KEIRAN HOWARD

RUBY TERMINAL APP

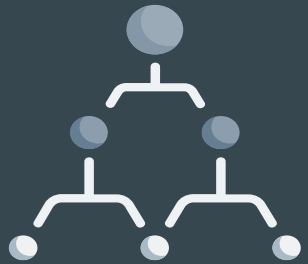
HANGMAN DESIGN PLAN

01 Project management

Create a detailed plan using Trello boards

02 Process planning

Pseudocode and Flowcharts

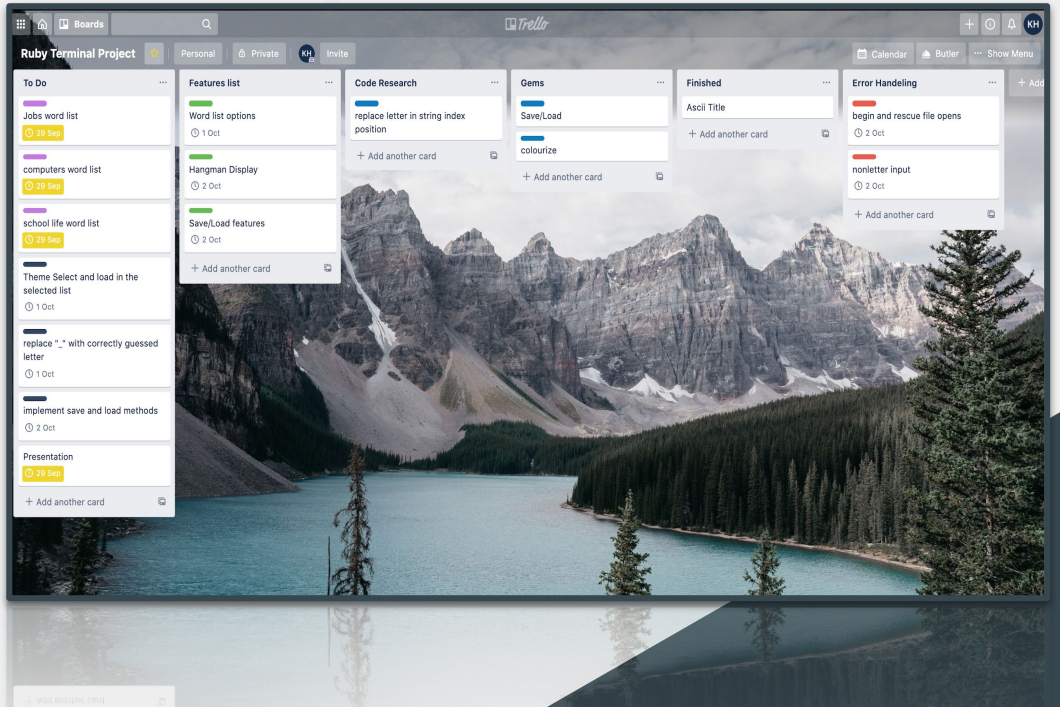


03 Code

Writing code and implementing gems

PROJECT MANAGEMENT

Trello Board

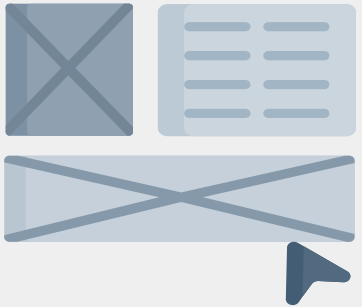


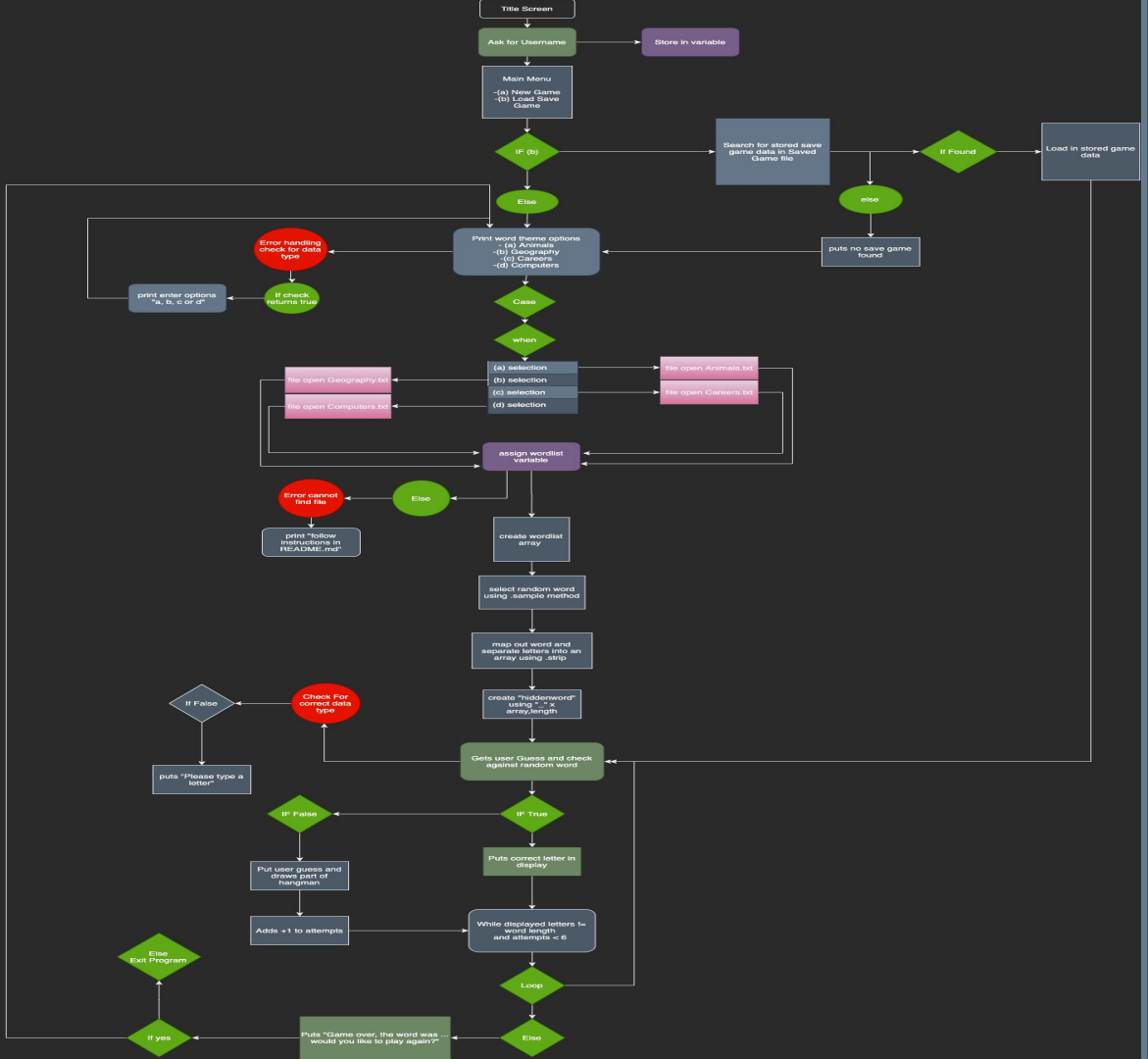
pseudocode.rb

```
prints loading screen

ask for username

gets username

if username entered begin game

get random word from questions

slice text to individual chars

asks for user guess and match against word

if user guesses a letter correctly add letters to board

else add  + 1 to attempts and draw charecter head

repeat until character is made or word is complete

if character is draw
    print "You lost"
    print "The word was ..."

if word is completed
    print "You Won!"

prints "Would you like to play again? type yes or no"

if user types yes
    repeat all

if user types no
    exit application



    make a class for menu

    and class for game
```

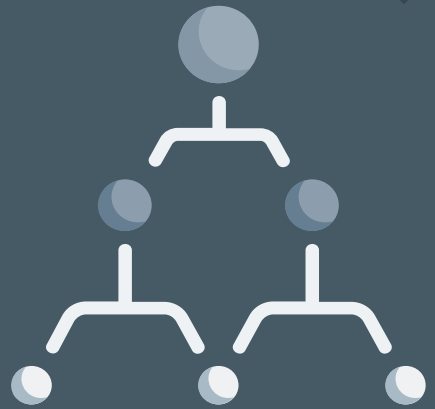
Pseudo Code

PLANNING PROCESS




Title Screen
Ask for Username
Store in variable
Main Menu
-(a) New Game
-(b) Load Save Game
IF (b)
Search for stored save game data in Saved Game file
If Found
Load in stored game data
else
puts no save game found
Else
Print word theme options
- (a) Animals
-(b) Geography
-(c) Careers
-(d) Computers
Error handling check for data type
If check returns true
print enter options "a, b, c or d"
Case
when
(a) selection
(b) selection
(c) selection
(d) selection
file open Geography.txt
file open Computers.txt
file open Animals.txt
file open Careers.txt
assign wordlist variable
Error cannot find file
Else
print "follow instructions in README.md"
create wordlist array
select random word using .sample method
map out word and separate letters into an array using .strip
create "hiddenword" using " " x array.length
If False
Check For correct data type
puts "Please type a letter"
Gets user Guess and check against random word
IF False
IF True
Put user guess and draws part of hangman
Puts correct letter in display
Adds +1 to attempts
While displayed letters != word length and attempts < 6
Else Exit Program
Loop
If yes
Puts "Game over, the word was ... would you like to play again?"
Else

DESIGN FLOW



USER INTERACTION

1. Input username
2. Choose new game or load saved game
3. Select a word list theme from the presented choices
4. Guess a letter until win or lose
5. Choose between new game or exit application

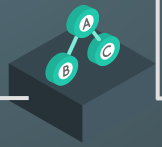
CODE

STAGE I

Code all basic functions of the application and design control flow.

STAGE 2

Implement Ruby gems and develop features.



STAGE 3

Run code tests and develop error handling methods.

Features

Word Theme

- User can choose between 4 word list.
- Animals, Geography, Careers and Computer.
- Application will pick a random word from selected list

```
wordlist = File.open("animals.txt")
wordlist = wordlist.to_a
word = wordlist.map {|word| word.upcase.strip}
secretword = word.sample
```

Save/Load Game

- Ability to save current game or load from a pre existing save state
- Utilizes Ruby gem “save_game” version 1.031
- Save states will be read and written into a relative text file then translated back to ruby code

Hangman Display

- Created in Ascii
- Sections will be displayed incrementally with each incorrect guess
- Once the image has been completed the application prompts user to initialize a new game and display hidden word

Concept art-

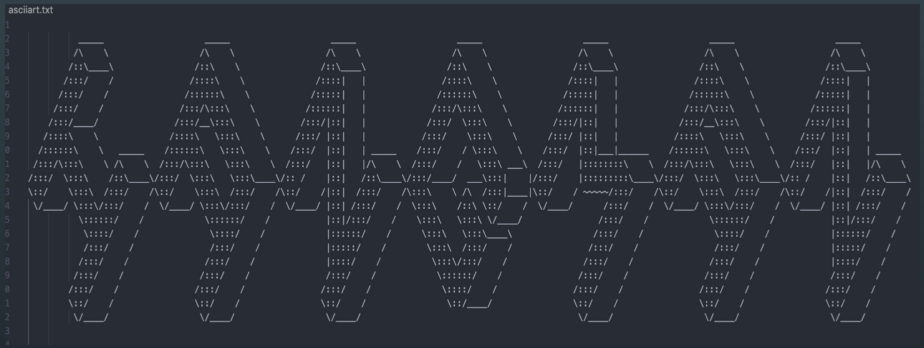
CONCEPT ART

2D Concept

```
# when attemps == 3 puts

#     "    --------   "
#     "    |      |   "
#     "    |      0   "
#     "    |     /|   "
#     "    |          "
#     "  -----------  "

# when attemps == 4 puts

#     "    --------   "
#     "    |      |   "
#     "    |      0   "
#     "    |     /|\  "
#     "    |          "
#     "  -----------  "

# when attemps == 5 puts

#     "    --------   "
#     "    |      |   "
#     "    |      0   "
#     "    |     /|\  "
#     "    |     /    "
#     "  -----------  "

# when attemps == 6 puts

#     "    --------   "
#     "    |      |   "
#     "    |      0   "
#     "    |     /|\  "
#     "    |     / \  "
#     "  -----------  "  + "Game over! the word was #{randomword}"
```

3D Concept

Title



ERROR HANDLING

Two main components requiring error exception handling

Loading in local files

Potential error. - unable to locate required files

- Handled by using BEGIN and RESCUE prompting user to refer to README documentation.

User Input

Potential error. - incorrect data type entered

- Handled by using creating a input check
- Create custom error message for incorrect data type