Project Walkthrough

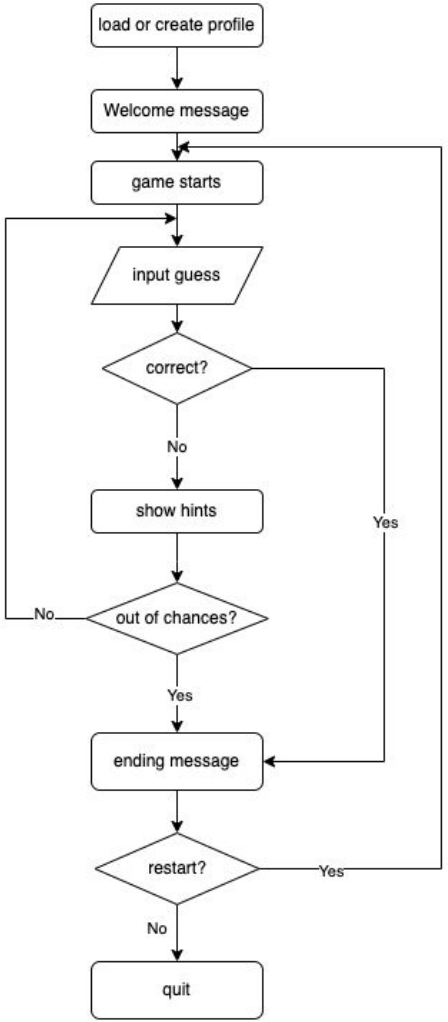
Wenxuan Pan T1A3

Outline

- Application walkthrough
  - Overall structure
  - Main features
  - Demo
- Code overview
  - Important parts
  - Application logic
- Development process
  - Plan
  - Challenges and takeaways

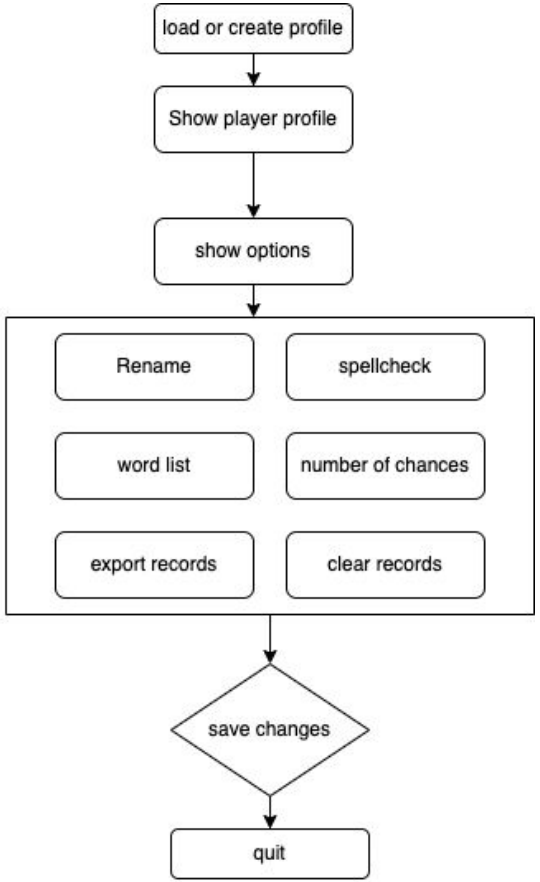
Application - flow

- Play
  - First time user: enter name and create a new profile
    - Existing user: loads data
  - Welcome message
  - Guess a word
  - Receive hints and continue guessing
  - Correct guess or out of chances:
    - game ends, showing success/failure message
  - Export records
  - Play again



Application - flow

- Settings
  - View player profile
  - A list of commands to choose from
    - rename, select word list, toggle spell-check etc.
  - Once happy: save and quit / quit without save



Application - features

- MVP features
  - Generate a random word from a list of words
  - Takes and validate user input
  - Analyse user's guess and show highlights
  - Show ending messages and restart/quit
- Bonus features
  - Export records as txt file (two ways)
  - Customise player settings
    - Rename, select word list, set number of chances, spellcheck
  - Save and load data

App walkthrough

Play - Welcome

```
Enter your name to create a profile:
jflkdajf adlkfj akldf jdklaj flkajd fkl ajklj flkajfd
Name should be 1-15 characters long. Please try again.
Enter your name to create a profile:
```

```
Enter your name to create a profile:
John Doe
        _       ______  ____  ____      ____  __________
       | |     / / __ \/ __ \/ __ \    / __ \/  _/ ____/
       | | /| / / / / / /_/ / / / /   / /_/ // // __/
       | |/ |/ / /_/ / _, _/ /_/ /   / ____// // /___
       |__/|__/\____/_/ |_/_____/   /_/    /___/_____/


  Hello, John Doe! Welcome to WORD PIE, a word guessing game.

  Mr. Python is hungry, and all he wants is that delicious but Poisonous Pie.
  He has to cast the correct healing spell to swallow it safely!
  Can you help him find the secret word in 6 attempts?
  Friendly Fairy agrees to reveal hints as you go along:
    - Green means the letter is in the secret word and at the same place.
    - Yellow means the letter is in the secret word but at the wrong place.
    - Grey means the letter is not in the secret word.

**Type '\r' to restart and '\q' to quit at any time of the game.**

==========================================
(Round: 1/6                SpellCheck: ON)

Take a guess: (5 letters)
```

Guess the word

```
Take a guess: (5 letters)
a
Input not valid. Please enter a 5-letter English word

Take a guess: (5 letters)
aakkk
Friendly Fairy warns you that the word is not in the dictionary. Try another word!
```

```
Mr. Python comes closer to the Poisonous Pie.

      A  P  P  L  E

==========================================
(Round: 2/6                SpellCheck: ON)

Take a guess: (5 letters)
apple
You already tried this word!
```

```
==========================================
(Round: 1/6                SpellCheck: ON)

Take a guess: (5 letters)
audio

Mr. Python comes closer to the Poisonous Pie.

      A  U  D  I  O

==========================================
(Round: 2/6                SpellCheck: ON)

Take a guess: (5 letters)
block

Poisonous Pie is starting to turn purple.

      A  U  D  I  O
      B  L  O  C  K

==========================================
(Round: 3/6                SpellCheck: ON)

Take a guess: (5 letters)
globe

Mr. Python comes closer to the Poisonous Pie.

      A  U  D  I  O
      B  L  O  C  K
      G  L  O  B  E

==========================================
(Round: 4/6                SpellCheck: ON)

Take a guess: (5 letters)
```

Win

```
==========================================
(Round: 4/6                SpellCheck: ON)

Take a guess: (5 letters)
elbow

Mr. Python thanks the Friendly Fairy.
```

A	U	D	I	O
B	L	O	C	K
G	L	O	B	E
E	L	B	O	W

```
==========================================
(Round: 5/6                SpellCheck: ON)

Take a guess: (5 letters)
noble
The spell works! Yum yum. Mr. Python was really happy.

**Progress auto saved. Head to user_data/save_data.json to copy backups.**
Enter '\s' to export current round as txt and start a new game.
Enter '\q' to quit.
Enter any other button to start a new game.
```

Lose

```
==========================================
(Round: 6/6                SpellCheck: ON)

Take a guess: (5 letters)
amber

Mr. Python thanks the Friendly Fairy.
```

H	E	L	L	O
A	M	E	N	D
A	G	E	N	T
A	S	I	D	E
A	S	S	E	T
A	M	B	E	R

```
You've run out of chances! The secret word is AFTER.
Friendly Fairy shrugged and flew away.


**Progress auto saved. Head to user_data/save_data.json to copy backups.**
Enter '\s' to export current round as txt and start a new game.
Enter '\q' to quit.
Enter any other button to start a new game.
```

Export record

```
**Progress auto saved. Head to user_data/save_data.json to copy backups.**
Enter '\s' to export current round as txt and start a new game.
Enter '\q' to quit.
Enter any other button to start a new game.
\s
Record exported! You can find it in user_data/record_20230513205726.txt
========================================
(Round: 1/6                SpellCheck: ON)
```

record_20230513205726.txt

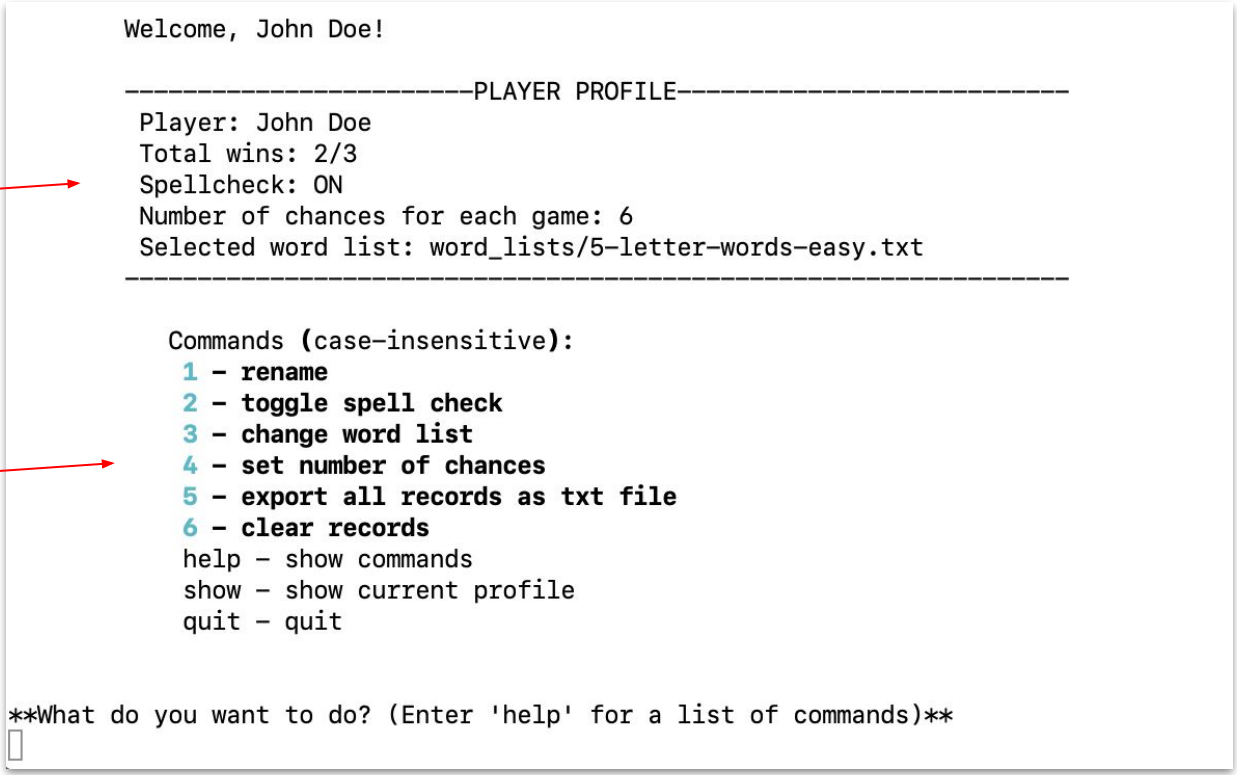
```
Start time: 2023-05-13 20:57:26
==================
  | H E L L O |
  | A M E N D |
  | A G E N T |
  | A S I D E |
  | A S S E T |
  | A M B E R |
==================
CORRECT WORD IS: AFTER
```

Quit and restart anytime

Type '\r' to restart and '\q' to quit at any time of the game.

```
========================================
(Round: 2/6                SpellCheck: ON)

Take a guess: (5 letters)
\r
========================================
(Round: 1/6                SpellCheck: ON)

Take a guess: (5 letters)
apple

Friendly Fairy looks at you encouragingly.

      A P P L E

========================================
(Round: 2/6                SpellCheck: ON)

Take a guess: (5 letters)
\q
Mr. Python seems disappointed. He hopes to see you soon!
wenxuanp@Wenxuan src %
```

Settings - profile and commands



Change settings

```
**What do you want to do? (Enter 'help' for a list of commands)**
1
Enter your new name:
Doe John
Renamed successfully! You are now called Doe John
```

1 - rename

```
**What do you want to do? (Enter 'help' for a list of commands)**
2
Spell check setting is now OFF.

**What do you want to do? (Enter 'help' for a list of commands)**
2
Spell check setting is now ON.
```

2- toggle spell check

3 - select word list

```
**What do you want to do? (Enter 'help' for a list of commands)**
3
       Selecting word list -
       You can upload custom txt files to the word_lists folder.
       (Note: the file should contain words longer than 2 characters)
       Current options:

0 - 5-letter-words-all.txt
1 - demo.txt
2 - first-names.txt
3 - 7-letter-words-common.txt
4 - 5-letter-words-easy.txt
please enter the file number you want to use (0 to 4)

0
Word list now set to word_lists/5-letter-words-all.txt
Note: toggle off spell check if the words cannot be found in dictionary.
```

Change settings

4 - set number of chances

```
**What do you want to do? (Enter 'help' for a list of commands)**
4
Enter number of chances:
10
You now have 10 attempts.
```

```
**What do you want to do? (Enter 'help' for a list of commands)**
6
Clearing all records? Type 'Y' to confirm.
y
Records cleared.
```

6 - clear all records

quit - exit settings
Y - confirm changes

```
**What do you want to do? (Enter 'help' for a list of commands)**
quit

**Save changes before quit?
Type 'Y' to save, type any other buttons to discard all changes.**
y
**Changes saved.**

See you next time, Doe John!
```

Export player records

```
**What do you want to do? (Enter 'help' for a list of commands)**
5
Record exported! You can find it in user_data/record_Doe John.txt
```

record_Doe John.txt

```
Start time: 2023-05-13 20:51:05
================
  | W A T E R |
  | N E C K S |
  | W A G O N |
================
CORRECT WORD IS: WAGON

Start time: 2023-05-13 20:52:53
================
  | A U D I O |
  | B L O C K |
  | G L O B E |
  | E L B O W |
  | N O B L E |
================
CORRECT WORD IS: NOBLE

Start time: 2023-05-13 20:57:26
================
  | H E L L O |
  | A M E N D |
  | A G E N T |
  | A S I D E |
  | A S S E T |
  | A M B E R |
================
CORRECT WORD IS: AFTER
```

Code overview

Packages and modules

Packages

- **rich, pyspellchecker, pytest**
- **random, datetime, sys, os, json**

Internal modules

- **main.py**
- **guess.py** - game's core features
- **player.py** - Player class and function to validate player
- **player_setting.py** - update player information
- **story.py** - narrative text
- **helper.py** - helper functions

Main

```
from guess import play_loop
from player_setting import setting
from sys import argv

if __name__ == "__main__":
    try:
        if argv[1] == "play":
            play_loop()
        elif argv[1] == "settings":
            setting()
        else:
            print("Invalid command. "
                  "Please enter 'play' or 'settings' and retry.")
    # catching error when user open py file directly
    except IndexError:
        print("Not enough arguments. "
              "Please follow the help doc and try again.")
```

Play and setting

```python
def play_loop():
    """Main play"""
    player = create_player()
    # display welcome message
    player_name = player.get_name()
    player_chances = player.get_num_chances()
    print_welcome(player_name, player_chances)
    try:
        # generate word list based on txt file
        list_path = player.get_list_path()
        word_list = get_word_list(list_path)
        # main play loop
        while True:
            try:
                play_once(player, word_list)
            # when user uses \r, raise exception to jump out
            # of current play round and restart from the beginning
            except StartAgainException:
                continue
    # exit the game when user raises keyboard interrupt (ctrl+c and \q)
    except KeyboardInterrupt:
        print(
            "[reverse]Mr. Python seems disappointed. "
            "He hopes to see you soon![/reverse]")
```

```python
def setting():
    """main settings"""
    # create a player object
    player = create_player()
    # get and display player name
    name = player.get_name()
    print(f"\n        Welcome, {name}!")
    # show player profile
    player.show_status()
    # show a list of commands
    show_options()
    # ask for user input
    change_settings_loop(player)
    # when loop ends, display quit message
    print(f"See you next time, {name}!")
```

Functions and variables

```python
def check_letter(answer, guess, word_length):
    """analyse user's guess and compare letter by letter"""
    # set up result array, 0 = wrong, 1 = misplaced, 2 = correct
    result = [0] * word_length
    # convert answer string to a list, to compare both letter and order
    answer_list = list(answer)
    # round 1: check for correct letters
    for index, letter in enumerate(guess):
        if letter == answer_list[index]:
            result[index] = 2
            # correct letter will be reset to 0
            # in the answer list to avoid duplicate match
            answer_list[index] = 0
    # round 2: check for misplaced letters
    for index, letter in enumerate(guess):
        if letter in answer_list:
            # avoid overwriting correct result
            if result[index] != 2:
                result[index] = 1
                # find the index of answer letter and
                # reset that to 0 to avoid duplicate match
                answer_list[answer_list.index(letter)] = 0
    # return result array ready for highlighting
    return result
```

```python
for i in range(1, num_chances+1, 1):
    print(
        "=========================================\n"
        f"(Round: {i}/{num_chances}"
        "              SpellCheck: "
        f"{player.display_spell_check_status()})\n")
    # get a valid word for analysis
    guess = check_input_word(guessed_list, word_length)
    # add the word to guessed list
    guessed_list.append(guess)
    # if guess matches answer, show win message and end loop
    if check_exact_match(answer, guess):
        print(
            "[italic reverse]The spell works! "
            f"{random.choice(win_messages)}[/italic reverse]")
        break
    # if not won, compare and show hints
    else:
        result = check_letter(answer, guess, word_length)
        hints += highlight_word(guess, result)
        print(
            f"\n[italic]{random.choice(hint_messages)}[/italic]\n{hints}")
```

Class

```python
class Player():
    """Player profile class.

    A class that stores player attributes,
    including player name, spell check status,
    number of chances, selected word list, and
    game records.

    Methods including getting, setting and toggling
    these attributes, as well as importing and exporting
    data using json.
    """

    def __init__(self):
        """Class constructor. Default values
        are set when instances are created to
        prevent save file corruption
        """
        self.name = "default_user"
        self.spell_check_enabled = True
        self.num_chances = 6
        self.list_path = "word_lists/5-letter-words-easy.txt"
        self.records = []
```

```python
def display_spell_check_status(self):
    """return spell check setting as user-friendly string"""
    return "ON" if self.spell_check_enabled else "OFF"

def toggle_spell_check_enabled(self):
    """toggle spell check on or off"""
    self.spell_check_enabled = not self.spell_check_enabled
    print(
        "Spell check setting is now "
        f"{self.display_spell_check_status()}.")
```

```python
def create_player():
    """this function will create a player object and
    1. try to load data from existing save file
    2. If no data is found, it will ask the user to
    enter name, and all the other attributes will be
    using default values. It will then export a copy of
    save data file
    """
    player = Player()
    try:
        player.load_data()
    except FileNotFoundError:
        player.create_new_save()
    return player
```

Conditions and loops

```python
match prompt:
    case "1":
        player.rename()
    case "2":
        player.toggle_spell_check_enabled()
    case "3":
        player.set_list_path()
    case "4":
        player.set_num_chances()
    case "5":
        player.export_records_all()
    case "6":
        player.clear_records()
    case "HELP":
        show_options()
    case "SHOW":
        player.show_status()
    case "QUIT":
```

```python
def check_exact_match(answer, guess):
    """check if the user's guess is the same as answer"""
    if guess.upper() == answer.upper():
        return True
    else:
        return False
```

```python
for i in range(1, num_chances+1, 1):
    print(
        "=========================================\n"
        f"(Round: {i}/{num_chances}"
        "                SpellCheck: "
        f"{player.display_spell_check_status()})\n")
    # get a valid word for analysis
    guess = check_input_word(guessed_list, word_length)
    # add the word to guessed list
    guessed_list.append(guess)
    # if guess matches answer, show win message and end loop
    if check_exact_match(answer, guess):
        print(
            "[italic reverse]The spell works! "
            f"{random.choice(win_messages)}[/italic reverse]")
        break
    # if not won, compare and show hints
    else:
        result = check_letter(answer, guess, word_length)
        hints += highlight_word(guess, result)
        print(
            f"\n[italic]{random.choice(hint_messages)}[/italic]\n{hints}")
# if loop ends without breaking, display losing message
else:
    print(
        "[italic reverse]You've run out of chances! "
        f"The secret word is [bold]{answer}[/bold].\n"
        f"{random.choice(lose_messages)}[/italic reverse]\n"
    )
```

Error handling

```
try:
    player.load_data()
except FileNotFoundError:
    player.create_new_save()
return player
```

```
try:
    if argv[1] == "play":
        play_loop()
    elif argv[1] == "settings":
        setting()
    else:
        print("Invalid command. "
              "Please enter 'play' or 'settings' and retry.")
# catching error when user open py file directly
except IndexError:
    print("Not enough arguments. "
          "Please follow the help doc and try again.")
```

```
try:
    with open(f"user_data/save_data.json") as f:
        save = json.load(f)
        self.name = save["name"]
        self.spell_check_enabled = save["spell_check_enabled"]
        self.num_chances = save["num_chances"]
        self.list_path = save["list_path"]
        self.records = save["records"]
# error handling for corrupted file
except KeyError and json.decoder.JSONDecodeError:
    print_red(
        "WARNING: Save file corrupted."
        "Upload backups to replace user_data/save_data.json,"
        "or default settings will be used.")
```

```
# quit game
if user_input.upper() == "\\Q":
    raise KeyboardInterrupt

# restart game
elif user_input.upper() == "\\R":
    raise StartAgainException
```

```
try:
    if int(num) > 0:
        self.num_chances = int(num)
        print(f"You now have {self.num_chances} attempts.")
        return
    # if input is integer but not positive, raise error
    else:
        raise ValueError
# if input is not positive integer, go back to loop start
except (TypeError, ValueError):
    print("Please input a positive whole number. Try again.")
```

```
try:
    # main play loop
    while True:
        try:
            play_once()
        # when user uses \r, raise exception to jump out
        # of current play round and restart from the beginning
        except StartAgainException:
            continue
# exit the game when user raises keyboard interrupt (ctrl+c and \q)
except KeyboardInterrupt:
    print(
        "[reverse]Mr. Python seems disappointed. "
        "He hopes to see you soon![/reverse]")
```

Validating user input

Loops

Conditions

Error handling

```
def set_num_chances(self):
    """set number of chances available per game play"""
    # loop and validate if input is a whole number
    while True:
        num = input("Enter number of chances:\n")
        # if input is positive integer, set number
        try:
            if int(num) > 0:
                self.num_chances = int(num)
                print(f"You now have {self.num_chances} attempts.")
                return
            # if input is integer but not positive, raise error
            else:
                raise ValueError
        # if input is not positive integer, go back to loop start
        except (TypeError, ValueError):
            print("Please input a positive whole number. Try again.")
```

```
while True:
    # convert to uppercase to compare with previous results
    guess = take_input(f"Take a guess: ({word_length} letters)\n").upper()
    # spellcheck the guess
    spell = SpellChecker()
    misspelled = bool(spell.unknown([guess]))
    # check if word is already guessed
    if guess in guessed_list:
        print("You already tried this word!\n")
    # check if input is an English word and the same length as answer
    elif not guess.isalpha() or len(guess) != word_length:
        print(
            f"Input not valid. "
            f"Please enter a {word_length}-letter English word\n")
    # if spell check enabled, check if it is misspelled
    elif player.get_spell_check_enabled() and misspelled:
        print(
            "Friendly Fairy warns you that the word "
            "is [bold]not in the dictionary[/bold]. "
            "Try another word!\n")
    # return guessed word if all validation passed
    else:
        return guess
```

```
def set_valid_name(self, prompt):
    """validate user input and set player name"""
    while True:
        name = input(prompt)
        # input 1-15 characters long is accepted
        if len(name) <= 15 and len(name) > 0:
            self.name = name
            return
        # print error message if word limits not met
        else:
            print("Name should be 1-15 characters long. Please try again.")
```

Testing

```python
def test_get_random_word_empty():
    """Tested function: get_random_word()
    test that empty list will display error message
    """
    word_list = []
    with pytest.raises(KeyboardInterrupt):
        guess.get_random_word(word_list)
```

```python
def test_get_random_word_valid():
    """Tested function: get_random_word()
    test that a random word can be drawn from a valid word list
    """
    word_list = ["apple", "water", "faint", "quiet", "33", "a"]
    word = guess.get_random_word(word_list)
    assert word.isalpha()
    assert word.lower() in [x.lower() for x in word_list]
```

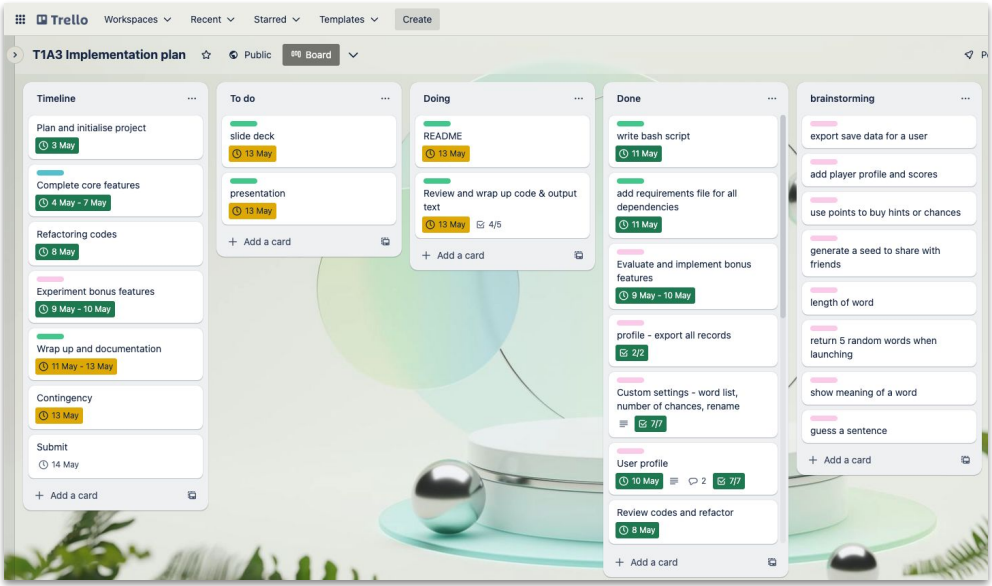
```python
def test_check_letter_duplicate():
    """Tested function: check_letter()
    Test some more complex cases, where there are duplicate
    letters. For example for the answer "eaten", if guess is
    "lever", the first "e" will be misplaced, while the second
    "e" will be correct. The order of the results must not be
    swapped.
    In answer "apple" and guess "puppy", the third "p" will be
    wrong because there are already two "p"s.
    """
    assert guess.check_letter(
        "eaten", "lever", 5) == [0, 1, 0, 2, 0]
    assert guess.check_letter(
        "apple", "puple", 5) == [1, 0, 2, 2, 2]
    assert guess.check_letter(
        "apple", "puppy", 5) == [1, 0, 2, 0, 0]
    assert guess.check_letter(
        "tight", "fight", 5) == [0, 2, 2, 2, 2]
    assert guess.check_letter(
        "elude", "ledge", 5) == [1, 1, 1, 0, 2]
```

Development process

Plan

```
WELCOME to the game! Have a guess:

=================
  | - - - - - |
  | - - - - - |
  | - - - - - |
  | - - - - - |
  | - - - - - |
  | - - - - - |
=================

=================
  | S U G A R |
  | - - - - - |
  | - - - - - |
  | - - - - - |
  | - - - - - |
  | - - - - - |
=================

You have 5 more guesses!

ABCDE is not a valid word. Please try another one.

Have a guess:

Congratulation! You've guessed the correct word.

Would you like to save your progress?
1. save this round only
2. save all progress so far
3. start again
```

- Trello
- Test Driven Development
- PEP 8 Style guide
- DRY principle



Challenges and takeaways

- Logic
- Class
- File handling
- Project management

Thanks for watching!