

CS3354 Software Engineering

Final Project Deliverable 2

Glossa

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

1. [5 POINTS] Well described delegation of tasks, i.e. who did what in the project. Now that your project is complete, you are required to submit the delegation of tasks from beginning of the project until the end. Please make sure to fairly distribute tasks in the team and remember that at the end of the semester, each member of a team will receive the same grade. See grading policy below for more detail. If no/poor contribution by a member, please specify clearly so that we can grade each student 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

PROJECT 1 DELIVERABLE CONTENT

[10 POINTS] Setting up a Github repository.

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

[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.

[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

[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 providing 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.

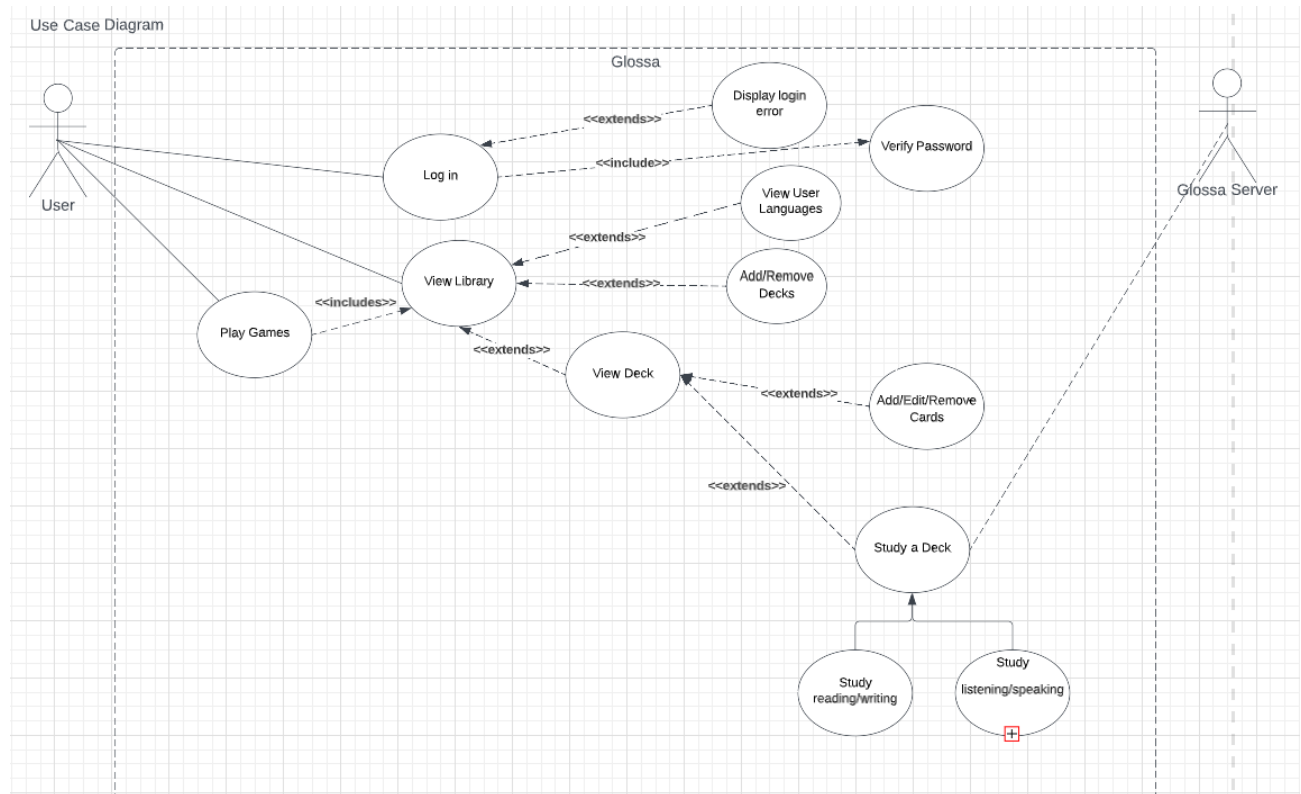
Regulatory: The app shall fall within FCC requirements and Android/Apple App Store requirements.

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.

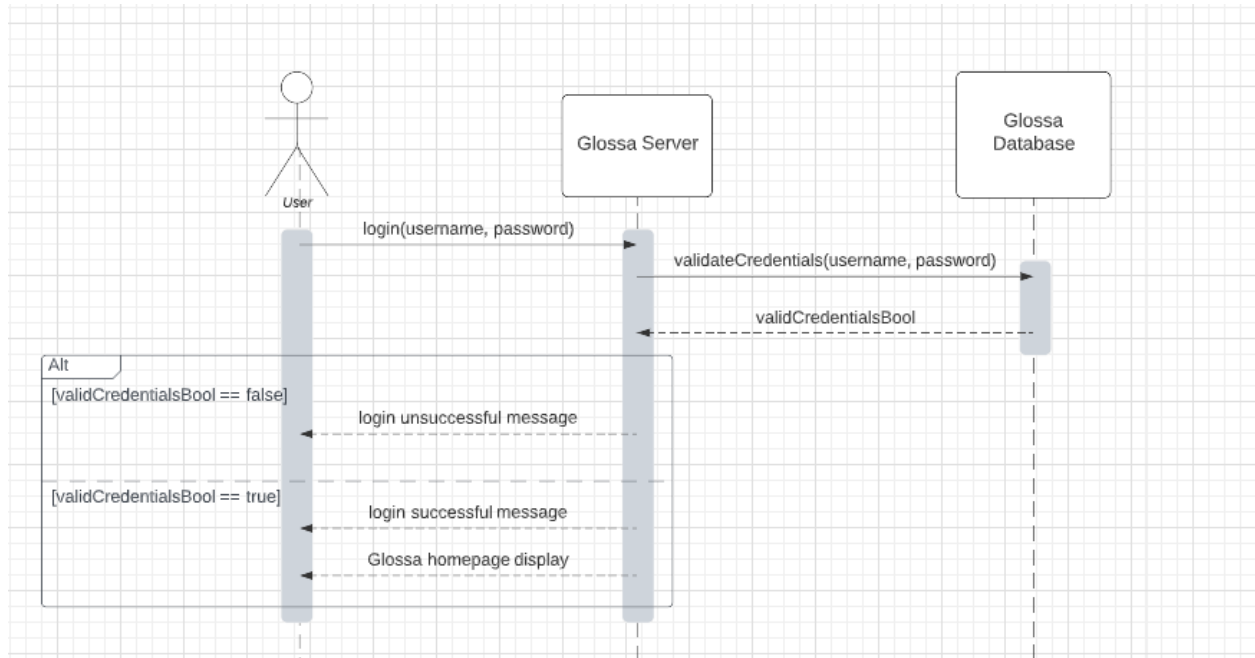
[15 POINTS] Use case diagram – Provide a use case diagram (similar to Figure 5.5) for your project. Please note that there can be more than one use case diagram as your project might be very comprehensive. (Ch 5 and Ch 7)



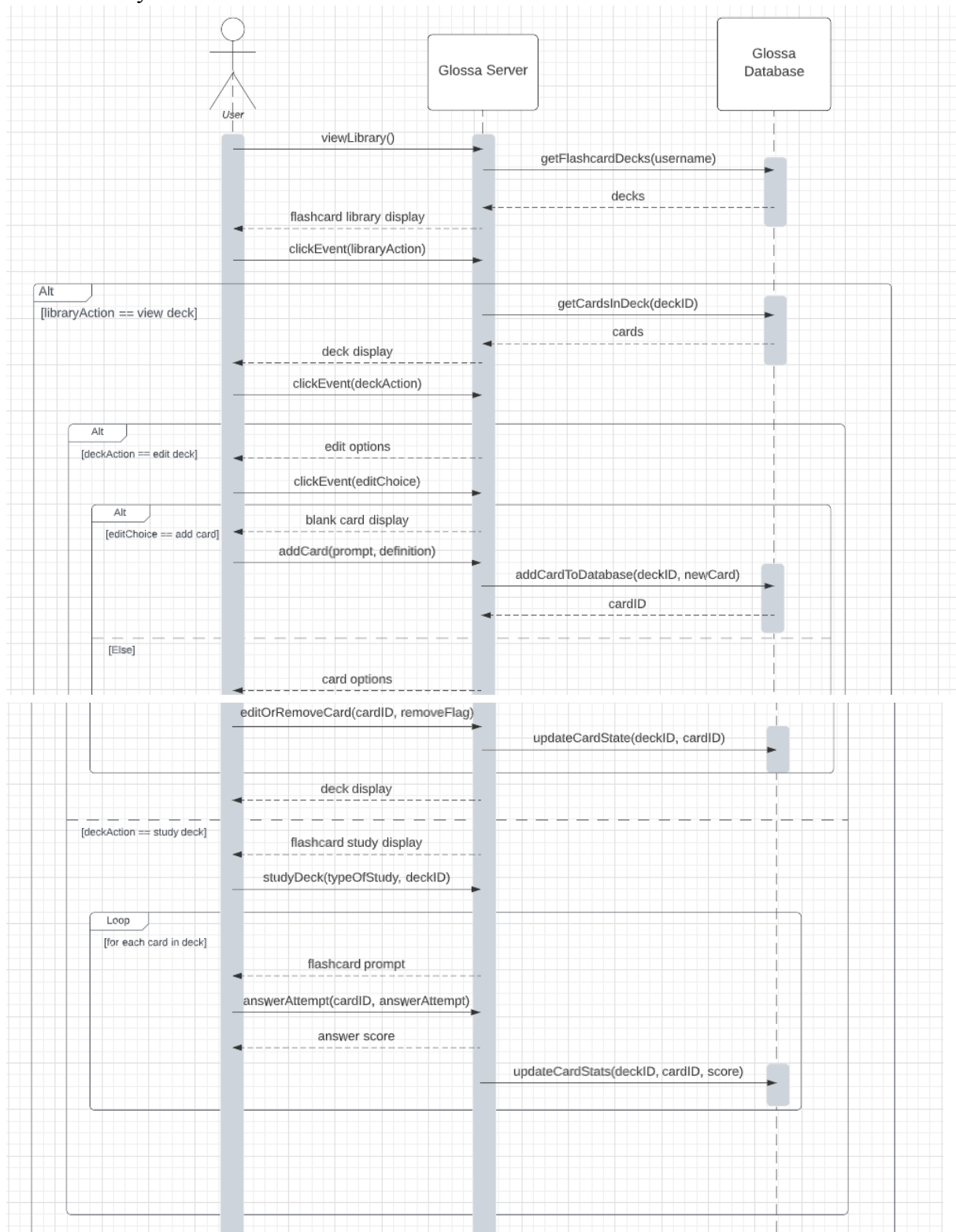
After logging in, the user is presented with the option to either play games or view the library. The game category uses decks from the library to generate games for the user. The view library function has powerful functionality that allows you to manipulate your studying decks by adding, removing, or 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.

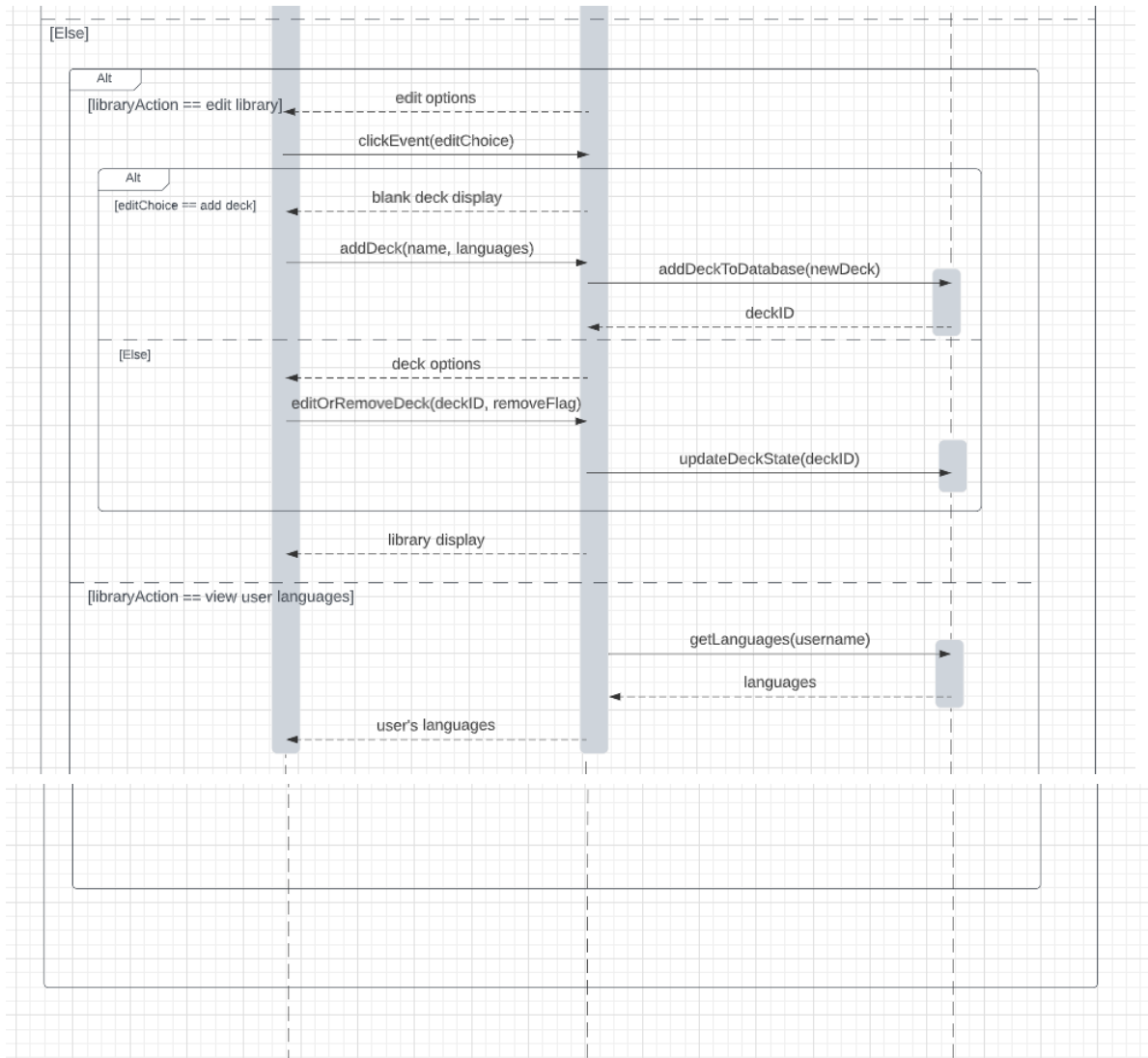
[15 POINTS] Sequence diagram – Provide sequence diagrams (similar to Figure 5.6 and Figure 5.7) for each use case of your project. Please note that there should be an individual sequence diagram for each use case of your project. (Ch 5 and Ch 7)

“Log In” Use Case:

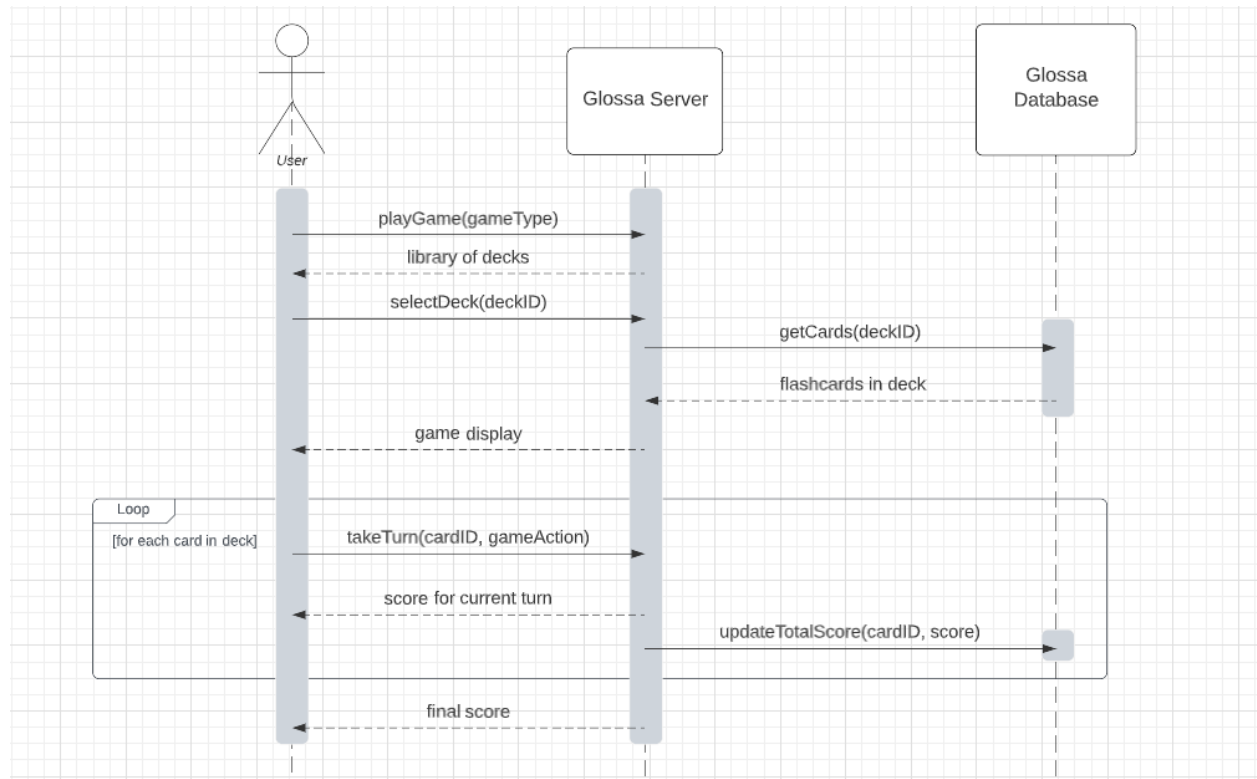


“View Library” Use Case:

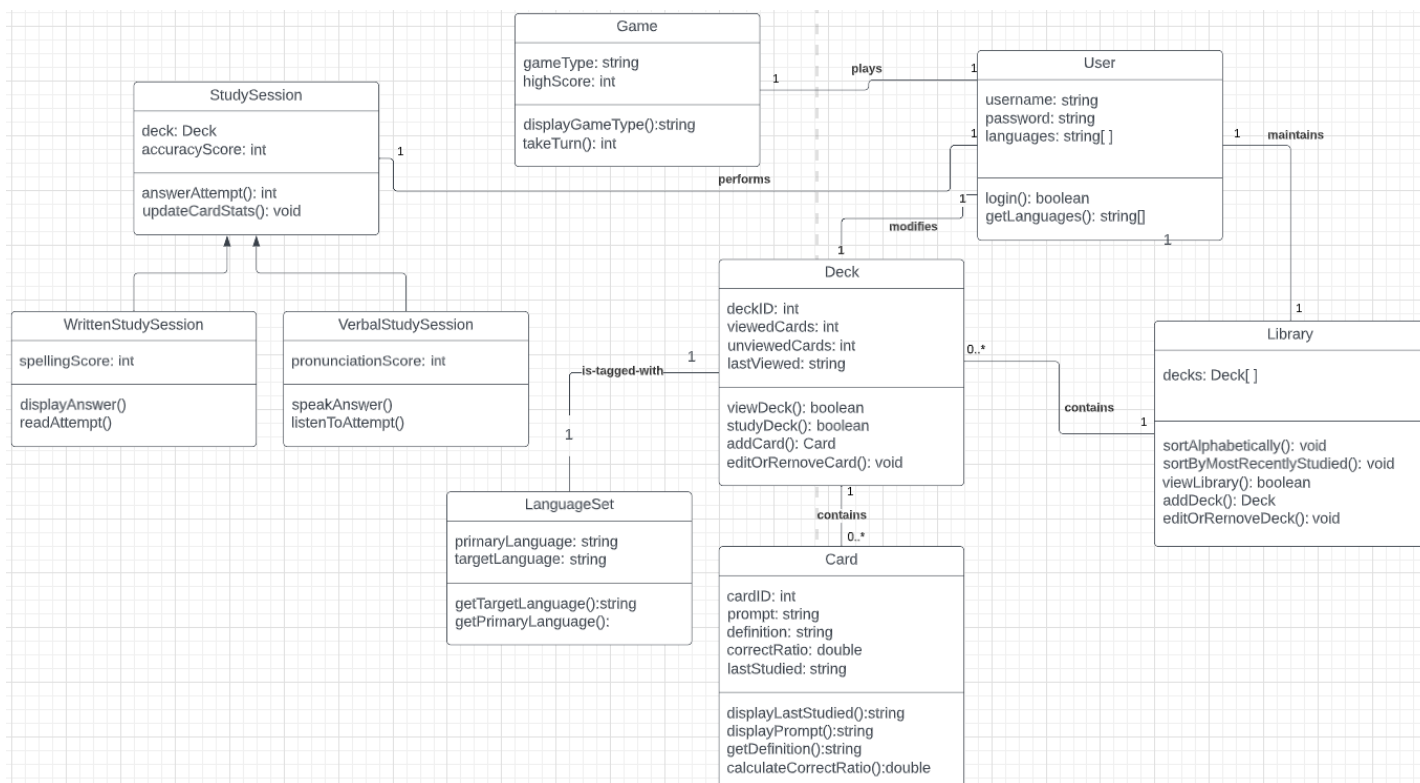




“Play Game” Use Case:



[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

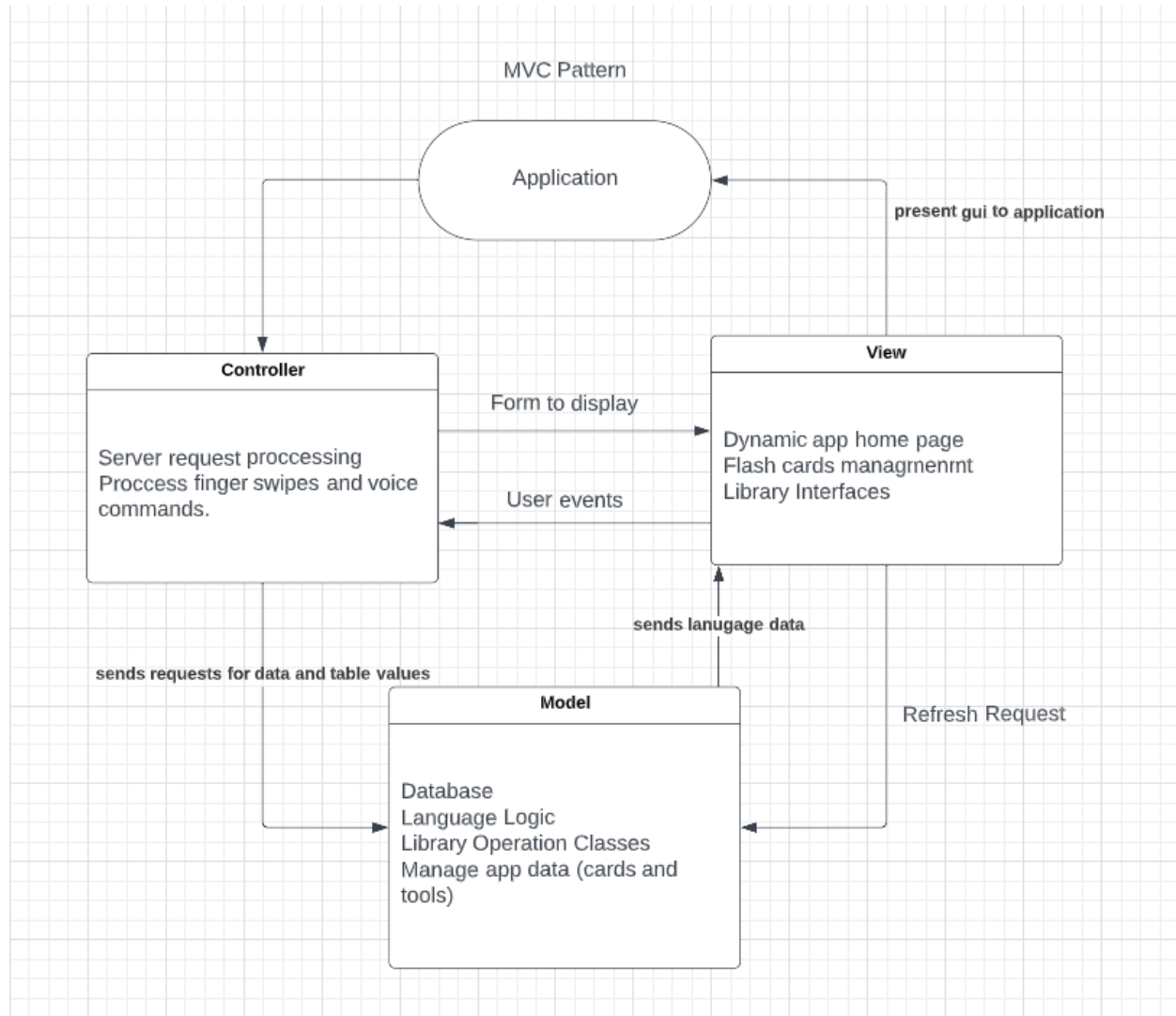
Library: Class that holds all the cards in an array called Deck[]. This class can sort the cards alphabetically, sort by recently studied. This class can also add decks, remove decks and edit decks to give the user more control of what they study

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

[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.

PROJECT DELIVERABLE 2 CONTENT

3. [35 POINTS] Project Scheduling, Cost, Effort and Pricing Estimation, Project duration and staffing: Include a detailed study of project scheduling, cost and pricing estimation for your project. Please include the following for scheduling and estimation studies:

3.1. [5 POINTS] Project Scheduling. Make an estimation on the schedule of your project. Please provide start date, end date by giving justifications about your estimation. Also provide the details for:

- **Whether weekends will be counted in your schedule or not**
- **What is the number of working hours per day for the project**

We estimate that our project will take about 6 months to complete. Development will take 3-4 months for a team of 4 developers based on the estimation in 3.2, and the remaining 2-3 months will be used for training, testing, and buffer time. Our team will work a standard schedule of 8 hours per business day (excluding weekends). The start date will be January 9th, 2023, and the end date will be July 7th, 2023.

3.2. [15 POINTS] Cost, Effort and Pricing Estimation. Describe in detail which method you use to calculate the estimated cost and in turn the price for your project. Please choose one of the two alternative cost modeling techniques and apply that only:

- **Function Point (FP)**
- **Application composition**

Our group is choosing to implement the COCOMO algorithmic estimation technique. Our justification for choosing COCOMO is that it is more objective than the FP method; for example, complexity is calculated from the number of views and data tables rather than chosen subjectively by us. We think using a more comprehensive model will help us get as accurate of an estimate as possible.

Step 1: Count the number of screens, reports, and 3GL components

A prototype of our flashcard-based language learning app has:

6 screens: 1 for the user's library, 1 for a deck of flashcards, 1 for a study session, 1 for a flashcard game, 1 for a results/score page, and 1 for the user profile.

3 reports: 1 produced after an individual study session, 1 with the results for all completed study sessions, and 1 for game results.

9 3GL components: 1 for each class in our class diagram

Step 2: Determine complexity levels of each screen and report

5 out of 6 of the screens have fewer than 7 views and access fewer than 4 data tables, so they are simple. The study session screen contains 3-7 views and 4-7 data tables, so its complexity is medium. Each report contains fewer than 4 sections and accessed fewer than 4 data tables, so they are all simple.

Step 3: Get complexity weights for each screen, report, and 3GL component

Screens: $1 * (\text{medium complexity weight}) + 5 * (\text{simple complexity weight})$
 $= 1 * 2 + 5 * 1 = 7$

Reports: $3 * (\text{simple complexity weight}) = 3 * 2 = 6$

3GL Components: $9 * (\text{difficult complexity weight}) = 9 * 10 = 90$

Step 4: Add weighted counts to get the Object Point count

OP count = screens weighted count + reports weighted count + 3GL components weighted count = $7 + 6 + 90 = 103$

Step 5: Estimate the percentage of reuse and compute New Object Points

There will be an estimated 5% reuse in our project. Some of the code related to flashcard studying sessions can be reused with some modifications for the code related to flashcard games.

$$\text{NOP} = \text{OP} * (100 - \text{Reuse}) / 100 = 103 * (100 - 5) / 100 = 97.85$$

Step 6: Determine object point productivity

Developer experience and capability: Low (7)

ICASE maturity and capability: Low (7)

PROD = Average of 7 and 7 = 7

Step 7: Compute person-month effort

$$\text{PM} = \text{NOP} / \text{PROD} = 97.85 / 7 = 13.98 = 14 \text{ person-months}$$

The estimated cost is 14 person-months.

3.3. [5 POINTS] Estimated cost of hardware products (such as servers, etc.)

The main hardware cost will be the server. We will get our server as part of a hosting plan that includes a web server, databases, and a domain name. We decided to use HostGator because it guarantees a high uptime percentage and is simple to use for Windows-based websites [1]. The monthly cost for the hosting service will be \$3.50 [1].

We also want to have an external hard drive to use as a backup. Although the price can vary depending on storage capacity and power requirements, we expect to pay around \$150 for several terabytes of storage [2].

Finally, we will need to purchase computers for our developers to work on. The cost for a reliable laptop suitable for development will be around \$1,000 [3]. For a team of four developers, the total cost would be around \$4,000.

3.4. [5 POINTS] Estimated cost of software products (such as licensed software, etc.)

We will use freeware for most of our software products to minimize the cost. For a text editor, we will use Visual Studio Code, which offers a free download [4]. We will use WinSCP for our FTP client [5]. For version control, we will use GitHub Team, which costs \$44 per user per year [6]. Since we will have four developers working on the project, the total annual GitHub cost would be \$176, which is about \$15 per month.

3.5. [5 POINTS] Estimated cost of personnel (number of people to code the end product, training cost after installation)

Since we want to finish development within half a year and we have an estimated cost of 14 person-months (from 3.2), we will have a team of four developers. This gives us an estimated completion time of 4 months with a 2 month buffer. The extra 2 months will be used to cover training. Our developers are entry level, so their salary will be \$90,000, or \$22,500 per month. For all four of them, the monthly cost is \$30,000.

4. [10 POINTS] A test plan for your software: Describe the test plan for testing a minimum of one unit of your software. As evidence, write a code for one unit (a method for example) of your software in a programming language of your choice, then use an automated testing tool (such as JUnit for a Java unit) to test your unit and present results. Clearly define what test case(s) are provided for testing purposes and what results are obtained (Ch 8). Include your test code as an additional document in your submitted zip file.

The test plan for our software is to perform unit tests on each method. For example, the unit test for getting the score when the user attempts to answer a prompt is below. Once each individual unit has been tested, we will do top-down integration testing by gradually integrating components together, starting with the main method. Finally, we will do validation testing by releasing a beta version to users and adjusting the code based on their feedback.

The method we tested returns a score based on the accuracy of the user's attempt. If the user gets the answer completely right, a score of 10 is returned. If more than half the characters in the attempt are correct, they get a score of 5. Otherwise, they get a score of 0. Our tests demonstrate that the logic is working by testing an example for each of the three cases.

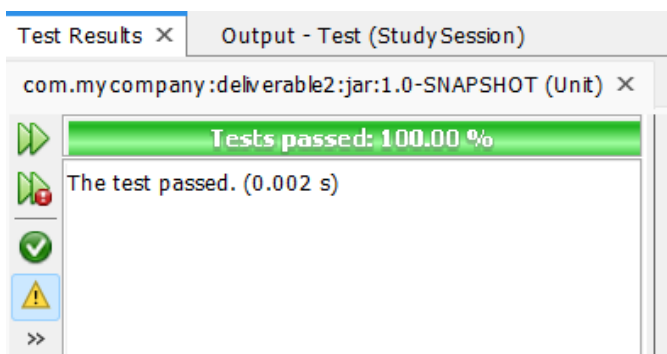
Code for the method:

```
1  /*
2  CS 3354.001
3  Glossa
4  */
5  public class StudySession {
6      public StudySession() {}
7
8      //Returns the accuracy score when user attempts to answer a prompt
9      public int answerAttempt(String expected, String attempt) {
10         //convert both strings to lowercase
11         expected = expected.toLowerCase();
12         attempt = attempt.toLowerCase();
13         //if the attempt matches the expected answer exactly, return a score of 10
14         if(expected.equals(attempt)) {
15             return 10;
16         }
17         //get the number of characters that differ between the attempt and expected answer
18         int length = Math.min(attempt.length(), expected.length());
19         int numDifferences = Math.abs(attempt.length() - expected.length());
20         for(int i = 0; i < length; i++) {
21             if(attempt.charAt(i) != expected.charAt(i)) {
22                 numDifferences++;
23             }
24         }
25         //if more than half the characters are correct, return a score of 5
26         if(numDifferences < expected.length() / 2) {
27             return 5;
28         }
29         //if fewer than half the characters are correct, return a score of 0
30         return 0;
31     }
32 }
```

Code for the tests:

```
1  /*
2   CS 3354.001
3   Glossa
4   */
5
6  import org.junit.jupiter.api.Test;
7  import static org.junit.jupiter.api.Assertions.*;
8
9  public class StudySessionTest {
10
11     public StudySessionTest() {}
12
13     //test the answerAttempt method
14     @Test
15     public void testAnswerAttempt() {
16         StudySession session = new StudySession();
17         //full credit test
18         assertEquals(10, session.answerAttempt("test", "test"));
19         //partial credit test
20         assertEquals(5, session.answerAttempt("tkst", "test"));
21         //no credit test
22         assertEquals(0, session.answerAttempt("cats", "test"));
23     }
24 }
25
```

Result:



The screenshot shows the JUnit test results interface. At the top, there are two tabs: "Test Results" and "Output - Test (StudySession)". Below the tabs, the test runner is identified as "com.mycompany:deliverable2:jar:1.0-SNAPSHOT (Unit)". A green progress bar indicates "Tests passed: 100.00 %". Below the bar, a message states "The test passed. (0.002 s)". On the left side, there are icons for a green checkmark (pass), a yellow warning triangle (warning), and a red X (fail), with a double arrow icon at the bottom.

Additional tests (not fully implemented):

```
/*
CS 3354.001
Samuel Jacobs
Arayan Patel
Test example file for Glossa language application
*/

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

public class GlossaTest {

    /**
     * Test of multiply method, of class Multiply.
     */
    @Test
    public void testGame() {
        //1A testing equals case
        System.out.println("testGame initial");
        int turn = 0; // check if the game initially has the users turn as false
        Game instance = new Game(); // create new game Object
        int highscore = 0; //creat
        String gameType = UserInput;
        assertNotNull("Testing if game object is not null", null, instance);
        assertEquals("Checking if UserName matches", userNameTest, "User");
    }
}
```

```
@Test
public void TeststudySession() {
    //1A testing equals case
    System.out.println("testUser ");
    StudySession instance = new StudySession(); // create new game Object
    instance.Deck = newDeck("English Study");
    String answer = "Password";
    int accuracyScore = 9;
    assertEquals("Checking accuracy", 9, accuracyScore);
}

@Test
public void testCard() {
    Card test1 = new Card();

    // checking if the attributes and methods have the correct data.
    assertEquals("Here is the the test for correctRatio", 30, correctRatio);
    assertEquals("Here is the the test for last studied", "10", test1.lastStudied);
}

@Test
public void testGame() {
    //1A testing equals case
    System.out.println("testGame initial");
    int turn = 0; // check if the game initially has the users turn as false
    Game instance = new Game(); // create new game Object
    int highscore = 0;
    String gameType = UserInput;
    assertNotNull("Testing if game object is not null", null, instance);
}
}
```

5. [10 POINTS] Comparison of your work with similar designs. This step requires a thorough search in the field of your project domain. Please cite any references you make.

Quizlet:

Quizlet is a flashcard application that allows users to create flashcard decks and use public decks that other users have created. Like our application, it provides users with a main library screen where they can view all of their saved and created decks [7]. Quizlet has the same type of matching games and simple study sessions as our application as well as additional games and quiz features [7]. However, our application has features tailored to language learning that Quizlet lacks. For example, Glossa users can choose a reading/writing study session or a listening/speaking study session, while Quizlet only offers an option for reading/writing [7]. Our application also helps users learn more effectively by providing an option to study cards based on how frequently the user gets them incorrect or by how infrequently the card is studied. Quizlet does not have this feature; users can only study cards in the order they appear in the deck or in a randomized order [7].

Rosetta Stone:

Our app is similar to Rosetta Stone in that it tries to help people learn new languages by providing an interactive platform that communicates with its users. It is also similar as they both use flashcards, sounds and activities to help the user grasp new concepts and languages [8]. The way that our app is different from Rosetta Stone is that we provide interactive games to help learn the language better, rather than simple memorization activities that Rosetta offers. We also have implemented a vast library to give the ability to choose what they need to learn. Also our app will be able to listen to the user, and help them pronounce words in the language of their choosing, instead of Rosetta just simply announcing the word [8].

Anki:

Anki is similar to our application. Anki is a flashcard based learning system that uses an user alterable spaced repetition algorithm to increase the efficiency of learning [9]. The flash cards in Anki are very versatile, accepting text, sound, pictures, video, and even latex data for each card [9]. The platform is open source licensed and available on Windows, Linux, MacOS, Android, and web platforms [9]. While these are very versatile cards, Glossa offers this alongside additional features such as games to keep the user interested and enjoying their language learning experience.

Duolingo:

Duolingo is one of the most popular language learning platforms [10]. While working to achieve the same thing, learning a language, it takes a completely different approach from our application Glossa. Duolingo is more similar to taking a class where it provides you with a structured path to

take with little flexibility. On the other hand, Glossa places a lot of emphasis on personal customization and flexibility. Both applications implement a review of sorts to help ensure that the learned knowledge has stuck. In Duolingo it is in the form of review tests after every X number of lessons [10]. For Glossa it is done by making cards that are missed appear more frequently. Both applications also have study features for reading, writing, listening, and speaking. A big difference between the two is that because Duolingo is structured like a course it has a limit to how much you can learn, at some point you will know everything that it can teach you. With Glossa because the decks are created by the users they can cover any kind of information and any level of difficulty whether it be grammar, vocab, or more complex things like idioms making it not have a point where it runs out of things to teach you. Duolingo also has several systems to try and engage the user in the app to keep them using it. They have a scoreboard to encourage competition between users, and they have a resource called hearts that restore over time and they give you access to certain things you don't have access to otherwise [10]. Glossa doesn't have anything directly comparable to that, but it does have games as a way to keep the user engaged by diversifying the study methods.

EWA:

EWA is a language learning app that takes a unique approach by allowing users to learn a foreign language from reading books and listening to videos [11]. They can tap on any word they don't know in a particular book and add it to a flash card deck to study. This concept is very similar to the app we plan to introduce, but EWA fails in execution for multiple reasons. For one the UI is hard to navigate. It's hard to tap on a word and add it to the deck that you want and not every word comes equipped with a translation. There is also only one deck that you can use and regularly update. This is on top of the app being buggy and generally unusable. One user echoes this sentiment by writing "Cool idea, execution needs work. The videos load sooo slowly. You can figure it out by the spacing before the video even loads. The audio when you click the speaker is slow, too. The UI isn't clear in the courses section" [11]. By contrast, Glossa's main focus will be on ease of use. We plan to emphasize ease of use and to equip the user with ready made card decks that include verb conjugations by default. These default decks will serve as special lessons the user can review. Our app will primarily be for people who already have limited working knowledge of the language who can jump right in and start using the app before knowing anything. Like EWA, we will include plenty of media (books, videos, etc.) for users to interact with since learning with media is the fastest and most efficient way to learn a language. However, our app will allow users to add words they don't know to their own custom decks, whereas EWA only allows the user to make one deck [11]. EWA also has a limited number of lessons a user can follow, whereas our app will periodically be updated with more lessons from time to time.

6. [10 POINTS] Conclusion - Please make an evaluation of your work, describe any changes that you needed to make (if any), if things have deviated from what you had originally planned for and try to give justification for such changes.

Glossa's goal is to give our users access to language learning flashcards and activities through a user friendly software application. We began this project using the spiral model due to the constant prototyping and upgrading that would need to be done throughout the early phases. The *spiral model* reduces the risk throughout the implementation phase and is something our group still thinks is in our best interest.

We initially intended to create a mobile app that would handle user interactions entirely locally, but given the vast language libraries we intended to use, we decided to implement the use of a server and a database. To facilitate this interaction we decided to host our application on a website instead as well.

Our functional requirements have changed slightly but the same idea has remained that we want a system that randomizes flash cards and will use this feature in a variety of games and studying tools. All non functional requirements that we stated in Deliverable 1 have been tested and proven to remain true throughout testing phases.

For our project we used the class, sequence, and case diagrams as general models for our design. These designs were followed and we haven't made any changes as of yet. The class diagram is somewhat implemented in our test plan but we have only been able to create a few working functions. If (or when) the project advances more classes and functions will be created according to our original diagrams.

The Architectural design that our group has chosen is The Model-View-Control (MVC) pattern. We chose this because we thought this would give us the best chance at running our app with the given resources we have while being able to interact/react with the user. This architectural design made sense due to the high volume of apps that follow the same pattern. Having multiple ways of viewing the flashcard data is essential to running our app.

According to project scheduling, our project will take until July 7th of next year to finish if a team of four developers worked on it from the 9th of January.

- This project will take 14 person-months to complete
- \$30,000 monthly cost of personnel
- \$15 a month for software products(github) / \$3.50 for hosting service
- \$4,000 a month for computers / \$150 for storage

For our testing, we plan to run a unit test on each method we created. These unit tests will check to make sure the methods return correct results; for example, our flashcard answer validating method returns a 10 for a correct answer, 5 for a partially correct answer, and 0 for a wrong answer. After these are tested and are working, we plan to open a beta version and adjust code based on user feedback. Once these tests come back positive it is our next goal to open Glossa to the public and review the feedback we get from it.

7. [5 POINTS] References: Please include properly cited references in IEEE paper referencing format.

Please review the IEEE referencing format document at the URL:

**<https://ieeeditaport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf>
(. It means that your references should be numbered, and these numbers properly cited in your project report.**

References

- [1] “Hosting for every website,” *HostGator*. [Online]. Available: <https://www.hostgator.com/web-hosting>. [Accessed: 04-Nov-2022].
- [2] M. S. Smith, “How to choose an external hard drive,” *Lifewire*, 20-Sep-2022. [Online]. Available: <https://www.lifewire.com/before-you-buy-external-hard-drive-6665942>. [Accessed: 04-Nov-2022].
- [3] C. Probst, “Best laptop for programming in 2022,” *TechRadar*, 04-Oct-2022. [Online]. Available: <https://www.techradar.com/news/best-laptop-for-programming>. [Accessed: 04-Nov-2022].
- [4] “IDE and code editor for software developers and teams,” *Visual Studio*, 31-Oct-2022. [Online]. Available: <https://visualstudio.microsoft.com/>. [Accessed: 04-Nov-2022].
- [5] “Free award-winning file manager,” *WinSCP*, 06-Oct-2022. [Online]. Available: <https://winscp.net/eng/index.php>. [Accessed: 04-Nov-2022].
- [6] “Pricing: plans for every developer,” *GitHub*. [Online]. Available: <https://github.com/pricing>. [Accessed: 04-Nov-2022].
- [7] “Learning tools, flashcards, and Textbook Solutions,” *Quizlet*. [Online]. Available: <https://quizlet.com/>. [Accessed: 04-Nov-2022].
- [8] “The experts in language learning,” *Rosetta Stone*. [Online]. Available: <https://www.rosettastone.com/>. [Accessed: 04-Nov-2022].
- [9] “Anki - powerful, intelligent flashcards,” *Ankiweb.net*, 2016. [Online]. Available: <https://apps.ankiweb.net/>. [Accessed: 06-Nov-2022].
- [10] Duolingo, “Learn a Language for Free,” *Duolingo*, 2019. [Online]. Available: <https://www.duolingo.com/>. [Accessed: 06-Nov-2022].

- [11] AppGrooves, “Positive & negative reviews: Ewa: Learn English & spanish - by Lithium Lab Pte Ltd - education category - 10 similar apps, 3 review highlights & 689,313 reviews - appgrooves: Save money on Android & iPhone Apps,” *AppGrooves*. [Online]. Available: <https://appgrooves.com/app/ewa-learn-english-words-easy-by-lithium-lab-llc/negative>. [Accessed: 06-Nov-2022].