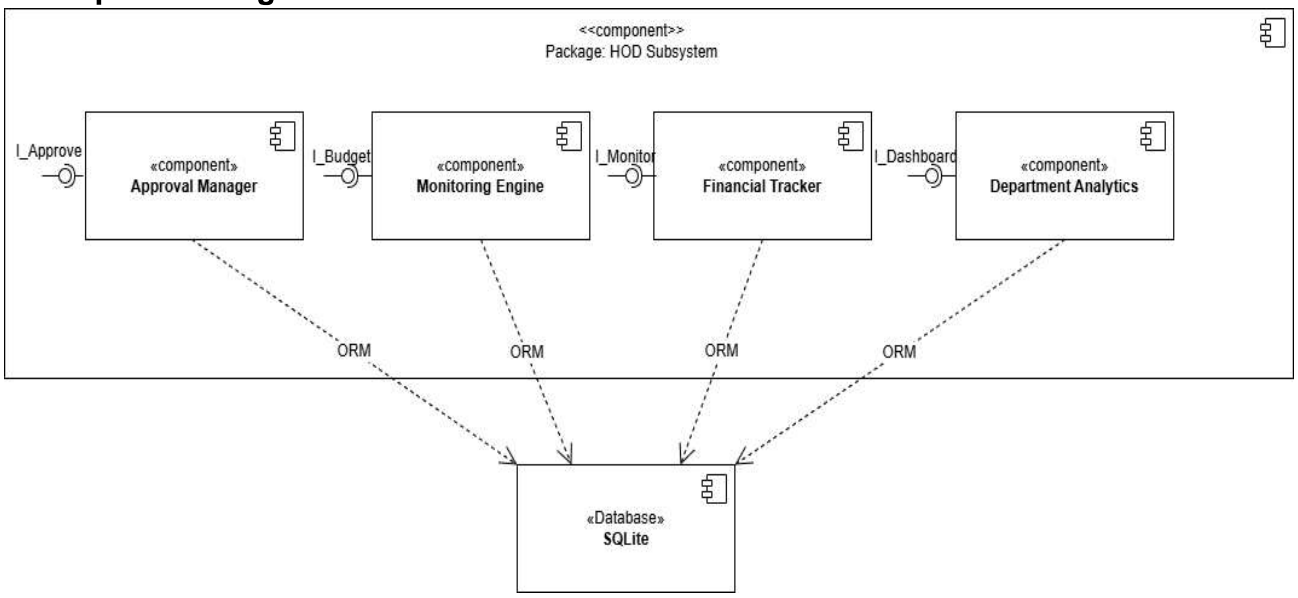
ALGORITHM finalize_review_transition(proposal):
1. UPDATE proposal.status to "Review Complete"
2. SAVE updated proposal record to Database
3. TRIGGER Notifications:
   a. CREATE Notification for HOD ("Ready for Decision")
   b. CREATE Notification for Researcher ("Review Done")
4. RETURN Success and Redirect to Reviewer Dashboard

END ALGORITHM



4.7 Component 3 (Subsystem 3: HOD)

4.7.1 Component diagram



4.7.2 Main Components

The Head of Department (HOD) subsystem is architected into three distinct functional modules located within the grants application. These components handle the lifecycle of a grant from approval to monitoring and financial oversight.

Component Name	Type	Description	Use Cases Covered
Approval Manager	Function Module	Manages the decision process, validates department funds, and initializes the grant entity.	UC 5 (Approve/Reject)
Financial Tracker	Function Module	Tracks real-time expenditures, calculates remaining balances, and processes budget top-ups.	UC 6 (Track Budget)
Monitoring Engine	Function Module	Aggregates project	UC 7 (Monitor

		health metrics (KPIs) and retrieves detailed progress reports.	Research Progress)
Dashboard Manager	Function Module	Serves as the central analytical hub for the HOD; aggregates departmental data from proposals and grants to calculate real-time performance metrics (KPIs) and provides administrative tools for exporting visual reports.	UC 8 (Monitor Department Analytics)

4.7.3 Component 1: Approval Manager

Description: The Approval Manager is the core decision-making module of the HOD subsystem. It handles the transition of a researcher's proposal into an active, funded grant. Its primary responsibility is to act as a financial gatekeeper by validating that the department has sufficient funds before any grant is authorized. It ensures data integrity by linking the grant to the original proposal and initializing the financial tracking records (Budget) simultaneously.

Pseudocode:

ALGORITHM ApproveProposal(user, proposal_id, input_amount, dates)

1. VERIFY user.role equals "HOD"

   ELSE RETURN "Access Denied"

2. RETRIEVE Proposal(proposal_id) AND HOD_User(current_user)

3. IF HTTP Method is POST (Submission):

   a. PARSE allocated_amount from form input

   b. VALIDATE Funds:

IF allocated_amount > hod_user.department_budget:

RETURN Error("Insufficient Department Funds")

c. CREATE or UPDATE Grant:

SET grant.proposal = proposal

SET grant.totalAllocatedAmount = allocated_amount

SET grant.dates = input_dates

SAVE Grant

d. IF Grant was newly created:

DEDUCT allocated_amount FROM hod_user.department_budget

CREATE Budget Record (linked to Grant, total_spent = 0.00)

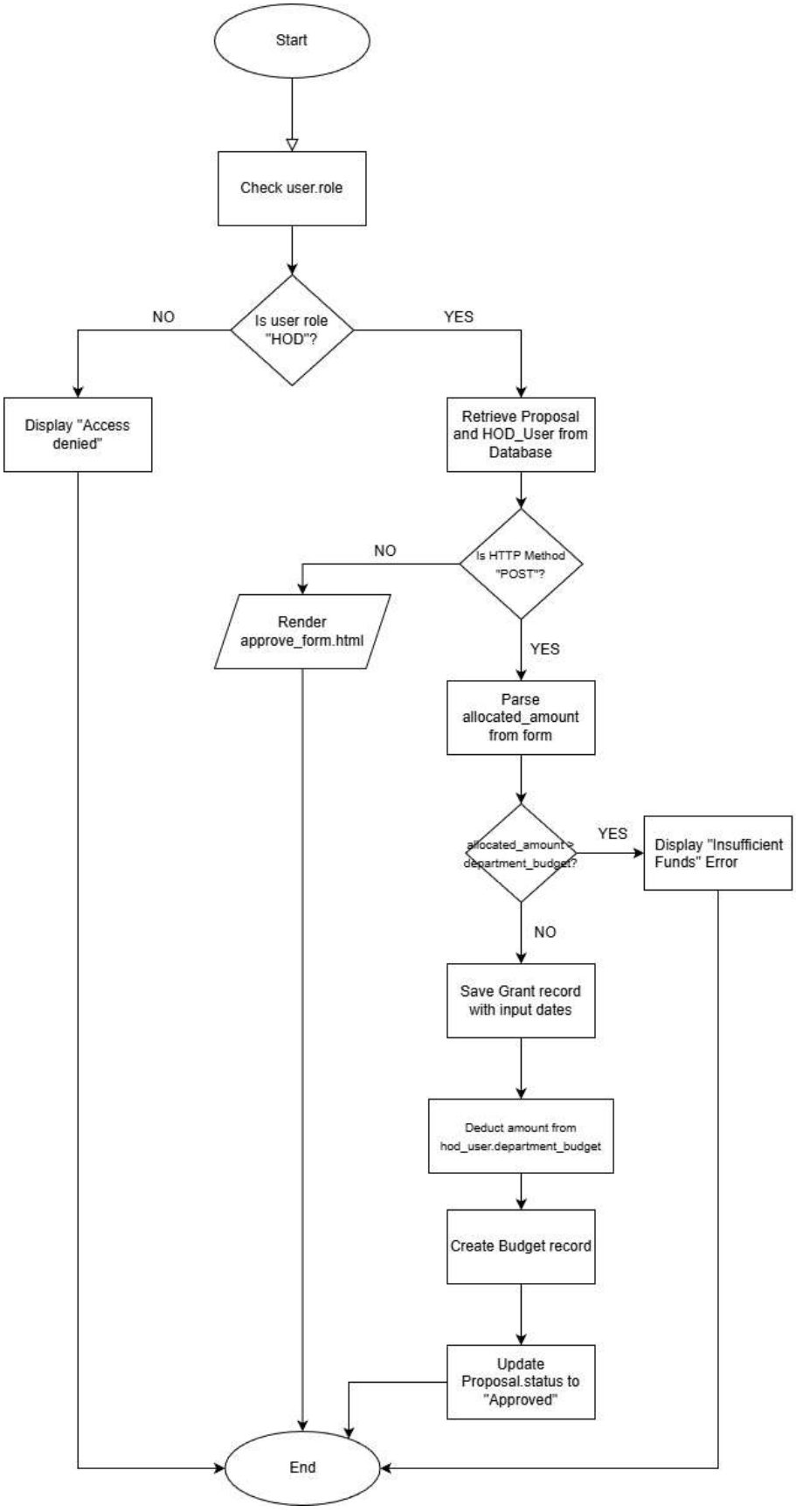
UPDATE Proposal.status = "Approved"

e. RETURN Success Message AND Redirect to Dashboard

4. ELSE (GET Request):

RENDER "approve_form.html" with Proposal details and Current Department Budget

END ALGORITHM


Start
Check user.role
Is user role "HOD"?
NO
YES
Display "Access denied"
Retrieve Proposal and HOD_User from Database
Is HTTP Method "POST"?
NO
YES
Render approve_form.html
Parse allocated_amount from form
allocated_amount > department_budget?
YES
Display "Insufficient Funds" Error
NO
Save Grant record with input dates
Deduct amount from hod_user.department_budget
Create Budget record
Update Proposal.status to "Approved"
End

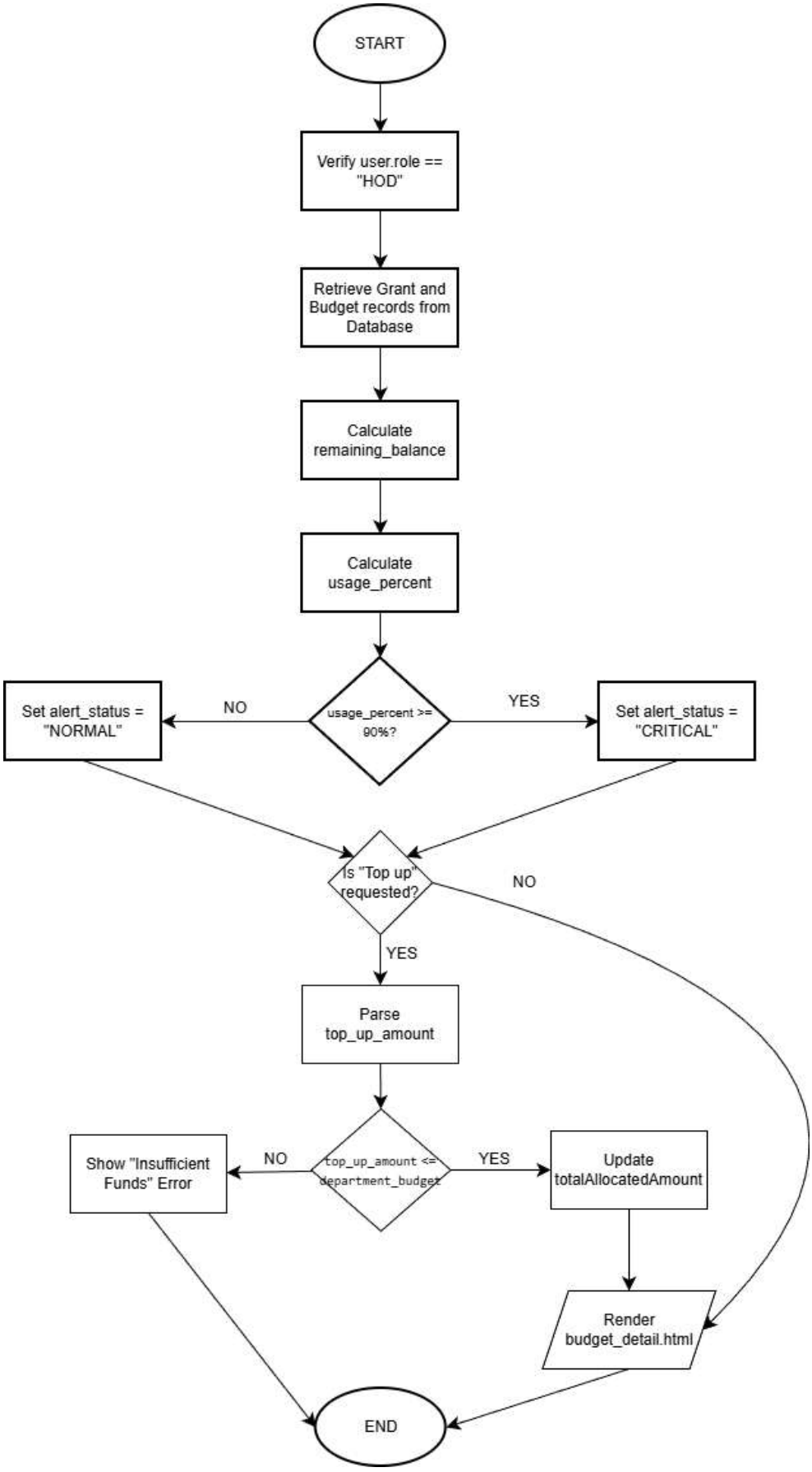
4.7.4 Component 2: Financial Tracker

Description: The Financial Tracker provides continuous fiscal oversight for active research grants. It calculates real-time metrics, such as the remaining balance and the percentage of funds consumed. A core business rule of this component is the "Critical Threshold" monitor, which triggers a visual alert when project spending exceeds 90% of the total allocation. Additionally, it handles "Top-Up" requests, allowing the HOD to inject further department funds into a project if required.

Pseudocode:

```
ALGORITHM TrackBudget(user, grant_id)
  1. VERIFY user.role equals "HOD"

  2. RETRIEVE Grant(grant_id) AND Budget(grant_id)

  3. CALCULATE Financial Metrics:
     SET total_spent = Budget.total_spent
     SET remaining = Grant.totalAllocatedAmount - total_spent
     SET usage_percent = (total_spent / Grant.totalAllocatedAmount) * 100

  4. MONITOR Threshold:
     IF usage_percent >= 90:
        SET alert_status = "CRITICAL"
     ELSE:
        SET alert_status = "NORMAL"

  5. IF Top-Up Requested (POST):
     a. PARSE top_up_amount
     b. IF top_up_amount <= hod_user.total_department_budget:
        UPDATE Grant.totalAllocatedAmount += top_up_amount
        UPDATE hod_user.total_department_budget -= top_up_amount
        SAVE records
```

```
        c. ELSE:
            RETURN Error("Insufficient Department Funds")

    6. RENDER "budget_detail.html" WITH metrics and alert_status
END ALGORITHM
```

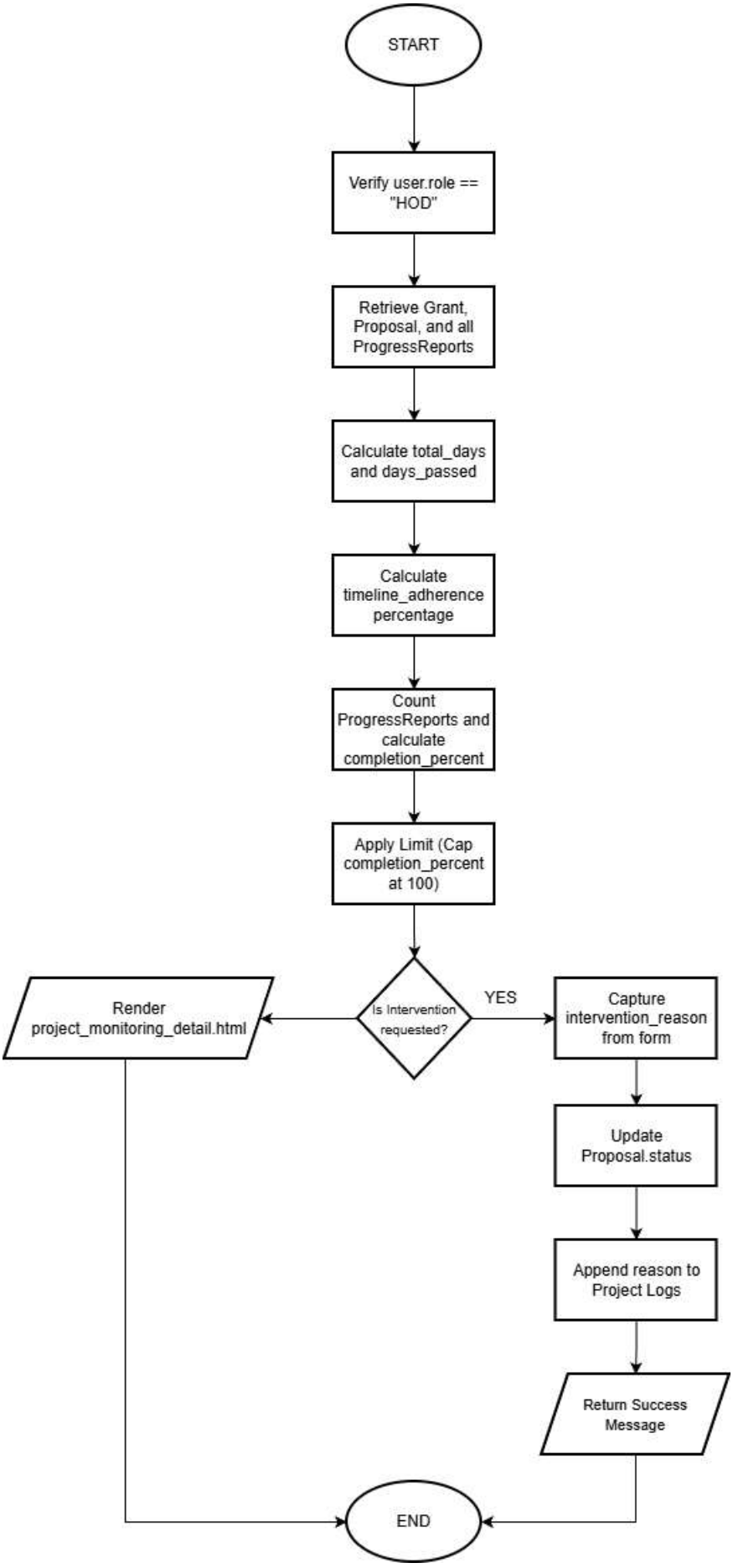

START
Verify user.role == "HOD"
Retrieve Grant and Budget records from Database
Calculate remaining_balance
Calculate usage_percent
usage_percent >= 90%?
NO
YES
Set alert_status = "NORMAL"
Set alert_status = "CRITICAL"
Is "Top up" requested?
NO
YES
Parse top_up_amount
top_up_amount <= department_budget
NO
YES
Show "Insufficient Funds" Error
Update totalAllocatedAmount
Render budget_detail.html
END

4.7.5 Component 3: Monitoring Engine

Description: The Monitoring Engine is responsible for the qualitative and quantitative oversight of research projects. It aggregates data from progress reports to evaluate project health through two primary Key Performance Indicators (KPIs): Timeline Adherence and Completion Percentage. This component allows the HOD to perform "Deep Dive" monitoring by reviewing individual report contents and provides a mechanism for "HOD Intervention" if a project is falling behind schedule or failing to meet milestones.

Pseudocode:

```
ALGORITHM MonitorProject(user, grant_id)
  1. VERIFY user.role equals "HOD"

  2. RETRIEVE Grant, associated Proposal, and List<ProgressReport>

  3. CALCULATE Timeline KPI:
     total_days = Grant.endDate - Grant.startDate
     days_passed = CurrentDate - Grant.startDate
     SET timeline_adherence = (days_passed / total_days) * 100

  4. CALCULATE Completion KPI:
     SET report_count = ProgressReport.count()
     SET completion_percent = report_count * 25  // Assuming 4 milestones
     LIMIT completion_percent TO 100

  5. IF Intervention Requested (POST):
     a. CAPTURE intervention_reason from form
     b. UPDATE Proposal.status = "Under Intervention"
     c. APPEND reason to Project Logs
     d. RETURN Success

  6. RENDER "project_monitoring_detail.html" WITH kpi_metrics and report_list

END ALGORITHM
```

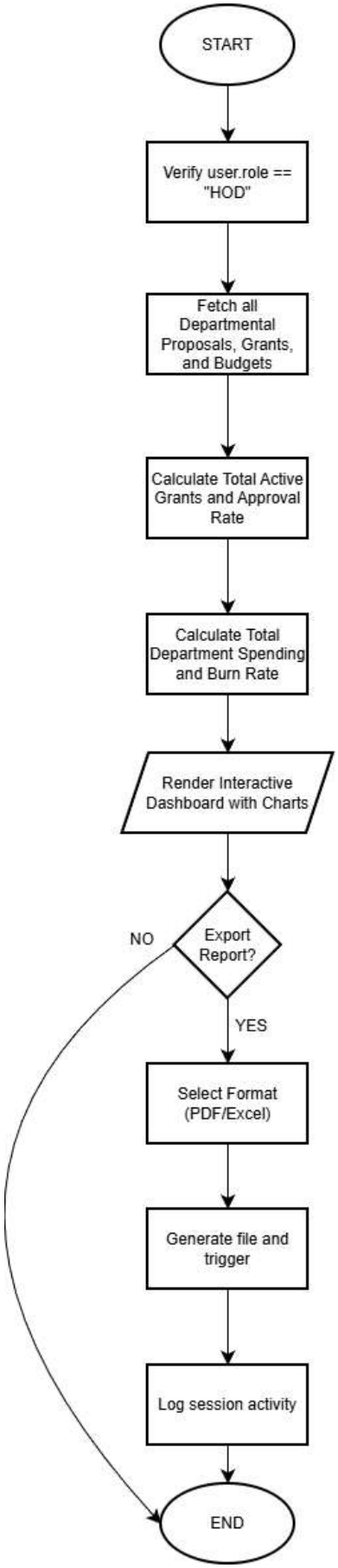

START
Verify user.role == "HOD"
Retrieve Grant, Proposal, and all ProgressReports
Calculate total_days and days_passed
Calculate timeline_adherence percentage
Count ProgressReports and calculate completion_percent
Apply Limit (Cap completion_percent at 100)
Is Intervention requested?
YES
Render project_monitoring_detail.html
Capture intervention_reason from form
Update Proposal.status
Append reason to Project Logs
Return Success Message
END

4.7.6 Component 4: Dashboard Manager (Department Analytics)

Description: The Dashboard Manager is the analytical hub of the HOD subsystem. It aggregates data from all proposals and active grants to provide a "helicopter view" of departmental performance. Its primary role is to calculate global metrics—such as the total budget burn rate and proposal success rates—and provide administrative tools for exporting these insights into formal reports (PDF/Excel).

Pseudocode:

```
ALGORITHM DashboardManager(user)
  1. VERIFY user.role equals "HOD"
     ELSE RETURN "Access Denied"

  2. RETRIEVE All Proposals AND All Grants linked to Department

  3. CALCULATE Department Metrics:
     SET total_active_grants = Count(Grants)
     SET approval_rate = (Count(Approved_Proposals) / Count(Total_Proposals)) * 100
     SET total_dept_spent = Sum(All_Budget.total_spent)
     SET total_dept_allocated = Sum(All_Grant.totalAllocatedAmount)
     SET overall_burn_rate = (total_dept_spent / total_dept_allocated) * 100

  4. IF Export Requested (POST):
     a. PARSE format (PDF or Excel)
     b. COMPILE metrics into report template
     c. TRIGGER file download
     d. LOG export activity in session logs

  5. RENDER "hod_dashboard.html" WITH aggregated_metrics AND charts
END ALGORITHM
```


START
Verify user.role == "HOD"
Fetch all Departmental Proposals, Grants, and Budgets
Calculate Total Active Grants and Approval Rate
Calculate Total Department Spending and Burn Rate
Render Interactive Dashboard with Charts
Export Report?
NO
YES
Select Format (PDF/Excel)
Generate file and trigger
Log session activity
END

4.8 Deployment Diagram

<TO DO: Describe the deployment diagram and place the diagram here.>

The deployment diagram for your **Research Grant Management System (RGMS)** provides a comprehensive view of how your Django application is physically and logically distributed. It strictly follows the **three-tier architecture** described in your design specification, separating the user interface, business logic, and data storage.

1. Client Node: User Workstation (<<device>>)

This node represents the hardware (PC, Laptop, or Mobile) used by the system's actors: **Researchers**, **Reviewers**, and **HODs.**

- **Web Browser Environment:** The system requires a modern browser (Chrome, Safari, Firefox) that supports HTML5 and ES6 standards.
- **RGMS UI Artifact:** Inside the browser, the **Presentation Layer** is active. It consists of dynamic templates rendered by the **Django Template Engine.**
- **Communication:** All requests are sent via a **URL Dispatcher** over the **HTTP/HTTPS protocol** to the central server.

---

2. Application Tier: Django Cloud Server (<<device>>)

This is the centralized hosting environment that acts as the "brain" of the RGMS.

- **Django Runtime:** The server operates within a **Python 3.x** and **Django 4.x** execution environment.
- **Modular Subsystems (Inner Artifacts):** This layer contains the specific logic modules that handle the system's workflows:
    - **Auth Module:** Manages role-based access control (RBAC), ensuring that an HOD sees the department dashboard while a Researcher sees their own proposals.
    - **Submission & Review Subsystems:** These artifacts handle the **Proposal Manager** (for version control) and the **Evaluation Manager** (for reviewer feedback).
    - **Admin & Monitoring Subsystems:** These artifacts power the **Financial Tracker** and **KPI Engine**, which handle budget updates and performance metrics.

---

3. Data Tier: Persistence & Storage (<<device>>)

The data layer is hosted on a SQL-compliant engine. As indicated in your architecture, this tier is split into two specialized storage environments to handle different types of data.

A. Relational Database (Inner Box 1)

- **Execution Environment:** The system is designed for **SQLite** during development but is prepared for **PostgreSQL** in production.

- **Artifacts (Grant_Records.db):** This stores all structured entities defined in your Data Dictionary, such as User, Proposal, Grant, Budget, and Evaluation records.
- **Relationship:** It communicates with the Application Tier using the **Django ORM**.

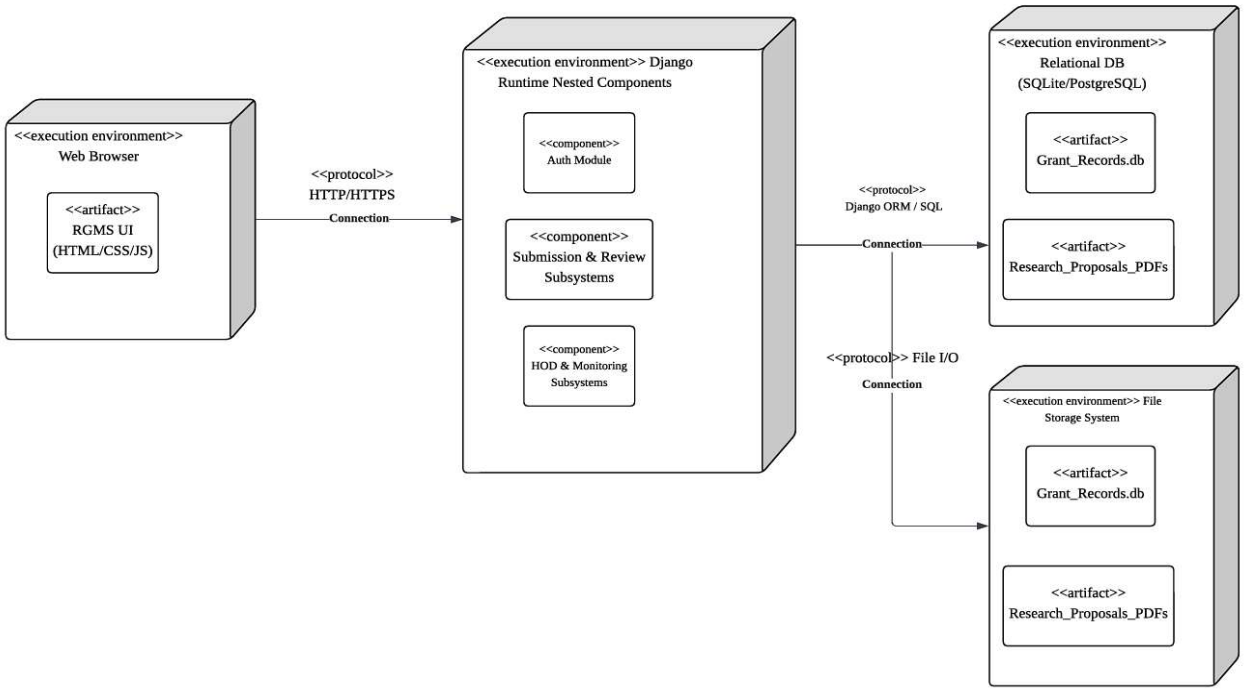
B. File Storage System (Inner Box 2)

- **Execution Environment:** A dedicated server-side file system.
- **Artifacts (Research_Proposals_PDFs):** This specifically manages unstructured data—the actual **PDF proposal documents** uploaded by researchers.
- **Relationship:** It is accessed via **File I/O** commands triggered by the Proposal Manager whenever a document is uploaded or viewed.

Summary of System Readiness

This deployment design confirms that the RGMS is ready for implementation because:

1. **Scalability:** The separation of the Database and File System allows for high-resolution chart rendering (via Matplotlib/Seaborn) without slowing down record lookups.
2. **Integrity:** The automated **Document Version Control** logic is integrated into the deployment flow, preventing duplicate submissions.
3. **Governance:** The **Financial Tracker** logic ensures that the HOD can monitor budget health in real-time, with critical alerts triggered at 90% usage

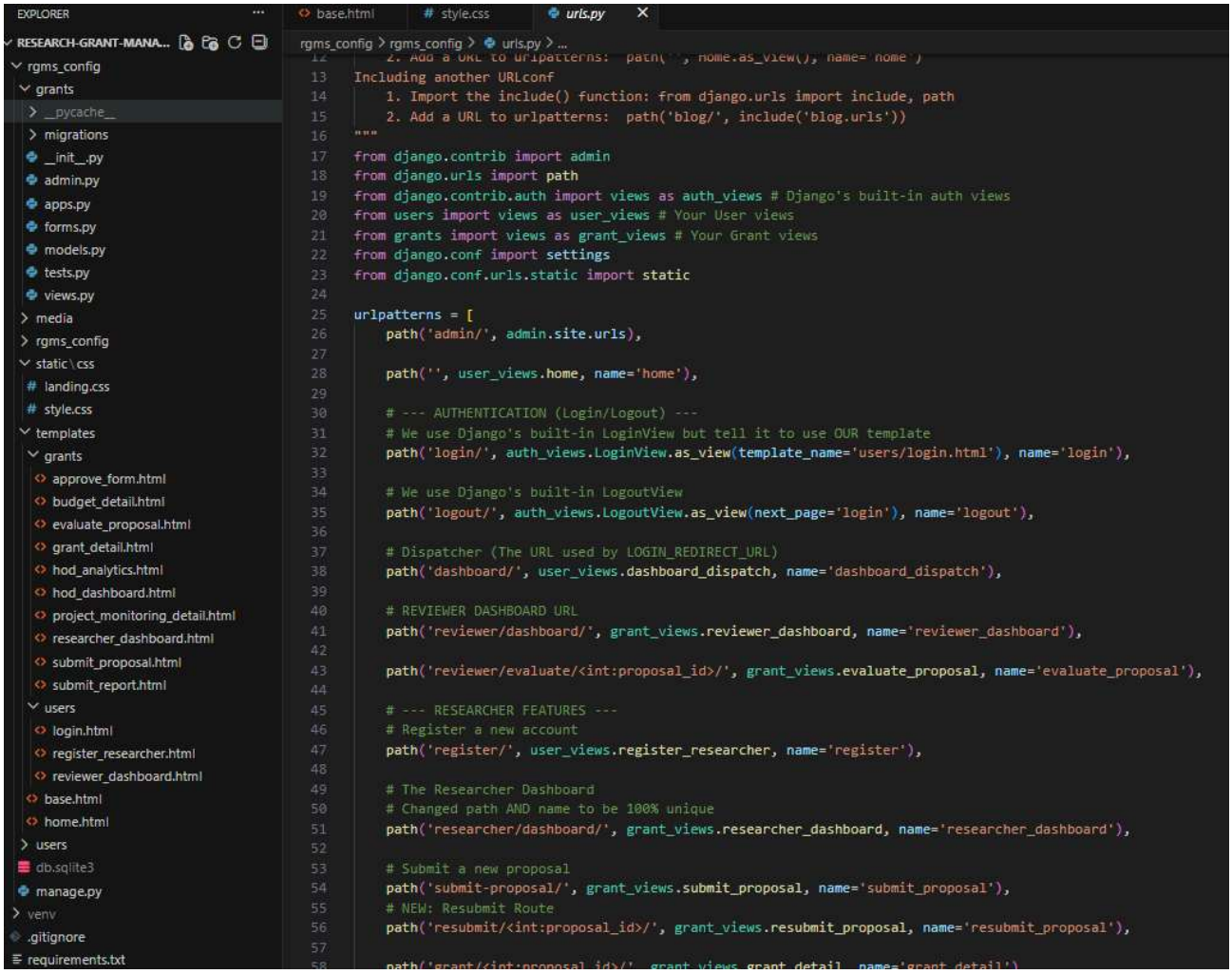


5 Implementation Details

5.1 Development Environment

The Research Grant Management System (RGMS) is developed using the **Django 6.0** framework on a **Windows 11** operating system using **Visual Studio Code (VS Code)** as the primary Integrated Development Environment (IDE). The system utilizes a robust stack to organize backend logic, database management, and frontend presentation.

- **Backend (Python/Django)**: The backend manages the core functionality, including user authentication, proposal versioning, and grant approval workflows. The models.py files define the database schema for users, proposals, and budgets, while views.py contains the business logic for handling user requests.
- **Frontend (HTML, CSS)**: The frontend is built using Django's template engine, with specialized CSS files (e.g., landing.css, style.css) providing a professional, responsive interface for different user roles.
- **Database (SQLite)**: All system data, including user profiles, PDF proposals, and financial logs, are securely stored in the db.sqlite3 relational database.
- **External Libraries**:
    - **Pandas, Matplotlib, and Seaborn**: Integrated for generating departmental analytics and budget utilization reports for HODs.
    - **xhtml2pdf**: Used to facilitate the generation of professional research proposal documents.


EXPLORER
RESEARCH-GRANT-MANA...
rgms_config
grants
__pycache__
migrations
__init__.py
admin.py
apps.py
forms.py
models.py
tests.py
views.py
media
rgms_config
static\css
landing.css
style.css
templates
grants
approve_form.html
budget_detail.html
evaluate_proposal.html
grant_detail.html
hod_analytics.html
hod_dashboard.html
project_monitoring_detail.html
researcher_dashboard.html
submit_proposal.html
submit_report.html
users
login.html
register_researcher.html
reviewer_dashboard.html
base.html
home.html
users
db.sqlite3
manage.py
venv
.gitignore
requirements.txt
base.html
style.css
urls.py
rgms_config > rgms_config > urls.py > ...
Including another URLconf
1. Import the include() function: from django.urls import include, path
2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from django.contrib.auth import views as auth_views # Django's built-in auth views
from users import views as user_views # Your User views
from grants import views as grant_views # Your Grant views
from django.conf import settings
from django.conf.urls.static import static
urlpatterns = [
path('admin/', admin.site.urls),
path('', user_views.home, name='home'),
--- AUTHENTICATION (Login/Logout) ---
We use Django's built-in LoginView but tell it to use OUR template
path('login/', auth_views.LoginView.as_view(template_name='users/login.html'), name='login'),
We use Django's built-in LogoutView
path('logout/', auth_views.LogoutView.as_view(next_page='login'), name='logout'),
Dispatcher (The URL used by LOGIN_REDIRECT_URL)
path('dashboard/', user_views.dashboard_dispatch, name='dashboard_dispatch'),
REVIEWER DASHBOARD URL
path('reviewer/dashboard/', grant_views.reviewer_dashboard, name='reviewer_dashboard'),
path('reviewer/evaluate/<int:proposal_id>/', grant_views.evaluate_proposal, name='evaluate_proposal'),
--- RESEARCHER FEATURES ---
Register a new account
path('register/', user_views.register_researcher, name='register'),
The Researcher Dashboard
Changed path AND name to be 100% unique
path('researcher/dashboard/', grant_views.researcher_dashboard, name='researcher_dashboard'),
Submit a new proposal
path('submit-proposal/', grant_views.submit_proposal, name='submit_proposal'),
NEW: Resubmit Route
path('resubmit/<int:proposal_id>/', grant_views.resubmit_proposal, name='resubmit_proposal'),


(venv) PS C:\Users\mario\Documents\CSE6214 TC2L TT7L\Project-20251113\Research-Grant-Management-System> cd .\rgms_config\
(venv) PS C:\Users\mario\Documents\CSE6214 TC2L TT7L\Project-20251113\Research-Grant-Management-System\rgms_config> python.exe .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
February 04, 2026 - 16:02:05
Django version 6.0, using settings 'rgms_config.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/6.0/howto/deployment/

5.2 Software Integration

The Research Grant Management System (RGMS) is composed of multiple subsystems, each responsible for specific functionalities such as user management, research proposal tracking, reviewer evaluations, and administrative oversight by the Head of Department (HOD). This section outlines the integration strategy that ensures seamless communication, data flow, and interoperability between these components. The system follows a modular architecture, where each Django app operates independently but interacts with others through database relationships and centralized routing mechanisms. This approach ensures scalability and efficient data sharing, enabling accurate tracking of research progress while facilitating collaboration between researchers, reviewers, and department heads.

5.2.1 Integration Strategy

The integration strategy is built on Django's MTV (Model-Template-View) architecture, which organizes the system into three core components: Models for database structure, Views for request processing, and Templates for rendering the user interface. Each subsystem interacts with others through:

- **URL Routing**: Directs user requests to the appropriate app-specific functional modules.
- **Database Relationships**: Ensures data consistency across apps, such as linking reviewer evaluations to specific researcher proposals.
- **Authentication Middleware**: Manages secure session handling and role-based access across all modules.

File	Description
`settings.py`	Configures the installed apps (`users`, `grants`), middleware, database settings, and template directories for the project.
`urls.py`	Defines the central URL routing system, mapping user requests to specific views in each app to ensure all functionalities are accessible.

models.py (per app)	Defines the database schema and relationships, including models for researchers, proposals, grants, and budgets.
views.py (per app)	Handles request processing and business logic, such as version control for resubmissions and financial validation for grant approvals.
forms.py (per app)	Manages user data entry and validation for proposal submissions and progress reports, bridging the frontend and backend.

5.2.2 System Configuration and Middleware Integration

The settings.py file plays a crucial role in configuring the RGMS, ensuring proper integration of all applications and security settings.

The INSTALLED_APPS section includes core Django applications and the custom users and grants apps, which handle key functionalities such as proposal submissions and budget tracking. Built-in middleware components ensure security, CSRF protection, and session handling for smooth user interactions.

Static files and uploaded media, such as research proposal PDF documents, are served correctly using STATIC_URL and MEDIA_URL.

```
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/6.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-8jrbsfd_!qdal+-ob+f3_p^!mnx9+8^#izc$ysao%r^uga&e+c'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

AUTH_USER_MODEL = 'users.User'

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # My apps
    'users',
    'grants',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'rgms_config.urls'
```

Django's default middleware stack is used for essential operations such as session management, security enforcement, and authentication handling. These middleware components facilitate request and response processing, ensuring efficient communication between the frontend and backend.

```
WSGI_APPLICATION = 'rgms_config.wsgi.application'


# Database
# https://docs.djangoproject.com/en/6.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/6.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/6.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/6.0/howto/static-files/

STATIC_URL = 'static/'

STATICFILES_DIRS = [
    BASE_DIR / 'static',
]

LOGIN_REDIRECT_URL = 'dashboard_dispatch'
```

5.2.3 URL Routing Integration

The urls.py files manage request routing and inter-app communication. The main urls.py integrates all app routes using path mapping, ensuring a centralized and well-structured request flow.

Feature-specific views, such as submit-proposal/ and hod/dashboard/, are mapped to corresponding URLs to allow for easy navigation within the system. This modular approach ensures that the routing remains organized as the system scales.

```
urlpatterns = [
    path('admin/', admin.site.urls),

    path('', user_views.home, name='home'),

    # --- AUTHENTICATION (Login/Logout) ---
    # We use Django's built-in LoginView but tell it to use OUR template
    path('login/', auth_views.LoginView.as_view(template_name='users/login.html'), name='login'),

    # We use Django's built-in LogoutView
    path('logout/', auth_views.LogoutView.as_view(next_page='login'), name='logout'),

    # Dispatcher (The URL used by LOGIN_REDIRECT_URL)
    path('dashboard/', user_views.dashboard_dispatch, name='dashboard_dispatch'),

    # REVIEWER DASHBOARD URL
    path('reviewer/dashboard/', grant_views.reviewer_dashboard, name='reviewer_dashboard'),

    path('reviewer/evaluate/<int:proposal_id>/', grant_views.evaluate_proposal, name='evaluate_proposal'),

    # --- RESEARCHER FEATURES ---
    # Register a new account
    path('register/', user_views.register_researcher, name='register'),

    # The Researcher Dashboard
    # Changed path AND name to be 100% unique
    path('researcher/dashboard/', grant_views.researcher_dashboard, name='researcher_dashboard'),

    # Submit a new proposal
    path('submit-proposal/', grant_views.submit_proposal, name='submit_proposal'),
    # NEW: Resubmit Route
    path('resubmit/<int:proposal_id>/', grant_views.resubmit_proposal, name='resubmit_proposal'),

    path('grant/<int:proposal_id>/', grant_views.grant_detail, name='grant_detail'),
    path('grant/report/<int:proposal_id>/', grant_views.submit_report, name='submit_report'),

    # --- HOD FEATURES ---
    path('hod/dashboard/', grant_views.hod_dashboard, name='hod_dashboard'),
    path('hod/approve/<int:proposal_id>/', grant_views.approve_proposal, name='approve_proposal'),
    path('hod/monitor/<int:grant_id>/', grant_views.project_detail, name='project_detail'),
    path('hod/budget/<int:grant_id>/', grant_views.track_budget, name='track_budget'),
    path('hod/analytics/', grant_views.hod_analytics, name='hod_analytics'),
]
```

5.2.4 Frontend-Backend Integration

The "Submit Research Proposal" function demonstrates effective frontend-backend integration by enabling researchers to upload documents and enter budget data. When a researcher submits a proposal, the frontend collects the input data and sends it to the backend via a POST request. The backend processes the request, validates the form data, calculates the appropriate version number (e.g., incrementing 1.0 to 1.1 for resubmissions), and updates the Proposal database entry while saving the PDF file to the media directory.

```
@login_required
def submit_proposal(request):
    if request.user.role != 'Researcher':
        return redirect('home')

    if request.method == 'POST':
        form = ProposalForm(request.POST, request.FILES)
        if form.is_valid():
            new_proposal = form.save(commit=False)
            new_proposal.researcher = request.user.researcher

            # --- LOGIC: VERSION CONTROL ---
            # Check if a proposal with this title already exists for this user
            existing_proposals = Proposal.objects.filter(
                researcher=request.user.researcher,
                title=new_proposal.title
            )

            if existing_proposals.exists():
                # Find the highest version number
                current_max = existing_proposals.aggregate(Max('version'))['version__max']
                new_proposal.version = current_max + 0.1  # Increment version (e.g., 1.0 -> 1.1)
                messages.info(request, f"New version {new_proposal.version:.1f} created.")
            else:
                new_proposal.version = 1.0 # First submission

            new_proposal.save()
            return redirect('researcher_dashboard')
    else:
        form = ProposalForm()
    return render(request, 'grants/submit_proposal.html', {'form': form})
```

5.2.5 Role-Based Dispatching Integration

The system uses a centralized dispatching mechanism to integrate the different user roles into a single entry point. The dashboard_dispatch view identifies whether a logged-in user is a Researcher, Reviewer, or HOD and redirects them to their respective dashboard. This ensures that users are consistently directed to the correct functional module based on their credentials while maintaining a unified authentication experience.

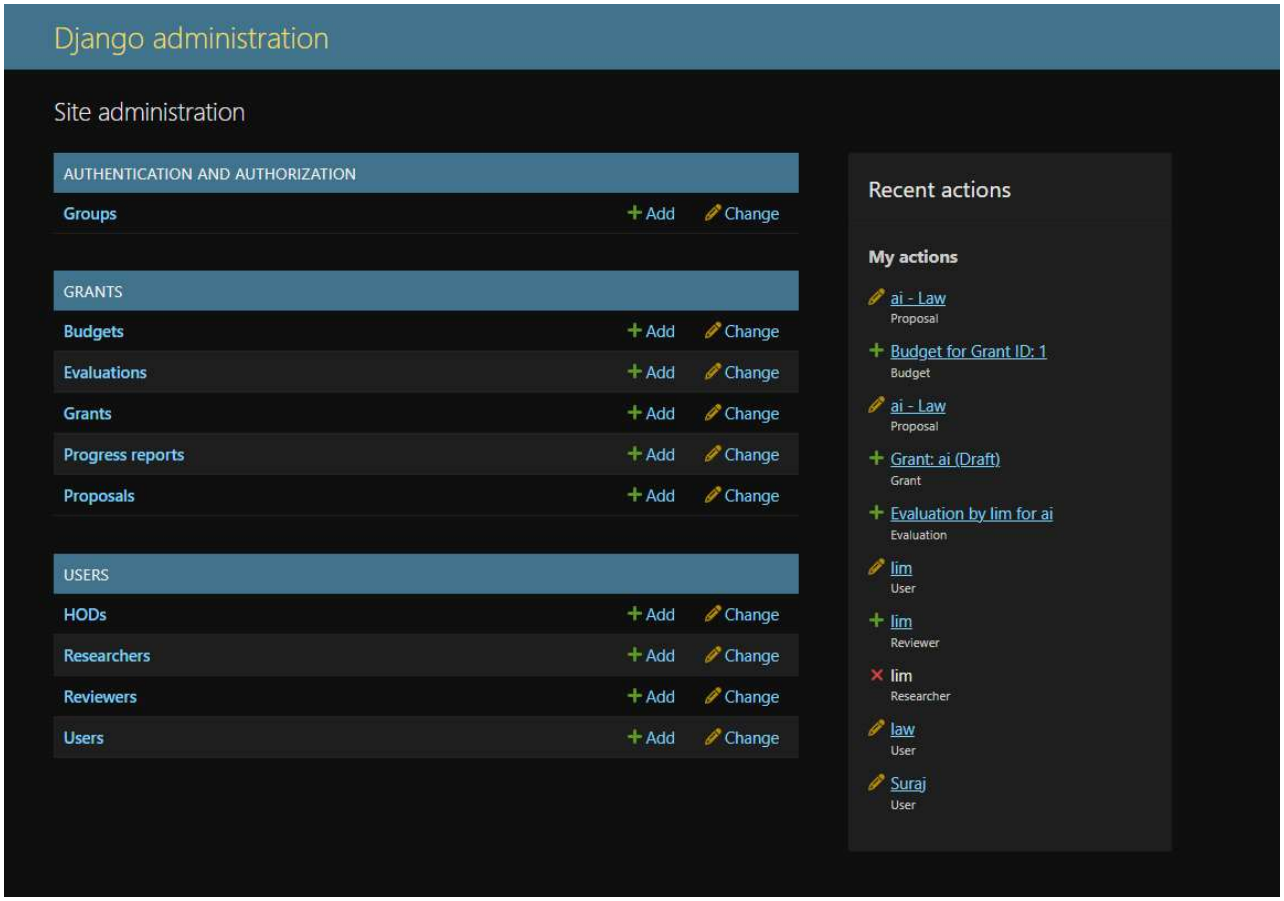
```
@login_required
def dashboard_dispatch(request):
    if request.user.role == 'Reviewer':
        return redirect('reviewer_dashboard')
    elif request.user.role == 'Researcher':
        return redirect('researcher_dashboard')
    elif request.user.role == 'HOD':
        return redirect('hod_dashboard')
    return redirect('home')
```

5.3 Database

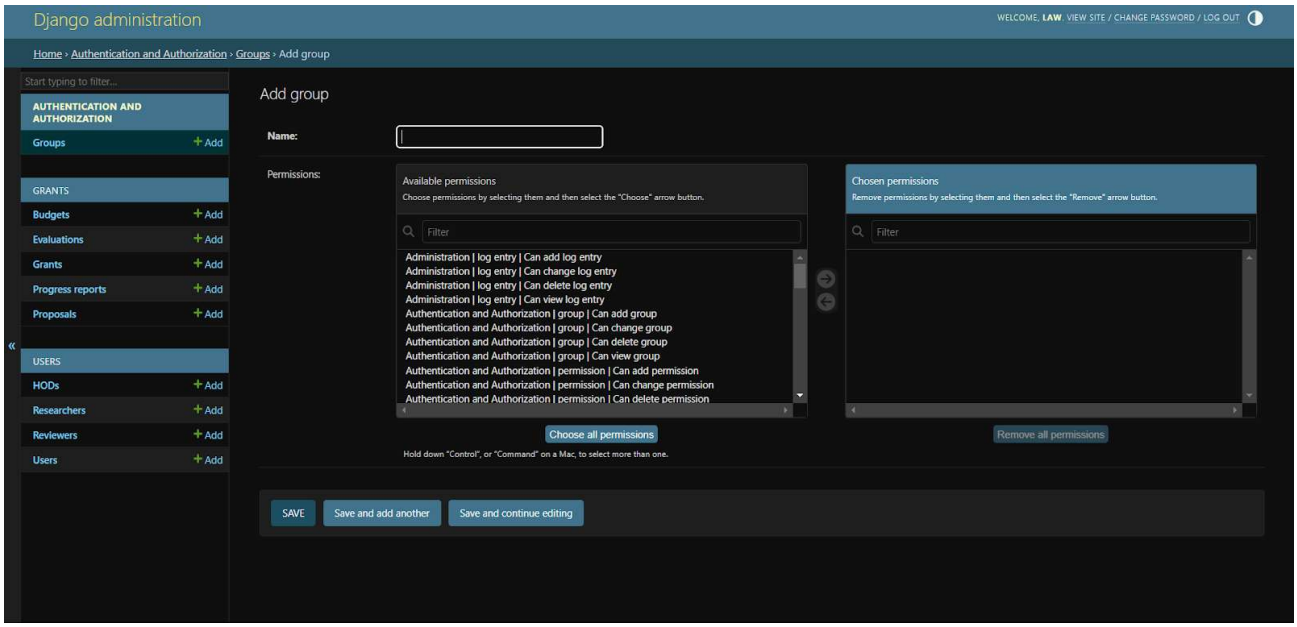
The system's database implementation utilizes the Django ORM to manage complex relationships between user roles and grant activities. To ensure efficient system administration, the **Django Administration** panel is utilized as a built-in interface for managing project data.

- **User Profiles**: The system uses a custom User model with defined roles (Researcher, Reviewer, HOD) and specialized profile models to store role-specific data, such as a Researcher's department or an HOD's total department budget.
- **Grant Lifecycle Tracking**: Data tables for Proposal, Evaluation, Grant, Budget, and ProgressReport are registered in the admin panel to allow for easy oversight and manual data adjustment if necessary.
- **Financial Integrity**: The Budget model tracks totalSpent against the totalAllocatedAmount, with logic in the views.py ensuring HODs are alerted when spending exceeds 90%.

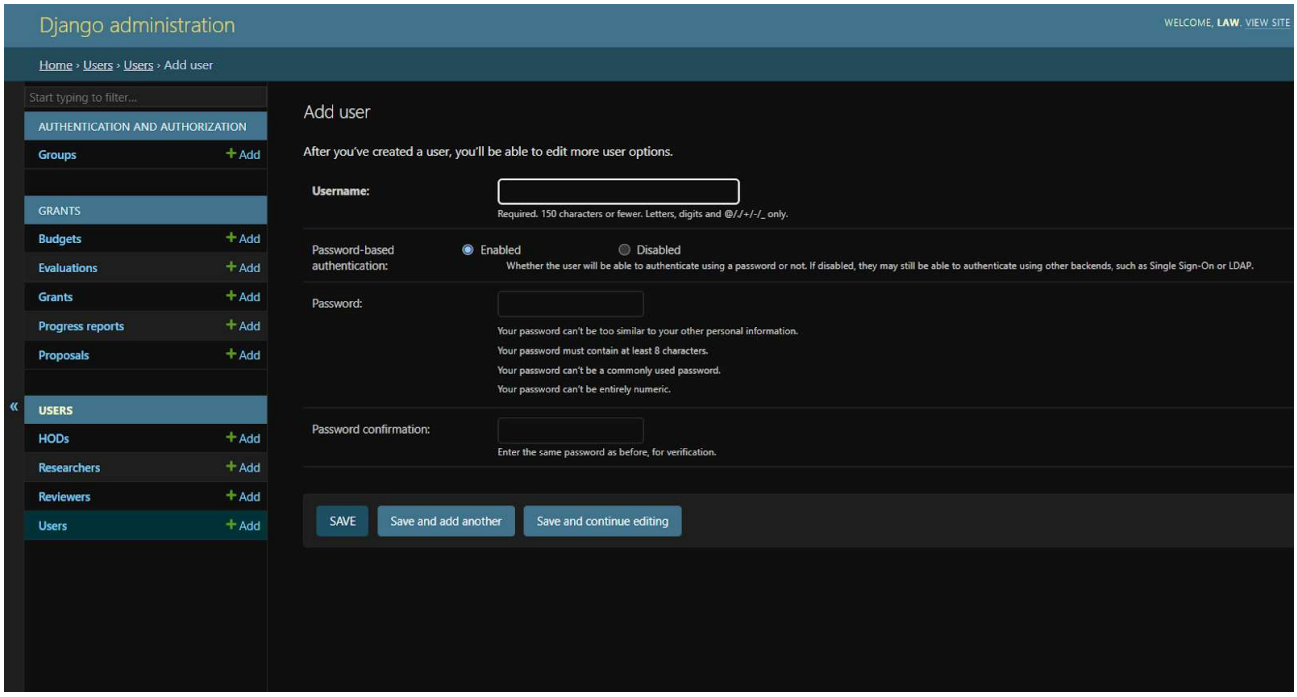
5.3.1 Homepage Django Administration



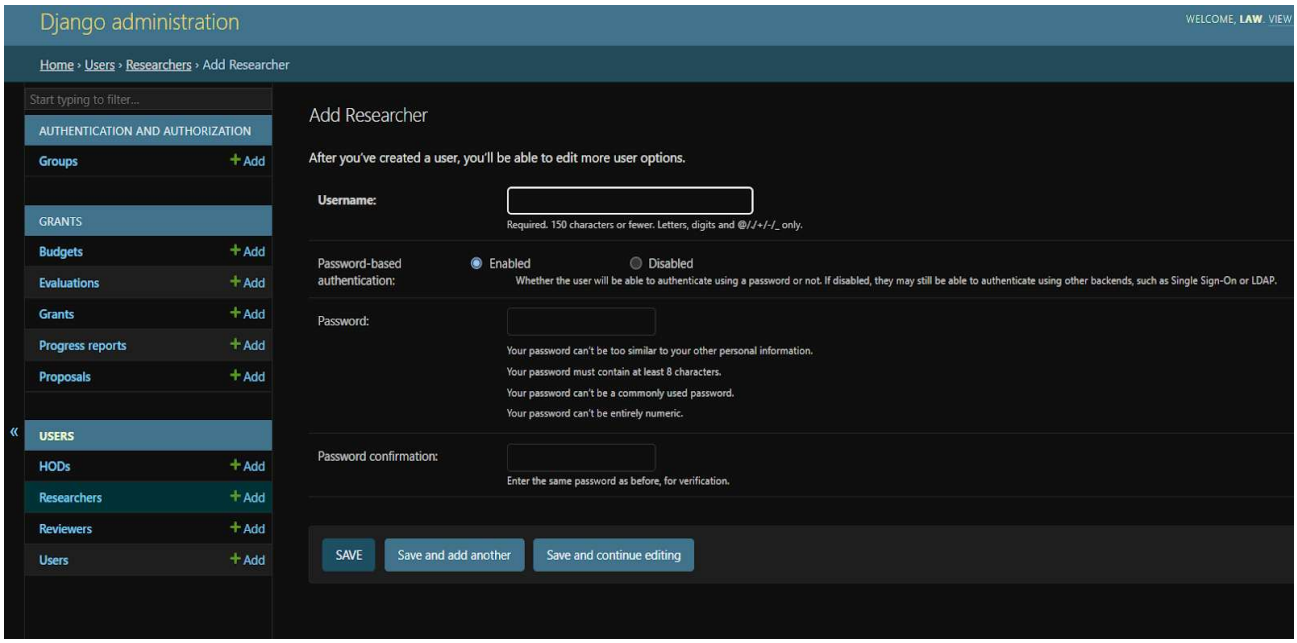
5.3.2 Add Group Django Administration

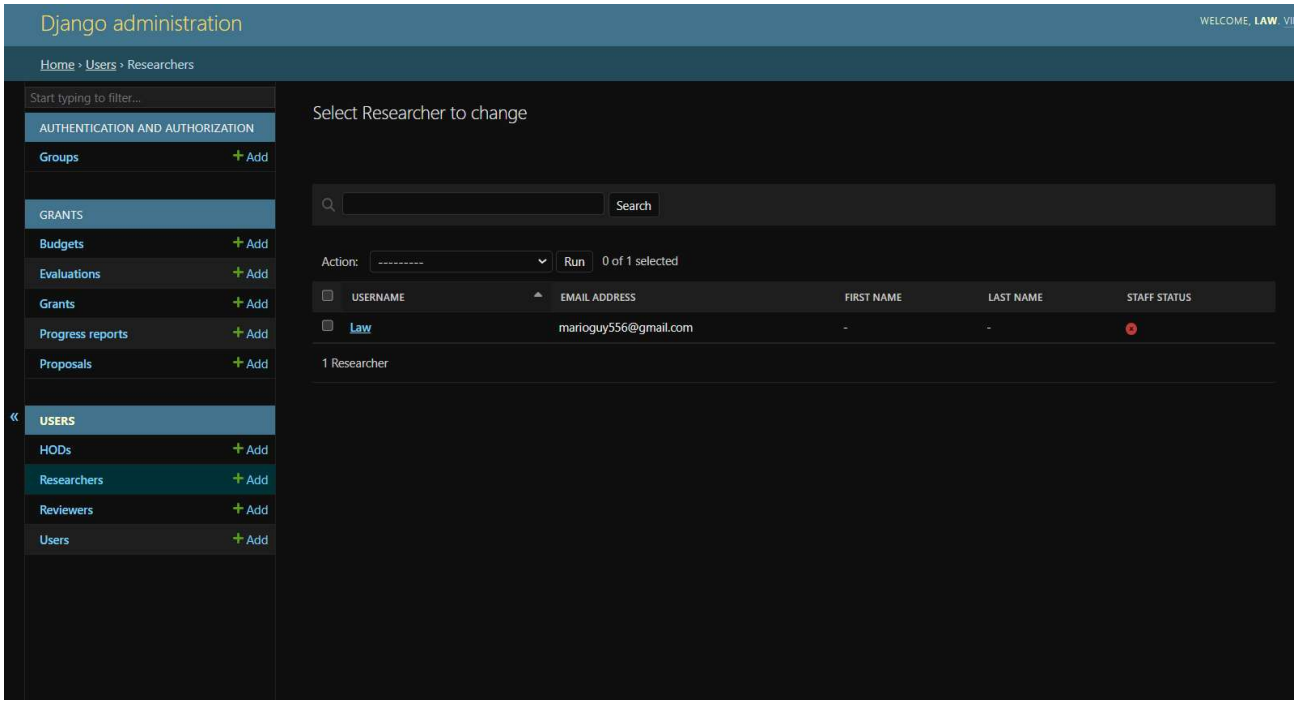


5.3.3 Add User Django Administration

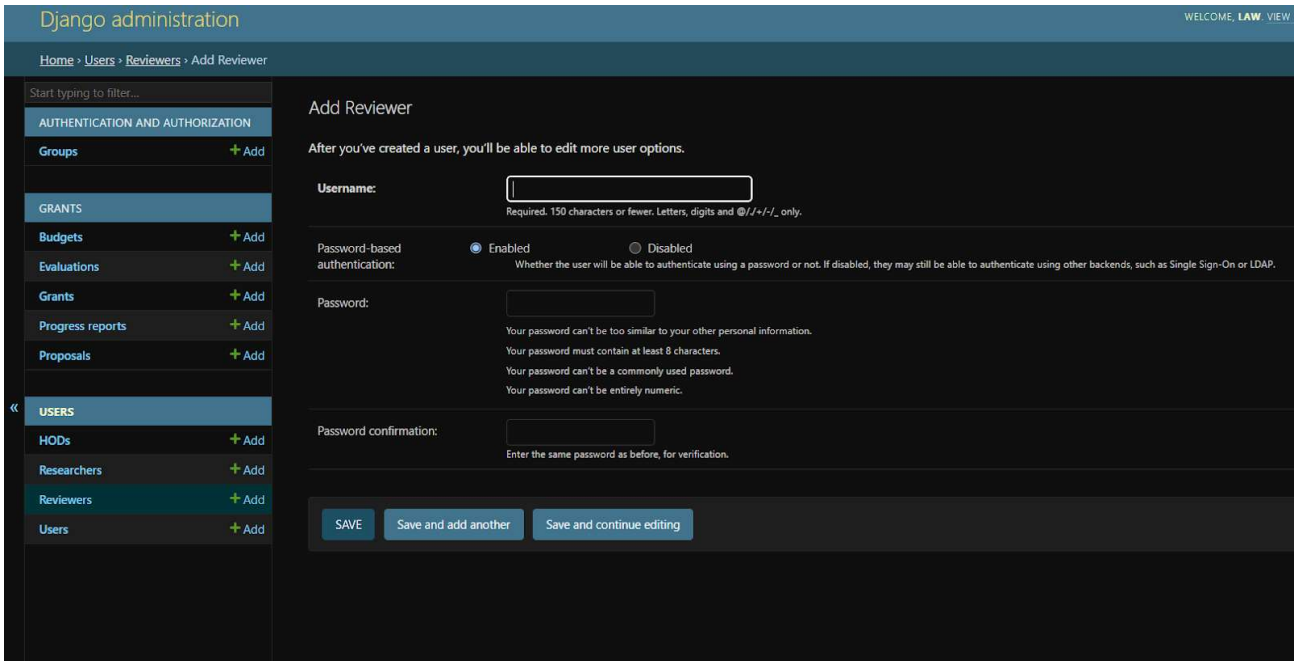


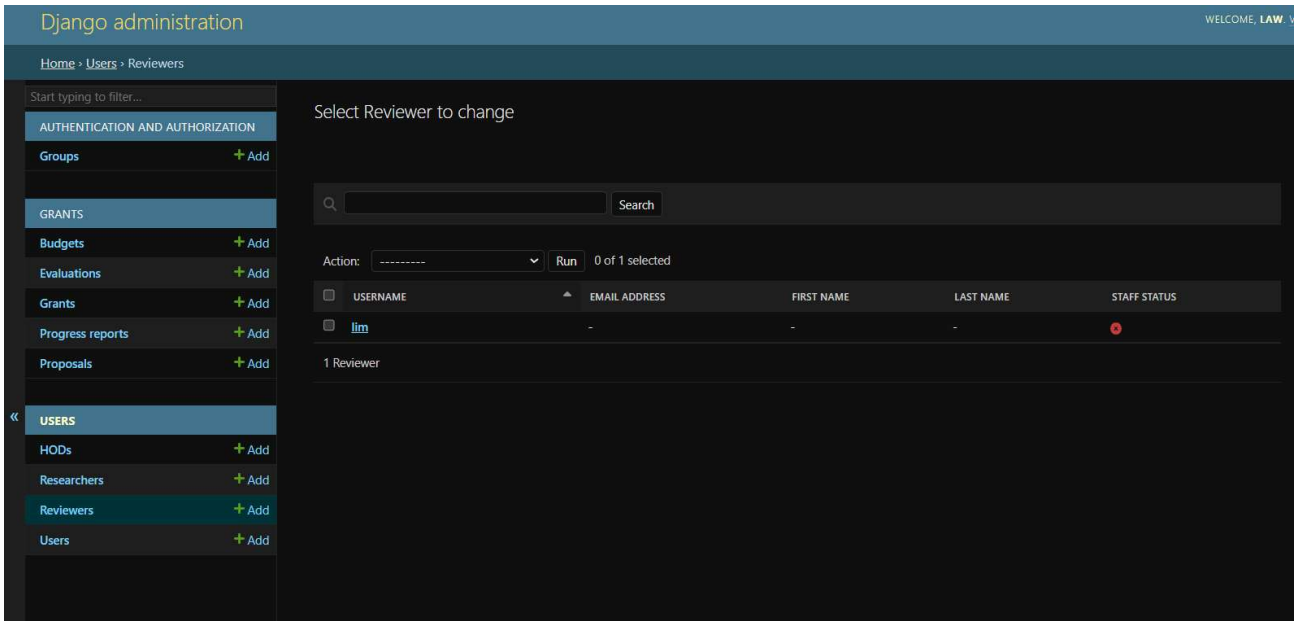
5.3.4 Add Researcher Django Administration



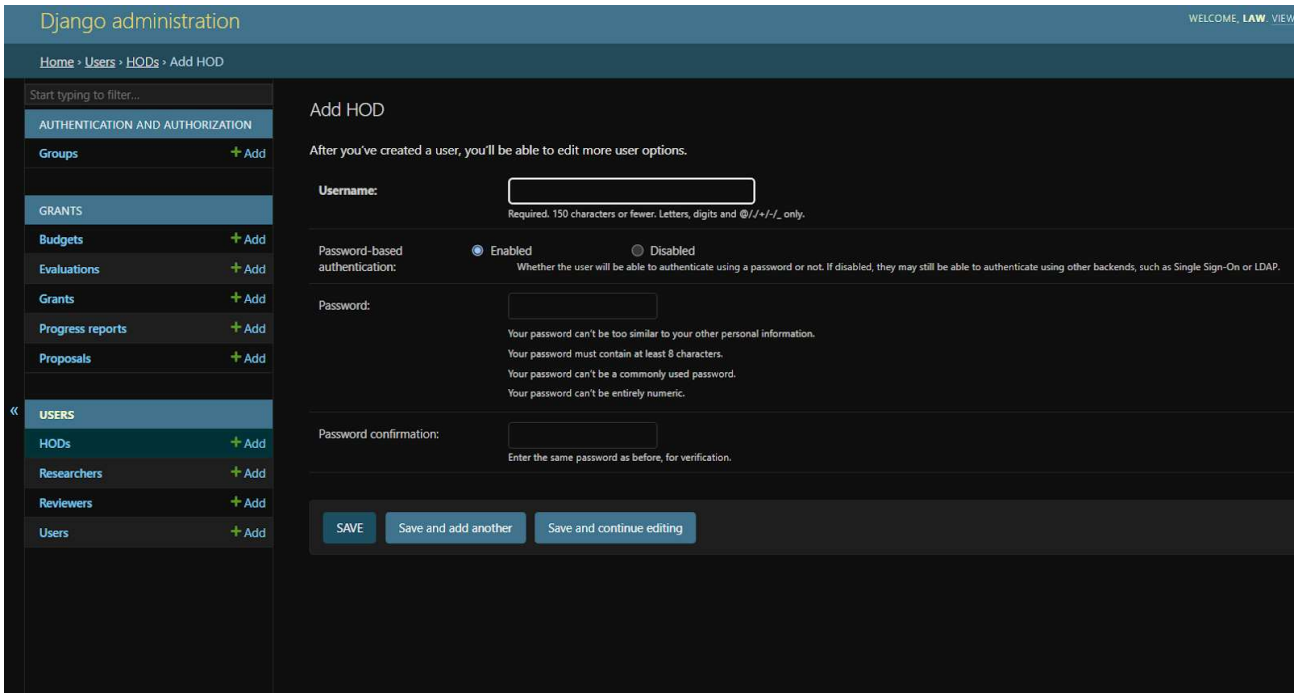


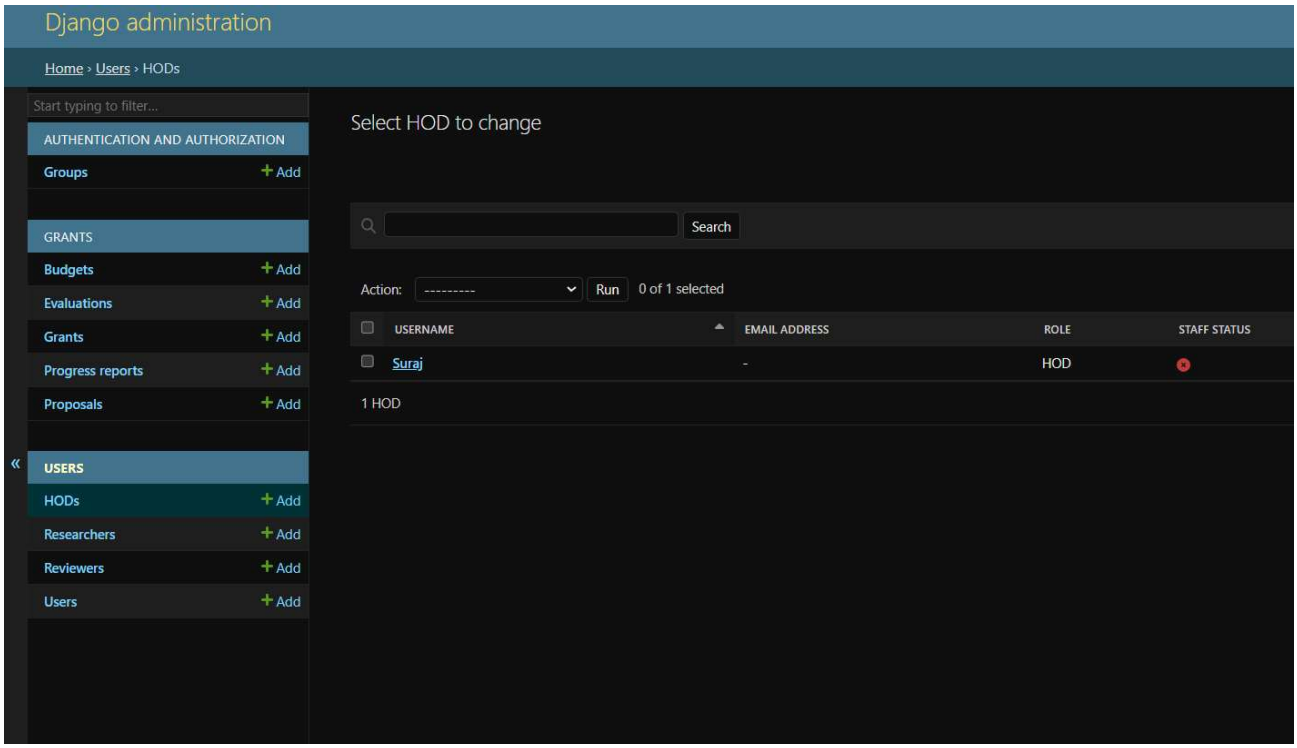
5.3.5 Add Reviewer Django Administration



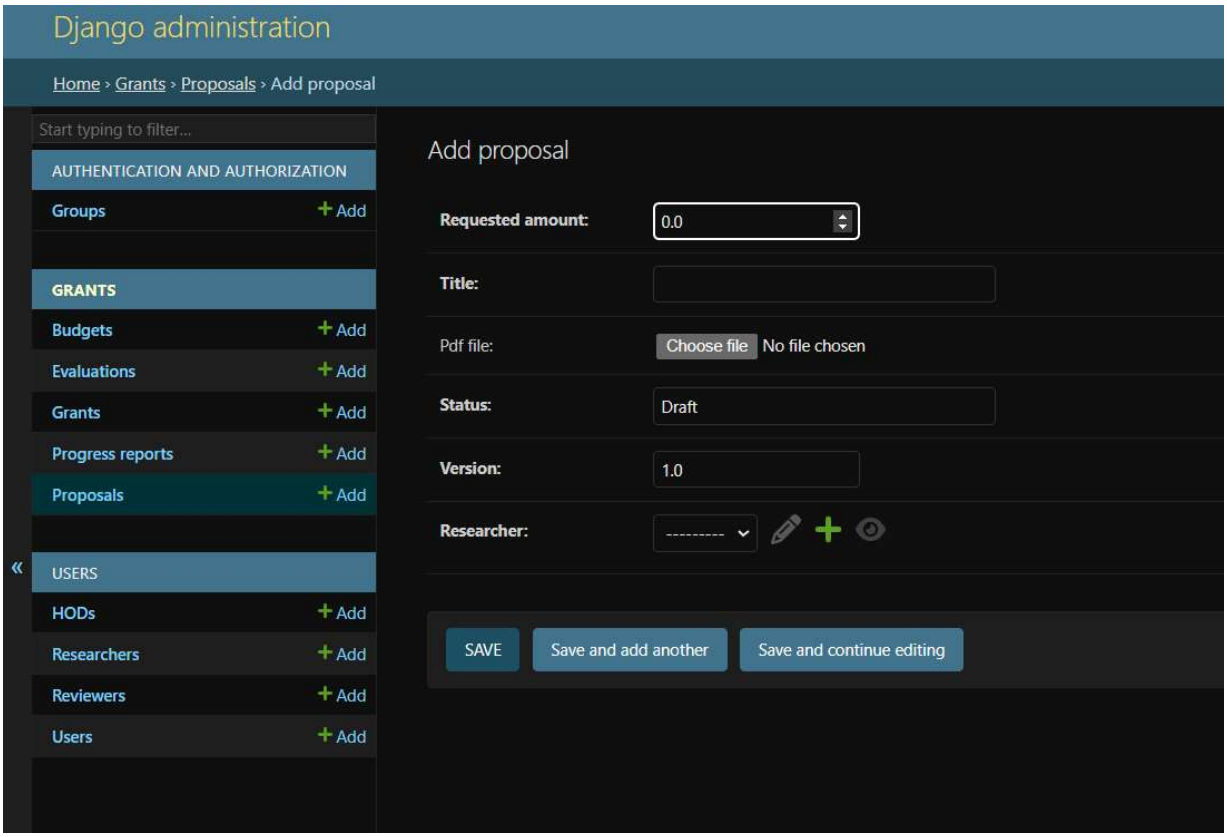


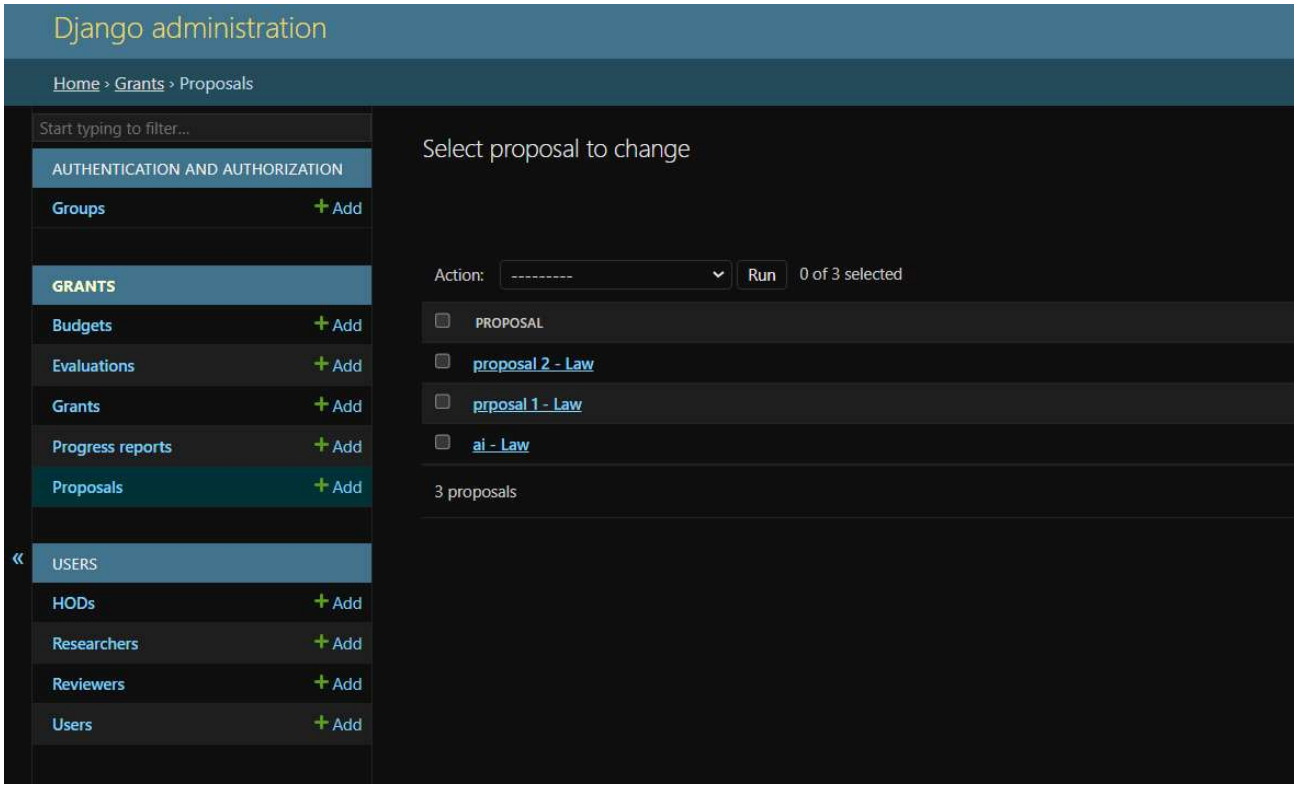
5.3.6 Add HOD Django Administration



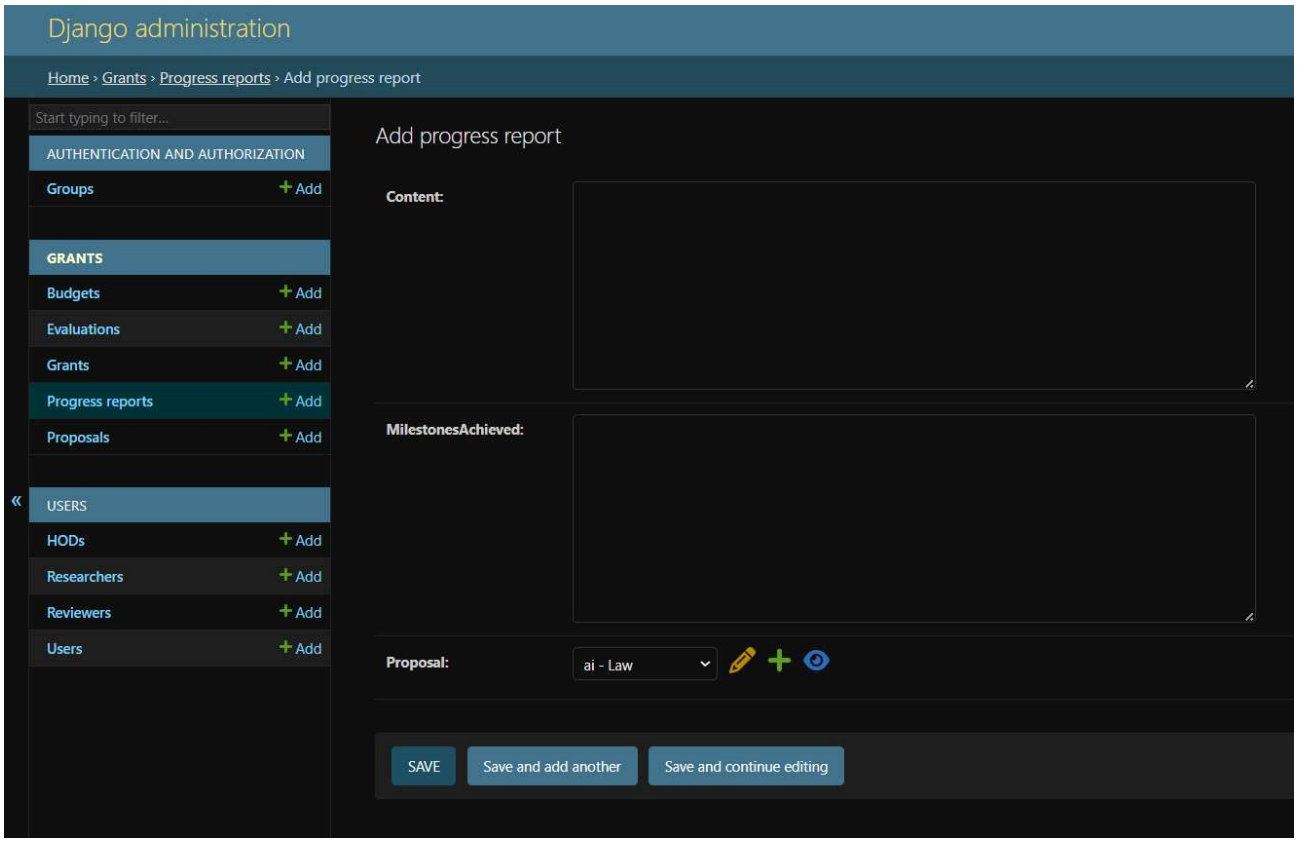


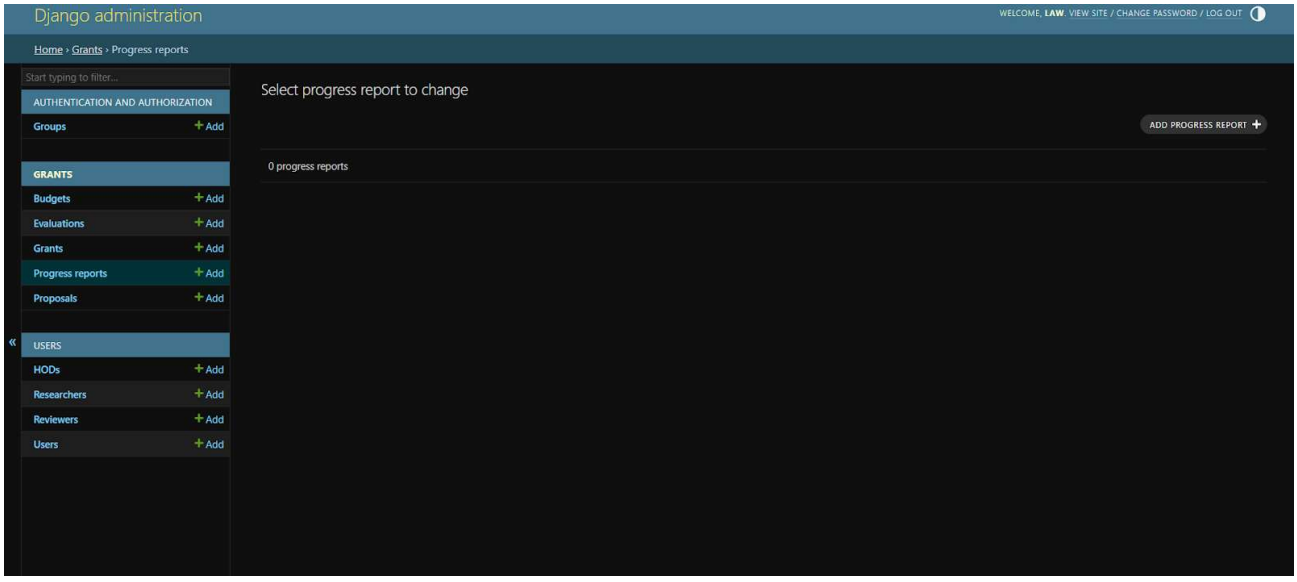
5.3.7 Add Proposal Django Administration



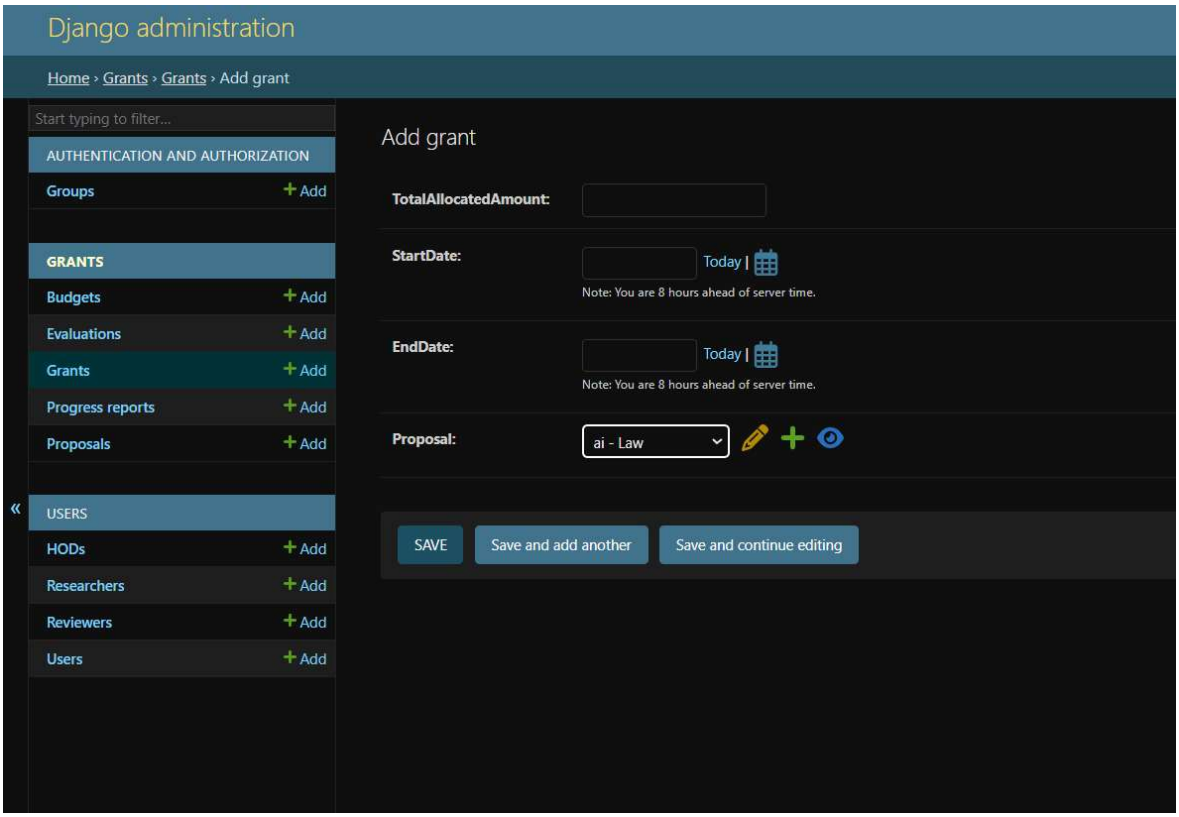


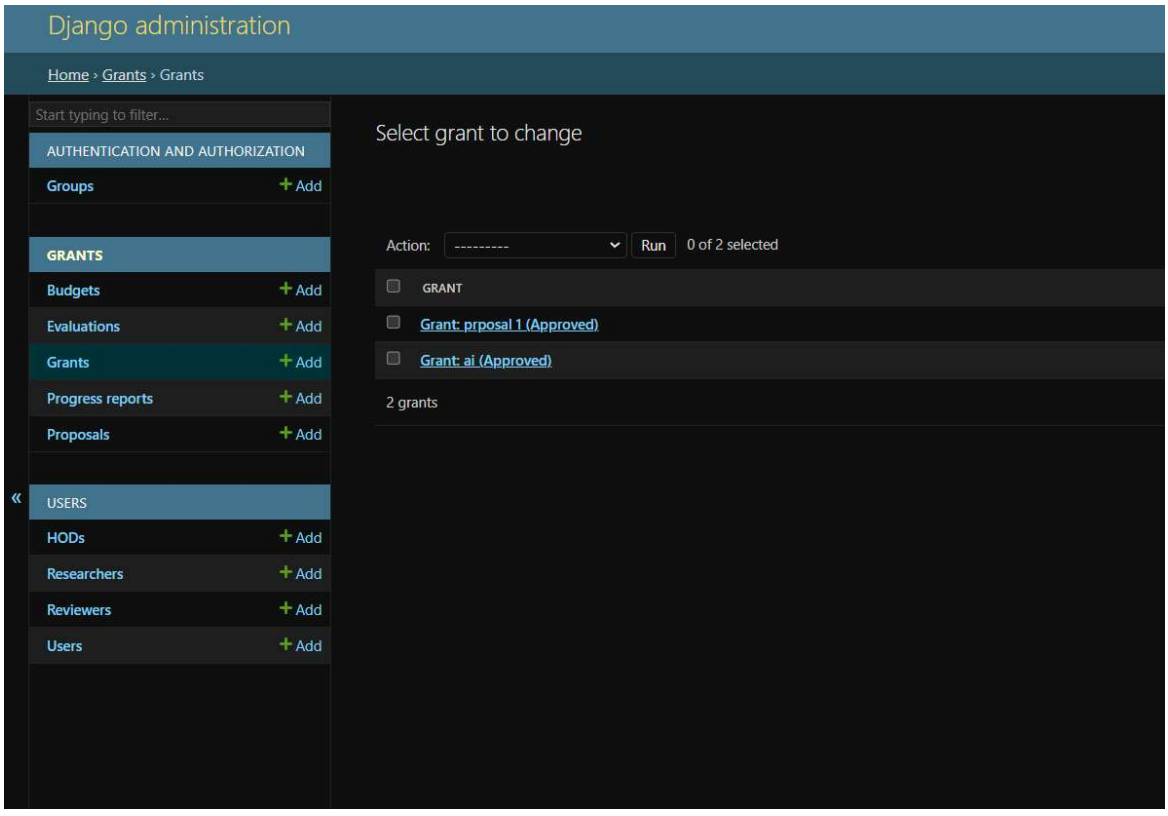
5.3.8 Add Progress Reports Django Administration



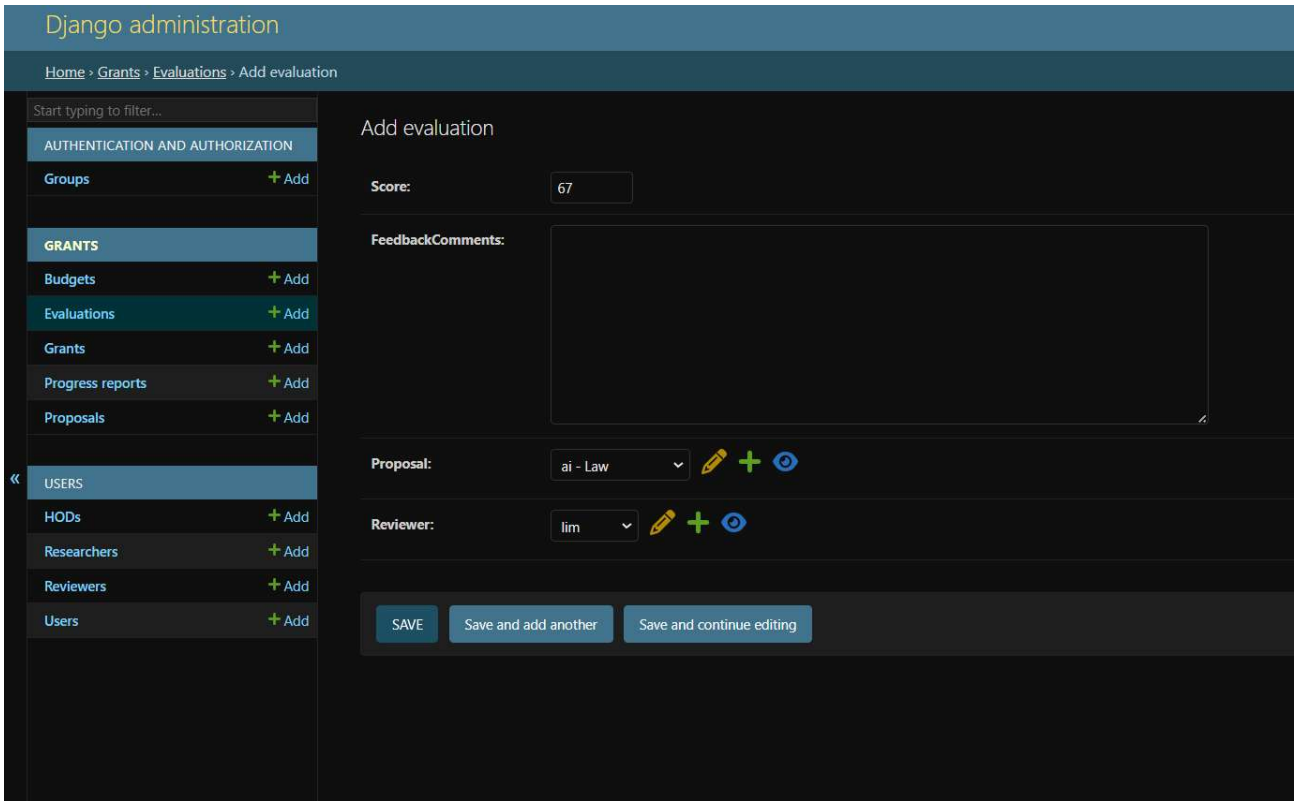


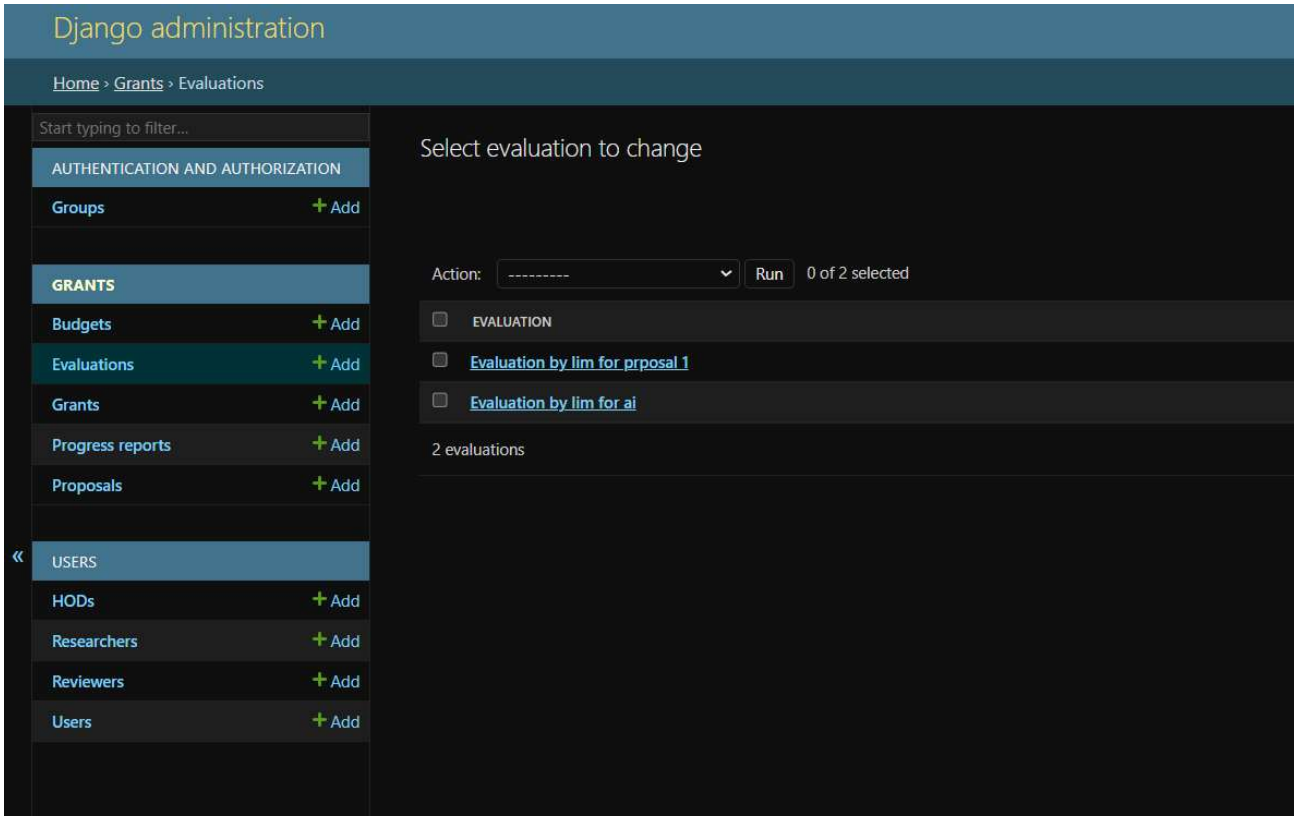
5.3.9 Add Grants Django Administration



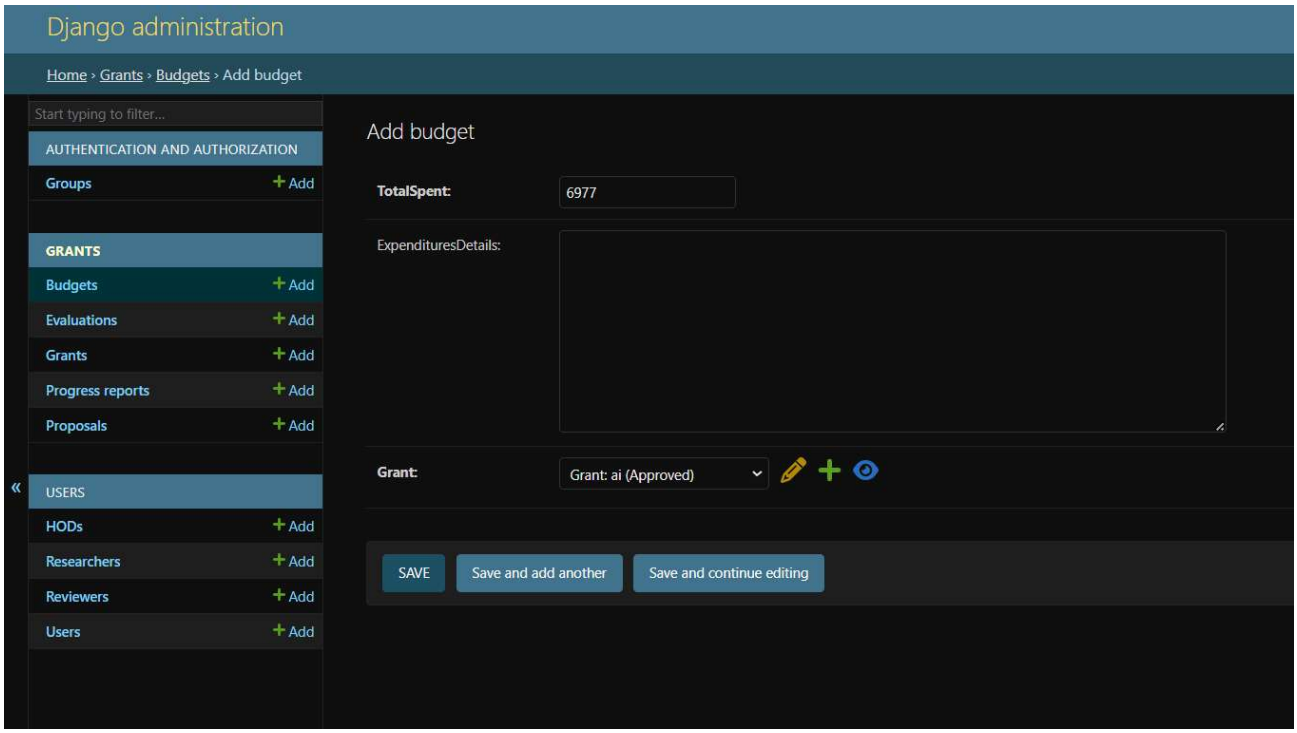


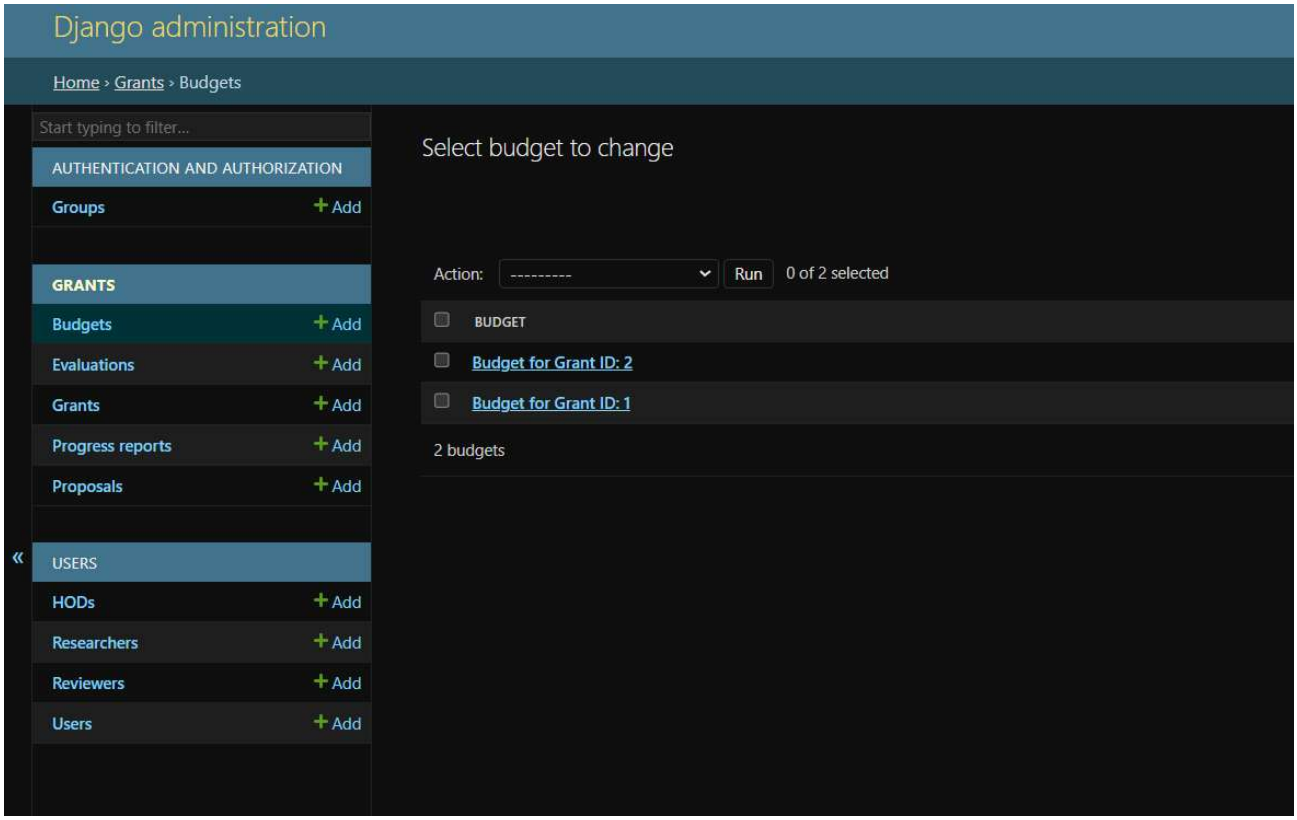
5.3.10 Add Evaluations Django Administration





5.3.11 Add Budgets Django Administration





6 Testing

6.1 Testing Strategy

<*TO DO: Describe the integration testing strategy here.*>

The testing phase for the Research Grant Management System (RGMS) was conducted using a **Bottom-Up Testing Approach**. This strategy ensures that individual modules (Models and Forms) are verified for correctness before being integrated into larger subsystems (Views and Templates), and finally tested as a complete system.

The testing process was divided into three distinct levels:

6.1.1 Unit Testing

Unit testing focused on the smallest testable parts of the application to ensure that the core logic functioned in isolation.

- **Data Integrity:** Verified that Django Models (`Proposal`, `Grant`, `Budget`) correctly enforced constraints (e.g., ensuring `requested_amount` is a positive integer).
- **Form Validation:** Tested `forms.py` to ensure user inputs were validated before processing (e.g., preventing submission of PDF files larger than the allowed limit).
- **Method Logic:** Verified specific internal methods, such as the `calculate_remaining_budget()` function in the Financial Tracker.

6.1.2 Integration Testing

Since the system architecture relies on a **Shared Repository Pattern** (SQLite Database), integration testing focused on the data flow between the three distinct subsystems:

- **Subsystem Interaction:** Verified that a proposal created in the **Researcher Subsystem** was immediately visible and accessible in the **Reviewer Subsystem**.
- **State Transitions:** Tested the workflow logic to ensure that an action in one module (e.g., HOD clicking "Approve") correctly triggered updates in related modules (e.g., creating a `Grant` record and deducting from the `DepartmentFund`).
- **Database Consistency:** Ensured that concurrent read/write operations (e.g., a Reviewer submitting a score while an HOD checks the dashboard) did not result in data corruption.

6.1.3 System Testing (User Acceptance Testing)

System testing involved "Black Box" testing of the **integrated application running in the local development environment**. This phase simulated real-world usage by the three primary actors (Researcher, Reviewer, HOD) interacting with the system via the web browser.

- **End-to-End Workflows:** Validated the complete lifecycle of a grant, from initial registration and proposal submission to final fund allocation and project monitoring.
- **Guard Condition Verification:** Specifically tested business rule enforcement, such as:
    - Preventing a Reviewer from grading a proposal they are not assigned to.
    - Preventing the HOD from approving a grant if the Department Reserve funds are insufficient.
- **UI/UX Responsiveness:** Verified that the Dashboard templates rendered correctly and that feedback messages (e.g., "Submission Successful") were displayed to the user effectively.

6.2 Test Data

<*TO DO: Provide a detailed description of the test data used to verify the system here.*>

6.2.1 Test Data (Researcher Actor)

To verify the functional requirements of the Researcher, the following datasets were designed to simulate the complete lifecycle of a research grant from submission and version control to budget tracking and reporting. These datasets ensure that the system's logic for document management and financial calculation remains accurate under both standard and edge-case conditions.

A. Actor Profile (The Researcher Account)

This account is used to validate all researcher workflows.

- **Username**: researcher_01
- **Role**: Researcher
- **Department**: FCI
- **Research Interests**: AI, Data Science, and Machine Learning

6.2.2 Test Data (Reviewer Actor)

6.2.3 Test Data (HOD Actor)

To verify the functional requirements of the Head of Department (HOD), the following specific datasets were created. These datasets simulate different financial scenarios to ensure the system's "Guard Conditions" and database logic function correctly.

A. Actor Profile (The HOD Account)

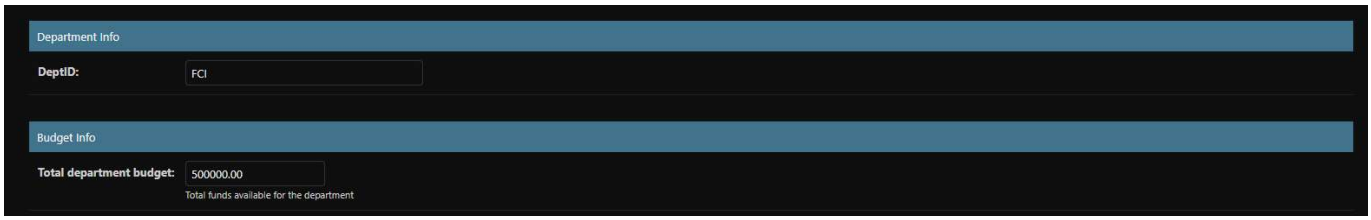
This account is used to perform all HOD actions.

Username: hod_admin

Role: Head of Department (HOD)

Department: FCI

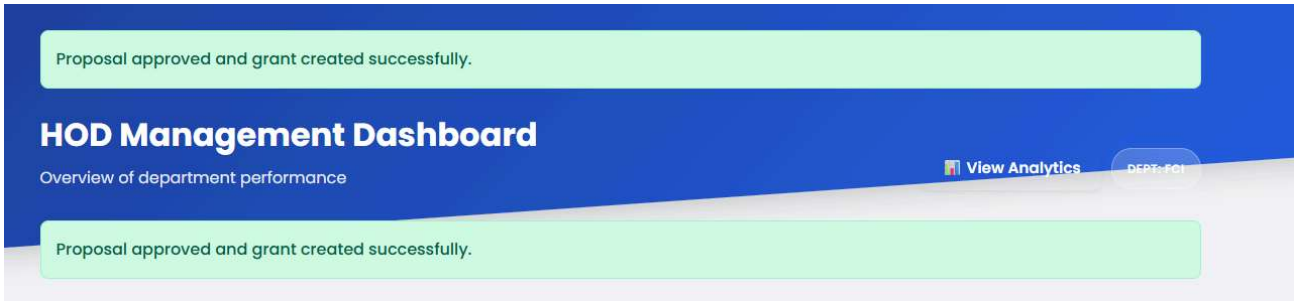
Total Department Budget: RM 500,000.00 *(This is the critical validation limit)*



Data Set A: For Use Case 1 (Approve/Reject Proposal)

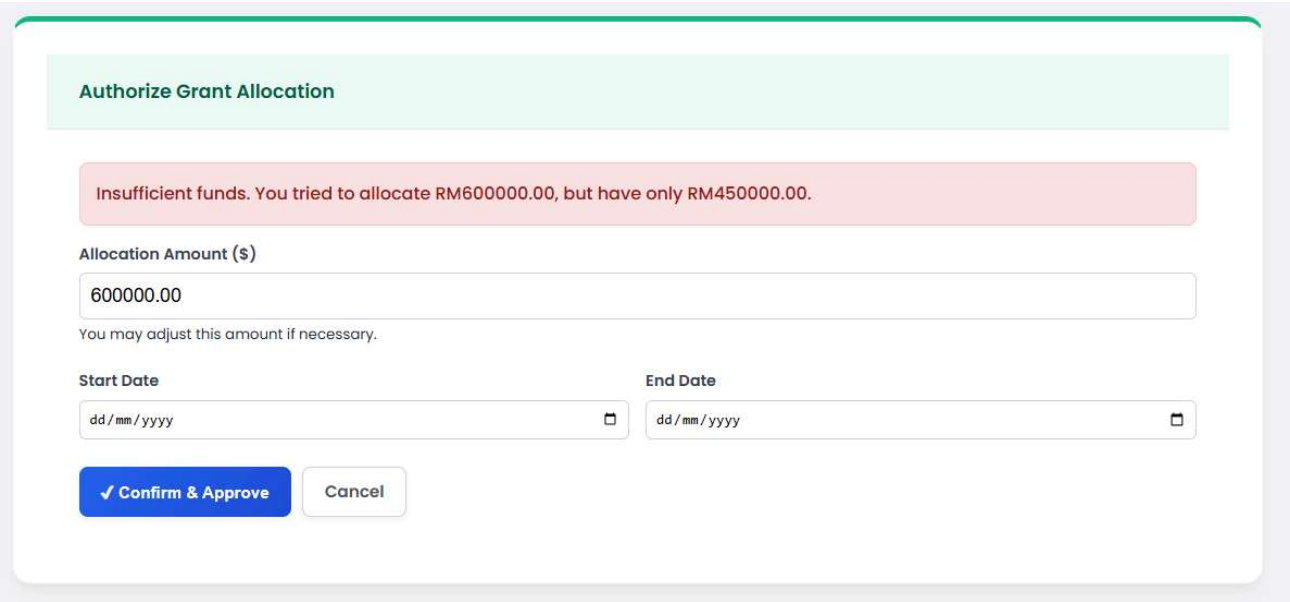
Used to test the decision logic and the "Insufficient Funds" guard condition.

- **Scenario 1 (Valid Approval):**
  - **Proposal:** "AI-Driven Crop Monitoring"
  - **Requested:** RM 50,000.00
  - **Action:** Approve **Expected:** Success.



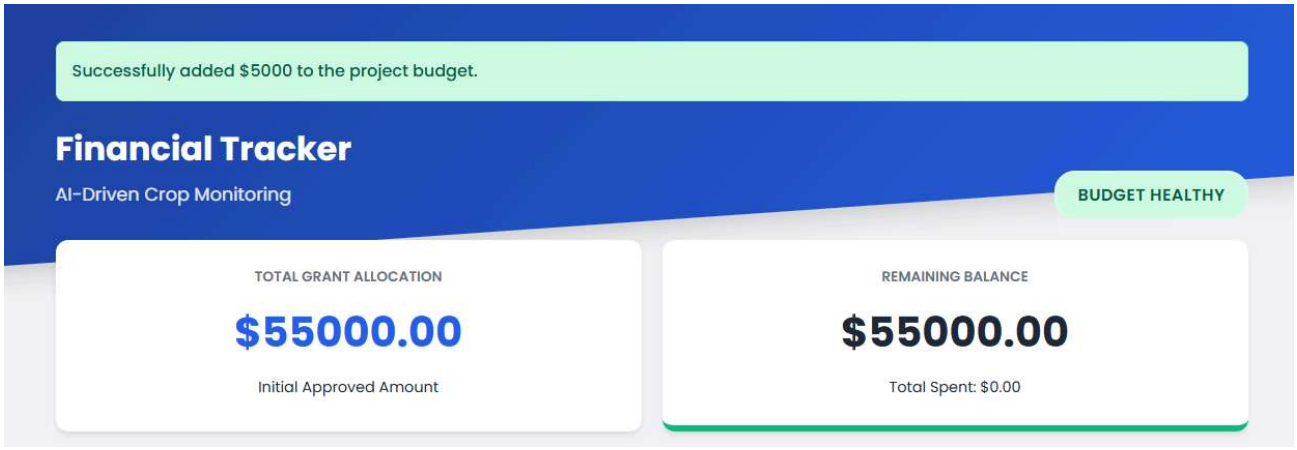
Department Budget Deducted:

- **Scenario 2 (Budget Overflow):**
  - **Proposal:** "Quantum Supercomputer Setup"
  - **Requested:** RM 600,000.00 (Exceeds Dept Limit)
  - **Action:** Approve **Expected:** System blocks action.

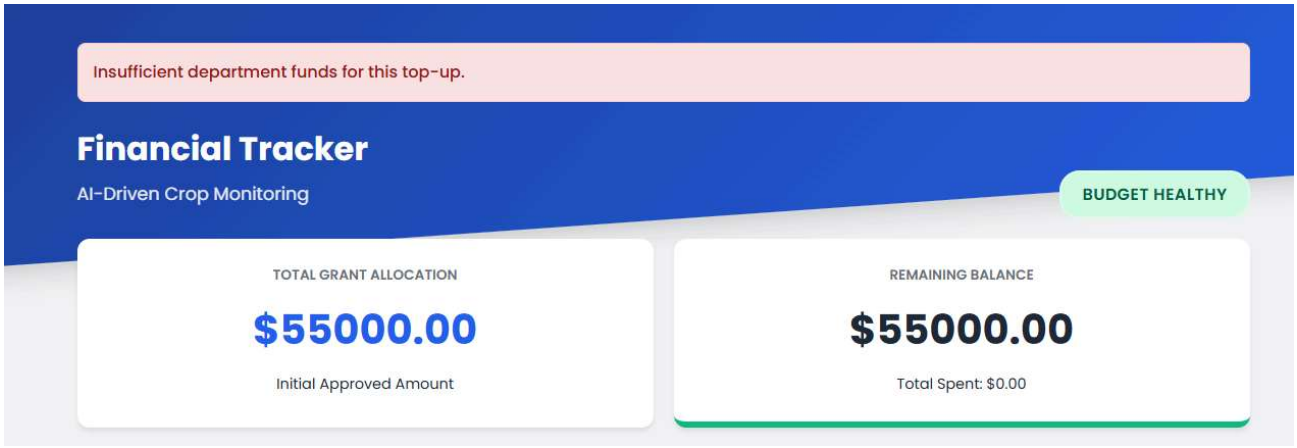


Data Set B: For Use Case 2 (Manage Budget / Top-Up)

- **Scenario B1 (Positive - Successful Top-Up):**
  - *Input:* Active Grant "AI-Driven Crop Monitoring".
  - *Action:* Add **RM 5,000.00**.
  - *Condition:* Dept Reserve has funds.
  - *Expected:* Grant Total increases to RM 55,000.

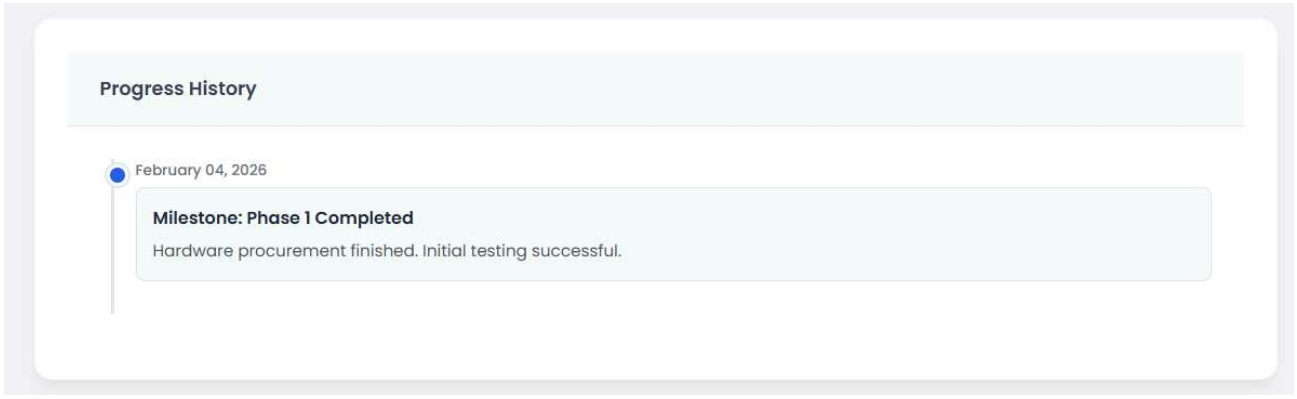


- **Scenario B2 (Negative - Top-Up Exceeds Reserve):**
  - *Input:* Active Grant "AI-Driven Crop Monitoring".
  - *Action:* Add **RM 450,000.00** (Assume Reserve is only RM 400k).
  - *Expected:* System Error. Top-up rejected to prevent department bankruptcy.



Data Set C: For Use Case 3 (Monitor Project Progress)

- **Scenario C1 (Positive - Report Visible):**
  - **Prerequisite:** Researcher has submitted a report for "Smart Traffic IoT".
  - **Input Data (Report):**
    - *Milestone:* "Phase 1 Completed"
    - *Content:* "Hardware procurement finished. Initial testing successful."
  - **Expected Result:** HOD sees "Phase 1 Completed" on the timeline. Status remains "On Track".



- **Scenario C2 (Negative - Needs Intervention):**
  - **Input Data (HOD Action):**
    - *Status Flag:* "Needs Intervention"

        - *Feedback:* "Project is delayed by 2 weeks. Please submit explanation."
    - **Expected Result:** Proposal status updates to **"Needs Intervention"**. A generic report entry is created in the timeline with the prefix "URGENT INTERVENTION".

Data Set D: For Use Case 4 (View Analytics)

- **Scenario D1 (Positive - Chart Calculation):**
    - **HOD Budget:** RM 500,000.00
    - **Total Spent (Actual):** RM 100,000.00 (Sum of `Budget.totalSpent`)
    - **Proposal Counts:** 3 Approved, 7 Pending/Rejected (Total 10).
    - **Expected Result:**
        - *Budget Chart:* 20% Spent (Red), 80% Remaining (Green).
        - *Status Chart:* 30% Approved (Blue), 70% Other (Grey).
- **Scenario D2 (Boundary - Zero Data):**
    - **Input:** System has **0 Proposals** and **RM 0.00 Spent**.
    - **Expected Result:**
        - `approval_rate` calculates to 0.
        - Charts render with 0 values without crashing the page.

6.3 Acceptance Testing

<*TO DO: Prepare the acceptance test for each team member*>

6.3.3

Criteria	Fulfilled	Remarks

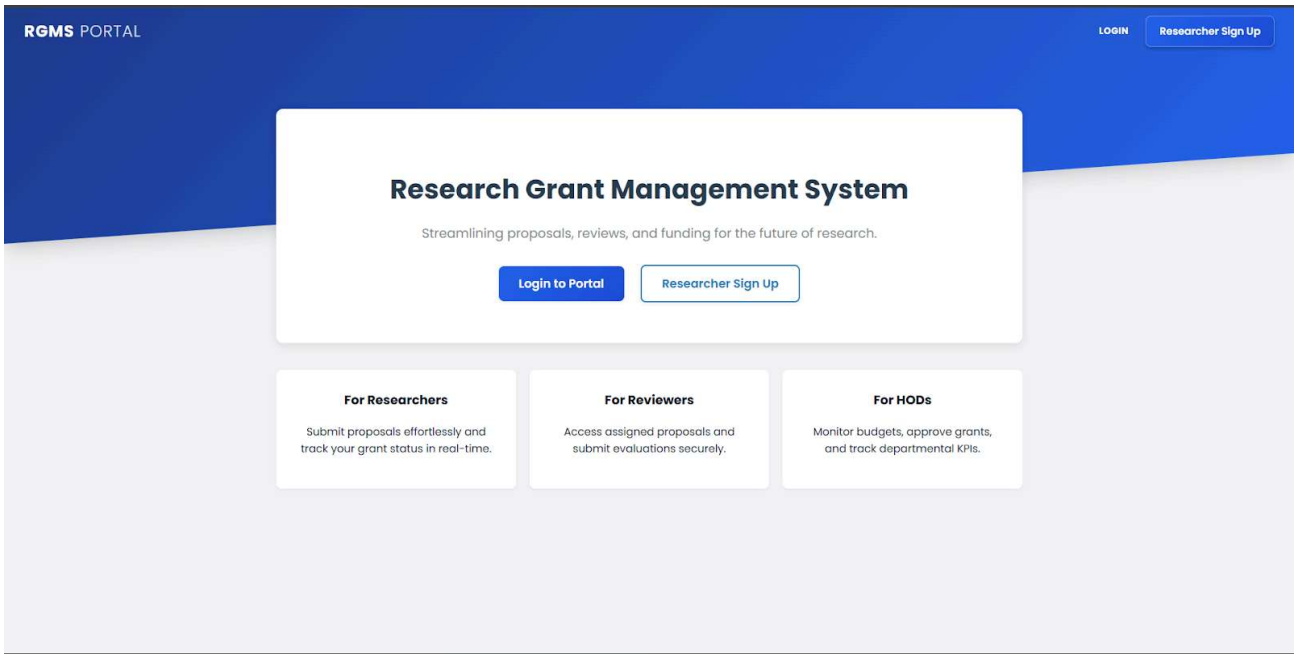
Approve/Reject Proposal	Complete	-
Manage/ Top-up budget	Complete	-
Monitor Project Progress	Complete	-
View Analytics	Complete	-

Date tested : 5/2/2026
% Complete : 100%
Tested by :Suraj A/L Prakash
Verified by :Law Jun Feng
…

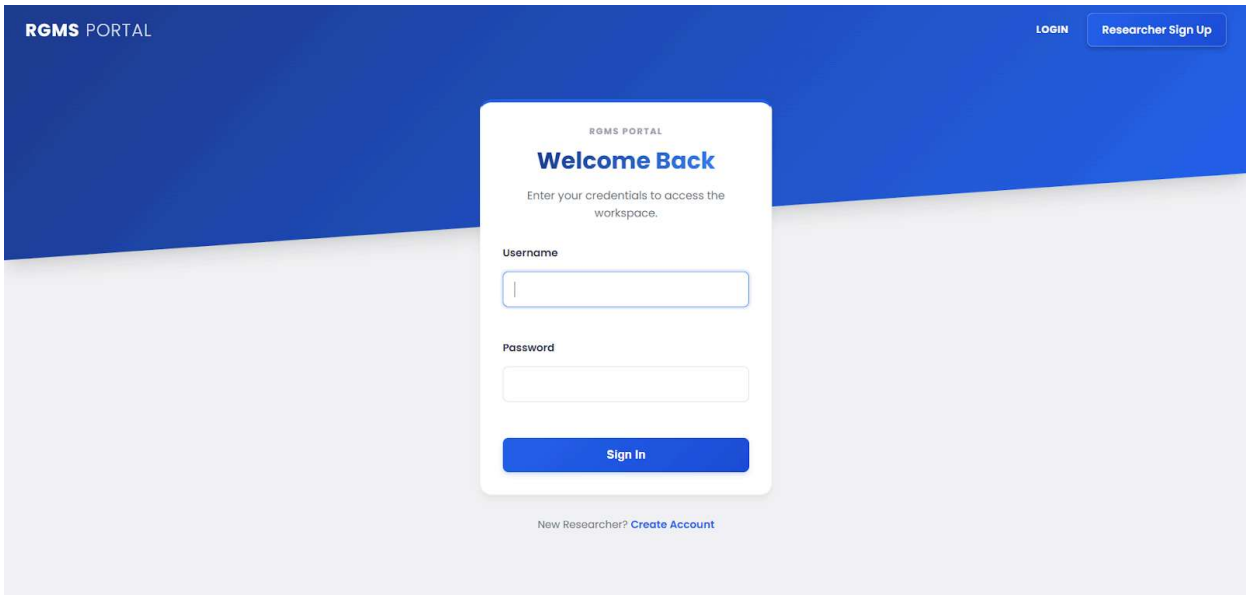
7 Sample Screens

7.2 Main Screen

The image below shows the **landing page** all users will first land on. From here users can either navigate to the login **portal** or **sign up** as a researcher.



The image below shows the **login page** for all users. Users who already have an account can login from here while users who have not signed up, can proceed to **sign up**.



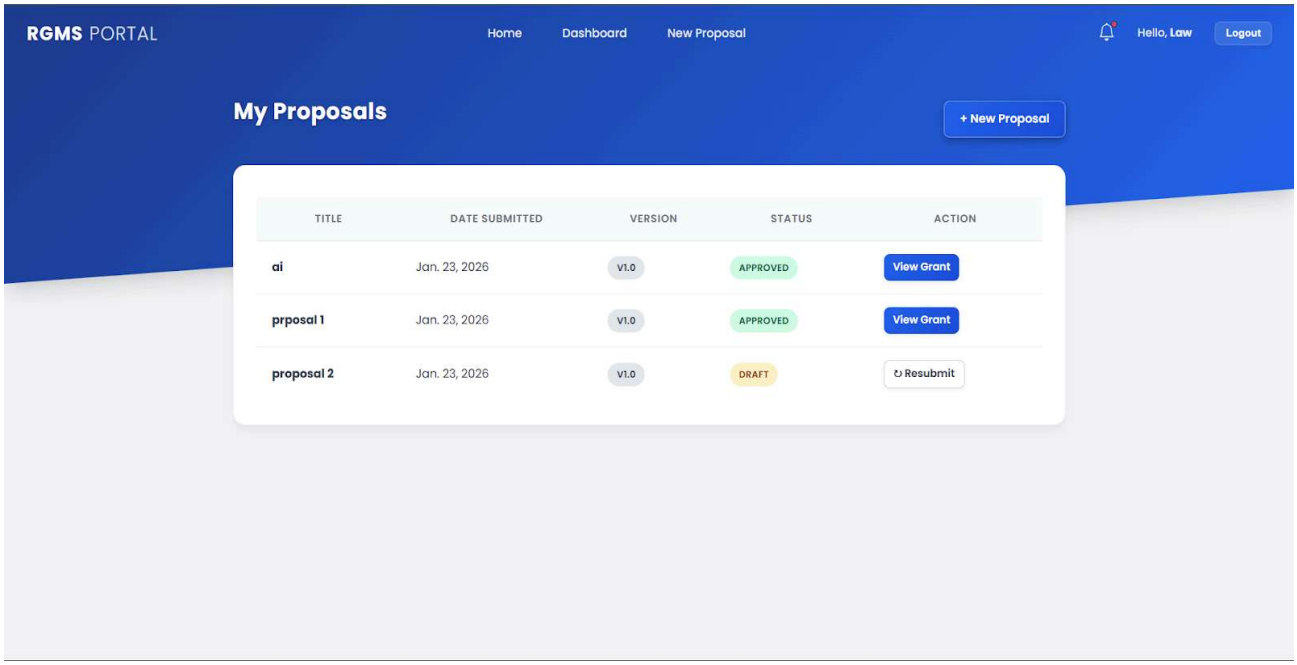
<*TO DO: Describe and place the main screens of the system here.*>

7.2 Subsystem 1 Screens: Researcher Subsystem

7.2.1 Researcher Dashboard

Researcher Dashboard The primary landing page that aggregates the Researcher's activities. The interface features a prominent Hero section with a skewed geometric blue background.

- **KPI Grid:** Uses a 4-column CSS grid (.info-grid) to display high-level metrics (e.g., Active Grants, Pending Proposals).
- **Summary Table:** A clean, spaced table layout listing recent proposals with color-coded status badges (.badge-success for Approved, .badge-warning for Pending).



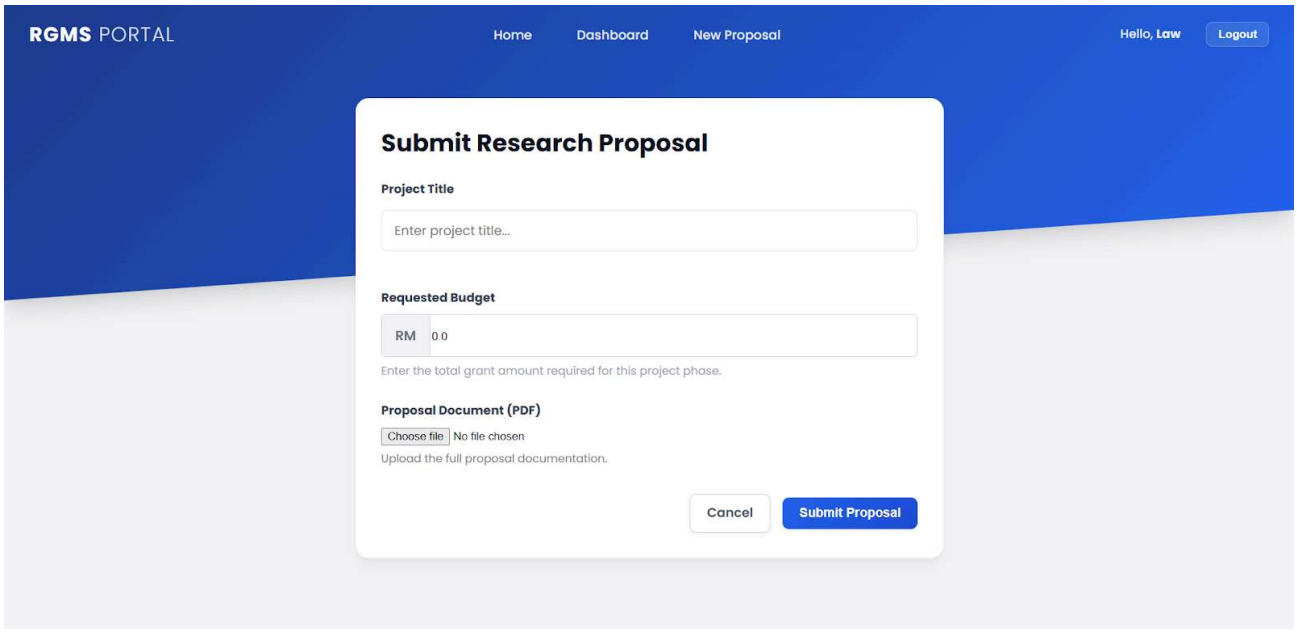
<*TO DO: Describe and place the screens of subsystem 1 here.*>

7.2.2 Submit Proposal or Resubmit Proposal

Submit Proposal Form A standardized data-entry interface designed for minimal friction.

- **Form Layout**: Clean form groups with subtle focus-state borders that guide the user's attention.

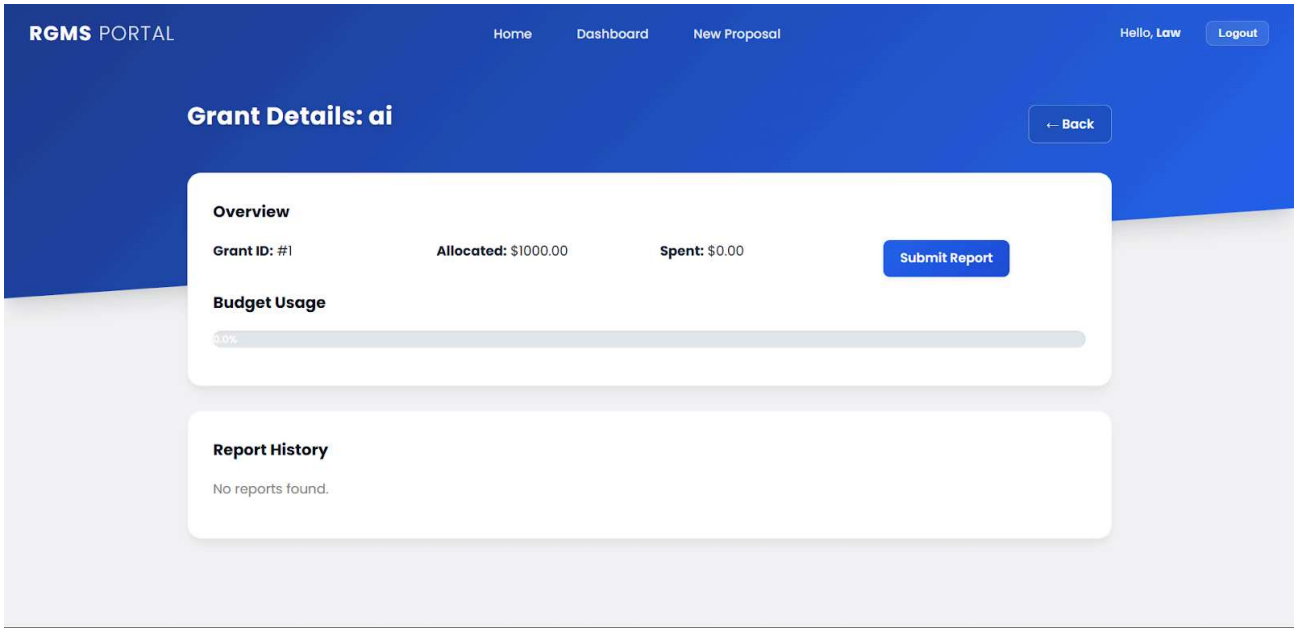
- **Document Upload:** Includes a dedicated file input field for the Research Proposal PDF.
- **Action Prompts**: A prominent, gradient-styled Primary Button confirms the submission.



7.2.3 Grant Details & Financial View

Grant Details & Financial View A read-only information hub for an approved grant. This interface uses the custom .card component to organize complex data.

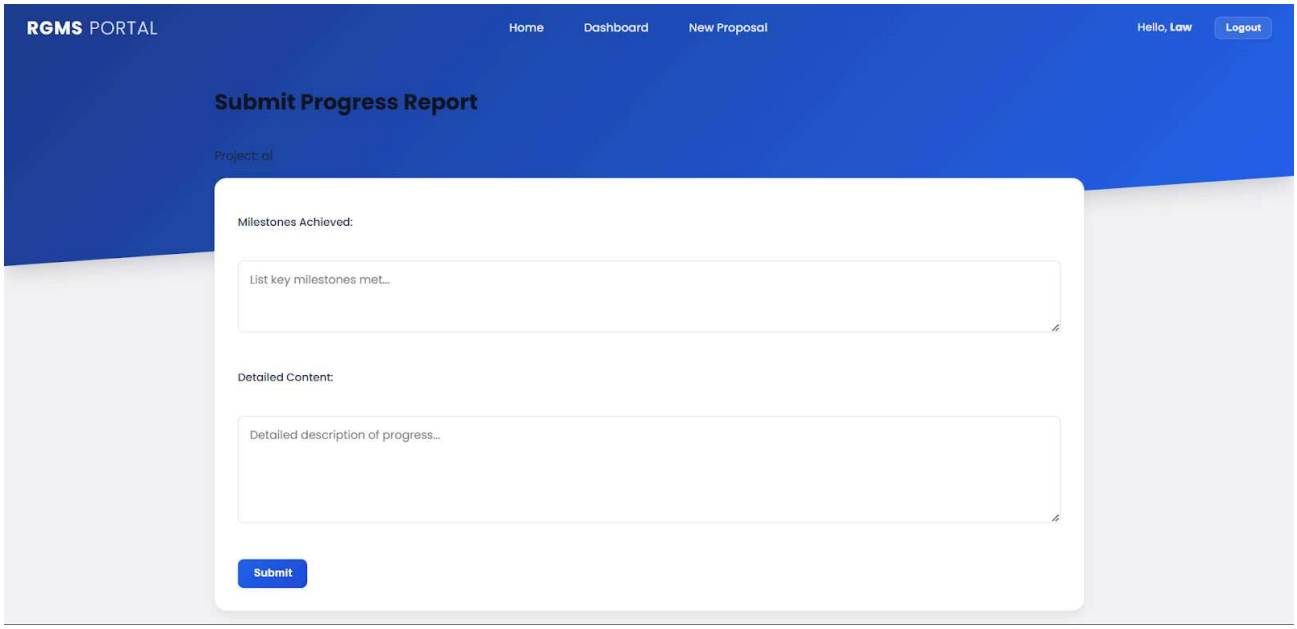
- **Budget Tracker:** Integrates native HTML5/CSS progress bars (.progress-fill) that dynamically display budget usage. Bars change color (Green, Yellow, Red) based on expenditure thresholds.
- **Expenditure Log:** A tabular view detailing date, item description, and receipt proofs for individual expenses.



7.2.4 Submit Progress Report

Submit Progress Report Form An interface tailored for qualitative data collection, allowing the researcher to update the department on their research milestones.

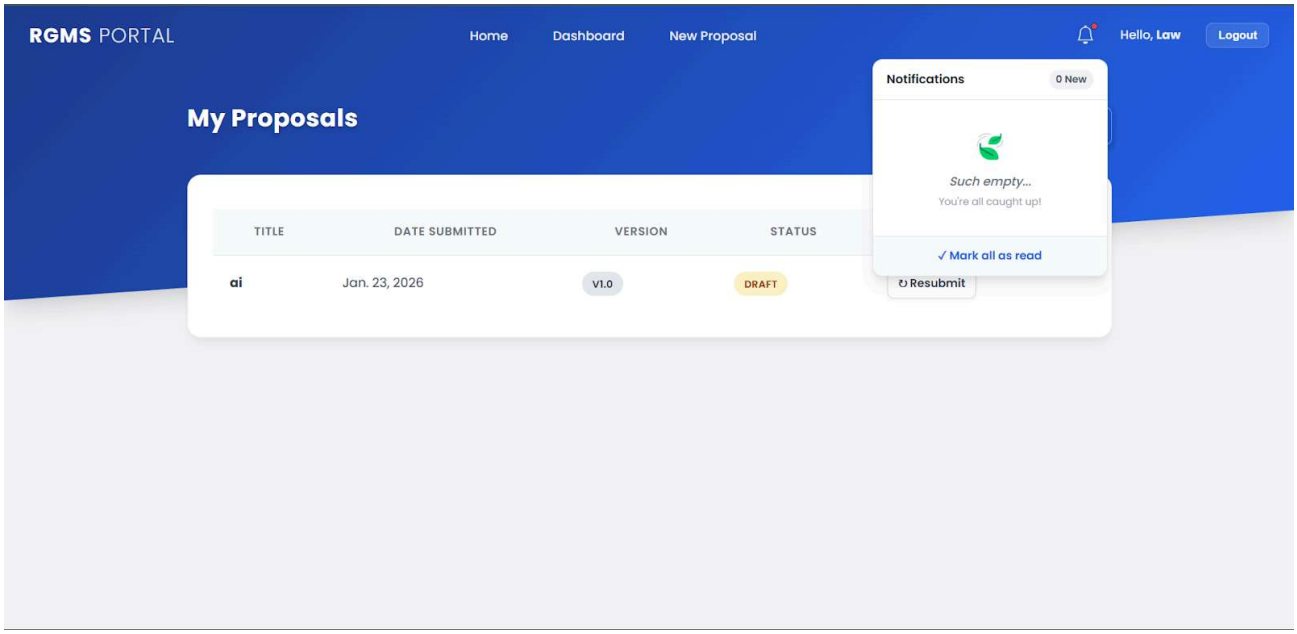
- **Narrative Input:** Features large, resizable text areas (.custom-textarea) for detailed milestone descriptions.
- **Context Preservation:** Displays the Grant Title and current timeline at the top of the card so the researcher retains project context while writing.



7.2.5 Notification System

Notification System Integrated directly into the navigation bar, the Notification Bell acts as the central hub for grant updates (e.g., "Proposal Approved", "Report Reviewed").

- **Visual Cue:** A custom red indicator dot (notif-badge) alerts the researcher to unread system messages.
- **Interactive Dropdown:** Clicking the bell reveals a floating, card-styled menu powered by pure CSS animations.
- **Empty State:** When no notifications are present, the system displays an illustrated "Such empty... " state, reducing visual clutter and confirming to the user that they are caught up.



7.3 Subsystem 2 Screens: Reviewer Subsystem

7.3.1 Interface Description: Reviewer Dashboard

Screen Name: Reviewer Dashboard

User Role: Reviewer

Purpose: To provide a streamlined list of all research proposals assigned to the reviewer for evaluation.

Key Interface Components:

1. **Hero Header:** A blue-themed header featuring the "RGMS Portal" branding and a personalized welcome message (e.g., "Welcome, Ijf") to confirm the user session.
2. **Proposal Evaluation Table:**

Data Columns: Displays essential metadata including Proposal ID, Title, and Researcher Name.

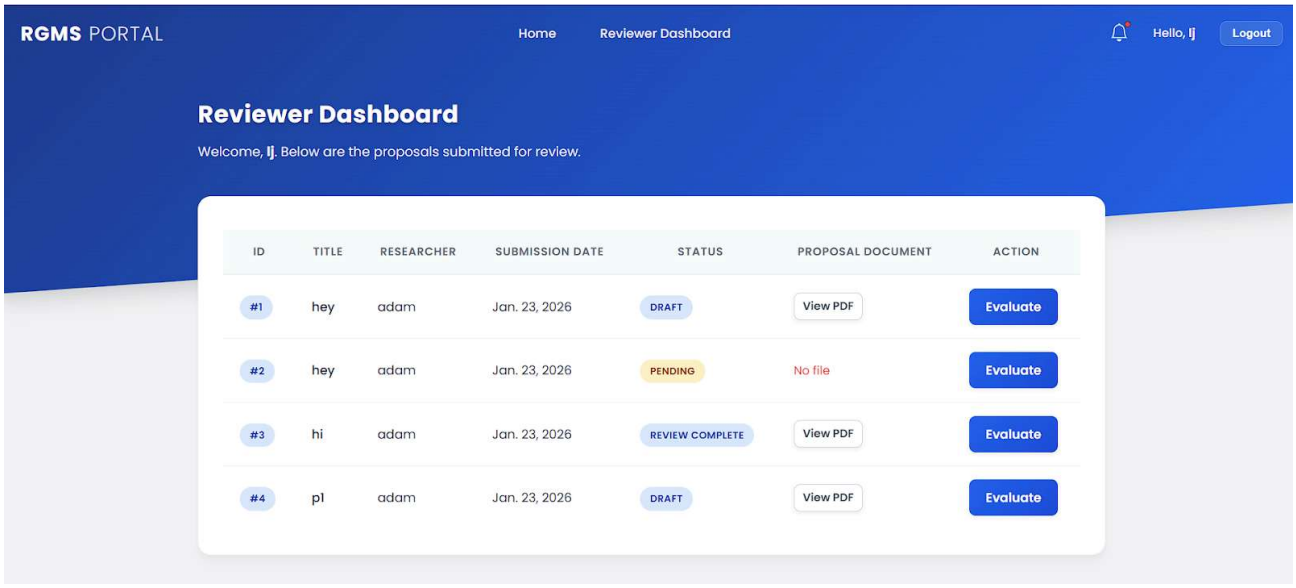
Submission Tracking: Shows the Submission Date (e.g., Jan 23, 2026) and current Status (e.g., "Draft") to help reviewers prioritize their workload.

3. **Action Hub:**

   View PDF: A primary blue button that allows the reviewer to open and read the proposal document without leaving the application environment.

   Evaluate: A secondary action button that triggers the transition to the formal assessment form

4. **Empty State Logic:** When no tasks are assigned, the table displays a "No proposals available for review" message, preventing UI clutter.



7.3.2 Interface Description: Proposal Evaluation Form

Screen Name: Evaluate Proposal Screen

User Role: Reviewer

Purpose: To capture the reviewer's quantitative score and qualitative feedback for a specific research submission.

Key Interface Components:

1. **Contextual Header:** The screen title dynamically updates to include the proposal title (e.g., "Evaluate Proposal: hey"), ensuring the reviewer knows exactly which project they are scoring.

2. **Researcher Attribution:** Clearly displays the "Submitted by" field to maintain transparency during the review process.
3. **Assessment Inputs:**

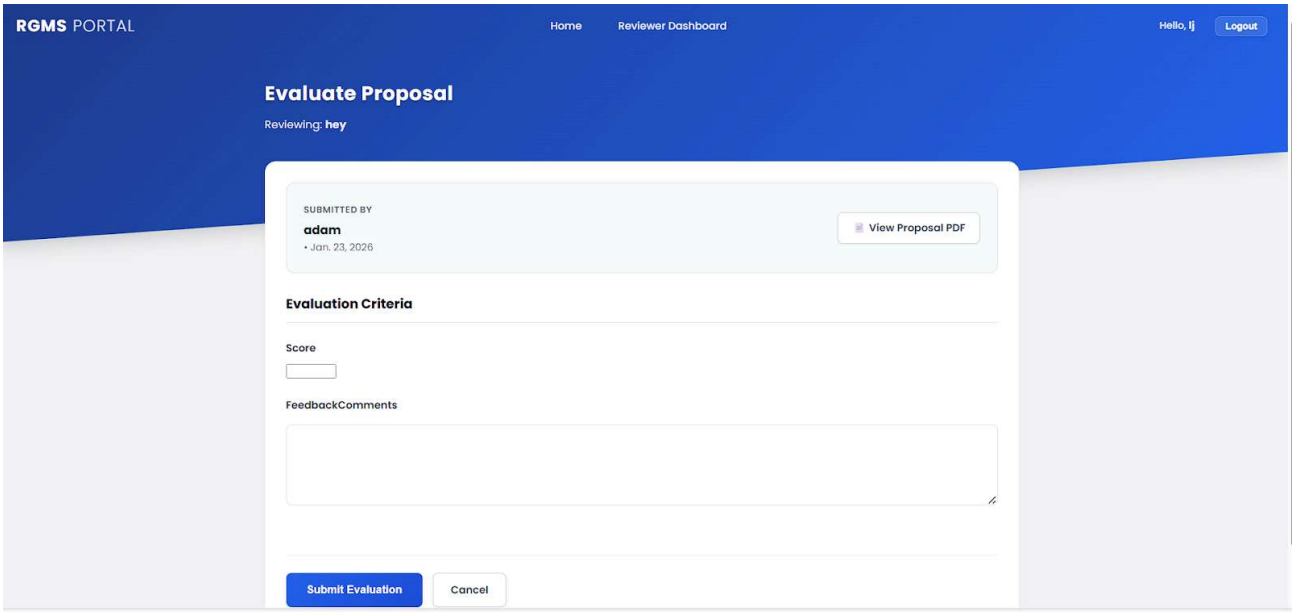
   Score Field: A dedicated numeric input box for the reviewer to provide a quantitative rating based on institutional rubrics.

   Feedback/Comments Box: A large, expandable textarea labeled "FeedbackComments" designed for detailed peer-review notes, critiques, and suggestions for the researcher.

4. **Form Submission Controls:**

   Submit Evaluation: A high-visibility blue button to finalize the review and commit the data to the system.

   Cancel: A neutral button that allows the user to exit the form and return to the dashboard without saving changes, serving as a safety mechanism against accidental edits.



7.4 Subsystem 3 Screens 9 (HOD)

7.4.1 Interface Description: HOD Management Dashboard

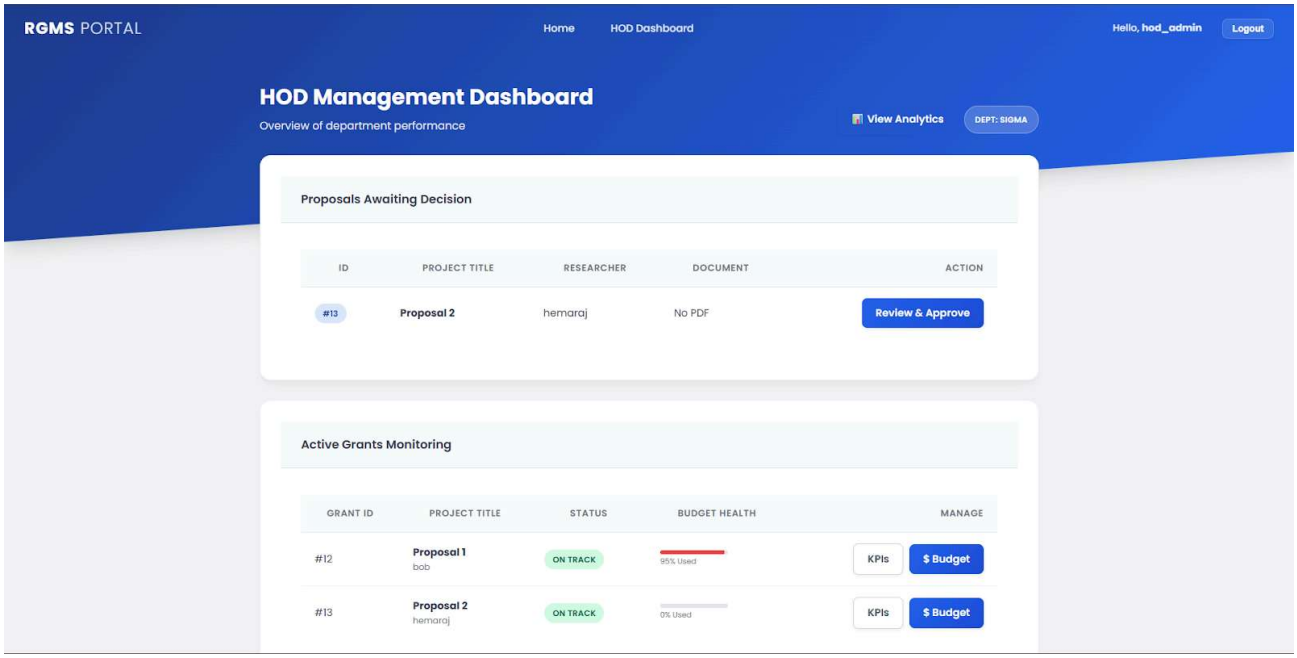
Screen Name: HOD Dashboard

User Role: Head of Department (HOD)

Purpose: To provide a centralized overview of departmental research activities, allowing the HOD to manage incoming proposals and monitor the financial and operational health of active grants.

Key Interface Components:

1. **Hero Header & Global Controls:**
    - **Department Identity:** The top section clearly identifies the user's jurisdiction with a specific badge (e.g., "DEPT: SIGMA"), ensuring the HOD knows which department's data is being viewed.
    - **Analytics Access:** A prominent **"View Analytics"** button is positioned in the header, providing single-click access to the graphical reports (Budget Utilization Doughnut Chart & Success Rate Bar Chart).
    - **Navigation:** Consistent top-bar navigation allows quick switching between Home and the Dashboard, with a user profile and logout option.
2. **Pending Proposals Section (Action Queue):**
    - This card displays a tabular list of **"Proposals Awaiting Decision".**
    - **Data Columns:** Displays critical info including Proposal ID , Project Title , Researcher Name , and Document availability.
    - **Primary Interaction:** The **"Review & Approve"** button directs the HOD to the detailed approval form for that specific proposal.
3. **Active Grants Monitoring (Project Oversight):**
    - A comprehensive table tracking **"Active Grants Monitoring"** for projects that have already been approved.
    - **Status Indicators:** Uses color-coded badges (e.g., Green for "ON TRACK") to give an immediate visual status of project health.
    - **Visual Budget Health:** Features a dynamic **Progress Bar** under the "Budget Health" column.
        - **Logic:** The bar visualizes the percentage of funds used. It automatically changes color to **Red** (Critical) when usage exceeds 90% (as seen in the "95% Used" example), alerting the HOD to potential overspending without needing to click into the details.
    - **Management Actions:** Provides two distinct action buttons for every grant:
        - **KPIs:** To review project milestones and timelines.
        - **$ Budget:** To view the financial ledger and perform fund top-ups



7.4.2 Interface Description: Approve and Grant Allocation

Screen Name: Grant Approval Decision

User Role: Head of Department (HOD)

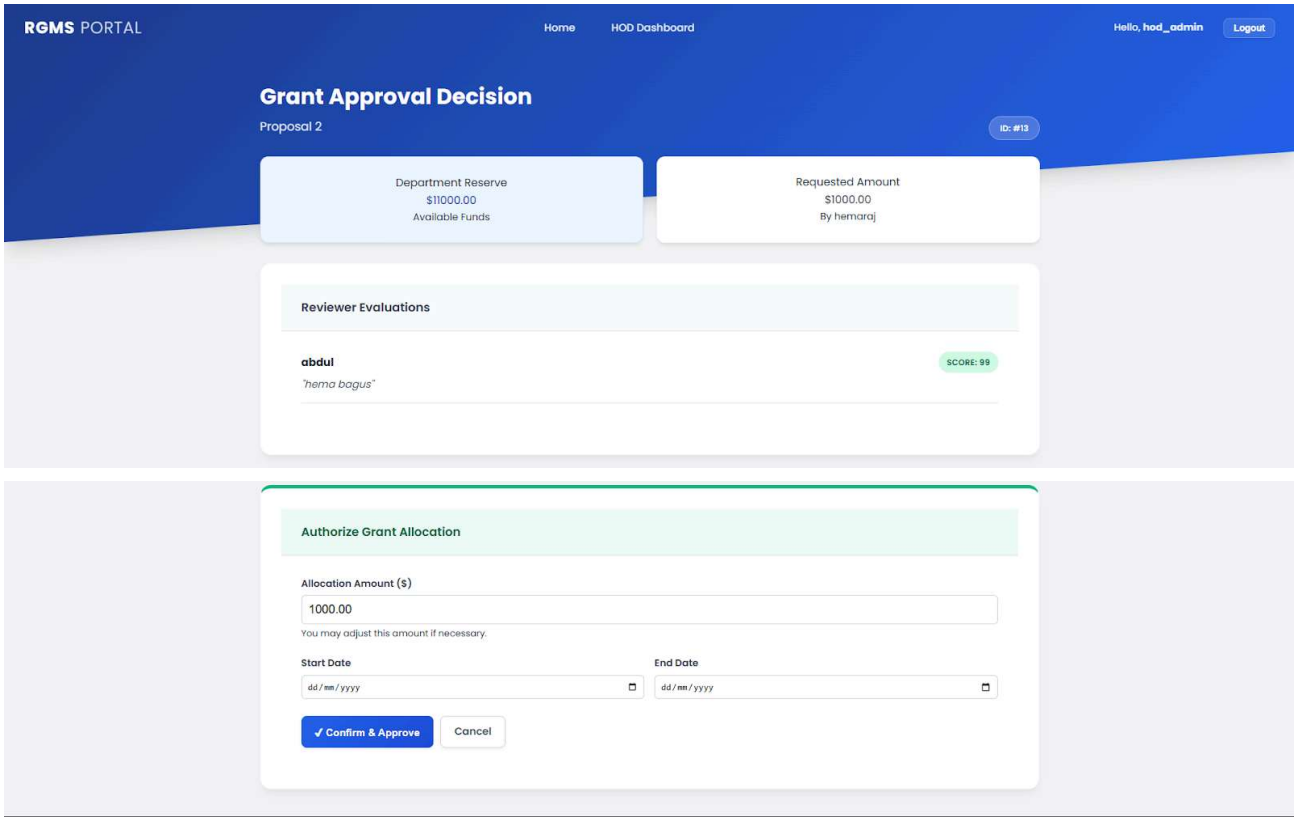
Purpose: To enable the HOD to review reviewer evaluations, assess budget feasibility, and formally authorize the allocation of funds for a specific research proposal.

Key Interface Components:

1. **Financial Context Header (Top Grid):**
    - **Purpose:** To provide immediate financial awareness before a decision is made.
    - **Component:** Two prominent "KPI Cards" displayed side-by-side.
        - **Department Reserve:** Displays the total available funds (e.g., "$11,000.00") in the department's account.
        - **Requested Amount:** Displays the specific amount requested by the researcher (e.g., "$1,000.00").
    - **Design Logic:** Placing these figures adjacent to each other allows the HOD to instantly verify if the department can afford the grant without navigating to a separate budget page.
2. **Evaluation Summary (Middle Section):**
    - **Section Title:** "Reviewer Evaluations".
    - **Content:** Aggregates qualitative and quantitative feedback from the review phase.

- **Data Points:** Displays the reviewer's username (e.g., "abdul"), their specific comments (e.g., "hema bagus"), and the assigned **Score Badge** (e.g., "SCORE: 99").
- **Visual Cue:** High scores are highlighted with green badges to quickly signal quality proposals.

3. **Authorization Form (Bottom/Action Section):**
   - **Section Title:** "Authorize Grant Allocation".
   - **Visual Distinction:** This card features a distinct **Green Top Border** to visually signify that this is the "Action Zone" for final approval.
   - **Editable Allocation:** The "Allocation Amount ($)" field is pre-filled with the requested amount ($1000.00) but remains editable, giving the HOD the flexibility to approve partial funding if necessary.
   - **Timeline Controls:** Mandatory "Start Date" and "End Date" pickers to define the grant's active duration.
   - **Primary Action:** A **"Confirm & Approve"** button that commits the transaction and creates the grant record.



7.4.3 Interface Description: Project KPI & Monitoring Review

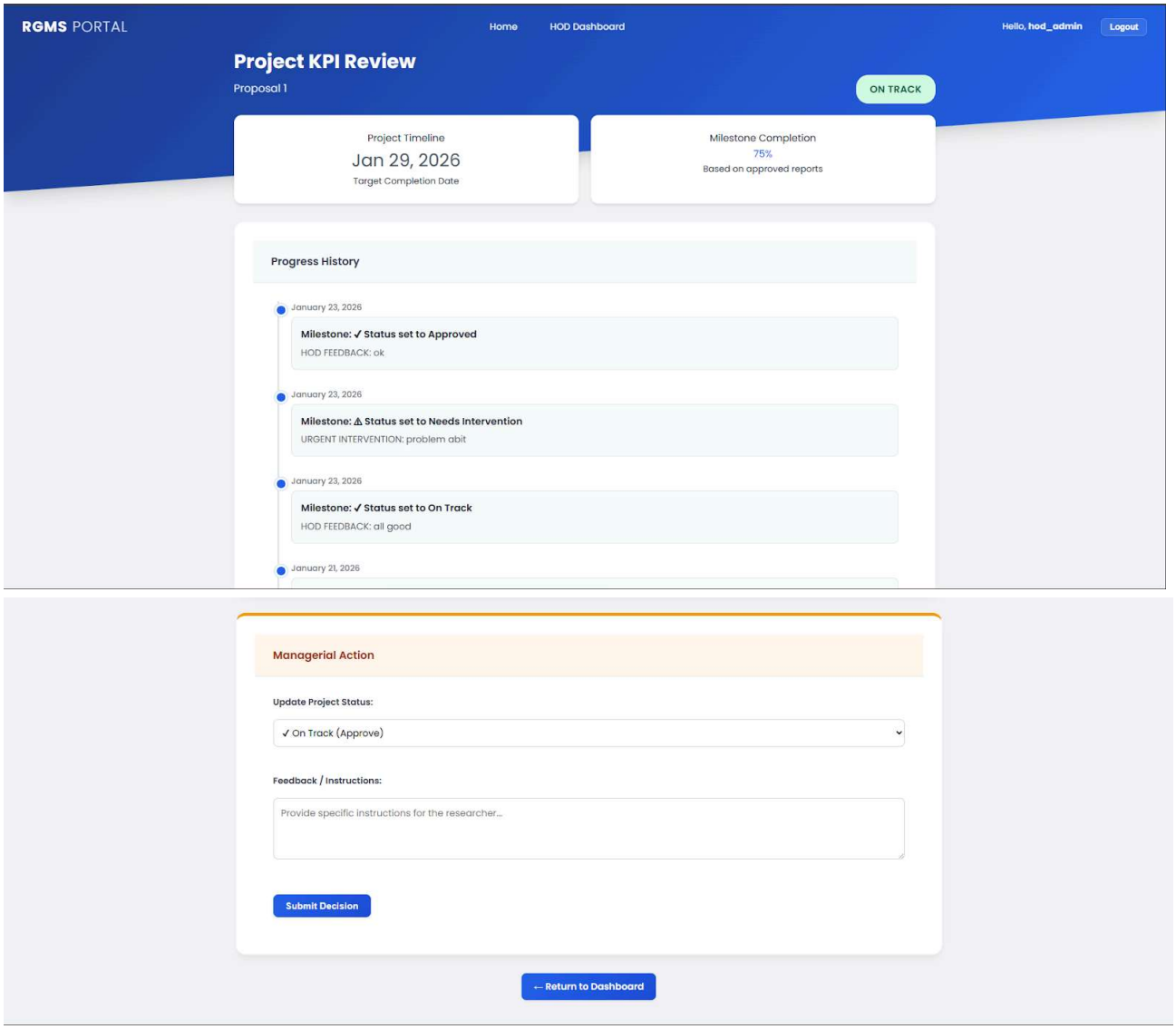
Screen Name: Project Monitoring Detail (project_monitoring_detail.html)

User Role: Head of Department (HOD)

Purpose: To provide a detailed historical and real-time view of a specific grant's progress, allowing the HOD to review milestone achievements and issue managerial directives.

Key Interface Components:

1. **Project Status Header:**
    - **Project Identity:** Clearly displays the project title (e.g., "Proposal 1") and the current status badge (e.g., "ON TRACK").
    - **Visual Context:** Uses the standard HOD blue hero header for consistency, ensuring the user knows they are in a management view.
2. **Performance Metrics (KPI Grid):**
    - **Timeline Tracking:** Displays the "Project Timeline" with the target completion date (e.g., "Jan 29, 2026"), helping the HOD assess if the researcher is behind schedule.
    - **Completion Metric:** A large numeric indicator for "Milestone Completion" (e.g., "75%"). This provides an instant quantitative measure of progress based on approved reports.
3. **Progress History (Vertical Timeline):**
    - **Component:** A chronological, vertical timeline on the left side of the screen.
    - **Data Points:** Each entry acts as a historical log, displaying:
        - **Date:** When the report/action occurred (e.g., "January 23, 2026").
        - **Event:** The specific milestone or status change (e.g., "Milestone: Status set to Needs Intervention").
        - **Feedback/Context:** Displays the content of the report or the HOD's previous feedback (e.g., "URGENT INTERVENTION: problem abit").
    - **Design Logic:** This timeline format allows the HOD to trace the "story" of the project—seeing how it went from "On Track" to "Needs Intervention" and back again.
4. **Managerial Action Panel (Intervention Form):**
    - **Purpose:** To give the HOD direct control over the project's lifecycle.
    - **Visual Distinction:** This card uses an **Orange/Amber Header** ("Managerial Action") to differentiate it from the informational cards, signaling that this is a control panel.
    - **Status Control:** A dropdown menu allows the HOD to force a status update (e.g., "✓ On Track (Approve)" or "⚠ Needs Intervention").
    - **Feedback Loop:** A text area for "Feedback/Instructions" enables the HOD to send specific directives directly to the researcher.
    - **Primary Action:** The "Submit Decision" button commits these changes to the system.



7.4.4 Interface Description: Project KPI & Monitoring Review

Screen Name: Budget Detail View

User Role: Head of Department (HOD)

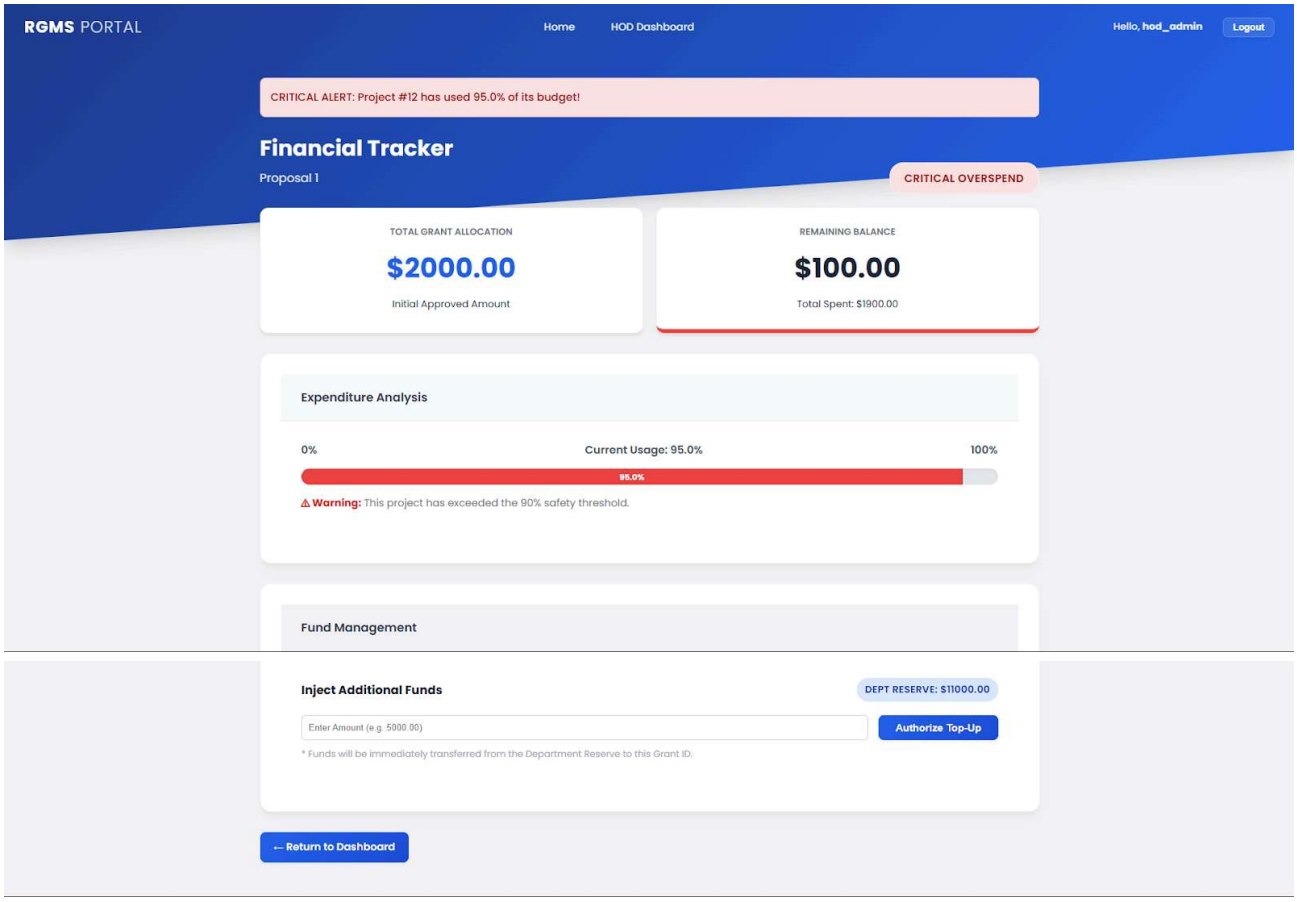
Purpose: To provide granular oversight of a specific grant's financial health, alert the HOD to critical overspending, and enable emergency funding interventions (top-ups).

Key Interface Components:

1. **Critical Alert System:**
    - **Banner Alert:** If a project exceeds the 90% spending threshold, a prominent **Red Alert Banner** appears at the top ("CRITICAL ALERT: Project #12 has used 95.0% of its budget!").

- **Visual Status:** A "CRITICAL OVERSPEND" badge in the header instantly signals that this project is in a danger zone.

2. **Financial KPI Cards (Grid Layout):**
   - **Total Grant Allocation:** Displays the original approved amount (e.g., "$2000.00").
   - **Remaining Balance:** Displays the live funds left (e.g., "$100.00").
   - **Visual Logic:** The "Remaining Balance" card dynamically applies a **Red Bottom Border** (class: border-critical) when funds are low, reinforcing the urgency visually.
3. **Expenditure Analysis (Progress Tracking):**
   - **Component:** A visual progress bar tracking "Current Usage" against a 0% - 100% scale.
   - **Logic:** The bar turns **Solid Red** when usage exceeds 90% (e.g., "95.0%").
   - **Diagnostic Text:** A warning message below the bar explicitly states the rule violation: "⚠ Warning: This project has exceeded the 90% safety threshold".
4. **Fund Management Panel (Action Zone):**
   - **Context:** Displays the "Dept Reserve" balance (e.g., "$11,000.00") so the HOD knows how much money is available to give.
   - **Input Control:** A precise numeric input field ("Enter Amount") allows the HOD to specify the exact top-up value.
   - **Primary Action:** The **"Authorize Top-Up"** button executes the transfer immediately.
   - **Disclaimer:** Clear instructional text confirms that funds will be deducted from the Department Reserve.



7.4.5 Interface Description: Department Analytics & Performance Reports

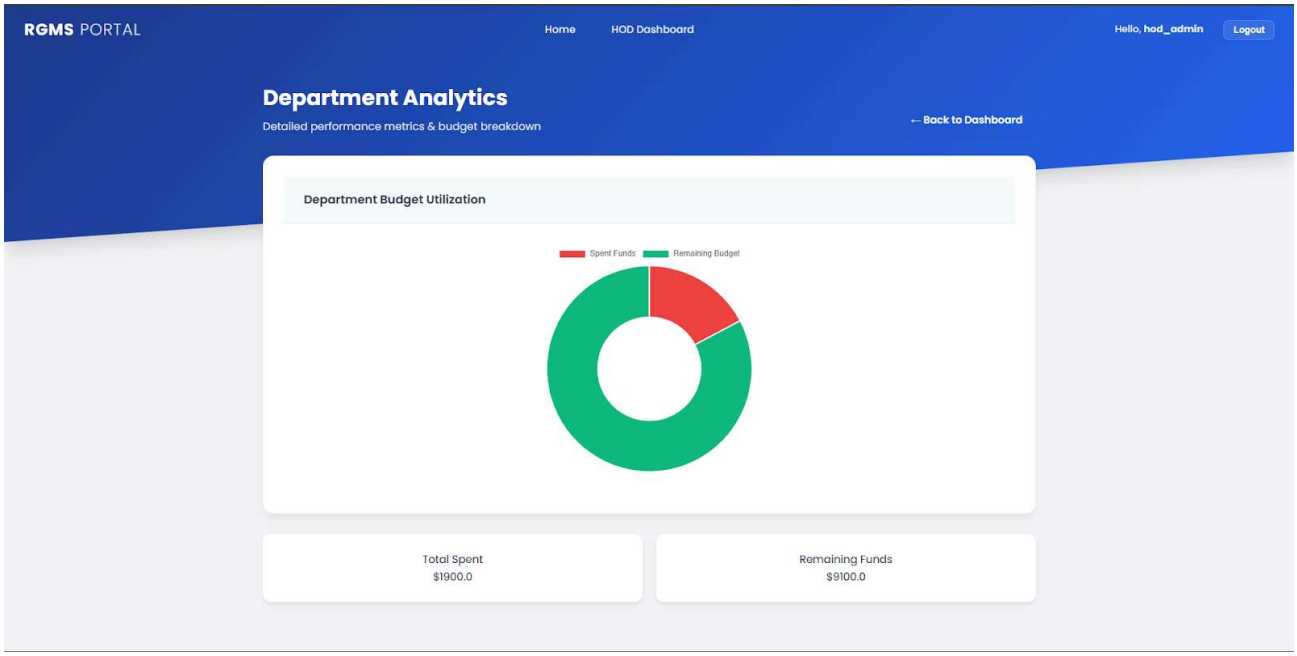
Screen Name: Department Analytics View

User Role: Head of Department (HOD)

Purpose: To provide high-level visual intelligence regarding the department's financial efficiency and proposal success rates, allowing for data-driven strategic planning.

Key Interface Components:

1. **Navigational Header:**
    - **Context:** The header clearly labels the page "Department Analytics" with a subtitle "Detailed performance metrics & budget breakdown," setting the expectation for statistical content.
    - **Back Navigation:** A prominent "← Back to Dashboard" button allows the HOD to quickly return to the main operational view after reviewing the data.
2. **Visual Analytics (Charts Section):**
    - **Budget Utilization Chart:** A Doughnut Chart visualizing the ratio of **"Spent Funds" (Red)** vs. **"Remaining Budget" (Green)**.
        - **Design Rationale:** The use of red/green color coding provides an immediate "traffic light" assessment of financial health. A large red segment warns of budget depletion at a glance.
    - **Proposal Success Metrics:** (Implicit in layout/code, though partially cut off in this specific crop) A Bar Chart comparing approved versus rejected proposals to track department research output quality.
3. **Summary KPI Cards (Bottom Grid):**
    - **Total Spent:** A dedicated card displaying the aggregate expenditure across all active grants (e.g., "$1900.0"). The text is color-coded **Red** to signify money flowing out.
    - **Remaining Funds:** A card displaying the available balance (e.g., "$9100.0"). The text is color-coded **Green** to signify available resources.



<*TO DO: Describe and the screens of subsystem 2 here.*>

Reflection and Learning Outcomes

< *Provide a reflective analysis of your learning and challenges faced.*>

Team member	Reflection and Learning Outcomes

User Guide

<*This section should the steps to be followed to use the system. Include step-by-step instructions for using your module.*>

Conclusion

<*This section should include details of completion of software, software quality assurance, group collaboration, problems encountered. Summarize your individual contribution and future recommendations.*>

References

<*List any references or sources used.*>