3. Reporting. The 6p (max) report must follow standards of scientific reporting starting with a justification of the design reflecting background knowledge in the domain of spoken interfaces, then including a description of work done along the 8 points above, then a critical evaluation of the system’s performance. Use numbered sections. The DLE submission will be in the form of a zip file that also includes the software. Students can also submit videos on an online repository showing example of userevaluation (this is required in the case of a referred coursework).

# Justification

justification of the design reflecting background knowledge in the domain of spoken interfaces

Databases often contain thousands of data entries across multiple tables. Tree-based Graphical User Interface (GUI) frontends to database backends have been used for a very time (REFERENCE). For many medium-large databases, these interfaces can often prove to be frustrating (REFERENCE?).

Spoken interfaces can provide a faster, more intuitive method for accessing complex databases by untrained users, whilst still providing methods of searching and exploring the database for users who don’t completely know what they are searching for yet.

This projects focus is the development of an interactive speech interface to a database of plants compiled by Plants For A Future (REFERENCE), with the aim of producing a system in which users can find a plant faster than they could with a traditional GUI interface.

## 1. Task domain vocabulary specification

/ e.g. from collected corpus.

## 2. Interaction scenario specification / or defined from conducted observations.

As this program is intended to serve as an assistant in garden centres,

## 3. Personality specification

## 4. Grammar design

## 5. Dialogue design

(will add to the grammar)

## 6. Error catching / handling

(will add to grammar and dialogue)

The primary error catching is performed by the “envCheck1” state, and error handling is performed between the “envCheck” and “envErrorFix” states.

### Error catching

Error catching is primarily performed in the “envCheck1” state. This state accesses the arrays which have search terms stored in them and vocalises these search terms to the user. The user is then asked whether the spoken terms are correct. If the terms are correct, the number of matching search results are queried and vocalised. If there are any mistakes, the “envPreFixCheck” state is called which asks the user if a spoken term was missed (false negative detected), or if a term was recognised that the user did not say (false positive detected). For false positives, “envCheck” is called. For false negatives, the user is returned to the “freeSpeech” state.

### Error handling

“envCheck” starts by asking the user which of the recently vocalised terms was misheard. The user then says which terms were incorrectly identified and the system will remove them from its arrays.

To allow the user to speak multiple corrections in one go, grammar slots were created for each possible search term which are set to a ‘1’ when they get filled. An individual if statement was written for each of these slots. Below is an example of one such if statement:

NLGetIntSlotValue(AppGetNLResult(app), "perennial\_climber\_said", &perennial\_climber);

if (perennial\_climber == 1) {

printf("%s\n", "You said Perennial Climber.");

strcpy(errorStoreArray[errorEnvStringCounter], "Perennial Climber");

errorEnvStringCounter = errorEnvStringCounter + 1;

}

To detect when the user had finished speaking corrections, the following code was written:

if ((AppGetRecognitionStage(app) == 4)) {

AppGoto(app, "envErrorFix");

}

Where “ …) == 4)” corresponds to “AFTER\_SPEECH\_STAGE” in the enumerated “RecognitionStage” structure returned by “AppGetRecognitionStage”.

## 7. Action / response generation

## 8. Evaluation with users / recommendations

(also attach dialogue examples). ( see examples in project folder).

## Description of work done

## Critical Evaluation

of the systems performance

# Appendices

## Appendix 1 – Full system flowchart

Imagine there’s a massive picture here…