Mad Math Quiz

A Terminal Application – Written in Ruby version 3.0.3 by

Kim Stocker

Overview

★ Mad Math Quiz, its features and functionalities (with Video & Live demo).

★ Logic behind Mad Math Quiz and code (view on VSCode).

★ Project Management process (Trello Board)

★ Challenges, ethical issues and favourite parts.

Features & Functionalities.

Feature 1: The Quiz
- ❖ 10 randomly generated arithmetic operations questions.
- ❖ Users can input their answers.
- ❖ Scores will be tallied up and displayed to the user.

Feature 2: Difficulty Levels
There are 3 difficulty levels
- ❖ Level 1: Each question in this level will be randomly generated from 2 arithmetic operators (either +, -, *), 2 single digit numbers and 1 double digit number.
- ❖ Level 2: This level include random questions with 2 arithmetic operators (either +, -, *), 1 single digit number and 2 double digit numbers.
- ❖ Level 3: Each question in this level will contain 3 arithmetic operators (either +,-,* or **), 1 single digit number, 2 double digit numbers and 1 triple digit number.

Feature 3: Saving Scorecard
Once finished their quiz, users can print out their scorecard that list out all 10 questions, the correct answers, their answers and their scores for each questions as well as their overall score.

Code Overview & Logic

The main Object of this application is The Quiz & the Scoreboard.

Quiz class include the below methods:

- .ask_questions

- .create_scorecard

- .display_scorecard

- .save_scorecard

Scoreboard class include the below methods:

- .read_file

- .max_score

- .update_score

- .write_file

Project Management Processes

Planning

★ Project Documentation requirements
★ Code requirements
★ Idea >> Trello Board with minimum MVP features & checklists, good to have features and deadlines.
★ Github Repo & folder set up

Development

★ Start with the core feature/function and prioritised list items that due first.
★ Code/Manual
★ Test/Refactor
★ Error handling.
★ Update Trello board.
★ Automatic Test with Rspec
★ Refactor
★ Test again

Deployment

★ Packaging/prepare script to help & guide user to install the application.
★ Update portfolio website
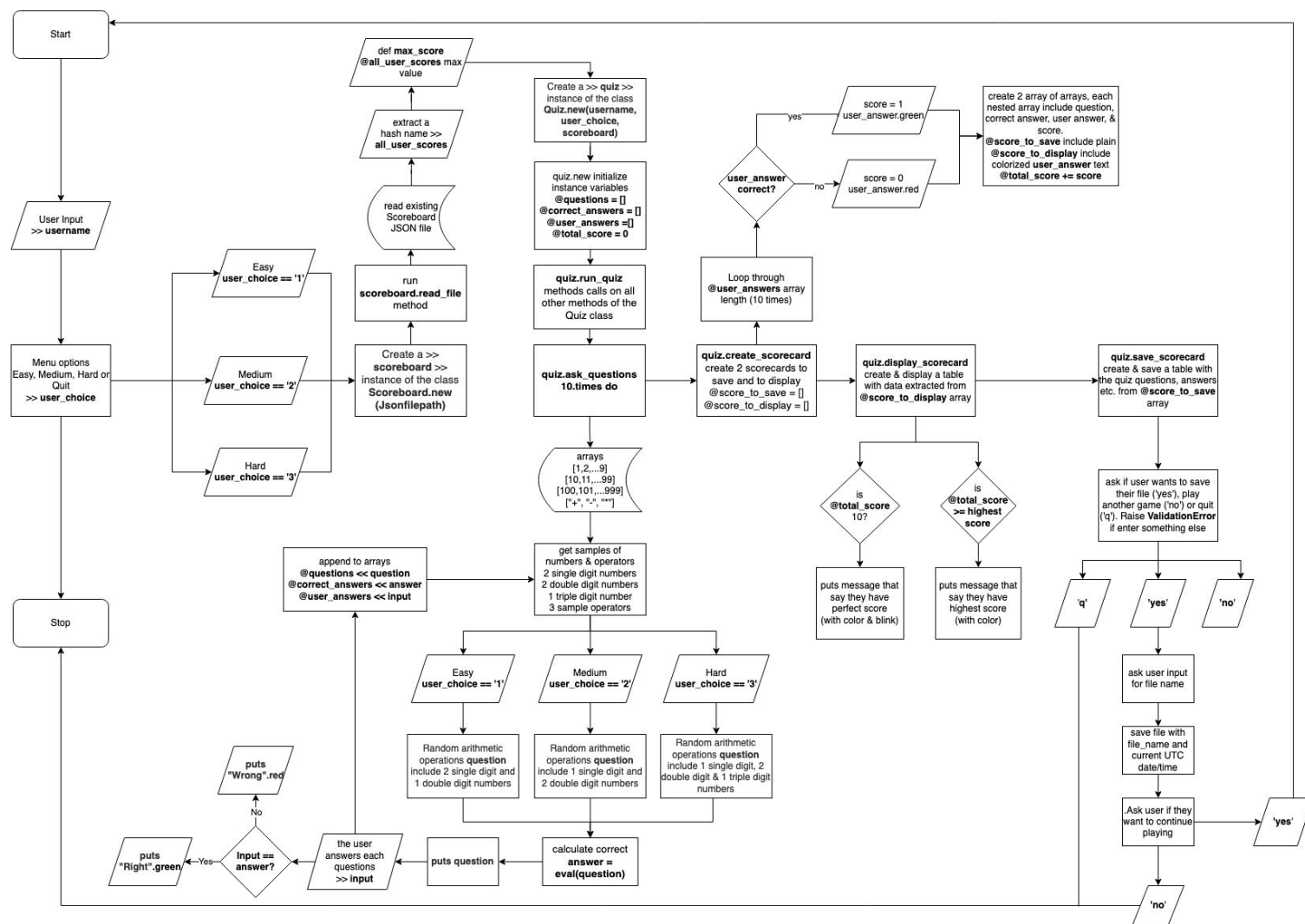★ Create a blog post on how it was developed.

Challenges

- ★ Planning
- ★ Use of Test Driven Development.

Favourites

- ★ Implementing Scoreboard Table with Terminal-table Gem.
- ★ Dynamic messaging when user hit the perfect score or on top of the leaderboard.

Ethical issues

- ★ Storing all users username and score.
- ★ Add disclaimer at the beginning of the quiz

Quiz Application Flowchart

```
Start
  │
  ▼
User Input >> username
  │
  ▼
Menu options
Easy, Medium, Hard or Quit
>> user_choice
  │
  ├──► Easy   user_choice == '1'
  ├──► Medium user_choice == '2'
  └──► Hard   user_choice == '3'
         │
         ▼
Create a >> scoreboard >> instance of the class
Scoreboard.new (Jsonfilepath)
         │
         ▼
run scoreboard.read_file method
         │
         ▼
read existing Scoreboard JSON file
         │
         ▼
extract a hash name >> all_user_scores
         │
         ▼
def max_score @all_user_scores max value
         │
         ▼
Create a >> quiz >> instance of the class
Quiz.new(username, user_choice, scoreboard)
         │
         ▼
quiz.new initialize instance variables
@questions = []
@correct_answers = []
@user_answers =[]
@total_score = 0
         │
         ▼
quiz.run_quiz methods calls on all other methods of the Quiz class
         │
         ▼
quiz.ask_questions 10.times do
         │
         ▼
arrays
[1,2,...9]
[10,11,...99]
[100,101,...999]
["+", "-", "*"]
         │
         ▼
get samples of numbers & operators
2 single digit numbers
2 double digit numbers
1 triple digit number
3 sample operators
         │
         ├──► Easy   user_choice == '1'
         │       │
         │       ▼
         │    Random arithmetic operations question include 2 single digit and 1 double digit numbers
         │
         ├──► Medium user_choice == '2'
         │       │
         │       ▼
         │    Random arithmetic operations question include 1 single digit and 2 double digit numbers
         │
         └──► Hard   user_choice == '3'
                 │
                 ▼
              Random arithmetic operations question include 1 single digit, 2 double digit & 1 triple digit numbers
                 │
                 ▼
calculate correct answer = eval(question)
                 │
                 ▼
puts question
                 │
                 ▼
the user answers each questions >> input
                 │
                 ▼
Input == answer?
   ├── Yes ──► puts "Right".green
   └── No  ──► puts "Wrong".red
                 │
                 ▼
append to arrays
@questions << question
@correct_answers << answer
@user_answers << input
```

```
user_answer correct?
   ├── yes ──► score = 1   user_answer.green
   └── no  ──► score = 0   user_answer.red
                 │
                 ▼
create 2 array of arrays, each nested array include question, correct answer, user answer, & score.
@score_to_save include plain
@score_to_display include colorized user_answer text
@total_score += score
```

```
Loop through @user_answers array length (10 times)
         │
         ▼
quiz.create_scorecard
create 2 scorecards to save and to display
@score_to_save = []
@score_to_display = []
         │
         ▼
quiz.display_scorecard
create & display a table with data extracted from @score_to_display array
         │
         ├──► is @total_score 10?
         │       │
         │       ▼
         │    puts message that say they have perfect score (with color & blink)
         │
         └──► is @total_score >= highest score
                 │
                 ▼
              puts message that say they have highest score (with color)
```

```
quiz.save_scorecard
create & save a table with the quiz questions, answers etc. from @score_to_save array
         │
         ▼
ask if user wants to save their file ('yes'), play another game ('no') or quit ('q'). Raise ValidationError if enter something else
         │
         ├──► 'q'
         ├──► 'yes'
         │       │
         │       ▼
         │    ask user input for file name
         │       │
         │       ▼
         │    save file with file_name and current UTC date/time
         │       │
         │       ▼
         │    .Ask user if they want to continue playing
         │       │
         │       ├──► 'yes'
         │       └──► 'no'
         │
         └──► 'no'

Stop
```

Thank You

Questions?