Software Design Specification

for

Academic Publication and Research Tracking System

Version <X.X>

Group No.: 1

Agilan A/L Buminathan	**243UC245F7**
Aniqah Nabilah Binti Azhar	**242UC244LQ**
Feqhah Delilah Binti Mohd Faizul	**242UC244FY**
Nur Aleez Dania Binti Mohd Shahrul Azman	**242UC244QB**

Date: **25 JANUARY 2026**

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 System Overview

2.1 Description

This system will enable both lecturers and students to upload details of academic publications, such as those appearing in journals, conference proceedings, and book chapters. Every submission requires supporting documents (PDF, DOI, acceptance letter), which afterwards shall be reviewed and verified by the Programme Coordinator or Admin.

Core processes include:

- User Registration & Authentication
- Publication Entry & Management
- Document Upload
- Verification & Approval Workflow
- Analytics Dashboard (by year, type, author, index)
- Exportable KPI Reports
- Notifications (reminders, missing documents)
- Research Profile linking (ORCID, Google Scholar)
- Admin System Settings & User Management

2.2 Actors

ADMIN	Manage Users Accounts
	Manage System Settings
	Manage Programme
	View System Reports
PROGRAM COORDINATOR	Verify publication
	Approve/Reject Publication
	Eligibility for Student's Graduation
	Generate Reports, KPI
	Monitor Research Output
LECTURE/STUDENT WHEN MAIN AUTHOR	Add Publication
	Upload Supporting Documents
	Edit Publication Details
	Delete Publication
	Shared publication to Professional Platforms
	Resubmit Publication (If program coordinator rejects publication)
LECTURER/STUDENT WHEN CO AUTHOR	View Publication Status
	View Comment/Remarks
	Share publication to Professional Platforms

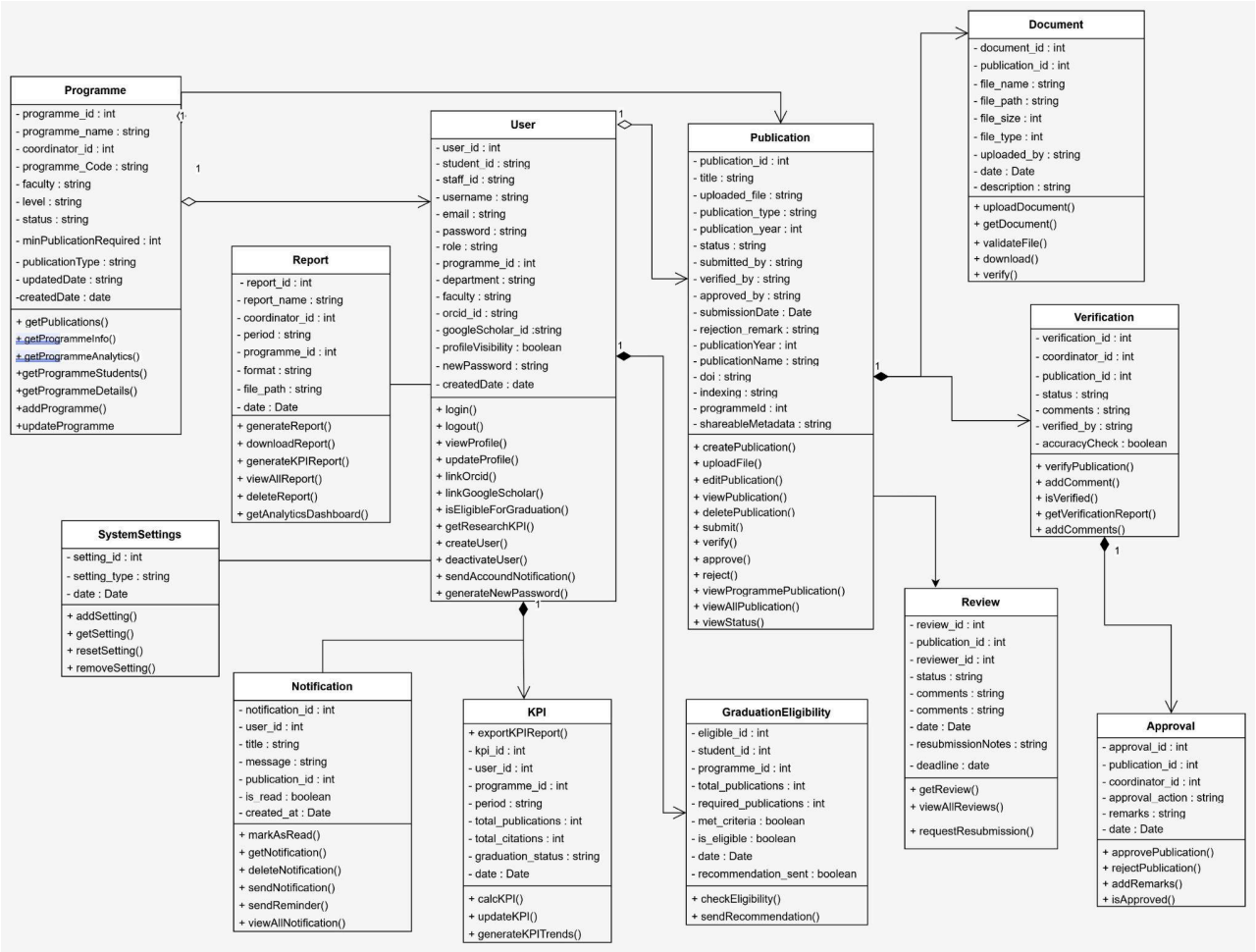
2.3 Assumptions and Dependencies

- Users must have a valid university email to register.
- The system assumes stable internet connectivity for document uploads.
- The verification workflow depends on Programme Coordinator/Admin availability.
- External integrations (ORCID, Google Scholar) depend on third-party API availability.
- Maximum publication file upload size is assumed to be ≤ 20MB.
- The university database and server infrastructure are available and accessible.

2.4 Use Case Diagram

2 Data Design

2.1 Design Class Diagram



2.2 Data Dictionary

Class/Entity	Attribute	Data Type	Description
User	user_id	Integer	Unique identifier for the system user.
	student_id	String	Official university student identification number
	staff_id	String	Official university staff

			identification number
	username	String	Unique login name
	email	String	Official email address for notifications and login
	password	String(Hashed)	Encrypted user authentication credential
	role	Enum	User permission level : coordinator, lecturer, student, admin
	programme_id	Integer(FK)	Links the user to a specific academic programme
	department	String	The academic department the user belongs to.
	faculty	String	The faculty associated with the user
	Orcid_id	String	Researcher's Open Researcher Contributor ID
	GoogleScholar_id	String	ID linking to user's Google Scholar profile
	profileVisibility	Boolean	Controls if the profile is public(true) or private(false)
	createDate	DateTime	Timestamp of account creation
Programme	programme_id	Integer	Unique identifier for the programme
	programme_name	String	Full name of the degree or course
	Coordinator_id	Integer(FK)	Reference to the user(staff) managing the programme
	minPublicationRequired	Integer	Minimum number of publications needed for graduation

	Level	String	Academic level Example : Masters, PHD, Undergraduate
	Status	String	Current status of the programme(Active/Inactive)
Publication & Documents	publication_id	Integer	Unique identifier for the publication record
	title	String	Title of the research paper of article
	status	Enum	Workflow state: pending, approved, rejected
	submitted_by	Integer(FK)	User ID of the student or lecturer who uploaded it
	file_path	String	Server path or URL to the stored document (PDF/DOCX)
	file_size	Float	Size of the file in MB
Verification & Approval	verification_id	Integer	Unique ID for the verification step
	approval_id	Integer	Unique ID for the final approval step
	action/status	Enum	Result of the review : Approved, Rejected, Pending
	comments/remarks	Text	Feedback provided by the Coordinator during review
KPI & GraduationEligibility	kpi_id	Integer	Unique identifier for the performance record.
	total_publications	Integer	Total count of approved publications for a user.
	total_citations	Integer	Total citations pulled from external IDs(Orcid/Scholar)
	is_eligible	Boolean	Flag indicating if a student has met all criteria

	met_criteria	Text	Summary of which requirements were met
Notification & SystemSettings	notification_id	Integer	Unique id for the alert
	message	Text	The content of the notification sent to the user
	is_read	Boolean	Status indicating if the user has seen the alert
	setting_type	String	Category of system setting Example : Email server, UI theme

2.3 Data Structures

2.3.1 Users

- User_id (int) : Unique identifier for the system user.
- Student_id (str) : Official university student identification number
- Staff_id (str) : Official university staff identification number
- Username (str) : Unique login name
- Email (str) : Official email address for notifications and login
- Password (str) : Encrypted user authentication credential
- Role (enum) : User permission level : coordinator, lecturer, student, admin
- Programme_id (int) : Links the user to a specific academic programme
- Department (str) : The academic department the user belongs to
- Faculty (str) : The faculty associated with the user
- Orchid_id (str) : Researcher's Open Researcher Contributor ID
- GoogleScholar_id (str) : ID linking to user's Google Scholar profile
- profileVisibility (bool) : Controls if the profile is public(true) or private(false)
- createDate (DateTime) : Timestamp of account creation

3.3.1 Users Data Structures

Attribute	**Data Type**
user_id	Integer
student_id	String
staff_id	String

username	String
email	String
password	String(Hashed)
role	Enum
programme_id	Integer(FK)
department	String
faculty	String
Orcid_id	String
GoogleScholar_id	String
profileVisibility	Boolean
createDate	DateTime

2.3.2 Programme

- Programme_id (int) : Unique identifier for the programme
- Programme_name (str) : Full name of the degree or course
- Coordinator_id (int) : Reference to the user(staff) managing the programme
- minPublicationRequired (int) : Minimum number of publications needed for graduation
- Level (str) : Academic level. Example : Masters, PHD, Undergraduate
- Status (str) : Current status of the programme(Active/Inactive)

3.3.2 Programme Data Structures

Attribute	**Data Type**
programme_id	Integer
programme_name	String
Coordinator_id	Integer(FK)
minPublicationRequired	Integer
Level	String

Status	String

2.3.3 Publication

- Publication_id (int) : Unique identifier for the publication record
- Title (str) : Title of the research paper of article
- Status (enum) : Workflow state: pending, approved, rejected
- Submitted_by (int) : User ID of the student or lecturer who uploaded it
- File_path (str) : Server path or URL to the stored document (PDF/DOCX)
- File_size (float) : Size of the file in MB

3.3.3 Publication Data Structures

Attribute	**Data Type**
publication_id	Integer
title	String
status	Enum
submitted_by	Integer(FK)
file_path	String
file_size	Float

2.3.4 Verification & Approval

- Verification_id (int) : Unique ID for the verification step
- Approval_id (int) : Unique ID for the final approval step
- action/status (enum) : Result of the review (Approved, Rejected, Pending)
- comments/remarks (text) : Feedback provided by the Coordinator during review

3.3.4 Verification & Approval Data Structures

Attribute	**Data Type**
verification_id	Integer
approval_id	Integer
action/status	Enum

comments/remarks	Text

2.3.5 KPIs & GraduationEligibility

- Kpi_id (int): Unique identifier for the performance record.
- Total_publications (int) : Total count of approved publications for a user.
- Total_citations (int) : Total citations pulled from external IDs(Orcid/Scholar)
- Is_eligible (boolean) : Flag indicating if a student has met all criteria
- Met_criteria (text) : Summary of which requirements were met

3.3.5 KPIs & Graduation Eligibility Data Structures

Attribute	**Data Type**
kpi_id	Integer
total_publications	Integer
total_citations	Integer
is_eligible	Boolean
met_criteria	Text

2.3.6 Notification & SystemSettings

- Notification_id (int) : Unique id for the alert
- Message (text) : The content of the notification sent to the user
- Is_read (boolean) : Status indicating if the user has seen the alert
- Setting_type (string) : Category of system setting.For example : Email server, UI theme

3.3.6 Notifications & System Settings Data Structures

Attribute	**Data Type**
notification_id	Integer
message	Text
is_read	Boolean
setting_type	String

3 Behavioral Modeling

3.1 Sequence Diagrams

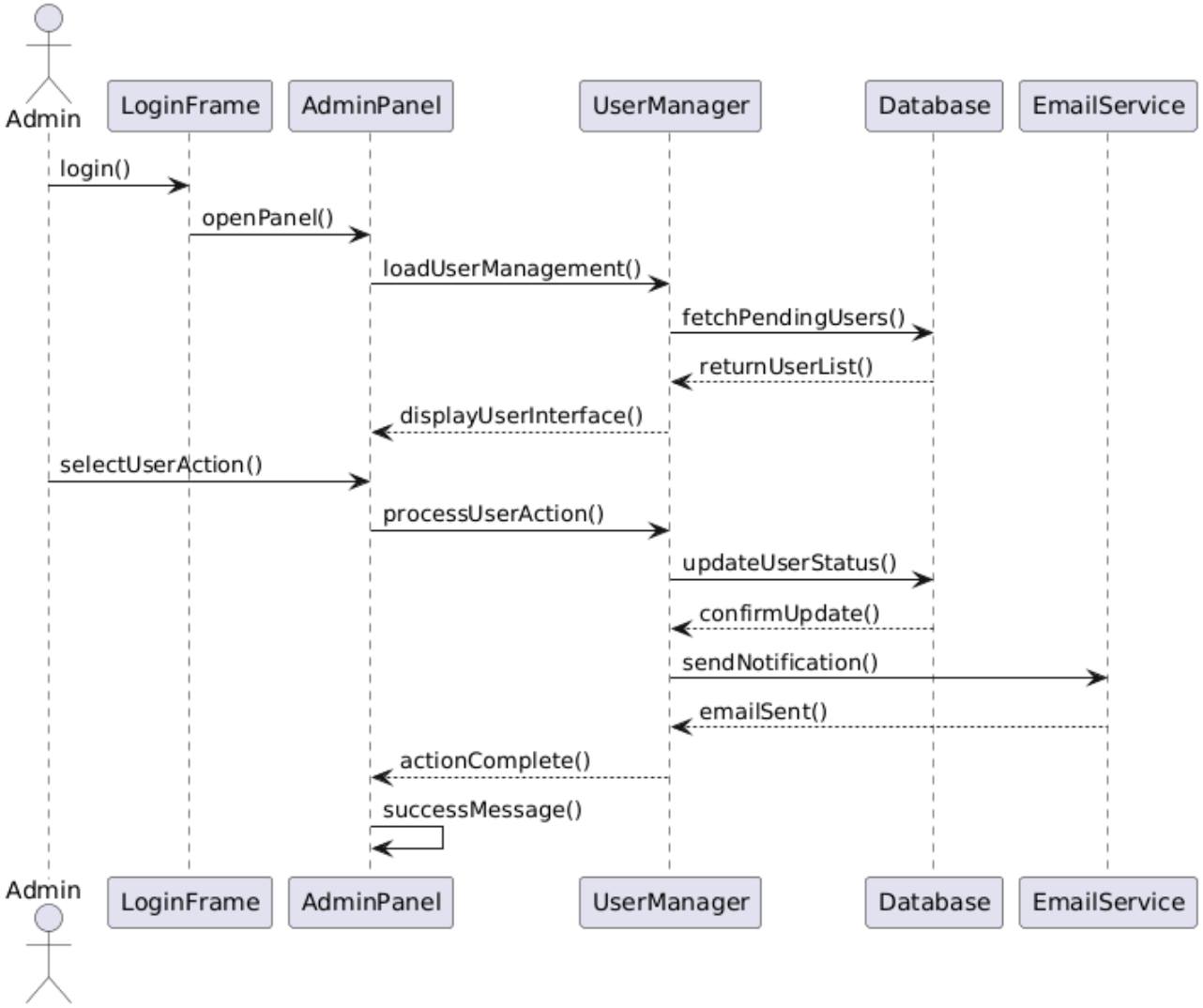
3.1.1 Admin

3.1.1.1 Manage Users Accounts

Table 3.1.1.1 Manage Users Accounts Descriptions

Use Case Name	Manage Users Accounts
Description	Admin approves new user registrations, edits existing user accounts, and deactivates accounts as needed.
Primary Actor	Admin
Precondition	1. The admin is logged into the system with appropriate privileges. 2. The user management module is accessible and operational.
Postcondition	1. User account data is accurately updated in the system database. 2. Relevant users are notified of account changes where applicable
Main Success Scenario	1. 1. Admin accesses user management. 2. 2. Views pending requests/existing accounts. 3. 3. Selects action (approve/edit/deactivate). 4. 4. Performs required action. 5. 5. System processes changes. 6. 6. Admin sees confirmation.
Alternative Scenario	Reject User Request 1. Admin reviews pending request. 2. Decides to reject instead of approve. 3. Provides rejection reason. 4. System notifies applicant. 5. Request removed from queue.

Figure 3.1.1.1 Manage Users Accounts Sequence Diagram



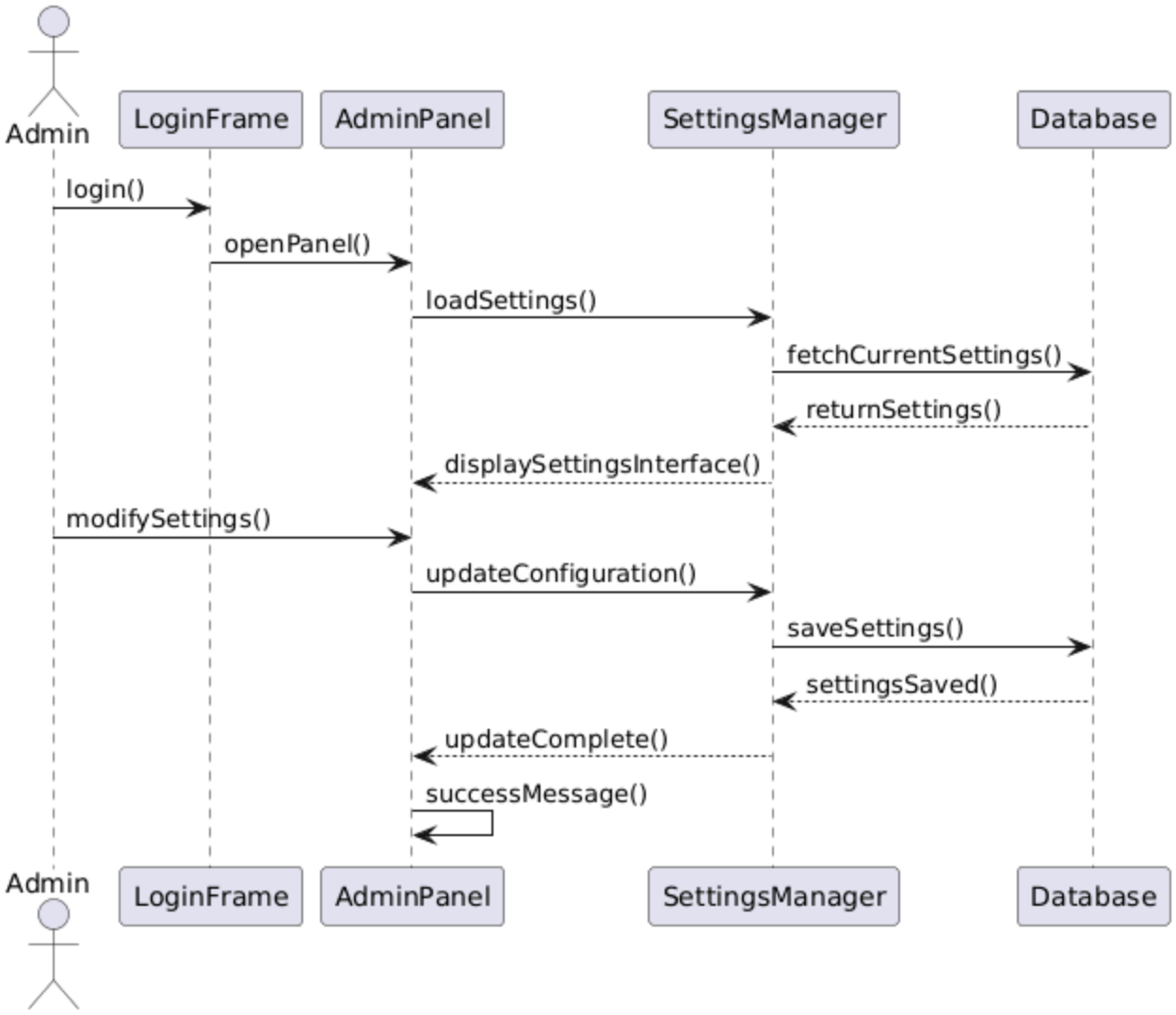
3.1.1.2 Manage System Settings

Table 3.1.1.2 Manage System Settings Descriptions

Use Case Name	Manage System Settings
Description	Admin configures system-wide settings including publication types, workflow rules, user permissions, and KPI calculations.
Primary Actor	Admin
Precondition	1. Admin is logged into the system. 2. Admin has system settings permissions.

	3. Settings interface is accessible.
Postcondition	System settings updated and applied immediately.
Main Success Scenario	1. Admin opens system settings. 2. Selects setting category. 3. Views current configuration. 4. Makes required changes. 5. Saves settings. 6. System applies changes. 7. Admin sees save confirmation.
Alternative Scenario	Invalid Settings 1. Admin enters invalid values. 2. System validates and rejects. 3. Shows error messages. 4. Admin corrects values. 5. Settings saved successfully.

Figure 3.1.1.2 Manage System Settings Sequence Diagram



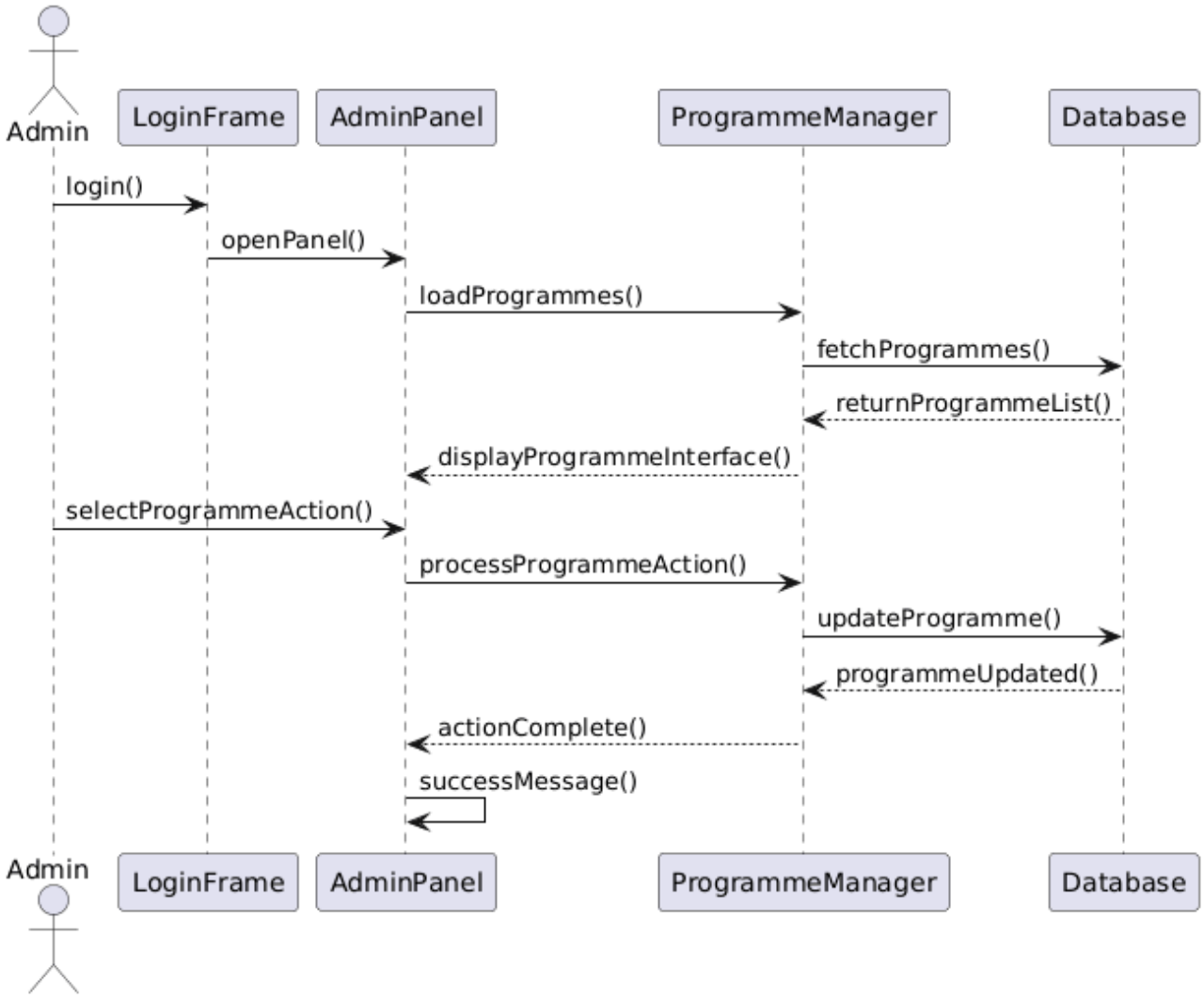
3.1.1.3 Manage Programme

Table 3.1.1.3 Manage Programme Descriptions

Use Case Name	Manage Programme
Description	Admin creates, updates, and manages academic programmes, assigns coordinators, and sets publication requirements.
Primary Actor	Admin
Precondition	1. Admin is logged into system. 2. Admin has programme management permissions.

	3. Programme data structure exists.
Postcondition	Programmes added/updated, coordinators assigned, requirements set.
Main Success Scenario	1. Admin accesses programme management. 2. Selects action (add/edit/assign/set). 3. Performs required operations. 4. System processes changes. 5. Relevant parties notified. 6. Admin sees completion confirmation.
Alternative Scenario	Programme Deactivation 1. Admin selects programme to deactivate. 2. System checks for existing users/publications. 3. Shows warning if dependencies exist. 4. Admin confirms deactivation. 5. Programme marked inactive.

Figure 3.1.1.3 Manage Programme Sequence Diagram



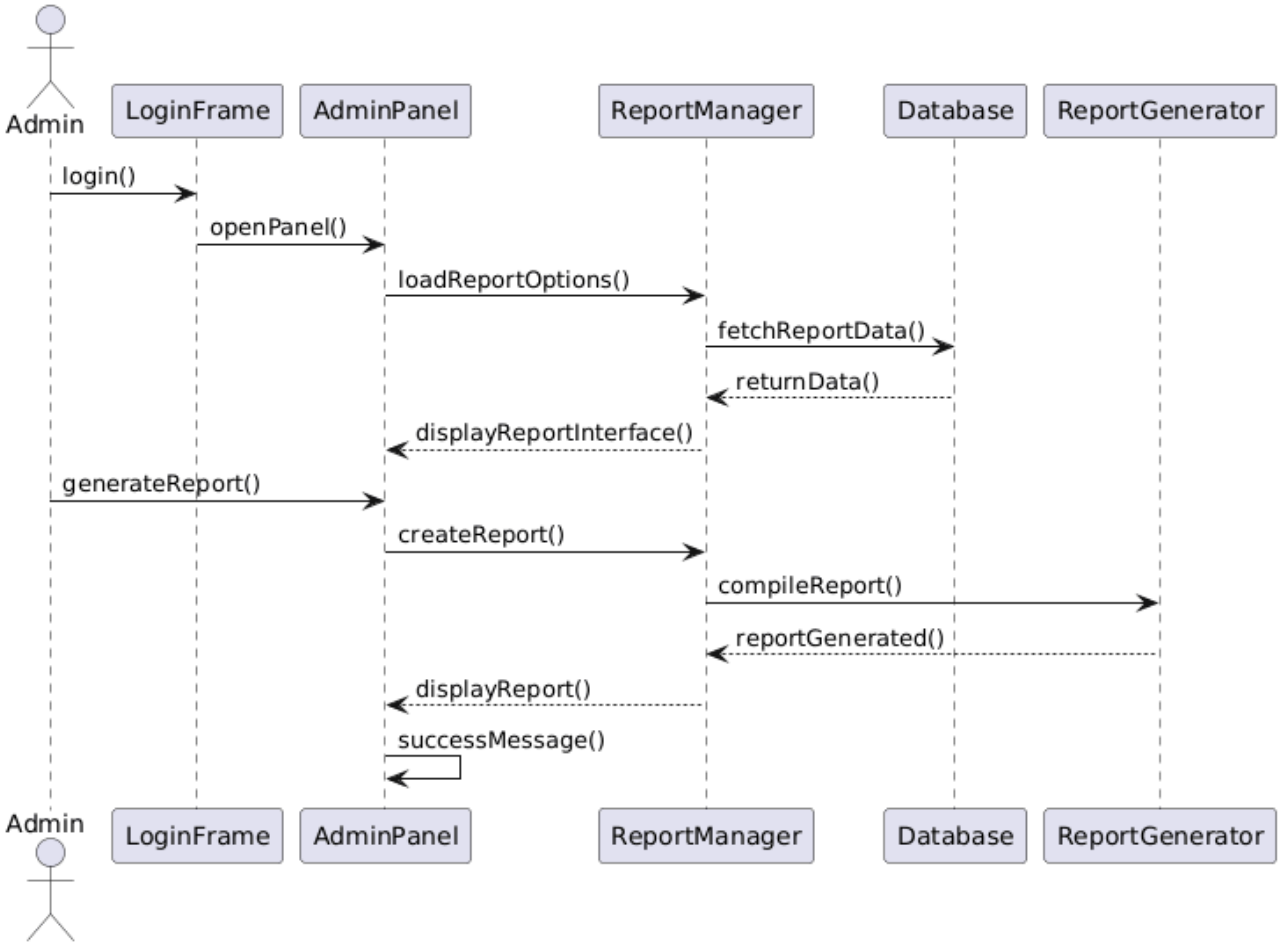
3.1.1.4 View System Reports

Table 3.1.1.4 View System Reports Descriptions

Use Case Name	View System Reports
Description	Admin generates and views comprehensive reports on publication activities, user behavior, and KPI performance across the system.
Primary Actor	Admin
Precondition	1. Admin is logged into system.

	2. Report data exists in system. 3. Analytics engine is operational.
Postcondition	Reports generated, viewed, and optionally exported.
Main Success Scenario	1. Admin accesses report dashboard. 2. Selects report type. 3. Applies filters. 4. Generates report. 5. Views results. 6. Optionally exports report. 7. Downloads exported file.
Alternative Scenario	No Data Available 1. Selected filters return no data. 2. System shows "No data available" message. 3. Admin adjusts filters. 4. Regenerates report.

Figure 3.1.1.3 View System Reports Sequence Diagram



3.1.2 Programme Coordinator

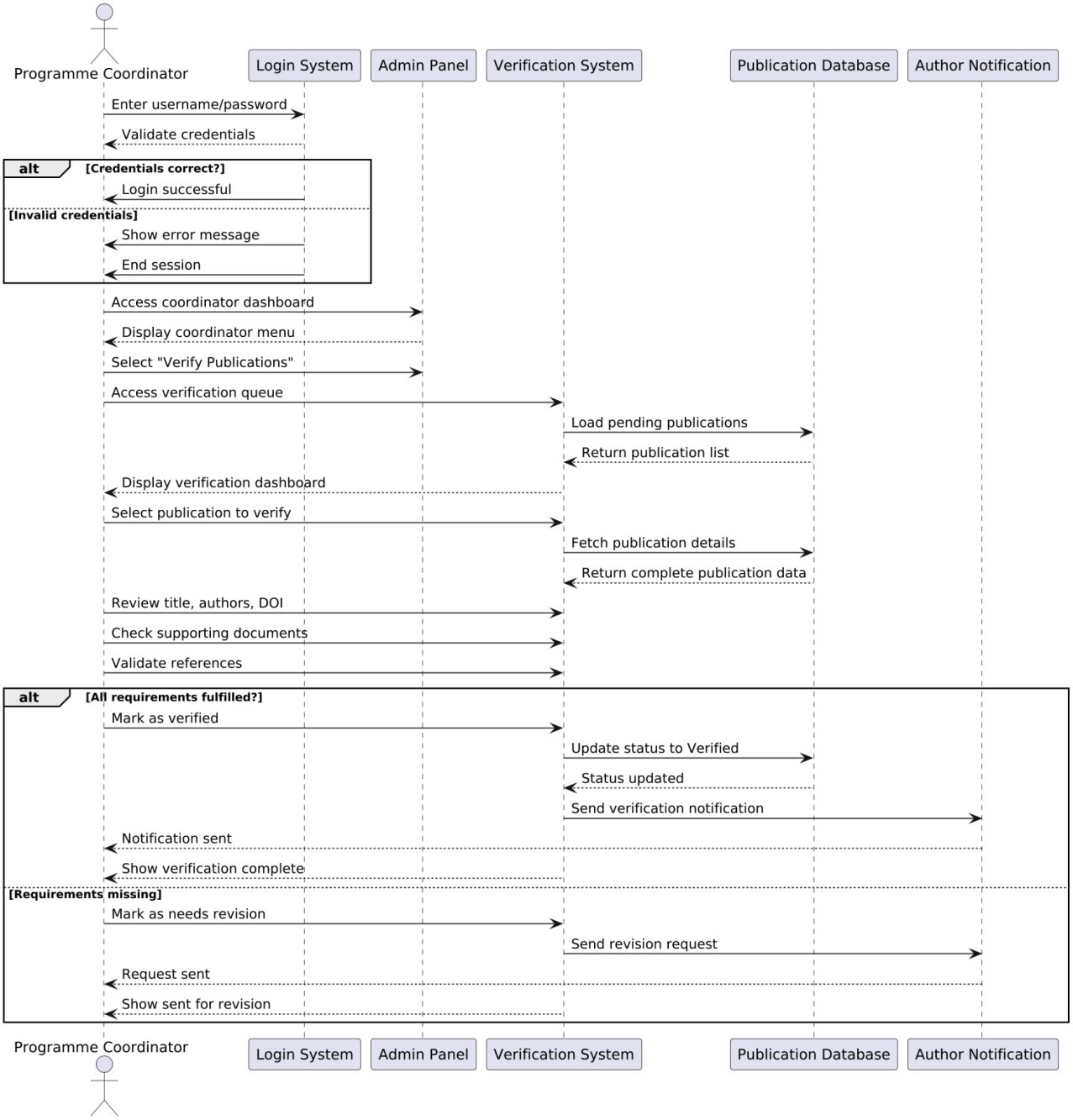
3.1.2.1 Verify Publication

Table 3.1.2.1 Verify Publication Descriptions

Use Case Name	Verify Publication
Description	Coordinator checks publication details for accuracy, completeness, and compliance with university standards before marking as verified.
Primary Actor	Programme Coordinator
Precondition	1. Coordinator is authenticated and logged in. 2. Admin Panel shows coordinator menu. 3. Publications are in "Pending Verification" status. 4. Coordinator has access to the verification queue.

Postcondition	Publication status updated to "Verified" or marked for revision.
Main Success Scenario	1. Coordinator logs in successfully. 2. Accesses Admin Panel → selects "Verify Publications". 3. Views pending publications. 4. Reviews publication details. 5. Marks as verified if all requirements met. 6. System updates status and notifies author.
Alternative Scenario	Login Failed 1. Invalid credentials entered. 2. System shows an error message. 3. Coordinator cannot access Admin Panel. Requirements Missing 1. Coordinator finds incomplete data. 2. Marks publication as needs revision. 3. System sends a revision request.

Figure 3.1.2.1 Verify Publication Sequence Diagram

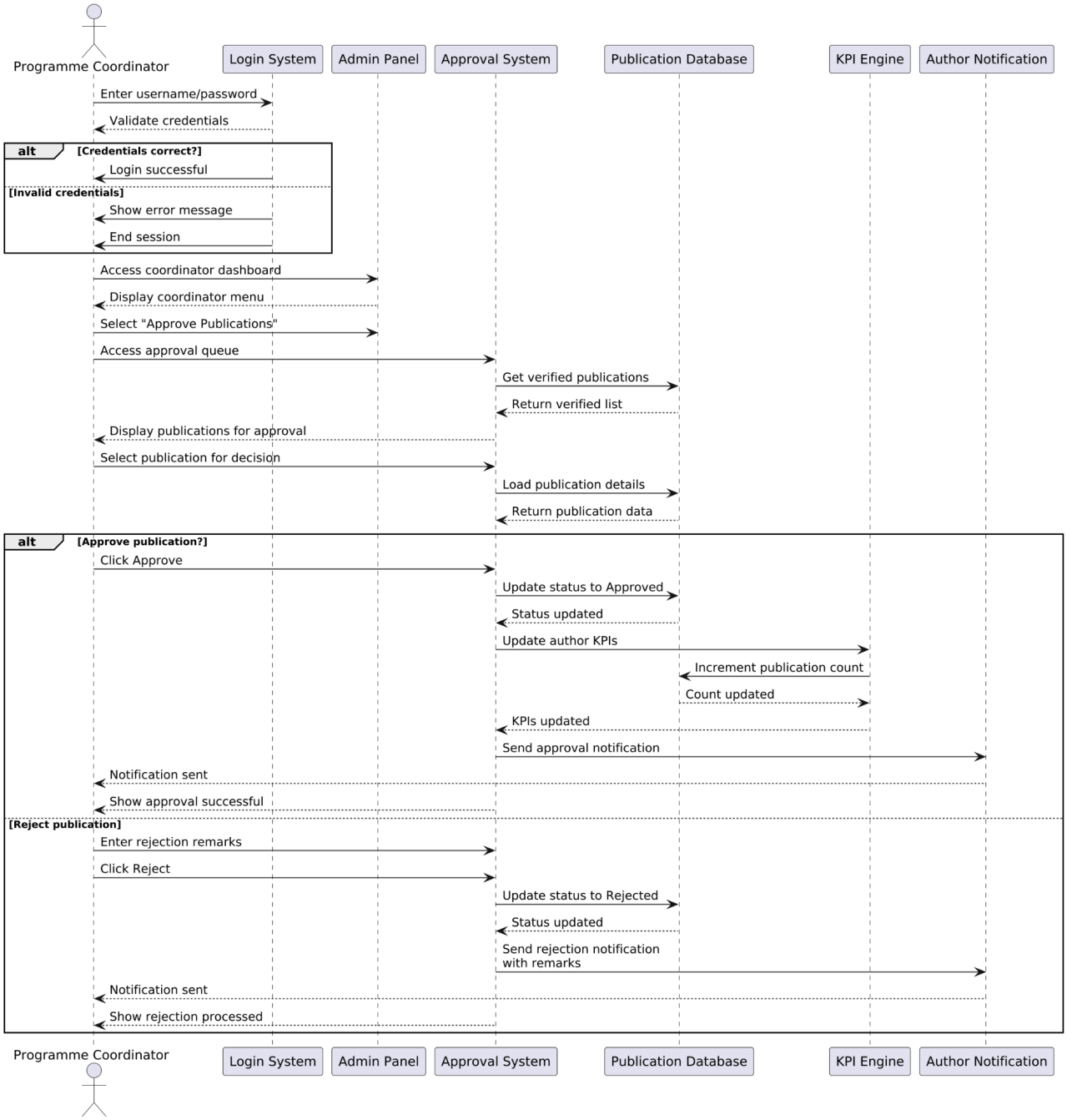


3.1.2.2 Approve/Reject Publication

Table 3.1.2.2 Approve/Reject Publication Descriptions

Use Case Name	Approve/Reject Publication
Description	Coordinator makes final decision on verified publications, either approving for official records or rejecting with feedback.
Primary Actor	Programme Coordinator
Precondition	1. Coordinator is authenticated and logged in. 2. Admin Panel shows coordinator menu. 3. Publications are in "Verified" status. 4. Coordinator has approval permissions.
Postcondition	Publication approved (added to research repository) or rejected (returned to author).
Main Success Scenario	1. Coordinator logs in successfully. 2. Accesses Admin Panel then selects "Approve Publications". 3. Views verified publications. 4. Reviews final details. 5. Clicks Approve. 6. System updates status, updates KPIs, notifies author.
Alternative Scenario	Reject Publication 1. Coordinator finds issues in verified publication. 2. Enters rejection remarks. 3. Clicks Reject. 4. System updates status and notifies author with feedback. 5. Publication returns to author for possible resubmission.

Figure 3.1.2.2 Approve/Reject Publication Sequence Diagram

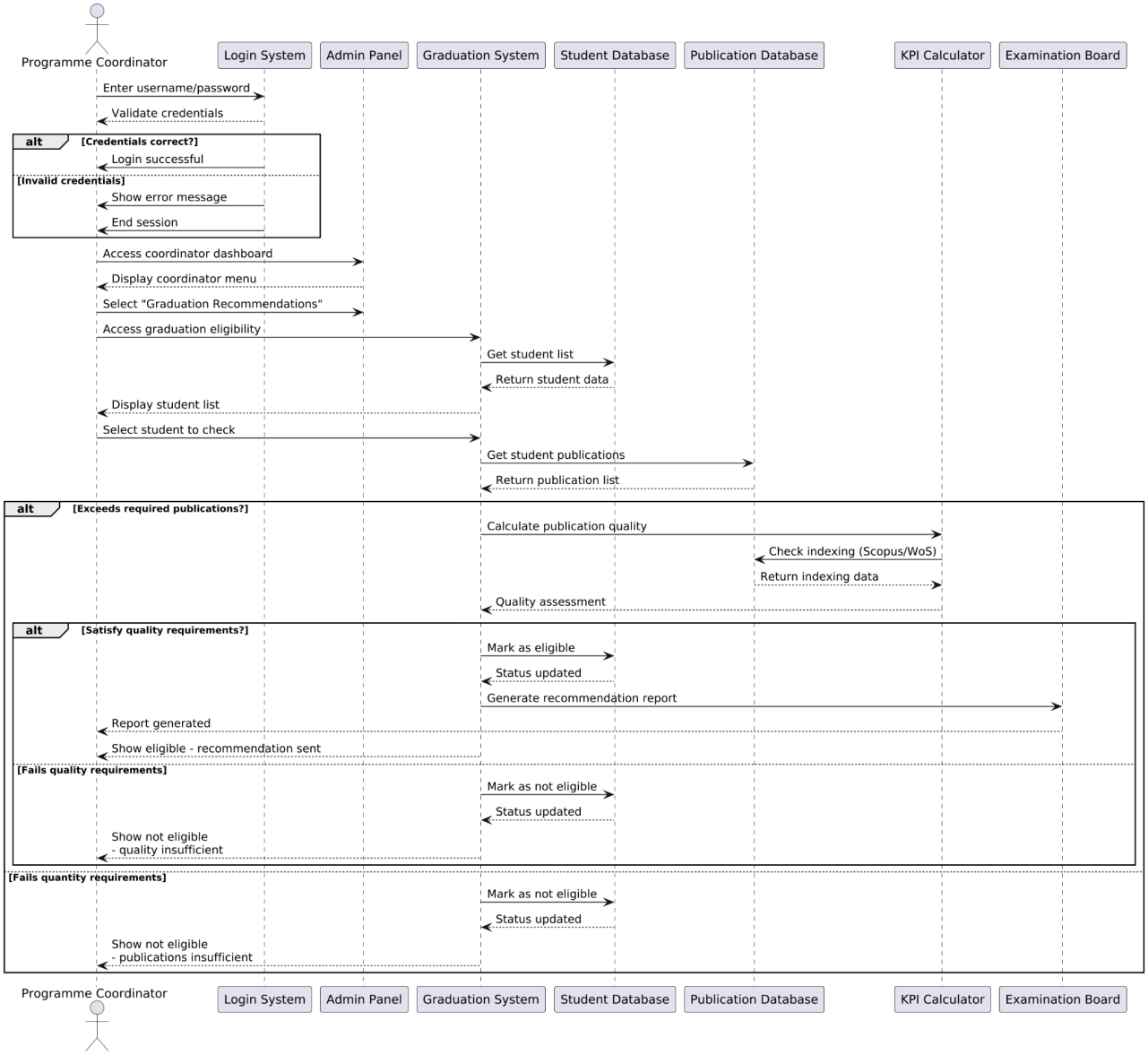


3.1.2.3 Recommend Students for Graduation

Table 3.1.2.3 Recommend Students for Graduation Descriptions

Use Case Name	Recommend Students for Graduation
Description	Coordinator checks if students meet publication requirements for graduation eligibility and generates recommendation reports.
Primary Actor	Programme Coordinator
Precondition	1. Coordinator is authenticated and logged in. 2. Admin Panel shows coordinator menu. 3. Student has completed coursework. 4. Student has submitted publications. 5. Graduation period is active.
Postcondition	Students marked as eligible/ineligible with recommendation report sent to examination board.
Main Success Scenario	1. Coordinator logs in successfully. 2. Coordinator accesses graduation eligibility. 3. Selects student to check. 4. System checks publication quantity. 5. System checks publication quality (indexing). 6. Both requirements met - marks as eligible. 7. System generates recommendation report. 8. Report sent to examination board. 9. Coordinator sees eligibility confirmation.
Alternative Scenario	Login Failed 1. Invalid credentials entered. 2. System shows error message. 3. Coordinator cannot access Admin Panel. Not Eligible 1. Student fails quantity or quality check. 2. System marks as not eligible. 3. Coordinator sees reason for ineligibility. 4. Student notified of missing requirements. No Students Found 1. No students in graduation period. 2. System displays "No students available". 3. Coordinator exits module.

Figure 3.1.2.3 Recommend Students for Graduation Sequence Diagram

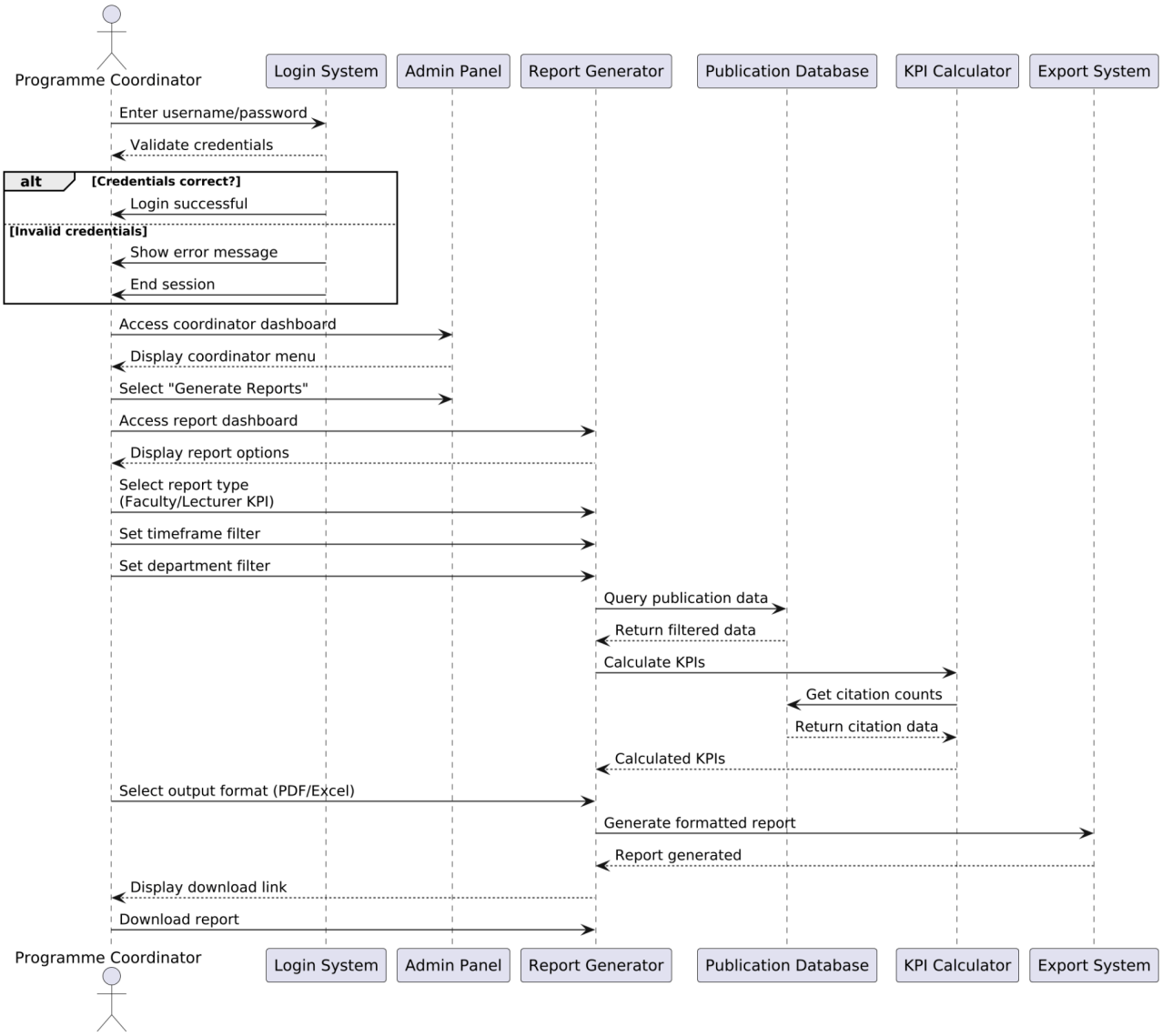


3.1.2.4 Generate Reports &KPI

Table 3.1.2.4 Generate Reports &KPIs Descriptions

Use Case Name	Generate Reports &KPI
Description	Coordinator generates customized reports on research performance, KPIs, and publication metrics for assessment and planning.
Primary Actor	Programme Coordinator
Precondition	1. Coordinator is authenticated and logged in. 2. Admin Panel shows coordinator menu. 3. Publication data exists in system. 4. Coordinator has report generation permissions. 5. Filters and parameters can be set.
Postcondition	Customized report generated and available for download.
Main Success Scenario	1. Coordinator logs in successfully. 2. Accesses Admin Panel then selects "Generate Reports". 3. Views report dashboard. 4. Selects report type (Faculty/Lecturer). 5. Sets filters (timeframe, department). 6. System queries data and calculates KPIs. 7. Selects output format (PDF/Excel). 8. System generates formatted report. 9. Downloads report.
Alternative Scenario	Login Failed 1. Invalid credentials entered. 2. System shows error message. 3. Coordinator cannot access Admin Panel. No Data Available 1. Selected filters return no data. 2. System displays "No data available" message. 3. Coordinator adjusts filters and retries. Export Error 1. Report generation fails. 2. System shows export error. 3. Coordinator retries or contacts admin.

Figure 3.1.2.4 Generate Reports &KPIs Sequence Diagram

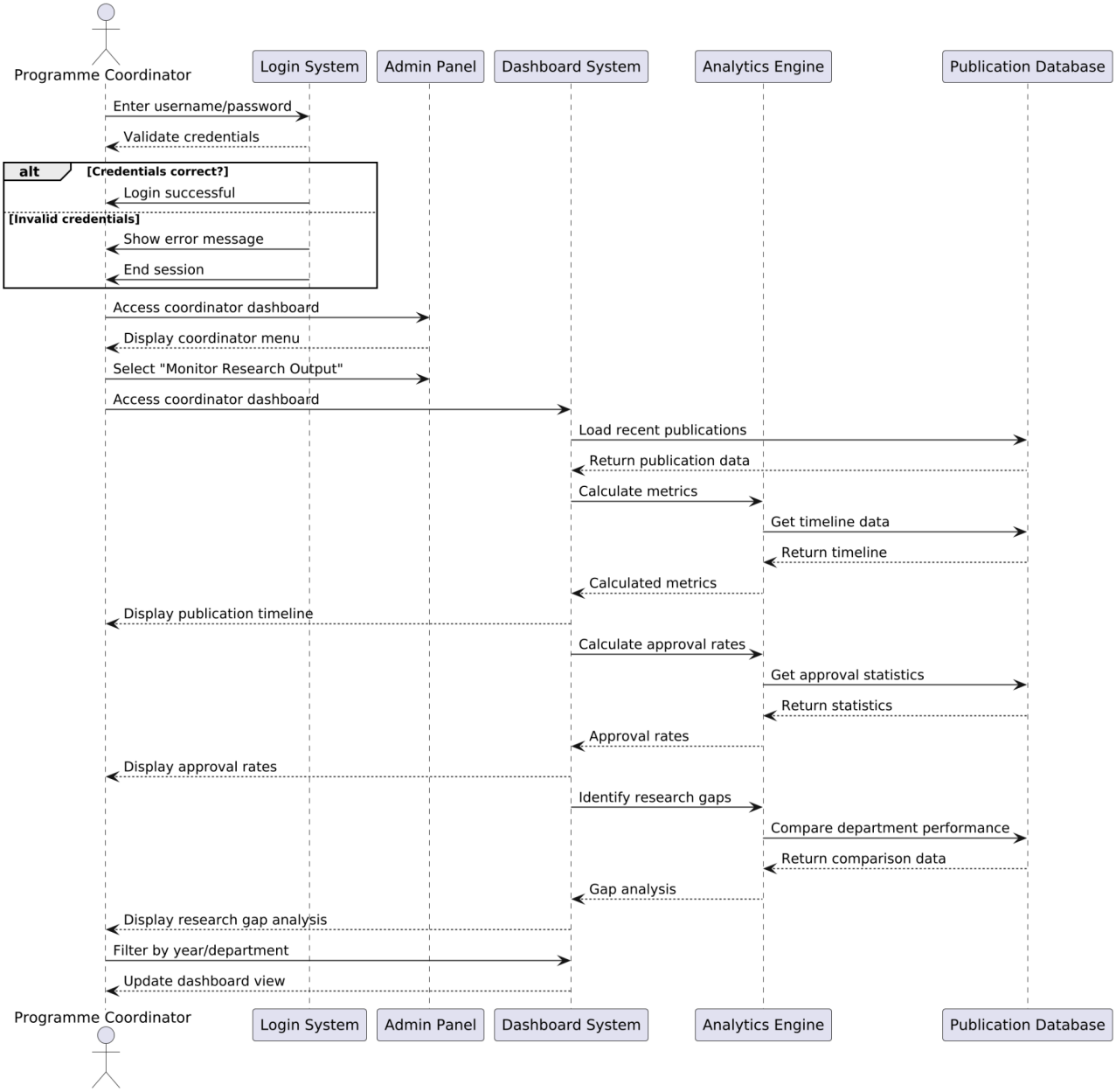


3.1.2.5 Monitor Research Output

Table 3.1.2.5 Monitor Research Output Descriptions

Use Case Name	Monitor Research Output
Description	Coordinator monitors real-time publication metrics, approval rates, research trends, and identifies performance gaps across departments.
Primary Actor	Programme Coordinator
Precondition	1. Coordinator is authenticated and logged in. 2. Admin Panel shows coordinator menu. 3. System has publication data. 4. Analytics engine is operational. 5. Coordinator has dashboard access.
Postcondition	Coordinator has overview of research performance and identified areas for improvement.
Main Success Scenario	1. Coordinator logs in successfully. 2. Accesses Admin Panel, selects "Monitor Research Output". 3. Views monitoring dashboard. 4. System displays publication timeline. 5. System shows approval rates. 6. System analyzes research gaps. 7. Coordinator reviews metrics and trends. 8. Coordinator filters views as needed. 9. Dashboard updates with filtered data.
Alternative Scenario	Login Failed 1. Invalid credentials entered. 2. System shows error message. 3. Coordinator cannot access Admin Panel. Data Loading Issues 1. System encounters data retrieval problems. 2. Dashboard shows partial/error state. 3. Coordinator retries or contacts admin. No Data Period 1. Selected timeframe has no publications. 2. Dashboard shows empty metrics. 3. Coordinator adjusts timeframe.

Figure 3.1.2.5 Monitor Research Output Sequence Diagram



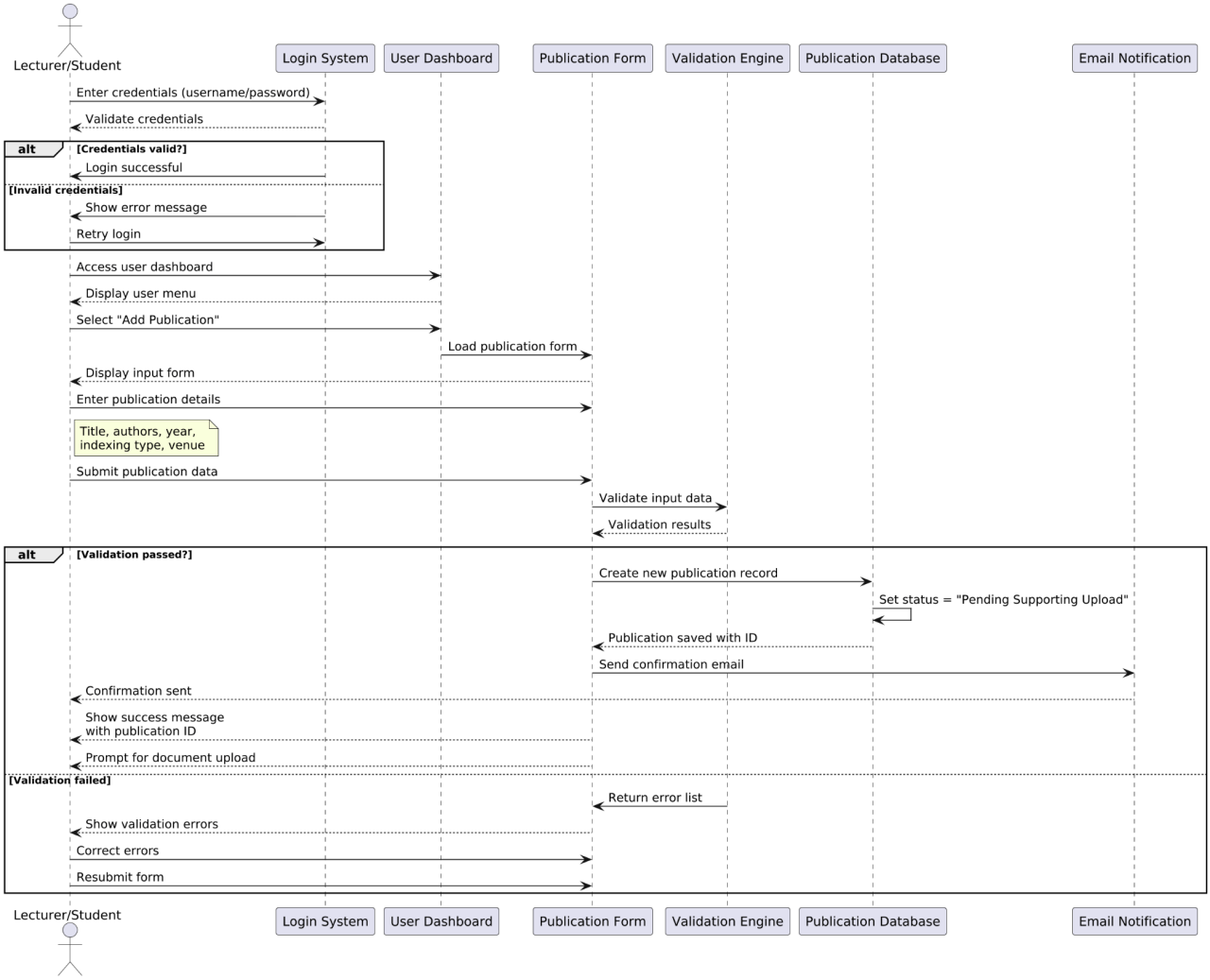
3.1.3 Lecturer/Student when Main Author

3.1.3.1 Add Publication

Table 3.1.3.1 Add Publication Descriptions

Use Case Name	Add Publication
Description	Lecturer or student creates a new publication entry in the system by inputting key publication metadata.
Primary Actor	Main Author (Lecturer/Student)
Precondition	1. Actor is authenticated and logged in. 2. User dashboard is accessible. 3. Actor has necessary publication information ready. 4. System publication module is operational.
Postcondition	Publication record created with "Pending Supporting Upload" status.
Main Success Scenario	1. Actor logs into system. 2. Navigates to "Add Publication" from dashboard. 3. Enters publication details (title, authors, year, indexing type, venue). 4. System validates input data. 5. System creates publication record. 6. System sets status to "Pending Supporting Upload". 7. System displays success message with publication ID. 8. System prompts for document upload.
Alternative Scenario	**Invalid Data** 1. Actor submits incomplete or invalid data. 2. System displays validation errors. 3. Actor corrects errors and resubmits. **Duplicate Entry** 1. Actor attempts to add duplicate publication. 2. System detects potential duplicate. 3. System asks for confirmation or modification. 4. Actor confirms or modifies details.

Figure 3.1.3.1 Add Publication Sequence Diagram

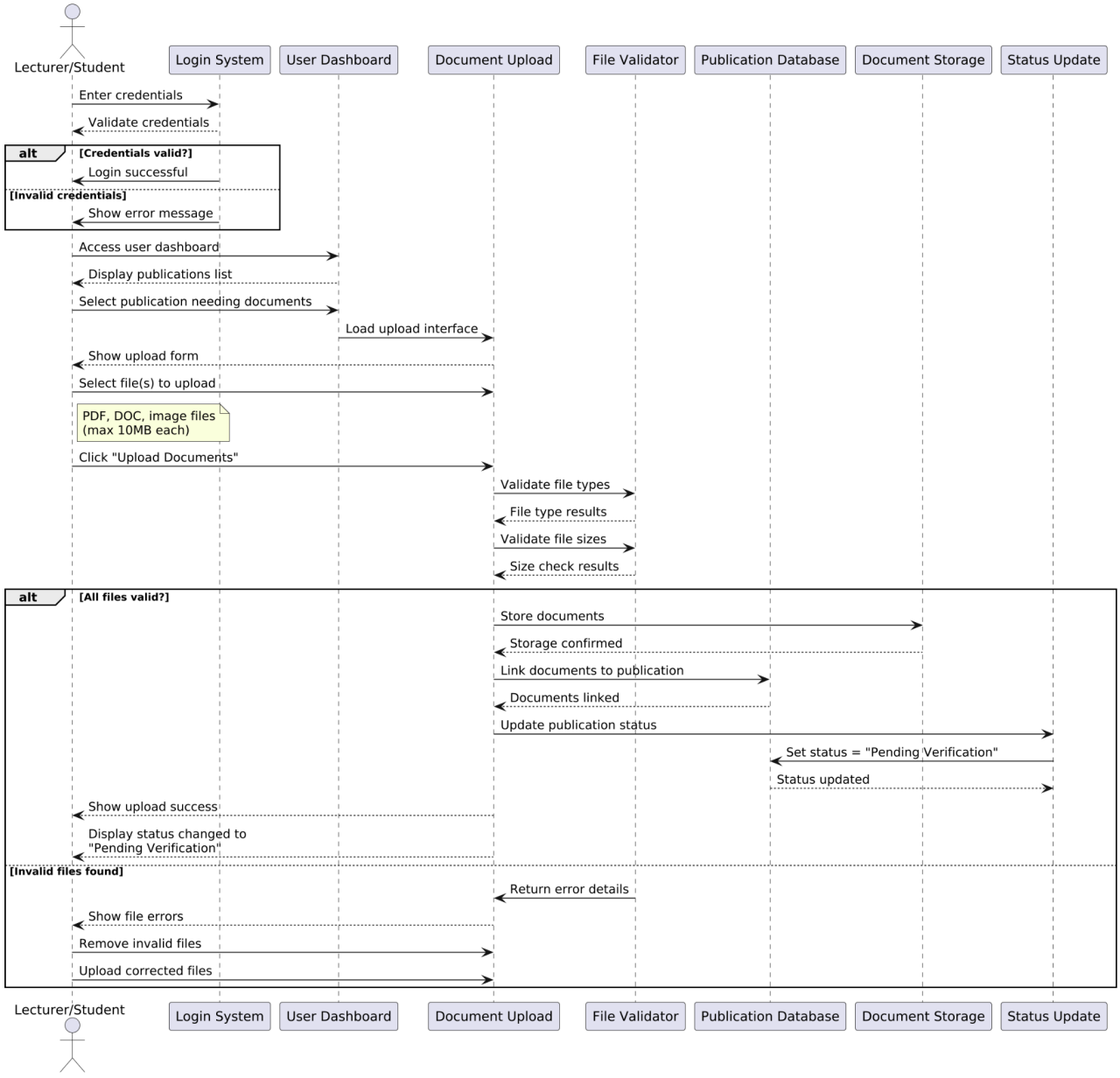


3.1.3.2 Upload Supporting Documents

Table 3.1.3.2 Upload Supporting Documents Descriptions

Use Case Name	Upload Supporting Documents
Description	Actor uploads supporting documents (PDFs, evidence files) to fulfill verification requirements for a publication.
Primary Actor	Main Author (Lecturer/Student)
Precondition	1. Actor is authenticated and logged in. 2. Publication exists with "Pending Supporting Upload" status. 3. Documents are ready for upload. 4. File formats are compatible with system requirements.
Postcondition	Documents attached to publication and status updated to "Pending Verification".
Main Success Scenario	1. Actor selects publication needing documents. 2. System displays upload interface. 3. Actor selects files to upload. 4. System validates file types and sizes. 5. System stores documents in database. 6. System links documents to publication. 7. System updates status to "Pending Verification". 8. System displays upload confirmation.
Alternative Scenario	**Invalid File Type** 1. Actor uploads unsupported file format. 2. System rejects file with error message. 3. Actor converts file to supported format. 4. Actor re-uploads corrected file. **File Size Exceeded** 1. File exceeds maximum size limit. 2. System displays size error. 3. Actor compresses file or selects smaller file. 4. Actor re-uploads corrected file.

Figure 3.1.3.2 Upload Supporting Documents Sequence Diagram

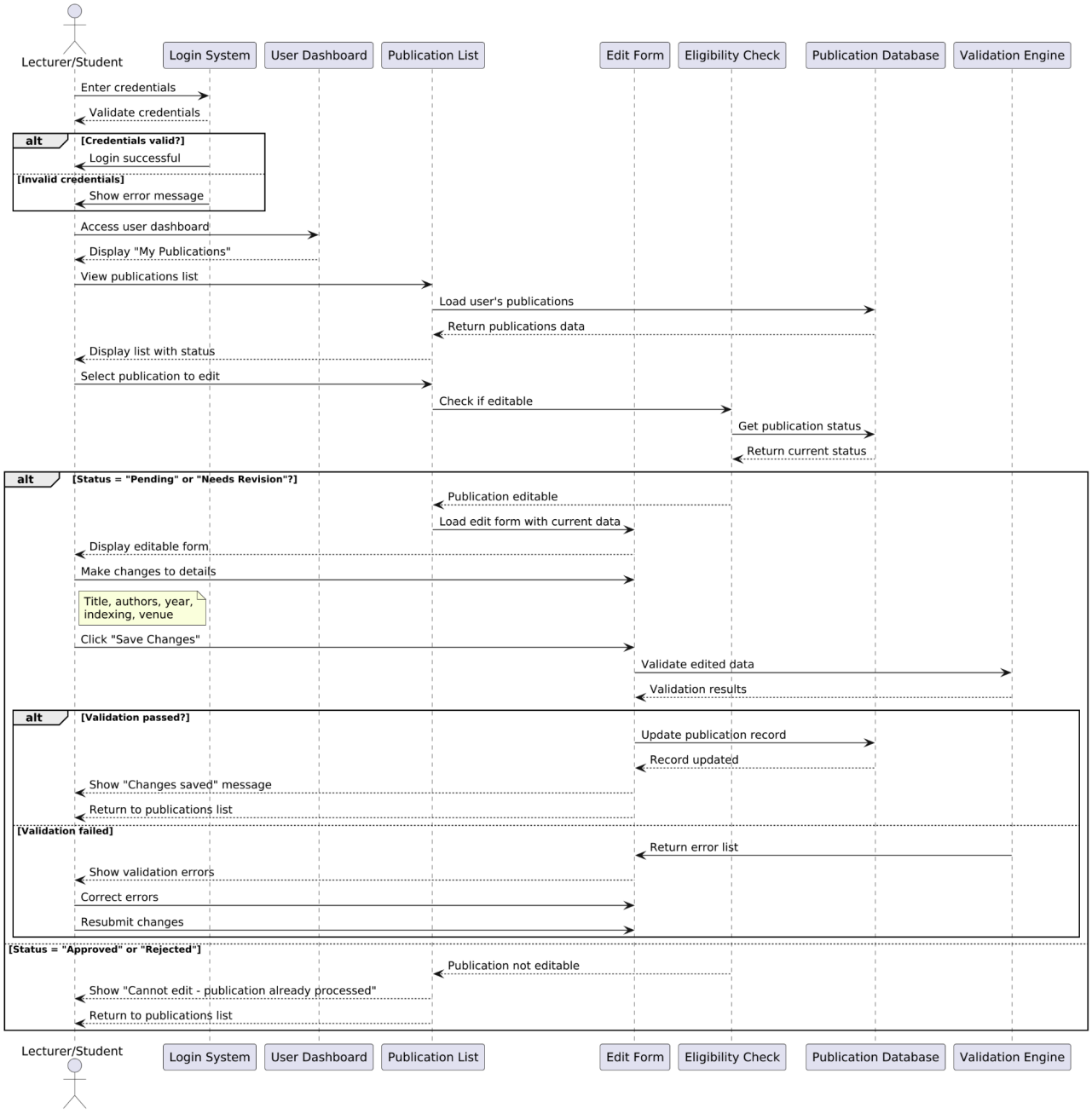


3.1.3.3 Edit Publication Details

Table 3.1.3.3 Edit Publication Details Descriptions

Use Case Name	Edit Publication Details
Description	Actor modifies publication information before it has been verified or approved by the coordinator.
Primary Actor	Main Author (Lecturer/Student)
Precondition	1. Actor is authenticated and logged in. 2. Publication exists and is editable (status: Pending or Needs Revision). 3. Publication has not been approved or rejected. 4. Edit permissions are granted for the publication.
Postcondition	Publication details updated with changes saved.
Main Success Scenario	1. Actor views "My Publications" list. 2. Actor selects editable publication. 3. System loads edit form with current data. 4. Actor modifies details (title, authors, year, etc.). 5. Actor saves changes. 6. System validates updated information. 7. System updates publication record. 8. System displays "Changes saved" message.
Alternative Scenario	**Non-editable Publication** 1. Actor attempts to edit approved/rejected publication. 2. System displays "Cannot edit" message. 3. Actor returns to publications list. **Validation Failure** 1. Edited data fails validation. 2. System displays specific errors. 3. Actor corrects errors and resaves.

Figure 3.1.3.3 Edit Publication Sequence Diagram

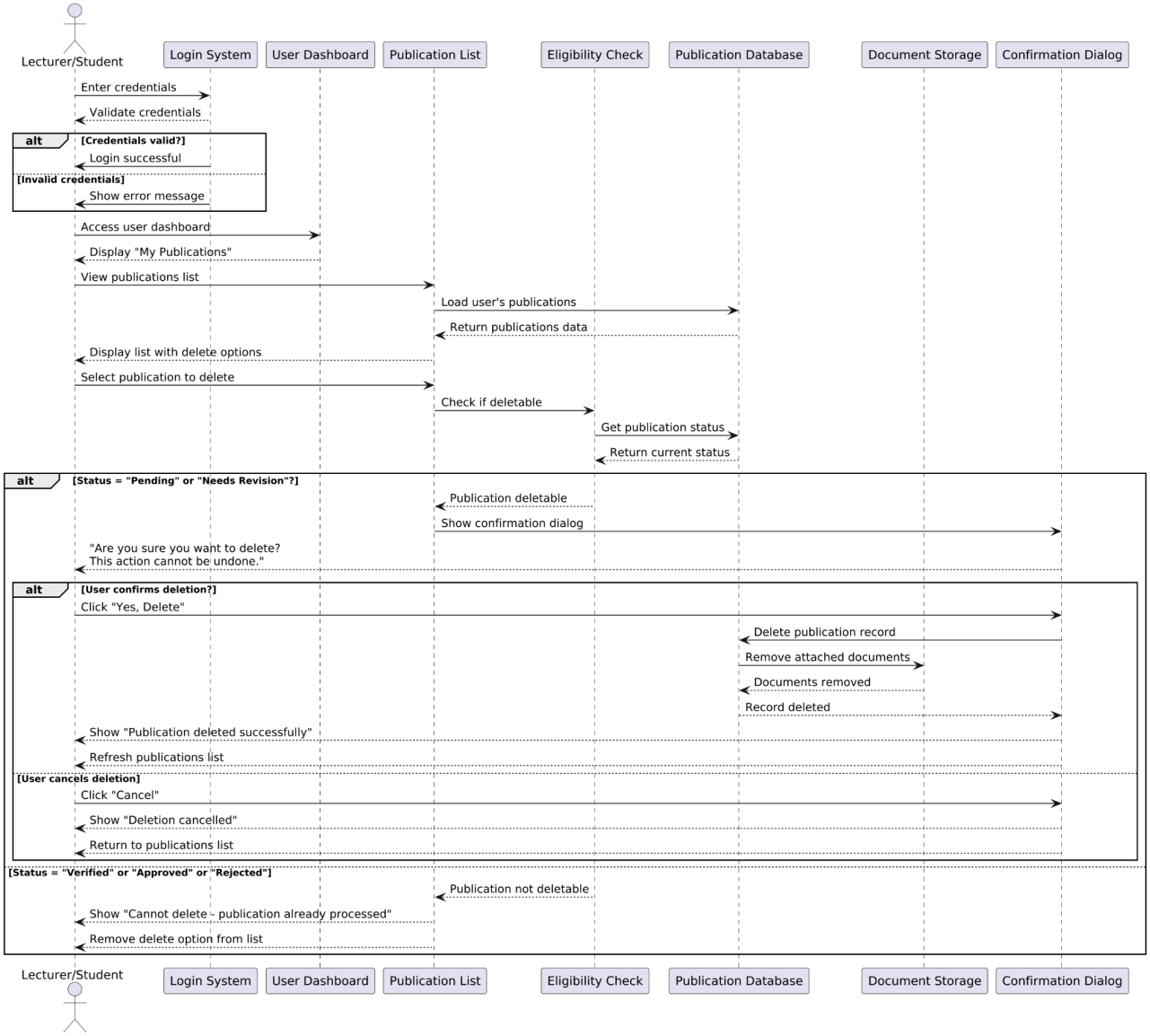


3.1.3.4 Delete Publication

Table 3.1.3.4 Delete Publications Descriptions

Use Case Name	Delete Publication
Description	Actor removes a publication entry that was submitted incorrectly or is no longer relevant.
Primary Actor	Main Author (Lecturer/Student)
Precondition	1. Actor is authenticated and logged in. 2. Publication exists and is deletable (status: Pending or Needs Revision). 3. Publication has not been verified or approved. 4. Delete option is available for the publication.
Postcondition	Publication and associated documents permanently removed from system.
Main Success Scenario	1. Actor views "My Publications" list. 2. Actor selects deletable publication. 3. Actor clicks "Delete" option. 4. System displays confirmation dialog. 5. Actor confirms deletion. 6. System removes publication record. 7. System removes associated documents. 8. System displays deletion success message.
Alternative Scenario	**Non-deletable Publication** 1. Actor attempts to delete verified/approved publication. 2. System removes delete option or shows error. 3. Actor cannot proceed with deletion. **Cancellation** 1. Actor initiates deletion. 2. System shows confirmation dialog. 3. Actor cancels deletion. 4. System returns to publications list.

Figure 3.1.3.4 Delete Publications Sequence Diagram

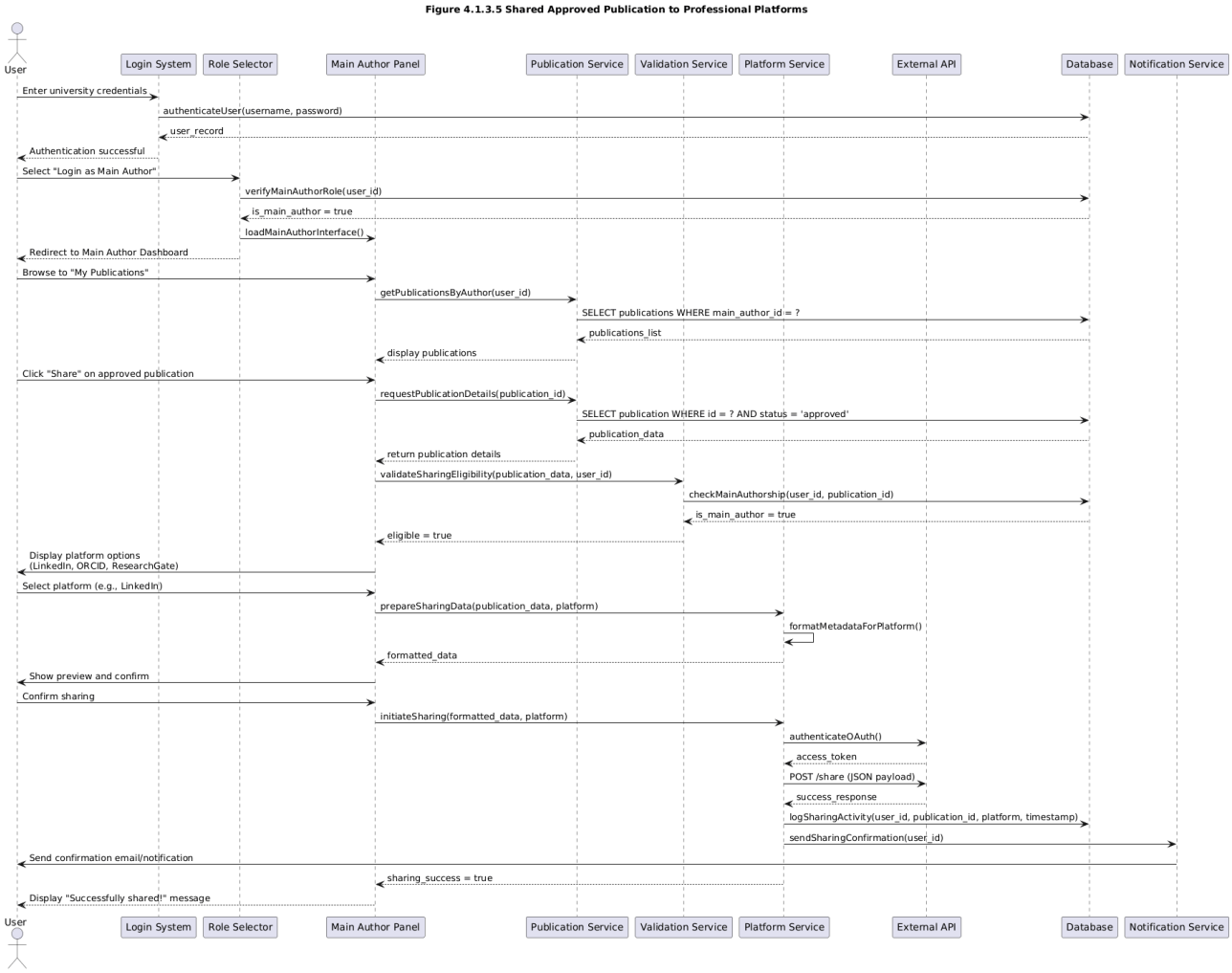


3.1.3.5 Shared Approved publication to Professional Platforms

Table 3.1.3.5 Shared Approved publication to Professional Platforms Descriptions

Use Case Name	Shared Approved publication to Professional Platforms
Description	Actor shares approved publication details directly to external professional platforms (LinkedIn, ResearchGate) to enhance visibility.
Primary Actor	Main Author (Lecturer/Student)
Precondition	1. Actor is authenticated and logged in. 2. Publication exists with "Approved" status. 3. External platform accounts are linked or accessible. 4. Sharing feature is enabled in system.
Postcondition	Publication shared on selected professional platform with formatted metadata.
Main Success Scenario	1. Actor views approved publications list. 2. Actor selects publication to share. 3. Actor clicks "Share Publication" option. 4. System formats publication details for sharing. 5. Actor selects target platform (LinkedIn/ResearchGate). 6. System redirects to platform with pre-filled content. 7. Actor confirms sharing on external platform. 8. System records sharing activity.
Alternative Scenario	**Platform Authentication Required** 1. External platform requires login. 2. System redirects to platform login. 3. Actor logs into platform. 4. Sharing process continues. **Sharing Failed** 1. External platform API returns error. 2. System displays sharing failure message. 3. Actor can retry or cancel sharing.

Figure 3.1.3.5 Shared Approved publication to Professional Platforms Sequence Diagram

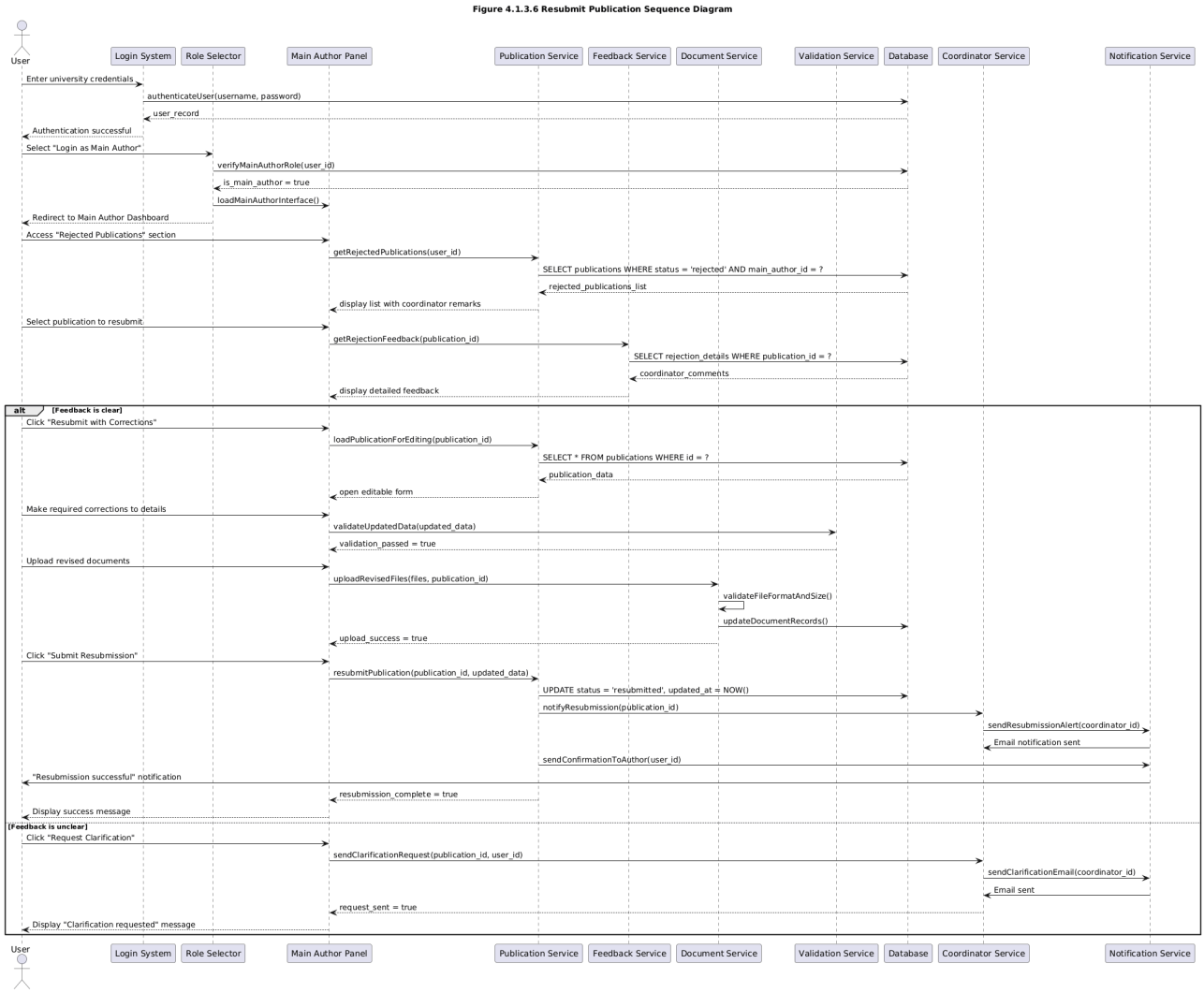


3.1.3.6 Resubmit Publication (If program coordinator rejects publication)

Table 3.1.3.6 Resubmit Publication Descriptions

Use Case Name	Resubmit Publication (If program coordinator rejects publication)
Description	Main author receives rejection notification, reviews feedback from coordinator, makes necessary corrections to publication details and documents, and resubmits for re-verification. The system validates changes, updates status, and notifies coordinator for re-review
Primary Actor	Main Author (Lecturer/Student)
Precondition	1. Main author is logged into the system. 2. Publication has been rejected by Programme Coordinator. 3. Rejection notification has been received.
Postcondition	Publication status is updated to "Resubmitted" and coordinator is notified for re-review
Main Success Scenario	1. Main author accesses "Rejected Publications" section. 2. System displays rejected publications with coordinator remarks. 3. Author reviews feedback and selects publication to resubmit. 4. System opens editable publication form with previous data. 5. Author makes required corrections to publication details. 6. Author uploads revised supporting documents. 7. System validates updated data and files. 8. System updates publication status to "Resubmitted." 9. Coordinator receives notification of resubmission. 10. Author receives confirmation of successful resubmission
Alternative Scenario	A1: Feedback is Unclear 1. Author finds coordinator feedback unclear. 2. Author requests clarification through system. 3. Coordinator receives clarification request. 4. Author receives "Clarification requested" message. A2: Invalid Files Uploaded 1. Author uploads files with invalid format/size. 2. System rejects files and displays error message. 3. Author corrects file issues and re-uploads.

Figure 3.1.3.6 Resubmit Publication Sequence Diagram



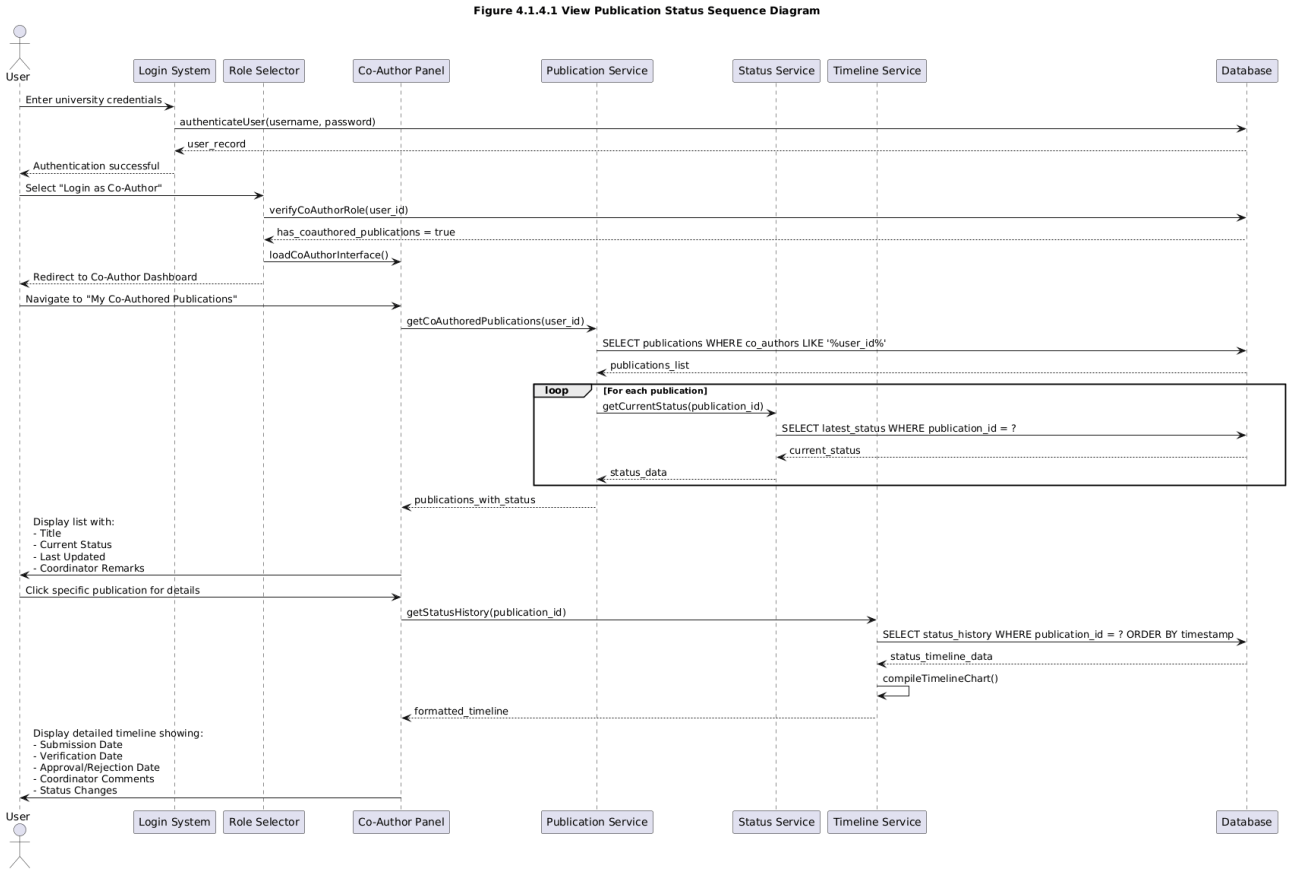
3.1.4 Lecturer/Student when Co-Author

3.1.4.1 View Publication Status

Table 3.1.4.1 View Publication Status Descriptions

Use Case Name	View Publication Status
Description	Co-author logs into system, navigates to co-authored publications section, views current status of all publications they contributed to, and can drill down to see detailed status history and timeline for specific publications
Primary Actor	Co-Author (Lecturer/Student)
Precondition	1. Co-author has valid system credentials. 2. Co-author is listed as author on at least one publication. 3. Publications exist in the system
Postcondition	Co-author has viewed publication status information; no changes made to system data
Main Success Scenario	1. Co-author logs into system using credentials. 2. System authenticates user and creates session. 3. Co-author navigates to "My Co-Authored Publications." 4. System queries database for publications where user is co-author. 5. Database returns publication list with basic information. 6. System retrieves current status for each publication. 7. System displays list with: Title, Current Status, Last Updated Date, Coordinator Remarks. 8. Co-author selects specific publication for detailed view. 9. System retrieves complete status history timeline. 10. System displays detailed status history chart with all status changes, timestamps, and comments.
Alternative Scenario	A1: No Co-Authored Publications 1. System queries database for co-authored publications. 2. Database returns empty result set. 3. System displays message: "No co-authored publications found." 4. Co-author returns to main dashboard. A2: Authentication Failure 1. Co-author enters invalid credentials. 2. System rejects login attempt. 3. System displays "Invalid username/password" error. 4. Co-author retries login or uses password recovery.

Figure 3.1.4.1 View Publication Status Sequence Diagram

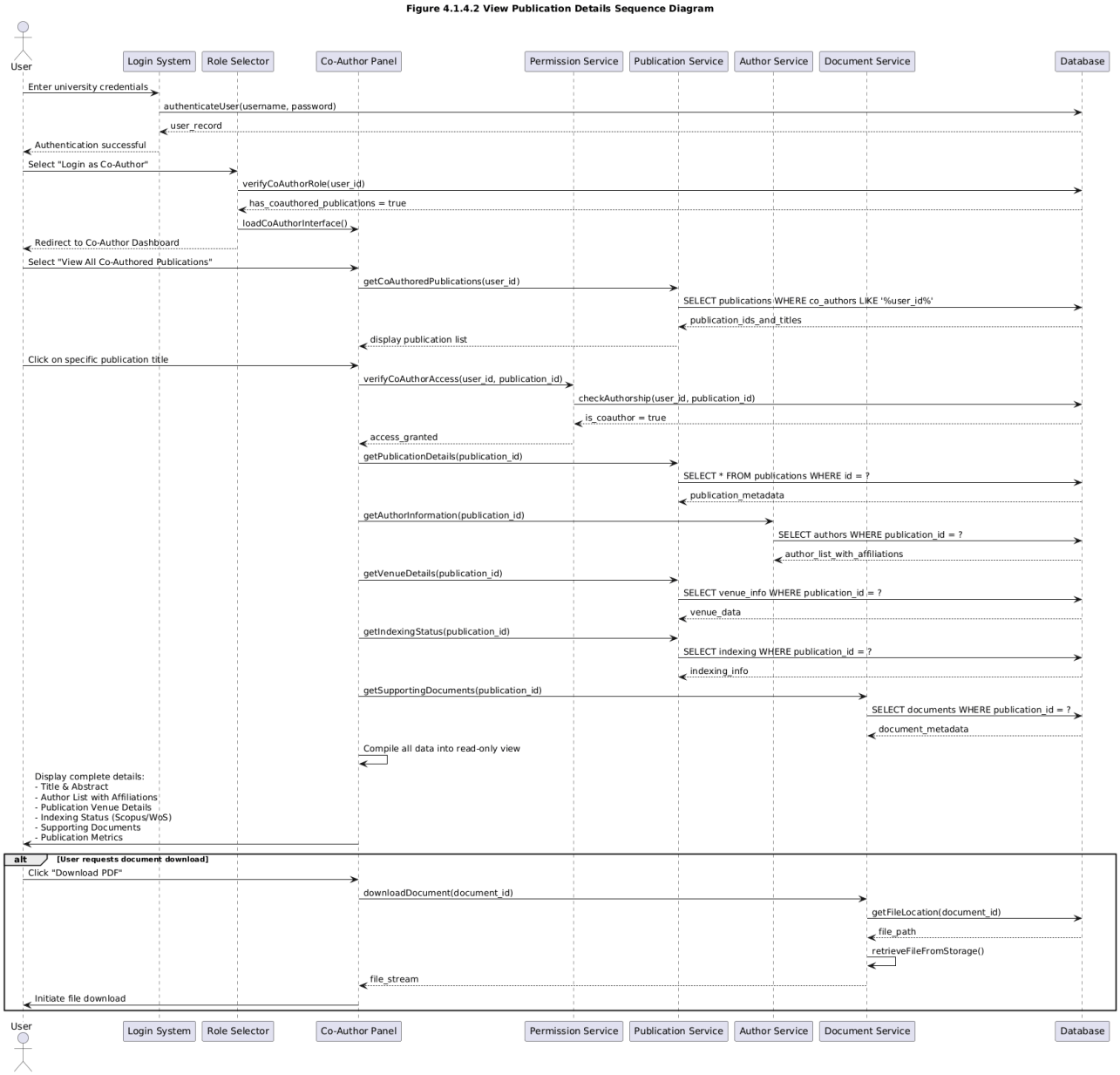


3.1.4.2 View Publication Details

Table 3.1.4.2 View Publication Details Descriptions

Use Case Name	View Publication Details
Description	Co-author accesses complete publication details in read-only mode, including metadata, author information, publication venue details, indexing status, and supporting documents. Can download files but cannot edit any information
Primary Actor	Co-Author (Lecturer/Student)
Precondition	1. Co-author is authenticated and logged in. 2. Publication exists in system. 3. Co-author is listed as author on the publication. 4. Publication details are stored in database.
Postcondition	Co-author has viewed publication details; system logs the access activity.
Main Success Scenario	1. Co-author selects "View All Co-Authored Publications." 2. System retrieves list of publications from database. 3. System displays publication titles in list format. 4. Co-author clicks on specific publication title. 5. System verifies user is co-author of selected publication. 6. Permission granted - system retrieves full publication details. 7. System fetches: metadata, author list, venue information, indexing status. 8. System retrieves supporting document information. 9. System displays complete read-only view with all details. 10. Co-author downloads PDF document (optional). 11. System logs download activity.
Alternative Scenario	A1: Permission Denied 1. Co-author attempts to view publication they are not author on. 2. System verifies authorship and denies access. 3. System displays "Access Denied - Not Author" message. 4. User returned to publication list. A2: Document Unavailable 1. Co-author requests document download. 2. System finds document file missing or corrupted. 3. System displays "Document Unavailable" message. 4. User continues viewing other details.

Figure 3.1.4.2 View Publication Details Sequence Diagram



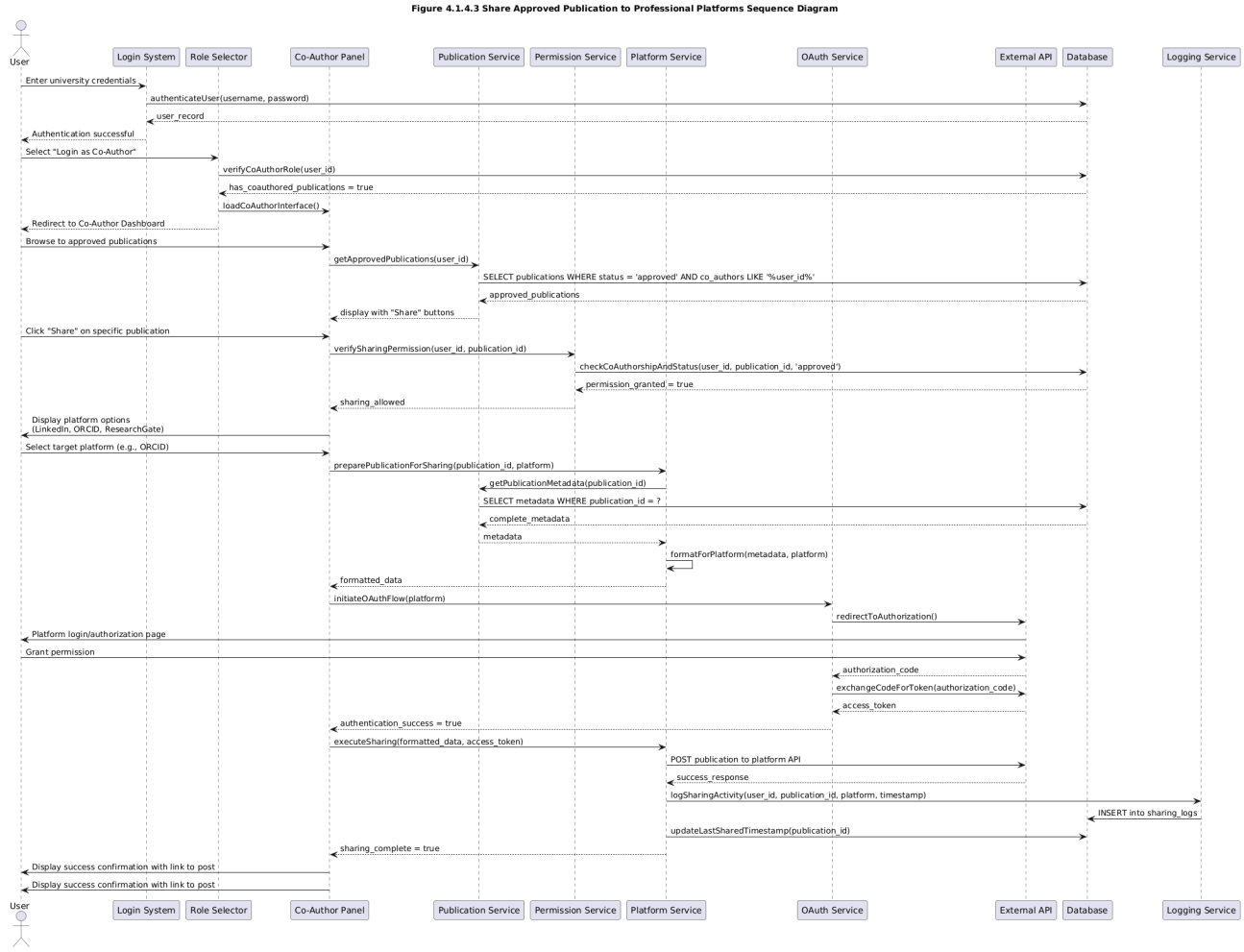
3.1.4.3 Share Approved Publication to Professional Platforms

Table 3.1.4.3 Share Approved Publication to Professional Platforms Descriptions

Use Case Name	Share Approved Publication to Professional Platforms
Description	Co-author shares an approved publication to external professional platforms (LinkedIn, ORCID, ResearchGate). System verifies publication approval status, formats metadata, handles OAuth authorization, posts to external API, and logs sharing activity
Primary Actor	Co-Author (Lecturer/Student)
Precondition	1. Co-author is logged into system. 2. Publication exists and has "Approved" status. 3. Co-author is listed as author on publication. 4. External platform APIs are accessible. 5. User has authorized platform access previously (or will during process)
Postcondition	Publication is posted to external platform; sharing activity is logged in system database.
Main Success Scenario	1. Co-author browses to approved publications section. 2. System displays approved publications with "Share" buttons. 3. Co-author selects publication and clicks "Share." 4. System verifies publication is approved and user is co-author. 5. System displays available platform options (LinkedIn, ORCID, etc.). 6. Co-author selects target platform (e.g., LinkedIn). 7. System formats publication metadata for external platform. 8. System requests OAuth authorization from user. 9. User authorizes platform access. 10. System posts publication to external platform API. 11. External API confirms successful post. 12. System logs sharing activity in database. 13. System displays success confirmation with link to post.
Alternative Scenario	A1: Publication Not Approved 1. Co-author attempts to share unapproved publication. 2. System verifies status and rejects request. 3. System displays "Cannot share - Publication not approved." 4. User returned to publication list. A2: Multi-Platform Sharing 1. After successful sharing, user selects "Share to another platform." 2. System shows remaining platform options.

	3. Process repeats for additional platforms.

Figure 3.1.4.3 Share Approved Publication to Professional Platforms Sequence Diagram

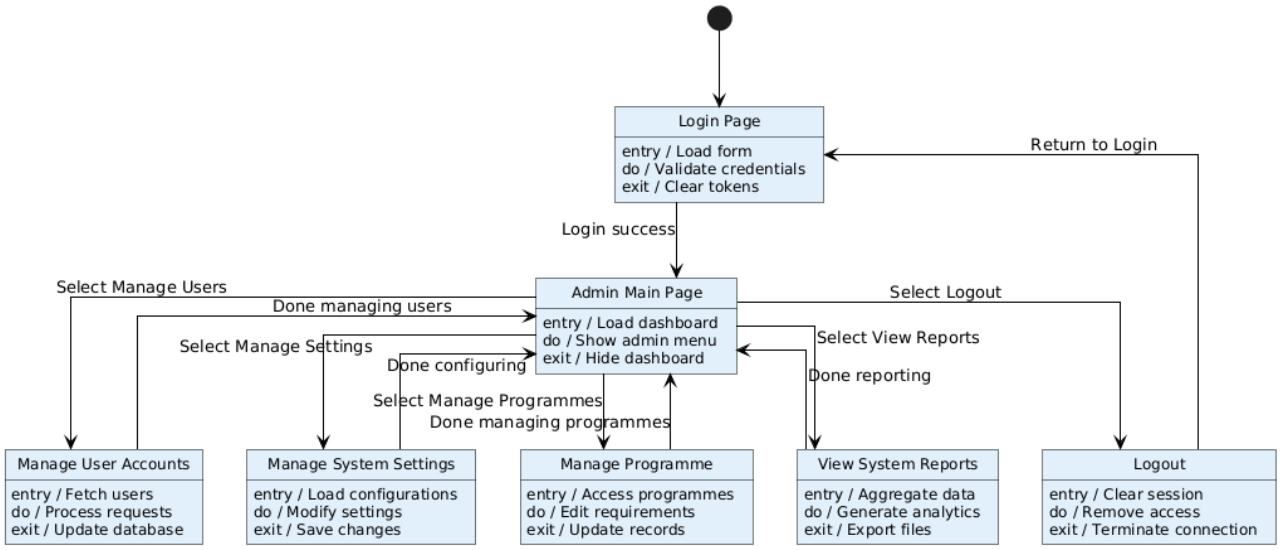


3.2 State Diagram

3.2.1 Admin

The state diagram for Figure 4.2.1 illustrates the complete administrative workflow, beginning with authentication at the Login Page before transitioning to the central Admin Main Page dashboard, from which the admin can navigate to four core functional states: Manage User Accounts for processing user requests, Manage System Settings for configuring system parameters, Manage Programme for editing academic requirements, and View System Reports for generating and exporting analytics each state completes its specific entry, processing, and exit actions before returning to the main dashboard, with the entire session concluding securely through the Logout state which terminates access and returns the user to the login interface.

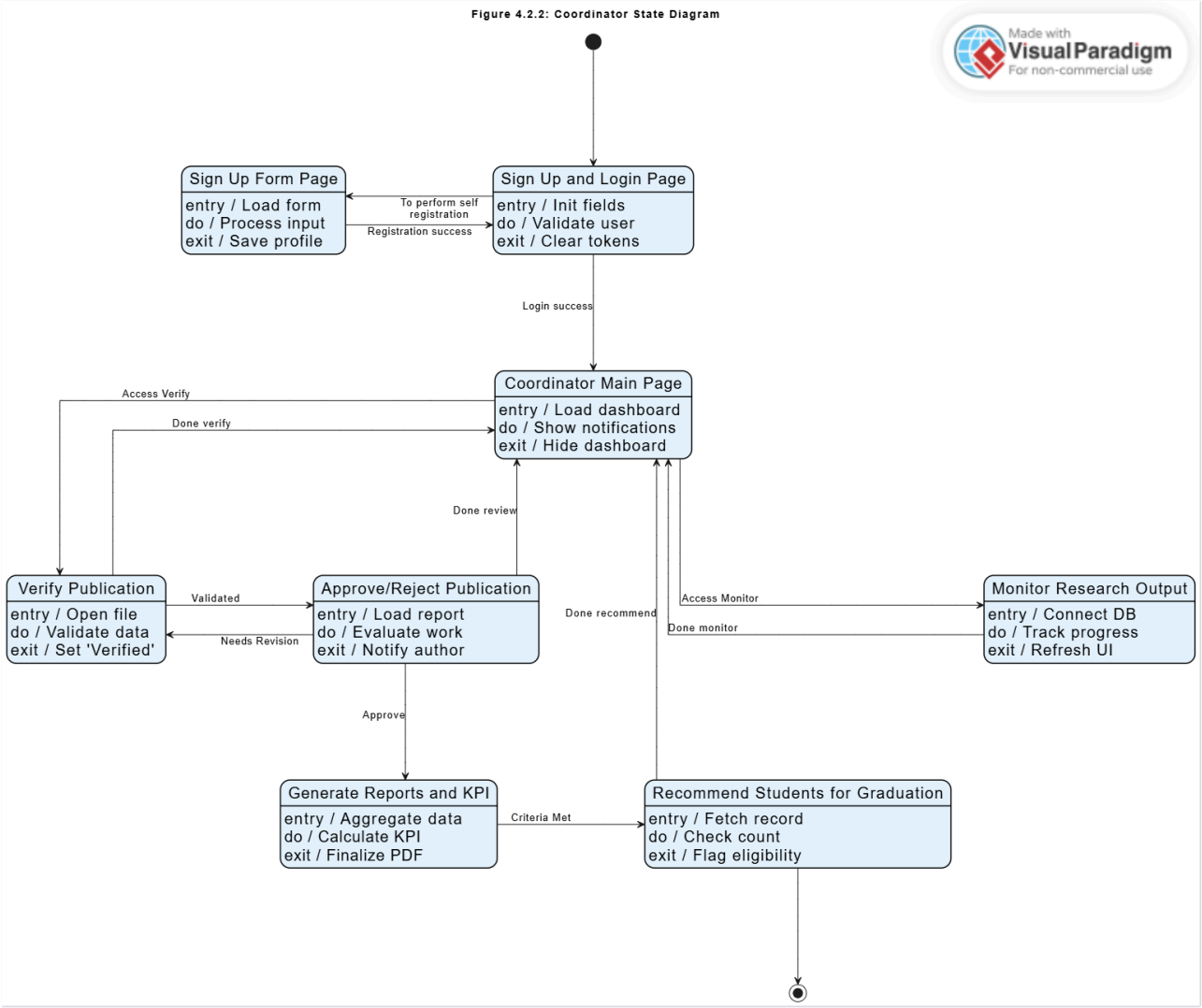
Figure 4.2.1: Admin State Diagram



3.2.2 Coordinator

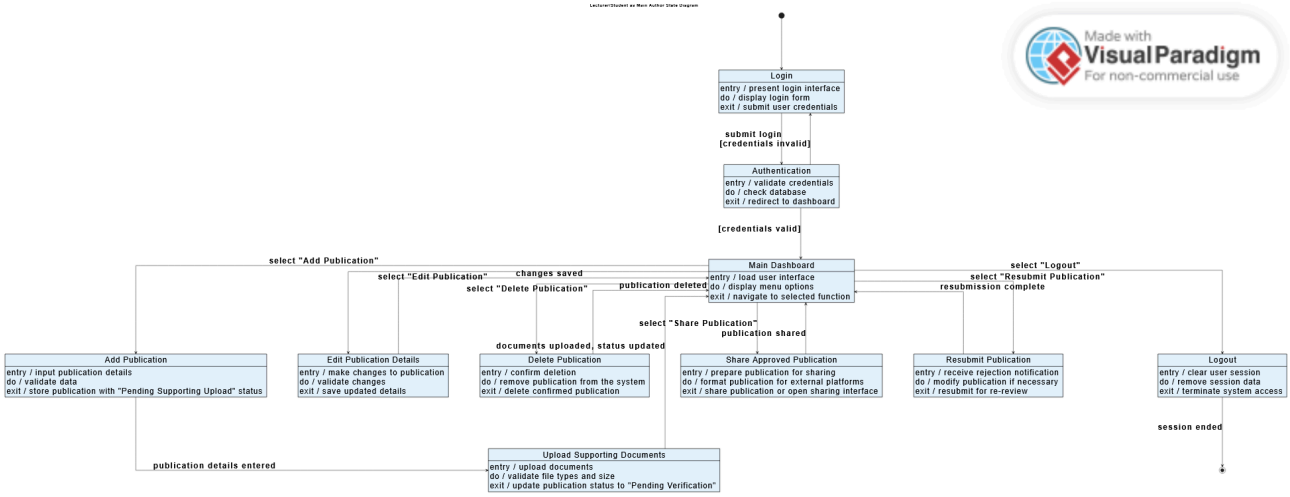
The state diagram for Figure 4.2.2 illustrates the operational lifecycle of the Coordinator within the publication and research tracking system. It begins with an authentication phase where users navigate between a Sign Up Form and the Login Page to establish secure access. Once authenticated, the Coordinator Main Page acts as a central hub, allowing the user to transition into specific functional modules such as Verify Publication, Approve/Reject Publication, or Monitor Research Output. Each functional state contains detailed internal activities including entry initialization, do ongoing tasks, and exit finalization to ensure data integrity. The flow is highly iterative; for instance, a publication can be sent back from the approval stage to verification if it needs revision, creating a bidirectional mapping between states. Finally, the process culminates in Generate Reports and Recommend Students for Graduation once specific criteria are met, with each module providing a return path back to the main dashboard upon completion.

Figure 4.2.2: Coordinator State Diagram



3.2.3 Lecturer/Student as Main Author

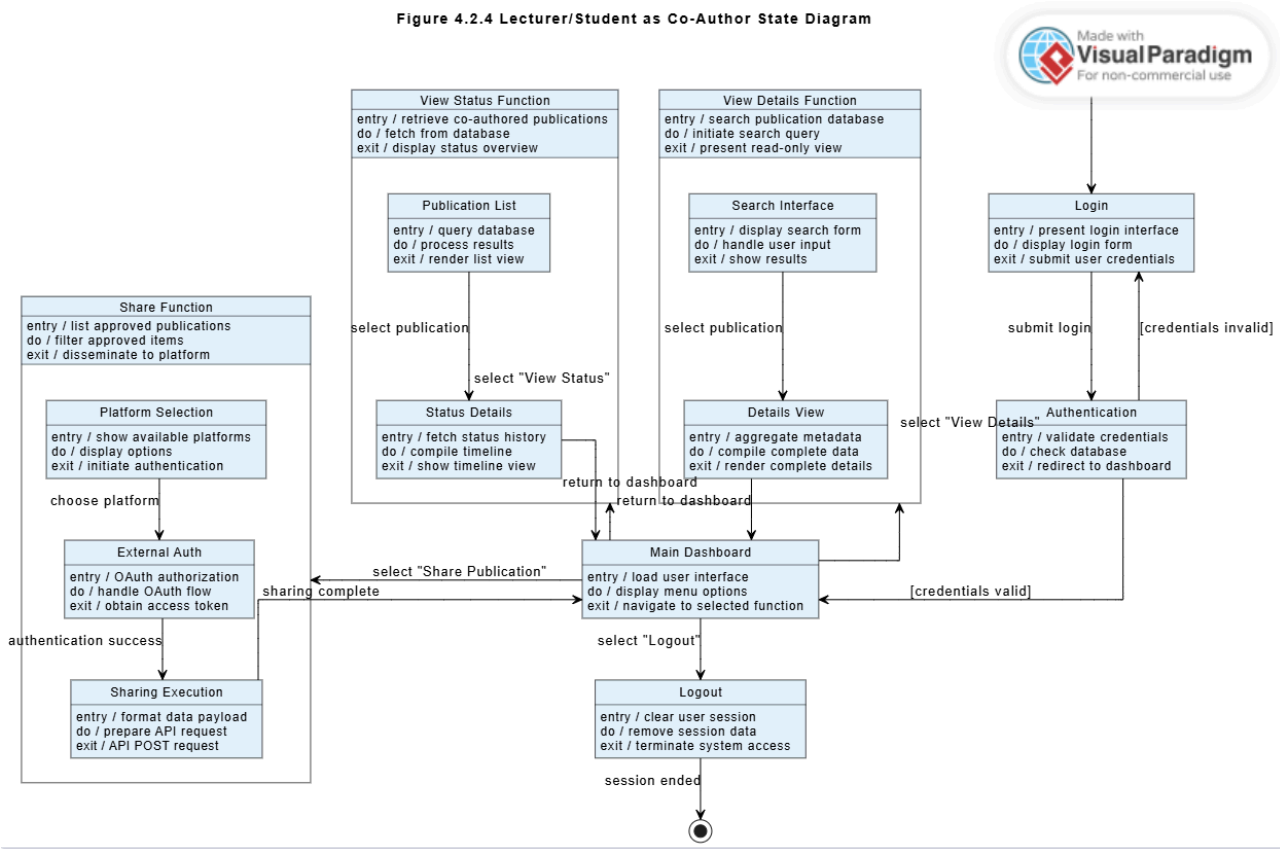
The state diagram for Figure 4.2.3 illustrates the operational lifecycle of the Lecturer/Student as Main Author for publication management workflow that begins with adding a publication, progresses through editing details and a mandatory validation phase for supporting documents, and culminates in sharing the approved publication. The process enforces a critical checkpoint where unclear or missing documents must be resolved, updating the status to "Pending Verification" before distribution can proceed. Integrated system management functions, such as session clearing and access monitoring, ensure security and traceability throughout, while a separate "Upgrade Supporting Documents" path allows for post-publication updates, reflecting a controlled yet adaptable lifecycle for publishing content.



3.2.4 Lecturer/Student as Co-Author

The state diagram for Figure 4.2.4 illustrates the operational workflow for a Lecturer or Student acting as a co-author within the publication management system. It begins with an authentication phase where the user logs in and is redirected to the main dashboard upon successful validation. From there, the user can navigate into core functional modules such as View Status, View Details, or Share Publication, each containing internal substates that handle specific tasks like retrieving publication lists, displaying detailed metadata, or executing external platform sharing via OAuth. The flow supports iterative navigation, allowing the user to return to the dashboard after completing any function, and concludes with a logout process that securely terminates the session.

Figure 4.2.4 Lecturer/Student as Co-Author State Diagram

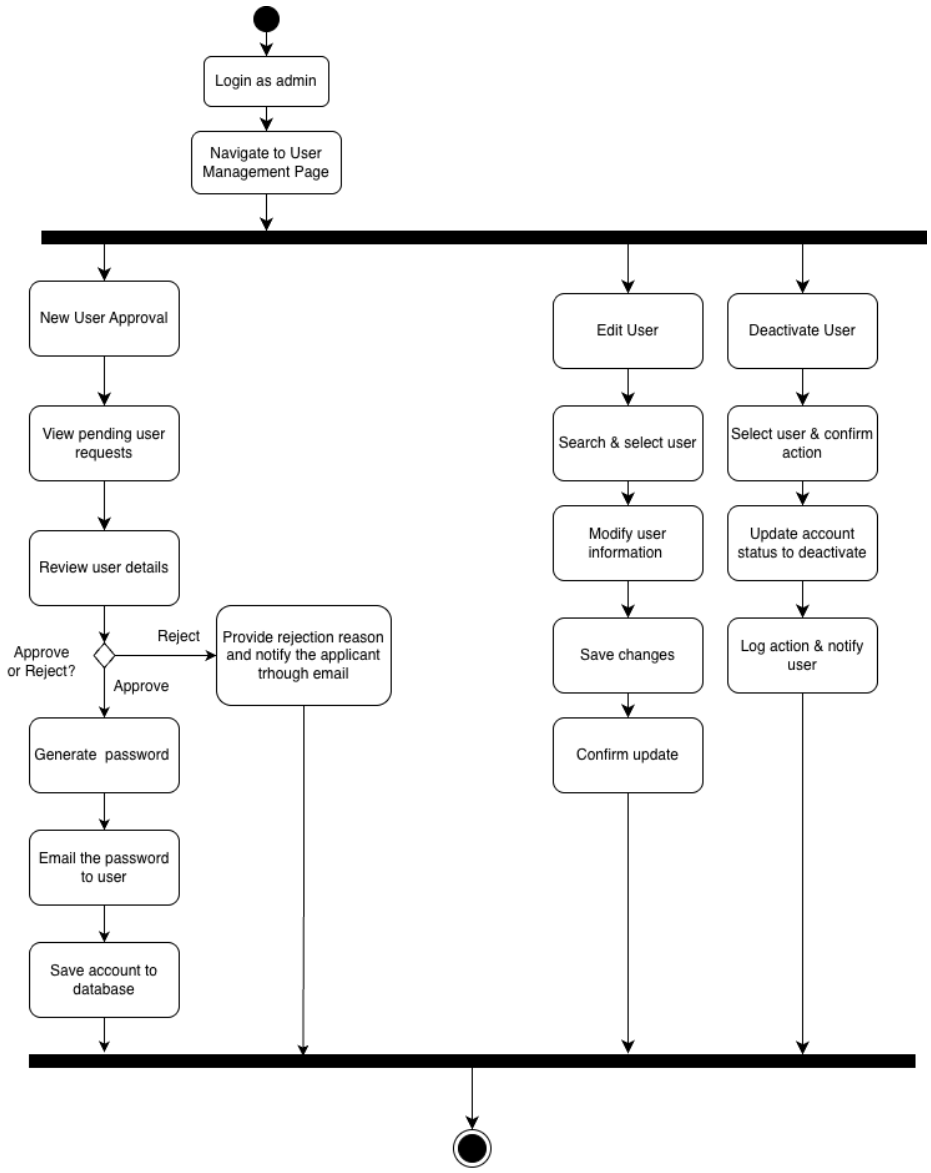


3. 3 Activity Diagram

3.3.1 Actor 1 : Admin

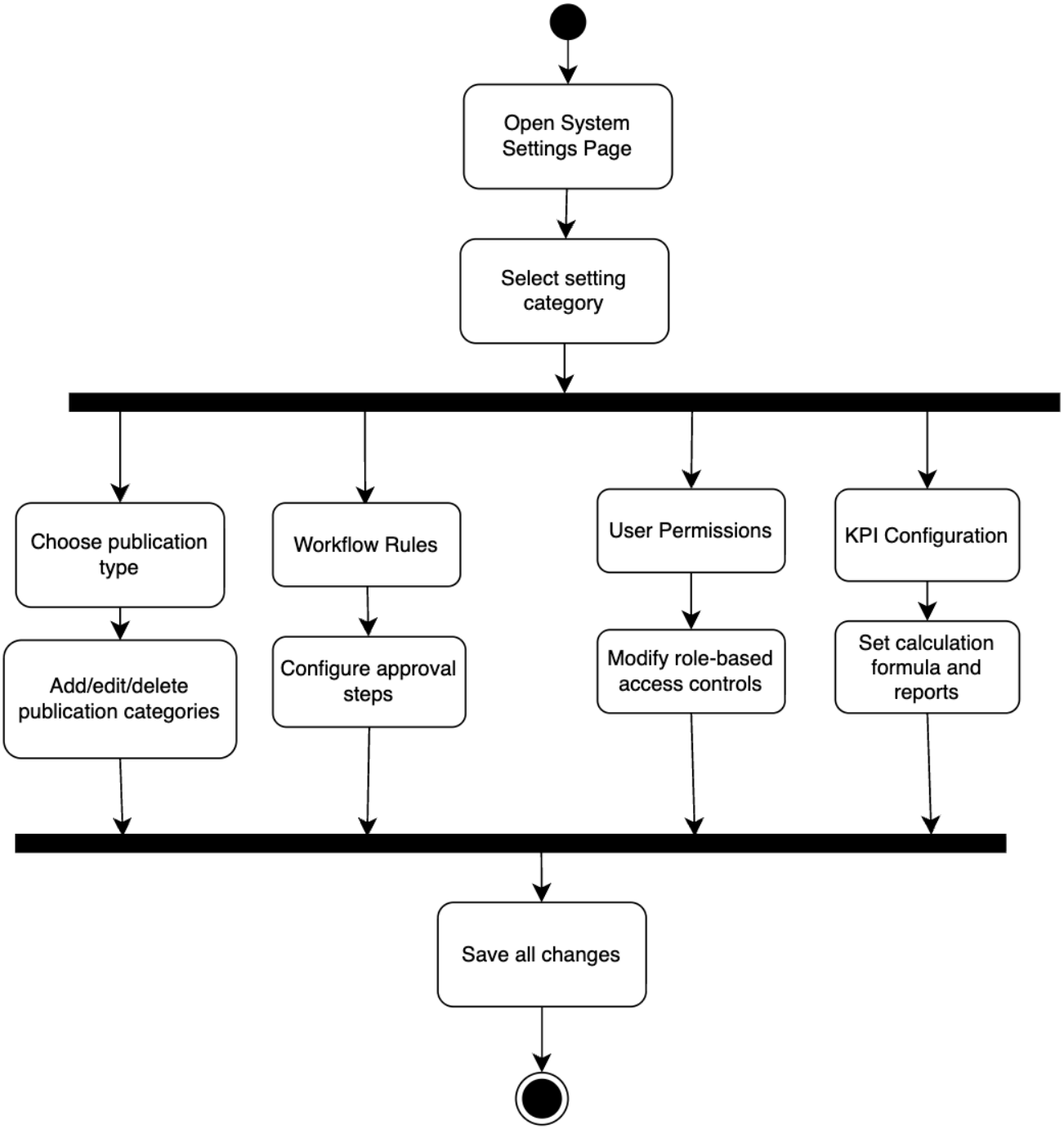
3.3.1.1 Manage Users Accounts

The Manage User Account function allows the administrator to control all user access within the system. The administrator can approve new user accounts by their request, generating a temporary password, and emailing it to the user before saving the account. For existing users, the administrator can edit account information by searching for the user, making changes, and saving the updates. The administrator can also deactivate accounts by selecting a user, confirming the action, and updating their status while notifying them. This ensures secure and accurate user management across the system.



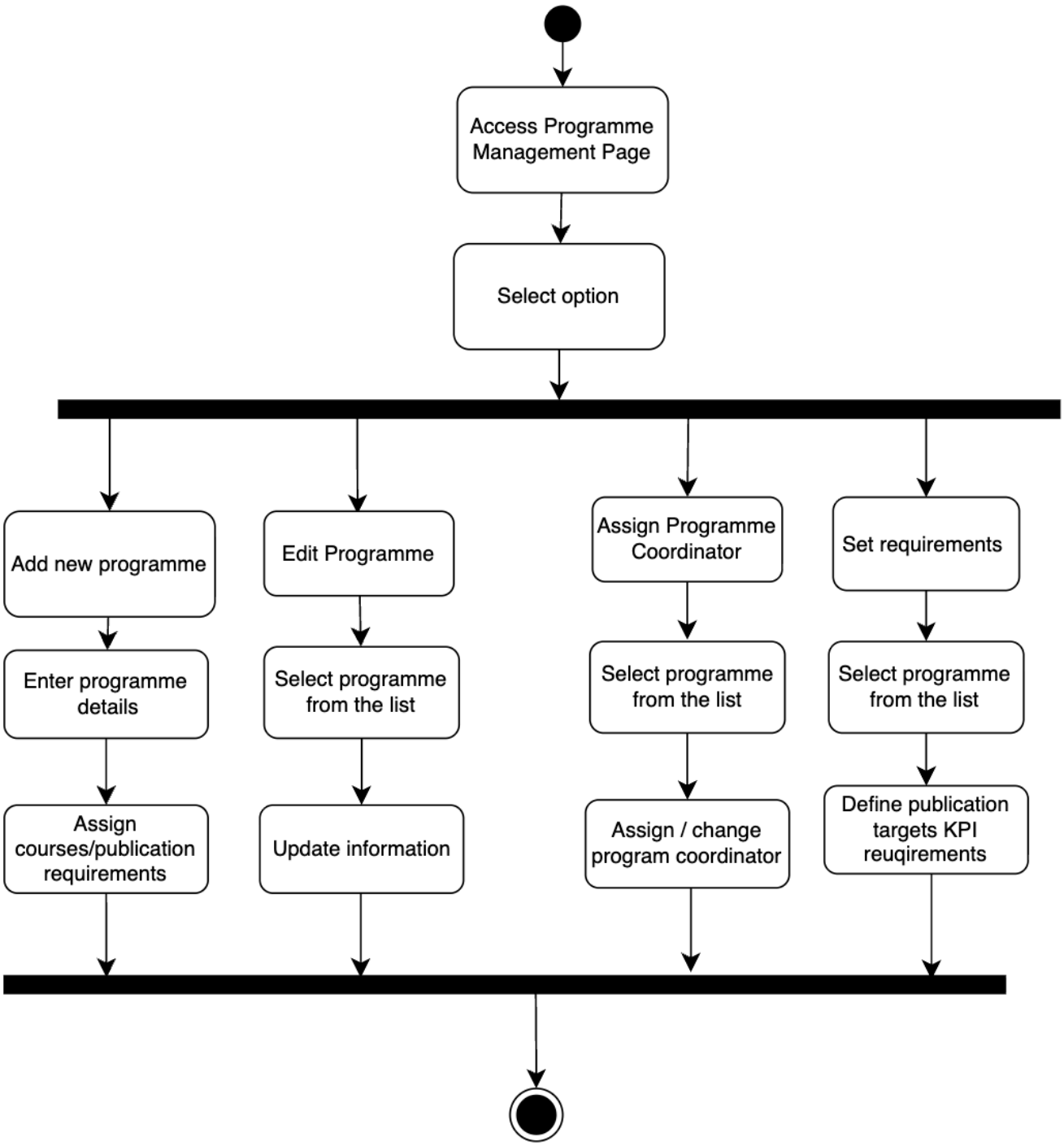
3.3.1.2 Manage System Settings

The administrator oversees the system's settings to make the platform work with the university's research policies. Utilizing the settings panel, the admin is able to determine the types of publications, workflow processes for approvals, and formulas used in KPIs for performance reporting. Besides the functions described earlier, the administrator may set user role permissions along with other operational parameters. All the settings modifications made within the settings menu are saved and applied immediately, allowing the system to keep up with current academic needs.



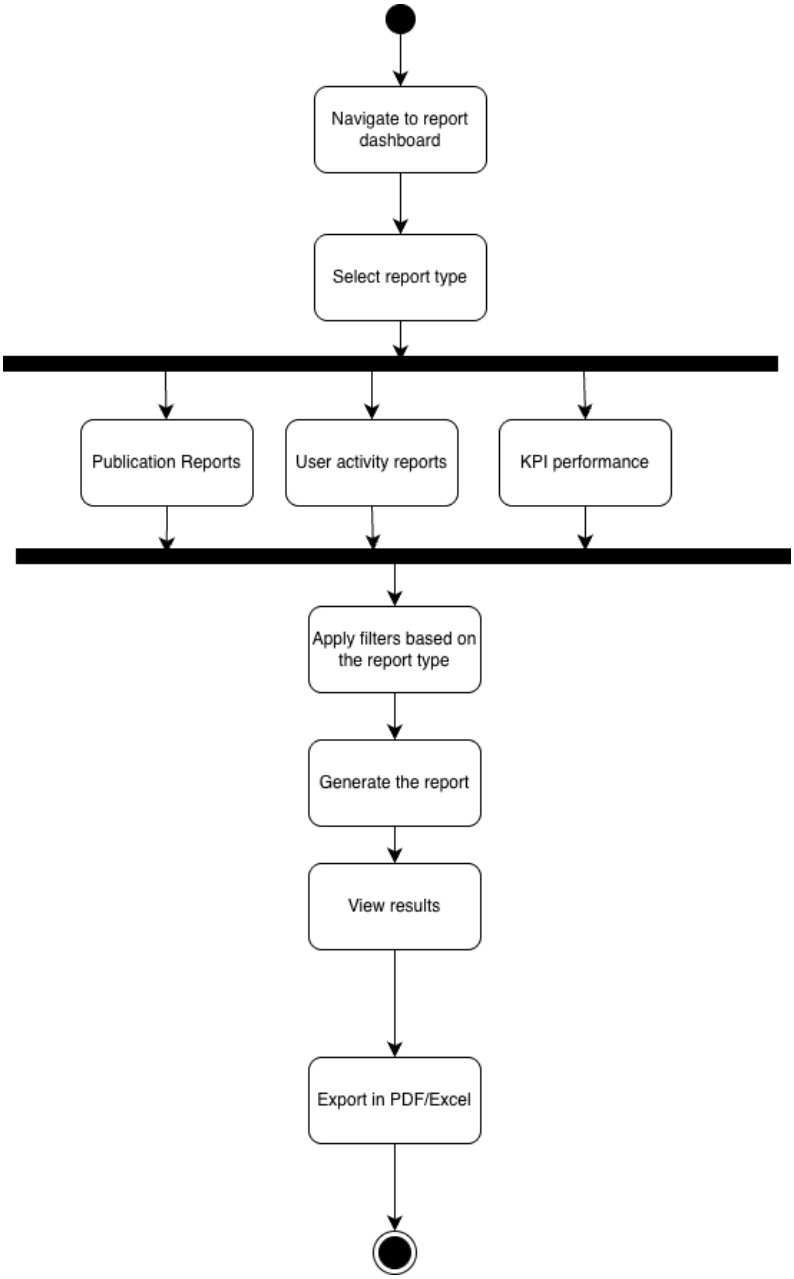
3.3.1.3 Manage Programme

The administrator manages all the academic programmes available in the system. With this functionality, the admin can add new programs and provide program details like the name, code, faculty, and level of study. The administrator may also update existing information on programs to maintain accuracy when the structures or even the names of the programs change. In the event that a program is no longer offered, the admin may remove or deactivate a program to prevent future enrollment. All amendments in the program management section are recorded by the system and used to update the list of programs that will be used in the publication record, user profile, and reporting feature.



3.3.1.4 View System Reports

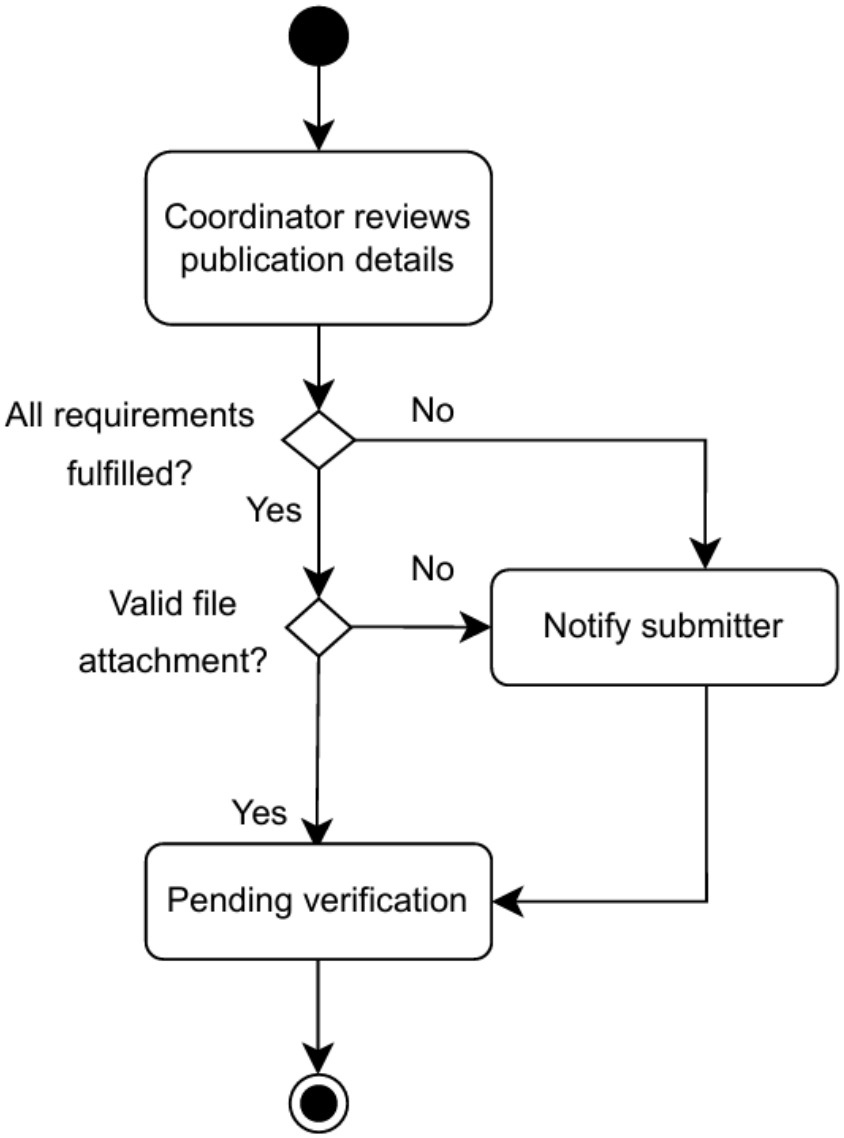
The administrator can view and generate comprehensive system reports that summarize research publication activities across departments and users. Using the report module, the admin selects the desired parameters, such as publication type, author, programme, or department. The system then generates detailed analytics, charts, and summary data that can be exported in formats such as PDF or Excel. These reports help support university assessments, performance evaluations, and strategic planning.



3.3.2 Actor 2 : Programme Coordinator

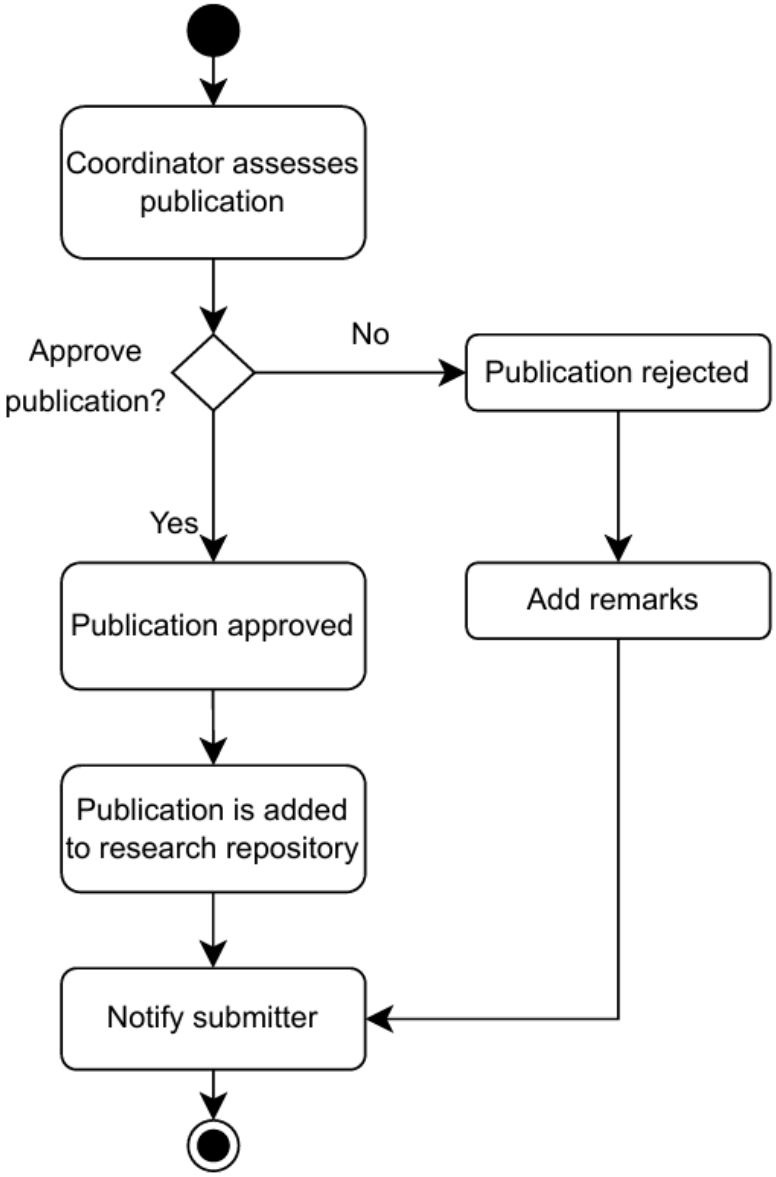
3.3.2.1 Verify Publication

The coordinator is responsible for checking the research paper published in the system for accuracy and publication details before approval. This process covers verifying the title, authors, DOI, supporting references, and evidence; it has to meet the university's criteria to ensure that only accurate and complete research papers are available in the system.



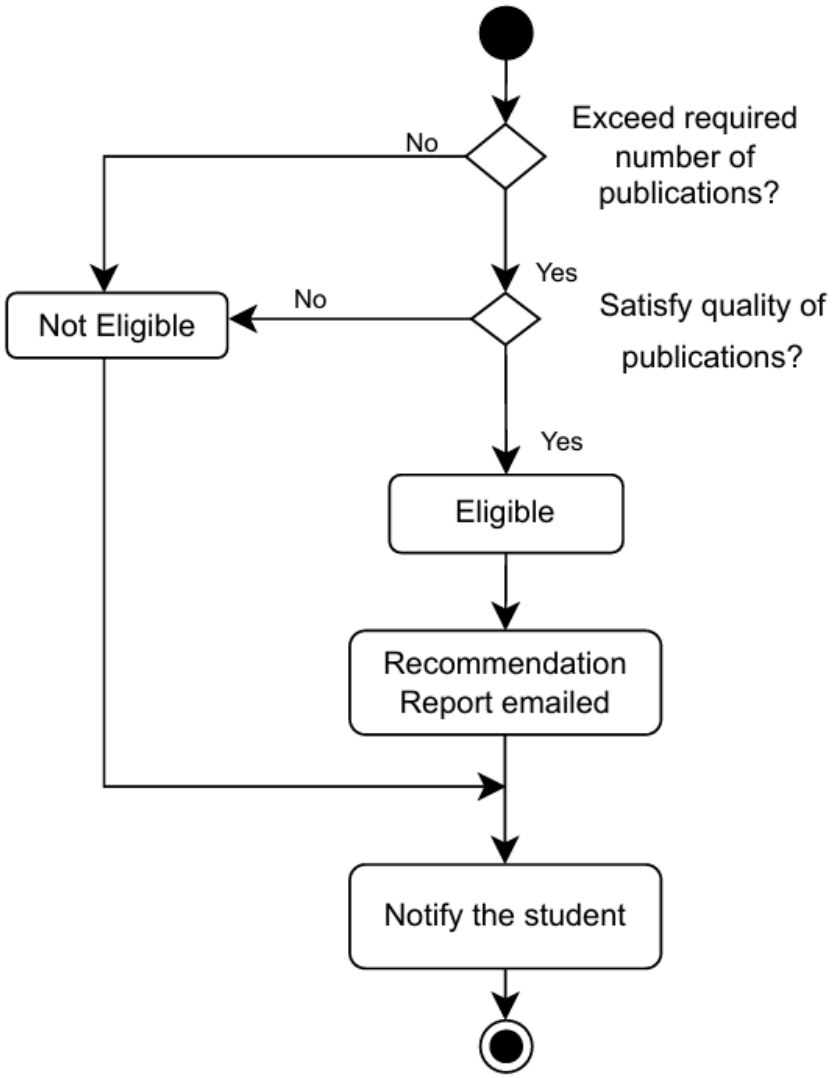
3.3.2.2 Approve/Reject Publication

The coordinator will assess verified publications, assessing each paper according to the university's format and policies. The coordinator then either will approve the paper by adding it to the university's research records or will reject the paper. Both results will be notified to the submitter.



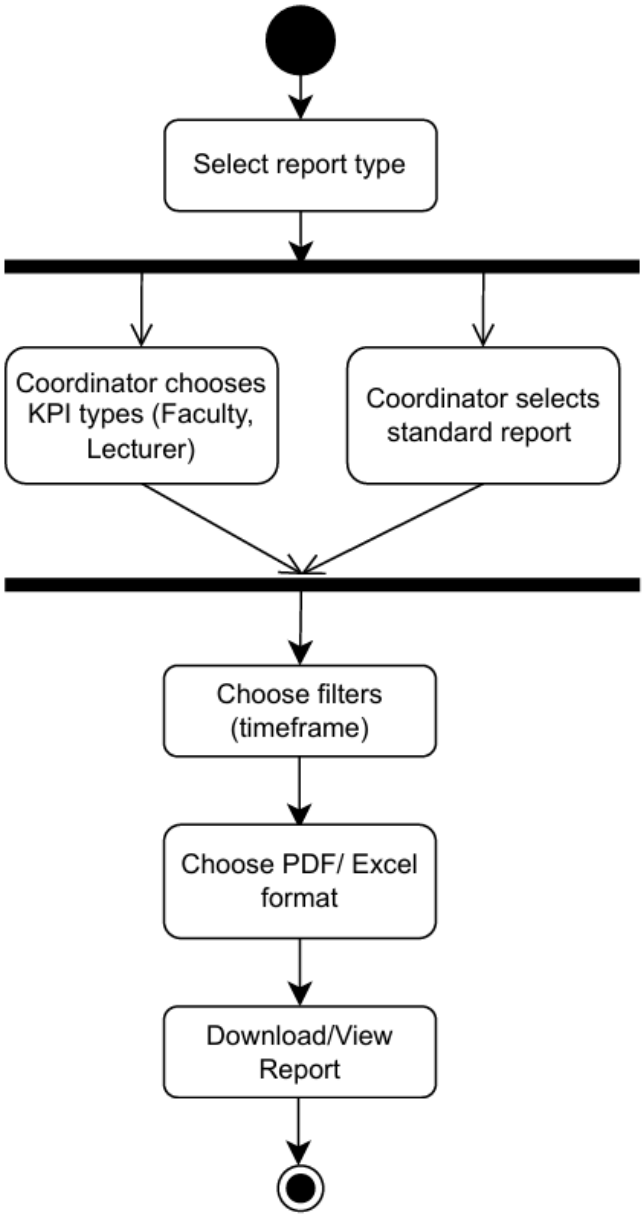
3.3.2.3 Recommend Students for Graduation

The Programme Coordinator checks on the student's graduation eligibility in two aspects. First, confirmation is done to ascertain if the student surpasses the minimum publication requirement of the programme. Secondly, assessment is made for the quality of such publications through indexing such as Scopus/WoS, authorship contribution, and impact metrics. If a student passes both requirements, the student is eligible, which would automatically generate a Recommendation Report to the Examination Board and notify the student. If either check fails, then that student is marked as not eligible, stopping the process.



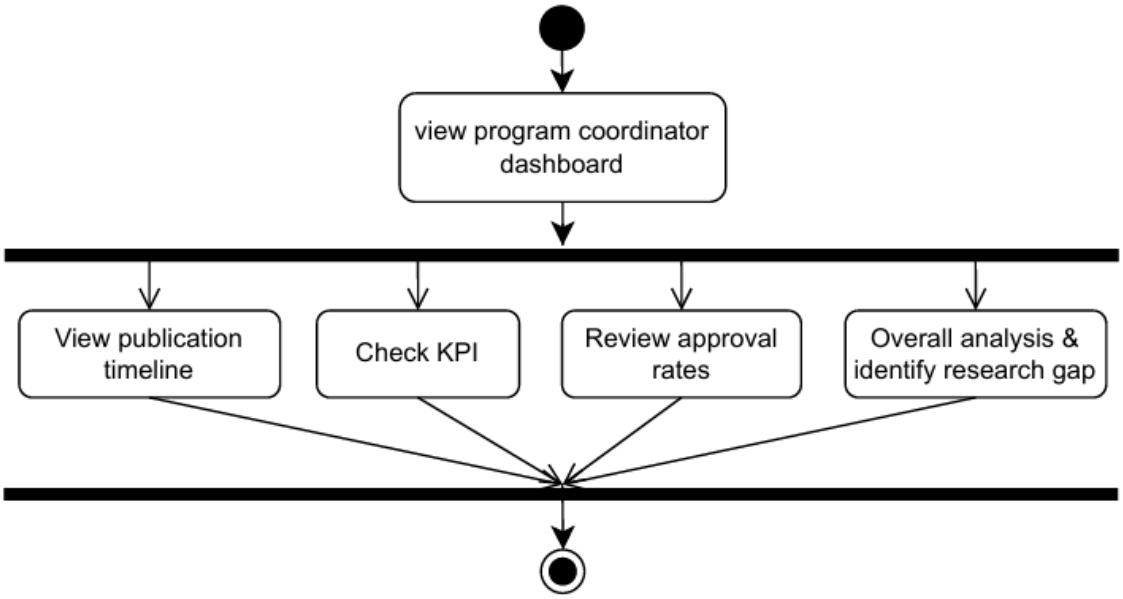
3.3.2.4 Generate Reports & KPI

Coordinator generates Faculty and Department or Lecturer KPI reports for research performance assessment. The coordinator enables selecting specific filter criteria like timeline and specialisation, then selects the desired output formats, either in PDF or Excel; this results in a downloadable customized report.



3.3.2.5 Monitor Research Output

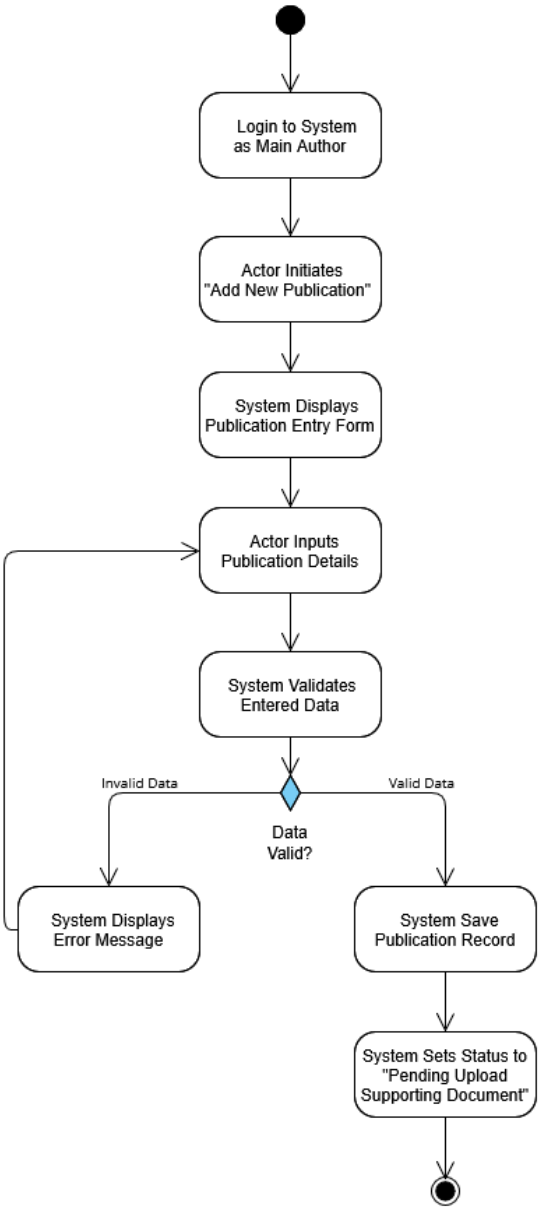
The coordinator sees the dashboards displaying publication metrics, monitors the approval rate and timeline, identifies gaps and high-performing areas and ensures the program maintains consistent research productivity and impact.



3.3.3 Actor 3 : Lecturer/Student when Main Author

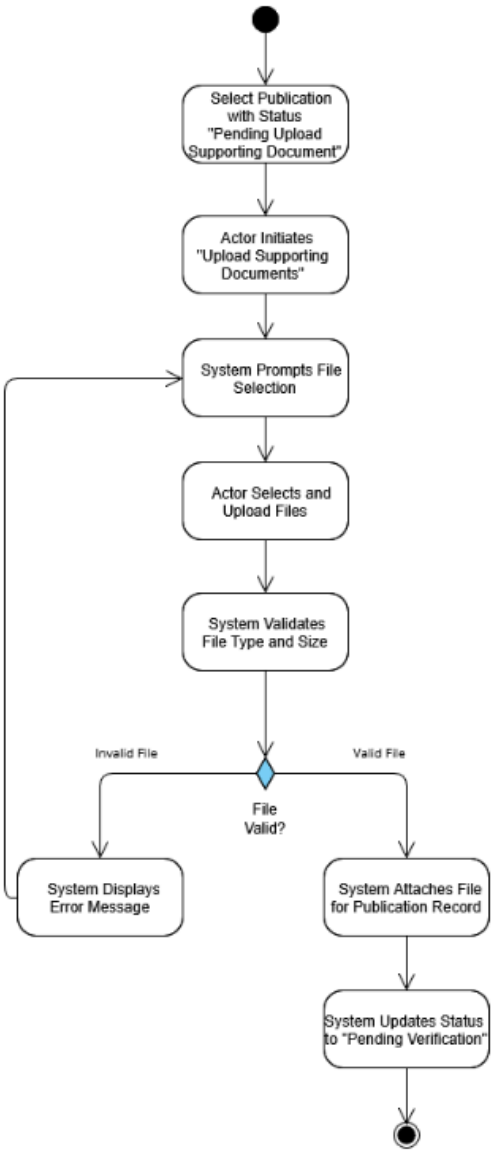
3.3.3.1 Add Publication

The lecturer or researcher can create a new publication entry in the system. The actor inputs the key publication information like title, authors, year, indexing type, and publication venue. The system validates the data entered and stores the publication with "Pending Supporting Upload" status. This starts the workflow for tracking publication and ensures all the publications are recorded before verification.



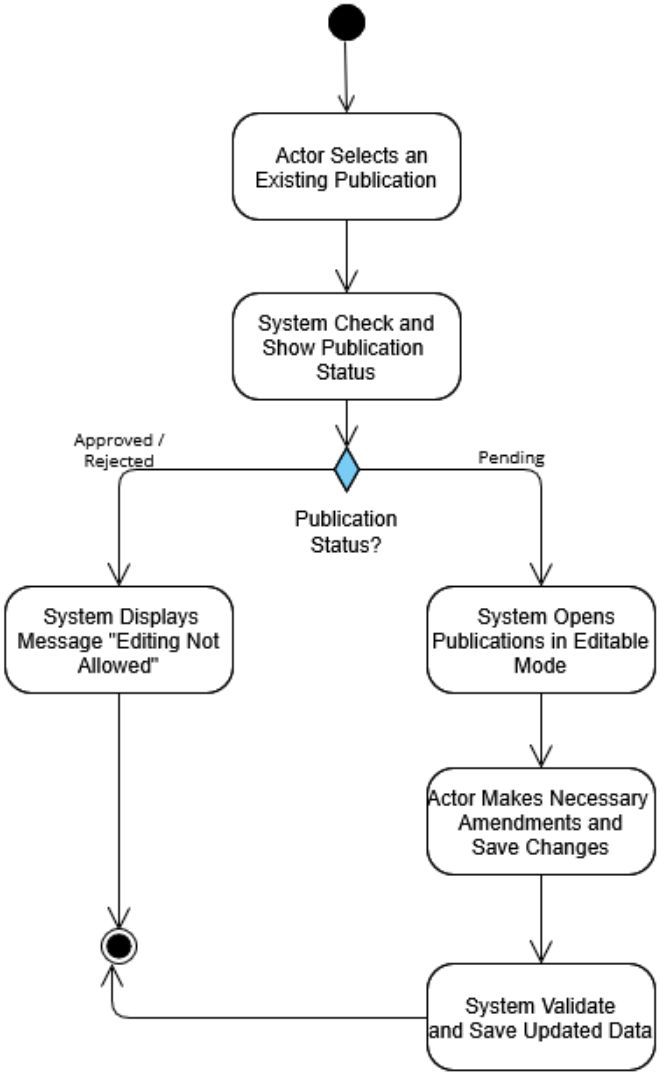
3.3.3.2 Upload Supporting Documents

The lecturer or researcher upload supporting documents to fulfill the requirement to proceed to the verification process. The system validates file type and size before attaching the documents to the publication record. The completion of this process results in an update of status that shows “Pending Verification”.



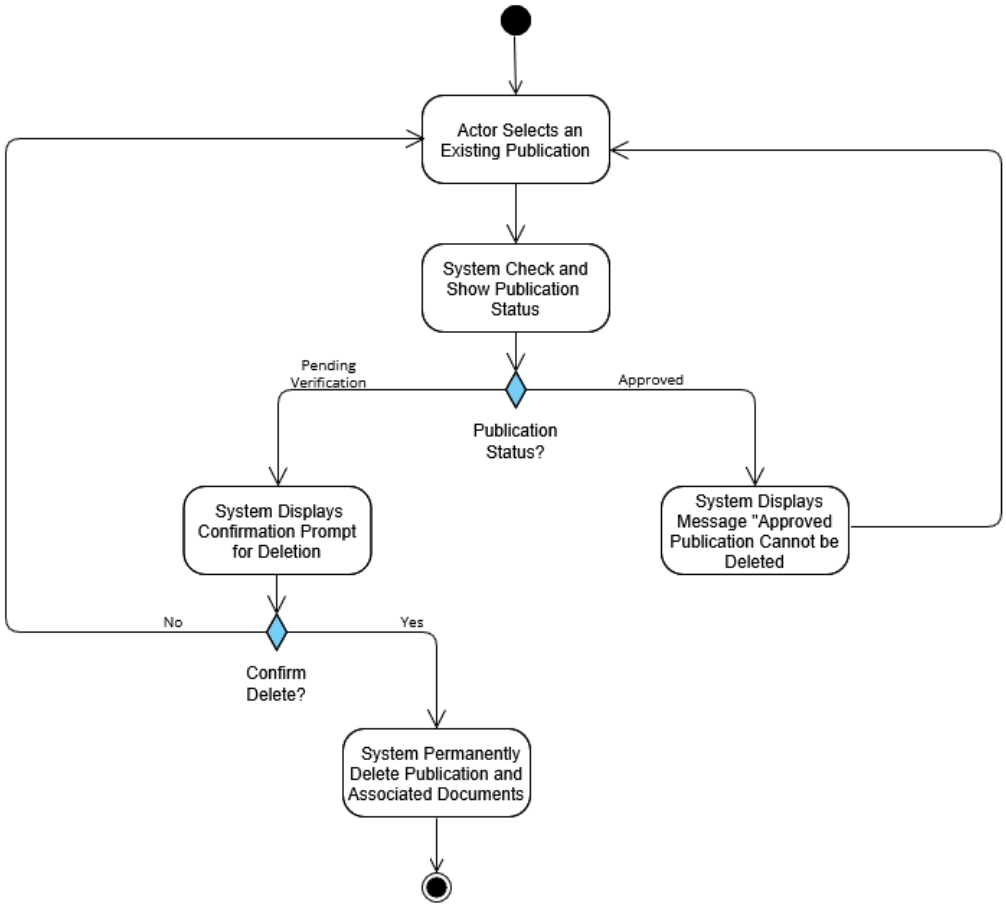
3.3.3.3 Edit Publication Details

This Use case enables the actor to make changes necessary in the publication details before the publication is verified. After the changes are recorded, the system saves the updated information and maintains data consistency. The editing can be allowed only when the publication has not been approved or rejected.



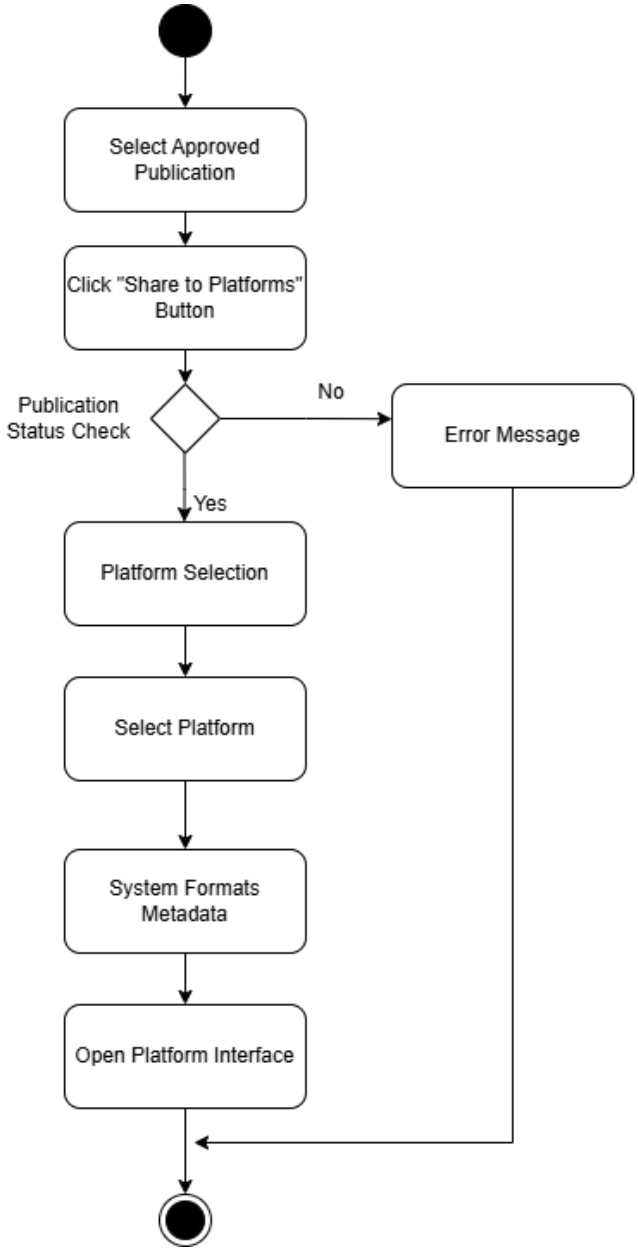
3.3.3.4 Delete Publication

This use case enables the lecturer or researcher to delete an existing publication entry that was submitted incorrectly or is no longer relevant. The system asks for confirmation of deletion to prevent accidental loss of data. Only unapproved publications can be deleted. After confirmation, it finally removes the publication and its associated documents from the system.



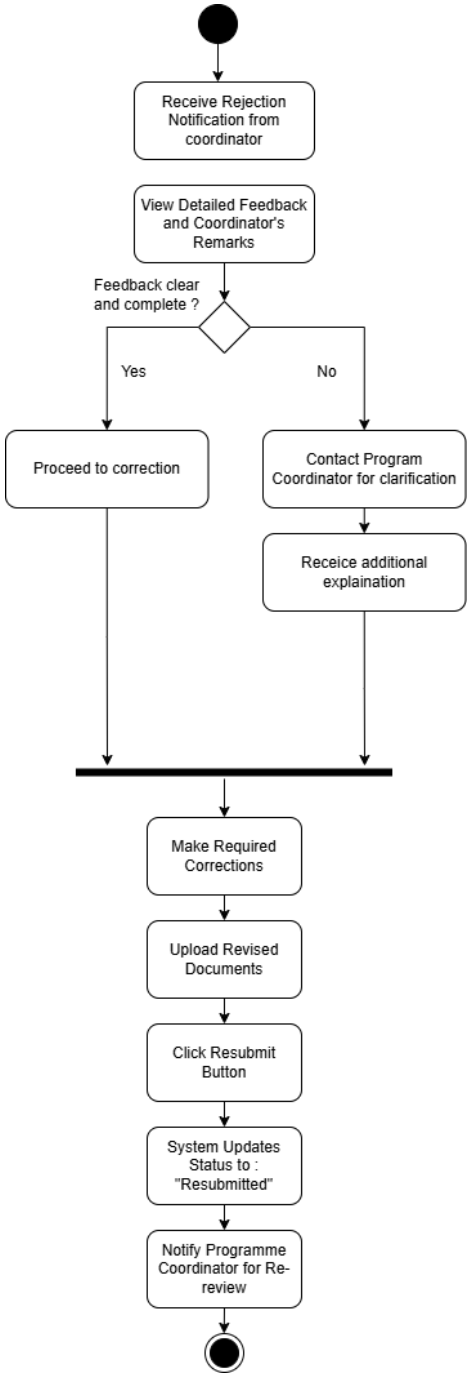
3.3.3.5 Shared Approved publication to Professional Platforms

It also allows direct sharing of approved publications by a lecturer or researcher on professional external platforms like LinkedIn. The system then formats the publication details of the title, authors, year, and journal into a shareable post or metadata template. The lecturer confirms the action, and the system will either post directly or open the platform's sharing interface. In this way, academic visibility can be improved, and professional networking will be supported.



3.3.3.6 Resubmit Publication (If program coordinator rejects publication)

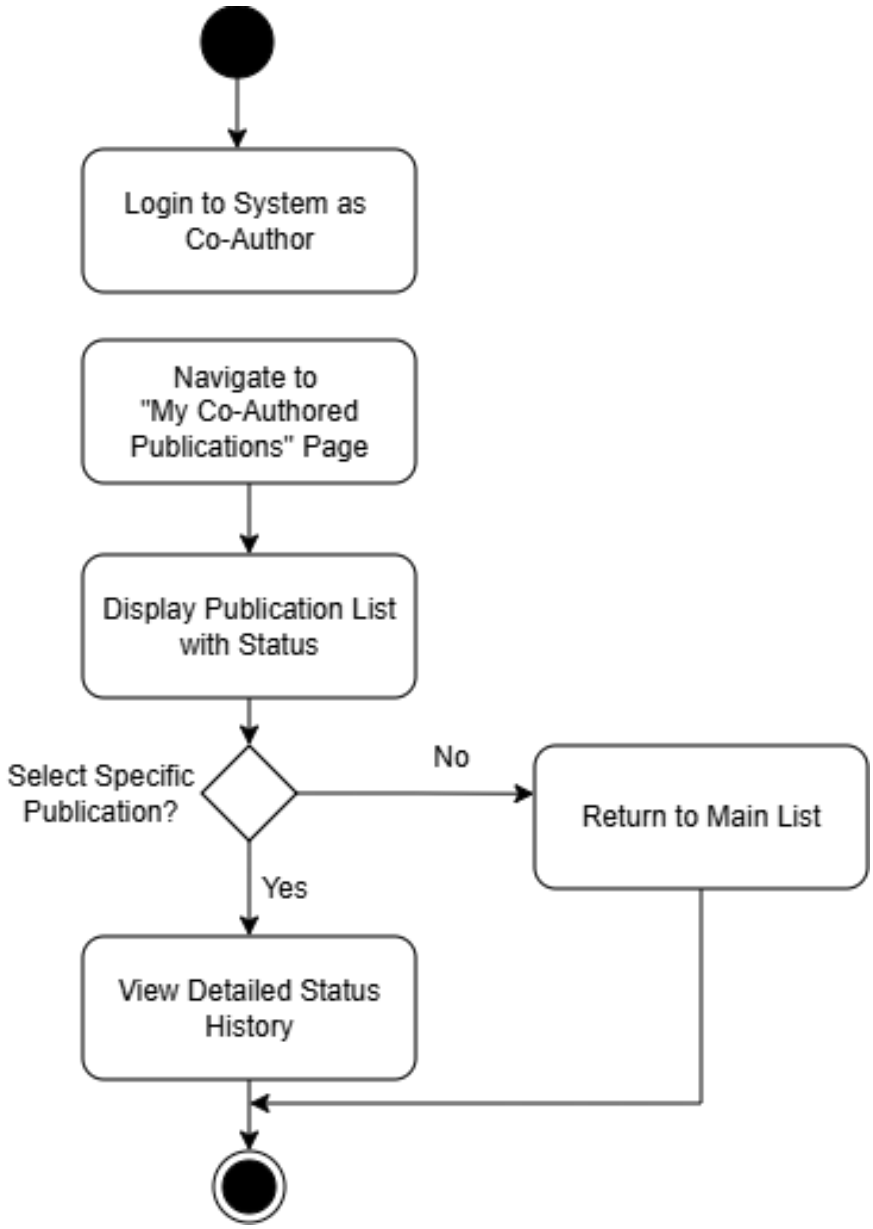
This activity diagram shows the complete workflow a main author needs to perform in order to resubmit a previously rejected publication by the Programme Coordinator. The process starts when the author receives notification of the rejection and finishes when the resubmission has been completed, awaiting re-review.



3.3.4 Actor 4 : Lecturer/Student when Co-Author

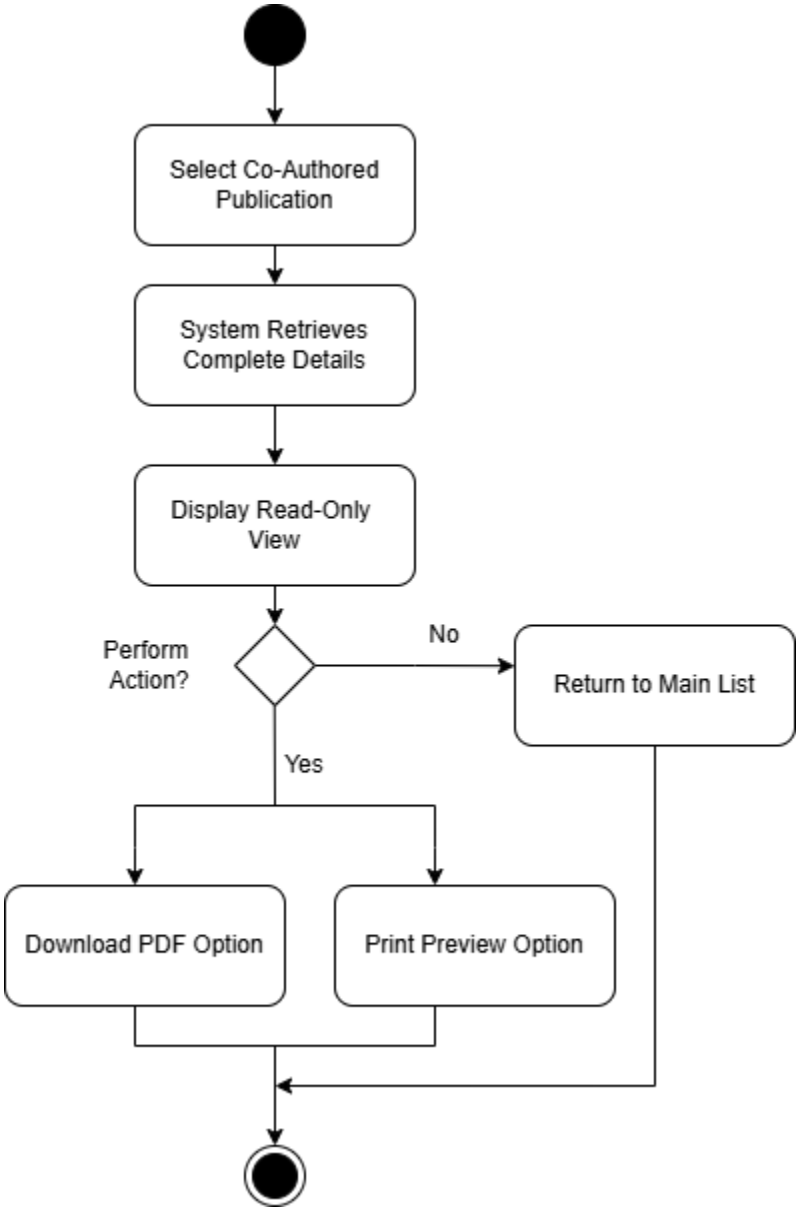
3.3.4.1 View Publication Status

Co-authors can see the current status of publications that they are listed on as a contributor, but not as the primary author. Examples of this include the ability to track verification progress from "Pending" to "Verified" to "Approved," being able to see if Programme Coordinators have requested revisions, and the final outcome in the publication submission process.



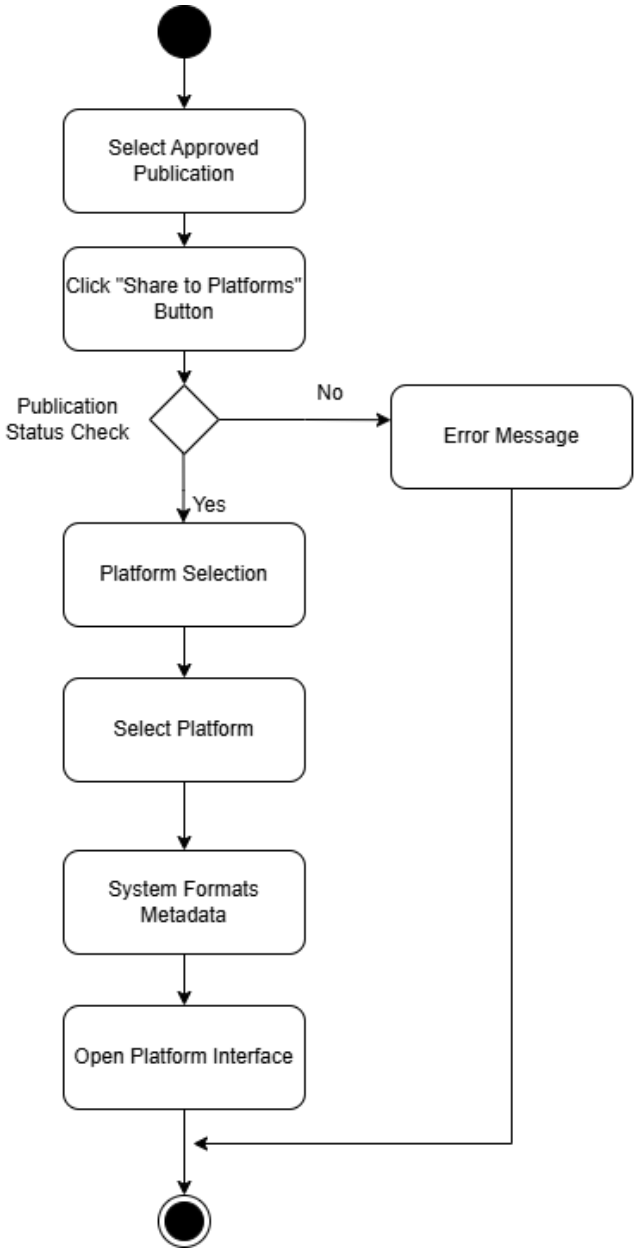
3.3.4.2 View Publication Details

Co-authors can view complete details of all the publications they are listed as an author for, such as title, abstract, author list, journal/conference, publication date, DOI, indexing status, and supporting documentation. Editing rights, however, are limited compared to the rights bestowed on a main author.



3.3.4.3 Share Approved Publication to Professional Platforms

The co-author can share with external professional platforms such as LinkedIn, ResearchGate, ORCID, or other academic social networks those verified publications, where he or she is listed as an author. The system will format the publication metadata according to the destination platform and either post automatically or open the sharing interface of that platform.



4 Architecture Design

4.1 Architecture Diagram

The Academic Publication and Research Tracking System utilizes a multi-tier (layered) architecture to ensure modularity, scalability, and separation of concerns. The system is structured into five distinct layers: Presentation, Business, Service, Data Access, and Data Layer. This design ensures that user interactions are decoupled from business logic and data storage, allowing for easier maintenance and future upgrades.

Presentation Layer

The Presentation Layer functions as the primary interface for all system actors, including the Admin, Program Coordinator, and Main Author/Co-Author. Interactions begin at the Login/Select Role UI, which serves as a central gateway to authenticate users and route them to their appropriate environments based on their credentials. Once authenticated, users are directed to role-specific dashboards which is the Admin Dashboard for system oversight, the Coordinator Dashboard for verification tasks, and the Main Author/Co-Author Dashboard for managing research submissions, ensuring a tailored and secure user experience.

Business Layer

The Business Layer encapsulates the system's core application logic, organized into three distinct subsystems to handle specific functional requirements. The Admin Subsystem manages high-level administrative tasks through the System Setting and User Manager modules. The Coordinator Subsystem is responsible for the academic review process, utilizing the Verification Workflow to assess submissions and the Graduation Eligibility Checker to validate student criteria. Simultaneously, the Main Author and Co-Author Subsystem facilitates the creation and modification of research entries via the Publication Management module, ensuring that business rules are applied consistently across all operations.

Service Layer

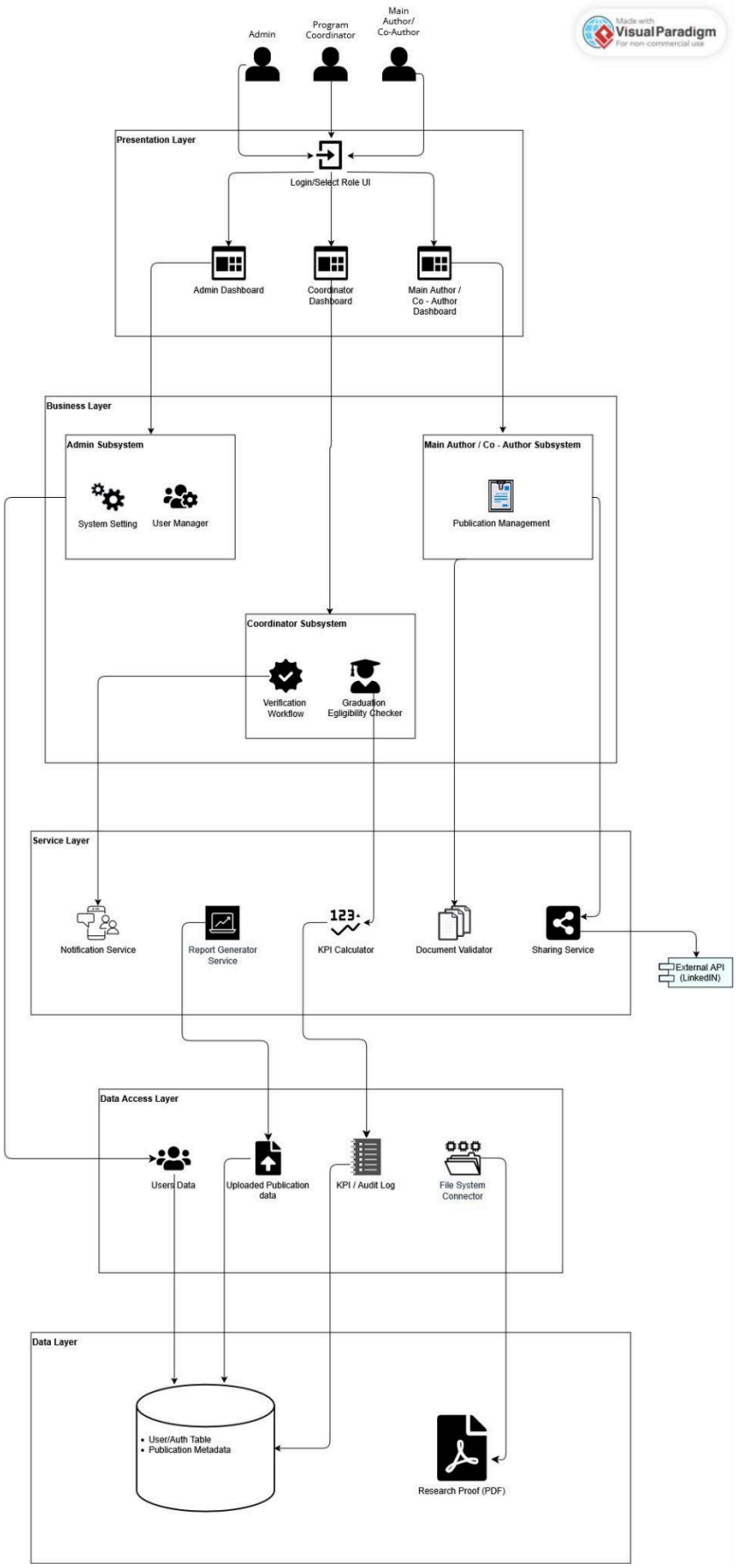
Supporting the business logic, the Service Layer provides specialized, reusable components that handle specific background processing tasks. This layer includes a Notification Service for managing user alerts, a Report Generator Service for compiling system data, and a KPI Calculator for computing performance metrics. It also features a Document Validator to ensure uploaded files meet system standards and a Sharing Service that interfaces with external APIs, such as LinkedIn, allowing authors to disseminate approved publications to professional networks.

Data Access Layer

The Data Access Layer acts as an abstraction barrier that manages communication between the application logic and the underlying data storage. It ensures data integrity and security by utilizing specific connectors for different data types. These include connectors for Users Data to manage profiles, Uploaded Publication Data for handling research metadata, and KPI or Audit Logs for tracking system performance. Additionally, a File System Connector is employed to manage the retrieval and storage of physical documents, isolating the business logic from direct database interactions.

Data Layer

The Data Layer forms the foundation of the architecture, responsible for the persistent storage of all system information. It employs a Relational Database to store structured records, such as the User and Auth Table for credentials and Publication Metadata for research details. Complementing this is a dedicated File Storage system specifically designed for managing large physical files, such as Research Proof (PDF) documents. This separation of structured data and file storage optimizes system performance and scalability.


Admin
Program Coordinator
Main Author/ Co-Author
Made with Visual Paradigm For non-commercial use
Presentation Layer
Login/Select Role UI
Admin Dashboard
Coordinator Dashboard
Main Author / Co - Author Dashboard
Business Layer
Admin Subsystem
System Setting
User Manager
Main Author / Co - Author Subsystem
Publication Management
Coordinator Subsystem
Verification Workflow
Graduation Egligibility Checker
Service Layer
Notification Service
Report Generator Service
KPI Calculator
Document Validator
Sharing Service
External API (LinkedIN)
Data Access Layer
Users Data
Uploaded Publication data
KPI / Audit Log
File System Connector
Data Layer
• User/Auth Table
• Publication Metadata
Research Proof (PDF)

4.2 Logical Architecture

Frontend (Client Browser)

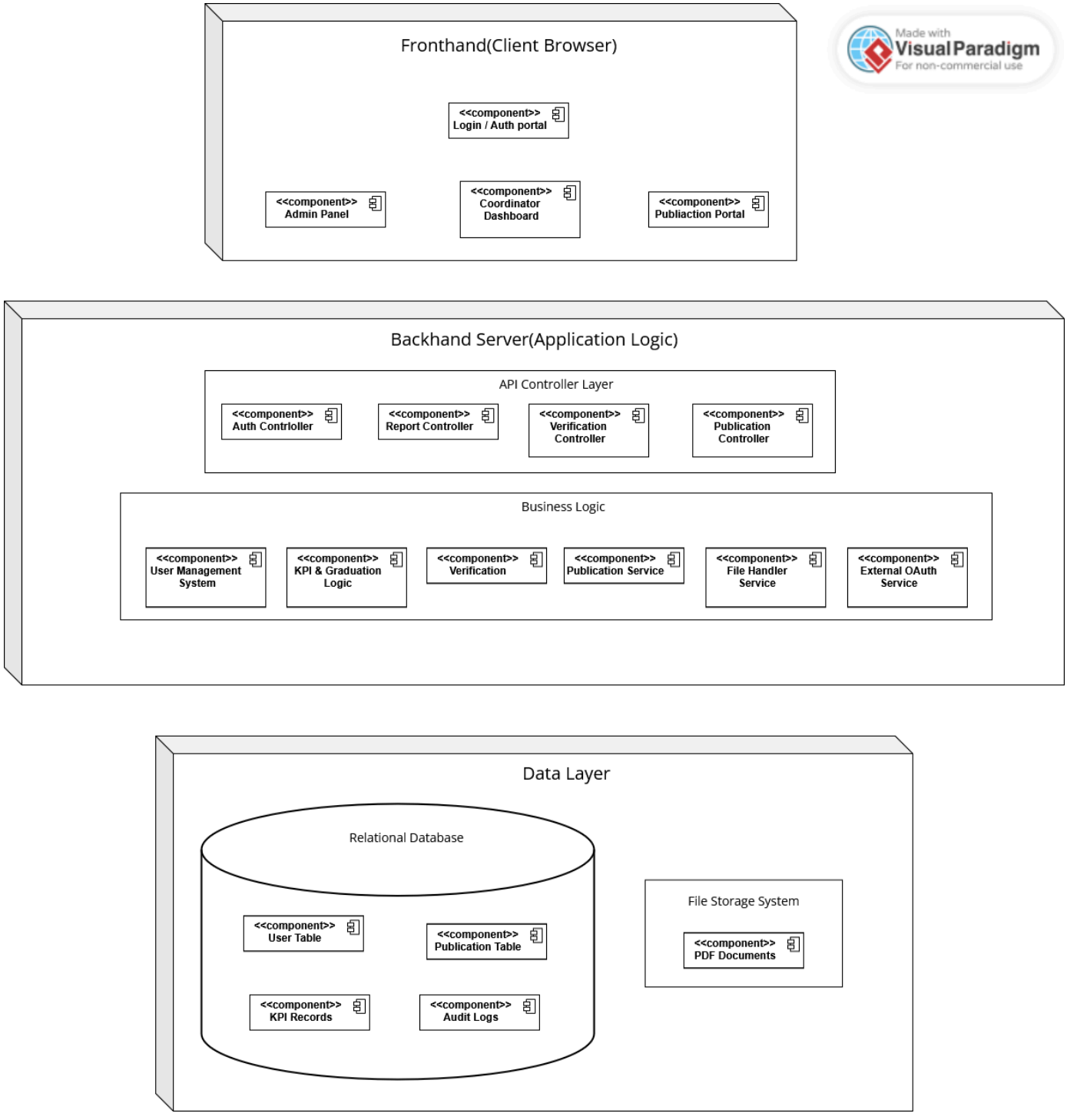
The Frontend layer serves as the direct interface for users, operating within the client browser to facilitate interaction with the system. It is composed of distinct portals tailored to specific user roles: the Login/Auth Portal manages secure access and entry, while the Admin Panel, Coordinator Dashboard, and Publication Portal provide specialized views and functionalities for administrators, coordinators, and authors respectively. This layer is responsible for capturing user inputs and presenting data retrieved from the server, ensuring a responsive and role-specific user experience.

Backend Server (Application Logic)

The Backend Server acts as the core processing unit of the system, divided into two sub-layers to separate request handling from business processing. The API Controller Layer sits at the forefront, managing incoming HTTP requests through specialized controllers such as the Auth, Report, Verification, and Publication Controllers. Below this lies the Business Logic layer, which executes the actual functional rules of the system. This includes components for the User Management System, KPI & Graduation Logic, Verification workflows, and Publication Service. It also houses the File Handler Service for managing document uploads and the External OAuth Service to handle integrations with third-party platforms.

Data Layer

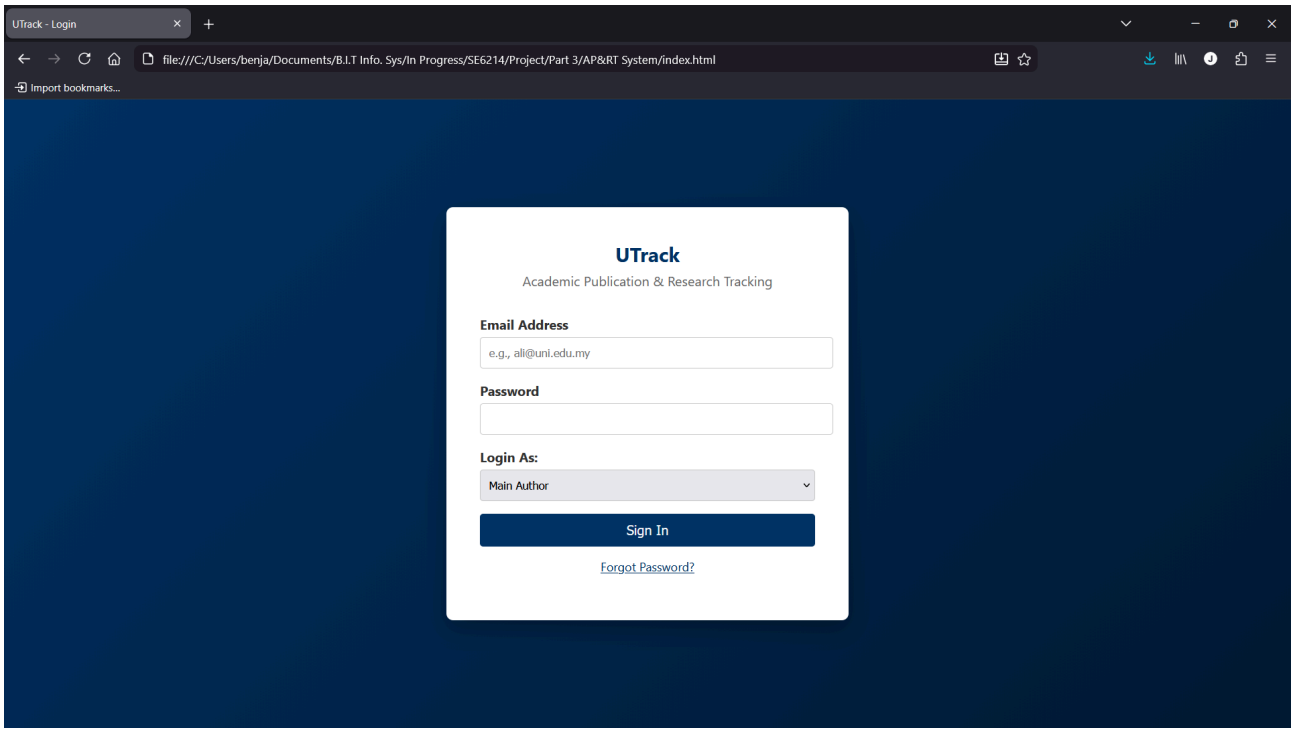
The Data Layer handles the persistent storage and retrieval of all system information, utilizing a hybrid storage approach to optimize performance. A Relational Database is used to store structured data, including the User Table for account details, Publication Table for metadata, KPI Records for performance metrics, and Audit Logs for tracking system activity. Complementing this is a dedicated File Storage System specifically architected to store unstructured binary data, such as the actual PDF Documents uploaded by researchers, ensuring that the main database remains lightweight and efficient.

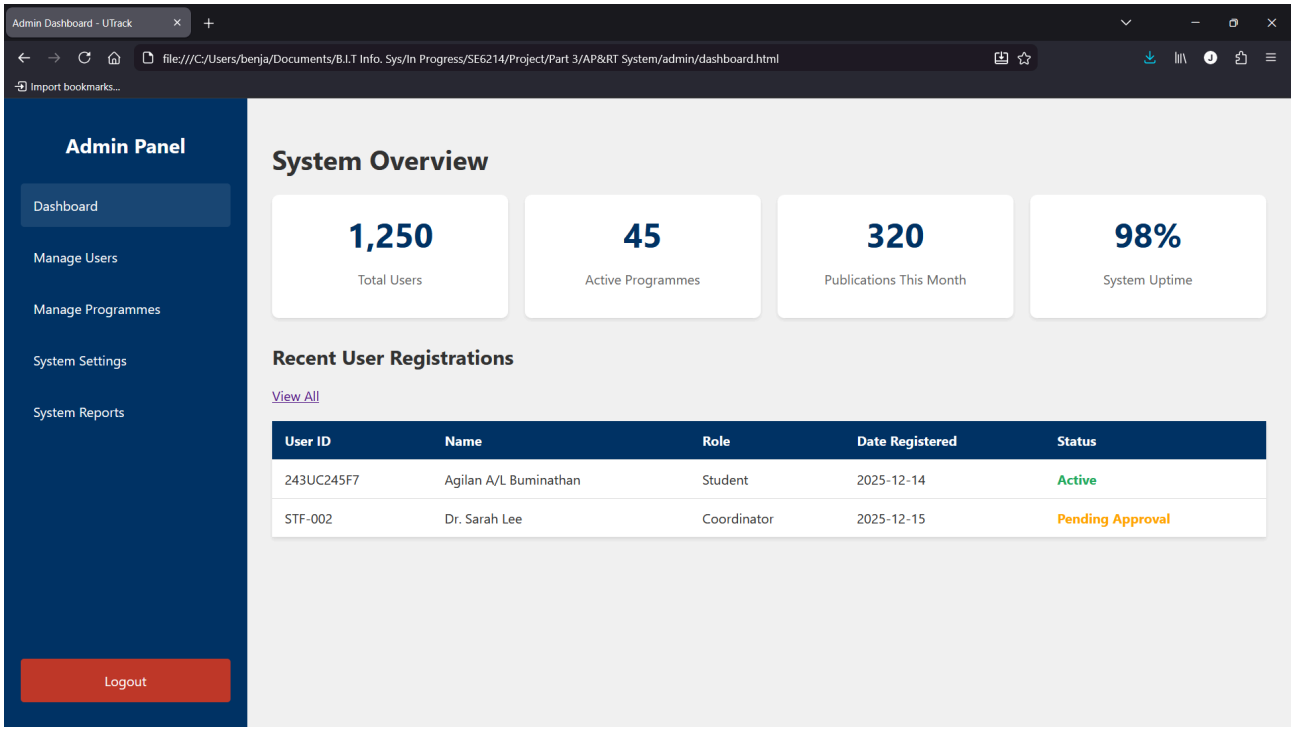


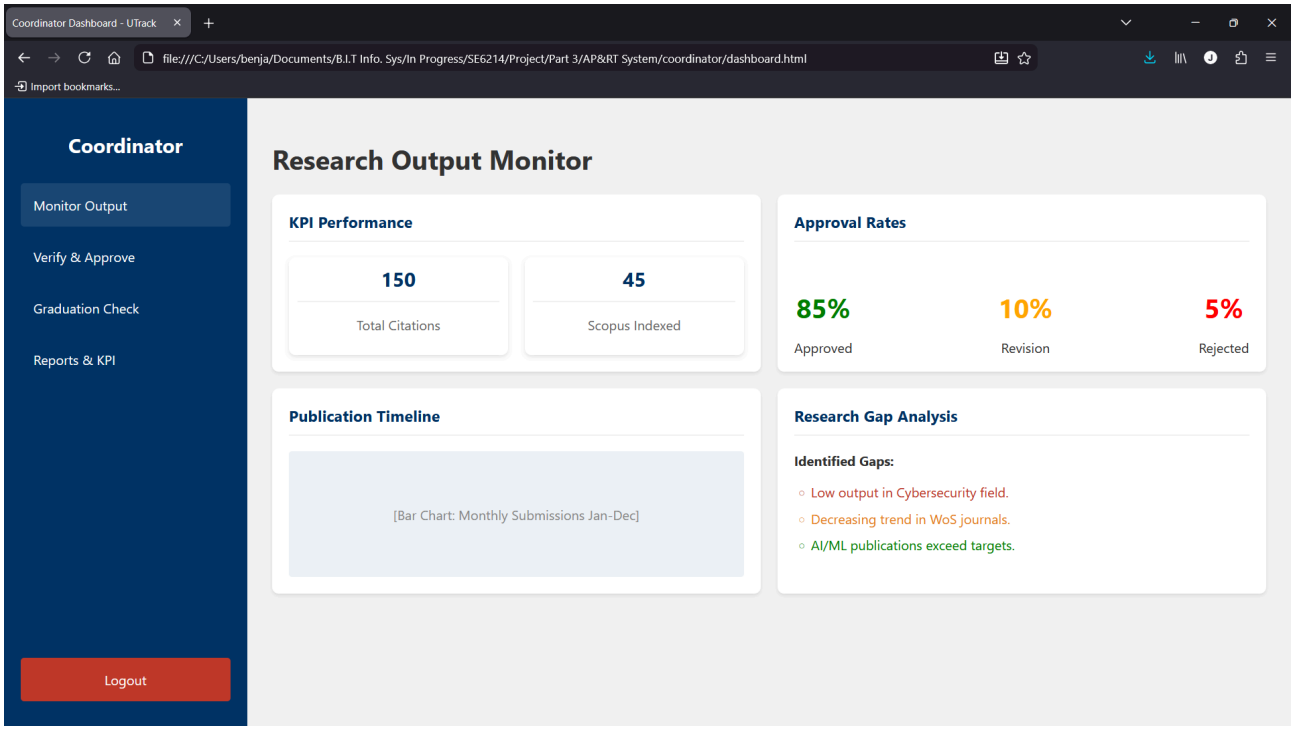
Made with VisualParadigm
For non-commercial use

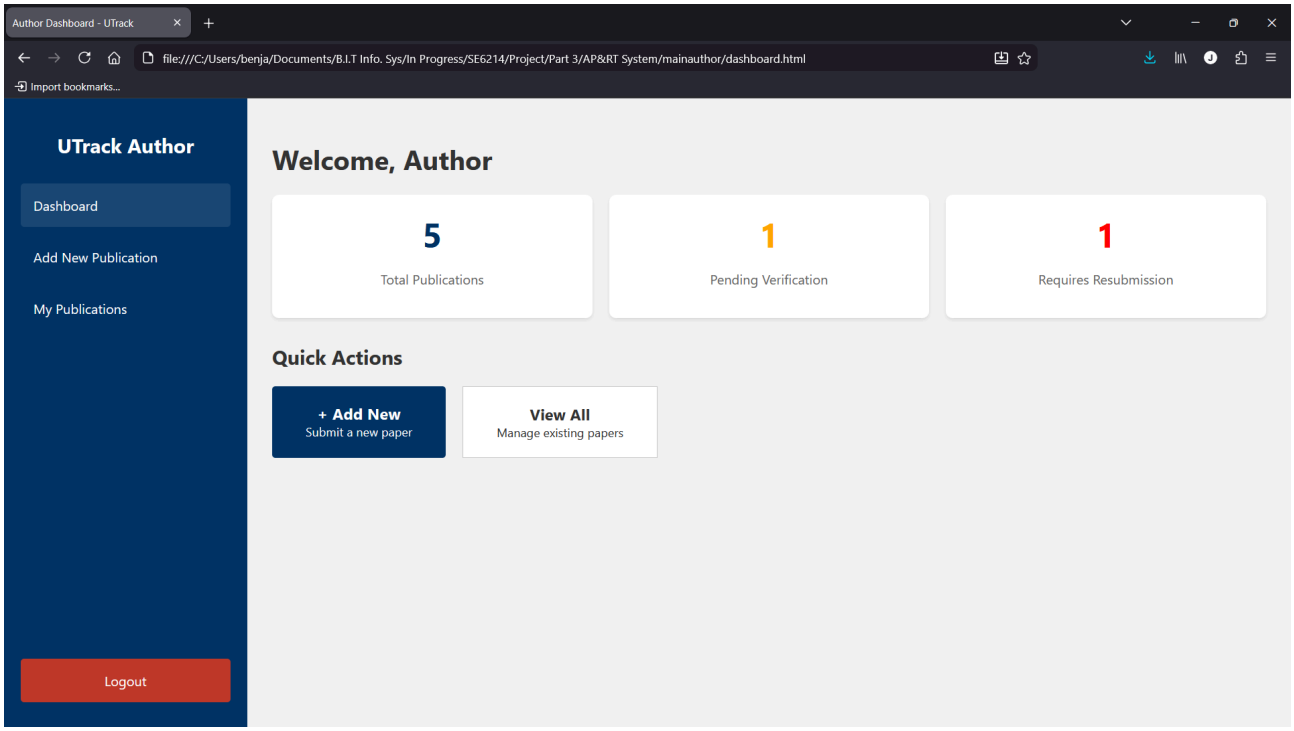
5 Interface Design

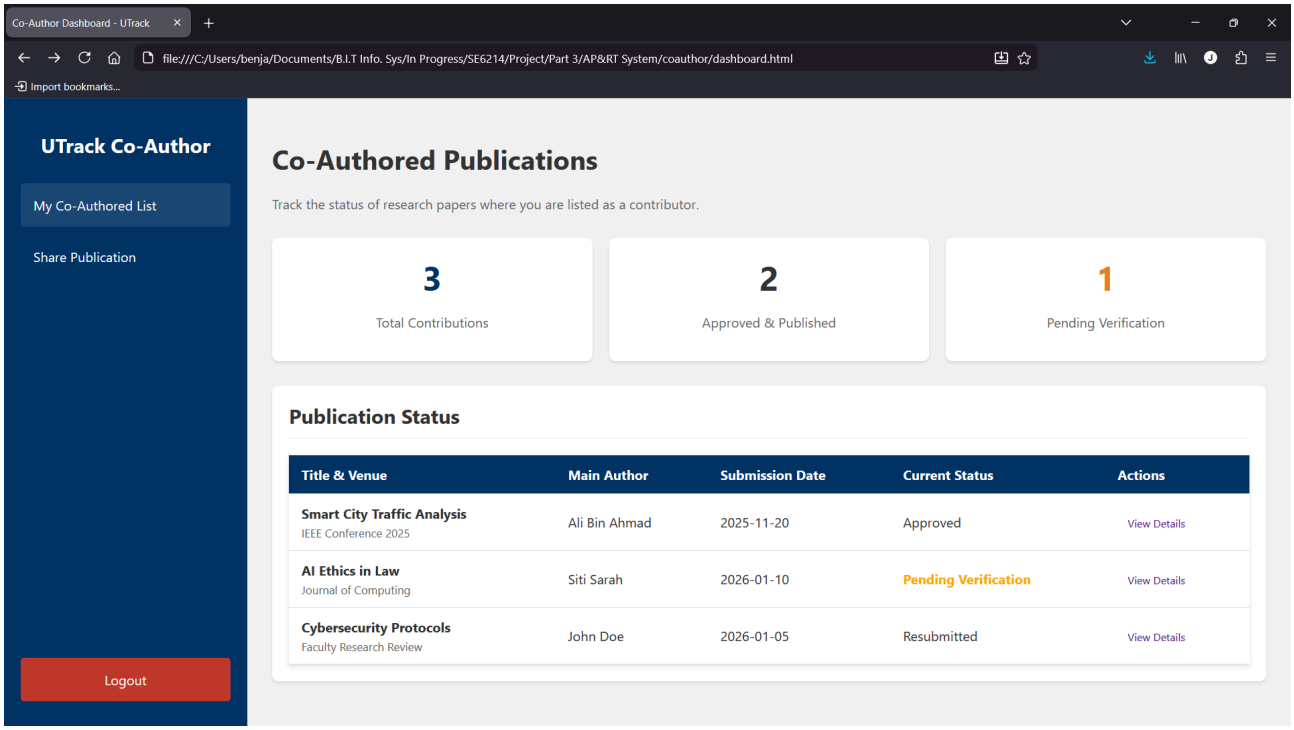
5. 1 Main Screens





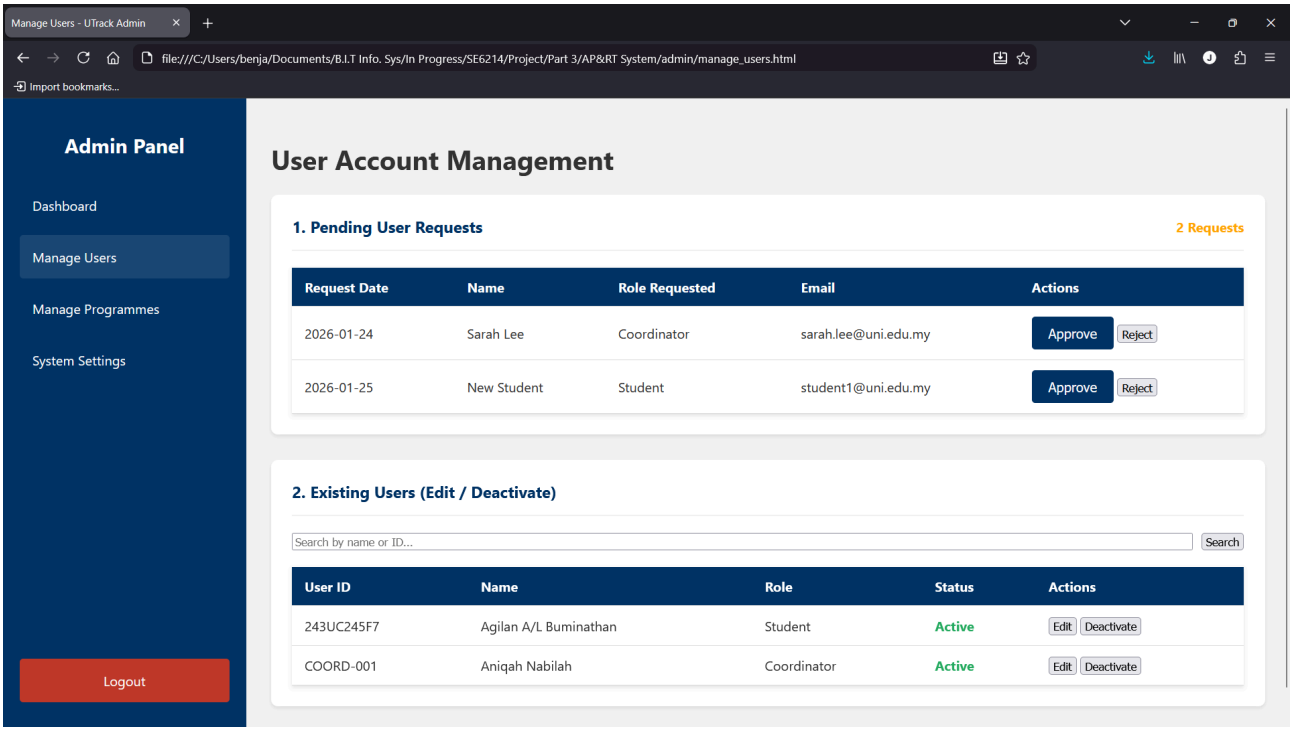

Coordinator Dashboard - UTrack
file:///C:/Users/benja/Documents/B.I.T Info. Sys/In Progress/SE6214/Project/Part 3/AP&RT System/coordinator/dashboard.html
Import bookmarks...
Coordinator
Monitor Output
Verify & Approve
Graduation Check
Reports & KPI
Logout
Research Output Monitor
KPI Performance
150
Total Citations
45
Scopus Indexed
Approval Rates
85%
Approved
10%
Revision
5%
Rejected
Publication Timeline
[Bar Chart: Monthly Submissions Jan-Dec]
Research Gap Analysis
Identified Gaps:
Low output in Cybersecurity field.
Decreasing trend in WoS journals.
AI/ML publications exceed targets.


Author Dashboard - UTrack
file:///C:/Users/benja/Documents/B.I.T Info. Sys/In Progress/SE6214/Project/Part 3/AP&RT System/mainauthor/dashboard.html
Import bookmarks...
UTrack Author
Dashboard
Add New Publication
My Publications
Logout
Welcome, Author
5
Total Publications
1
Pending Verification
1
Requires Resubmission
Quick Actions
+ Add New
Submit a new paper
View All
Manage existing papers

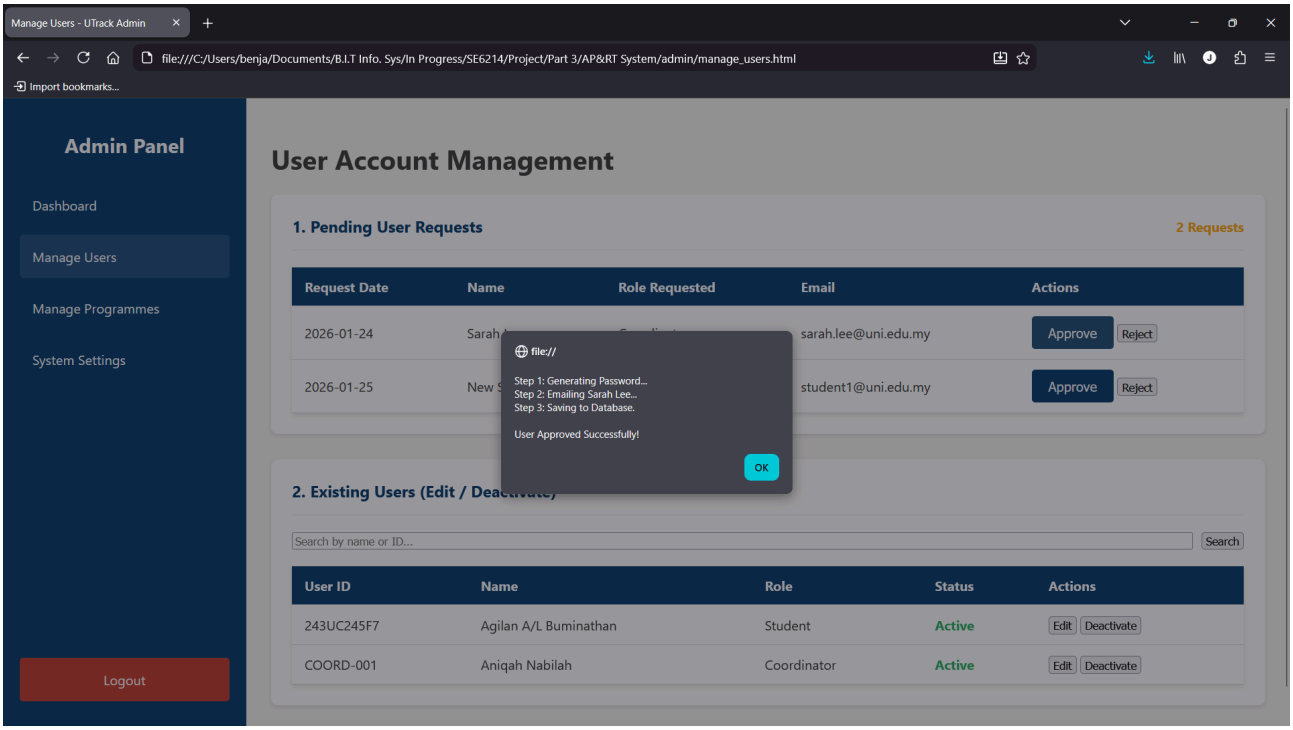


5.2 Subsystem 1 Screens (Admin)

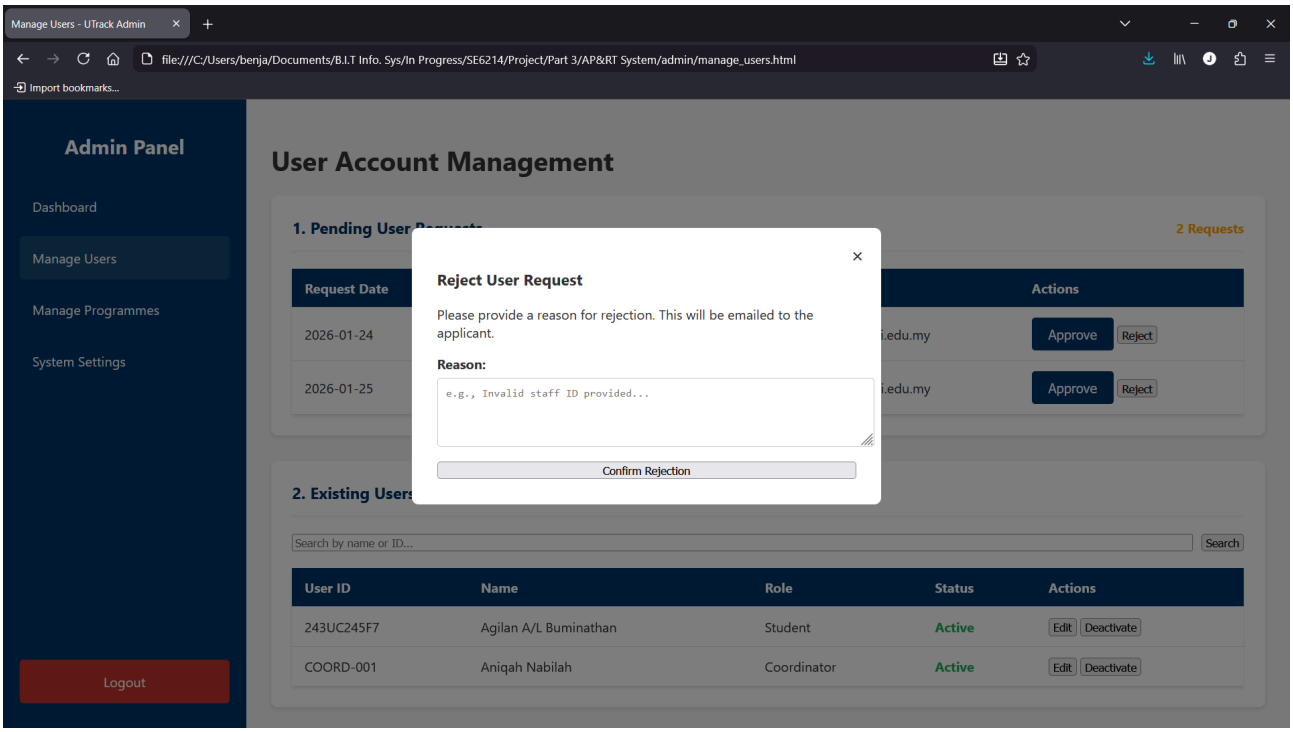
5.2.1 Manage Users



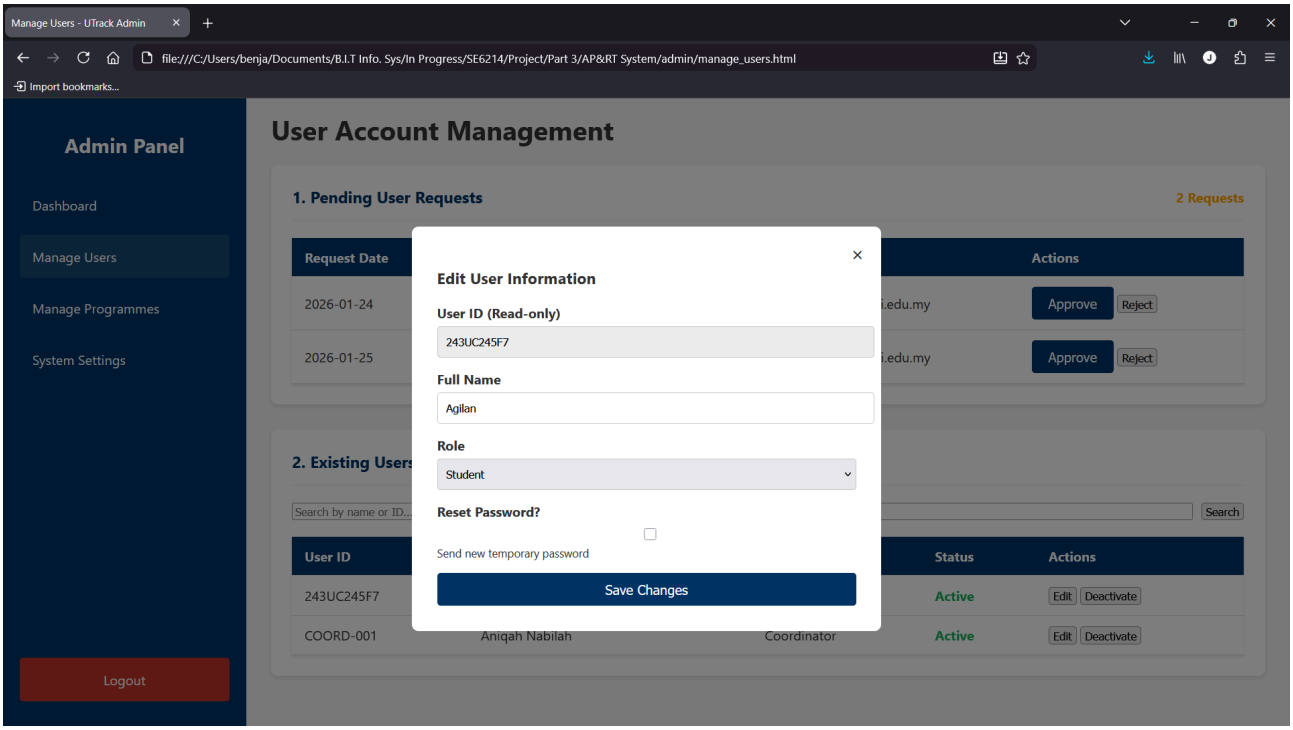
Manage Users main page



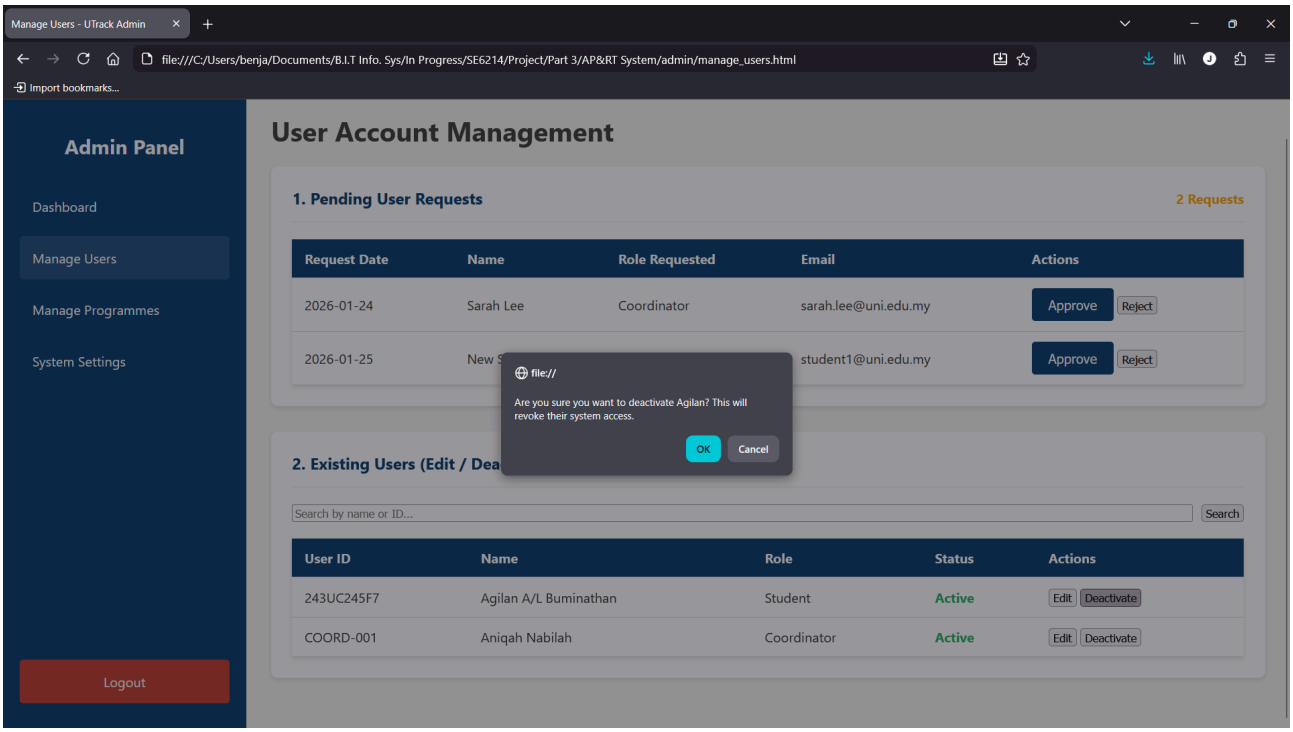
Approve users



Reject Users

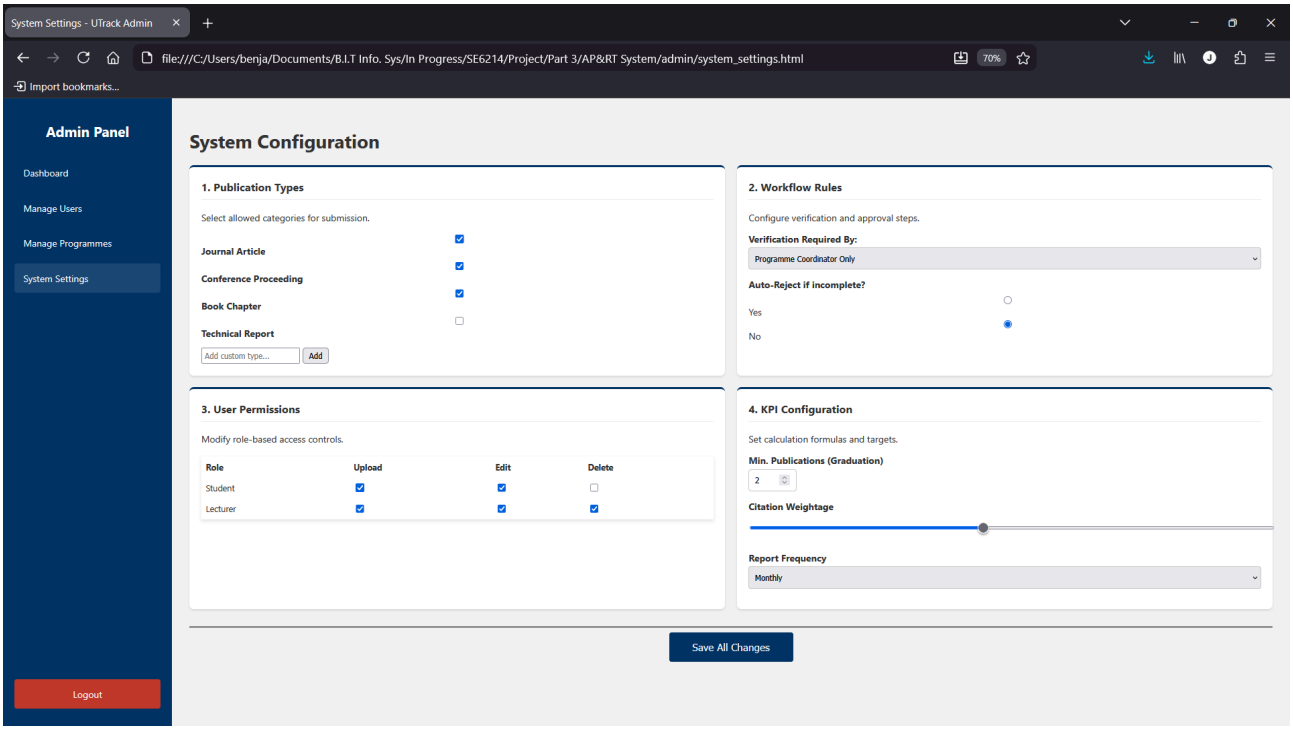


Edit users

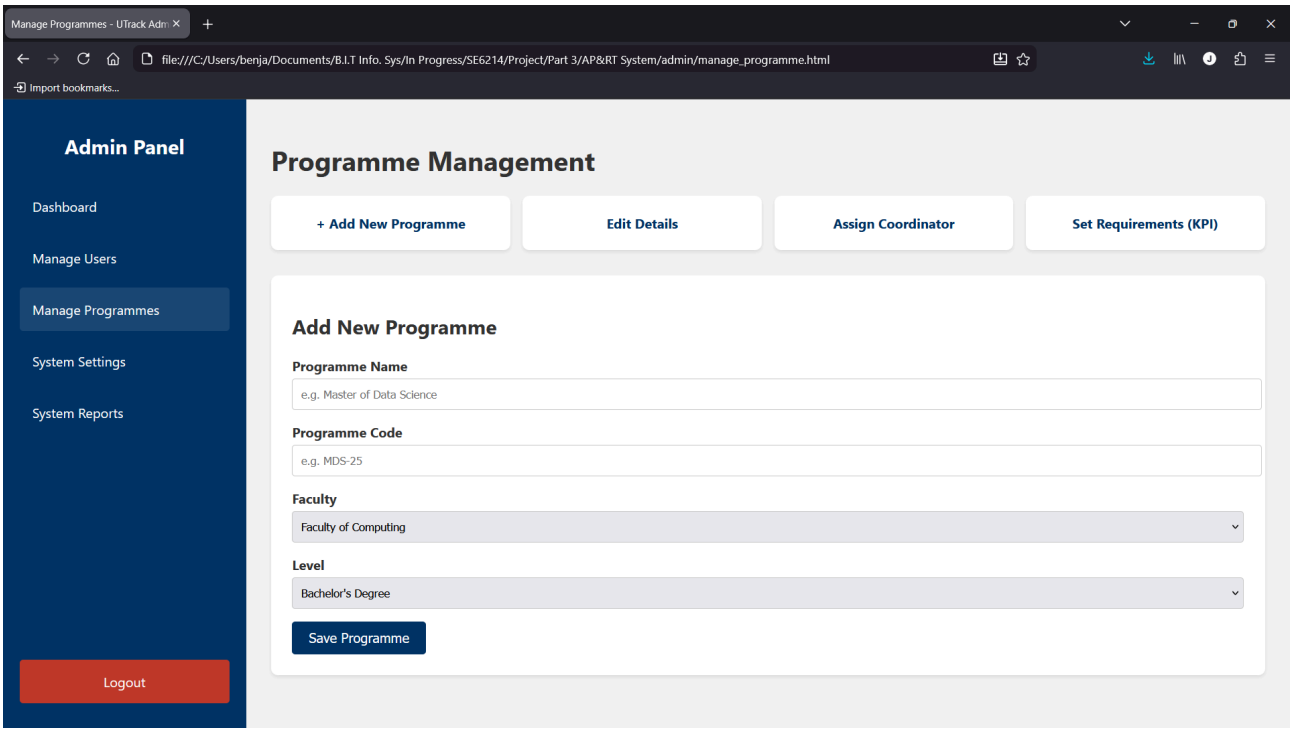


Deactivate users

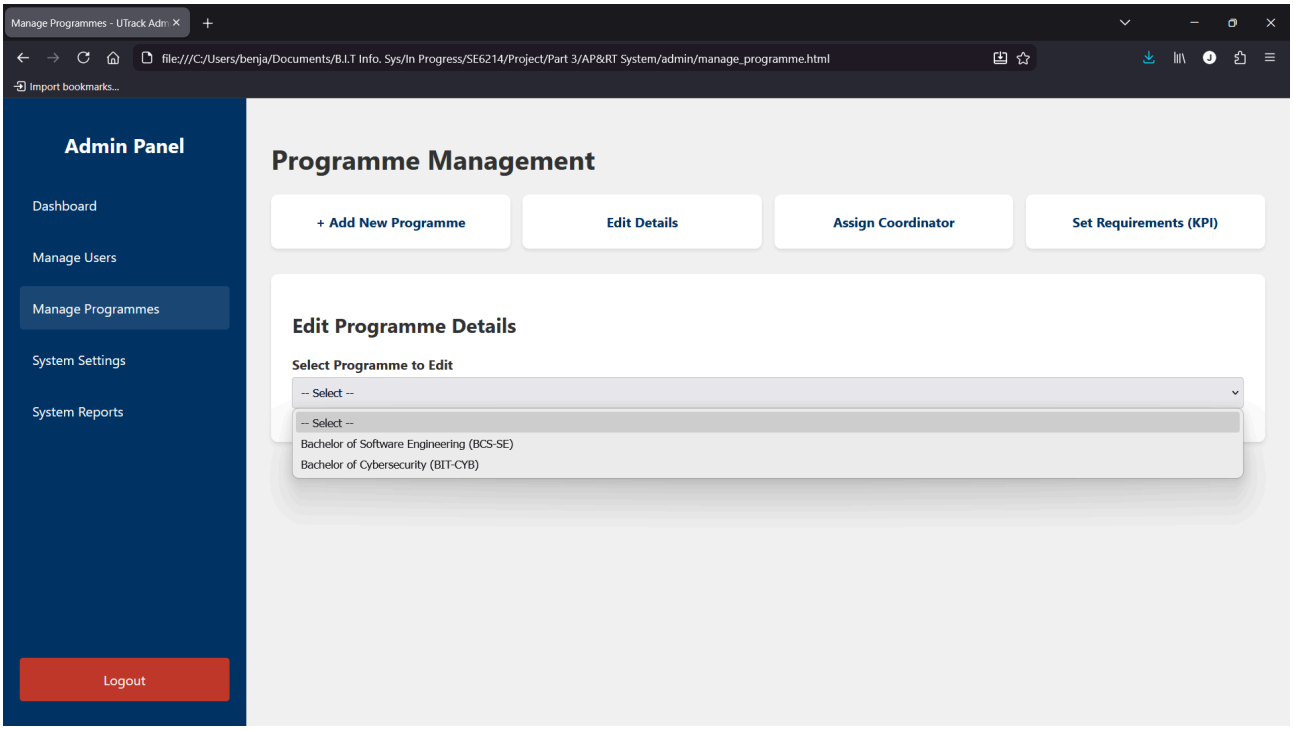
5.5.2 Manage system settings



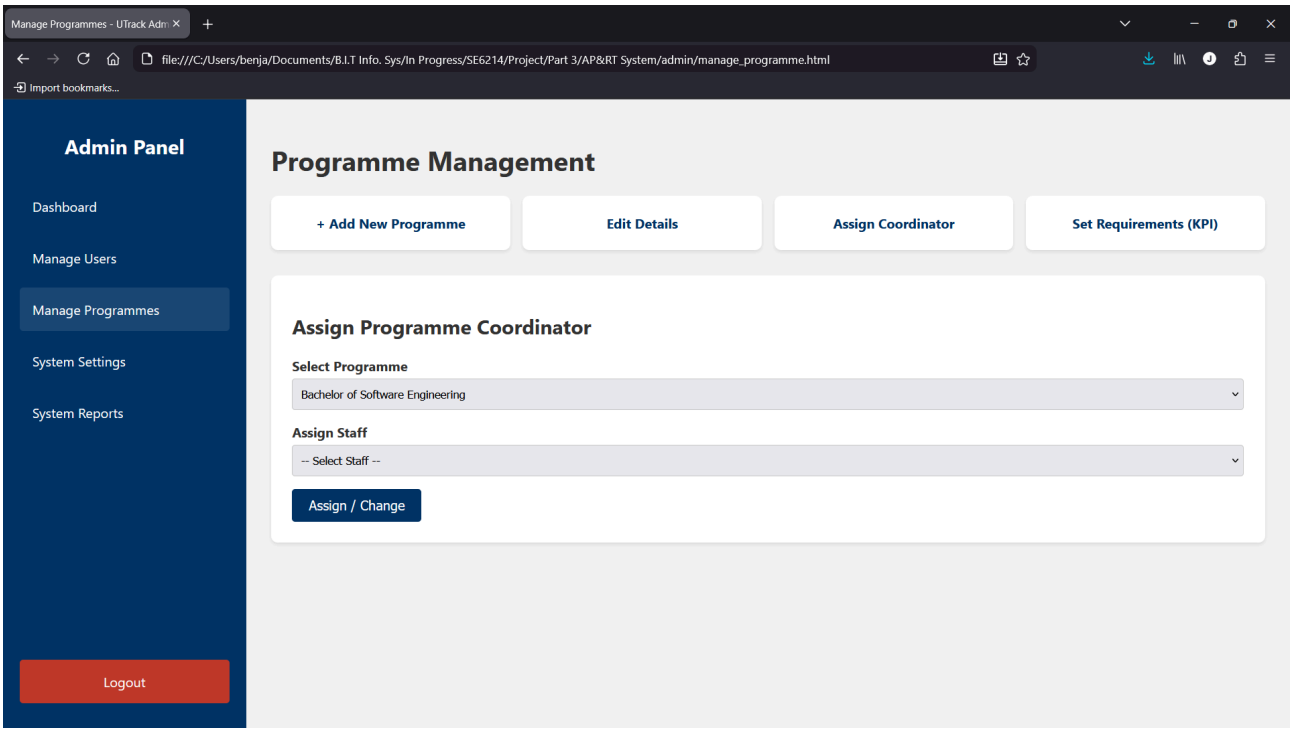
5.5.3 Manage Programme



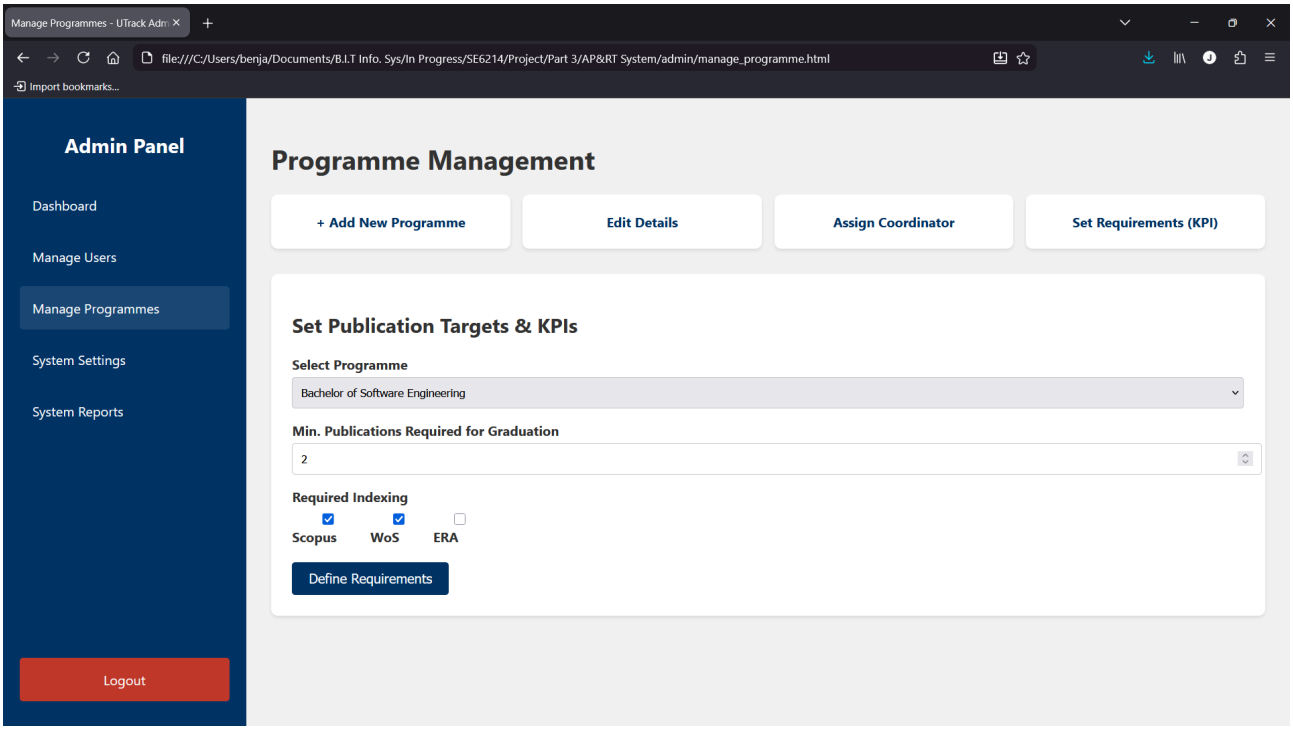
Add new programme



Edit programme details

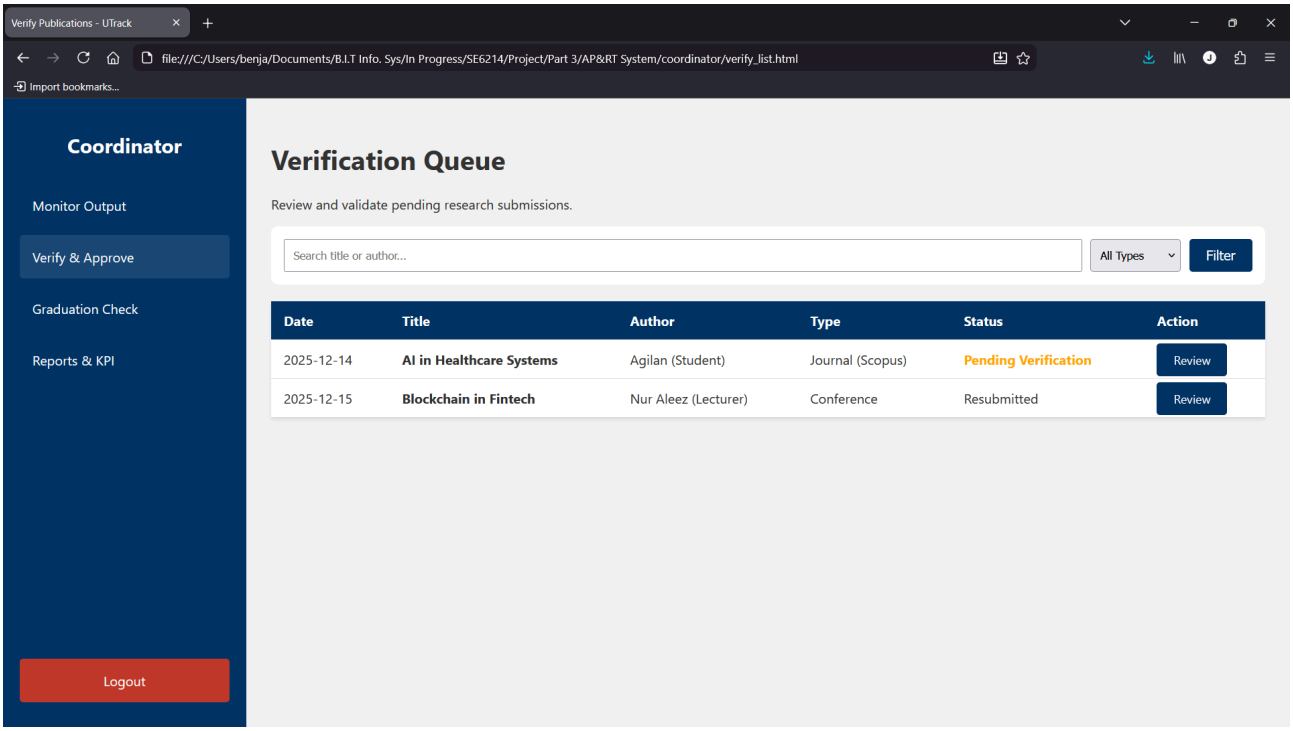


Assign Coordinator

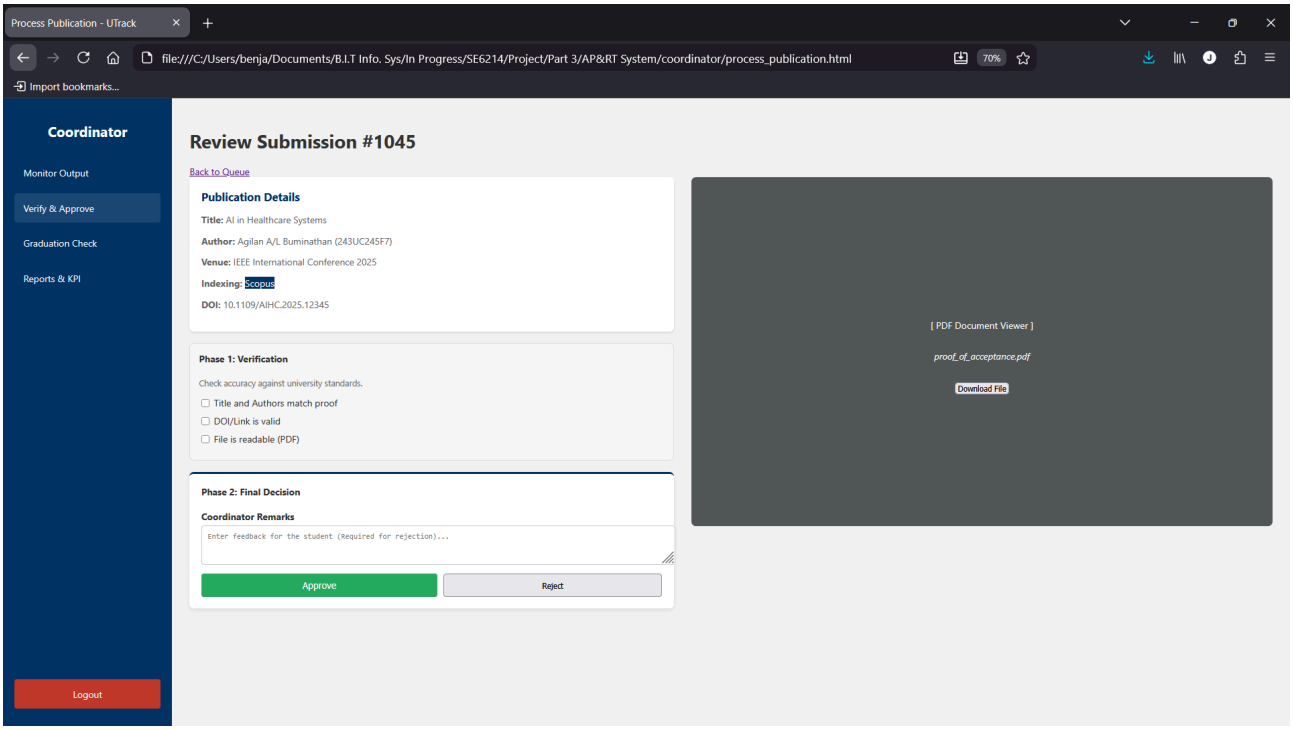


Set publication target & KPIs

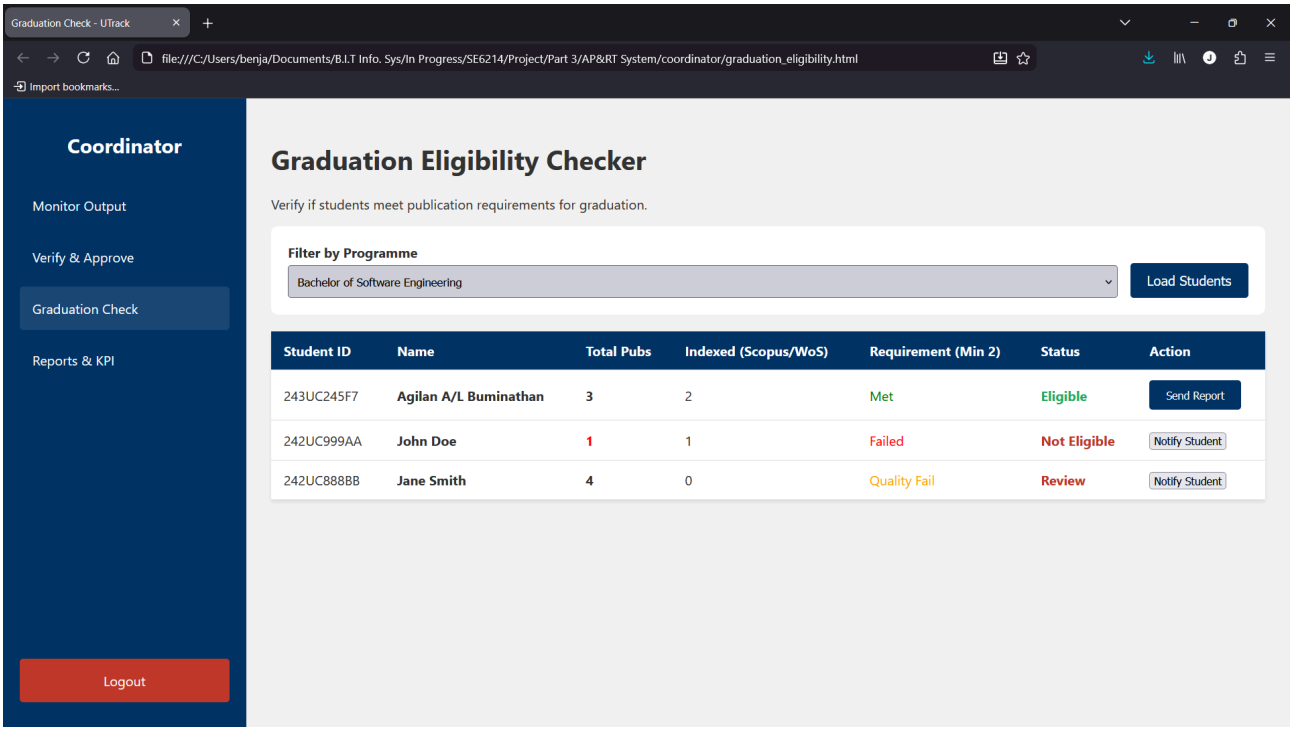
5.3 Subsystem 2 Screens (coordinator)



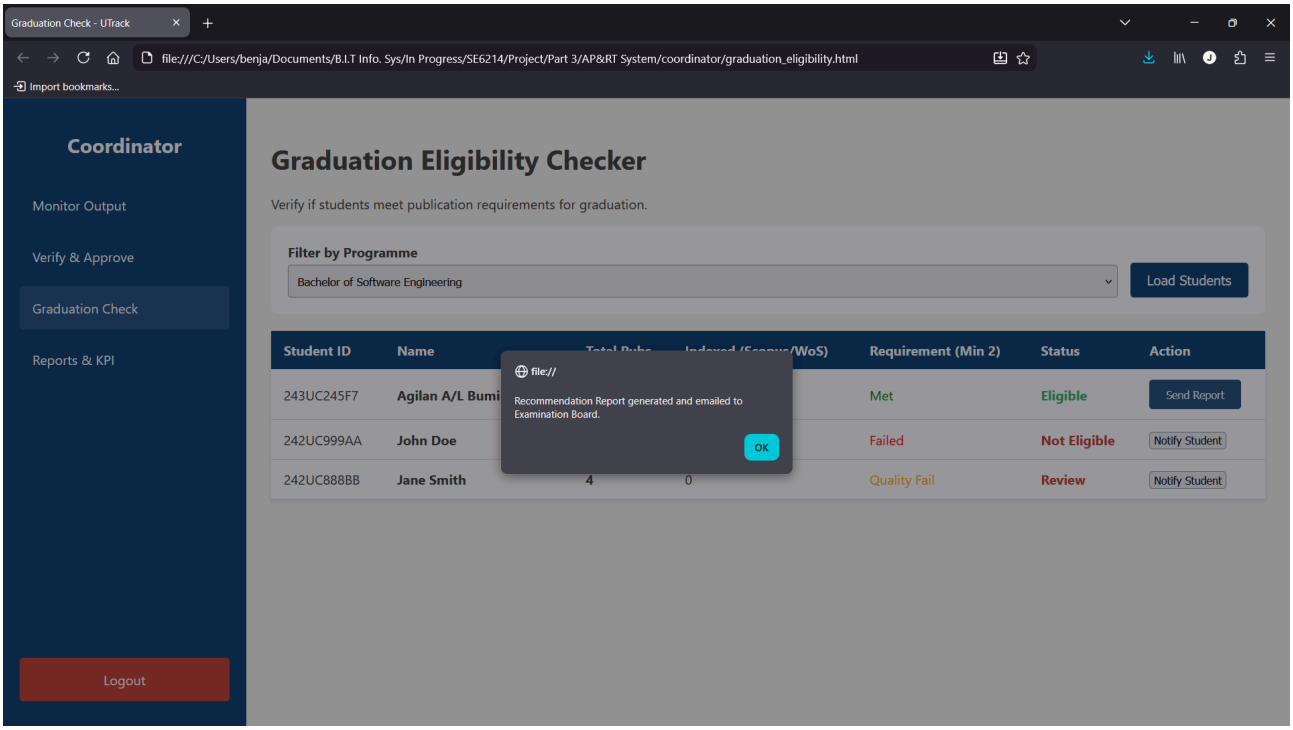
Verification Queue Display



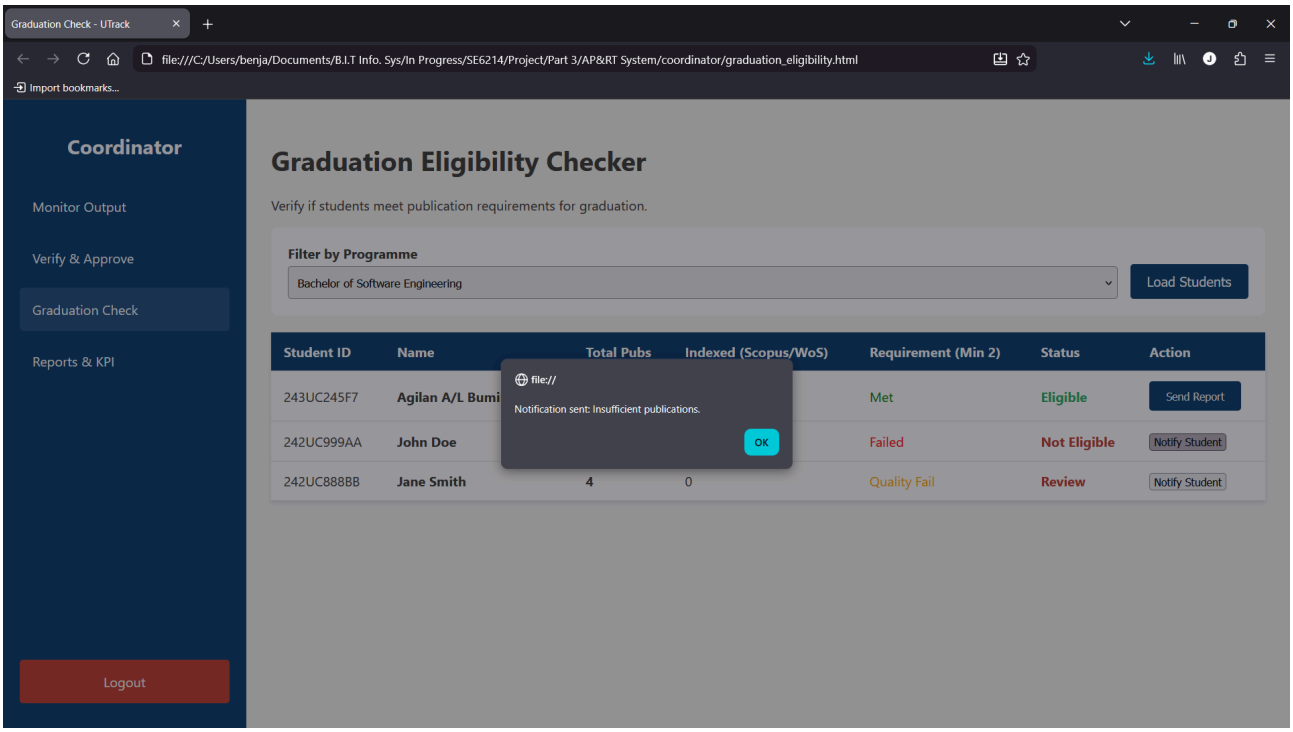
Verification page



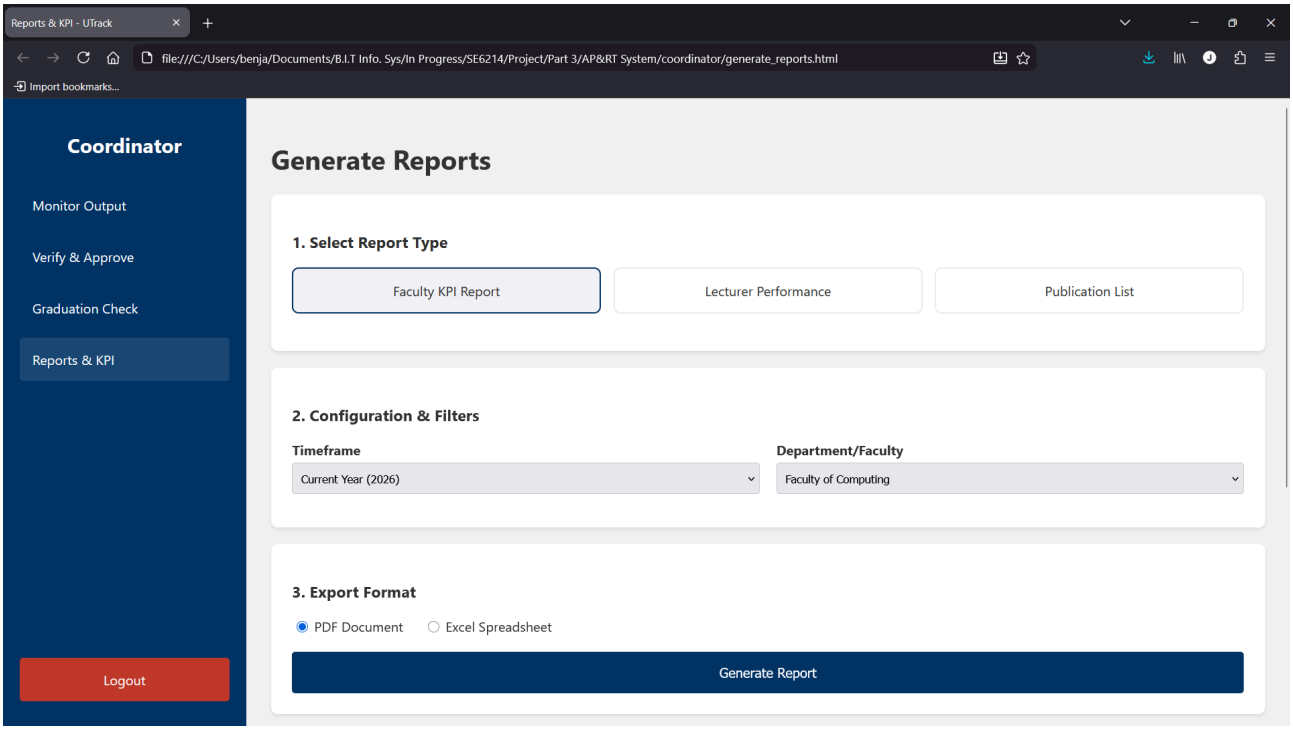
Graduation eligibility checker



If student eligible to graduate

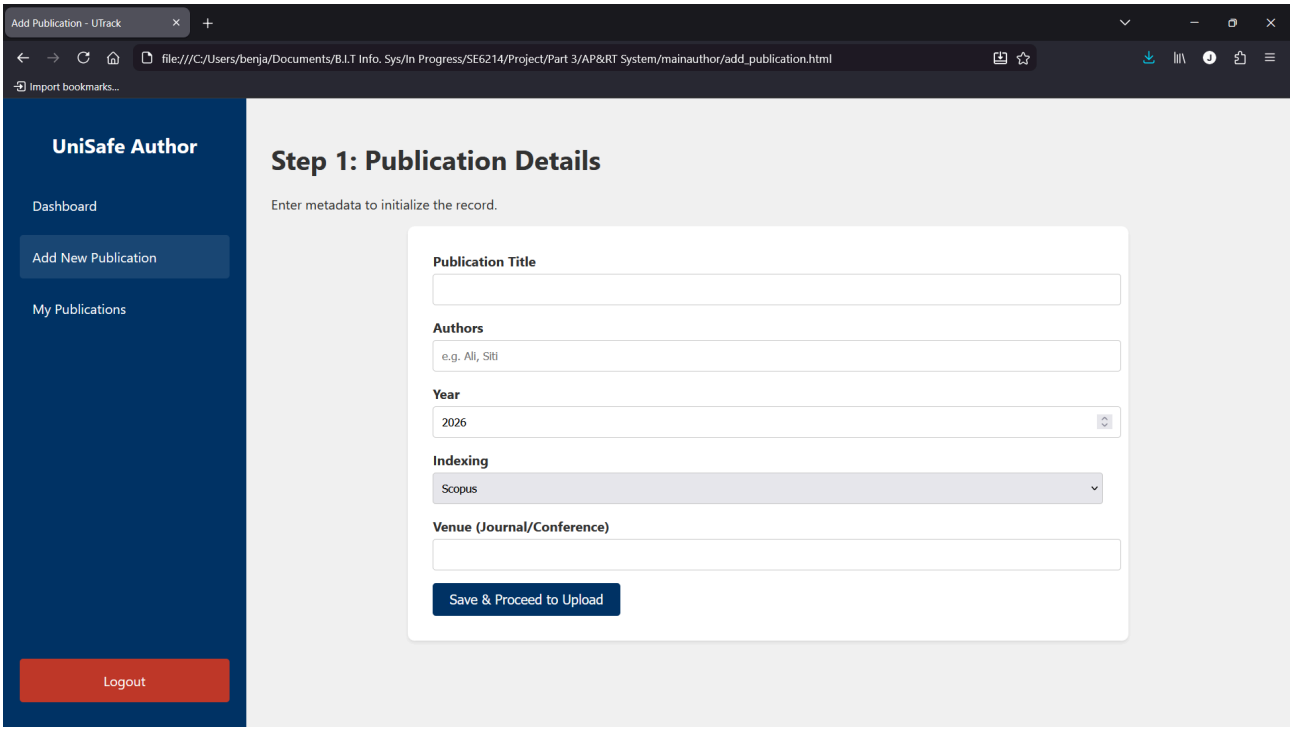


Student not eligible due to insufficient publication

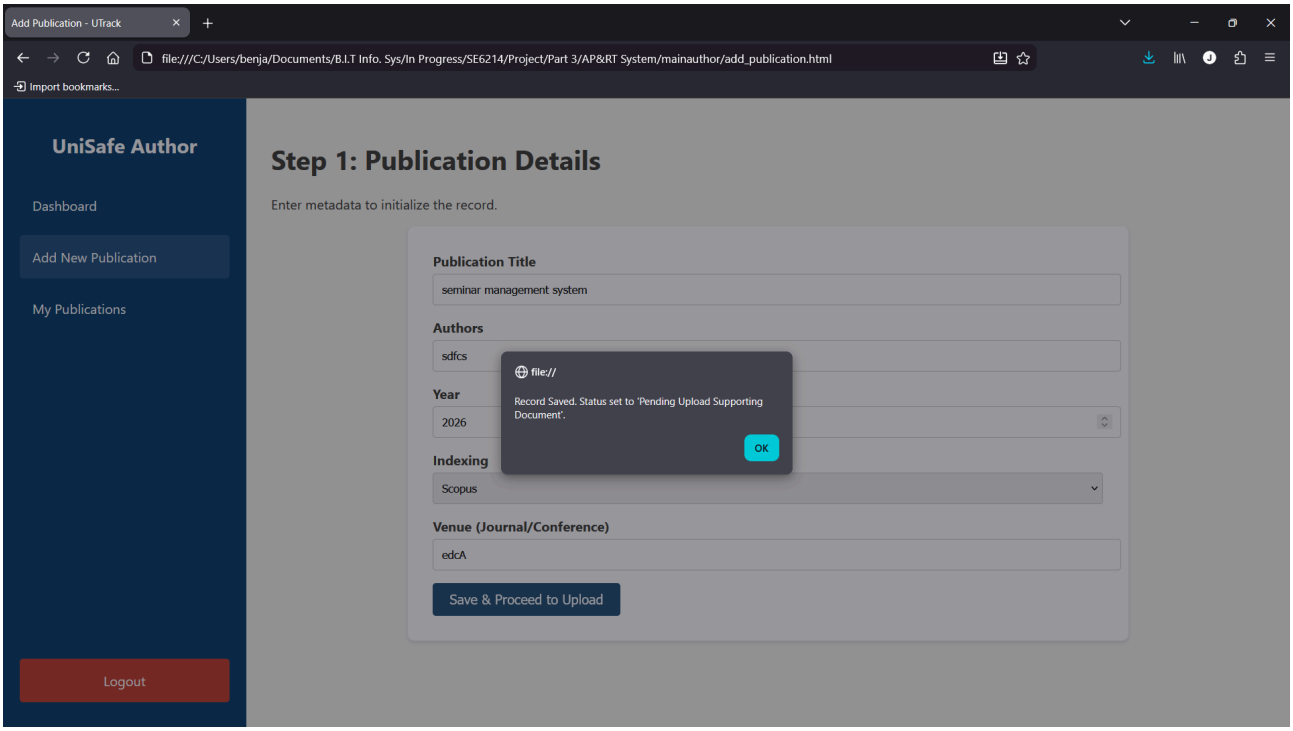


Generate reports

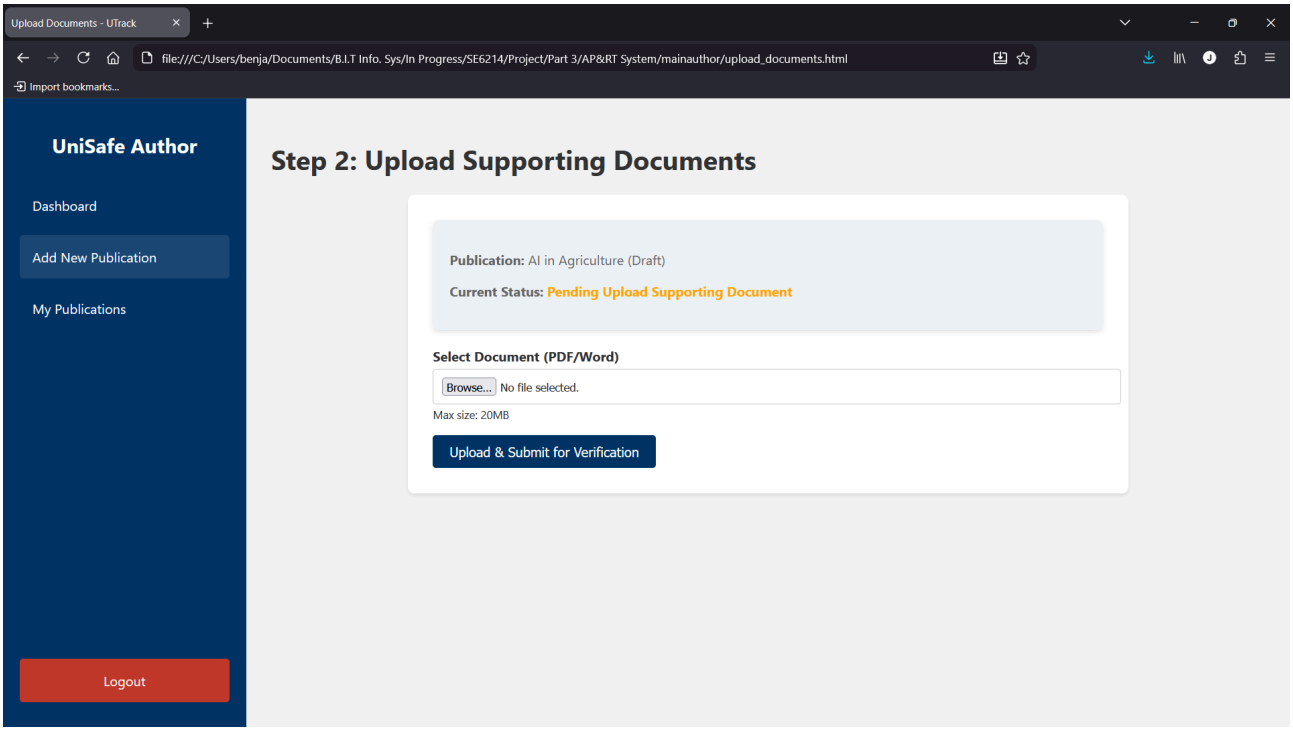
5.4 Subsystem 3 Screens (Main Author)



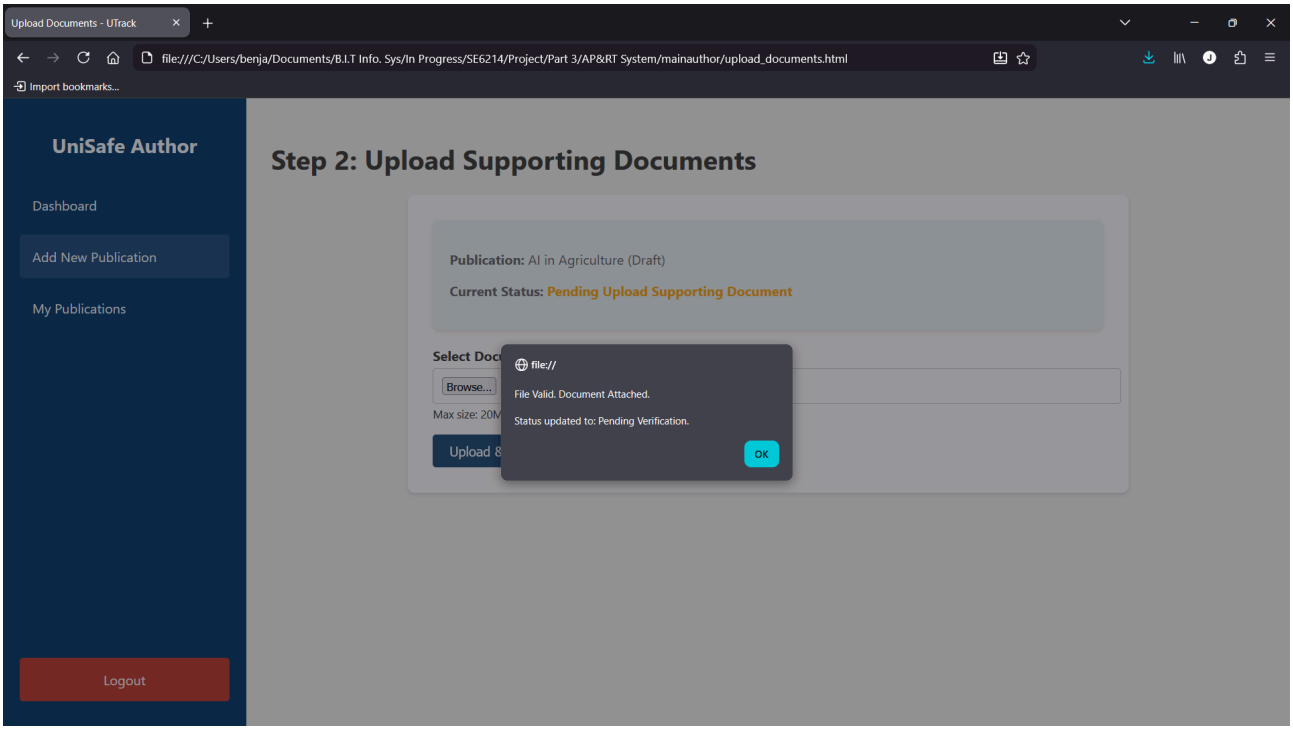
Add publication details



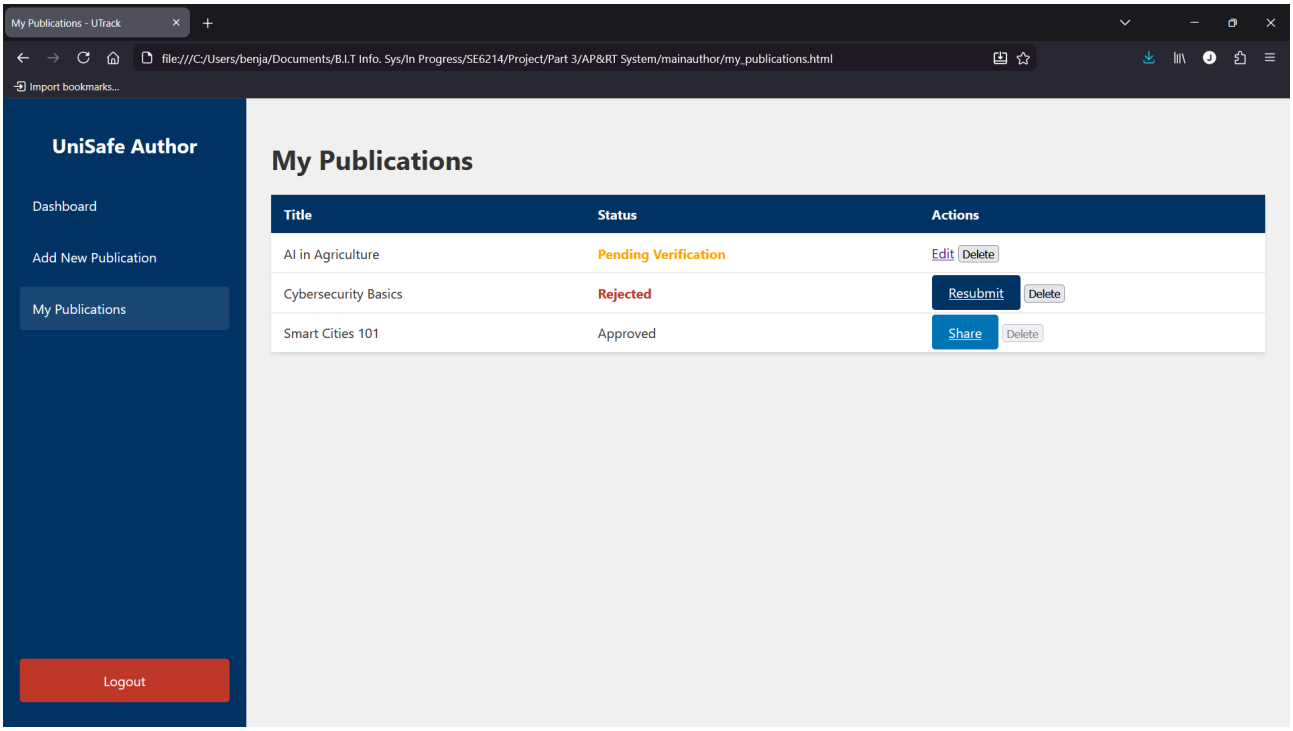
Publication details saved, status changed



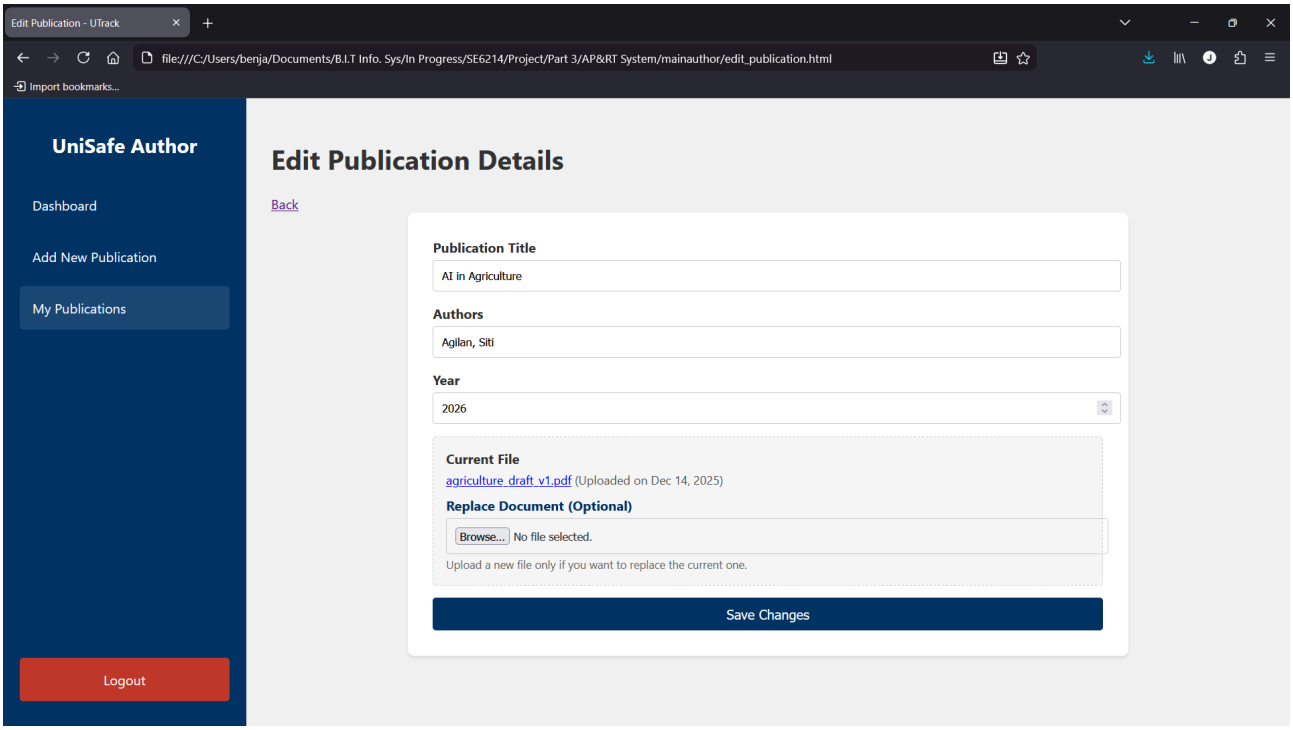
Upload supporting documents



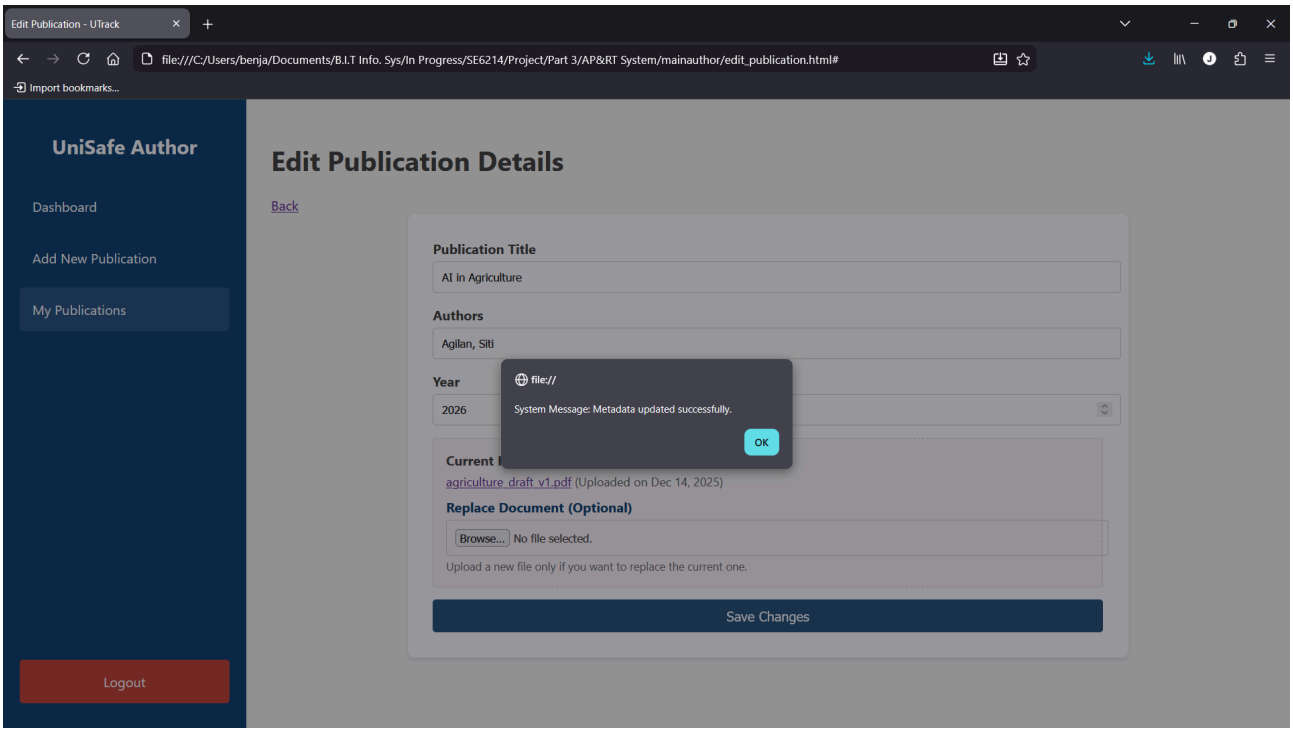
File uploaded with no uploading error, status changed



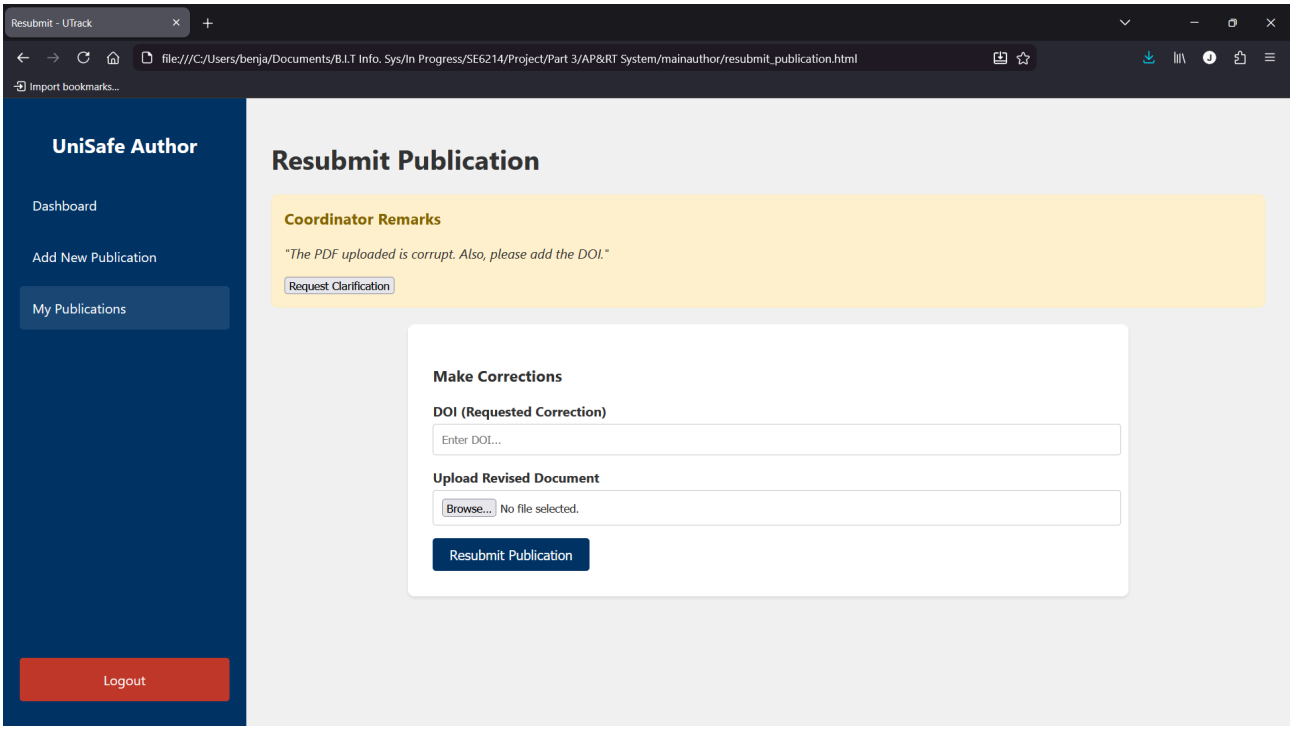
List of publications with status



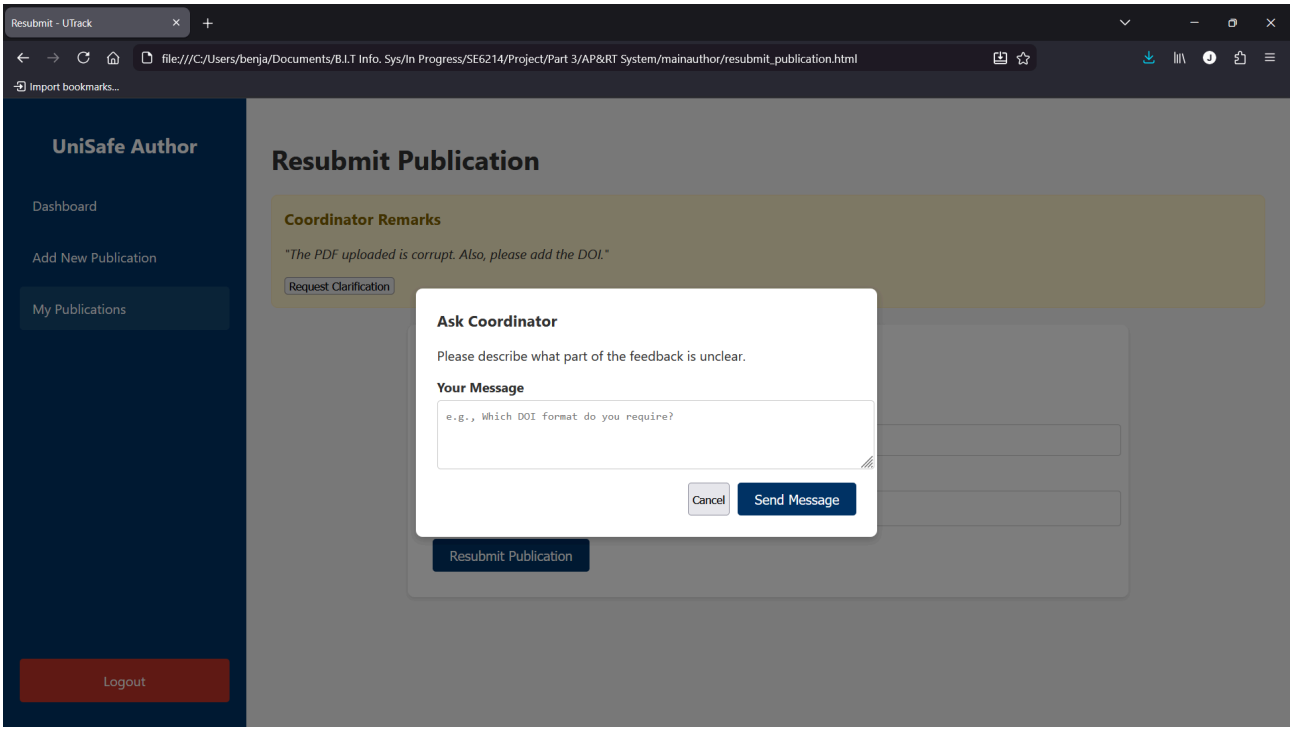
Edit publication



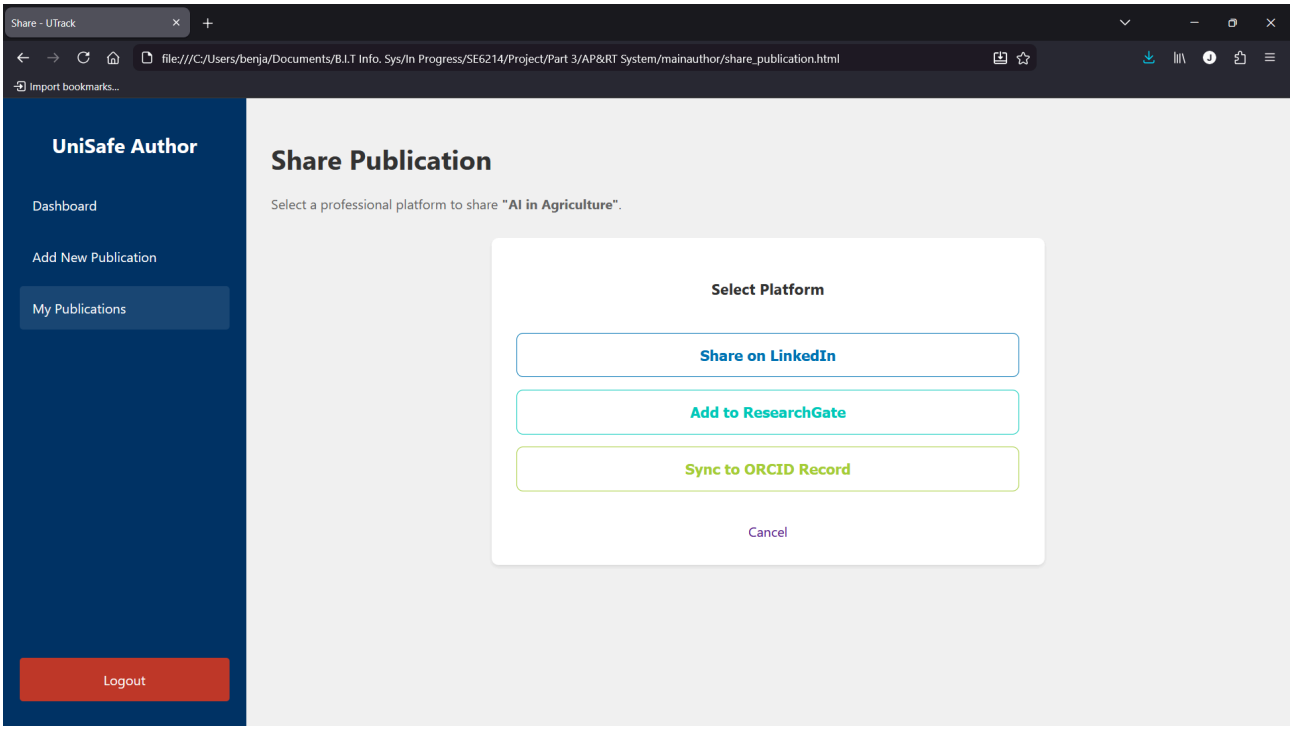
Metadata of the edited publication is updated



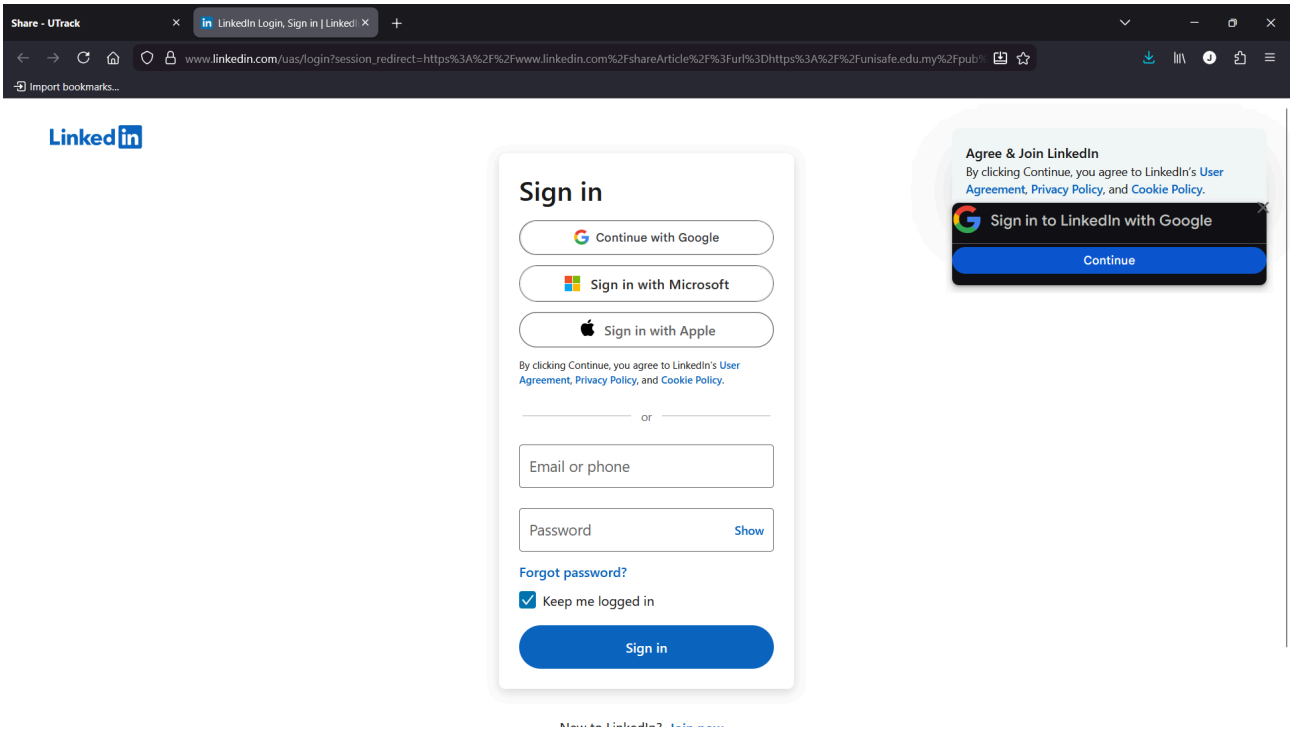
Resubmit publication



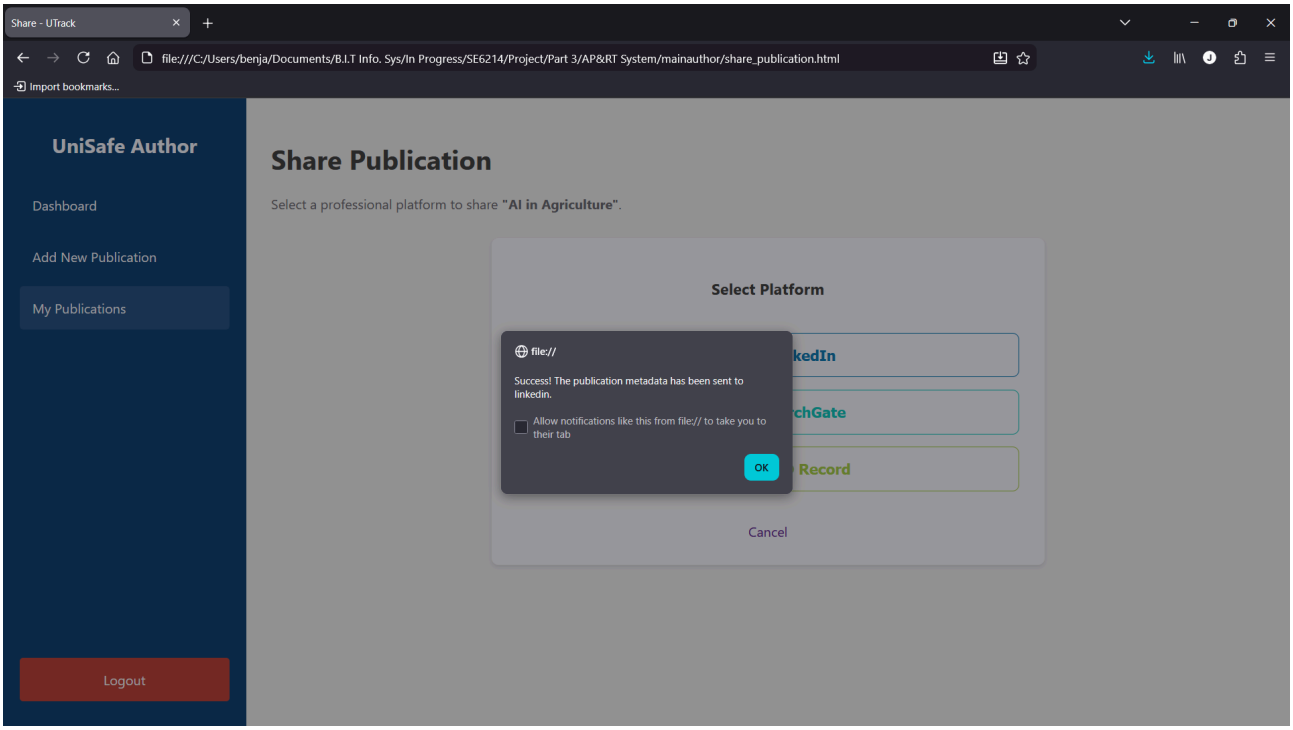
Contacting coordinator for clarification



Share publication to professional platforms

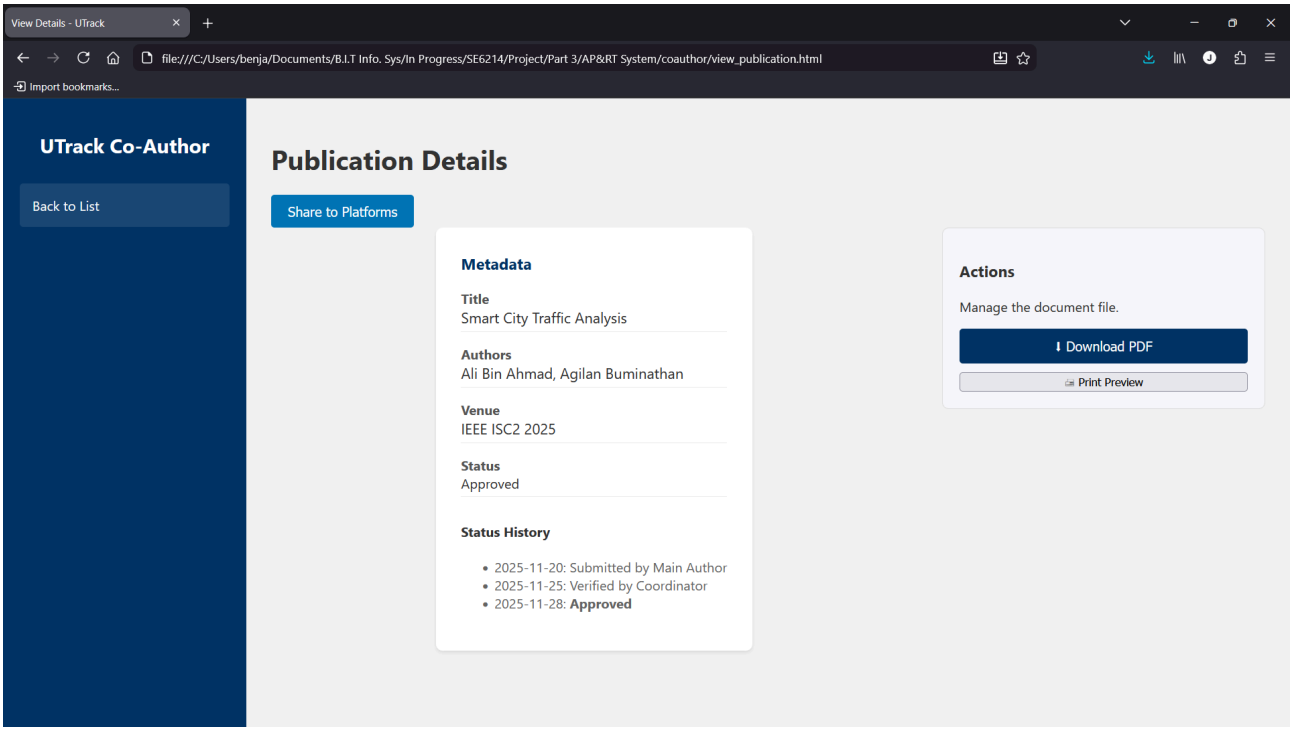


Redirecting to the sharing platform

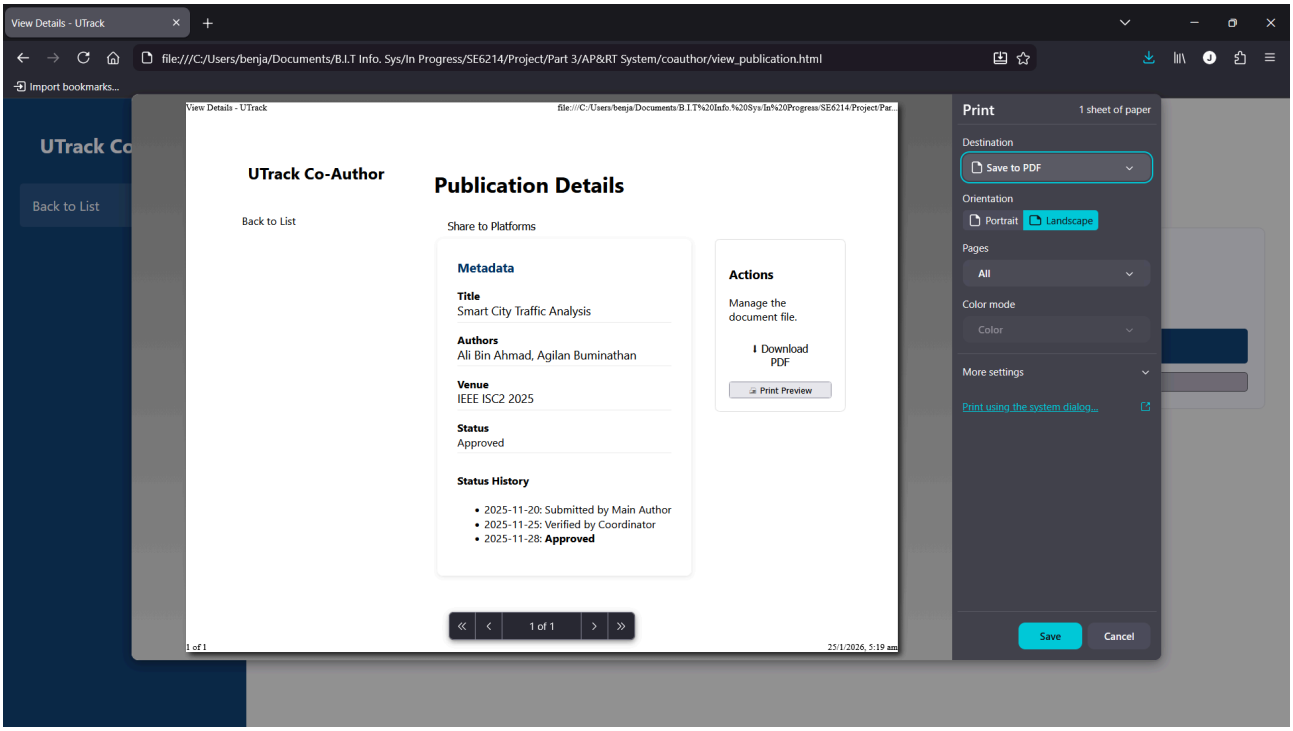


Show successful sharing message

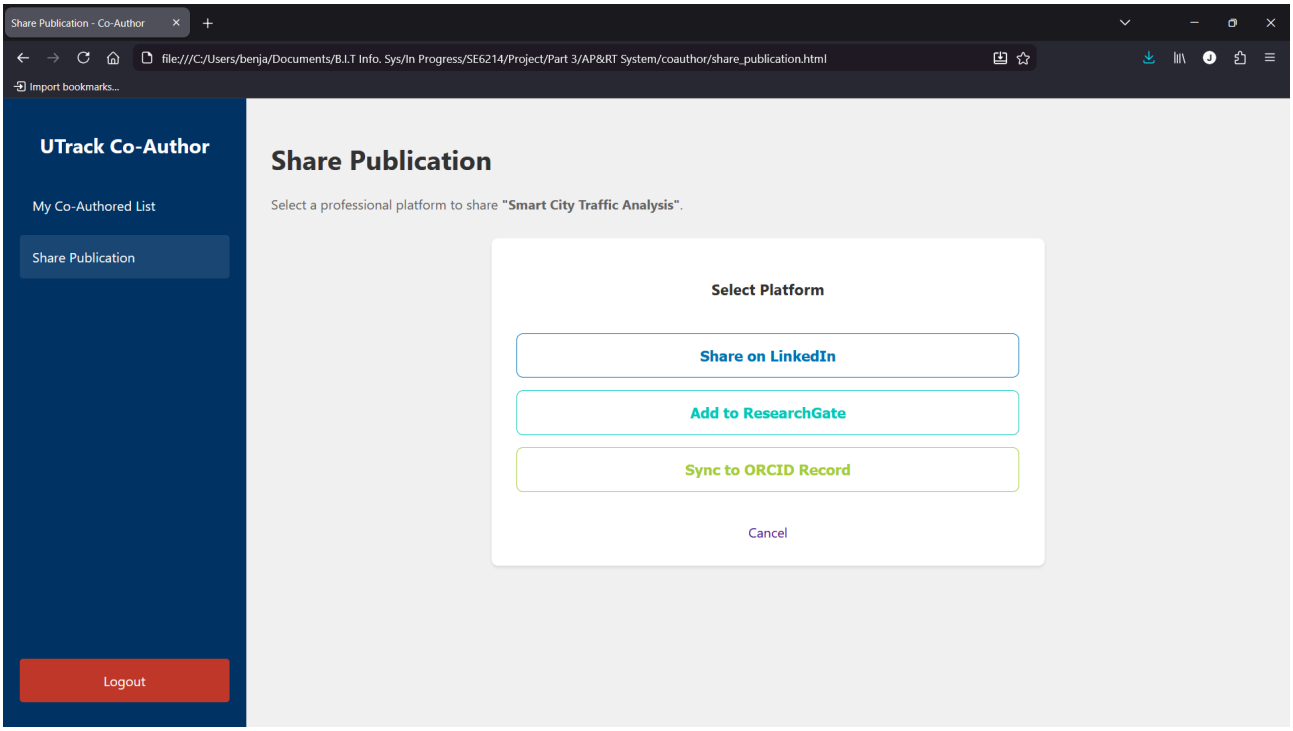
5.5 Subsystem 4 Screens (Co - Author)



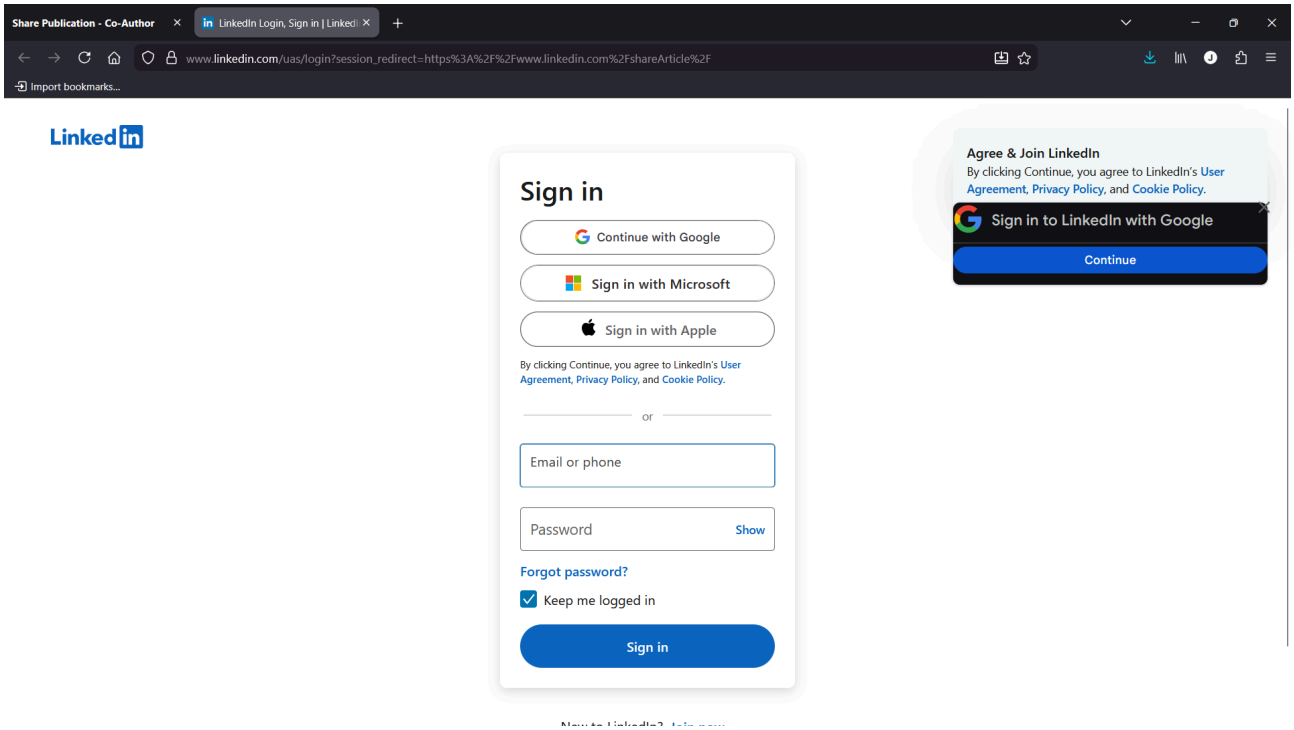
View details of co-authored publications



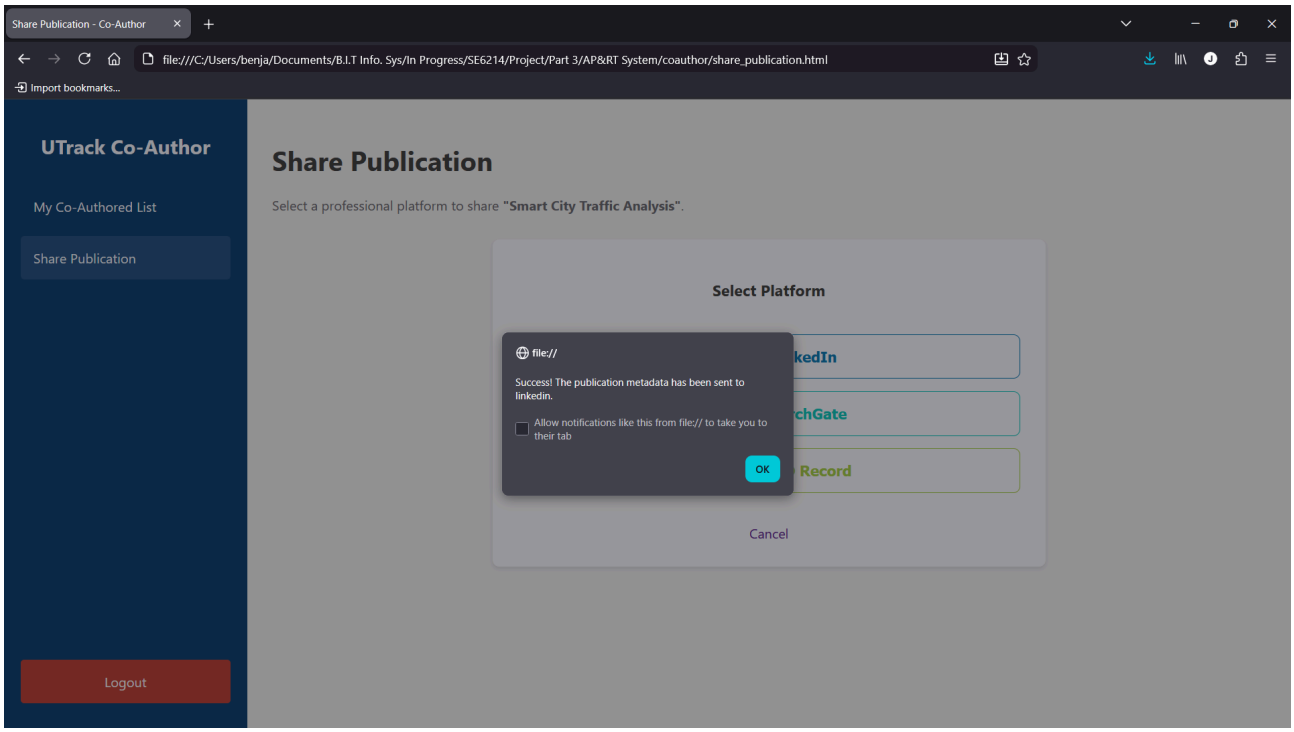
Print preview



Share publications to professional platforms



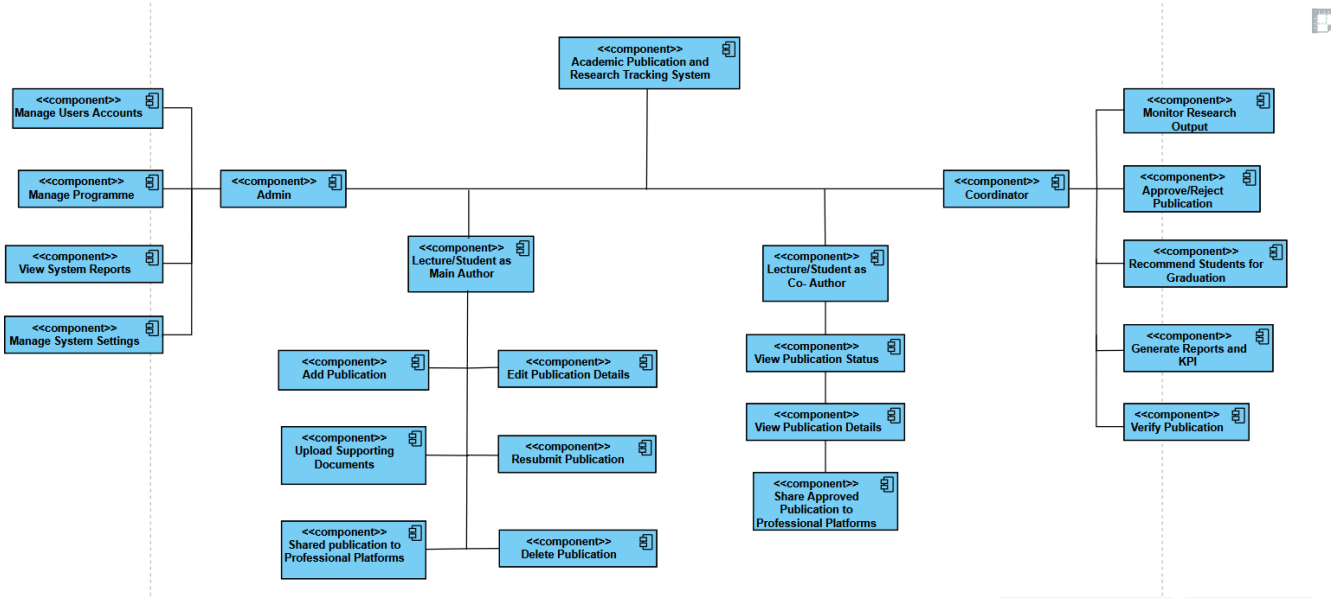
Redirecting to the sharing platform



Show successful sharing message

6 Component Design

6.1 Main Components



Component / Subsystem	Type	Description
Admin Subsystem	Functional Module	Handles user management, system settings configuration, programme management, and system-wide reporting. • Manage Users Accounts • Manage System Settings • Manage Programme • View System Reports
Coordinator Subsystem	Functional Module	Manages publication verification, approval/rejection workflows, graduation eligibility checks, and KPI generation. • Verify publication • Approve/Reject Publication • Eligibility for Student's

		Graduation ● Generate Reports, KPI ● Monitor Research Output
Main Author Subsystem	Functional Module	Enables students/lecturers to create publications, upload documents, edit details, and share approved works. ● Add Publication ● Upload Supporting Documents ● Edit Publication Details ● Delete Publication ● Shared publication to Professional Platforms ● Resubmit Publication (If program coordinator rejects publication)
Co-Author Subsystem	Functional Module	Provides read-only access to publication details, status tracking, and sharing capabilities for contributing authors. ● View Publication Status ● View Comment/Remarks ● Share publication to Professional Platforms
Auth Service	Shared Service	Manages user login, authentication tokens, and role-based security (Login/Auth Portal).
Database Server	Data Component	Centralized storage containing User, Publication, Report, and Notification databases.
File Storage System	Data Component	dedicated storage for physical documents (PDFs, evidence files) uploaded by

		authors.

6.1.1 Subsystem 1 : Admin

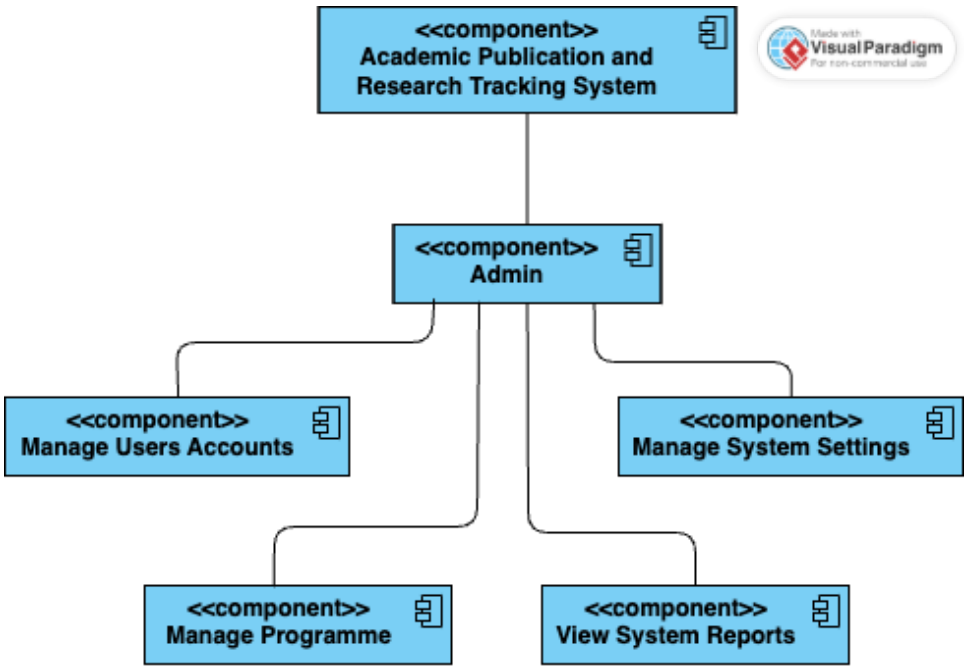
The Admin subsystems include:

Manage Users Accounts : Admin can control user registration by approving new user registrations, edits existing user account and also deactivates account as needed.

Manage System Settings : Admin can configure and customize system behavior, including publication types, workflow rules, user permissions and KPI calculation.

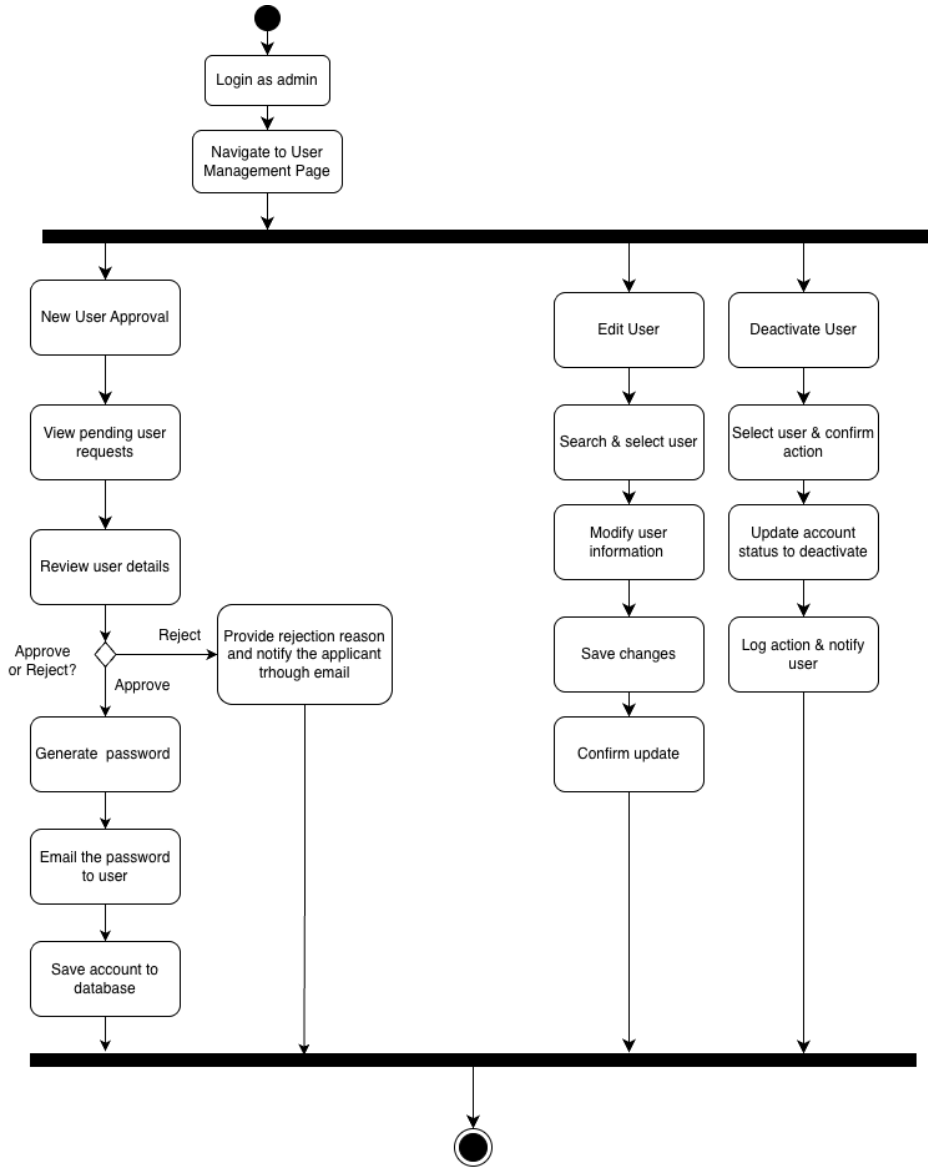
Manage Programme : Admin can oversee academic programme setup, updates requirements, and assigns coordinators to align with university students.

View System Reports : Admin can generates, visualizes, and exports detailed reports on research output, user activity and KPIs for strategic review.



4.4.1 Admin subsystem diagram

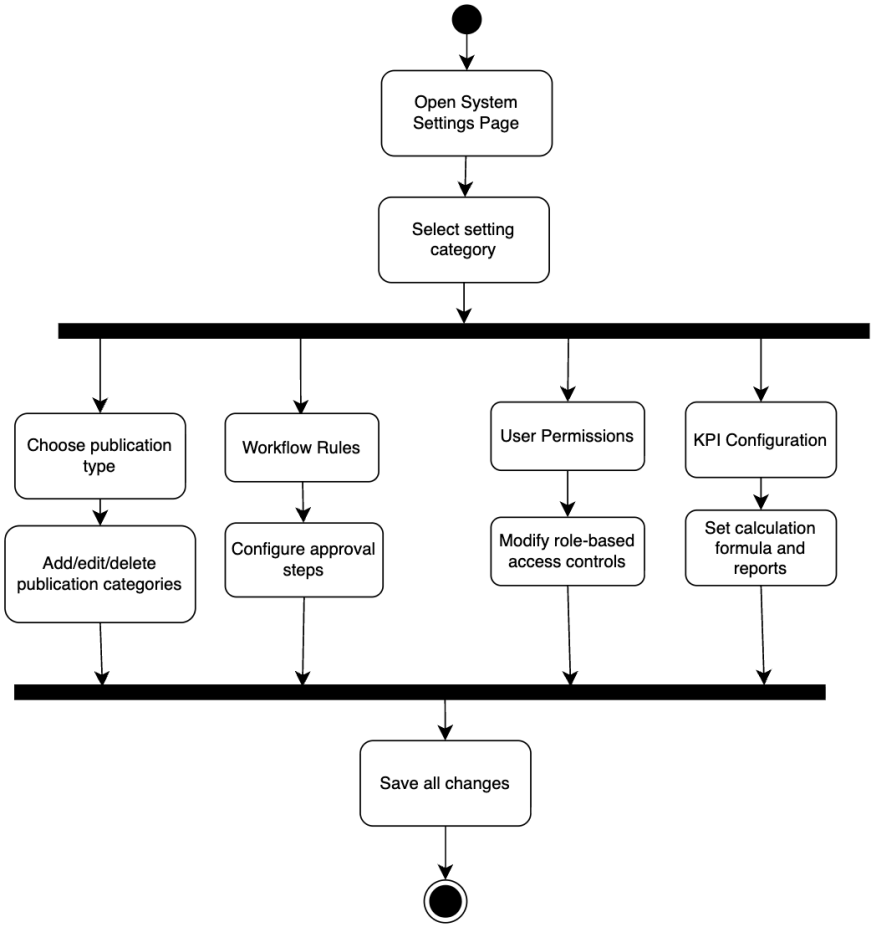
6.1.1.1 Admin Use case 1 : Manage Users Accounts




```
BEGIN

   CALL Login_As_Admin()
   CALL Navigate_To_User_Management_Page()

   SWITCH Action_Selected:
      CASE "New User Approval":
         CALL View_Pending_User_Requests()
         CALL Review_User_Details()

         IF Admin_Decision IS "Approve" THEN
            GENERATE temporary_password
            SEND_EMAIL(user, temporary_password)
```

```
            SAVE_ACCOUNT_TO_DATABASE(user_info)
         ELSE IF Admin_Decision IS "Reject" THEN
            INPUT rejection_reason
            SEND_NOTIFICATION_EMAIL(user, rejection_reason)
         END IF

      CASE "Edit User":
         SEARCH_AND_SELECT_USER()
         MODIFY_USER_INFORMATION()
         SAVE_CHANGES()
         DISPLAY "Update Confirmed"

      CASE "Deactivate User":
         SELECT_USER_AND_CONFIRM_ACTION()
         UPDATE_ACCOUNT_STATUS("Deactivated")
         LOG_ACTION()
         SEND_NOTIFICATION_EMAIL(user, "Account Deactivated")

   END SWITCH

   DISPLAY "Process Completed"
   EXIT WORKFLOW

END
```

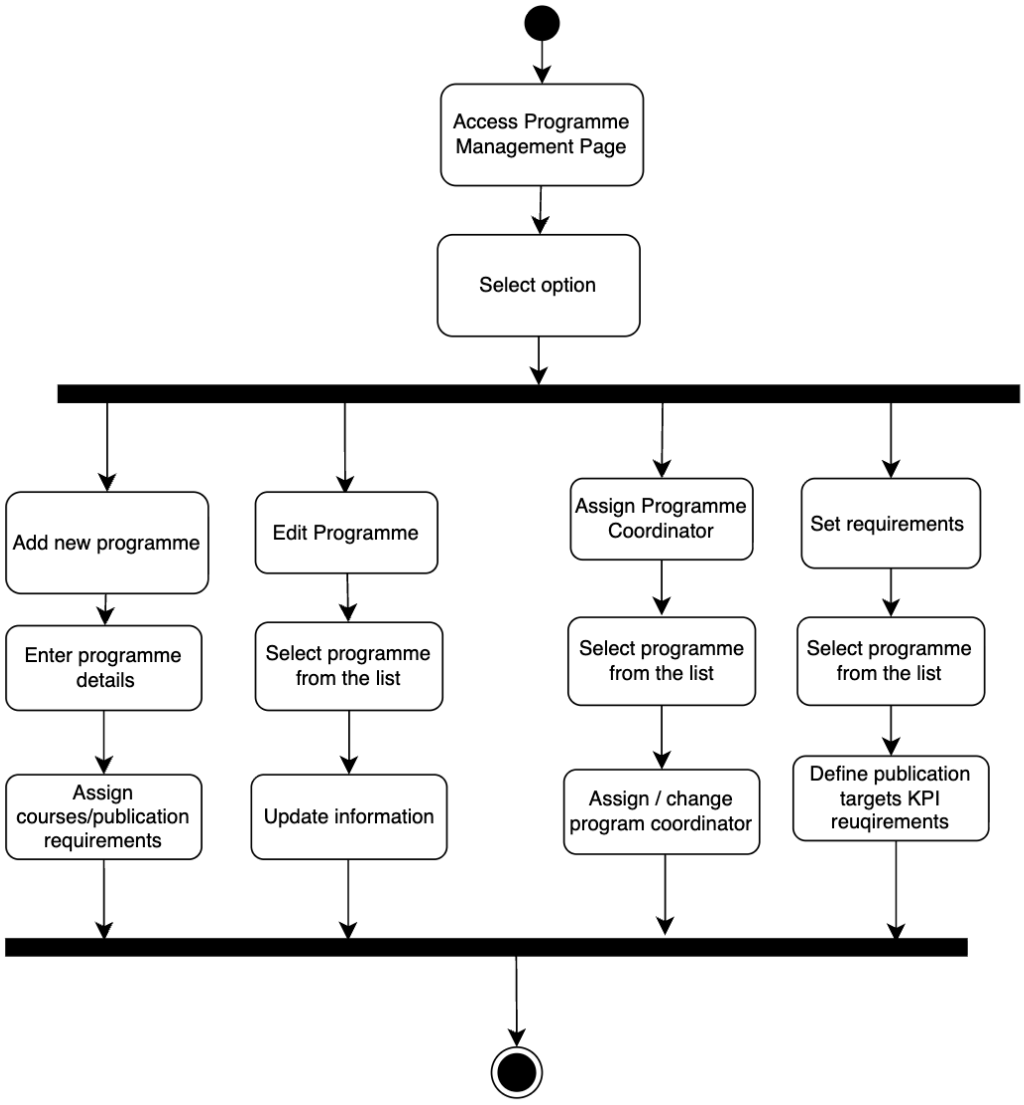
6.1.1.2 Admin Use case 2 : Manage System Settings




```
BEGIN

   CALL Open_System_Settings_Page()
   CALL Select_Setting_Category()

    SWITCH Configuration_Type:

      CASE "Publication Type":
         CALL Choose_Publication_Type()
         CALL Add_Edit_Delete_Publication_Categories()

      CASE "Workflow Rules":
         CALL Workflow_Rules()
         CALL Configure_Approval_Steps()

      CASE "User Permissions":
         CALL User_Permissions()
         CALL Modify_Role_Based_Access_Controls()
```

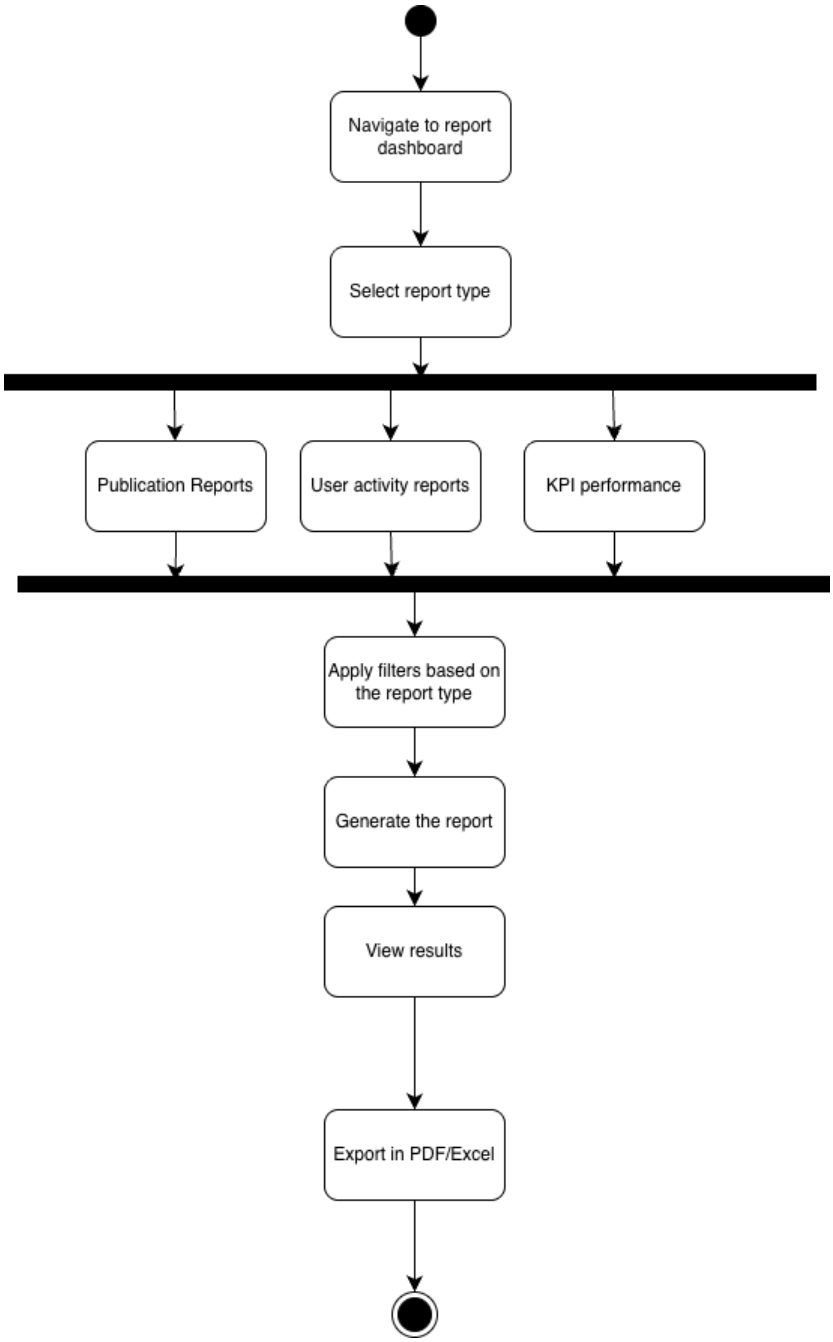
```
        CASE "KPI Configuration":
            CALL KPI_Configuration()
            CALL Set_Calculation_Formula_And_Reports()

    END SWITCH

    CALL Save_All_Changes()

    EXIT WORKFLOW

END
```

6.1.1.3 Admin Use case 3 : Manage Programme



```
BEGIN

   CALL Access_Programme_Management_Page()
   INPUT menu_option

    CHOOSE menu_option:

      CASE "Add New":
         CALL Add_new_programme()
         INPUT programme_details
         EXECUTE Assign_courses_or_publication_requirements()

      CASE "Edit":
         CALL Edit_Programme()
         SELECT target_programme FROM list
         EXECUTE Update_information()

      CASE "Coordinator":
         CALL Assign_Programme_Coordinator()
         SELECT target_programme FROM list
         EXECUTE Assign_or_change_program_coordinator()

      CASE "Requirements":
         CALL Set_requirements()
         SELECT target_programme FROM list
         EXECUTE Define_publication_targets_KPI_requirements()

   END CHOOSE

   EXIT Programme_Management_Page

END
```

6.1.1.4 Admin Use case 4 : View System Reports




```
BEGIN

   CALL Navigate_to_report_dashboard()
   CALL Select_report_type()

   FORK:
      BRANCH A: Publication_Reports
```

```
    BRANCH B: User_activity_reports
    BRANCH C: KPI_performance
  END FORK (Join)

  CALL Apply_filters_based_on_the_report_type()
  CALL Generate_the_report()
  CALL View_results()
  CALL Export_in_PDF_or_Excel()

END
```

6.1.2 Subsystem 2 : Coordinator

The Coordinator subsystems include:

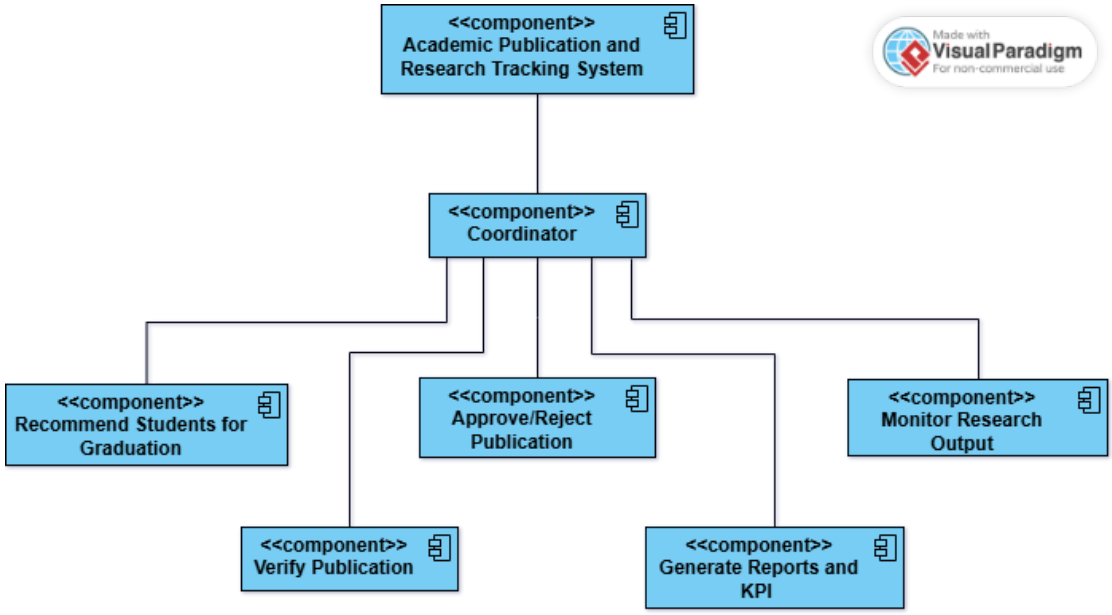
Verify Publication: This use case involves checking the submitted research data for accuracy, ensuring that all metadata (like author names, dates, and citations) is correct before it moves further in the workflow.

Approve/Reject Publication: This is the decision-making stage where a researcher's work is either officially accepted into the system or sent back for revisions, based on the verification results.

Monitor Research Output: This function provides a continuous overview of all research activities, allowing administrators to see which projects are active and how much work is being produced in real-time.

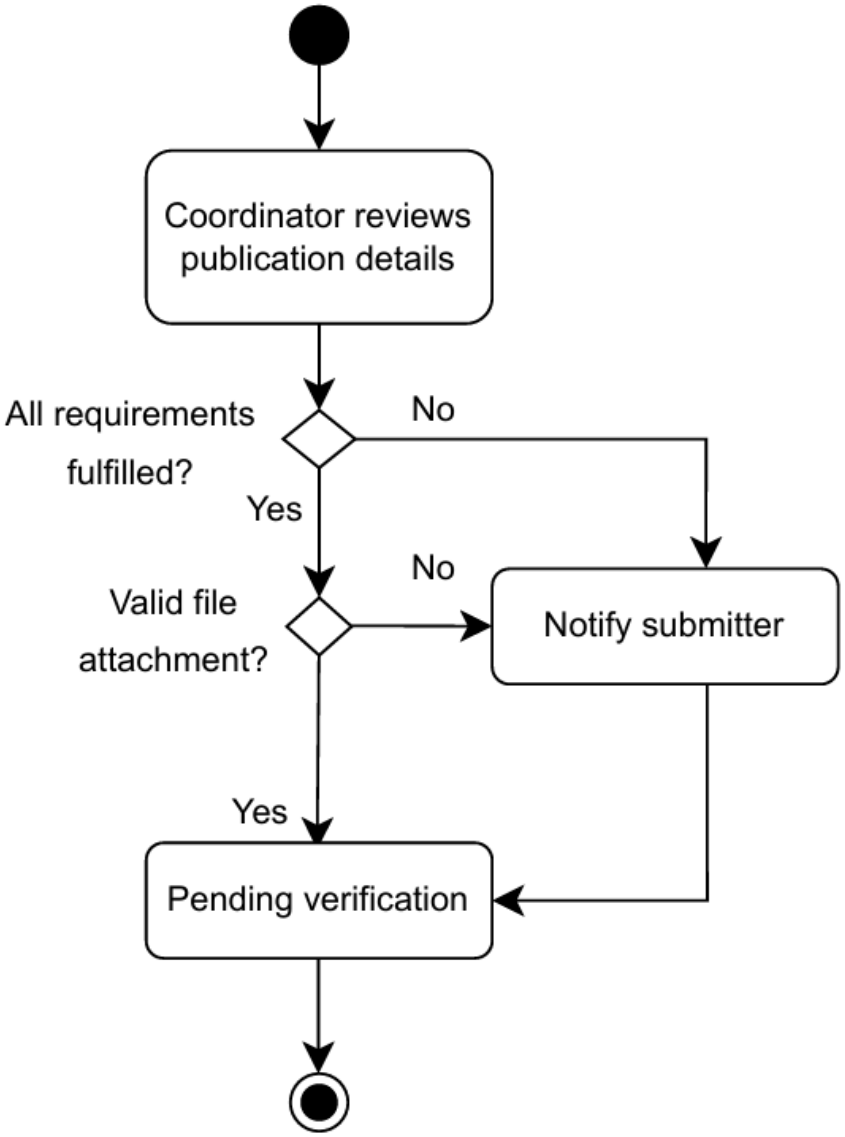
Generate Reports and KPI: This use case processes raw data into formal documents and "Key Performance Indicators," helping the institution measure its success against specific research goals.

Recommend Students for Graduation: This final function tracks a student's research progress and automatically flags them as eligible for graduation once they have met all necessary publication requirements.



6.1.2 Coordinator subsystem diagram

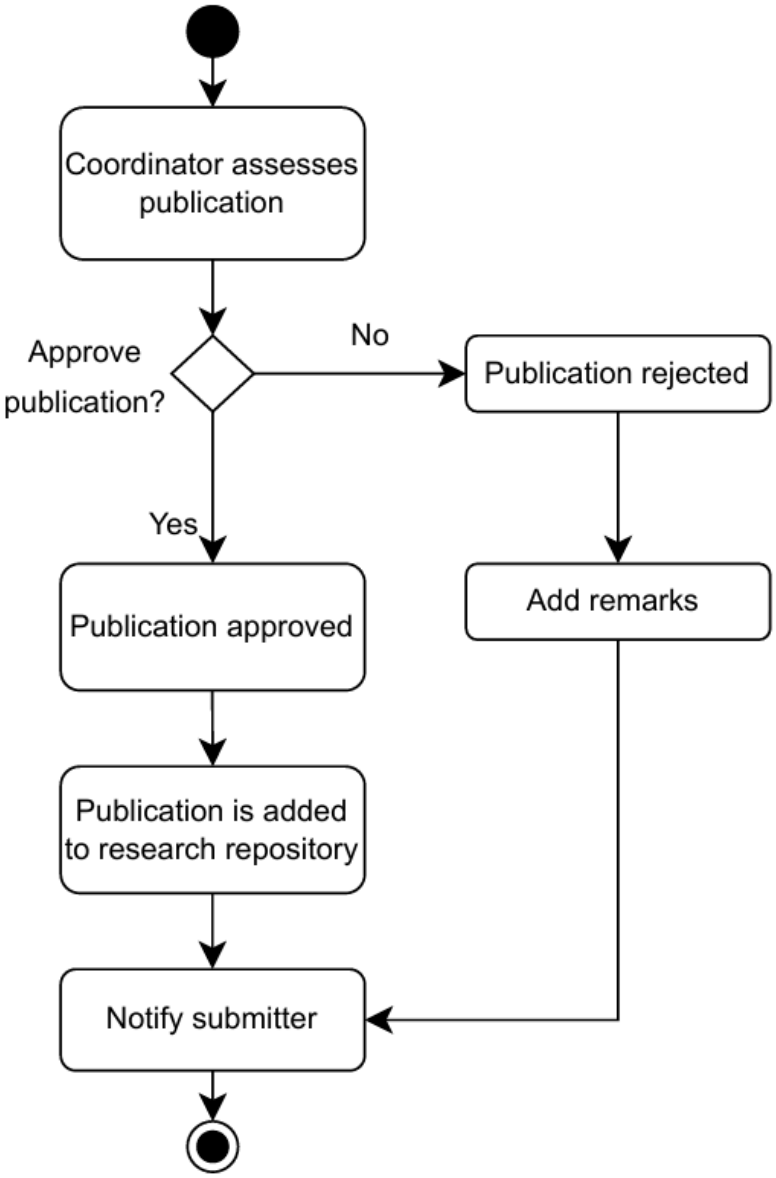
6.1.2.1 Coordinator Use case 1 : Verify publication




```
FUNCTION reviewPublication(publicationId, coordinatorId)
    publication = getPublicationDetails(publicationId)
    requirements = getPublicationRequirements()

    IF checkRequirements(publication, requirements) == TRUE
        IF validateFileAttachment(publication) == TRUE
            SET publication.status = "pending_verification"
            UPDATE publication IN database
            RETURN "Publication pending verification"
        ELSE
            notification = createNotification(
```

```
            recipient: publication.submitterId,
            message: "Invalid file attachment",
            type: "error"
        )
        sendNotification(notification)
        RETURN "Invalid attachment - submitter notified"
    END IF
  ELSE
    notification = createNotification(
        recipient: publication.submitterId,
        message: "Requirements not fulfilled",
        type: "rejection"
    )
    sendNotification(notification)
    RETURN "Requirements not met - submitter notified"
  END IF
END FUNCTION
```

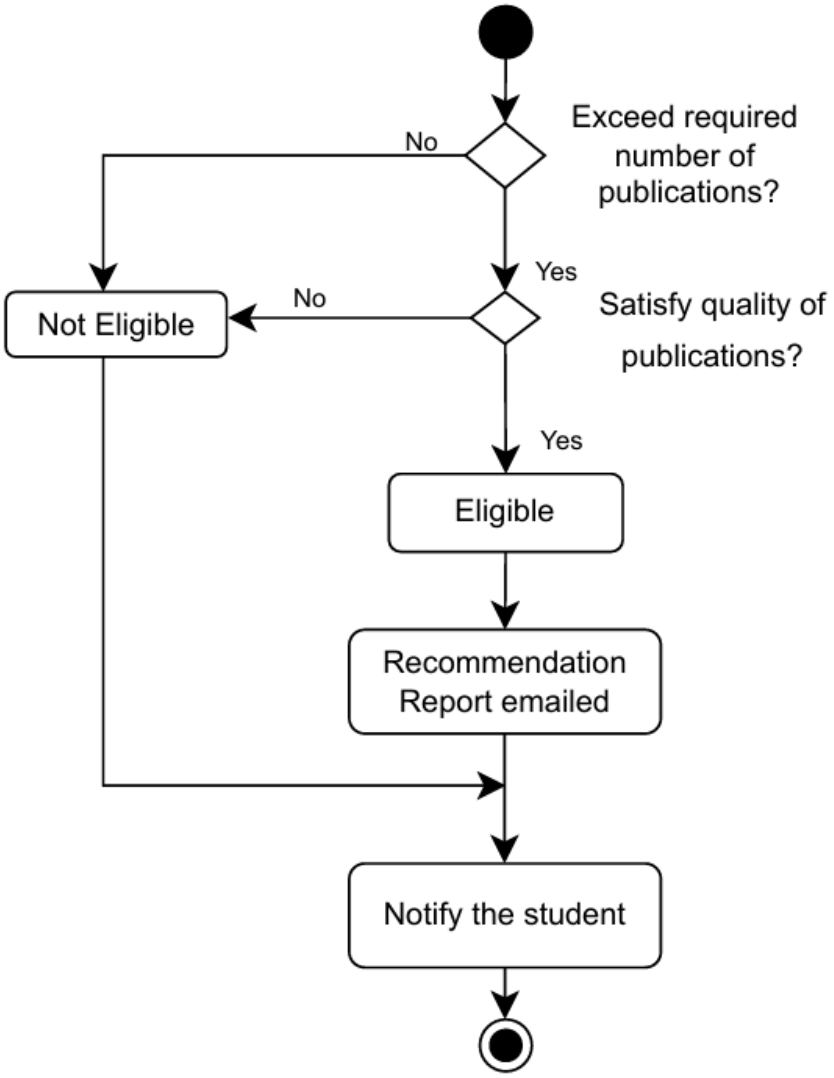
6.1.2.2 Coordinator Use case 2 : Approve/Reject Publication




```
FUNCTION assessPublication(publicationId, coordinatorId)
    publication = getPublicationDetails(publicationId)
    assessment = getAssessmentCriteria()

    DISPLAY publication.details TO coordinator
    DISPLAY "Approve this publication? (Yes/No)"
    coordinatorDecision = GET coordinatorInput()

    IF coordinatorDecision == "Yes"
        publication.status = "approved"
        publication.approvalDate = CURRENT_DATE()
```

```
      publication.approvedBy = coordinatorId

      addToResearchRepository(publication)

      notification = createNotification(
         recipient: publication.submitterId,
         message: "Publication approved and added to repository",
         type: "approval"
      )
      sendNotification(notification)

      UPDATE publication IN database
      RETURN "Publication approved and notification sent"

   ELSE IF coordinatorDecision == "No"
      publication.status = "rejected"
      publication.rejectionDate = CURRENT_DATE()
      publication.rejectedBy = coordinatorId

      DISPLAY "Enter rejection remarks:"
      remarks = GET coordinatorInput()
      publication.rejectionRemarks = remarks

      UPDATE publication IN database

      notification = createNotification(
         recipient: publication.submitterId,
         message: "Publication rejected. Remarks: " + remarks,
         type: "rejection"
      )
      sendNotification(notification)

      RETURN "Publication rejected with remarks"
   END IF
END FUNCTION
```

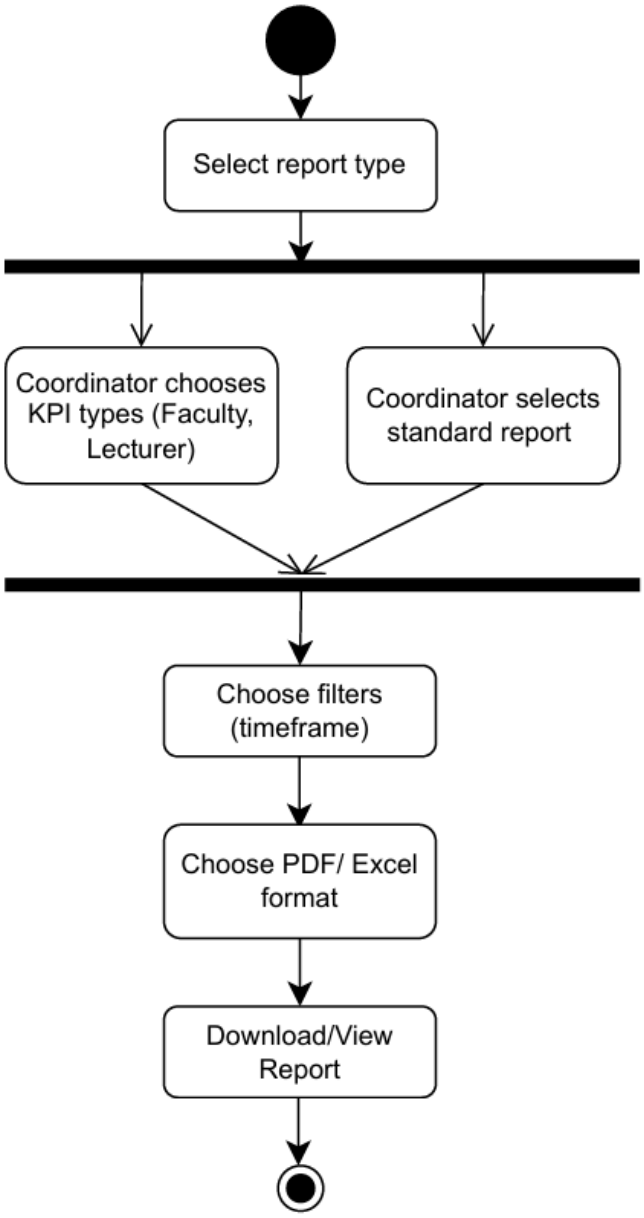
6.1.2.3 Coordinator Use case 3 : Recommend Students for Graduation




```
FUNCTION checkStudentEligibility(studentId, programId)
    student = getStudentDetails(studentId)
    program = getProgramRequirements(programId)

    studentPublications = getStudentPublications(studentId)

    // Check quantity requirement
    IF studentPublications.count < program.requiredPublicationCount
        RETURN "Not Eligible - Insufficient publications"
    END IF
```

```
    // Check quality requirement
    eligibleCount = 0
    FOR EACH publication IN studentPublications
        IF publication.qualityScore >= program.minimumQualityScore
            AND publication.type IN program.allowedPublicationTypes
            eligibleCount = eligibleCount + 1
        END IF
    END FOR

    IF eligibleCount >= program.requiredPublicationCount
        // Student is eligible
        report = generateRecommendationReport(student, program, studentPublications)

        notification = createNotification(
            recipient: studentId,
            message: "Eligible for graduation. Recommendation report generated.",
            type: "eligibility"
        )

        emailReportToStudent(report, student.email)
        sendNotification(notification)

        RETURN "Eligible - Report emailed to student"
    ELSE
        RETURN "Not Eligible - Quality requirements not met"
    END IF
END FUNCTION
```

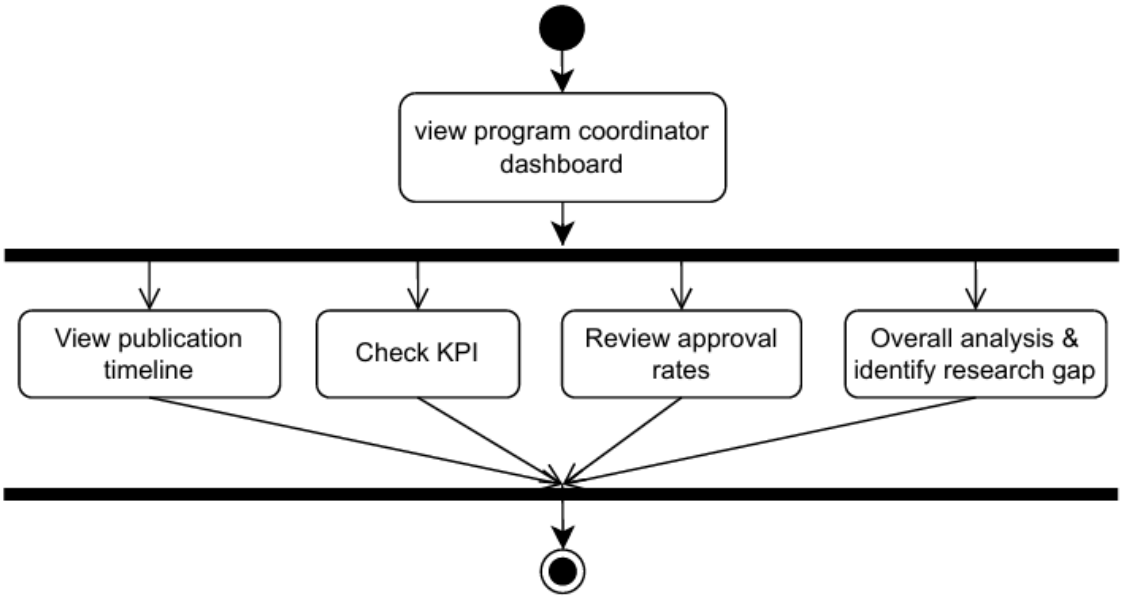
6.1.2.4 Coordinator Use case 4 : Generate Reports & KPI




```
FUNCTION generateReport(coordinatorId)
    DISPLAY "Select Report Type:"
    DISPLAY "1. Faculty Report"
    DISPLAY "2. Lecturer Report"
    DISPLAY "3. Standard Report"

    reportType = GET coordinatorSelection()

    IF reportType == "1" OR reportType == "2"
```

```
        kpiTypes = getKPITypesForRole(reportType)
        DISPLAY "Available KPI Types: " + kpiTypes
        selectedKPIs = GET coordinatorSelection(kpiTypes)
    END IF

    // Get timeframe filters
    DISPLAY "Select Timeframe:"
    DISPLAY "1. Last Month"
    DISPLAY "2. Last Quarter"
    DISPLAY "3. Last Year"
    DISPLAY "4. Custom Range"

    timeframe = GET coordinatorSelection()

    IF timeframe == "4"
        startDate = GET dateInput("Start Date")
        endDate = GET dateInput("End Date")
    ELSE
        dateRange = calculateDateRange(timeframe)
        startDate = dateRange.start
        endDate = dateRange.end
    END IF

    // Format selection
    DISPLAY "Select Output Format:"
    DISPLAY "1. PDF"
    DISPLAY "2. Excel"

    outputFormat = GET coordinatorSelection()

    // Generate report
    reportData = fetchReportData(
        reportType: reportType,
        kpis: selectedKPIs,
        startDate: startDate,
        endDate: endDate
    )

    reportFile = formatReport(reportData, outputFormat)

    DISPLAY "Report generated successfully!"
    DISPLAY "1. Download Report"
    DISPLAY "2. View Report"

    action = GET coordinatorSelection()

    IF action == "1"
        downloadFile(reportFile)
    ELSE IF action == "2"
        displayReportPreview(reportFile)
    END IF

    RETURN "Report generation completed"
END FUNCTION
```

6.1.2.5 Coordinator Use case 5 : Monitor Research Output




```
FUNCTION displayCoordinatorDashboard(coordinatorId, programId)
    // Load dashboard view
    dashboard = initializeDashboard()

    // 1. View publication timeline
    publicationTimeline = getPublicationTimeline(programId)
    dashboard.addComponent("timeline", publicationTimeline)

    // 2. Check KPIs
    kpiData = getKPIForProgram(programId)
    dashboard.addComponent("kpi", kpiData)

    // 3. Review approval rates
    approvalStats = calculateApprovalRates(programId)
    dashboard.addComponent("approval_rates", approvalStats)

    // 4. Overall analysis and research gap identification
    analysis = performResearchAnalysis(programId)
    researchGaps = identifyResearchGaps(analysis)
    dashboard.addComponent("analysis", analysis)
    dashboard.addComponent("research_gaps", researchGaps)

    // Display dashboard
    renderDashboard(dashboard)

    // Interactive elements
    WHILE dashboard.isOpen()
        userAction = waitForUserInteraction()

        IF userAction == "filter_timeline"
            applyTimelineFilter(getUserFilter())
```

```
      ELSE IF userAction == "export_kpi"
         exportData(kpiData, "kpi_report.csv")

      ELSE IF userAction == "drilldown_approval"
         showDetailedApprovalData(getSelectedPeriod())

      ELSE IF userAction == "address_research_gap"
         gap = getSelectedResearchGap()
         initiateResearchInitiative(gap)
      END IF
   END WHILE

   RETURN "Dashboard session ended"
END FUNCTION
```

6.1.3 Subsystem 3 :Lecturer/Student as Main-author

The Main Author subsystems include:

Add Publication: This use case allows the lecturer or student to submit a new research publication into the system by entering key details such as title, authors, year, and venue, which then awaits supporting document upload.

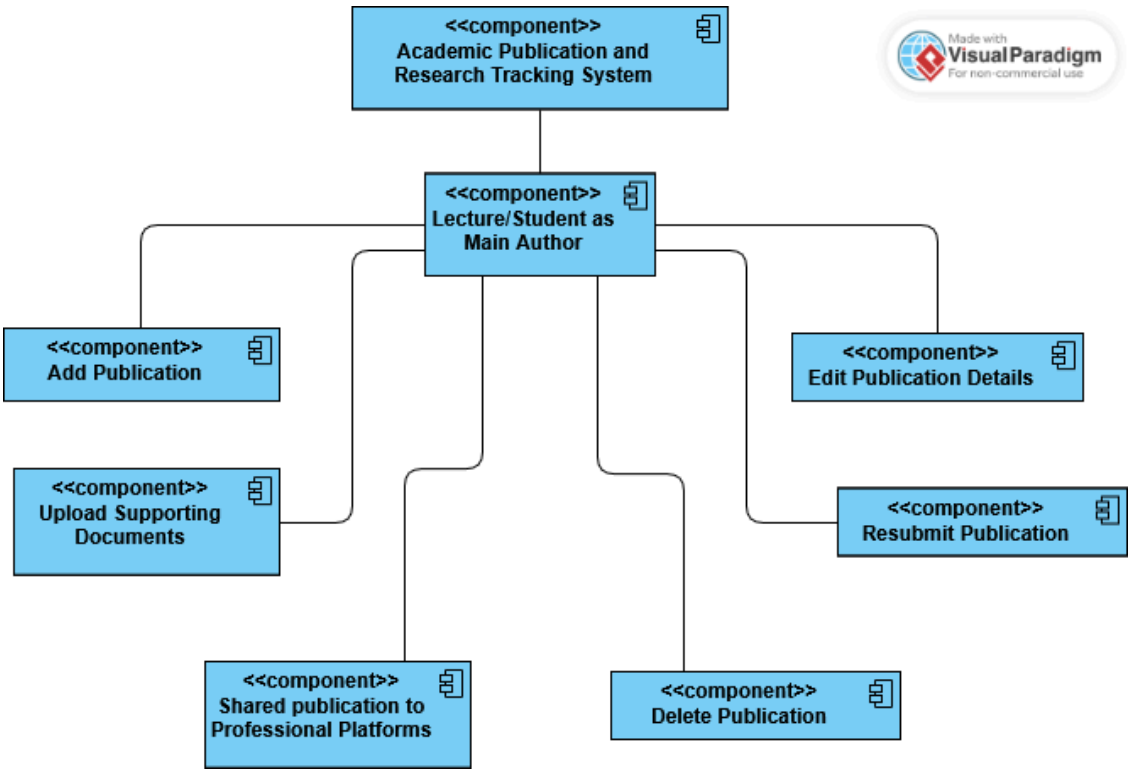
Upload Supporting Documents: This function enables the main author to attach required files such as PDFs, acceptance letters, or DOI links to complete the submission process and move it to verification.

Edit Publication: This allows modifications to publication information before it is verified, ensuring data accuracy and completeness during the review phase.

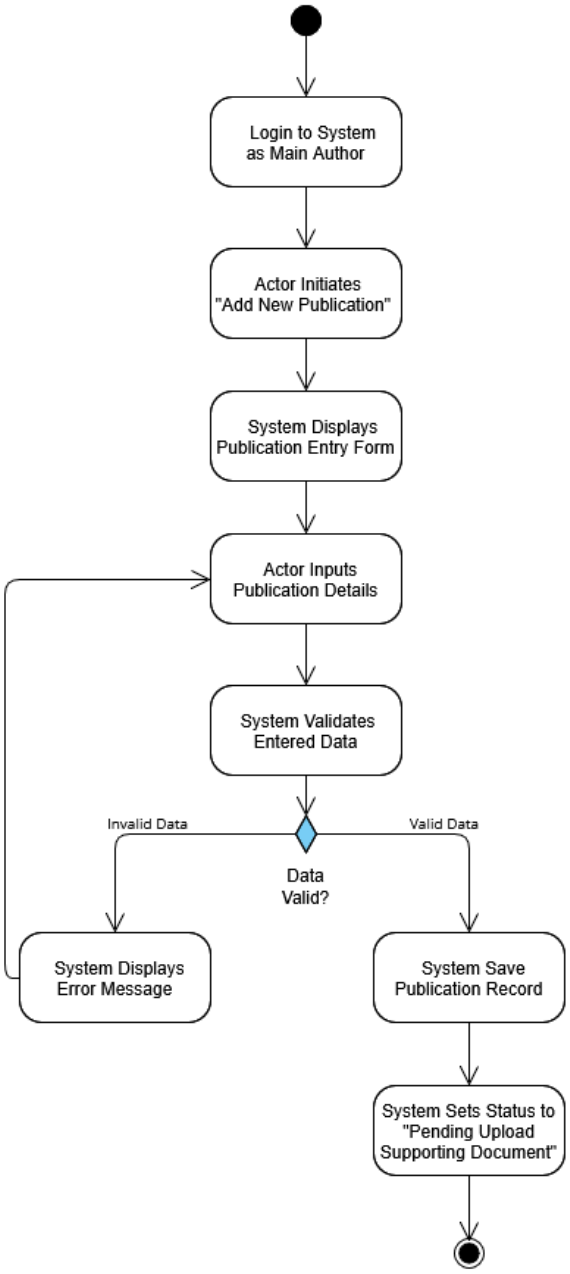
Delete Publication: This function permits the removal of a publication entry if it was submitted incorrectly or is no longer relevant, but only if it has not yet been approved.

Share Publication to Professional Platforms: This use case lets the main author share approved publications to external academic networks such as LinkedIn or ResearchGate to increase research visibility.

Resubmit Publication: This function allows the main author to revise and resubmit a publication that was previously rejected by the coordinator, incorporating feedback for re-review.

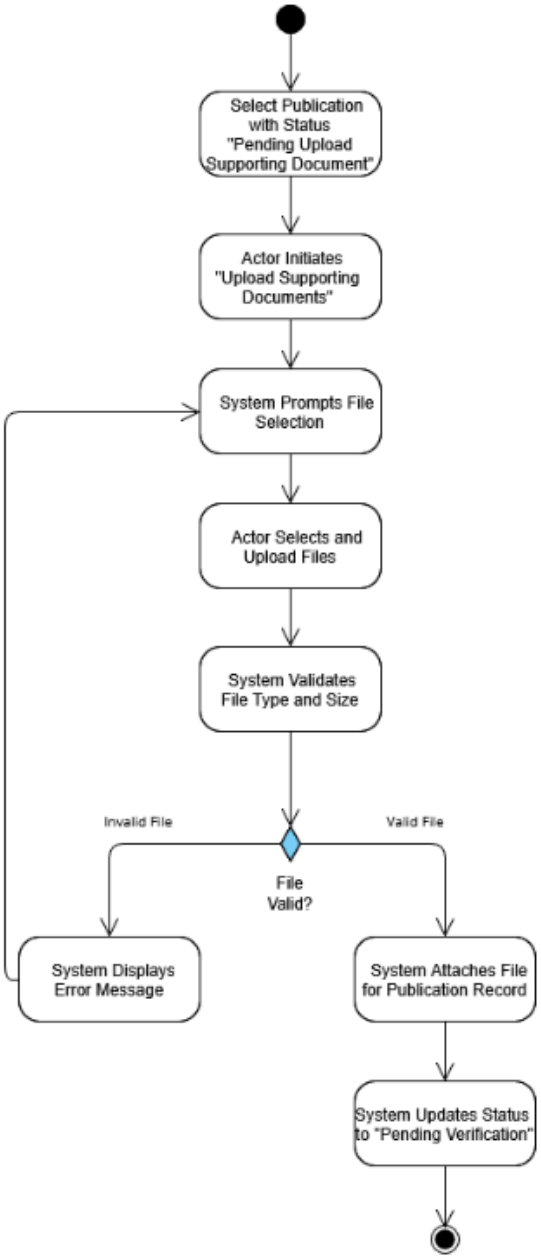


6.1.3.1 Main-author Use case 1 : Add publication



```
BEGIN

FUNCTION addNewPublication()

   LOGIN_AS_MAIN_AUTHOR()

   INITIATE_ADD_PUBLICATION()

   DISPLAY_ENTRY_FORM()

   publicationData = INPUT_PUBLICATION_DETAILS()

   isValid = VALIDATE_DATA(publicationData)

   IF isValid THEN

      SAVE_PUBLICATION(publicationData)

      SET_STATUS("Pending Upload Supporting Document")

      RETURN "Publication added successfully"
   ELSE
      DISPLAY_ERROR_MESSAGE()
      RETURN "Validation failed"
   END IF

END FUNCTION
```

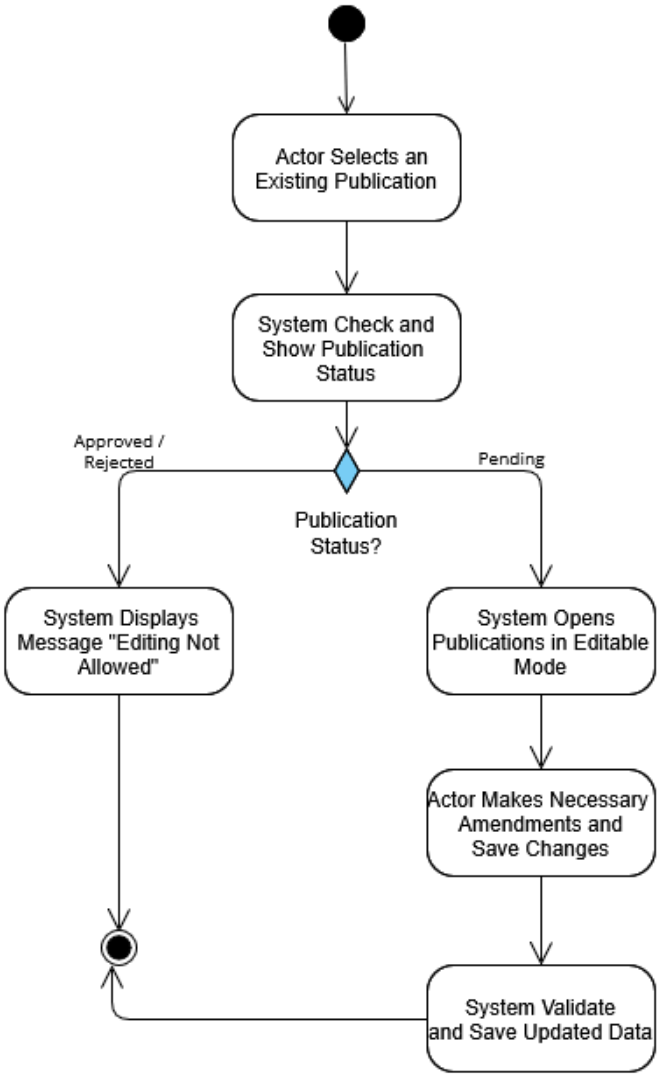
6.1.3.2 Main-author Use case 2 : Upload Supporting Documents




```
BEGIN

FUNCTION uploadSupportingDocuments()

   // 1. Select Publication with Status "Pending Upload Supporting Document"
   publication = SELECT_PUBLICATION_WITH_STATUS("Pending Upload Supporting Document")
```

```
    // 2. Actor Initiates "Upload Supporting Documents"
    INITIATE_UPLOAD()

    // 3. System Prompts File Selection
    SHOW_FILE_SELECTOR()

    // 4. Actor Selects and Uploads Files
    files = SELECT_AND_UPLOAD_FILES()

    // 5. System Validates File Type and Size
    isValid = VALIDATE_FILES(files)

    IF isValid THEN
        // 7. System Attaches File for Publication Record
        ATTACH_FILES_TO_PUBLICATION(publication, files)

        // 8. System Updates Status to "Pending Verification"
        UPDATE_STATUS(publication, "Pending Verification")

        RETURN "Files uploaded successfully"
    ELSE
        // 6. System Displays Error Message
        DISPLAY_FILE_ERROR()
        RETURN "File validation failed"
    END IF

END FUNCTION
```

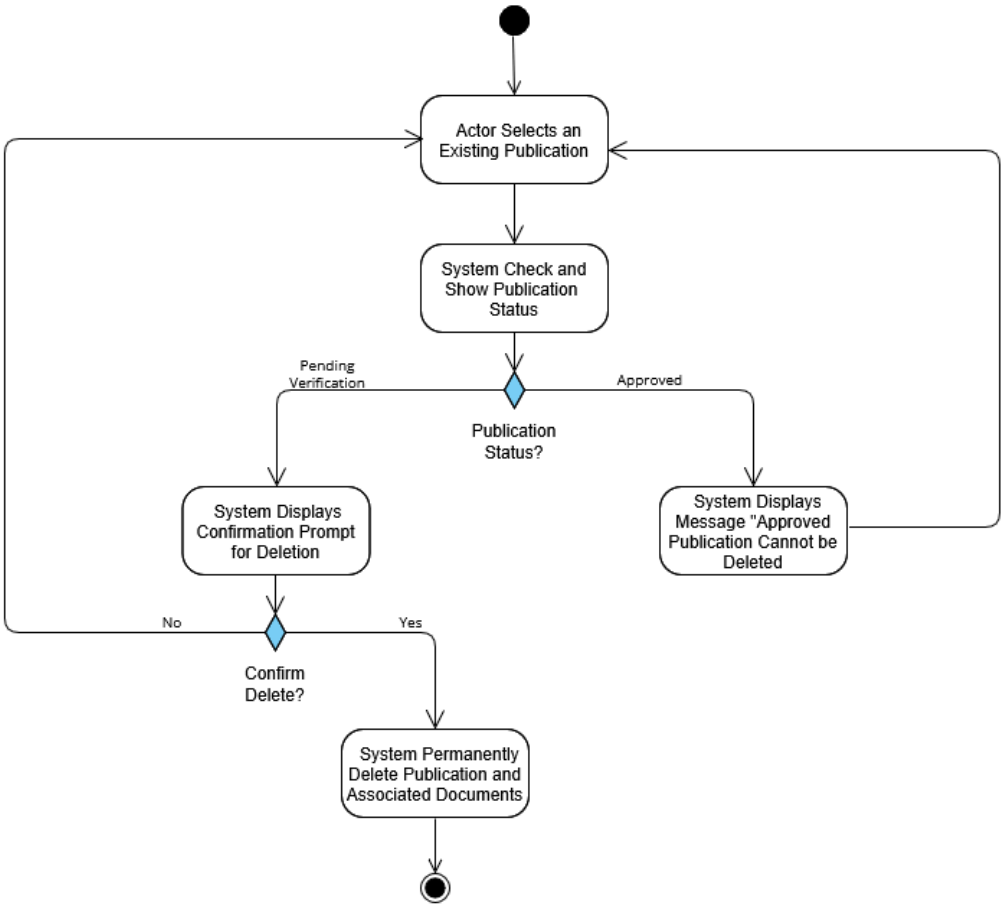
6.1.3.3 Main-author Use case 3 : Edit Publication Details




```
BEGIN

FUNCTION editPublicationDetails()

    publication = SELECT_EXISTING_PUBLICATION()
```

```
    status = GET_PUBLICATION_STATUS(publication)

    IF status == "Approved" OR status == "Rejected" THEN
        DISPLAY_ERROR("Editing Not Allowed")
        RETURN "Cannot edit - publication is " + status
    ELSE
        OPEN_EDITABLE_VIEW(publication)

        updatedData = GET_USER_EDITS()

        IF VALIDATE_DATA(updatedData) THEN
            SAVE_UPDATES(publication, updatedData)
            RETURN "Publication updated successfully"
        ELSE
            DISPLAY_VALIDATION_ERROR()
            RETURN "Validation failed"
        END IF
    END IF

END FUNCTION
```

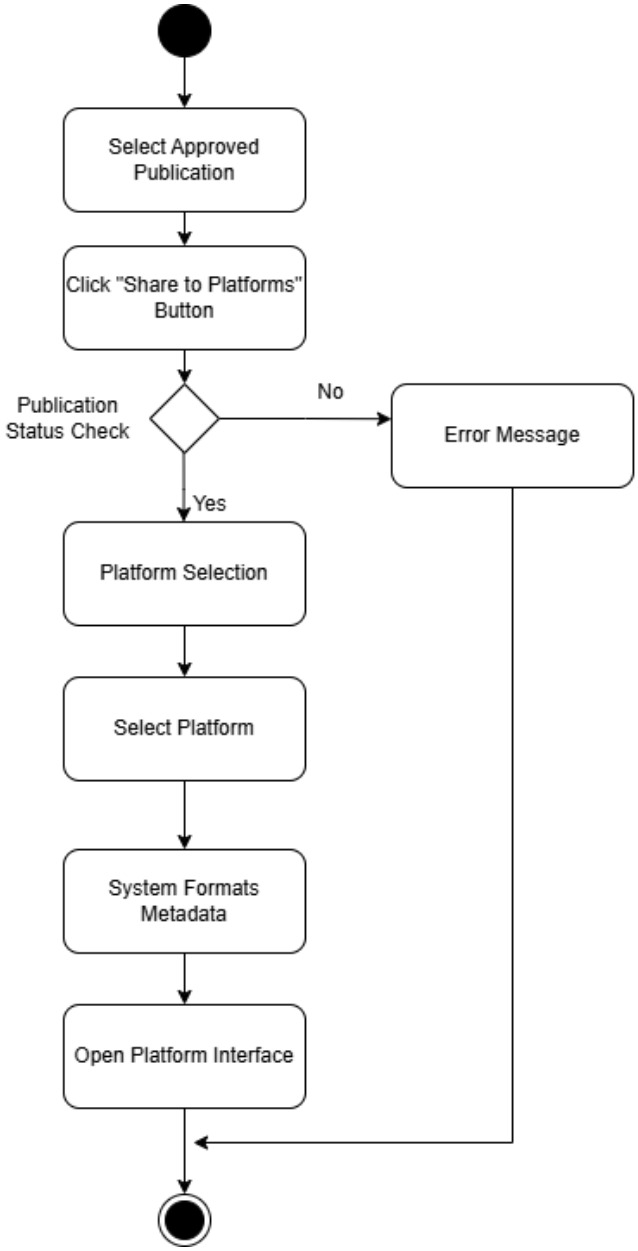
6.1.3.4 Main-author Use case 4 : Delete Publication




```
BEGIN

FUNCTION deletePublication()

   publication = SELECT_EXISTING_PUBLICATION()

   status = GET_PUBLICATION_STATUS(publication)

   IF status == "Approved" THEN
      DISPLAY_ERROR("Approved Publication Cannot be Deleted")
      RETURN "Deletion not allowed"
   ELSE
      isConfirmed = SHOW_CONFIRMATION_PROMPT()

      IF isConfirmed THEN
         DELETE_PUBLICATION(publication)
         RETURN "Publication deleted successfully"
      ELSE
         RETURN "Deletion cancelled"
```

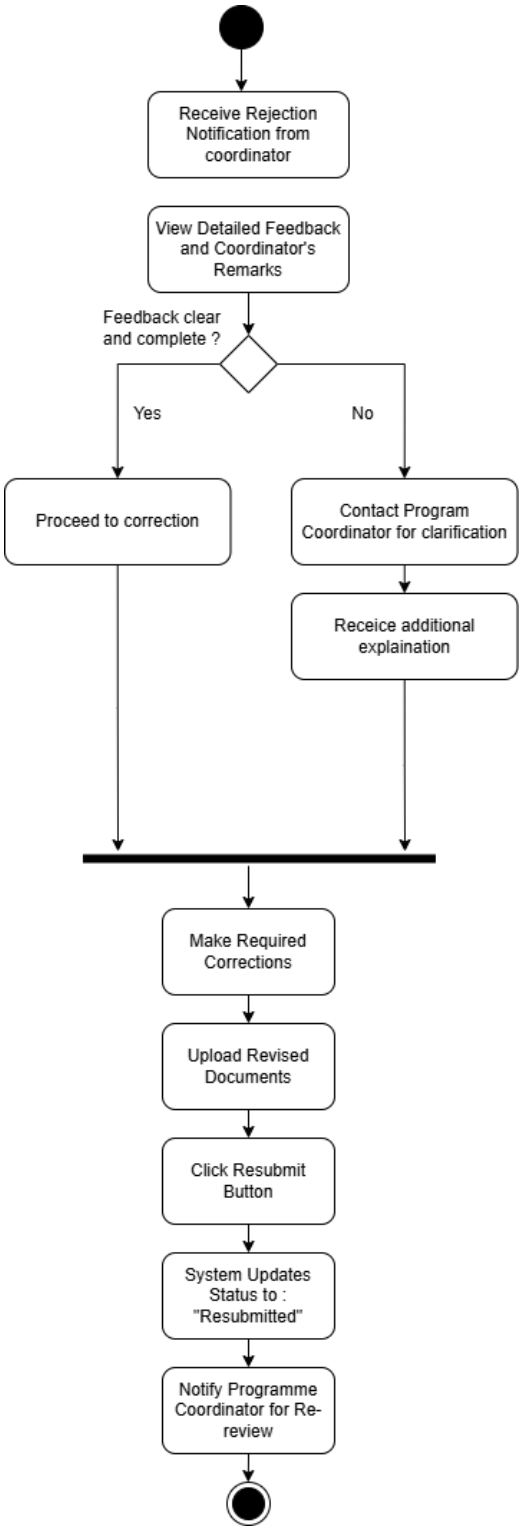
```
    END IF
  END IF

END FUNCTION
```

6.1.3.5 Main-author Use case 5 : Shared Approved publication to Professional Platforms



```
BEGIN

FUNCTION shareToProfessionalPlatforms(publication)

   selectedPublication = GET_SELECTED_PUBLICATION()

   IF userTriggersShareButton() THEN

      IF selectedPublication.status == "Approved" THEN

         targetPlatforms = USER_SELECTS_PLATFORMS()

         formattedMetadata = FORMAT_METADATA(selectedPublication)

         OPEN_PLATFORM_INTERFACE(targetPlatforms, formattedMetadata)

         RETURN "Sharing process initiated successfully"

      ELSE
         DISPLAY_ERROR_MESSAGE("Publication is not approved for sharing")
         RETURN "Error: Publication status not approved"
      END IF

   END IF

END FUNCTION
```

6.1.3.6 Main-author Use case 6 : Resubmit Publication (If program coordinator rejects publication)



```
BEGIN

FUNCTION resubmitRejectedPublication()

    rejectionNotification = RECEIVE_NOTIFICATION("rejection")

    displayNotificationDetails(rejectionNotification)
    feedbackDetails = rejectionNotification.feedback
    coordinatorRemarks = rejectionNotification.remarks

    IF isFeedbackClearAndComplete(feedbackDetails, coordinatorRemarks) THEN

        correctionsMade = makeRequiredCorrections(feedbackDetails)

    ELSE

        coordinatorResponse = CONTACT_COORDINATOR_FOR_CLARIFICATION()

        updatedFeedback = MERGE_FEEDBACK(feedbackDetails, coordinatorResponse)

        correctionsMade = makeRequiredCorrections(updatedFeedback)

    END IF

    revisedDocuments = UPLOAD_REVISED_DOCUMENTS()

    IF userClicksResubmitButton() THEN

        UPDATE_PUBLICATION_STATUS("Resubmitted")

        NOTIFY_COORDINATOR("Publication resubmitted for review")

        RETURN "Publication resubmitted successfully"

    END IF

END FUNCTION
```

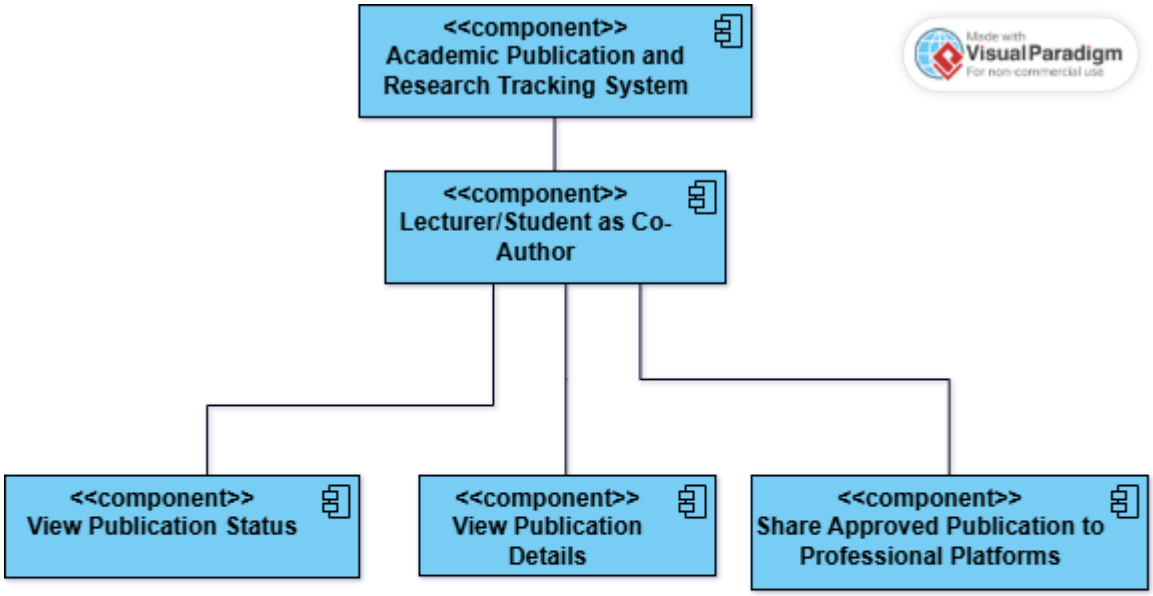
6.1.4 Subsystem 4 : Lecturer/Student as Co-author

The Lecturer/Student as Co-Author subsystems include:

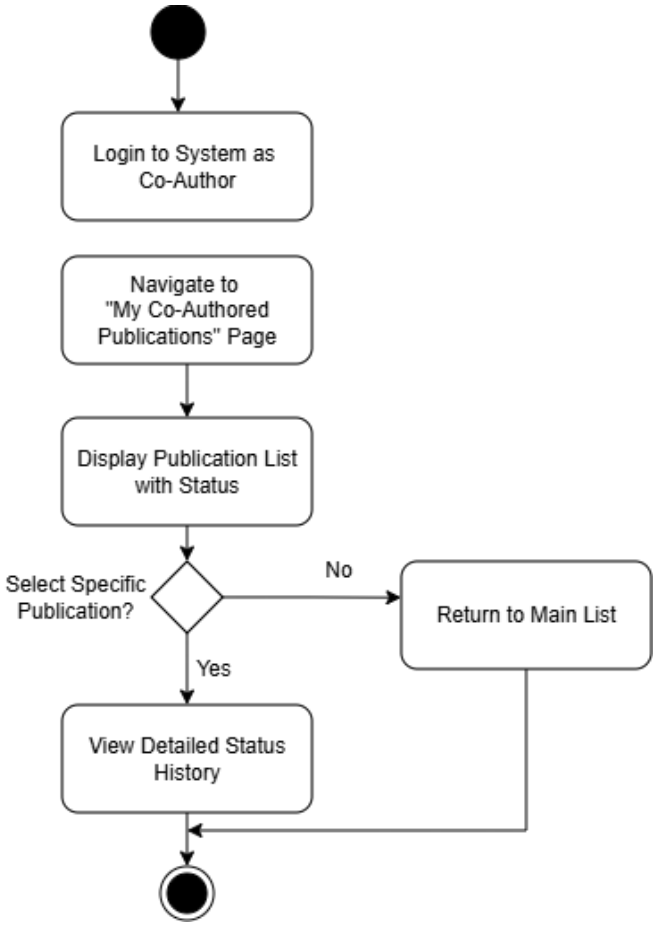
View Publication Status: Co-Authors can monitor the current verification and approval progress of publications they have contributed to, tracking status changes from submission to final decision with coordinator remarks.

View Publication Details: Co-Authors have read-only access to complete publication metadata, author information, venue details, indexing status, and supporting documentation without editing permissions.

Share Approved Publication to Professional Platforms: Co-Authors can disseminate approved publications to external academic networks like LinkedIn, ORCID, and ResearchGate to enhance research visibility and professional networking.

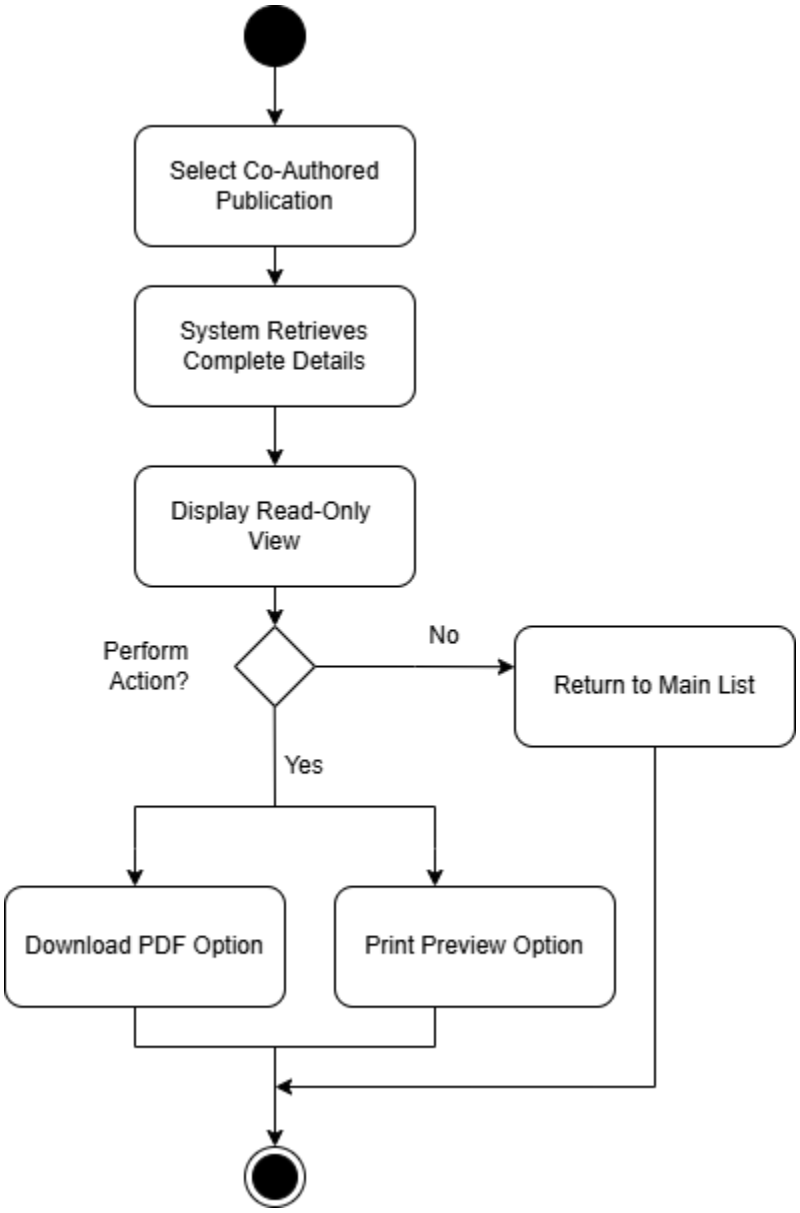


6.1.4.1 Co-Author Use case 1 : View Publication Status




```
BEGIN

FUNCTION viewPublicationStatus()

   publications = GET_COAUTHORED_PUBLICATIONS()
   DISPLAY_PUBLICATIONS(publications)

   IF USER_WANTS_DETAILS() THEN
      publication = SELECT_PUBLICATION()
      history = GET_STATUS_HISTORY(publication)
      DISPLAY_HISTORY(history)
   END IF

END FUNCTION
```

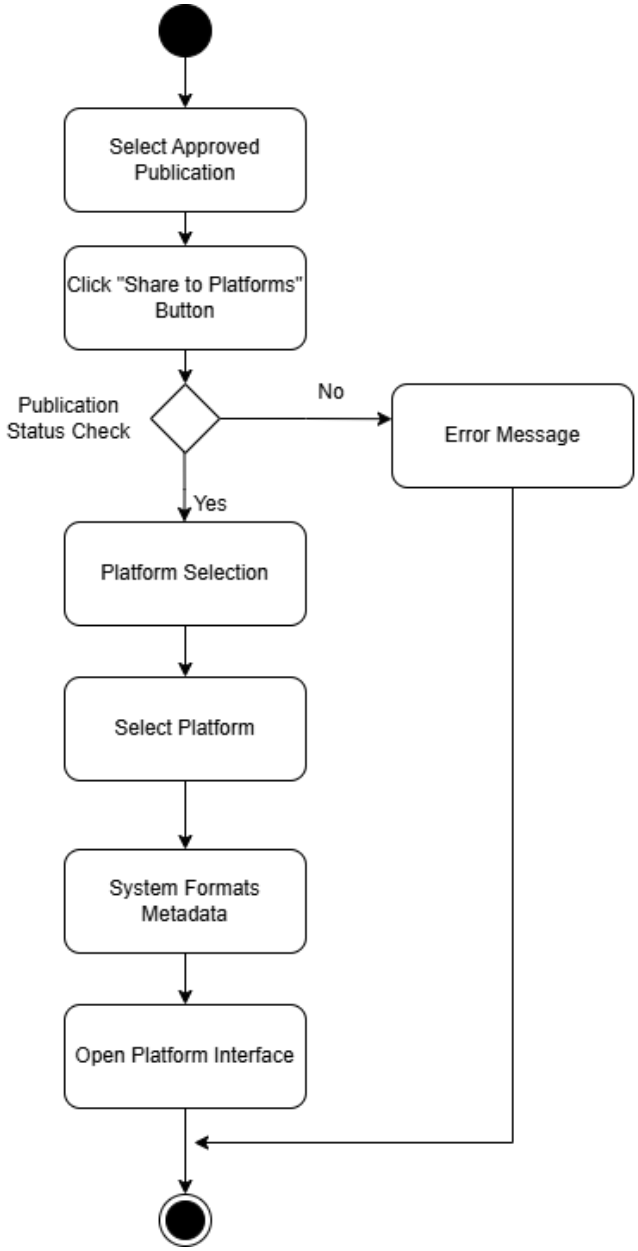
6.1.4.2 Co-Author Use case 2 : View Publication Details




```
BEGIN

FUNCTION viewPublicationDetails()

    publication = SELECT_COAUTHORED_PUBLICATION()
    details = GET_PUBLICATION_DETAILS(publication.id)
    DISPLAY_DETAILS(details)

    action = GET_ACTION()
    IF action == "download" THEN DOWNLOAD_PDF(details)
    IF action == "print" THEN PRINT_PREVIEW(details)
```

```
    RETURN_TO_LIST()

END FUNCTION
```

6.1.4.3 Co-Author Use case 3 : Share Approved Publication to Professional Platforms



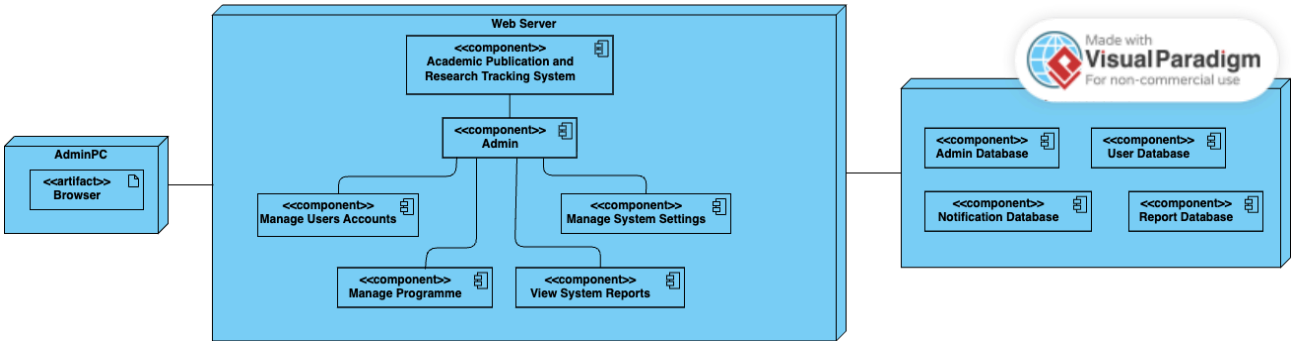
```
BEGIN

FUNCTION coauthorShareToProfessionalPlatforms()

   selectedPublication = GET_SELECTED_COAUTHOR_PUBLICATION()

   IF userTriggersShareButton() THEN

      IF selectedPublication.status == "Approved" THEN

         targetPlatforms = USER_SELECTS_PLATFORMS()

         formattedMetadata = FORMAT_METADATA(selectedPublication)

         OPEN_PLATFORM_INTERFACE(targetPlatforms, formattedMetadata)

         RETURN "Sharing process initiated successfully"

      ELSE
         DISPLAY_ERROR_MESSAGE("Publication is not approved for sharing")
         RETURN "Error: Publication status not approved"
      END IF

   END IF

END FUNCTION
```

7 Deployment Design

7.1 Deployment Diagram

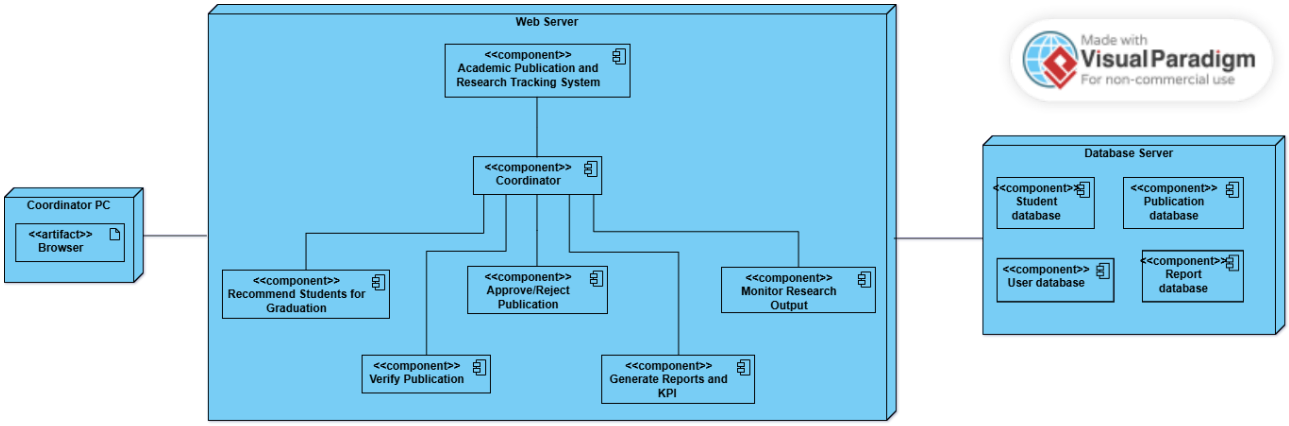
7.1.1 Admin

This deployment diagram illustrates the three-tier architecture of the Academic Publication and Research Tracking System specifically for the Admin actor. The Admin PC hosts a web browser artifact and components for managing user accounts, manage system settings, manage programmes and view system reports. These administrative application components interact with a Database Server containing the User Profile, Notification, Admin, and Report databases.



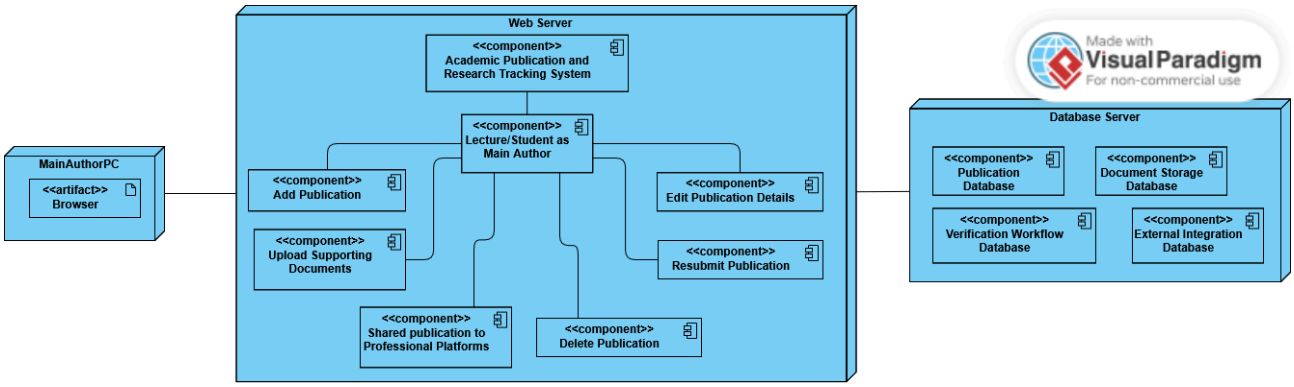
7.1.2 Coordinator

This deployment diagram illustrates the three-tier architecture of the Academic Publication and Research Tracking System. The Coordinator PC, hosts a web browser artifact and components for recommending students and managing publications, which communicate via a Web Server, running the core system and coordinator-specific services for approval, monitoring, and reporting. These application components interact with a Database Server containing the Student, User, and Report databases, ensuring data persistence and separation of concerns across the system's layers.



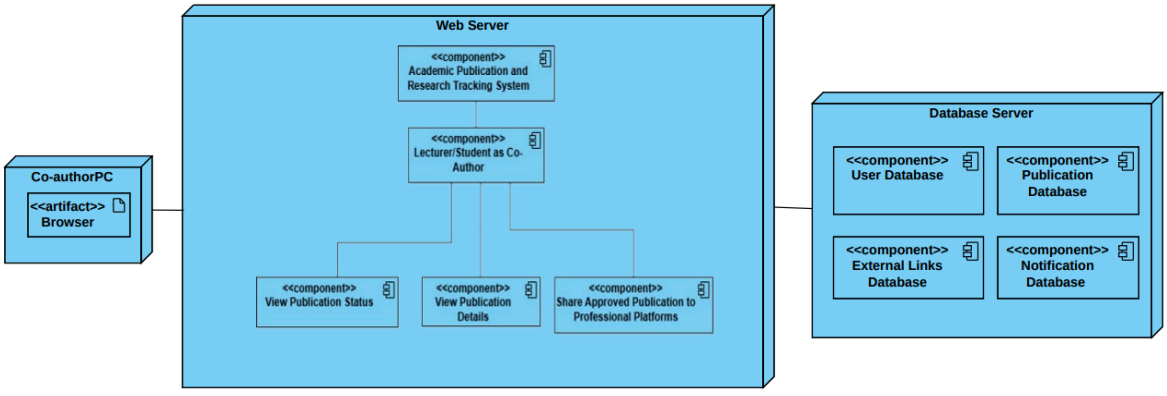
7.1.3 Lecturer/Student as Main-author

This deployment diagram shows how the Academic Publication and Research Tracking System is set up for a Main Author (a lecturer or student). The Main Author uses a Browser on their computer (MainAuthorPC) to access the system through a Web Server, which hosts the entire application. The Web Server runs the core system and the specific Main Author module, which includes features like adding publications, uploading documents, and sharing to external platforms. In the background, three separate database servers store different types of data: the Publication Database holds publication details and metadata, the Document Storage Database keeps uploaded files like PDFs, and the External Integration Database manages connections to platforms like LinkedIn and ORCID. This setup ensures the Main Author can smoothly submit, track, and share their research while the system keeps everything organized and secure.



7.1.4 Lecturer/Student as Co-author

This deployment diagram illustrates the three-tier architecture of the Academic Publication and Research Tracking System specifically for the Lecturer/Student (Co-Author) actor. The Co-Author PC hosts a web browser artifact and components for viewing publication status, tracking research progress, and sharing research to external professional platforms (such as LinkedIn or ORCID). These application components interact via the Web/Application Server with a Database Server containing the User Profile, Publication, and Notification databases to ensure real-time access to research data and feedback from coordinators.



8 Summary

Based on the Software Design Specification, the design approach adopts a robust multi-tier architecture comprising Presentation, Business, Service, Data Access, and Data layers, which ensures modularity, scalability, and a clear separation of concerns between the user interface and backend logic . The system is logically structured into four distinct functional subsystems which are Admin, Coordinator, Main Author, and Co-Author. These subsystems are supported by a Service Layer that handles reusable components like notifications and external API integrations . Data management utilizes a hybrid approach, employing a relational database for structured records alongside a dedicated file storage system for efficiently handling physical document uploads .

Regarding readiness for implementation, the system is well-prepared for the coding phase, evidenced by comprehensive behavioral and structural modeling. Detailed sequence and activity diagrams have been developed for every use case, mapping out precise logic flows for complex processes such as publication verification, graduation eligibility checks, and system reporting. Furthermore, the database schema is fully defined through a complete data dictionary and entity-relationship models, providing a clear blueprint for backend development. While the document indicates that specific screen designs were pending at the time of writing, the detailed definition of API controllers, business logic components, and state transitions confirms that the core application logic is fully specified and ready for development.