Ulysses

Technical Requirements Document

Version 1.0

November 19th, 2018

Anastasiia Roldugina: Project Manager / Team Lead

Tamara Saldina: User Interface Designer

Jordan Pike: User Experience Designer

Jerome Ching: Software Developer

Morgan Gill: Quality Assurance

Prepared for:

COMP3059 CAPSTONE PROJECT 1

Instructor: Professor Anjana Shah

Fall 2018

Ulysses aims to take advantage of several primary subsystems that will incorporate a variety of different types of software in order to realize its functionality to the fullest of its potential.

The web application that the majority of clients will use shall be heavily leveraging HTML, CSS, JavaScript and SASS in order to provide the user with a clean interface and responsive experience. CSS and SASS will allow us to provide clean and stylized interfaces for our users in order to enable them to navigate quickly and intuitively. JavaScript will allow us to incorporate event triggers and client-side validation in order to lighten the processor load on our server.

The incorporation of an address prediction bot via “U.I. Path Bots” will allow us to harness machine learning in order to aid in auto completion of address information provided by the user. This will alleviate the user of some of the input required, allowing them to save time while still being just as productive.

The desktop application that the qualified industry individuals (architects, engineers, etc…) will be using shall be running primarily on Java. Java has an immense amount of code libraries and resources incorporated within it that will allow us to heavily tailor Ulysses and fine-tune it to our stringent specifications. Through Java, we will be able to incorporate many security and efficiency features such as data encapsulation, as well as synchronous and asynchronous resource allocation and access in addition to multi-thread task execution. (in order to ensure a high level of system compatibility.)

The server shall be run largely on PHP incorporating “Laravel” in order to handle server requests, responses and database queries that may be sent by clients or passed to the database by users of the desktop application. We chose Laravel largely based upon its “Model View Controller” architectural pattern. This makes it easier to integrate in a modular fashion which has the added benefit of easing the process of updating the system and providing on-going maintenance. Laravel also includes various interfaces through which we can access relational databases, providing even more freedom to development staff.

And finally, the database will be comprised of an SQL relational database following a strict schema following the conventions of “3rd Normal Form” data storage doctrine. This will allow us to reduce data redundancy, saving space and resources on our database. This carries the additional benefit of increasing memory efficiency, reducing the amount of maintenance and additional supervision of the database.