Evan Waddicor, Marquis Jones, Cameron DiBenedetto
CIS 264
Professor Robidoux
28 April 2020

Final Project: No Trivial Matter

PRODUCT OVERVIEW

Description: A dynamically loaded Presidential Trivia game with 3 different difficulty levels and at least 10 questions for the user to answer per level. The game tests the user on their knowledge of the U.S. Presidents throughout history. It provides an interactive GUI which allows the user to answer multiple choice questions. The user can select different difficulty levels and then answer questions while the game tracks their score.

Software Requirement:

Dynamic Attributes:

- Scalability, easy to make the game larger in size while not breaking the file by adding more questions.
- Usability, easy to use GUI with self explanatory buttons and prompts.
- Reliability, executable experiences no runtime failures or errors and did not crash during testing.

Static Attributes:

- Maintainability, extremely easy to go in and add new questions or change the font colors/size
- Presidential Trivia User Manual

Resources: List the hardware and software resources needed for development and implementing the testing plan. Be sure to include at least one software testing tool.

Java Eclipse, IntelliJ IDEA

ITEMS TO BE TESTED

List and describe the software features that are being tested by this plan. This must include at least the dynamic software requirements.

1. Scalability - The Presidential Trivia Game will be supported so that more questions will be able to be added and the game itself can grow in size for users. In addition to always being able to add more questions there's difficulty levels that carry a large amount of questions for each.

2. Reliability - The Presidential Trivia Game will need to be reliable so that their scores won't be lost mid-game and the app will also not shut off on them. Set variables to answers deemed correct and incorrect for a final count after one game.
3. Performance - The Presidential Trivia Game will need to be quick in responses to button clicks along with dynamically and quickly saving and dividing their scores.
4. Score - A collective tally of incorrect answers and correct answers are stored until the final game screen.
5. Streaks - A count based on number of correct answers in a row or resets once a question is answered incorrectly. The longest streak of gameplay is displayed at the final screen.
6. User feedback - Every answer clicked triggers a pop-up window informing the user whether they were correct and brought to the next question or if they were incorrect and display the correct.
7. Multiple Choice - Each question is multiple choice and the data that is read in for the answers comes from a database with incorrect and correct answers.

DELIVERABLES

List and describe the deliverables from this testing plan. Each deliverable must be included with the submission of the final project. Examples may include, but are not limited to...

- Beta version of the software (other iterative versions may also be included)
- Testing results
- Bug reports

Included in the Deliverables folder is:

- TriviaFinal_V01
- TriviaFinal_V02
- TriviaFinal_V03
- TriviaFinal_V04
- Bug Reports (Cumulative)
- TriviaFinalTest.java (JUnit test cases)

Sprint #1: Basic Gameplay

- GOALS: User can start game and answer multiple questions. The user should receive messages of correct or incorrect.
- DELIVERABLES (taken from Product Backlog)
    - Array structure with 20 questions/answers
    - Version 0.1 of trivia game
    - Product Backlog

TEST SPECIFICATION: SPRINT #1		
Test Cases		
ID	**Feature**	**Generation Method**
UserInput_ValidCorrect	User Story: Simple Gameplay	Equivalence Partitioning
UserInput_ValidIncorrect	User Story: Simple Gameplay	Equivalence Partitioning

Test Specification: Sprint 1			**Test Case ID:** UserInput_ValidCorrect			
Test Case Description: The aim of these set of test cases is to test the invalid versus valid user input. Valid user input comes in the form of selecting a choice from the 4 multiple choices. Invalid user input comes in the form of selecting somewhere that is not in the valid range of user input such as outside of the 4 choices.			**Prerequisite(s):** ● A set of questions for the test user to input valid and invalid inputs ● A choice of 4 questions in which the user is able to click through the 4 questions in any random order and get feedback			
Step	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**	**Notes**
1	User select a multiple choice option	George Washington (“a2”)	George Washington	George Washington	PASS	User selected the option “a2” for the sake of testing and received the proper answer back
2	Answer selected is checked internally and user receives positive feedback	Correct answer	Correct message	Correct message	PASS	Went through the code manually to see the tracing of when the user click is listened to what

						occurred after

Test Specification: Sprint 1			**Test Case ID:** UserInput_ValidIncorrect			
Test Case Description: The aim of these tests is to see what occurs during the valid input of a user but the wrong answer.			**Prerequisite(s):** ● A set of questions for the test user to input valid and invalid inputs ● A choice of 4 questions in which the user is able to click through the 4 questions in any random order and get feedback			
Step	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**	**Notes**
1	User select a multiple choice option but is incorrect purposely for some of the options	A multiple choice option follow a1,a2,a3,a4	1 right answer and 3 wrong answers for valid input	George Washington and an option pane for what is right or wrong along with displaying the correct answer	PASS	User checked 4 different inputs. 1 right answer and 3 wrong and then was expecting the George Washington output for the correct answer.
2	Answer selected is check internally and user receives negative feedback	Incorrect answer from one of the 4 multiple choices	Incorrect message	Incorrect message pane showing what the correct answer was supposed to be	PASS	The correct answer was supposed to be George Washington and when the wrong answer was clicked, the user was prompted with the correct answer

Sprint #2: Testing Difficulty Settings

- GOALS: User can start game and after redirected by the “Start Game” button, they are asked to choose their difficulty from “Easy”, “Medium”, or “Hard”
- DELIVERABLES (taken from Product Backlog)
    - Working GUI with a start screen and difficulty select screen.
    - Version 0.2 of trivia game
    - Added an easy and medium difficulty along with a set of test questions to check out the difficulties

TEST SPECIFICATION: SPRINT #2		
Test Cases		
ID	**Feature**	**Generation Method**
MethodTest_DifficultySelect	User Story: Simple Gameplay	Cause-Effect Graphing
MethodTest_StartNewGame	User Story: Simple Gameplay	Cause-Effect Graphing

Test Specification: Sprint 2			**Test Case ID:** MethodTest_DifficultySelect			
Test Case Description: Software must display selected difficulty once the user hits the button to begin the game. In the database there are questions grouped by different difficulty levels that are generated into the game upon the selection of that difficulty.			**Prerequisite(s):** ● Database containing at least 1 question/answer pair ● Database containing questions and answers for each difficulty			
Step	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**	**Notes**
1	Select difficulty	Easy	Easy level questions and answers	Easy level questions and answers to the easy level questions	PASS	Problems at first discerning between the easy questions and medium questions.
2	Select difficulty	Medium	Medium level questions and answers	Medium level questions and answers to the medium level questions	PASS	Problems at first discerning between the medium and hard questions.
3	Select difficulty	Hard	Hard level questions and answers	Hard level questions and answers to the hard	PASS	Questions viewed as hard were sometimes viewed as

				level questions		to obscure for the game and did not create a fun user experience

Test Specification: Sprint 2			**Test Case ID:** MethodTest_StartNewGame			
Test Case Description: Software must display difficulty screen after user prompts “Start New Game”			**Prerequisite(s):** ● Database containing at least 1 question/answer pair ● Database containing questions and answers for each difficulty			
Step	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**	**Notes**
1	Start game	Start Game button	Difficulty screen	Difficulty screen	PASS	Could potentially use a change in color to depict the different difficulties visually
2	Complete trivia game	Answer all questions	Score screen	Score screen	PASS	Score screen could use alignment of values along with encouragin g text
3	Start new game	Start New Game button	Difficulty screen	Difficulty screen	PASS	After the score screen there could be some visual to how that you have started a new game and what your previous high score was

Sprint #3: Leaderboard

- GOALS: User can start game and answer multiple questions. The user should receive messages of correct or incorrect. At the end of the game a final screen appears with stats from the game like number of incorrect/correct answers, the longest streak achieved during the game and whether or not the user's final score
- DELIVERABLES (taken from Product Backlog)
    - Data structure with 20 questions/answers per difficulty level
    - Version 0.3 of trivia game
    - Score count (correct/incorrect)
    - Streak count (consecutive correct answers)
    - Highscore
    - Bug Report: A list of failed test cases

TEST SPECIFICATION: SPRINT #3		
Test Cases		
ID	**Feature**	**Generation Method**
MethodTest_NewHighScore	User Story: Simple Gameplay	Equivalence Partitioning
MethodTest_CalculateScore	User Story: Simple Gameplay	Equivalence Partitioning

Test Specification: Sprint 3			**Test Case ID:** MethodTest_CalculateScore			
Test Case Description: Software must accurately present final statistics			**Prerequisite(s):** • Database containing at least 1 question/answer pair • Count running score			
Step	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**	**Notes**
1	Complete trivia game with all questions correct.	N/A	Longest streak count: 20, Points: 210, Number Correct: 20 Number Incorrect: 0	Longest streak count: 20, Points: 210, Number Correct: 20 Number Incorrect: 0	PASS	Could implement some sort of cheering for the people scoring 20 correct straight
2	Complete trivia game with all questions incorrect.	N/A	Longest streak count: 0, Points: 0, Number Correct: 0 Number Incorrect: 20	Longest streak count: 0, Points: 0, Number Correct: 0 Number Incorrect: 20	PASS	Could implement messages of criticism for people getting 20 wrong

Test Specification: Sprint 3			**Test Case ID:** MethodTest_NewHighScore			
Test Case Description: Software must display the newest accurate high score			**Prerequisite(s)**: ● Database containing at least 1 question/answer pair ● Database containing questions and answers for each difficulty ● Working arithmetics for final statistics			
Step	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**	**Notes**
1	Complete trivia game for the first time with a score < 50 and score > 0	N/A	New high score message on final screen	New high score message on final screen	PASS	High scores shown on a run should have some more redeeming qualities so the user is enticed into coming back for more gameplay
2	Complete trivia game for the first time with a score > 50.	N/A	New high score message on final screen	New high score message on final screen	PASS	
3	Complete trivia game for the first time with a score < 50.	N/A	No message	No message	PASS	

Sprint #4: Testing Difficulty Settings

- GOALS: User can start game and after redirected by the “Start Game” button, they are asked to choose their difficulty from “Easy”, “Medium”, or “Hard” and after they select their difficulty they are never prompted with the same set of questions in a row back to back
- DELIVERABLES (taken from Product Backlog)
    - Data structure with 20 questions/answers for another difficulty level (revised).
    - Version 0.4 of trivia game
    - Added a randomizer of questions so the user isn’t able to memorize the location of each answer in a question without actually learning the answers

TEST SPECIFICATION: SPRINT #4

Test Cases		
ID	**Feature**	**Generation Method**
MethodTest_RandomizeAnswers	User Story: Simple Gameplay	Equivalence Partitioning

Test Specification: Sprint 4			**Test Case ID:** MethodTest_RandomizeAnswers			
Test Case Description: Software must validate the answers for each question are randomized and provide 3 random answer options and 1 correct answer option			**Prerequisite(s):** ● Database containing at least 1 question/answer pair			
Step	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**	**Notes**
1	Start trivia game	Start New Game button	3 random answers and 1 correct answer	3 random answers and 1 correct answer	PASS	Formatting becomes messy with the randomization of answers and have to make sure that only presents are outputted for answers