**Trump v Kanye Quiz CLI**

**Table of contents**

[1. Project Links](https://github.com/Finbob12/ToDoList-CLI#Project-Links)  
[2. What does ToDoList CLI do?](https://github.com/Finbob12/ToDoList-CLI#What-does-ToDoList-CLI-do?)  
[3. What problem does this app solve? and why am I developing it?](https://github.com/Finbob12/ToDoList-CLI#What-problem-does-this-app-solve?-Why-am-I-developing-it?)  
[4. Target audience and how they will use it](https://github.com/Finbob12/ToDoList-CLI#Target-audience-and-how-they-will-use-it)  
[5. ToDoList CLI Features](https://github.com/Finbob12/ToDoList-CLI#ToDoList-CLI-Features)  
[6. An outline of user interaction](https://github.com/Finbob12/ToDoList-CLI#An-outline-of-user-interaction)  
[7. Control flow diagram for the app](https://github.com/Finbob12/ToDoList-CLI#Control-flow-diagram-for-the-app)  
[8. Implementation plan - Trello](https://github.com/Finbob12/ToDoList-CLI#Implementation-plan---Trello)  
[9. System requirements](https://github.com/Finbob12/ToDoList-CLI#System-requirements)  
[10. How to install and run ToDoList CLI](https://github.com/Finbob12/ToDoList-CLI#How-to-install-and-run-ToDoList-CLI)

**Project Links**

[Github: ToDoList CLI](https://github.com/Finbob12/ToDoList-CLI)

Variables

Loops

Input and Output

**Develop** a statement of **purpose** and **scope** for your application. It must include:  
- **describe** at a high level what the application will do  
- **identify** the problem it will solve and **explain** why you are developing it  
- **identify** the target audience  
- **explain** how a member of the target audience will use it

**Purpose and scope of the application**

**What does Trump v Kayne Quiz CLI do?**

The Trump v Kanye Quiz is a command line quiz that provides the user with 15 multiple choice questions. Users are asked to enter their name, and the program refers to them as this name from then on. If they do not enter a value, they are prompted to re-enter their name. The instructions for the quiz are then displayed on the screen, with the design aiming to be clear and easy to read. The quiz then starts, with 15 quotes from Twitter and users have to identify the author – Trump or Kanye. If they get the correct answer, they receive a point are told their cumulative score. If they get it wrong, they are told their answer is incorrect.

At the end of the quiz, the user is given their score out of the total answers. They are then given the opportunity to double or nothing – to gamble all their points on the last question. They are then told their new score based on their answer to the final question.

**What problem does this app solve?**

This app doesn’t necessarily solve a problem - possibly boredom! It does improve user’s knowledge of the Donald Trump’s and Kanye West’s social media outputs; perhaps it’s a commentary on the state of politics and entertainment…

**Target audience**

This is a fun tongue-in-cheek app to entertain users and allow them to test their knowledge about popular culture figures. It is designed to be played by users with minimal to no knowledge of the command line, as they only have to answer simple prompts. Users who have never heard of Trump of Kanye can still play, as there are only two options to pick from so they still have a chance of being right.

Inputs are entered as ‘Name’, typing ‘Trump’ or ‘Kanye’ for each question (case insensitive) and then selecting Yes or No at end when deciding whether or not to gamble.

**Develop** a list of features that will be included in the application. It must include:  
- at least THREE **features**  
- **describe** each feature  
  
**Note:** **Ensure** that your features above allow you to demonstrate your understanding of the following language elements and concepts:  
- use of variables and the concept of variable scope  
- loops and conditional control structures  
- error handling  
  
**Consult with your educator** to check your features are sufficient .

**Trump v Kanye Features**

Some of the features of the app include asking users to input their name, using colour and boxes to improve the usability of the app and the UX, displaying the cumulative score as users move through the quiz and allowing users to gamble their points at the end.

Feature 1 – Name prompt

The first step of the app asks the user to enter their name, in order to engage them from the beginning and to personalise the user experience. The program then includes the user name in the intro text, and also at the end when telling them their score. Error handling is utlised in the name prompt. If the user presses enter without typing their name, they are given an error message “*Name cannot be empty*”. They are given three opportunities to enter their name – if a blank value is returned each of the three times, the program will exit.

NOTE ON ERROR HANDLING: A possible weakness in the app is that a user may mis-type ‘Trump’ or ‘Kanye’ and not get the points, though they did get the answer right (the code only allows for inputs to be rendered case insensitive). This could be mitigated by utlising TTY-prompt (as is done for the end question where users are only given the two options). However, having the user enter the answers increases their involvement and engagement with the game. When they are given instructions at the beginning of the game, they are warned to be careful of their spelling.

Feature 2 – Display output

Ruby gems have been used in the app to enhance readability and make the user experience positive. This includes use of coloured text, of bold and italic text stylings, boxes to separate the text to ensure instructions are read and a hyperlink gem to allow the user to visit the credited side directly from the terminal.

Feature 3 – Cumulative score total

The app also includes the feature of giving users a cumulative total of their points as they go through the quiz questions. To improve user experience, the message for an incorrect answer is shown in red, and the message for a correct answer is shown in green. If the answer is correct they are shown the total. This is done through setting the score to originally to be 0, looping through each question in the questions list and increasing the score by 1 when the answer is equal to the answer in the questions array. Once the method has been executed for all the questions, it moves on to the next stage of the program.

Feature 4 – Gamble your points

After the user is given their

**An outline of user interaction**

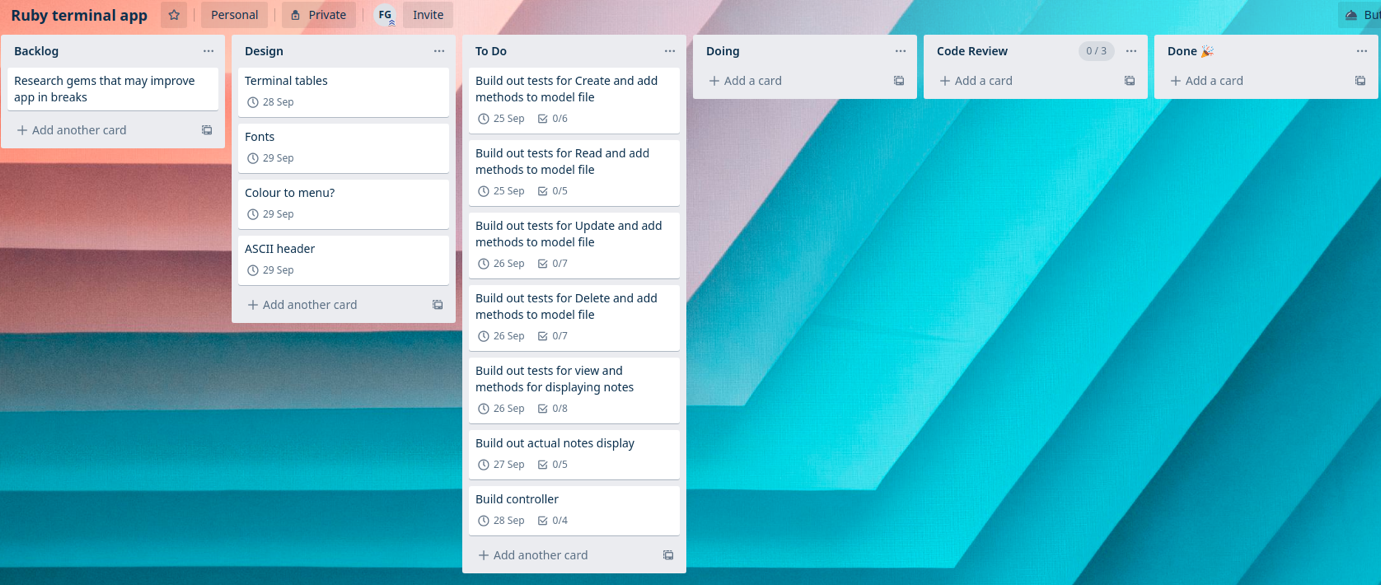
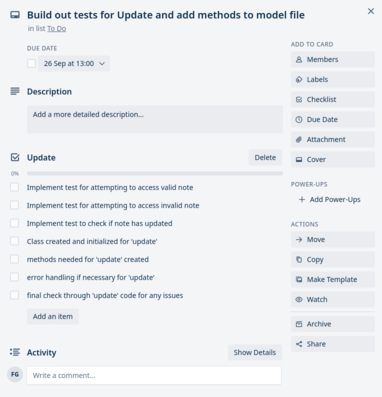
When the user first launches the application they will be greeted with a very simple, clear and consise menu that should be self-explantory. If the user is still after further clarification on what they should do, they can select the help option which will provide them with a detailed beakdown of the menu options available. Each feature can be easily used by selecting it from the main menu. Show, Update and Delete also require an ID number to be inputted for their functionality to be accessed. If these ID's are forgotten, the user can simply use the List feature to view them again. At any time if the user incorrectly inputs something, or attempts to input nothing for a note, the system will give them a message telling them what the exact issue was and how they can resolve it.

**Control flow diagram for the app**

**Implementation plan - Trello**

Link to the live Trello board is [here](https://trello.com/b/XyIxIzrm/ruby-terminal-app)

The below two images are from my initial Trello plan. I've set myself acceptable timeframes to complete each task, allowing for extra days before the project is due for any complications or extra features I may want to add. This initial plan changed a few times as I edited my board to better fit the exact scope of the project. An MVP was the main goal of the process, while also keeping in mind design elements that I wanted to add to make the project more appealing. Further screenshots of my Trello board as it was updated day to day are located [here](https://github.com/Finbob12/ToDoList-CLI/blob/master/readme_images/trello)

[](https://github.com/Finbob12/ToDoList-CLI/blob/master/readme_images/trello/24-sep-trello.png) [](https://github.com/Finbob12/ToDoList-CLI/blob/master/readme_images/trello/24-sep-checklist.png)

**System requirements**

The only system requirement is a computer that has bash script capability.

**Installation of the application**

* Follow these instructions to [download and install](https://www.ruby-lang.org/en/documentation/installation/) Ruby on your computer.
* Download the Zip file and extract it onto your computer, or clone the Github repository.
* Open your terminal and run the following command after navigating to the projects main folder.

$ gem install bundler

* After bundle is installed, run below command:

$ bundle install

* Then to launch the application enter below:

$ ./run\_app.sh

**Dependencies for the application**