**ENG 1002**

**Major Project Self-Assessment**

**A1860763**

**Introduction**

Below as illustrated I have self-assessed each component of the MATLAB major projects rubric and provided relevant scope for the received score. Greater evidence and demonstration of the scores are included within the appendix. Additionally, the resources I used to gauge foundational knowledge within appdesigner are listed.

**Conceptual Coverage (Score: 20/20)**

- Input/Output: The application is almost entirely input/output based. Buttons, sliders, and text boxes all take user input and produce output, only the output is completely based on the function that is being used. All buttons utilised within each function and UI, act on pressing input established through the call-back functions that accompany the buttons used within the application. Moreover, for the translator and challenge modes, user input can be experienced further by text-based input given within the edit text panels.

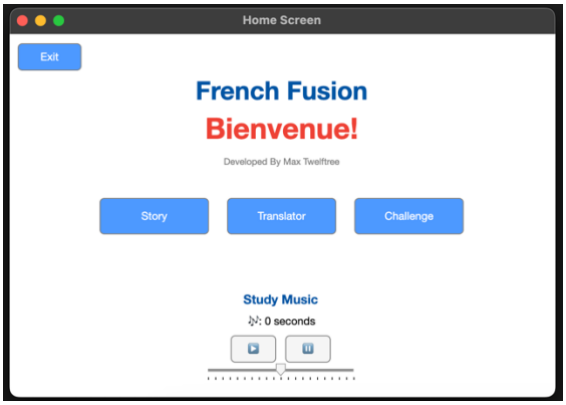

Figure 1.1: French Fusion GUI.

- Visualisation: Within the application, visualisation is very apparent through its entirety. When the user is prompt with the loading screen, a changing state within the 'Loading…' is animated with three different stages. This was developed through using an array of dots and frames where the number of dots correspond to a frame. The frame sequence cycles back infinitely. Furthermore, each application mode resembles a different state entirely. The UI for each mode and its contents change, offering a visualization of the application state. The most notable changing state would involve the runtime with the 'Study Music' feature, where the time the music has been playing clocks with a time that is later reset.

- Programming Concepts:
  1. Loops: For loops were often utilised within the application to iterate over and process lists. Additionally, for loops were generally utilised to append to lists using end+1 indexing and involved checking conditions. While loops were not used

within the code for the application due to their issues with GUI's. They can block the GUI as GUI frameworks run on a single thread that handles both logic and user interface updates. This can result in unresponsive interfaces if the loop continues for an extended period and blocked operations. Despite this I tried to add while loops within my code where I deemed possible but issues with responsiveness and functionality arose. See the following example of a for loop, with conditional execution used within the project:

```
for i = 1:length(lines)
        line = lines{i};
        parts = strsplit(line, ';');

        % Check if the line has the expected number of parts
        if numel(parts) >= 4
            frenchWords{end+1} = parts{1};
            englishTranslations{end+1} = parts{4};
        else
            % Print a warning or handle missing translations as needed
            fprintf('Warning: Line %d does not have the expected format\n', i);
        end
    end
```

Within the for loop, I iterate over the length of the lines (previously initialised lines = to the content within each level text function). Moreover, the loop initialises another variable line which is equal to the current index of I in lines.

2. Vectors/Matrices: Both vectors and matrices are primarily used to manipulate and sort string data within the application. Notably within the for loops, vectors were frequently used. To illustrate this, see the following piece of code taken from myApp.m:

```
for i = 1:numel(frenchWords)
```

Here, i iterates over a vector of values from 1 to the number of elements (numel) in frenchWords.

Moreover, matrices were thoroughly used when positioning and adding colours to GUI elements i.e.: [0 85/255 164/255]. Such matrices were used for the entirety of the project allowing for a more aesthetic and personal theme. Outside of vectors and matrices, many more similar data structures were used. Cell arrays, logical vectors, and arrays were all used throughout the program's entirety. Additionally, MATLAB's dictionary data structure was used for key-value pairing. With this, the dictionary data structure MATLAB has followed the same approach a HashMap would with quick lookups ensuring the time my application took for some functions wasn't extended.

3. Conditional Execution: If, else and else if conditional statements were frequently used within the projects code. However, a switch statement was also used to determine the appropriate level file based on the selected level.

```
switch selectedLevel
  case 1
     % If the selected level is 1, set the level file to level1.txt
     levelFile = 'level1.txt';
   case 2
      % If the selected level is 2, set the level file to level2.txt
     levelFile = 'level2.txt';
   case 3
     % If the selected level is 3, set the level file to level3.txt
      levelFile = 'level3.txt';
    otherwise
      % For any other value of selected level, default to level1.txt
       levelFile = 'level1.txt';
end
```

Within the switch statement cases are utilised as per the previous call-back buttons for each level and reflected in the shown code. The provided code could have been

executed with conditional statements (if, else and else if) but due to rubric requirements I attempted to deliberately use different functionality within various parts of the project where I deemed suitable.

4. Function: Several functions were used throughout the project's entirety, each of them with their own purpose and use case for the application. Due to the structure GUI's host, many functions were needed to allow for specific tasks to be performed. Furthermore, utilising several functions had great benefit when debugging code due to specify and the ease of finding bugs with MATLAB's command window returning warnings for functions and the exact line within that function. With several functions being used I was able to expand upon each of them quite rapidly and had my main one's acting as building blocks for others to expand off. This included the main modes, the buttons associated within them and later the call-backs that responded to other functions within the application.

**Value Add (Score: 20/20)**

- Functionality: A GUI and MATLAB's appdesigner were utilised to add value with great performance and execution. Appdesigner was chosen as the best way to add the greatest value despite the tight timeframe I was given to execute the project. I was also very excited to use it due to my experience with other application development frameworks like flutter and my experience with application development programming languages, Dart and Swift. In addition, my main goal was to create something others and myself can utilise and find value in using. Due to being close with French learning communities I hope that my application can give a potential way people can find value in learning French and I found appdesigner to be the only mean within MATLAB to be able to execute such a goal. On top of utilising appdesigner I added, text-based animation, sounds, and game like functionality to add the most value I deemed I could within the time frame I had.

- Extensions: I added many extensions to the application as shown within the progress files located within the project's repository. The main extensions to eh application:

1. Multiple levels of challenges
2. Dynamic content loading

3. Streak and error tracking, with conditional level placement
4. Error handling
5. Adaptive UI responses
6. Navigation features
7. Additional features (study music, story mode, volume slider and more)
8. Clean UI that follows the French theme of the application

- New MATLAB Features Used: My projects makes use of new techniques within the course and other techniques that haven't been covered. 'fileread' was used to allow for dynamic content loading. As well as nested functions, timers, and audio management. UI control and creation was evidently greatly used despite the course not covering them as features. More importantly it uses a GUI and appdesigner, two features the course has not covered, yet the main principle of my application.

**Incremental Development (Score: 15/15)**

- 15 various progress files with images, writeups and code are located within the projects repository to show the development and time the project took to develop. Additionally, videos are shown for some stages of the app's development. For example, when the 'light' and 'dark' mode were introduced and properly working a screen record was taken. However due to issues with them they were later removed from the project.

**Testing Evidence (Score: 20/20)**

- Test documentation was added for each of the progress files to showcase the applications state and what it struggled with at each stage of development. Goals for each test were included to give myself direction as to what I wanted to improve when I further developed the application from its current state at that given time. Due to the GUI being different to the courses content I couldn't utilise normal test cases that judge output on expected vs actual output and instead incorporated PDF files where I would run the code in its state and provide a reflection and result based on the output of the code. I also thoroughly connected what I aimed to do next for given test cases and what had seen great amounts of progression. Moreover, individual components had been tested for as the project had incremental development, where each new function was tested in a progress write up with errors and photos.

**Comments and Style (Score: 20/20)**

- Evidently there are several comments throughout the code's entirety, in which a function first structure was used. Additionally, each function has its own file within the repository allowing for greater readability. Descriptive and useful comments were added to aid in the users understanding. I also followed a function first approach where I capitalised the function the comment was based on and then gave a description of its purpose and the following lines involved within the function. See the example:

  ```
  %% Start the challenge
  % STARTCHALLENGE: starts the challenge with an extensive amount of
  % functionality allowing for users to gain streaks and increase levels
  ```

**References:**


1. au.mathworks.com. (n.d.). *Adding a New UI Element to a MATLAB App with App Designer Video*. [online] Available at: https://au.mathworks.com/videos/adding-a-new-ui-element-to-a-matlab-app-with-app-designer-1509047047549.html?s_tid=srchtitle_videos_main_5_App%20Designer [Accessed 6 Sep. 2023].
2. au.mathworks.com. (n.d.). *App Designer Overview Video*. [online] Available at: https://au.mathworks.com/videos/app-designer-overview-1510748719083.html [Accessed 6 Sep. 2023].
3. au.mathworks.com. (n.d.). *Building MATLAB Apps with App Designer Video*. [online] Available at: https://au.mathworks.com/videos/matlab-and-simulink-robotics-arena-building-apps-with-matlab-and-app-designer-1513378634144.html?s_tid=vid_pers_recs [Accessed 6 Sep. 2023].
4. au.mathworks.com. (n.d.). *Cell Array, Table, Timetable, Struct, or Dictionary? Choosing a Container Type Video*. [online] Available at: https://au.mathworks.com/videos/cell-array-table-timetable-struct-or-dictionary-choosing-a-container-type-1683356755755.html?s_tid=srchtitle_videos_main_8_dictionary [Accessed 6 Sep. 2023].
5. au.mathworks.com. (n.d.). *How to Build a GUI in MATLAB using App Designer - Video*. [online] Available at: https://au.mathworks.com/videos/gui-building-in-matlab-97169.html?s_tid=srchtitle_videos_main_10_App%2520Designer [Accessed 6 Sep. 2023].
6. au.mathworks.com. (n.d.). *Introduction to Dictionaries in MATLAB Video*. [online] Available at: https://au.mathworks.com/videos/introduction-to-dictionaries-in-matlab-1663800706532.html?s_tid=srchtitle_videos_main_2_dictionary [Accessed 6 Sep. 2023].
7. au.mathworks.com. (n.d.). *Pause on Error with MATLAB App Designer Video*. [online] Available at: https://au.mathworks.com/videos/pause-on-error-with-matlab-app-designer-1664822531898.html?s_tid=srchtitle_videos_main_4_App%20Designer [Accessed 6 Sep. 2023].

## Appendix:

# Project Self-Assessment

Initial Idea: Arithmetic study with quests
- Other initial ideas included in repository
- Changed due to functionality issues

Final Idea: French learning application

Progress 1: Loading screen with a 'translator' button
↳ English words could be translated to the 50 most common French words

# initial ideas code & files in repository

Progress 2: Home screen added with a new 'start' button.
↳ 'Translator' button moved to the home screen

Progress 3: Added a new 'dictionary.txt' file containing significant amounts of translation.
↳ Utilised MATLAB functions to read the text file & subsequently greater translation was enabled.

Progress 4: Formulated a text animation for the loading screen & changed text, background & button colours to match French flag colours:

Progress 5: 'Study Music' feature added to the home screen with a runtime label showing runtime of the .wav file.
↳ lofi music used due to my own love for it

Progress 6: 'Exit' button & additional labels such as, 'Developed by', 'Bienvenue!' & volume slider added to let users toggle volume for study music.

Progress 7: 'Challenge' mode added initially mimicking the translation buttons functionality.

Progress 8: 'Challenge' mode changed to allow for streaks & word guessing.
↳ Levels feature added with different text files.
↳ Level 1: 50 most common words
↳ Level 2: Next 100 most common words
↳ Level 3: 100 far harder words
↳ 'Streaks' & responses added to user guesses.
↳ Levels increment when user gets 5 in a row also implemented.

Progress 9: 'Story' mode added with 3 graded readers:
- Shorlock Holmes
- Baskerville Hall
- Fixing the Nets.

Progress 10: 'Select your story' page added mimicking 'Select your level'.
↳ Extra exit buttons added to streamline functionality.

# Rubric

Conceptual Coverage:

- Input/output:
  - The application is essentially all input & output
  - Buttons, sliders & text box's (translate/challenge) functions all take user input & produce output.
  - The output is greatly dependent on the function.
- Loops:
  - ↳ for loops, while loops
  - A combination of for loops were included.
  - While loops weren't used primarily due to their issues with GUI:
    - Blocking & State management.
- Vectors/Matrices:
  - ↳ As well as: cell arrays, logical vectors (boolean) & arrays
  - The primary vectors/matrices used within code for manipulating & sorting string data. Often in the parsing mode by the translation/challenge functions in the application.
  - Matrices are also often used for position & colour content. ie: [100, 100, 600, 400] for the challenge UI.
  - Cell arrays were also often used, ie for:
    - french words {}
    - english translations {}
    - ↓
    - Appending to cell arrays was also executed:
      - frenchwords{end+1} = parts{1};
- Conditional Execution:
  - ↳ switch, if, else, else if
  - Switch, if, else & else if conditional statements were all used several times:
    - Switch: used for levels in challenge mode.
    - If & else: used several times for example:
      - line 710-715 for exiting out of challenge
    - If, else & elseif: story type, line 317-323.
- Functions:
  - 30+ functions made with their own files for each.
  - Vital to the application for UI creation, interactivity & navigation.

# Review

- I initially had the idea to make an arithmetic quest game but due to drawbacks with functionality & a desire to use MATLAB's appdesigner I elected to develop something I would actually use.
- Incremental development was utilised as illustrated above
- 30+ functions & several more assets used
- Test cases ran with every progress in development

Value Add:

- functionality:
  - GUI & appdesigner used to add substantial value with great performance & execution
  - It can be used for any French learner such that they find value in all three main modes.
  - Text based animation & sounds adhering to the app's core purpose also execute perfectly
- Extensions:
  - Multiple levels of challenge
  - Dynamic content loading
  - Streak & error tracking with conditional level placement
  - Error handling, making the app more user friendly

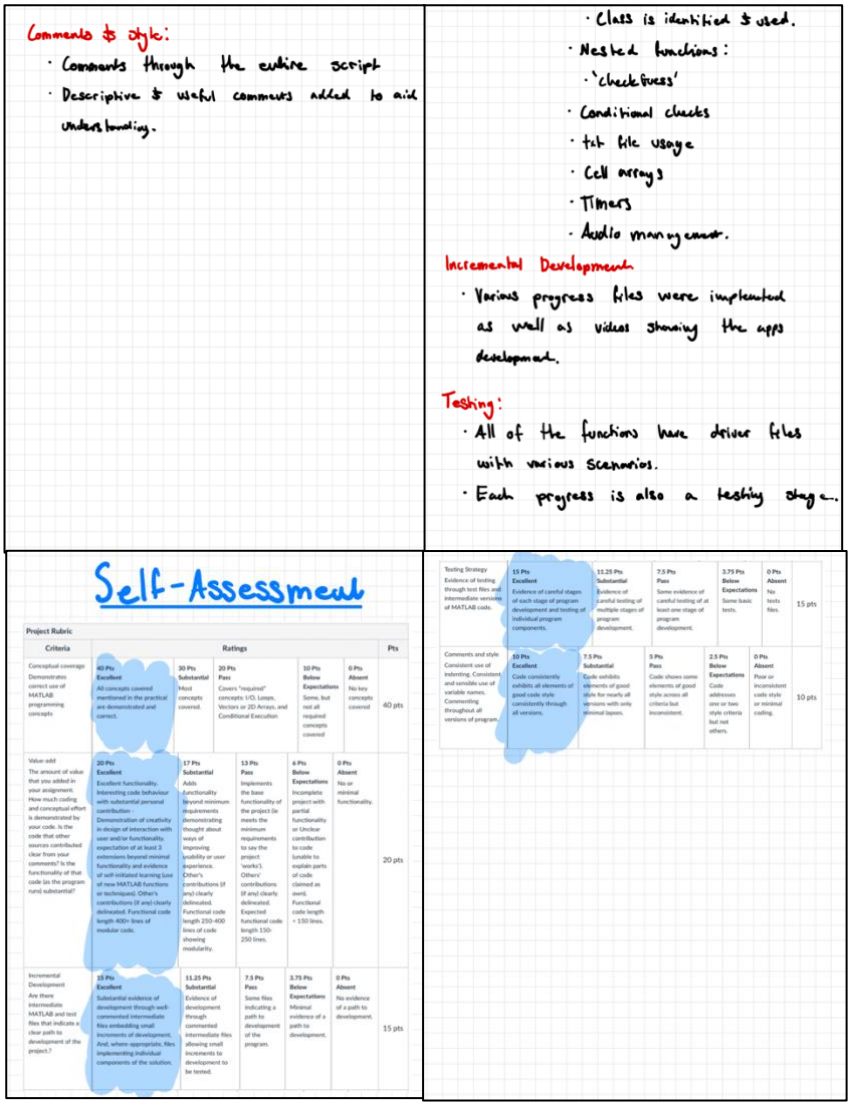


Figure 2.1: Self-Assessment PDF taken throughout the development of the application. The document acted as a tool to evaluate the progress while crafting the application (progress notes are generalised to the time of writing).