## Report Three

|  |
| --- |
| Inventory Management System |

## FHSU Software Engineering Fall 2018

## Group ABLM

Members:

###### Ross Monroe

###### Joshua Larson

###### Venance Agnero

###### Sheun Badmus

###### 

URL: <https://github.com/SoftwareClassProject/SoftwareProject2.git>

###### November 30th 2018

Individual Contributions Responsibilities

###### Ross Monroe:

* + Worked on correcting/improving functionality for the project
  + Created Gantt Chart showing the tasks schedule
  + Created the Resources section
  + History of Work, Current Status, Future Work

###### Joshua Larson:

* Updated Glossary of Terms
* Wrote Reflective Essay
* Worked on Design Patterns
* Working on User Interface Updates
* Worked on OCL Contracts

###### Venance Agnero:

* System Requirement
* Functionality Requirement
* Interaction Diagram
* Domain Analysis

Table of Contents

1. Customer Statement of Requirements (CSR) ……..3
   1. Problem Statement ……..3
2. Glossary of Terms ……..5
3. System Requirements ……..7
   1. Enumerated Functional Requirements ……..7
   2. Enumerated Nonfunctional Requirements ……..8
   3. On-Screen Appearance Requirements ……..9
4. Functionality Requirements Specification ……10
   1. Stakeholder ……10
   2. Actors and Goal ……10
   3. Use Cases...............................................................................................................10
   4. System Sequence Diagrams…………………………………………….………..20
5. Effort Estimation using Use Case Points ……21
6. Domain Analysis ……22
   1. Domain Model ……22
   2. System Operation Contracts..................................................................................25
7. Interaction Diagrams ……26
   1. Sequence Diagrams ……26
   2. UML Interaction Diagrams ……30
   3. Design Patterns ……30
8. Class Diagram and Interface Specification………………………………………………31
   1. Class Diagram ……………………………………………………………………31
   2. Data types and signatures………………………………………………………...31
   3. Traceability Matrix………………………………………………………………33
   4. Design Patterns…………………………………………………………………..34
   5. OCL Contracts…………………………………………………………………...34
9. System Architecture and System Design ………………………………………………..35
   1. Architectural Styles………………………………………………………………35
   2. Identifying Subsystems ………………………………………………………….35
   3. Mapping Subsystems to Hardware………………………………………………35
   4. Persistent Data Storage…………………………………………………………..35
   5. Network Protocol……………………………………………………………..….36
   6. Global Control Flow……………………………………………………………..36
   7. Hardware Requirements………………………………………………………….36
10. Algorithm and Data Structures (Not Applicable)
11. User Interface Design and Implementation……………………………………………...37
12. Design of Tests…………………………………………………………………………..47
13. History of Work, Current Status, and Future Work.……………………………………..49
14. References. …....50
15. Customer Statement of Requirements (CSR)
    1. Problem Statement

Here at Widgets and Gizmo’s we have been making due with the tried and true methodology of the primitive pen and paper for our store’s inventory management. This has been the philosophy since our inception and we continued following the principles of, “if it ain’t broke, why fix it?” However, as the number of items has been increasing, and the vendors that we receive our products from going up in numbers as well, we are continually finding ourselves in difficult predicaments with keeping our records straight. Our current inventory management is fraught with errors, inefficient, excruciatingly time-consuming, and just frankly old-fashioned.

Taking inventory by hand is always causing errors. We never seem to be able to get an accurate count of all the items that are in the store at any given time. Even with multiple attempts to coordinate counting every item, and then tallying them accordingly, there are always discrepancies. Trying to keep all the inventory cards that we fill out in order and keeping the item numbers/names/etc. consistent, is a nightmare as well. Having a system that would allow us to keep track of everything electronically and with unique numbers/names would be the ideal to help alleviate the issues that we have been having with our outdated style. It would also save ample time since there would be no need for manually counting and checking multiple times to try and achieve a certain level of accuracy.

Multiple items recorded multiple times or accidently omitted. This has been an issue from the beginning. We never have a precise checker, immune to human, error that we can count on to see if an item has been added to our registry more than once. In the same regard, sometimes we accidently remove an item that is still available at our store. Ideally, we would design a software system that would be able to verify with us whether we are trying to add a duplicate item to the roster, or whether we are trying to sell something that is not appearing on our current roster. This would really save us time and some redundancies when it comes to keeping our inventory in a near perfect state that we can easily search through and pull data from.

Horribly inefficient style for adding new items or removing item from stock. Our business is constantly changing our product line to try and align with our customers need. Too often this requires creating new inventory ledgers, and hoping our human error does not leave out, or duplicate items into our inventory again. At times when we get the items numbered correctly per our ledger, they are not categorized in any decipherable manner, it would be nice to be able to have a software system that could automatically keep track/update our item numberings and be able to add some numbering scheme that would help us identify what type of items fall within certain ranges. This would help us be able to track what type of items are moving off the shelves, or which ones are not.

No reminder or notification when items are running low, or out of stock. We have missed out on many valuable sale opportunities when our poor management style left us out of stock when our customers came looking for a product that we normally have on hand. It would be nice to have a software system implemented where this would be a thing of the past for our business and allow us to keep track of the quantity of each item on our shelves. Another feature that would be handy for us, would be to add the ability to send us notifications if an item in stock is running low. This would give us the ability to reflect on how many of them we sold, in what time frame, and therefore allow us to order additional accordingly before we run out and lose these potential sales. This system would also help ordering inefficiencies due to wrong items, non-optimal quantities being ordered. Our current strategy does not allow us to track what would be best for certain areas, or times of the year. The ability to review our entire sale history, in a comprehensive manner, would really add a major marketing benefit to our business model.

Inefficient means of setting and updating prices on items throughout the store. Whenever we have sales in our store, or are required do any form of re-pricing, it makes for a nightmare on our books. Having a software system that would be able to update features like this on our products instantly in an electronic dynamic ledger would be immensely helpful. Temporary sales on items requires a plethora of extra work that must be redone far too often to make it worth our time with our current inventory management. This is also the case when it comes time to discount our out-of-season, non-prime products that we are reducing for a quicker sale.

Non-optimal means of keeping track of daily sales/activity. Prone to human error with current set-up, time-consuming. Every time we have had the need for any form of audit, tax preparation time, or any form of sales/profits reports for any certain period, it has been met with difficulties. Being able to create a software system that would allow us to instantly pull sales activity, or any other measurable activity out of our database for review or documentation would save a great deal of time and hassle for us compared to what we are currently dealing with. Not only would it save us a large amount of time, it would make all our accounting practices far more streamlined and accurate. In addition to, it would give us the far better grasp as to what our marketing team should be focusing on, in terms of what items are selling, what time of year they are selling, and in what area these particular products are selling.

* 1. Glossary of Terms
* Customer Statement of Requirements – an informal description of what the customers thing they need from a software system to do for them.
* Priority Weight – a level of measurement pertaining to the importance of each requirement. May be assigned by the customer or derived from the urgency-to-deliver the requested capabilities to the customer.
* Stakeholder – is an individual, team, or organization with interests in, or concerns related to, the system-to-be.
* Actors – human or other external system that interacts with the system-to-be.
* Use cases – a list of actions or event steps typically defining the interactions between a role and a system to achieve a goal.
* System sequence diagrams – a sequence diagram that shows, for a particular scenario of a use case, the events that external actors generate, their order, and possible inter-system events.
* User effort – the required amount of work to use the interface designed for the system to be described in mouse clicks, keystrokes, etc.
* User interface – the means by which the user and a computer system interact, in particular the use of input devices and software.
* Design Patterns – general reusable solution to a commonly occurring problem in software design. This is not a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations.
* Domain Model – a conceptual model of the domain that incorporates both behavior and data.
* System Operation Contracts – These describe detailed system behavior in terms of state changes to objects in the Domain Model, after a system operation has executed
* Architectural Styles – refers to the high level structures of a software system and the discipline of creating such structures and systems
* Hardware Requirements - The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware
* Interaction Diagrams – Models that describe how a group of objects collaborate in some behavior – typically a single use-case

2. System requirements

1. Enumerated Functional Requirement

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Priority Weight** | **Requirement** |
| REQ - 0 | 3 | A new user shall be registered in the system, and must create a login password and username before being able to login |
| REQ - 1 | 2 | Every user shall login with him/her credential, before enter data or retrieve information from the system |
| REQ - 2 | 1 | Every item registered shall have its SKU number, including all its possible descriptions |
| REQ - 3 | 1 | All item shall be registered with its price and quantity |
| REQ - 4 | 4 | The system shall ask to create an account for new user. |
| REQ - 5 | 5 | The system shall ask for name and Id Number for registration |
| REQ - 6 | 2 | The system shall ask for credential before login |
| REQ - 7 | 2 | The system shall ask to choose an option of inputs |
| REQ - 8 | 2 | The system shall ask to input name |
| REQ - 9 | 2 | The system shall ask to input item quantity |
| REQ - 10 | 3 | The system shall ask to input item price |
| REQ - 11 | 2 | The system shall ask to choose an option of activities |
| REQ -12 | 2 | The system shall be able to display the status of the inventory |
| REQ - 13 | 3 | The application shall be compatible to any operating system |
| REQ – 14 | 1 | The application shall be installed on any workstation |

1. Enumerated Non-functional Requirement

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Weight Priority** | **Requirement** |
| REQ - 0 | 3 | Any modification of information in the system shall be authorized by the supervisor |
| REQ - 1 | 2 | Any modification of information in the system must be approved by the manager. |
| REQ - 2 | 2 | Every new price of item shall be authorize by the manager in the system |
| REQ - 3 | 1 | Every new price of item shall be added in the system by the manager |
| REQ - 4 | 4 | Every new price of item shall be verified and approved by the company in the system |
| REQ - 5 | 2 | Every item newly entering the inventory list shall be verified by the manager |
| REQ - 7 | 3 | Every inventory shall be approved by the company |
| REQ - 8 | 1 | User shall login to the system at company site only |
| REQ – 9 | 1 | Owner shall be allowed to access the system remotely |
| REQ – 10 | 3 | Data shall be backed up every night after hours |
| REQ – 11 | 1 | User account shall be automatically disable if user is out of work |
| REQ – 12 | 1 | User account shall be deleted by the manager or owner if user no longer work for the company |

1. On Screen Appearance Requirements

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Weight Priority** | **Requirement** |
| REQ – 0 | 5 | The screen shall display the application login page |
| REQ – 1 | 5 | The data entry page shall be display on the screen |
| REQ – 2 | 3 | The users list shall be display on the screen |
| REQ – 3 | 2 | The items list shall be display on the screen |
| REQ – 4 | 2 | The price of item shall be display on the screen |
| REQ – 5 | 2 | The quantity of the item shall be display on the screen |
| REQ – 6 | 2 | The inventory shall be display on the screen |
| REQ - 7 | 4 | The menu of the application shall be display |
| REQ - 8 | 2 | The choice menu option shall be display on the screen |

3. Functionality Requirements Specification

1. Stakeholders
   1. Users
   2. Supervisors
   3. Managers
   4. Owner/Company
   5. Customers
2. Actors and Goal
   1. Initiating
      1. Users
      2. Supervisors
      3. Managers
      4. Customers
   2. Participating
      1. Computers
      2. Barcode scanners
      3. Printers
      4. Fax machines
      5. Power sources
3. Use Cases
   1. Casual Description
      * 1. UC-1: The login/ create an account page

The User Interface will display the login page which will ask the users to enter their credentials before using the system.

* + - 1. UC-2: View menu

The user interface after user login will display the system menu which will allow the users to choose the window they would like to work in.

* + - 1. UC-3: Data entry page

The User Interface will display the data entry page which will ask users to enter information for new users, and also names, numbers, quantities, and prices for products.

* + - 1. UC-4: View entries lists

The User Interface will display the different lists of entities one at a time, according the user in which the user would like to work.

* + - 1. UC-5: return to menu

The User Interface will display the possibility to the user to go back to the menu from one window and go to another one.

* + - 1. UC-6: Close windows and application

The User Interface will display the possibilities to users to close an opened window and close the application.

* 1. Use Case Diagram

Use Case 1:

User login

Use Case 2:

User menu access

Database

Application

Use Case 3:

Data entries

User

Data storage

User Interface

Use Case 4:

Lists views

Use Case 5:

Back to menu

Use Case 6:

Close app/ window

* 1. Traceability matrix

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | UC-1 | UC-2 | UC-3 | UC-4 | UC-5 | UC-6 |
| REQ 0 |  |  | x |  |  |  |
| REQ 1 | x |  |  |  |  |  |
| REQ 2 |  |  | x |  |  |  |
| REQ 3 |  |  | x |  |  |  |
| REQ 4 |  |  | x |  |  |  |
| REQ 5 |  |  | x |  |  |  |
| REQ 6 | x |  |  |  |  |  |
| REQ 7 |  | x |  |  |  |  |
| REQ 8 |  |  | x |  |  |  |
| REQ 9 |  |  | x |  |  |  |
| REQ 10 |  |  | x |  |  |  |
| REQ 11 |  | x |  |  |  |  |
| REQ 12 |  |  |  | x |  |  |
| REQ 13 |  |  |  |  |  |  |
| REQ 14 |  |  |  |  |  |  |
| REQ 15 |  |  | x |  |  |  |
| REQ 16 |  |  | x |  |  |  |
| REQ 17 |  |  | x |  |  |  |
| REQ 18 |  |  |  |  |  |  |
| REQ 19 |  |  | x |  |  |  |
| REQ 20 |  |  |  |  |  |  |
| REQ 21 |  |  | x |  |  |  |
| REQ 22 |  |  |  |  |  |  |
| REQ 23 |  |  |  |  |  |  |
| REQ 24 | x |  |  |  |  |  |
| REQ 25 | x |  |  |  |  |  |
| REQ 26 |  |  |  |  |  |  |
| REQ 27 |  |  |  |  |  |  |
| REQ 28 |  |  |  |  |  |  |
| REQ 29 | x |  |  |  |  |  |
| REQ 30 |  |  | x |  |  |  |
| REQ 31 |  |  |  | x |  |  |
| REQ 32 |  |  |  | x |  |  |
| REQ 33 |  |  |  | x |  |  |
| REQ 34 |  |  |  | x |  |  |
| REQ 35 |  |  |  | x | x |  |
| REQ 36 |  | x |  |  |  |  |
| REQ 37 |  | x |  |  |  | x |
| Tot. PW | 13 | 10 | 36 | 13 | 4 | 2 |

* 1. Fully-dressed description

Our inventory management system software is made to help users to manage all inventory activity of the stock of production related to the manufacturing. So users who need to use this system are given the requirement to create their account and login credential. The software is used to enter and retrieve information about users and products from the database of the system. The user can remain logged as long as is would like to use the system, but must logout after use.

|  |
| --- |
| Use Case UC-1 Login/create an account |
| Related requirements: REQ 1, REQ 6, REQ 24, REQ 25, REQ 29  Initiating Actor: Users/Employees  Actor Goal: Register in the database  Participating Actors: Computers  Preconditions: The system will require to users the information needed to  register.  Postconditions: Users information will be stored in the database server. |
| Flow of events for Main Success Scenario |
| 1 - The User opens the software to the login page.  2 - The system ask to the user to create an account for new user, or enter  credentials for login.  3 - the user can create account by entering the requested information, or  enter his credentials to login.  4 - The system will collect the information given and insert in  the database for new user, or verify the given credential for existing  user.  5 - After successful verification the system will give access to the user,  overwise the system return to step 2 .  6 - The user will be using the system for his need. |

|  |
| --- |
| Use Case UC-2 View Menu |
| Related requirements: REQ 7, REQ 11, REQ 36, REQ 37  Initiating Actor: Users/Employees  Actor Goal: Choose the part of the system they would like to work with.  Participating Actors: Computers  Preconditions: The users have the menu page to choose what they would like  to work with.  Postconditions: the section of work is opened after user made their choice. |
| Flow of events for Main Success Scenario |
| 1 - Users access menu page  2 – System display options to the users  3 – Users choose work option  4 – System opens chosen work option to users  5 – Users access section for work needed. |

|  |
| --- |
| Use Case UC-3 Data Entry |
| Related requirements: REQ 0, REQ 2, REQ 3, REQ 4, REQ 5, REQ 8,  REQ 9, REQ 10, REQ 15, REQ 16, REQ 17, REQ 18  REQ 21, REQ 30  Initiating Actor: Users/Employees.  Actor Goal: Enter data in the system.  Participating Actors: Computers, barcode scanners  Preconditions: The system ask to enter data as needed  Postconditions: Data is stored in the system server. |
| Flow of events for Main Success Scenario |
| 1 – User access the data entry page  2 – The system ask to enter data  3 – User enter data as needed |
|  |

|  |
| --- |
| Use Case UC-4 View Entries lists |
| Related requirements: REQ 12, REQ 31, REQ 32, REQ 33, REQ 34,  REQ 35  Initiating Actor: Users/Employees  Actor Goal: View the data previously entered from the data lists.  Participating Actors: Computers  Preconditions: The users click on the list they want to access to  .  Postconditions: the list open and user have access to it |
| Flow of events for Main Success Scenario |
| 1 - Users access click on the list they want to access  2 – System displays the list to the users  3 – Users access the list for view. |

|  |
| --- |
| Use Case UC-5 Return to Menu |
| Related requirements: REQ 35  Initiating Actor: Users/Employees  Actor Goal: to go back to the main menu.  Participating Actors: Computers  Preconditions: The users click to the menu button    Postconditions: the system open the menu page |
| Flow of events for Main Success Scenario |
| 1 - Users click on menu button  2 – System displays the menu to the users. |

|  |
| --- |
| Use Case UC-6 Close application/Windows |
| Related requirements: REQ 37  Initiating Actor: Users/Employees  Actor Goal: To close a window or the application login page.  Participating Actors: Computers  Preconditions: The users click on the exit button    Postconditions: The page or the application closes. |
| Flow of events for Main Success Scenario |
| 1 - Users click on the exit button.  2 – System closes the window or the application in used. |

1. System Sequence Diagrams

This system sequence diagram below describes the interactions between main actors of the system.

UC – 1: Login page

Users System Database

Open login page by users

Request to enter info / credentials

Users input information / credentials

Register / verify entries by the system

Allowance of access to users

Use of the system by users

UC – 2: View menu

Users System Database

Open menu page by users

Options given to users to choose

Users choose Option

Access of database by the system

Opens chosen option to users

Allowance to access chosen option

UC – 3: Data entry

Users System Database

Access to data entry page by users

Request to enter data to users

Entries of data by users

Store data in the server by the system

UC – 4: View Entries lists

Users System Database

Request to enter lists by users

Retrieve data from database by system

Allowance to access list by system

Access to lists by users

5. Effort Estimation using Use Case Points:

UCP = UUCP x TCF x ECF (4.1 from lecture notes)

135.641 = 118 x 0.95 x 1.21

Duration = UCP (use case points) x PF (Productivity Factor)

Duration = 135.641 x 28 (assumed via instructions)

Duration = 3797.95 / hours

So for 4 software engineers working 30/hrs a week a piece with 10 hours weekly going to corporate overhead, it would take:

3797.95 \ (120) = 31.6 weeks for this project.

6. Domain Analysis

1. Domain Model
   * 1. Domain model Diagram

-Application instructions

-Employee Info -Product Info

System code

Database

User Interface

-Data Entrée -Data Retrieve

-Application Storage -Application Usage

Computer

Employee

* + 1. Concept definition(D-Doing; K-Knowing, N-Nether)

|  |  |  |
| --- | --- | --- |
| **Responsibility Description** | **Type** | **Concept Name** |
| Stores the system application | D | Computer |
| Allow user to input data | K | User Interface |
| Check if Data is Input | k | User Interface |
| Instructs computer to communicate with User | D | System Code |
| Allows User to store information | D | Database |
| Allow communication application-User | D | Computer |
| Allows user to retrieve data | D | User Interface |
| Allows communication User-Database | D | System Code |
| Allows user to retrieve information from | D | Database |
| Allows communication user-Database | D | User Interface |
| Display information from users | D | User Interface |
| communicate with user interface | D | Database |
| Displays information from Database | D | User Interface |

* + 1. Association Definition

|  |  |  |
| --- | --- | --- |
| **Concept Pair** | **Association Description** | **Association Name** |
| Computer | The System application is stored into the computer storage. | StoreApp |
| User Interface | The user interface get data from the user and the database | GetData |
| System Code | The system code gives command the computer system the execute the code instruction | Giveinstruction |
| Database | The database stores information from the user. | StoreInfo |
| Computer | The computer allows user to get information from database. | GetInfo |
| Database | The user collect information from the database. | ProvidInfo |
| User Interface | The user interface input data into the database. | InputData |

* + 1. Attribute Definitions

|  |  |  |
| --- | --- | --- |
| **Concept** | **Attribute** | **Attribute Description** |
| Computer | AppStorage  UserAppTool | The computer serves as a storage for the application  The user uses a computer ton interact with the application system |
| User Interface | InputInterface  OutputInterface | The user interface is used to input data into the database  The user interface is used to output information from database |
| System Code | SystemInstuction | The system code hold instruction with are uses put the processor to control the communication between the user and the database. |
| Database | UserInfoList  ProductInfoList | The database holds the list of information about the employees  The database holds the list of information about the products |

* + 1. Traceability Matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Priority Weight | Use Case | Computer | User Interface | Database | System Code |
| 13 | UC1 | x | x |  | x |
| 10 | UC2 | x | x |  | x |
| 36 | UC3 | x |  | x | x |
| 13 | UC4 | x |  | x | x |
| 14 | UC5 | x | x |  | x |
| 2 | UC6 | x |  |  |  |

1. System Operations Contract(Responsibility Description)

|  |  |
| --- | --- |
| Operation | Register |
| Precondition | * No account for the User * User will create account by entering:   Username – Password - Name – address- Title |
| Postcondition | * User information is stored into the database |
|  |  |
| Operation | Login |
| Precondition | * User has his /her login information stored in the database |
| Postcondition | * User access system menu |
|  |  |
| Operation | Input Data |
| precondition | * User select input data option |
| Postcondition | * Data input is stored into the database |
|  |  |
| Operation | Output Data |
| Precondition | * Information is stored into the database * Information reference Id is input |
| Postcondition | * Data is display on user interface |
|  |  |
| Operation | View Data List |
| Precondition | * View list option is selected |
| Postcondition | * Data list is display on user interface |

1. Interaction Diagrams
   1. Sequence Diagrams

These sequence diagrams below show the logical function of the system with the different Use Cases. They show the interaction between Objects and Classes involved into the system functions.

Participating actor

Participating actor

Initiating actor

as

**System**

**Server**

**User**

**Database**

USE CASE

GUI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **UC 1: User login scenario**   1. System access request signal sent to system 2. Identifications request signal is sent to user 3. Identifications provided signal is sent to Sys.     6. Access authorization signal sent to user  **UC 2: Menu access scenario**   1. Access authorization to menu signal 2. Chosen menu’s item open request signal 3. Open item authorization signal sent   **UC 3: Data entry/retrieve scenario**   1. Data entry/retrieve signal sent to system | Separation block | Signal messages   1. Identification verification request 2. Successful verification signal 3. Data store/retrieve request 4. Data store/retrieve authorization 5. Successful Data store /retrieve |  |

Participating actor

USE CASE

Participating actor

Initiating actor

as

**Database**

**Server**

**System**

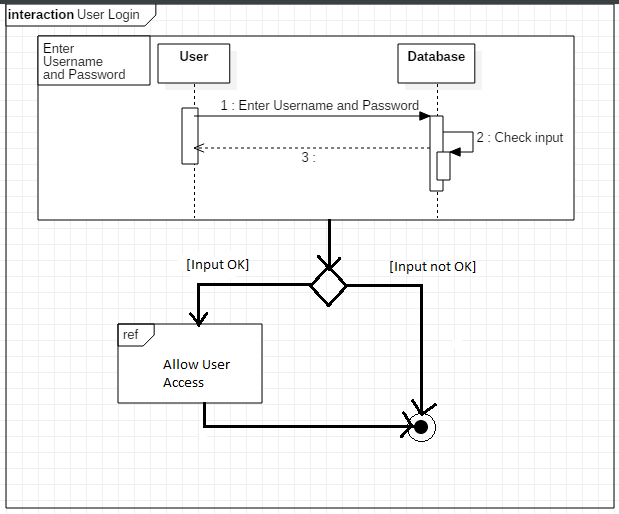
GUI

**User**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **UC 4: View data lists scenario**   1. Access list request signal sent to system      1. Access list authorization signal 2. Gain access to list signal sent to system       **UC 5: back to Menu scenario**   1. Back to menu request signal 2. Access to menu authorization signal 3. Gain access to Menu signal   **UC 6: Closing App/window scenario**   1. Close App/window request signal 2. Close App/window authorization signal 3. Successfully close App/window signal | Separation block | Signal messages   1. Data access request signal 2. Data access authorization signal 3. Successful Data retrieve signal |  |

* 1. UML Interactions Diagram

This diagram shows the interaction for the user login scenario

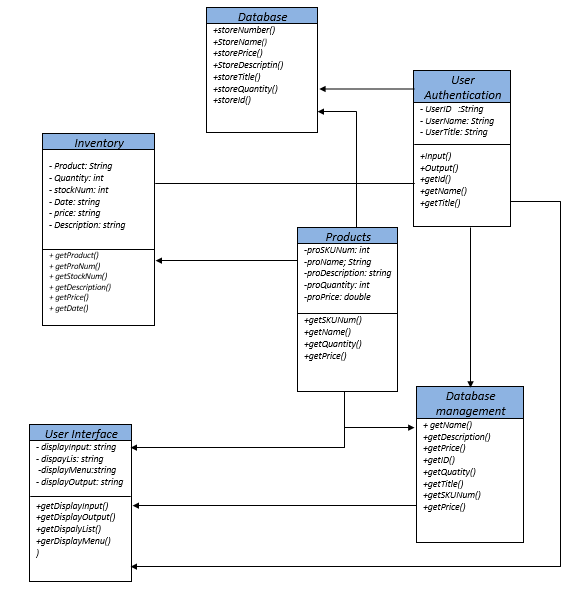


* 1. Design Patterns:

A Pub-Sub Pattern Strategy was utilized in our design since there are many objects that need to have a reference to other methods to be able to pull different portions of data from our servers. These portions of data cross over into several of the categories that will be displayed simultaneously from our server/database. Also, knowing how much this portion of code is used we wanted to make sure that, going forward, it would be easily adaptable and reusable.

A Proxy Pattern was used in our design to be able to access different memory locations without having to go through complex networking protocols. This type of pattern allows us to access the database and display the needed information without increasing the demands on resources and time.

1. Class Diagram and Interface Specification
2. Class Diagram



1. Data Types and Operation Signatures

Database:

+StoreNumber(): stores numbers of products

+StoreName() : stores names of products and users

+StorePrice() : Stores price of products

+StoreDescriptin(): Store products description

+StoreTitle(): Stores users titles

+StoreQuantity() : Stores quantities

+StoreId() : Stores users Identification

User authentication:

-String UserID: User identification

-String UserName: User name

-String UserTitle: User title

+ input() : input user information

+output(): output user information

+getID() : get user identification

+getName() :get user name

+getTitle() : get user title

Inventory:

-String product: name of product

-int quantity: the quantity of the product

-int stockNum: the stock number of the product

-String Date: date of the inventory

-String price: price of the product

-String description: description of the product

+getProduct() : get the product name

+getquantity(): get the quantity of product

+getStockNum():get stock number

+getDate() : get date

+getPrice(): get price of the product

+getDescription: get product description

Product:

-String proName: product name

-String proDescription: product description

-int proQuantity: product quantity

-double proPrice: product price

-int proSKUNum: product SKU Number

+getSKUNum(): get product SKU number

+getName(): get product name

+getQuantity(): get product quantity

+getPrice(): get product price

User interface:

-String displayInput: display input page

-String displayList: display lists

-String displayMenu: display menu

-String displayOutput: display output

+getDisplayInput(): get display input

+getDisplayOuput(): get display output

+getDisplayList(): get display list

+getDisplayMenu(): get display menu

Database management:

+getName() : get product, user name

+getDescription(): get product description

+getPrice(): get price

+getID(): get user identification

+getQuantity(): get product quantity

+getTitle(): get user title

+getSKUNum(): get product SKU number

+getPrice(): get product price

1. Traceability matrix

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | Class |  |  |  |
| Domain  concept | Database | User authentication | Inventory | Product | User interface | Database management |
| ID | x | x |  |  |  | x |
| Name | x | x | x | x |  | x |
| Title | x | x |  |  |  | x |
| Quantity | x |  | x | x |  | x |
| Number | x |  | x | x |  | x |
| Date |  |  | x |  |  |  |
| Price | x |  | x | x |  | x |
| Description | x |  | x | x |  | x |
| Input |  |  |  |  | x |  |
| Menu |  |  |  |  | x |  |
| List |  |  |  |  | x |  |
| Output |  |  |  |  | x |  |

1. Design Patterns

* A Pub-Sub Pattern Strategy was utilized in our design since there are many objects that need to have a reference to other methods to be able to pull different portions of data from our servers. These portions of data cross over into several of the categories that will be displayed simultaneously from our server/database. Also, knowing how much this portion of code is used we wanted to make sure that, going forward, it would be easily adaptable and reusable.

1. Object Constraint Language Contracts

* Context Database

Inv: self. StoreNumber() >= 0

Inv: self. StorePrice() >= 0

Inv: self. StoreQuantity() >= 0

Inv: self. StoreID() >= 0

* Context UserAuthentication

Inv: self.UserID() >= 0

Inv: self.getID() >= 0

Context String::UserName(parameter1 : string1…): String

Pre: parameter1 != null

Post: result != null

* Context Inventory

Inv: self.quantity() >= 0

Inv: self.stockNum() >= 0

Inv: self.price() >= 0

Inv: self.getquantity() >= 0

Inv: self.getStockNum() >= 0

Inv: self.getPrice() >= 0

Context String::Product(parameter1 : string1…): String

Pre: parameter1 != null

Post: result != null

* Context Product

Inv: self.proQuantity() >= 0

Inv: self.proPrice() >= 0

Inv: self.proSKUNum >= 0

Inv: self.getQuantity() >= 0

Inv: self.getPrice() >= 0

Context String::Proname(parameter1 : string1…): String

Pre: parameter1 != null

Post: result != null

* Context Database Management

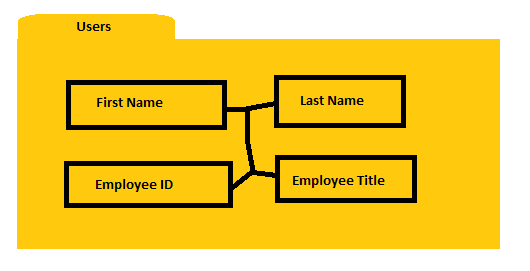
Inv: self.getPrice() >= 0

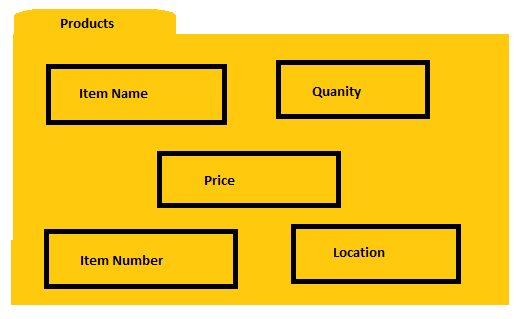
Inv: self.getID() >= 0

Inv: self getQuantity() >= 0

Inv: self.getSKUNum() >= 0

1. System Architecture and System Design
2. Architectural Styles:
   * Data-Centric Architecture – The main proponents of our design were focused around database implementation. We used a relational database management system for accessing all the stored and added data within our Inventory Management System
   * Component-based Architecture – All of our system processes are placed in similarly related modules that encapsulates related functions (or data). We also incorporate a User Interface that allows us to navigate among these different related components etc.
3. Identifying Subsystems:





1. Mapping Subsystems to Hardware:
   * Our System can run on the same machine, there will be no need to place computation onto separate machines.
2. Persistent Data Storage
   * We will be using a relational database to save all the items in our inventory, including other information about them as location, price, and quantity.
3. Network Protocol
   * N/A
4. Global Control Flow
   * *Execution orderness*: The User Interface will allow us to access any of the branches of the system within 2 clicks. (once to go back to main menu, then select your preferred component)
   * *Time dependency*: There is a time-limit in our system that will automatically log the user out. (potentially around 5-10 mins)
   * *Concurrency*: N/A
5. Hardware Requirements
   * Standard Screen display will be sufficient. Disk storage space will be dependent on the amount of information added to the inventory items, and also how many items are going to be added to the system. Scanners used for inputting inventory into the system. Printers used for document control
     1. User Interface Design and Implementation

a. Preliminary Design

Inventory Management System User Interface

**INVENTORY MANAGEMENT SYSTEM**

**User Login**

**Username**

**Password**

Login

Exit

1. Project Management

Create an Account

**INVENTORY MANAGEMENT SYSTEM**

**Create an Account**

**First Name**

**Last Name**

**Employee ID**

**Employee Title**

Exit

SUBMIT

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INVENTORY MANAGEMENT SYSTEM** | | | | | |
| MAIN MENU | | | | | |
| Users | Products | SKU Numbers | Prices | Quantities | Stock Locations |
| Back  Exit | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INVENTORY MANAGEMENT SYSTEM** | | | | | |
| USERS | | | | | |
| Users | Products | SKU Numbers | Prices | Quantities | Stock Locations |
| Name | | Id Number | | Position | |
|  | |  | |  | |
|  | |  | |  | |
|  | |  | |  | |
|  | |  | |  | |
| Exit  Back | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INVENTORY MANAGEMENT SYSTEM** | | | | | |
| Products | | | | | |
| Users | Products | SKU Numbers | Prices | Quantities | Stock Locations |
| Description | | SKU Number | | Location | |
|  | |  | |  | |
|  | |  | |  | |
|  | |  | |  | |
|  | |  | |  | |
| Exit  Back | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INVENTORY MANAGEMENT SYSTEM** | | | | | |
| SKU Numbers | | | | | |
| Users | Products | SKU Numbers | Prices | Quantities | Stock Locations |
| SKU Number | | Products | | Locations | |
|  | |  | |  | |
|  | |  | |  | |
|  | |  | |  | |
|  | |  | |  | |
| Exit  Back | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INVENTORY MANAGEMENT SYSTEM** | | | | | |
| Prices | | | | | |
| Users | Products | SKU Numbers | Prices | Quantities | Stock Locations |
| Prices | | Products | | SKU Number | |
|  | |  | |  | |
|  | |  | |  | |
|  | |  | |  | |
|  | |  | |  | |
| Exit  Back | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INVENTORY MANAGEMENT SYSTEM** | | | | | |
| Quantities | | | | | |
| Users | Products | SKU Numbers | Prices | Quantities | Stock Locations |
| Quantities | | Products | | SKU Numbers | |
|  | |  | |  | |
|  | |  | |  | |
|  | |  | |  | |
|  | |  | |  | |
| Exit  Back | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INVENTORY MANAGEMENT SYSTEM** | | | | | |
| Stock Locations | | | | | |
| Users | Products | SKU Numbers | Prices | Quantities | Stock Locations |
| Stock locations | | Products | | SKU Numbers | |
|  | |  | |  | |
|  | |  | |  | |
|  | |  | |  | |
|  | |  | |  | |
| Exit  Back | | | | | |

1. User Effort Estimation

Fully Dressed UC-1 – Register Account / Log-in

Navigation: Log-in screen appears as soon as software is opened

* + First Screen displayed once software is ran

Data Entry: total 3 mouse clicks, and keystrokes suitable to enter Username and Password as follows,

* + Determine if you are logging in or created an account then click in the correct Username box
  + Enter the respective Username
  + Click on the password box that follows what you are trying to do (login/create)
  + Enter the respective password
  + Click on the login button.

Fully Dressed UC-5 – Return to the main menu

Navigation: When using the system, the main menu is always one click away

* + From any screen after login, just click on the back button

Data Entry: N/A

Fully Dressed UC-6 – Exit the Program

Navigation: When using the system, exiting the program is always one click away

* + From any screen throughout the entire system, just click on the exit button

Data Entry: N/A

The main change to the UI was adapting all the elements to accept the forms and create the links between all the components. The major “Ease-of-Use” change was with the navigation between the different parts of the program. Instead of requiring two-button clicks to return to main menu, and then select what portion of the program you require, you can now change to each individual portion from every screen that is included in the program.

* + 1. Design of Tests

The design of test is to testing all the actors involved in the well functioning of the system. As units that need to be tested there are: The user interface The system application The database.

1. Testing the user interface

Goal: The user interface is the system interface that is use to input an retrieve information about user or item from the system.

Test cases:

* **Check if input requirements are satisfied**

This step allow the system to make sure all the information required are input before the system can proceed for verification.

* **Check for valid input**

This step is to verify if the information input are correct.

* **Check if the system display**

This step is to make sure that the information which are the system output are displayed.

1. Testing the system application

Goal: The system application is the software part of the system, it allows the exchange of data between database and user interface.

Test cases:

* **Check for input**

In this case, the application verifies if input has been made.

* **Check for valid input**

In this case the application verifies if the input required are valid

* **Check if data are output or display**

When user request for data, the application check to make sure the data have been provided to the user.

1. Testing database

Goal: The database is the container of the system that stores all the information about the system.

Test case:

* **Store data**

This step is to check if the database stores data.

* **Store input data**

This case is to make sure the input is stored in the database.

* **Check input**

This step is to check if the input matches with the stored data

* **Check for data retrieve**

This case is to check if the data can be retrieved from database.

* **Check for output**

This case is to check if data retrieve from database is output.

**13. History of Work, Current Status, and Future Work**

History of Work

Comparing our project milestones from the previous reports, we met each timeline except for one, which was implementing the user interface, database, and functionality all together (which was set to be done between 10/29 and 11/4 (see attached Gantt Chart)). Timelines in the Gantt chart are based off of projected timelines at the beginning of the class (meaning this disregards assignment due dates that were pushed back).

Key Accomplishments

* All due dates were on time
* All milestone projections (except for implementing the UI, database, and functionality together) were on time
* Created a real word software application that has great functionality
* Formed a successful team that worked well together
* Learned and grew as professionals from this software project journey

Future Work

Future work for this project will be to work on implementing the functionality, UI, and database together to work as one. One issue specifically we will be addressing is adding employee ids to the user functionality, so that the IDs will be the primary key in our database for the users.

# 14. References

Marsic, I. (2012, February 16). *Report #2: SYSTEM DESIGN*. Retrieved from Software Engineering Project Report: http://www.ece.rutgers.edu/~marsic/Teaching/SE/report2.html