INF 121 Design Studio 3:

Collaborative Math Learning

Team Members:

Ahaz Bhatti	#47367726
Nehal Desai	#71764655
Tam Nguyen	#12972468
Luke Li	#40629074
Keanna Medina	#0000000

Introduction

Context

An educational software company wants to build a math learning application for children aged 5 through 9. The goal is to create an application that is both educational and engaging for young children. Furthermore, it should be tightly integrated with parents, teachers, and caretakers, so that a child's progress can be guided and observed. This data will also be collected by the company in order to better discern patterns of learning in children.

Audience

There are 3 main classes for the audience of this application: children, parents/caretakers, and teachers.

- Students (ages 5 through 9)
    - Young students are the main audience for our application. They will interact with Alexa vocally. We separated this class by "skill level," as our curriculum is based on web research regarding standard math levels for each grade.
- Teachers
    - Teachers interact solely with our web application. With a teacher user account, a teacher can create Session Plans, assigning concepts for each day throughout the week, which can be exported to associated student accounts. In this way, teachers can guide student learning.
- Parents and Caretakers
    - Parents and caretakers will use a parent user account. They are able to edit the Session Plans for the account's associated students. In addition, for each student associated with the parent account, the parent is able to view the achievements and log. This way, the parent can observe how well their student is doing and how long s/he uses the application.

Stakeholders

For this application, teachers and educational researchers may have ideas on how to improve the software for students and make it more engaging. Teachers may also have feedback for the functionality of their user accounts, particularly with regards to session plan creation and export.

Caretakers and parents are also stakeholders, as they are the primary buyers for this application and will likely want an application that is a good practice tool their children want to use. In the same vein, parenting blogs and magazines are stakeholders that may influence buyers by reviewing the application, mentioning benefits and possible improvements.

Finally, the educational company is also a stakeholder. The company will oversee the development and functionality of the application, and may require additional features based on feedback from the above stakeholders.

Goals

Since the main audience for our application are children, the Alexa skill should be easy to interact with. The target age range is five- through nine-year-olds, so the app must focus on math skills that are standard for these ages (i.e. counting, addition, multiplication, etc. instead of calculus).

It is important that the skill is also both educational and engaging, so it can help students practice math in a way that is fun, or at least not mind-numbingly boring. Lastly, the application should also provide feedback to some extent regarding how many questions the student gets correct.

Constraints

As mentioned, the application is geared towards children, so it must be easy to interact with. Children tend to have shorter attention spans and may require simplified language. Furthermore, if there are too many choices, it may confuse the students instead of augmenting engagement.

Another constraint is that the app must be supported by the Alexa virtual assistant. Moreover, for the primary audience, students, the application will be voice-activated. This limits visual methods of engagement, unless the user has an Alexa-enabled device with a screen. For users without one of those devices, our application will be entirely voice-activated.

Assumptions

- Children will be the primary audience of our Alexa skill.
- All users of the app can speak and understand English.
- Parents will accept the Terms of Use on educational company's website.
- Parents will be closely involved with the children using the app, and will be able to configure settings and proficiency levels when the account is created.

- Teachers will create Session Plans, if necessary, which contain concepts that children can work on.

Application Design

Main Functionality

- Allow parent user to configure proficiency levels on the web application for each student
    - These levels can be configured for each concept, and help tailor the diagnostic test to further understand the student's skill level
- Create a diagnostic test
    - The diagnostic test will be based on the above settings.
    - If a parent configures these settings to note that the student has not learned division yet, the diagnostic test will not test on division. It will only test on concepts that the student has encountered or learned, in order to understand where she needs practice.
- Assign skill levels based on curriculum standards for US elementary students
    - Our application assigns a skill level based on the diagnostic test. If a child is more advanced, she will receive a higher skill level, which relates to higher math standards.
    - For example, an average second-grader might be assigned a skill level of 2. However, an advanced second-grader could be assigned a skill level of 3, indicating that her math skills are at a third-grade level.
- Automatically generate session plans according to skill level
    - A session plan spans 7 days, and for each day the session plan decides which topics to cover based on the curriculum standards for the student's skill level.
- Allow parent users the option to edit the session plan on the web application
    - For example, a student has an exam on adding two-digit numbers, but Alexa has not covered it yet. The parent may choose to edit the session plan so the student would practice this concept before their exam.
- Allow parent user the option to create a schedule on the web application
    - The schedule states times for when the student is supposed to practice math with the application. Alexa will notify the student with a sound according to the scheduled time.
- Give students "achievements" based on performance
    - For example, a student can earn an achievement for the longest streak of correct answers, or for spending more time on the application than usual, or for following the schedule.
- Give students hints when they are stuck on questions
    - Students would feel frustrated if they are stuck on a question for too long, and
- Create a log of time intervals that students use the application, and their percentage of correct answers per session

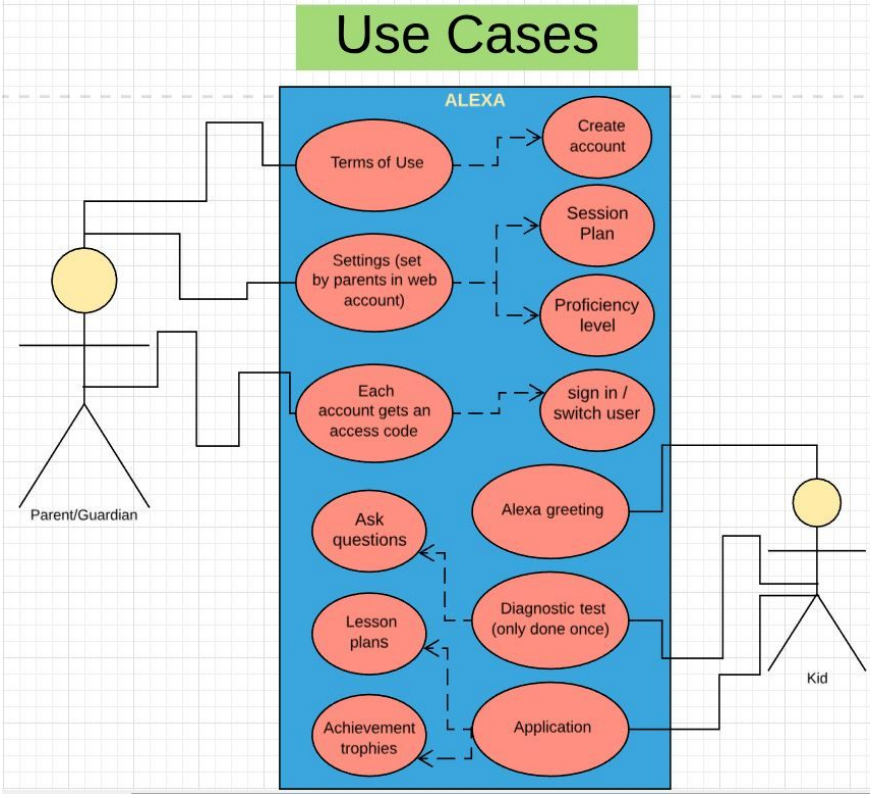
Process

1. Terms of Use
    a. Tell user to create account on educational company's website
    b. Parent creates an account on educational company's website
        i. Data collection - approve in terms of use for alexa skill
2. Settings (set by parents in web account)
    a. Proficiency/Comfort with different concepts
        i. Counting, multiplication, word problems that are related to concepts, etc
        ii. Parents/guardians will work with the child to configure these settings
        iii. Slider scale
        iv. Schedule days and times for Alexa to remind kid to learn math/use app as well as specify lesson plans for that day
    b. Account details for each kid
        i. Name, age, grade level, configure settings for each kid
        Ii. Themes for problems (i.e. Spongebob theme -> "If we have 2 Krabby Patties and we make 2 more, how many do we have now?"
3. Each account gets an access code
    a. Parents can use this to sign in
    b. Only needs to be used the first time
    c. Add access codes for multiple students
    d. Option to switch user to another account and log out
4. Alexa greeting
    a. Hello [kid's name]! Let's learn some math!
5. Diagnostic test (only done once)
    a. Ask questions based on the settings that parents specified (level, which concepts to include, etc)
        i. The younger the kid, the less questions
        ii. Up to 15 questions
6. Application
    a. Alexa will automatically start the kid on a lesson plan appropriate for their grade level curriculum, parent-defined settings, diagnostic test results
    b. Alexa will ask questions that focus on that concept at the user's specified level
        i. Based on diagnostic test
    c. Achievements gained during the session are updated on the user account to be viewed later
        i. Ex: Gold/silver trophy, highest consecutive streak per concept
    d. After a certain number of incorrect answers, Alexa will offer to take the lesson back to more basic concepts: "Let's go back"
    e. After a streak of correct answers, Alexa will ask the kid: "Do you want to switch?" (Possible answers: "Yes" and "No")
        i. If the answer is no, continue the current lesson plan

ii. If the answer is yes, Alexa will behave as though the kid issued a “switch” command (specified below)

f. If the kid issues the command “switch”, Alexa will pause the lesson and ask the kid: “Do you want to switch to harder questions or something different?” (Possible answers: “Harder” and “Different”)
   i. The kid can exit from this command with “cancel”

Use Cases



Architectural Design

Entities

- **User**

  Attributes

  - String userId // The user’s unique ID
  - String userEmail // The user’s email

- Student[ ] students // The collection of students belonging to this account (the parent's children)

  Methods
- addStudent(Student s) // Adds a new student to this account
- removeStudent(Student s) // Removes a student from this account

- **Parent** // Inherits from User

  Methods
  - toggleSettings(Student s) -> st.setSettings(args) // Changes this student''s settings
  - displayAchievements(Student s) // Displays this Student's achievements
  - checkStudentLog(Student s) // Checks this student's log
  - editSessionPlan(Student s) -> s.setSession(args) // Edits this student's session plan

- **Teacher** // Inherits from User

  Attributes
  - Session[ ] publicSessionPlan // The teacher's own session plans that are importable by students.

  Methods
  - updateSessionPlan() // Makes changes to the teacher's own session plan.
  - resetSessionPlan() // Clears the current session plan.
  - exportSessionPlan() // Updates all students' teacherPlan in the Students Array.

- **Student**

  Attributes
  - String studentId // Student's unique ID
  - String name // Student's name
  - int age // Student's age
  - int skillLevel // (1-5, kindergarten - 4th grade)
  - private Session[ ] teacherPlan // The student's teacher's session plan
  - private Settings studentSettings // The unique settings for this student
  - private Session currentSession // The current session this student is on
  - private Session[ ] mySessionPlan // The session plan that the student uses
  - public Achievement unlockedAchievements[ ] // This student's achievements
  - public Log studentLog // This student's log

Methods

- public setSettings(args[]) -> st.makeSettings(args[]) // sets this student's settings
- public setSession(args[]) // sets this student's current session. If multiple arguments are passed, sets this student's session plan.
- public generateSession() // generates a series of sessions based on the student's skill levels.
- private switchSession(Session s) -> this.currentSession = s // allows this student to switch their session to another one
- public importTeacherPlan() // Called by the teacher's exportPlan function to update teacherPlan
- private syncPlan() // Synch the teacher's session plan to be the using plan

- **Settings**

  Attributes

  - int addProf // The student's proficiency in Addition
  - int subProf // The student's proficiency in Subtraction
  - int multProf // The student's proficiency in Multiplication
  - int divProf // The student's proficiency in Division
  - int countProf // The student's proficiency in Counting
  - int wordProf // The student's proficiency in Word Problems

  Methods

  - void makeSettings(args[]) // Adjusts the student account's settings with 7 arguments passed to the parameter.

- **Session**

  Attributes

  - String concept // This session's concept
  - Question[] questions // The collection of questions available for this session
  - String[] studentAnswers // The collection of answers to the session's questions
  - double performanceScore // The calculated score based on correct answers for this session
  - int streakCount // The highest number of correct answers in a row for this session

  Methods

  - double calculateScore() // Calculates and sets performanceScore
  - int getNumOfQs() // gets the number of questions in this session (questions.length)
  - void addStudentAnswer() // Updates studentAnswers array when the student answers a question
  - String[] getAnswers() // returns an array of the answers given by a student

- void generateQuestion() // pulls information from data and creates a Question object with that information

- **Question**

  Attributes

  - <String, String> problem // a question-answer pair
  - int difficulty // The difficulty level of the question on a scale of 1-5
  - String answer // The answer to the question

- **Achievement**

  Attributes

  - String title // The title of the achievement
  - String timeAndDate // The specific time and date of which the student receives this achievement.

  Methods

  - displayAchievement() // Displays all achievements that a student has received either through voice or visually on a device that connects to Alexa.

- **Log**

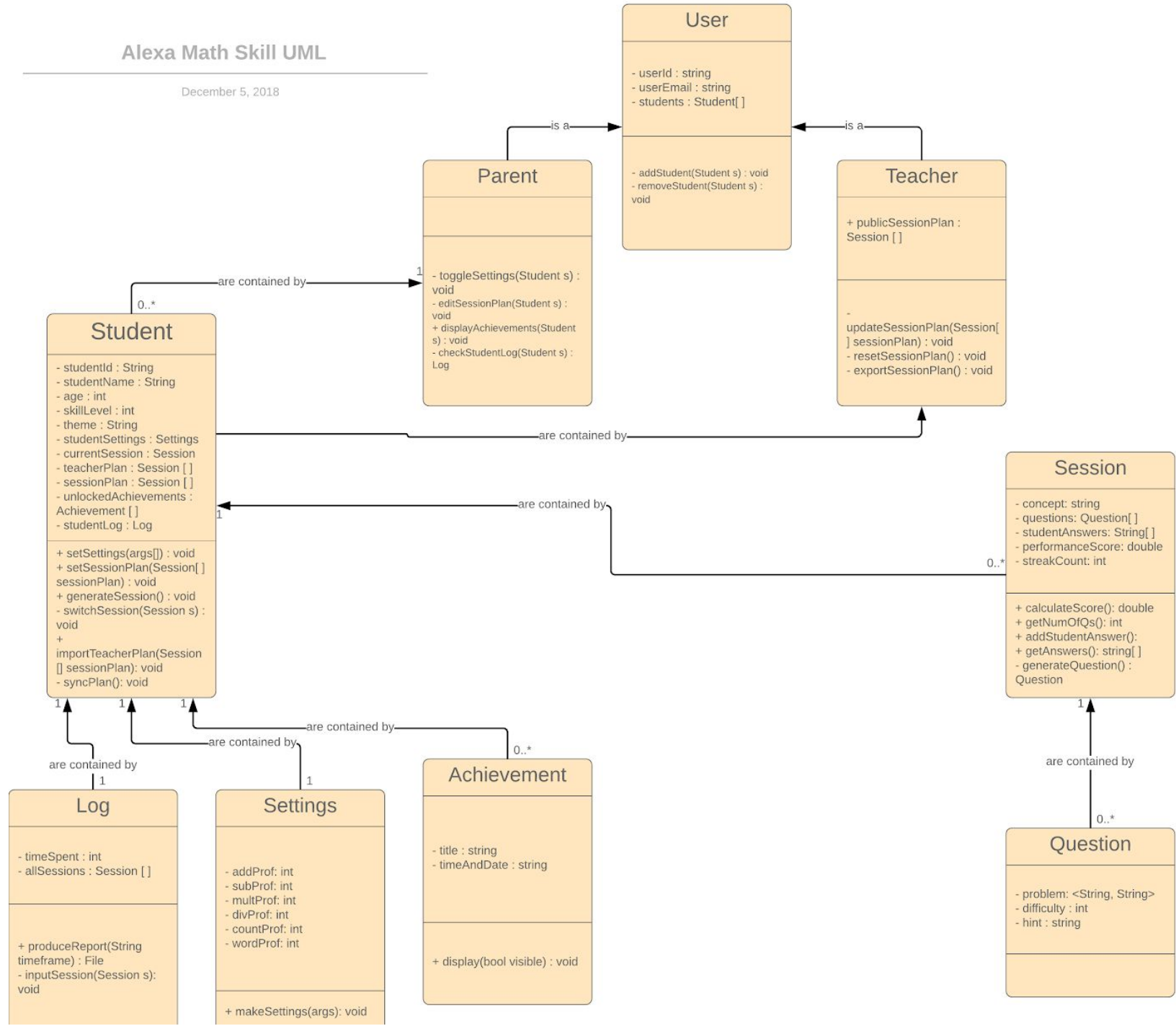
  Attributes

  - int timeSpent // The total time that a student has spent using the Alexa skill
  - Session[ ] allSessions // The array which contains all the sessions that a student has learnt.

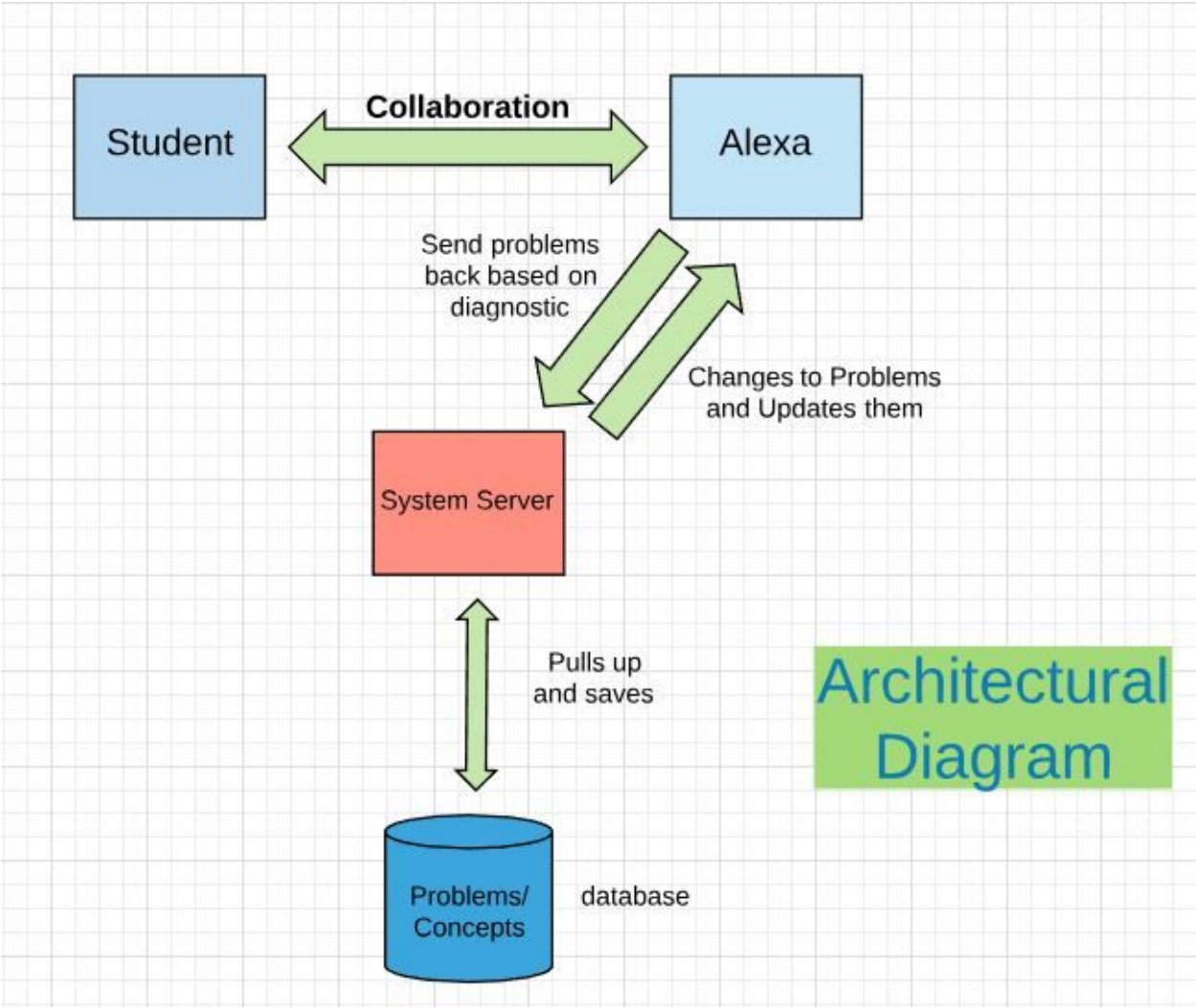
  Methods

  - produceReport(String timeframe) : File // timeframe is “daily”, “weekly”, “monthly”, if no actual parameter is passed, report is produced from start of student account to the present.
  - inputSession(Session s): void // Adds a session to allSessions

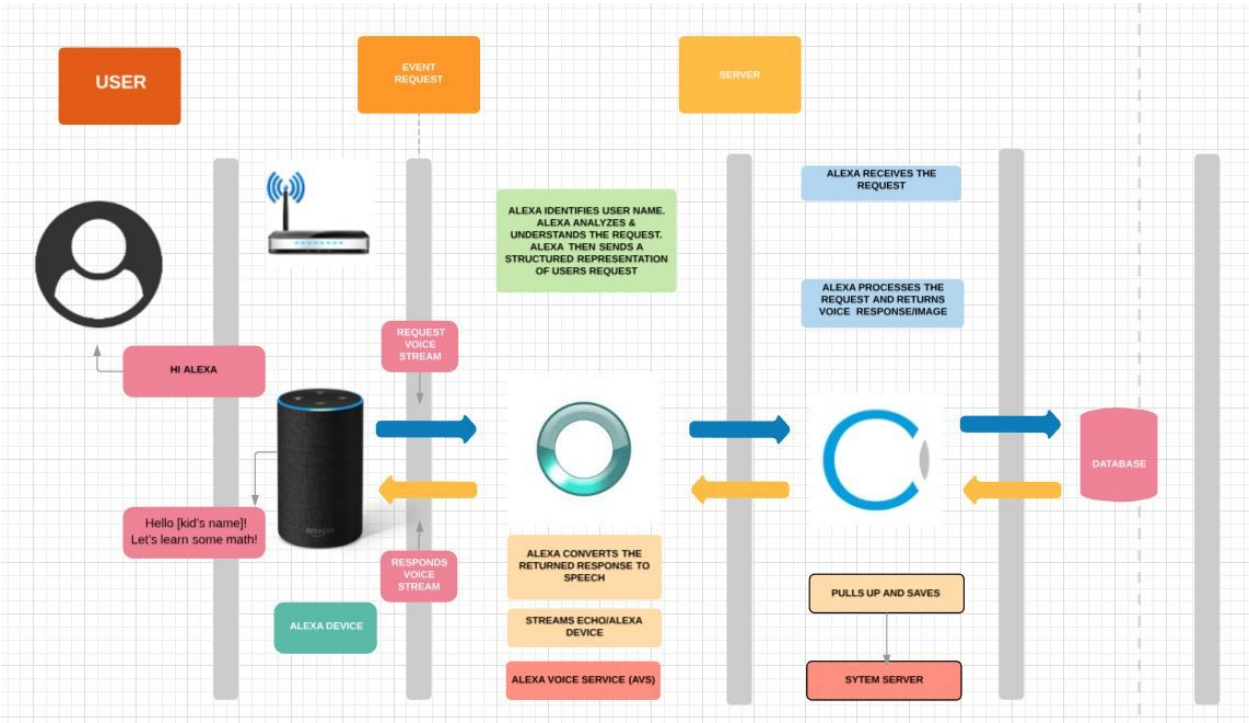
UML Diagram


Alexa Math Skill UML
December 5, 2018
User
- userId : string
- userEmail : string
- students : Student[ ]
- addStudent(Student s) : void
- removeStudent(Student s) : void
is a
is a
Parent
- toggleSettings(Student s) : void
- editSessionPlan(Student s) : void
+ displayAchievements(Student s) : void
- checkStudentLog(Student s) : Log
Teacher
+ publicSessionPlan : Session [ ]
- updateSessionPlan(Session[ ] sessionPlan) : void
- resetSessionPlan() : void
- exportSessionPlan() : void
are contained by
1
0..*
Student
- studentId : String
- studentName : String
- age : int
- skillLevel : int
- theme : String
- studentSettings : Settings
- currentSession : Session
- teacherPlan : Session [ ]
- sessionPlan : Session [ ]
- unlockedAchievements : Achievement [ ]
- studentLog : Log
+ setSettings(args[]) : void
+ setSessionPlan(Session[ ] sessionPlan) : void
+ generateSession() : void
- switchSession(Session s) : void
+ importTeacherPlan(Session [] sessionPlan): void
- syncPlan(): void
are contained by
1
0..*
Session
- concept: string
- questions: Question[ ]
- studentAnswers: String[ ]
- performanceScore: double
- streakCount: int
+ calculateScore(): double
+ getNumOfQs(): int
+ addStudentAnswer():
+ getAnswers(): string[ ]
- generateQuestion() : Question
are contained by
are contained by
are contained by
1
1
1
1
1
0..*
are contained by
1
Log
- timeSpent : int
- allSessions : Session [ ]
+ produceReport(String timeframe) : File
- inputSession(Session s): void
Settings
- addProf: int
- subProf: int
- multProf: int
- divProf: int
- countProf: int
- wordProf: int
+ makeSettings(args): void
Achievement
- title : string
- timeAndDate : string
+ display(bool visible) : void
are contained by
0..*
Question
- problem: <String, String>
- difficulty : int
- hint : string

Architectural Diagram



Architectural Details Diagram



Implementation Design

Algorithms and Pseudocode

Student

- setSettings (args[ ]):
    1. Passed (6) arguments which represent the six proficiency levels of each concept (add, sub, mult, div, counting, word problems)
        - setSettings(3, 5, 8, 6, 10, 1);
    2. Calls **Settings**'s *makeSettings(args[ ])* to make the actual changes to the private attributes.
        - Student.studentSettings.makeSettings(3, 5, 8, 6, 10, 1);

- generateSession(String concept, int skillLevel) :

1. Generates a **Session** object using the passed-in concept, *studentSettings* object, and skillLevel
2. Automatically appends this **Session** to the end of the *sessionPlan* array

- setSessionPlan(Session[ ] sessionPlan):
    1. Receives a *sessionPlan* as a result of **Parent** calling *editSession(Student s)* on this student
    2. Sets *this.sessionPlan = sessionPlan*

- switchSession(Session s):
    1. *this.currentSession* = s;

- importTeacherPlan(Session[ ] sessionPlan):
    1. Called by **Teacher**, therefore is marked **public**
    2. Calls *setSessionPlan(sessionPlan*) to set the student's *teacherPlan* to the teacher's *publicSessionPlan*

- syncPlan():
    1. Called only by student, therefore is marked **private**
    2. Makes student's *mySessionPlan* to be *teacherPlan*, then clears *teacherPlan* to be an empty array (therefore allowing for updates from the teacher when the teacher makes a new plan)

Teacher

- exportSessionPlan():
    1. Operates on the entire Student[ ] array
    2. Calls each student's *importTeacherPlan(Session[ ] sessionPlan)* function with this teacher's session plan array
        - for (student : students) { student.importTeacherPlan(publicSessionPlan) }

Parent

- toggleSettings(Student s):
    1. Sets a **Student**'s settings to a given **Settings** object
        - s.setSettings(new Settings(3, 5, 8, 6, 10, 1));

- editSessionPlan(Student s):
    1. Sets a **Student**'s *sessionPlan* to a given **Session** array of length 7
        - s.setSessionPlan([new Session(), new Session(), …])

- displayAchievements(Student s):
    1. Calls the *display()* method on all elements of the **Student'**s Achievements array
        - for (a : s.unlockedAchievements)
          {
          a.display(false) // if no external Alexa-enabled device with screen enabled
          a.display(true) // if device is enabled
          }

- checkStudentLog(Student s):
    1. Returns a **Log** object containing this **Student**'s log information
        - return s.studentLog;

Log

- produceReport(String timeFrame):
    1. *timeFrame* specifies how much information should be in the report.
        - "day": generates a report with session information for the past day
        - "week": generates a report with session information for the past week
        - "month": generates a report with session information for the past month
    2. Uses information from the collection of **Session** objects that have been attempted or completed by the **Student** of this log to generate a holistic report of Student's average scores, strengths and weaknesses and which sessions were completed or not completed
    3. Returns a (text) File object containing this information formatted in a comprehensive manner that is easy to read for parents

Achievement

- display(bool visible):
    1. If visible is **false**, an Alexa-enabled device is not paired or connected with this device, and therefore, achievement cannot be displayed to an external screen
        - Alexa will speak the achievement *title* and *timeAndDate* as audio
    2. If visible is **true**, an Alexa-enabled device is paired or connected with this device, and therefore achievement can be displayed on the screen with the student's chosen theme and any other engaging sound effects and/or music

Session

- generateQuestion():
    1. Queries the company database to generate a **Question** object
    2. Return this **Question** object to be manipulated by Session

Voice Commands

The list of available commands that the Student can use when inside the Alexa skill, and their functionalities.

"Alexa help"

1. Presents the user with a list of commands that are available for use within this app
2. Informs user on how to properly format and answer questions so that the app can parse and interpret the responses

"Alexa switch"

1. Prompts student for "Do you want to try something else or try something harder?"
    a. Alexa.prompt("Do you want to try something else or try something harder?");
2. Possible responses:
    a. *"Something else"* :
        i. Randomly select a concept from ("Addition", "Subtraction", "Multiplication", "Division", "Counting", "Word Problem") at same *skillLevel*

ii. Call *generateSession()* on selected concept with same *skillLevel* and return this new session
iii. Call *switchSession()* on the returned session

b. *"Something harder"* :
i. Call *generateSession(currentSession.concept, skillLevel + 1)* which generates a session with the same concept at one skill level higher
ii. Call *switchSession()* on the returned session

c. "Cancel":
i. Exits this "Alexa switch" command and resume Session

"Alexa I'm stuck"

1. Speaks the Question's hint and waits for a student's answer
   a. Alexa.speak(Question.hint)
   b. Alexa.wait();
2. If Student's answer is wrong after the hint is given, speak the Question's answer and advance to the next Question

Answers (examples and responses)

Addition **(Subtraction, Division, and Multiplication have similar implementations)**	*Alexa:* "What is 13 + 10"?	Question q = new Question(("What is 13 + 10", "23"), 1, "Try adding 10 + 10 first, then add 3"); Alexa.wait();

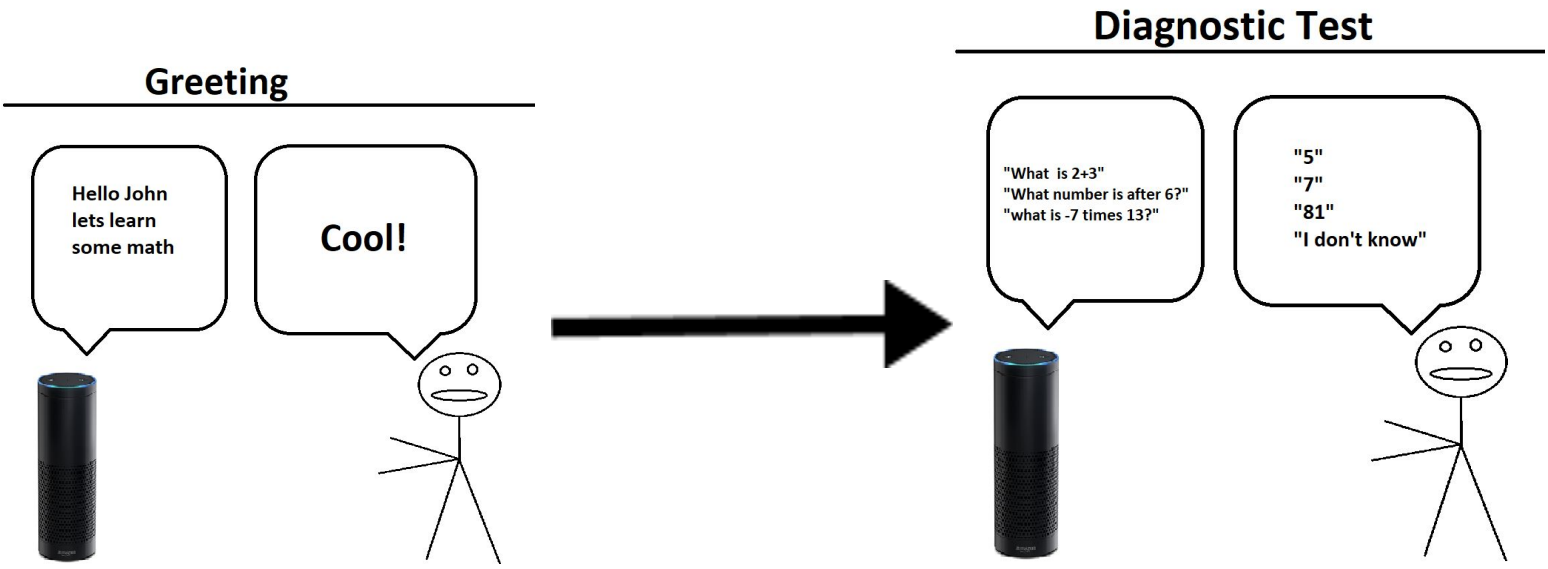
	Student: "23"	int answer = Alexa.parseInt("23") ;
		if (answer != q.problem[1]) {
		Alexa.speak("Not quite. Here's a hint!");
		Alexa.speak(q.hint);
		Alexa.wait();
		} else {
		Alexa.speak("That's right!");
		Alexa.nextQuestion();
		}

Counting	*Alexa:* “What number comes after -3?”
	
	Student: “-2”
		Question q = new Question(
		(“What number comes after -3”, “-2”),
		3,
		“Try adding 1 to -3, or subtracting 3 from 1”);
		Alexa.wait();
		
		int answer = Alexa.parseInt(“-2”);
		if (answer != q.problem[1]) {
		  Alexa.speak(“Not quite. Here’s a hint!”);
		  Alexa.speak(q.hint);
		  Alexa.wait();
		} else {
		  Alexa.speak(“That’s right!”);
		  Alexa.nextQuestion();
		}

Word Problem	*Alexa:* “If I have twenty-five boxes and each box has 20 pencils, how many pencils do I have in total?”
	
	Student: “500 pencils”
		Question q = new Question(
		(“If I have twenty-five boxes and each box has 20 pencils, how many pencils do I have in total?”, “500 pencils”),
		5,
		“Try multiplying 20 times 20 first, and then adding that to 20 times 5”);
		Alexa.wait();
		
		String answer = Alexa.parse(“500 pencils”);
		if (answer != q.problem[1]) {
		  Alexa.speak(“Not quite. Here’s a hint!”);
		  Alexa.speak(q.hint);
		  Alexa.wait();
		} else {
		  Alexa.speak(“That’s right!”);
		  Alexa.nextQuestion();
		}

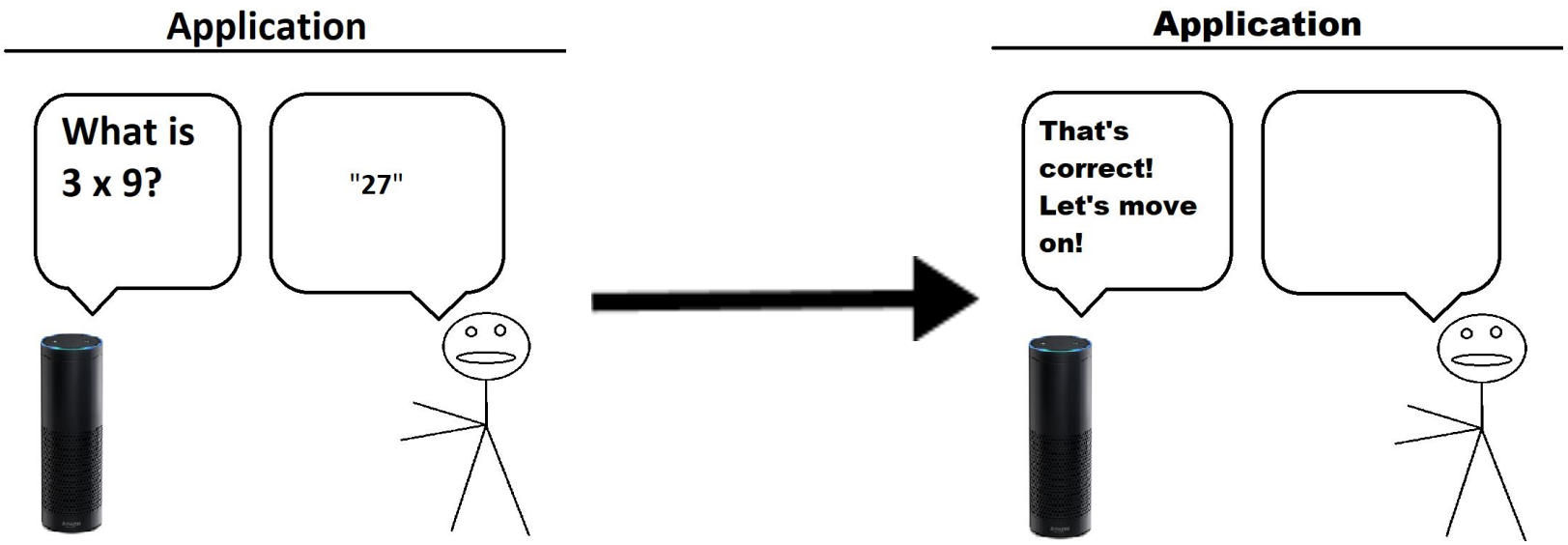
Interaction Design

Set-up

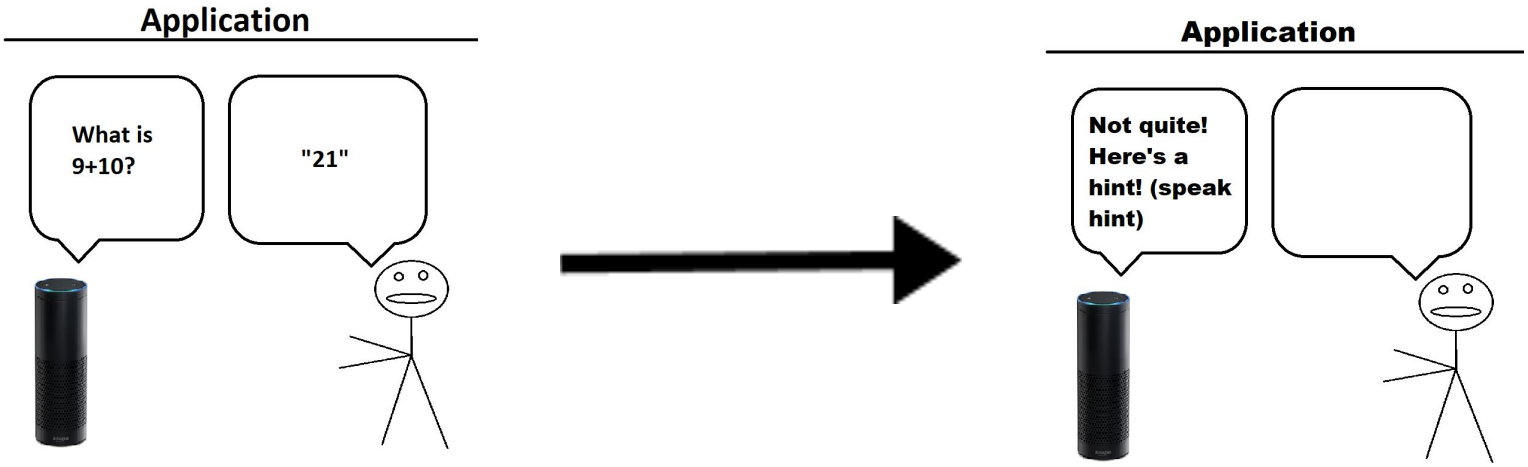


Main Application

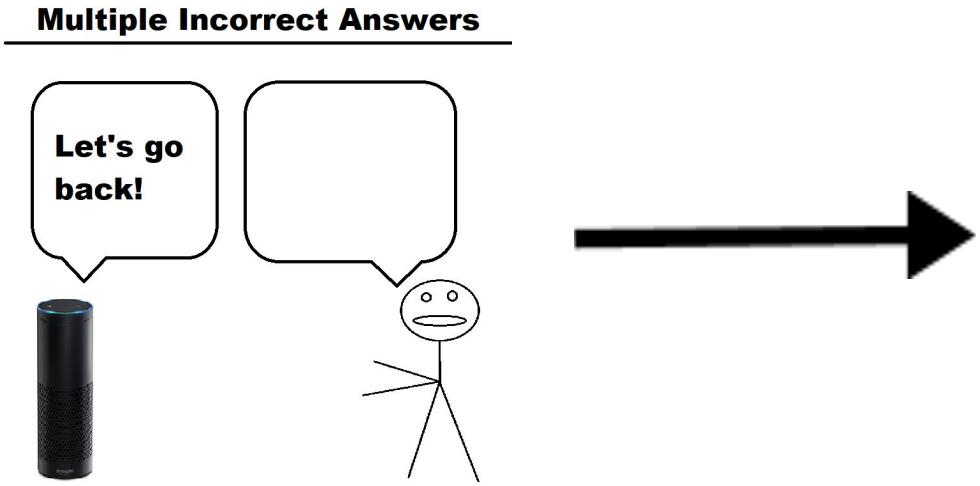
Correct answer



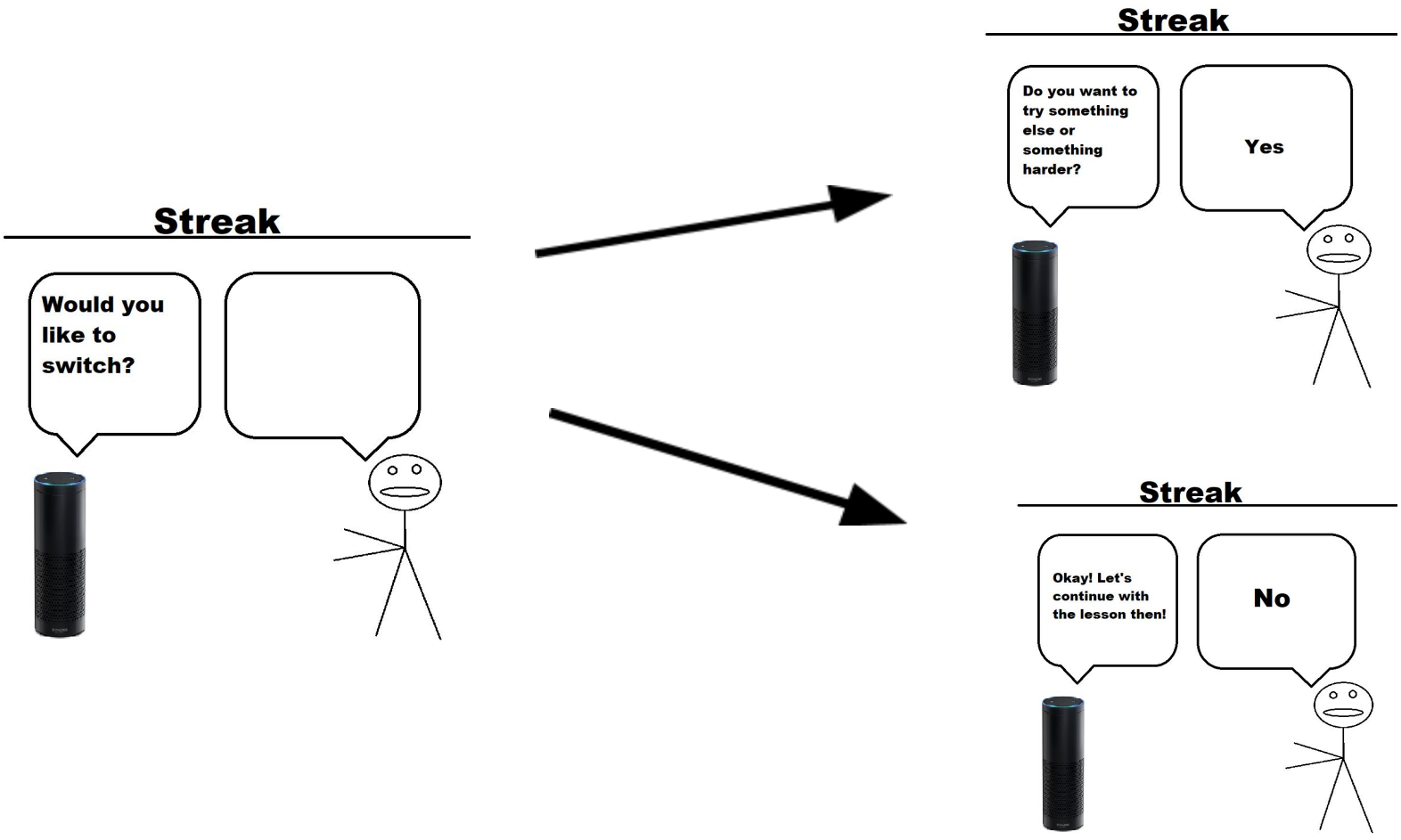
Incorrect Answer



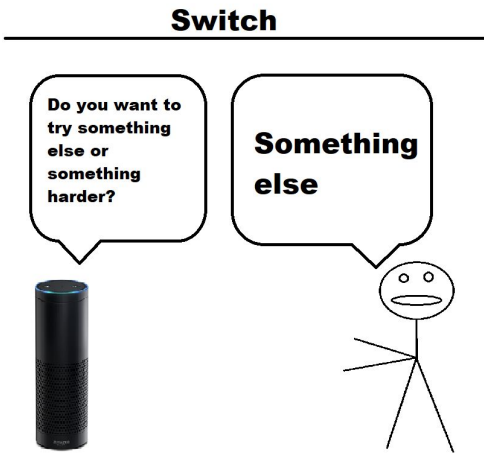
Multiple Incorrect Answers



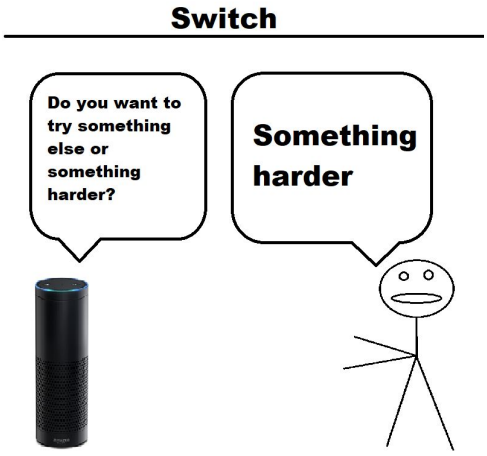
Multiple Correct Answers (streak)



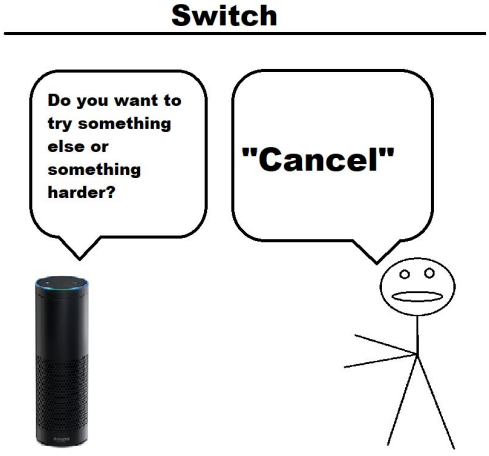
“Switch” command



DIFFERENT SESSION



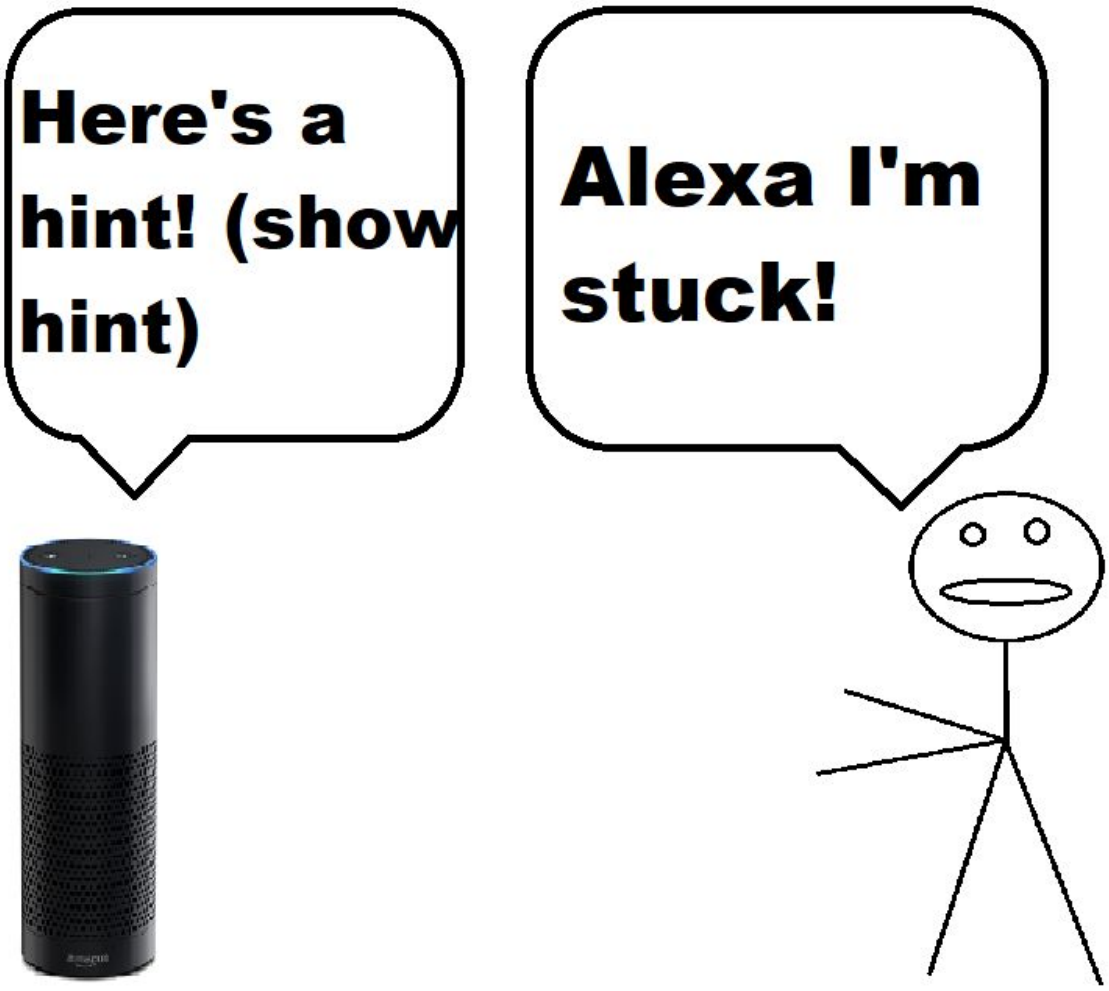
HARDER SESSION



CURRENT SESSION

“Alexa I’m stuck”

Application



Design Log

11/20 Group meeting: Nehal, Tam, Luke
- Mindmap
- Developing ideas
- Researching similar applications
- Researching math skills for students

11/20 Individual work: Ahaz
- Developing ideas
- Architectural Style

11/27 Group meeting: Nehal, Tam, Keanna, Ahaz
- Started Application design
- More ideas
- Mapping out step-by-step process

11/27 Group meeting: Nehal, Tam, Ahaz, Luke
- Worked more on application design and step-by-step process

11/30 Group meeting: Nehal, Tam, Keanna, Luke, Ahaz
- Talked further about how to make the application more engaging

12/3 Individual work: Ahaz
- Use case diagram

12/4 Group meeting: Luke, Nehal, Ahaz, Tam
- Listed entities and their attributes and methods
- Started UML Diagram
- Roleplay:
  - Ahaz: child
  - Luke: Alexa
  - Nehal: parent
  - Tam: teacher

12/5 Group meeting: Luke, Nehal, Ahaz, Tam
- Created Architecture Design graphic
- Finished UML Diagram
- Finished Interaction Design
- Added more details to clarify Application Design
- Completed Design Document

Other Artifacts