**Document 06 – Sprint 2 Report**

CS 4321 – Fall 2021

This document is contained in your GitHub repository in a folder named *docs*.

| | Group | 1 | | --- | --- | | Group Member Names |  |
| --- | --- | --- | --- | --- |
|  | | 1. | Alex Acorn | | --- | --- | | 2. | Gregg Buehring | | | 3. | Sean Vickers | | --- | --- | | 4. | Nathan Bailey | | 5. | David Truong??? | |

1. **Video Demo**

| **Deliverable**  Create a video (up to 20 minutes) for your demo and post the link here. **Preferably, post your video(s) on Youtube**. The agenda for your demo:   1. (3-5 minutes) Discuss the design of your system using a class diagram.  * Explain at a high-level what each (or the most important) classes’ responsibilities are. Your goal is to give me a feel for your architecture and how the pieces fit together. * Don’t read off the list of methods! You can mention some key methods, or just describe what responsibilities each class has. If you need to go into more detail, it is fine, to show portions of the design and explain. * The class diagram should **not** show methods nor instance variables for any Gui classes (in StarUML, select the class, right-click, and choose: *Format, Suppress Operations* and *Suppress Attributes*. You can use the handles on the selected class to manually resize appropriately.) * If you need several versions of the class diagram, or need to break it into pieces that is fine. However, indicate how they join together.  1. (5-15 minutes) Demo as many user stories as you want, starting from the top priority (lowest number) and work your way down. Then, for each:  * Display a user story (text, in word or whatever) and read it, Expand on it if necessary. * Illustrate how the software fulfills it.   You can produce several videos if needed, *i.e.* part 1, part 2, *etc.* |
| --- |

| Video Link | Part 1(Class Diagram)-<https://youtu.be/oSEA8k62zqY>  Part 2(GUI)- <https://youtu.be/FvStLju4tP4> |
| --- | --- |

1. **Development Status**

| **Deliverable**  Provide a list of user stories, ordered by their priority that were worked on during Sprint 2. Include the user story number (“Num” in table below), status, title, and any comments if needed. For the status, use these symbols below. This is the *User Story Priority* table in Document 02, with Status and Comment columns added.   | **Symbol** | **Description** | | --- | --- | | ✔ | Complete (tested) | | ~ | In progress | | X | Not started | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |

| **Priority** | **Num** | **Status** | **Title** | **Comment** |
| --- | --- | --- | --- | --- |
| 1 | 1 | ~ | View Product Names and Prices in a VM | Can see name, but not price. |
| 2 | 2 | ✔ | Purchase Product with Cash | Works! |
| 6 | 3 | ✔ | Purchase Product with Card | Works! |
|  | 4 | ✔ | Cancel Pending Purchase | Works! |
| 3 | 5 | ✔ | Restock VM with Products | Works! |
|  | 6 | ~ | Remove Expired Products | Should work on the backend, didn’t think it was implemented frontend. |
| 4 | 7 | ✔ | Remove Cash from VM | Works! |
| 5 | 8 | ✔ | Restock Cash in VM | Works! |
|  | 9 | ✔ | Add a Manufacturer | Works! |
|  | 10 | ✔ | Add a Non-Refrigerated Product | Works! |
|  | 11 | ✔ | Add a Refrigerated Product | Works! |
|  | 12 | ✔ | Add a Location | Works! |
|  | 13 | ✔ | Add a Non-Refrigerated VM | Works! |
|  | 14 | ✔ | Add a Refrigerated VM | Works! |
|  | 15 | ~ | Edit a Manufacturer |  |
|  | 16 | ~ | Edit a Non-Refrigerated Product |  |
|  | 17 | ~ | Edit a Refrigerated Product |  |
|  | 18 | ✔ | Edit a Location | Works! |
|  | 19 | ✔ | Edit a Non-Refrigerated VM | Works! |
|  | 20 | ✔ | Edit a Refrigerated VM | Works! |
|  | 21 | ✔ | Remove a Manufacturer | Works! |
|  | 22 | ✔ | Remove a Non-Refrigerated Product | Works! |
|  | 23 | ✔ | Remove a Refrigerated Product | Works! |
|  | 24 | ✔ | Remove a Location | Works! |
|  | 25 | ✔ | Remove a Non-Refrigerated VM | Works! |
|  | 26 | ✔ | Remove a Refrigerated VM | Works! |
|  | 27 | ✔ | Generate Machine Report | Works! |
|  | 28 | ✔ | Generate Product Report | Works! |
|  | 29 | ✔ | Generate Detailed Machine Report | Works! |
|  | 30 | ✔ | Generate Location Report | Works! |
|  | 31 | ✔ | Generate Company Report | Works! |
|  | 32 | ✔ | Generate Dry Goods Report | Works! |
|  | 33 | ✔ | Generate Refrigerated Report | Works! |
|  | 34 | ✔ | Generate Products Report | Works! |
|  | 35 | ✔ | Generate Manufacturer Report | Works! |
|  | 36 | ✔ | Generate Expired Inventory Report | Works! |

1. **Class Diagram**

| **Deliverable**  Use StarUML to develop at least two neat, legible, UML class diagrams showing your current design that exactly reflects your code. The first class diagram should show only the classes and associations/relationships (not methods nor attributes). The second should show only public members of the classes and associations/relationships. **These should be saved in your GitHub repository in a folder named docs** |
| --- |

1. **System Tests**

| **Deliverable**  Provide a number list of system tests that are performed manually. Can have multiple tests for a user story. Use the template below. Remove the prompts that are surrounded by brackets. An example follows. |
| --- |

| Test Num | 1 |
| --- | --- |
| US Num-Title | 2 - Purchase Product with Cash |
| Description | Purchase Item With Just Enough Cash |
| Flow of events | 1. Create Vending Machine 2. Create Item Costing $1 3. Create Customer with exactly $1 4. Purchase Item With Cash |
| Expected output/result | Expected: Purchase Successful with $0 change  Result: Purchase Successful with $0 change |
| Comments |  |

| Test Num | 2 |
| --- | --- |
| US Num-Title | 2 - Purchase Product with Cash |
| Description | Purchase Item With Just More Cash |
| Flow of events | 1. Create Vending Machine 2. Create Item Costing $1 3. Fill Vending Machine with 5 nickels, dimes, quarters, and dollars 4. Create Customer with exactly $2.50 5. Purchase Item With Cash |
| Expected output/result | Expected: Purchase Successful with $1.50 change  Result: Purchase Successful with $1.50 change |
| Comments |  |

| Test Num | 3 |
| --- | --- |
| US Num-Title | 2 - Purchase Product with Cash |
| Description | Purchase Item Failed With Not Enough Cash |
| Flow of events | 1. Create Vending Machine 2. Create Item Costing $1 3. Create Customer with exactly $.50 4. Purchase Item With Cash |
| Expected output/result | Expected: Purchase Failed  Result: Purchase Failed |
| Comments |  |

| Test Num | 4 |
| --- | --- |
| US Num-Title | 2 - Purchase Product with Cash |
| Description | Purchase Item With Just More Cash |
| Flow of events | 1. Create Vending Machine 2. Create Item Costing $1 3. Fill Vending Machine with 1 nickel, dime, quarter, and dollar 4. Create Customer with exactly $2.50 5. Purchase Item With Cash |
| Expected output/result | Expected: Purchase Failed  Result: Purchase Failed |
| Comments |  |

| Test Num | 5 |
| --- | --- |
| US Num-Title | 3 - Purchase Product with Card |
| Description | Purchase Item With Card |
| Flow of events | 1. Create Vending Machine 2. Create Item Costing $1 3. Purchase Item With Card |
| Expected output/result | Expected: Purchase Successful  Result: Purchase Successful |
| Comments |  |

| Test Num | 6 |
| --- | --- |
| US Num-Title | 6 - Restock VM with Products |
| Description | Restock from the starting quantity 0 to quantity 5 |
| Flow of events | 1. Create Vending Machine with quantity 0 with capacity 5 2. Create Item 3. Have technician restock 5 of the items |
| Expected output/result | Expected: Restocked 5 items  Result: Successfully restocked 5 items |
| Comments |  |

| Test Num | 7 |
| --- | --- |
| US Num-Title | 7- Remove Cash from VM |
| Description | Remove $1 |
| Flow of events | 1. Have a customer buy a $1 item 2. Have technician remove $1 from vending machine |
| Expected output/result | Expected: Vending Machine Balance: $0  Result: Vending Machine Balance: $0 |
| Comments |  |

| Test Num | 8 |
| --- | --- |
| US Num-Title | 8- Restock Cash In Vending Machine |
| Description | Add $1 |
| Flow of events | 1. Make a Vending Machine 2. Have Technician add $1 |
| Expected output/result | Expected: Vending Machine Balance: $1  Result: Vending Machine Balance: $1 |
| Comments |  |

| Test Num | 9 |
| --- | --- |
| US Num-Title | 9 - Add a Manufacturer |
| Description | Add Manufacturer |
| Flow of events | 1. Make a Vending Machine with Manager 2. Make Item with name “Pepsi” that has a manufacturer with name “PepsiCo” with Manager. |
| Expected output/result | Expected: Pepsi Manufacturer Name: PepsiCo  Resultl: Pepsi Manufacturer Name: PepsiCo |
| Comments |  |

| Test Num | 10 |
| --- | --- |
| US Num-Title | 10 - Add a Non-Refrigerated Product |
| Description | Adding A non-refrigerated product |
| Flow of events | 1. Make a vending Machine without a temp 2. Add an item without a temp |
| Expected output/result | Expected Output: 1 Non refrigerated item  Result: 1 Non refrigerated item |
| Comments |  |

| Test Num | 11 |
| --- | --- |
| US Num-Title | 11 - Add a Refrigerated Product |
| Description | Adding A refrigerated product |
| Flow of events | 1. Make a vending Machine with a temp 2. Add an item with a temp |
| Expected output/result | Expected Output: 1 refrigerated item  Result: 1 refrigerated item |
| Comments |  |

| Test Num | 12 |
| --- | --- |
| US Num-Title | 12 - Add a Location |
| Description | Add a Location |
| Flow of events | 1. Make a Vending Machine with Location “123 Electric Avenue” |
| Expected output/result | Expected: VM address: “123 Electric Avenue”  Result: VM address: “123 Electric Avenue” |
| Comments |  |

| Test Num | 13 |
| --- | --- |
| US Num-Title | 13- Add a Non-Refrigerated VM |
| Description | Add a Non-Refrigerated VM |
| Flow of events | Make a Vending Machine without a temp |
| Expected output/result | Expected: 1 non-refrigerated vm  Result: 1 non-refrigerated vm |
| Comments |  |

| Test Num | 14 |
| --- | --- |
| US Num-Title | 14- Add a Refrigerated VM |
| Description | Add a Refrigerated VM |
| Flow of events | Make a Vending Machine with a temp |
| Expected output/result | Expected: 1 refrigerated vm  Result: 1 refrigerated vm |
| Comments |  |

| Test Num | 15 |
| --- | --- |
| US Num-Title | 18- Edit a Location |
| Description | Change a vending machine’s location |
| Flow of events | 1. Make a vending Machine 2. Use Manager to change vending machine’s location |
| Expected output/result | Result: Changed Location |
| Comments |  |

| Test Num | 16 |
| --- | --- |
| US Num-Title | 19 - Edit a Non-Refrigerated VM |
| Description | Edit a Non-Refrigerated VM |
| Flow of events | 1. Make a Vending Machine with building “Student Union” 2. Use Manager to edit the Vending Machine’s building to “Nevin’s Hall” |
| Expected output/result | Expected: VM building: Nevin’s Hall  Result: VM building: Nevin’s Hall |
| Comments |  |

| Test Num | 17 |
| --- | --- |
| US Num-Title | 20 - Edit a Refrigerated VM |
| Description | Edit a Refrigerated VM |
| Flow of events | 1. Make a Refrigerated Vending Machine with building “Student Union” 2. Use Manager to edit the Refrigerated Vending Machine’s building to “Nevin’s Hall” |
| Expected output/result | Expected: VM building: Nevin’s Hall  Result: VM building: Nevin’s Hall |
| Comments |  |

| Test Num | 18 |
| --- | --- |
| US Num-Title | 24- Remove a Location |
| Description | Removing a location |
| Flow of events | 1. Make a vending machine with a location 2. delete the vending machine |
| Expected output/result | Results: 0 vending machines |
| Comments |  |

| Test Num | 19 |
| --- | --- |
| US Num-Title | 25- Remove a Non-Refrigerated VM |
| Description | Removing a non - refrigerated vm |
| Flow of events | 1. Make a vending machine without a temp 2. delete the vending machine |
| Expected output/result | Results: 0 vending machines |
| Comments |  |

| Test Num | 20 |
| --- | --- |
| US Num-Title | 26- Remove a Refrigerated VM |
| Description | Removing a refrigerated vm |
| Flow of events | 1. Make a vending machine without a temp 2. delete the vending machine |
| Expected output/result | Results: 0 vending machines |
| Comments |  |

| Test Num | 21 |
| --- | --- |
| US Num-Title | 27- Generate Machine Report |
| Description | Generating a Machine Report |
| Flow of events | 1. Make a vending machine 2. add a bunch of items 3. buy a bunch of items 4. generate machine report |
| Expected output/result | Result: Successful Machine Report |
| Comments |  |

| Test Num | 22 |
| --- | --- |
| US Num-Title | 28- Generate Product Report |
| Description | Generate Product Report |
| Flow of events | 1. Make a vending machine 2. add a bunch of items 3. buy a bunch of items 4. generate product report |
| Expected output/result | Result: Successful Product Report |
| Comments |  |

| Test Num | 23 |
| --- | --- |
| US Num-Title | 29- Generate Detailed Machine Report |
| Description | Generate Detailed Machine Report |
| Flow of events | 1. Make a vending machine 2. add a bunch of items 3. buy a bunch of items 4. generate Detailed Machine report |
| Expected output/result | Result: Successful Detailed Machine Report |
| Comments |  |

| Test Num | 24 |
| --- | --- |
| US Num-Title | 30- Generate Location Report |
| Description | Generate Location Report |
| Flow of events | 1. Make a vending machine 2. add a bunch of items 3. buy a bunch of items 4. generate Location report |
| Expected output/result | Result: Successful Location Report |
| Comments |  |

| Test Num | 25 |
| --- | --- |
| US Num-Title | 31- Generate Company Report |
| Description | Generate Company Report |
| Flow of events | 1. Make a vending machine 2. add a bunch of items 3. buy a bunch of items 4. generate Company report |
| Expected output/result | Result: Successful Company Report |
| Comments |  |

| Test Num | 26 |
| --- | --- |
| US Num-Title | 32- Generate Dry Goods Report |
| Description | Generate Dry Goods Report |
| Flow of events | 1. Make a vending machine 2. add a bunch of items 3. buy a bunch of items 4. generate Dry Goods report |
| Expected output/result | Result: Successful Dry Goods Report |
| Comments |  |

| Test Num | 27 |
| --- | --- |
| US Num-Title | 33- Generate Refrigerated Goods Report |
| Description | Generate Refrigerated Goods Report |
| Flow of events | 1. Make a vending machine 2. add a bunch of items 3. buy a bunch of items 4. generate Refrigerated Goods report |
| Expected output/result | Result: Successful Refrigerated Goods Report |
| Comments |  |

| Test Num | 28 |
| --- | --- |
| US Num-Title | 34- Generate Products Report |
| Description | Generate Products Report |
| Flow of events | 1. Make a vending machine 2. add a bunch of items 3. buy a bunch of items 4. generate Products report |
| Expected output/result | Result: Successful Products Report |
| Comments |  |

| Test Num | 29 |
| --- | --- |
| US Num-Title | 35- Generate Manufacturer Report |
| Description | Generate Manufacturer Report |
| Flow of events | 1. Make a vending machine 2. add a bunch of items 3. buy a bunch of items 4. generate Manufacturer report |
| Expected output/result | Result: Successful Manufacturer Report |
| Comments |  |

| Test Num | 30 |
| --- | --- |
| US Num-Title | 36- Generate Expired Inventory Report |
| Description | Generate Expired Inventory Report |
| Flow of events | 1. Make a vending machine 2. add a bunch of items 3. buy a bunch of items 4. generate Expired Inventory report |
| Expected output/result | Result: Successful Expired Inventory Report |
| Comments |  |

1. **Retrospective**
2. Read this short page about what a software retrospective is and why it is important:

<https://searchsoftwarequality.techtarget.com/definition/Agile-retrospective>

1. Meet as a group and discuss the following questions and provide a group written response below:
2. What worked well for us?

| **Answer**  Team work makes the dream work. Communication was king for this project; Especially when it came to fixing bugs or errors in our code. We were able to communicate thoroughly amongst one another. |
| --- |

1. What did not work well for us?

| **Answer**  Scheduling and lack of communication would be our downfall in retrospect. Procrastinating was another “elephant in the room.” Our schedules were somewhat hectic, so it was really difficult to hone in our concentration, at first. |
| --- |

1. What actions can we take to improve our process going forward?

| **Answer**  This is the end of the class, but one action that we could do to improve our process if we were to continue working on the project would be consistent self-given deadlines for various aspects of our project. Also, establishing clear cut goals and following up on those goals would improve the process tremendously. |
| --- |