Running Head: CSSIS Documentation

Computer Shop Sale and Inventory System Documentation

Code Catalysts

Nathan Miller

Brayden Qualman

Jake Robinson

Central Washington University

6/1/2023

Table of Contents

[Project Description 3](#_Toc136950479)

[User Guide 3](#_Toc136950480)

[Customer: 3](#_Toc136950481)

[Employee: 3](#_Toc136950482)

[Manager: 4](#_Toc136950483)

[UML Diagram 4](#_Toc136950484)

[Use Case Diagram 5](#_Toc136950485)

[Testing Overview 5](#_Toc136950486)

[Known Bugs 5](#_Toc136950487)

[Possible Feature Additions 6](#_Toc136950488)

Computer Shop Sale and Inventory System Documentation

# Project Description

The Computer Shop Sale and Inventory System (CSSIS) is a database-oriented application that supports inventory management and ecommerce system. It supports both customer and employee users. Customers and employees both share a sign in screen. Customers can sign-up for an account and login, which takes them to the customer dashboard. The customer view features include viewing and ordering individual computer component parts and prebuilt computers, creating repair order requests, and viewing their pending orders. Employees and managers (who must be hardcoded into database when hired) can log in and be taken to the employee dashboard. Employee view features include viewing inventory, adding new parts to the ordering catalog, requesting part orders (and accepting orders if a manager), creating prebuilt computers from parts currently available in inventory, and fulfilling customer part, prebuilt, and repair orders.

# User Guide

## Setup:

The program is launched as a single executable, which is compiled with a hard coded link to the database, which would be setup during installation. The program can be run by double clicking the executable jar file.

## Customer:

Upon starting the application, you either need to log into your account or register for a new one. Then you’ll have the ability to order computer parts or prebuilt computers from the system, create a request for a repair order, and can view your pending orders after making the purchase. Pending orders are viewable on the dashboard, and you can view order details by double clicking an order.

## Employee:

The employees' side is similar; first, you must connect to the database before anything can be done. Then, you’re able to log in as an employee, and from there you'll have the option to add new parts to the database so customers can order them. The employee can also create a new part order request or modify existing ones for when you need more parts due to demand. Employees can also fulfill repair requests from customers as well as fulfill their parts and prebuilt orders. The final feature of an employee is the ability to assemble prebuilts with available parts from the database.

## Manager:

Managers have the same abilities as employees but have the authority to approve part order requests from the employees.

# UML Diagram

A screenshot of a computer program

Description automatically generated with medium confidence

# Use Case Diagram



# Testing Overview

Due to the nature of our database-oriented project and limited time, we couldn’t implement robust unit testing. However, we did be sure to implement robust error detection and input cleansing to ensure all action listeners and functions are functioning as expected, as well as testing edge cases in domains for each user entered field. We’ve tested all features any they are working properly.

# Known Bugs

* Rounding issues on the order screen.

# Possible Feature Additions

There are a couple feature additions that could be implemented into this project. One feature we cut from our original design was the ability for the database and application to support multiple different stores, each with their own employees, manager, and inventory. We unfortunately had to cut this due to limited development time. Our system currently doesn’t check if parts are compatible with each other when creating prebuilt computers, so functionality could be added to ensure parts better reflect the real-life setting (like CPUs only fitting into certain socket types on motherboards for example). We could also add better a better GUI system instead of swing, and display pictures of items and prices. We could also implement maps to show the closest store to the user if we’re working with multiple locations. To make queries easier, we can also implement a factory design pattern instead of hard-coding everything. Finally, we could add full payment processing support if we ever decided to take this to a live setting.