1. **The program should be as crash-proof as possible. You must submit a test plan and test data which demonstrates how you checked for stability.**

Step One: Prioritize features and sections used the most often

The sections of my codet that I spent the most time checking stability were the most visible, the most frequently used and the ones that had the highest possible rate of failure. Using these three variables I ranked what features I spent the most testing, and which ones still may have some issues.

Step Two: Follow Through with Three Stages of Testing

Since the file input/output are most likely to cause issues that will crash the program I spend the majority of my time trying to fix them. I ran on three types of test cases: no data in the files, a valid data file, and an invalid data file. Use the following csv files in the folders named the following NoData, ValidData, and InValidData. [Not these are included in the zip file and are the required test data, however due to the lack of time, the InValidData is not included as the program is not created to handle those issues]

Step Three: Lower Priorities

The next issues after file input/output are rarely large enough to crash the GUI (which makes it noticeable to the user). These includes purposely setting inputs from buttons and textfields to things that do not make logical sense. However, my code is filled with many tries and catches and if statements to catch invalid inputs to make sure that the code never tries to use invalid input in a method created. This includes changing input from simply ints to longs (to make it standardized) and some lists to ArrayLists (to make sure there are no null pointers).

Step Four: Looking back

The stability of the program is indeed ensured however I'm sure despite the thought I put into continuous testing there may be some bugs throughout the code but I'm certain they are not large enough to crash the GUI, but they may cause some displaying to be messed up. However, if I had more time I would spend more like making a proper search for the jobs as my search displays all results that have even one of the filters, not all of them.

2. **A user manual file in PDF format providing an instruction manual for using the program.**

Step One: Login or Sign Up

Launch the "UserInterface" class and wait for the home page to load up. If you wish to create a new account feel free to complete the sign up page. If not, feel free to use the login in: admin1 to admin20 for both username and password. (I recommend admin1 or admin2 as they have a message strand with each other)

Step Two: Test out the Features

Go to the profile page and double check the information in the Employees CSV file. Send a message to any other user that exists and go to the view past messages to check the message is being received. Open the Message CSV file to see the message is being encrypted. Go to the jobs menu and click view. Use the filter to search different things such as job title, location, availability and rank. (wage and requirement are not included). Go to the homepage and see the stories being displayed. (Note the order then close the browser and reopen). Can you spot the difference?

Step Three: Logout

At the top right corner, there is a logout hyperlink. Feel free to logout (without restarting the browser) and logging in with another account again.

3. What was completed and what was not

Unfortunately I was unable to complete:

- Procedure Help (regular jobs versus dangerous jobs)
- Transferring Money (see funding for specific trips versus buying illegal items)
- Jobs to do (regular to do list versus missions)

However I was able to make a sign up and login that is very stable and makes sure that everything created or is happening is correctly formatted. Of the classes that I did make, I make thoroughly follow through with my plan as they used variables that logically made sense.