Overview

In this day and age, technology advances faster than ever, but it creates a gap between the older and the younger generations. This results in social isolation between one another, which only worsens due to the recent pandemic and lockdowns. Our client Jon Mc Namara, who is an IBM employee, came up with this project to hopefully resolve this issue.

The purpose of the game is to solve social isolation (particularly for the elderly) causing acute problems relating to health, both mental and physical, it requires an engaging activity that brings joy while also exercising the mind. The game can be interacted via verbal commands to take our users on an engaging and enjoyable but also challenging experience.

The game is a storytelling game that revolves around a linear storyline with multiple choices that may lead to bad ends, and puzzles to be solved. There will be interactions with AI NPCs that react differently depending on the player's emotional response. For the player to advance through the game, he or she has to pick the correct response under different circumstances. The story's setting is situated in real parts of Bristol (local landmarks, areas of interest) and builds on local knowledge that long-time Bristolians would be aware of. This enables the player to explore the city without the need to go outdoors.

The game utilizes Watson Assistant (which incorporates Natural Language Understanding) and Tone Analyser (to determine emotional responses). It also uses Alexa skill (parses player's spoken words into strings), so that those at home can enjoy it without the need to download, configure and setup.

Requirements

System Stakeholders

Internal Stakeholders

1. **Client**: Our client, John McNamara, is a Master Inventor at IBM Hursley, Winchester, as well as the IBM UK University Programs Lead. He is responsible for a team of technologists in the IBM Hursley Innovation Labs whose job is to help innovate IBM Partners, including Watson which our team will be utilizing for part of our software.

2. **Software Development Team**: Our team, who will create, implement, test, and develop this Open Source Software for the client, which will be released as an Amazon Alexa Skill for the Amazon Alexa. The members of the team include:

- Valentino Chen
- Emerson Suter
- Zihong Lin
- Ash Zhuang

3. **Coms20006 Supporting Team**: The University of Bristol members affiliated with the Software Project Engineering (SPE) Unit. This includes:

- **Our Mentor**: Navya Zaveri, a Masters student in the Computer Science Course. He will be offering support and assisting us with technical difficulties on the project should the need arise.
- **Development Managers**: Dr Daniel Schien and Dr Simon Lock. They will supervise us throughout the development of our software as well as be teaching us the fundamental knowledge that is needed for proper software development.

External Stakeholders

1. **End Users**: The user’s of our software. The people that will be playing our Audio Game. Our player base is limited to people that possess an Amazon Alexa as that is the “console” (hardware) that the game is being made for.

User Stories

The Player: The end users who play our game using Amazon Alexa

- **Starting the game**: The player starts the Audio game Alexa skill through their Amazon Alexa and connects to our Watson Assistant session so that they can receive text which their Alexa will then convert into Audio for them to listen to.

1. The player starts up their Amazon Alexa
2. The player tells Alexa to start the Audio Game Skill
3. The Amazon Alexa connects to the AWS Lambda and our Watson Assistant session
4. The introduction text of our game is passed through AWS Lambda to the Alexa
5. The Alexa Skill converts the text of the game's introduction into audio and outputs it
6. The player listens to the riveting introduction of the game

Exceptional Flow

At any point during steps 3 and 4 if there is a connection issue, the Alexa Skill should notify the player of the issue and where it arose- either from connecting to the session or from passing text back to the Alexa.

- **Solving a riddle**: The player receives a riddle from the Watson Assistant session and must answer vocally which the Alexa skill will convert into a string and send back to the Watson Assistant session for it to decide if the answer should be accepted or not and then inform the player of this.

1. The game state progresses by the Watson Assistant session moving to the next section
2. The riddle text is passed through AWS Lambda to the player's Alexa
3. The Alexa Skill converts the text of the riddle into audio and outputs it
4. The player listens to the riddle and replies, verbally, with an answer to it
5. The Alexa Skill converts the answer into a string (text) and sends it to the cloud/ session
6. Watson Assistant reads the string and matches it with a list of correct solutions
7. Watson Assistant accepts the string (wright answer) and sends back text to the Alexa

   (In the fashion of steps 2-4) to let the player know they have succeeded

8. Watson Assistant advances the session to the next section, advancing the game state

Alternative Flow

7. Watson Assistant rejects the string (wrong answer) and sends back text to the Alexa, (In the fashion of steps 2-4) to let the player know they have failed and need to try again

8. Watson Assistant does not advance the session and asks for input (repeat from step 2)

Exceptional Flow

At any point during steps 2, 5, 7, or Alt 8, if there is a connection issue, the Alexa Skill should notify the player of the issue and where it arose- either from connecting to the session or from passing text back to the Alexa.

- **Questioning suspects**: The player is given a choice between different characters to speak to by the Watson Assistant session, through their Alexa, and then must state their choice so that the session can transition to the correct section, i.e. the game can transition to the right state.

1. The game state progresses by the Watson Assistant session moving to the next section
2. The player is asked which suspect/ witness they wish to question, by passing text through Alexa Lambdas to the player's Alexa
3. The Alexa Skill converts the question into audio and outputs it
4. The player listens to the questions and responds with their choice
5. The Alexa Skill converts the response into a string and sends it to the cloud/ session
6. Watson Assistant reads the string and matches it with a list of accepted answers
7. Watson Assistant correctly identifies the right match and proceeds the session to the section corresponding to the player's choice
8. Watson Assistant starts a series of back and forth dialogue with the player
9. Watson Assistant then changes the game state back to what it was before and begins again from step 2.

Alternative Flow

7. Watson Assistant identifies that the given "choice" is invalid and sends back text to the Alexa to let the player know this.

8. Watson Assistant does not advance the session and asks for input (repeat from step 2)

Exceptional Flow

At any point during steps 2, 5, or 8 if there is a connection issue, the Alexa Skill should notify the player of the issue and where it arose- either from connecting to the session or from passing text back to the Alexa.

Implementation Features

For the second user story, “Solving A Riddle” many requirements are needed in order for the software to function as intended.

Prior to step one, the player’s Amazon Alexa needs to connect to the AWS Lambda from there to our IBM Watson Assistant Session. This requires us to have created an Amazon Alexa Skill that creates a connection through the AWS Lambda to a Watson Assistant session. It also requires that we create a Watson Assistant session, in the AWS Lambda, for the Alexa Skill to connect to.

In Step one, the game should advance to the section containing the riddle. For this to be possible we need to structure the Watson Assistant session so that it will move to the wanted part from the previous section, therefore advancing the state of the game.

In Step two the text containing the riddle is passed from the Watson Assistant session through AWS Lambda to the player’s Amazon Alexa, therefore our Alexa Skill needs to be able to receive a string from the AWS Lambda.

In Step three the riddle text is converted into Audio and output to the player. The Alexa skill must therefore be able to convert strings into the type that an Alexa understands as Audio.

In Steps four and five the player gives a verbal response to their Alexa which sends that response back to the Watson Assistant session. The Alexa skill must also be capable of the reverse of the previous two requirements- it must be able to convert the Audio into a string as well as send that string back through AWS Lambda to the Watson Assistant session, without any loss.

In the final three steps the response string is received and evaluated. This means that the Watson Assistant session should have a list of accepted answers that accurately predict ways the player might have formulated the correct answer to the riddle. The session must then also return a string indicating whether or not the answer was correct and advance itself to the next section, or stay in the current one depending on the case.

Personal Data, Privacy, Security and Ethics Management

This project does not require any form of data from the user, the only data that is stored is the save files of the player's progress throughout the game. Therefore there are no ethical concerns or any privacy issues for this project.

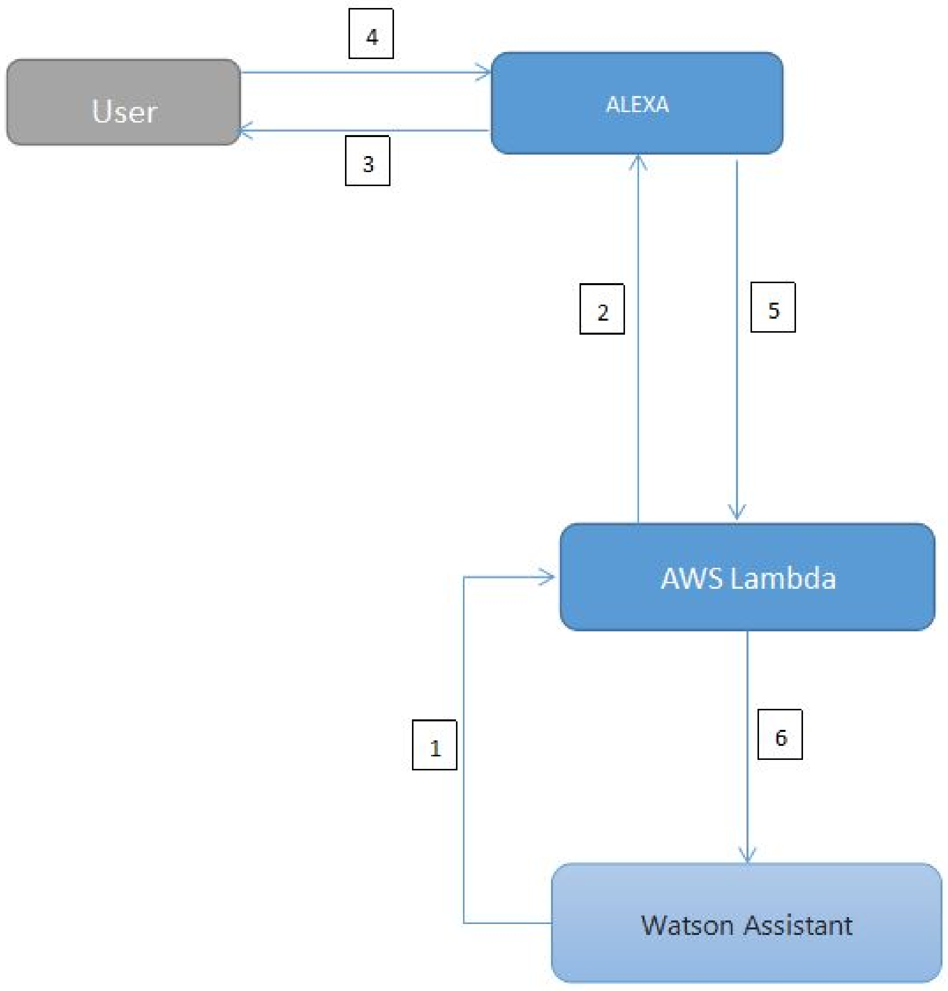
Architecture

Because this game is purely audio based, Alexa is a good choice since it can handle the audio input from the user and parse them into strings for further usage.

AWS Lambda is used for creating an online server and allows access to the Alexa skills and is scalable and synergies well with one another.

Watson Assistant can help determine the user's input and create a suitable response from the pre-made choices, which allows the branching of different outcomes in the game's storyline.

We chose Java as the programming language because it's object-oriented feature is suitable for building applications such as this.



1:Riddle is sent to the AWS Lambda as a string from Watson Assistant.

2:AWS Lambda sent the string to Alexa.

3:Alexa turns the string into audio and sends it to the user.

4:The user gives a verbal response back to Alexa.

5:Alexa turns the audio response into a string and sends it back to the AWS Lambda.

6:AWS Lambda sends the string back to Watson Assistant.

Development Testing

By adopting the TDD strategy throughout the development, we will ensure that test cases are written before the actual implementation of features. Every class method will be tested and we will use JaCoCo to measure and improve the code coverage. We will use Junit for unit testing and Mockito for dependency injections.

As for Continuous Integration, Circle CI will be used to verify any PR doesn't break existing features. To further improve the maintainability, any open Alexa sessions will be logged for the sole debugging purpose and will be cleaned periodically.

There are three key components involved in the project: Alexa, AWS Lambda, and Watson API. Since the actual speech-to-text and conversations are handled by third-party providers, we don't have direct access therefore we are unable to test the actual performance of their services. Instead, we will focus on testing the reliability of the endpoints to make sure an immersive and fluid user experience. In case of disconnection, Alexa should be able to reattempt the same request the second time. This kind of scenario can be tested by sending automated JSON requests against our lambda function.

Furthermore, our services will only respond to Alexa requests, this functionality can be tested as follows:

https://s3.amazonaws.com/echo.api/echo-api-cert.pem	Valid
https://s3.amazonaws.com:443/echo.api/echo-api-cert.pem	Valid
http://google.com	Not Valid
https://s3.amazonaws.com:563/echo.api/echo-api-cert.pem	Not Valid
https://s3.amazonaws.uk/echo.api/echo-api-cert.perm1	Not Valid

Release Testing

Due to the intricate nature of the conversation, it is extremely difficult to fully evaluate whether a chatbot can respond to any question the user might have as there is always room to improve. However, we can try to automate some of the conversations if the user behaves as intended. By using SMAPI provided by Amazon, it's possible to set up simulated conversation tests locally, which serve also as a regression test, to ensure that new features don’t mess up the old ones. If we pick the user story "starting the game" as an example:

We programmatically invoke Alexa by sending "hi, Alexa",	Expects answers from Alexa
We programmatically tell Alexa that we want to invoke "audio game skill"	Expects skill opening, and introduction of the story
We "listen" the story and ask questions to Alexa	Expects changes of story path and the Alexa responds according to the script
If we ask something that is not scripted	Expects Alexa be able to realize that and provide possible answers/hints
Repeat until the end of the story	

By building several automated conversations like this, we can be confident about the quality of the user experience and that Alexa can bring the user back to the flow in case of an exception.
Furthermore, we will perform beta-testing by distributing the game to a group of testers and have their feedback to further improve on the gameplay and the completeness of questions we can expect from the user.