**Section 4: Evaluating our Product**

We seek to evaluate our product by considering how far it satisfies our customers’ requirements. In particular, earlier in Section 2 we constructed a Value Proposition Canvas, considering various benefits we can provide to customers (both in the sense of gain creators and pain relievers) that seek to address these requirements. The three main benefits that we have identified are as follows:

1. Our application should be simple and easy to use.
2. Our application works without the user needing to know *precisely* where his tasks can be accomplished.
3. Our application helps users plan routes that are efficient and optimal or near-optimal.

Thus, to some extent at least we can measure the success of meeting customers’ requirements by determining how successful our application is in providing said benefits.

**4.1: Simplicity and Ease of Use**

This benefit was motivated by users’ concerns that our routing application might be too difficult or inconvenient to use. As we have a sound knowledge of our application’s user interface, it is difficult for us to fairly assess this on our own; thus, we have sought feedback from various users in evaluating.

We are meeting with our PS on a weekly basis, and use the demo environments mentioned in Section 3 to show her the front-ends that we are providing to the user. During these meetings, a member of the team (usually Jeremy) will take careful, detailed notes as to her reaction to our work, especially if we notice that she has any difficulties in navigating the user interface.

In addition, we are also performing hallway testing with users from the various stakeholder groups to help ascertain if our product is usable and/or if there seems to be a customer interest in it. Typically, these tests involve asking users to carry out certain tasks on the application (for example, in the case of students, a possible scenario could be “use the application to help you post a letter, visit a bank and grab a coffee during your lunch break”.) We consider the time users take to complete these tasks (lower times are generally better), determine if they’re able to understand the output presented, as well as note if they struggle on any particular parts of the task.

A possible extension of this which would scale to a larger user-base could be to incorporate automated logging features into our front-end, so that timing data is automatically collected when users use the application (such as computing the time taken for the user to add each task, find the map button, *etc.*) for later analysis.

To some extent, system tests also play a part in determining whether our system is easy to use. We expect that busy users are likely to find slow systems inconvenient to use, and thus we plan to use tests which monitor the performance (service time) of our system and fail should a significant performance regression arise.

**4.2: Determining Locations to Complete Tasks**

This benefit was motivated by users’ desires to find new places to accomplish tasks, especially when this would mean that their errands could be completed more efficiently. It was also motivated by gaps in users’ existing knowledge of locations where they can perform said tasks, which limited their ability to plan the most efficient routes.

To some extent, we are evaluating our system’s performance in this dimension, by using system tests of our back-end that are driven by existing domain knowledge. For example, many team members personally know the area around South Kensington and Gloucester Road fairly well, and can identify that there are several places which individuals can go to complete certain tasks – we can check that these points are returned by the algorithm. During the course of our hallway tests, we also ask users whether they notice any deficiencies/problems with regard to the locations identified.

We acknowledge that it is not possible for our system to capture *every* possible task that a user might want to do. In order to alleviate this, we allow users to indicate specific points (where they *know* they can do what they need to) to include in the route. We also plan on developing a feedback system which allows users to suggest possible future tasks that we should add support for.

**4.3: Planning Optimal / Efficient Routes**

This benefit was motivated by users’ needs to plan itineraries to allow them to complete tasks within time limits, as well as their desires to accomplish said tasks as efficiently as possible.

Much of our strategy for evaluating the correctness of our back-end logic involves the use of automated tests to verify the accuracy and reasonability of our code. In particular, a large part of the algorithm underlying our system involves solving the Travelling Salesman Problem (TSP). This is an NP-hard problem and thus we presently do not know of algorithms that solve it more quickly than in exponential time. However, as speed is important (explained in 4.1), we consider finding a route that is fairly close to the optimal solution acceptable. As a first step, we have ensured that certain invariants are preserved (for example, that the final route is some permutation of the input set of points, and contains at least 1 location where each errand can be performed). We have also constructed several simple cases for which we have manually determined the optimal solution.

Thankfully, as the TSP is a widely-studied problem, there are numerous known TSP problems available online that are used as benchmarks for TSP solvers. We plan to compare the output of our algorithm with the best results achieved by state-of-the-art algorithms; as long as our algorithm’s output is within a tolerance margin (of say 10%) from the best found solution, we consider it acceptable. Our algorithm is also not as likely to deal with large numbers of locations (say *n* = 10000), so for efficiency of running the tests we have not considered very large instances of the problem.