1) Team member names, tasks done by you and tasks done by each of your teammates.(Done by each person)

2) Project description(the WHAT)

The project is to create a fully functioning Lexicon game that has a wait time in under a minute, and includes a dictionary. Lexicon is played by generating 9 letters and displaying those letters and a timer to the user. Then the user enters possible four letter words that can be made from the letters (always including the center letter), which are checked against the dictionary of correct words. The length of correct inputs are used for scoring. The game ends when the timer counts down to.

3) How the program was implemented (e.g. using the dictionary in a text file, James' tool, etc…)(the HOW)

The frontend runs of the code runs on interrupts, and a section of code is dedicated to determining which interrupt was generated, and how the program should proceed. The backend subroutines are called in the main body of the code. There are two primary subroutines BackendInt and BackendSearch. BackendInt is called once when the program is first run, while Backend Search is called at the beginning of each new game. BackendInt loads a dictionary of words, and a list of 9-letter words to the RAM. These are used in BackendSearch where a random 9 letter word is selected and is then passed to a search subroutine. The subroutine searches the dictionary for possible words made from the nine letters and generates a wordlist. If the word list is too small (under 15 words) then the process is repeated until a large enough wordlist is generated. Before passing the wordlist back to the main body the nine letter word used to help generate the wordlist is added to the wordlist. Once the wordlist is created, the saved registers are initialized (score to zero, timer to 60, etc.), and the MMIO system is set up. The timer is initialized and is refreshed when needed as part of the interrupt system. Once input is entered the StringCheck subroutine will be called. StringCheck will compare each letter of the input with each letter of each word in the wordlist. Once a letter of either word doesn’t match, the code moves on to the next word. If a match is found, the subroutine checks a list of bytes to check for duplicate input. Duplicates and wrong input will return -1, while correct inputs will have the length of the correct input added to the score and also the length will be returned. The process of input entry, and input checking will continue until the timer reaches zero, for which the score and the wordlist are displayed.

4) Process the team took to tackle the assignment

The project was divided into two parts: the front end and the back end. The front end consisted of displaying the letters and timer to the user and checking the user’s input against a list of correct inputs for the 9 letters generated. The back end includes an algorithm that selects a 9 letter word from a dictionary (stored in a text file) and finds the other words in the dictionary that can be made from the letters in the 9 letter word. Each part was handled by two group members and then once all the coding was complete, the group all got together in a single location and combined the code to form a working program.

5) Challenges met during the project on how the team overcame them

The biggest challenge was forming a working program from the individual subroutines made by separate members of the group. We were able to put the pieces of code together when we all met in person and debugged everyone’s code as a group.

6) What you have learned doing the project. (Done by each person)