CMSC 345

Software Design and Development

Fall 2013

**Code to Joy**

Matthew Barteau

Benjamin Walsh

Sean Hutchinson

Amanda Hartman

Xuanzhu Zhu

John Pham

**Customer:**

Vishnugopal Rajamanickam

**Code Inspection Report**

Code to Joy – Map Maestro

Code Inspection Report

**Table of Contents**

Page

1. Introduction 3
   1. Purpose of This Document 3
   2. References 3
   3. Coding and Commenting Conventions 3
   4. Defect Checklist 3
2. Code Inspection Process 4
   1. Description 4
   2. Impressions of the Process 4
   3. Inspection Meetings 4
3. Modules Inspected 5
4. Defects

Appendix A – Team percent contributions, Team sign off, Customer acceptance

1. **Introduction**

1.1 Purpose of This Document

The purpose of this document is to provide a well-documented analysis of the code written for this product. This analysis will include an examination of the coding standards used, any defects found within the current build and the methods of reviewing our code employed for this document.

* 1. References
* System Requirements Specification
* Software Design Document
  1. Coding and Commenting Conventions

Code Conventions for the Java Programming Languages is the list of standard conventions supported by UMBC. The group consulted these standards when writing the Map Maestro. Some standards followed included in-line curly braces, camel case variable names, and standard method headers in Java. These standards were followed to make easy to read, organized code.

* 1. Defect Checklist

Currently, the project has some methods that are not functioning as intended. These program defects include:

Functional Defects:

* Adding locations as an admin does not add locations
* Removing locations as an admin does not remove locations
* Modify locations as an admin does not modify locations
* Currently, there is no email function set up to email the user their trip
* Jar file does not function like the program does in Eclipse
* Email system unable to send trip as attachment
* If text files removed by users, program has no file to draw from
* Program does not notify user that user could not be created because an existing user has that name
* Program does not notify user that user could not be created because passwords do not match
* Help documentation is non-existent

Security Oversights:

* Text files easily accessible by users to change status to admin
* No way within program to change user status to admin
* Nonexistent security of usernames and passwords

Other Defects:

* Logout button is too close to Next button on Locations select panel
* 2. **Code Inspection Process**
  1. Description

Code was inspected through the testing. After sections of code had been written, they would be tested to confirm functionality. After this incremental testing and unit testing was completed for each class, members met and merged their code. During this process, much testing was performed in order to ensure continued functionality after the edits. This incremental testing ended after the program functioned correctly in all areas tested.

After this building process, next came the cleaning up of the existing code. Comments were altered to more closely follow a uniform model, and more of the code was commented for clarity. After this, code was further edited, mainly the gui, in order to improve aesthetic appeal. This overhaul of the gui required further testing in order to confirm functionality a final time. This last confirmation of clean, successful code ended our review process.

2.2 Impressions of the Process

This was an effective way of evaluating the code written. Testing while inspecting allowed the group to confirm that the project has functionality and readability. Ideally, comments would be completed during the coding process, rather than after, but enough description was given to have a basic understanding of what each program did. The process could have been improved through the use of writing and inspecting together rather than coding separately and joining at the end.

Within the code, there are classes likely to contain few defects and classes likely to contain many defects. Based on the difficulty of implementation, LoginGUI class is likely to contain many defects, as it is the driver for both a GUI and the entire project.

* 1. Inspection Meetings

3. **Modules Inspected**

4. **Defects**

Appendix A – Team percent contributions, Team sign off, Customer acceptance

**Team Review Signoff**

I agree that I have read the above document and reviewed its content. If I do not agree with anything pertaining to the content or format of the document, I shall write it in the comments section below.

**Members of the Team: Signatures: Date:**

Amanda Hartman \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

Comments: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sean Hutchinson \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

Comments: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Matthew Barteau \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

Comments: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Xuanzhu Zhu \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

Comments: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

John Pham \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

Comments: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Benjamin Walsh \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

Comments: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_