Testing:

Our program has gone under severe testing to ensure that it is bug and error free. To ensure that the program is bug free, we have implemented multiple testing listed in the table below

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type of test | Why | How | Result | Normal cases | Extreme cases |
| White Box Testing | To ensure that the input and output are correct | We have put multiple if else and print () statement in the program. This is to ensure that the during the coding period, we can make sure that the functions works while coding. | The result and expectation match every time I finish a function which is good for the later stages of the program | passed | N/A |
| Random Testing | To ensure that the functions will work with any kind of inputs and outputs | test all functions and compare it with its output. We have done it just by calling the function and input values of our own. We also used junits to help us test the codes. Using the input and expected output to see if we can pass all tests | Both ways of the testing gave us pleasing results. We passed all the tests, and it is a good sign that our program is in good condition | Passed | Passed |
| Partition Testing | To ensure that the functions will work with any kind of inputs and outputs that represent different kind of inputs. | This can be seen in the testing.py too. I have randomly selected some tests and chosen some specific tests. The specific tests are mainly to check whether the code works for that specific partition. For example, if Kraaifontein to Brackenfell generate the output that is to our expectation. Then any start and end location on the area north train route will work. | The results are pleasing because all functions worked as expected. We used normal and extreme cases and they are all able to run effectively | Passed | Passed |
| Acceptance test | To see whether the user will be able to use the program to their satisfaction | We have manually checked whether the output created is correct and able to use. We see if the shortest time is the shortest if the closest time to depart is the closest time. | When we found a something that we do not like about the output, we tried to change our code so that the output is acceptable | Passed | Passed |
| Use-case based black box: | To ensure the customers satisfactory | We have run the website on different computers to ensure that it works with them. We have ask multiple people to use the application to hear from their feedback. | The website is able to function just fine regardless of the computer type and speed. From their feedback, we strive for better website for our customers. | Passed | Passed |
| Recovery test | To ensure that data is not lost due to break down or accidental close of the application | We do many different testings, once bug is found, we fix them immediately. We tried to store and close website immediately, check if the database was able to record the data. | The program is working better every time we fix a bug. Everything is recorded in the database so nothing is lose when website is closed suddenly after the clicking of the button | Passed | Passed |
| Security test | To ensure the privacy of the users | we have created a user and password protection to ensure that all users have their own account if they want it. We tried to log in with different accounting using different passwords. Check if the user is able to log in. | When the user an username with wrong password, there will be an error preventing them from accessing the other account’s data | Passed | N/A |
| Performance test | To improve efficiency | We import time.  User time.start() and time.time() to measure the time taken to execute different inputs | Simon’s Town – Strand on Monday took 0.006  Mowbray – Stellenbosch on Saturday took 0.005  Cape Town to Avondale on Monday took 0.004.  The time normally won’t go over 0.01 seconds | Passed | Passed |

