Software Design and Analysis (SE-2002)

Assignment 3

Software: AnkiDroid

Group#10

Sana Idrees(23i-2039), Manahil(23i-3000),

Maryam Fatima(23i-3007) and Ayesha Khan(23i-3037)

BS-Software Engineering(4B)

Submitted to: Dr. Behjat Zuhaira

Client: Group # 1

Use Cases Correction:

#1:

Use Case Name	Manage Decks

Scope	Ankidroid Enhanced Learning System
Level	User Goal
Primary Actor	Learner/Student
Stakeholders and Interests	- **Student:** Wants to efficiently manage decks for better learning organization. - **System Administrator:** Ensures deck management operations function smoothly. - **Institution (if applicable):** May track user engagement with decks if consent is given.
Preconditions	- The student must be logged in. - At least one deck must be available for management.
Success Guarantee (Postconditions)	- The system successfully updates the decks based on the performed action. - User receives confirmation of the modification.
Main Success Scenario	1. The user accesses the "Manage Decks" section. 2. The system displays a list of available decks. 3. The user selects a deck and an action (create, rename, delete, import/export, share). 4. If renaming, the user enters a new name, and the system updates it. 5. If deleting, the system requests confirmation before removing the deck. 6. If importing/exporting, the user selects a file, and the system processes it. 7. If sharing, the user enters recipient details, and the system sends the deck. 8. The system updates the deck list and provides confirmation to the user.
Extensions (Alternate Success/Failure Scenarios)	- **Invalid Deck Name:** The system prevents renaming to an empty or duplicate name. - **Deck Deletion Warning:** The system asks for confirmation before deletion. - **Import Failure:** If the file format is incorrect, the system notifies the user and rejects the import. - **Sharing Error:** If the recipient is invalid or sharing fails, the system alerts the user.
Special Requirements	- Deck operations should complete within **2**

	seconds for a smooth experience. - The system should **support multiple concurrent users** managing decks. - Data security must be maintained for shared decks.
Technology & Data Variations List	- Decks may be stored **locally or in the cloud** based on user preferences. - Import/export may support **Anki-specific formats**.
Frequency of Occurrence	- Learners may manage decks **daily or weekly** based on study needs.
Testing Considerations	- Ensure **rename, delete, import/export, and share** work as expected. - Test **large deck files** for performance stability.
Implementation Timing	- Deck management can be **iteratively enhanced** with cloud storage options and AI-powered recommendations in future updates.
Miscellaneous (Open Issues)	- Should the system allow **deck recovery after deletion**? - How to prevent **accidental sharing of sensitive study materials**?

#2

Use Case Name	Manage Cards

Scope	Ankidroid Enhanced Learning System
Level	User Goal
Primary Actor	Learner/Student
Stakeholders and Interests	- **Student:** Wants to efficiently manage decks for better learning organization. - **System Administrator:** Ensures deck management operations function smoothly. - **Institution (if applicable):** May track user engagement with decks if consent is given.
Preconditions	- The student must be logged in. - At least one deck must be available to manage cards within it.
Success Guarantee (Postconditions)	- The system successfully updates the cards based on the performed action. - User receives confirmation of the modification.
Main Success Scenario	1. The user accesses the "**Manage Cards**" section.

	2. The system displays a list of available cards in the deck. 3. The user selects a card and an action (add, rename, delete, edit, or reorganize). 4. If adding, the user enters the content on the front and back of the card. 5. If renaming, the user enters a new name, and the system updates it. 6. If deleting, the system requests confirmation before removing the card from the deck. 7. If editing, the user selects the card, modifies its content, and confirms the update 8. If reorganizing, the user adjusts the order or grouping of cards, then changes are applied. 9. The system updates the card list and provides confirmation to the user.
Extensions (Alternate Success/Failure Scenarios)	- **Empty Fields:** The system prevents renaming to an empty or duplicate name. -**Duplicate Card:** The system may warn if the card with identical content already exists - **Card Deletion Warning:** The system asks for confirmation before deletion. - **Save Error:** If there is storage or sync error related issues the system alerts the user and do not save the changes.
Special Requirements	- Card actions should complete within **2 seconds** for a smooth experience. - The system should support good quality text,images,audios and other attachment formats - Undo functionality for accidental edit or deletion cases -Offline support with automatic sync when online
Technology & Data Variations List	- Cards may contain only text, or include media like images or audio clips or other attachment formats - Cards may be stored locally or synced to cloud services(e.g. Anki Web) - Different card types (basic,cloze, reversed e.t.c.) may affect editing behaviour.
Frequency of Occurrence	- Learners may manage cards frequently- **daily or weekly-** based on study needs.
Testing Considerations	- Ensure **rename, delete,save** work as expected. -Validate form in input (e.g empty field or some sort of special characters). -Test media attachment support (images/audio upload) -Ensure proper sync of cards across different

	devices. - Test card management under **large deck sizes.**
Implementation Timing	- Card management can be **iteratively enhanced** with cloud storage options -Basic card management should be implemented early -Advanced features like media support, bulk editing, and AI-powered recommendations can be added in later implementations
Miscellaneous (Open Issues)	- Should the system allow **card recovery after deletion**? - How to prevent **accidental edits during review sessions**? -Should the app suggest edits based on performance analytics?

#3:

Use Case Name	**Review Flashcards**

Scope	AnkiDroid Enhanced Learning System
Level	User Goal
Primary Actor	Learner/Student
Stakeholders and Interests	- **Student:** Wants an efficient review system that optimizes learning. Also wants an engaging and rewarding way to become consistent with their reviewing routine - **System Administrator:** Ensures the review system functions correctly and spaced repetition scheduling works. Also ensures proper functioning of gamified features - **Institution (if applicable):** May analyze student engagement trends (if permitted).
Preconditions	- The user must have at least one deck with scheduled flashcards available. - The spaced repetition system must be active.
Success Guarantee (Postconditions)	- The learner completes a review session with flashcards scheduled based on their difficulty ratings. - The system updates the user's progress statistics and adjusts future review schedules. -The learner completes a review session that meets or exceeds the daily challenge goal. -The system updates review progress and challenge streaks, awards points, and adjusts the spaced repetition algorithm accordingly
Main Success Scenario	1.The student is notified by the system about a

	daily review challenge (for example, “Review 20 cards to maintain your streak”) 2. The student selects a deck or multiple decks for review. 3. The system presents a flashcard based on the spaced repetition schedule. 4. The student recalls the answer and flips the card to reveal the correct response. 5. The student rates their recall (**again, hard, or good**). 6. The system updates the review schedule for the card based on the selected rating and tracks challenge progress, for example number of reviewed cards daily. 7. Steps 3-6 repeat until all scheduled cards are reviewed until one of the following occurs: ● The daily challenge goal is completed. ● All due flashcards has been reviewed ● The student choose to exit session 8. The system updates the student’s progress and displays statistics (e.g., number of reviewed cards, time spent). 9. The System updates the streaks, points, badges, or rewards ,And adjust future review schedules based on rating
Extensions (Alternate Success/Failure Scenarios)	- **No Cards Available:** The system notifies the student that no flashcards are due for review. - **Early Exit:** The student can exit the session at any time, and progress will be saved. - **Review Customization:** The student may change the display settings, such as enabling auto-flip or dark mode. - **Missed Sessions:** Overdue cards are prioritized in the next session - **Adjust Review Order/Intervals:** User can override the default order or spacing - **Error Handling:** If a flashcard fails to load, the system displays an error message and allows the user to skip it. -**Challenge Customization:** User can adjust challenge difficulty (e.g., increase goal from 20 to 30 cards). -**Missed Challenges:** Notify users of streak loss and offer a recovery challenge.
Special Requirements	- The system must ensure that flashcards are presented **smoothly and efficiently**. - The spaced repetition algorithm must adjust **review frequency** accurately based on recall ratings.

	- The system should display **progress statistics** at the end of each session. -**Real-time challenge tracking** UI, e.g., progress bar, streak indicator. -The reward system like badges, streaks must integrate with the flashcard review engine.
Technology & Data Variations List	- Spaced repetition scheduling may use the **SuperMemo 2 (SM-2) algorithm** or a similar system. - The system may support different flashcard formats, including **text, images, and audio**.
Frequency of Occurrence	- Students may review flashcards **multiple times per day** based on their learning schedule.
Testing Considerations	- Validate **spaced repetition accuracy** to ensure cards are scheduled correctly. - Ensure the **progress statistics update in real-time**. - Test performance with **large decks** to ensure stability. -Ensure streaks are rewarded correctly.
Implementation Timing	- The basic spaced repetition review system should be implemented first, followed by a daily challenges notification system, then streak tracking, which is furthur followed by additional features like **AI-powered review recommendations**.
Miscellaneous (Open Issues)	- Should the system allow users to **customize their review schedule** (e.g., override the algorithm)? - Should the system provide **hints or explanations** for incorrect answers? - How should the system handle **missed review sessions** (e.g., rescheduling overdue cards)? -Should user be allowed to set their own daily challenge goals

#4:

Use Case Name	**Track Progress**

Scope	AnkiDroid Enhanced Learning System
Level	User Goal

Primary Actor	Registered User
Stakeholders and Interests	- **User:** Wants to track study performance, maintain streaks, and improve retention. - **System Administrator:** Ensures accurate data tracking and visualization. -**Group Members**: Interested in viewing team progress and comparisons within a shared leaderboard environment -**Group Admin :**Controls the visibility in groups or handling statistics. - **Educational Institutions (if applicable):** May provide insights on user learning trends.
Preconditions	- The user must have completed at least one study session. - The system must have recorded study data for progress tracking. -The user must be part of a group to view group-related statistics.
Success Guarantee (Postconditions)	- The user successfully views their study statistics, including reviewed flashcards, time spent, and retention rates. - The system generates accurate progress graphs based on study history. -User motivation is enhanced through meaningful data and insights
Main Success Scenario	1. The user navigates to the **Progress Tracking** section. 2. The system **retrieves stored study data** including flashcards reviewed, time spent, retention rates, and study history. 3. The system **displays key statistic**s, such as flashcards reviewed, time spent, and retention rates. 4. The system **generates graphical representations of study trends.** 5. The user **analyzes their progress** using the provided data. 6. The system updates statistics dynamically as the user continues studying. 7.**System calculates and displays current streak status,**showing consecutive study days,missed days,option for recovery if available. 8. System **check if user is part of a study group**. 9.If part of a group ,the system retrieves group study statistics including average study time,top performers in group and group streaks. 10.System displays group stats using leaderboard,bar charts or comparisons etc.

Extensions (Alternate Success/Failure Scenarios)	- **No Study Data Available:** If the user has no recorded study sessions, the system displays a message encouraging them to start studying. - **Partial Data Retrieval:** If some data is missing, the system provides available statistics and notifies the user of any missing information. - **Incorrect Statistics:** If the user suspects an error, they can request a data refresh or contact support. -**Streak Broken:** System highlights missed days and shows a motivation message or recovery tips -**Group not joined:** If the user is not in any group, the system displays “You have not joined any group. Join one to see view Group Study Statistics”.
Special Requirements	- The system must ensure **data accuracy and real-time updates** for progress tracking. - The UI should be **visually engaging**, using charts and graphs for better understanding. - Data should be stored **securely and efficiently** to prevent loss. -Offline support for personal progress using cached data.
Technology & Data Variations List	- The system may use **SQLite** or **cloud-based storage** to log progress data. - Graphs may be generated using **built-in UI libraries** or external visualization tools. -**Group data retrieva**l through group Id token
Frequency of Occurrence	- Users may check their progress **daily or weekly** based on their study habits.
Testing Considerations	- Verify that **progress calculations are accurate**. - Ensure that **graphs update correctly** after new study sessions. - Test **performance with large datasets** to maintain a smooth user experience.
Implementation Timing	- Initial implementation should focus on **basic statistics**, followed by **advanced analytics and trends**. **Phase 1**: Personal progress and streak data with basic visuals. **Phase 2**: Group statistics, leaderboard integration, and streak warnings. **Phase 3**: Export options and smart insights (e.g., “Best Study Day”).

Miscellaneous (Open Issues)	- Should users be able to **customize statistics** (e.g., choose what data to display)? - How should the system handle **long study breaks** (e.g., resetting trends vs. maintaining historical data)? - Should users receive **progress notifications** to encourage continued studying? -Should **user be able to toggle between individual and group performance** view? -Should **past streak data** be hidden or visible -Should **group admins moderate visibility** of individual group member's stats?

#5:

Use Case Name	**Manage FlashCard Battles**
Scope	**AnkiDriod Enhanced Learning System**
Level	**User Goal**
Primary Actor	**Student/Learner**
Secondary Actor(s)	1. **Battle Opponent** (Student/Group Member) – Another user who accepts the battle invitation. 2. **AI Study Buddy** (Optional) – If no human opponent is available, the AI acts as a competitor or generates adaptive questions.
Stakeholders & Interests	1. **Students** – Want an engaging, competitive way to test knowledge retention. 2. **Group Members** – Need smooth real-time interaction, fair scoring, and balanced difficulty. 3. **System Admin** – Ensures battle stability, prevents cheating, and maintains server performance.

Preconditions	1. User must be **logged in.** 2. At least **one flashcard deck must be available** (either personal or shared). 3. Stable **internet connection** (for real-time sync). 4. At least **one opponent** (human/ AI) must be available. 5. **Battle settings** like time limit, scoring rules must be pre-configured.
Success Guarantee (Postconditions)	1. Battle is **successfully initiated** and completed. 2. Both **players receive questions** in real-time with synchronized timing. 3. **Scores are calculated fairly** both in accuracy + speed. 4. **Winner is declared based on predefined rules** e.g., first to 5 points. 5. **Battle statistics** like win/loss, accuracy, response time are **recorded**. 6. Ranking ,if enabled, are updated post-battle.

Main Success Scenario	1. User clicks **"Start Battle"** from the dashboard. 2. **System prompts to select a flashcard deck** (personal/group/shared). 3. User chooses opponent type: - **Friend** (from contacts) - **Random Player** (matchmaking) - **AI Study Buddy** (difficulty settings: Easy/Medium/Hard) 4. Opponent receives **invitation** and accepts. 5. **System loads the deck and starts battle in real-time**: - Questions **alternate** between players. - Each player has **10-15 seconds** to answer. 6. System validates answers and assigns points: - **Correct answer or fast response = Higher score.** **- Incorrect/slow answer = No points**. 7. Battle **continues for fixed rounds** (e.g., 5) or until a player reaches a set score. 8. System **declares winner** and updates: - Leaderboard. - User stats win rate, accuracy. 9. **Post-battle** options: - Review mistakes (incorrect answers highlighted). - Rematch (same opponent). - Return to menu.
Extensions/(Alternative Scenarios	4a. **Opponent Declines Invitation** - System notifies initiator and suggests other opponents. 4b. No Human Opponent Available - System defaults to AI Study Buddy. 5a. Player Disconnects Mid-Battle - If reconnection within 30 sec, resumes. - Else, battle ends (disconnected player forfeits). 6a. Player Doesn't Answer in Time - System marks as incorrect, moves to next question. 7a. Tie at Final Round - Sudden-death round: First correct answer wins. 8a. AI Study Buddy Post-Battle Feedback - Provides weakest topics based on incorrect answers.

	- Suggests related decks for improvement.
Special Requirements	1. Real-Time Performance – Max latency of 500ms for smooth gameplay. 2. Anti-Cheating – Disable copy-paste, detect bot-like behavior. 3. Fair Scoring – Adjust for network delays, prevent point farming. 4. Data Privacy – Battle history stored securely (GDPR compliance).
Technology & Data Variations	1. Real-Time Sync – WebSockets or Firebase for live updates. 2. AI Opponent – GPT-based question generation (adjusts difficulty). 3. Battle Customization – Allow users to set: - Time per question (5s/10s/15s). - Win condition (points/rounds). - Deck restrictions (e.g., only biology cards).
Frequency of Occurrence	High during exam prep periods; moderate otherwise. - Requires scalable matchmaking to handle peak loads.
Testing Considerations	1. Scoring Logic – Verify speed + accuracy weighting. 2. Network Resilience – Test under poor connectivity. 3. AI Behavior – Ensure questions match selected difficulty. 4. Load Testing – Simulate 1000+ concurrent battles.
Implementation Phases	Phase 1 (MVP) – Single-player vs. AI, basic scoring. Phase 2 – Multiplayer (invite-only), leaderboards. Phase 3 – Public matchmaking, advanced stats.
Miscellaneous (Open Issues)	1. Should battles be ranked (affecting ELO) or casual? 2. Penalty for rage-quitting? (e.g., temporary ban).

	3. Allow spectators in group battles?

Part A: Object-Oriented Design

Domain Model:

1. **Identifying Conceptual Classes by Category Lists:**

Category	**Classes**
Places	Deck Section, Progress Section, Study Group
Occurrence/Event	Review Session, Flashcard Battle, Shared Study Session, Daily Challenge
Structure	Deck, Study Session, Flashcard Review Flow
Thing	Flashcard, Deck, Progress, Question, Score, Statistic, challenge
Roles	Student/Learner, Group Study Member

2. **Identifying Conceptual Classes by Noun Phrases:**

A **Student** begins their daily learning by opening a **Deck**. The **Student** selects a **Flashcard** from the **Deck**. The **Student** then starts a **Flashcard Review Session**, where they go through a series of **Flashcards** one by one. For each card, they flip it and rate their recall, which helps decide when the card should appear again. After finishing the session, the **Review History is updated**, keeping track of what was reviewed. This information is used to generate the **Statistics**, which show how the **Student** has been performing over time.

To maintain consistent learning, the **Student** participates in a **Daily Challenge**. For fun and competition, the **Student** joins a **Flashcard Battle** with other **Students** or friends. This battle uses a specific **Deck** and tracks how well each player performs. The **Student** may also join a **Shared Study Session** with friends, where everyone studies the same **Deck** together in real time. Throughout these activities, a **Progress Tracker** monitors the **Student**'s performance, using their **Statistics** and **Review History**.

At times, based on their performance trends, **Recommended Cards** are suggested. These are specific **Flashcards** chosen because they relate to the **Student**'s **Weak Areas** or haven't been reviewed in a while.

Valid Conceptual Classes:

1. Core Conceptual Classes (Key Domain Entities)

Student
Deck
Flashcard
Flashcard Review Session
Consistency Assessment
Flashcard Battle
Shared Study Session
ProgressTracker

2. Derived Conceptual Classes

Statistics

Review History

Weak Areas

Recommended Card

Leaderboard

3. Datatype classes

Name

Date

Time

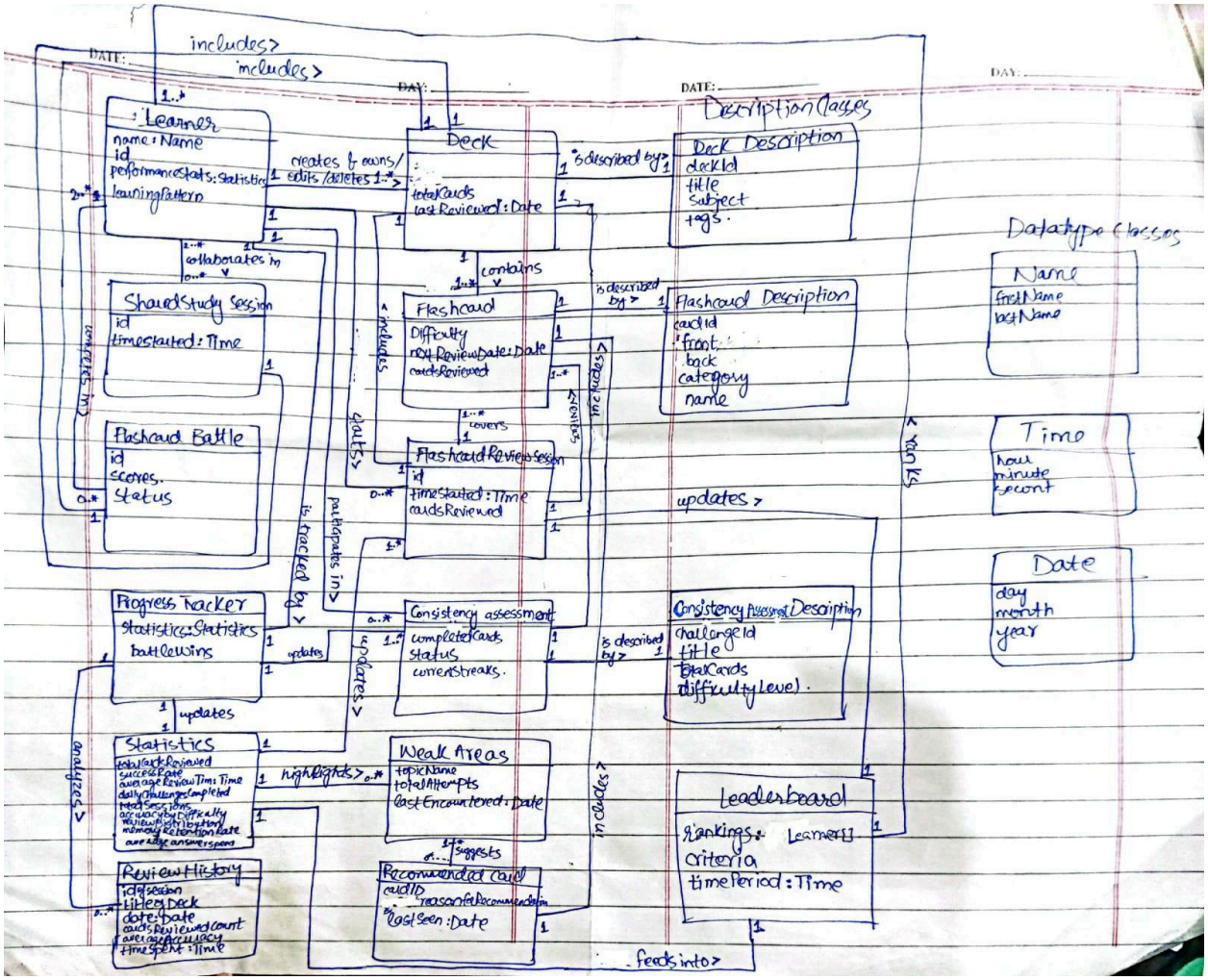
4. Description classes

Deck description

Flashcard description

Challenge description

Domain Model



Operation Contracts

Contract CO1:

- **Operation:** createFlashcard(flashcardName: String)
- **Reference:**
  - Use Case: Manage Flashcards
  - SSD: Manage Flashcards

- **Preconditions:**

  - A Deck with deckID exists

    - The Deck is owned by the current Student.

- **Postconditions:**

    - A Flashcard instance *fc* was created (instance creation).
    - Attributes of *fc* were initialized (attribute modification).
    - *fc.title* became *`flashcardName`* (attribute modification).
    - *fc* was associated with the Deck (association formed).
    - The Deck's **`totalCards`** increased by 1 (attribute modification).

Contract CO2:

- **Operation:** `createDeck(deckName: String)`
- **Reference:**
    - Use Case: Manage Decks
    - SSD: Manage Decks

- **Preconditions:**

    - The Student is logged in.
    - No other Deck owned by the Student has the same *`deckName`*
- **Postconditions:**

    - A Deck instance *d* was created (instance creation).
    - *d.title* became *`deckName`* (attribute modification).
    - *d* was associated with the Student (association formed).
    - *d* was added to the Student's deck list (association formed).

Contract CO3:

- **Operation:** `selectDeck(deckID: int)`
- **Reference:**
    - Use Case: Review Flashcards
    - SSD: Review Flashcards
- **Preconditions:**
    - A Deck with `deckID` exists.
    - The Deck contains at least one Flashcard.
    - The Student is logged in and authorized to access the Deck.
    - A `Flashcard Review Session` is in progress

- **Postconditions:**

    - **ReviewSession** instance was created (instance creation).
    - The ReviewSession was associated with the Deck (association formed).
    - The ReviewSession was associated with the Student (association formed).
    - The Deck's *lastReviewed* attribute was set to the current date/time (attribute modification).

Contract CO4:

- **Operation:** `calculateStreak(studentID: int)`
- **Reference:**
    - Use Case: Track Progress
    - SSD: Track Progress

- **Preconditions:**

    - A Student with the specified *studentID* exists.
    - The Student has completed at least one DailyChallenge.
- **Postconditions:**

    - The Student's *currentStreak* was updated (attribute modification).
    - If the streak continued, the Student was associated with the Leaderboard (association formed).
    - If the streak was broken, a Notification instance was created (instance creation).
    - The Notification was associated with the Student (association formed).

Contract CO5:

- **Operation:** `endBattle(battleID: String)`
- **Reference:**
    - Use Case: Manage Flashcard Battles
    - SSD: Manage Flashcard Battles

- **Preconditions:**

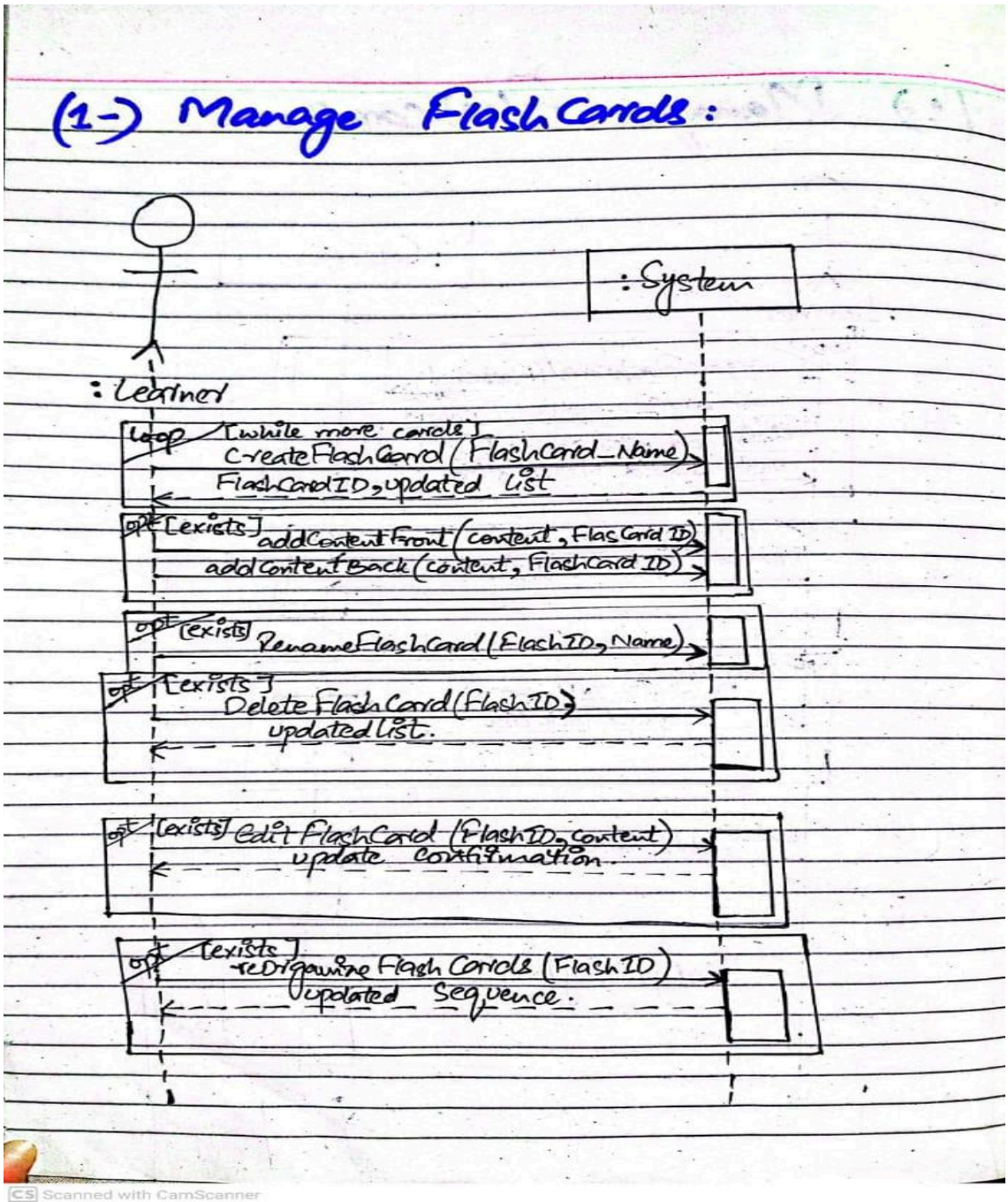
    - A FlashcardBattle with *battleID* exists.
    - All battle rounds are completed.
- **Postconditions:**

    - The FlashcardBattle's *winner* attribute was set (attribute modification).

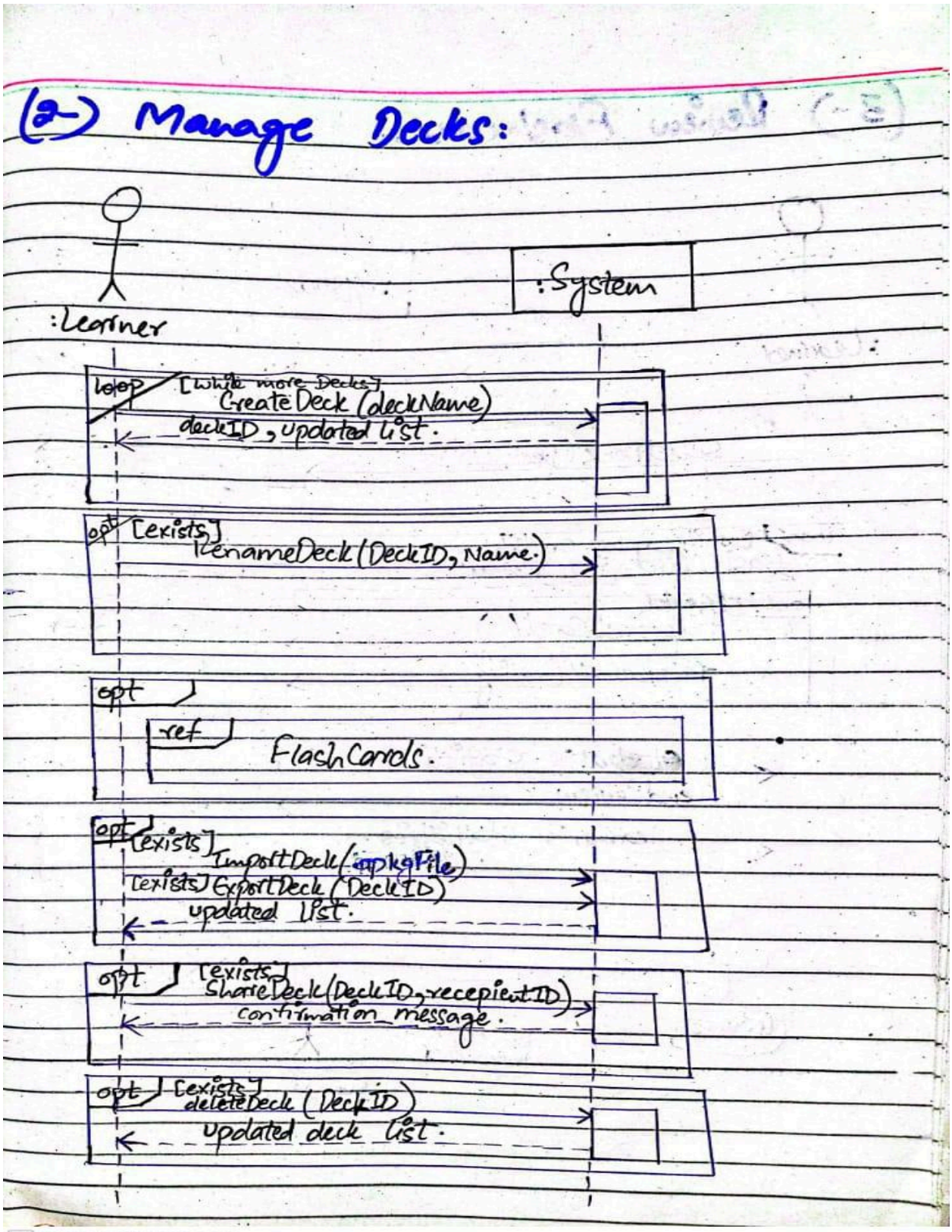
- The winning Student's `battleWins` attribute was incremented (attribute modification).
- The Leaderboard's rankings were updated (attribute modification).

System Sequence Diagrams:

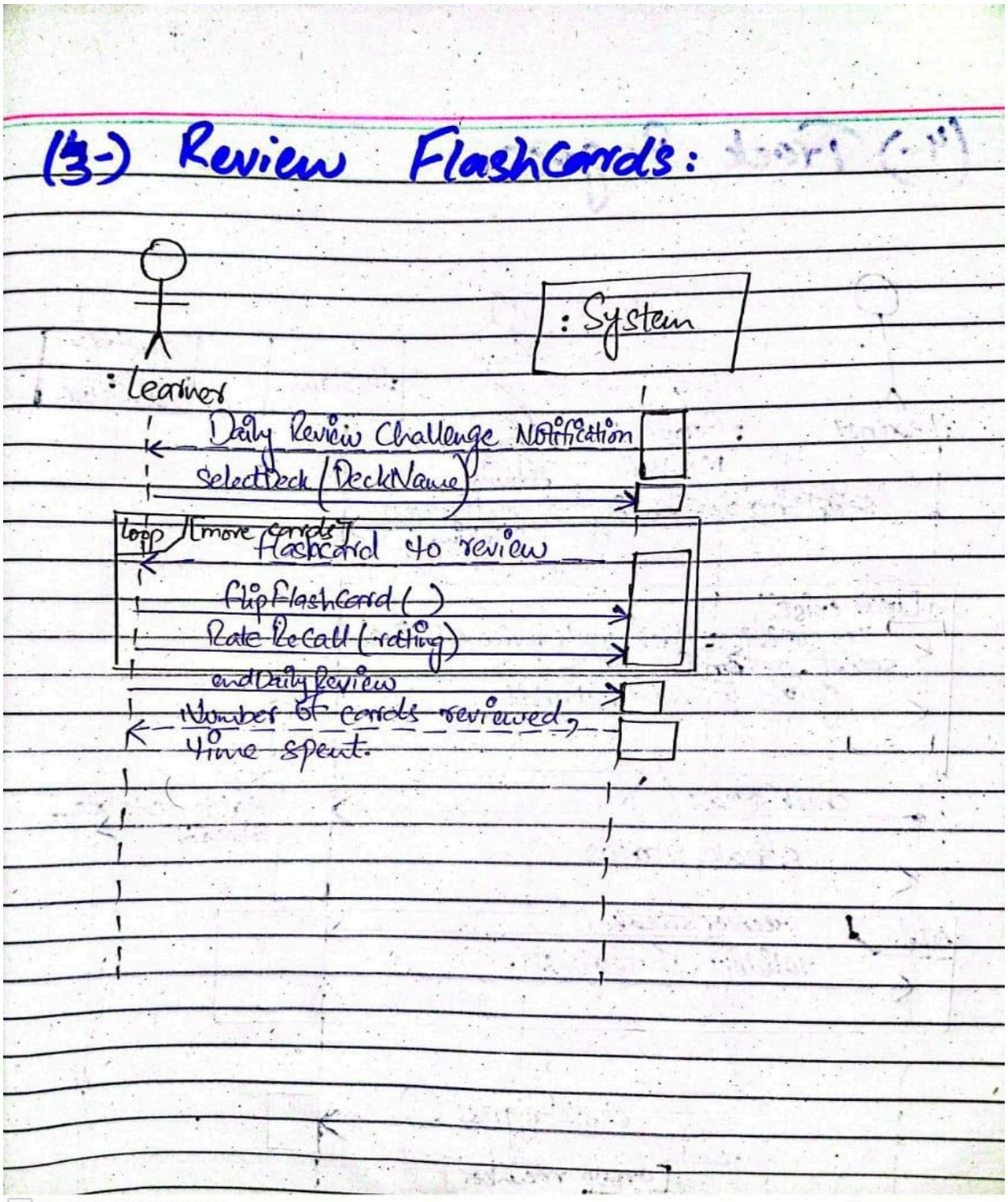
#1:Manage Decks:



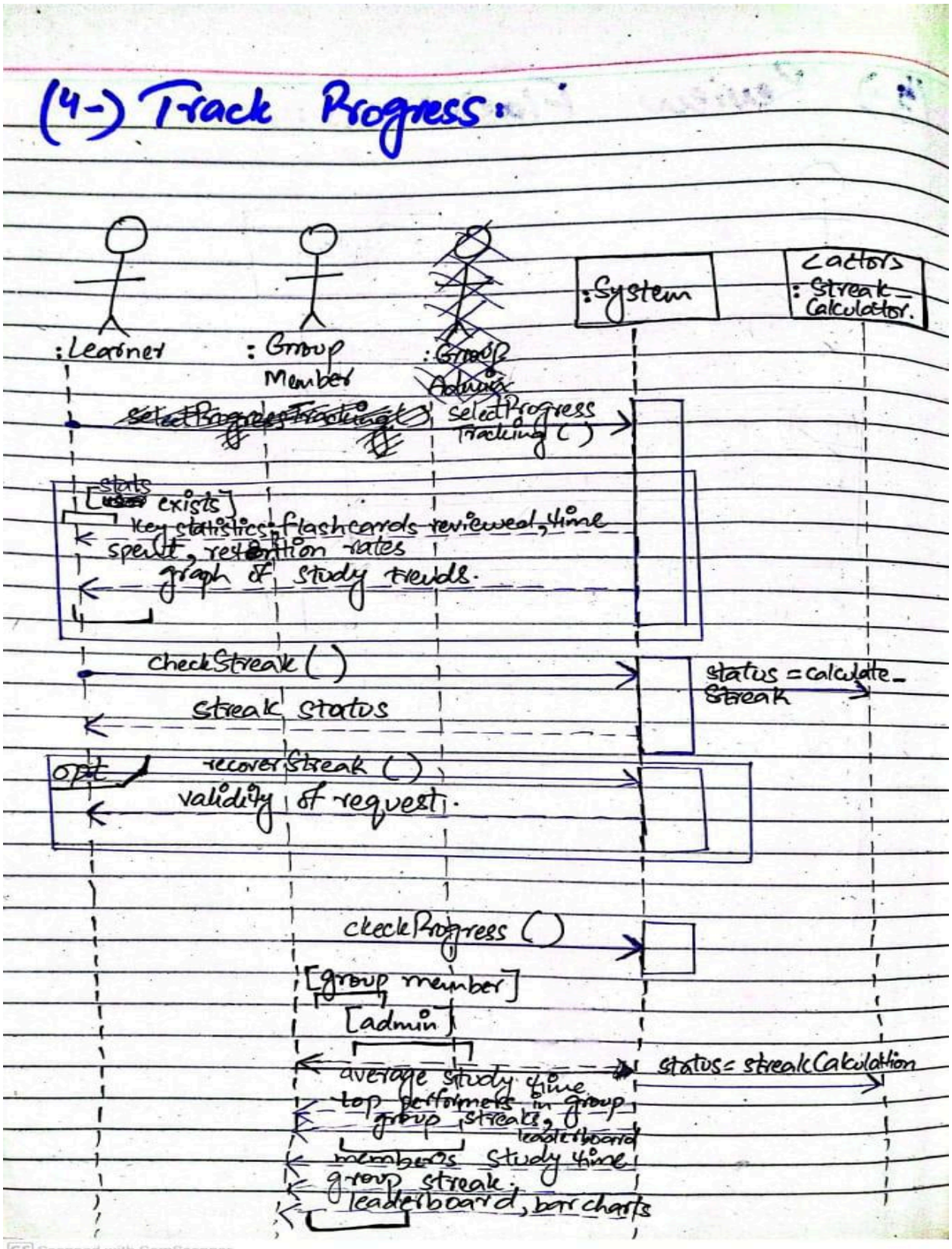
#2:Manage Cards:



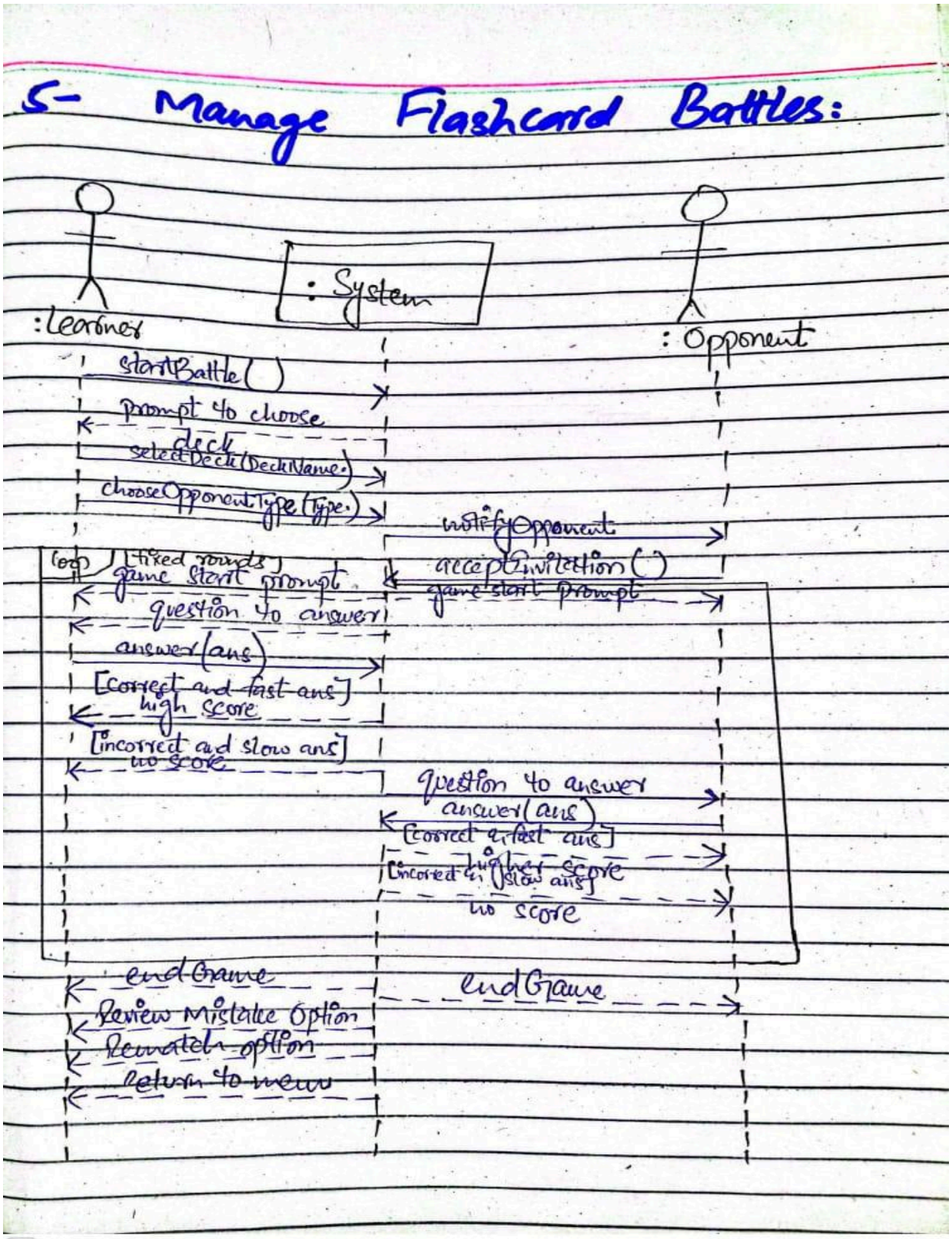
#3:Review FlashCards:



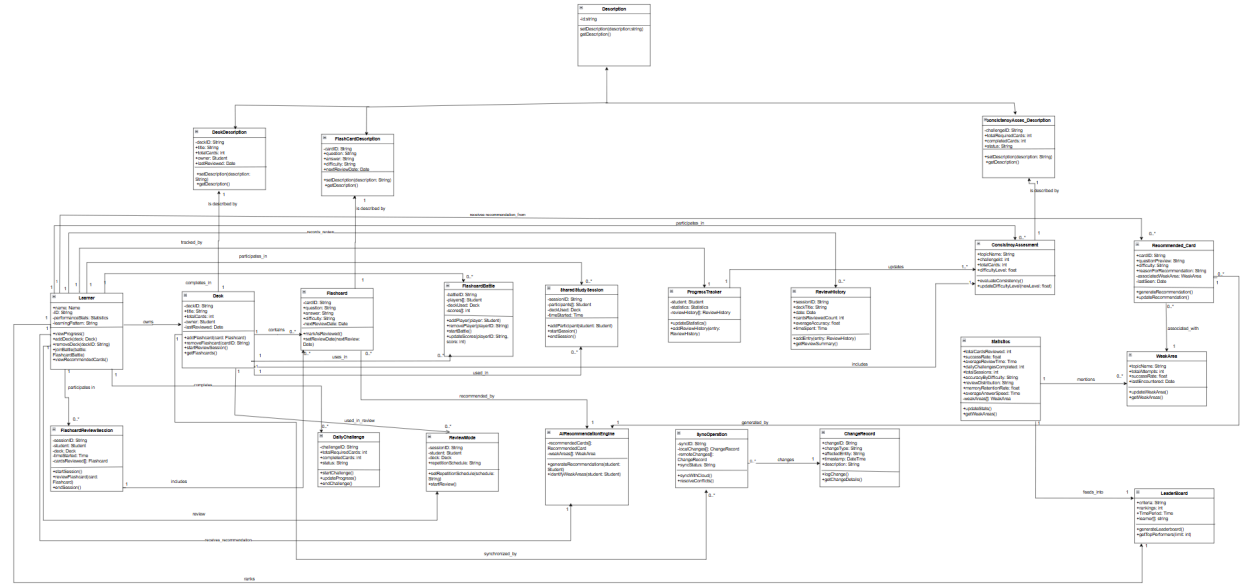
#4:Track Progress:



#5:Manage FlashCard Battles:



Class Diagram:



DFD CORRECTION: