|  |
| --- |
| IUPUI CSCI 45200 – SPRING 2014 |
| The Healthy People Pharmacy System |
| Agile Development – Elaboration Phase |
|  |
| **Naeem Tai** |
| **3/3/2014** |

|  |
| --- |
| This document is prepared for the Healthy People Pharmacy system during the Elaboration phase of the Agile Development process. This document is prepared in accordance with the Assignment 2 requirement and contains Answers to the Asked Questions, Conceptual Class List, Risk Analysis & Organization, UML Interaction Diagrams, UML Class Diagrams, and System Architecture Discussion. |

# Answers to the Questions:

1. List of Potential Conceptual Classes:
2. Physical Objects: Register
3. Business Transactions: Sale, Payment
4. Roles of Payment: Employee, Cashier, Pharmacist
5. Events: Sale
6. Products: Item, General Merchandise Item, Pharmaceutical Item, Rental Item
7. Record: Receipt
8. Critical Risks of the System:

Readdress risks defined in Inception phase specifically. Look at Risk Analysis and Organization page.

1. UML Interaction & UML Class Diagram:

UML Interaction & UML Class Diagrams derived from Use Case UC 1 and Use Case UC 2 defined in detail in Inception Phase. See attached diagrams.

1. System Architecture Discussion:

We plan to organize the Healthy People Pharmacy system in a strict layered architecture. See discussion in the System Architecture Discussion section.

# Conceptual Class List:

* Physical Objects:
  + Register
* Business Transactions:
  + Sale
  + Payment
* Roles of Payment:
  + Employee
  + Cashier
  + Pharmacist
* Events:
  + Sale
* Products:
  + Item, General Merchandise Item, Pharmaceutical Item, Rental Item
* Record:
  + Receipt
  + Store

# Risk Analysis and Organization:

* Business
  + Changes in items’ prices: update prices as they change

This is the most critical risk since the items’ prices change frequently with marketing strategies. The system will interact with a product catalog that contains all the information about all the items. The product catalog will contain the item’s id, description, price, and other information such as rental period for rental items. Any price changes will be made to the product catalog. When an item is entered by the cashier, the system will interact with the product catalog to present its description, price, and other related information. This way our system can easily comply with changes in items’ prices.

* Laws/Rules/Regulations
  + Changes in medicine regulations: collaborate with new Medicine Regulations System

The system will interact with the external Medicine Regulations System to comply with regulations on sale of pharmaceutical items. Since the Medicine Regulations System is an external system, we can easily switch it out to a new system. Thus it will be easy to comply with changes in regulations on sale of pharmaceutical items.

* Technical
  + System fail/error: erase any incomplete transactions

To account for system fail/error in the middle of a transaction, the system will not record any incomplete transactions. The system will only record a transaction when it is fully completed, meaning that the payment is accepted and the sale is completed. In the event of a system fail/error, the user will no longer be able to use the system until the issue has been resolved. The system will be restarted with no knowledge of the incomplete transaction it had previously.

* Resource
  + Loosing connection with external systems: alert error and suggest alternative way to perform operation

The system will check its connection with all the external systems periodically. In the event of losing connection with any external system, the system will display an error message and prevent performing a transaction. This way, we can assure that the system is in a working condition all the time.

* Schedule
  + Spend about six weeks on each iteration; enhance the system every iteration

This is the least critical risk since it is up to the development team to decide how much to take on every iteration and what to do. Depending on the availability of needed resources, developers may decide to work ahead or put something off till the next iteration. Since we are in the agile development process, we have the continuous cycle of refinement process to improve the system.

# System Architecture Discussion:

We plan to organize the system in a layered architecture. The system will be separated into the following layers: user interface, application, technical, and foundation. The user interface layer will make up the part of the system the users see on the registers. The user interface will be connected to the supplication layer, which will handle the tasks specified by the users, control the logic in the application, and controls the user interface layer. The application layer will be connected to the technical layer to pass down the information to the lower level of the system. The technical layer will provide a connection with the foundation layer of the system to record and update data. The foundation layer, also the lowest level of our layered architecture, will be consisting of the inventory system, Payment Authorization System, and the Medicine Regulations System.

Specifically, we plan to organize the system in a strict layered architecture. This way any layer can only communicate with a layer directly below or above it. For example, we do not want direct communication between the user interface layer and the foundation layer. This will ensure us a better security for the system. Additionally, it will also make the development of the system simpler because we will be able to designate layer specific responsibilities to developers who will not need to intercommunicate constantly. The strict layered approach will also make system upgrades and refinement easier and quicker because we will only need to make changes to a specific layer without worrying about disturbing the whole system. If needed, we will easily be able to switch out a layer in future. For example, switching out the Medicine Regulations System when regulations on sale of pharmaceutical items change. Thus, organizing the system in a strict layered architecture will allow us to separate the concerns from the high level to the low level of the system, reduce coupling, support reuse of a layer, and make the system clear to make development easier.