HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SOFTWARE ENGINEERING

REQUIREMENT ANALYSIS REPORT

FLASHCARD WEBSITE

GROUP 17

Name	Student's ID
Ngô Duy An	20235883
Nguyễn Trọng Minh	20235976
Đồng Đại Huy	20235945
Nguyễn Việt Hòa	20235936
Phan Đăng Đạt	20235908

Instructor: Dr. Nguyễn Quốc Tuấn

CONTENTS

1. Problem Definition & Stakeholder Analysis

1.1. Project Objective

In modern learning, students and self-learners increasingly rely on online tools to support memorization and concept reinforcement. Among these, flashcards stand out as one of the most effective methods for active recall and spaced repetition. To provide them with a proper tool, this project aims to develop an interactive and user-friendly **Flashcard Website** that enables learners to **create, organize, and share flashcards and decks** efficiently.

The website will allow users to build decks consisting of two-sided flashcards (question–answer or term–definition), share decks publicly to support collaborative learning, and copy shared decks for personal customization. To ensure content quality and community standards, the system will include administrative features for **moderation and user management**, such as censoring inappropriate decks and sending warnings to violators. The overall objective is to create a reliable, scalable, and easy-to-use platform that encourages both **individual learning and community engagement**.

1.2. Stakeholders analysis

The project involves several key stakeholders, each plays a vital role in the system's design, implementation, and use:

- **End Users (Learners):** Create, edit, organize, and review flashcards and decks; share and copy decks for personal use. From their perspective, the system must provide a simple and user-friendly interface that supports quick content creation, efficient study workflows, and smooth interaction between users. Learners expect the platform to enhance study effectiveness through accessibility, collaboration, and flexibility.
- **Administrators:** Monitor public decks, manage user activities, censor inappropriate or harmful content, and send warnings to users who violate community standards. Administrators require a reliable set of tools that allow them to efficiently review, search, and manage content while maintaining the overall safety and integrity of the community.
- **Project Development Team:** Design, develop, and maintain the system, including frontend, backend, and database components. The developers focus on building robust, scalable, and maintainable applications that meet user and administrative requirements. We can achieve this through well-documented requirements and regular communication.

1.3. Overview of the report

This document is divided into four main chapters:

- **Chapter 1 – Introduction:** Outlines the problem and project's objective, stakeholders, and the structure of the report.
- **Chapter 2 – System Requirements:** Provides a detailed breakdown of functional requirements (e.g., account system, deck creation, reviews, search features) and non-functional requirements (e.g., performance, security, usability, and compliance).
- **Chapter 3 – System Model:** Illustrates the system's logical structure through diagrams such as the Data Flow Diagram (DFD), Finite State Machine (FSM), Entity-Relationship (ER) model, Use Case Diagram, and Activity Diagram, offering a visual representation of system behavior and data interaction.
- **Chapter 4 – Conclusion:** Summarizes the key findings and provides an overview of how the system meets the stated objectives and constraints.

2. Requirement Specification

2.1 Functional requirement

2.1.1 Account

FR-A1 Sign in

- **Description**: New users create an account by providing **display name**, **login name**, and **password** (display name 1–40 chars). Optional profile fields (profile picture, short bio) can be added after registration. The user record is stored in the database and is immediately available for login, personalization, and interaction across the platform.
- **Acceptance**:
    - Given a display name (1–40 chars), **unique login name**, and strong password, when signing up, then an **active** user record is created in the DB and a success confirmation is shown.
    - Given the newly created credentials, when the user signs in, then sign-in succeeds and the display name is visible in the UI header/profile.
    - Given a **login name** that already exists, when signing up, then the system shows a **login name already in use** error and no account is created.
    - Given an active session after first sign-in, when the user uploads a profile picture (≤ 5 MB) and/or enters a short bio (≤ 160 chars) and saves, then the profile updates persist and are visible after refresh.

FR-A2 Log in/Log out

- **Description**: Allow users to authenticate with **login name + password** to establish a session that **persists across page refresh** until it expires or they log out. Logging out **revokes session cookies/tokens** and immediately blocks access to protected routes, redirecting the user to the log in page.
- **Acceptance**:
    - Given an **active** account and correct credentials, when submitting, then a session is issued and persists across a refresh.
    - Given an **active** session, when clicking log out, then session cookies/tokens are revoked; protected pages redirect to log in.

2.1.2 Decks & Cards

FR-D1 Create Deck

- **Description**: Create a new deck owned by the current user with **title** and optional **description**. Duplicate titles are allowed; new decks default to **Private** and persist **`created_at`** and **`updated_at`** timestamps; these fields are shown on Deck Detail.
- **Acceptance**:
    - Given a signed-in user, when creating with **title** and optional **description**, then a **Private** deck is created and visible under **My Decks**, and the following fields are persisted: **description** (if any), **visibility** (Public or Private)
    - Given a duplicate title **for the same owner**, when creating, then duplicates are **allowed** and the deck is created without adding a suffix or returning an error.
    - Given a newly created deck, when opening the **Deck Detail** page, then it displays the deck **description** (if any), **visibility** chip (Public/Private), **creation time** (`created_at`), and **last update time** (`updated_at`); on creation, `updated_at` equals `created_at`.

FR-D2 Edit/Delete Deck

- **Description**: The owner can update deck metadata (title, description). Editing updates **`updated_at`**; deleting a deck removes its cards and study reviews and de-indexes it from Public.
- **Acceptance**:
    - **Edit with validation & normalization**: Given an owned deck, when updating **title**, **description**, and clicking **Save**, then:
        - changes persist to the database and are visible after reload,
        - duplicate titles **are allowed** for the same owner, and
        - `updated_at` changes while `id` remains the same and `created_at` remains unchanged.
    - **Delete with cascade & deindex**: Given a deck with cards, when **Delete** is clicked and the confirmation dialog is accepted, then:
        - the deck disappears from **My Decks** and from the **Public** catalog (if previously Public),
        - associated **cards** and **study reviews** are removed, and
        - searching the former deck title in the public catalog returns no results.

FR-C1 Add/Edit/Delete Card

- **Description**: Within an owned deck, the user can add cards, edit existing cards, and delete cards. New cards start with status new. Deleting a card removes its related study reviews. UI reflects changes immediately and API exposes updated state.
- **Acceptance**:
    - **Add**: Given an owned deck, when adding a card with and clicking **Save**, then:
        - the card appears in the deck's card list without page reload,
        - deck counters update (Total **+1**), and
        - the new card is scheduled as **new** for study.
    - **Edit**: Given an existing card, when changing **front/back** and saving, then:
        - the updated content is visible after reload,
        - `updated_at` changes while the card `id` stays the same, and
        - any leading/trailing whitespace is trimmed.
    - **Delete**: Given an existing card, when **Delete** is confirmed, then:
        - the card is removed from the list,
        - deck counters update (Total **−1**),
        - related **study reviews** for that card are removed, and

2.1.3 Study Session

FR-S1 Start Study Session

- **Description**: Start a simple study session for a selected deck. The session opens on the first card's **front** and allows linear navigation (**Next/Previous**) and **flip**. No scheduling or rating (Known/Repeat) is recorded.
- **Acceptance**:
    - Given a deck with ≥ 1 card, when clicking **Study**, then the first card's **front** is shown.
    - When clicking **Next/Previous**, the next/previous card's **front** is shown (bounds respected: no wrap unless explicitly enabled).
    - When the user opens Study mode, the system updates the deck's per-user **`last_seen`** timestamp to **now**.

FR-S2 Flip Card

- **Description**: In study mode, the learner can **flip** a card to reveal the back and flip again to return to the front. No Known/Repeat actions are available.
- **Acceptance**:
    - Given the card **front** is visible, when triggering **Flip** (button or key), then the **back** is revealed.
    - Given the **back** is visible, when triggering **Flip** again, then the **front** is shown.

2.1.4 Public catalog, Search & Clone

FR-PB1 Publish/Unpublish

- **Description**: The deck owner can toggle a deck's **visibility** between **Private** and **Public**. When set to Public, the deck is listed in the public catalog. Switching back to Private removes it from public listings and makes it inaccessible via catalog/ browse/ search.
- **Acceptance**:
    - Given an owned deck, when toggling **Public** and saving, then the deck appears in public listings.
    - Given the same deck toggled back to **Private**, when browsing/searching the catalog, then the deck **does not appear** and is no longer accessible from public views.

FR-PB2 Browse Public Deck

- **Description**: Visitors and signed-in users can browse the **Public** catalog to discover decks. The catalog shows a **paginated** list with deck title, author label, and brief stats. Opening a public deck shows its read-only detail view, including its cards; editing is not allowed from this view.
- **Acceptance**:
    - Given public decks exist, when opening the **Public** page, then a **paginated** list (default 20/page) with title, author label, brief stats appears.
    - When clicking a deck, then read-only details and its full cards are viewable (no editing).

FR-PB3 Search/Filter

- **Description**: Provide a **keyword search** over public decks limited to **title** and **tags.** Results are paginated and exclude full-text search on card bodies in the MVP.
- **Acceptance**:
    - Given the seed dataset, when searching by a keyword contained in **title.**
    - Given no matches, when searching, then an **empty state** is displayed.

FR-PB4 Clone Public Deck

- **Description**: A signed-in user can **clone** any public deck into their account. The system creates a **new Private copy** with all cards and metadata under the new owner. The clone is independent (new ids), and changes to the clone or the original do not affect one another.
- **Acceptance**:

- Given a signed-in user viewing a public deck, when clicking **Clone**, then a **new Private copy** appears in **My Decks** with all cards.
- Edits to the clone **do not** change the original deck.

FR-PB5 Rate Public Deck

- **Description**: On a public deck's detail view, a **signed-in** user rates via an **intuitive star widget**: click/tap to fill **1–5** stars. Each user has **at most one rating per deck** and may update it by clicking a different star. Show the **average rating (to one decimal)** and **total ratings count** on the deck detail.
- **Acceptance**:
  - Given a signed-in user on a public deck, when clicking a star in the 5-star widget, then exactly that many stars appear filled and the rating is saved (per user, per deck); the **average** and **count** update immediately.
  - Given the user has rated before, when clicking a different star, then their prior rating is replaced, the average/count recompute, and the widget reflects the new number of filled stars.

FR-PB6 Comment on Public Deck

- **Description**: On a public deck's detail view, a **signed-in** user can **post comments** (text up to 500 chars). Comments display newest first with author display name and timestamp. A commenter can **delete their own** comment.
- **Acceptance**:
  - Given a signed-in user, when posting a comment (≤ 500 chars), then the comment appears at the top of the list with their display name and current timestamp and is persisted.
  - Given a comment authored by the signed-in user, when clicking **Delete**, then the comment is removed from the list and cannot be fetched by API (404) within **5 seconds**.

2.1.5 AI Assist

FR-AI1 Generate AI Answer

- **Description**: In the card editor, when the user enters a question on the **Front** of a flashcard, they may optionally click **Ask AI** to get an answer suggestion. The system calls the **Gemini API** with the front text (and optional deck context) and returns **one proposed answer**. The answer appears in a **popup** beside the front field. If the user clicks the **"✓"** button in the popup, the suggested answer is **auto-filled into the Back field** (prefill) for further editing; otherwise the user can ignore/close the popup and nothing is inserted. This is a synchronous, **suggest-only** feature (no auto-publish) with a **≤ 15s** target response and a visible caution that AI may be inaccurate.

- **Acceptance**:
    - Given a card with **Front** text (1–500 chars), when the user clicks **Ask AI**, then a popup displays either a suggested answer within **≤ 15s** or an error message; a loading indicator is shown during the call.
    - Given the popup shows an answer, when the user clicks **"✓"**, then the **Back** field is populated with the suggested answer (trimmed) and the popup closes.
    - Given the popup shows an answer, when the user clicks **"×"** (or **Ignore**), then the **Back** field remains unchanged and the popup closes.
    - Given the answer was inserted, when the user edits the **Back** field and clicks **Save**, then the edited content is persisted in the card.
    - The popup contains a visible disclaimer: "AI output may contain errors; please review."

2.1.6 Monitoring

FR-ADM1 Admin Account Provisioning

- **Description**: Administrator accounts are **pre-provisioned** (seeded) by the system at deployment; users cannot self-elevate to admin via UI or API. At least **one admin** must exist in all environments.
- **Acceptance**:
    - Given a fresh deployment with seed data, when signing in with the provided **admin** credentials, then role = admin and the **Admin Panel** is accessible.

FR-ADM2 Admin Role & Access

- **Description**: **Administrator** role login with the **same Sign-in form** as regular users; upon successful login the system **auto-detects the admin role** and shows an **Admin Panel** entry and moderation controls on relevant pagesThe admin UI mirrors the user UI with additional controls (delete/hide/show public decks, delete/hide/show comments, lock/unlock accounts, send warnings).
- **Acceptance**:
    - Given an **admin** account, when signing in **via the standard Sign-in page**, then an **Admin Panel** link is visible in the navigation

FR-ADM3 Moderate Decks & Comments

- **Description**: Admins can **Hide/Show** and **Delete** both **public decks** and their **comments**. Hiding a deck unpublishes it (becomes **Private**, removed from catalog/search); showing republishes it (**Public**). Deleting a deck permanently removes the deck and cascades its **cards**, **public comments**, and **ratings**. Hiding a comment makes it invisible to regular users (still visible to admins with a **Hidden** badge); showing restores visibility. Deleting a comment removes it permanently.

- **Acceptance**: **Decks**
    - **Hide**: Given an admin on a public deck detail, when clicking **Hide** and confirming, then the deck disappears from the Public catalog/search. The owner sees it in **My Decks** as **Private** with a **Removed by Admin** note.
    - **Show**: Given a previously hidden deck, when an admin clicks **Show** and confirms, then the deck reappears in the Public catalog/search and the owner sees visibility **Public** again.
    - **Delete**: Given an admin on a public deck detail, when clicking **Delete deck** and confirming, then `GET /api/decks/{id}` returns 404 and the deck is removed from the owner's My Decks; associated **cards**, **comments**, and **ratings** are removed.
- **Acceptance**: **Comments**
    - **Hide**: Given an admin viewing a deck's comments, when clicking **Hide** on a comment and confirming, then the comment disappears for non-admin users.
    - **Show**: Given a hidden comment, when the admin clicks **Show**, then the comment becomes visible to all users.
    - **Delete**: Given an admin clicks **Delete** on a comment and confirms, then the comment disappears and `GET /api/comments/{id}` returns **404**.

FR-ADM4 Lock/Unlock Accounts

- **Description**: Admin can **lock** a user account (prevent sign-in and revoke active sessions) and **unlock** it. Locking is recorded with a reason.
- **Acceptance**:
    - Given an admin locks a user, when that user attempts to **Sign-in**, then access is **denied** with a message "Account locked".
    - Given the user had an active session, when making the next authenticated API call after being locked, then the call is rejected and the session is invalidated (user is forced to Sign-in screen).
    - Given the account is **unlocked** by admin, when the user signs in with correct credentials, then access **succeeds**.

FR-ADM5 Send Warnings

- **Description**: Admin can **send a warning** to a user for violations. The warning is stored as an in-app notification and shown as a **banner** on the user's next sign-in (and on the Account page) until dismissed.
- **Acceptance**:
    - Given an admin composes a warning and clicks **Send**, then the warning is stored with user_id, admin_id, warning_id, action, reason, created_at.
    - Given the warned user signs in, then a **warning banner** appears until dismissed.

2.2 Non-Functional Requirements

Non-functional requirements describe the qualitative characteristics and constraints that ensure the Flashcard Website operates efficiently, securely, and reliably. They define how the system should perform rather than what it should do, supporting overall user satisfaction, scalability, and maintainability.

2.2.1 Performance Requirements

Description:
The system must deliver fast and consistent performance when users interact with core features such as browsing decks, starting study sessions, or generating AI-assisted content.

Rationale:
Efficient performance enhances the user experience, prevents frustration, and encourages consistent study habits. As the platform supports learners worldwide, minimizing latency is essential for engagement and accessibility. A fast system also supports scalability as more users join.

Acceptance Criteria:

- Page load time for Home, Decks, and Study pages ≤ **3 seconds** on a standard network (10 Mbps).
- User interactions (e.g., flipping cards, saving decks) complete within **2 seconds**.
- System supports **≥ 500 concurrent users** without server timeouts.
- AI Assist feature returns a response within **15 seconds** or displays a timeout notification.
- Database queries (e.g., search, clone deck) execute within **1 second** for 95% of requests.

2.2.2 Usability and Accessibility

Description:
The system shall provide an intuitive, consistent, and user-friendly interface that is accessible to all learners, including those using mobile devices or assistive technologies.

Rationale:
Flashcard-based learning depends heavily on user interaction and repetition. A confusing or inconsistent interface discourages users from continuous engagement.

Adhering to accessibility standards ensures inclusivity and broader adoption across devices and user groups.

Acceptance Criteria:

- The website interface complies with **WCAG 2.1 Level AA** accessibility guidelines.
- Users can complete core tasks (register, create deck, start study) without external instructions.
- All pages are **responsive** on screens ≥ 320px wide.
- Clear visual and textual feedback (loading spinners, confirmation alerts, error messages) is provided for all actions.
- Navigation and interactions are operable via both mouse and keyboard.

2.2.3 Reliability and Availability

Description:
The Flashcard Website must provide consistent service availability and preserve user data under normal and fault conditions.

Rationale:
As users depend on the platform for tracking their learning progress, any unexpected downtime or data loss could harm trust and learning continuity. Ensuring reliability and recoverability enhances system credibility and user satisfaction.

Acceptance Criteria:

- System uptime ≥ **99%**, excluding scheduled maintenance.
- User data (decks, cards, progress) automatically persisted to the database after creation or update.
- Automated **daily backups** and restoration procedures are maintained.
- In case of service failure, system recovery occurs within **10 minutes**.
- No data loss occurs after unplanned restarts or connectivity interruptions.

2.2.4 Security Requirements

Description:
The system shall protect user data and prevent unauthorized access, modification, or leakage through proper authentication, authorization, and encryption methods.

Rationale:
Learners may store personal information and study materials that should remain private. Data breaches or unauthorized edits can compromise user trust and violate ethical standards. Implementing strong security safeguards aligns with privacy regulations and good development practice.

Acceptance Criteria:

- All communications use **HTTPS (TLS 1.2 or higher)**.
- Passwords are hashed using **bcrypt** or **Argon2** before database storage.
- Input validation is implemented to prevent **SQL injection**, **XSS**, and **CSRF** vulnerabilities.
- Session tokens expire automatically after inactivity (e.g., 30 minutes).
- Role-based access control is enforced for administrator and user operations.
- Administrative actions (deck removal, user warnings) are recorded in activity logs.

2.2.5 Scalability Requirements

Description:
The system must be designed to handle increasing workloads and expanding user bases without requiring major architectural changes.

Rationale:
As adoption grows, the platform must scale seamlessly to accommodate new users, decks, and study activities. Scalable architecture ensures long-term sustainability and reduces operational costs by leveraging cloud services efficiently.

Acceptance Criteria:

- Supports **horizontal scaling** via cloud services (Vercel, Render, Supabase).
- Caching mechanisms and database indexing implemented for faster retrieval.
- Pagination applied to public deck browsing and search results.
- System handles at least a **5× increase** in dataset size (e.g., decks and users) with no critical failures.
- New features (e.g., advanced SRS, analytics) can be integrated without changing core architecture.

2.2.6 Maintainability and Extensibility

Description:
The system's architecture, source code, and documentation shall support easy maintenance, debugging, and future enhancements.

Rationale:
Given the project's educational nature and evolving scope, maintainability ensures that future teams or contributors can continue improving the system efficiently. Extensibility allows for new modules (e.g., mobile apps or gamification) without rewriting the existing codebase.

Acceptance Criteria:

- Source code adheres to modular structure with clear separation of frontend, backend, and database layers.
- Documentation (README, API reference, database schema) is up-to-date and stored in version control.
- Continuous Integration/Continuous Deployment (CI/CD) pipelines handle automated testing and deployment.
- Code follows consistent naming conventions and style guidelines.
- Unit and integration test coverage ≥ **60%** of codebase.
- Change logs and version histories maintained for all releases.

2.2.7 Portability

Description:
The web application shall function reliably across multiple operating systems, browsers, and screen sizes without requiring custom installation.

Rationale:
Portability increases accessibility for learners using diverse devices and ensures consistent user experience regardless of platform or environment. It also reduces dependency on specific hardware or software configurations.

Acceptance Criteria:

- Fully functional on major browsers: **Chrome, Firefox, Edge, Safari**.
- Compatible with operating systems: **Windows, macOS, Linux, Android, iOS**.
- Responsive UI adapts to various screen sizes (mobile, tablet, desktop).
- Deployment possible on standard web hosting services without proprietary dependencies.

2.2.8 Legal, Ethical, and Privacy Requirements

Description:
The platform must respect user privacy, intellectual property rights, and ethical use of AI technologies while ensuring compliance with legal and institutional policies.

Rationale:
Educational systems often handle sensitive personal and user-generated content. Compliance with privacy principles and ethical AI use is critical to maintain transparency, fairness, and academic integrity. Protecting intellectual property also prevents copyright violations in shared decks.

Acceptance Criteria:

- Privacy Notice and Terms of Service clearly explain data usage, retention, and user rights.
- No private decks or personal data are used for AI model training without explicit user consent.
- Users can export or delete their personal data on request.
- Public decks include attribution fields and allow creators to select a **Creative Commons license**.
- AI-generated suggestions include visible disclaimers stating “AI output may contain inaccuracies.”
- Reporting and moderation mechanisms available for inappropriate or copyrighted content.

3. Requirement modeling

3.1 Use case diagram

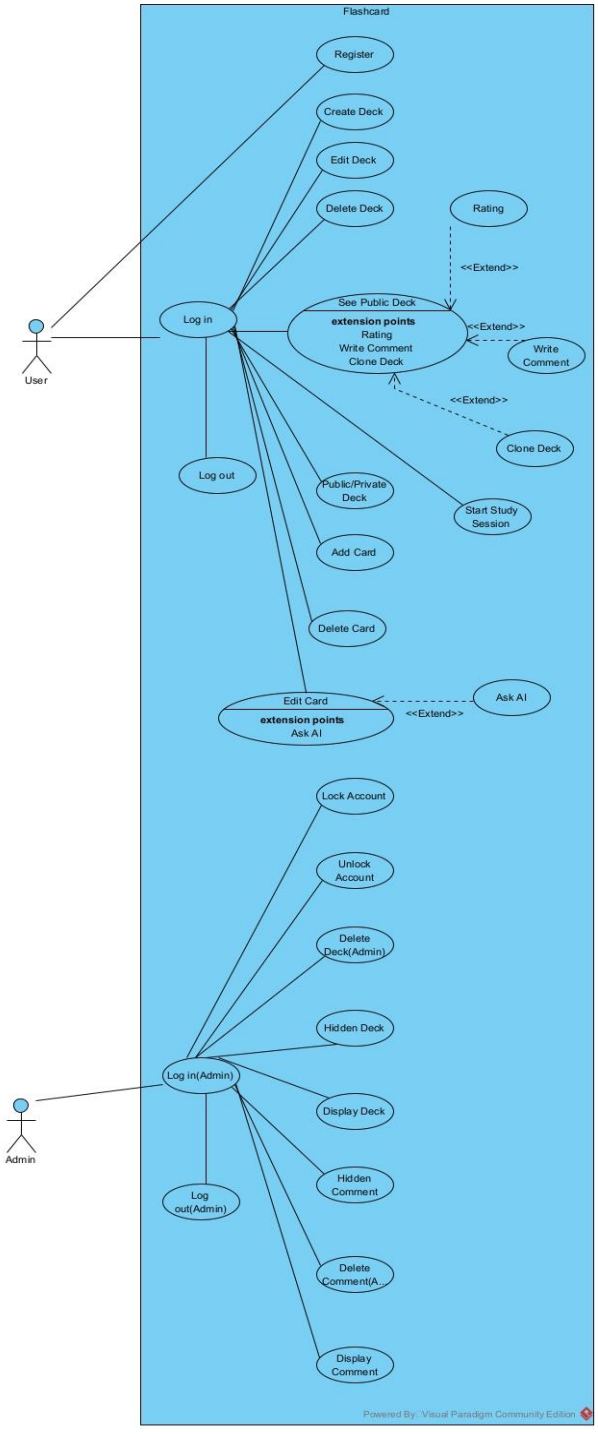

Figure 3.1 Use case diagram

3.2 Data flow diagram

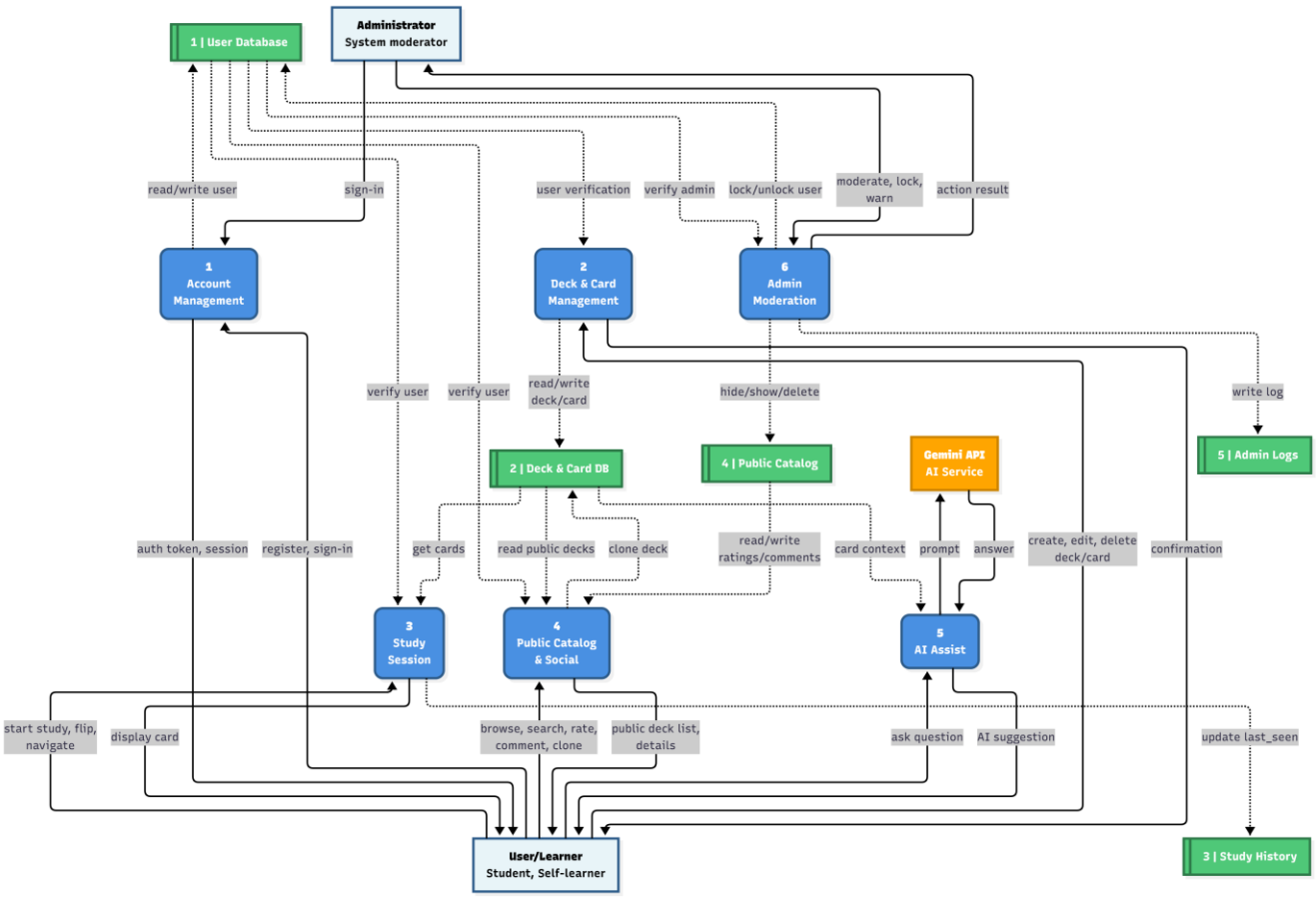

Figure 3.2: Data flow diagram

3.3 Entity relationship diagram

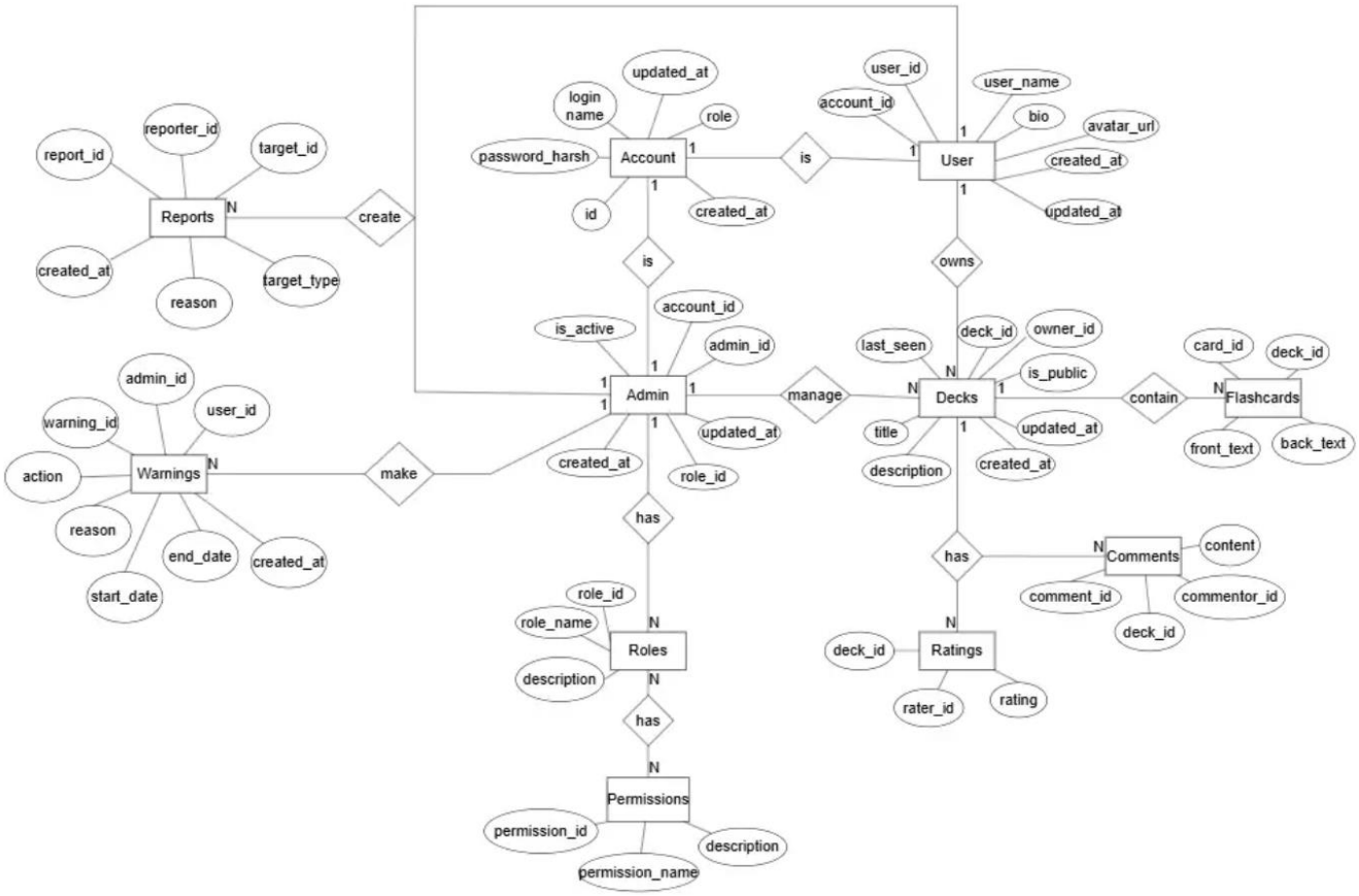

Figure 3.3: Entity relationship diagram

3.4 Finite state machine diagram

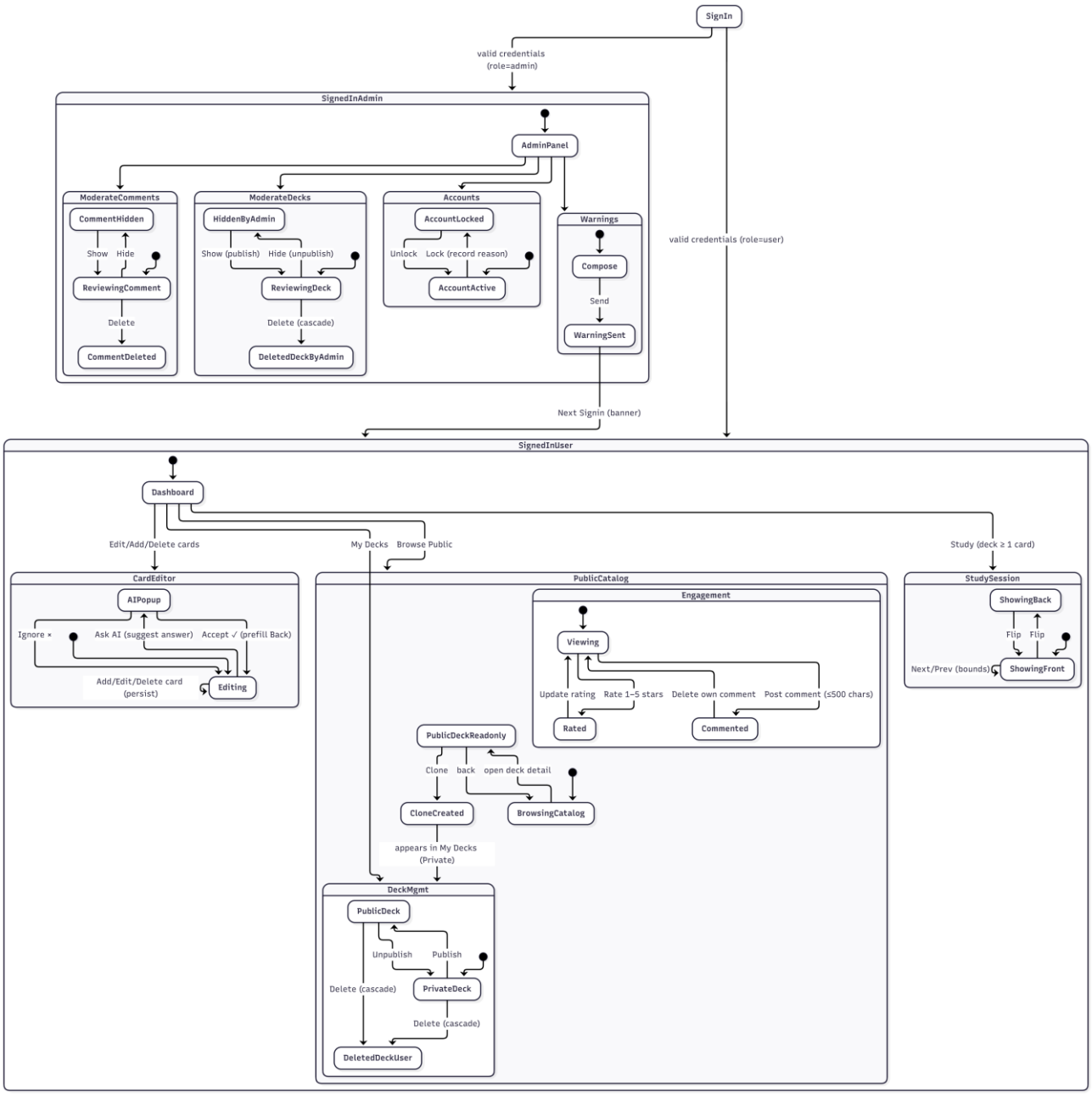

Figure 3.4: Finite state machine diagram