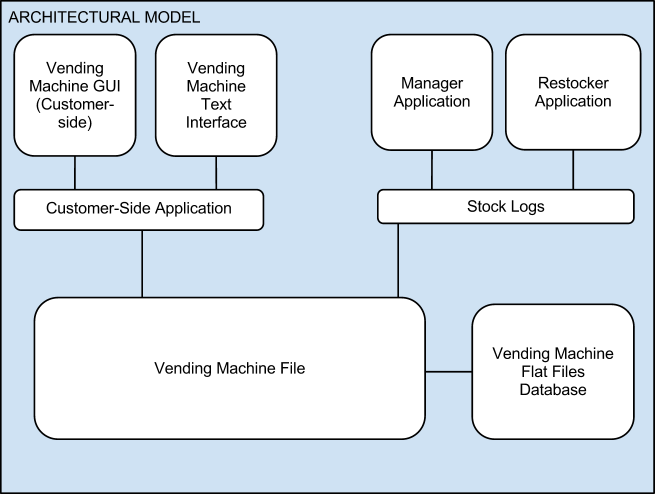
**Product Design**

|  |  |
| --- | --- |
| **Team** | Group S361-02A **Chalupas** |

# Architectural Model

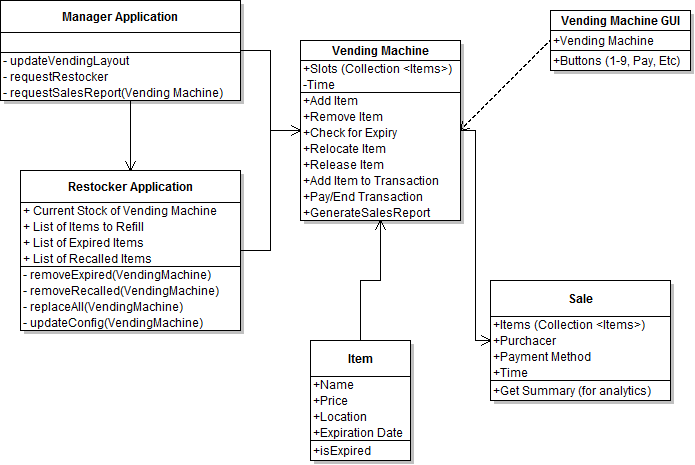
This diagram represents the major subsystems of the product. Initially we want to focus on the domain layer and its components before decomposing the user interface component. Note that a common interface allows both the GUI and a Command Line Interface to access the domain model in the same manner without regard to the type of presentation technique.

**Components and Functions**

*Identify the components (logical groupings of software modules that provide a related set of services). For each component, specify the information it maintains and the functionality it provides. Provide sufficient detail so that the purpose of each component in the design is clear, and so your instructor can assess each component’s appropriateness for the problem at hand.*

|  |  |
| --- | --- |
| Vending Machine Application | Component State   * Maintains a collection of items in the vending machine. * E.g., a printing subsystem might hold the current status of all the printers it controls as well as the queue of print jobs waiting to be printed.   Component behavior   * Execute sales -> update its collection of items * Simulates time * E.g., the printing subsystem might support the queuing up of new jobs, estimating the time until a given job completes, or emailing status information at the end of a job. |
| Restocker Application | Component State   * Current Stock of Current Vending Machine * List of Expired Items in Current Vendingg Machine * List of Recalled Items in Current Vending Machine * List of Purchased Items in Current Vending Machine   Component behavior   * Removes Recalled or Expired items in the current vending machine. * Rearranges the Vending Machine, if demanded by Manager * Refills Current Vending Machine with fresh items. |
| Manager Application | Component state   * List of all vending machines currently in operation   Component behavior   * Reads Analytical data from Vending Machine * Rearranges item placement based on sales * Sends updated data back out to Restocker application. * Can request a Restocker to a location, if stocks are too low. * Can add and remove vending machines from the network. |

# Class Diagram(s)



MANAGER UPDATE: Make sure to include the list of vending machines as an ATTRIBUTE of the Manager application. Manager needs to be able to add/remove vending machines, specify rows, columns, and items per slot if adding.

ADD MULTIPLICITY

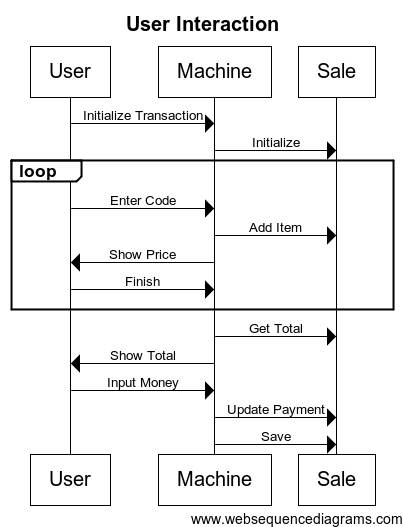
# Sequence Diagram(s)

UML sequence diagram for the agreed significant use case.

*While you may reference an external file here, most instructors* ***strongly*** *prefer embedded images, or, failing that, external files in generic formats such as JPEG, PNG or PDF.*

CREATE A SEQUENCE DIAGRAM BASED ON CUSTOMER INTERACTION.

We think that the customer interaction is the most significant use case, and it will be used much more often than the manager or restocking application.



CODE FOR EDITING AT <http://www.websequencediagrams.com/#> REMOVE BEFORE SUBMISSION

# Design Rationale

*Currently chosen concept is in bold.*

* Data management
  + Concept A: Store all data for every vending machine in a single file
    - Pros: Easy to manage a single file
    - Cons: If any of the applications needs to make a change to any vending machine they have to lock the entire. Corruption affects every vending machine’s data, and only one machine can alter it at once.
  + **Concept B: Store data for each vending machine in a separate file. Store data for each vending machine in a separate file. Store vending machine metadata for all vending machines in a single file.**
    - Pros: Modifying one vending machine’s information does not require locking every vending machine’s data. Corruption of one vending machine’s data does not affect any other data. Locking can be managed in a separate layer (metadata) than the actual data. Can run multiple vending machines at once, as you would in the real world.
    - Cons: Lots of files to keep track of. More complicated coordination.