KENNY ZHANG

Application Release Testing

App to Test: codingChallenge2
Date of Test: Sun May 19, 2019
Number of Screens/Windows: 2
Results: (Pass) (Unsuccessful)

Functionality:
Loads onto Simulator ( ) 2nd Try with initial simulator (iPhone 8 Plus)

[redacted]

User Experience and Compliance with Problem Statement: OMIT FOR APP

[redacted]

Loads successfully. Cells behave as expected when tapped on; they display image. Rapid touch test (randomly and quickly tap areas around the screen): No unexpected behaviour. Is consistent. Doesn't behave unexpectedly. Search bar functions as expected. Even the random type test doesn't crash the app or slow it down. Unexpected symbols don't cause problems. Layout is consistent, nothing strains eyes. Images scale properly; no tearing/stretching or any aesthetic issues. Memory management is excellent It stays within the bounds of 60-70 MB and doesn't drop too low or rise excessively high.

In conclusion, based on the results of this assessment, the app passes the test with its consistent memory management, ability to handle high stresses and maintains high quality standards. (Add 2 marks)

Stress Testing and Code Base: (22.5) out of 25
Performance: (Poor) (Barely Useable) (Useable, but some issues) (Excellent)
Test out each screen in the app. Try to apply unusual and unexpected actions and see how the app responds. For each screen, list any errors here. In addition, evaluate the code base using the denoted guidelines. Use Practical Tests section for reference.

- Correctness
- Naming
  - Prose
  - Delegates
  - Use Type Inferred Context
  - Generics
  - Class Prefixes
  - Language
- Code Organization 3

[redacted]

EVALUATION RESULTS

[redacted]

STRESS TESTING AND CODE BASE: (22.5) of 25

[redacted]

TOTAL: (22.5) of 25

Total to pass: 22 of 25 (86% or greater)

Notes and Remarks:

Evaluation Instructions

Loads into Simulator (Execution):
Run the app directly from download from repository. If it fails to run, attempt to run it again. If it fails to do so, run from a different simulator. If it still fails to run, the app fails immediately as features cannot be evaluated from the executable.

[redacted]

[redacted]

Practical Tests to carry out (Stress Testing and Code Base):
FUNDAMENTALS OF COMPUTERS: CPU/RAM/HDD/HID
Press ALL buttons in the scene and note any broken or unexpected functions.
Enter text in the entry fields in each scene. Try simple text first. Then attempt alphanumeric characters before symbols and even foreign characters.
Activate any toggle switches present. See how it responds. Does it do anything?
Mix up everything. Type into the field and randomly activate buttons/switches.
Test all features in the app. Does anything appear abnormal?

Examine the layout and design of the app. Does anything appear eye-straining? Is the layout very confusing to use? Are the constraints out of place? Is everything in a controlled manner or does it seem out of place?

Test the app's memory management. Duplicate and rapidly create large volumes of information. Document when the performance starts to decay. Abruptly exit the app midway. Does it lose control? How about in the middle of a saving operation?

[redacted]

Are there any functions that reduce the app's performance? If so, list them.
Using System Monitor (or equivalent) monitor the app's CPU/RAM/HDD usage.
Is resource usage abnormally high for this type of app and situation? (3D apps and games is normal)

Code Base:
Evaluate the source code and its compliance with these guidelines (Through hyperlinks)

- Correctness
- Naming
  - Prose
  - Delegates
  - Use Type Inferred Context
  - Generics
  - Class Prefixes
  - Language
- Code Organization 3
  - Protocol Conformance
  - Unused Code
  - Minimal Imports
- Spacing
- Comments
- Classes and Structures 6
  - Use of Self
  - Protocol Conformance
  - Computed Properties
  - Final
- Function Declarations
- Function Calls
- Closure Expressions 9
- Types

Taken from “The Official raywenderlich.com Swift Style Guide.” (Raywenderlich)

[redacted]