Software Requirements Specification

for

Research Grant Management System

Version <1.0>

Group No.: 3

Suraj A/L Prakash	**243UC247CL**
Lim Jian Feng	**243UC247BZ**
Law Jun Feng	**243UC247CP**

Date: 14/12/2025

CONTENTS

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
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 Introduction

1.1 Team Members

Name	Actor/Processes
Suraj A/L Prakash	HOD
Lim Jian Feng	Reviewer
Law Jun Feng	Researcher

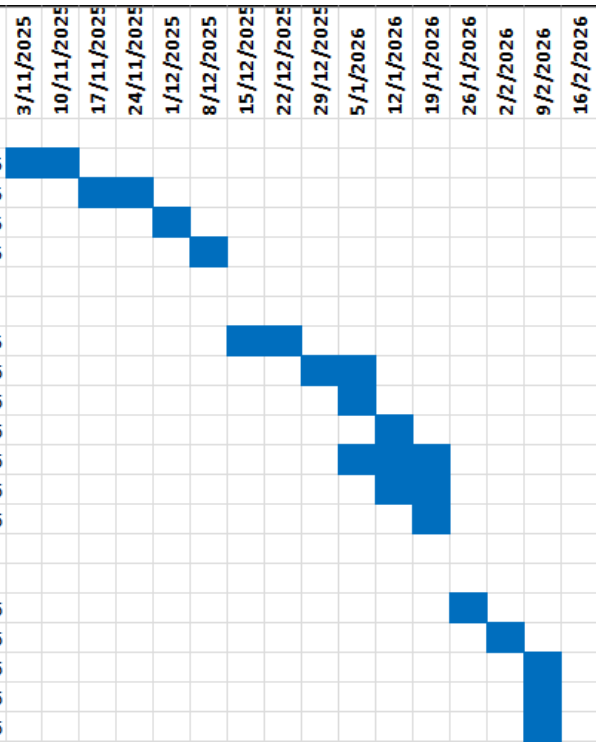
1.2 Problem statement

Currently, the University's research grant process relies on decentralized and manual workflows, primarily utilizing email correspondence and static spreadsheets. This approach presents four significant challenges:

1. **Workflow Inefficiency:** The manual routing of proposals from Researchers to Reviewers leads to delays, lost documents, and missed deadlines due to a lack of automated notifications.
2. **Lack of Financial Integrity:** Decentralized budget tracking results in data inconsistency, making it difficult for the Head of Department to monitor grant utilization accurately.
3. **Version Control Conflicts:** Managing proposal iterations via email frequently results in reviewers evaluating obsolete versions of documents, compromising the integrity of the review process.
4. **Limited Administrative Oversight:** Without a centralized dashboard, stakeholders lack real-time visibility into funding status and overall research output (KPIs), hindering effective decision-making.

1.3 Project Schedule

Task	Start	End
Phase 1: Project Planning		
Group Formation	3/11/2025	16/11/2025
Project Introduction & System Overview	17/11/2025	30/11/2025
Functional Requirements & System Models	1/12/2025	7/12/2025
Refining & Submission of Part 1	8/12/2025	14/12/2025
Phase 2: System Design (Part 2)		
Architectural & Data Design	15/12/2025	28/12/2025
Behavioral Modeling & Component Design	29/12/2025	11/1/2026
Database Setup & User Login	5/1/2026	11/1/2026
Interface UI & Deployment Design	12/1/2026	18/1/2026
Early Implementation (Prototyping)	5/1/2026	25/1/2026
Core Features	12/1/2026	25/1/2026
Refining & Submission of Part 2	19/1/2026	25/1/2026
Phase 3: Final Implementation (Part 3)		
Final Coding & Integration	25/1/2026	2/2/2026
Testing & Debugging	2/2/2026	8/2/2026
Presentation & Demo Prep	3/2/2026	6/2/2026
Final Documentation Compilation	4/2/2026	7/2/2026
Final Submission of Part III	8/2/2026	8/2/2026



2 System Overview

2.1 Description

The RGMS acts as a centralized hub for all grant-related activities, ensuring governance, auditability, and efficiency. Its primary goal is to solve key problems like workflow inefficiency, lack of financial integrity, version control conflicts, and limited administrative oversight.

The major functions the system will perform for its users are:

- **User Registration and Authentication:** Managing secure access for all stakeholders (Researcher, Reviewer, Head of Department).
- **Proposal Submission and Review:** Providing a structured, online portal for researchers to submit proposals and for reviewers to assess and score submissions.
- **Grant Allocation & Budget Tracking:** Centralizing the approval process, allocating funds, and enabling the Head of Department to accurately monitor grant utilization and expenditures.
- **Document Version Control:** Ensuring all stakeholders are working with the most current version of proposals and documents to maintain the integrity of the review process.
- **Progress Report Submission:** Allowing researchers to submit project updates and for administrators to track progress and compliance requirements.
- **Notification for Deadlines and Approvals:** Automating alerts to prevent delays and missed deadlines that result from manual routing.
- **Dashboard (Funding Status, Research Output):** Providing real-time, centralized visibility into key performance indicators (KPIs), funding status, and overall research output for effective decision-making.

Actors and Key Processes

Researcher: Submission of Proposals, Progress Reports, Document Management

Reviewer: Evaluation and Scoring of Proposals, Document Review

Head of Department: Grant Approval/Rejection, Budget Monitoring and Tracking

2.2 Actors

Actor	Use Cases
Researcher	● Upload research proposals and reports ● Document Version Control ● Track grant status & budget usage(track own budget) ● Receive notifications (deadlines, approvals)
Reviewer	● Review submitted proposals ● View researcher documents ● Give evaluation / feedback(visible for both HOD & Researcher) ● Receive notifications for pending reviews(from researcher)
Head of Department(HOD)	● Approve / reject grant proposals ● Monitor research project progress ● Grant Allocation & Budget Tracking ● Monitor Research Dashboard (KPIs)

2.3 Assumptions and Dependencies

2.3 Assumptions and Dependencies

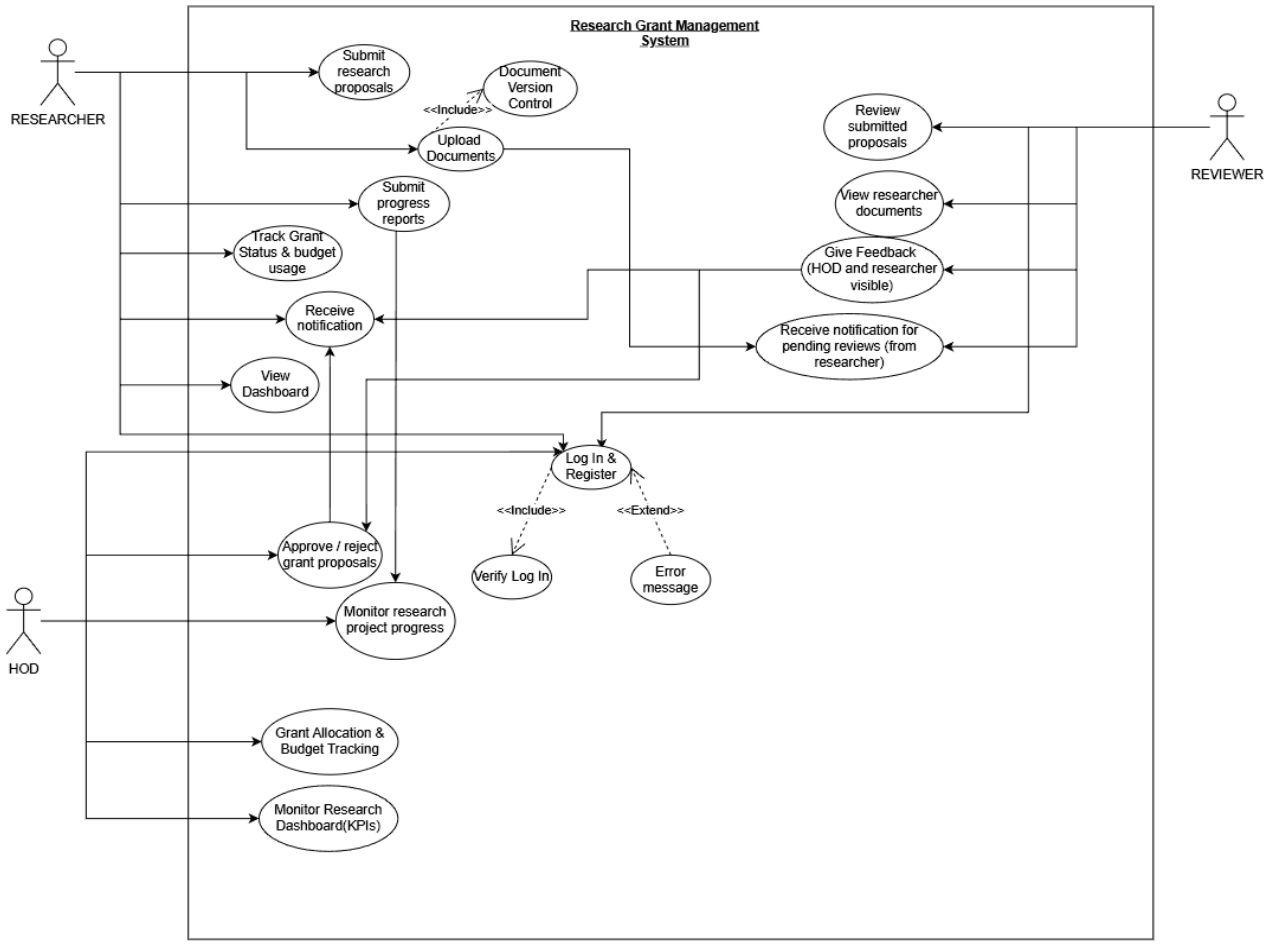
Assumptions

- **Technical Proficiency:** It is assumed that the development team is proficient in the Python programming language and the Django framework.
- **Internet Connectivity:** The system assumes that all users (Researchers, Reviewers) will have a stable internet connection to access the web-based interface.
- **Browser Compatibility:** It is assumed that users will access the system using modern web browsers (Chrome, Edge, Firefox, Safari) that support HTML5 and JavaScript.

Dependencies

- **Django Framework:** The system implementation depends entirely on the Django Web Framework (Python). Any updates or deprecations in the framework version may affect development.
- **Database Engine:** The system relies on a relational database compatible with Django ORM (e.g., SQLite for development, PostgreSQL/MySQL for production).
- **Hosting Environment:** The deployment of the system is dependent on a server environment that supports Python WSGI/ASGI applications.

2.4 Use Case Diagram

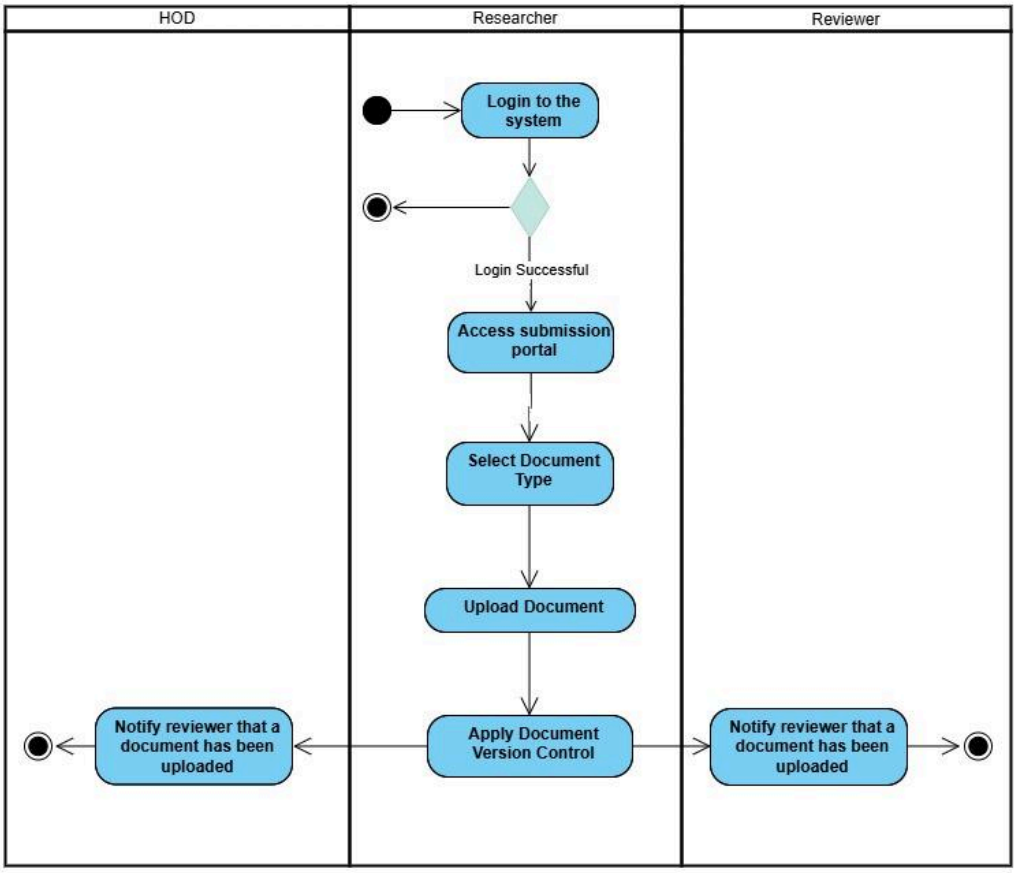


3 Functional Requirements

3.1 Actor 1: Researcher

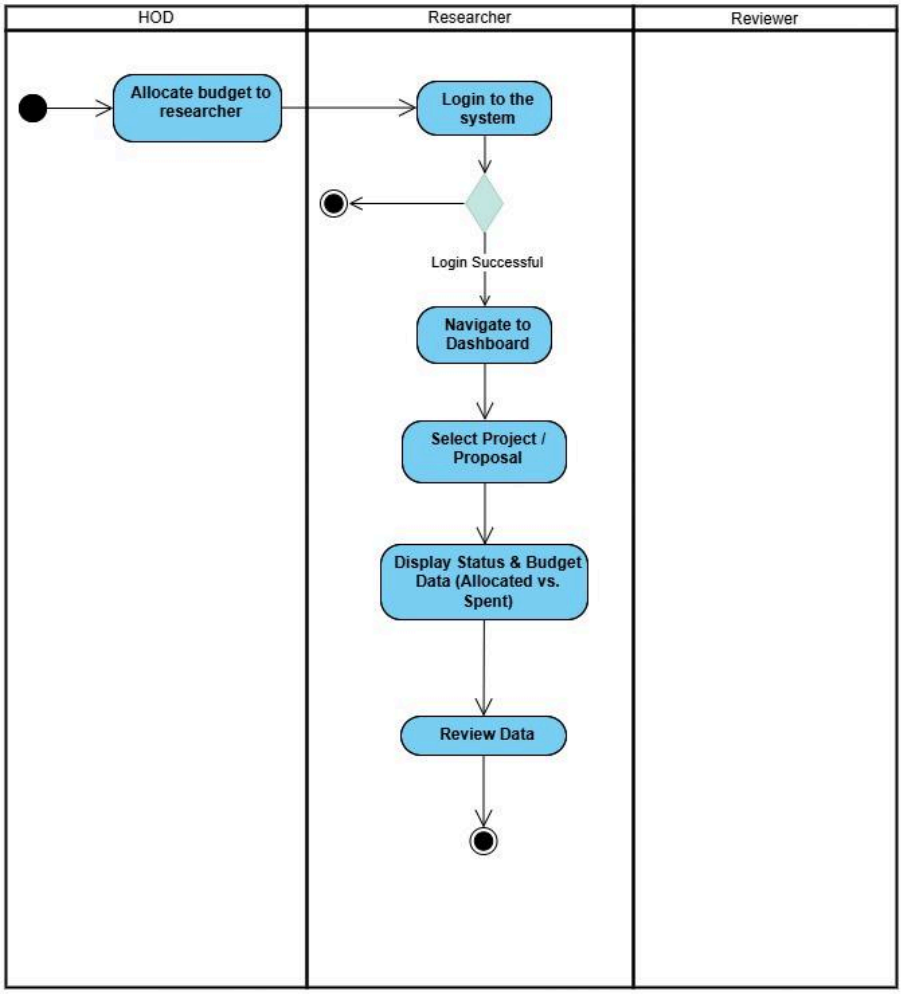
3.1.1 Use Case 1 : Upload Research Proposals and Reports

This critical use case covers the submission of all required documents to the RGMS. The Researcher uses an online portal to submit research proposals to initiate a grant request and, subsequently, to upload progress reports for active projects. Successful uploads trigger the system's Document Version Control and may trigger notifications to Reviewers.



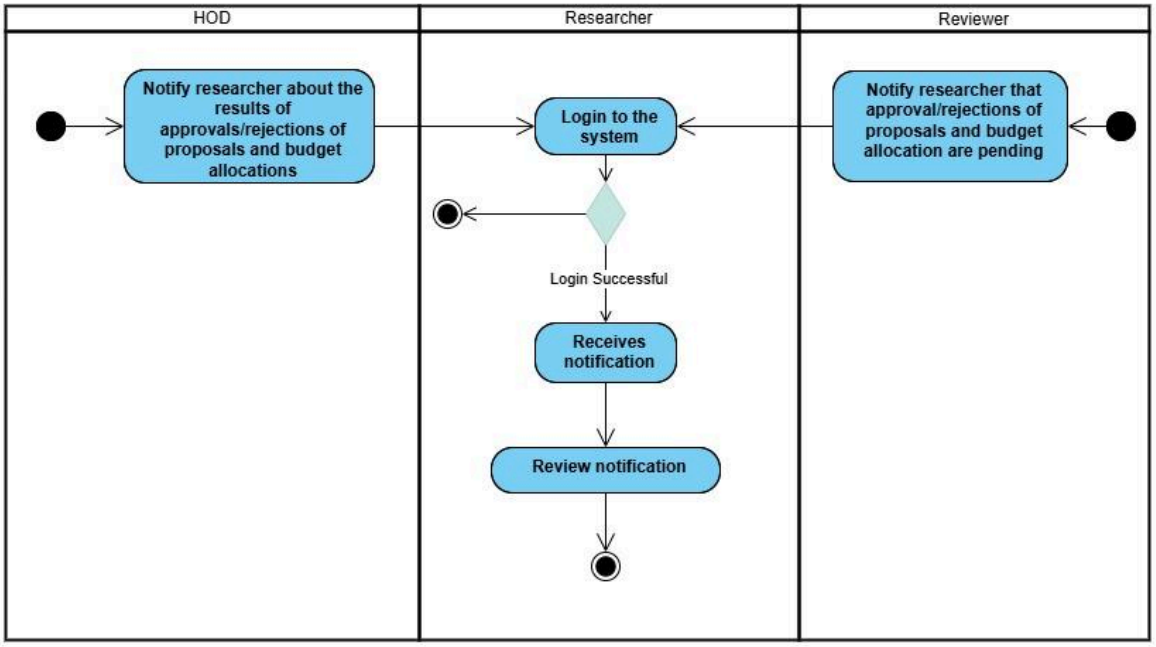
3.1.2 Use Case 2 : Track Grant Status & Budget Usage

This use case enables the Researcher to actively monitor the status of their submitted proposals throughout the review cycle and review the financial health of any active project. They access a centralized Dashboard to view funding status and project progress (KPIs), specifically tracking their own allocated budget and expenditure.



3.1.3 Use Case 3 : Receive Notifications (Deadlines, Approvals)

This use case details the system's ability to automate alerts to the Researcher. The Researcher receives notifications for critical workflow events, such as impending deadlines (e.g., for progress reports) and the outcome of proposal reviews (approvals or rejections). Notifications ensure the Researcher stays informed and prevents delays caused by manual correspondence.



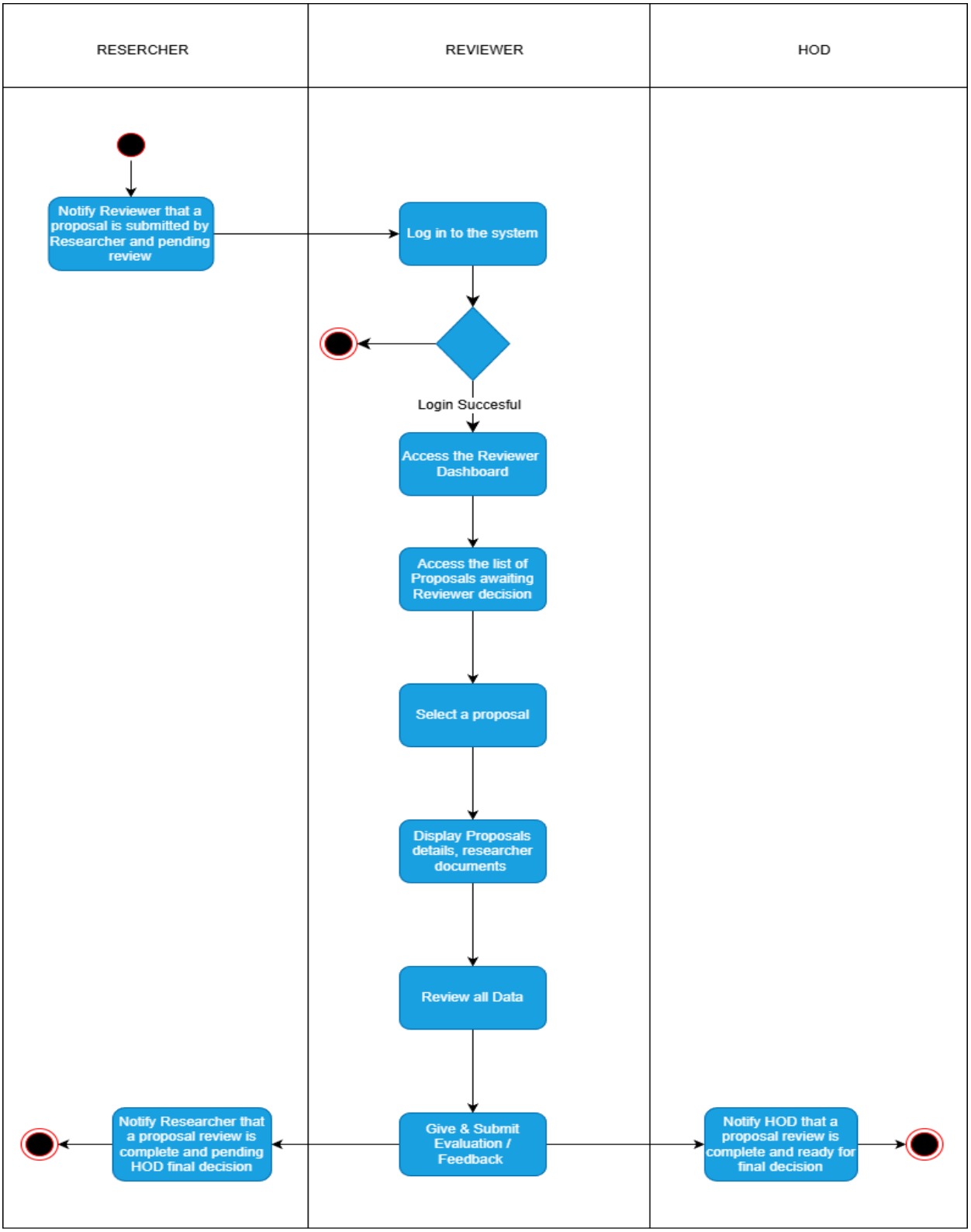
3.2 Actor 2 : REVIEWER

3.2.1 Review & Feedback Researcher documents / proposals

This use case defines the end-to-end evaluation process. The **Authorized Reviewer** begins by having the system display a list of their assigned proposals, along with key metadata like the **Proposal ID**, **title**, and **submission date**. The system retrieves and displays the **full research proposal document** and any associated **researcher documents** from storage for the Reviewer's assessment. After reviewing the material, the Reviewer enters the necessary **evaluation scores** (e.g., numerical ratings for methodology) and provides **detailed textual feedback/comments**. The Reviewer concludes their input by submitting a **final review.**Upon submission, the system stores the complete **Review Record**, including all scores, feedback in the database, and updates the **Proposal Status** to 'Review Complete'.

3.2.2 Receive notification for pending reviews(from researcher)

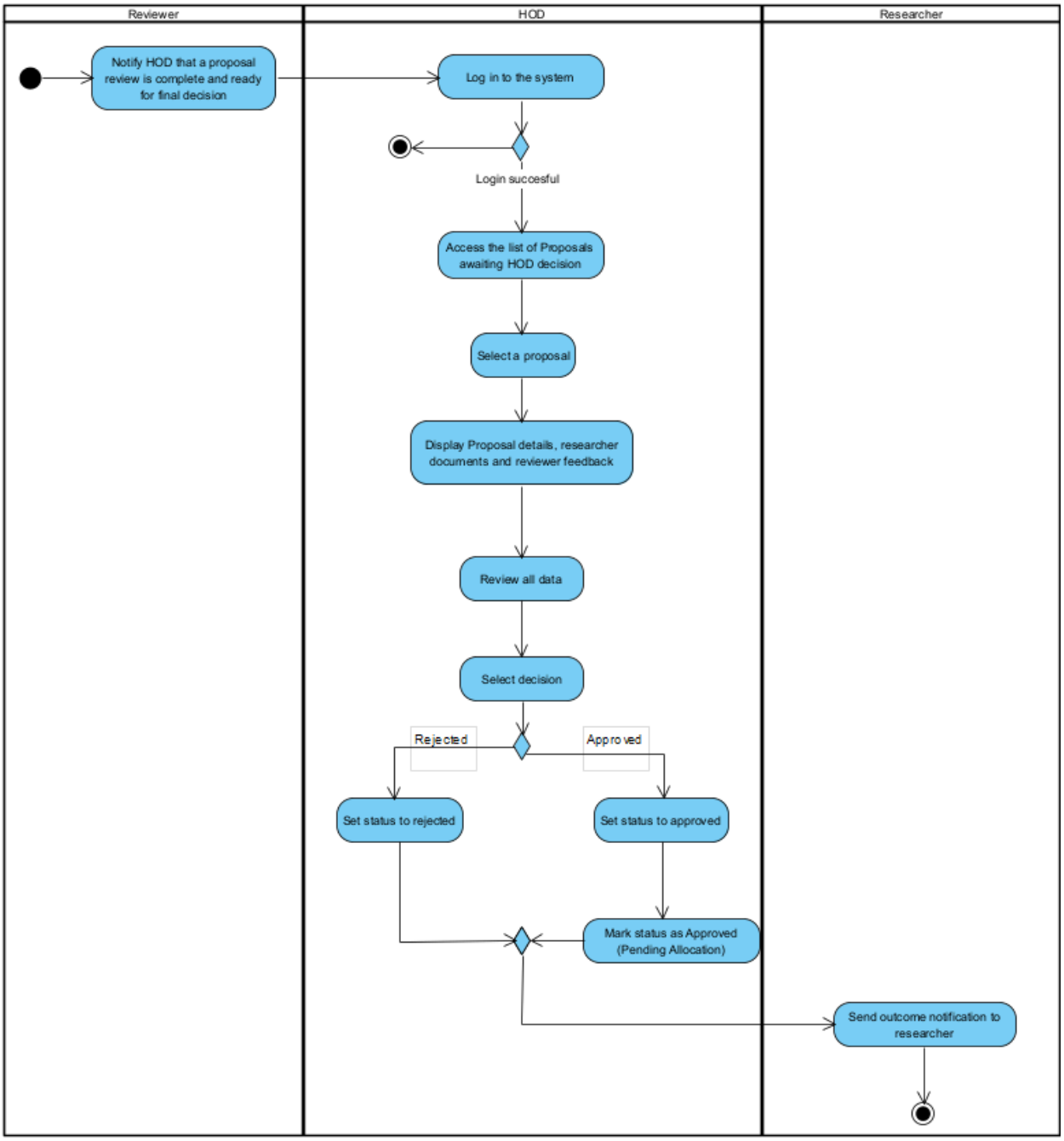
This use case ensures the Reviewer is promptly informed when a new proposal or revised version is submitted and awaiting evaluation. Triggered by the Researcher's submission, the system retrieves the **Reviewer's contact information** (e.g., email address) and the **Proposal's metadata** (ID, title, and calculated review deadline) from the database. The system then sends a **Notification** (via email or in-system alert) to the Reviewer, displaying the **Proposal Title**, **ID**, **review deadline**, and a **call-to-action link** to begin the review. The system logs a **Notification Record** in storage, noting the Reviewer ID, Proposal ID, and the **timestamp of the notification send**. To access the proposal, the Reviewer must interact with the notification by clicking the link and then successfully **logging into the system**.


RESERCHER
REVIEWER
HOD
Notify Reviewer that a proposal is submitted by Researcher and pending review
Log in to the system
Login Succesful
Access the Reviewer Dashboard
Access the list of Proposals awaiting Reviewer decision
Select a proposal
Display Proposals details, researcher documents
Review all Data
Give & Submit Evaluation / Feedback
Notify Researcher that a proposal review is complete and pending HOD final decision
Notify HOD that a proposal review is complete and ready for final decision

3.3 Actor 3: HOD

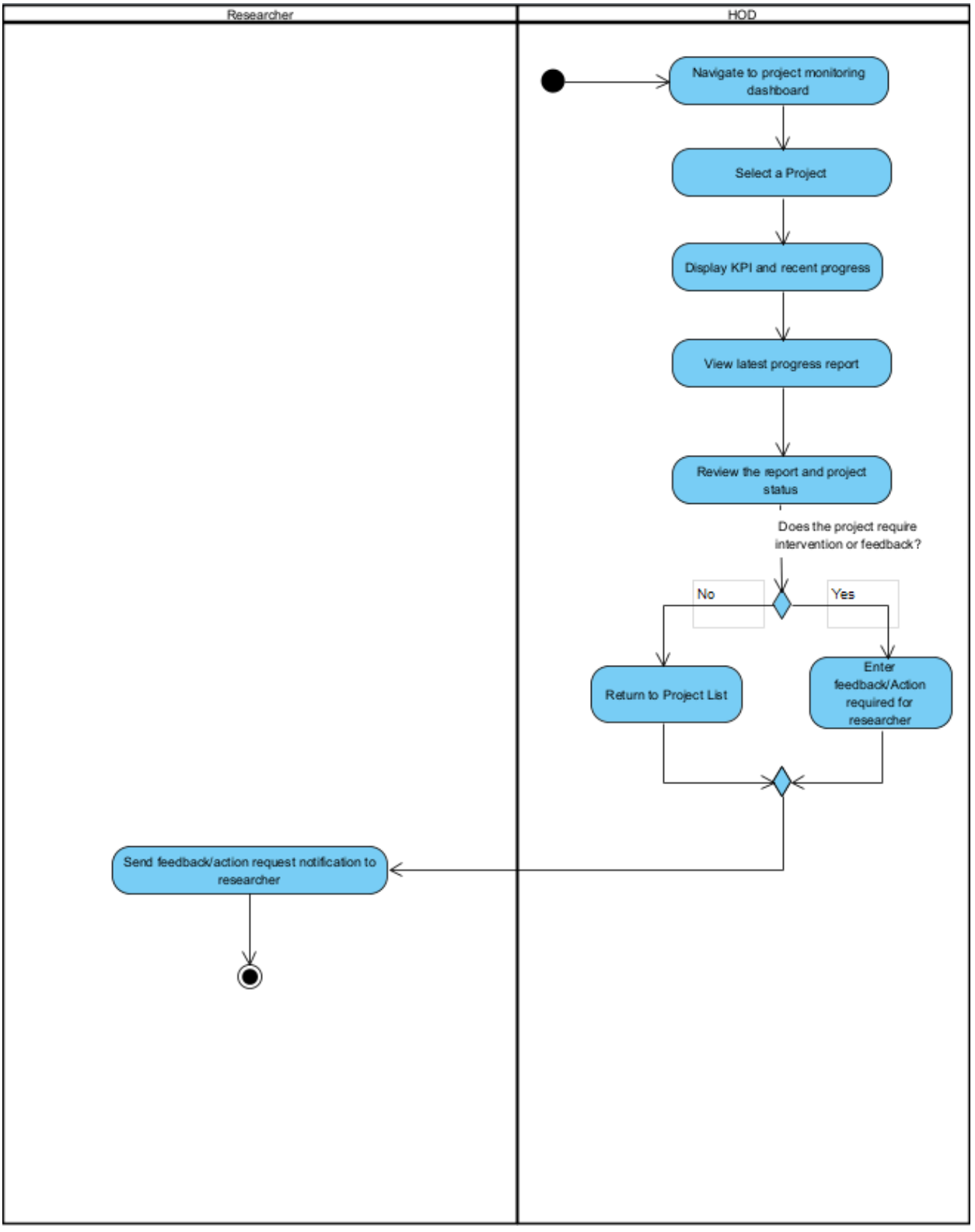
3.3.1 Approve/Reject Grant Proposals

The HOD initiates this process by retrieving a list of pending grant applications from the database. The system displays the applicant's profile, the full research proposal, and the requested budget details for the HOD to review. After evaluating the merit of the application, the HOD enters a final decision (Approve or Reject) along with specific review comments. Finally, the system stores the updated proposal status and the HOD's feedback into the database to conclude the transaction.



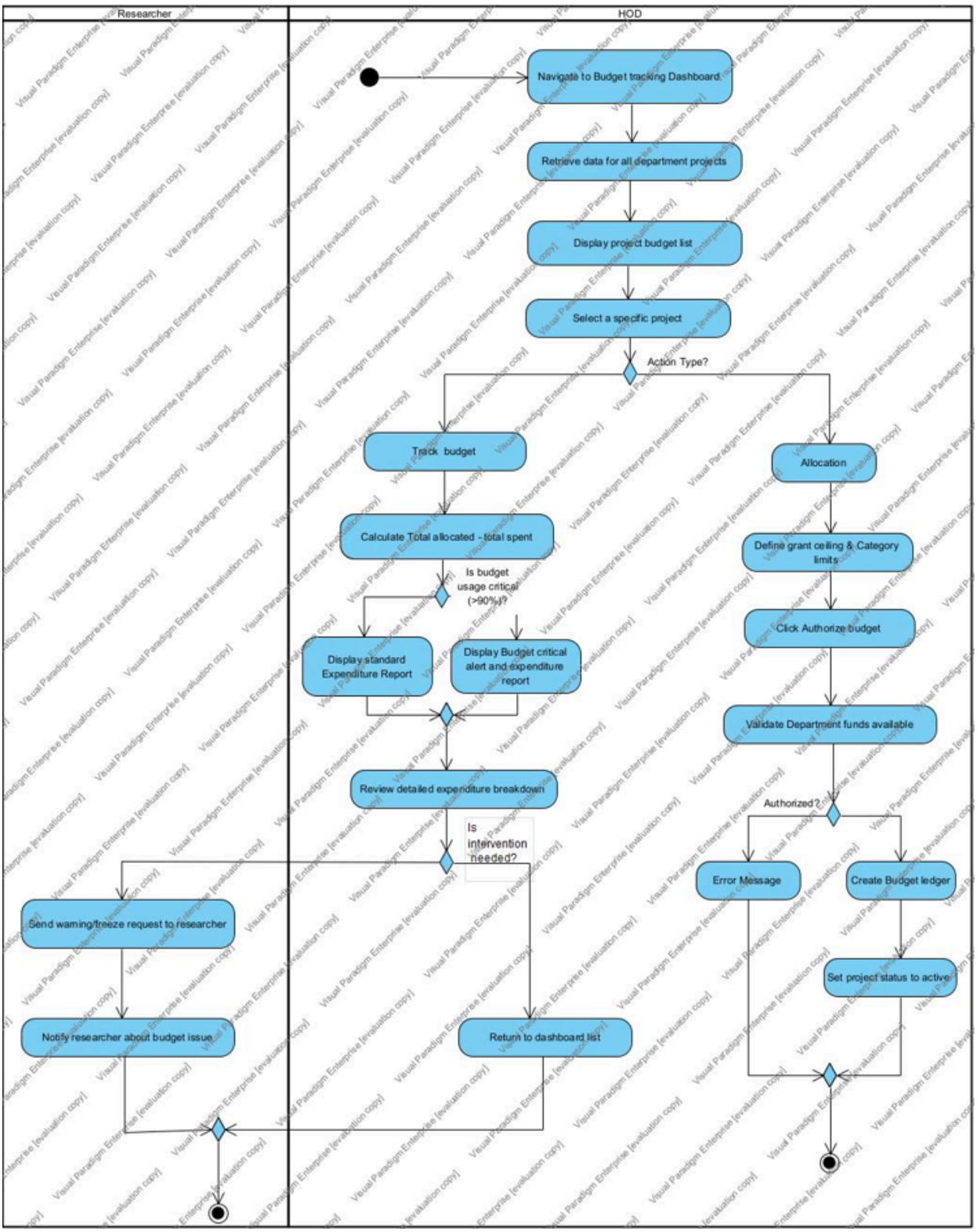
3.3.2 Monitor Research Project Progress

To track performance, the HOD enters a search for a specific project, prompting the system to retrieve the relevant milestones and submitted deliverables. The system displays the project's current progress timeline, percentage of completion, and any adherence issues. Based on this view, the HOD enters a status flag (e.g., "On Track" or "Needs Intervention"), which the system then stores as a progress log in the project file for future auditing.



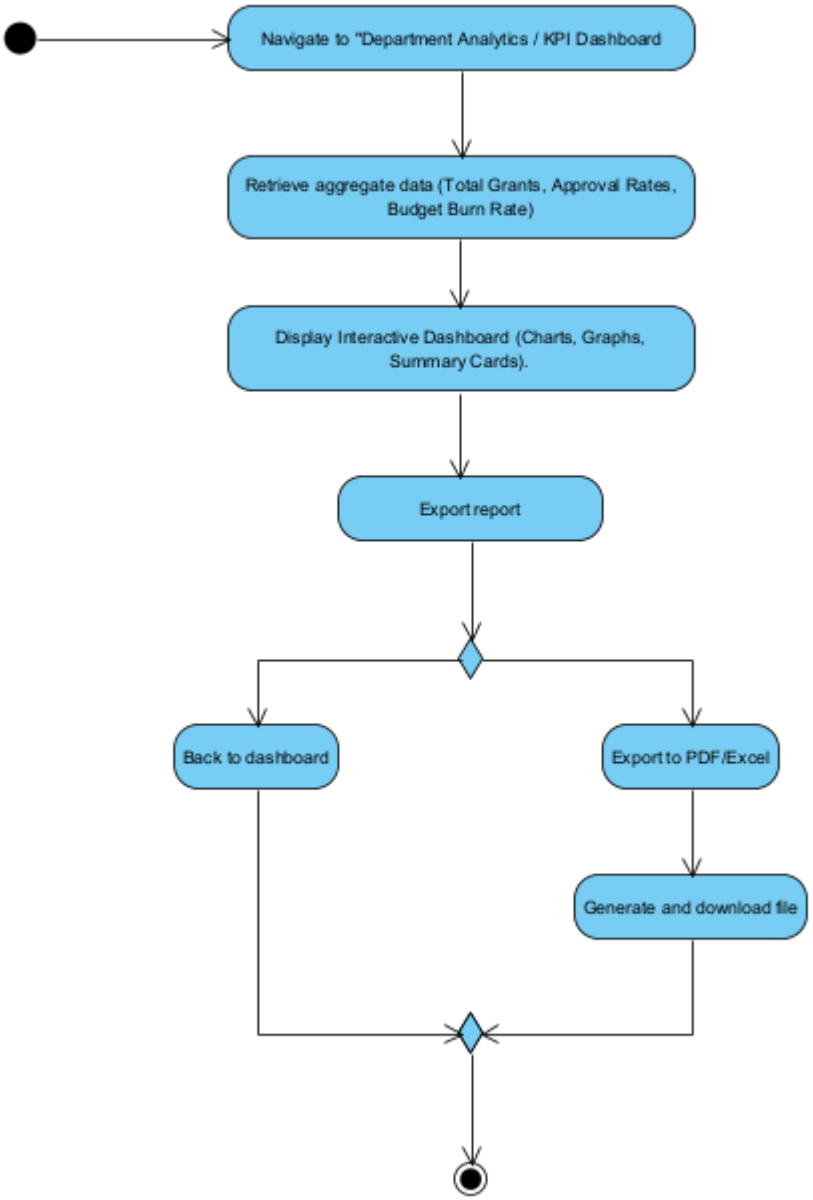
3.3.3 Grant Allocation & Budget Tracking

This use case describes how the **Head of Department (HOD)** monitors the financial health of active research projects. The system automatically calculates budget usage (Total Allocated vs. Total Spent) and provides visual indicators for the HOD. The process focuses on identifying projects with critical budget usage (e.g., >90%) or those exceeding limits, allowing the HOD to review detailed expenditure reports and intervene by sending warnings or freezing spending if necessary.


Researcher
HOD
Navigate to Budget tracking Dashboard
Retrieve data for all department projects
Display project budget list
Select a specific project
Action Type?
Track budget
Calculate Total allocated - total spent
Is budget usage critical (>90%)?
Display standard Expenditure Report
Display Budget critical alert and expenditure report
Review detailed expenditure breakdown
Is intervention needed?
Send warning/freeze request to researcher
Notify researcher about budget issue
Return to dashboard list
Allocation
Define grant ceiling & Category limits
Click Authorize budget
Validate Department funds available
Authorized?
Error Message
Create Budget ledger
Set project status to active

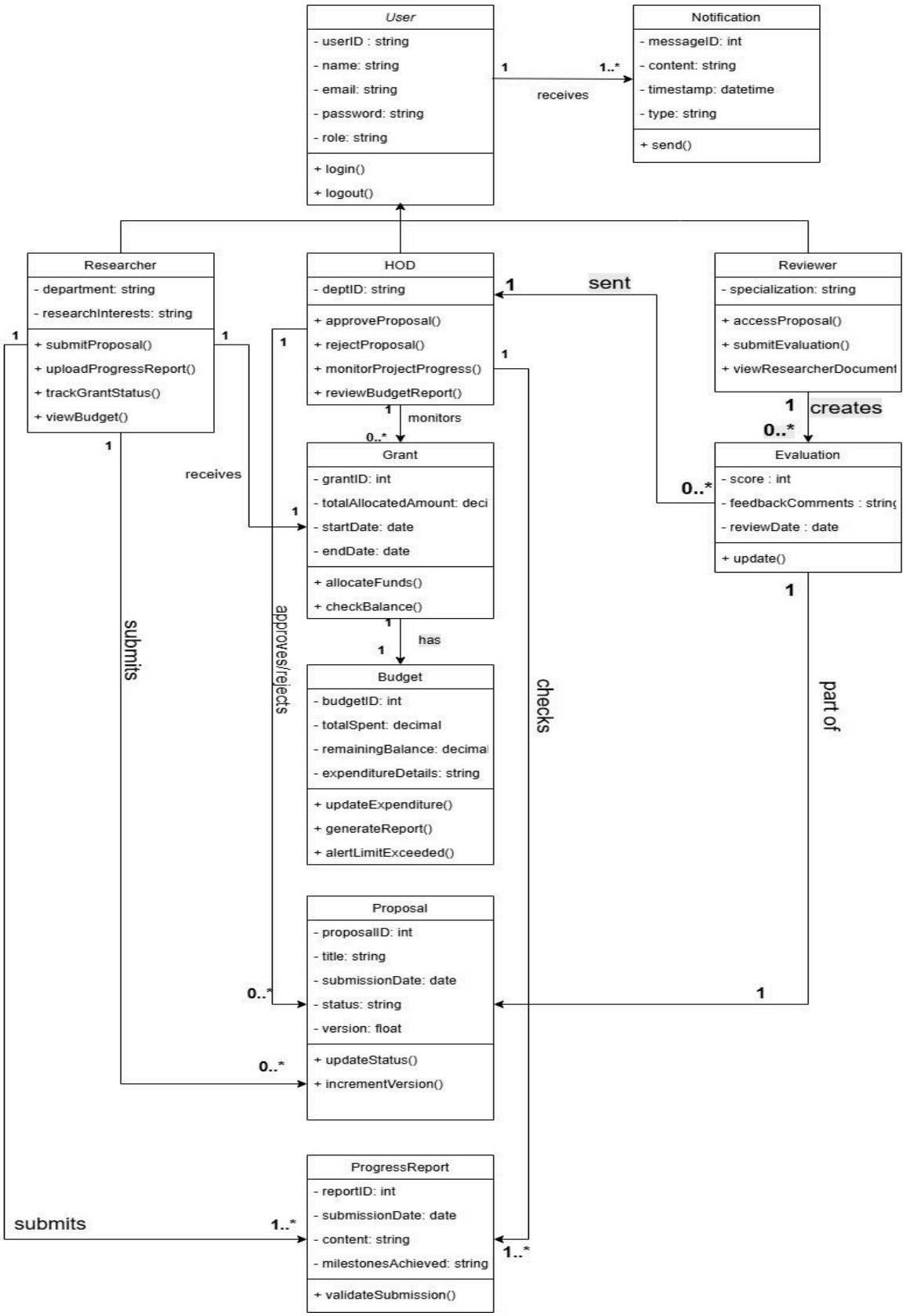
3.3.4 Monitor Research Dashboard (KPIs)

The HOD analyzes departmental trends by entering specific filter criteria, such as a date range (e.g., 2024-2025) or faculty name. The system retrieves aggregated data on all proposals and funds to calculate performance metrics. It then displays visual Key Performance Indicators (KPIs), such as success rate charts and total funding graphs, while storing the HOD's query parameters and session activity for usage tracking.



4 System Models

4.1 Class Diagrams / ERD



4.2 Classes / Entities

Class / Entity	Description	Relationship
User	The base class for all system users. It has attributes like **userID**, **name**, **email**, **password**, and **role**. Operations include **login()** and **logout()**.	**Parent Class** for Researcher, Reviewer, and HOD. **1** User receives **0 to many** Notifications.
Notification	Represents a message received by a User. Attributes include **messageID**, **content**, **timestamp**, and **type**. The operation is **send()**.	**Associated with** User (**0 to many** Notifications belong to **1** User).
Researcher	A type of user who can **submitProposal()**, **uploadProgressReport()**, **trackGrantStatus()**, and **viewBudget()**. Has attributes for **department** and **researchInterests**.	**Inherits from** User. **1** Researcher submits **0 to many** Proposals. **1** Researcher uploads **0 to many** ProgressReports.
Reviewer	A type of user responsible for the review process. Can **accessProposal()**, **submitEvaluation()**, and **viewResearcherDocuments()**. Has a **specialization** attribute	**Inherits from** User. **1** Reviewer creates **0 to many** Evaluations. **1** Reviewer assesses **0 to many** Proposals.
HOD	A type of user who can **approveProposal()**, **rejectProposal()**, **monitorProjectProgress()**, and **reviewBudgetReport()**. Has a **deptID** attribute.	**Inherits from** User. Associated with Evaluation (**1** HOD receives **0 to many** Evaluations) **1** HOD approves/rejects **0 to many*** Proposals. **1** HOD monitors **0 to many** Grants.

Evaluation	Represents a Reviewer's assessment of a Proposal. Contains a numerical **score**, **feedbackComments**, and the **reviewDate**. Can be updated with **update()**.	**Associated with** Reviewer (**1** Evaluation is created by **1** Reviewer). **Part of** Proposal (**0 to many** Evaluations belong to **1** Proposal via Composition) Associated with HOD (**0 to many** Evaluations are sent to **1** HOD)
Grant	Represents awarded funding for an approved Proposal. Attributes are **grantID**, **totalAllocatedAmount**, **startDate**, and **endDate**. Operations are **allocateFunds()** and **checkBalance()**.	**Associated with** Proposal (**1** Grant comes from **1** Proposal). **Has** Budget (**1** Grant has **1** Budget via Composition).
Proposal	Represents a research proposal submitted by a Researcher. Attributes are **proposalID**, **title**, **submissionDate**, **status**, and **version**. Operations include **updateStatus()** and **incrementVersion()**.	**Associated with** Researcher (**0 to many** Proposals are submitted by **1** Researcher). **Has** Evaluation (**1** Proposal has **0 to many** Evaluations via Composition).
Budget	Details the financial status associated with a Grant. Attributes include **budgetID**, **totalSpent**, **remainingBalance**, and **expenditureDetails**. Operations are **updateExpenditure()**, **generateReport()**, and **alertLimitExceeded().**	**Part of** Grant (**1** Budget belongs to **1** Grant via Composition). **Associated with** HOD (**1** Budget is monitored by **1** HOD).
ProgressReport	Documents the progress of a research project. Attributes are **reportID**, **submissionDate**, **content**, and **milestonesAchieved**. The operation is **validateSubmission().**	**Associated with** Researcher (**0 to many** Reports are uploaded by **1** Researcher). **Associated with** HOD (**0 to many** Reports are monitored by **1** HOD).

5 Non-Functional Requirements

*<This section is **Optional**. Define any other requirements not covered elsewhere in the SRS. This might include performance, security, usability, and reliability requirements. Add any new sections that are pertinent to the project.>*

6 References

[1] GeeksforGeeks, "Unified Modeling Language (UML) | Class Diagrams," GeeksforGeeks. Available: https://www.geeksforgeeks.org/system-design/unified-modeling-language-uml-class-diagrams/

[2] GeeksforGeeks, "Unified Modeling Language (UML) | Use Case Diagrams," GeeksforGeeks. Available: https://www.geeksforgeeks.org/system-design/use-case-diagram/ [Accessed:

[3] Visual Paradigm, "Drawing Class Diagrams," Visual Paradigm User Guide. Available: https://www.visual-paradigm.com/support/documents/vpuserguide/94/2576/7190_drawingclass.html

[4] Visual Paradigm, "Drawing Activity Diagrams," Visual Paradigm User Guide. Available: https://www.visual-paradigm.com/support/documents/vpuserguide/94/2580/6713_drawingactiv.html