Scribble Dash Milestone #2 Requirements

This requirements document explains Scribble Dash's Milestone #1 requirements. For each milestone, a new requirements document and an updated Figma file will be created. You can find all current milestones in the members area.

The mockups show the exact look and colors of a specific UI element. The app has a single theme, but no light or dark theme.

Note that this serves to give you an overall impression of what the app should be able to do. Feel free to decide how you implement specific things (e.g., how you display a specific loading progress or how you specify error messages).

You can find the mockups for Scribble Dash here: https://www.figma.com/design/TTnnjuj2SSijQWo2KI5Eg0/Milestones%3A-ScribbleDash?node-id=4002-623&t=cMEPJGsItty1nquo-0

Milestone #2 Goal

No app variants - just pure skill and precision!

Time to level up! 🔥 Build a fully functional flow for the ***One Round Wonder*** game mode. Load & display reference images for users to recreate. Implement an accuracy algorithm to evaluate their drawing skills.

You will develop a game where users recreate a given drawing as accurately as possible in a single try. Users will see a reference image for a few seconds and then draw it from memory. Once the drawing is submitted, it will be scored based on how closely it matches the original using an algorithm that compares two bitmap images. **I'll share with you exactly what it needs to do alongside a visual**

guide to help better communicate the intent in the Technical Requirements section.

Game Flow Overview

- Choose the game mode ***One Round Wonder*** from the *Home Screen*
- Choose the difficulty level
- Preview the reference drawing (with countdown)
- After the countdown ended, users can draw their version
- On submission, the comparison algorithm is run to get an accuracy score
- User is shown their score and rating with an option to try again

Icons

All icons for the app can be Material design icons or taken from the mockups as SVG (in case an equivalent Material icon doesn't exist)

Drawings

35 drawings will be provided as SVG files in the attachments of the challenge that you can import as (xml) vector drawable files. You are free to extend these with more drawings - they are all AI generated.

Feedback Text

Each rating has a a few texts associated with them that can be shown to the user on the *ResultsScreen.* You can download them here from the resources section in Memberspot. Alternatively, they have also been included at the bottom of this document.

Technical Requirements

- Setup

- When the app is launched, it needs to load all the XML drawing resources as a list of paths (each path consists of a list of x and y coordinates).
- If the resources have already been loaded, it should not be done again.

▼ Path Comparison Algorithm

Overview

This algorithm will compare the user's drawing to the reference drawing. To ensure a fair comparison between the two drawings, we need to make adjustments to both. This includes offsetting (moving/translating) them to the same positions and scaling them to equal size. Without doing this, even a perfect recreation of a drawing could result in a bad score if it is simply drawn in the wrong place on the canvas.

Step-By-Step Breakdown

Comparing the similarity of two paths is not trivial. In the context of ScribbleDash, please stick to the following algorithm:

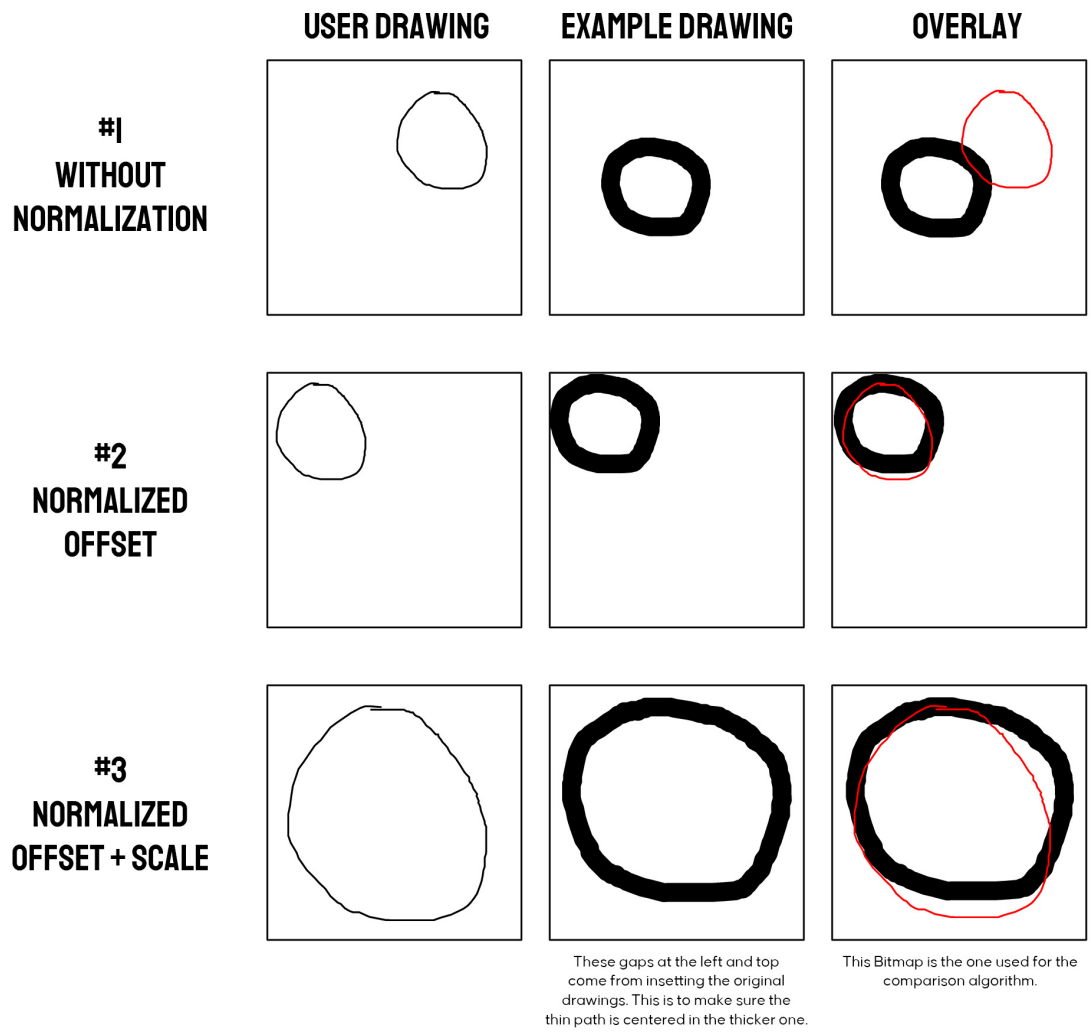
1. Take the list of paths from the example drawing and change the stroke width of the entire drawing to be a multiple of the stroke width from the user's drawing. The exact multiple depends on the difficulty. For difficulty "Beginner", the path for the example Bitmap is drawn with a 15x thicker path than the user's drawing. For "Challenging", the multiple is 7x and for "Master" it's just 4x. **Note, that this is just done internally for this algorithm - it's nothing the app's user should see or notice.**
2. Next, we need to make sure the offset and scale of a drawing have no impact on the final comparison of the similarity of the shapes. Therefore, apply the following transformations on the user and example paths in order to normalize them for better comparison **(you can find a visualization below)**:
    a. Calculate the total bounds of the shapes (total bounds → The dimensions of the rectangle the paths perfectly fit in)
    b. Inset both these bounds by half the drawing's stroke width. Additionally, inset just the user's drawing by `(exampleStrokeWidth -`

`userStrokeWidth) / 2`. From this point on use these inset bounds to calculate the dimensions of the shape, e.g. for the following scale factor.

   c. Offset each drawing so that its bounding box has a position of (0, 0) - the top left of the canvas.

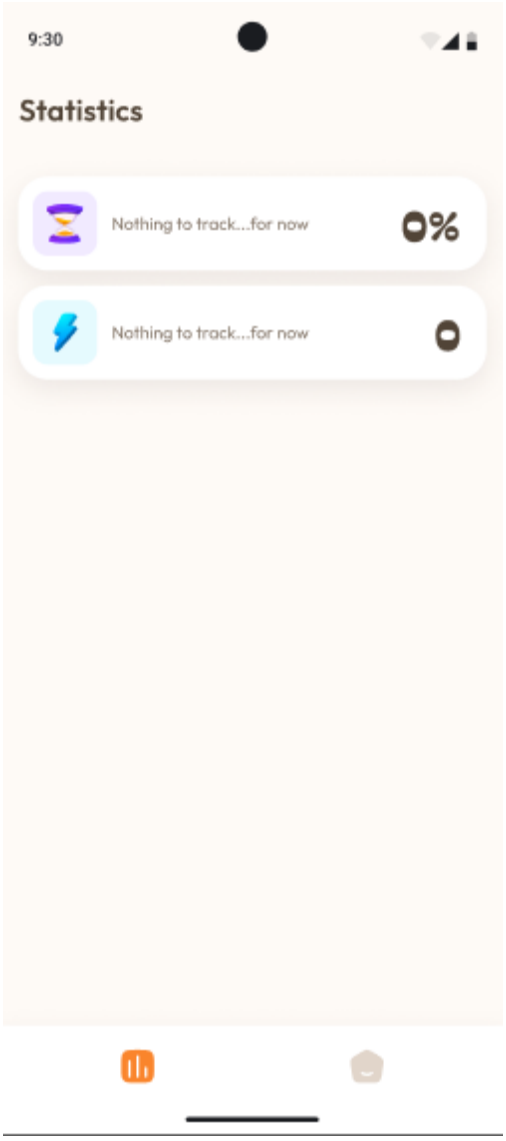
   d. Scale up each drawing to perfectly fit the canvas size. Again, this is not visible for the user, but just to make sure both drawings have the same size. **Scaling should not distort the drawings, but consider their aspect ratio.**

3. Now, a `Bitmap` of both the user and example drawing is created which has the size of the canvas and only contains the normalized paths for each drawing and fully transparent pixels for everything else. This means, you will have one Bitmap of the normalized user drawing's path and one Bitmap of the thicker normalized exmaple drawing path.
4. Next, iterate over every single pixel in these Bitmaps and compare the user's drawing with the example drawing.
5. If the current pixel is transparent in both Bitmaps → Ignore.
6. If the current pixel is non-transparent in the user's Bitmap → Count as a visible user pixel.
7. If additionally, the current pixel is non-transparent in both Bitmaps → Count as a matched user pixel.
8. After this loop, you can calculate the coverage percentage via `matchingUserPixelCount / visibleUserPixelCount`. In this calculation, `matchingUserPixelCount` is the amount of pixels in the user drawing Bitmap that is **also** visible in the example Bitmap. `visibleUserPixelCount` is the total amount of non-transparent pixels in the user drawing Bitmap.

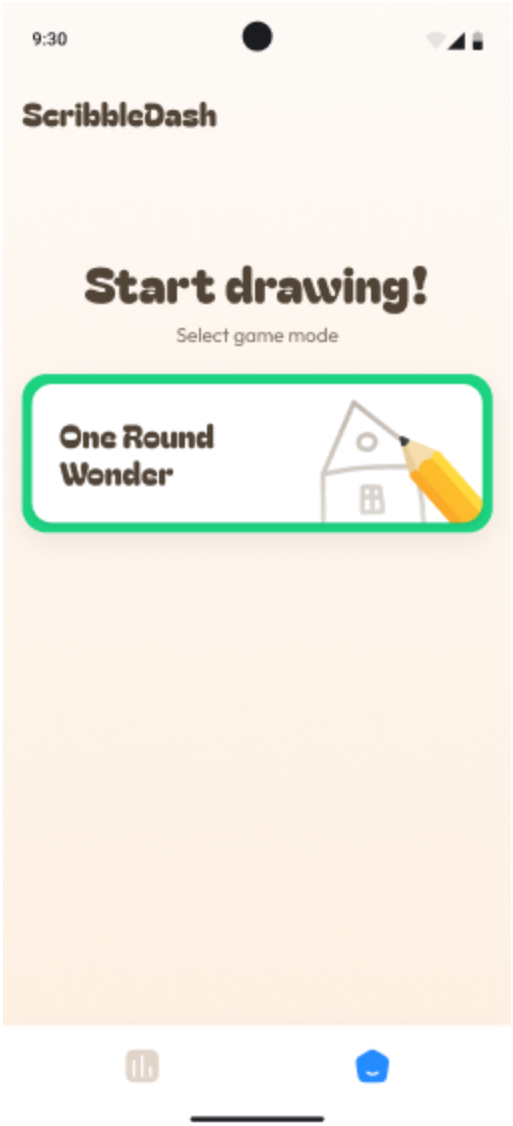


Visual of how each normalization step affects the drawings

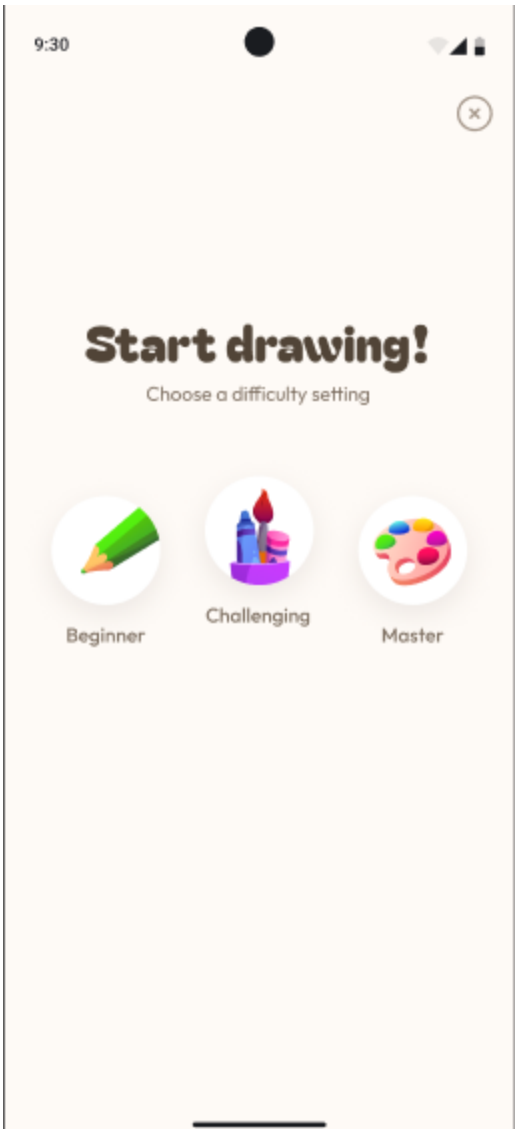
- **Statistics Screen**
  - *For Milestone #2, this screen should use only static UI; it does not need dynamic data*
  - Title at the top center of the screen: “Statistics”
  - Custom UI components to show a number and another to show a percentage
    - Max 4 digits allowed for the far right text
    - Each has an icon far left
    - Description text in between those elements that labels what the score or percentage is for



- **Home Screen (updated)**
  - The additional navigation destination is for the *StatisticsScreen*



- **One Round Wonder game mode flow (updated)**
  - **Difficulty Selection Screen**
    - The brush stroke size of the user **which is only used during the path comparison algorithm** is set based on the difficulty level selected
      - Beginner - 15x the default stroke size
      - Challenging - 7x the default stroke size
      - Master - 4x the default stroke size



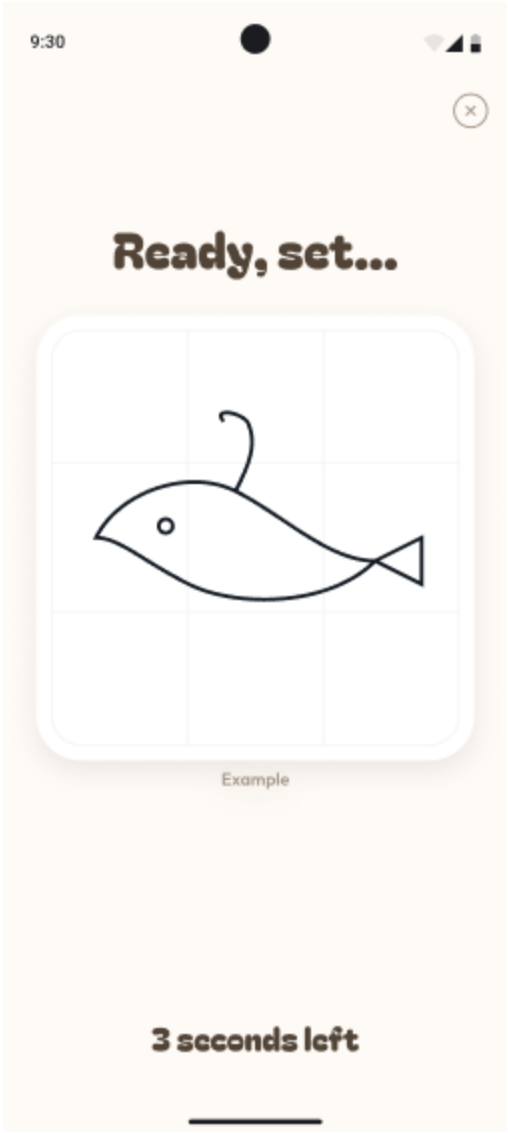
▼ **Draw Screen (updated)**

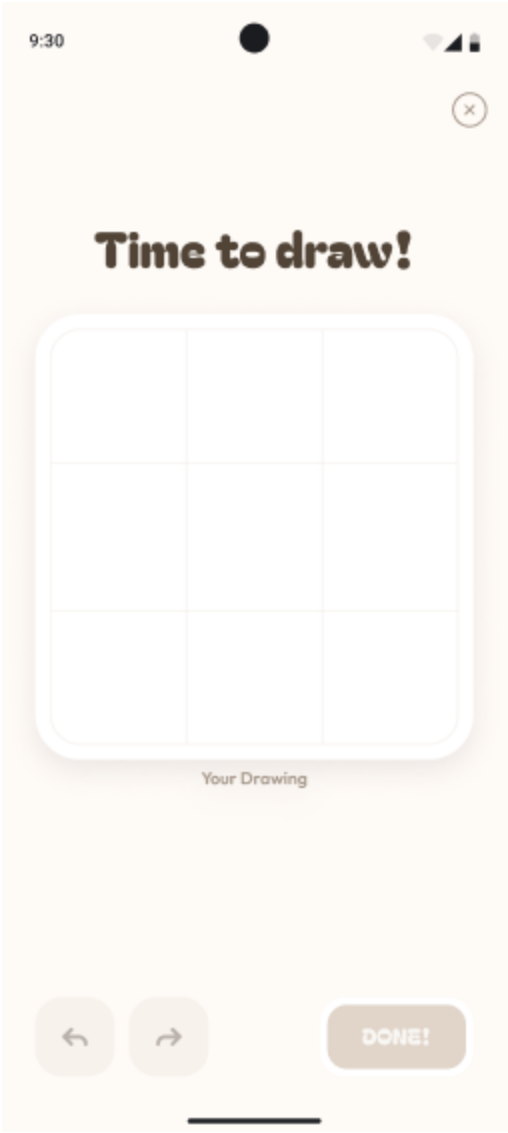
- New state where a preview of the image to be drawn is shown
  - Close icon top right of screen
    - On tap pop screen to return to *HomeScreen*
  - Randomly select a drawing to draw onto the canvas and display it to the user for 3 seconds
    - This drawing should be centered in the canvas and have a reasonable padding
  - Title: “Ready, set...”

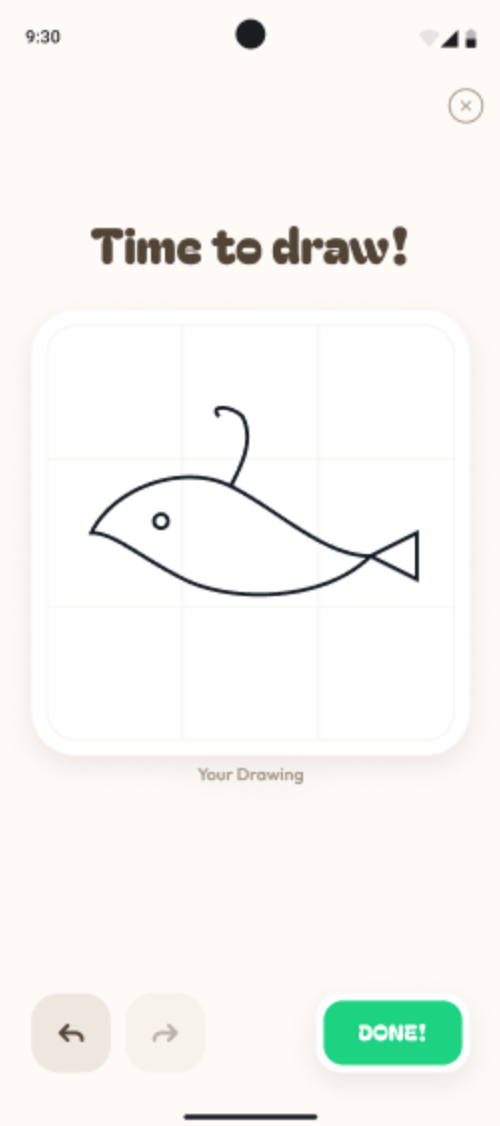
    - Canvas with drawing to mimic
    - Bottom caption on canvas: “Example”
    - The user should not be able to draw on the canvas while in this state
    - The bottom text counts down the remaining time left. The text needs to account for plurals as shown below:
        - “3 seconds left”
        - “2 seconds left”
        - “1 second left”
    - After the countdown ends switch screen state to drawing mode
- Draw Mode state
    - Done button (replaces the Clear Canvas button)
        - On tap
            - Navigate to the *ResultsScreen*
        - Disabled when the canvas has no paths drawn on it
    - Comparison Formula
        - Get the coverage percentage: This is how much of the original drawing’s shape is covered by the user’s drawing (e.g. 100% if user’s all pixels are overlayed by user’s drawing)
        - Path length comparison: We want to penalise the user if their path is shorter than 70% of the example shape’s length. The missing percentage is then subtracted from the final score
        - **Final Score (%) = Coverage (%) - Missing Length Penalty (%)**
            - If the User Path Length is less than 70% of the Example Path Length:
                - Missing Length Penalty (%) = 100 - (Total User Path Length / Total Example Path Length * 100)

- Otherwise (if User Path Length is ≥ 70% of the Example Path Length):
    - Missing Length Penalty (%) = 0
- The final score can only be between 0% and 100%.

Coverage	User path length / Example path length	Penalty	Final Score
100%	40%	60%	40%
100%	75%	0%	100%
85%	80%	35%	50%
65%	90%	0%	65%


9:30
Ready, set...
Example
3 seconds left

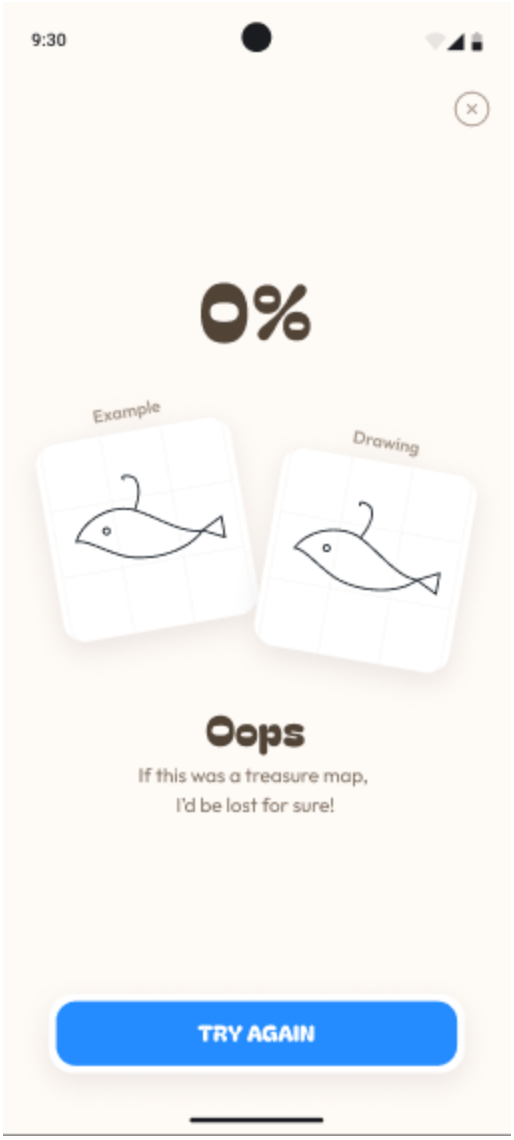

9:30
Time to draw!
Your Drawing
DONE!



▼ **Results Screen**

- Close icon top right of screen
  - On tap pop screen to return to *HomeScreen*
- Bottom button
  - “Try Again”
  - On tap, restart the *One Round Wonder game mode flow*
    - The user should be taken to the *Difficulty Selection Screen*
    - When on that screen, if the user taps the system back button, they should be taken to the *Home Screen*

- Main content shows the user drawing next to the example drawing with a rating and accuracy percentage
    - Title: The accuracy percentage score achieved (e.g. “70%”)
    - Two canvases, each rotated slightly
        - One with the example drawn
        - Another with the user’s drawing
    - Rating
        - The title is based on the final score percentage
        - The feedback message is chosen randomly from the pool associated with the rating achieved
- Tapping the system back button should not take the user back to the *Home Screen*



Ratings

The rating achieved is based on the user drawing accuracy score.

- Oops: 0% - 39%
- Meh: 40% - 69%
- Good: 70% - 79%
- Great: 80% - 89%

- Woohoo!: 90% - 100%

Hints

💡 **Importing the vector drawables as path data -** This requires a process called “parsing” where we can take the string text we extract from the XML file and convert it to code that we can use. Take a look at the `XMLPullParser` and `PathParser` classes to help you with parsing.

💡 **Navigation**

- Check out Philipp’s video on bottom bar navigation: https://www.youtube.com/watch?v=c8XP_Ee7iqY

💡 **Tips to implement this algorithm**

- It could make sense to model the bounds of each list of paths as a `RectF` object, since that provides useful utility like `rectF.inset(...)`
- For the normalization transformations, Android Paths (those from `android.graphics.Path`) provide a `path.transform(matrix)` function - matrices are perfect to change the translation and scale of a path.
- During development, it can help to save the bitmaps in a file to see whether you’re on the right track.

Resources

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- Oops - Meh Feedback -->
    <string name="feedback_oops_1">Looks like Picasso had a rough day!</str
ing>
    <string name="feedback_oops_2">Is this a drawing or a modern art statem
ent? Either way, I&amp;apos;m confused!</string>
    <string name="feedback_oops_3">I didn&amp;apos;t know stick figures could h
```

```
ave such deep emotions!</string>
    <string name="feedback_oops_4">This is what happens when you let a cat hold the pencil!</string>
    <string name="feedback_oops_5">Did you use a potato for inspiration? Because this is a-peeling!</string>
    <string name="feedback_oops_6">If this was a treasure map, I&apos;d be lost for sure!</string>
    <string name="feedback_oops_7">I see you&apos;re going for the &apos;abstract&apos; look. Mission accomplished!</string>
    <string name="feedback_oops_8">Is this a drawing or a game of Pictionary gone wrong?</string>
    <string name="feedback_oops_9">I can see the effort... right next to the confusion!</string>
    <string name="feedback_oops_10">This masterpiece is brought to you by the letter &apos;W&apos; for &apos;What is that?&apos;</string>

    <!-- Good - Great Feedback -->
    <string name="feedback_good_1">Not bad! Just a few more practice sessions and you might rival Picasso!</string>
    <string name="feedback_good_2">Almost there! With a little more practice, you&apos;ll be the next Van Gogh—minus the ear incident!</string>
    <string name="feedback_good_3">Good effort! Just remember, even the best artists had their &apos;meh&apos; days!</string>
    <string name="feedback_good_4">Solid attempt! Just think of this as your warm-up sketch for greatness!</string>
    <string name="feedback_good_5">You&apos;re on the right track! Just a few more strokes and you&apos;ll be a legend!</string>
    <string name="feedback_good_6">Nice try! With a little more practice, you&apos;ll be drawing like a pro in no time!</string>
    <string name="feedback_good_7">Great start! Just remember, every masterpiece begins with a little chaos!</string>
    <string name="feedback_good_8">Not too shabby! Keep at it, and soon you&apos;ll be the talk of the art world!</string>
    <string name="feedback_good_9">Good job! Just think of this as a stepping stone to your future art career!</string>
    <string name="feedback_good_10">You&apos;ve got potential! With a bit m
```

```
ore doodling, you&apos;ll be unstoppable!</string>

    <!-- Woohoo Feedback -->
    <string name="feedback_woohoo_1">Wow, did you just summon the spirit o
f Michelangelo?</string>
    <string name="feedback_woohoo_2">This is so good, I&apos;m convinced
you have a secret art degree!</string>
    <string name="feedback_woohoo_3">If this was a competition, you&apos;d
be the reigning champion of awesomeness!</string>
    <string name="feedback_woohoo_4">Is this a drawing or a portal to anothe
r dimension? Because I&apos;m ready to step through!</string>
    <string name="feedback_woohoo_5">You&apos;ve officially raised the bar!
I&apos;m going to need a ladder to reach it!</string>
    <string name="feedback_woohoo_6">This is so impressive, I&apos;m prett
y sure it just became the new Mona Lisa!</string>
    <string name="feedback_woohoo_7">If this were a food, it would be a gour
met meal—deliciously artistic!</string>
    <string name="feedback_woohoo_8">You&apos;ve got skills! I&apos;m star
ting to think you might be a wizard with that pencil!</string>
    <string name="feedback_woohoo_9">This drawing deserves its own fan cl
ub! Can I be the president?</string>
    <string name="feedback_woohoo_10">Is there a gallery nearby? Because t
his masterpiece needs to be on display!</string>
</resources>
```