الكلية الجامعية للعلوم التطبيقية

رائدة الإبداع

Project #5

Design Document

Academic Paper Management System

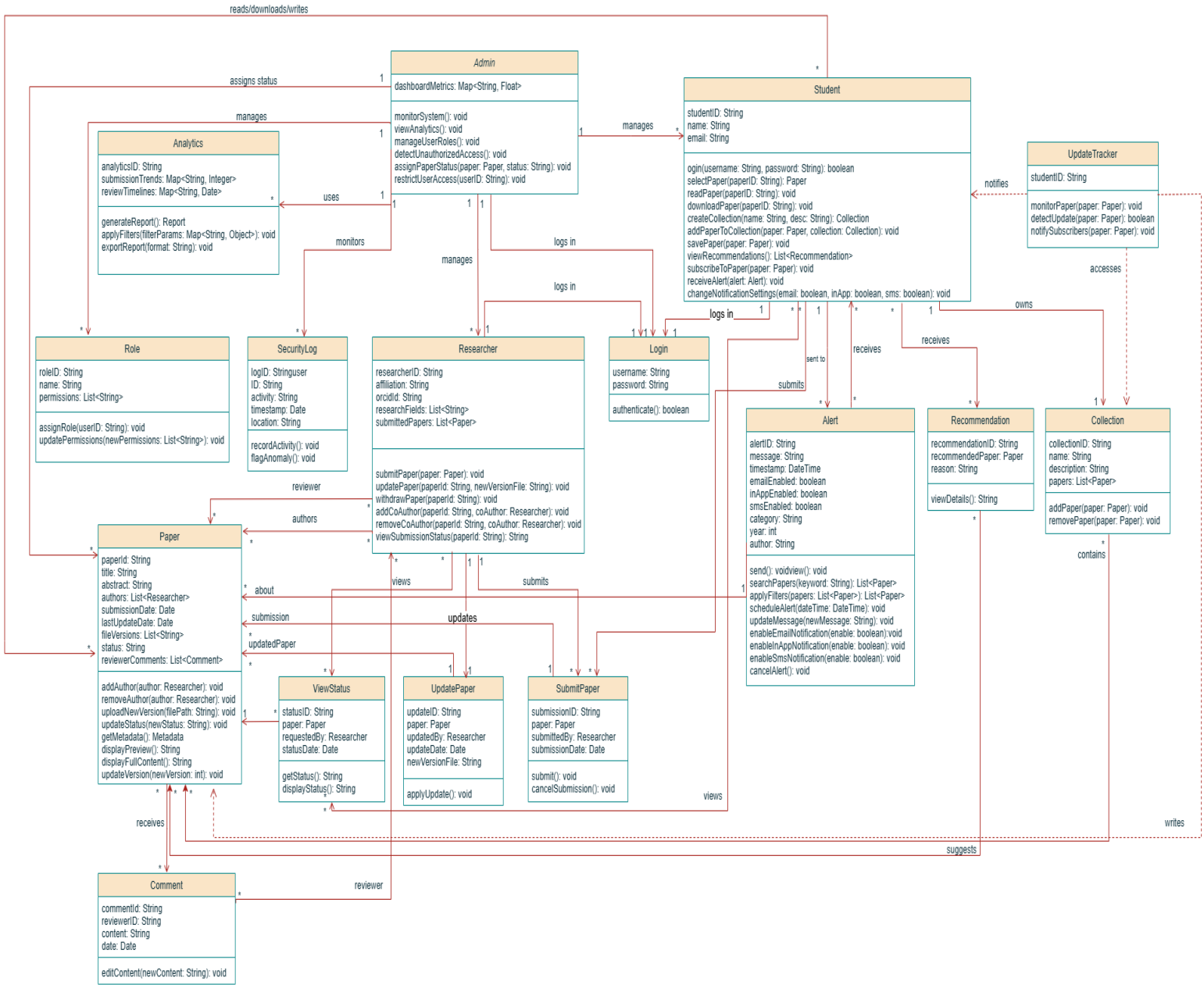
Software Engineering Project. By:

Hebatallah AbuHarb - 220210448
Shahd El-Thalathini - 220210527
Salma Shaheen - 220210654

Class Diagram

A high-level object-oriented representation of the system, illustrating class relationships, public methods, and attributes. The diagram provides insight into how the system is structured, ensuring clarity, maintainability, and extensibility.

link of diagram here : [Academic Paper Management System - Class Diagram **]**



◆Class Descriptions

Outlines the purpose and behavior of each class in the system. The following design enables separation of concerns, data encapsulation, and supports future feature enhancements.

Class Name	Class State & Behavior
Researcher	**Class state:** • Researcher ID, affiliation, ORCID ID, research fields, list of submitted papers. **Class behavior:** • Submit, update, withdraw papers. • Manage co-authors. • View paper submission status.
SubmitPaper	**Class state:** • Submission ID, paper reference, researcher who submitted, submission date. **Class behavior:** • Submit a paper. • Cancel submission if needed.
UpdatePaper	**Class state:** • Update ID, paper reference, researcher who updated, update date, new version file. **Class behavior:** • Apply updates to papers. • Manage version updates.
ViewStatus	**Class state:** • Status ID, paper reference, researcher requesting status, status date. **Class behavior:** • Retrieve and display paper status to researcher.
Login	**Class state:** • Username, password. **Class behavior:** • Authenticate user login credentials. • Enable secure system access.
Comment	**Class state:** • Reviewer ID, comment content, date of comment. **Class behavior:** • Edit comment content. • Manage reviewer feedback.

Admin	**Class state:** • Dashboard metrics, managed roles, monitored users and activities. **Class behavior:** • Monitor system performance. • View analytics reports. • Manage user roles and access rights. • Detect unauthorized access. • Assign or update paper status. • Restrict or revoke user access.
Analytics	**Class state:** • Analytics ID, submission trends, review timelines. **Class behavior:** • Generate analytical reports. • Apply filters to report data. • Export reports in various formats (PDF, CSV, etc.).
SecurityLog	**Class state:** • Log ID, user ID, activity description, timestamp, location. **Class behavior:** • Record system activity. • Flag anomalies or suspicious behavior.
Paper	Class State: • Paper ID, title, abstract, list of authors, submission date, last update date, file versions, status, reviewer comments. **Class behavior:** • Add/remove authors. • Upload and manage new versions. • Update paper status. • Retrieve metadata. • Display previews and full content.
Student	**Class state:** • Student ID, name, email. **Class behavior:** • Login to the system using credentials • Select, read, and download academic papers • Create collections of papers for future reference • Add/remove papers to/from collections • Save favorite papers • View personalized paper recommendations • Subscribe to paper updates • Receive alerts and manage notification preferences

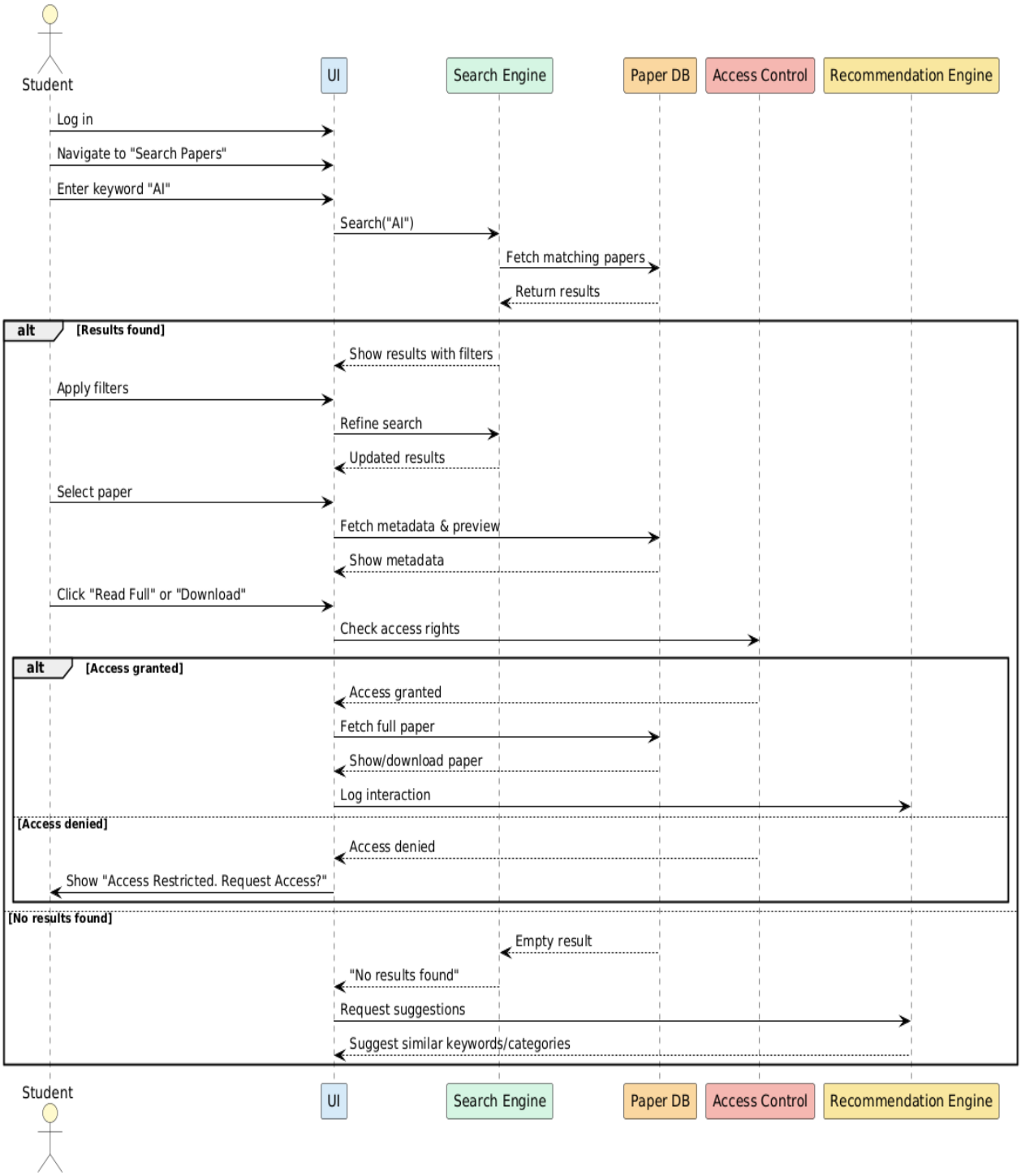
Collection	**Class state:** • Collection ID, name, description, list of papers. **Class behavior:** • Add or remove papers from a collection.
Alert	**Class state:** • Alert ID, message, timestamp, notification settings (email, in-app, SMS), category, author, year. **Class behavior:** • Send or cancel alerts. • View and update alert messages. • Search and filter related papers. • Schedule alerts. • Manage notification preferences.
UpdateTracker	**Class state:** • (Stateless helper class) **Class behavior:** • Monitor papers for changes. • Detect paper updates. • Notify subscribers and update collections accordingly.
Recommendation	**Class state:** • Recommendation ID, recommended paper, reason for recommendation. **Class behavior:** • View detailed information about the recommendation.

◆ Sequence Diagrams

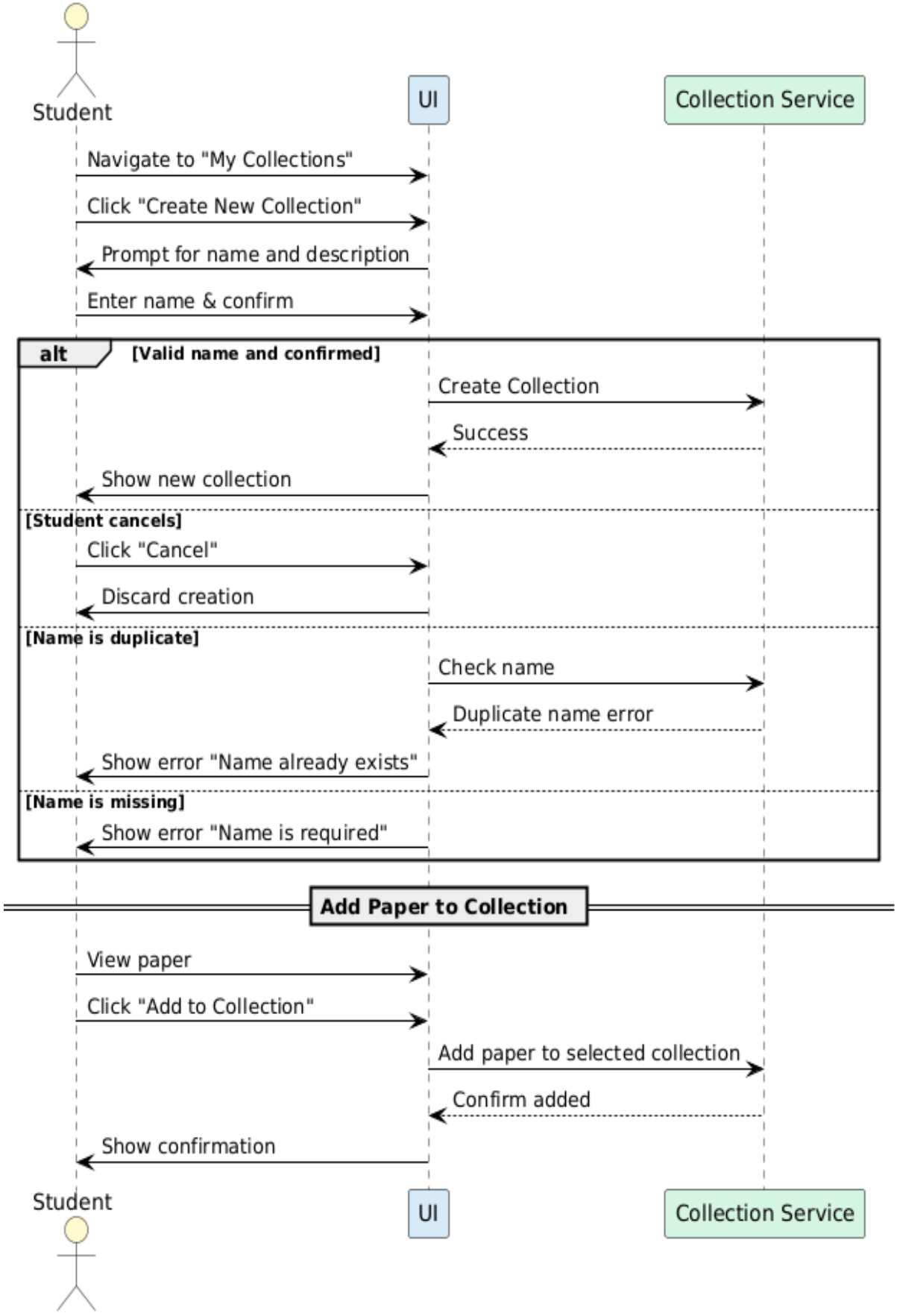
Sequence diagrams illustrate how objects in the system interact over time to carry out specific use cases. They capture the dynamic behavior of the system by detailing the sequence of messages exchanged between components to complete key functionalities. These diagrams provide clarity on the flow of logic and support communication among developers and stakeholders during design and implementation.

Student Actor (5 use cases):

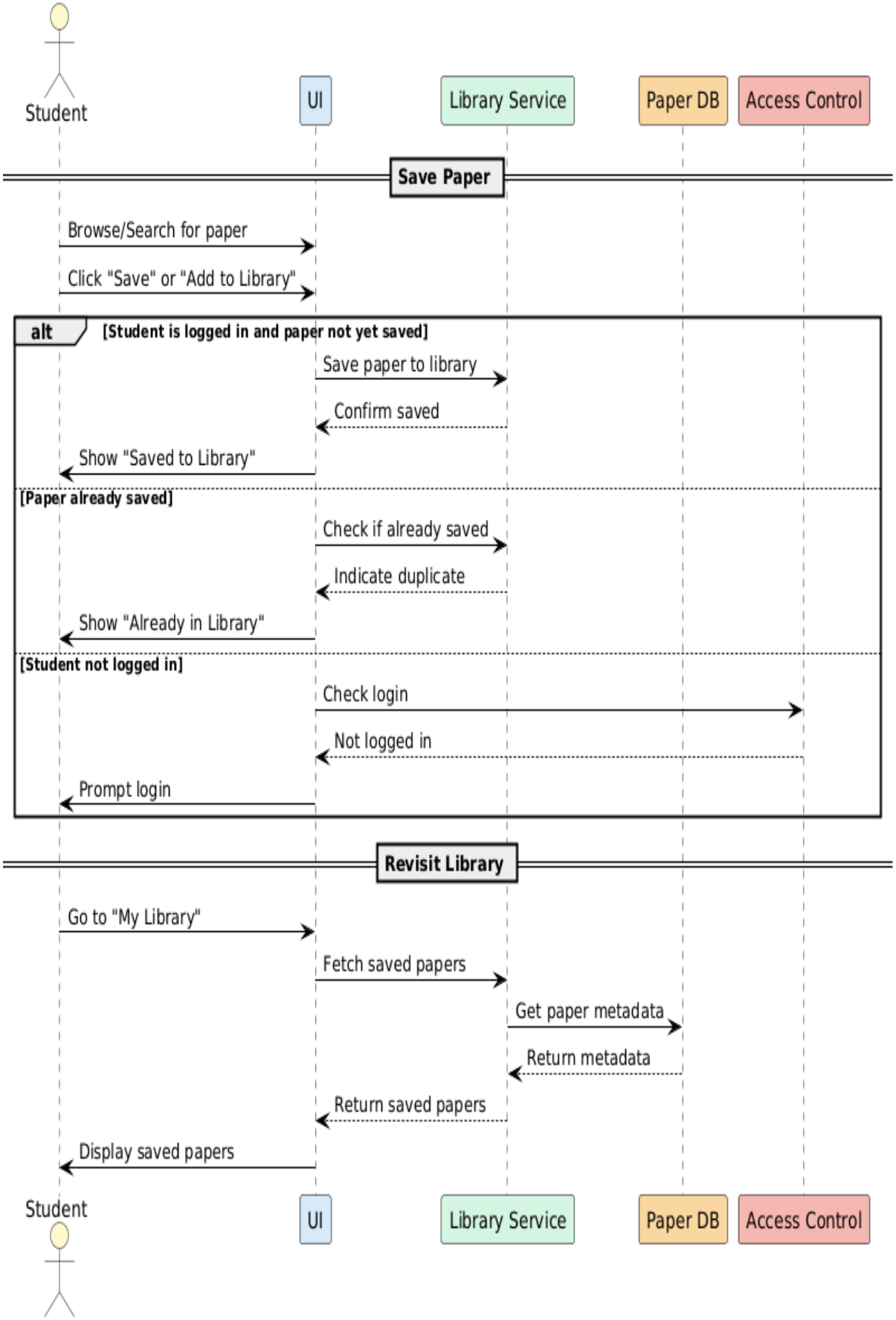
UC-S1: Search and Read Academic Papers



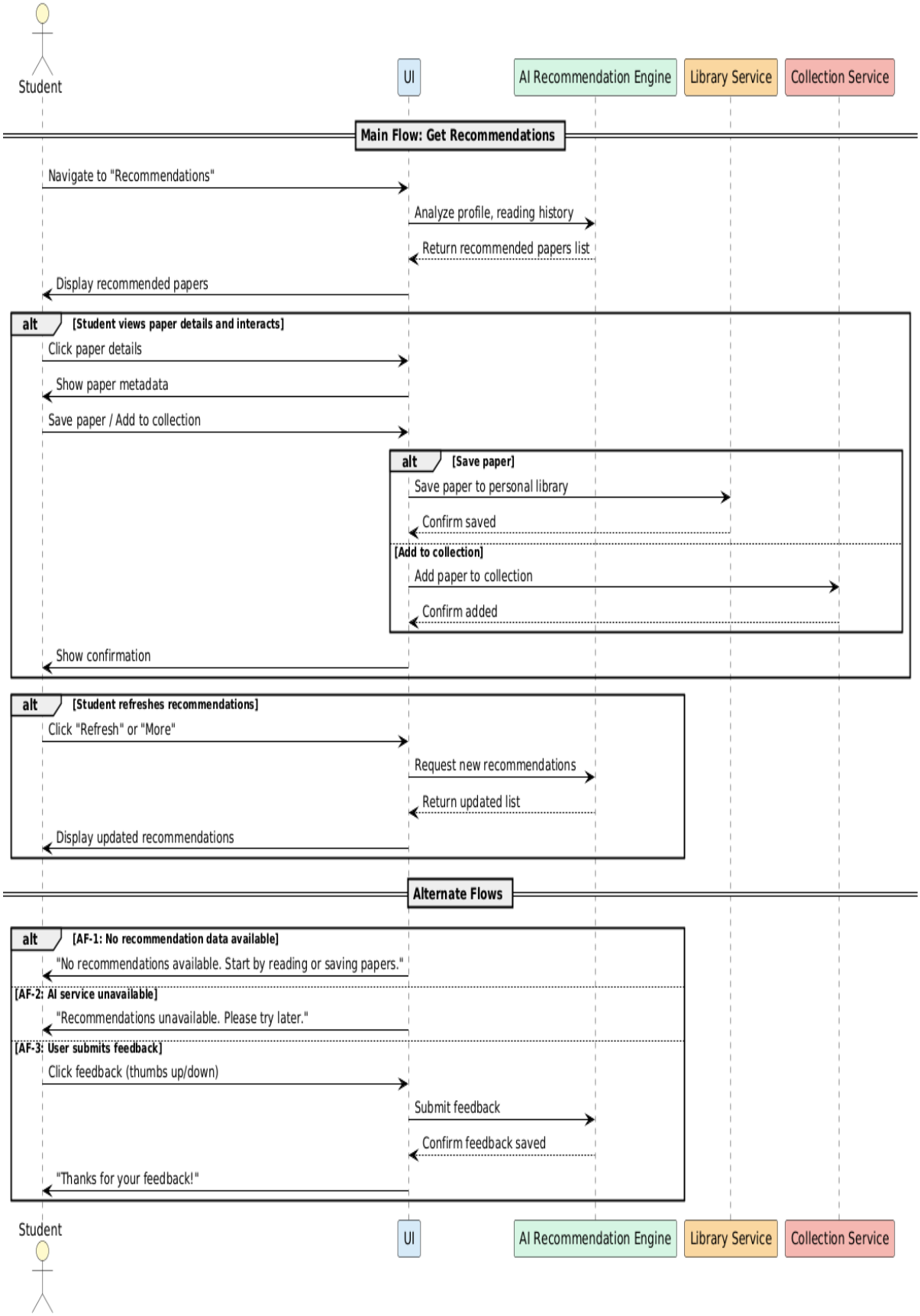
UC-S2: Create Personal Collections of Papers



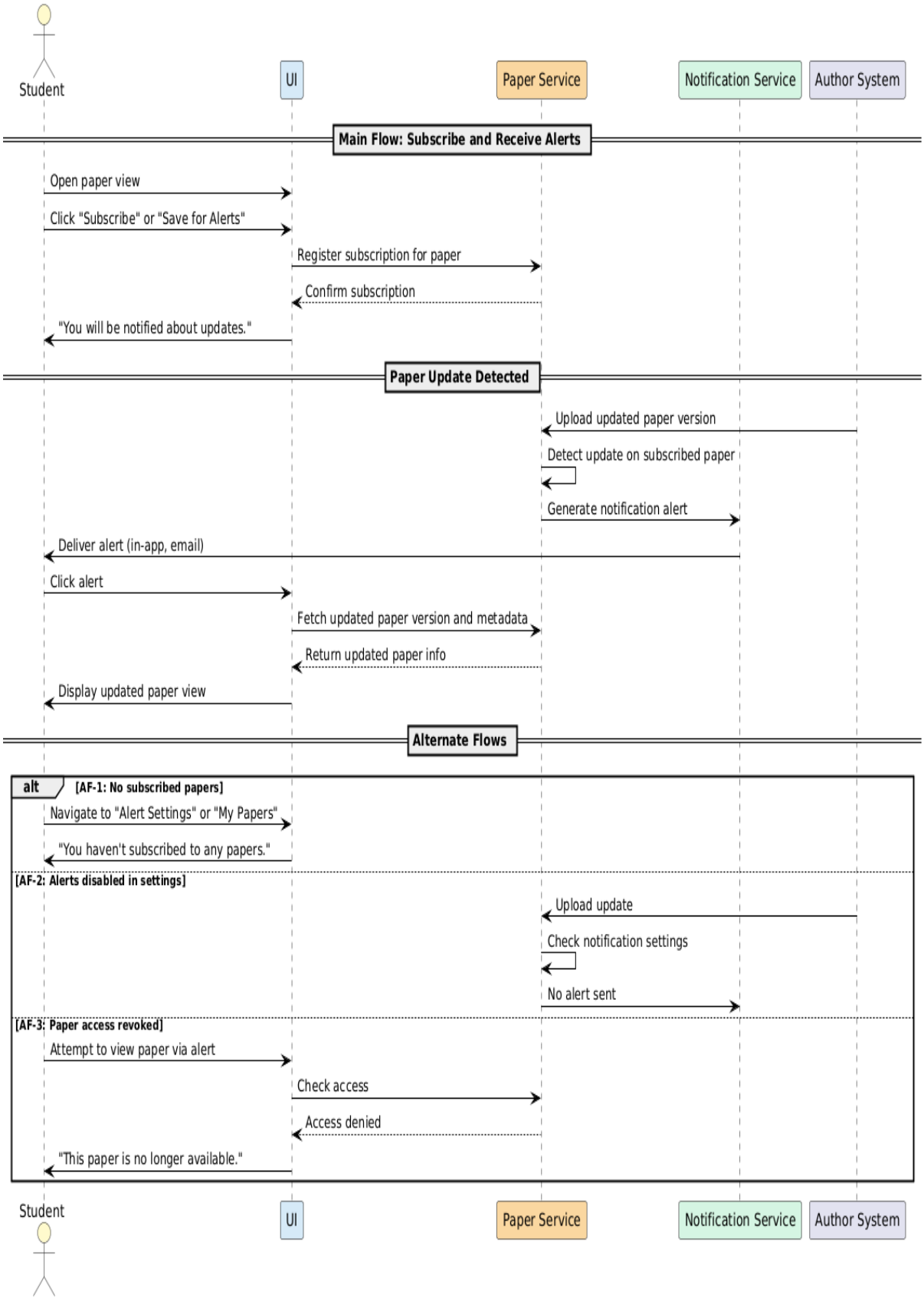
UC-S3: Save and Revisit Interesting Papers



UC-S4: Discover Paper Recommendations

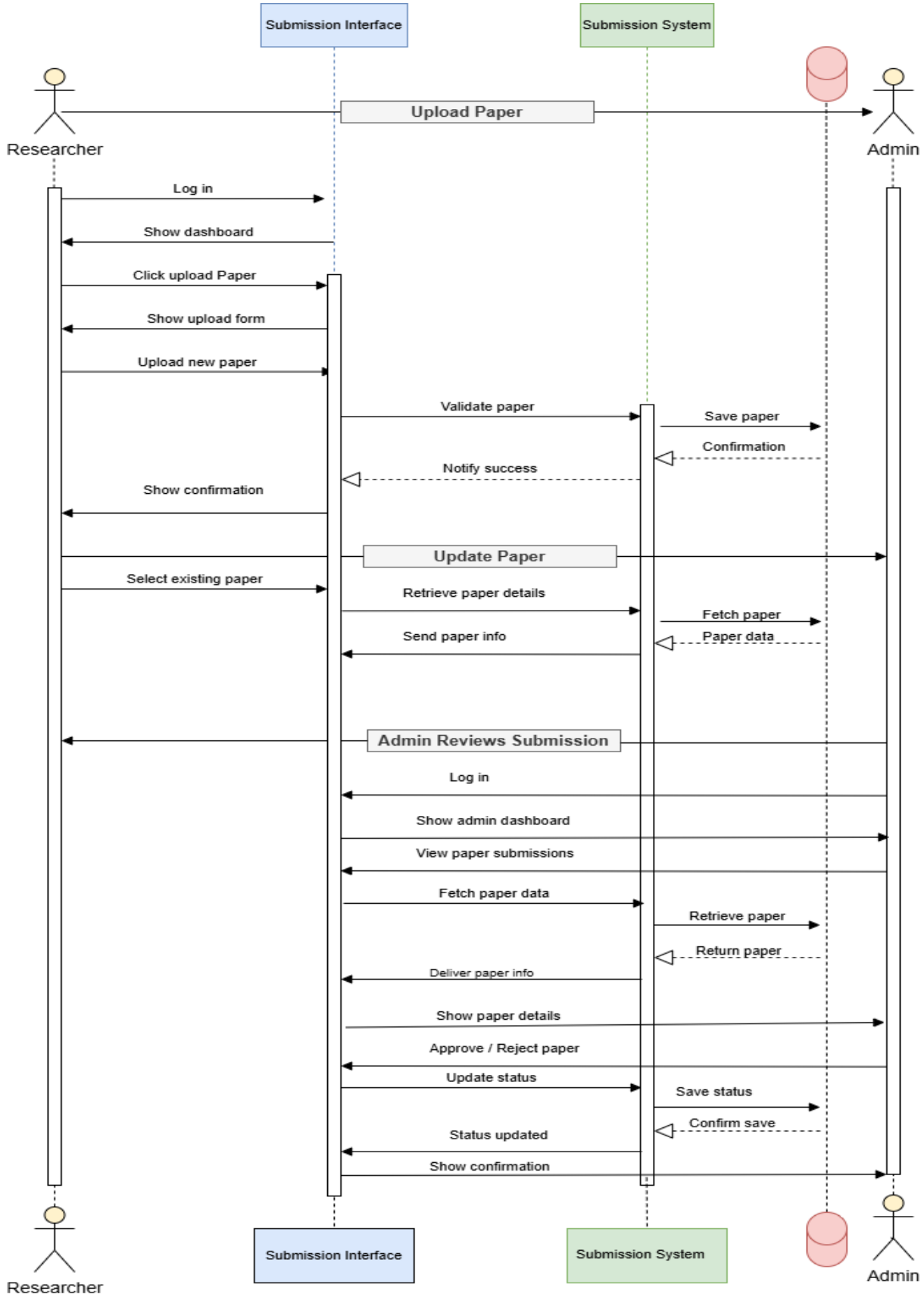


UC-S5: Receive Alerts When a Paper is Updated

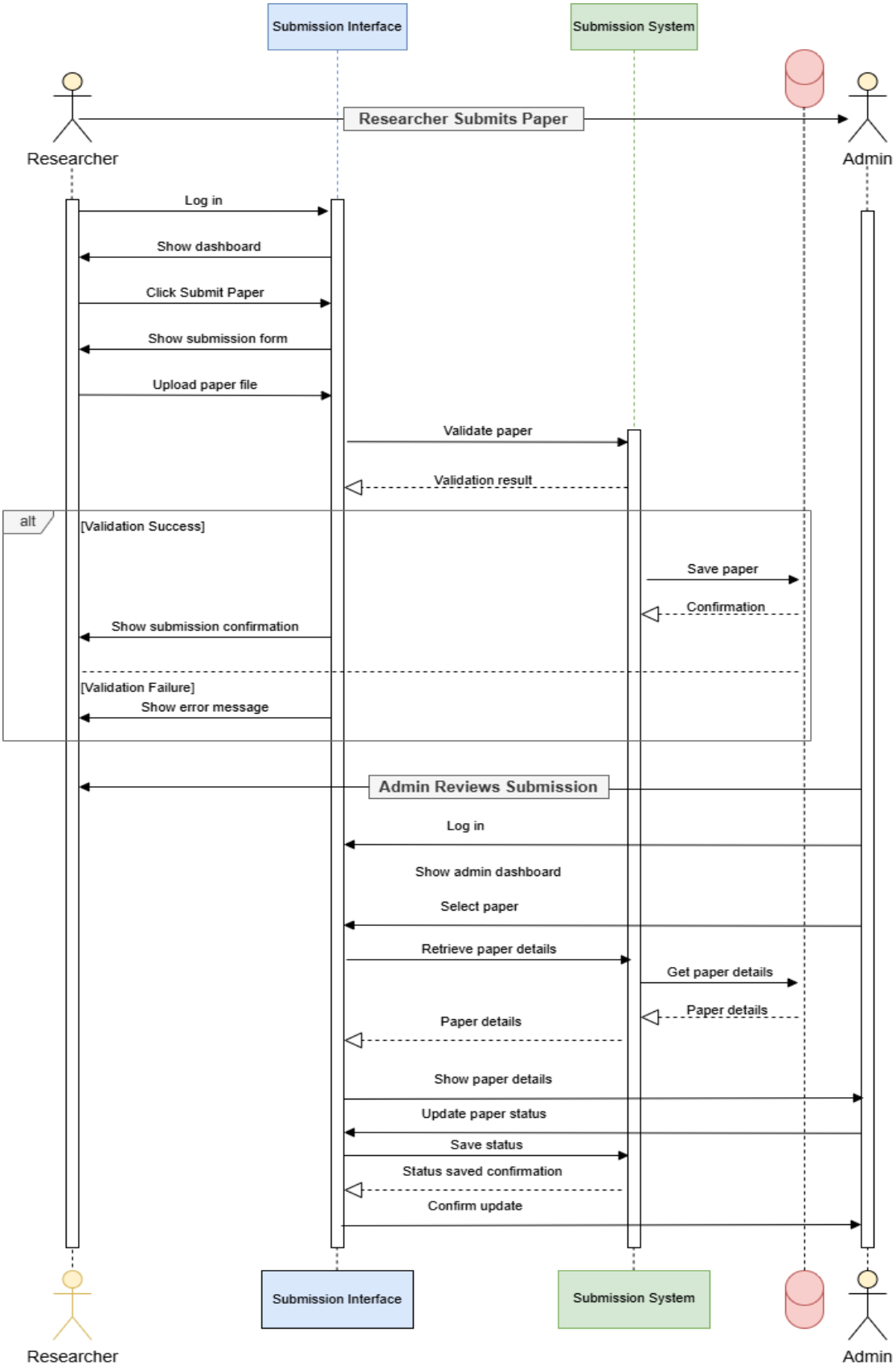


Researcher Actor (3 use cases):

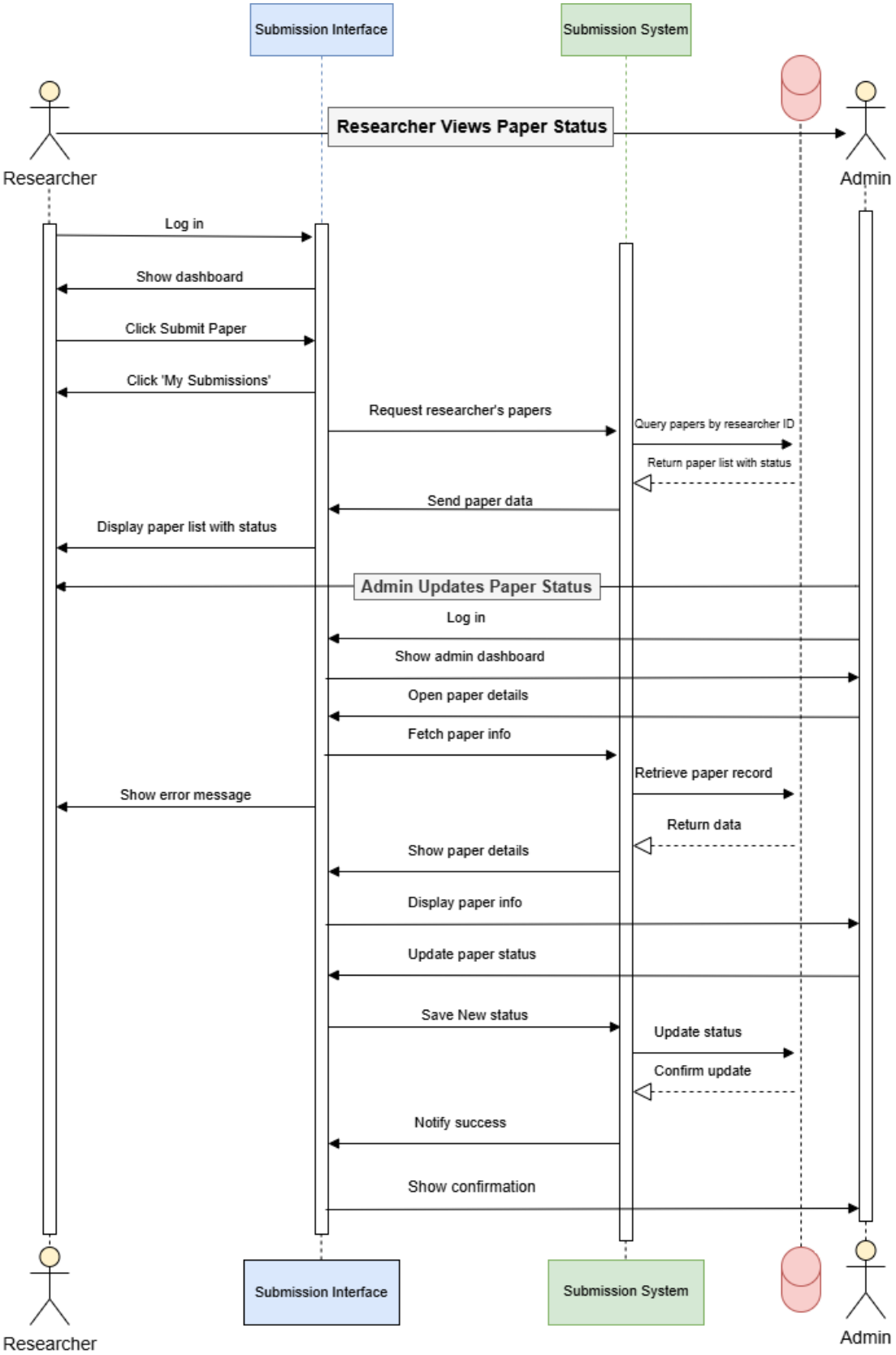
UC-R1: Paper Submission by Researcher



UC-R2:Paper Upload and Update by Researcher

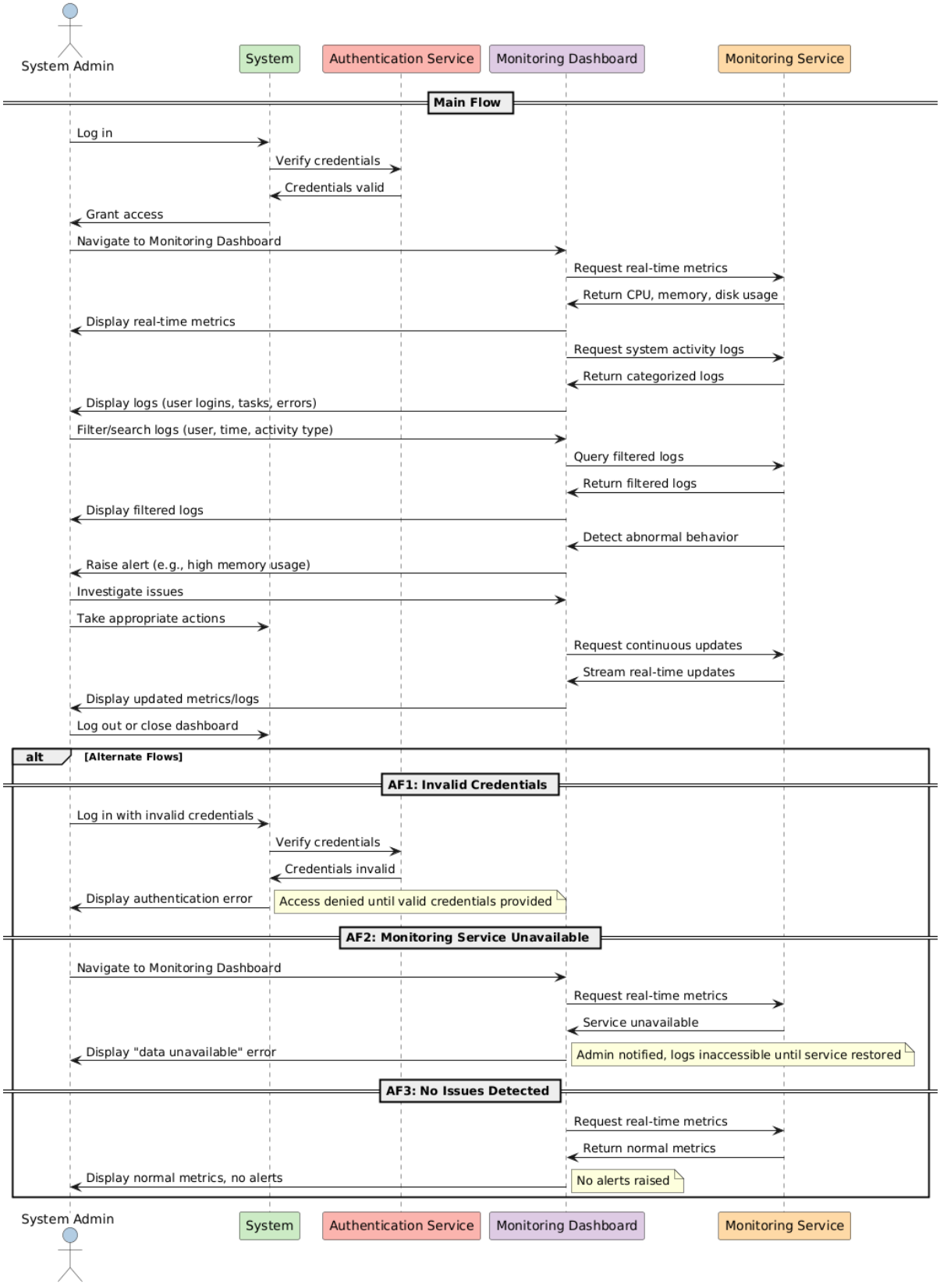


UC-R3: View Status of Submitted Papers

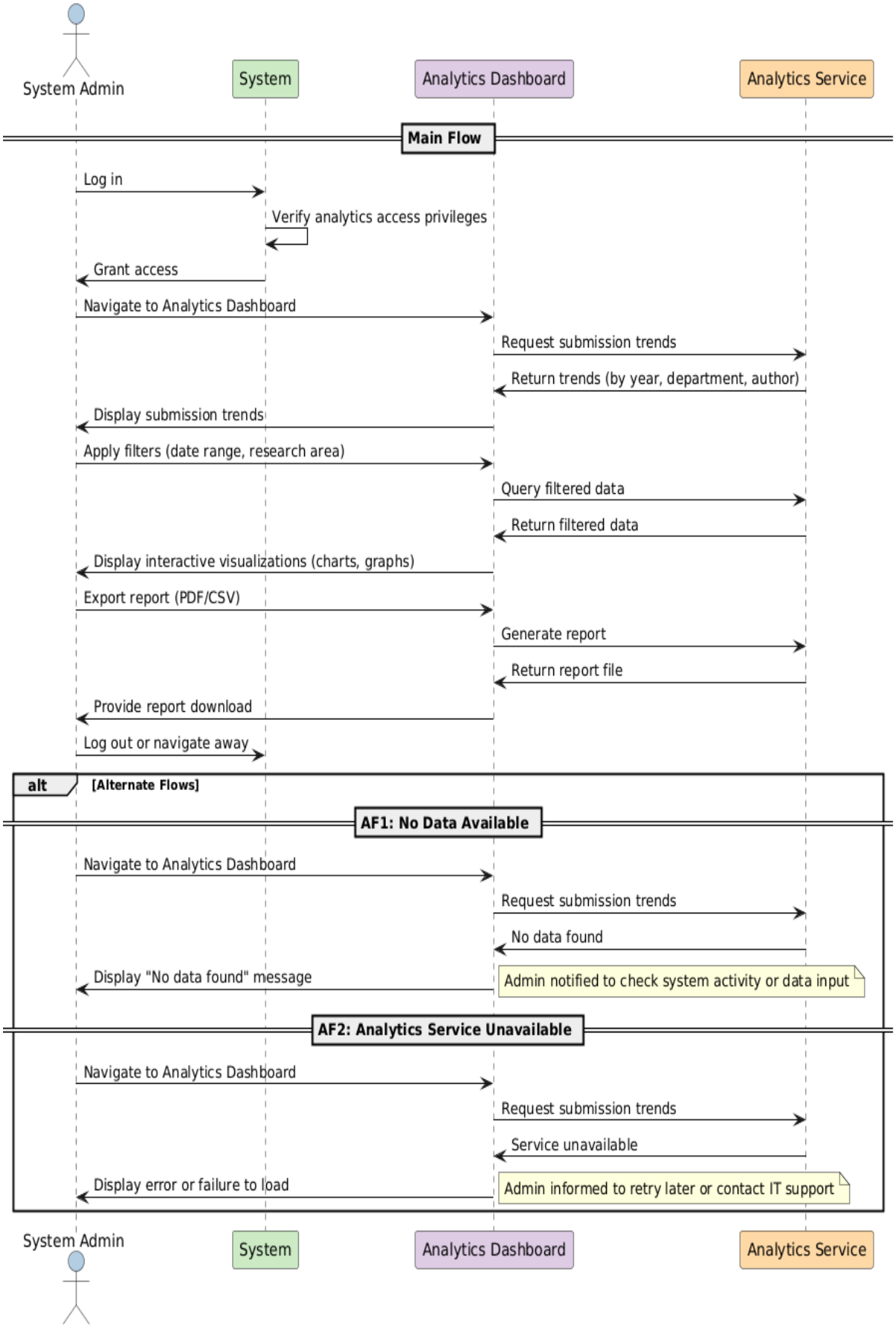


Admin Actor (5 use cases):

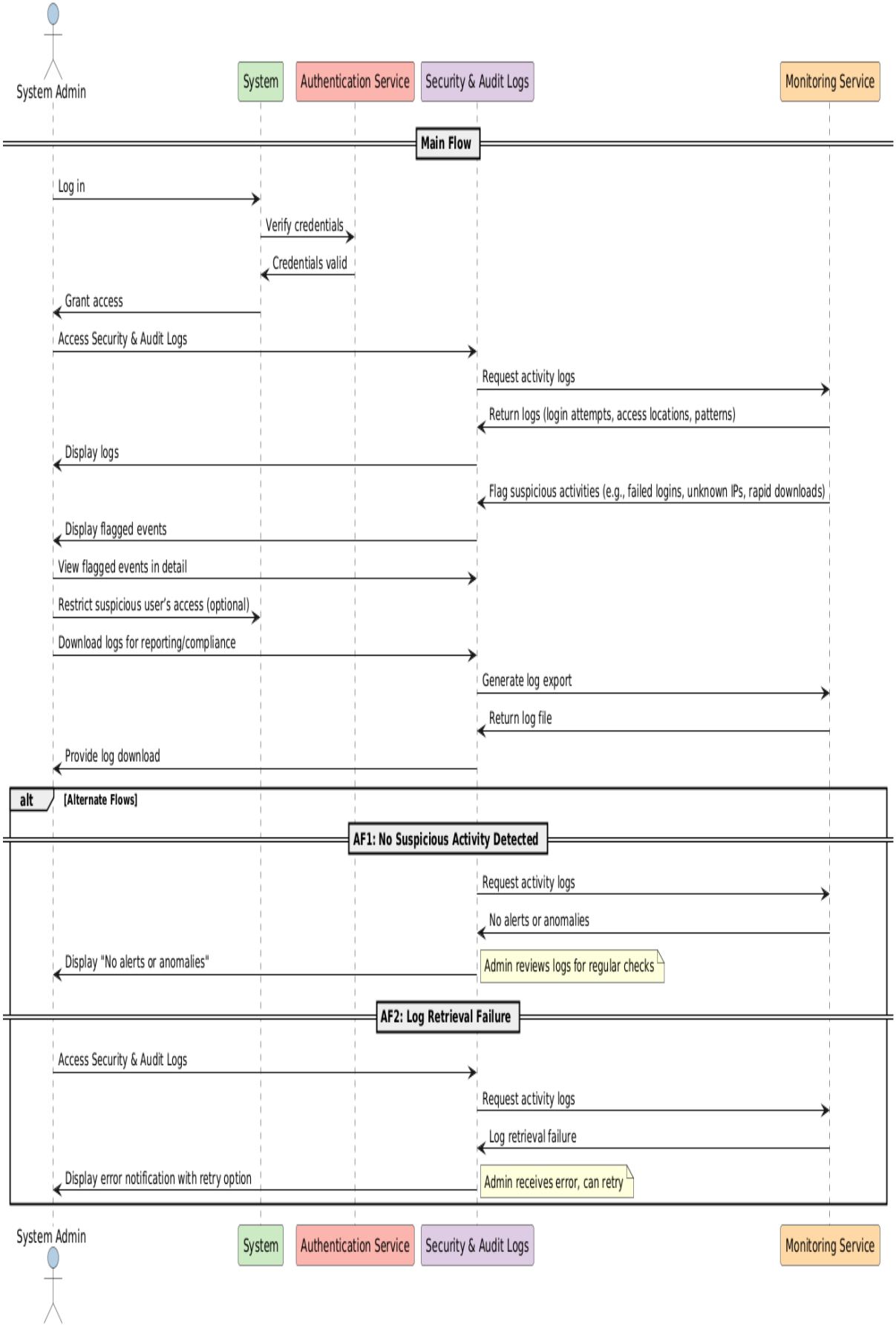
UC-A1:Monitor all system activities



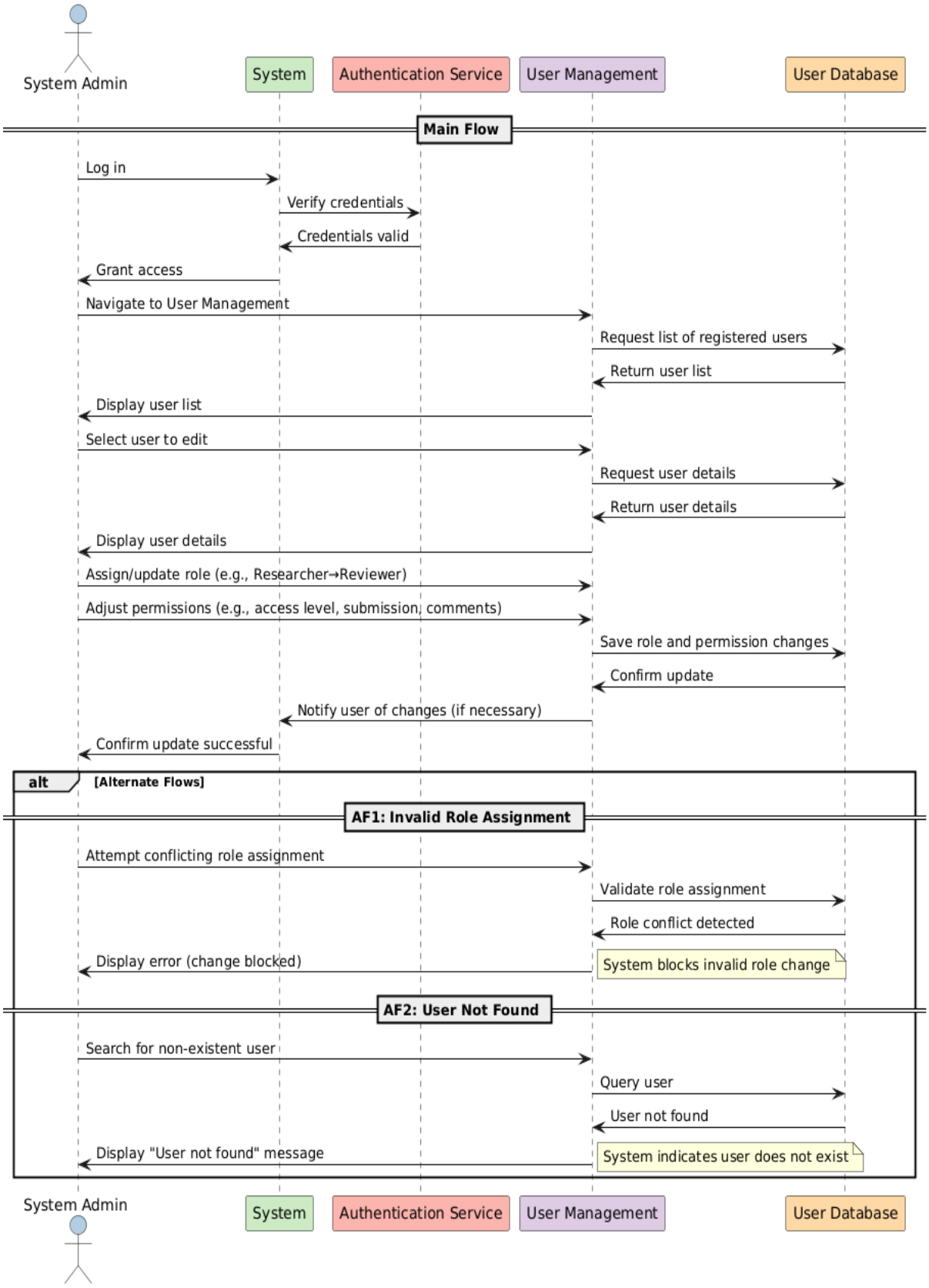
UC-A2 : Access Analytics on Submissions



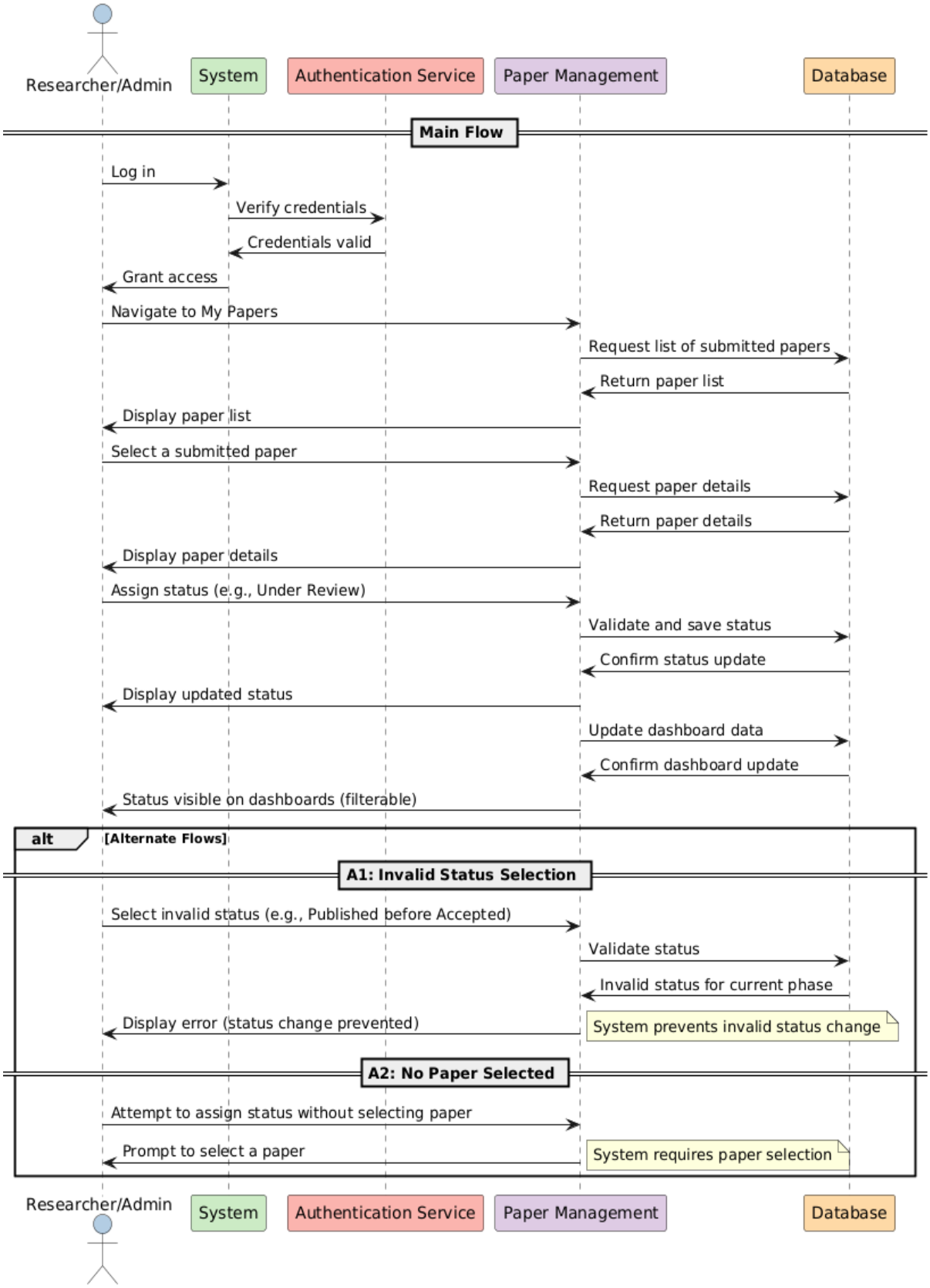
UC-A3 : Detect Unauthorized Access or Suspicious Behavior



UC-A4 : Manage User Roles and Permissions



UC-A5 ：Assign and Track Paper Status



◆Design Rationale

Choice of Software Process Model:

Decision: **Incremental Software Development Model**

Alternatives Considered:

1. Waterfall Model: A linear approach where each phase (requirements, design, implementation, testing) is completed sequentially before moving to the next.

Strengths	Weaknesses
• Clear structure • well-suited for projects with stable requirements • easy to manage for small teams.	• Inflexible to changes • late delivery of working software • high risk if requirements evolve (e.g., adding new AI features).

2. Agile (Scrum): Iterative development with short sprints, emphasizing frequent stakeholder feedback and adaptability.

Strengths	Weaknesses
• Highly flexible • supports continuous feedback • delivers small, functional increments quickly.	• Requires significant team coordination which could be challenging for a student-led project with limited time. May lead to scope creep without strict control.

3. Incremental Model: Delivers the system in small, functional increments, allowing early deployment of core features and iterative addition of advanced features.

Strengths	Weaknesses
• Early delivery of working software • reduced risk through incremental testing • flexibility to incorporate feedback (e.g., from Professor Ahmed).	• Requires careful planning to define increments • integration of later features (e.g., AI components) may introduce complexity.

◆ Rationale for Choice:

The Incremental Model was chosen because it aligns with the project's need for progressive feature delivery. It supports early deployment of essential functionalities (e.g., paper upload, search) while allowing advanced AI features (e.g., metadata extraction, summarization) to be added in later increments. The model accommodates continuous feedback from stakeholders, reducing risk by isolating potential issues to specific increments. Unlike Waterfall, it handles evolving requirements, and unlike Agile, it requires less frequent coordination, suiting the student team's constraints.

◆ Trade-offs:

Pros	Cons
Early delivery provides immediate value to users (e.g., Researchers can start uploading papers). Incremental testing reduces the risk of major failures. Scalability allows for future AI enhancements without disrupting core functionality.	**Planning increments requires upfront effort to define feature priorities. Integration of AI features (e.g., NLP for search) may introduce technical debt if not carefully designed.**