

System and Unit Test Report

Project Alnfant
Team Michael and the Boyz
11/22/2016

User Stories	User Stories
1. As an app user, I need a basic user interface, so that the app is easy to use	Sprint 1
2. As a developer, I need to have a database setup so that the AI can store/retrieve data and information about the English language	Sprint 1/2 (Readded)
3. As an app user, I need to have a second screen appear on the UI where I can “teach” the AI a word (construct a word) with its given attributes so I can add it to the database with all of its attributes/connotations.	Sprint 3
4. As a developer, I need to build a parser that will construct a grammar tree out of the tokens inputted so we can identify the structure of each sentence.	Sprint 3
5. As a user of the app, I want to be able to know if the sentence I entered is correct so that the AI will be able to communicate if it does not understand.	Sprint 3

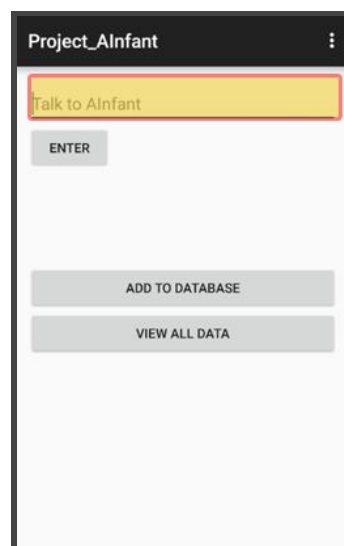
System Test scenarios

Sprint 1

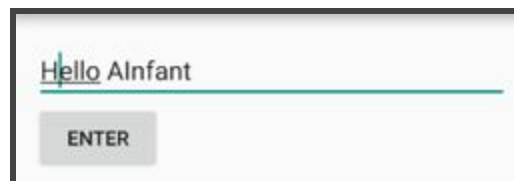
A. As an app user, I need a basic user interface, so that the app is easy to use

Scenario 1

1. Start the Alnfant application
2. Select the text input box



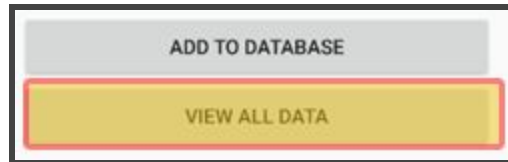
3. Push the send button



4. The sentence is then accepted to be added to the database
5. Press the "ADD TO DATABASE" button to add to database



5. Press the “VIEW ALL DATA” button to view what words the Alnfant knows



Sprint 2

A. User story 1 from sprint 1: As a developer, I need to have a database setup so that the AI can store/retrieve data and information about the English language.

Scenario 2

Name	Type	Schema
▼ Tables (13)		
▼ adjectives		CREATE TABLE
ID	INTEGER	'ID' INTEGER PR
WORD	text	'WORD' text
TYPE	text	'TYPE' text
▶ adverbs		CREATE TABLE
▶ android_metadata		CREATE TABLE
▶ conjunctions		CREATE TABLE
▶ determiners		CREATE TABLE
▶ input_table		CREATE TABLE
▶ interjections		CREATE TABLE
▶ nouns		CREATE TABLE
▶ prepositions		CREATE TABLE
▶ pronouns		CREATE TABLE
▶ sqlite_sequence		CREATE TABLE
▶ verbs		CREATE TABLE
▶ words		CREATE TABLE
Indices (0)		
Views (0)		
Triggers (0)		

1. Displayed here is all of the tables in our database. We are able to visualize each table for each specific part of speech using the DB Browser for SQLite.
2. Each table holds words of the specific part of speech, along with its attributes.
 - a. For example, adjective (as shown), displays each word's ID, the WORD itself, and its TYPE. In the case of adjectives, its TYPE is defined by its connotation (Positive, Negative, Neutral)
3. Upon construction of each word in the UI, a word object is added into the database in its respective table, with its respective attributes. **(See Scenario 4 for details)**

Sprint 3

- A. As an app user, I need to have a second screen appear on the UI where I can “teach” the AI a word (construct a word) with its given attributes so I can add it to the database with all of its attributes/connotations.
- B. As a developer, I need to build a parser that will construct a grammar tree out of the tokens inputted so we can identify the structure of each sentence.
- C. As a user of the app, I want to be able to know if the sentence I entered is correct so that the AI will be able to communicate if it does not understand.

Scenario 4

1. When the user submits a word that the AI does not know (is not in the database) then it opens up this screen

(continue to next page)

The screenshot shows a mobile application interface titled "DropDownMenu". It contains two dropdown menus. The first is labeled "Select part of speech:" and has "Adjective" selected. The second is labeled "Select connotation:" and has "Positive" selected. Below these is a grey button labeled "ADD TO DATABASE".

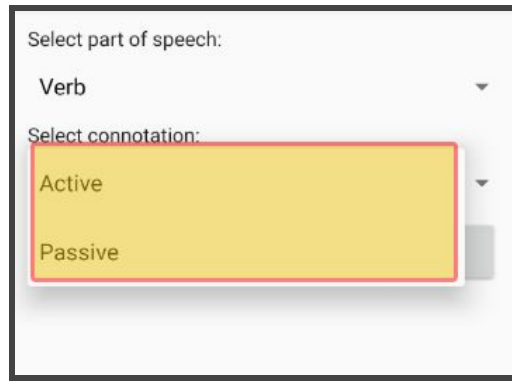
2. The user will first select the button labeled "Select part of speech"

This is a close-up of the "Select part of speech:" dropdown menu. The text "Adjective" is displayed in the selection box, which is highlighted with a yellow background and a red border.

3. This will then open a drop down menu that will allow the user to select from a number of different parts of speech as shown below

This screenshot shows the expanded dropdown menu for "Select part of speech:". The menu is a yellow box with a red border, listing the following parts of speech: Adjective, Adverb, Conjunction, Determiner, Interjection, Noun, Pronoun, and Verb. The list is scrollable, as indicated by a grey scrollbar on the right side of the menu.

4. After selecting one of the parts of speech the user will then select some connotation variables that further describe the word such as in the case of an active or passive verb seen below



The screenshot shows a web form with two dropdown menus. The first dropdown is labeled "Select part of speech:" and has "Verb" selected. The second dropdown is labeled "Select connotation:" and has a yellow box highlighting the "Active" and "Passive" options. The "Active" option is currently selected.

5. Finally the user presses the “ADD TO DATABASE” button that actually stores the word, part of speech, and connotation variables into the database



The screenshot shows a single yellow button with the text "ADD TO DATABASE" in black capital letters.

Scenario 5

1. A list of word objects is passed to the grammar checker function (**see how this works in more detail in Scenario 5**)
2. Make an instance of SyntaxCheck. This is an object that checks the validity of the sentence syntax against grammatical rules with a list of Word objects in SyntaxCheck's constructors arguments.
3. Call the SynaxCheck() function on the word list that was passed in
 - a. Word objects represents different strings of words with various flags to denote the parts of speech.
4. Convert the word list into a list of parsetree.node objects that will be used to create the parse tree (the parsetree object a variable in Syntax Check)
5. Call isValidSentence() and that will build the parse tree with respect to the rules specified in CheckRules and the list of Words passed into SyntaxCheck.

6. The CheckRules grammar defines what parts of speech can shift reduce with another
 - a. ex : the (determiner) , dog (Noun) -> parent (Noun)->children(the,dog)
7. The grammar rules that can be retroactively modified in the CheckRules class in case of future changes in grammar definitions.
8. **isValidSentence()** :This portion of code is done in a loop:
 - a. Check if the current node object in the list can shift reduce with the next node object based off of the check CheckRules class (**See line 7 for details**)
 - i. **If it can be shift reduced:** shift reduce with the next node
 - ii. **If it can not be shift reduced:** do nothing
 - b. Change current node to the next node in the end of the list is reached
9. Repeat that loop until there only two nodes left in the list (the first node is the subject and the second node is the verb phrase)
10. **IsValidSentence()** then returns **true** if the sentence was reduced to two nodes(making it valid) or returns **false** it stopped before it reach two nodes

Scenario 6

1. User inputs a sentence
2. The sentence is submitted
3. The sentence is separated into strings
4. The strings are each looked up in the database
 - a. If the string is found: continue to the next string
 - b. If the string is not found add the word to a list of words to be added to the database
5. **If there were words not found:**The application will then open the activity to add the words there any words that are not in the database
6. **If the application found all the words:** then it will continue to grammar checking the sentence
7. To grammar check the sentence the list of strings is then converted into a list of word object by taking the relevant information associated from the database and creating a word with those variables (where a word can be in the form Noun,Verb, Adjective etc.)
8. The list of word objects is then passed to the SyntaxCheck() module that constructs the parse tree and returns a boolean values as defined above
 - a. **If the value is true:** then the Alnfant will say that it understood your sentence
 - b. **If the value is false:** it will tell the user that it doesn't understand