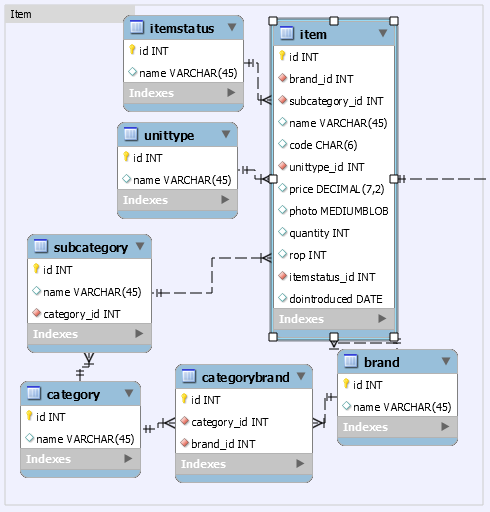
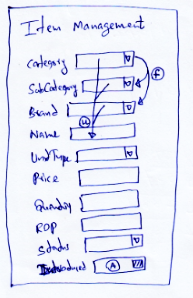
|  |  |
| --- | --- |
| **Branch** | **Colombo** |
| **Name** | **Dimuthu Suranga** |
| **Project** | **Purchase, Sales and Inventory Management System for Harvest-Supermarket** |
| **Sprint** | **3(a) – Item Management** |
| **Duration** | **16.07.2018 – 24.07.2018** |

**Data Design** (Template : Employee) (Imagine the Form if it is difficult to design data)

**Form-Interactivity other than Regex-Based and Default-Binding** (Imagine the Form using Master-Table)

Category 🡪 When Category Selected, Relevant Brands and Subcategories must be selected automatically. (Not Binding)

Category/Brand/Subcategory 🡪 Name will generate based on these values

**Sample Data of Supportive (FK-Tables)**

Brand( Nipuna, Maliban, Anchor, Manchee ) , Unit Type ( Unit, Retail-1kg, Retail-100g)

Item Status ( Available, Discontinued, Pending)

Category( Busicuit, Dairy, Rice ) Subcategory ( Rice-> Samba, Kakulu, Nadu, Dairy->Milk-Powder,Yoghurt,Chees)

**Master**

|  |  |  |  |
| --- | --- | --- | --- |
| Property | Regex/Validation | Sample-1 | Sample-2 |
|  |  |  |  |
| Brand | NN | Nipuna | Anchor |
| Category | NN | Rice | Dairy |
| Subcategory | NN | Samba | Milk-Powder |
| Code | ^[\d]{6}$ | 010001 | 020001 |
| Name | ^[a-zA-Z]{3} | Nipuna-Keerisamba-5kg | Anchor-Milk-Powder-400g |
| Unittype | NN | Unit | Unit |
| Price | ^([\d]+)(([.][\d]{2})?)$  Max-9999.99 | 850.00 | 325.00 |
| Photo | Optional | Image result for keeri samba 5kg packet | Image result for anchor 400g |
| Quantity | ^[0-9]?$ | 15 |  |
| Dostarted | Yyyy-mm-dd | 2015-02-12 | 2015-06-23 |
|  |  |  |  |

**Table Design**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Photo | Category | Brand | Price | ROP | Quantity |

**Search Criteria:**  Name, Category, Brand

**Review of the Current Sprint** (Modifications/Corrections-Ongoing Changes and End-Review)

|  |
| --- |
| ItemStatus, Item.description are needed (ER-Drawing) (Initially Empty) |

Phase-1 (Sprint Approval)

1. Update the Product Backlog
2. Update the Sprint Sequence
3. Create a Folder for the last sprint as “Sprint-N”
4. Copy the content of the “Sprint-Current” into the newly created Folder
5. Save a Database Backup relevant to the “Sprint-N”
6. Rename the ER relevant to the new Sprint as “eucbitprojectSn”
7. Open the ER and add the relevant Tables of the new Sprint
8. Change the Order of the Attributes to be used in the Form
9. Export the ER as a “.png” file
10. Create the Sprint-Plan-N
11. Get the Supervisor Approval for the new Sprint-Plan
12. Update Product-Backlog/Overall-ER If needed, Revise Phase-1 if needed.

Phase-2 (DB-Update, View Data, Define Table and Fill-Data)

1. Synchronize the ER with the Database
2. Insert Data into Supportive Tables. Make sure to click “Apply” function in MySQL-Workbench.
3. Insert Data into Master Table(s).
4. **Test-1** : Observe Data in the tables using MySQL-Workbench-Refresh Function
5. Generate Entity Classes using NetBeans’s Auto-Entity-Generation Wizard
6. **Test-2** : Run the Project to verify that there is no Errors exists in the last Sprint (Smoke Test)
7. Copy the New Entities into the “entity” folder in the “Sprint-Current” of the Project
8. Update Existing Entities that has modified Attributes and Relationships
9. **Test-3** : Run the Project to verify that there is no Errors after adding new Entities and Changing Existing (Smoke Test)
10. Add needed Annotation (JPA/Jackson) to the Entity-Classes.

(Depended Objects must be loaded using “FetchType.EAGER” and JSON-loops must be stopped using “@JsonIgnore”)

1. Declare X-Controller and X-Dao to implement “findAll” Service
2. **Test-4** : Request URL for “findAll” Service as “http://localhost:8080/employees?page=0&size=1”
3. Create the UI needed for the Module
4. Update “UiController” to deliver the UI
5. **Test-5** : Request URL for the UI
6. Define the Table with Pagination
7. Load and Fill data into the Table
8. **Test-6** : Request URL for the UI with the table that would filled with data

Phase-3 (Load-Data, Define Form, Validation with Regex and Add)

(A)

1. Add the Regexes to the Entity Classes using “@Pattern” Annotation
2. Update the “RegexController” to provide needed Regex Service related to the Module
3. **Test-1** : Request URL “http://localhost:8080/regexes/user”

(B)

1. Define Dao methods (list) with Constructors in the Entity Classes for Supportive Tables
2. Update the “DatalistController” to provide needed data. Auto wire needed Daos.
3. **Test-2** : Request URL “http://localhost:8080/datalists/roles” (userstatuses, employees)

(C)

1. Create the Form
2. Update initialize function by registering Event-Handlers with operational buttons
3. Get the Regexes from the Server and Apply them with the relevant Input Fields
4. Load the Form with Supportive Data and fill them with
5. **Test-3** : Combo Must be loaded with Supportive Data
6. **Test-4** : Validation Color Changes (Custom “Validation and Data-Binding” may be need to defined and register in “initialize()”)