Classroom Oriented Voice Activated Learning Assistant

Design Document

3/30/17

CS495 Spring 2017
Christina Noe, Bria Lowe, James Hodge

Change History

Date	Draft	Author	Change
2/1/16	A	Christina Noe	First Draft
2/11/17	B	James Hodge	Edits to Introduction Section
2/21/17	C	Christina Noe	Project Description & Requirements
2/22/17	D	James Hodge	Added Use Case Diagram & High Level Class Diagram
2/26/17	E	Christina Noe	Updated document organization
2/28/17	F	Team	Added functionality priorities
2/28/17	G	James Hodge	Edits to High Level Class Diagram
3/1/17	H	Christina Noe	Revised system architecture, added more descriptions
3/1/17	I	Bria Lowe	Added activity diagrams
3/2/17	J	Christina Noe	Added description of non-functional requirements. Added potential risks list
3/2/17	K	Bria Lowe	Added the remaining activity diagrams and descriptions for each activity diagram
3/2/17	L	James Hodge	Finalizing, formatting.
3/24/17	M	James Hodge	Updated formatting to Design Document, added diagrams for Class and Database
3/24/17	N	James Hodge	Added descriptions for Class/Database diagrams.
3/26/17	O	Christina Noe	Added Voice Interaction Model
3/28/17	P	James Hodge	Added Page #s, Updated Use Case Diagram
3/29/17	Q	Bria Lowe	Added sequence diagrams

Table of Contents

Introduction

Motivation

Google and Amazon have been developing voice controlled smart speakers for home use. These smart speakers act as artificial personal assistants and can respond to a variety of commands. This project focuses on adapting this technology from a home setting to a classroom setting. We aim to give teachers the tools to automate repetitive tasks, and provide a hands free way of managing classroom records.

Goals

This project specifically deals in creating software that will accomplish two tasks. The first task is creating new actions for the voice controlled speaker that can be used in a classroom. The second task is to connect the software to Learning Management Systems to provide easy control over classroom records.

Key Definitions

- **Learning Management Systems (LMS)** - A digital administration and record keeping service for use in education.

- **Intelligent Personal Assistant (IPA)** - Software that can perform assistant tasks such as schedule management or question answering.

- **Amazon Echo Dot** - A smaller version of the Amazon Echo IPA.The Dot has a smaller speaker that the standard Echo, but it's less expensive, lighter, and comes with bluetooth capability.

- **Amazon Web Services (AWS)** - A suite of cloud-computing services offered by Amazon. Amazon suggests using AWS to host services for Echo Dot projects.

- **Alexa Skills Kit** - Offered by Amazon, a collection of APIs, documentation, and code samples for developing Alexa skills.

Project Description

System Overview

Our classroom assistant will consist of an Amazon Echo Dot device and a website hosted by Amazon Web Services. Using the Echo Dot device, voice commands can be issued that ask Alexa, the Echo Dot OS, to perform various classroom tasks. These tasks include taking student attendance and setting timed voice alerts for tests (see a detailed functionality table in the Requirements section). The accompanying website will act as a bridge between the Echo Dot and a learning management system. This LMS can be connected to an external online service or consist of an internal database that our service will host. The website will also provide classroom assistant functionality, as well as help documentation.

Hardware Requirements

- Intelligent Personal Assistant → Amazon Echo Dot
- A device to access the website

Software Requirements

- Learning Management System (LMS)
- A service to process Amazon Alexa commands → Amazon Web Service
- Web application for accessing and editing a LMS → Amazon Web Service

System Components

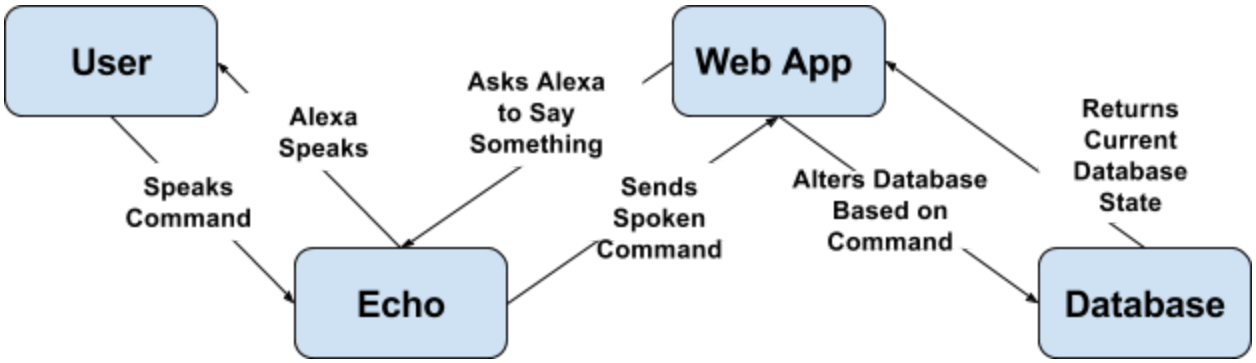
The voice activated classroom assistant will consist of the following components:

- **Echo Dot Device -** The interface through which users give Alexa spoken commands.

- **Web Application -** The interface that receives and processes Alexa commands, and through which users can display student attendance information. Users will be able to edit student information manually. The web application will also handle the business logic of accessing and editing the student database.

- **Database: Learning Management System or SQL -** The mechanism for storing student information. Users can link to an external online LMS or create an internal LMS.

Component Communication Diagram



The diagram above details the communication between all components of the classroom assistant system. First, the user speaks a command to the Echo Dot. The Echo Dot then sends this spoken command to the Amazon Web Service-hosted web application to be processed.

Once the command is processed, the web application determines what action should be taken with what parameters. The web application then accesses the backend database and applies the action to the database.

Once the action is applied, the database will return its current state to the web application. Based on the returned state of the database, the web application will then send a request for Alexa to speak the returned information back to the user.

Potential Risks

- Unauthorized student control when the teacher is absent
- The Echo Dot not correctly recognizing voice commands
- Accessing more than one class database for a single LMS account
- Conflicts when connecting to an LMS

Functional Requirements

Echo Dot-Initiated Functionality

Echo Dot-Initiated Functionality refers to functionality that is initiated by communication with the Echo Dot device. Echo Dot-Initiated Functionality can be split into three types: General, Database, and Testing.

Priority Key
Critical - Must be implemented in order for the system to be operable
High - Functionality key to meeting our functionality goals
Moderate - Will be implemented if time permits
Low - Implementation possibility is questionable

General Functionality
Functionality that is necessary, but separate from classroom assistant functionality.

#	Functionality	Priority
1	Start Alexa Skill	Critical
2	Close Alexa Skill	Critical

Database Functionality
Functionality for accessing and editing the student database.

#	Functionality	Priority
3	Take roll	High
4	Return roll at date	High
5	Return roll for particular student at date	High
6	Add student as present	High
7	Remove student as present	High
8	Give bonus points to student for current date	Moderate

Testing Functionality

Functionality that aids in student testing.

#	Functionality	Priority
9	Set timer to for a certain amount of time	High
10	Return time remaining	High
11	Stop timer	High
12	Pause timer	High

Webapp-Initiated Functionality

Webapp-Initiated Functionality refers to functionality that is initiated through the system's accompanying web application. Webapp-Initiated Functionality can be split into three types: Database, Testing, and Miscellaneous.

Priority Key

Critical - Must be implemented in order for the system to be operable
High - Functionality key to meeting our functionality goals
Moderate - Will be implemented if time permits
Low - Implementation possibility is questionable

Database Functionality

Functionality for accessing and editing the student database.

#	Functionality	Priority
13	Link user to exterior database	High
14	Setup interior database	Critical
15	Display student database	High
16	Take roll	High
17	Display called student names on screen	Moderate
18	Return roll at date	High
19	Return roll for particular student at date	High
20	Manually add student as present	High
21	Manually remove student as present	High
22	Give bonus points to student for current date	Moderate
23	Export student database as excel document	High

Testing Functionality

Functionality that aids in student testing.

#	Functionality	Priority
24	Set timer to for a certain amount of time	High
25	Display time remaining	High
26	Pause Timer	High
27	Stop timer	High

Misc. Functionality

“What-if” functionality that is dependent on time and implementability.

#	Functionality	Priority
28	Add student nickname	Low
29	Change student name phonetic pronunciation	Low
30	Nickname suggester	Very Low

Non-Functional Requirements

Notable non-functional requirements include time constraints on how long the system requires to complete roll call, the ability to add as many students to the student database as needed, having help documentation available, and concern over unauthorized student access of the system.

Priority Key
Critical - Must be implemented in order for the system to be operable
High - Functionality key to meeting our functionality goals
Moderate - Will be implemented if time permits
Low - Implementation possibility is questionable

#	Functionality	Priority
31	The system must take no longer than 3 minutes to complete a roll call of 25 students.	High
32	The user should be able to add as many students to the student database as required.	High
33	When the website is viewed on a projector, the displayed timer should be large enough that all students in a classroom can easily read it.	High
34	The website will include help documentation that details all voice commands that the system is able to respond to.	Moderate
35	When a timer is running (such as during a test), the Echo Dot should ignore typical Alexa search requests (Example: “Alexa, what is 2+2?”)	High
36	There must be some security mechanism such that a student cannot alter their attendance or bonus points while a teacher is absent.	High
37	Student attendance and bonus point information must be stored and searchable by day.	High
38	The system should be designed such that it isn't dependent on a particular LMS, so that adding functionality for an additional LMS is as simple as programming another LMS adapter.	High
39	When the system is linked to an external LMS, that LMS should accurately reflect any changes requested by the user by voice command (verifiable by checking the online version of the LMS).	Moderate

Diagrams

Web-App Projected Classes

Below are the classes that we anticipate will form the architecture of our project.

- **Application Driver -** The main class that drives the application. Will most likely contain instances of the other classes. Will most likely setup the Alexa Command Receiver to listen for commands from the Amazon Web Service.

- **Database Communication Manager -** The class that contains access and edit functions to the backend student database. This database will either be an internal SQL database or an external online Learning Management System.

- **Alexa Command Receiver -** The class that will handle interpreting Alexa commands received from the Amazon Web Service.

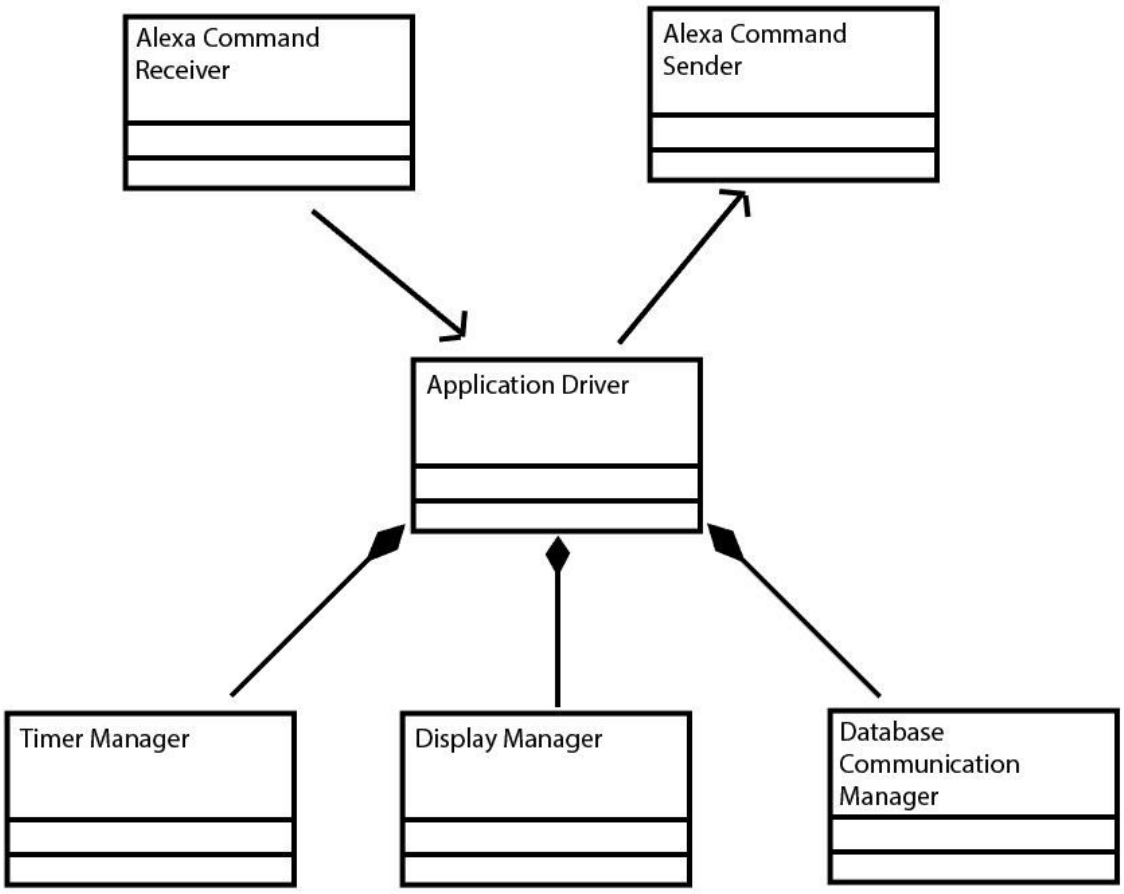
- **Alexa Command Sender -** The class that will handle sending Alexa speech requests back to the Echo Dot device.

- **Display Manager -** The class that will handle any display functionality, such as displaying received database information from the Database Communication Manager.

- **Timer Manager -** The class that will handle the control of quiz/test timers.

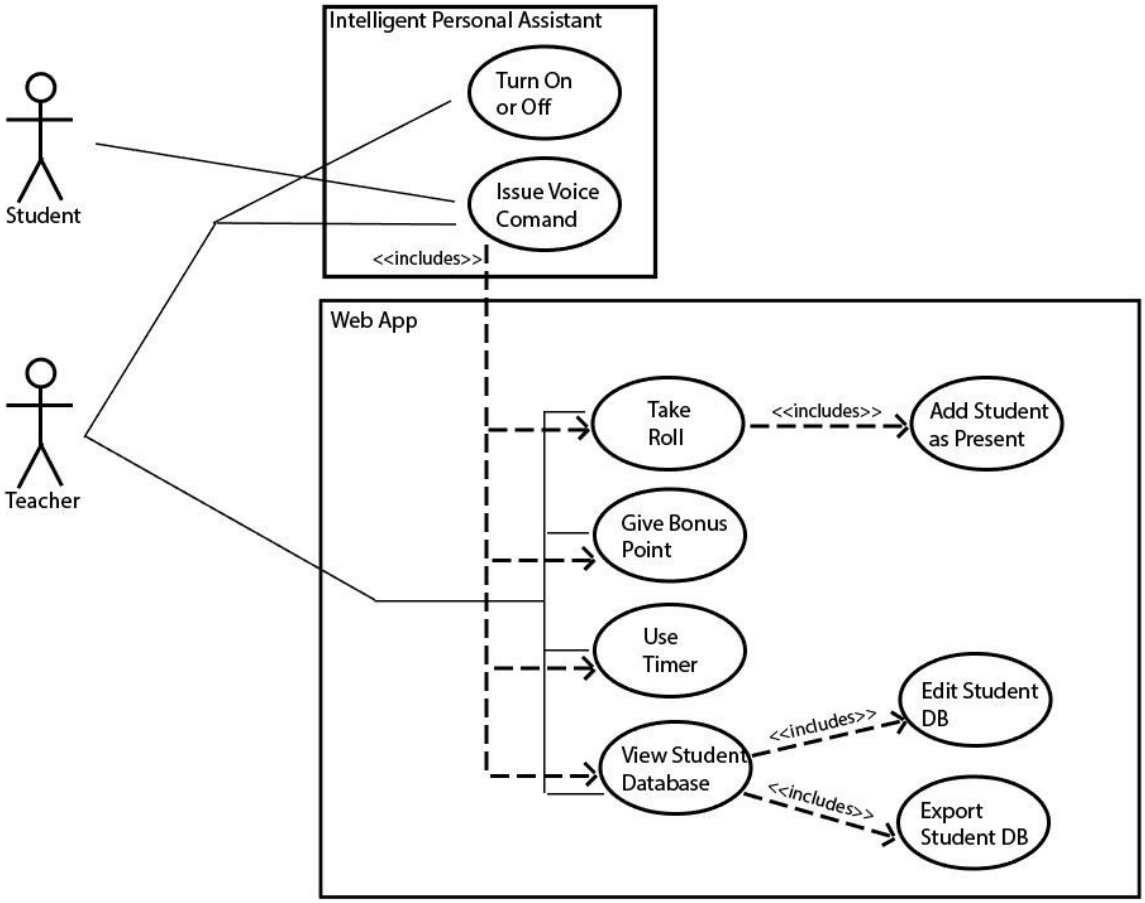
High Level Class Diagram

Below is a UML representation of the class architecture described above. Just as described, the Time Manager class, the Display Manager class, and the Database Communication Manager will most likely be instantiated within the Application Driver Class. The Alexa Command Receiver and the Alexa Command Sender will communicate with the Application Driver as needed.



Use Case Diagram

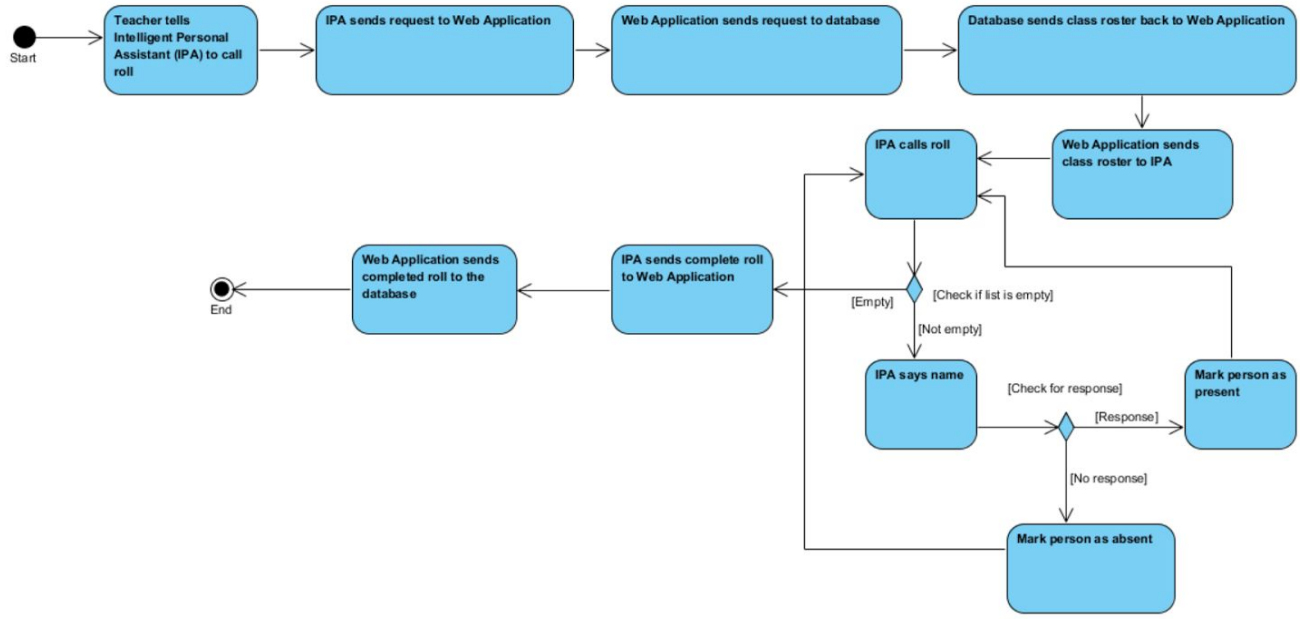
This use case diagram shows how both students and Teachers use the Intelligent Personal Assistant as a way to issue commands to the web app. It also shows how teachers can manually interact with the web app itself.



Activity Diagrams

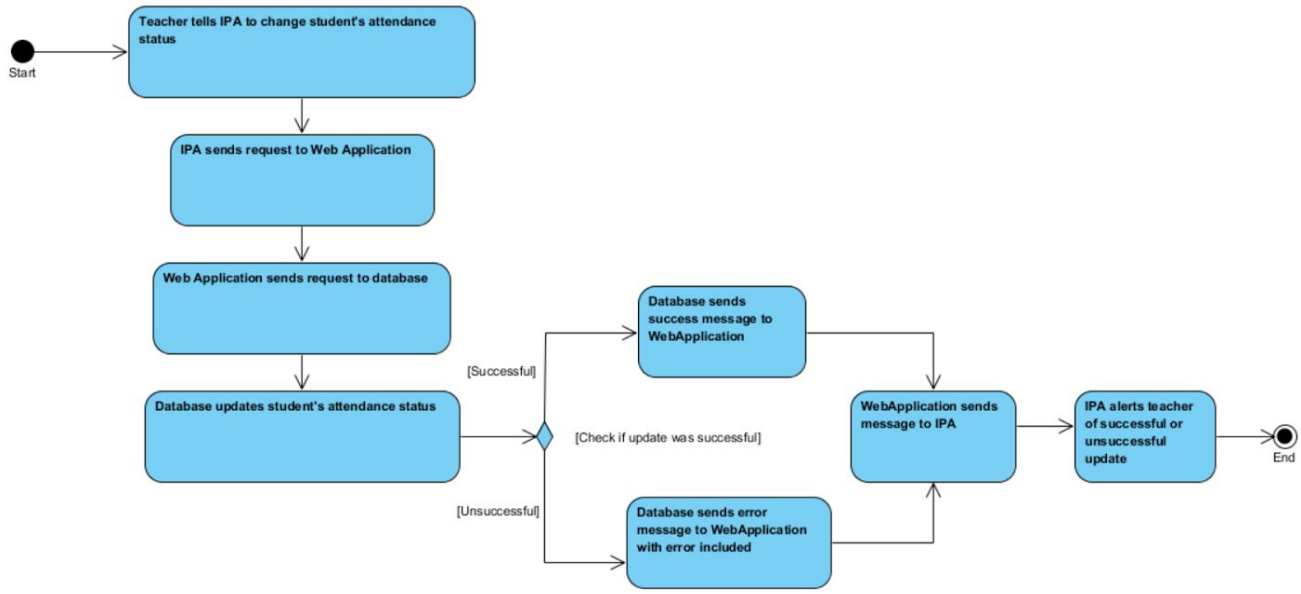
Call Roll

The following diagram walks through the process of calling roll. The teacher will send a voice command to the IPA indicating that he or she would like to call roll. The command will be sent to the Web Application and then the database. The database will return the class roster to the IPA through the Web Application. The IPA will then begin calling rolls, pausing for a response, and marking students absent or present depending on a student's response or lack thereof respectively. Once all the names are called, the Web Application will take the completed roll and send it back to the database where it will be recorded.



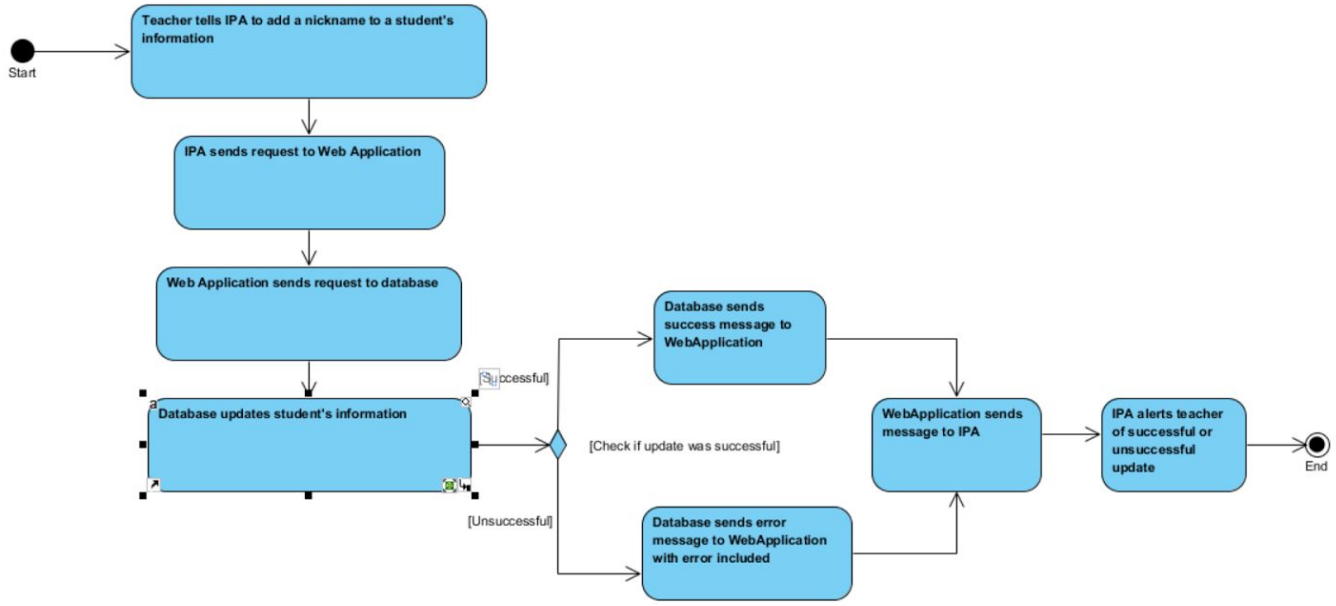
Update student's attendance status

The following diagram walks through the process of updating a student's attendance status for the day. The teacher will request that the IPA change a particular student's attendance status. The IPA will send the request to the database through the Web Application. The database will then attempt to update the student's attendance status for today's date. The database will then alert the IPA through the Web Application of the successful or unsuccessful update and this message will be relayed to the teacher through the IPA.



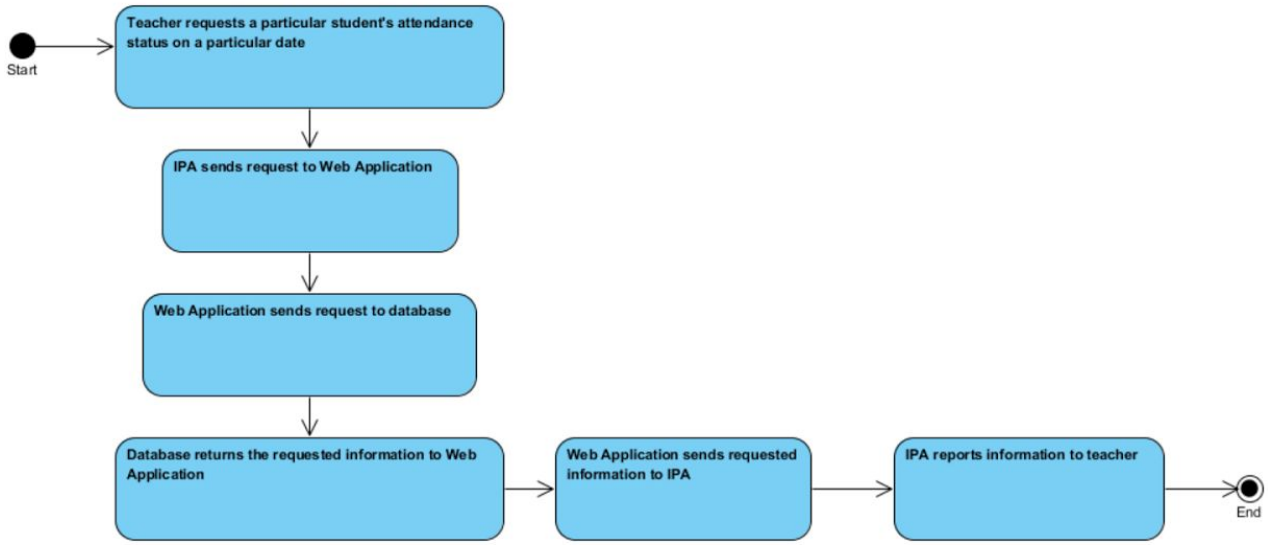
Add a nickname for a student

The following diagram shows the process of adding a nickname to a student's record. The teacher will request that the IPA add a nickname to a particular student's information. The IPA will send the request to the database through the Web Application. The database will then attempt to update the student's information. The database will then alert the IPA through the Web Application of the successful or unsuccessful update and this message will be relayed to the teacher through the IPA.



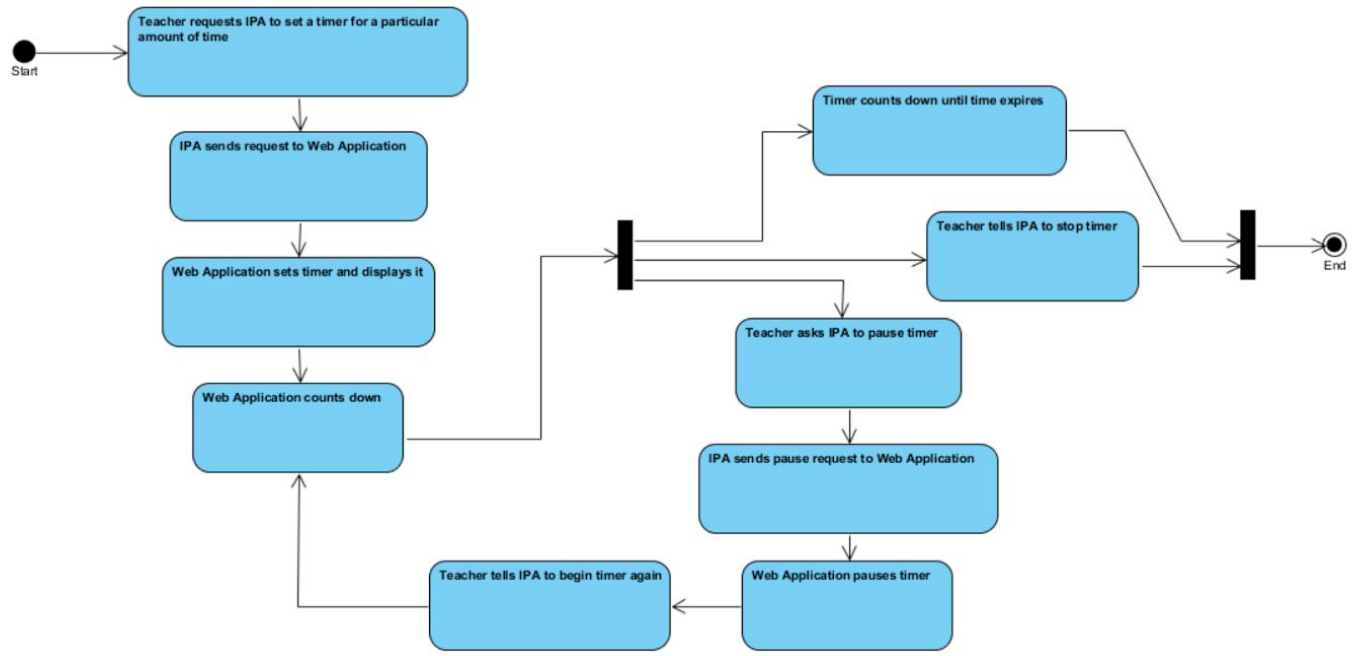
Request student's attendance status on a particular date

The following diagram shows the process of a teacher requesting a particular student's attendance status on a particular date. The teacher will send the voice command to the IPA. The IPA will then send the command to the database through the Web Application. The database will return the requested information to the IPA through the Web Application, and the information will be reported to the teacher by the IPA.



Set quiz/test timer

The following diagram shows the quiz or test timer activity. The teacher sends a request to the IPA to start a timer for a particular amount of time. The IPA sends the request to the Web Application and the Web Application creates a timer. The Web Application displays the timer on screen and waits for the voice command to start counting down. The teacher tells the IPA to start the timer and the IPA sends this command to the Web Application. The Web Application continues counting down until time expires unless the teacher requests the timer to be paused or stopped.



Design Diagrams

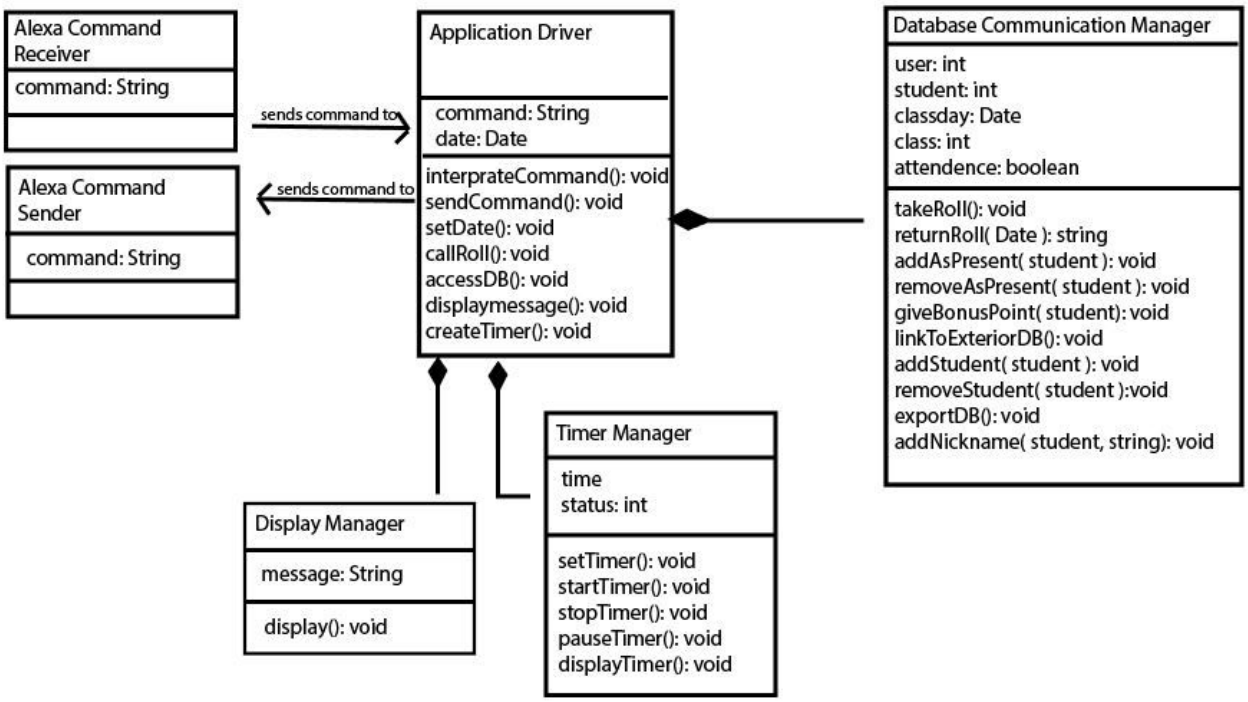
Class Diagram

The following diagram shows a more detailed look at the classes involved in the webapp. On the left are functions for receiving and sending commands to the connected Amazon voice assistance device. These connect to the central Application Driver which is used to interpret commands and run tasks involving three others classes.

The Database Communication Manager connects to either the built in database or to an external learning management system and can get and set certain values given the command received.

The Timer Manager controls the quiz/test timer and can desolate the timer information on screen.

The Display Manager is used to display other various messages onto the webapp.

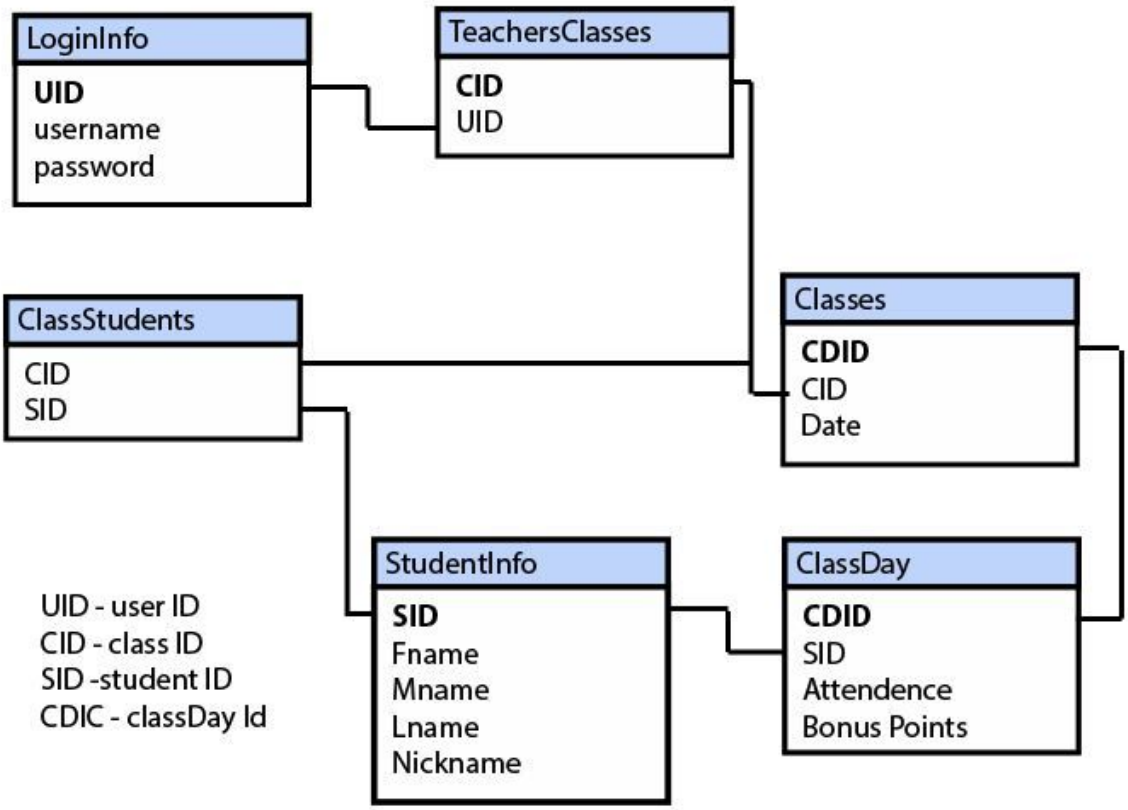


Database Diagram

Below is the structure for the built in database, when connecting to other databases the different structures will need to be taken account. In the bottom left of the diagram there is a key for the different acronyms used in the database. The database is designed to provide easy access to student information, class information, and the record of attendance for each day of each class.

The tables used in the database are:

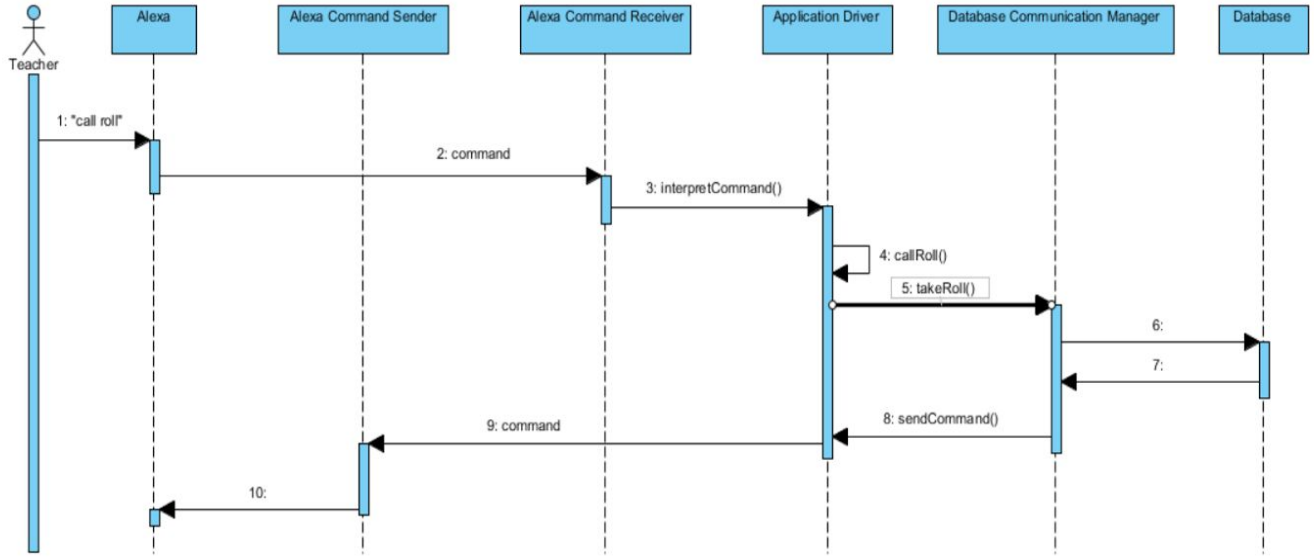
- LoginInfo
- TeachersClasses
- ClassStudents
- Classes
- StudentInfo
- ClassDay



Sequence Diagrams

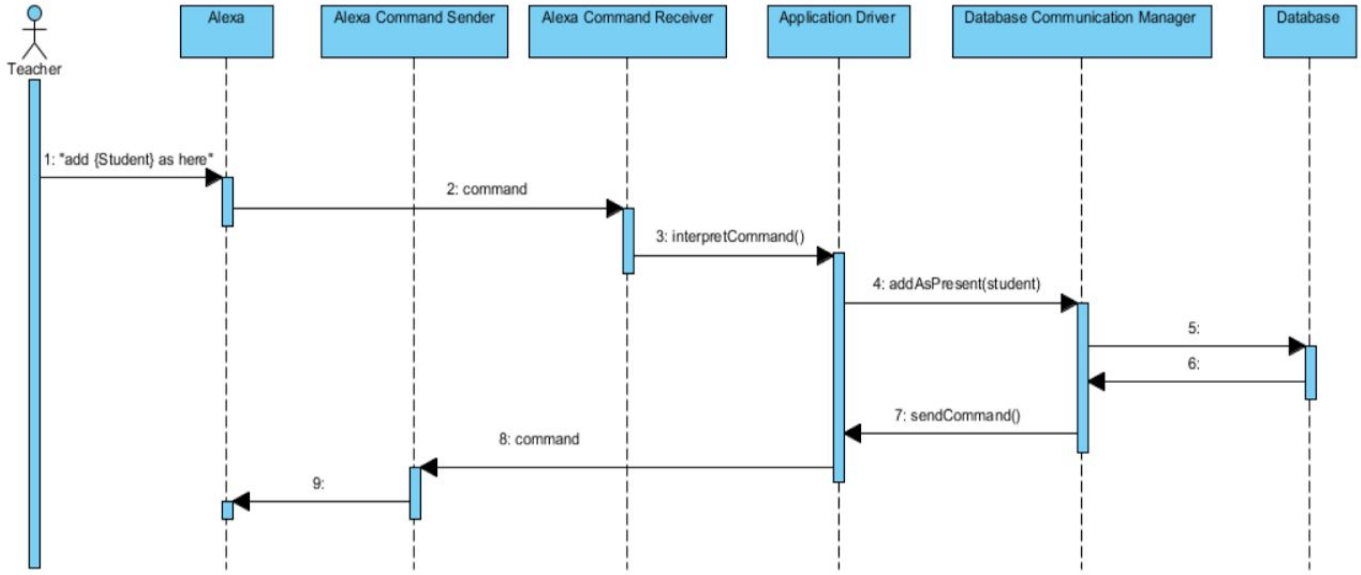
Call Roll

This following sequence diagram shows the process of calling roll. The teacher sends the command “call roll” to Alexa. That command gets passed to the Alexa Command Receiver and then the Application Driver where it is interpreted as call roll. The call roll method is invoked and calls the take roll method for the Database Communication Manager. The Database Communication Manager collects the necessary information from the database and sends it back to the Application Driver. The Application Driver sends the commands to the Alexa Command Sender, which sends the commands to Alexa. Alexa carries out the commands and calls roll.



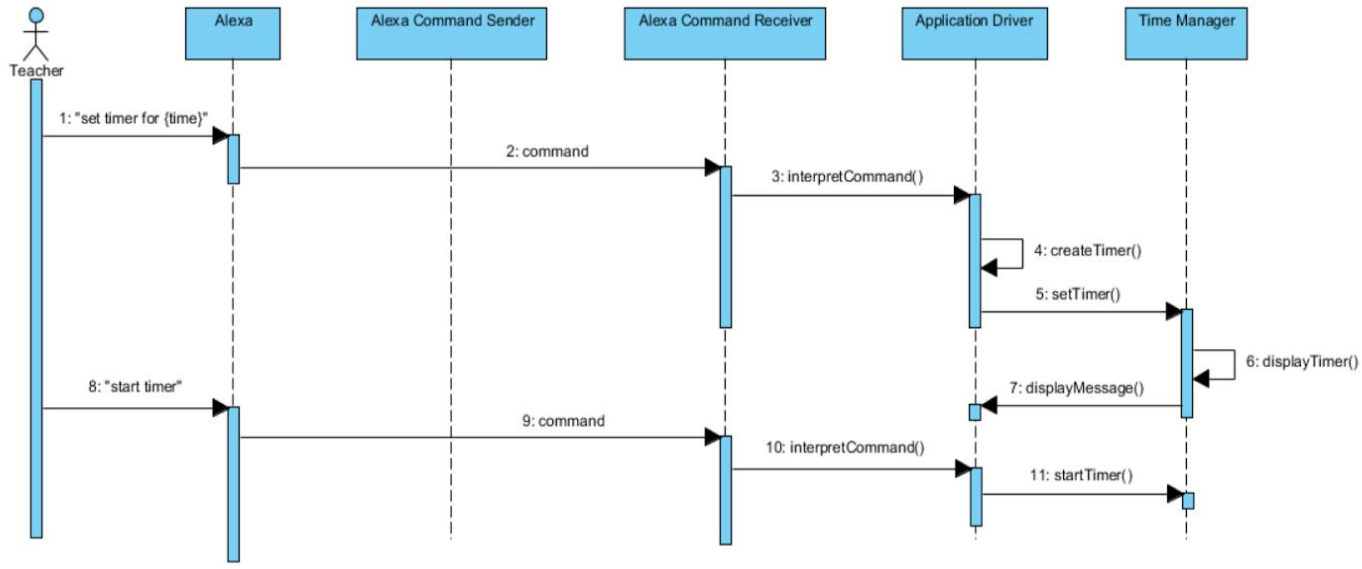
Update Student's Attendance Status

The following sequence diagram shows the process of changing a student's attendance status to present for the current date. The teacher sends the command "add {student} as here" to Alexa, where {student} is the particular student's first and last name. This command gets passed to the Alexa Command Receiver and then the Application Driver where it is interpreted as a change to a student's attendance status in the database. The addAsPresent method for the Database Communication Manager is invoked and the proper changes are sent to the database. If the change is successful, a success message is passed from the Database all the way to Alexa as shown in steps 6, 7, 8, and 9. Otherwise a failure message is passed from the Database to Alexa as shown in steps 6, 7, 8, and 9.



Set Timer

The following sequence diagram depicts a teacher setting a timer for a particular amount of time and starting that timer. The teacher sends the command “set timer for {time}” to Alexa, where {time} is a particular amount of time such as 10 minutes. This command is passed to the Alexa Command Receiver and then the Application Driver where it is interpreted as a command to create a timer. The Application Driver creates a timer instance and sets the timer up by calling the setTimer method for the Time Manager. The Time Manager displays the timer. The teacher then sends the command “start timer” to Alexa. This command is passed to the Application Driver through the Alexa Command Receiver where it is interpreted as a command to start the timer. The message to start the timer is sent to the Time Manager and the Time Manager starts the timer.



Voice Interaction Model

Part of creating an Alexa skill is creating a voice interaction model for that skill. A Voice Interaction Model is a model what commands users must speak to interact with your Alexa skill. There are two main components of a voice interaction model, an Intent Schema and Sample utterances.

Intent Schema

An Intent is an action that a user wishes an Alexa skill to execute. Whenever a user speaks a command to an Alexa skill, Alexa maps their spoken command to one of the skills pre-specified intent. Alexa then sends a JSON object containing the intent and the intent's parameters to the server program to be processed. An Intent Schema is a JSON object that defines the set of valid intents for an Alexa skill. The Intent Schema also defines the set of parameters or "slots" that a user can supply with an intent.

Below are the intents for our Covala skill.

Roll Call Intents

```
{
    "intents": [
        {
        "intent": "callRoll",
        "slots": [
            {
                "name": "date",
                "type": "AMAZON.DATE"
            }
        ]
    },
        {
        "intent": "callRollForStudent",
        "slots": [
            {
                "name": "date",
                "type": "AMAZON.DATE"
            },
            {
                "name":"studentFirstName",
                "type":"AMAZON.US_FIRST_NAME"
            },
            {
                "name":"studentLastName",
                "type":"AMAZON.US_LAST_NAME"
            }
        ]
    },
```

Attendance Intents

```
{
    "intent": "addStudentAttendance",
    "slots": [
        {
            "name":"studentFirstName",
            "type":"AMAZON.US_FIRST_NAME"
        },
        {
            "name":"studentLastName",
            "type":"AMAZON.US_LAST_NAME"
        },
        {
            "name":"status",
            "type":"ATTENDANCE_STATUS"
        },
        {
            "name":"date",
            "type": "AMAZON.DATE"
        }
    ]
},
{
    "intent": "removeStudentAttendance",
    "slots": [
        {
            "name":"studentFirstName",
            "type":"AMAZON.US_FIRST_NAME"
        },
        {
            "name":"studentLastName",
            "type":"AMAZON.US_LAST_NAME"
        },
        {
            "name":"status",
            "type":"ATTENDANCE_STATUS"
        },
        {
            "name":"date",
            "type": "AMAZON.DATE"
        }
    ]
},
```

Timer Intents

```
        "intent": "setTimer",
        "slots": [
            {
                "name":"time",
                "type":"AMAZON.TIME"
            }
        ]
    },
    {
        "intent": "stopTimer"
    },
    {
        "intent": "pauseTimer"
    },
    {
        "intent": "resumeTimer"
    }
  ]
}
```

Database Intents

```
    {
        "intent": "addStudentToDatabase",
        "slots": [
            {
                "name":"studentFirstName",
                "type":"AMAZON.US_FIRST_NAME"
            },
            {
                "name":"studentLastName",
                "type":"AMAZON.US_LAST_NAME"
            }
        ]
    },

    {
        "intent": "removeStudentFromDatabase",
        "slots": [
            {
                "name":"studentFirstName",
                "type":"AMAZON.US_FIRST_NAME"
            },
            {
                "name":"studentLastName",
                "type":"AMAZON.US_LAST_NAME"
            }
        ]
    },
```

Sample Utterances

Sample Utterances must also be declared for an Alexa skill. Sample utterances are the set of possible ways that a user can invoke an intent for a skill. An intent can have multiple sample utterances. When declaring sample utterances, developers also indicate in brackets which words in the utterances represent slot parameters.

Below are the sample utterances for the Covala skill. There is at least one utterance for each intent. Some intents have an additional utterance for cases where an intent is being called relative to a certain date.

callRoll call roll
callRoll call past roll for {date}
callRollForStudent call roll for {studentFirstName} {studentLastName}
callRollForStudent call roll for {studentFirstName} on {date}

addStudentAttendance add {studentFirstName} {studentLastName} as here
removeStudentAttendance remove {studentFirstName} {studentLastName} as here
addStudentAttendance add {studentFirstName} {studentLastName} as here on {date}
removeStudentAttendance remove {studentFirstName} {studentLastName} as here {date}

addStudentToDatabase add {studentFirstName} {studentLastName} to database
removeStudentFromDatabase remove {studentFirstName} {studentLastName} from database

giveStudentBonusPoints give {studentFirstName} {bonusPoints} bonus points
removeStudentBonusPoints remove {bonusPoints} for {studentFirstName} {studentLastName}

setTimer set timer for {time}
stopTimer stop timer
pauseTimer pause timer
resumeTimer resume timer