Hi Lo Game

|  |
| --- |
| ***You MUST provide evidence showing how the problem has been decomposed, how the components have been developed and trialed, and of how they have been assembled and tested to create a final, working outcome.*** |

### Outline / Decomposition

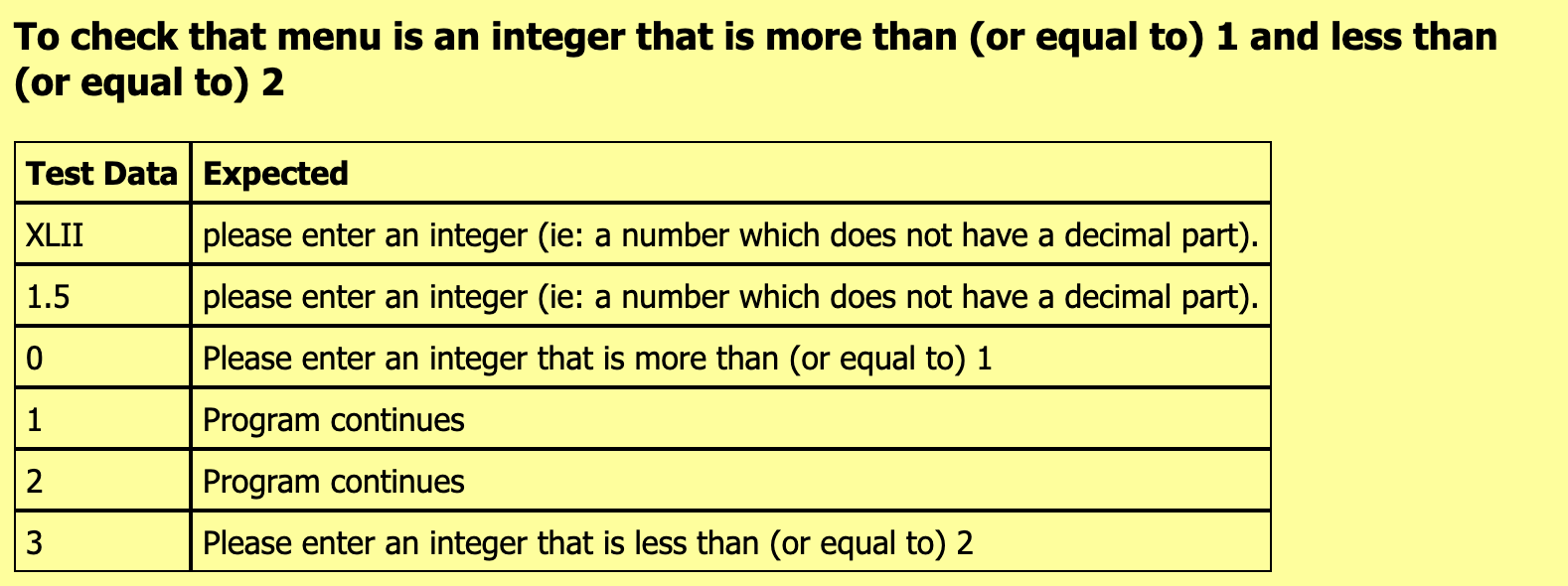
1. *Show menu*
   1. *If 1 continue*
   2. *If 2 quit*
2. *Ask the user how much they want to deposit into the game*
3. *Ask the user how much money they want to play with in that round*
4. *Computer randomly generates two numbers that is between 0-100*
5. *Comparing user guess with numbers generated*
6. *Calculate the win percentages with the two numbers generated*
7. *Winning system*
   1. *If user guess ‘high’ and next number is higher than the previous one. Add “win\_amount” to balance.*
   2. *If user guess ‘low’ and next number is lower than the previous one. Add “win\_amount” to balance.*
   3. *If user guess ‘high’ and next number is lower than the previous one. Subtract ‘user bet’ from balance.*
   4. *If user guess ‘low’ and next number is higher than the previous one. Subtract ‘user bet’ from balance.*
   5. *Show the user feedback*
      1. *If user is correct print ‘win\_feedback’ and ‘win\_amount’*
      2. *If user is incorrect print ‘lose\_feedback’*
8. *End Game System*
   1. *If balance is greater than $1, User can hit ‘2’ to quit or press ‘1’ to continue playing*
      1. *If user hits <1> number 2 becomes number 1 and number 2 get regenerated randomly*
      2. *If user enters <2> or cancel, end game*
   2. *If balance is equal to 0, Output “Sorry you don’t have enough money to continue”, end game*

### Version Log

[Version 1](https://github.com/Brad123ghost/HiLowGame/blob/master/hilowgame/index.html)

[Version 2](https://github.com/Brad123ghost/HiLowGame/blob/master/hilowgame/indexV2.html)

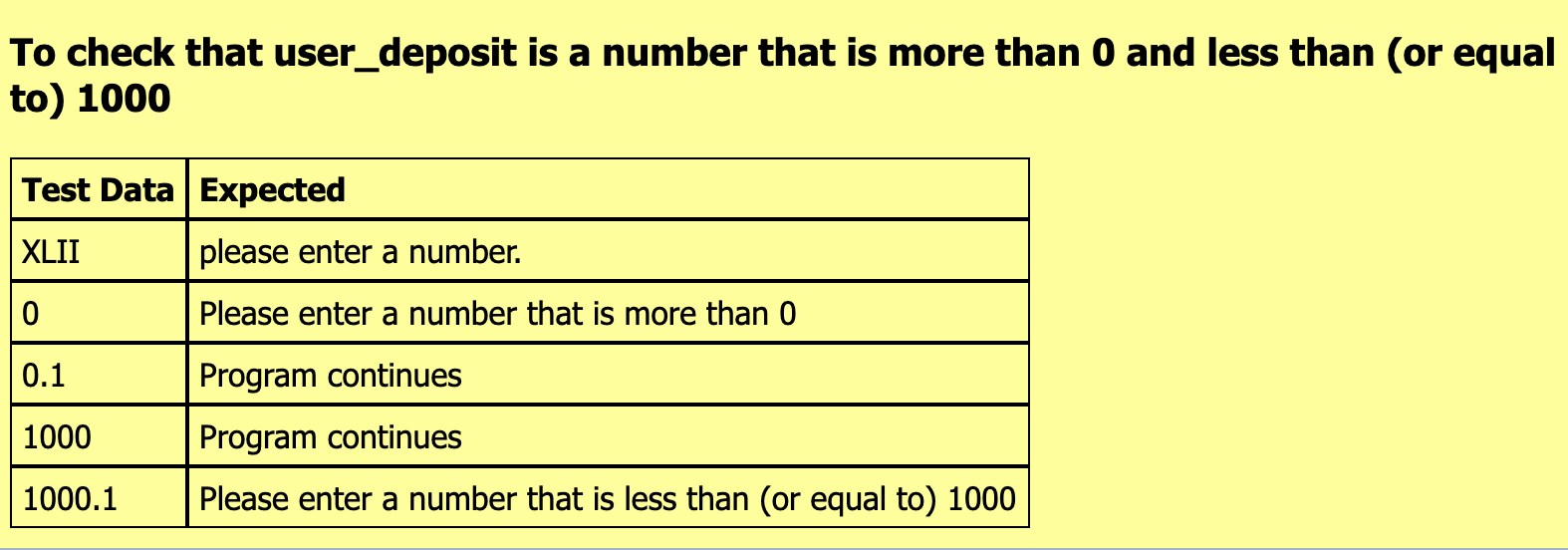
### Component Testing

**Component 1: Test that ‘menu’ is equal to 1 or 2**

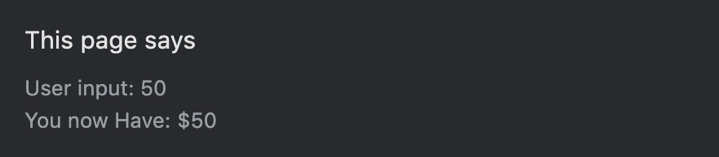
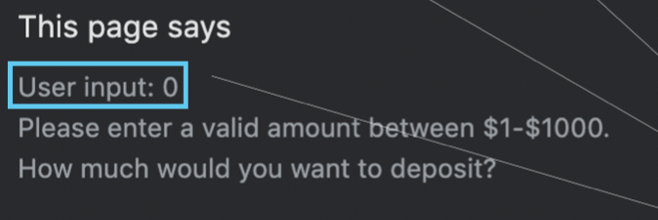
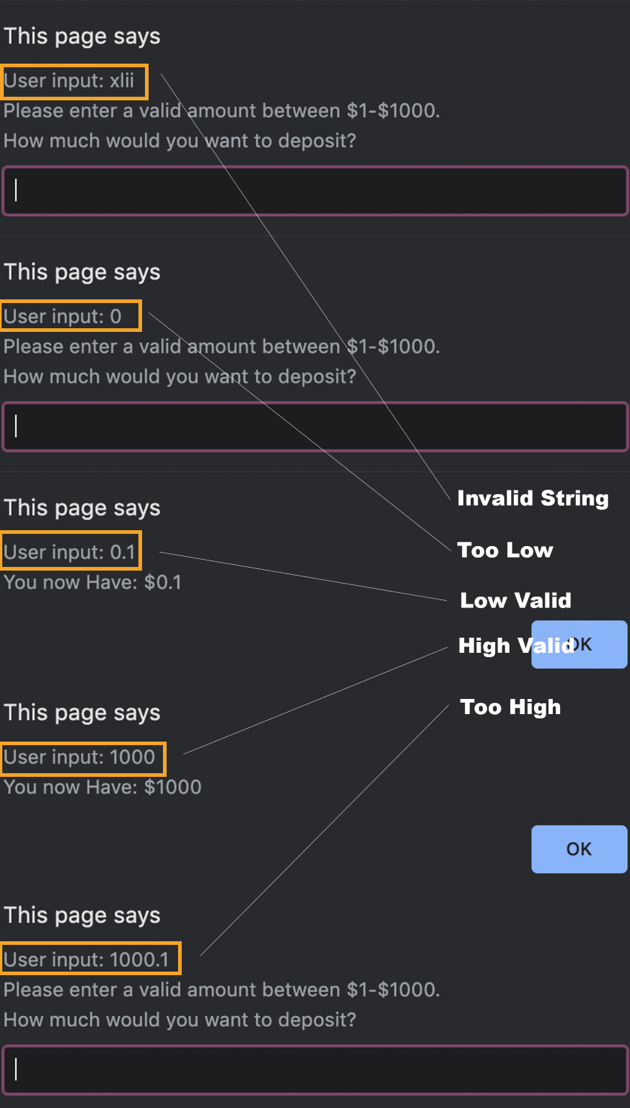
|  |  |
| --- | --- |
| **Test Data** | **Expected** |
| Menu: 1  Menu: 2 | Program continues  Program quits |

****

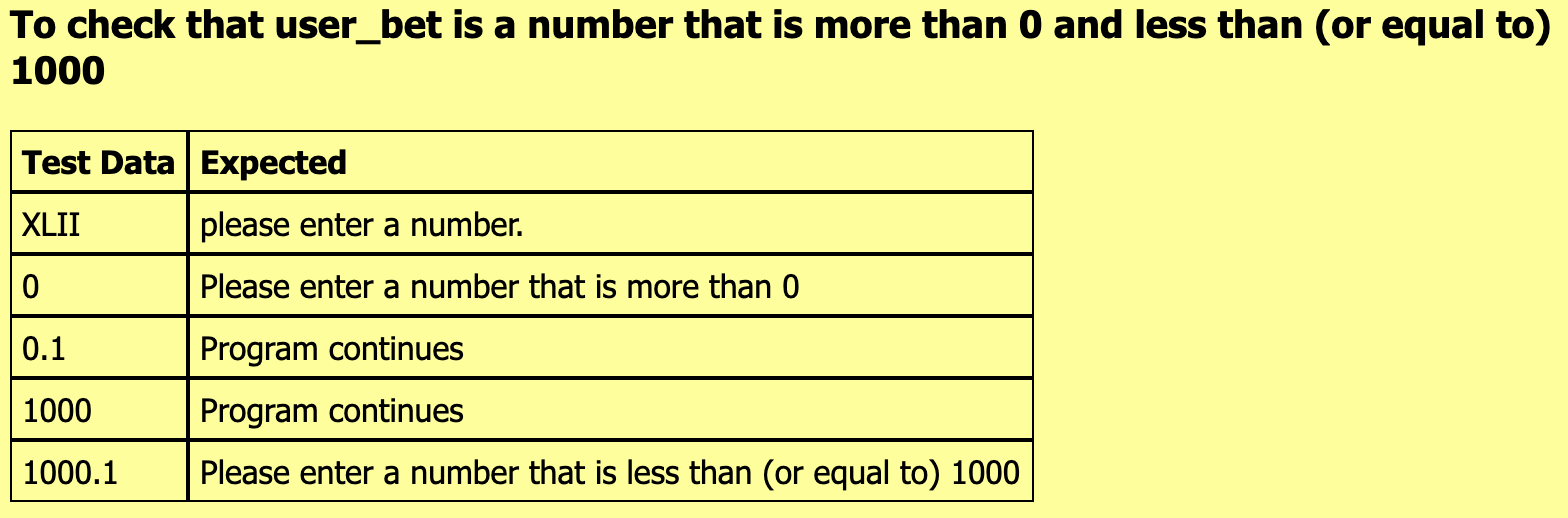
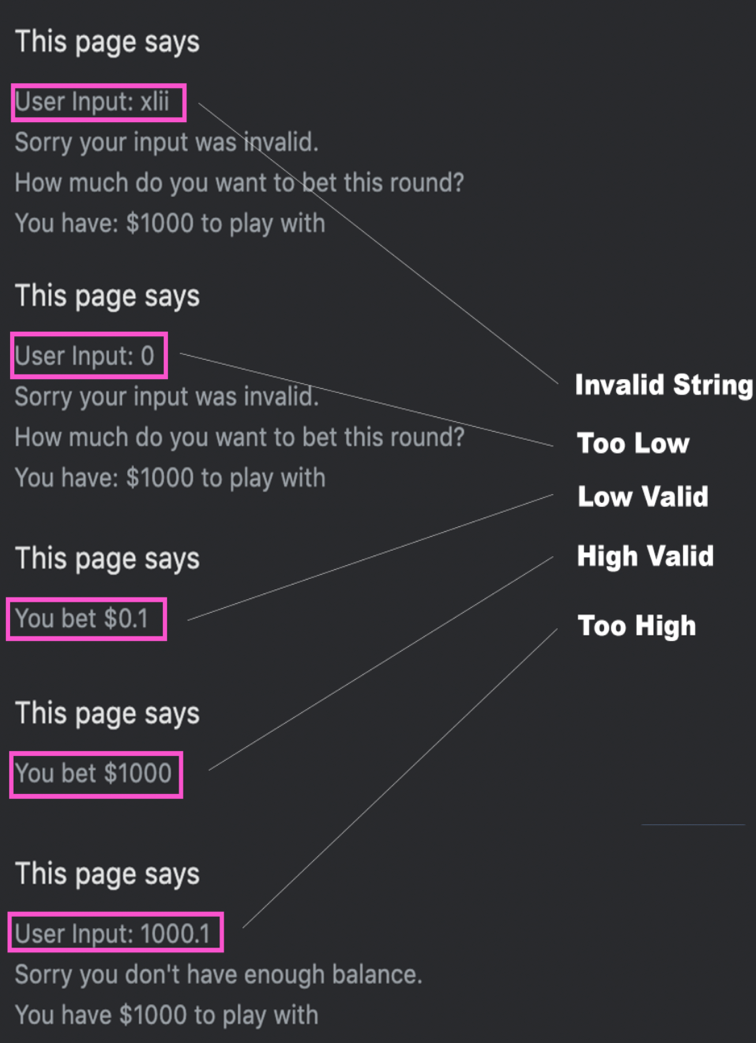
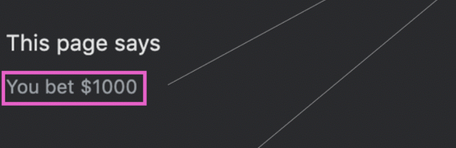
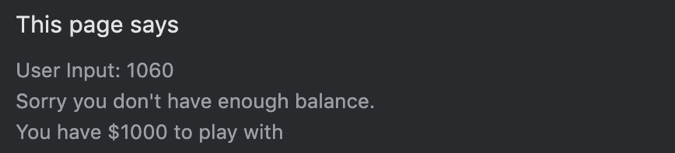
**Component 2: User deposit**

*****The numbers in purple have been hard coded in for testing purposes*

|  |  |
| --- | --- |
| **Test Data** | **Expected** |
| User balance: $50 | The program continues and alerts  “You now have $50”  <continues> |
| User deposit: $0 | Alerts “Please enter a valid amount  How much would you like to deposit?” |

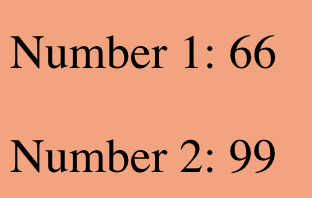
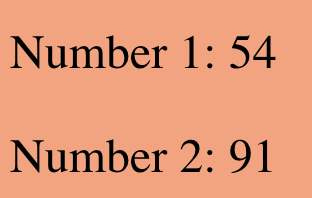
****

**Component 3: User bet**

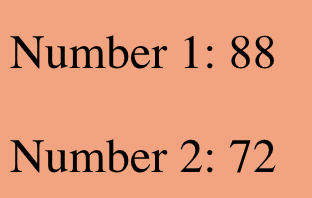
*The numbers in purple have been hard coded in for testing purposes*

|  |  |
| --- | --- |
| **Test Data** | **Expected** |
| User balance: $1000  User bet: $1000 | The program should continue and alert “You have placed $1000” |
| User balance: $1000  User bet: $1060 | The program will run into a loop until valid amount has been entered alerts “Sorry you don’t have enough balance to play with.  You have $1000” |

**Component 4: Random Number Generation**

****

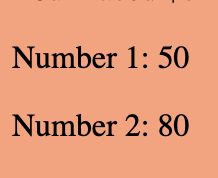
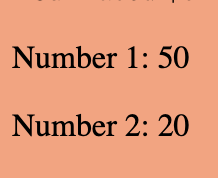
|  |  |
| --- | --- |
| **Test Data** | **Expected** |
| Low: 0  High: 100  Generate number 1 | The program should generate 1 number between 0 and 100 |
| Low: 0  High: 100  Generate number 2 | The program should generate 1 number between 0 and 100 |

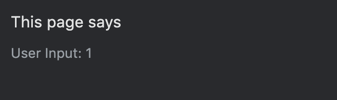
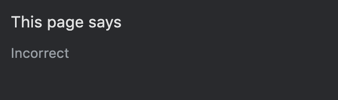
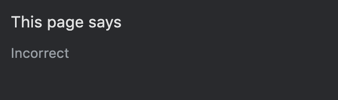
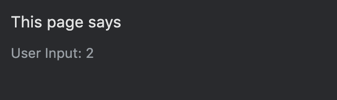
****

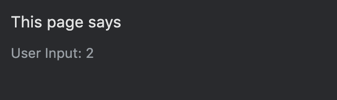
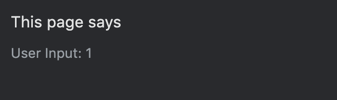
**Component 5: Compare user guess with numbers**

*The numbers in purple have been hard coded in for testing purposes*

|  |  |
| --- | --- |
| **Test Data** | **Expected** |
| Number 1: 50  Number 2: 80  Guess: High (1) | “Correct” |
| Guess: Low (2) | “Incorrect” |
| Number 1: 50  Number 2: 20  Guess: High (1) | “Incorrect” |
| Guess: Low (2) | “Correct” |

****

****

****

**Component 6: Win Multiplier**

*The numbers in purple have been hard coded in for testing purpose*

|  |  |
| --- | --- |
| **Test Data** | **Expected** |
| Number 1: 80  Number 2: 90  Bet: $10  Guess: High | 100-80=20 (20% of getting high)  100/20 = 5  highWin = bet\*5  winAmount = $50 |
| Number 1: 80  Number 2: 70  Bet: $10  Guess: Low | 100/80=1.25 (80% of getting low)  lowWin = bet\*1.25  winAmount = $12.5 |
| Number 1: 10  Number 2: 5  Bet: $10 Guess: Low | 100/10=10 (10% of getting low)  highWin = bet\*10  winAmount = $100 |
| Number 1: 10  Number 2: 15  Bet: $10  Guess: High | 100-10=90  100/80=1.11 (90% of getting high)  highWin = bet\*1.11  winAmount = $11.10 |

**Component 7: Win/Loss Mechanics**

*The numbers in purple have been hard coded in for testing purposes*

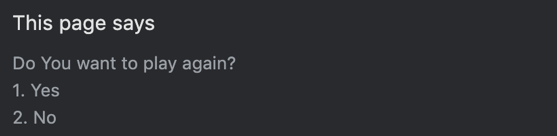
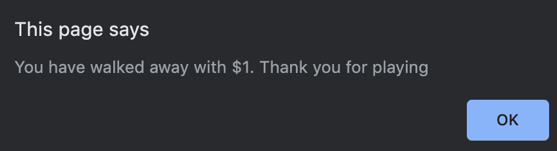
|  |  |
| --- | --- |
| **Test Data** | **Expected** |
| Start amount: $20  Bet Amount: $10  Number 1: 80  Number 2: 90  Guess: High | End amount: $60  Feedback: “Correct, you won $50”  Amount Left: “You have $60 remaining” |
| Start amount: $20  Bet Amount: $10  Number 1: 80  Number 2: 70  Guess: Low | End amount: $22.50  Feedback: “Correct, you won $12.50”  Amount Left: “You have $22.50 remaining” |
| Start amount: $20  Bet Amount: $10  Number 1: 80  Number 2: 70  Guess: High | End amount: $10  Feedback: “Incorrect, you didn’t win anything”  Amount Left: “You have $10 remaining” |
| Start amount: $20  Bet Amount: $10  Number 1: 80  Number 2: 90  Guess: Low | End amount: $10  Feedback: “Incorrect, you didn’t win anything”  Amount Left: “You have $10.00 remaining” |

**Component 8: Test End mechanics**

|  |  |
| --- | --- |
| **Test Data** | **Expected** |
| Start amount: $10  Bet amount $10  Number 1: 50  Number 2: 80  Guess: Low | End amount: $0  Feedback: “Sorry you don’t have any money remaining”  Farewell: “Thanks for playing.”  <ends> |
| **Number 1 set to 50 and number 2 set to 80 for testing purposes**  Start amount $10  Bet amount: $10  Guess: high  Again: <Ok>  **Number 1 set to 80 and number 2 set to 20 for testing purposes**  Start amount: $20  Bet amount: $20  Guess: Low  Again: <Cancel> | End amount: $20.00  Feedback: “Correct! You have won $20”  Amount Left: “You have $20 remaining”  Again: “Do you want to continue playing? <Ok> Yes, <Cancel> No”    End amount: $25.00  Feedback: “Correct! You have won $25”  Amount Left: “You have $25 remaining”  Again: “Do you want to continue playing? <Ok> Yes, <Cancel> No”    Farewell: “Thanks for playing. You walked away with $25”  <ends> |

**Component 8a: Test ‘2/quit’ etc**

|  |  |
| --- | --- |
| **Test Data** | **Expected** |
| Again? 2 | Game quits |

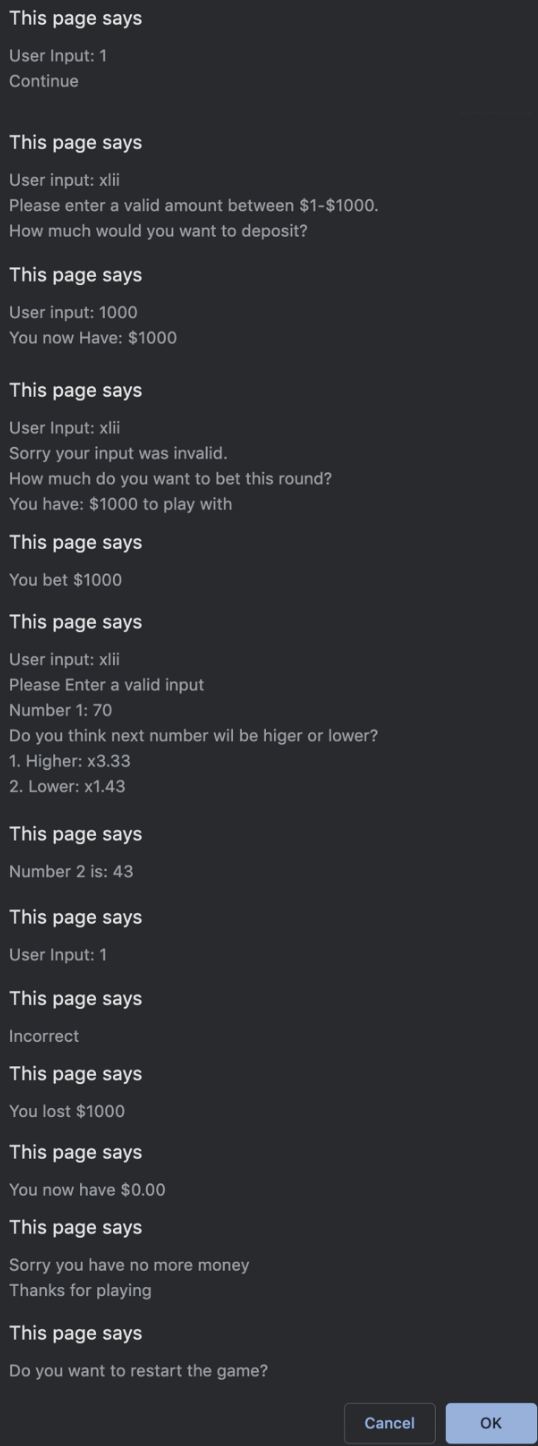
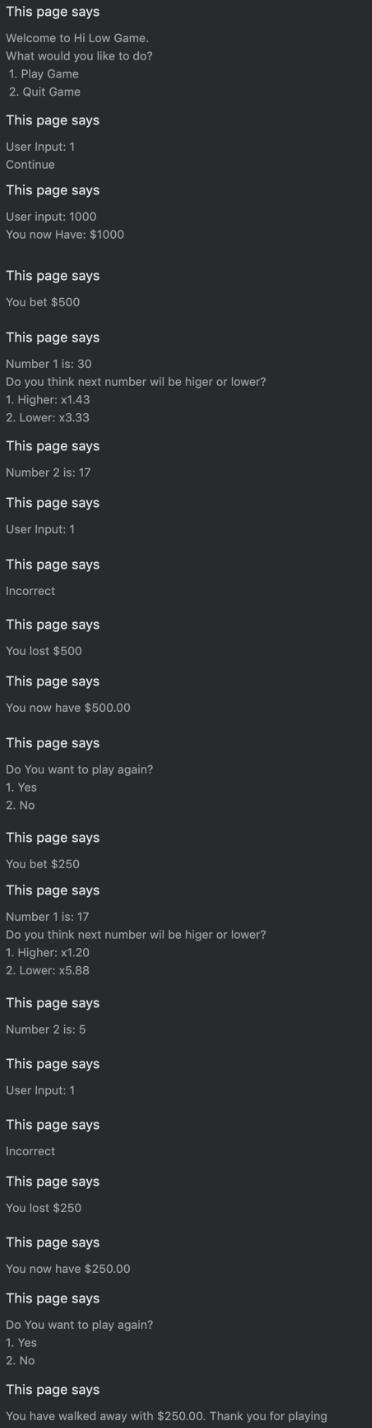


### Outcome Testing

*Please show testing for your assembled outcome below. This should include a test plan followed by screenshot proof*

*Money is always set to $1000 for testing purposes unless specified.*

|  |  |
| --- | --- |
| **Test Data** | **Expected** |
| What would you like to do? 1  How much would you like to deposit? xlii  How much would you like to deposit? 1000  How much would you like to bet? xlii  How much would you like to bet? 1000  Will the next number be higher or lower? xlii  Will the next number be higher or lower? 1 (Higher)  Would you like to play again? <cancel> | The game should continue to ask how much user wants to deposit.  The game should keep on asking how much they want to deposit until they enter a valid amount.  The game should then ask the user how much they wish to bet each round (amount >= balance), it should continue to ask until the user enters a valid amount. The game should allow the user to play again unless they run out of money.  The program should keep on asking the user if the next number will be higher or lower. This would continue until a valid option is entered.  The user would be asked if they wanted to play again or if they want to quit. (If user loses and has no more money, game would output farewell message and prompts if they want to restart the game) |
| Start with $1000, user plays 2 rounds and then quits | The program should generate random numbers and ask user if next number will be higher or lower. It should then allow the user to quit while he/she still have some money left. |

**

### Usability Testing

I got my sister to try my game out and below is the feedback I received…

* The game needed some instructions and introduction, she got confused at first but got the hang of it later.
* When you had won or loss there was too many popups.
* As she was playing the game, she found no problems itself. Although she had tried to exit the game by hitting cancel, but it wouldn’t work. Only when she was on the main menu.
* Color could be improved a little.

As a result of further testing and the feedback from my sister, I decided to change a few things with the game, I had made these following changes…

* Added a button which when clicked shows you a set of instructions of how to play the game and an example.
* Condensed the outcome so instead of it displaying each outcome individually and taking so long it displays all the different outputs in one alert.
* Change the background color and the text of the button to white which made it the same color as the outline.
* Also changed it so when you hover over buttons there is a transition from normal then it fades instead of instant.
* All changes made can be found in V2 of my game.

### Post Usability Test…

*Show that your post usability testing program works correctly*

|  |  |
| --- | --- |
| **Pre-Usability Test** | **Post Usability Test** |
|  |  |

### Social and End User Considerations…

**How did you ensure that your task was suitable for your chosen audience?**

*I ensured that my program is suitable for my chosen audience by making my game easy to understand the mechanics and how it’s play. I have also made changes to the outputs to make the game a bit quicker. Also, I made it suitable by using a volunteer to play test the game and based off their feedback I changed my game.*

*I made sure that it was suitable as I wanted my game to be played by everyone which meant it had to be simple, the changes I made to the outputs saves the users time as there isn’t as much prompts to click through.*

**How have you honored copyright?**

*I have honored copyright issues for my game as it includes no images that were made by others and throughout the game it does not include any trademarked names. I have also honored copyright as the code for the game was made from scratch and I used no extra code from others.*

*I wanted to honor copyright as I didn’t want to infringe on copyright, and I didn’t want to plagiarize something as it is not allowed when doing internals. I also wanted the code to be written by me and not anyone else*

**How did you make your game easy to use?**

*I made my game easy to use by reducing the amount outputs you have to go through and also adding some basic instructions that should make it clarify any problems on how to play. I also made my game a bit easier by using a volunteer to play test the game.*

*My final outcome of the game contains some of these features that should help playing the game a lot simpler:*

* *Changed the accepted input to either “1” or “2” for options. This made it easier instead of typing “high” or “low”, it also makes it more efficient as you won’t have to type as much characters.*
* *If the user at any point enters an invalid input, it will display a clear error message which will ask them to try again.*
* *At the end of each round played, it shows the user how much money they have left. It also asks them if they want to continue playing and start a new round or to quit and walk away with their money*

*I wanted to make my game easier for the end user to play as this will allow a wider range of users to play and it would make the game simpler to understand. It also saves time but condensing the outputs, this makes the game more enjoyable for the user to play*