Test ID	Description	Expected Results	Actual Results

Test #1
 	
testID: testCompress
 	
Strategy: Equivalence Class - Test Loading a test file and Compressing it [UC 2]
	Preconditions:
	
	• The user has started the CompressionManager UI (UC 1)
	• The user can input the `decompressed.txt` into the UI's prompt
	
	Steps:
	
	1. **The user has started UI and types in "`input/decompressed.txt`" as the file path**
	2. **The user types in the command "`compress`"**
	3. **UI displays the compressed version of the file** `decompressed.txt`
	4. **The user then types in the command "`quit`" and the application terminates**
		The CompressionManager displays the following
		
		`Compressed Output {`
		
		`  Line 1:One fish Two 2 Red 2 Blue 2`
		
		`  Line 2:Black 2 5 2 Old 2 New 2`
		
		`  Line 3:This one has a little car`
		
		`  Line 4:9 10 11 12 13 star`
		
		`  Line 5:Say What 12 lot of 2 there are`
		`}`
		
		The program then terminates after the quit command
			`Compressed Output {`
			`  Line 1:One fish Two 2 Red 2 Blue 2`
			`  Line 2:Black 2 5 2 Old 2 New 2`
			`  Line 3:This one has a little car`
			`  Line 4:9 10 11 12 13 star`
			`  Line 5:Say What 12 lot of 2 there are`
			`}`
			
			Was output to the console
			
			The program ended after typing quit

Test #2
 	
testID: testDecompress
 	
Strategy: Equivalence Class - Test Loading a text file and decompressing it [UC 3]
	Preconditions:
	
	• The user has started the CompressionManagerUI (UC 1)
	• The user can input the `compressed.txt` into the UI's prompt
	
	Steps:
	
	1. **The user has started the UI and types in** `"input/compressed.txt"` **as the file path**
	2. **The user types in the command "**`decompress`**"**
	3. **The UI displays the decompressed version of** `compressed.txt`
	4. **The user then types in the command "**`quit`**" and the application terminates**
		The CompressionManager displays the following
		
		`Decompressed Output {`
		
		`  Line 1:One fish Two fish Red fish Blue fish`
		
		`  Line 2:Black fish Blue fish Old fish New fish`
		
		`  Line 3:This one has a little car`
		
		`  Line 4:This one has a little star`
		
		`  Line 5:Say what a lot of fish there are`
		`}`
		
		The program then terminates after the quit command
			`Decompressed Output {`
			`  Line 1:One fish Two fish Red fish Blue fish`
			`  Line 2:Black fish Blue fish Old fish New fish`
			`  Line 3:This one has a little car`
			`  Line 4:This one has a little star`
			`  Line 5:Say What a lot of fish there are`
			`}`
			Was output to console
			
			The program ended after typing quit

Test #3
 	
testID:
 	
testEmptyFile
 	
Strategy: Unexpected Input - Test handling an unexpected empty text file [UC 2,3 E 1]
	Preconditions:
	
	• The user has started the CompressionManagerUI
	• The user can input the `empty.txt` into the UI
	
	Steps:
	
	1. **The user has started the UI and types "`input/empty.txt`" as the file path**
	2. **The user types in the command "`compress`" and notes the output**
	3. **The user types in the command "`decompress`" and notes the output**
	4. **The user then types in the command "`quit`" and the application terminates**
		The messages should be output by the UI for the following commands.
		
		For compress
		
		"The provided input file has no text to compress"
		
		For decompress
		
		"The provided input file has no text to decompress"
		
		The program then terminates after the quit command
			`The provided input file has no text to compress.`
			
			`The provided input file has no text to decompress.`
			
			These messages are displayed in the console and the program continues to run until quit

Test #4 **testID:** **testNoFile** **Strategy:** **Unexpected Input - Test an unexpected value of selecting a file that does not exist [UC 1 E 1]**	**Preconditions:** • The user has started the CompressionManagerUI • The user can input a file that does not exist **Steps:** **1. The user types in the location of a file that does not exist in the file explorer.** **a. Eg (**`"no-file.txt"`**)** **2. The UI displays a prompt to the user and terminates on its own.**	The following messaged is displayed to the console and terminates "The provided input file is empty"	`The provided input file is empty.` The program then terminates after the message is displayed to console

Test #5 **testID: testQuit** **Strategy: Equivalence Class - Test that the UI properly closes the application [UC 4]**	**Preconditions:** • The user has selected `compressed.txt` as the input file and has started the CompressionManagerUI • The CompressionManagerUI is currently running (UC 1) **Steps:** 1. **The user has started the CompressionManager application and typed in a valid file path** 2. **The user types in the command "**`quit`"	Nothing is printed to the console other than the prompt asking for a command	`Compress or Decompress the file? Type "quit" to quit.` This message is printed to the console. After inputting quit the program terminates.

Test #6
 	
testID: testNoCompression
 	
Strategy: Boundary Value - Test a boundary case where no compression is needed [UC 2]
	Preconditions:
	
	• The user has started the CompressionManagerUI
	• The user can input the `unique.txt` into the UI
	
	Steps:
	
	1. **The user has started the UI and selects the** `unique.txt`
	2. **The user types in the command "**`compress`**"**
	3. **The CompressionManager UI displays** `unique.txt` **with no changes**
		The CompressionManager displays the following
		
		`Compressed Output {`
		
		`  Line 1: This has only unique words`
		
		`  Line 2: No repeats here`
		
		`  Line 3: So no changes will be made`
		`}`
			`Compressed Output {`
			`  Line 1:This has only unique words`
			`  Line 2:No repeats here`
			`  Line 3:so no changes will be made`
			`}`
			
			This is output to the console and the program terminates after typing in quit.

Test #7
 	
testID: testNoDecompression
 	
Strategy: Boundary Value - Test a boundary case where no compression is needed [UC 3]
	Preconditions:
	
	• The user has started the CompressionManagerUI
	• The user can input the `unique.txt` into the UI
	
	Steps:
	
	1. **The user has started the UI and selects the** `unique.txt`
	2. **The user types in the command "**`decompress`**"**
	3. **The CompressionManagerUI displays** `unique.txt` **with no changes**
		The CompressionManager displays the following
		
		`Decompressed Output {`
		
		`  Line 1: This has only unique words`
		
		`  Line 2: No repeats here`
		
		`  Line 3: So no changes will be made`
			`Decompressed Output {`
			`  Line 1:This has only unique words`
			`  Line 2:No repeats here`
			`  Line 3:so no changes will be made`
			`}`
			
			This is output to the console and the program terminates after typing in quit.

Test #8
 	
testID: ManyCompresses
 	
Strategy: Boundary Value - Test a boundary case the same file is compressed multiple times[UC 2]
	Preconditions:
	
	• The user has started the CompressionManager UI (UC 1)
	• The user can input the `decompressed.txt` into the UI's prompt
	
	Steps:
	
	1. **The user has started UI and types in "`input/decompressed.txt`" as the file path**
	2. **The user types in the command "`compress`"**
	3. **The user types in the command "`compress`"**
	4. **The user types in the command "`compress`"**
	5. **UI displays the compressed version of the file thrice** `decompressed.txt`
	6. **The user then types in the command "`quit`" and the application terminates**
		The CompressionManager displays the following three times with no changes between every iteration.
		
		`Compressed Output {`
		
		`  Line 1:One fish Two 2 Red 2 Blue 2`
		
		`  Line 2:Black 2 5 2 Old 2 New 2`
		
		`  Line 3:This one has a little car`
		
		`  Line 4:9 10 11 12 13 star`
		
		`  Line 5:Say What 12 lot of 2 there are`
		`}`
		
		The program then terminates after quit.
			`Compressed Output {`
			`  Line 1:One fish Two 2 Red 2 Blue 2`
			`  Line 2:Black 2 5 2 Old 2 New 2`
			`  Line 3:This one has a little car`
			`  Line 4:9 10 11 12 13 star`
			`  Line 5:Say What 12 lot of 2 there are`
			`}`
			
			`Compressed Output {`
			`  Line 1:One fish Two 2 Red 2 Blue 2`
			`  Line 2:Black 2 5 2 Old 2 New 2`
			`  Line 3:This one has a little car`
			`  Line 4:9 10 11 12 13 star`
			`  Line 5:Say What 12 lot of 2 there are`
			`}`
			
			`Compressed Output {`
			`  Line 1:One fish Two 2 Red 2 Blue 2`
			`  Line 2:Black 2 5 2 Old 2 New 2`
			`  Line 3:This one has a little car`
			`  Line 4:9 10 11 12 13 star`
			`  Line 5:Say What 12 lot of 2 there are`
			`}`
			
			Is displayed with no changes between iterations

Test #8
 	
testID: ManyDecompresses
 	
Strategy: Boundary Value - Test a boundary case the same file is decompressed multiple times[UC 3]
	Preconditions:
	
	• The user has started the CompressionManager UI (UC 1)
	• The user can input the `decompressed.txt` into the UI's prompt
	
	Steps:
	
	1. **The user has started UI and types in "`input/compressed.txt`" as the file path**
	2. **The user types in the command "`decompress`"**
	3. **The user types in the command "`decompress`"**
	4. **The user types in the command "`decompress`"**
	5. **UI displays the decompressed version of the file thrice** `compressed.txt`
	6. **The user then types in the command "`quit`" and the application terminates**
		The CompressionManager displays the following
		
		`Decompressed Output {`
		
		`  Line 1:One fish Two fish Red fish Blue fish`
		
		`  Line 2:Black fish Blue fish Old fish New fish`
		
		`  Line 3:This one has a little car`
		
		`  Line 4:This one has a little star`
		
		`  Line 5:Say what a lot of fish there are`
		`}`
		
		The program then terminates after quit.
			`Decompressed Output {`
			`  Line 1:One fish Two fish Red fish Blue fish`
			`  Line 2:Black fish Blue fish Old fish New fish`
			`  Line 3:This one has a little car`
			`  Line 4:This one has a little star`
			`  Line 5:Say What a lot of fish there are`
			`}`
			
			`Decompressed Output {`
			`  Line 1:One fish Two fish Red fish Blue fish`
			`  Line 2:Black fish Blue fish Old fish New fish`
			`  Line 3:This one has a little car`
			`  Line 4:This one has a little star`
			`  Line 5:Say What a lot of fish there are`
			`}`
			
			`Decompressed Output {`
			`  Line 1:One fish Two fish Red fish Blue fish`
			`  Line 2:Black fish Blue fish Old fish New fish`
			`  Line 3:This one has a little car`
			`  Line 4:This one has a little star`
			`  Line 5:Say What a lot of fish there are`
			`}`
			
			Is displayed with no changes between iterations