Systems Modeling

Project Name

Applied Language Research

Prepared by

Applied Language Team

Date

18 October 2020

Overview

System models are an important step in the creation of software. They provide a layout of how the software should be structured, which can be referred back to at later points of development as a guiding tool. They also help developers identify users' requirements and how those requirements work into the software's mechanisms. Creating a solid model before beginning development gives a development team easy steps to follow in what may seem like an otherwise highly complicated, intricate system of interconnected software and hardware. An additional benefit is that conflicts in the system may be spotted in the model before beginning development, allowing time to be saved by fixing those conflicts before development begins.

Different models represent different aspects of a system, so the team chose to include three different types of models to represent the project software. The first model presented is a swimlane diagram, which is a type of flowchart that distinguishes what each user is capable of doing within the system. The second model is a data flow diagram that shows how data will flow through our software. This includes the processes involved in moving data around, storing it, and fetching it for users. The third model is an entity-relationship diagram that shows the relationships between the tables of our database.

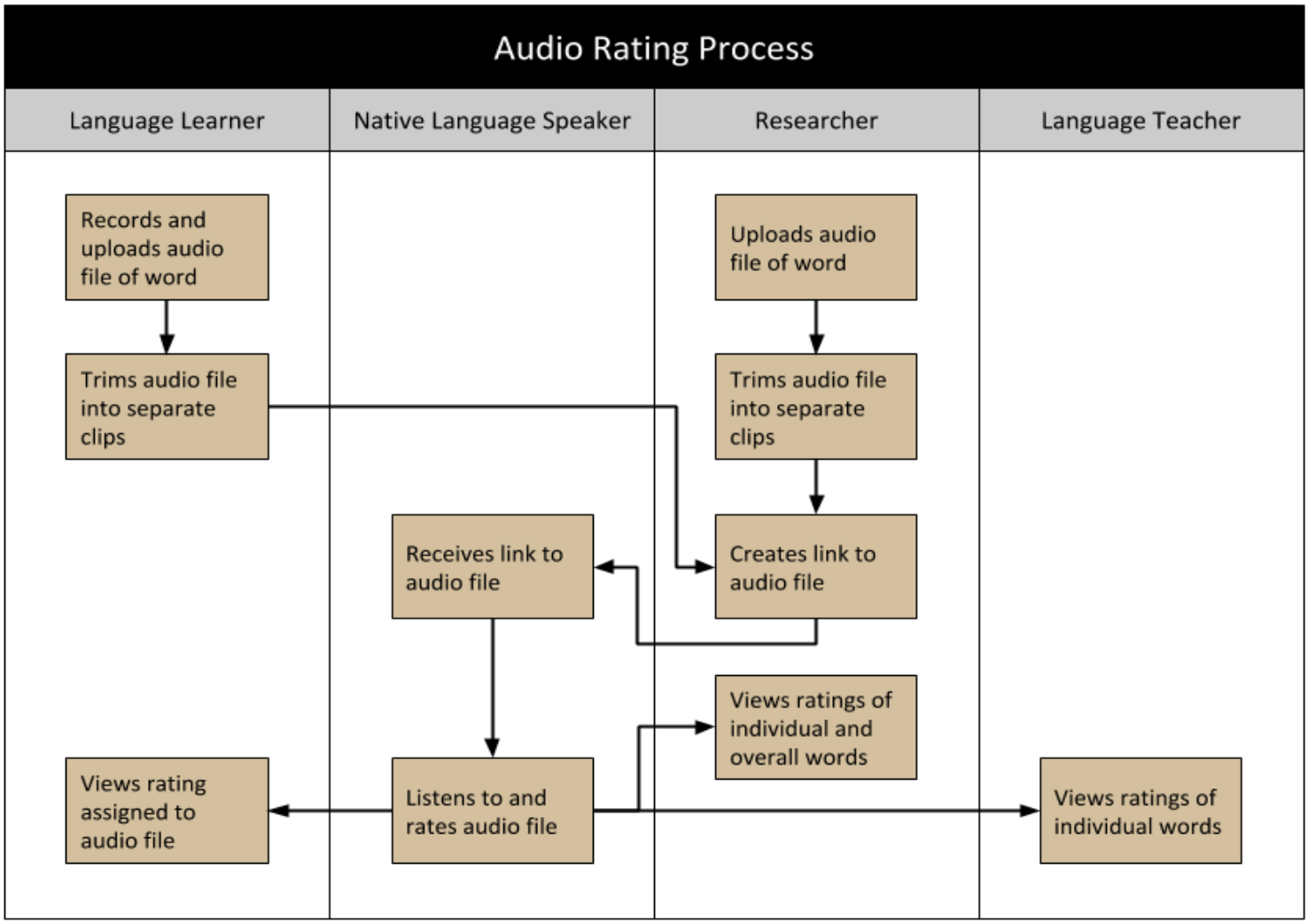
1. Model #1 – Swimlane Diagram

A swimlane diagram illustrates how different users interact with the system. Each distinct user group is given a column. In the corresponding columns, user actions are put into rectangular boxes. Arrows point from one action box to another action box (or multiple boxes) to demonstrate the order in which actions take place.

This swimlane diagram represents the process of uploading and rating an audio file. This model was chosen because it is the most important process of the system and is necessary to implement to meet the qualification of a minimum viable product.

The audio rating process can begin with either of two types of users: a researcher or language learner uploads an audio file containing a spoken word or words in a particular language. The uploader can then trim this audio file into smaller individual audio files. The researcher will then email a Native language speaker with a link to rate that audio file. The Native language speaker accesses the audio file through the link, which doesn’t require a login, and rates it. If the audio file is uploaded by a researcher, that rating can then be viewed by all researchers. Suppose the audio file was uploaded by a language learner. In that case, its ratings can be viewed by the language learner, the language teacher to which the language learner is assigned (if applicable), and all researchers.

Completing the swimlane model gave the team a good overview of the audio uploading process. This model confirmed some of the system requirements in terms of user functionality. No missing functionality was identified in the completion of this model.



2. Model #2 – Data Flow Diagram

A data flow diagram represents how data flows through a system, where the data is produced in a system, and where it is consumed throughout a system. This model is a good representation of our system because we have data being produced from language learners in the form of audio files with text descriptions, language teachers in the form of ratings and class assignments, and lastly, Native language speakers in the form of comments, ratings, and audio. The researchers and language teachers consume data in the form of audio ratings and audio files. Native language speakers consume language learner audio. Language learners consume the data that they produce as well as comments and ratings they receive. With the data flow diagram, we can show data flows within the system, from it being gathered to it being stored and consumed.

This data flow diagram depicts the flow of data between users and storage. Language learners can produce data by recording audio uploaded to the *user big audio* data store and then trimmed by the system and stored in *user trimmed audio*. Language learners expend data by viewing their submissions from *user big audio* and their trimmed audio timestamps

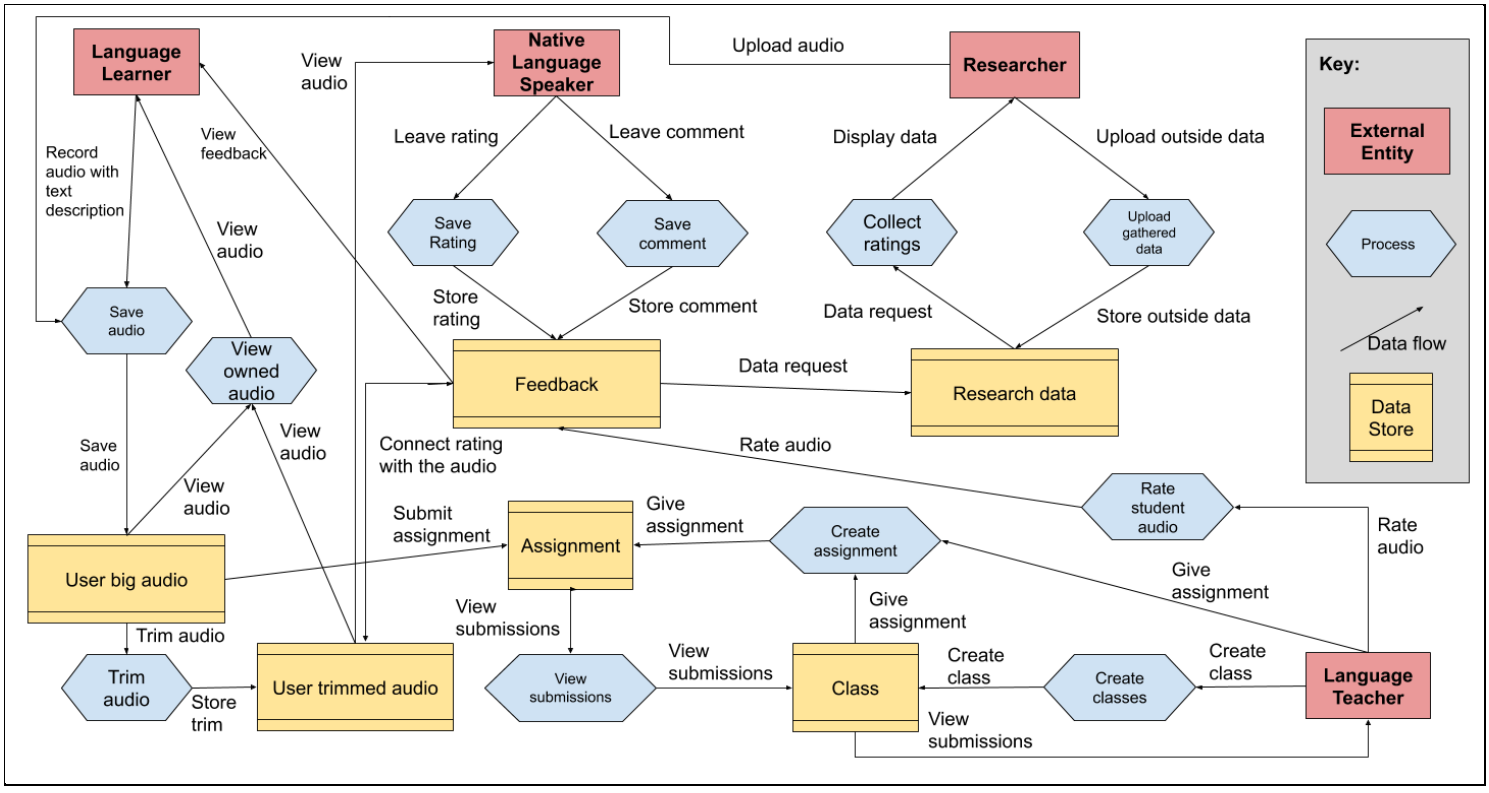
from *user trimmed audio.* Any feedback they have received from other users is taken from the *feedback* data store.

Native language speakers expend data from the trimmed audio in the *user trimmed audio* data store to then produce ratings and comments that get stored in the *feedback* data store.

Researchers can produce data in the form of past Excel files stored in the *research data* data store and in the form of audio files stored in the *user big audio* data store. Researchers can utilize all feedback data from the *feedback* data store.

Language teachers produce data by creating classes that are stored in the *class* data store. The language teacher can then generate assignment data to be stored in the *assignment* data store connected to the *user big audio* data store to enable the language teacher to view submitted audio files. When the language teacher examines submitted audio, they rate that audio which is then stored in the *feedback* data store.

From this model, we have realized that the ALR software's existing structure is a tangled and convoluted set of systems that needs some revision. While making this model, we have discovered that the ALR database structure could also have many tables condensed into one table to reduce redundancy and clutter. One example of this would be to aggregate all feedback (*alr_comment*, *alr_review*) tables into a single table. The naming scheme of existing tables could also be improved to be more readable. This model also has revealed some overlap in the required functionality of different user groups, e.g., language learners and researchers needing to upload audio.



3. Model #3 – Entity Relationship Diagram

An entity-relationship (ER) diagram shows entity types, their attributes, and their interrelationships. This type of diagram can be used to represent the structure of the data in a system and serve as a design guide for implementing the database. The ER diagram was chosen to help represent our system because data storage is an integral aspect of our system. A carefully planned database will ensure that important information will not be left out.

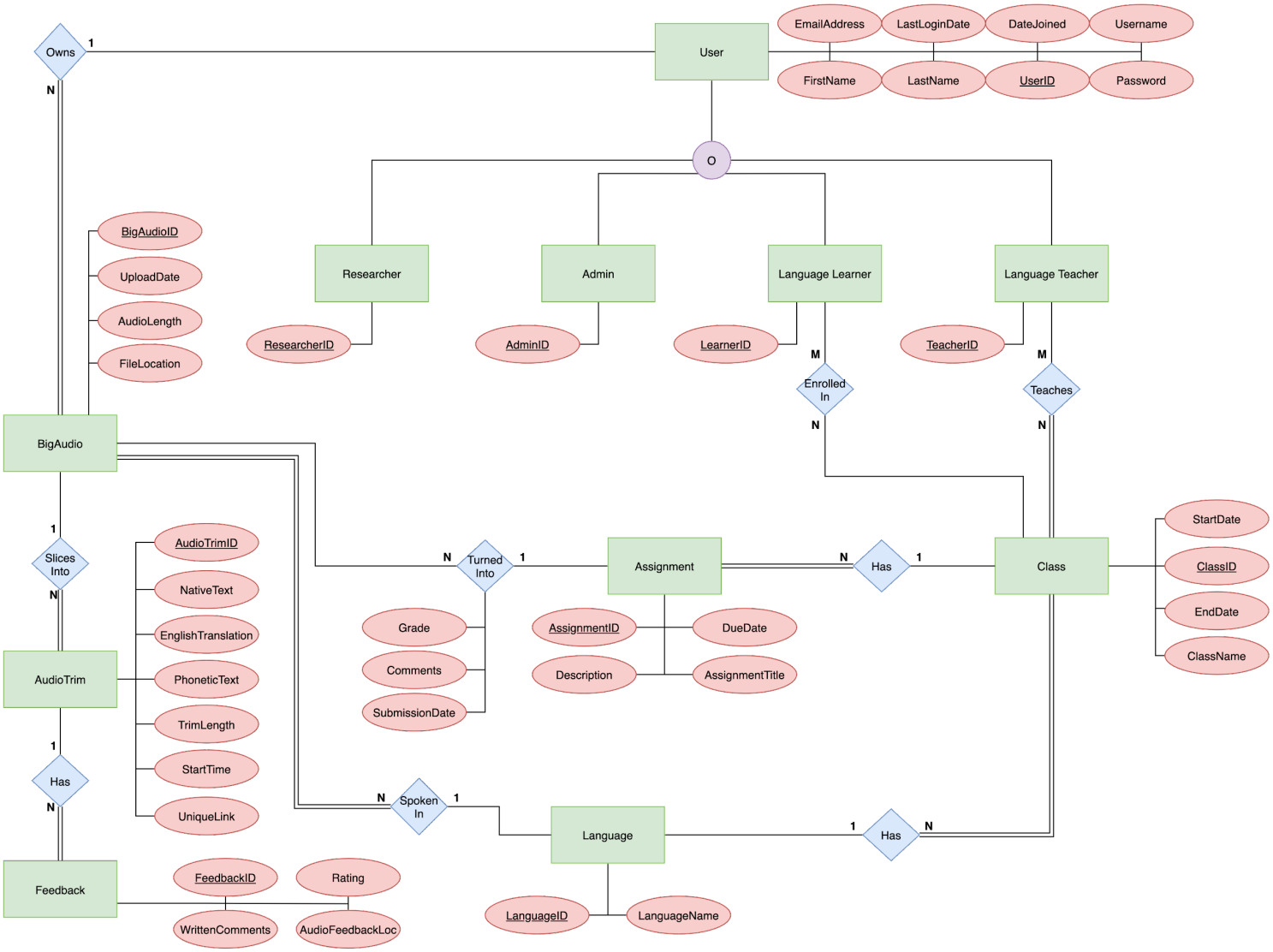
In the diagram, all user groups inherit properties of the user entity: administrator, researcher, language learner, and language teacher. These inherited properties include email address, username, password, first name, last name, user ID, last login date, and date joined. Further, each specific user type has a unique identifying attribute corresponding to their user group, which can be used for user authorization during login on the website. The Native language speaker user group is not included because this type of User will not have persistent data associated with them stored in the database.

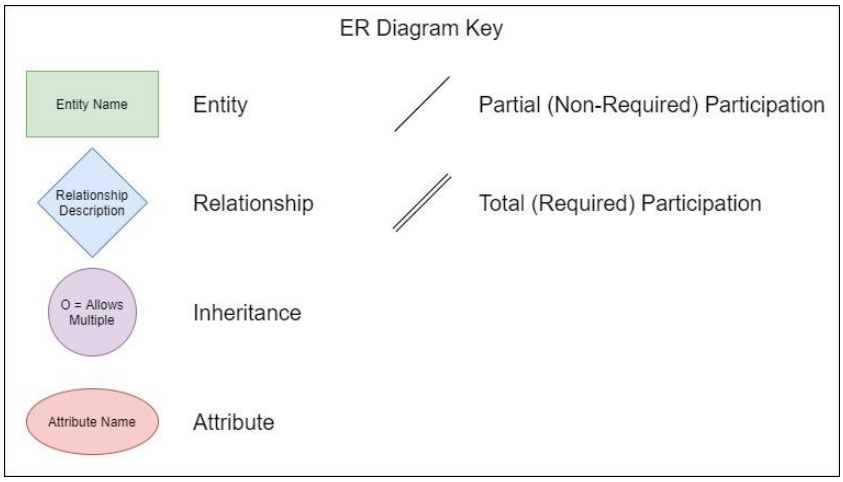
The language teacher and language learner entities can belong to a class entity, though not every learner needs to be a student. One language teacher and one or more student learners make up a class. This is representative of a real-world classroom setting. A class also has a single language and can have multiple assignments for its student learners. A language is its separate entity, allowing numerous languages to be used in the system.

When a user uploads a new audio file, the database creates a new BigAudio entity in the database attached to the user who made it and the language spoken in the audio. The website also slices that audio file based on the individual words' location and creates an audio trim in the database for each slice, which all associate back to the original BigAudio entity. In the case of a learner submission, the audio that is uploaded by the user can be associated with a class assignment, where it can receive a grade and comments from the teacher.

Audio trims are stored by their timestamp in the full audio file and the audio trim's length. Audio trims can receive feedback from multiple Native speakers, including written comments, a rating between 1-5, and audio feedback.

In the end, creating this model gave us a good insight into how we want to structure the database. Seeing an overview of the database structure allows us to make better tables because we can see the relationships much clearer. It has also shown us some characteristics may have been initially assigned to the wrong entity, particularly in the BigAudio and AudioTrim entities. Without this model, it would be much harder to create the database because we would lose track of where things are located due to all of the additional information.


Owns
1
N
User
EmailAddress
LastLoginDate
DateJoined
Username
FirstName
LastName
UserID
Password
O
Researcher
Admin
Language Learner
Language Teacher
ResearcherID
AdminID
LearnerID
TeacherID
M
Enrolled In
N
M
Teaches
N
BigAudio
BigAudioID
UploadDate
AudioLength
FileLocation
1
Slices Into
N
AudioTrim
AudioTrimID
NativeText
EnglishTranslation
PhoneticText
TrimLength
StartTime
UniqueLink
1
Has
N
Feedback
FeedbackID
Rating
WrittenComments
AudioFeedbackLoc
N
Turned Into
1
Assignment
Grade
Comments
SubmissionDate
AssignmentID
DueDate
Description
AssignmentTitle
N
Has
1
Class
StartDate
ClassID
EndDate
ClassName
N
Spoken In
1
Language
LanguageID
LanguageName
1
Has
N


ER Diagram Key
Entity Name
Entity
Partial (Non-Required) Participation
Relationship Description
Relationship
Total (Required) Participation
O = Allows Multiple
Inheritance
Attribute Name
Attribute

4. Execution and Acknowledgement

The team members hereby indicate by their signatures below that they have read and agreed with the specifications of this document.

18 OCT 2020
Team Member / Date

18 OCT 2020
Team Member / Date

18 OCT 2020
Team Member / Date

18 OCT 2020
Team Member / Date

11/12/20
Client Name / Date