Formal Meeting Records

-------------------------------------------------------------------------------------------------------------------------

Construction Phase 1 – Meeting 1

Date:

Time:

Agenda:

We chose the roles that we wanted to take for this construction phase of our project. Sidd, our manager, then assigned us the tasks that we need to deliver for each of us.

Sidd and Evan were then tasked to do the programming of our project together. They would then split up the functions among themselves to work on. They also have to discuss the important algorithms to take note, and also at the same time, they would have to make sure that their codes combined would work.

While the 2 of them work on the programming part of the project, the rest of the team members would have to do other parts concurrently. We would then clarify with one another when we had any questions.

|  |  |
| --- | --- |
| Sidd (Manager, designer) | Programming – inventory, stock, encryption |
| Evan (Lead designer) | Programming – user, login, interface, encryption |
| Darren (System integration, Documenter) | SRS, Meeting minutes, summary report, risk analysis |
| Justine (System integration, Documenter) | Deployment diagram, SRS, summary report, gantt chart |
| Dominic (Documenter) | State diagram, SRS, Activity diagram |
| Pavinder (???) | ??? |

Lack – test case

-------------------------------------------------------------------------------------------------------------------------

Construction Phase 2 – Meeting 2

Date:

Time:

Agenda:

A detailed and planned gantt chart prepared by Justine was sent out to the team such that each member would have a rough idea of when their task needs to be done and delivered to the team leader.

Most part of the coding was expected to be done. The team then reviewed the code together. As of yet, user, inventory and stock classes have been completed. Basic user interface is also working. Encryption and validation are halfway done. The team also then identify some areas of the code where it could be improved on, and the 2 programmers made the changes on the spot.

The team then reviewed all the currently completed diagrams together. Deployment diagram, state diagram and SRS. As a team, we all went through each item and identified quite a number of mistakes. The person in charge of each item had to amend to the necessary changes to make it for the second iteration

-------------------------------------------------------------------------------------------------------------------------

Transition Phase – Iteration 1

Date:

Time:

Agenda:

As the team approaches transition phase of the project, the Warehouse Management Tool has been completed without any compiling errors. The team then did a quick review of the program to make sure that there isn’t any errors and validations are done. The team then review the test case documents that was done by \_\_\_\_\_\_. The team then checked all the diagrams from the previous meeting, to make sure that all the mistakes made were rectified before placing it into report and SRS. Some small and final changes were made on several test cases to better reflect the purpose of those functions. Changes to the actual program was expected. Gantt chart was updated to fit the actual progress of the team. After that, a final report is to be done by Justine and Darren.

-------------------------------------------------------------------------------------------------------------------------

# Risk Analysis and Counter Measures

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Impact Type** | **Risk Seriousness (%)** | **Likelihood of Occurrence (%)** | **WBS**  **(affected work / task )** | **Risk Description** |
| 1 | Estimation of project | 80% | 10% | AL | Certain phase of the program development process was miscalculated, thus more time is needed to complete. This then results in lesser time left for other phases of program development. |
| 2 | Loss of employee | 90% | 10% | ALL | When the developer / employee leaves the company, a certain set of skills would be lacking in that particular aspect – such as a manager who manages the stock leaves. Thus, delaying project. |
| 3 | Lack of specific skillset | 90% | 10% | ALL | Not all team members would be exceptionally good in doing coding, for example C++. |
| 4 | Data Loss | 100% | 2% | ALL | Data corruption or failure to back up |
| 5 | Poor productivity | 70% | 10% | ALL | As quite a long duration of time was given for the project, the sense of urgency would be very low, and team members would tend to procrastinate. This then results in time lost in the early phase of the project. |
| 6 | Deadline | 60% | 10% | ALL | Too many concurrent projects ongoing at the same time |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Risk Description** | **Proposed Management Plan** | **(Possible) Reduction in Risk Seriousness (%)** |
| 1 | Certain phase of the program development process was miscalculated, thus more time is needed to complete. This then results in lesser time left for other phases of program development. | Allocate a longer period of time for the harder parts of the program during the planning phase. | -30% |
| 2 | When the developer / employee leaves the company, a certain set of skills would be lacking in that particular aspect – such as a manager who manages the stock leaves. Thus, delaying project. | Have the team share a same set of skills and knowledge, therefore they could cover up for one and other when in times of need. | -20% |
| 3 | Not all team members would be exceptionally good in doing coding, for example C++. | Some codes could be outsourced online, or get help from friends etc | -50% |
| 5 | As quite a long duration of time was given for the project, the sense of urgency would be very low, and team members would tend to procrastinate. This then results in time lost in the early phase of the project. | Manager should have schedule checks on team members regularly, checking on their progress. Alternatively, a set deadline could be given to each member so that they would have to submit their part by the time. | -50% |
| 4 | Data corruption or failure to back up | Plan 1: Do multiple backups of data on different devices; e.g. each member keep a copy of the data | -100% |
| Plan 2: Save regularly | -90% |
| 6 | Too many concurrent projects ongoing at the same time | Plan 1: Prioritize the more important ones to finish first | -30% |
| Plan 2: Request for an extend of deadline | -50% |

-------------------------------------------------------------------------------------------------------------------------

SRS part

-------------------------------------------------------------------------------------------------------------------------

3.1 SYSTEM FEATURES - Add Stock

|  |  |  |  |
| --- | --- | --- | --- |
| Number | What is it? | Function | Explanation |
| 3.1.1 | Description and Priority | Add Stock | Adding a stock is the utmost basic requirement of any system. Of course, this function has a high priority because it would definitely be a need for the Management. |
| 3.1.2 | Stimulus/Response Sequences |  | 1. User selects Add Stock option 2. – Program prompts for stock id 3. User enters desired stock id 4. – Program do a check to see if stock name is unique 5. – Program would retrieve latest stock ID + 1 and store into the array if stock name is unique 6. – Program prompts for item description 7. User enters item description 8. – Program stores string of information into stock file 9. – Program prompts for item category 10. User enters item category 11. – Program stores string into stock file 12. – Program prompts for sub item category 13. – User enters sub item category 14. – Program stores string into stock file 15. – Program prompts for stock quantity available 16. User enters stock quantity 17. – Program would do a validation check to ensure if data entered is a positive integer before adding into stock file 18. – Program prompts for item price 19. User enters item price 20. - Program would do a validation check to ensure if data entered is a positive integer before adding into stock file 21. – Program prompts user for date 22. User enters date 23. – Program stores string into stock file 24. – Program will then display item id and item description on screen for user and updates stock file |
| 3.1.3 | Functional requirements |  | * -------------------------------------------------- |
|  |  |  |  |

3.1 SYSTEM FEATURES – Remove Stock

|  |  |  |  |
| --- | --- | --- | --- |
| Number | What is it? | Function | Explanation |
| 3.1.1 | Description and Priority | Remove Stock | Removing stock is also the utmost basic requirement of any system. It could cater to mistakes made by the user or a particular stock has been sold. Of course, this function has high priority because it would definitely be a need for the Management |
| 3.1.2 | Stimulus/Response Sequences |  | 1. User selects Remove Stock Option 2. – Program requests for stock id 3. User enters stock id 4. – Program does a check to see if stock id exists in stock file 5. – Program would extract data by set delimiter and store into a temp variable 6. – Program would then move each and every data via the temp variable and copy into a new file called temp.txt 7. – Program repeats step 4 to 6 until it matches said stock id from user, and skips it, continues to copy from stock file to temp 8. – When program is done with copying all records over, program deletes stock file 9. – Program renames temp file into stock file |
| 3.1.3 | Functional requirements |  | * ----------------------------------------------- |
|  |  |  |  |

3.1 SYSTEM FEATURES – Stock Summary

|  |  |  |  |
| --- | --- | --- | --- |
| Number | What is it? | Function | Explanation |
| 3.1.1 | Description and Priority | Add Stock | This function is for the convenience of the user to view all the stock in a single page which the user can view, example, stocks that are required to top up, at one go. Of course , this function has a high priority, definitely from the Manager’s perspective. |
| 3.1.2 | Stimulus/Response Sequences |  | 1. User selects stock summary option 2. – Program takes out all the dates of stocks from stock file 3. – Program stores dates into vector and sort them accordingly 4. – Program print out items with matching dates |
| 3.1.3 | Functional requirements |  | * -------------------------------------------------- |
|  |  |  |  |

3.1 SYSTEM FEATURES - Edit Stock

|  |  |  |  |
| --- | --- | --- | --- |
| Number | What is it? | Function | Explanation |
| 3.1.1 | Description and Priority | Edit Stock | Editing stock is a function for user to change particular fields that were incorrect or require a change. Of course , this function has a high priority, definitely from the Manager’s perspective. |
| 3.1.2 | Stimulus/Response Sequences |  | 1. User selects Edit Stock Option 2. – Program requests for stock id 3. User enters stock id 4. – Program does a check to see if stock id exists in stock file 5. – Program would extract data by set delimiter and store into a temp variable 6. – Program would then move each and every data via the temp variable and copy into a new file called temp.txt 7. – Program repeats step 4 to 6 until it matches said stock id from user, and pauses 8. – Program would generate a menu option for user to choose what they want to edit 9. User selects an option 10. – If user selects description, program would prompt user for a new description 11. – if user selects main category, program would prompt user for a new main category 12. – If user selects sub category, program would prompt user for a new sub category 13. – If user selects price, program would prompt user for a new price 14. – If user selects quantity, program would prompt user for a new quantity 15. – If user selects transaction date, program would prompt user for a new transaction date 16. User enters required information needed by the program to proceed 17. – Program takes the new string of data and place into temp.txt, variable by variable until the whole item is update before moving on to next item 18. – Program copy over until there is no more records 19. – When program is done with copying all records over, program deletes stock file 20. – Program renames temp file into stock file 21. – Program displays a message to alert user that item is successfully updated |
| 3.1.3 | Functional requirements |  | * -------------------------------------------------- |
|  |  |  |  |

3.1 SYSTEM FEATURES - Search Stock

|  |  |  |  |
| --- | --- | --- | --- |
| Number | What is it? | Function | Explanation |
| 3.1.1 | Description and Priority | Search Stock | Searching stock is a function for user to search for stocks with a particular keyword. Of course , this function has a high priority, definitely from the Manager’s perspective. |
| 3.1.2 | Stimulus/Response Sequences |  | 1. User selects search stock options 2. – Program prompts user for a keyword to search 3. User enters keyword 4. – Program would check user input with each field of data in stock file 5. – Program will then display all items with matching data out on the screen |
| 3.1.3 | Functional requirements |  | * -------------------------------------------------- |

4 External Interface Required

**4.1 User Interfaces**

Back-end software: Quincy (C++)

**< DISPLAY SCREENSHOTS OF QUINCY >**

**4.2 Hardware Interfaces**

- Windows

- Browser that supports CGI, HTML & JavaScript.

**4.3 Software Interfaces**

|  |  |
| --- | --- |
| **Software Used** | **Description** |
| Operating System | Windows has higher consumers, thus, users should be more comfortable with using Windows. In addition, has the best support and user-friendliness |
| Database | 2 text files which contain user and stock information respectively will be encrypted and decrypted when certain actions are detected, example, when user logs in, when user logs out... etc. |
| Quincy | We chose Quincy because every programmer is experienced in using it and are more comfortable using it. |

**4.4 Communication Interfaces**

**-** It supports every web browser

- System interacts with the text file, but only AFTER user successfully logs in and decrypts the file, which only then the user is able to view the content of the text files.

- After user logs out, file is encrypted and content of the text files are no longer able viewable.

5 Other Nonfunctional Requirements

**5.1 Performance Requirements**

The system shall function in real-time: any operation on the stored information, triggered by the User, shall complete in less than 10 seconds. The system shall allow simultaneous use by at least 100 users, without data corruption.

**5.2 Safety Requirements**

If anything cropped up in the middle of system usage, any changes made post log in not be saved. User will have to re-do any changes made again.

**5.3 Security Requirements**

In order to keep the stock/user files safe, user must first authenticate themselves by username and password. The system shall not allow access if the user fails to provide correct information.

The user is given 3 chances to key in the right information, else, the account will be locked.

**5.4 Software Quality Requirements**

**AVAILABILITY:** This system is available 24/7 as long as user wants to make a change.

**CORRECTNESS:** The stock should correspond to the stock file and update correctly based on user input.

**MAINTAINABILITY:** Programmers should maintain the functions, keep up with the times and update accordingly based on any technology change.

**USABILITY**: This system should satisfy the number of users who needs to use this system.