Kasim Rashid

Jackie Giang

Trevor Holm

Davis Ramirez

Carter Alemania

The Generic Company Test Plan

Test Plan ID: 1

Purpose of Test Plan: Design a well suited GUI application that supports the intentions and purposes for College Touring. We will be testing that there are no errors present throughout the software process in order to ensure that the program runs successfully and meets all of its requirements.

Scope of Test Plan: We will be testing the existing contents and requirements of the given agile stories. Some of the story testing represents the following:

- Based on the college student's input, validate the legitimacy of the calculations pertaining to the souvenirs chosen.
- Based on the college student's input, validate the legitimacy of the calculations pertaining to the distances between campuses.
- From an administrative perspective, when adding a college campus, confirm the existence of the campus through a message box.
- Validate the functionality for the widget connections. Confirm that it will switch to the correct widget based on the story.
- The most logical variable types and correct values are used for the different tables in the

database. The values must be linear to the values given on the excel sheet.

Overall test strategy:

- Team members will notify the Product Owner once they have completed the various tasks given via story.
- The Product Owner will go through a thorough bug/error-check to test the developer's code. All tasks on the story must be completed.
- If the Product Owner encounters bugs/error, the Product Owner must notify the developer immediately. From there, the developer must debug until the bugs/errors are non-existent.
- Each developer must follow the definition of finished code. In other words, the code must meet all the requirements given via story and properly error checking user inputs/outputs/boundary values.

What features will be tested from a user's perspective:

- The correct storage of information within the tables in the database.
- The correct display of information from the tables within the database.
- Each widget has their own functionality and works properly.
- Sending and receiving of information from the database
- The addition of campuses or souvenirs to the database from a txt file
- The deletion or addition of souvenirs based on existing campuses in the database
- Valid and invalid inputs and the outputs each of them produce.
- The total distance and the most efficient order of the campuses the user

selected.

What features will not be tested from a user's perspective and what the system does:

- Code that links the QT application and SQL database together.
- The SQL queries that allows for the execution of certain information from the database
- How widgets are connected with one another (Signals and Slots).
- Much of the code that implements the specific functions needed to meet the requirements for all college students and administrators.
- Code that converts the variable types in a txt file to a QT variable type thus transfers the data to a SQL database.

Entry Criteria:

- Each developer must have their code written so that they can check for proper functionality. Each developer will test their code during the development process

Exit Criteria:

- When the program has no bugs present and much of the testing strategies have been completed with successful outcomes.
- The program accounts for invalid or undesirable inputs and prevents any harm to the code/executions.

Suspension Criteria: The case where bugs are present despite various testings are conducted.

Approval process: The case will be approved once the product manager agrees with the condition.

Schedule: A developer will utilize testing strategies, such as unit testing, when working on a story. This will allow the developer to debug code if necessary. Furthermore, every developer must utilize Black Box testing nearing the conclusion of the sprint. This will allow the developer to confirm that their code meets the necessary requirements of the story.

Necessary training needed for testing: Each developer has familiarity with using version control systems such as Github. Advanced knowledge of QT or C++ isn't required, but each developer must be proficient with both. Same ideas apply to SQL and using the SQL database.

Environment description:

Hardware: laptops/desktops that have good internet connections and works well.

Software: QT creator, SQLITE database browser such as DB Browser.

Configuration management (GITHUB):

Finished stories must be uploaded on the developers own branch. The Product owner will run the code with the developer and check for errors. If any errors are found, the developer must debug and reupload the debugged code onto their own branch, thus the process will resume again. Nearing the end of each sprint, the developer must have their branches equipped with the proper and completed code. Furthermore, Black Box testing will begin. Where the testing fails, the whole entire team must work together to

debug. Once Black Box testing is completed without any existing errors, the code will be merged into the master branch.

Documents that support the test plan: UML diagrams, QT reference pages, SQL reference pages