ABSTRACT

The overall idea of the project is to generate the bills for customers and maintain the sales report daily, monthly and yearly for a restaurant online. The restaurant owners can easily keep the record of their sales and generate the report. The cashier in the counter can easily take down the orders from their customers and give them the receipt by using this online point of sale. It is convenient for the cashier to calculate the bill amount and also keep the record of the currency exchanged. The mode of payments varies from customer to customer so there is a facility of the user to select the mode of payment like Cash, Card and Pay tm.Since, the database has the record of the bill generated, it is easy to view the sales report daily, monthly and yearly. The customer can receive the softcopy of receipt as a message on their mobile phones.

It is developed in HMVC codeigniter framework and PHP as a frontend and MySql as a backend while XAMPP as its controller.Besides using the technologies mention above its an attempt to use different technology where following facility can be implemented in future.Most of the business owners has more than one source of income due to which they cannot focus their time to handle the cash counter so they employee someone to handle it. At this case the employee can also cheat by deleting some transaction. For this problem the software is also connected to the owner‘s mobile phone so that every transaction record can be viewed even if he or she is not present in the restaurant.

1.INTRODUCTION

1.1 DOMAIN STUDY

Online point of sale is designed for a restaurant to easily calculate their bills and generate report daily, monthly and yearly. Since this is online it will be easier for the business owner to use the facility of cloud storage. If the owner of the business is not present in the premises then they can easily check the record of their sales.

1.1 STATEMENT OF THE PROBLEM

Traditional way of billing is quite a bit difficult to keep record of customers and sales report. Here are some points on how it is difficult:

- Billing record gets higher and higher where it is physically impossible to secure.
- Difficult to search for the old records.
- All the details are not recorded properly.
- Data might be changed by third party.
- Chances of data loss.

Besides this problem the existing systems are not online in which there can be a problem of security. For example: If the cashier in the cash counter deletes some data or plays around with sales record, there is a loss for the business.

2. SYSTEM REQUIREMENTS

2.1 LITERATURE SURVEY

The literature survey plays a very important role in the research process. It is a source from where research ideas are drawn and developed into concepts and finally theories. It also provides the researcher a bird's eye view about the research done in that area so far. Depending on what is observed in the literature review, a researcher will understand where his/her research stands.

2.2 EXISTING SYSTEM

- Existing system are offline which can keep the record offline but not online.
- Not portable

2.2.1 DESCRIPTION

- Existing systems can give the same facility of point of sale of keeping record and managing the bills and receipt for the customer.
- When we compare to the online pos it is much better in terms of faster access, interaction with customer through online messages, etc.

2.2.2 DRAWBACKS

- Risk of mismanagement of the data when the project is under development.
- There are some security concerns.
- No proper coordination between different Applications and Users.
- Fewer User - Friendly.
- The bulk data recorded has to be stored offline which extends to concern with security
- Existing system are not online which is a risk for a business from theft by their own employee.

2.3 PROPOSED SYSTEM

2.3.1 FUNCTIONAL SPECIFICATION

Functional Requirements

The aim of the proposed system is to address the limitations of the current system. The requirements for the system have been gathered from the defects recorded in the past and also based on the feedback from users of previous metrics tools. Following are the objectives of the proposed system:

1. Admin Login:

   Login to the system by the entering valid user name and password.

2. Create receipt and record sales :

   Able to record sales and receipt.

3. Content management.

4. View the number of sales and records.

Non-functional requirement:

1. **Security:**

Only the registered user and the administrator can do the login. The system's back-end servers shall only be accessible to authenticated administrators. Sensitive data will be encrypted before being sent over insecure connections like the internet.

2. Reliability:

The system provides storage of all databases on redundant computers with automatic switchover. The reliability of the overall program depends on the reliability of the separate components. The main pillar of reliability of the system is the backup of the database which is continuously maintained and updated to reflect the most recent changes.

Thus the overall stability of the system depends on the stability of container and its underlying operating system.

3. Availability:

The system should be available at all times, meaning the user can access it using a web browser, only restricted by the down time of the server on which the system runs. In case of a of a hardware failure or database corruption, a replacement page will be shown. Also in case of a

hardware failure or database corruption, backups of the database should be retrieved from the server and saved by the administrator.

4. Maintainability:

A commercial database is used for maintaining the database and the application server takes care of the site. In case of a failure, a re-initialization of the program will be done. Also the software design is being done with modularity in mind so that maintainability can be done efficiently.

5. Portability:

The application can run on any windows system, with the proper installation of the software.

2.3.2 BENEFITS OF THE PROPOSED SYSTEM

- Business like restaurant and supermarket and easily keep record of sales.

2.4 HARDWARE AND SOFTWARE REQUIREMENTS

- **Software Requirement Specification**

Operating system : Windows XP/Vista/7/8/8.1/10
Framework : HMVC Codeignitor
Front End : Php/javascript
Back End : MySQL
Platform : Xampp Server
Documentation : Microsoft Office 2010

- **Hardware Requirement Specification (Minimum)**

Computer Processor	Dual Core
Processor Speed	2.8 GHz Processor or
Hard Disk	40 GB or more
RAM	2 GB

2.5 TOOLS SURVEY

INTRODUCTION TO PHP

PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

Instead of lots of commands to output HTML (as seen in C or Perl), PHP pages contain HTML with embedded code that does "something" (in this case, output "Hi, I'm a PHP script!"). The PHP code is enclosed in special start and end processing instructions <?php and ?> that allow you to jump into and out of "PHP mode."

What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was. You can even configure your web server to process all your HTML files with PHP, and then there's really no way that users can tell what you have up your sleeve.

The best things in using PHP are that it is extremely simple for a newcomer, but offers many advanced features for a professional programmer. Don't be afraid reading the long list of PHP's features. You can jump in, in a short time, and start writing simple scripts in a few hours.

Although PHP's development is focused on server-side scripting, you can do much more with it. Read on, and see more in the What can PHP do? section, or go right to the introductory tutorial if you are only interested in web programming.

INTRODUCTION TO MYSQL

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons:

- MySQL is released under an open-source license. So you have nothing to pay to use it.

- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
- MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

HMVC CODEIGNITER

The largest practical benefit of using an HMVC architecture is the "widgetization" of content structures. An example might be comments, ratings, Twitter or blog RSS feed displays, or the display of shopping cart contents for an e-commerce website. It is essentially a piece of content that needs to be displayed across multiple pages, and possibly even in different places, depending on the context of the main HTTP request.

Traditional MVC frameworks generally don't provide a direct answer for these types of content structures, so people generally end up duplicating and switching layouts, using custom helpers, creating their own widget structures or library files, or pulling in unrelated data from the main requested Controller to push through to the View and render in a partial. None of these are particularly good options, because the responsibility of rendering a particular piece of content or loading required data ends up leaking into multiple areas and getting duplicated in the places it is used.

HMVC, or specifically the ability to dispatch sub-requests to a Controller to handle these responsibilities is the obvious solution. If you think about what you're doing, it fits the Controller structure exactly. You need to load some data about comments, and display them in HTML format. So you send a request to the comments Controller with some params, it interacts with the Model, picks a View, and the View displays the content. The only difference is you want the comments displayed inline, below the blog article the user is viewing instead of a completely separate full comments page (though with an HMVC approach, you can actually serve both internal and external requests with the same controller and "kill two birds with one stone", as the saying goes). In this regard, HMVC is really just a natural byproduct of striving for increased code modularity, re-usability, and maintaining a better separation of concerns. THIS is the selling point of HMVC.