Rural Grocery Network Testing Documentation

Prepared by:

Colin Cope
Zachery Brunner
Ali Alhamadah
Kevin Nolde

05/09/2020

TABLE OF CONTENTS

1.0 INTRODUCTION

This document outlines the testing of the Rural Grocery Network website. The mission of the Rural Grocery Initiative (RGI) is to provide resources to help sustain and enhance independently-owned rural grocery stores. RGI assists communities and citizens to strengthen rural grocery operations and improve access to healthy foods.

This application is meant to aid rural grocers by optimizing their distribution networks. It takes in information about stores or other distribution points of interest and will calculate the best way to load trucks, the order to distribute to stores in, etc. The application gives the user control to manipulate nodes (a store or distribution center), trucks, demand, and more. To accomplish the routing portion of the calculations, the application utilizes ArcGIS Online, a service that can find the best routes between nodes on a map.

The end goal of this application is to provide rural grocery store owners a tool to easily figure out the best route for a truck for a shared delivery.

2.0 OBJECTIVES AND TASKS

2.1 Objectives

This document aims to outline the expected testing practices and suggested methods for going about testing and goes over the important sections of the application that will need testing.

3.0 SCOPE

General:

The core sections of this application that will need to be tested are the login functionality and the scenario import functionality. Data that is being saved to the database should also be checked to make sure extraneous data is not going with it.

Tactics:

Testing of the login functionality and other parts of the UI can be tested using Selenium which is a portable framework for testing web applications. Some tests written with selenium are already included with the project and can be used as an example. The data that is being sent to the database can be tested with unit tests.

4.0 TESTING STRATEGY

4.1 Unit Testing

Unit testing can be completed used xUnit in the RuralGroceryNetwork.Test project. bUnit is also included as a dependency and can be used to test UI elements of Blazor apps.

4.2 Integration and System Testing

This website is designed to be run on Ubuntu 18.04 LTS and should be tested on it as well. Integration testing of the application should be to test the completed system such as whether or not proper UI elements are being displayed on the correct pages. These tests can be conducted on a working version of

the website using Selenium which can record and repeat user actions do automatically verify if the functionality is still consistent with how it was intended.

4.3 User Acceptance Testing
The purpose of acceptance testing is to confirm that the system is ready for operational use. During acceptance testing, end-users (customers) of the system compare the system to its initial requirements.

5.0 HARDWARE REQUIREMENTS
Server: Azure App Service

6.0 ENVIRONMENT REQUIREMENTS
6.1 Test Environment
Tests should be run on Linux, specifically Ubuntu, to make sure it closely resembles the actual server environment of the application. Tests using Selenium can be run on the hosted version of the website so that they are as accurate as possible.

7.0 TEST SCHEDULE
Testing should keep up with production. Meaning that unit tests should be written along with development, or the tests should be written first and the code can be written to comply with the tests. Testing of the UI should also be done along with development instead of done all at once to make sure that features are not being written on top of broken code.

8.0 CONTROL PROCEDURES
Bug Reporting:
Bugs should be reported using the bug tracker built into the Github repo. Reporting bugs on GitHub allows for the bugs to be easily tracked and can be closed with a git commit message.

9.0 FEATURES TO BE TESTED
- Login functionality
- Registration functionality
- Importing scenarios from CSV file
- Route calculation

10.0 FEATURES NOT TO BE TESTED
- None

11.0 RISKS/ASSUMPTIONS
- Assumes that testing all parts of the application is possible
- Failing to test some parts could fail to catch if features are missing, incomplete, or not working fully
- Assumes that application is in a working state

12.0 TOOLS

- xUnit
- bUnit
- Moq
- Selenium
- Github Actions
- Github Bug Tracker