QEC Programming Challenge: Pizza Pal

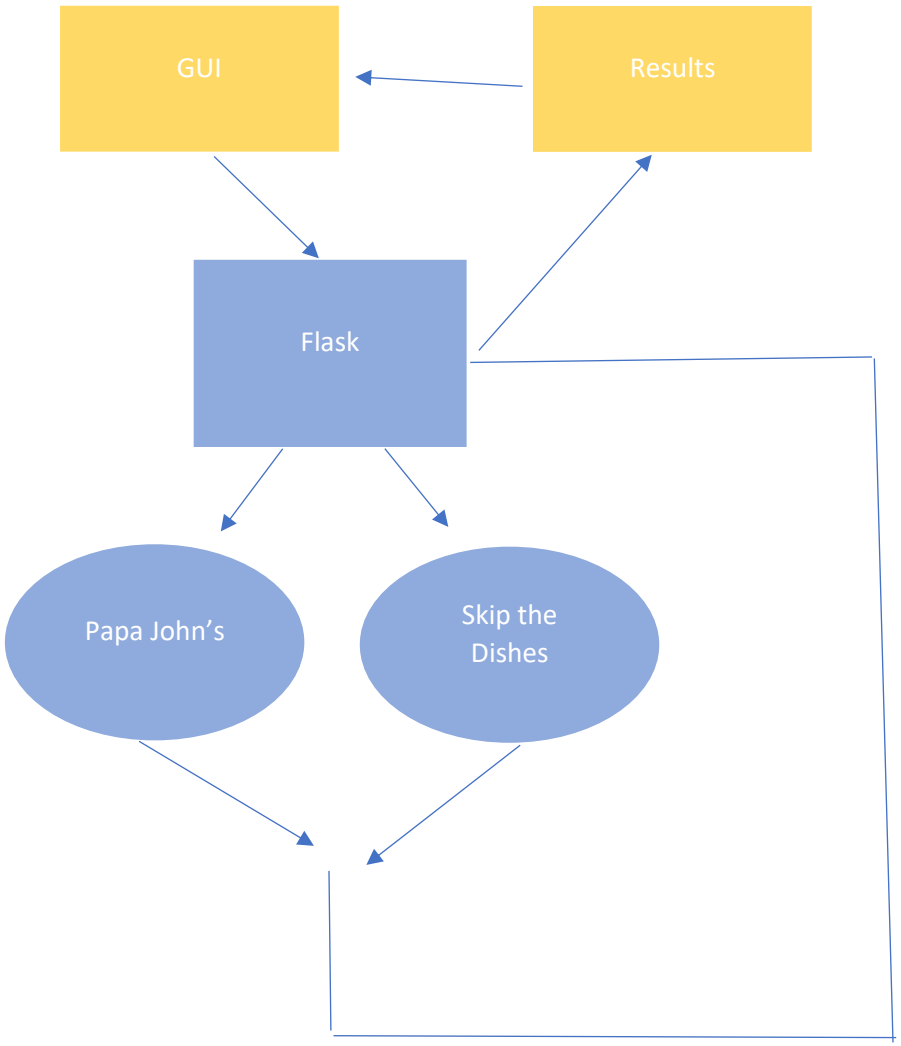
Team 8: Nicole Ooi, Patrick Lenover, Andrew Cramp, Ozzie Kirkby

1. Functionality:

This documentation is written in accordance with the QEC Programming Challenge Outlines 2018. The solution presented is a web-based application permitting users to find the cheapest pizza within their area. Users may choose the number of pizzas, size, number of toppings, and the search radius for an establishment – as it affects the delivery charge. Users must input their delivery address on the website. The following table shows the methods and languages used to implement the solution.

Front-end	HTML, CSS
Back-end	Python

1.1 Steps:



1.3 GUI:

As outlined above, the GUI will be a website written using HTML and CSS. These files were simply used to render the browser. No Javascript was used in connecting the front-end to the back-end. The user will input all required details on the website. An HTML form was used to encompass the state of the input elements. Upon hitting submit, all elements in the form will be transferred to Flask through a post method. The street address input was converted to longitude and latitude using MapBox.

1.4 Flask:

Flask was used as the framework for running the python server. This is how all components of the solution communicate with each other. The GUI serves as the input interface for the user and it gets transferred through the Flask server to connect to the web-scraping modules.

1.5 Web Scraping:

Beautifulsoup and Selenium are used to scrape the webpages of Papa John's and Skip the Dishes – based on the front-end user inputs. This information is ordered and output to the results page using Flask.

1.6 Results:

The ordered pizza options are displayed for the user. The user cannot order from the solution – they must order from the vendor listed.

2. Testing

The team tested the solution by manually inputting different cases and checking that the options truly were the cheapest. More thorough testing is included in the recommendations.

3. Recommendations:

The team recommends more rigorous testing be conducted on the solution. Unit testing would be required to test the code's logic and that it covers all conditions. PyUnit is the recommended library for this testing. Before the solution is released to the public, a bot would be needed to hit the flask server/endpoint to simulate a real user. This would ensure that the solution can handle the traffic of a real user. Additionally, the solution should implement asynchronous requests for the web-scraping as it is more efficient.

To make the solution more accurate and accessible, the website would need more input options such as multiple pizza sizes for different numbers of pizza – then adding the total. There should be more criteria as well such as the quantifiable size of the pizza because this differs per vendor. In order to accurately scrape the webpages and take the differing sizes into account, natural language processing would need to be implemented.

Currently, the solution is only functional for Skip the Dishes in Kingston. This should be broadened to Uber Eats and Skip the Dishes for a larger region such as Canada. Uber Eats is preferred to Papa John's because it handles different vendors in itself by nature of its service.