UPPSALA
UNIVERSITET

Självständigt arbete i informationsteknologi
25 maj 2023

A System for Creating, Sharing and Listening to Interactive Stories

Krister Emrén
Nicke Löfwenberg
Alexander Sellström

Civilingenjörsprogrammet i informationsteknologi

Master Programme in Computer and Information Engineering

UPPSALA
UNIVERSITET

Institutionen för informationsteknologi

Besöksadress:
ITC, Polacksbacken
Lägerhyddsvägen 2

Postadress:
Box 337
751 05 Uppsala

Hemsida:
https://www.it.uu.se

Abstract

A System for Creating, Sharing and Listening to Interactive Stories

Krister Emrén
Nicke Löfwenberg
Alexander Sellström

Perhaps nothing explains the concept of **interactive stories** better than *The road not taken* by Robert Frost [Fro15]. His poem tells the story of a man following a road through the woods, when he comes to a fork in the road. Which way should he go? And how will this choice affect the rest of his life? In an interactive story, the author expands their story along multiple branches, and invites the reader to choose for themselves which path to take. They can also restart the story to go back and see what would lie down that other road.

When an interactive story is played on a smartphone, there are additional ways of making those choices besides simply using vocal input. For example, going left or right at a fork in the road could be done by walking left or right, so that the phone's positioning service detects a change in position. A long journey could require the user to wait until the next day to find out what happens.

Based on an existing system for creating and playing interactive stories, we added new interaction possibilities for the listener and made the creation of stories easier by improving the user interface and adding features.

Extern handledare: Matthew Davis, Uppsala University
Handledare: Mats Daniels, Björn Victor och Tina Vrieler
Examinator: Björn Victor

Sammanfattning

Kanske förklarar ingenting begreppet **interaktiva berättelser** bättre än *The road not taken* av Robert Frost [Fro15]. Hans dikt berättar historien om en man som följer en väg genom skogen, när han kommer fram till ett vägskäl. Vilken väg ska han gå? Och hur kommer detta val att påverka resten av hans liv? I en interaktiv berättelse expanderar författaren sin berättelse längs flera grenar och uppmanar läsaren att själv välja vilken väg att gå. De kan också starta om historien för att gå tillbaka och se vad som skulle ligga längs den andra vägen.

När en interaktiv historia spelas på en smartphone finns det ytterligare sätt att göra dessa val, utöver röst. Till exempel kan man välja vänster eller höger vid ett vägskäl genom att själv gå åt vänster eller höger, så att telefonens platstjänst upptäcker en förändring i position. En lång resa kan kräva att användaren väntar till nästa dag för att få reda på vad som händer sen.

Baserat på ett befintligt system för att skapa och spela interaktiva historier har vi lagt till nya interaktionsmöjligheter för lyssnaren och gjort skapandet av berättelser lättare genom att förbättra användargränssnittet och lagt till nya funktioner.

Contents

1 Introduction

Perhaps nothing explains the concept of interactive stories better than *The road not taken* by Robert Frost [Fro15]. His poem tells the story of a man following a road through the woods, when he comes to a fork in the road. Which way should he go? And how will this choice affect the rest of his life? Every choice we make means not choosing something else, and it seems a common trait in humans to never really stop wondering what would have happened if we had chosen otherwise. In an interactive story, the author expands their story along multiple branches, and invites the reader to choose for themselves which path to take. They can also re-start the story to go back and see what would lie down that other road.

Audiobooks in general are a convenient medium for stories, since they leave the user's hands free for other tasks. Making the stories interactive can increase engagement, as demonstrated by Green and Jenkins [GJ14]. However, interactive audiobooks, when compared with non-interactive ones, often lose the advantage mentioned earlier: the interaction with the listener is usually through clicking buttons on the screen of the device playing the audiobook. Instead of being something to listen to while cleaning or doing the dishes, they require the user to use their hands to interact with the story.

Creating an interactive audiobook is in some ways more complicated than creating a regular, non-interactive, one. The narration must be cut up into individual segments, and the links between these segments defined. Then the links need to be labelled in a way that lets the listener identify and select one of the links to follow to a new segment. Non-interactive audiobooks are stored as a single audio file, meaning that any audio player can play them to the listener, but an interactive audiobook needs the audio player to have additional control logic to read those links and use them, together with the listener's choice, to direct the story in the correct direction.

To aid authors in creating and publishing interactive audiobooks, we present Augmented Audio, a system of two parts: a web application to let authors create interactive audiobooks in an easy way, and a player for smartphones to let listeners play stories created in the web application. With this, we hope that by making the creation tool available to all it will lower the barrier of entry for many aspiring authors of interactive audiobooks.

We continue the work done by Alstergren et al. [AAHM20] on this topic, who created a mobile application and a web-editing tool which could be run on a local computer to create voice-controlled interactive audiobooks. Our focus in this project is adding real-world interactions to the stories and improving the looks and usability of the web tool.

Real-world interactions, described in Sections 2.3, 8, 10.4.2, and Appendix C, means

using the sensors present in smartphones in order to infer actions undertaken by the user in the physical reality and apply these actions in some form to the digital story. If the character in the story is travelling to a nearby city, for example, the step counter could be used to influence how far along the road they are. If the character needs to recover from an injury, the story can require 8 hours to pass before the wound is healed.

Our improvements to *looks and usability* are described in Section 10.4. While the previous system had all functionality needed to create and publish a story, there were many small details that could be improved. Also, the sensor support added to the mobile player meant changes were needed in the web application as well. Making the editor available as a web application, rather than a program to download and run on a local computer, makes it much easier to use for non-technical authors (compare using Overleaf with editing and compiling LaTeX documents on your own computer).

1.1 Division of Labor

This project is the result of equal work by all authors. We have all worked on both applications, but Alexander spent most time on the mobile application, in particular the speech-to-text functions, while Nicke and Krister spent most time on the web application. This is in part reflected in the implementation sections of the report, although we have had daily edit meetings to go over what was written by an individual.

2 Background

In this section, we will use the words *listener*, *reader*, and *user* to refer to a person interacting with a story. A *listener* refers to a user interacting with an audiobook, a *reader* is specifically a user interacting with a paper book, and a *user* is the generic term used when the medium is unspecified or unimportant.

Audiobooks are nothing new conceptually, with some of the first words ever recorded in audio form being a recital of ”Mary Had a Little Lamb” by Thomas Edison [Rub11] They have followed the trends of music with regard to delivery medium: starting from vinyl records and cassette tapes of stories, over CDs to MP3 files and today often provided as a streaming service. They occupy a popular niche in daily life: like music, they leave your hands free for other tasks, and like regular books, they are entertaining and engaging for our minds.

Using the medium of a paper book to tell an interactive story, where the reader can

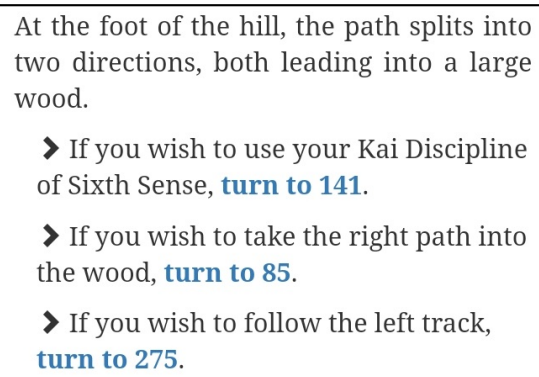


Figure 1 A screenshot showing choices in the mobile app *Kai Chronicles*.

influence the actions of characters to change how the story ends, became popular in the 1980s. One early example is the Lone Wolf series [Dev], where the reader follows and influences the story of the single survivor from a monastery of warrior monks. At the start of the story, the reader chooses a number of skills that the monk has studied before the monastery fell. The story was divided into hundreds of chapters, each of them one or a few paragraphs long, and many chapters had a list of different choices at the end, where each choice led to a different chapter. Some choices were only available if the reader had chosen a particular skill, or found a particular item.

In today's digital offerings of interactive stories, the choices of how to progress the story are usually made by clicking a button on the screen. See Figure 1 for an example from *Kai Chronicles* [LSI] (one adaptation of the Lone Wolf series to smartphones), also showing how an extra option is available by having a specific skill.

This works fine for interactive stories in general, but when adapted to the audiobook as a medium, it loses one of the advantages of audiobooks: the listeners' hands are needed to progress the story. For an audio medium, it would make more sense to let the listener choose an action by voice.

2.1 Speech Recognition

To recognize speech from a user, the system must store and process the sound of the user's speech. The simplest way to store audio data on a computer is as a sequence of amplitude values that can be sent to the speaker in order to reproduce the original sound. Typically, 44100 values are stored for one second of sound, so a new value is sent to the speaker 44100 times per second. To convert these amplitudes from time-series data to

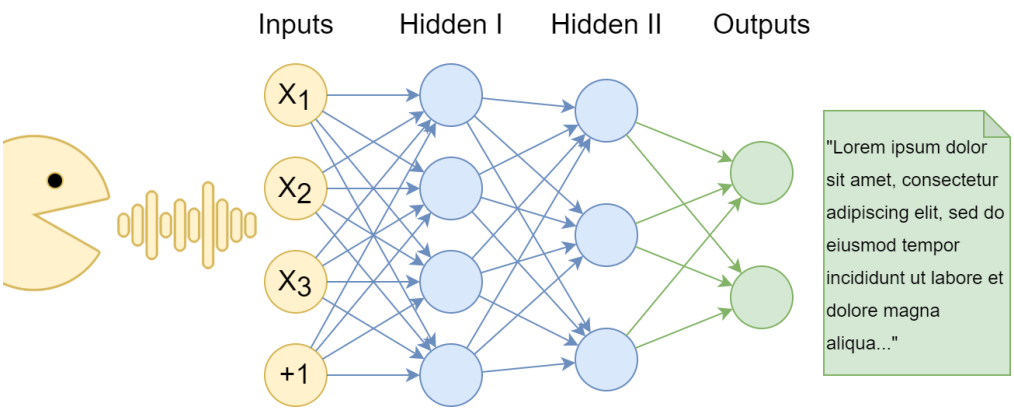


Figure 2 How a neural net converts speech to text. The amplitude of each frequency range comes in on the left. Each node in a hidden layer takes in the input from all nodes to its left, and a weighted sum decides what value the node will pass on to the right. At the end sit the output nodes, each corresponding to one word.

a range of frequencies, the *Fast Fourier Transform* [Smi03] is used. One advantage of going from time to frequency is that a given word, for example "then", looks the same no matter where in the sentence it appears. This frequency distribution is then used as input to a neural net.

The area of speech recognition is one where neural nets (an overview is given by van Veen in his article Neural Network Zoo Prequel: Cells and Layer [vV17]), which is software modeled after the way our brains are presumed to work, are very useful. The idea is that data (inputs X_1 through X_3 in Figure 2) – in this case, the amplitude of the audio input at different frequencies – enters a first layer of nodes (hidden I in the figure), also called neurons, as an array of values. Each node has a different set of weights they give to each input, and if the weighted sum of the inputs exceeds a threshold value the node sends its value on to the next layer of nodes (hidden II in the figure). This threshold value is usually the same constant for all nodes, but an additional constant input (yellow +1 in the figure) called a bias, with its own different weight in each node, in practise allows each node have a different threshold. The collection of all these weights, after they have been adjusted during training, is called the *model* of the neural net. At the end are the output nodes. Each output node outputs its confidence that the input data represents one specific word that that node has been trained to recognize.

The details of how the network converts a spoken sentence into a text string is a vast and interesting topic, which we will not delve into further here. A good place to start for the interested reader would be the paper by Ferrucci [Fer12] describing IBM's Watson, the speech-to-text service we use.

2.2 Databases

Most individual pieces of data is related to others in some way. A typical example is the first name, last name, and age of a person. We will use *record* to refer to all data related to each other, and *row* to refer to that record when stored in a database. When kept in a file, it is up to the user or the program they use to define and maintain this relation. Databases contain extra information about the relations between different pieces of data, and can use that information to maintain the relations when data is updated.

For this project, two types of databases were considered. The first type is the relational database, also called SQL database after the language (Structured Query Language, or SQL) used to create, retrieve, update and delete data. Data is stored in tables. Each table has a number of columns, with each column holding one datum of the record, for example the name or age of each person, as well as the type of that datum (string, integer, boolean, date and so on). The set of all tables is called the schema for the database. One of the columns is usually set to be the index for that table. The index value of each row must be unique within the table. Each record in the database occupies a separate row. A column can also be flagged as a primary or foreign key. A primary key, like an index, must have a unique value. A foreign key can only have a value that is a primary key in another table. Unlike indexes, there can be several key columns in each table. A row cannot be deleted if one of its primary keys is the value of a foreign key in a row in another table. When a row is added or updated, there are builtin verification functions that e.g. ensures a person's age is not "Green" (the age column has been told to only store integers). It is also possible to write additional checks for a table to make sure that the new data adheres to relations that fall outside the builtin rules. For example, a table of employees could have an added rule that says that age must be between 18 and 65.

The other type is a variety of "Not only SQL" databases, or NoSQL databases, called Document Store. Rather than storing data in tables, it is stored in a more flexible form called a document. This document replaces columns with arbitrary key-value pairs, meaning that each person can have a different set of columns for their row. The downside of this is that consistency needs to be maintained by the program accessing the database.

Whatever their internal organization, databases share the same semantics for accessing them. A query applies a set of filters to the data in a table or documents in a store and returns all rows or documents that match the filters. The connections are usually made through so-called database drivers, to make it easier for developers to switch from one database to another. The programming language libraries provides a generic interface to connect to a database and perform queries against it. This interface in turn calls a driver that is provided by the database vendor.

Figure 3 A screenshot showing the controls in the mobile app *Gunship Battle*. Tilting the phone is detected by a sensor and used as a control input in the game.

These concepts will aid understanding when we describe the database used by our system in Section 9.1.

2.3 Smartphone Sensors

Modern smartphones have a lot of sensors that can be used by applications to allow new modes of interaction. As an example, Gunship Battle [Joy], where the player controls a helicopter during combat missions, uses the accelerometer sensors of the phone to control the helicopter (Figure 3), while Pokémon GO [Nia] uses GPS position to move the player's avatar in the game world, which is a map of the physical world around them (Figure 4).

2.4 Our Stakeholder

The project originated with an external supervisor, Matthew Davis, a PhD student at Uppsala University. He envisioned a tool to create interactive audiobooks, and a way to listen to them on smartphones that made use of the different sensor capabilities to drive different game dynamics. A part of the motivation was that they wanted a system that allowed them to investigate how interactive audiobooks with non-screen interactions

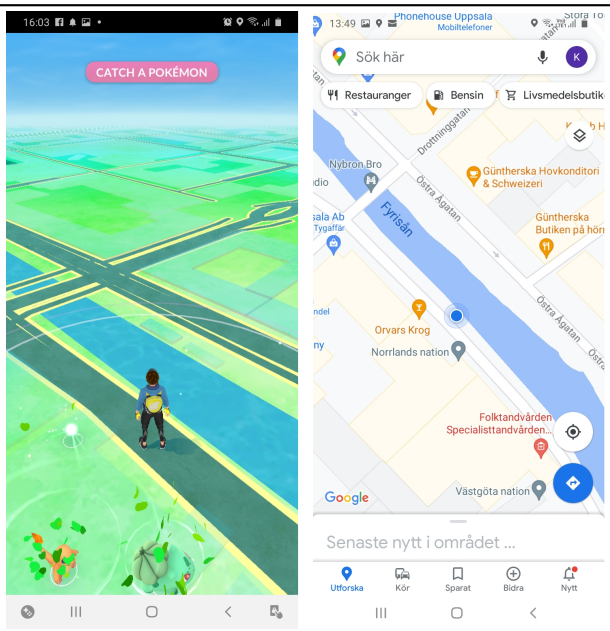


Figure 4 A screenshot showing the interface in the mobile app *Pokémon GO* side by side with *Google Maps* from the same location. The phone's location service is used to move the player's avatar in the game world.

could affect the formation of positive sleep habits. One aspect of the phone application is to facilitate these screen-less interactions, which would also theoretically help avoid blue light emissions late at night [TPP19].

3 Purpose, Aims, and Motivation

Self-publishing is not as easy for interactive audiobooks as it is for non-interactive audiobooks or books in general, which is elaborated on in Section 5.1. The purpose of this project was to simplify the publishing and creation of interactive audiobooks, as well as to allow new forms of interaction.

To accomplish this, the project aimed to provide a simple yet powerful system for authors to create and publish interactive, sensor-based stories in. These interactive stories require a system for listeners to play. Thus the creation of such a system was a complementary aim.

While earlier examples of voice-controlled interactive audiobooks exist, it is very rare for these to incorporate real-world sensor data such as step counters, light levels or

position. Likewise, games based on sensor data have existed for several years, but tend to focus on a pure gameplay experience rather than storytelling. Thus, another purpose was to make it possible for authors to include smartphone sensors in their stories.

By providing a system for creating and listening to interactive stories, with support for story elements incorporating smartphone sensor data, this project aimed at allowing the creation of new experiences. The inclusion of sensor data also helps offload the technical burden from story creators as the simplified creation tools for these experiences allows interactive stories to access that data without needing to manually interact with the sensors.

3.1 Delimitations

Controlling the various menus in the mobile application via voice commands (and thus not just the stories themselves) could be good for improving accessibility and further reducing the need for touch-based interaction. However, this was deemed too time-consuming to implement with any degree of accuracy and usability, and was thus not implemented in the project.

4 Ethics

It is important to consider various ethical factors when developing software, especially software based on smartphones due to their wide-spread nature. In the case of the system developed in this paper, Augmented Audio, two primary concerns arise: Data Privacy and Sustainability.

4.1 Data Privacy

Due to the system utilizing sensor data in its functionality, the issue of data privacy immediately arises. In recent years exploitation and misuse of personal data collected in various mobile applications has become widespread, and users grow more and more suspicious of sharing personal data with companies. At the same time, what this data can be used for has also expanded. Previously unimagined experiences such as Niantic's position-based Pokémon GO [Nia] have been made possible by the incorporation of this personal data (the player's position, in this case). This creates a potential conflict of interest: creators (in our case, storybook authors) might want to have access to as

much sensor data as possible to create new forms of experiences, but users may not want to share their sensor data. In an attempt to mitigate these problems, Augmented Audio implements a solution where the stories receive information of what sensors it has permission to use while letting the user know what sensors a story requires to function. When the story tries to read a sensor the user does not wish to share with the application, a predefined default value will be returned instead. Due to how permissions are handled by the operating system on smartphones, these permissions apply to all stories played by the application rather than being granted or denied on a per-story basis. While it would be possible to add our own permissions system on top of that provided by the operating system in order to have finer granularity, this remains an area of future work.

Another aspect of this data issue is the storage of data. While there are legal requirements in Europe regulating the storage and collection of data (most famously the General Data Protection Regulation, GDPR), there are also ethical concerns surrounding where and how data should be stored. It can in many cases be tempting to temporarily store data on a remote server while it is being processed, especially in cases where complicated processing is necessary (such as when running speech recognition). While this can be handled in a safe and ethical manner, in many cases these transfers either go to third parties with little oversight, or the data is kept for significantly longer than necessary. Due to this, the data can potentially fall into the wrong hands, or be used for purposes the user did not intend for. This is one of the problems with cloud-based speech recognition services such as Google Speech [Good], which often reserve the right to use any data submitted to them in order to improve their software. This was one reason why we switched to IBM's Watson [IBM], and why we ultimately want to run the speech-to-text service on the phone itself.

The recordings of the user's voice is saved twice in our system, once on the phone and again in the back-end. Both files are deleted as soon as the speech recognition service returns a result.

4.2 Sustainability

By enabling stories to use sensors, and preparing for the inclusion of future sensor solutions, this system can be used to create a wide range of different experiences. If used as an educational medium (e.g. by asking questions and having a ”right” or ”wrong” path), it could potentially help the UN:s fourth goal for sustainability, quality education: ”[to] ensure inclusive and equitable quality education and promote lifelong learning opportunities for all” [Uni].

A generic interface for sensors, rather than handling each sensor directly at different

places in the code, means that it is not necessary to make major changes to the system every time a new sensor is to be added. Supporting innovation over time and encouraging the development of new technologies is part of the UN:s ninth goal for sustainability: ”[to] build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation”. [Uni].

It should be emphasized that the system itself is not directly aimed at providing quality education, but rather assist others in working towards that goal. It does this by providing a platform where multi-sensory learning can be facilitated, which has been shown to be superior to single-sensory learning in several aspects [SS08].

5 Related Work

Interactive audiobooks have been around since at least 2006. In 2007 a paper was written by Huber et al. documenting the history of interactive audiobooks [HRHM07]. These books are still usually limited to either screen-based or speech-based interactions despite the fact that smartphone sensors today enable other means of interactivity and are used in games [way][Ket]. As such, the works related to this field can be loosely grouped up into interactive audiobooks without sensor input, smartphone apps with sensor input but not classified as interactive audiobooks, and lastly the few systems which incorporate both this sensor control and can be classified as interactive audiobooks.

5.1 Voice-Controlled Interactive Audiobooks

Audible [Auda], a popular service for non-interactive audiobooks, has an offering [Audb] using Amazon’s smart home assistant Alexa [Amaa] to interact by voice, but it is currently limited to only two stories. These audiobooks are fully voice-controlled, similarly to our app. However, unlike Augmented Audio, they do not implement any sensor data in their stories, meaning the only input available is the user’s speech. Also unlike this project, availability for story creators is not a priority; each story is manually approved and implemented by Audible centrally, adding an intermediary between the author and the end-user.

5.2 Smartphone Apps Using Sensor Data

Many training and fitness apps, like RunKeeper [ASI] and Running Distance Tracker [Fit], use the step counter sensor and positioning service available in smartphones, and can optionally use pulse monitors commonly available in smart watches, in order to keep track of a user's exercise.

Other sensor data can be used to predict and evaluate sleep quality, as done by Bai et al. in 2012 [BXM+12]. Their Android app called SleepMiner uses sensor and communication data to create a model that is used to predict the users' sleep quality with 78% accuracy (comparing the model's predictions against self-reported quality by the test subjects).

Toss 'n' Turn [MDW+14] is another smartphone app created in a study for detecting and evaluating sleep through sensor data. Toss 'n' Turn uses machine learning to create models for evaluating sleep, but is unique in that it can create individual models to achieve greater accuracy.

As mentioned in Section 2.3, and also in the beginning of this section, many games take advantage of the sensors to broaden the range of inputs available for the player to control the game.

5.3 Sensor-based Interactive Audiobooks

In the book *The Mobile Story: Narrative Practices with Locative Technologies*, edited by Jason Farman [Far13], several ways to use the location of the listener in a narration are explored. In a tour guide application for a historical site, the positioning service of the phone is used to select which chapter to read, while an art gallery can use proximity sensors to know what the listener is looking at right now. While these examples are certainly using smartphone sensors to change the story, they are not interactive in the sense we use here: there are no alternative storylines to explore. There is, for example, no location to visit at Gettysburg where the digital tour guide will tell how the South won that battle and went on to win the American Civil War.

There are several products that offer interactive audiobooks, as shown in Section 5.1. One of these, called Iyagi [KLM+17], had a similar purpose to Augmented Audio: adding real-world interactivity to the story. Iyagi aimed to help parents create a healthy bedtime routine for their kids with interactive storytelling through multiple rooms by interacting with physical objects, such as toothbrushes. Its approach was distinctly different from Augmented Audio: instead of using the sensors built into mobile devices,

Iyagi intended to rely on hardware units distributed throughout the house as well as sensors built into Internet of Things-compatible devices [XYWV12]. Three proof-of-concept videos were published in 2017 demonstrating a simple projector device and the visual interface of the mobile application. After 2017 no information has been found relating to Iyagi, so it is unclear whether a working prototype of the system was ever developed.

DreamScape, a system for creating and playing interactive audiobooks, was developed by Alstergren et al. [AAHM20] in 2020. They divided their system into three parts: a web tool for creating the stories, a mobile application to play the stories, and a back-end to store the stories and make them available to the mobile application. A web application is itself composed of two major parts: the user interface, running in a web browser on the user's computer, and a control logic running as a web server. This web server is usually not running on the user's computer, but rather on a dedicated server that is always available and can be accessed by many users at the same time. They, however, made the choice to run the web server on the user's computer, which meant that in order to use the tool a user had to be able to install and configure the web server as well.

Their mobile application used Google's cloud service for speech recognition [Good]. They also chose not to implement a login functionality, but instead defined a single user account that would always be used when the app was running.

Their back-end used two separate services to store story data: the stories themselves were kept in a database provided by Mongo Atlas [Monb], while the audio files were kept in Amazon's Simple Storage Service [Amab] (S3). This is the recommended (by Parse [Par]) way to do it. The purpose of the back-end is to provide a point of access such that both the web application and the mobile application can access the shared data storage services without having a direct connection. Many systems use the local hard drive to provide persistent storage, but distributed or cloud-based systems generally rely on a server on the Internet to provide this.

6 Method

We will first describe the language, frameworks and libraries used in the project, and then go on to describe external services that are used. In all cases, we chose to remain with the choices made by Alstergren et al. [AAHM20], whose code formed the base we built upon. This was largely due to a desire not to have to re-create any existing functionality, but also that the differences between the alternatives are small. Any unique feature presented by one of the options will soon be present in the others as well. Below, we will describe these choices, as well as some of the alternatives available.

6.1 Language

JavaScript [Orab] and Dart [Gooa] are two popular languages that can be used for both web and mobile applications. The benefits of using a single language for all parts of the system will be presented below, when compared with using different languages.

Native mobile applications can also be written in Java [Oraa] or Kotlin [Jet] (for Android) or Swift [App] (for iOS). This gives an edge in performance, and also allows programs to access all features of the platform's operating system. However, it means that each operating system being targeted needs its own version of the source code. Aside from the obvious drawbacks of wasted space and developer time, this also risks the versions drifting apart when it comes to features and bugs/vulnerabilities.

For developing web applications, the choices are nearly infinite. While the client part, running in the user's web browser, is mostly limited to JavaScript, Dart or Java, the code running on the back-end can be written in any language that runs on that computer. While the server code needs to be separate from client and mobile code for all languages, using the same language for all three parts makes it easier for them to communicate with one another (for example, class objects will have the same representation and so be easier to serialize and send over the network). It also aids in code re-use, where common functionality (for example, creating a new user) can be shared between them.

We chose to use JavaScript because the existing codebase was written in that language.

6.2 Frameworks and Libraries

Three popular frameworks for working with web applications in JavaScript are React [Rea], Angular [Ang], and Vue [Vue]. They all provide so-called ”widgets”, UI elements such as buttons, text input boxes and drop-down menus. They all present an abstract copy of the Document Object Model (DOM) [WHA], essentially a tree graph representing all objects on a web page, and their widgets will take care of updating the DOM when needed. The biggest difference is that Angular is a richer (featurewise), and thus heavier (in terms of code size) framework, while Vue is relatively new and still growing fast.

The biggest reason for choosing React was, as with language, that the existing codebase used it.

For the mobile application, React Native [Fac] was used. It is very similar to React, but optimized for mobile apps rather than web apps. In their report, Alstergren et al. say that React Native was an important factor of why they chose React, but it should

be noted that NativeScript [Ope] brings the same convenience to Angular and Vue. As mentioned in the beginning, Dart is a popular alternative to JavaScript, and it has its own framework for creating both web and mobile apps: Flutter [Gooc].

We also used Expo [Exp], which is a framework for developing mobile applications in JavaScript. It provides many libraries to access the operating system, in particular expo-av (used for recording and playing sound), expo-permissions and expo-sensors. This intermediate layer meant that we were not able to access all features we wanted on the phone, but the advantages of a single code-base (we would otherwise need to write different code for Android and iOS) and easy deployment of the application through their own app store outweighed the disadvantages. One area of future work would be to remove Expo and call the operating system directly, in order to be able to use more sensors.

The back-end chosen was Parse Server [Par]. The back-end connects front-ends (web and mobile apps) with third-party services, in our case the database and speech-to-text engine (described in the next section). Another popular choice for back-end is Google's Firebase [Goob]. Again, they provide the same functionality, and our choice to stay with Parse Server was based on existing code.

6.3 External Services

The database chosen was MongoDB [Mona], a NoSQL database. Parse also provides a driver for PostgreSQL [Pos], which as the name implies is an SQL database. Refer back to Section 2.2 for the differences. The choice of MongoDB was mainly influenced by development speed, since no initialization of the database is needed, unlike with an SQL database. In the scope of this project, the differences between the two are small, but one future work would be to switch over to PostgreSQL as that would be a better fit for the data that is stored in the database.

Alstergren et al. originally used Amazon's Web Storage (AWS or S3) [Amab] to store audio files, since they rented their database and thus paid more the more they stored. AWS was cheaper per unit of storage, but did not provide the database semantics needed to store more complex data. Since we run our database on a server we rent, we do not pay per megabyte and therefore decided to store everything in the database.

Speech-to-text conversion was done using Google's speech-to-text service [Good] in the original code, but we switched to IBM's Watson [IBM] for privacy reasons described in Section 4.

We use certbot [Ele] to handle renewal of the server's TLS certificate, which is used to

encrypt traffic between front-end applications and the back-end server.

7 System Architecture

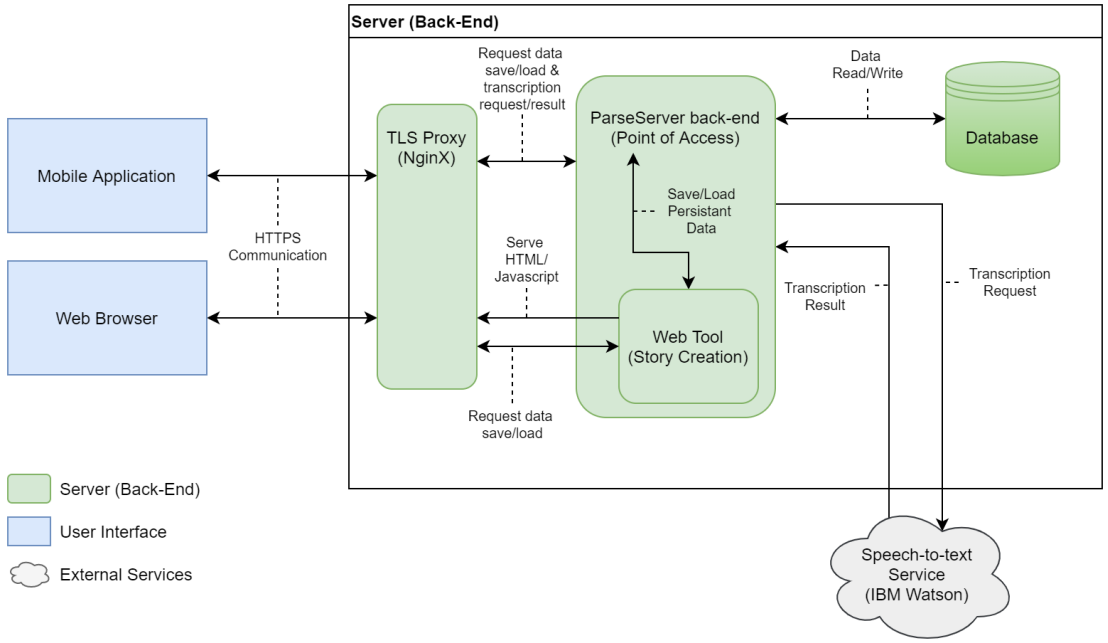


Figure 5 The general layout of the system architecture. When a user uses a web browser to access the web tool, a large part of the functionality is sent to the browser and executes there, only calling back to Parse when it needs to access the database.

The system consists of three main parts, which will be described in more detail below:

- A mobile application which plays the interactive audiobooks and processes sensor input
- A back-end giving clients access to the database and speech-to-text service
- A web tool for creating interactive audiobooks for the system, and storing them in a format ready for uploading to the database

Augmented Audio follows a traditional client-server model. It is a centralized system architecture in which multiple clients send requests to a server for processing. The

server communicates with a database and a speech-to-text service. On the left side of Figure 5 are the clients, either using their web browser to access the web tool or their smartphone to access the story player. Much of the functionality of the story creation tool is delivered to the browser and runs locally, communicating with the ParseServer back-end when data needs to be stored or retrieved from the database. In our implementation, they run on the same machine as the NginX proxy and the database, but the latter two can also be separated to run on other machines if desired.

The system created by Alstergren et al. [AAHM20] provided the basis for our system, but we investigated replacing the existing solution for speech-to-text based on Google [Good] with a new solution that can be run locally on the mobile device itself, e.g. as suggested in [MPA+16], as well as combining the database and storage service into one component. Unfortunately, running speech-to-text on the mobile device was not possible in Expo, and we instead switched from Google Speech to a similar service provided by IBM's Watson [IBM], which was cheaper to use and offered the ability to tell the service what words to look for.

7.1 Mobile Application

The mobile application is where users can play stories created in the web tool. After logging in, the user's data is retrieved from the database. This data includes a list of stories the user has access to, called their library. When looking for a new story to play, or starting/resuming one they have found before, the database is asked for a list of all stories or a single story, respectively. If a user wishes to pause a story, to resume later or to switch to another story, their position in the story is saved in their library, so it will not interfere with another user listening to the same story.

7.2 Back-end

As seen in Figure 5, the database and speech-to-text service are unreachable directly by the clients and must be accessed through the back-end server. The database stores users and stories. The stories can be further divided into four types of data: the story itself, the separate chapters, the links between chapters, and the audio files narrating each chapter. User audio input is not stored in the database, but rather processed and transcribed into command keywords.

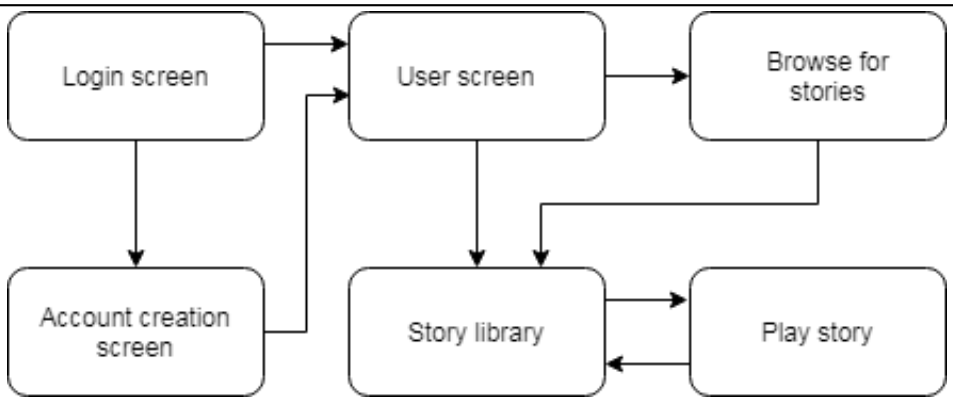


Figure 6 The views and navigation options of the mobile application.

7.3 Web Tool

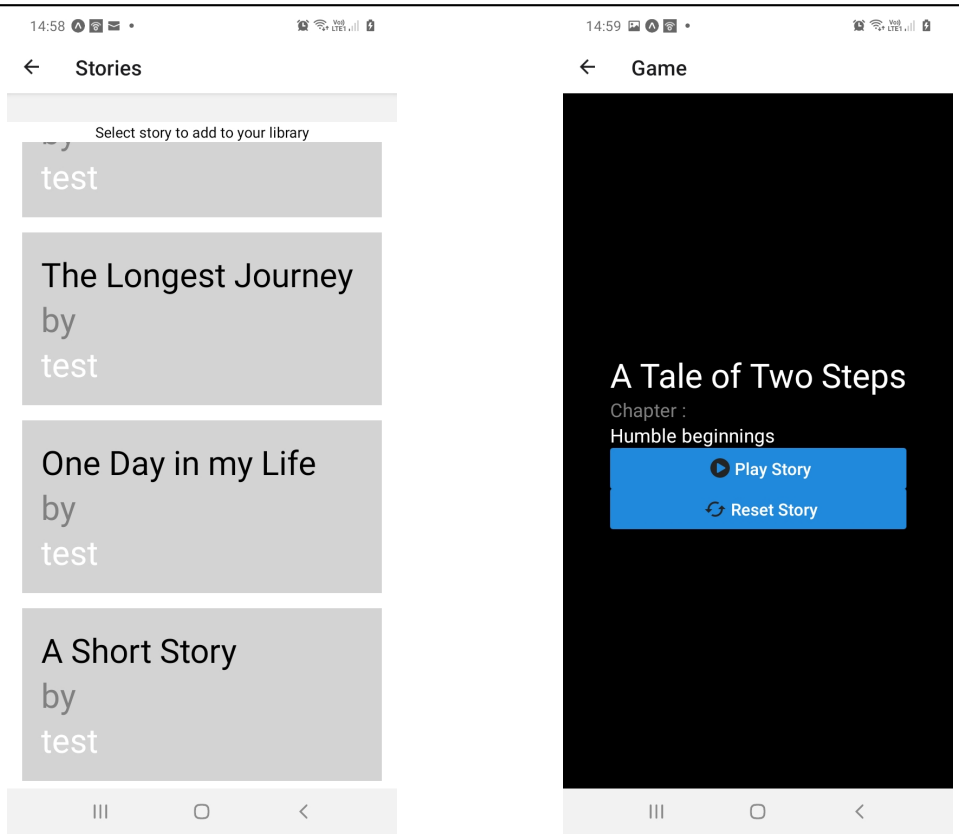
The web tool is hosted on a cloud server rented by the external stakeholder. The server allows content creators to access its functionality without downloading or installing any programs. In it, the creators can create and edit interactive stories, as well as save these stories in the database and publish them to make them visible to users of the mobile application.

8 Implementation of the Mobile Application for Playing Interactive Stories

The mobile application is built in Expo, and consists of three internal components and six screen, as shown in Figure 6. A screen in a mobile application is analogous to a page on a website. When we need to refer to the physical screen on the mobile, we will use the word *display*.

The login screen has fields to input a username and password, and buttons to sign in or create a new account. If the username or password is incorrect, an error message is displayed. The account creation screen has fields to input user profile details, and two buttons to either create the account or reset all fields. If the account could not be created, for example if the account name is already in use, an error message is displayed. Otherwise, the newly created account is logged in and the application navigates to the user screen.

The user screen has three buttons: one that goes to the browsing page, one that goes to



(a) Browsing for stories to add to the library **(b)** A story to play has been chosen

Figure 7 Screenshots of the mobile application

the library of stories the user has downloaded to their phone, and one that performs a logout and returns the application to the login screen. In the browsing screen, shown in Figure 7a, the user can search for stories, and choose to add any of them to their library.

From the library screen, it is possible to select any story and start playing it, or to resume playing it if the story has been started but not yet finished, as shown in Figure 7b. When a story starts playing, the display is dimmed and the buttons inactivated. The story can be paused at any time by tapping the display, which will restore its previous brightness and reactivate the buttons.

Aside from those screens, there are three other components of the app: the variables module, the speech-to-text service, and the interface used by sensors. Conceptually, they are part of the Play story screen, since that is the only place they are called (more precisely, the Play story screen calls the variables module, which in turn calls the various sensors through the interface). The loop used to play a story is shown in Figure 8.

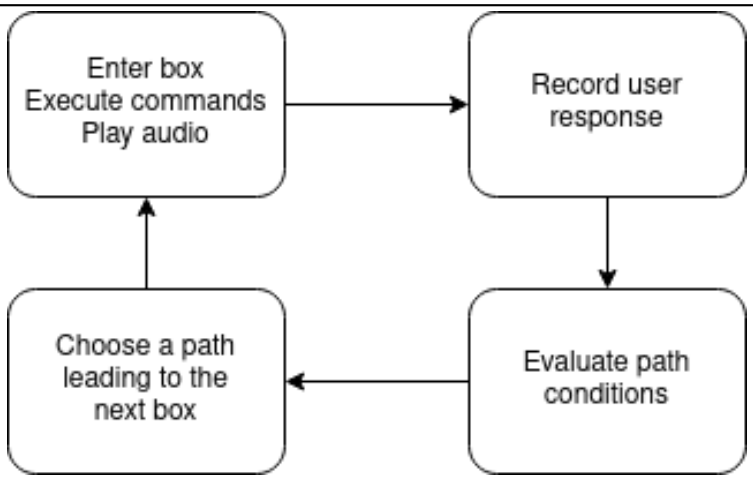


Figure 8 The main loop while playing a story.

Each story consists of a number of boxes or chapters, and paths connecting the boxes. All these data structures are described in Section 9.1. A box is a data structure containing the information required to play the associated sound. One box is called the starting box, and there is a reference to its ID in the story data. If the user is resuming a story they have begun earlier, the current box is read from their library. Each box has an audio URL that contains a narration of that part of the story, as well as a string containing commands to be executed when the story enters that box. The narration starts playing when the box is entered. Each path has a reference to the two boxes it connects, with a direction implicit in the ordering of the references. It also has a keyword string and a condition string. After the audio file of a box has finished playing, the user is prompted to choose an action. The response is recorded and converted to a string. Each path leading away from the box is checked in turn, until one is found that can be taken. If this response string contains the keyword for the path, and if the condition string is either empty or results in a non-zero value, then that path is taken, and the box it points to is entered. No automated method for informing the user of what conditions the condition string contains is implemented. Instead, it is the author's responsibility to inform the listener in their recorded narration. For more information on how the condition string is evaluated, see appendix C. If no path is taken, perhaps because the speech-to-text service mistranslated a word, then a message is played asking the user to repeat what they said, and a new response is recorded.

There is no formal marking of a box intended as an end point to the story. Instead, any box that has no paths leading away from it is considered to be an end point of the story, and the user is returned to the library screen.

The value of a sensor is made available via special variables that can be compared to constant values or other variables in the condition string of a path. Each sensor is a separate module with functions to check if the sensor is available, read its value (or a default value if the sensor is not available), and set the default value.

9 Implementation of the Back-end and Database Storing Interactive Stories

The back-end contains built-in functions (in Parse Server) to handle user creation and authentication, database access to load and store boxes and paths, and updating user information and password.

It also has a function to manage access to Watson's speech-to-text service. Watson wants the recorded speech sent in a file format that is not available in Expo, so the recording from the mobile must be converted before sent on to Watson. This is also where the authentication key is added to the request package, since we do not want that key to be known to the clients.

9.1 Database

The database consists of four tables: Users, Stories, Boxes and Paths, as shown in Figure 9. The audio files are stored in a special file-type table that does not quite work the same as regular tables, and is handled automatically by the Parse Server. Regular tables are accessed using Parse.Query, taking an Object argument to tell it which table to look in. Files are accessed using a URL instead.

Each table has an **id** field that is the index for that table. The Users table contain username, password, name, and email address of each user, as well as a list of all stories they have added to their private library. The Stories table contains title, author, description, reference to the starting box, and a flag saying if the story is published, i.e. if it can be found by users browsing for stories in the mobile app. The author field, unlike all other references in the schema, points to the username instead of the id of the user who created the story. This is in order to save an extra lookup when listing stories, and since the username is also unique, it works just as well. The Boxes table contains a title, a descriptive text, ID of the story it belongs to, the URL of an audio file, and the command string. It also has an (x,y) coordinate that is used by the editor to position the box on the canvas in the web tool. The Paths table contains a reference to the two boxes it connects,

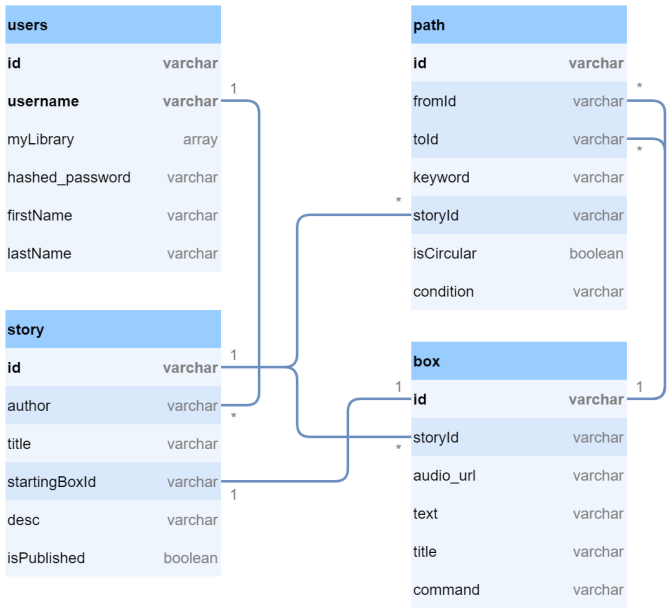


Figure 9 A schema of the database. The **id** field is the index for that table. The fields with blue background are keys. If the name is **bold**, it is a primary key, otherwise it is a foreign key.

a reference to its story, a keyword string, a condition string, and a flag saying if the path should be drawn straight or curved.

10 Implementation of the Web Tool for Creating Interactive Stories

The web tool for creating interactive stories uses Node.js to run on the server and render each page for the user. Parts of the code is included as part of the rendered page and run in the user's web browser. It consists of four pages as shown in Figure 10, each implementing different parts of the application. These parts are the login page, the account creation page, the user homepage, and the story editing page.

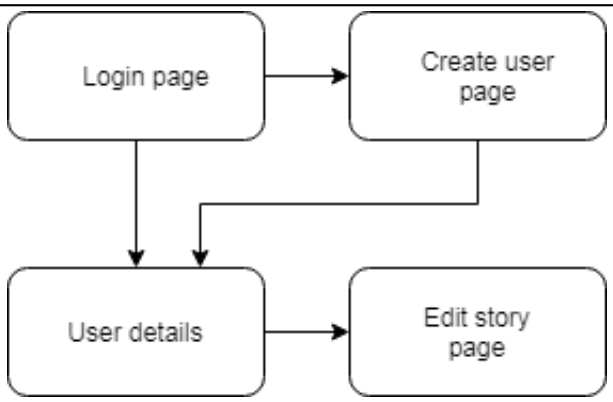


Figure 10 The page layout and navigation options of the web tool.

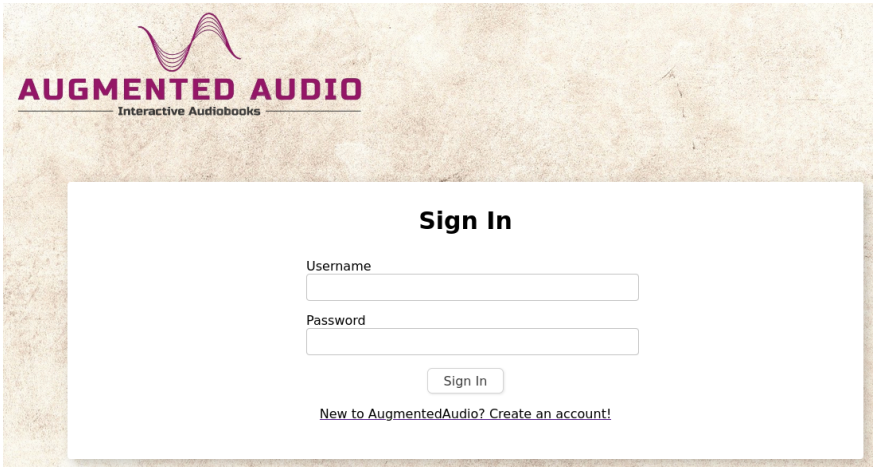


Figure 11 The login page of the web tool.

10.1 Login Page

The login page, shown in Figure 11, is the starting page when a user opens the web tool in their browser. If the user has an account already, they can input their username and password and click ”Sign in” to go to the user homepage. If they do not yet have an account, they can click ”Create account” to go to the account creation page.

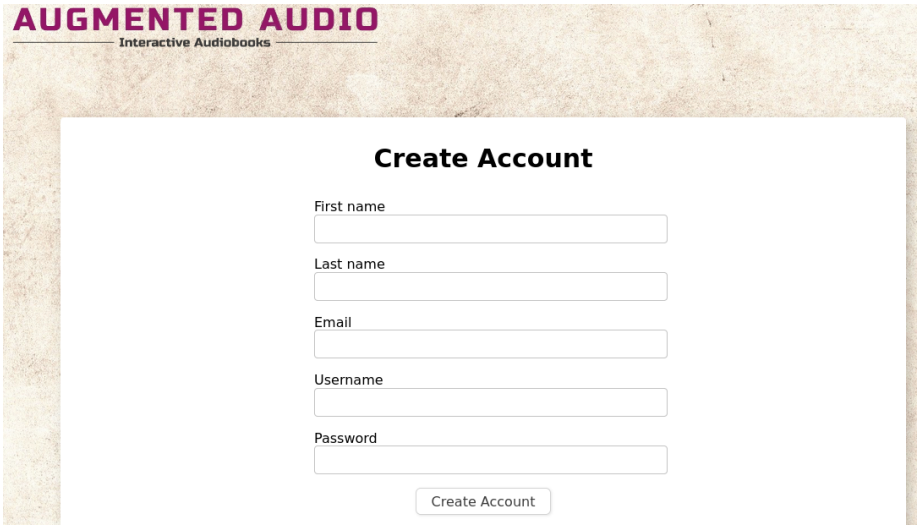


Figure 12 The account creation page of the web tool.

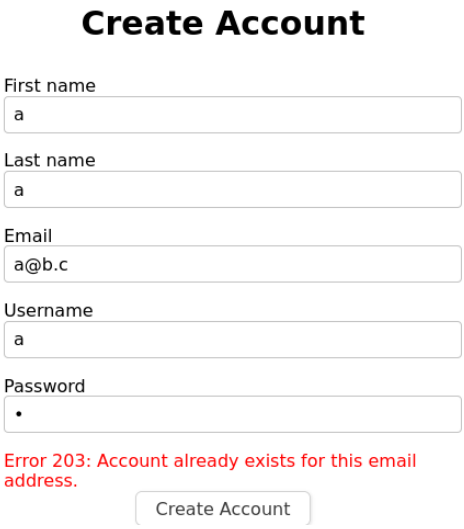


Figure 13 Errors are displayed to the user.

10.2 Account Creation Page

On this page, shown in Figure 12, the user is asked for their first and last names, their email address, desired username and password. Validation of input is minimal: the email must contain an '@' character and a '.' after the '@', the username must not be in use already, and the password must not be empty. When the account is successfully cre-

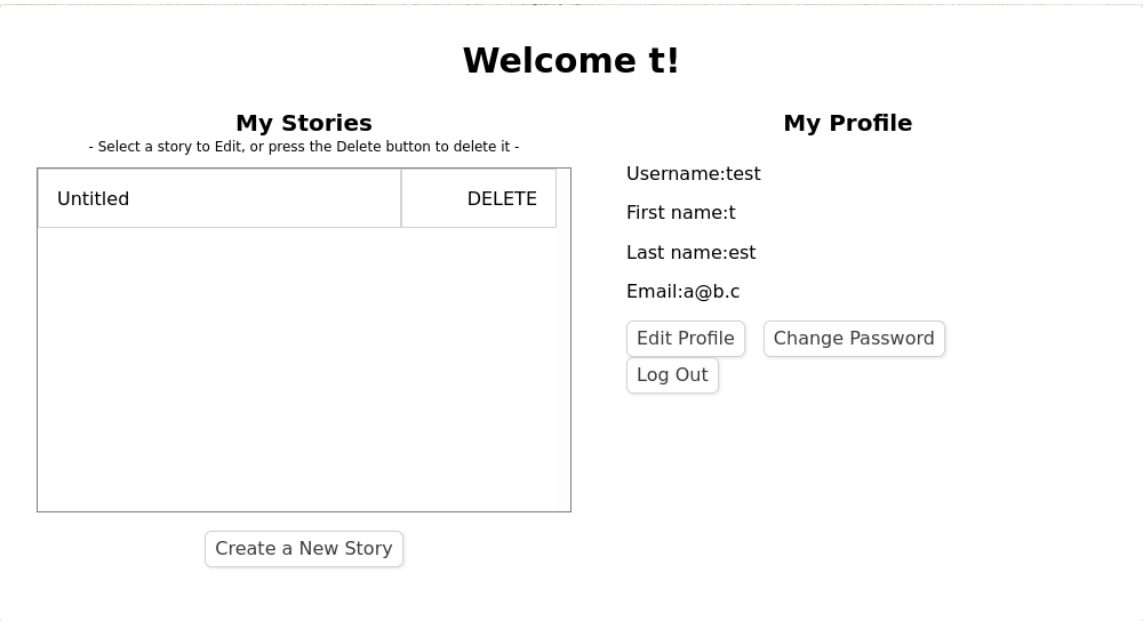


Figure 14 The user homepage.

ated and stored in the database, they are redirected to the user homepage. If something went wrong, a message is shown to the user, as shown in Figure 13.

10.3 User Homepage

This page, shown in Figure 14, consists of three parts: The left side is a list of all stories the user has created, with a delete button next to each story, and a button to create a new story. Clicking that button, or one of the existing stories, transfers the user to the story editing page.

The top half of the right side lists the user's profile information. The bottom half initially contains two buttons to edit profile information or to change the password. If either button is pressed, the bottom half expands to provide input fields for either changing name and email, shown in Figure 15, or to change password.

While the norm today for changing password is that an email or an SMS is sent with a verification code, we have emphasized that our users need not provide valid email addresses to their accounts. We also do not ask for their mobile phone numbers. This is largely due to the known security vulnerabilities of some libraries used by the system [Nata] [Natb] [Natc], but partially due to a desire to collect as little personal information

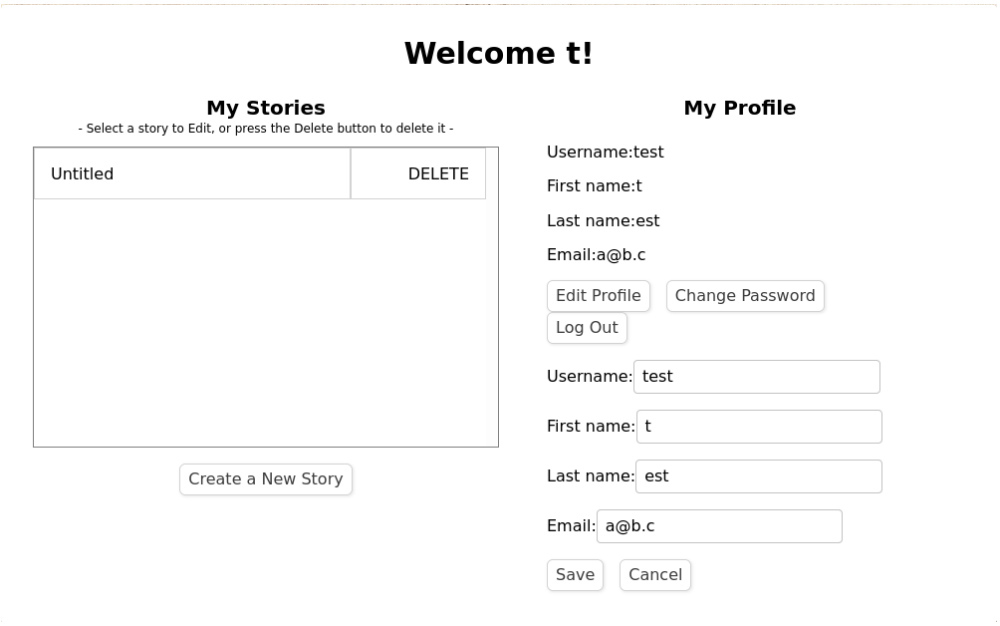


Figure 15 Editing the user's profile information.

as possible. Therefore, we resorted to an earlier best practice. The form for changing password consists of three fields: first the current password, and then the new password and a repeat of the new password. Asking for the new password twice is done because the password is hidden, each character replaced by a dot. To avoid a typo resulting in the user being unable to access their account, we ask them to enter the new password twice, reasoning that they are unlikely to enter the same typo twice. If the current password does not match what is currently stored for that user, or the two new passwords do not match, an error message is displayed and no changes are made. If all three input fields are correct, then the user's password is changed.

It should be emphasized that there is no automated function to recover an account if the user forgets their current password. There is a manual workaround available: an administrator of the server running the web tool (technically, the server running the database, but in our implementation they both run on the same server) can change the password stored for any user. While this is sufficient for the duration of the project, an automated way to handle this is desirable is a finished product.

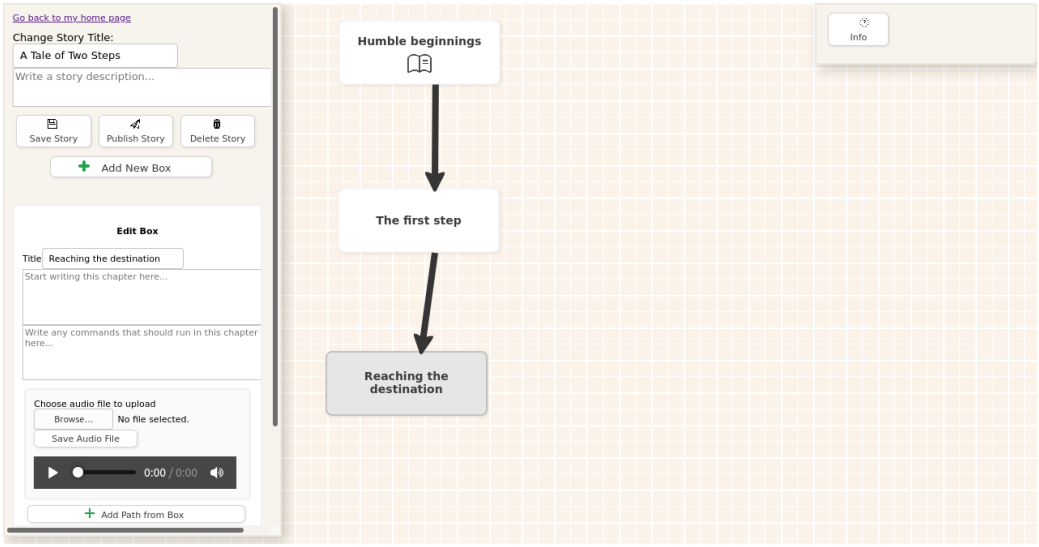


Figure 16 The story editing page.

10.4 Story Editing Page

The story editing page, shown in Figure 16, is divided into two parts: on the left, there is a menu with options to return to the homepage, change the title of the story, and save, publish or delete the story. There is also a button to create a new box. Any status messages are shown below the *Add new box* button.The rest of the page is the canvas component that lets the user describe the chronological layout of the story. In the top right corner, there is a help button with information about how to use the editor, shown in Figure 17.

10.4.1 Left-hand Menu

Publishing a story sets a flag in the database entry for the story so that is becomes visible in the mobile application ”Browse for stories” page, while deleting a story will remove it and all boxes and paths is consists of from the database and return the user to the homepage.

Below this, when a box or a path is selected, is information about that box or path, as shown in Figures 18 and 19. Note how the selected box or path changes color. Aside from input fields to update parameters of the box or path, the box information has a component that allows uploading an audio file to be associated with that box, a button to create a path from this box to another one, mark this box as the starting point of the

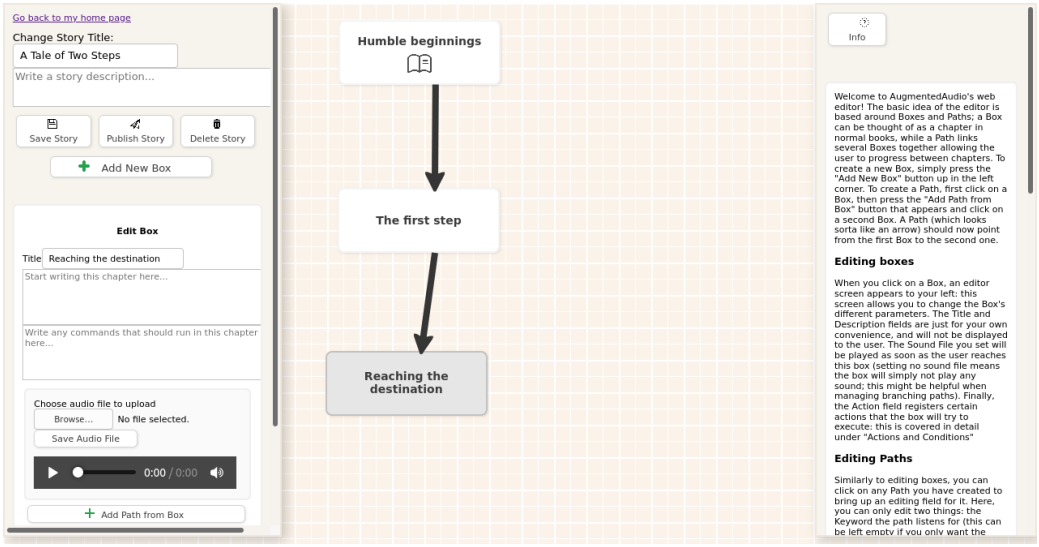


Figure 17 The help information.

story, and to delete the box. The path information only has a button to delete the path, aside from the input fields.

Deleting a story, a box, or a path causes a popup to appear asking the user to confirm the deletion, shown in Figure 20. When a box is deleted, all paths connected to that box are also deleted. Due to files being stored differently to other data objects by ParseServer, with each file being split up into several chunks and only sharing a URL reference, audio files related to a box are not deleted in the process. While in theory it should be possible to use the URL of each audio file, which is stored in the box information, to delete the audio file as well, we were unable to get that to work. In the development version, we simply accept that our database will contain audio files that are not needed by any story. In the finished product, there are two ways to handle this. Either a delete function can be added to the REST API that allows the web tool to delete files from the database, or a maintenance program can go through the list of all audio files, for example once a day at midnight, and delete any audio files that are not referenced by a story.

10.4.2 Canvas Component

The canvas component displays the chronological layout of the story, as desired by the author. Each story is divided into separate components, called *boxes*, and *paths* connecting these boxes together. One box is marked as the starting point of the story. Which box is considered the starting point can be changed. When adding a new box

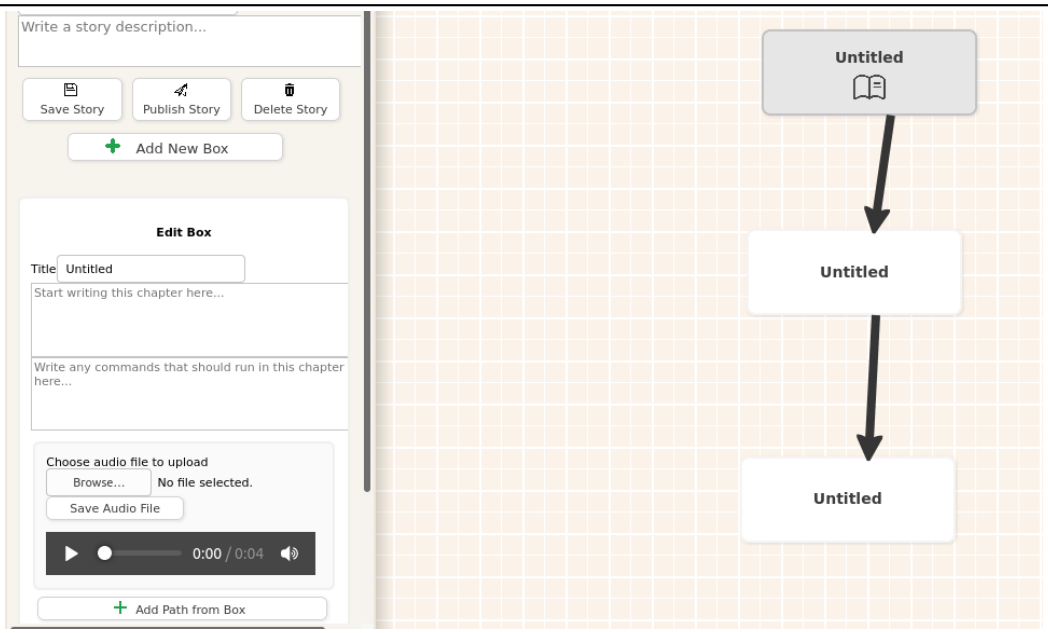


Figure 18 Editing the information of a box.

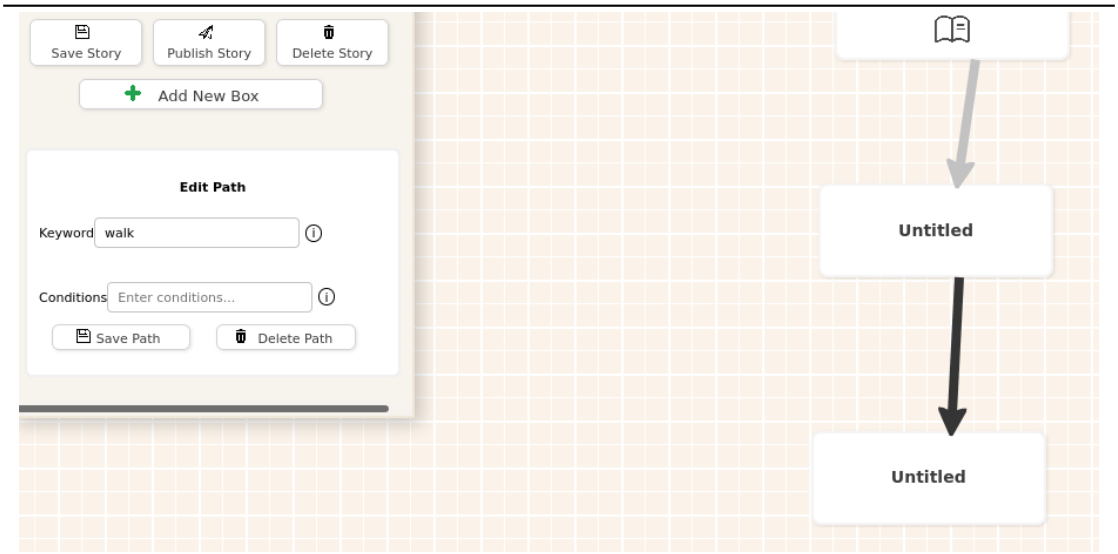


Figure 19 Editing the information of a path.

to the story, it was considered too computationally expensive to find a free section of the canvas, while remaining close to the currently visible region, so a fixed spawning position is adjusted by two random numbers to displace the new box slightly in both the X and Y dimensions. Once placed on the canvas, any box can be dragged to any

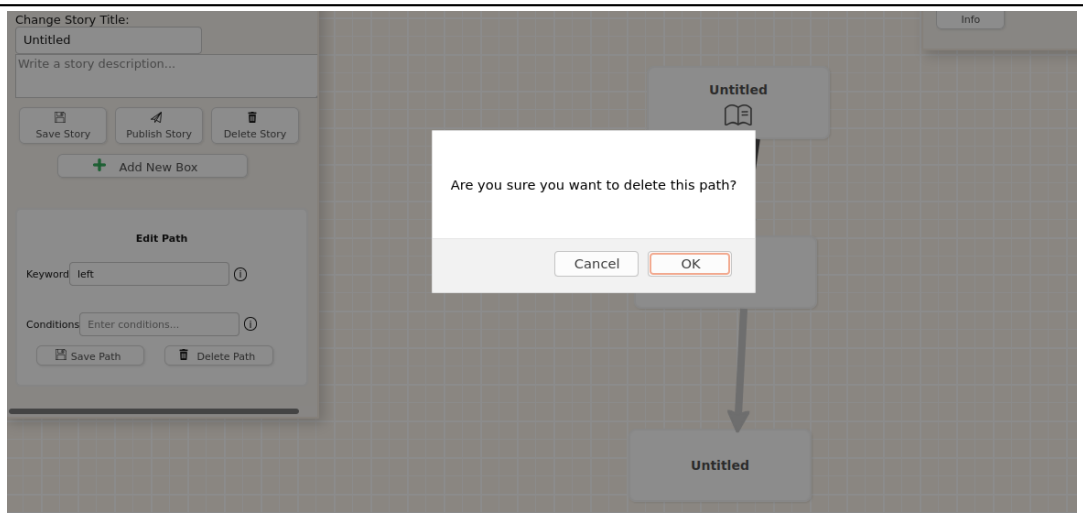


Figure 20 Confirmation dialog before deleting.

position, and the canvas itself is very large in both dimensions. One limitation is that the canvas will not scroll when a box is dragged to the edge of the visible area, but this was considered as a low-priority feature improvement. For performance reasons, paths connected to the box being dragged will not be redrawn until the box has reached its final position.

Both boxes and paths have an information field called *command*. This is how we have chosen to implement variables, and smartphone sensors are represented by special reserved variable names (beginning with an '@' character). While both command strings are evaluated in the same way by the mobile application, our intention is that command instructions in a box will set variables, while command instructions in a path will test variables to determine if this path is available to take. Evaluating a single instruction from the command string will return a numerical value, and the return value of the last instruction will be considered to be the return value of the command string as a whole. The return value from evaluating the command string of a box is ignored. A non-zero return value means that the path is available to take, while a return value of zero means that the path is not available to take at this point in time, although if the user pauses the story at this point and resumes it at a later time, it is possible that re-evaluating the command field will yield a different result. The result of evaluating the command string of a path is not communicated to the user explicitly. For more information on the command field syntax, see appendix C.

Deciding which points of two boxes to connect with a path is still not perfect. The tool considers a line between the center points of the two boxes, and the magnitude and sign

of the difference in X- and Y-coordinates of the two endpoints are used to partition the edge into one of 8 cases. This partitioning is not perfect, and there are several edge cases (pun intended) that are rendered in a graphically unattractive way. Still, the common cases are handled well, and are a considerable improvement over the implementation in the system developed by Alstergren et al. [AAHM20], where the arrow was always pointing straight down. The case of an edge from one box to that same box (a loop) is handled as a special case.

Clicking on a box or path in the canvas will bring up its information in the left-hand menu. Clicking anywhere else in the canvas will de-select the box or path and close the information listing in the left-hand menu. When a box is selected, it is possible to click on the ”Add path from box” button. If the user clicks on a part of the canvas that is not a box, the ”Add path” command is cancelled. Otherwise, a new path is created from the previously selected box to the most recently selected box.

11 Requirements and Evaluation Methods

Our requirements were that both the mobile application and the web tool should be easy to use. It should be possible to use the web tool to create and publish a story that uses sensor data or other variables, and it should be possible to use the mobile application to find and play that story.

The evaluation was divided into functional and usability testing. The functional tests were performed by ourselves as system tests to evaluate that sensor integration and variables work as intended and graded on a simple pass-fail scale, while the usability tests asked potential end-users to perform a given set of tasks in the two applications and grade their opinion of how easy they are to use.

11.1 Functional Testing

Often, each component in a system is accompanied by a set of unit tests to verify correct behaviour by that component in isolation. Once all components pass all their unit tests, components are combined and put through integration tests to verify that the components interact as desired. These integration tests combine more and more components, until all components are included in the test. It is then called a system test. Finally, after the developers are satisfied that system testing demonstrates that their system fulfills all requirements, the system is turned over to the end-users who perform an acceptance test.

Unit and integration tests are usually setup in such a way that they can be automated, and re-run at any time, for example once every day or every time a source code file is changed. System and acceptance tests, in contrast, are usually performed manually, sometimes by developers and sometimes by dedicated testing persons, since the conditions that determine if the test is successful are often harder to encode in a suitable format that can be understood by a computer.

Unit testing, and by extension integration testing, is more difficult for web applications since each function modifies a small part of a large shared system state. There are frameworks available to assist with this, but in this project system testing was instead done directly without prior unit tests. This was in part motivated by confidence in the correctness of the system provided by Alstergren et al. [AAHM20], and in part due to time constraints. While it is generally true that time spent developing unit tests is repaid several times over during the lifetime of a system, for the short development cycle imposed by this project it was felt that testing the components manually would be sufficient. Also, while the system as a whole has a large shared state, each component affects a very small part of this state, and indeed it was correct that changes to one component were unlikely to affect other components.

To test the step counter, two short stories were created: “A long journey” and “A longer journey”. The first story requires a single step, as reported by the step counter, to complete, while the second requires 5 steps. The ability to complete the stories without giving the application permission to access the step counter was also tested.

11.2 Usability Testing

The web app should be intuitive for end users, meaning that they should be able to complete a set of common tasks without detailed instructions, and thus a survey was conducted to get feedback on how easy users found using the web tool to perform the following tasks:

- Create a new story
- Saving and later resuming editing of a story
- Marking a story as complete and publishing the story
- Use a variable representing a sensor

A similar survey was also conducted for the mobile application, where users were asked to grade how easy it was for them to perform the following tasks:

- Browse for stories
- Starting a story
- Use voice commands to progress a story
- Pausing and then resuming a story
- Completing a story

To find participants for the surveys, the aid of the external stakeholder was enlisted. The full instructions given to users participating in the usability evaluation can be found in appendix A. The participants were asked to grade each step on a scale of 0 to 5, with 0 meaning they were unable to complete the step and a 5 meaning that they found it very easy to perform that step. They also had the option to submit a comment on each task that would form feedback for future improvements but not be included in the formal evaluation. Asking people to evaluate ease or difficulty in this manner is highly influenced by individual personalities, and therefore it was decided that an average grade of 3 or higher on all steps was considered to mean that the goal of creating an easy-to-use system had been achieved.

As mentioned in Sections 12.2 and 14, the participants were unable to complete the evaluation of the mobile application using their own phones. Instead, one in our group invited friends to come one by one and perform the evaluation on a phone where the application worked as intended.

As the system utilizes built-in sensor systems which vary depending on the user's hardware, the performance and accuracy of these sensors was not deemed relevant for the performance of the system as a whole. The sole exception to this was the microphone for the purposes of using voice commands, as the performance of the voice recognition is almost entirely software-related, and the speed of this is an integral part of the responsiveness of the mobile application.

12 Evaluation Results

The functional testing of both the web tool and the mobile application was completed with satisfying results. A story can be created with paths containing a keyword, a condition, or both, and it behaves as expected in the mobile application. One minor issue was that for a path with both a keyword and a condition, the user was sometimes asked to repeat the keyword when the issue was that the condition was not satisfied. This has been logged as a bug in our issue tracker on GitHub.

The individual results, as well as written feedback, of the usability evaluations can be found in Appendix B.

12.1 Usability Evaluation for the Web Application

We were only able to find 3 people to evaluate the web application. Unlike with the mobile application, below, they were able to do the tests as intended on their own machines. From the feedback we received, it is clear that the initial box is not marked clearly enough, and it was also unclear that the paths could be selected. There was also a mismatch between terms used in the instructions and in the application.

The grades are shown in Figures 21 and 22. We decided to split the data into two graphs since there was a total of 11 tasks to grade.

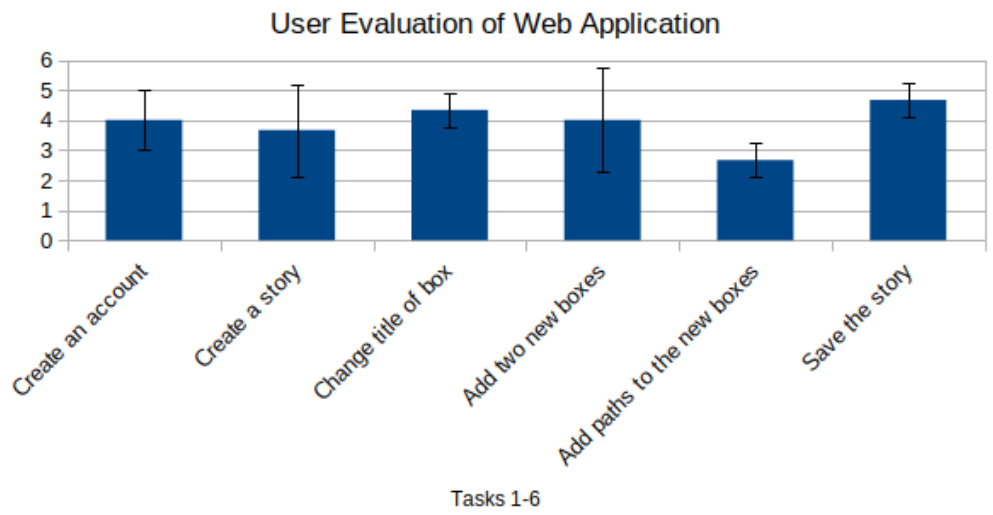


Figure 21 Average and standard deviation of user evaluation of the web application for tasks 1-6. A total of 3 participants were interviewed.

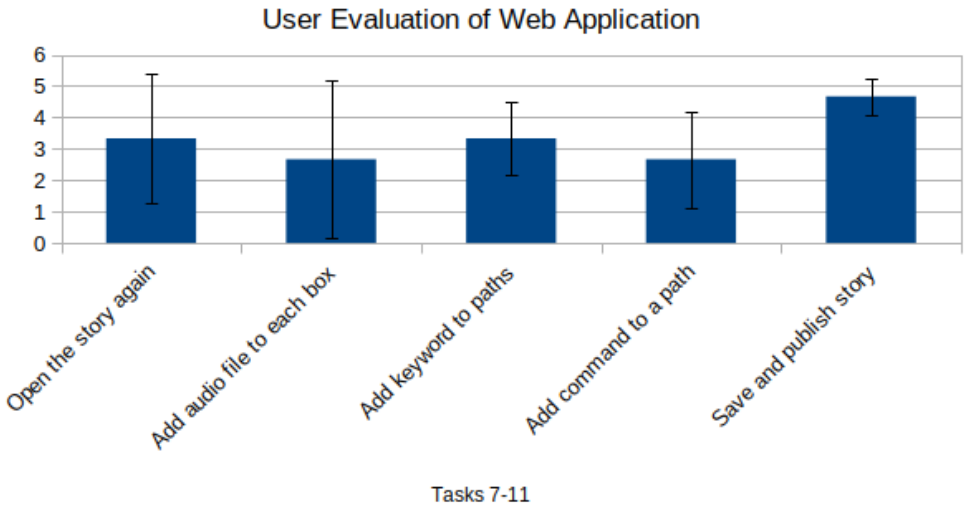


Figure 22 Average and standard deviation of user evaluation of the web application for tasks 7-11. A total of 3 participants were interviewed.

Our requirements were that each task should receive a grade of 3 or higher. This failed for three of the tasks, “Add paths to the new boxes”, “Add audio file to each box” and “Add command to a path”. Many of the other tasks were only slightly above 3 with a fairly large standard deviation. Looking at the individual data, it is not a single person having increased impact due to the small size of our pool of participants. One person had problem with the initial tasks, while another had problem with the latter ones.

12.2 Usability Evaluation for the Mobile Application

Due to a bug related to application permissions, our participants were unable to test the mobile application on their own phones. This will be expanded upon in Section 14. Instead, we went out and asked our friends to conduct the evaluation using our phones. For one of the subjects, we forgot to close the application and switch back to Expo, leading to that person being unable to launch the application (since it was already running). Their grading of this task has been removed from the data set shown in Figure 23. Two persons used an existing account, according to the feedback we received, and graded the task as a 3, as instructed. These two grades have been kept in the data set, leading to a lower average and larger deviation than would otherwise have been the case. An alternate graph, with no removal from the first task and instead removing the two entries from the second task, is presented in Figure 24.

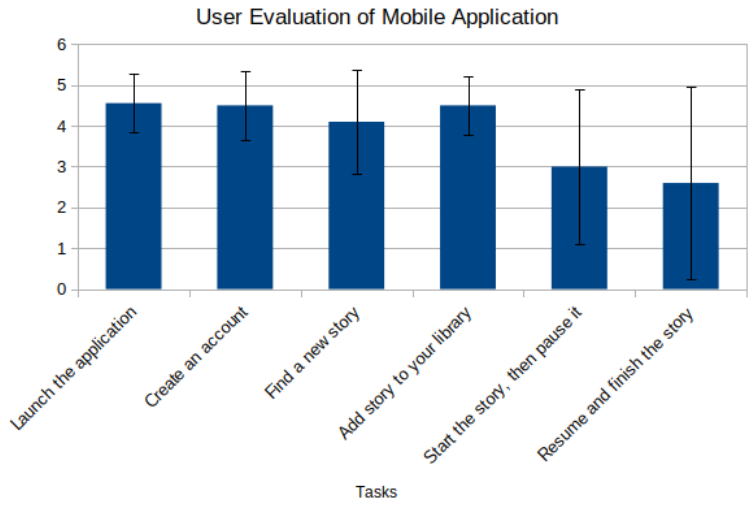


Figure 23 Average and standard deviation of user evaluation of the mobile application, with one data point removed from the first task. A total of 10 participants were interviewed.

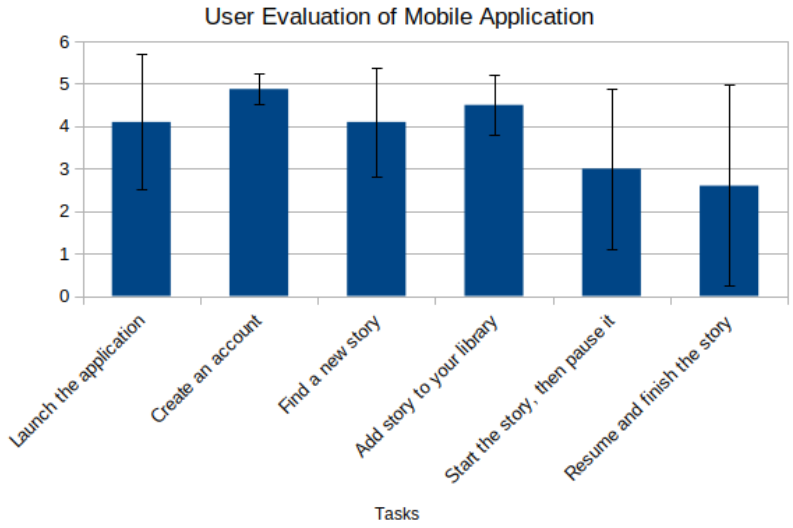


Figure 24 Average and standard deviation of user evaluation of the mobile application, with 2 data points removed from the second task. A total of 10 participants were interviewed.

The task ”Find a new story” was difficult since the name in the application did not match the name given in the instructions document. Pausing and then resuming the story was very difficult for many of the participants, which is something we had not expected. It should be made clear to users that they can pause the story at any time by touching the screen. Some participants were also unclear that a story had been added to their library. One way to fix this would be for stories in the library to be rendered in a different style to make the difference more obvious.

Our requirements were that each task should receive a grade of 3 or higher. This clearly failed for the last task, “Resume and finish the story”, and the one before it, “Start the story, then pause it”, only barely passed, with two people unable to complete it and one grading it as a 2.

13 Results

The result of the project was a two-part system for creating and listening to interactive sensor-based audiobooks. The creation part of the system utilizes a web tool where boxes can be linked together with paths in a graphical user interface (see Figure 16 on page 26). Each box represents the equivalent of a chapter in a more traditional interactive book, while paths represent possible choices. The listening part of the system instead relies on a mobile application, which downloads these audiobooks from a shared server and plays them interactively for the user.

As can be seen in Section 12, functional testing of both systems was completed and successful. Usability testing was also performed, but due to the low number of participants the results were not necessarily representative.

For a detailed list of what was done, please refer to Appendix D.

14 Discussion

The purpose of the project included the creation of a web tool for making interactive stories. The web tool allows authors to use sensor data in addition to voice commands to influence the story’s progression, so this has been achieved. While the evaluations show that not all parts were as easy to use as we desired, it was still possible for everyone to create stories using both voice commands and sensor data to progress the story. One way this could be addressed would be a series of short instruction videos, for example published on YouTube, demonstrating how to use the editor.

The mobile application achieved most but not all desired functionality, mainly due to technical limitations inherent to the Expo framework, which did not allow native libraries to be used. While this did not affect the sensors implemented during the project, any future extension would need to address this as it means very few sensors are available. The project also initially intended to run speech-to-text locally on the mobile device, but this was kept as a cloud-based service due to a lack of Expo-compatible speech-to-text libraries.

Switching from the previous speech-to-text service by Google [Good] to instead use IBM's Watson [IBM] was an improvement from a privacy perspective due to IBM not storing audio recordings. It also allows us to improve accuracy by sending along a list of words that are likely to be present in the speech recording, in our case a list of keywords on paths leading away from the current box.

The participants in our usability study were unable to play stories in the mobile application on their own phones. After asking for permission to use the microphone, instead of starting the story, the application would be reloaded and returned to the login screen. We believe this is due to Expo thinking that the application has been updated when it is being granted a permission, and that it worked on our phones since our application had that permission stored from running the development version. It was decided that we did not have time to track down and fix the issue, and instead the mobile usability survey was conducted using our own phones. This did have the benefit of allowing us to observe the participants as they performed the tasks, in addition to reading the feedback they submitted along with their ratings.

15 Conclusions

This project delivered a platform for creating and listening to interactive stories, focusing on allowing the utilization of several different sensor measurements and enabling easy future extensions to the number and types of sensors available. The basis for the platform was already developed by a previous project, though this basis lacked support for sensors and did not take future improvements into mind in the same manner. By extending the previously developed platform with a simple yet powerful variable management language and easily extendable hooks for sensor inputs this project manages to provide a system with a wide range of possible uses, for example using the current time to stop all progress after 10pm, thereby encouraging the users to go to sleep.

Unlike previous software, Augmented Audio allows interactive stories to rely on a wide range of sensor input rather than exclusively on voice input. What stories this will result in is still unknown. Some potential uses are the gamification of sleeping habits

or applying the already-successful gamification of exercise to the realm of audiobooks. Further research is however necessary to verify the validity of these use cases.

Functional testing was performed and showed that our implemented functions work as intended. The results of the usability evaluations show that not all parts of the applications are as intuitive as we would have wished. While all tasks could be completed, it was unclear how to do some of them, and as a result they needed to stop and think, interrupting their work flow.

16 Future Work

There are several ideas that we would have liked to implement, but did not have time to do. Some of these remain from ideas suggested by Alstergren et al. [AAHM20], while others came up during our development.

16.1 Marketplace for Stories

The current listing of all published stories could be improved in several ways. Rather than only displaying the title and author of each story, a short summary could be presented. This information can be entered in the story editing page of the web tool, but it is not shown in the mobile app. In addition, an optional cover photo could be uploaded for each story, and also displayed here. A rating system for stories could be added as a feature of the mobile app, possibly with the requirement that you need to have finished the story before you can give it an above average rating. Finally, as implied by the title of this section, a way for authors to earn money on their stories could be added. There are many ways to do this, ranging from a Patreon link asking for voluntary donations to a pricing system using a third-party payment processor or the payment methods provided by the mobile operating systems.

16.2 Switching App Frameworks

Using Expo is convenient for rapidly developing and deploying a mobile app to both iOS and Android, but it also imposes severe limitations. By either switching to another framework or coding the app using only React Native, it would be possible to access more sensors, as well as do the speech-to-text conversion on the mobile phone instead of using a cloud service.

16.3 Delete Audio Files from the Database

The HTTP extension used by the back-end (REST) supports a DELETE method. It should be possible to use this, together with the audio file URL saved in the box, to delete the current audio file when a new file is uploaded, as well as when the box is deleted.

16.4 Multiple Audio Files per Box

A common problem with speech-to-text systems, especially when using a cloud service (a local implementation can, to some degree, be trained to recognize a particular user's pronunciation), is that the spoken words are mistranslated. Rather than just asking the user to repeat their choice, it would be nice if the narration related to each box was divided into two parts, one narrating the fixed story element and another listing all choices available to the user. This list could then be replayed before asking the user to repeat what they said. Once multiple audio files are supported, a third file could contain a very brief summary of the story element in that box, and all these summaries could optionally be played when the user resumes a story.

16.5 Password Reset

An active decision was made to not connect each user account to a working email address. If this decision is reversed, Parse Server has builtin functionality to reset the password if a user forgets it. This is usually combined with an activation email sent when an account is created, to confirm that the address given is one that can be accessed by the user. If this function should turn out not to be builtin, it can be added as an external library or implemented from scratch.

References

[AAHM20] I. Alstergren, A. Andersson, L. Hedman, and D. Maric, "Dreamscape, a platform for creating, sharing and listening to interactive stories," dissertation, 2020. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1439802&dswid=4359

[Amaa] Amazon Alexa. Amazon. [Accessed : 2021-04-30]. [Online]. Available: https://www.amazon.com/smart-home-devices/b?ie=UTF8&node=9818047011

[Amab] Amazon simple storage service. Amazon. [Accessed : 2021-04-30]. [Online]. Available: https://aws.amazon.com/s3/

[Ang] Angular. [Accessed : 2021-04-30]. [Online]. Available: https://angular.io/

[App] Apple. [Online]. Available: https://developer.apple.com/swift/

[ASI] ASICS Digital, Inc. [Accessed: 2021-05-13]. [Online]. Available: https://play.google.com/store/apps/details?id=com.fitnesskeeper.runkeeper.pro

[Auda] Audible. [Accessed : 2021-05-13]. [Online]. Available: https://www.audible.co.uk/

[Audb] Audible: Choose your own adventure. Audible. [Accessed : 2021-04-30]. [Online]. Available: https://www.audible.co.uk/ep/chooseyourownadventure

[BXM+12] Y. Bai, B. Xu, Y. Ma, G. Sun, and Y. Zhao, "Will you have a good sleep tonight?: sleep quality prediction with mobile phone." in *BODYNETS*, 2012, pp. 124–130, [Accessed : 2021-05-01]. [Online]. Available: https://www.researchgate.net/publication/262389789_Will_You_Have_a_Good_Sleep_Tonight_Sleep_Quality_Prediction_with_Mobile_Phone

[Dev] J. Dever. Lone wolf(series). [Accessed : 2021-04-30]. [Online]. Available: https://www.projectaon.org/en/Main/Home

[Ele] Automated tls certificate renewal. Electronic Frontier Foundation. [Accessed : 2021-06-13]. [Online]. Available: https://certbot.eff.org/

[Exp] Expo. [Accessed : 2021-04-30]. [Online]. Available: https://expo.io/

[Fac] React Native. Facebook. [Accessed : 2021-04-30]. [Online]. Available: https://reactnative.dev/

[Far13] J. Farman, *The Mobile Story: Narrative Practices with Locative Technologies*. Routledge, 2013.

[Fer12] D. A. Ferrucci, "Introduction to "this is watson"," *IBM Journal of Research and Development*, vol. 56, no. 3.4, pp. 1:1–1:15, 2012.

[Fit] Fitness22. [Accessed: 2021-05-13]. [Online]. Available: https://play.google.com/store/apps/details?id=com.fitness22.running

[Fro15] R. Frost. (1915) [Accessed: 2021-06-04]. [Online]. Available: https://www.poetryfoundation.org/poems/44272/the-road-not-taken#guide

[GJ14] M. C. Green and K. M. Jenkins, "Interactive narratives: Processes and outcomes in user-directed stories," *Journal of Communication*, vol. 64, no. 3, pp. 479–500, 2014, [Accessed : 2021-04-30]. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/jcom.12093

[Gooa] Dart. Google. [Accessed : 2021-06-11]. [Online]. Available: https://dart.dev/

[Goob] Firebase. Google. [Accessed : 2021-06-11]. [Online]. Available: https://firebase.google.com/

[Gooc] Flutter. Google. [Accessed : 2021-06-11]. [Online]. Available: https://flutter.dev/

[Good] Google cloud speech-to-text. Google. [Accessed : 2021-04-30]. [Online]. Available: https://cloud.google.com/speech-to-text

[HRHM07] C. Huber, N. Röber, K. Hartmann, and M. Masuch, "Evolution of interactive audiobooks," in *Audio Mostly - A Conference on Interaction with Sound*. Ilmenau, Germany: ACM, September 2007, [Accessed : 2021-04-30]. [Online]. Available: https://www.researchgate.net/publication/259308580_Evolution_of_Interactive_Audiobooks

[IBM] Ibm Watson speech-to-text. IBM. [Accessed : 2021-05-10]. [Online]. Available: https://www.ibm.com/cloud/watson-speech-to-text

[Jet] JetBrains. [Online]. Available: https://kotlinlang.org/

[Joy] Gunship battle. Joy City. [Online]. Available: https://play.google.com/store/apps/details?id=com.theonegames.gunshipbattle&hl=sv&gl=US

[Ket] Scream go hero. Ketchapp. [Accessed : 2021-05-01]. [Online]. Available: https://apps.apple.com/us/app/scream-go-hero-eighth-note-yasuhati/id1212039074

[KLM+17] M. Krishnaswamy, B. Lee, C. Murthy, H. Rosenfeld, and A. S. Lee, "Iyagi: An immersive storytelling tool for healthy bedtime routine," in *Proceedings of the Eleventh International Conference on Tangible, Embedded, and Embodied Interaction*, ser. TEI '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 603–608. [Online]. Available: https://doi.org/10.1145/3024969.3025076

[LSI] Kai chronicles. LSI, S.A. [Accessed : 2021-04-30]. [Online]. Available: https://play.google.com/store/apps/details?id=org.projectaon.kaichronicles

[MDW+14] J.-K. Min, A. Doryab, J. Wiese, S. Amini, J. Zimmerman, and J. I. Hong, "Toss 'n' turn: Smartphone as sleep and sleep quality detector," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 477–486. [Online]. Available: https://doi.org/10.1145/2556288.2557220

[Mona] MongoDB. [Accessed : 2021-04-30]. [Online]. Available: https://www.mongodb.com/2

[Monb] MongoDB Atlas. MongoDB. [Accessed : 2021-05-07]. [Online]. Available: https://www.mongodb.com/cloud/atlas

[MPA+16] I. McGraw, R. Prabhavalkar, R. Alvarez, M. G. Arenas, K. Rao, D. Rybach, O. Alsharif, H. Sak, A. Gruenstein, F. Beaufays, and C. Parada, "Personalized speech recognition on mobile devices," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5955–5959.

[Nata] Vulnerability CVE-2021-23337. National Institute of Standards and Technology. [Accessed: 2021-05-13]. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-23337

[Natb] Vulnerability CVE-2021-23368. National Institute of Standards and Technology. [Accessed: 2021-05-13]. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-23368

[Natc] Vulnerability CVE-2021-28918. National Institute of Standards and Technology. [Accessed: 2021-05-13]. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-28918

[Nia] Pokémon GO. Niantic. [Online]. Available: https://play.google.com/store/apps/details?id=com.nianticlabs.pokemongo&hl=US&gl=US

[Ope] OpenJS. [Online]. Available: https://nativescript.org/

[Oraa] Java. Oracle. [Online]. Available: https://www.java.com/sv/

[Orab] JavaScript. Oracle. [Online]. Available: https://developer.oracle.com/javascript/

[Par] Parse. [Accessed : 2021-04-30]. [Online]. Available: https://parseplatform.org/

[Pos] PostgreSQL. [Accessed : 2021-04-30]. [Online]. Available: https://www.postgresql.org/

[Rea] React. ReactJS. [Accessed : 2021-05-01]. [Online]. Available: https://reactjs.org/

[Rub11] M. Rubery, *Audiobooks, Literature, and Sound Studies*, ser. Routledge research in cultural and media studies. Routledge, 2011. [Online]. Available: https://books.google.se/books?id=PCV4RAAACAAJ

[Smi03] S. W. Smith, "Chapter 12 - the fast fourier transform," in *Digital Signal Processing*, S. W. Smith, Ed. Boston: Newnes, 2003, pp. 225–242. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780750674447500492

[SS08] L. Shams and A. R. Seitz, "Benefits of multisensory learning," *Trends in Cognitive Sciences*, vol. 12, no. 11, pp. 411–417, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364661308002180

[TPP19] L. Tähkämö, T. Partonen, and A.-K. Pesonen, "Systematic review of light exposure impact on human circadian rhythm," *Chronobiology International*, vol. 36, no. 2, pp. 151–170, 2019, pMID: 30311830. [Online]. Available: https://doi.org/10.1080/07420528.2018.1527773

[Uni] The 17 goals. United Nations. [Online]. Available: https://sdgs.un.org/goals

[Vue] Vue. [Accessed : 2021-04-30]. [Online]. Available: https://vuejs.org/

[vV17] F. van Veen. (2017) Neural network zoo prequel: Cells and layers. Asimov Institute. [Accessed : 2021-06-11]. [Online]. Available: https://www.asimovinstitute.org/neural-network-zoo-prequel-cells-layers/

[way] Fast like a fox. waybefore_. [Accessed : 2021-04-30]. [Online]. Available: https://play.google.com/store/apps/details?id=com.waybefore.fastlikeafox

[WHA] WHATWG. [Online]. Available: https://dom.spec.whatwg.org/

[XYWV12] F. Xia, L. T. Yang, L. Wang, and A. Vinel, "Internet of things," *International journal of communication systems*, vol. 25, no. 9, p. 1101, 2012, [Accessed : 2021-04-30]. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.2417

A Instructions for Usability test

Here are the instructions for the two usability evaluation surveys.

A.1 Web Application

Thank you for helping us to evaluate the web application for creating interactive audio-book stories! The test is expected to take no more than 15 minutes.

A.1.1 Preparation Instructions

- Open the following URL to reach the starting page of the web application: https://augmentedaud.io/
- Open the following Google Form URL where you will fill in your answers: https://forms.gle/pRUvZSFVRCpxPifc6
- Make sure you have either a working microphone or a selection of sound files (between 15 and 30 seconds are best)

A.1.2 Tasks to Perform

1. Create an account for the application. The data you input in the fields can be anything at all, except that the email address must contain an '@' character and a '.' after the '@' (for example, 'a@b.c'). The username you choose here is one of the questions in the Google form, and will let us connect your answers with log entries on the server. Please do NOT use a password for this test that is in use anywhere else; this demo-server cannot assure password security.
2. Create a new story. Give it a title that is the same as your username.
3. The newly created story contains one box already. Change the title of the box to "Once upon a time...".
4. Add two more boxes, one titled "... I took a walk" and the other titled "... I stayed at home".
5. Add paths from the starting box to both of the other boxes.
6. Save the story and go back to your homepage.

7. Open the story you created earlier.
8. Add an audio file to each box, either a recording of your voice or another file. There is no recording function in the web tool itself, but most computers have recording software installed on them already.
9. Add the keyword ”home” to the path from the starting box to the box titled ”... I stayed at home”.
10. Add the keyword ”walk” to the path from the starting box to the box titled ”... I took a walk”.
11. Add the command ”@step>10” to the path from the starting box to the box titled ”... I took a walk”.
12. Save the story and publish it.

A.1.3 Data Analysis and Retention

We will use your answers, as well as log files from the server, to grade how easy the web application is to use. No extra data (for example, browser agent identification or sessions cookies) will be stored in connection with your user profile. Once our summation of the results is finished, we will delete your user account as well as all content you have created.

A.2 Mobile Application

Thank you for helping us to evaluate the mobile application for listening to interactive audiobook stories! The test is expected to take no more than 15 minutes.

A.2.1 Preparation Instructions

- Download the ”Expo Go!” app from your app store.
- Open the following Google Form URL where you will fill in your answers: https://forms.gle/piRaJbt5Mzs2K8Co8
- Follow the instruction on this link: https://expo.io/@augmented-audio/AugmentedAudio

A.2.2 Tasks to Perform

1. Start Expo Go and select our application to launch it.
2. Create a new user (or login as the user you created for testing the web application, if you did that). The data you input in the fields (if you choose to create a new user) can be anything at all, except that the email address must contain an '@' character and a '.' after the '@' (for example, 'a@b.c'). The username you choose here is one of the questions in the Google form, and will let us connect your answers with log entries on the server. Please do NOT use a password for this test that is in use anywhere else; this demo-server cannot assure password security.
3. Go to the "Browse stories" page.
4. Search for the story you created in the previous survey. Add this story to your library.
5. Start playing the story, then go back to "My stories" after the audio has stopped playing.
6. Resume the story you started before and finish it by speaking one of the keywords, either "home" or "walk". If you choose "walk", you also have to permit the application to use the step counter on your phone and walk at least 11 steps before speaking.

A.2.3 Data Analysis and Retention

We will use your answers, as well as log files from the server, to grade how easy the mobile application is to use. No extra data (for example, phone identification or location) will be stored in connection with your user profile. In particular, any voice recordings sent to our server will be deleted as soon as they have been converted to text. Once our summation of the results is finished, we will delete your user account.

B Individual Evaluation Results

This appendix is split in two parts, one for the web application and one for the mobile application. The individual gradings of each task are presented first, and after that we will present feedback for the tasks that received it. Some of this feedback was written in Swedish, and has been translated by us. The original feedback in Swedish is included in parentheses after the translation.

B.1 Web Application Results

In Tables 1 and 2 the individual grades for each task are presented. Below is the feedback for each task.

Create an account	Create a story	Change title of box	Add two new boxes	Add paths to the two new boxes	Save the story
5	5	4	5	3	5
4	2	4	2	3	5
3	4	5	5	2	4

Table 1 Individual ratings for each task of the web application evaluation, first 6 tasks.

Open the story again	Add audio file to each box	Add keyword to paths	Add command to a path	Save and publish story
5	5	4	3	5
4	3	4	4	5
1	0	2	1	4

Table 2 Individual ratings for each task of the web application evaluation, last 5 tasks.

Add two new boxes: “Had been easier if it first box had been clearly marked.”

Add paths to the new boxes: “Problem wasto know witch box to start with se above” (sic), “Say ’Click on box’ to inform how to get the ’add path’ option.”

Open the story again: “Easy to do but the text says ’Select story to edit or delete’, not open.”

Add keyword to paths: “Say ’Click on arrow’ to tell how to select the path.”

Add command to a path: “The web text says ’Conditions’ but the instruction says ’Command’. Use the same phrase!”

B.2 Mobile Application Results

In Table 3 the individual grades for each task are presented. Below is the feedback for each task.

Launch the application	Create an account	Find a new story	Add story to your library	Start the story, then pause it	Resume and finish the story
0	3	1	5	0	0
5	3	5	4	0	0
5	4	5	5	5	5
5	5	4	5	5	5
5	5	5	5	3	0
5	5	4	4	2	0
5	5	5	4	4	3
4	5	4	3	3	3
4	5	5	5	5	5
3	5	3	5	3	5

Table 3 Individual ratings for each task of the mobile application evaluation.

Launch the application: “Already chosen.”, “Had no problem with that part.”, “Easy to decide but hard to know what to say if no one is near to explain (Enkelt att avgöra men svårt att veta vad man ska säga om ingen står framför en och förklarar).”, “Slightly unclear name (Något oklart namn).”

Create an account: “Used the same account as in creating the story.”, “Had prior account.”, “Slight UI issues, worked fine though.”, “Was easy to create an account.”, “Clear and good (tydligt och bra).”

Find a new story: “’Browse stories’ does not exist. It should be ’Find new stories’.”, “Wasn’t named ’browse stories’.”, “Easy when you found one.”, “Unclear name (Oklart namn).”

Add a story to your library: “Didn’t understand that I added the story to my library at first. But when I got it described I understood that.”

Start the story, then pause it: “It looks as if it starts, but return to login page within a second.”, “App crashed/restarted.”, ”Easy to start, missed to pause it due to the story

being so short (Start enkelt, paus missade jag pga kort historia).", "Worked good."

Resume and finish the story: "App crashed/restarted.", "Worked on the 3rd attempt."

C Programming Language Used In Command Fields

The text entered into the command field of a box or a path is called a *program*. A program consists of any number of lines, each line containing any number of *statements*. Each statement, in turn, consists of one operator and two operands.

- ':=' Assignment. The left operand is assigned the value of the right operand. The value of the statement is the new value of the left operand.
- '+' Addition. The left and right operands are added together. The value of the statement is the sum of the two operands.
- '<', '>', '==' Comparison. The left and right operands are compared (numerically, since our language only has integer variables). The value of the statement is either 0 or 1, depending on the result of the comparison.
- If no operator is present in the statement, it is assumed to be either a *variable* or a *literal*. Separating the two is done by the JavaScript function `isNaN`. If the function considers the statement to be a number, it is used as a literal, otherwise it is used as a variable. Variables whose name starts with an '@' character are considered system variables, usually connected in some way to one of the sensors of the smartphone. The part of a system variable name after the '@' is considered a reserved name, and cannot be used by other variables. Non-sensor system variables cannot be assigned a new value (they are not considered *lvalues*, to use the vocabulary of the C language, among others). Assigning a value to a sensor system variable will set the default value that is returned when reading that sensor if the user has not given permission for the application to use that sensor, or if the sensor is not present on this particular smartphone. The value of a literal is simply its numerical value, while the value of a variable is the current value of that variable at this point in time.

The value of a program is the value of its last statement. Lines are evaluated in a pure left-to-right order, without considering priorities of the different operators. There is no equivalent of the parentheses used in other languages to enforce a different operator priority, instead it is assumed that temporary variables can be used to store intermediate results and so achieve the same final result.

The left operand of an assignment operator is used as key in a key-value map of variable names to their values, and not evaluated in any way. The right operand is recursively evaluated, just as the original statement was, and this allows a single line to contain several assignments, for example. The addition and comparison operators recursively evaluate both operands.

D List of Features Improved and Implemented

In the lists below, we use the following abbreviations to indicate what part of the system the change can be found in: **[WT]** means the web tool, **[MA]** means the mobile application, and **[DB]** means the database. The following existing features in the system by Alstergren et al. described in Section 5.3 were improved:

- **[WT]** New boxes in the edit story page are now spawned with a random offset so that underlying boxes are partially visible.
- **[WT]** The paths connecting boxes in the edit story page are now drawn between the closest edge or corner of each box. The arrow terminating the path is correctly rotated to correspond to the incoming angle of the path.
- **[WT]** Deleting a box in the edit story page now also deletes all paths connected to that box.
- **[WT]** Deleting a story now deletes all boxes and paths related to that story.
- **[WT]** Any audio format can now be used for the audio files belonging to a box in the edit story page.
- **[DB]** Story data and audio files related to stories are now stored in the same database.

In addition, the following new features were implemented:

- **[WT]** It is now possible to edit profile information and change password from the user homepage.
- **[WT]** It is now possible to delete a story from the user homepage.
- **[WT]** It is now possible to publish and delete a story from the edit story page.
- **[WT]** It is now possible to read and set variables in a box or path in the edit story page.
- **[WT]** The web tool can now run as a web service instead of needing to be run locally on the user's computer.
- **[MA]** It is now possible to login as any user, instead of always using the same hard-coded one.

- **[MA]** The step counter is available as a system variable that can be used in stories.

Aside from this, we have also upgraded several of the frameworks and libraries used by the system to more recent versions, and thereby removed some, but not all, known security vulnerabilities.