System and Release Testing

Scenario Tests

Scenario 1:
Bobert has created a file in pick-a-path and once he finished working on it at the time, he saved his file and closed the program. Later, he decided that he wanted to add a few more scenarios and decisions, so he re-opened pick-a-path, opened his saved file, and added more boxes and arrows. He is satisfied with his work and saves his file and closes the program. Then later opens it and has his friend, Fred, play through it. After getting feedback from Fred about his game, Bobert decides to add even more scenarios and decisions, so he opens pick-a-path again and adds more boxes and arrows, then saves the game again and exits the program.
Result:
The entire scenario runs smoothly without any crashes. All the files are successfully saved and accessed later no matter how many times the user needs it.

Scenario 2:
Bobert has been a working on a program for an hour. He saves the program. But after a couple of minutes he starts to make changes in the program again when abruptly his laptop runs out of battery and switches off.
Result:
After restarting the application, he finds that his most recent changes that were unsaved are gone. The file only consists of the data from the last save point.

User Tests:

We approached two users to test our software:

1. The user found it easy to use, though the “add arrow” part was a little confusing until we explained it to her. Other than that, she thought it was easy and it was fun to create a story in a “choose your own adventure” format. After looking at the manual/having a developer explain things, she was able to use it easily to make test stories.

2. The user read the manual before he used the software. He said the manual made it easier to understand the software, but thinks the software is easy and user-friendly to use even if the manual was absent. He thought it was fun and very creative to create whatever he wished to. He also thinks that this software can be helpful in creating flowcharts and search engines.

Stress Tests:

We conducted four stress tests to test our software:

1. A lot of boxes (connected) and arrows were added manually to the software to see if there's a long story with a lot of options that will break the program.
   Result:
   The program doesn't crash on account of the number of boxes and arrows. The user can add as many boxes and arrows as they want. But these boxes and arrows need to contain text (names/numbers) of some sort in them in order for the program to work. Empty boxes and arrows can be saved in a .pap file but will crash in the player mode.

2. Using multiple windows of the program at the same time.
   Result:
   The software doesn't crash. The user can work on the same project in multiple windows and update it, as they want. The most recent save is the most recent save for the file.

3. An auto clicker, which made a box every 0.1 seconds for five minutes was used to see if the software crashes in the editor or the player mode, due to overload of boxes.
   Result:
   The software did not crash or even slow down in this stress test. It keeps populating the canvas with boxes at a constant rate until the auto clicker was stopped. The file was later saved and accessed again without any problem.

4. An auto-typer was used to see if the JtextArea in the editor mode GUI would slow down or crash.
   Result:
   The software was able to store all the data inputted by the auto-typer and never crashed, slowed down or stopped functioning.