CS3354 Software Engineering
Final Project Deliverable 1

Glossa

Samuel Jacobs
Matthew Cundiff
Hamza Daruger
Aryan Patel
Ashton LaRoche
Kelly Holl
Brandon Buchanan

1. [5 POINTS] Please attach here the Final Project draft description (that contains the instructor feedback). It is ok to include a picture of the original document. Address the feedback provided for your proposal by listing what you did / plan to do to comply with those proposed changes and or requests for additions to your project

Project Proposal

Project Title: Glossa

Team Members:

- Aryan Patel
- Ashton Laroche
- Samuel Jacobs
- Kelly Holl
- Matthew Cundiff
- Hamza Daruger
- Brandon Buchanan

Project Description:

- Language learning app focused on flashcards

Motivation:

We chose this project because we like learning other languages. The current language learning options are inefficient. There is a desire in the group to travel and communicate with other people.

Feedback to Learner

9/20/22 4:04 PM

Good choice for a topic! Using technology to teach yourself a new language sounds very cool.

Your team should consider a detailed break down of the tasks as described in the project specs. There is no need to resubmit this proposal, though. Just make sure that you don't miss any task.
In the final report, please make sure to include comparison with similar applications -if any-, make sure that you differentiate your design from those, and explicitly specify how.
No delegation of tasks provided.
Please share this feedback with your group members.
You are good to go. Have fun with the project and hope everyone enjoys the collaboration.

To address the feedback, we created a breakdown of who would complete each task for both of the project deliverables. The breakdown is below:

Task Delegation:

Project Deliverable 1

GitHub (must be three different people)

Task	Group Member
Create repository and add members	Samuel
Make first commit	Hamza
Make project scope commit	Kelly

Report

Task	Group Member
Task delegation	Everyone
Address feedback	Ashton
Software process model selection	Aryan
Functional requirements	Matthew
Non-functional requirements	Aryan
Use case diagram	Hamza
Sequence diagram	Kelly
Class diagram	Brandon
Architectural design	Samuel

Project Deliverable 2

Task	Group Member
Task delegation	Everyone
Project scheduling	Kelly
Cost/effort/pricing estimation	Aryan
Test plan	Matthew
Comparison with similar designs	Ashton
Conclusion	Brandon
Presentation slides	Everyone

2. [10 POINTS] Setting up a Github repository.

Link: https://github.com/SjacobsUTD/3354-Glossa

3. [5 POINTS] Delegation of tasks: Who is doing what. If no contribution, please specify as it will help us grade each group member fairly.

Task Delegation:

Project Deliverable 1

GitHub (must be three different people)

Task	Group Member
Create repository and add members	Samuel
Make first commit	Hamza
Make project scope commit	Kelly

Report

Task	Group Member
Task delegation	Everyone
Address feedback	Ashton
Software process model selection	Aryan
Functional requirements	Matthew
Non-functional requirements	Aryan
Use case diagram	Hamza
Sequence diagram	Kelly
Class diagram	Brandon
Architectural design	Samuel

Project Deliverable 2

Task	Group Member
Task delegation	Everyone
Project scheduling	Kelly
Cost/effort/pricing estimation	Aryan
Test plan	Matthew
Comparison with similar designs	Ashton
Conclusion	Brandon
Presentation slides	Everyone

While each task was completed individually, the group provided feedback for each person's tasks. We decided to do this since the majority of the tasks are sequential and require the completion of previous tasks.

4. [5 POINTS] Which software process model is employed in the project and why. (Ch 2)

Spiral model:

- We may want to modify our requirements, and the spiral model will allow us to make changes with each circuit
- We want to build the application in increments so that we have the fundamental features working before adding details
- We want to reduce risk throughout the implementation phase

5. [15 POINTS] Software Requirements including

5.a.) [5 POINTS] Functional requirements. To simplify your design, please keep your functional requirements in the range minimum 5 (five) to maximum 7 (seven). (Ch 4)

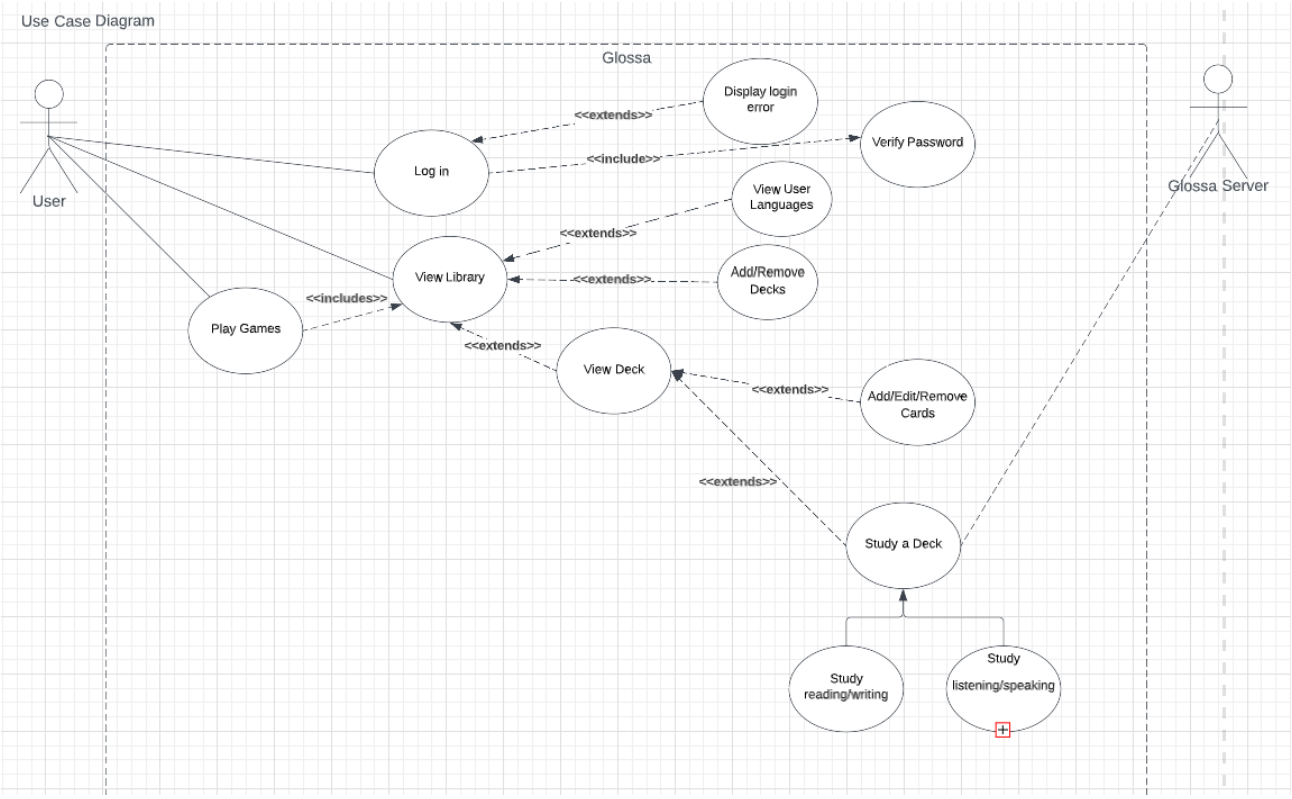
Functional:

1.The system shall randomize the order of the flashcards
2. A user shall be able to use their flashcards to play matching games
3. The system shall save user progress after the user studies a card
4. A user shall be able to add and remove flashcards from a deck
5. A user shall be able to add and remove decks from a library
6. A user shall be able to view a deck of flashcards
7. A user shall be able to tag decks with languages

5.b.) [10 POINTS] Non-functional requirements (use all non-functional requirement types listed in Figure 4.3 - Ch 4. This means provide one nonfunctional requirement for each of the leaves of Figure 4.3. You can certainly make assumptions, even make up government/country based rules, requirements 4 to be able to provide one for each. Please explicitly specify if you are considering such assumptions.)

Performance: The system shall randomize card sets in 0.2 seconds.
Space: The application shall take up no more than 1 GB of device storage.
Usability: A user shall be able to navigate to a flashcard deck within 5 seconds of opening the app.
Dependability: The system shall be available 24/7. Downtime within a given day shall not be more than 1 minute.
Security: Users shall authenticate themselves using a username and password.
Environmental: The app shall be able to work in iOS and Android operating systems.
Operational: The screen refresh time shall not exceed 2 seconds.
Development: The system shall remain functional during update deployments.
Regulatory: The app shall fall within FCC requirements and Android/Apple App Store requirements.
Ethical: User data shall not be sold.
Accounting: Payment plan shall complete in 2 minutes.
Safety/Security: Passwords shall be encrypted so that user accounts are secure.

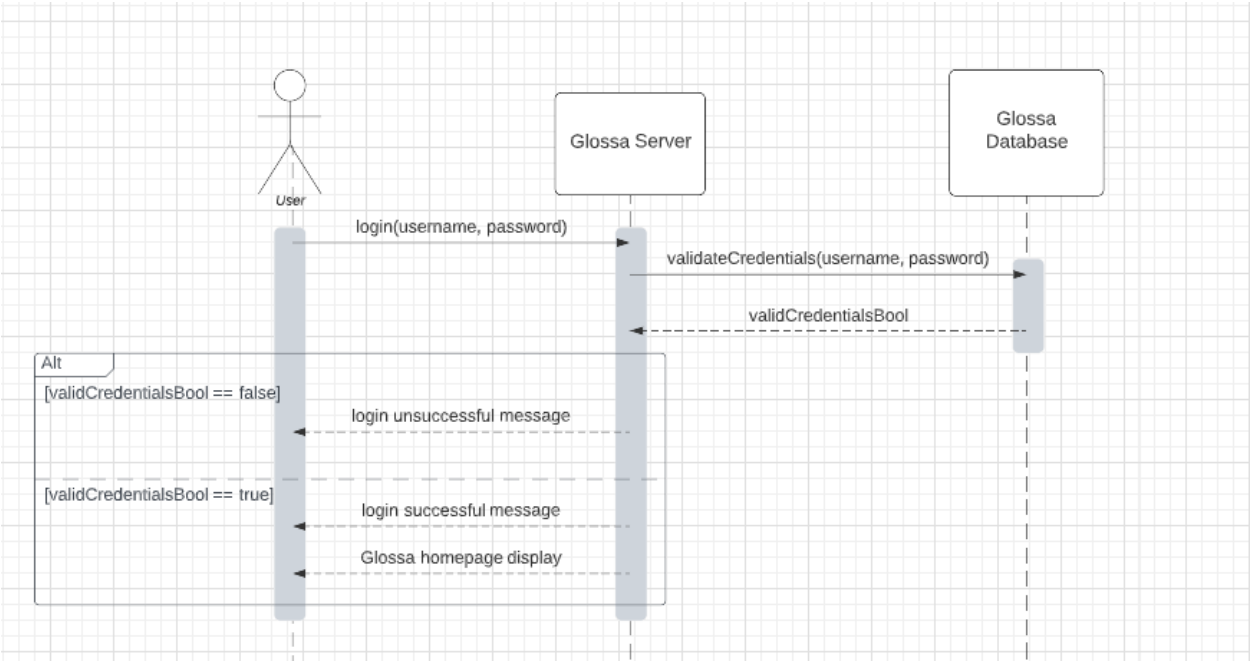
6. [15 POINTS] Use case diagram – Provide a use case diagram (similar to Figure 5.5) for your project. Please note than there can be more than one use case diagrams as your project might be very comprehensive. (Ch 5 and Ch 7)



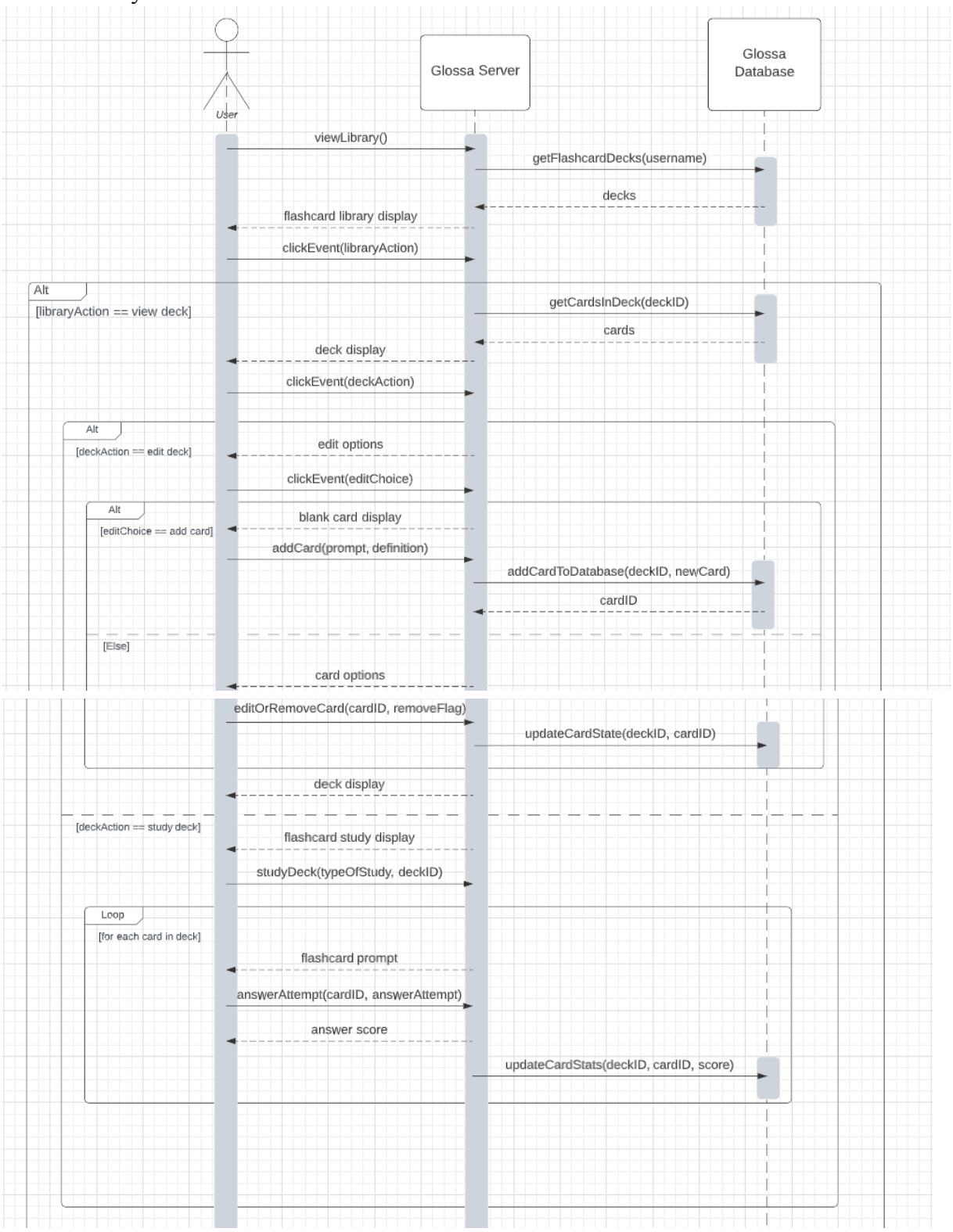
After logging in, the user is presented with the option to either play games or view the library. The game category uses decks from the library to generate games for the user. The view library function has powerful functionality that allows you to manipulate your studying decks by adding, removing, or editing elements of the deck. From there you can study decks with options to read/write or listen/speak. The Glossa server is responsible for maintaining the library and the data inside of it.

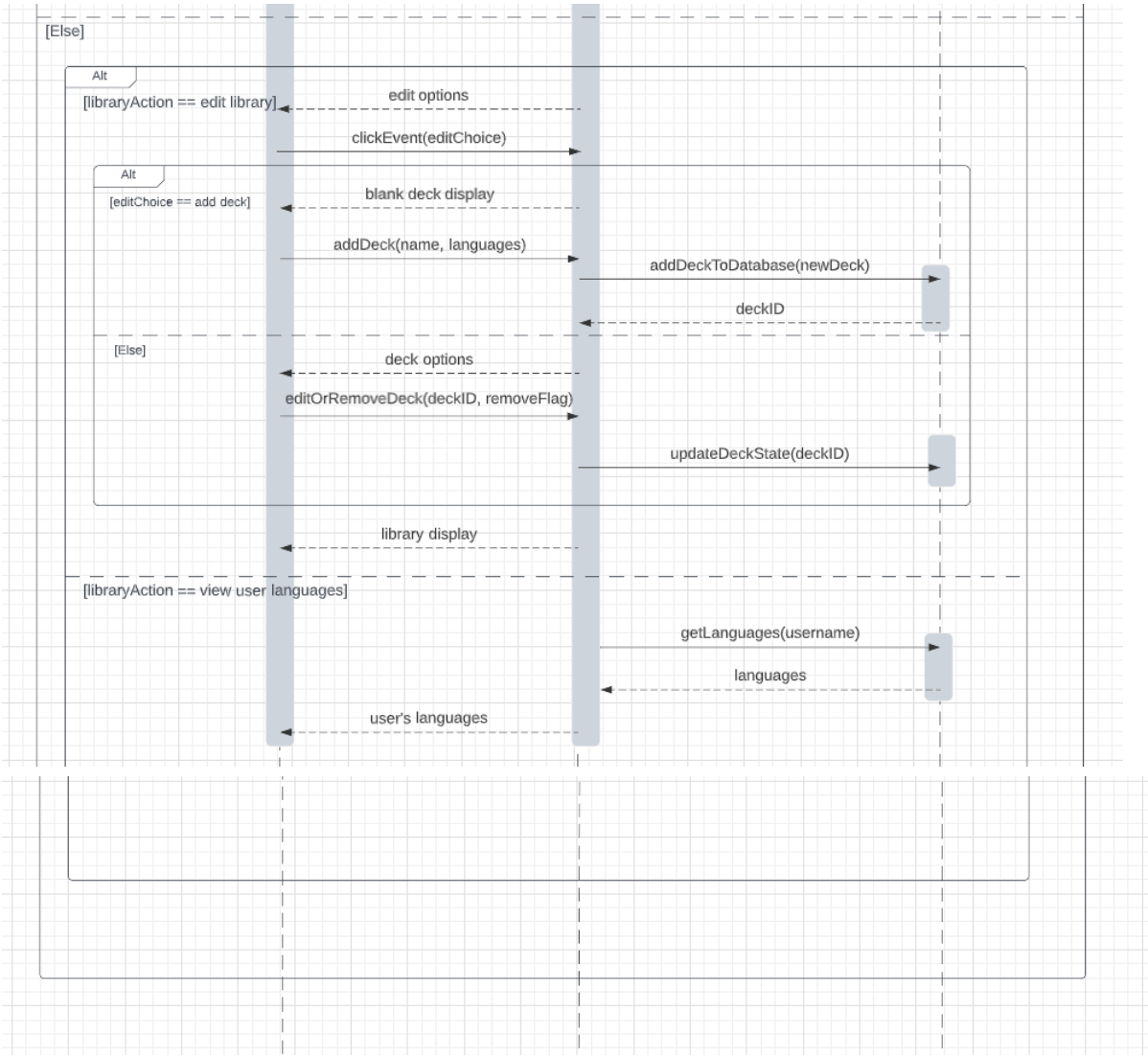
7. [15 POINTS] Sequence diagram – Provide sequence diagrams (similar to Figure 5.6 and Figure 5.7) for each use case of your project. Please note that there should be an individual sequence diagram for each use case of your project. (Ch 5 and Ch 7)

"Log In" Use Case:

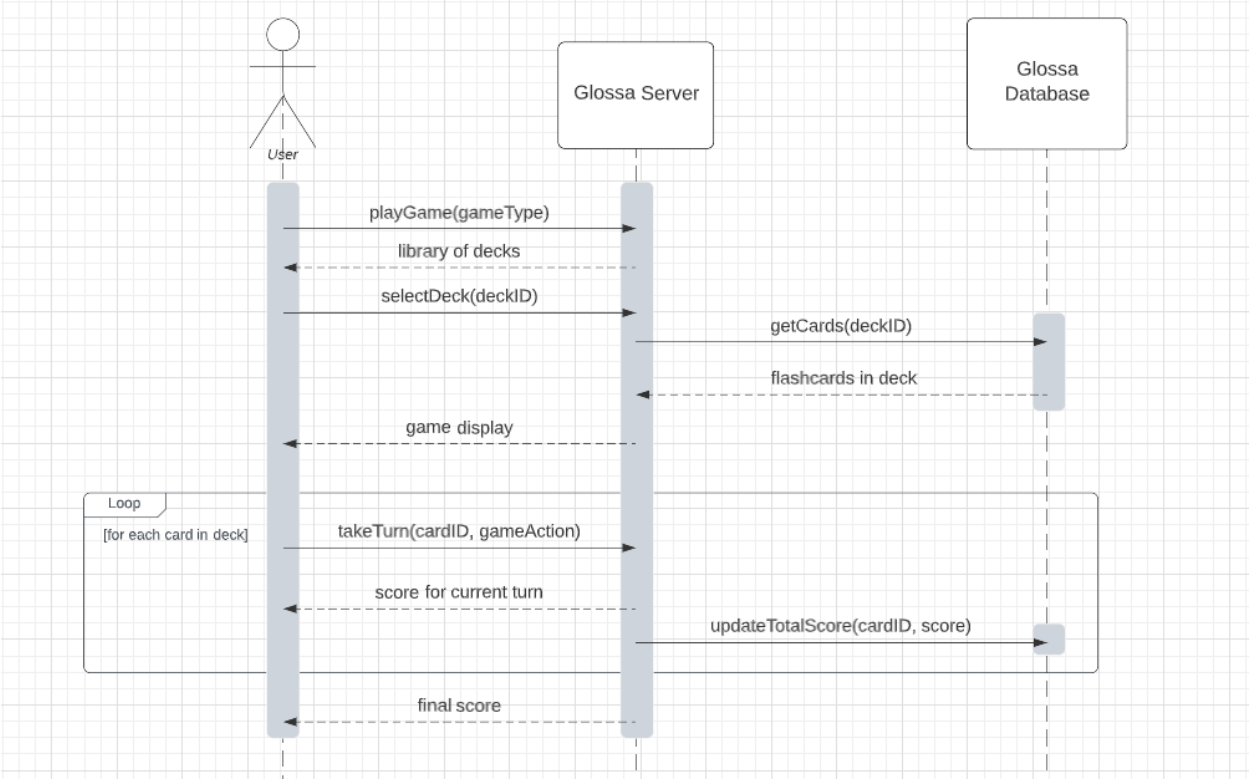


“View Library” Use Case:

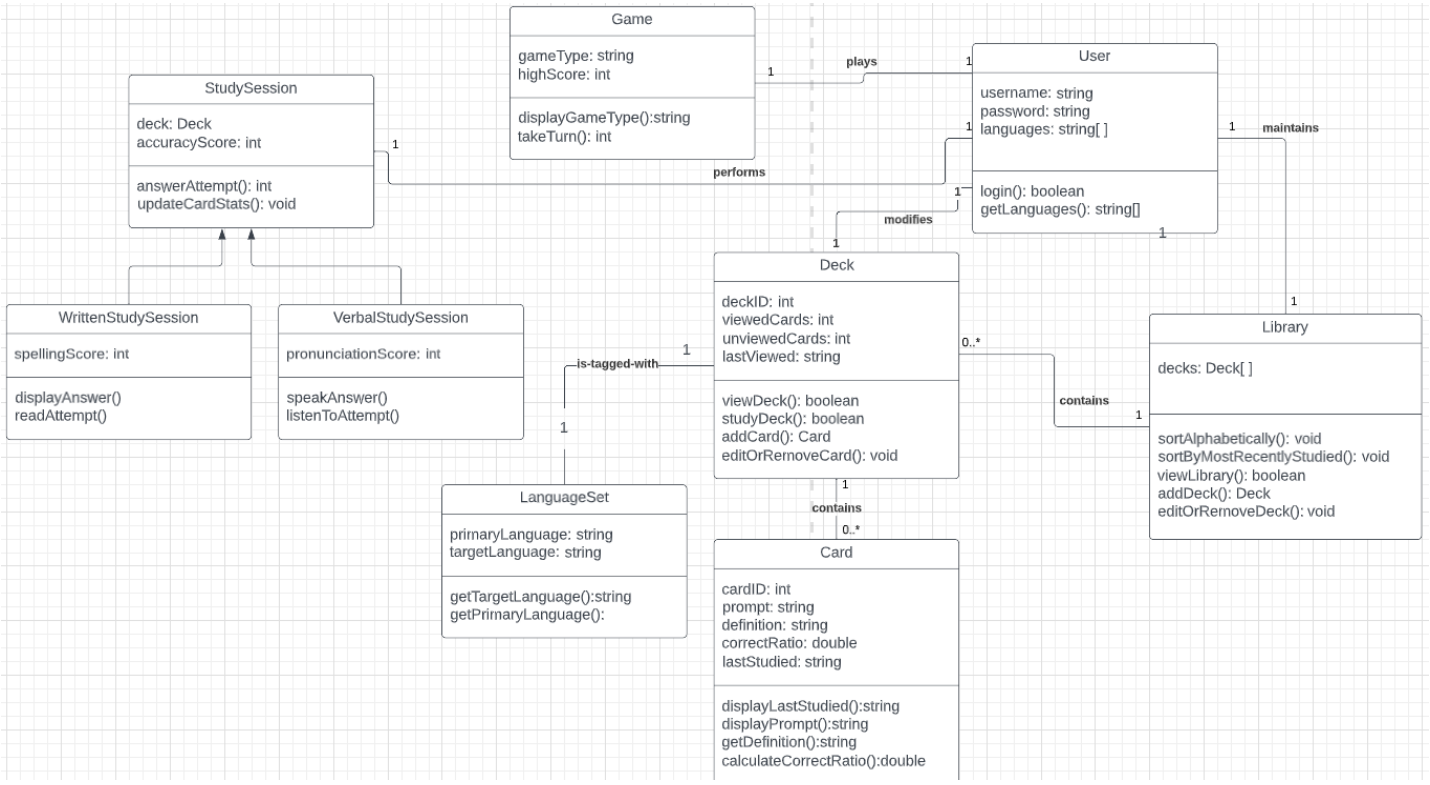



[Else]
Alt
[libraryAction == edit library]
edit options
clickEvent(editChoice)
Alt
[editChoice == add deck]
blank deck display
addDeck(name, languages)
addDeckToDatabase(newDeck)
deckID
[Else]
deck options
editOrRemoveDeck(deckID, removeFlag)
updateDeckState(deckID)
library display
[libraryAction == view user languages]
getLanguages(username)
languages
user's languages

“Play Game” Use Case:



8. [15 POINTS] Class diagram – Provide a class diagram (similar to Figure 5.9) of your project. The class diagram should be unique (only one) and should include all classes of your project. Please make sure to include cardinalities, and relationship types (such as generalization and aggregation) between classes in your class diagram. Also make sure that each class has class name, attributes, and methods named (Ch 5).



Classes:

Game: Game class shows the types of games available to the user.

User: User Class contains parameters to verify the login information of the user.

Deck: Deck Class allows the cards to be sorted alphabetically or by recently studied. Decks can be added,modified, and deleted.

Language: Language class holds the current language String of the Users chosen deck.

Card: This class constructs what the cards are made of including a prompt, the prompts definition, a correct ratio that tracks how many times you get a card right, and a last studied variable that tracks the last prompt you studied

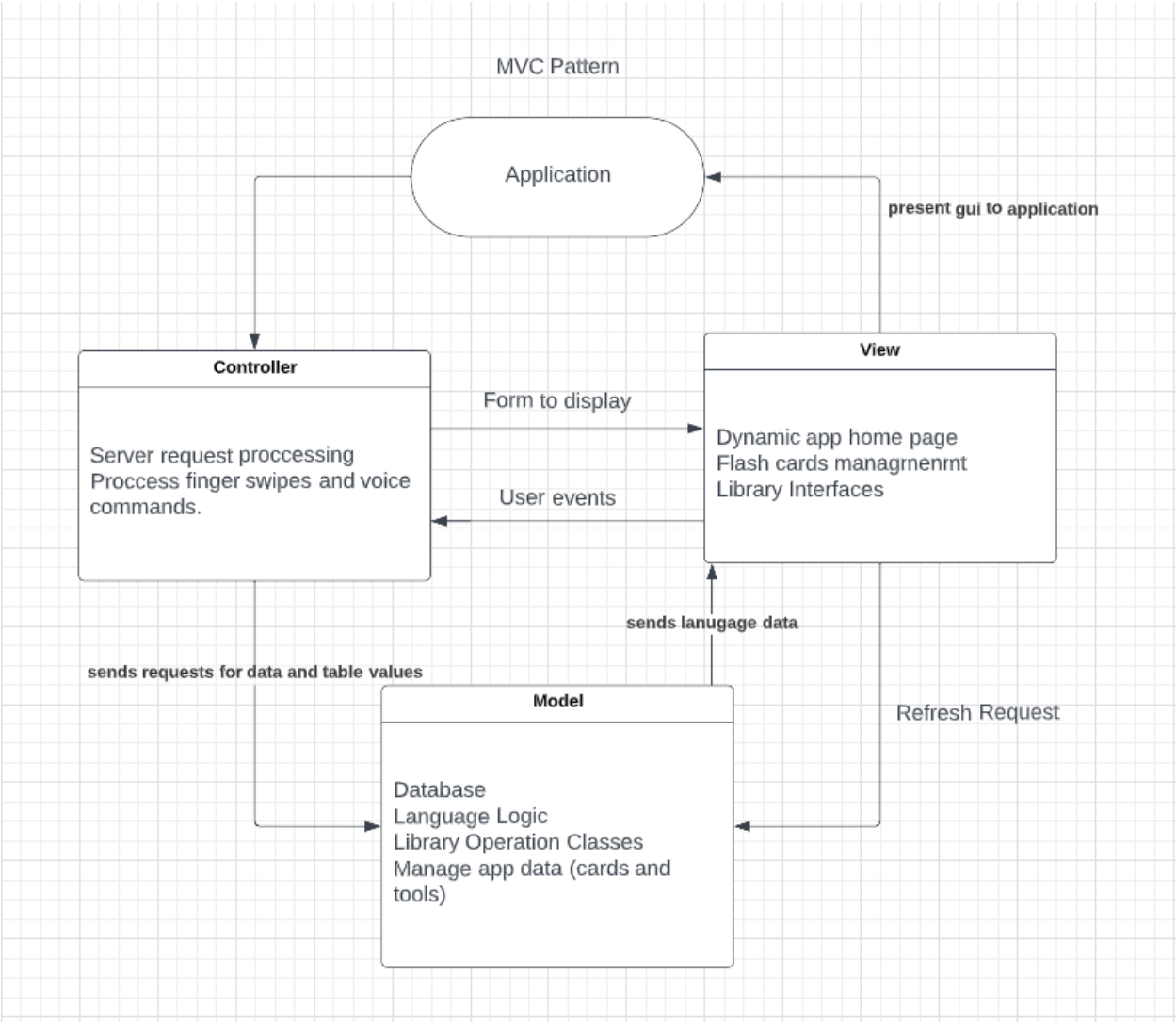
Library: Class that holds all the cards in an array called Deck[]. This class can sort the cards alphabetically, sort by recently studied. This class can also add decks, remove decks and edit decks to give the user more control of what they study

StudySession: Represents a session of a user studying a flashcard deck

WrittenStudySession: Type of StudySession focused on literacy

VerbalStudySession: Type of StudySession focused on verbal fluency

9. [15 POINTS] Architectural design – Provide an architectural design of your project. Based on the characteristics of your project, choose and apply only one appropriate architectural pattern from the following list: (Ch 6 section 6.3) 9.1. Model-View-Controller (MVC) pattern (similar to Figure 6.6) 9.2. Layered architecture pattern (similar to Figure 6.9) 9.3. Repository architecture pattern (similar to Figure 6.11) 9.4. Client-server architecture pattern (similar to Figure 6.13) 9.5.
Pipe and filter architecture pattern (similar to Figure 6.15)



The Model View Controller model was chosen because of its design. It naturally separates the components so the system's components are forced to interact with each other and are not completely centralized. We wanted users to have multiple ways of viewing the flashcard data.

The Model component: This would manage our users slide deck containing each of their individualized flashcards. It would also contain the languages containing the syntax and

grammar of a user's chosen language. It would contain the user's account data and progress in a given language diagram.

The View component: The view component contains the User interface which should be standardized for all users. This includes the presentation of slide decks and library interface, and user account details.

The Controller component: The controller component will regulate the user's interaction with the application. The controller will need to select cards and libraries by tapping the screen as well as swiping through flashcards.