UNIVERSITY OF LUXEMBOURG

# Software Engineering 1

# Research Showcase Portal: Deliverable 1

**2025-10-31**

## Team data

**Team number:** 10
**GITHUB:** SE1-Research-Showcase-Portal

| Student number | First name | Last name |
|---|---|---|
| 0231297359 | Yaraslaw | Akhramenka |
| 0231364350 | Adelaide | Danilov |
| 0220502202 | Demyan | Faguer |
| 0230901309 | Oleksandr | Marchenko Breneur |
| 023159503B | Oleksandr | Yeroftieiev |

## Requirements Identification

### Functional Requirements

1. **Account**: Users can register, log in, and reset passwords.

2. **Profiles**: Users have posts and reports history. Each user has a profile with, optionally, personal information (ORCID, arXiv, social networks etc.).

3. **Institutional Verification**: Optional verification via institutional email for researchers.

4. **Create Research Post**: Authenticated users can commit research papers. Each research paper must have a title, authors, abstract and may have attachments and tags. After committing, the post is immediately visible in the search.

5. **Becoming a Researcher**: If an authorized user has at least one published research paper with 3 positive reviews from the researchers, they become a researcher and can now review other posts.

6. **Comments & Threads**: Authenticated users can comment and reply; target a nested/threaded view.

7. **Voting**: Upvote/downvote on posts and comments; score updates immediately with one vote per user per item (comment or post).

8. **Reporting & Moderation**: Users can report Terms & Conditions (T&C) violations; moderators can review, delete or allow an item.

9. **Tagging & Discovery**: Tags are an optional attribute that allows faster querying by the users.

10. **Search**: Keyword search over titles; sort by recency. Allow the filtering by author, publication date and tags through the UI.

11. **Share/Cite**: Generate share links and citations (BibTeX format) for posts.

12. **Review**: Researchers can provide their reviews for the published posts of other users, which are displayed separately from comments.

### Non-functional Requirements

1. **Performance**: Search response and page rendering should be performed within 1 and 2 seconds respectively.
2. **Scalability**: An API capable of horizontal scaling.
3. **Reliability**: Durable storage for posts/comments.
4. **Security**: Utilize industry standard practices for account data (for instance passwords) protection (such as Argon2 hashing with salt). Use rate-limits to minimize chances of DoS attacks, keep clear security logging.
5. **Usability**: Intuitive UI, clear error messages.
6. **Maintainability**: Clean module boundaries, formal automated and manual testing, target 80% code coverage.

# Requirements Elicitation

## Stakeholder Roles

1. **Anonymous User**: Can search, read, and share/cite published research.
2. **Registered User**: All of the above, plus comment, publish research (elevating to the Researcher role after 3 positive reviews from other researchers), up/down vote, and report T&C violations, receives an *Author* plaque shown on their posts and when replying within their own threads.
3. **Researcher**: All of the above; researchers may review the research posted by other users; may mark affiliation, verify the account with an institutional email.
4. **Moderator**: Monitors content, handles T&C reports, manages tags, reviews and approves verification requests, and applies moderation actions; Approve the researchers initially to avoid cold-start problem.

## User Stories

1. As a user, I want to search by keywords and tags so I can discover relevant research.
   *Expectations:* Given indexed posts exist, when I search for ”Theory of Relativity”, a results page renders within 1s with items whose titles or tags match; results are sorted by recency by default. I can adjust the results by applying the filters such as post date or author through the UI without the full page reload.
2. As an anonymous user, I want to register, log in, and reset my password.
   *Expectations:* I can create an account, confirm via email, log in, and request a time-limited password reset link; errors are shown clearly.
3. As a registered user, I want to edit my profile (bio, ORCID, arXiv, social links) so others can understand my identity.
   *Expectations:* Profile edits persist and appear on my public profile; private fields (e.g., email) are not exposed.
4. As a researcher, I want to verify my institutional affiliation, so my posts display an author plaque.
   *Expectations:* After submitting an institutional email and confirming the challenge link, my profile shows the plaque and my replies in my own threads are labeled accordingly.
5. As a registered user, I want to publish a research post with title, abstract, links/files, and optional tags, so others can find and discuss it.
   *Expectations:* On submit, the post is durably stored, appears on its page, and becomes searchable and available for reviewing by researchers; attachments upload successfully and are retained.
6. As a registered user, I want to comment and reply in threads so I can discuss findings.
   *Expectations:* Comments and multi-level replies appear immediately without page reload; deleted items show a placeholder.

7. As a registered user, I want to upvote or downvote posts and comments.
   *Expectations:* I can vote once per item and change my vote; the score updates immediately and persists after refresh.

8. As a user, I want to report Terms & Conditions violations, so moderators can review inappropriate content.
   *Expectations:* I can submit a report with category and notes; moderators see it in a queue, review context, and mark the item as deleted or allowed; my report shows a resolved status.

9. As a user, I want to use tags for faster discovery so I can navigate topics efficiently.
   *Expectations:* While creating or editing a post, tag autocomplete suggests existing tags; clicking any tag opens a tag page with results and related tags.

10. As a user, I want to generate a share link and a BibTeX citation for a post so I can cite it elsewhere.
    *Expectations:* Each post provides a permanent share URL and a copy-to-clipboard BibTeX entry.

11. As a user, I want search and page loads to be responsive, so the site feels fast.
    *Expectations:* Typical indexed searches respond within 1s and post pages render within 2s; pagination or infinite scroll prevents UI stalls.

## Backlog

### Search & Discovery (Must)

1. Deliver fast keyword search over titles and tags with client-side filters for date, author, and tags (Must).

2. Results sort by recency, update without full page reload, and preserve state. Implement responsive search UI with loaders and accessible navigation (Must).

3. Add tag autocomplete and tag pages with related tags (Should).

4. Query results filtering through the UI by authors, tags and dates (Should).

5. Infinite scroll/pagination (Could).

### Accounts & Profiles (Must)

1. Provide registration with email verification, secure login/session management, and time-limited password resets (Must).

2. Profiles allow editing bio/ORCID/arXiv/socials with privacy safeguards (Should).

3. Optional activity/notifications can come later (Could).

### Publishing & Discussions (Must)

1. Authenticated users can publish research (title, abstract, links/files, optional tags) with durable storage and immediate availability in search (Must).

2. Researchers can post their reviews on the respective publication page; If there are more than 3 positive reviews on the page of the publication, then the author is promoted to the researcher (Must).

3. Comments render instantly and show moderated placeholders when removed (Must).

4. File upload supports PDFs/common archives with size/type checks and retention (Must).

5. Comments support multi-level threads (Should).

### Moderation & Reporting (Must)

1. Users can report T&C violations with category and notes (Must)

2. Moderators review a queue, inspect context, and mark items deleted/allowed; actions are logged and reporter sees resolution (Must)

### Verification & Plaques (Should)

1. Institutional email verification grants a *Verified* plaque on profile. *Author* plaque is added to the comments, and replies within own threads (Should).

### Voting & Reputation (Must)

1. Users can up/down vote posts and comments with idempotent, single-vote semantics; scores update instantly and persist (Must).

### Sharing & Citation (Should)

1. Each post exposes a stable share URL and copy-to-clipboard BibTeX; include DOI/URL when available (Should).

## Sprints

1. General architecture discussions and implementation.
2. Core functionality: authentication, roles, post creation and display.
3. Further functionality: search through the search bar, filters, tags, comments/replies, voting, reporting.
4. Non-critical functionality: verification plaques, share + BibTeX, pagination.

# Use-case Models

## UC-01 Tag and Keyword search

**Primary Actor**: Anonymous User, Registered User, Researcher
**Goal**: Discover relevant research.
**Preconditions**: Indexed posts exist.
**Trigger**: Visitor enters a query (e.g., "Theory of Relativity").
**Main Success Scenario**:

1. Visitor types keywords and/or selects tags.
2. System queries the Search Index.
3. System returns results within **1s** sorted by recency.
4. Visitor adjusts filters (date, author) via UI controls.
5. Results update client-side without full page reload.

**Postconditions**: Result list visible; filters reflected in URL state.

## UC-02 Register

**Primary Actor**: Anonymous User.
**Preconditions**: Unique email not in use.
**Trigger**: Visitor submits sign-up form.
**Main Success Scenario**

1. Visitor provides email, password; accepts T&C.
2. System creates pending account; sends verification email.
3. Visitor clicks time-limited verification link.
4. System activates account; signs user in.

**Postconditions**: Account active, session established.

## UC-03 Authentication

**Primary Actor**: Registered User, Researcher, Moderator.
**Preconditions**: Active account.
**Trigger**: User submits credentials.
**Main Success Scenario**

1. User enters email/password or uses SSO.
2. System verifies and issues session token.
3. User is redirected to prior or default page.

**Postconditions**: Authenticated session active.

## UC-04 Password Reset

**Primary Actor**: Registered User, Researcher.
**Preconditions**: User is known to system.
**Trigger**: Actor requests password reset.
**Main Success Scenario**

1. Actor submits email.
2. System sends time-limited reset link.
3. Actor opens link, sets new password.
4. System invalidates older sessions and confirms success.

**Postconditions**: Password updated; session optional.
**Includes**: Authentication

## UC-05 Edit Profile

**Primary Actor**: Registered User, Researcher
**Preconditions**: Logged in.
**Trigger**: User saves profile edits.
**Main Success Scenario**

1. User edits bio, ORCID, arXiv, social links.
2. System validates and stores public vs. private fields.
3. Public profile updates immediately.

**Postconditions**: Profile changes persisted; private data hidden.
**Includes**: Authentication

## UC-06 Verify Institutional Affiliation

**Primary Actor**: Researcher
**Preconditions**: Logged in; institution email entered.
**Trigger**: User requests verification.
**Main Success Scenario**

1. User provides institutional email.
2. System sends challenge link to that email.
3. User confirms via link.
4. System marks affiliation verified; adds author plaque and label for replies in own threads.

**Postconditions**: Profile shows verified plaque; metadata applied; able to review papers of other users.
**Includes**: Authentication

## UC-07 Publish Research Post

**Primary Actor**: Registered User, Researcher
**Preconditions**: Logged in; form data valid.
**Trigger**: User submits new post.
**Main Success Scenario**

1. User enters title, abstract, links/files, optional tags.
2. System uploads attachments to Storage; computes checksums.
3. System stores post transactionally in DB.
4. System enqueues indexing; post page renders.
5. Post appears in search and on tag pages.

**Postconditions**: Post stored and discoverable for other users.
**Includes**: Authentication

## UC-08 Comment

**Primary Actor**: Registered User, Researcher, Moderator
**Preconditions**: Logged in; target post exists.
**Trigger**: User submits comment.
**Main Success Scenario**

1. User enters comment or reply.
2. System stores item and updates thread.
3. UI renders new item, no full reload.

**Postconditions**: Comment/reply persisted; visible in thread.
**Includes**: Authentication

## UC-09 Vote

**Primary Actor**: Registered User, Researcher
**Preconditions**: Logged in; item exists.
**Trigger**: User clicks upvote/downvote.
**Main Success Scenario**

1. User casts a vote.
2. System records one vote per user per item (insert/update).
3. Score updates and persists across refresh.

**Postconditions**: Vote state stored; aggregate score updated.
**Includes**: Authentication

## UC-10 Generate Citation/ Share Link

**Primary Actor**: Anonymous/Registered User/Researcher
**Preconditions**: Post exists and has stable ID/metadata.
**Trigger**: User opens share/citation options.
**Main Success Scenario**

1. System shows permanent share URL.
2. System renders BibTeX entry from post metadata.
3. User copies to clipboard.

**Postconditions**: The citation/link is displayed to the user and copied to the clipboard.

## UC-11 Review the research post

**Primary Actor**: Researcher.
**Preconditions**: Post exists and has stable ID/metadata;
**Trigger**: Researcher opens the review menu
**Main Success Scenario**

1. Researcher writes a review in a dedicated field.
2. The system saves the review in a review table.
3. The review is displayed on the research post page.
4. If it is the 3rd positive review of this publication, then its author is promoted to the researcher.

**Postconditions**: The research post has a new review from the Researcher. The registered user may be promoted to the researcher (if the post gets 3 positive reviews).
**Includes**: Authentication.

## UC-12 Report T&C Violation

**Primary Actor**: Anonymous User, Registered User, Researcher
**Preconditions**: Target item visible.
**Trigger**: User submits report with category and notes.
**Main Success Scenario**

1. User opens report dialog and selects category, adds notes.
2. System records report and queues it for Moderators.
3. Moderator reviews context in queue (see Figure 8).
4. System notifies reporter of resolution; item marked deleted or allowed.

**Postconditions**: The report is sent.

## UC-13 Moderate Reports

**Primary Actor**: Moderator
**Preconditions**: Moderator authenticated with privileges.
**Trigger**: Moderator opens moderation queue.
**Main Success Scenario**

1. Queue lists reports with context (item, history, reporter notes).
2. Moderator views item/thread and author info.
3. Moderator selects action: allow, delete (soft), or escalate/ban.
4. System applies action, leaves placeholder on deletion.

**Postconditions**: Queue item resolved; audit trail recorded.
**Includes**: Authentication System records report with target reference (targetType/targetId) and queues it for Moderators… Moderator reviews context… System notifies reporter of resolution; item marked deleted or allowed.

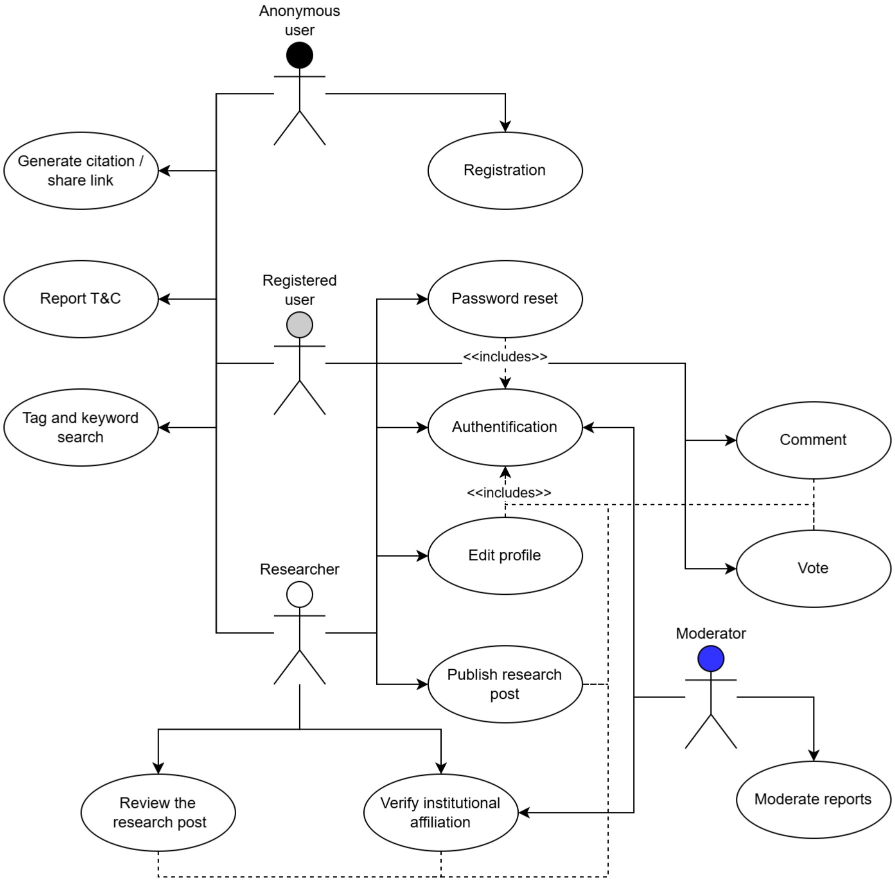

Figure 1: Use-case Model

# Behavioral Aspects

The UML diagrams depicting the behavioral aspects for each of the use-cases are presented in the Figures 3 through 8 in the appendix.

# Domain Model

## Relationships (Semantics and Multiplicities)

This subsection explains why each association exists and how its multiplicities reflect the real world:

- **User →Post (poster / authoredPosts)** ***1 ↔ 0..****
  A published post must have exactly one poster (i.e. the author; not to be confused with the plain text `authors` attribute, which lists the authors' names and is only used for display purposes, hence a quirky name).

- **User →Comment (commenter / authoredComments)** ***1 ↔ 0..****
  A published comment must have exactly one author; any user may comment as many posts as they want.

- **Post →Comment (comments / commentedPost)** ***1 ↔ 0..****
  Every comment belongs to exactly one post (context is required), while a post may accumulate many comments or none.

- **(Optional) Comment nesting** ***parent/replies***
  Threading is modeled by `Comment.parent : Comment`. We omit the explicit arrow for diagram clarity; conceptually it is *0..1* parent ↔ *0..** replies.

- **Researcher →Review (reviewer / authoredReview)** *1 ↔ 0..**
  Researchers have a power to write an extensive critical review of the post, each review must have exactly one author and each researcher can write several reviews.
- **Post →Tag (tags / posts)** *0..* ↔ 0..**
  A post can carry multiple tags, and a tag can describe many posts. This supports flexible discovery and filtering.
- **Moderator →Report (handledReports / responsible)** *0..* ↔ 0..1*
  A report may be unassigned (*0*) or assigned to exactly one moderator. A moderator can handle many reports.
- **Post →Attachment (attachments / post)** *0..* ↔ 1*
  Each attachment is tied to one post; a post can have none or many attachments (PDFs, images, archives).
- **User →Profile (profile / user)** *1 ↔ 1*
  Each user has exactly one profile, and each profile belongs to exactly one user. This separation isolates personal and institutional data (e.g., email, ORCID, arXiv links) from authentication, posts, and reports logic.
- In this model, the `Report` class is abstracted from the specific items it can reference. A report may concern a post, a comment, a user, or any other issue type (for example, tag misuse).

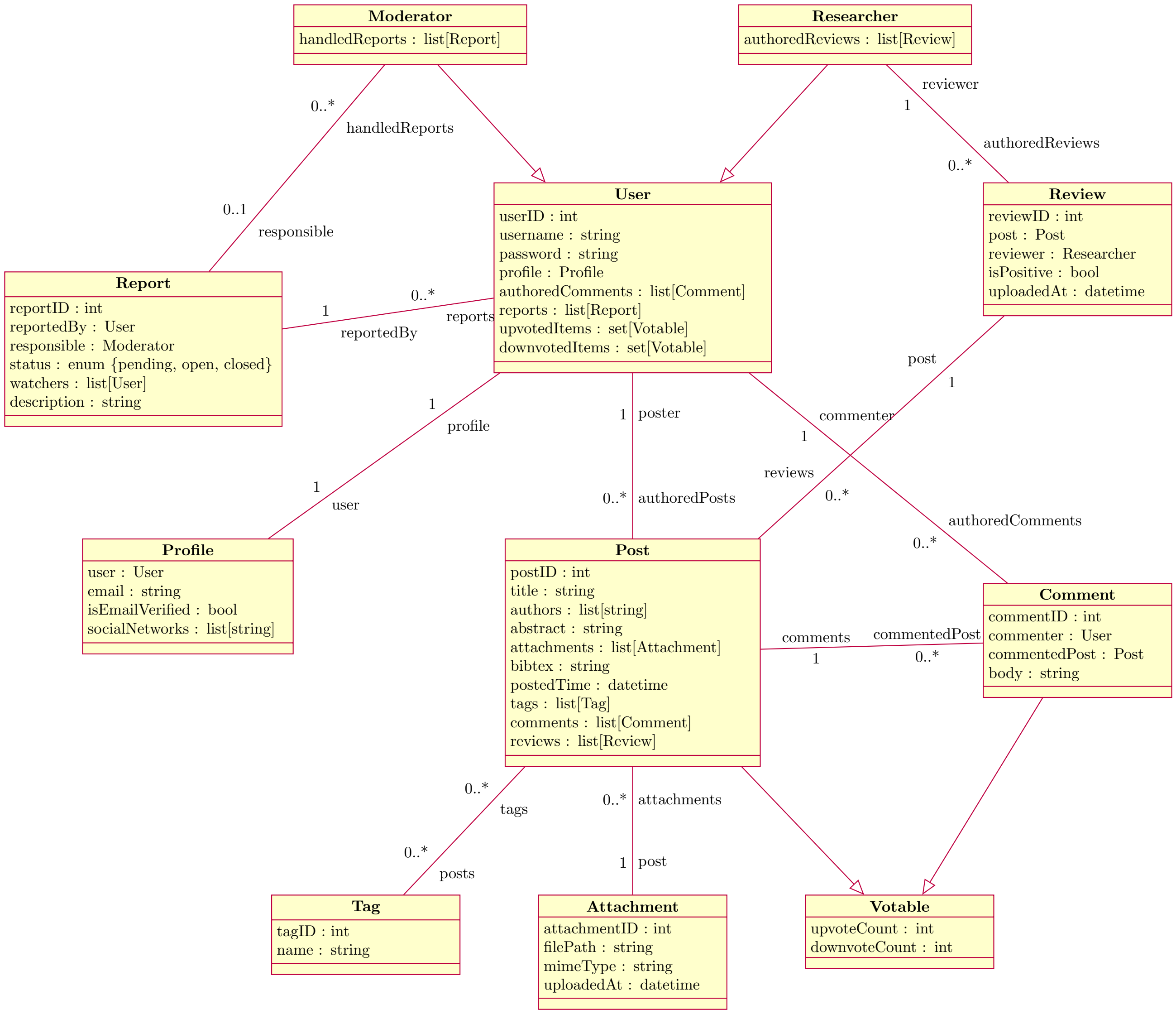

Figure 2: Domain Model

# Appendix

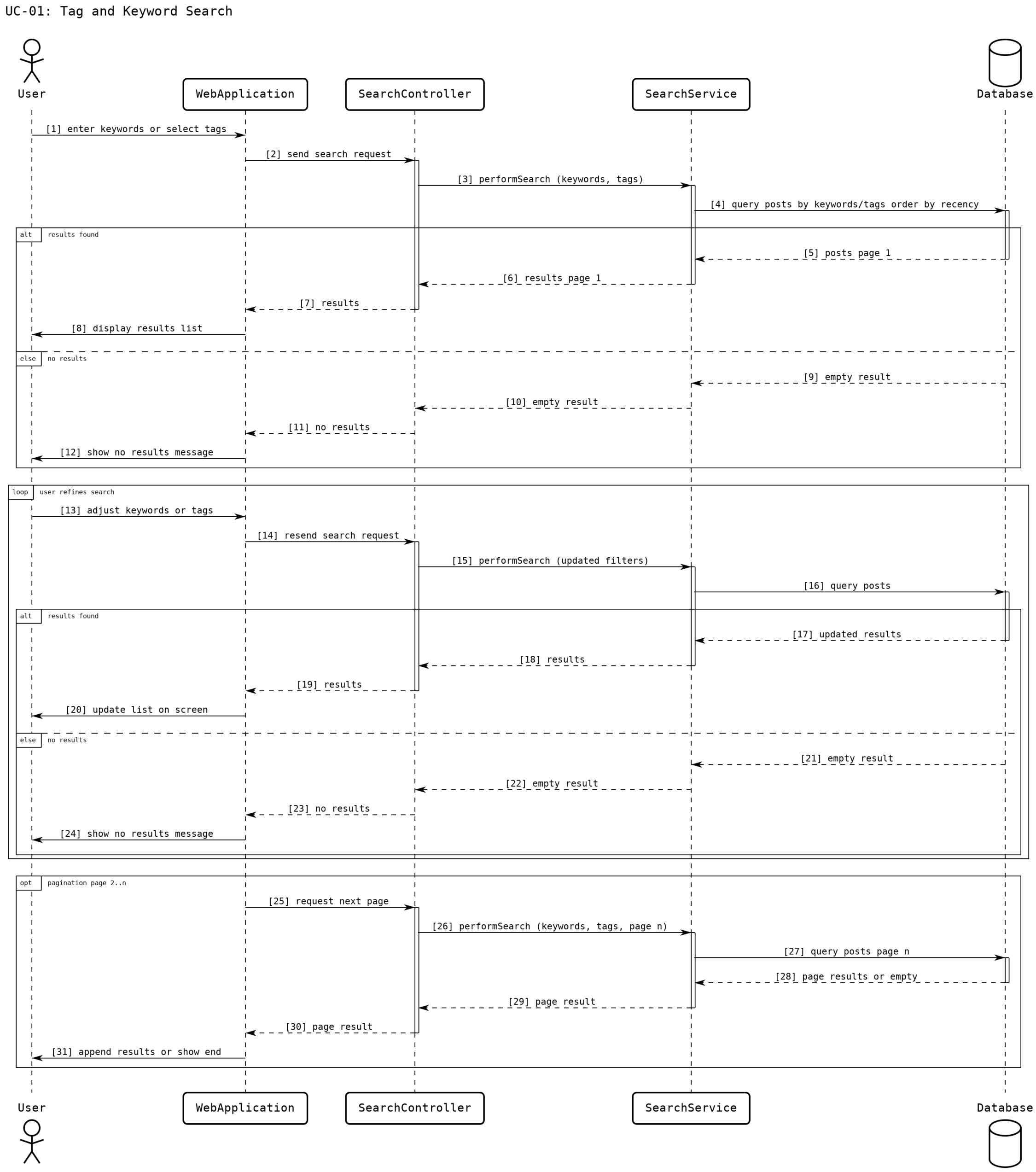

Figure 3: UC01 Search

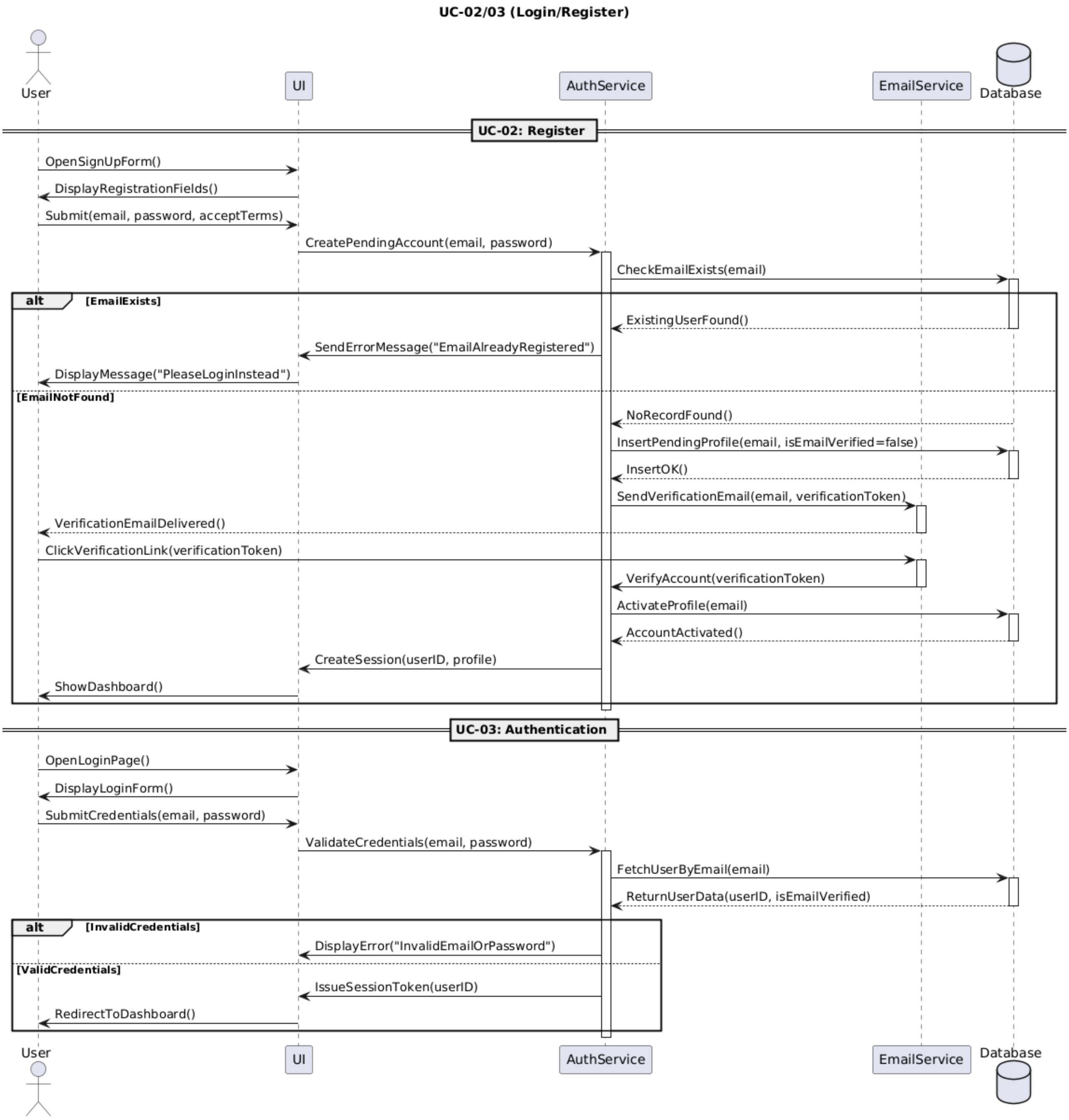
UC-02/03 (Login/Register)
User
UI
AuthService
EmailService
Database
UC-02: Register
OpenSignUpForm()
DisplayRegistrationFields()
Submit(email, password, acceptTerms)
CreatePendingAccount(email, password)
CheckEmailExists(email)
alt
[EmailExists]
ExistingUserFound()
SendErrorMessage("EmailAlreadyRegistered")
DisplayMessage("PleaseLoginInstead")
[EmailNotFound]
NoRecordFound()
InsertPendingProfile(email, isEmailVerified=false)
InsertOK()
SendVerificationEmail(email, verificationToken)
VerificationEmailDelivered()
ClickVerificationLink(verificationToken)
VerifyAccount(verificationToken)
ActivateProfile(email)
AccountActivated()
CreateSession(userID, profile)
ShowDashboard()
UC-03: Authentication
OpenLoginPage()
DisplayLoginForm()
SubmitCredentials(email, password)
ValidateCredentials(email, password)
FetchUserByEmail(email)
ReturnUserData(userID, isEmailVerified)
alt
[InvalidCredentials]
DisplayError("InvalidEmailOrPassword")
[ValidCredentials]
IssueSessionToken(userID)
RedirectToDashboard()
User
UI
AuthService
EmailService
Database

Figure 4: UC02/03 Login/Register

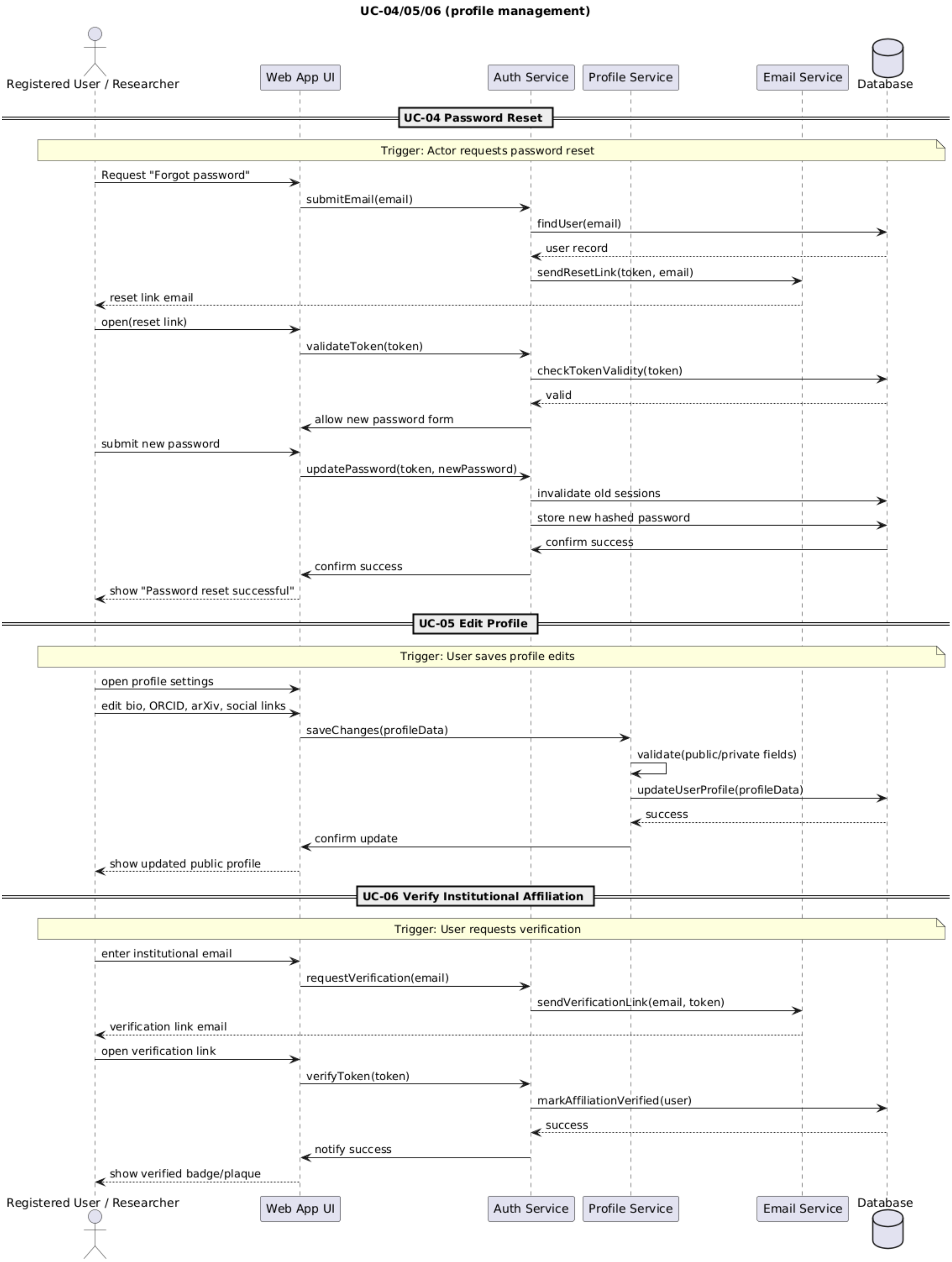
UC-04/05/06 (profile management)
Registered User / Researcher
Web App UI
Auth Service
Profile Service
Email Service
Database
UC-04 Password Reset
Trigger: Actor requests password reset
Request "Forgot password"
submitEmail(email)
findUser(email)
user record
sendResetLink(token, email)
reset link email
open(reset link)
validateToken(token)
checkTokenValidity(token)
valid
allow new password form
submit new password
updatePassword(token, newPassword)
invalidate old sessions
store new hashed password
confirm success
confirm success
show "Password reset successful"
UC-05 Edit Profile
Trigger: User saves profile edits
open profile settings
edit bio, ORCID, arXiv, social links
saveChanges(profileData)
validate(public/private fields)
updateUserProfile(profileData)
success
confirm update
show updated public profile
UC-06 Verify Institutional Affiliation
Trigger: User requests verification
enter institutional email
requestVerification(email)
sendVerificationLink(email, token)
verification link email
open verification link
verifyToken(token)
markAffiliationVerified(user)
success
notify success
show verified badge/plaque
Registered User / Researcher
Web App UI
Auth Service
Profile Service
Email Service
Database

Figure 5: UC04/05/06 Profile Management

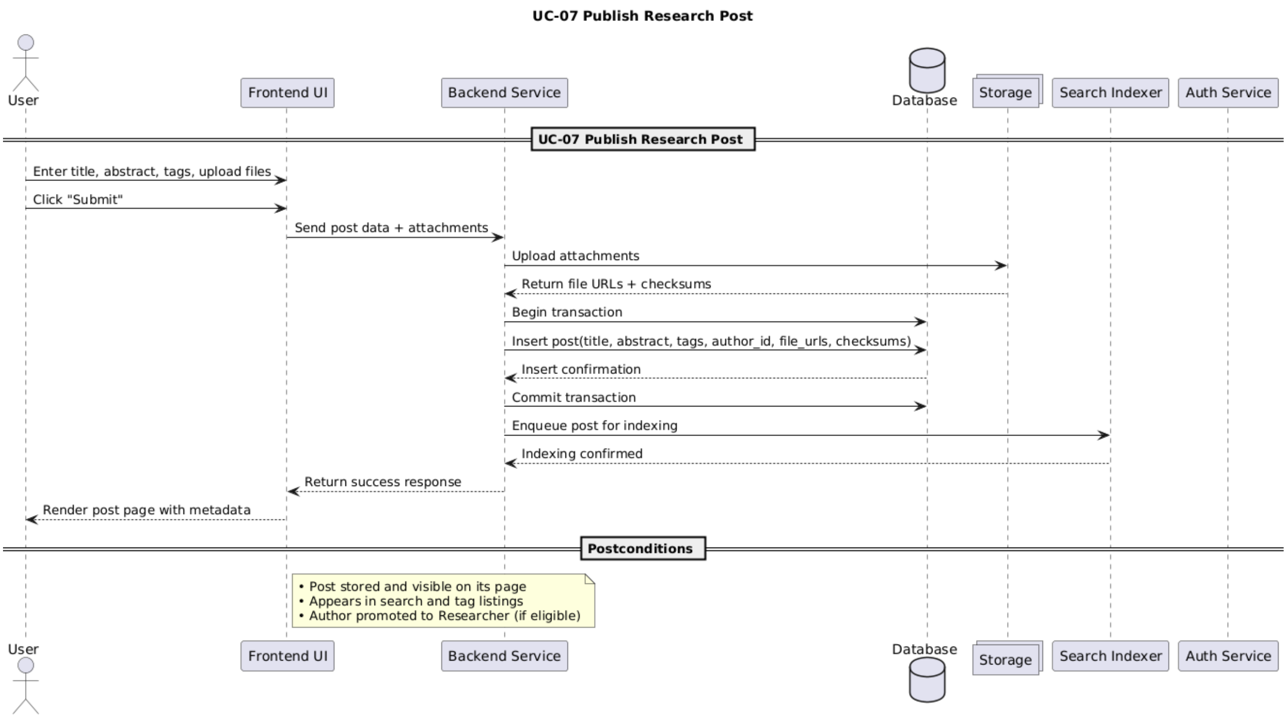
UC-07 Publish Research Post
User
Frontend UI
Backend Service
Database
Storage
Search Indexer
Auth Service
UC-07 Publish Research Post
Enter title, abstract, tags, upload files
Click "Submit"
Send post data + attachments
Upload attachments
Return file URLs + checksums
Begin transaction
Insert post(title, abstract, tags, author_id, file_urls, checksums)
Insert confirmation
Commit transaction
Enqueue post for indexing
Indexing confirmed
Return success response
Render post page with metadata
Postconditions
• Post stored and visible on its page
• Appears in search and tag listings
• Author promoted to Researcher (if eligible)

Figure 6: UC07 Post, Comment, Vote, Review, Cite

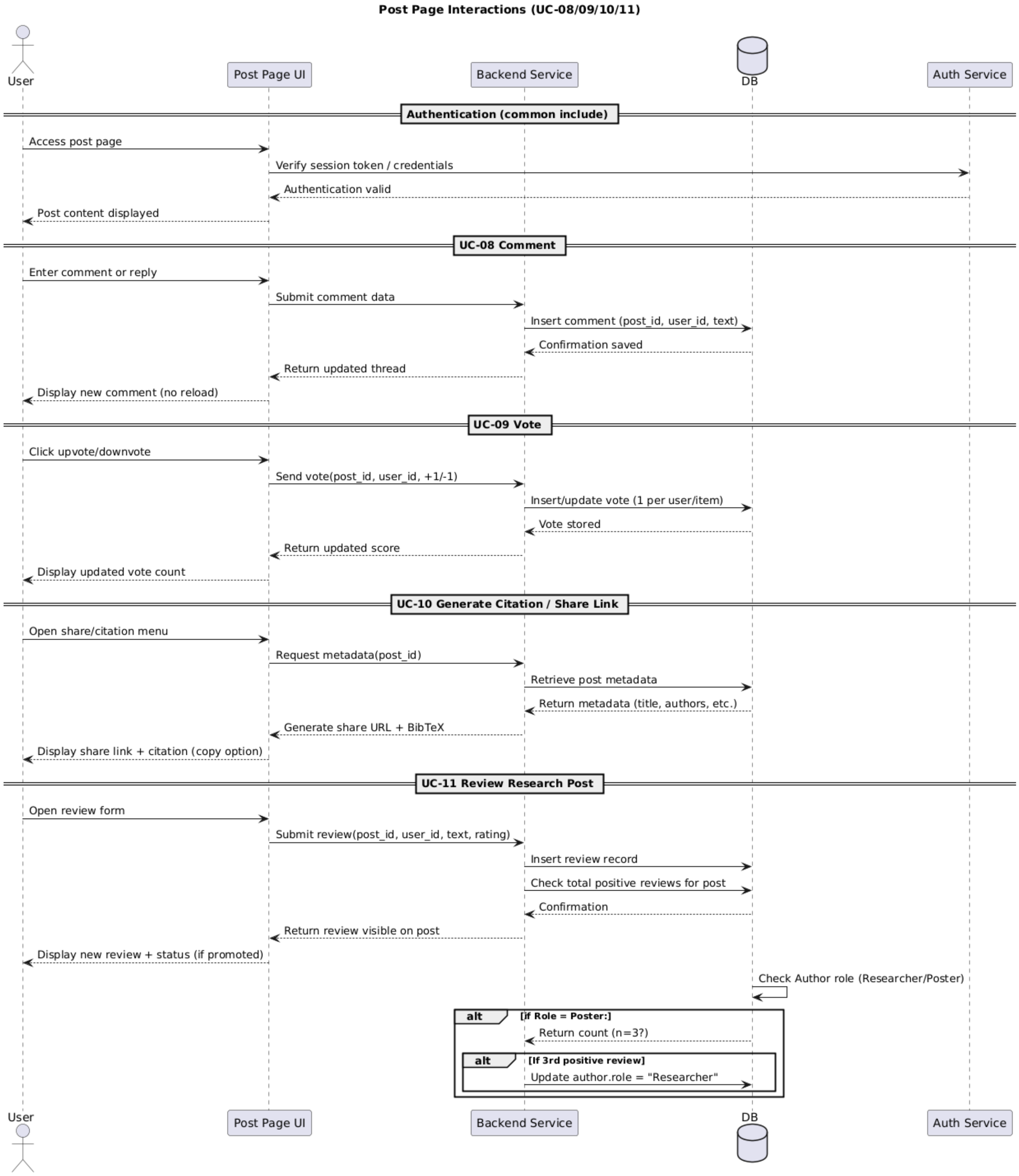
Post Page Interactions (UC-08/09/10/11)
User
Post Page UI
Backend Service
DB
Auth Service
Authentication (common include)
Access post page
Verify session token / credentials
Authentication valid
Post content displayed
UC-08 Comment
Enter comment or reply
Submit comment data
Insert comment (post_id, user_id, text)
Confirmation saved
Return updated thread
Display new comment (no reload)
UC-09 Vote
Click upvote/downvote
Send vote(post_id, user_id, +1/-1)
Insert/update vote (1 per user/item)
Vote stored
Return updated score
Display updated vote count
UC-10 Generate Citation / Share Link
Open share/citation menu
Request metadata(post_id)
Retrieve post metadata
Return metadata (title, authors, etc.)
Generate share URL + BibTeX
Display share link + citation (copy option)
UC-11 Review Research Post
Open review form
Submit review(post_id, user_id, text, rating)
Insert review record
Check total positive reviews for post
Confirmation
Return review visible on post
Display new review + status (if promoted)
Check Author role (Researcher/Poster)
alt
[if Role = Poster:]
Return count (n=3?)
alt
[If 3rd positive review]
Update author.role = "Researcher"
User
Post Page UI
Backend Service
DB
Auth Service

Figure 7: UC/08/09/10/11 Post

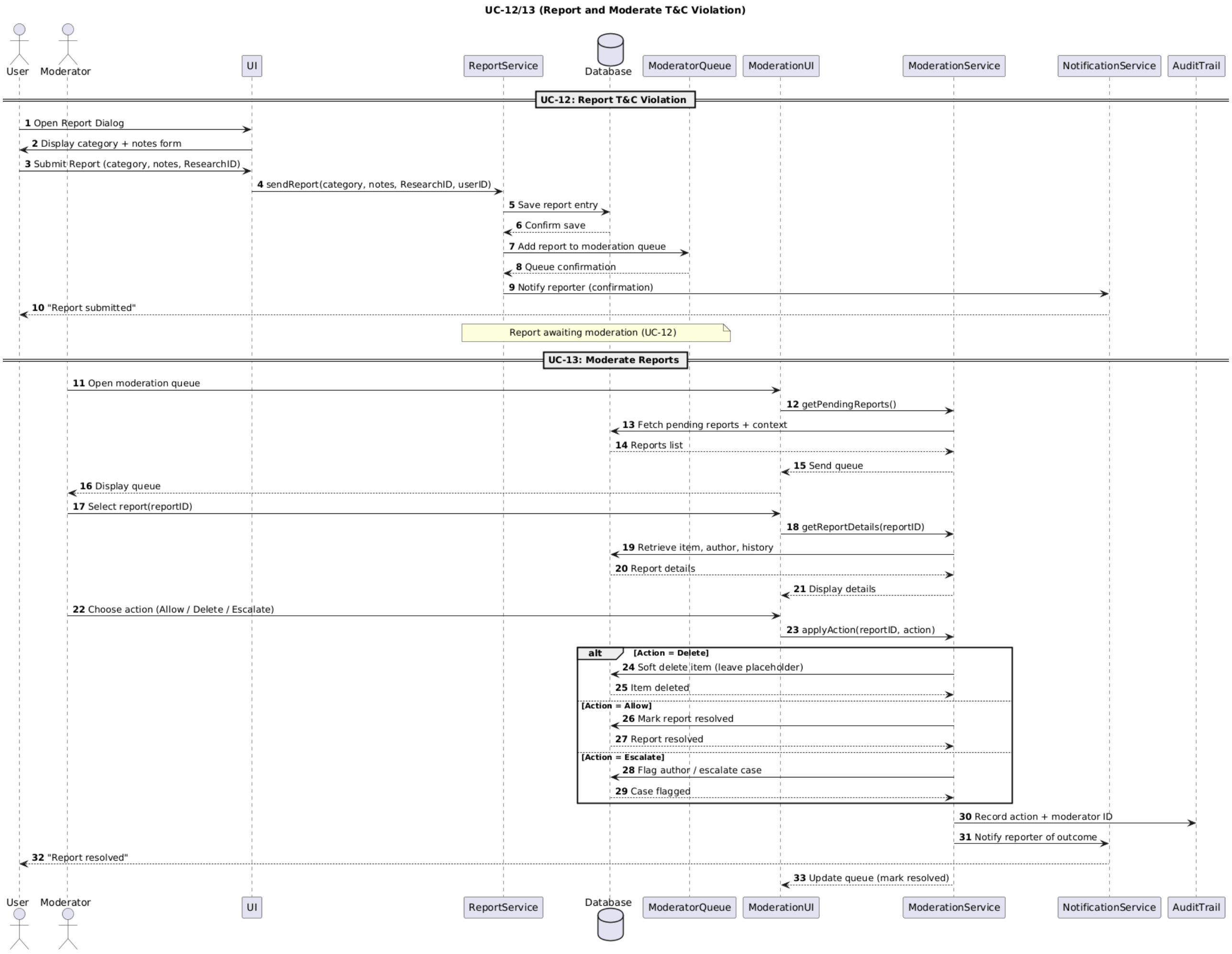
UC-12/13 (Report and Moderate T&C Violation)
User
Moderator
UI
ReportService
Database
ModeratorQueue
ModerationUI
ModerationService
NotificationService
AuditTrail
UC-12: Report T&C Violation
1 Open Report Dialog
2 Display category + notes form
3 Submit Report (category, notes, ResearchID)
4 sendReport(category, notes, ResearchID, userID)
5 Save report entry
6 Confirm save
7 Add report to moderation queue
8 Queue confirmation
9 Notify reporter (confirmation)
10 "Report submitted"
Report awaiting moderation (UC-12)
UC-13: Moderate Reports
11 Open moderation queue
12 getPendingReports()
13 Fetch pending reports + context
14 Reports list
15 Send queue
16 Display queue
17 Select report(reportID)
18 getReportDetails(reportID)
19 Retrieve item, author, history
20 Report details
21 Display details
22 Choose action (Allow / Delete / Escalate)
23 applyAction(reportID, action)
alt
[Action = Delete]
24 Soft delete item (leave placeholder)
25 Item deleted
[Action = Allow]
26 Mark report resolved
27 Report resolved
[Action = Escalate]
28 Flag author / escalate case
29 Case flagged
30 Record action + moderator ID
31 Notify reporter of outcome
32 "Report resolved"
33 Update queue (mark resolved)
User
Moderator
UI
ReportService
Database
ModeratorQueue
ModerationUI
ModerationService
NotificationService
AuditTrail

Figure 8: UC12/13 Report and Moderate T&C violations