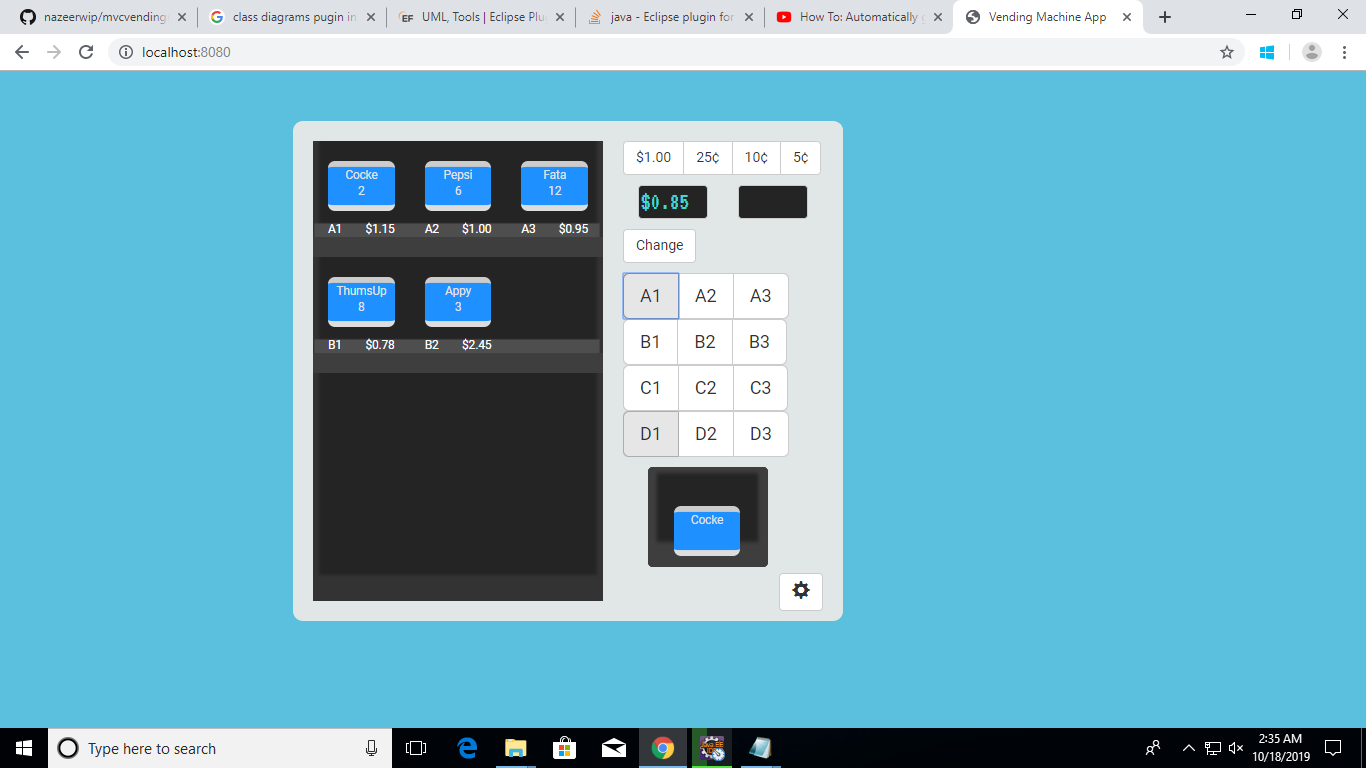
**Vending Machine Application**

**By Shaik Nazeer**

We have implemented basic Angular with JSP using spring boot technology. When hit the url <http://localhost:8080/> getting the below screen



1. **Design Considerations:**

Scalability & Flexibility: You can change items, price and quantity being dispensed, promotion without any code changes.

**Technical Considerations:**

1. Database can change from one to other by making changes in configuration file.
2. External webserver’s not required to make it run, we have embedded servers with spring boot technology.
3. Abstraction of implementation layer (interfaces) to support changes in vending machines.
4. **Technical Architecture Design:**

There are mainly into four main categories: **presentation, application, domain, and infrastructure**. Each of the layers contains objects related to the particular concern it represents.

* **The presentation layer** contains all of the classes responsible for presenting the UI to the end-user or sending the response back to the client (in case we’re operating deep in the back-end).
* **The application layer** contains all the logic that is required by the application to meet its functional requirements and, at the same time, is not a part of the domain rules. In most systems that I've worked with, the application layer consisted of services orchestrating the domain objects to fulfill a use case scenario.
* **The domain layer**represents the underlying domain, mostly consisting of domain entities and, in some cases, services. Business rules, like invariants and algorithms, should all stay in this layer.
* **The infrastructure layer (also known as the persistence layer)**contains all the classes responsible for doing the technical stuff, like persisting the data in the database, like DAOs, repositories, or whatever else you’re using.

There are two important rules for a classical Layered Architecture to be correctly implemented:

1. All the dependencies go in one direction, from presentation to infrastructure. (Well, handling persistence and domain are a bit tricky because the infrastructure layer often saves domain objects directly, so it actually knows about the classes in the domain)
2. No logic related to one layer’s concern should be placed in another layer. For instance, no domain logic or database queries should be done in the UI.
3. **How to use the Vending Machine System:**

We are displaying list of items in vending machine from backend services and displays on UI with Quantity, Cost and Item name.

**Operating steps:**

1. Enter the amount using amount on top
2. When deposited amount if greater than price of the items (A1, A2 etc.. ) those options can be selected. Items with price more than deposited amount will remain disable.
3. Select item’s with price within amount deposited. Item will be delivered and change will be dispersed.
4. **Class & Sequence Diagram:**

To view below diagram change zoom to 200%

