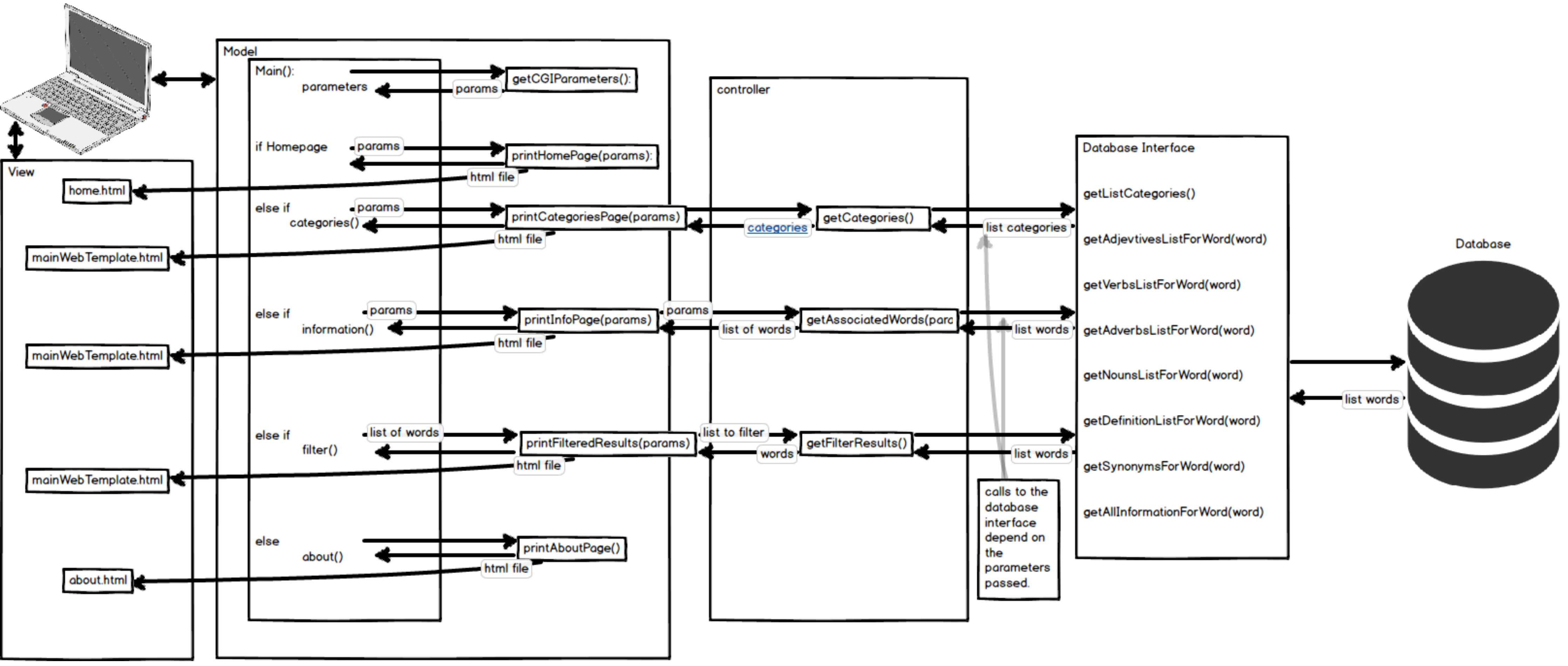
Model
Main():
parameters
params
getCGIParameters():
if Homepage
params
printHomePage(params):
html file
View
home.html
else if
categories()
params
printCategoriesPage(params)
html file
mainWebTemplate.html
controller
getCategories()
categories
list categories
Database Interface
getListCategories()
getAdjevtivesListForWord(word)
getVerbsListForWord(word)
getAdverbsListForWord(word)
getNounsListForWord(word)
getDefinitionListForWord(word)
getSynonymsForWord(word)
getAllInformationForWord(word)
Database
list words
else if
information()
params
printInfoPage(params)
html file
mainWebTemplate.html
params
getAssociatedWords(para
list of words
list words
else if
filter()
list of words
printFilteredResults(params)
html file
mainWebTemplate.html
list to filter
getFilterResults()
words
list words
calls to the database interface depend on the parameters passed.
else
about()
printAboutPage()
html file
about.html

Xword Deliverables

VIEW

Home.html

//Template that displays the home page and has the search bar for user to enter the word they are searching for.

mainWebTemplate.html

//Main template html file that is going to house the different information returned from the database.

about.html

//About page template that explains the usage of Xword and how users can get the most out of the application. This will also cite the websites used to get the information that populated the database.

mainStyle.css

//This will be the main cascading style sheet for all of the html files that we are using. The file will contain the generic styling of the webpages of the site and all html files will be linked to it.

MODEL

def main():

// Main function of the webapp. It will have a set of conditional statements so that the parameters can be passed into the right method for processing. Main will call getCGIParameters() which will be a method responsible for getting the parameters and returning them after they have been filtered and cleaned.
Main also calls the different methods that print the desired pages in the webapp.

def testMain():

//What: Testing if main calls the right methods given a set of CGI Parameters.
//How: Given CgiParameters compare the returned values from the method called with the hard coded expected values of method expected to be called by main.

def getCGIParameters():

// Gets parameters and 'cleans' them (removes illegal characters and spaces).
// Returns the set of cleaned parameters.

def printHomepage():

// Calls and displays the home.html file to the screen of the user. The home page is guaranteed to be displayed properly, as no parameters are necessary for this function to run properly.
// Does not return anything.

```
def printCategoriesPage(word):
        // Takes a word as a parameter and displays the all categories that the user can choose
        from for the associated words of the given word. The parameter is only necessary for
        displaying the correct header.
        // Calls getCategories() which in turn queries the database for the categories.
        // Returns nothing but prints the categories onto a template html.
        // The method is guaranteed to display the full set of categories regardless of the word, as
        it simply queries the database for the set categories.

def testPrintCategoriesPage(word):
        //Compare the expected page displayed with the actual page that a given word will
        display.
        //inspect the content printed on the html template to see if it is the right info printed.
        //Test cases:
                If there is no word input, there should be no display.
                If the word is not found the categories section will be replaced by an error
                message.

def printInfoPage(word):
        // Populates the mainWebTemplate.html with the info retrieved from processing the
        parameters. Calls getAssociatedWords(word) to retrieve the appropriate list of words.
        // Returns nothing, but prints a new website.
        // The method guarantees to display the list of words returned from
        getAssociatedWords().

def testPrintInfoPage(word):
        //Compare the expected page displayed with the actual page that a given word will
        display.
        //inspect the content printed on the html template to see if it is the right info printed.
        //Test cases:
                If there is no word input, there should be no display.
                If the word is not found the categories section will be replaced by an error
                message.

def printFilteredResults(size, character_list, word_list):
        // Takes the resulting list of associated words from the database and applies a user
        specified filter. The user can filter on length of words and position of specific letters. The
        method guarantees to find the words that fulfill all the criteria as specified by the user and
        displays only that content.
        // The character_list parameter should (preferably) be a list of the size as the size
        parameter. The desired characters should be in the position where they are supposed to be
        and contain the character '-' for unspecified letters. [-, r, -, n, -, e] for example, would
        return the word 'orange' among others. The word_list parameter is the list of words to be
        filtered.
        // Non-positive size is interpreted to mean that size does not matter.
        // Returns nothing, but prints a new website containing the filtered results.
```

def testPrintFilteredResults(word):

//Compare the expected page displayed with the actual page that a given word will display.

//inspect the content printed on the html template to see if it is the right info printed.

//Test cases:

If word input in None, home.html is displayed.

If the word is not found the categories section will be replaced by an error message.

def printAboutPage():

// Prints a page with useful information regarding Xword.

// Returns nothing.

CONTROLLER

def getCategories():

// Gets the complete list of categories of words available in our database. This list is guaranteed to be the same every time the function is called, as all categories are returned and it does not depend on any parameters.

// Returns the list of categories based on column headings in the database.

def TestGetCategories():

// Calls getCateogries() to get the list of categories from the database. Given that we know which categories are offered, this method will compare the known list of categories with the one received from getCategories().

def getAssociatedWords(type, word):

// Gets the list of all words associated to with the given word that fall within the specified type. Types include nouns, adjectives, verbs, adverbs, and definitions.

// The list of associated words returned can be empty, in which case there are no associated words with the given word in that category.

// Returns a list of words after calling a function from the database interface.

def testGetAssociatedWords(type, word):

//What: Testing the returned list of Associated words.

//How: Assert that the returned list of words is the same as the expected one by hard coding an expected list of words and checking if the returned list is the same as the hard coded one.

def getFilterResults(size, character_list, word_list):

// Filters the given list of words by length of a word and letter positions. Creates a new list containing all the words in the given list of words that fulfill the criteria.

// Returns a list of words from the given list of words that pass the criteria of length and character positions.

def testGetFilterResults():

// Calls getFilterResults() with a set of parameters for which we know what should be returned. Compares the expected return values with the actual return values.
// Non-positive sizes, empty character_list, empty word_list, valid character_list, positive size will be tested.
// In the case of non-positive sizes, size should not be a filter, meaning that the filter results can be of unequal length. However, for positive sizes the result will be asserted to only contain words of the correct length.
// Assert that when character_list is empty, the position of characters does not matter, meaning that the list of words should only be filtered by length of the words. Since we know the number of words of respective lengths, we can easily assert that the output of the method is correct or not. The same goes for a non-empty character_list, except that in this case certain letter has to be in certain positions in the word.
// Assert that en empty word_list returns nothing but an empty list, regardless of what the other parameters are.

DATABASE INTERFACE

def getListCategories():

// Queries the database for the column headings and returns the headings in a list.
// The headings are guaranteed to be the same every time the method is called, as it does not depend on any parameters and what page the user it as at in the webapp.
// If the method fails to retrieve the information from the database an empty list is returned.
// An empty list is returned if the database connection failed.

def testGetListCategories():

//What: Testing the returned list of Categories.
//How: Assert that the returned list of categories is the same as the expected one by hard coding an expected list of categories and checking if the returned list is the same as the hard coded one.

def getAdjevtivesListForWord(word):

// Queries the database for the adjectives associated with a given word. The complete list of associated adjectives is guaranteed to be returned. If the method fails to retrieve the information from the database an empty list is returned.
// An empty list is returned if the database connection failed.

def testGetAdjectivesListForWord(word):

//What: Testing the returned list of adjectives associated with word.
//How: Assert that the returned list of adjectives is the same as the expected one by hard coding an expected list of words and checking if the returned list is the same as the hard coded one.

```
def getVerbsListForWord(word):
        // Queries the database for the verbs associated with a given word. The complete list of
        associated verbs is guaranteed to be returned. If the method fails to retrieve the
        information from the database an empty list is returned.
        // An empty list is returned if the database connection failed.

def testGetVerbsForWords(word):
        //What: Testing the returned list of verbs.
        //How: Assert that the returned list of verbs is the same as the expected one by hard
        coding an expected list of verbs and checking if the returned list is the same as the hard
        coded one.

def getAdverbsListForWord(word)
        // Queries the database for the adverbs associated with a given word. The complete list of
        associated adverbs is guaranteed to be returned. If the method fails to retrieve the
        information from the database an empty list is returned.
        // An empty list is returned if the database connection failed.

def testGetAverbsListForWords(word):
        //What: Testing the returned list of adverbs.
        //How: Assert that the returned list of adverbs is the same as the expected one by hard
        coding an expected list of adverbs and checking if the returned list is the same as the hard
        coded one.

def getNounsListForWord(word)
        // Queries the database for the nouns associated with a given word. The complete list of
        associated nouns is guaranteed to be returned. If the method fails to retrieve the
        information from the database an empty list is returned.
        // An empty list is returned if the database connection failed.

def testGetNounsListForWords(word):
        //What: Testing the returned list of nouns.
        //How: Assert that the returned list of nouns is the same as the expected one by hard
        coding an expected list of nouns and checking if the returned list is the same as the hard
        coded one.

def getDefinitionListForWord(word)
        // Queries the database for the definitions of the given word. The complete list of
        definitions is guaranteed to be returned. If the method fails to retrieve the information
        from the database an empty list is returned.
        // An empty list is returned if the database connection failed.

def testGetDefinitionListForWords(type, word):
        //What: Testing the definition of word returned.
        //How: Assert that the returned definition is word for word identical to a hard coded
        expected definition.
```

```
def getSynonymsForWord(word)
    // Queries the database for the synonyms of the given word. The complete list of
    synonyms is guaranteed to be returned. If the method fails to retrieve the information
    from the database an empty list is returned.
    // An empty list is returned if the database connection failed.

def testGetSynonymsForWords(type, word):
    //What: Testing the returned list of synonyms.
    //How: Assert that the returned list of words is the same as the expected one by hard
    coding an expected list of words and checking if the returned list is the same as the hard
    coded one.

def getAllInformationForWord(word)
    // Queries the database for all the associated words of the given word. The complete list
    of associated words is guaranteed to be returned. The lists of associated words are
    returned in the form of a dictionary with the specific types as keys and the respective lists
    of associated words within each category as values.
    //If the method fails to retrieve the information from the database an empty list is
    returned.
    // An empty list is returned if the database connection failed.

def testGetAllInformationForWords(word):
    //What: Testing the returned dictionary where the keys are types and the values are lists
    of associated words for the respected types.
    //How: Assert that the returned dictionary is the expected one by checking if it contains
    the expected keys and their corresponding value.
```

DATABASE STRUCTURE

Column headings:

- **Words**
- **Nouns**
- **Adverbs**
- **Adjectives**
- **Verbs**
- **Definition**

Each Cell will be a list of words, which are associated with the word listed in the first column as the parent word of that row see below;

WORD	DEFINITION	NOUNS	ADVERBS	VERBS	ADJECTIVES
food	noun. Any substance that can be metabolized by an animal to give energy and build tissue.	Digestion, Digestive Gourmet Forage Raiment Shortage Dispenser Morsel Nutrition Calorie Scarcity Nourishment	Hungrily Plentifully Greedily Chemically	Cook Digest Forage Nourish Abstain Supply Eat Fodder Store Taste Starve Partake Shovel Dish	Nutritious Nourishing Cooking Palatable Scarce Animal Vegetable Canned Cooked Appetizing Edible Famished Unwholesome Wholesome

Xword Development Schedule

May 1
- Have the scraping script for WordAssociations.net done and ready to go.

May 4
- Have completed all the scraping and have the data stored all in csv files.
- Have completed getCGIParameters().
- Have completed all the controller methods. The controller methods basically only need to call the database interface for the print<page>Page() methods.

May 5
- Have completed the System Architecture, Deliverables, and Development Schedule document!
- Have completed all the tests for all methods. What happens when any parameter = “” (nothing)? What happens when length = negative #?
- Have the filtering algorithm done. The database will not have been set up at this point, but can still write this method because we know what format the database interface will return the data in → lists / dictionary.

May 7
- Database structure is defined; datasets are loaded into the database. Loading stuff into the database may be a lot of work, depending on the form you have it in, so be aware that this phase may take a little while.

May 9
- Have database interface up and working properly.
- Have completed the print<page>Page() methods. The methods need to call the appropriate database interface methods and produce the html files.
- Main() should be done. Main() is basically only lots of if/elif/else statements that determine which print<page>Page() method to call.

May 10
- Have all html and css files done and ready to go.

May 11
- Version 1 of your webapp. It should have all the basic features that your original plan called for, with a reasonable front-end. It should not use hard-coded dummy data, but instead get what it needs from the database. The code should be well structured and properly commented, and should use good style.

May 12
- Have the scraping script for Therausus.com done and ready to go.
- Have completed most of the actual scrape.

May 15
- Have the synonyms dataset (csv) loaded to the database.

- Have modified the html and css files to accompany synonyms.
- Have written a printSynonymsPage() method that works correctly.
- Have added a working getSynonymsListForWord() method to the database interface.
- Have modified the getAssociatedWords() method such that it also works for synonyms.

May 17

- Have modified main() to accompany synonyms.
- Have passed tests for synonyms.

May 18

- Final version of webapp due!