## Addis_Ababa_University_logo

## Addis Ababa institute of Technology

**School of Electrical and Computer Engineering**

|  |  |
| --- | --- |
| **Stream** | Computer |
| **Course Title** | Database Systems |
| **Semester/Year** | 1st /4th |
| **Instructor** | Biruk A. |

|  |  |
| --- | --- |
| **Project Title** | **Restaurant Billing and Management System Documetation** |

|  |  |
| --- | --- |
| **Submission Date** | **02/02/2018** |

|  |  |
| --- | --- |
| **Name** | **ID No** |
| Abel Tefera | ATR/4360/07 |
| Belayneh Mathewos | ATR/3742/07 |
| Terbinos Getachew | ATR/9045/07 |
| Selamu Abo | ATR/7293/07 |
| Nahom Girmay | ATR/5368/07 |

**Purpose**

The restaurant management and billing system not only keeps the record of various people like customer and products but as well as it reduces the extensive paper work in the present system.

**Overview**

This restaurant billing and management system is an application to automate the process of ordering and billiing of a specific order . This system is also aimed to add and maintain records of available prodducts , add andd maintain customer details , add and maintain description of new products, provides financial reports to the customer.This restaurant billing and management system also provides convinient solution off billing pattern, by giving an easy and user friendly enviroment for the users.

**Introduction**

This restaurant billing and management system is going to be designed to solve some problems those some restaurants are facing. To do this design practical the system has some functional specifications and for those specifications there are requerments.

The functionality specification is closely related to the use cases. The main  
functions are ordering, generate report, receipt and administration. A description of each of the functionality will be described in detail. It includes the output of the system and the input it expects from the users.In case of Technical specificationthe system is made up of two layers. At the top there is the GUI (Graphical User Interface) layer, and the bottom layer is the underlying database.

**External Interface Requirements**

**Hardware Interface**

Since our application doesn’t have any designated hardware, it doesn’t have any direct hardware interface. But so to speak, the hardware connection to the database server is managed by the underlying operating system on the machine and the database server.

**Software Interface**

This application is a desktop application. It uses java for interaction with the user and a MySQL database is used to store data. The applications works best with Java JDK 1.8 and MySQL version 5.6.\*. The minimum requirements are given below.

* The minimum requirement of java SDK on system is java 7 (JDK 1.7)
* The minimum requirement of MySQL database is version 5.0

The software can run on most platforms including Windows, Mac OS or Linux distributions. The minimum requirements of Java and MySQL mentioned above should be installed for each platform before running the application.

**Functional Requirement**

1. **Login**

Use case name: Log in

Primary actor: Staff member (Manager/Cashier)

Precondition: User has an account in the system

Main success scenario:

1. User: Enters his/her credentials
2. User: Clicks the login button
3. System: if user is an Administrator, he will be redirected to a administrator/manager window otherwise, he will be redirected to the billing window
4. System: displays the name of the user account in the right top corner of the window

Exception Scenarios:

* User has entered wrong credentials

System: notifies user “Wrong user or password entered”

1. **Logout**

Use case name: Log out

Primary actor: Staff member (Manager/Cashier)

Precondition: User has logged in

Main success scenario:

1. User: clicks drop down menu on the right top corner
2. User: clicks on logout
3. System: returns the user back to the log in page

Exception Scenarios: None

1. **Manager**

Use case 1: Add to menu

Primary actor: Manager

Precondition: Manager has logged in

Main success scenario:

1. Manager: selects the edit menu tab
2. Manager: Inserts Item name, Item type, Price
3. Manager: clicks on Add to menu button
4. System: adds the new menu to the database and updates the display

Exception scenarios: None

Use case 2: Replace menu item

Primary actor: Manager

Precondition: Manager has logged in

Main success scenario:

1. Manager: selects the edit menu tab
2. Manager: selects a menu item from the list
3. Manager: Inserts the new item information i.e. Item name, Item type, Price used to replace the selected one
4. Manager: clicks on Replace selected item button
5. System: new menu item replaces the old one in the database and updates the display

Exception scenarios: None

Use case 3: Remove menu item

Primary actor: Manager

Precondition: Manager has logged in

Main success scenario:

1. Manager: selects the edit menu tab
2. Manager: selects a menu item
3. Manager: clicks on remove from menu button
4. System: removes the menu item from the database and updates the display

Exception scenarios: None

Use case 4: create manager or cashier account

Primary actor: Manager

Precondition: Manager has logged in

Main success scenario:

1. Manager: selects the manage staff tab
2. Manager: Inserts user name, password and a status of whether he/she is a manager
3. Manager: clicks on Add user button
4. System: adds the new user to the database and updates the display

Exception scenarios: None

Use case 5: Replace/modify user account

Primary actor: Manager

Precondition: Manager has logged in

Main success scenario:

1. Manager: selects the manage staff tab
2. Manager: selects the the user account to be replaced or modified
3. Manager: Inserts the new user information i.e. user name, password, status
4. Manager: clicks on Replace user button
5. System: modifies the user account in the database and updates the display

Exception scenarios: None

Use case 6: Remove user account

Primary actor: Manager

Precondition: Manager has logged in

Main success scenario:

1. Manager: selects the manage staff tab
2. Manager: selects a user account
3. Manager: clicks on remove user button
4. System: removes the user from the database and updates the display

Exception scenarios: None

Use case 7: create a new waiter

Primary actor: Manager

Precondition: Manager has logged in

Main success scenario:

1. Manager: selects the waiters tab
2. Manager: Inserts first name and last name in the input boxes provided
3. Manager: clicks on Add waiter button
4. System: adds the new waiter to the database and updates the display

Exception scenarios: None

Use case 8: Replace/modify waiter information

Primary actor: Manager

Precondition: Manager has logged in

Main success scenario:

1. Manager: selects the waiters tab
2. Manager: selects the the waiter to be replaced or modified from the list
3. Manager: Inserts the new waiter information i.e. first name and last name
4. Manager: clicks on Replace waiter
5. System: updates the waiter information in the database and updates the display

Exception scenarios: None

Use case 9: Remove a waiter information

Primary actor: Manager

Precondition: Manager has logged in

Main success scenario:

1. Manager: selects the waiter tab
2. Manager: selects a waiter from the list
3. Manager: clicks on remove waiter button
4. System: removes the waiter information from the database and updates the display

Exception scenarios: None

User case 10: Review history of orders served by the restaurant and generate report

Primary actor: Manager

Precondition: Manager has logged in

Main success scenario:

1. Manager: selects the orders list tab
2. Manager: apply filters to narrow down the history of orders served using waiter name, cashier name, table number, ordered item, the start and end date of the order
3. Manager: clicks on search
4. System: updates the list of orders served that fullfill the given filter
5. Manager: selects a specific entry in the orders list and clicks on show details. Alternatively manager can double click on the selected item.
6. System: displays a new window containing the detailed information
7. Manager: closes the window and clicks on Generate Report
8. System: prompts user to choose location of file
9. System: saves an excel compatible, csv file that contains all the orders filtered with the manager’s selected combination of filters in the choosen location
10. **Cashier**

Use case 1: Add a single order

Primary actor: Cashier

Precondition: Cashier has logged in

Main success scenario:

1. Cashier: inserts table number, menu item and the number of menu items consumed
2. Cashier: clicks on Add order button
3. System: updates the list of menu items consumed for the given table in the restaurant

Exception scenarios: None

Use case 2: Remove a single order from a particular restaurant table

Primary actor: Cashier

Precondition: Cashier has logged in

Main success scenario:

1. Cashier: selects a table number from the drop down menu
2. Cashier: selects an item from the list of orders in the selected table
3. Cashier: clicks on Remove selected order button
4. System: removes the selected item from the list of orders for that particular table

Exception Scenarios: None

Use case 3: Print receipt

Primary actor: Cashier

Precondition: Cashier has logged in and at least one item should be consumed at the table

Main success scenario:

1. Cashier: selects a table number from the drop down menu
2. Cashier: selects the waiter from a dropdown menu
3. Cashier: clicks on print receipt button
4. System: updates the database with the receipt information and prints the receipt as a PDF.

Exception Scenarios: None

**Entity Relationship Model( ER Diagram)**

****

**3NF relationship table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | **Columns** | **Type** | **Primary Key** | **Foreign Key** | **Referenced Table** | **References Column** |
| staff\_members | Id | INT | **YES** |  |  |  |
| user\_name | VARCHAR(30) |  |  |  |  |
| Password | VARCHAR(30) |  |  |  |  |
| Is\_manager | TINYINT |  |  |  |  |
| last\_login\_date | DATE |  |  |  |  |
| last\_login\_time | TIME |  |  |  |  |
|  |  |  |  |  |  |  |
| Menu | Id | INT | **YES** |  |  |  |
| Item\_name | VARCHAR(30) |  |  |  |  |
| Item\_type | VARCHAR(30) |  |  |  |  |
| Price | FLOAT |  |  |  |  |
|  |  |  |  |  |  |  |
| Waiters | Id | INT | **YES** |  |  |  |
| first\_name | VARCHAR(30) |  |  |  |  |
| last\_name | VARCHAR(30) |  |  |  |  |
| last\_date\_worked | DATE |  |  |  |  |
| Last\_time\_worked | TIME |  |  |  |  |
|  |  |  |  |  |  |  |
| orders\_list | order\_id | INT | **YES** |  |  |  |
| table\_number | INT |  |  |  |  |
| Date | DATE |  |  |  |  |
| Time | TIME |  |  |  |  |
| Waiter\_id | INT |  | **YES** | waiter | id |
| staff\_member\_id | INT |  | **YES** | Staff\_members | id |
|  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Individual\_orders | Item\_id | INT |  | **YES** | menu | id |
| Quantity | INT |  |  |  |  |
| total\_price | FLOAT |  |  |  |  |
|  |  |  |  |  |  |  |
| number\_of\_tables | Total | INT |  |  |  |  |

**Non-functional requirements**

The system should be able to do the following constraints.

* Performance – system will give a response within 3 seconds as much as possible.
* Error Tolerance – system must be error tolerable. That means, when a user give a miss-matched input, the system should tolerate without crashing and give a pop-up message for the user.

**Challenges and Solutions**

**Challenge:** The setup and implementation of the overall architecture was one of the most challenging tasks at the beginning of this project. The group has some trouble reaching a decision on the necessary functionalities required for different stuffs of the restaurant billing and management system.

**Solution:** The used case and relationship models discussed during the building blocks for this project. Using the fully dressed descriptions, we were able to implement use cases and modify them according to the individual needs of the host, waiter, manager. The relationship model was crucial in tying everything together. It became evident that everything is interlinked and much of the code can be easily reused for different functions.

**Challenge:** When we began the implementation of the database and coding, the structure and organization of the class as well as their functionalities were very unclear. We were not sure of the required parameters and information required to implement the database. The various functionalities of different users caused some unnecessary repetitiveness in the database.

**Solution:** Using the class diagram and interface specification discussed in class, we were able to coordinate and implement an organized database containing only the necessary information and eliminating unnecessary repetitiveness. This also made the distribution of tasks easier within the group as the classes and functionalities required had now become very clear.

**References**

* A concise introduction to software engineering, By Pankaj Jalote.
* IEEE Software Engineering Standards Committee,”IEEE Std. 830-1998, IEEE
* Recommended Practice for Software Requirements Specifications”, October 20, 1998.
* Fundametals of Database Systems by Elmasri Navathe, Sixth Edition