Introduction

**Abstract**

The**“Pizza Ordering System (POS)”**objective is to provide a system which manages the sales activity in a pizza shop for each day and its calculation which is very huge. The users will consume less amount of time when compared to manual paper work through the automated system.

Pizza Ordering System project is a desktop application developed using VB.Net Language. The Project is based on a concept to maintain all the pizza orders for delivery of a pizza house. The system will take care of all the sales servicing activity in a quick manner.  Data storing is easier. It will be able to check any report at any time. Paper work and manual work is reduced. The system is user friendly and easy to use. At the end of the day report is generated to calculate the payment for each user in each day.

**Project Module**

**Modules in this projects are**

**Admin Module :** This module provides administrator related functionality. Administrator manages all information and has access rights to add, delete, edit and view the data related to the application like sales, orders, staffs etc..

**Staff Details :** The admin will provide a username and password to the staff where he used to login with the provided details. The user has the limited permission he can only take orders, billing, customer feedback and registration of client details.

**Orders :**  This module contains the order details placed by the clients to the staffs and the details are centralized stored for billing and reporting purpose

**Billing :** Once the order is placed and over the billing details are logged by the user.

**Clients Details** : Client details are stored in this module. If there is any new offer it will be informed to the regular clients.

**Purchase** : This module contains the details of the purchase items which needs for the kitchen.

**Pizza Master :** This module contains the details of the pizza varieties like veg pizza, Non veg pizza, Burger Pizza etc…

**Reports :** This Module is for the Admin level and it contains the full reporting details.

**SYSTEM REQUIREMENT**

**HARDWARE REQUIREMENT**

|  |  |
| --- | --- |
| **Hardware Specification** | |
| **Processor:** | **Intel(R) Core(TM) i3-4570** |
| **Hard Disk** | **1 TB** |
| **Ram** | **6Gb** |

**SOFTWARE REQUIREMENT**

|  |  |
| --- | --- |
| **Software Specification** | |
| Text Editor | **Notepad ++** |
| **Front End** | **Html, Css, Bootstrap, Javascript** |
| **Back End** | **PHP** |
| **Database** | **MySql** |
| **Server** | **Wamp Or Xamp Server** |
| **Operating System** | **Windows 10 or Higher** |

**Software Descriptions**

**About PHP :**

PHP: Hypertext Preprocessor (the name is a recursive acronym) is a widely used, general-purpose scripting language that was originally designed for web development to produce dynamic web pages. For this purpose, PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document. As a general-purpose programming language, PHP code is processed by an interpreter application in command-line mode performing desired operating system operations and producing program output on its standard output channel. It may also function as a graphical application. PHP is available as a processor for most modern ib servers and as standalone interpreter on most operating systems and computing platforms. PHP was originally created by Rasmus Lerdorf in 1995 and has been in continuous development ever since. The main implementation of PHP is now produced by The PHP Group and serves as the de facto standard for PHP as there is no formal specification. PHP is free software released under the PHP License, which is incompatible with the GNU General Public License (GPL) because restrictions exist regarding the use of the term PHP.

Hypertext refers to files linked together using hyperlinks, such as HTML (Hypertext Markup Language) files. Preprocessing is executing instructions that modify the output. Below is a demonstration of the difference between HTML and PHP files.

**Accessing an HTML Page**

* Your browser sends a request to that web page's server (computer) for the file (HTML or image) you wish to view.
* The web server (computer) sends the file requested back to your computer.
* Your browser displays the file appropriately.
* If you request a PHP file (ends with ".php"), the server handles it differently.

**Accessing a PHP Page**

* Your browser sends a request to that web page's server for the PHP file you wish to view.
* The web server calls PHP to interpret and perform the operations called for in the PHP script.
* The web server sends the output of the PHP program back to your computer.
* Your browser displays the output appropriately.

**Benefit of PHP**

Because the server does processing, the output of PHP files changes when its input changes. For example, most of the pages on the Horticulture site have only two (2) PHP commands:

* Include the header file that defines the links on the left, the banner, and the quick links at the top.
* Include the footer file that displays the mission statement and Horticulture contact information.

Because including the files is performed every time the PHP file is accessed, when the header/footer files change, the new content will be immediately updated. In other words, if you add a new link, every page that includes the header will immediately display the new link.

**Security**

About 30% of all vulnerabilities listed on the National Vulnerability Database are linked to PHP. These vulnerabilities are caused mostly by not following best practice programming rules: technical security flaws of the language itself or of its core libraries are not frequent (23 in 2008, about 1% of the total). Recognizing that programmers make mistakes, some languages include taint checking to detect automatically the lack of input validation which induces many issues. Such a feature is being developed for PHP, but its inclusion in a release has been rejected several times in the past. There are advanced protection patches such as Suhosin and Hardening-Patch, especially designed for Web hosting environments.

**Syntax**

The PHP interpreter only executes PHP code within its delimiters. Anything outside its delimiters is not processed by PHP (although non-PHP text is still subject to control structures described in PHP code). The most common delimiters are <?php to open and ?> to close PHP sections. <script language="php"> and </script> delimiters are also available, as are the shortened forms <?or<?= (which is used to echo back a string or variable) and ?> as well as ASP-style short forms <% or <%= and %>. While short delimiters are used, they make script files less portable as support for them can be disabled in the PHP configuration, and so they are discouraged. The purpose of all these delimiters is to separate PHP code from non-PHP code, including HTML.

The first form of delimiters, <?php and ?>, in XHTML and other XML documents, creates correctly formed XML 'processing instructions'. This means that the resulting mixture of PHP code and other markup in the server-side file is itself well-formed XML.

Variables are prefixed with a dollar symbol, and a type does not need to be specified in advance. Unlike function and class names, variable names are case sensitive. Both double-quoted ("") and here-doc strings provide the ability to interpolate a variable's value into the string. PHP treats newlines as whitespace in the manner of a free-form language (except when inside string quotes), and statements are terminated by a semicolon. PHP has three types of comment syntax: /\* \*/ marks block and inline comments; // as well as # are used for one-line comments. The echo statement is one of several facilities PHP provides to output text, e.g., to a Web browser.

In terms of keywords and language syntax, PHP is similar to most high level languages that follow the C style syntax. if conditions, for and while loops, and function returns are similar in syntax to languages such as C, C++, Java and Perl.

**Data types**

PHP stores whole numbers in a platform-dependent range, either a 64-bit or 32-bit signed integer equivalent to the C-language long type. Unsigned integers are converted to signed values in certain situations; this behavior is different from other programming languages. Integer variables can be assigned using decimal (positive and negative), octal, and hexadecimal notations. Floating point numbers are also stored in a platform-specific range. They can be specified using floating point notation, or two forms of scientific notation. PHP has a native Boolean type that is similar to the native Boolean types in Java and C++. Using the Boolean type conversion rules, non-zero values are interpreted as true and zero as false, as in Perl and C++. The null data type represents a variable that has no value. The only value in the null data type is NULL. Variables of the "resource" type represent references to resources from external sources. These are typically created by functions from a particular extension, and can only be processed by functions from the same extension; examples include file, image, and database resources. Arrays can contain elements of any type that PHP can handle, including resources, objects, and even other arrays. Order is preserved in lists of values and in hashes with both keys and values, and the two can be intermingled. PHP also supports strings, which can be used with single quotes, double quotes, nowdoc or heredoc syntax.

**MY SQL**

MySQL is the world's most used open source relational database management system (RDBMS) as of 2008 that run as a server providing multi-user access to a number of databases. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack—LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL.

For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, Joomla, Word Press, phpBB, MyBB, Drupal and other software built on the LAMP software stack. MySQL is also used in many high-profile, large-scale World Wide Web products, including Wikipedia, Google(though not for searches), ImagebookTwitter, Flickr, Nokia.com, and YouTube.

**Interimages**

MySQL is primarily an RDBMS and ships with no GUI tools to administer MySQL databases or manage data contained within the databases. Users may use the included command line tools, or use MySQL "front-ends", desktop software and web applications that create and manage MySQL databases, build database structures, back up data, inspect status, and work with data records. The official set of MySQL front-end tools, MySQL Workbench is actively developed by Oracle, and is freely available for use.

**Features**

As of April 2009, MySQL offered MySQL 5.1 in two different variants: the open source

MySQL Community Server and the commercial Enterprise Server. MySQL 5.5 is offered

under the same licenses. They have a common code base and include the following features:

 A broad subset of ANSI SQL 99, as well as extensions

 Cross-platform support

 Stored procedures

 Triggers

 Cursors

 Updatable Views

 Information schema

 Strict mode (ensures MySQL does not truncate or otherwise modify data to conform

to an underlying data type, when an incompatible value is inserted into that type)

 X/Open XAdistributed transaction processing (DTP) support; two phase commit as

part of this, using Oracle's InnoDB engine

 Independent storage engines (MyISAM for read speed, InnoDB for transactions and

referential integrity, MySQL Archive for storing historical data in little space)

 Transactions with the InnoDB, and Cluster storage engines; savepoints with InnoDB

 SSL support

 Query caching

 Sub-SELECTs (i.e. nested SELECTs)

 Replication support (i.e. Master-Master Replication & Master-Slave Replication) with

one master per slave, many slaves per master, no automatic support for multiple masters per slave.

 Full-text indexing and searching using MyISAM engine

 Embedded database library

 Unicode support (however prior to 5.5.3 UTF-8 and UCS-2 encoded strings are

limited to the BMP, in 5.5.3 and later use utf8mb4 for full Unicode support)

 ACID compliance when using transaction capable storage engines (InnoDB and

Cluster)

 Partitioned tables with pruning of partitions in optimizer

 Shared-nothing clustering through MySQL Cluster

 Hot backup (via mysqlhotcopy) under certain conditions

Multiple storage engines, allowing one to choose the one that is most effective for each table in the application (in MySQL 5.0, storage engines must be compiled in; in MySQL 5.1, storage engines can be dynamically loaded at run time): Native storage engines (MyISAM, Falcon, Merge, Memory (heap), Federated, Archive, CSV, Blackhole, Cluster, EXAMPLE, Maria, and InnoDB, which was made the default as of 5.5). Partner-developed storage engines (solidDB, NitroEDB, ScaleDB, TokuDB, Infobright (formerly Brighthouse), Kickfire, XtraDB, IBM DB2). InnoDB used to be a partner-developed storage engine, but with recent acquisitions, Oracle now owns both MySQL core and InnoDB.

**SYSTEM DESIGN**

**SYSTEM DESIGN**

**DEFINITION**

The most creative and challenging face of the system development is System Design. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Design goes through the logical and physical stages of development.

In designing a new system, the system analyst must have a clear understanding of the objectives, which the design is aiming to fulfill. The first step is to determine how the output is to be produced and in what format. Second, input data and master files have to be designed to meet the requirements of the proposed output. The operational phases are handled through program construction and testing.

Design of a system can be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Thus system design is a solution to “how to” approach to the creation of a new system. Thus important phase provides the understanding and the procedural details necessary for implementing the system recommended in the feasibility study. The design step provides a data design, architectural design, and a procedural design.

**Input Design**

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

Therefore, the quality of system input determines the quality of system output. Welldesigned input forms and screens have following properties −

* It should serve specific purpose effectively such as storing, recording, and retrieving the information.
* It ensures proper completion with accuracy.
* It should be easy to fill and straightforward.
* It should focus on user’s attention, consistency, and simplicity.
* All these objectives are obtained using the knowledge of basic design principles regarding −
  + What are the inputs needed for the system?
  + How end users respond to different elements of forms and screens.

**Objectives for Input Design**

The objectives of input design are −

* To design data entry and input procedures
* To reduce input volume
* To design source documents for data capture or devise other data capture methods
* To design input data records, data entry screens, user interface screens, etc.
* To use validation checks and develop effective input controls.

**Data Input Methods**

It is important to design appropriate data input methods to prevent errors while entering data. These methods depend on whether the data is entered by customers in forms manually and later entered by data entry operators, or data is directly entered by users on the PCs.

A system should prevent user from making mistakes by −

* Clear form design by leaving enough space for writing legibly.
* Clear instructions to fill form.
* Clear form design.
* Reducing key strokes.
* Immediate error feedback.

Some of the popular data input methods are −

* Batch input method (Offline data input method)
* Online data input method
* Computer readable forms
* Interactive data input

**Input Integrity Controls**

Input integrity controls include a number of methods to eliminate common input errors by end-users. They also include checks on the value of individual fields; both for format and the completeness of all inputs.

Audit trails for data entry and other system operations are created using transaction logs which gives a record of all changes introduced in the database to provide security and means of recovery in case of any failure.

**Output Design**

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

**Objectives of Output Design**

The objectives of input design are −

* To develop output design that serves the intended purpose and eliminates the production of unwanted output.
* To develop the output design that meets the end users requirements.
* To deliver the appropriate quantity of output.
* To form the output in appropriate format and direct it to the right person.
* To make the output available on time for making good decisions.

Let us now go through various types of outputs −

**External Outputs**

Manufacturers create and design external outputs for printers. External outputs enable the system to leave the trigger actions on the part of their recipients or confirm actions to their recipients.

Some of the external outputs are designed as turnaround outputs, which are implemented as a form and re-enter the system as an input.

**Internal outputs**

Internal outputs are present inside the system, and used by end-users and managers. They support the management in decision making and reporting.

There are three types of reports produced by management information −

* **Detailed Reports** − They contain present information which has almost no filtering or restriction generated to assist management planning and control.
* **Summary Reports** − They contain trends and potential problems which are categorized and summarized that are generated for managers who do not want details.
* **Exception Reports** − They contain exceptions, filtered data to some condition or standard before presenting it to the manager, as information.

**Output Integrity Controls**

Output integrity controls include routing codes to identify the receiving system, and verification messages to confirm successful receipt of messages that are handled by network protocol.

Printed or screen-format reports should include a date/time for report printing and the data. Multipage reports contain report title or description, and pagination. Pre-printed forms usually include a version number and effective date.

## Forms Design

Both forms and reports are the product of input and output design and are business document consisting of specified data. The main difference is that forms provide fields for data input but reports are purely used for reading. For example, order forms, employment and credit application, etc.

* During form designing, the designers should know −
  + who will use them
  + where would they be delivered
  + the purpose of the form or report
* During form design, automated design tools enhance the developer’s ability to prototype forms and reports and present them to end users for evaluation.

### Objectives of Good Form Design

A good form design is necessary to ensure the following −

* To keep the screen simple by giving proper sequence, information, and clear captions.
* To meet the intended purpose by using appropriate forms.
* To ensure the completion of form with accuracy.
* To keep the forms attractive by using icons, inverse video, or blinking cursors etc.
* To facilitate navigation.

### Types of Forms

**Flat Forms**

* It is a single copy form prepared manually or by a machine and printed on a paper. For additional copies of the original, carbon papers are inserted between copies.
* It is a simplest and inexpensive form to design, print, and reproduce, which uses less volume.

**Unit Set/Snap out Forms**

* These are papers with one-time carbons interleaved into unit sets for either handwritten or machine use.
* Carbons may be either blue or black, standard grade medium intensity. Generally, blue carbons are best for handwritten forms while black carbons are best for machine use.

**Continuous strip/Fanfold Forms**

* These are multiple unit forms joined in a continuous strip with perforations between each pair of forms.
* It is a less expensive method for large volume use.

**No Carbon Required (NCR) Paper**

* They use carbonless papers which have two chemical coatings (capsules), one on the face and the other on the back of a sheet of paper.
* When pressure is applied, the two capsules interact and create an image.

## Software Detailed Design

A software module is the lowest level of design granularity in the system. Depending on the software development approach, there may be one or more modules per system. This section should provide enough detailed information about logic and data necessary to completely write source code for all modules in the system (and/or integrate COTS software programs).

If there are many modules or if the module documentation is extensive, place it in an appendix or reference a separate document. Add additional diagrams and information, if necessary, to describe each module, its functionality, and its hierarchy. Industry-standard module specification practices should be followed. Include the following information in the detailed module designs:

* A narrative description of each module, its function(s), the conditions under which it is used (called or scheduled for execution), its overall processing, logic, interfaces to other modules, interfaces to external systems, security requirements, etc.; explain any algorithms used by the module in detail
* For COTS packages, specify any call routines or bridging programs to integrate the package with the system and/or other COTS packages (for example, Dynamic Link Libraries)
* Data elements, record structures, and file structures associated with module input and output
* Graphical representation of the module processing, logic, flow of control, and algorithms, using an accepted diagramming approach (for example, structure charts, action diagrams, flowcharts, etc.)
* Data entry and data output graphics; define or reference associated data elements; if the project is large and complex or if the detailed module designs will be incorporated into a separate document, then it may be appropriate to repeat the screen information in this section
* Report layout

# FILE Design AND DATABASE DESIGN

Interact with the Database Administrator (DBA) when preparing this section. The section should reveal the final design of all database management system (DBMS) files and the non-DBMS files associated with the system under development. Additional information may add as required for the particular project. Provide a comprehensive data dictionary showing data element name, type, length, source, validation rules, maintenance (create, read, update, delete (CRUD) capability), data stores, outputs, aliases, and description. Can be included as an appendix.

## Database Management System Files

This section reveals the final design of the DBMS files and includes the following information, as appropriate (refer to the data dictionary):

* Refined logical model; provide normalized table layouts, entity relationship diagrams, and other logical design information
* A physical description of the DBMS schemas, sub-schemas, records, sets, tables, storage page sizes, etc.
* Access methods (such as indexed, via set, sequential, random access, sorted pointer array, etc.)
* Estimate of the DBMS file size or volume of data within the file, and data pages, including overhead resulting from access methods and free space
* Definition of the update frequency of the database tables, views, files, areas, records, sets, and data pages; estimate the number of transactions if the database is an online transaction-based system

## Non-Database Management System Files

In this section, provide the detailed description of all non-DBMS files and include a narrative description of the usage of each file—including if the file is used for input, output, or both; if this file is a temporary file; an indication of which modules read and write the file, etc.; and file structures (refer to the data dictionary). As appropriate, the file structure information should:

* Identify record structures, record keys or indexes, and reference data elements within the records
* Define record length (fixed or maximum variable length) and blocking factors
* Define file access method—for example, index sequential, virtual sequential, random access, etc.
* Estimate the file size or volume of data within the file, including overhead resulting from file access methods
* Define the update frequency of the file; if the file is part of an online transaction-based system, provide the estimated number of transactions per unit time, and the statistical mean, mode, and distribution of those transactions

**SYSTEM TESTING AND VALIDATION**

**SYSTEM TESTING**

System testing is defined as testing of a complete and fully integrated software product. This testing falls in black-box testing wherein knowledge of the inner design of the code is not a pre-requisite and is done by the testing team. It is the final test to verify that the product to be delivered meets the specifications mentioned in the requirement document. It should investigate both functional and non-functional requirements.

**UNIT TETSING**

The first test in the development process is the unit test. The source code is normally divided into modules, which in turn are divided into smaller units called units. These units have specific behavior. The test done on these units of code is called unit test. Unit test depends upon the language on which the project is developed. Unit tests ensure that each unique path of the project performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**INTEGRATION TESTING**

In integration testing modules are combined and tested as a group. Modules are typically code modules, individual applications, source and destination applications on a network, etc. Integration Testing follows unit testing and precedes system testing. Testing after the product is code complete. Betas are often widely distributed or even distributed to the public at large in hopes that they will buy the final product when it is released.

**VALIDATION**

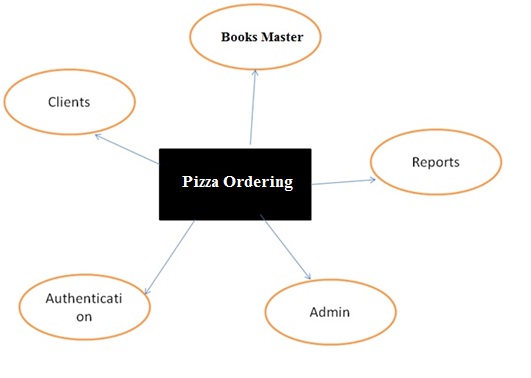
Validation is the process of evaluating the final product to check whether the software meets the customer expectations and requirements. It is a dynamic mechanism of validating and testing the actual product. Validation is determining if the system complies with the requirements and performs functions for which it is intended and meets the organization’s goals and user needs. Validation helps in building the right product as per the customer’s requirement and helps in satisfying their needs.

**DFD Diagram**

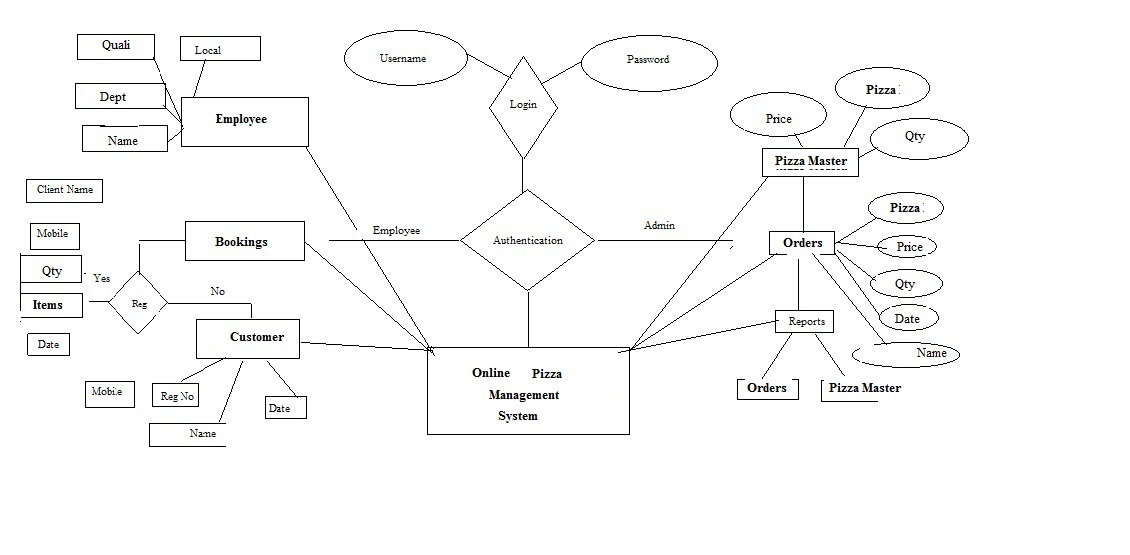
**Level 1**



**Level 2**



**ER Diagram**

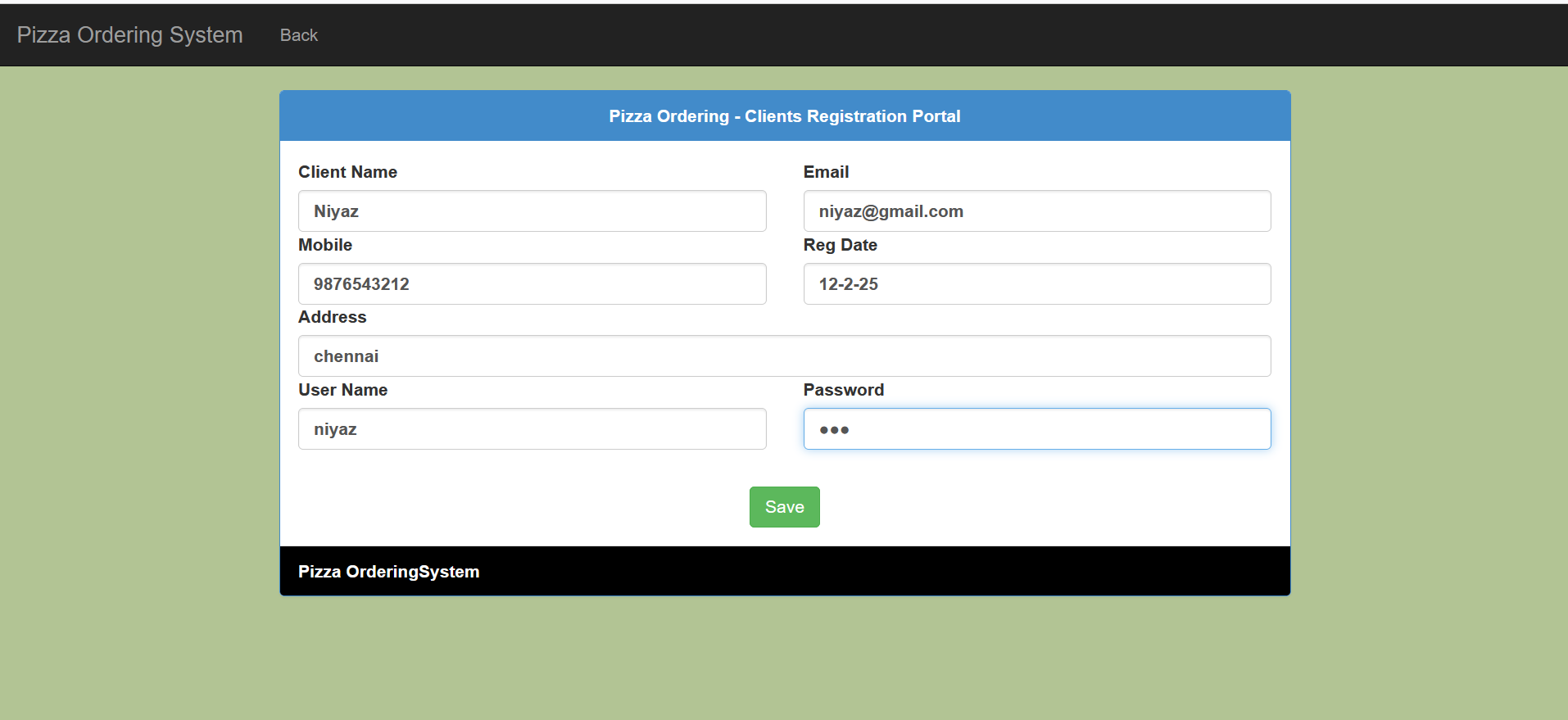


**Form Layout**

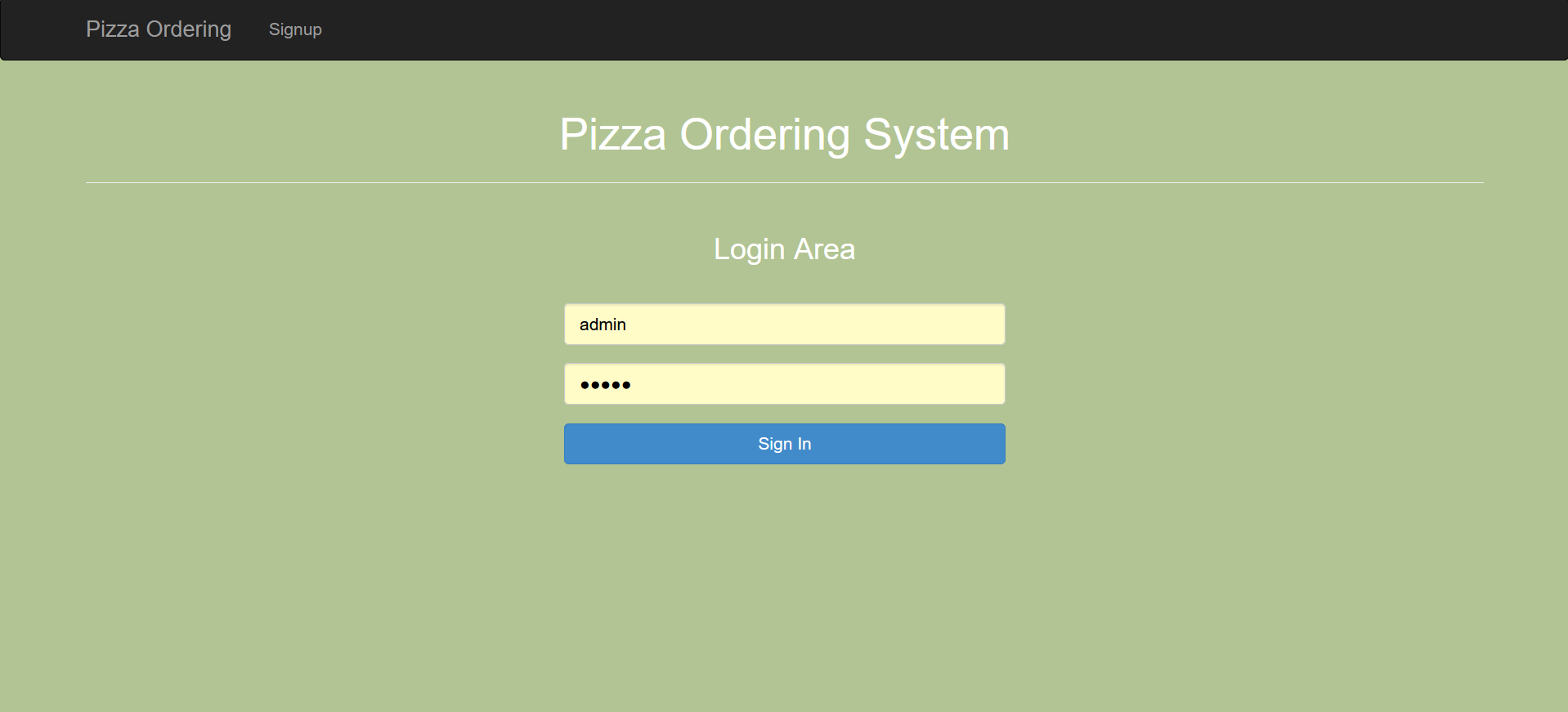
**Index Page**

****

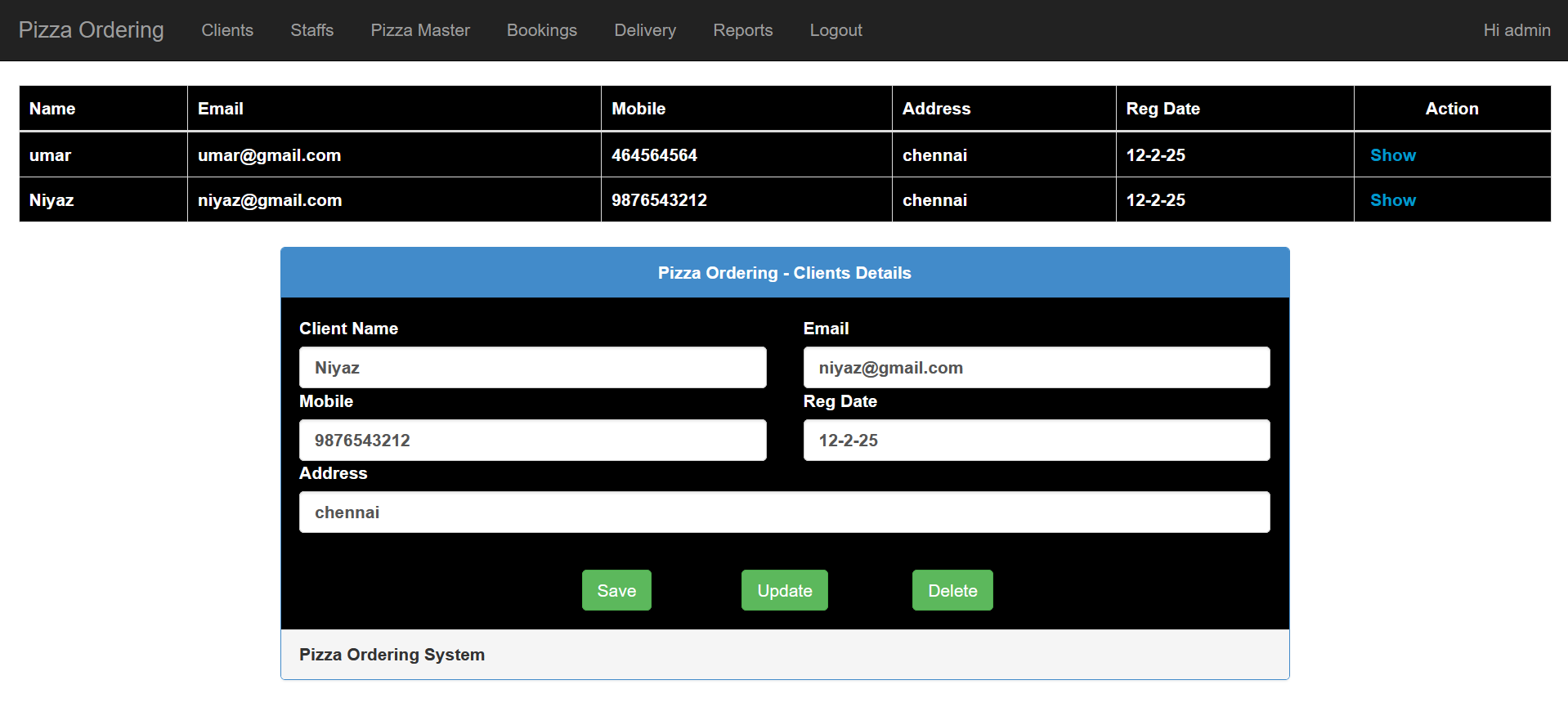
**Signup**

**\***

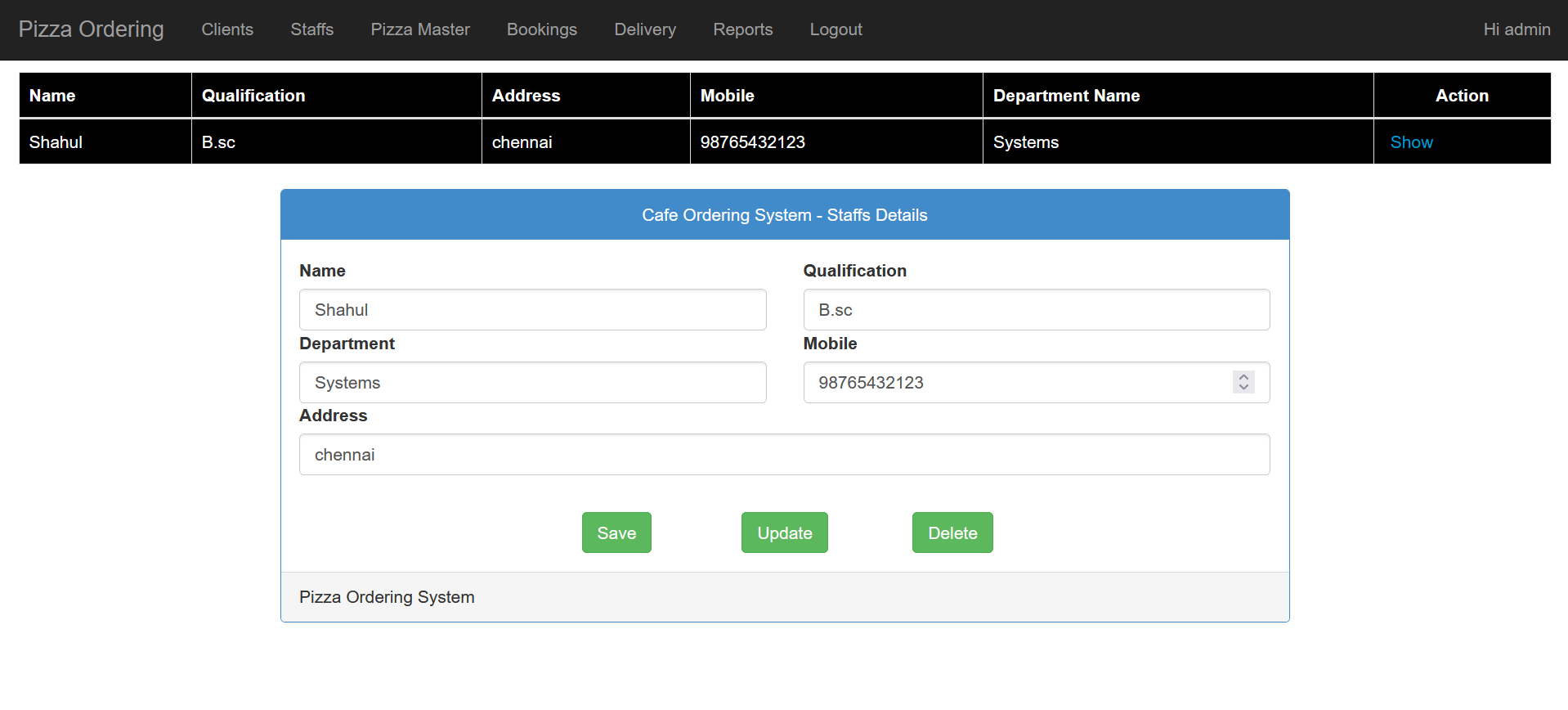
**Login**

****

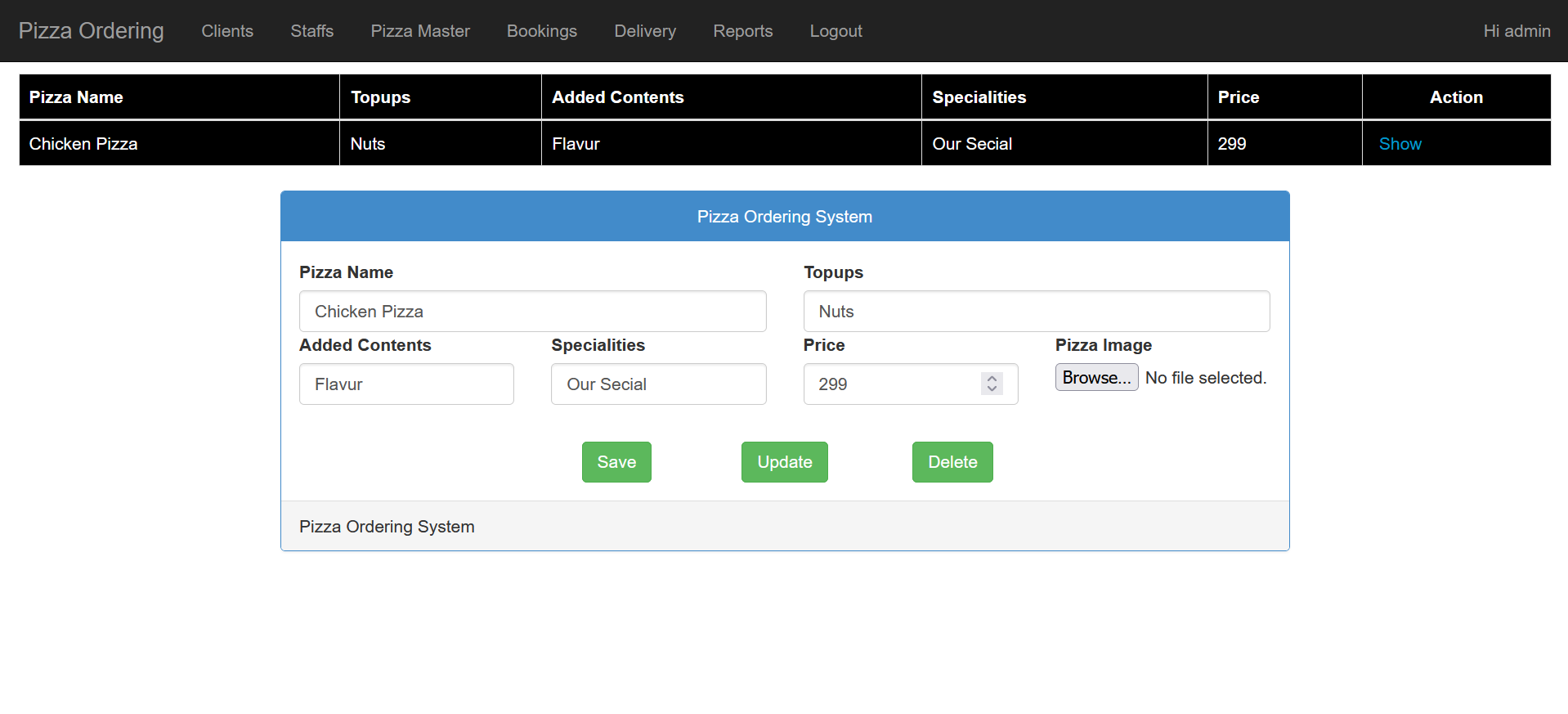
**Clients**

****

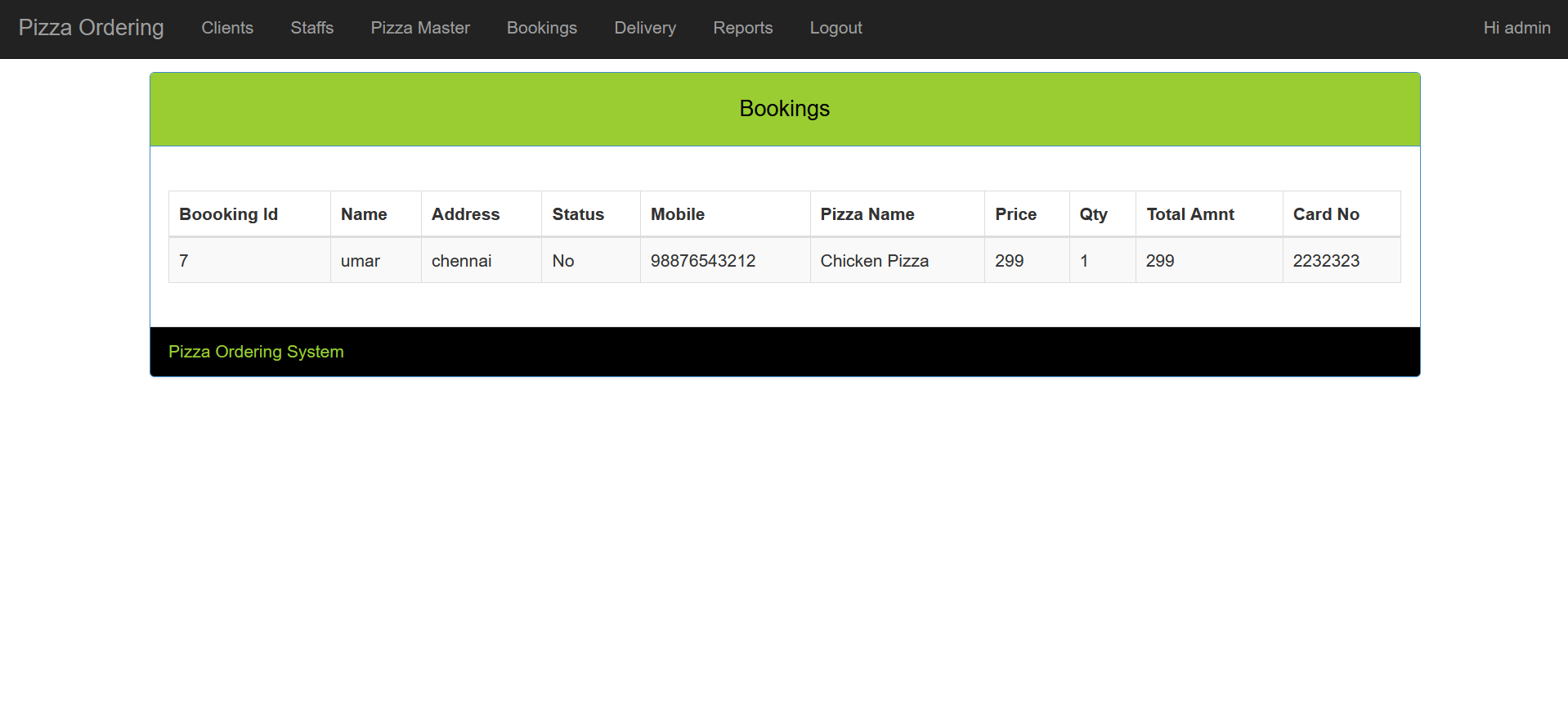
**Staffs Info**

****

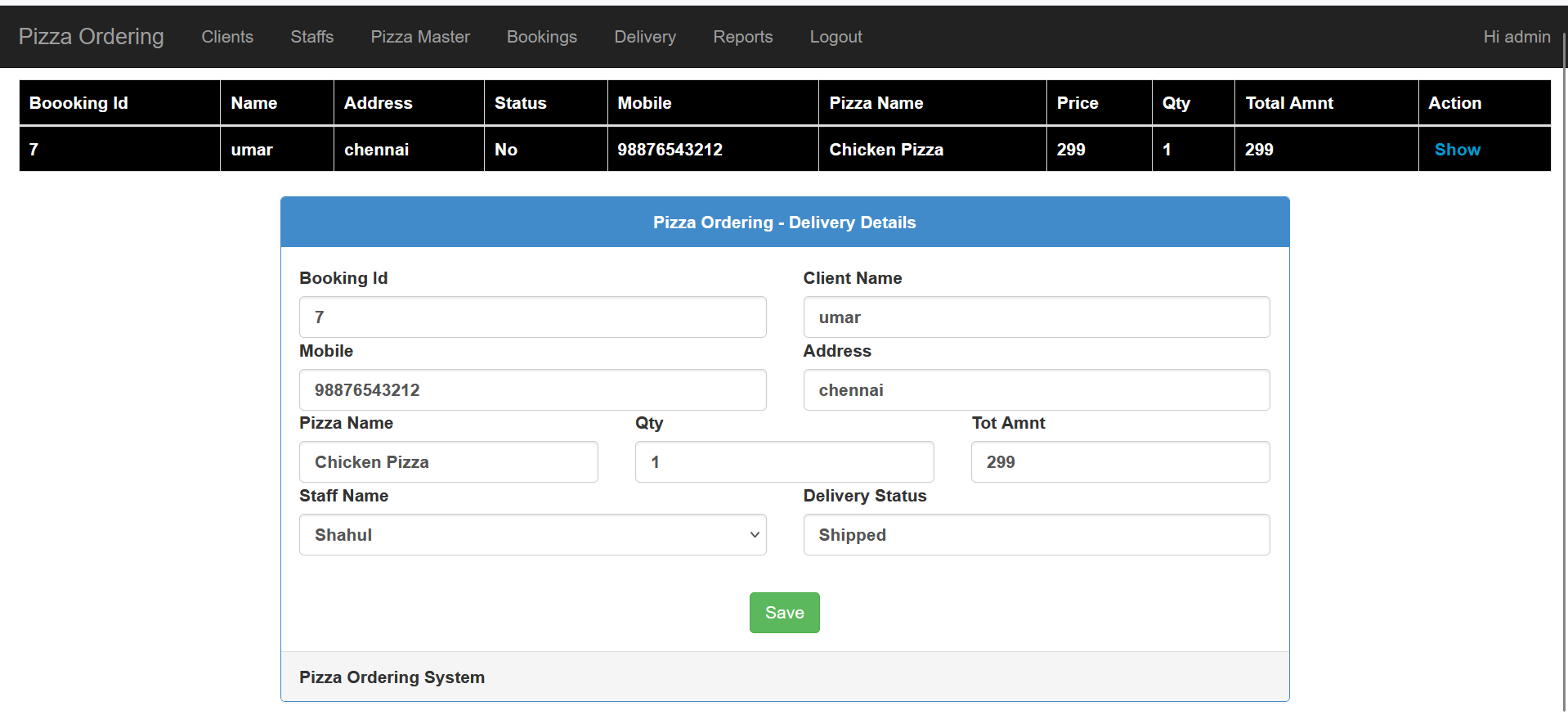
**Pizza Master**

****

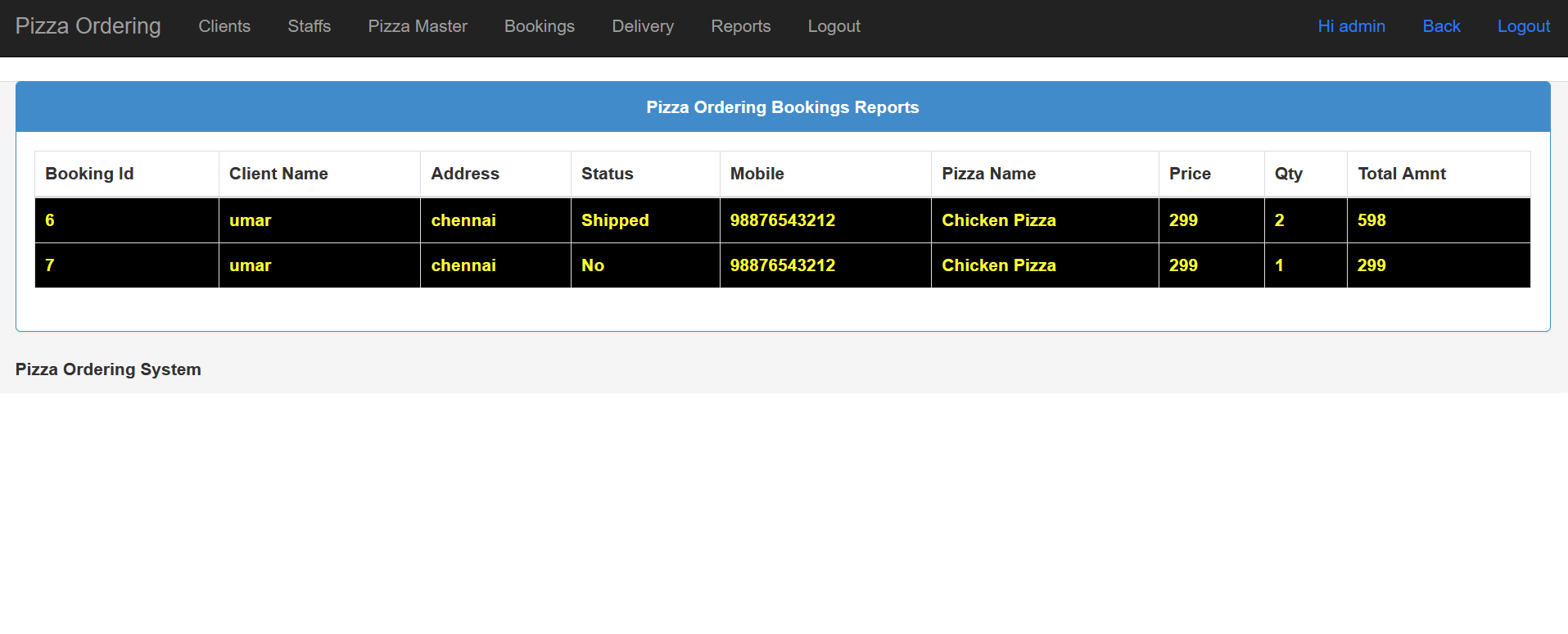
**Bookings Info**

****

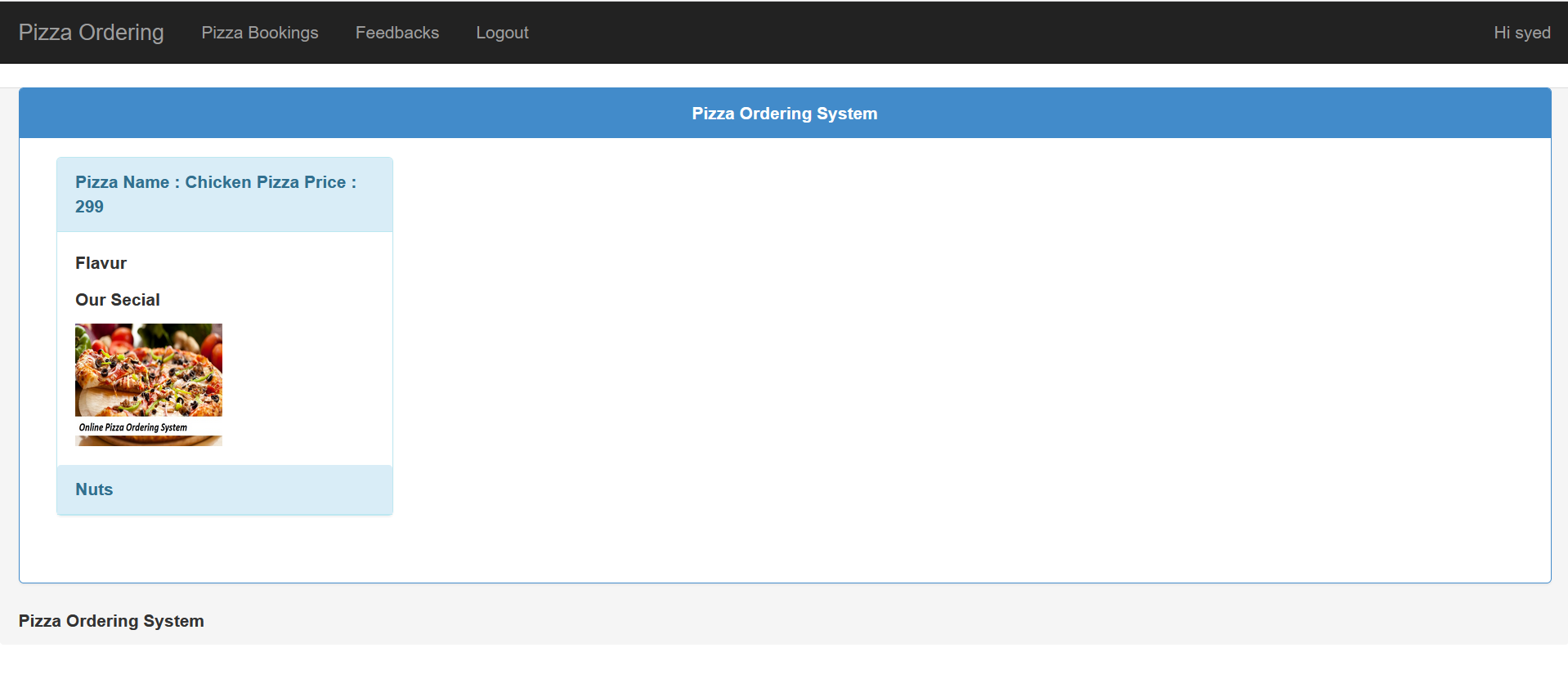
**Delivery**

****

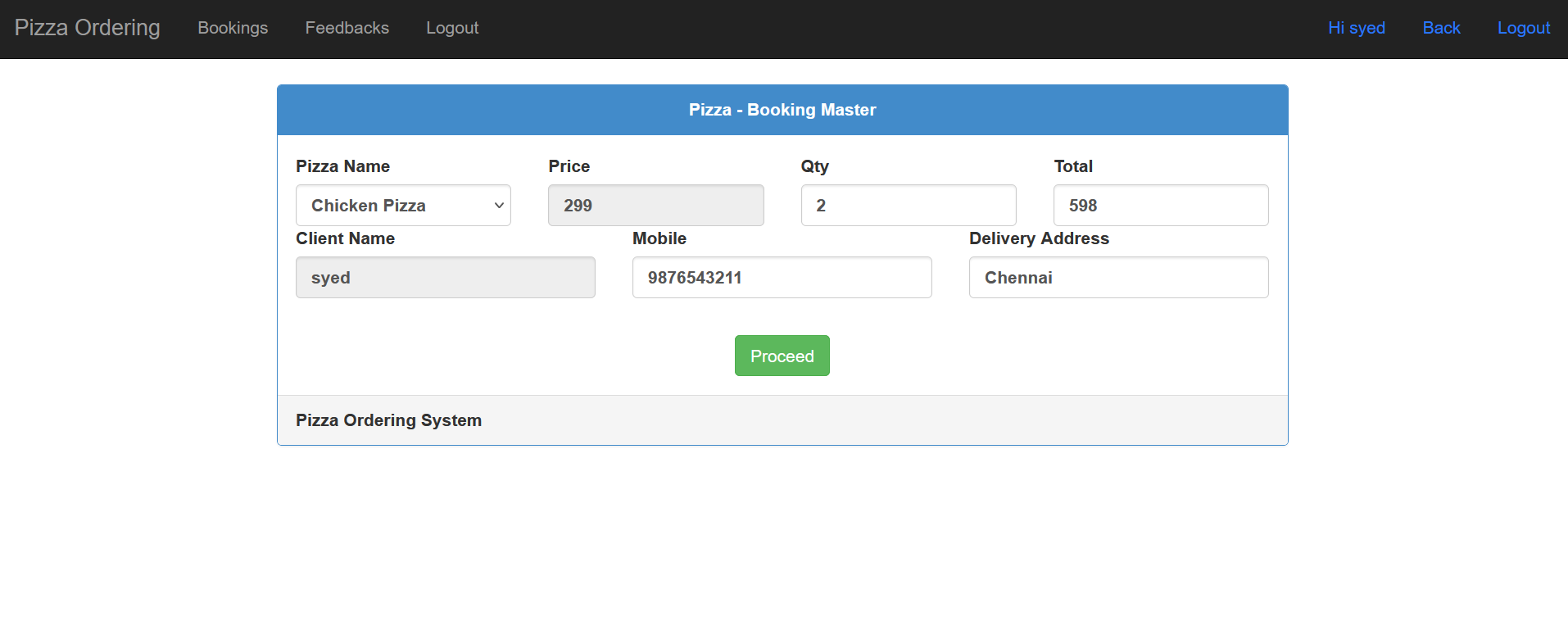
**Reports**

****

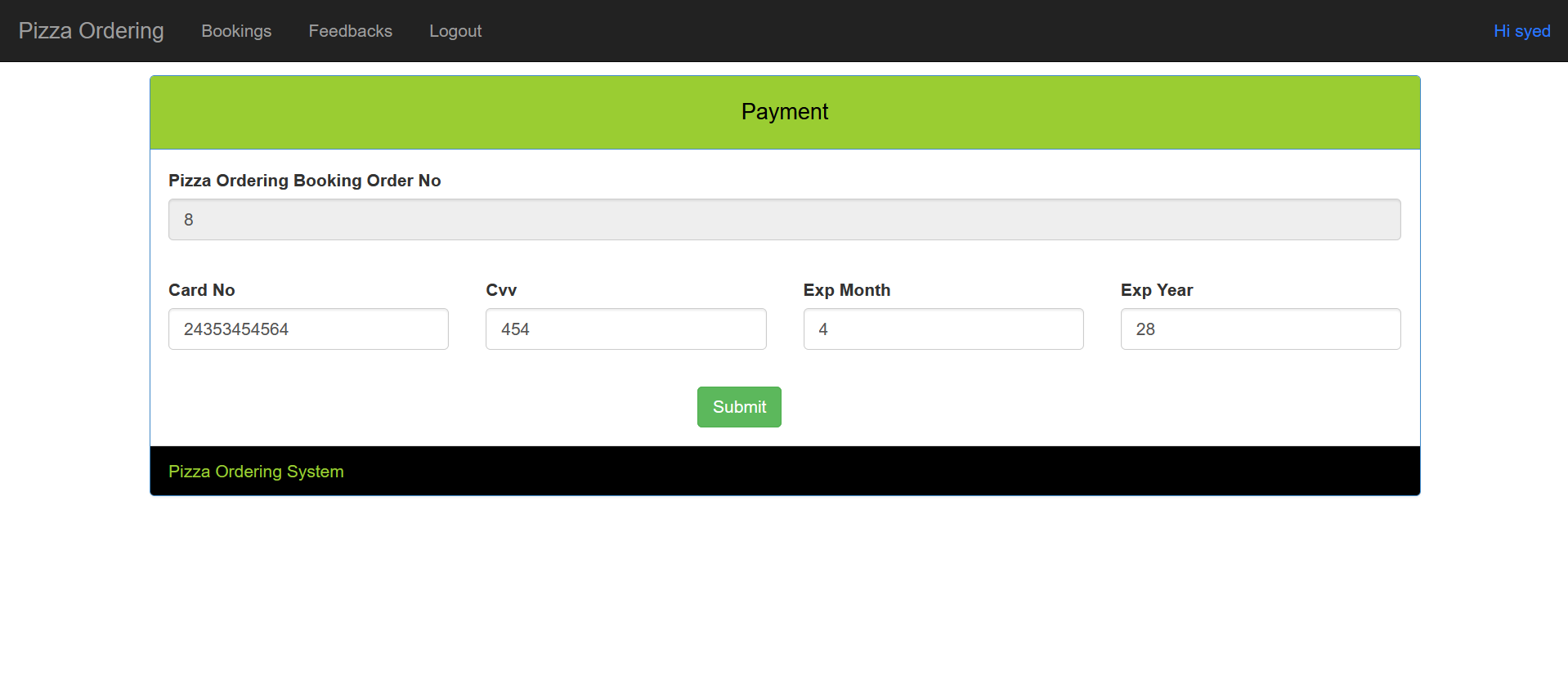
**Client Home Page**

****

**Client Bookings**

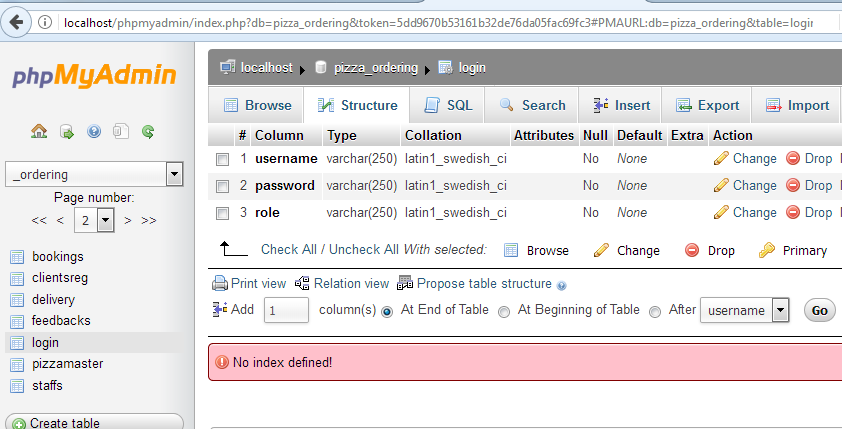
****

**Payments**

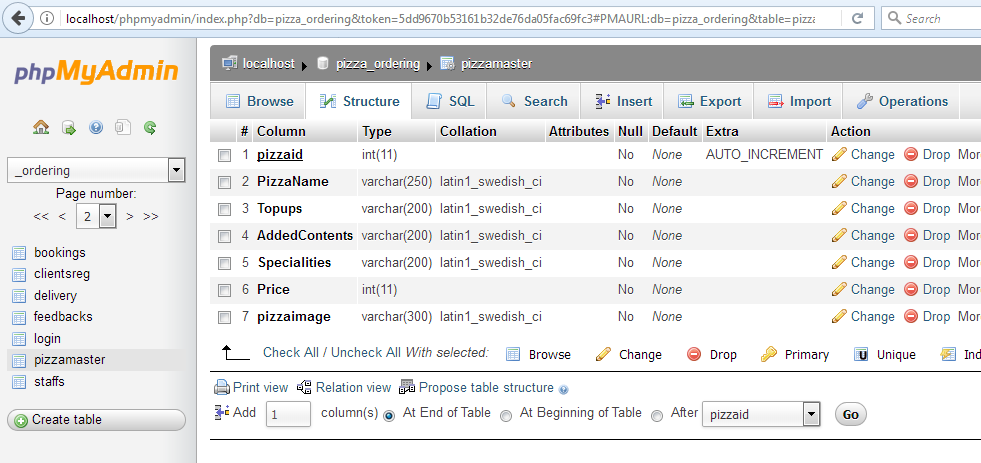
****

**Table Structure**

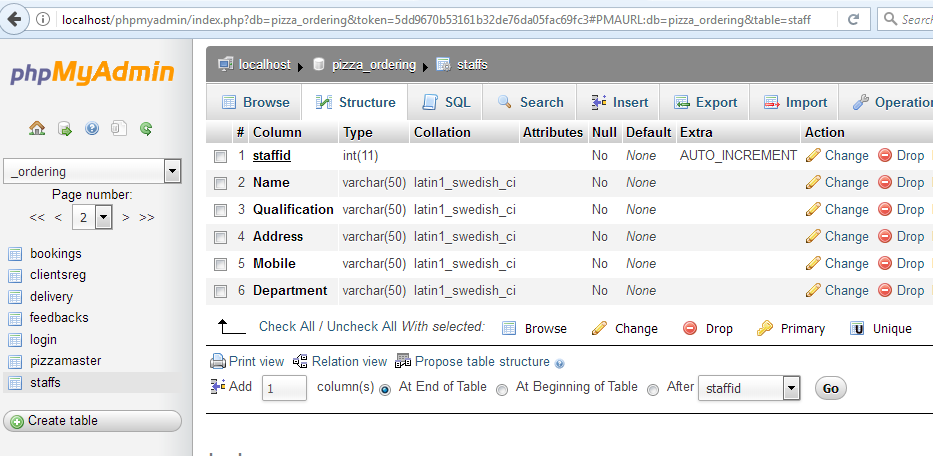
Pizza Ordering Database Design

Login Table

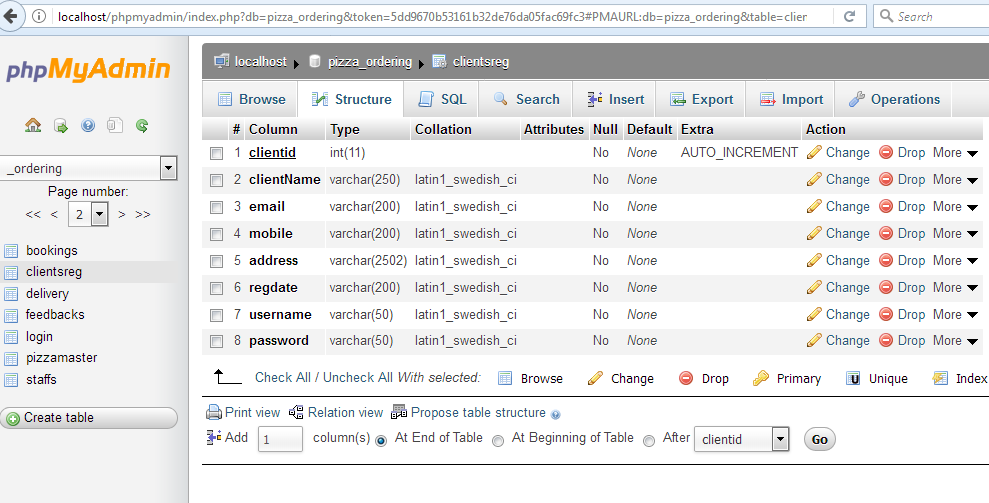
Pizza Master



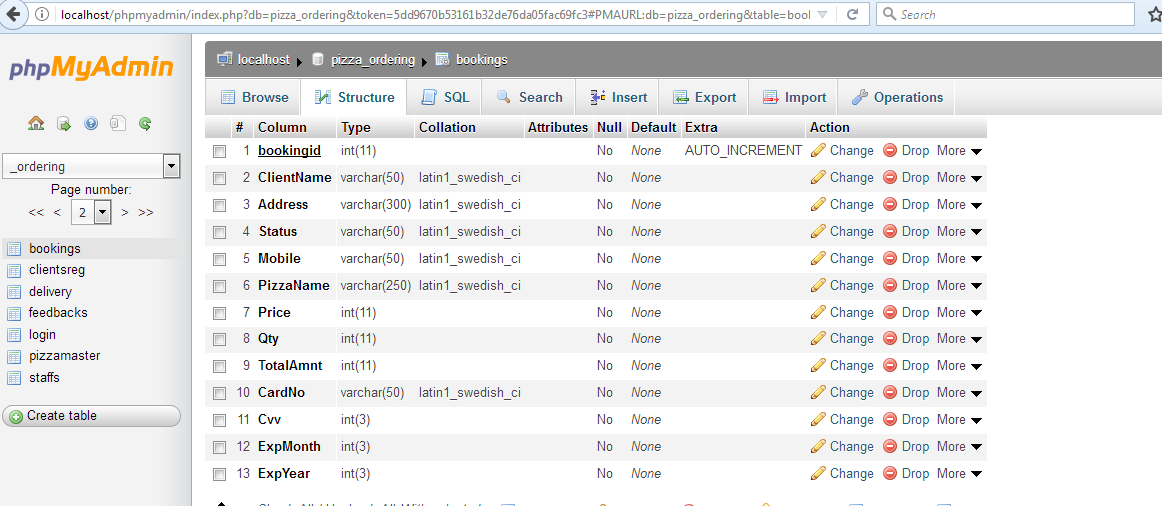
Staffs



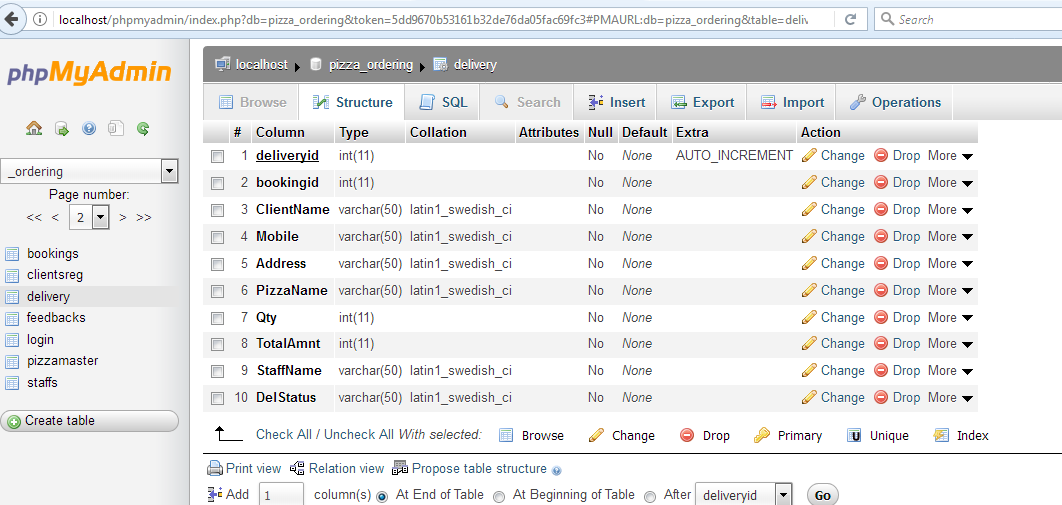
Clients Reg



Bookings



Delivery



**Source Code**

**Index Page**

<?php

if(isset($\_POST["btnsignup1"]))

{

header("location:signup.php");

}

?>

<!DOCTYPE html>

<html>

<head>

<title>Pizza Ordering System</title>

<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7; IE=EmulateIE9">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1.0, user-scalable=no"/>

<link rel="stylesheet" type="text/css" href="css/bootstrap.css" />

</head>

<body style="background-image:url(images/1.jpg);color:#fff;">

<nav class="navbar navbar-inverse" role="navigation" >

<div class="container">

<!-- Brand and toggle get grouped for better mobile display -->

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="#">Pizza Ordering</a>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

<ul class="nav navbar-nav">

<li>

<a href="register.php">Signup</a>

</li>

<li>

<a href="login.php">Sign In</a>

</li>

</ul>

</div>

<!-- /.navbar-collapse -->

</div>

<!-- /.container -->

</nav>

<div class="container">

<div class="row">

<div class="col-lg-12">

<div id="header">

<h1 align="center" style="color:#fff;">Pizza Ordering System</h1>

</div>

<hr>

</div>

<div class="col-lg-12">

<br>

</div>

</div>

</div>

<div class="container">

<div class="row">

<div class="col-md-3"></div>

<div class="col-md-6"></div>

<div class="col-md-3"></div>

</div>

<div class="row">

<div class="col-md-4"></div>

<div class="col-md-4">

<div class="table-responsive">

</div>

</div>

<div class="col-md-4"></div>

<footer>

<div class="row">

<div class="col-lg-12">

<p align="center" style="background-color:#000;color:#fff;">Copyright &copy; Pizza Ordering System</p>

</div>

</div>

<!-- /.row -->

</footer>

<script src="jquery.js"></script>

<!-- Bootstrap Core JavaScript -->

<script src="bootstrap.min.js"></script>

</body>

</html>

Login

<?php

if(isset($\_POST["btnsignup1"]))

{

header("location:signup.php");

}

?>

<!DOCTYPE html>

<html>

<head>

<title>Pizza Ordering System</title>

<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7; IE=EmulateIE9">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1.0, user-scalable=no"/>

<link rel="stylesheet" type="text/css" href="css/bootstrap.css" />

</head>

<body style="background-image:url(images/1.jpg);color:#fff;">

<nav class="navbar navbar-inverse" role="navigation" >

<div class="container">

<!-- Brand and toggle get grouped for better mobile display -->

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="#">Pizza Ordering</a>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

<ul class="nav navbar-nav">

<li>

<a href="register.php">Signup</a>

</li>

</ul>

</div>

<!-- /.navbar-collapse -->

</div>

<!-- /.container -->

</nav>

<div class="container">

<div class="row">

<div class="col-lg-12">

<div id="header">

<h1 align="center" style="color:#fff;">Pizza Ordering System</h1>

</div>

<hr>

</div>

<div class="col-lg-12">

<h3 align="center" style="color:#fff;"> Login Area</h3>

<br>

</div>

</div>

</div>

<div class="container">

<div class="row">

<div class="col-md-3"></div>

<div class="col-md-6"></div>

<div class="col-md-3"></div>

</div>

<div class="row">

<div class="col-md-4"></div>

<div class="col-md-4">

<div class="table-responsive">

<table class="table table-bordered" style="bckground-color:#000;color#fff;">

<form class="col-md-12" action="login1.php" method="post">

<div class="form-group">

<input type="text" class="form-control input-md" id="txtUserName" name="txtUserName" placeholder="Username">

</div>

<div class="form-group">

<input type="password" class="form-control input-md" id="txtPWD" name="txtPWD" placeholder="Password">

</div>

<div class="form-group">

<button class="btn btn-primary btn-md btn-block">Sign In</button>

</div>

</form>

</table>

</div>

</div>

<div class="col-md-4"></div>

<footer>

<div class="row">

<div class="col-lg-12">

<p align="center" style="background-color:#000;color:#fff;">Copyright &copy; Pizza Ordering System</p>

</div>

</div>

<!-- /.row -->

</footer>

<script src="jquery.js"></script>

<!-- Bootstrap Core JavaScript -->

<script src="bootstrap.min.js"></script>

</body>

</html>

Master Admin

<?php

session\_start();

require "Connection.php";

?>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Pizza Ordering System</title>

<link rel="stylesheet" href="css/bootstrap.min.css"/>

<script src="js/jquery-2.2.2.min.js"></script>

<script src="js/bootstrap.min.js"></script>

<script src="js/main.js"></script>

</head>

<body style="background-image:url(images/3.jpg);">

<div class="navbar navbar-inverse navbar-fixed-top">

<div class="container-fluid">

<div class="navbar-header">

<a href="#" class="navbar-brand">Pizza Ordering</a>

</div>

<ul class="nav navbar-nav">

<li>

<a href="clients.php">Clients</a>

</li>

<li>

<a href="staffs\_info.php">Staffs</a>

</li>

<li>

<a href="pizzamaster.php" >Pizza Master</a>

</li>

<li>

<a href="booking\_delivery\_view.php"> Bookings</a>

</li>

<li>

<a href="delivery.php"> Delivery</a>

</li>

<li>

<a href="reports.php">Reports</a>

</li>

<li><a href="logout.php">Logout</a></li>

</ul>

<ul class="nav navbar-nav navbar-right">

<li><a href="#" style='color:#2979ff;'><?php echo "Hi ".$\_SESSION["uname"]; ?></a>

</li>

</ul>

</div>

</div>

</body>

</html>

Master User

<?php include("SessionInclude.php");

require "Connection.php";

?>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Pizza Ordering System</title>

<link rel="stylesheet" href="css/bootstrap.min.css"/>

<script src="js/jquery-2.2.2.min.js"></script>

<script src="js/bootstrap.min.js"></script>

<script src="js/main.js"></script>

</head>

<body style="background-image:url(images/3.jpg);">

<div class="navbar navbar-inverse navbar-fixed-top">

<div class="container-fluid">

<div class="navbar-header">

<a href="#" class="navbar-brand">Pizza Ordering</a>

</div>

<ul class="nav navbar-nav">

<li>

<a href="booking\_master.php">Pizza Bookings</a>

</li>

<li>

<a href="feedback.php">Feedbacks</a>

</li>

<li><a href="logout.php">Logout</a></li>

</ul>

<ul class="nav navbar-nav navbar-right">

<li><a href="#"><?php echo "Hi ".$\_SESSION["uname"]; ?></a>

</li>

</ul>

</div>

</div>

<p><b></p>

<p><br></p>

<div class="container-fluid">

<p><br></p>

<div class="row">

<div class="col-md-12">

<div class="panel panel-primary">

<div class="panel-heading text-center">Pizza Ordering System</div>

<div class="panel-body">

<table class="table table-bordered" style="background-color:#000;color:#fff;">

<thead class="thead-dark">

</thead>

<tbody>

<?php

$results = mysqli\_query($con,"select \* from pizzamaster");

while ($row=mysqli\_fetch\_array($results))

{

$prod\_id=$row["pizzaid"];

$pizzaname=$row["PizzaName"];

$flavor=$row["Topups"];

$contents=$row["AddedContents"];

$special=$row["Specialities"];

$price=$row["Price"];

$pizzaimage=$row["pizzaimage"];

echo "

<div class='col-md-3'>

<div class='panel panel-info'>

<div class='panel-heading'><b>Pizza Name : $pizzaname </b> Price : $price</div>

<div class='panel-body'>

<p>$contents</p>

<p>$special</p>

<img src='uploads/$pizzaimage' style='width:120px; height:100px;float:left;'/>

</div>

<div class='panel-heading'><b>$flavor</b>

</div>

</div>

</div>

";

}

?>

</tbody>

</table>

</div>

</div>

</div>

<div class="panel-footer">Pizza Ordering System</div>

</div>

</div>

</body>

</html>

Pizza Master

<?php

session\_start();

$\_SESSION['msg']="";

require "Connection.php";

$txtname = "";

$txtflavour ="";

$txtcontents="";

$txtspecial="";

$txtprice="";

$id=0;

$edit\_state=false;

If(isset($\_GET['edit']))

{

$id=$\_GET['edit'];

$rec=mysqli\_query($con,"select \* from pizzamaster where pizzaid=$id");

$record=mysqli\_fetch\_array($rec);

$txtname = $record['PizzaName'];

$txtflavour = $record['Topups'];

$txtcontents = $record['AddedContents'];

$txtspecial = $record['Specialities'];

$txtprice = $record['Price'];

$id=$record['pizzaid'];

}

function fill\_Id($con)

{

$sql = "select max(pizzaid)+1 from pizzamaster";

$result=mysqli\_query($con,$sql);

while($data = mysqli\_fetch\_row($result))

{

echo $data[0];

}

}

if(isset($\_POST['btnsave']))

{

$file = rand(1000,100000)."-".$\_FILES['file']['name'];

$file\_loc = $\_FILES['file']['tmp\_name'];

$file\_size = $\_FILES['file']['size'];

$file\_type = $\_FILES['file']['type'];

$folder="uploads/";

// new file size in KB

$new\_size = $file\_size/1024;

// new file size in KB

// make file name in lower case

$new\_file\_name = strtolower($file);

// make file name in lower case

$final\_file=str\_replace(' ','-',$new\_file\_name);

$txtname = $\_POST['txtname'];

$txtflavour = $\_POST['txtflavour'];

$txtcontents = $\_POST['txtcontents'];

$txtspecial = $\_POST['txtspecial'];

$txtprice = $\_POST['txtprice'];

if(move\_uploaded\_file($file\_loc,$folder.$final\_file))

{

$sql1 = "insert into pizzamaster set

PizzaName = '$txtname',

Topups = '$txtflavour',

AddedContents = '$txtcontents',

Specialities = '$txtspecial',

Price = '$txtprice',

pizzaimage = '$file'";

if(mysqli\_query($con,$sql1))

{

echo "

<div class='alert alert-success'>

<p><br></p><p><br></p>

<a href='#' class='close' data-dismiss='alert' aria-label='close'>&times;</a>

<b>Pizza Addded Successfully</b>

</div>

";

}

else

{

echo "

<div class='alert alert-success'>

<p><br></p><p><br></p>

<a href='#' class='close' data-dismiss='alert' aria-label='close'>&times;</a>

<b>Failed To Add</b>

</div>

";

}

}

}

//Update Record

if(isset($\_POST['btnedit']))

{

$txtname = $\_POST['txtname'];

$txtflavour = $\_POST['txtflavour'];

$txtcontents = $\_POST['txtcontents'];

$txtspecial = $\_POST['txtspecial'];

$txtprice = $\_POST['txtprice'];

$id=$\_POST['id'];

mysqli\_query($con,"update pizzamaster set PizzaName='$txtname',Topups='$txtflavour',AddedContents='$txtcontents',Specialities='$txtspecial',Price='$txtprice' where pizzaid='$id'");

header("location: pizzamaster.php");

}

//Delete Record

if(isset($\_POST['btndelete']))

{

$id=$\_POST['id'];

mysqli\_query($con,"delete from pizzamaster where pizzaid='$id'");

header("location: pizzamaster.php");

}

// Show Records

$results = mysqli\_query($con,"select \* from pizzamaster");

?>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Pizza Ordering System</title>

<link rel="stylesheet" href="css/bootstrap.min.css"/>

<script src="js/jquery-2.2.2.min.js"></script>

<script src="js/bootstrap.min.js"></script>

<script src="js/main.js"></script>

</head>

<body style="background-image:url(images/3.jpg);">

<div class="navbar navbar-inverse navbar-fixed-top">

<div class="container-fluid">

<div class="navbar-header">

<a href="#" class="navbar-brand">Pizza Ordering</a>

</div>

<ul class="nav navbar-nav">

<li>

<a href="clients.php">Clients</a>

</li>

<li>

<a href="staffs\_info.php">Staffs</a>

</li>

<li>

<a href="pizzamaster.php">Pizza Master</a>

</li>

<li>

<a href="booking\_delivery\_view.php"> Bookings</a>

</li>

<li>

<a href="delivery.php"> Delivery</a>

</li>

<li>

<a href="reports.php">Reports</a>

</li>

<li><a href="logout.php">Logout</a></li>

</ul>

<ul class="nav navbar-nav navbar-right">

<li><a href="#"><?php echo "Hi ".$\_SESSION["uname"]; ?></a>

</li>

</ul>

</div>

</div>

<div class="container-fluid">

<p><br></p>

<p><br></p>

<div class="row">

<div class="col-md-12">

<table class="table table-bordered" style="background-color:#000;color:#fff;">

<thead class="thead-dark">

<tr>

<th>Pizza Name</th>

<th>Topups</th>

<th>Added Contents</th>

<th>Specialities</th>

<th>Price</th>

<th class="text-center">Action</th>

</tr>

</thead>

<tbody>

<?php

while ($row=mysqli\_fetch\_array($results))

{

?>

<tr>

<td><?php echo $row['PizzaName']; ?></td>

<td><?php echo $row['Topups']; ?></td>

<td><?php echo $row['AddedContents']; ?></td>

<td><?php echo $row['Specialities']; ?></td>

<td><?php echo $row['Price']; ?></td>

<td><a style="text-decoration:none;padding:2px 5px;color:#0099cc;border-radius:3px;backgroud:#800000;" href="pizzamaster.php?edit=<?php echo $row['pizzaid']; ?>">Show</a></td>

</tr>

<?php

}

?>

</tbody>

</table>

</div>

<div class="col-md-2"></div>

<div class="col-md-8">

<div class="panel panel-primary">

<div class="panel-heading text-center">Pizza Master</div>

<div class="panel-body">

<form method="post" enctype="multipart/form-data" action="#">

<input type="hidden" name="id" value=" <?php echo $id; ?>">

<div class="row">

<div class="col-md-6">

<label for="Name">Pizza Name</label>

<input type="text" class="form-control" id="txtname" name="txtname" value="<?php echo $txtname; ?>">

</div>

<div class="col-md-6">

<label for="Name">TopUps</label>

<input type="text" class="form-control" id="txtflavour" name="txtflavour" value="<?php echo $txtflavour; ?>">

</div>

</div>

<div class="row">

<div class="col-md-3">

<label for="Name">Added Contents</label>

<input type="text" class="form-control" id="txtcontents" name="txtcontents" value="<?php echo $txtcontents; ?>">

</div>

<div class="col-md-3">

<label for="Name">Specialities</label>

<input type="text" class="form-control" id="txtspecial" name="txtspecial" value="<?php echo $txtspecial; ?>">

</div>

<div class="col-md-3">

<label for="Name">Price</label>

<input type="text" class="form-control" id="txtprice" name="txtprice" value="<?php echo $txtprice; ?>">

</div>

<div class="col-md-3">

<label for="Name">Pizza Image</label>

<input type="file" name="file" />

</div>

</div>

<p></br></p>

<div class="row">

<div class="col-md-3">

</div>

<div class="col-md-2">

<button type="submit" id="btnsave" name="btnsave" class="btn btn-success btn-md center-block">Save</button>

</div>

<div class="col-md-2">

<button type="submit" id="btnedit" name="btnedit" class="btn btn-success btn-md center-block">Update</button>

</div>

<div class="col-md-2">

<button type="submit" id="btndelete" name="btndelete" class="btn btn-success btn-md center-block">Delete</button>

</div>

<div class="col-md-3">

</div>

</div>

</div>

</form>

<div class="panel-footer">Pizza Ordering System</div>

</div>

</div>

<div class="col-md-2"></div>

</div>

</div>

</body>

</html>

Pizza Master View

<?php

session\_start();

require "Connection.php";

?>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Pizza Ordering System</title>

<link rel="stylesheet" href="css/bootstrap.min.css"/>

<script src="js/jquery-2.2.2.min.js"></script>

<script src="js/bootstrap.min.js"></script>

<script src="js/main.js"></script>

</head>

<body>

<div class="navbar navbar-inverse navbar-fixed-top">

<div class="container-fluid">

<div class="navbar-header">

<a href="#" class="navbar-brand">Pizza Ordering</a>

</div>

<ul class="nav navbar-nav">

<li>

<a href="booking\_master.php">Bookings</a>

</li>

<li>

<a href="feedback.php">Feedbacks</a>

</li>

<li><a href="logout.php">Logout</a></li>

</ul>

<ul class="nav navbar-nav navbar-right">

<li><a href="#"><?php echo "Hi ".$\_SESSION["uname"]; ?></a>

</li>

</ul>

</div>

</div>

<p><b></p>

<p><br></p>

<div class="container-fluid">

<p><br></p>

<div class="row">

<div class="col-md-12">

<div class="panel panel-primary">

<div class="panel-heading text-center">Pizza - Varieties</div>

<div class="panel-body">

<table class="table table-bordered" style="background-color:#000;color:#fff;">

<thead class="thead-dark">

</thead>

<tbody>

<?php

$results = mysqli\_query($con,"select \* from pizzamaster");

while ($row=mysqli\_fetch\_array($results))

{

$prod\_id=$row["pizzaid"];

$pizzaname=$row["PizzaName"];

$flavor=$row["Topups"];

$contents=$row["AddedContents"];

$special=$row["Specialities"];

$price=$row["Price"];

$pizzaimage=$row["pizzaimage"];

echo "

<div class='col-md-3'>

<div class='panel panel-info'>

<div class='panel-heading'><b>Pizza Name : $pizzaname </b> Price : $price</div>

<div class='panel-body'>

<p>$contents</p>

<p>$special</p>

<img src='uploads/$pizzaimage' style='width:120px; height:100px;float:left;'/>

</div>

<div class='panel-heading'><b>$flavor</b>

</div>

</div>

</div>

";

}

?>

</tbody>

</table>

</div>

</div>

</div>

<div class="panel-footer">Pizza Ordering System</div>

</div>

</div>

</body>

</html>

Feedback

<?php

require "Connection.php";

session\_start();

if($\_POST)

{

$txtname = $\_POST['txtname'];

$txtmobile=$\_POST['txtmobile'];

$txtfeedback=$\_POST['txtfeedback'];

mysqli\_query($con,"insert into feedbacks set

Name = '$txtname',

Mobile = '$txtmobile',

feedback = '$txtfeedback'")or die('Query Not Working 2 : '.mysqli\_error());

header("location: feedback.php");

}

?>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Pizza Ordering</title>

<link rel="stylesheet" href="css/bootstrap.min.css"/>

<script src="js/jquery-2.2.2.min.js"></script>

<script src="js/bootstrap.min.js"></script>

<script src="js/main.js"></script>

</head>

<body style="background-color:#000;">

<div class="navbar navbar-inverse navbar-fixed-top">

<div class="container-fluid">

<div class="navbar-header">

<a href="#" class="navbar-brand">Pizza Ordering</a>

</div>

<ul class="nav navbar-nav">

<li>

<a href="booking\_master.php">Bookings</a>

</li>

<li>

<a href="feedback.php">Feedbacks</a>

</li>

<li><a href="logout.php">Logout</a></li>

</ul>

<ul class="nav navbar-nav navbar-right">

<li><a href="#"><?php echo "Hi ".$\_SESSION["uname"]; ?></a></li>

</ul>

</div>

</div>

<p><br></p>

<p><br></p>

<div class="container-fluid" style="background-color:#000;">

<div class="row">

<div class="col-md-1"></div>

<div class="col-md-10">

<div class="panel panel-info" style="background-image:url(images/6.jpg)">

<div class="panel-heading">Pizza Ordering - Feedbacks</div>

<div class="panel-body">

<form method="post" id="frmfeedback" action="#">

<div class="row">

<div class="col-md-6">

<label for="Name">Name</label>

<input type="text" class="form-control" id="txtname" name="txtname">

</div>

<div class="col-md-6">

<label for="Name">Mobile</label>

<input type="text" class="form-control" id="txtmobile" name="txtmobile">

</div>

</div>

<div class="row">

<div class="col-md-12">

<label for="Name">Feedback</label>

<textarea rows="10" id="txtfeedback" name="txtfeedback" class="form-control"></textarea

</div>

</div>

<p></br></p>

<div class="row">

<div class="col-md-5">

</div>

<div class="col-md-2">

<button type="submit" id="btnsave" name="btnsave" class="btn btn-success btn-lg center-block">Save</button>

</div>

<div class="col-md-5">

</div>

</div>

</div>

</div>

<div class="panel-footer" style="background-image:url(images/6.jpg)">&copy; 2025 Pizza Ordering</div>

</div>

</div>

<div class="col-md-1"></div>

</div>

</div>

</body>

</html>

**Conclusion**

All the objectives that had been charted out in the initial phases were achieved successfully. System Features: System satisfies all the requirements for which the company developed the system. System has strong security. System is fully GUI based. It is easy operate and user friendly. Platform includes the inbuilt backup and recovery facility. Working on the project was a good experience. Working together in teams helped us to communicate better. We understand the importance of planning and designing as a part of Online Pizza Ordering System development. The concept of peer-reviews helped to rectify the problems as and when they occurred and also helped us to get some valuable suggestions that were incorporated by us. Developing the project has helped us to gain some experienced on real time development procedures.

**FUTURE ENHANCEMENT**

The project has covered almost all the requirements. Further requirements and improvements

can easily be done since the coding is mainly structured or modular in nature. Improvements

can be appended by changing the existing modules. In future this project will be useful for real time AI Based application system with slight modification in the design and also in future this project can be made mobile based web application.

**Bibliography**

**BOOK REFERENCES**

* + Felke-Morris, Web Development &amp; Design Foundations with HTML5, 10th Edition,
  + Addison-Wesley, 2020.
  + W3Techs - World Wide Web Technology Surveys. W3Techs. 26 April 2021.
  + Retrieved 26 April 2021.
  + Learning PHP,MySQL With jQuery, CSS &amp; HTML5 (Learning PHP, MYSQL,
  + Javascript, CSS &amp; HTML5) Robin Nixon - Jun 12, 2018.
  + PHP &amp; MySQL: Server-side Web Development is the long-awaited book by author
  + Jon Duckett. 2022
  + The Joy of PHP Programming: A Beginner’s Guide to Programming Interactive Web
  + Applications with PHP and MySQL, Alan Forbes. Fifth Edition

**WEBSITES REFERENCES**

* https://www.tutorialspoint.com/php/php\_tutorial.pdf
* www.freetechbooks.com
* www.slideshare.com
* www.w3schools.com
* www.programmersheaven.com
* www.phpreferencebook.com