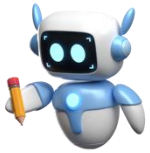
HDC – HGP - Assignment 01
UI Design Document



Student Name: Anriel Almeida
Student Number: 3168178

1. Using an application such as draw.io, canva or similar, create a wireframe of the layout of your UI, including explanation of what containers you used and why, as well as the naming convention used for each of your components.

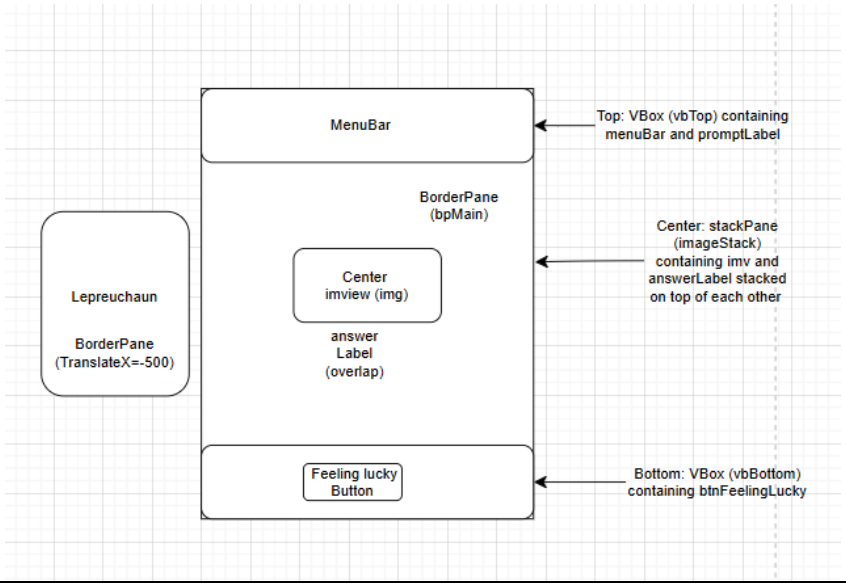
Wireframe and explanation:

BorderPane (bpMain): I chose BorderPane as the main layout container because it provides natural divisions for different areas of the application. This layout divides the screen into 5 regions (top, bottom, left, right, center), making it easy for organizing the UI components.

- **VBox (vbTop)**: Used at the top region to vertically stack the MenuBar and the prompt label. I chose VBox because it naturally arranges elements in a vertical sequence with controlled spacing, which works well for header elements.
- **StackPane (imageStack)**: Used in the center region to overlay the answer label on top of the Magic 8 Ball image. (so the answer text to appear directly on the ball)
- **VBox (vbBottom)**: Used for the bottom region to hold the "Feeling Lucky" button with appropriate padding and alignment. (Extra feature)

Component naming convention follows Hungarian notation:

- **Controls**: Prefixes indicate type (btn for Button, lbl for Label, etc.)
- **Containers**: Prefixes indicate layout type (vb for VBox, bp for BorderPane)
- **Camel case**: Used for multi-word identifiers (btnFeelingLucky, miFileChange)



2. Include screenshots of your Main User Interface. Why does your UI look the way it does? What design decisions did you make for this interface? Justify your decisions with a foundation of UI Design Principles.

Screenshots and explanation:

MenuBar

- **Location**: Positioned at the top of the application as this is the standard location for menus in desktop applications, following standard conventions and user expectations.
- **Style**: Simple and clean design that follows JavaFX default styling to maintain a familiar look and feel. CSS code makes it responsive on hover.
- **Functionality**: Contains File and About options, providing access to image customization and application information.

Prompt Label

- **Location**: Placed at the top, immediately below the MenuBar to guide users on how to interact with the application.
- **Size**: Medium-sized text to ensure readability without stealing attentions from the interface.
- **Style**: Simple text that provides clear instructions, disappears after first interaction to reduce visual clutter and -fx-font-family: "Palatino Linotype" to match theme.

Magic 8 Ball Image

- **Location**: Centered in the application to serve as the center of focus, following the principle of emphasis.
- **Size**: Dynamically sized to half the window width while preserving aspect ratio, ensuring it works well on different screen sizes (responsive design).
- **Functionality**: Interactive element that responds to clicks with animations and displays answers.

Answer Label

- **Location**: Centered directly on the Magic 8 Ball image to create the illusion that the answer comes from the ball itself.
- **Style**: Gold Text styled to stand out against the ball background for readability. -Font: -fx-font-family: "Palatino Linotype" to match the theme.
- **Size**: Appropriately sized to fit within the "window" of the Magic 8 Ball.

"Feeling Lucky" Button

- **Location**: Positioned at the bottom center of the application, aligning naturally from top to bottom.
- **Color**: Green color to indicate its special function and draw attention as a secondary action, (Green matching its Gaelic theme).
- **Size**: Larger than standard buttons with padding to draw users attention to it.
- **Style**: Custom styling with glow effect when activated to provide glow visual feedback.

Overall Color Scheme

- **Background**: Teal color (#168178) complements the black Magic 8 Ball, gold menu bar text, black menu bar, gold answer, gold prompt and green Feeling lucky button.
- **MenuBar, buttons, answer, prompt :** I chose these colours to remain the same for both themes for consistency.
- **Irish Theme**: Dark brown (#381819) background when the "Feeling Lucky" button is clicked, creating a distinct visual mode that signals the new function with Gaelic answers in gold which complements the green ball and button and brown. Chose to change the background colour and ball image so that they match the theme and balances every elements colour scheme.

Design Principles Applied

- **Contrast**: Dark ball against lighter background creates visual separation. Themed colour change green ball and brown background used to also create visual separation.

- **Alignment**: Components are carefully aligned to create a balanced, organized layout.
- **Proximity**: Related elements are grouped together (menu items, button with its effects).
- **Consistency**: to create visual consistency

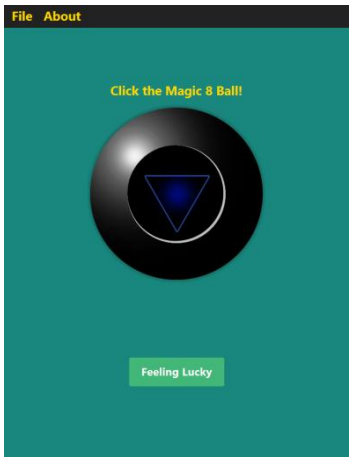
1. Similar styling for answer and prompt.
2. Same colour scheme and font for buttons, menubar and answer and prompt between themes

- **Feedback**: Visual and animation feedback for user actions improves usability.

Hover effect CSS styling is used to give user feedback. And glow effect from Irish theme. TranslateTransition for shaking effect for both themes.

Added a Theme switching behaviour for background

BorderPane Design Decision: I deliberately made the bpMain BorderPane an **instance variable** (rather than a local variable in the start method) to enable access across different parts of the application. This decision allows multiple methods to modify the same layout, particularly important for theme switching between the default and Irish modes.
When the ball is click it goes to default theme. The background matches the ball. When it is Irish Themed the ball changes to reflect the theme and the background matches the ball. The button and answer colours are consistent between themes to maintain similarity principle and consistency.



Referenced sites : 1) https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html#typecolor

Didnt use buttin chose to change image

miFileChange is directly used to make it similar to miAboutHelp

3. Briefly explain the functionality of each of the methods in your application.

Explanation:

Constructor (Magic8Ball()): Initializes all UI components, loads default images, and sets up the menu structure. It also calls methods to read prediction text from CSV files.

start(Stage primaryStage): Sets up the primary stage with appropriate dimensions, creates and configures all layouts, adds components to their respective containers, and applies styling. This method establishes the visual hierarchy and applies CSS styling.

init(): Sets up all event handlers for UI interactions, including file selection, 8 ball clicking, and "Feeling Lucky" button functionality (theme change+ easter egg). This method separates behavior from UI construction.

main(String[] args): The application entry point that launches the JavaFX application.

Custom methods:

readMagicAnswers(String magicAnswers): Reads standard predictions from the specified CSV file and stores them in an ArrayList for random selection.

readIrishAnswers(String irishAnswersFile): Reads Gaelic-themed predictions from a CSV file for use with the "Feeling Lucky" feature.

showDialog(): Displays an information dialog with details about the application and student information.

shakeImage(ImageView imv): Creates a shaking animation effect for the 8 ball image using TranslateTransition to provide visual feedback when clicked.

peekLeprechaun(ImageView leprechaunView): Controls the animation sequence for the leprechaun Easter egg, creating a slide-in, pause, and slide-out effect.

Referenced sites:

1.https://stackoverflow.com/questions/33190849/how-to-move-shapes-in-javafx/36666659?newreg=42eacd96e49c4a04923df128793f9100

2.https://www.w3resource.com/java-exercises/javafx/javafx-basic-exercise-10.php

3.https://docs.oracle.com/javafx/2/animations/basics.html

4.https://www.tutorialspoint.com/javafx/javafx_translate_transition.html

4. EXTRA FEATURE: Explain what your extra feature is, why you chose it, how it works etc. Include any relevant screenshots of your extra feature in action. Remember you must complete this section in detail to receive marks for your extra feature.

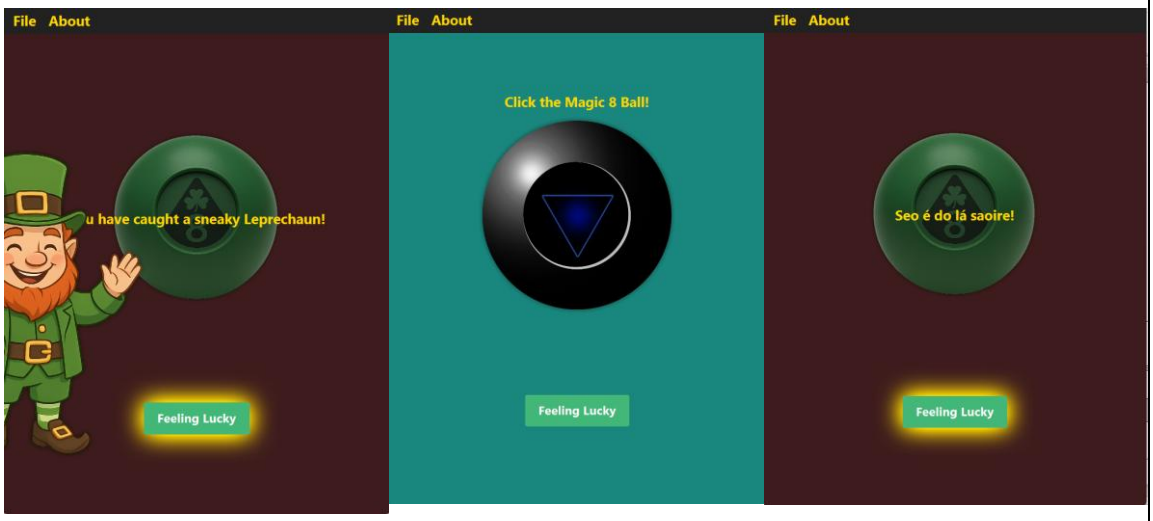
Screenshots and explanation:

Irish Lucky Theme

When users click the "Feeling Lucky" button, the application transforms with:

- An Irish-themed Magic 8 Ball image replaces the standard one
- The background color changes to a dark burgundy (#381819) that complements the green theme
- A special set of Gaelic-themed predictions/answers are used instead of the standard ones
- The button gets a green glow effect using CSS styling

This theme provides a culturally distinct alternative that adds variety and playfulness to the application.



Leprechaun Easter Egg

After clicking the "Feeling Lucky" button **five times, a hidden Easter egg is triggered**:

- A leprechaun image slides in from off-screen using animation effect.
- A special message "You have caught a sneaky Leprechaun!" appears in the answer label
- The animation sequence uses TranslateTransition and SequentialTransition for smooth movement
- After showing briefly, the leprechaun slides back off-screen

I chose this extra feature because:

1. It adds an element of surprise for users
2. It demonstrates more advanced JavaFX techniques
3. It adds to the magical theme of a Magic 8 Ball
4. It encourages exploration and rewards curiosity
5. I wanted to add a language feature and decided to combine all these features in a theme.

The feature required creating additional data files, implementing animation sequences, and managing state (click counting) to trigger the Easter egg at the right moment.

Transition effect on the magic 8 ball

shakeImage(ImageView imv): Creates a realistic shaking animation effect for the 8 ball image using TranslateTransition. I designed this as a separate method to promote code reuse, as it's called both when the standard 8 ball is clicked and when the Irish theme is activated. The animation provides visual feedback that simulates the physical action of shaking a real Magic 8 Ball.

5. Describe the interactive elements in your UI. How did you make sure they are intuitive and easy to use?

Explanation:

Magic 8 Ball Image

- **Intuitive Interaction**: Users naturally want to click or shake a Magic 8 Ball, so making the image clickable follows real-world Magic 8 ball.
- **Visual Feedback**: The shake animation provides immediate feedback that the action was received.
- **Clear Results**: The answer appears directly on the ball where users are already looking, following the principle of least surprise.

"Feeling Lucky" Button

- **Clear Labeling**: The button text clearly indicates its purpose and differentiates it from standard functionality.
- **Visual Distinction**: Styled differently from other elements to highlight its special function.
- **Feedback**: Activates a glow effect when clicked, providing visual confirmation.
- **Progressive Discovery**: The Easter egg feature encourages exploration without being necessary for core functionality.

MenuBar Options

- **Standard Positioning**: Following standard conventions for menu location and structure.
- **Grouping**: File operations and About information are separated into appropriate menus.
- **Descriptive Labels**: Menu items use clear, action-oriented text (e.g., "Change File").

File Chooser

- **Filtered Options**: Only allows PNG files to be selected, preventing user errors.
- **Default Location**: Opens in the Assets folder where example images are stored.

- **Standard Dialog**: Uses the platform's native file chooser for familiar interaction.

All interactive elements follow standard UI patterns to ensure users can easily understand how to interact with the application without extensive instructions.

6. What do you envision to be the user base for this application? Justify your response. Describe the typical user of this application.

Explanation:

The Magic 8 Ball application is designed primarily for:

1. **Casual Users Seeking Entertainment**: People looking for a fun, lighthearted way to "answer" questions or make trivial decisions.
2. **Young Adults and Teenagers**: The nostalgic appeal of the Magic 8 Ball combined with modern digital implementation appeals to this demographic who enjoy novelty apps.
3. **Office Workers**: As a stress-relief tool or icebreaker during meetings or breaks.
4. **Irish and Gaelic language students:** Due to my themed feature.

The typical user would be:

- Comfortable with basic computer applications
- Looking for brief, casual interactions rather than extended use
- Appreciates nostalgic elements with modern implementation
- Enjoys elements of surprise (like the Easter egg feature)
- May use the app socially (showing friends) or during breaks

I designed the interface to be immediately understandable without instructions, with large clickable areas and clear visual feedback for casual users. The additional Irish theme adds variety for returning users, while the Easter egg rewards exploration.

7. How would you redesign this application? Draw a new wireframe detailing what you believe the ideal design to be for the Magic8Ball application.

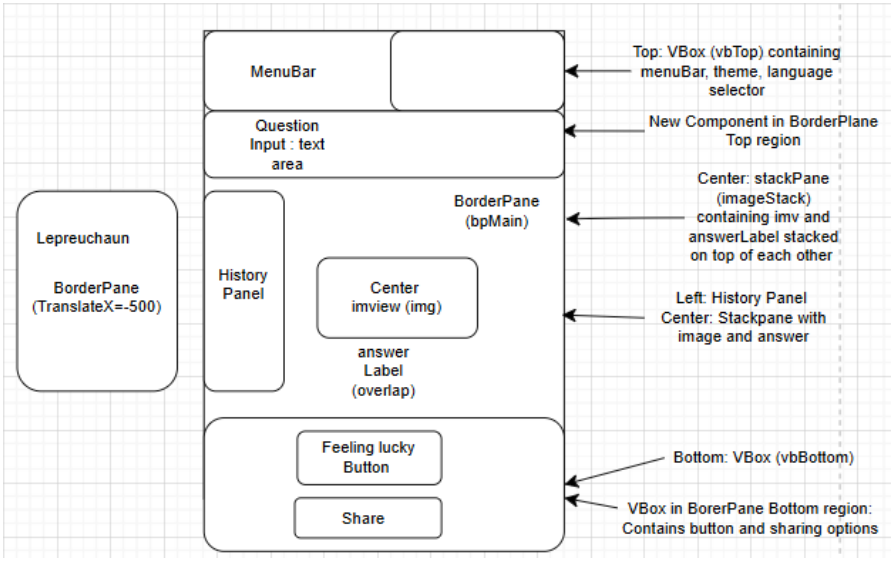
Wireframe and explanation:

My redesign would enhance the Magic 8 Ball experience with these improvements:

1. **Question Input Field**: Adding a text field at the top where users can type their question before clicking the ball. This creates a more complete simulation of the Magic 8 Ball experience.
2. **History Panel**: A collapsible side panel showing previous questions and answers, allowing users to track their session history.
3. **Theme Selector**: Instead of just one alternative theme, a drop-down theme selector would allow multiple visual styles with corresponding answer sets.
4. **Audio Feedback**: Adding subtle sound effects for shaking/clicking the ball and revealing answers would enhance the sensory experience.
5. **Accessibility Improvements**: Higher contrast text options and keyboard shortcuts for all functions.
6. **Language Option**: For a wider user base who don’t have English as their first language and to design the theme and Easter eggs to be culturally appropriate.
7. **Social Sharing**: Options to share interesting or funny answers on social media.

This redesign maintains the simplicity of the original while adding features that enhance

engagement and user return.



8. Create a unique color hex code by replacing the first digit of your student number with #. For example, if your student number is 3123456, the hex code becomes #123456. Set your UI background to this color, take a screenshot of the updated design, and comment on whether the new color improves your design, explaining why or why not.

Screenshot and Explanation:

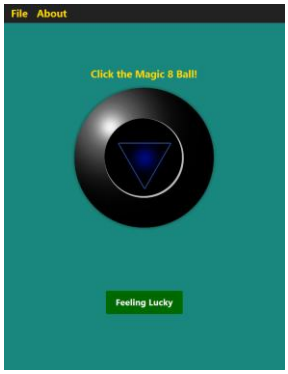
My unique hexcode#168178 is Dark Teal.

This color works well for the application because:

1. **Contrast**: The teal provides excellent contrast with the black Magic 8 Ball, making the central element stand out clearly.
2. **Mood**: The color has a calming quality that aligns with the fortune-telling theme of a Magic 8 Ball.
3. **Differentiation**: It clearly distinguishes the standard mode from the Irish theme's burgundy background.
4. **Readability**: The color is dark enough to provide contrast with white text but not so dark that it creates a gloomy atmosphere.

The color improves the design by creating an appropriate atmosphere without drawing attention away from the primary interactive elements. It creates a cohesive visual identity for the application that supports its purpose.

I could work with the design by adding contrasting gold colour to the menu bar, prompt and labels.



Reasons why it wasnt ideal

1. I had an Irish theme for my extra feature which didn't blend too smoothly with the teal.
2. The teal also didnt go too well with the dark blue highlights in the magic 8 ball image.
3. A background colour which would go well with the highlights as well as the button would be ideal.
   Like a shade of blue from the magic ball.

9. Include a screenshot of your *magicAnswers.csv* file, populated with random responses for the Magic 8 Ball. Describe the steps you followed to populate the list of magical answers and explain how a random answer is selected from the file.

Explanation and Screenshots:

For populating the Magic 8 Ball answers, I created a CSV file with traditional Magic 8 Ball responses. The process I followed was:

```
Yes!!
No!!
Maybe
Absolutely Not
Try again tomorrow
It is certain.
Without a doubt
Yes - definitely
Outlook good
Better not tell you now
Doubtful
My sources say yes
A resounding maybe
```

```
Tá sé cinnte!
Ní hé anois an t-am.
B'fhéidir.
Gheobhaidh tú é.
Ní dóigh liom é.
Seo é do lá saoire!
Tá súil agam go bhfuil tú sásta.
Ní
Tá
```

1. **Research**: I researched Magic 8 Ball responses to ensure the app reflected the toy and appropriate Gaelic counterpart.
2. **File Creation**: I created a CSV file named "magicAnswers.csv" in the Assets folder.
3. **Response Variety**: I included a balance of positive, negative, and neutral responses to provide varied user experiences.

The random answer selection works through these steps:

1. **File Reading**: The readMagicAnswers() method reads each line from the CSV file and adds it to an ArrayList called predictions1.
2. **Random Selection**: When the Magic 8 Ball is clicked, the event handler generates a random index using Math.random() * predictions1.size().
3. **Display**: The randomly selected prediction is retrieved from the ArrayList and displayed in the answer label.

(used help from https://stackoverflow.com/questions/40074840/reading-a-csv-file-into-a-array)

This approach allows for easy modification of possible answers by simply editing the CSV file without changing the application code. Similar logic was used for the Gaelic answers.

10. Describe how you implemented the functionality that allows the user to select a new image for the Magic 8 Ball. Explain the steps you took to enable image selection and how you ensured that only .png files were allowed. Include any relevant screenshots and discuss any challenges you faced during implementation.

Explanation and Screenshots:

The image selection feature allows users to personalize their Magic 8 Ball by choosing a different image. I implemented this functionality through these steps:

1. **Menu Integration**: Added a "Change File" option under the File menu that triggers the file selection process.
2. **FileChooser Configuration**:
   - Created a new FileChooser object with a descriptive title
   - Set the initial directory to the Assets folder where sample images are stored
   - Added an ExtensionFilter that only allows PNG files to be selected using the pattern "*.png"
3. **File Processing**:
   - got the selected file using fc.showOpenDialog(null)
   - checked if the file isn't null before processing
   - Converted the file to a URL string format that JavaFX's Image class can use
4. **Image Update**:
   - Created a new Image object with the file path
   - Updated the existing ImageView (imv) with the new image
   - Maintained the same sizing properties for consistency

5. **Error Handling**:
    - try-catch blocks to handle potential errors during file loading
    - Added console error logging to help troubleshoot issues

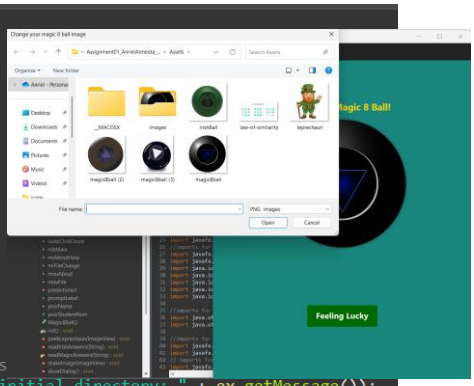
The PNG restriction ensures that only images with transparency are used, which is important for the clean appearance of the Magic 8 Ball against the background.

Implementation challenges included:

- Ensuring proper file path handling across different operating systems
- Managing potential exceptions when loading user-selected files
- Maintaining image proportions when changing to images with different dimensions

The solution works effectively, allowing users to customize their app while maintaining the functionality and layout of the application.

```
miFileChange.setOnAction(event ->{
    System.out.println("miFileChange was clicked!");
    //show a system dialogue to choose a new image
    FileChooser fc = new FileChooser();
    fc.setTitle("Change your magic 8 ball image");
    //set initial directory to Assets - volatile component
    // Set the initial directory to "Assets/"
    try {
        // Attempt to set the initial directory to "Assets/"
        File assetsDir = new File("Assets/");
        if (assetsDir.exists()) {
            fc.setInitialDirectory(assetsDir);
        } else {
            System.err.println("Assets folder does not exist!");
        }
    } catch (Exception ex) {
        // Handle unexpected errors (e.g., file system access iss
        System.err.println("An error occurred while setting the initial directory: " + ex.getMessage());
        ex.printStackTrace();
    }
    //filter out only image files ( .png)
    ExtensionFilter onlyImages = new ExtensionFilter("PNG  images","*.png");

    //add filter to file chooser
    fc.getExtensionFilters().add(onlyImages);
eTogether...
    //show dialog and get the chosen file
    File chosenFile = fc.showOpenDialog(null);

    //make sure the file isnt null
    if(chosenFile !=null) {


        //show the image in the UI (try/catch!)

        try { // load selected image and update the imageView
            img = new Image (chosenFile.toURI().toURL().toString());
            imv.setImage(img); // we dont create a new image , it just update the image
        }

        catch  (Exception oopsie) {
            System.out.println("Error reading user selected image.");
            oopsie.printStackTrace();
```



11. If any part of your submission isn't working, please detail this below including relevant screenshots.

Screenshots and explanation:

All features of my Magic 8 Ball application are fully functioning as described, including:

- Core Magic 8 Ball functionality with random answers
- Menu system with About dialog and image selection
- Irish theme with the "Feeling Lucky" feature
- Leprechaun Easter egg animation
- File reading for both standard and Irish predictions
- Image scaling and animations

There are no non-working components to report.