Dennis Lim
Michael Doan

3010 Proposal

1)

Our project is a study-helper iOS application that will be using the idea of space reputation and active recall. It will require the user to enter a certain subject/class. The certain subject that would be broken down to sub-sections that will hold the flashcards that the users will have to create. As the user uses the application, it will keep track of which flash cards that user had trouble with by having the user reflect with these three categories: easy, starting to understand, need to review. With these three options, it will rearrange the flashcard deck in a certain order having the user review the card that they are having a hard time understanding to be shown up more often. If the user finds a certain flashcard easy the user would have two options: to stop review it or to keep reviewing it. If the user chose to keeping reviewing it would have the card to be placed onto the back of the flashcard deck/stack. If the user chooses the "understand" option, then it will have a longer push back onto the stack and the user will eventually review the card. The goal is by the end study session the user will have reviewed all his/her cards and have chosen the option easy and chose the option to stop review all the cards. By then the user has now finished studying. So, each option will decide where the card will be placed in the stack. For example, "easy" option, the flash card would be pushed back to back of the deck, "understand" option will have the card be placed in cards middle of the deck and "needs review" option will have the card be placed in forth of the deck. Each decision the user decides to choose will influence the space repetition algorithm. Meaning if the user was struggling with a specific card, the algorithm will make sure that specific card would be tested more often than the others, but the algorithm will also give a good amount of time in between when the user had studied that specific card to when he/she needs to study it again. The big part about the flash card is that its emphasis the study technique of active recall. It is forcing the user to retrieve concepts from his or her brain and apply it.

The next highlight of this application is its space reputation algorithm and features (note we are stilling trying to develop our algorithm (might implement the space repetition algorithm SM-2 or maybe model it based on the Fibonacci), but we are going to give you guys the gist of what its desired purpose). So, I guess we are going to start with the definition of space repetition, it is the idea of studying a specific idea or topic in a spread-out time and using active recall (so our flashcards) at different intervals of time. The purpose of it is to actually to learn the topic and get it into our long-term memory instead of cramming the material and have it in short term memory and forget after we take our exams. So, our algorithm will keep track of which flashcards the user needs to study and have a time-stamp when they last studied the topic and give a user a reminder the he or she needs to review this certain topic on their phone or when they open up this application. A feature that our application will have is a layout of all the class/subject with a subtopic:

Ex.
CSCI 3010:
- C++, objects, pointers & references (02/21)
- Bash Scripting (02/21)
- Const, Constructor, Enum (02/19)

Const, good coding practices, terminal (02/20)
Version Control, Git (02/18)

So CSCI 3010 is our class/subject and the topics/subsections are indented. There would be specific dates that would recommend the user what he or she should study first. The user would also get to reflect on which subject he/she understands by choosing the color green for good, yellow for moderate, and red for don't understand. We actually might have the algorithm calculate those colors by seeing how many times the users is unsure - sure for a certain deck.

2)

We are planning to use swift and create an iOS application and have our backend handled with CoreData. We both are new to iOS development; we are currently learning the technology at this moment in time. We do understand that the checkpoints for this project cannot contain us learning the tools, so we will be learning the tools on our own time and make sure that we will complete our checkout points.

3)

So the essential parts of the project are:

- The flash cards
  - The algorithm to place the order of the cards
- The algorithm for space reputation
- Our data base and how the flash card is stored
- How our GUI design
- The feature designs

Our project won't work if it doesn't have the flashcard GUI and functionality. This is important because this is basically the bulk of the application. The backup for this is how the cards are rearranged, instead of breaking the card in froths, halves, and end we would use time to place which card comes next.

Our project won't work if it doesn't have an algorithm for space reputation. We are currently trying to design this, but we are going to try to base it on the SM-2 algorithm because out of the other space repetition algorithm this is supposed to be the most basic. If we cannot implement the SM-2 algorithm we would be base our space reputation with the Fibonacci sequence but with a cap.

Our project won't work if it doesn't have if we don't have a database or a place to store our flash cards. If we cannot get our application to work with Core Data, we will fall to using Realm or SQLite databases because these are supposedly the most popular databases for iOS development.

Our application will prompt the user to make flashcards in the beginning, so we won't be having any flashcards preloaded in the application, but once the user has already made flash cards, we will load the cards the user needs to study.

4)

At the moment of time we do not need any outside recourses. All the technology we are using are free to use for open source projects.

5)

Our front-end part would be the GUI of the application. Basically, how our menu is going to displayed, how the flashcard is shown to the user, the little feature I mention before. I'm not sure where the algorithm is going to be placed in the front/back end area but it will get information from the user reaction and once the user is done with his/her study session, it will give the information to a specific method/object that would be acting as an observer. Our front end will inform change to that specific object and that object will then give the certain change whether it is the new deck created, information of the user's study session with the output to the back end and have it stored into the database. When a user is asking to pull a card deck, then we will retrieve our information through the proper commands for Core Data and load all the flashcards in a stack data structure.

We will have the observer design pattern integrated in our model where it will wait for either a user to create a flashcard deck or finish a study session where the algorithm will then spit out information on which cards the user needs to study next to the backend. We will also have our flashcard class be implementing a factory or strategy pattern where we have a flashcard with certain traits that are implemented by one of these patterns.

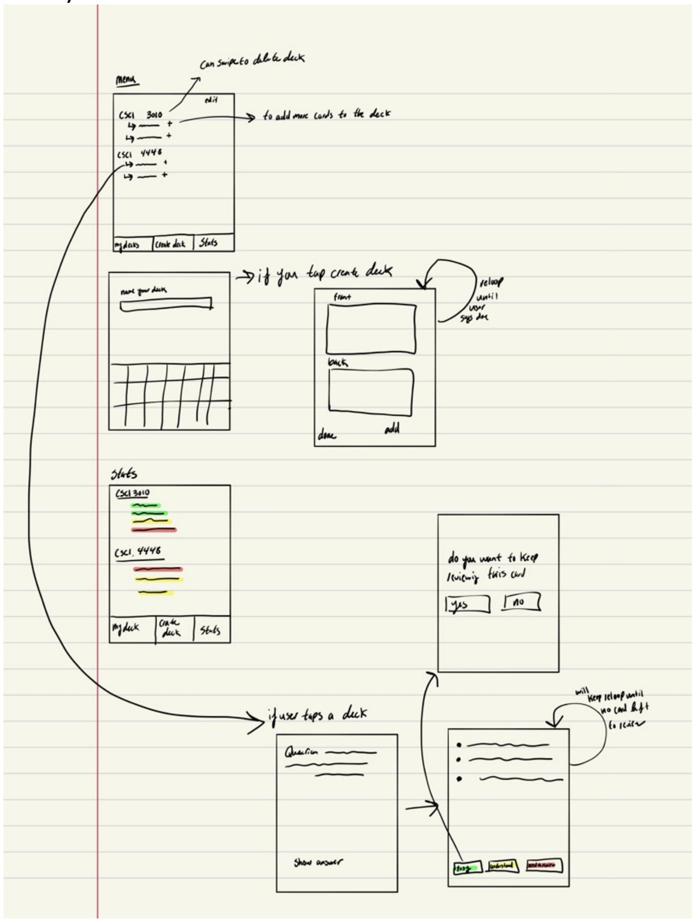
The architecture for our application:

- We have the flash card class that will consist of
  - private attributes:
    - Header – which is the front of the flash card
    - Info – which is the back of the flash card
    - Scores = [#easy, #understand, #review]
      - Basically, the application will keep track of how many times the users has selected this options and will be used for our space repetition algorithm
    - Score – is going to be a heuristic for the priority of the certain card in the stack
  - Public methods:
    - GetScore()
    - UpdateScore(String)
    - getHeader()
    - getInfo()
    - setHeader()
    - setInfo()
- We have a Study class with the
  - protected attributes:
    - Flashcard stack
    - Done stack
  - Private abstract method
    - Format()
  - Public method:
    - Exit()
    - Edit()
  - This class would have subclasses:
    - Front

            - Front will inherit the exit and edit method from study and the flashcard stack and done stack and override the format function
                - Public method:
                    - Show action – is the action that will cause transition to the back of the flashcard
            - Back
                - Same as the front it will inherit the exit edit method and the flashcard stack and the done stack. It will also overwrite the format function
                - Public method:
                    - response() – will update the flashcard with the user choice

- We have a create abstract class
    - Private variables:
        - Flashcard stack
        - deckCount
    - public method:
    - will be a sub class: deckCreation
        - makeDeck()
            - will loop and add cards to the flashcard stack
        - Done
            - will exit out of this interface once the user click this button
- We will also have a class for stats
    - This will retrieve the information from the decks
    - Will inform user which decks they need to review more and have a specific date they should study the cards,

6)



7)

Checkpoint 1 February 18th - 23rd (DESIGNING/GETTING THE APPLICATION RUNNING)

- Dennis:
  - Turn in: Part 2—a revised project proposal and a link to your GitHub with your Part 3 code on Sunday, February 23rd
- Dennis and Michael:
  - Have a UML diagram drawn out
  - Get our layout for each screen done, doesn't need any functionality done yet
  - Get our database design drawn out

Checkpoint 2 March 1st – 16th (CREATING THE FLASH CARD AND GETTING BACKEND WORKING)

- Dennis:
  - Can run one card and test the three actions: easy, understand, need to review
    - Have our algorithm of how the cards are placed back into the flashcard stack when the user chooses one of the 3 options

- Michael:
  - Get data base set up where we can store things in it
    - Need to get it be able to pull and push items into the database
  - We should be able to add cards and the database should update the existing deck
- Dennis and Michael:
  - We are able to create a flashcard deck and load it to the database
    - Need to be able to learn how to get its information to the database
    - Need to learn how to retrieve the information to put it on the card
    - NOTE: SHOULD BE TESTING THIS BUT NOT FULLY INTERGRATE THIS
      - Checkpoint 3 should have a more polish version of this
  - Should be able to create new decks
  - Should be able to remove decks to, so edit feature should be implemented here

<u>Checkpoint 3 March 17th - April 3rd (APP SHOULD BE USABLE)</u>

- Dennis:
  - In the display, in our deck we should be able to see the deck pop and choose it to study
- Michael:
  - And we should start the stats tab
    - Will lay out what decks there are with the subjects
    - Color will be added for which user needs to study more
- Michael and Dennis:
  - Should plan and test a space repetition algorithm
  - Polishing up our database design:
    - When we retrieve a deck, we need to parse the information and make it into a flashcard format
      - (If we got time, we can use multi-threading to make this process much faster)

<u>Checkpoint 4 April 4th - April 28th (SPACE REPEITION ALGORITHM SHOULD BE WORKING and APP SHOULD BE ALMOST DONE)</u>

- Dennis and Michael:
  - Our space repetition algorithm should be integrated with the flash cards
    - We can test our cards and see if we can go through all the cards in the decks
      - Should get the output of our algorithm for the space repetition and see how we should store it into our database (sanity check)
  - The stat tab should be all done and can display new decks when made

  - When pulling the subject of each decks, should have:
    - Name of the deck
      - Name of the topic (date when need to study) + <= to add more cards

<u>Checkpoint 5 April 29th APP IS DONE!</u>

- App should be fully functional and be able to perform all the task that was mention in this proposal!

Note:

- When it listed both of our names, it means we are going to meet up on campus or a library and try work out the problems together

8)

We will still come to class and stay engage with the certain topics that are presented. We will also stay active with the in-class lecture assignments and learn all the necessary topics. We will also stay up to date with every program exercise and finish them all in a timely manner.

Requirement summary:

1) Version control (+ continuous integration, strongly recommended)
    - We are planning to use Git for our version control
        - o We both know how to use this tool
    - Travis CI for continuous integration
        - o Using Travis CI because there is tech support for Mac/IOS and its free for open source projects on GitHub
2) A testing framework
    - XCTest
        - o Because it's a built-in framework that apple built for XCode
3) Swift
4) Flash card
5) Object-oriented language or paradigm
    - Will be observer pattern, factory, and strategy pattern