**ONLINE VOTING ANALYSIS SYSTEM**

**Abstract**

The Online Voting Analysis System delivers a seamless and user-friendly experience while ensuring robust data management and security. In Our Project we use front-end as PHP and Back-End as MYSQL. Online Voting Analysis System is a dynamic platform designed to revolutionize the electoral process, offering administrators powerful tools to oversee and manage elections effectively. Administrators have the capability to view user information and add candidate profiles, ensuring that the system remains updated and organized. For users, the process begins with registration, where they provide their email addresses. Upon registration, users receive a unique one-time password (OTP) via email for authentication. This stringent verification process ensures that only legitimate users can access the system, enhancing security and trust. Once authenticated, users gain access to their personalized voting interface, where they can view candidate profiles and cast their votes securely. Each vote is recorded anonymously, guaranteeing confidentiality and integrity throughout the voting process. After submitting their vote, users receive a confirmation email acknowledging their participation. This not only provides users with assurance but also serves as a record of their engagement in the electoral process.

**Introduction:**

Online Voting Analysis System represents a significant advancement in the realm of electoral processes, introducing a sophisticated platform that harnesses the power of technology to enhance the efficiency, transparency, and security of elections. In today's digital age, traditional voting methods are being increasingly supplemented or replaced by online voting systems due to their convenience, accessibility, and potential for increased voter turnout. This system aims to address the challenges and limitations of traditional voting systems while providing administrators and users with a comprehensive suite of features and functions

At its core, the Online Voting Analysis System is designed to facilitate the entire electoral process, from voter registration to result analysis, in a seamless and user-friendly manner. Administrators play a pivotal role in overseeing the system, with capabilities to manage user registrations, add candidate profiles, monitor voting activities, and analyze election results. This centralized control not only streamlines administrative tasks but also ensures the integrity and fairness of the electoral process

For users, the system offers a secure and intuitive platform to exercise their voting rights. Through a simple registration process, users are authenticated via email and provided with unique credentials to access their personalized voting interface. This interface presents them with comprehensive information about candidates and enables them to cast their votes securely and anonymously. By leveraging encryption and authentication mechanisms, the system safeguards the confidentiality and integrity of each vote, instilling trust and confidence among user.

Furthermore, the Online Voting Analysis System incorporates robust analytical tools to analyze voting patterns, trends, and outcomes. By generating detailed reports and visualizations, administrators can gain valuable insights into voter behavior, candidate performance, and overall election dynamics. These insights not only inform decision-making but also contribute to the continuous improvement of the electoral process.

**Existing System:**

Traditional voting methods have long relied on physical polling stations, paper-based ballots, and manual vote counting to facilitate elections. While these methods have been effective to some extent, they are often plagued by various challenges and limitations that can undermine the integrity and efficiency of the electoral process.

**Drawbacks:**

1. Physical Polling Stations
2. Manual Vote Counting
3. Limited Accessibility
4. Security Concerns

**Proposed System:**

The proposed Online Voting Analysis System (OVAS) offers a modern solution to the traditional voting process by harnessing the power of technology. With OVAS, users can register online using their email addresses, receiving a unique One-Time Password (OTP) for secure authentication. Once logged in, voters can conveniently access their personalized dashboard to cast their ballots electronically. Meanwhile, election administrators can easily manage candidate registrations, monitor the voting process in real-time, and generate comprehensive reports for analysis. By automating and streamlining the voting process, OVAS aims to enhance accessibility, transparency, and efficiency in elections, ensuring a fair and credible outcome..

**Advantages:**

1. Online Registration and Authentication
2. User-Friendly Interface
3. Candidate Registration and Management
4. Secure Voting Mechanism

**MODULES**

1. Admin
2. User
3. Add Candidate
4. Reports

**Admin:**

Admin need to login with valid login credentials.Admin view all the registered users, types, current and historical unit readings and alert status. Allows Admin to filter, search, and export user data for analysis and reporting purposes.

**User:**

User can register with all their details. Registered user can Login with their mailid.User can register with basic information like id, username, email, address, mobile number,etc..After registration user can vote the candidate and sent the mail alert to the user.

**Add Candidate:**

By using this module, Admin can add candidate with their perceptive details like id,name,symbol etc..

**Report:**

By this module the Admin gets the final report of the work. The admin who receives the candidate voting report after the user vote .admin can view the result count in their page..

**Software Requirements:**

Operating Front end **:**  PHP

Data Base **:** My SQL

Operating System  **:** Windows 10

**Hardware Requirements:**

* Processor – Dual Core
* Hard Disk – 500 GB
* Memory – 2GB RAM

# PHP Programming

**PHP** is a [scripting](http://en.wikipedia.org/wiki/Scripting_programming_language) language designed to fill the gap between [SSI](http://en.wikipedia.org/wiki/Server_Side_Includes) (Server Side Includes) and [Perl](http://en.wikibooks.org/wiki/Perl), intended for the Web environment. Its principal application is the implementation of Web pages having dynamic content. PHP has gained quite a following in recent times, and it is one of the frontrunners in the Open Source software movement. Its popularity derives from its C-like syntax, and its simplicity. The newest version of PHP is 5.5 and it is heavily recommended to always use the newest version for better security, performance and of course features.

If you've been to a website that prompts you to login, you've probably encountered a server-side scripting language. Due to its market saturation, this means you've probably come across PHP. [PHP](http://en.wikipedia.org/wiki/PHP) was designed by [Rasmus Lerdorf](http://en.wikipedia.org/wiki/Rasmus_Lerdorf) to display his resume online and to collect data from his visitors.

Basically, PHP allows a static webpage to become dynamic. "PHP" is an acronym that stands for "**P**HP: **H**ypertext **P**reprocessor". The word "Preprocessor" means that PHP makes changes before the HTML page is created. This enables developers to create powerful applications that can publish a blog, remotely control hardware, or run a powerful website such as Wikipedia or Wikibooks. Of course, to accomplish something such as this, you need a database application such as MySQL.

Before you embark on the wonderful journey of Server Side Processing, it is recommended that you have a basic understanding of the [HyperText Markup Language (HTML)](http://en.wikibooks.org/wiki/HTML). But [PHP](http://en.wikipedia.org/wiki/PHP) can also be used to build [GUI](http://en.wikipedia.org/wiki/GUI)-driven applications for example by using [PHP-GTK](http://en.wikipedia.org/wiki/PHP-GTK).

**PHP** is a [server-side scripting](http://en.wikipedia.org/wiki/Server-side_scripting) language designed for [web development](http://en.wikipedia.org/wiki/Web_development) but also used as a [general-purpose programming language](http://en.wikipedia.org/wiki/General-purpose_programming_language). As of January 2013, PHP was installed on more than 240 million [websites](http://en.wikipedia.org/wiki/Website) (39% of those sampled) and 2.1 million [web servers](http://en.wikipedia.org/wiki/Web_server). Originally created by [Rasmus Lerdorf](http://en.wikipedia.org/wiki/Rasmus_Lerdorf) in 1994, the [reference implementation](http://en.wikipedia.org/wiki/Reference_implementation) of PHP (powered by the [Zend Engine](http://en.wikipedia.org/wiki/Zend_Engine)) is now produced by The PHP Group. While PHP originally stood for *Personal Home Page*, it now stands for *PHP: Hypertext Preprocessor*, which is a [recursive](http://en.wikipedia.org/wiki/Recursive_acronym) [backronym](http://en.wikipedia.org/wiki/Backronym).

PHP code can be simply mixed with [HTML](http://en.wikipedia.org/wiki/HTML) code, or it can be used in combination with various [templating engines](http://en.wikipedia.org/wiki/Web_template_system) and [web frameworks](http://en.wikipedia.org/wiki/Web_framework). PHP code is usually processed by a PHP [interpreter](http://en.wikipedia.org/wiki/Interpreter_%28computing%29), which is usually implemented as a web server's native [module](http://en.wikipedia.org/wiki/Plugin_%28computing%29) or a [Common Gateway Interface](http://en.wikipedia.org/wiki/Common_Gateway_Interface) (CGI) executable. After the PHP code is interpreted and executed, the web server sends resulting output to its client, usually in form of a part of the generated web page; for example, PHP code can generate a web page's HTML code, an image, or some other data. PHP has also evolved to include a [command-line interface](http://en.wikipedia.org/wiki/Command-line_interface) (CLI) capability and can be used in [standalone](http://en.wikipedia.org/wiki/Computer_software) [graphical applications](http://en.wikipedia.org/wiki/Graphical_user_interface).

The canonical PHP interpreter, powered by the Zend Engine, is [free software](http://en.wikipedia.org/wiki/Free_software) released under the [PHP License](http://en.wikipedia.org/wiki/PHP_License). PHP has been widely ported and can be deployed on most web servers on almost every [operating system](http://en.wikipedia.org/wiki/Operating_system) and [platform](http://en.wikipedia.org/wiki/Computing_platform), free of charge.

Despite its popularity, no written [specification](http://en.wikipedia.org/wiki/Formal_specification) or standard existed for the PHP language until 2014, leaving the canonical PHP interpreter as a [*de facto*](http://en.wikipedia.org/wiki/De_facto) standard. Since 2014, there is ongoing work on creating a formal PHP specification.

PHP development began in 1994 when [Rasmus Lerdorf](http://en.wikipedia.org/wiki/Rasmus_Lerdorf) wrote a series of [Common Gateway Interface](http://en.wikipedia.org/wiki/Common_Gateway_Interface) (CGI) binaries in C, which he used to maintain his [personal homepage](http://en.wikipedia.org/wiki/Personal_homepage). He extended them to add the ability to work with [web forms](http://en.wikipedia.org/wiki/Web_form) and to communicate with [databases](http://en.wikipedia.org/wiki/Database), and called this implementation "Personal Home Page/Forms Interpreter" or PHP/FI.

PHP/FI could be used to build simple, dynamic [web applications](http://en.wikipedia.org/wiki/Web_application). Lerdorf initially announced the release of PHP/FI as "Personal Home Page Tools (PHP Tools) version 1.0" publicly to accelerate [bug](http://en.wikipedia.org/wiki/Software_bug) location and improve the code, on the [Usenet](http://en.wikipedia.org/wiki/Usenet) discussion group *comp.infosystems.www.authoring.cgi* on June 8, 1995. This release already had the basic functionality that PHP has as of 2013. This included Perl-like variables, form handling, and the ability to embed HTML. The [syntax](http://en.wikipedia.org/wiki/Syntax) resembled that of Perl but was simpler, more limited and less consistent.

Early PHP was not intended to be a new programming language, and grew organically, with Lerdorf noting in retrospect: "I don’t know how to stop it, there was never any intent to write a programming language […] I have absolutely no idea how to write a programming language, I just kept adding the next logical step on the way." A development team began to form and, after months of work and [beta](http://en.wikipedia.org/wiki/Beta_development_stage) testing, officially released PHP/FI 2 in November 1997.

One criticism of PHP is that it was not originally designed, but instead it was developed organically; among other things, this has led to inconsistent naming of functions and inconsistent ordering of their parameters. In some cases, the function names were chosen to match the lower-level libraries which PHP was "wrapping", while in some very early versions of PHP the length of the function names was used internally as a [hash function](http://en.wikipedia.org/wiki/Hash_function), so names were chosen to improve the distribution of hash values.

[Zeev Suraski](http://en.wikipedia.org/wiki/Zeev_Suraski) and [Andi Gutmans](http://en.wikipedia.org/wiki/Andi_Gutmans) rewrote the [parser](http://en.wikipedia.org/wiki/Parser) in 1997 and formed the base of PHP 3, changing the language's name to the [recursive acronym](http://en.wikipedia.org/wiki/Recursive_acronym) *PHP: Hypertext Preprocessor*. Afterwards, public testing of PHP 3 began, and the official launch came in June 1998. Suraski and Gutmans then started a new [rewrite](http://en.wikipedia.org/wiki/Rewrite_%28programming%29) of PHP's core, producing the [Zend Engine](http://en.wikipedia.org/wiki/Zend_Engine) in 1999. They also founded [Zend Technologies](http://en.wikipedia.org/wiki/Zend_Technologies) in [Ramat Gan](http://en.wikipedia.org/wiki/Ramat_Gan), Israel.

On May 22, 2000, PHP 4, powered by the Zend Engine 1.0, was released. As of August 2008 this branch reached version 4.4.9. PHP 4 is no longer under development nor will any security updates be released.

On July 13, 2004, PHP 5 was released, powered by the new Zend Engine II. PHP 5 included new features such as improved support for [object-oriented programming](http://en.wikipedia.org/wiki/Object-oriented_programming), the PHP Data Objects (PDO) extension (which defines a lightweight and consistent interface for accessing databases), and numerous performance enhancements. In 2008 PHP 5 became the only stable version under development. [Late static binding](http://en.wikipedia.org/wiki/Late_static_binding) had been missing from PHP and was added in version 5.3.

Many high-profile open-source projects ceased to support PHP 4 in new code as of February 5, 2008, because of the GoPHP5 initiative, provided by a consortium of PHP developers promoting the transition from PHP 4 to PHP 5.

Over time, PHP interpreters became available on most existing [32-bit](http://en.wikipedia.org/wiki/32-bit) and [64-bit](http://en.wikipedia.org/wiki/64-bit) operating systems, either by building them from the PHP source code, or by using pre-built binaries. For the PHP versions 5.3 and 5.4, the only available [Microsoft Windows](http://en.wikipedia.org/wiki/Microsoft_Windows) binary distributions were 32-bit [x86](http://en.wikipedia.org/wiki/X86) builds, requiring Windows 32-bit compatibility mode while using [Internet Information Services](http://en.wikipedia.org/wiki/Internet_Information_Services) (IIS) on a 64-bit Windows platform. PHP version 5.5 made the 64-bit [x86-64](http://en.wikipedia.org/wiki/X86-64) builds available for Microsoft Windows.

**PHP 6 and Unicode**

PHP received mixed reviews due to lacking native [Unicode](http://en.wikipedia.org/wiki/Unicode) support at the core language level. In 2005, a project headed by Andrei Zmievski was initiated to bring native Unicode support throughout PHP, by embedding the [International Components for Unicode](http://en.wikipedia.org/wiki/International_Components_for_Unicode) (ICU) library, and representing text strings as [UTF-16](http://en.wikipedia.org/wiki/UTF-16) internally. Since this would cause major changes both to the internals of the language and to user code, it was planned to release this as version 6.0 of the language, along with other major features then in development.

However, a shortage of developers who understood the necessary changes, and performance problems arising from conversion to and from UTF-16, which is rarely used in a web context, led to delays in the project. As a result, a PHP 5.3 release was created in 2009, with many non-Unicode features back-ported from PHP 6, notably namespaces. In March 2010, the project in its current form was officially abandoned, and a PHP 5.4 release was prepared containing most remaining non-Unicode features from PHP 6, such as traits and closure re-binding. Initial hopes were that a new plan would be formed for Unicode integration, but as of 2014 none has been adopted.

**PHP 7**

As of 2014, work is underway on a new major PHP version named PHP 7. There was some dispute as to whether the next major version of PHP was to be called PHP 6 or PHP 7. While the PHP 6 unicode experiment had never been released, a number of articles and book titles referenced the old PHP 6 name, which might have caused confusion if a new release were to reuse the PHP 6 name. After a vote, the name PHP 7 was chosen.

PHP 7 gets its foundations from an experimental PHP [branch](http://en.wikipedia.org/wiki/Branching_%28revision_control%29) that was originally named *phpng* (*PHP next generation*), which aims at optimizing PHP performance by refactoring the Zend Engine while retaining near-complete language compatibility. As of 14 July 2014, [WordPress](http://en.wikipedia.org/wiki/WordPress)-based benchmarks, which serve as the main benchmark suite for phpng project, show an almost 100% increase in performance. Changes from phpng are also expected to make it easier to improve performance in the future, as more compact data structures and other changes are seen as better suited for a successful migration to a [just-in-time](http://en.wikipedia.org/wiki/Just-in-time_compilation) (JIT) compiler. Because of the significant changes, this reworked Zend Engine will be called *Zend Engine 3*, succeeding the Zend Engine 2 used in PHP 5.

In terms of new language features, PHP 7 will add features such as return type declarations, which will complement its existing parameter type declarations. PHP 7 will also contain an improved variable syntax which is internally consistent and complete, resolving a long-standing issue in PHP, what will allow use of ->, [], (), {}, and :: operators with arbitrary meaningful left-hand-side expressions.

## Syntax

The following [Hello world program](http://en.wikipedia.org/wiki/Hello_world_program) is written in PHP code embedded in an [HTML](http://en.wikipedia.org/wiki/HTML) document:

<!DOCTYPE html>

<html>

<head>

<title>PHP Test</title>

</head>

<body>

<?php echo '<p>Hello World</p>'; ?>

</body>

</html>

However, as PHP does not need to be embedded in HTML or used with a web server, the simplest version of a Hello World program can be written like this, with the closing tag omitted as preferred in files containing pure PHP code (prior to PHP 5.4.0, this short syntax for echo() only works with the short\_open\_tag configuration setting enabled, while for PHP 5.4.0 and later it is always available):

<?= 'Hello world';

The PHP interpreter only executes PHP code within its [delimiters](http://en.wikipedia.org/wiki/Delimiter). Anything outside its delimiters is not processed by PHP (although non-PHP text is still subject to [control structures](http://en.wikipedia.org/wiki/Control_structure) 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](http://en.wikipedia.org/wiki/String_%28computer_science%29) or [variable](http://en.wikipedia.org/wiki/Variable_%28programming%29)) and ?> as well as [ASP](http://en.wikipedia.org/wiki/Active_Server_Pages)-style short forms <% or <%= and %>. Short delimiters make script files less portable, since support for them can be disabled in the local PHP configuration, and they are therefore 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](http://en.wikipedia.org/wiki/XHTML) and other [XML](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/Dollar_sign), and a [type](http://en.wikipedia.org/wiki/Primitive_type) does not need to be specified in advance. PHP 5 introduced *type hinting* that allows functions to force their parameters to be objects of a specific class, arrays, interfaces or [callback functions](http://en.wikipedia.org/wiki/Callback_function). However, type hints can not be used with scalar types such as integer or string.

Unlike function and class names, variable names are case sensitive. Both double-quoted ("") and [heredoc](http://en.wikipedia.org/wiki/Heredoc) strings provide the ability to interpolate a variable's value into the string. PHP treats [newlines](http://en.wikipedia.org/wiki/Newline) as [whitespace](http://en.wikipedia.org/wiki/Whitespace_character) in the manner of a [free-form language](http://en.wikipedia.org/wiki/Free-form_language), and statements are terminated by a semicolon. PHP has three types of [comment syntax](http://en.wikipedia.org/wiki/Comparison_of_programming_languages_%28syntax%29#Comments): /\* \*/ 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++, C#, Java and Perl.

### Data types

PHP stores whole numbers in a platform-dependent range, either a 64-bit or 32-bit [signed](http://en.wikipedia.org/wiki/Signed_number_representations) [integer](http://en.wikipedia.org/wiki/Integer_%28computer_science%29) equivalent to the [C-language long type](http://en.wikipedia.org/wiki/C_variable_types_and_declarations). 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](http://en.wikipedia.org/wiki/Octal), [hexadecimal](http://en.wikipedia.org/wiki/Hexadecimal), and [binary](http://en.wikipedia.org/wiki/Binary_code) notations.

[Floating point](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/Scientific_notation). PHP has a native [Boolean](http://en.wikipedia.org/wiki/Boolean_datatype) type that is similar to the native Boolean types in [Java](http://en.wikipedia.org/wiki/Java_%28programming_language%29) and [C++](http://en.wikipedia.org/wiki/C%2B%2B). Using the Boolean type conversion rules, non-zero values are interpreted as true and zero as false, as in [Perl](http://en.wikipedia.org/wiki/Perl) and C++.

The null data type represents a variable that has no value; *NULL* is the only allowed value for this data type.

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](http://en.wikipedia.org/wiki/Hash_table) with both keys and values, and the two can be intermingled. PHP also supports [strings](http://en.wikipedia.org/wiki/String_%28computing%29), which can be used with single quotes, double quotes, nowdoc or [heredoc](http://en.wikipedia.org/wiki/Heredoc) syntax.

The Standard PHP Library (SPL) attempts to solve standard problems and implements efficient data access interfaces and classes.

### Functions

PHP has hundreds of functions provided by the core language functionality and thousands more available via various extensions; these functions are well documented in the online PHP documentation. However, the built-in library has a wide variety of naming conventions and associated inconsistencies, as described under [history](http://en.wikipedia.org/wiki/PHP#ORGANIC) above.

Additional functions can be defined by the developer:

function myAge($birthYear) // defines a function, this one is named "myAge"

{

$yearsOld = date('Y') - $birthYear; // calculates the age

return $yearsOld . ' year' . ($yearsOld != 1 ? 's' : ''); // returns the age in a descriptive form

}

echo 'I am currently ' . myAge(1981) . ' old.'; // outputs the text concatenated

// with the return value of myAge()

// As the result of this syntax, myAge() is called.

// In 2014, the output of this sample program will be 'I am currently 33 years old.'

In PHP, normal functions are not [first-class](http://en.wikipedia.org/wiki/First-class_function) and can only be referenced by their name directly, or dynamically by a variable containing the name of the function (referred to as "variable functions"). User-defined functions can be created at any time without being [prototyped](http://en.wikipedia.org/wiki/Function_prototype). Functions can be defined inside code blocks, permitting a [run-time decision](http://en.wikipedia.org/wiki/Dynamic_dispatch) as to whether or not a function should be defined. Function calls must use parentheses, with the exception of zero-argument class [constructor](http://en.wikipedia.org/wiki/Constructor_%28computer_science%29) functions called with the PHP new operator, where parentheses are optional.

Until PHP 5.3, support for true [anonymous functions](http://en.wikipedia.org/wiki/Anonymous_functions) or [closures](http://en.wikipedia.org/wiki/Closure_%28computer_science%29) did not exist in PHP. While create\_function() exists since PHP 4.0.1, it is merely a thin wrapper around eval() that allows normal PHP functions to be created during program execution.Also, support for variable functions allows normal PHP functions to be used, for example, as [callbacks](http://en.wikipedia.org/wiki/Callback_function) or within [function tables](http://en.wikipedia.org/wiki/Function_table). PHP 5.3 added support for closures, which are true anonymous, first-class functions, whose syntax can be seen in the following example:

function getAdder($x)

{

return function($y) use ($x)

{

return $x + $y;

};

}

$adder = getAdder(8);

echo $adder(2); // prints "10"

In the example above, getAdder() function creates a closure using passed argument $x (the keyword use imports a variable from the lexical context), which takes an additional argument $y, and returns the created closure to the caller. Such a function is a first-class object, meaning that it can be stored in a variable, passed as a parameter to other functions, etc.

The [goto](http://en.wikipedia.org/wiki/Goto) flow control statement is used as in the following example:

function lock()

{

$file = fopen('file.txt', 'r+');

retry:

if (!flock($file, LOCK\_EX | LOCK\_NB))

goto retry;

fwrite($file, 'Success!');

fclose($file);

}

When flock() is called, PHP opens a file and tries to lock it. The target label retry: defines the point to which execution should return if flock() is unsuccessful and goto retry; is called. The goto statement is restricted and requires that the target label be in the same file and context.

The goto statement has been supported since PHP 5.3.

### Objects

Basic [object-oriented programming](http://en.wikipedia.org/wiki/Object-oriented_programming) functionality was added in PHP 3 and improved in PHP 4. Object handling was completely rewritten for PHP 5, expanding the feature set and enhancing performance. In previous versions of PHP, objects were handled like [value types](http://en.wikipedia.org/wiki/Value_type). The drawback of this method was that the whole object was copied when a variable was assigned or passed as a parameter to a method. In the new approach, objects are referenced by [handle](http://en.wikipedia.org/wiki/Handle_%28computing%29), and not by value.

PHP 5 introduced private and protected [member variables](http://en.wikipedia.org/wiki/Member_variable) and methods, along with [abstract classes](http://en.wikipedia.org/wiki/Abstract_type), [final classes](http://en.wikipedia.org/wiki/Final_type), [abstract methods](http://en.wikipedia.org/wiki/Abstract_method), and [final methods](http://en.wikipedia.org/wiki/Final_method). It also introduced a standard way of declaring [constructors](http://en.wikipedia.org/wiki/Constructor_%28computer_science%29) and [destructors](http://en.wikipedia.org/wiki/Destructor_%28computer_science%29), similar to that of other object-oriented languages such as [C++](http://en.wikipedia.org/wiki/C%2B%2B), and a standard [exception handling](http://en.wikipedia.org/wiki/Exception_handling) model. Furthermore, PHP 5 added [interfaces](http://en.wikipedia.org/wiki/Interface_%28computing%29) and allowed for multiple interfaces to be implemented. There are special interfaces that allow objects to interact with the runtime system. [Objects](http://en.wikipedia.org/wiki/Object_%28computer_science%29) implementing ArrayAccess can be used with [array](http://en.wikipedia.org/wiki/Array_data_type) syntax and objects implementing [Iterator](http://en.wikipedia.org/wiki/Iterator) or [IteratorAggregate](http://en.wikipedia.org/wiki/IteratorAggregate) can be used with the foreach [language construct](http://en.wikipedia.org/wiki/Language_construct). There is no [virtual table](http://en.wikipedia.org/wiki/Virtual_table) feature in the engine, so [static variables](http://en.wikipedia.org/wiki/Static_variable) are bound with a name instead of a reference at compile time.

If the developer creates a copy of an object using the reserved word clone, the Zend engine will check whether a \_\_clone() method has been defined. If not, it will call a default \_\_clone() which will copy the object's properties. If a \_\_clone() method is defined, then it will be responsible for setting the necessary properties in the created object. For convenience, the engine will supply a function that imports the properties of the source object, so the programmer can start with a by-value [replica](http://en.wiktionary.org/wiki/replica) of the source object and only override properties that need to be changed.

The following is a basic example of [object-oriented programming](http://en.wikipedia.org/wiki/Object-oriented_programming) in PHP:

class Person

{

public $firstName;

public $lastName;

public function \_\_construct($firstName, $lastName = '') { // optional second argument

$this->firstName = $firstName;

$this->lastName = $lastName;

}

public function greet() {

return 'Hello, my name is ' . $this->firstName .

(($this->lastName != '') ? (' ' . $this->lastName) : '') . '.';

}

public static function staticGreet($firstName, $lastName) {

return 'Hello, my name is ' . $firstName . ' ' . $lastName . '.';

}

}

$he = new Person('John', 'Smith');

$she = new Person('Sally', 'Davis');

$other = new Person('iAmine');

echo $he->greet(); // prints "Hello, my name is John Smith."

echo '<br />';

echo $she->greet(); // prints "Hello, my name is Sally Davis."

echo '<br />';

echo $other->greet(); // prints "Hello, my name is iAmine."

echo '<br />';

echo Person::staticGreet('Jane', 'Doe'); // prints "Hello, my name is Jane Doe."

The [visibility](http://en.wikipedia.org/wiki/Visibility_%28computer_science%29) of PHP properties and methods is defined using the [keywords](http://en.wikipedia.org/wiki/Keyword_%28computer_programming%29) public, private, and protected. The default is public, if only [var](http://en.wikipedia.org/wiki/Variable_%28programming%29) is used; var is a synonym for public. Items declared public can be accessed everywhere. protected limits access to [inherited classes](http://en.wikipedia.org/wiki/Inherited_class) (and to the class that defines the item). private limits visibility only to the class that defines the item. Objects of the same type have access to each other's private and protected members even though they are not the same instance. PHP's member visibility features have sometimes been described as "highly useful." However, they have also sometimes been described as "at best irrelevant and at worst positively harmful."

## Implementations

The original, only complete and most widely used PHP implementation is powered by the [Zend Engine](http://en.wikipedia.org/wiki/Zend_Engine) and known simply as PHP. To disambiguate it from other implementations, it is sometimes unofficially referred to as "Zend PHP". The Zend Engine [compiles](http://en.wikipedia.org/wiki/Compiler) PHP [source code](http://en.wikipedia.org/wiki/Source_code) on-the-fly into an internal format that it can execute, thus it works as an [interpreter](http://en.wikipedia.org/wiki/Interpreter_%28computing%29). It is also the "reference implementation" of PHP, as PHP has no formal specification, and so the semantics of Zend PHP define the semantics of PHP itself. Due to the complex and nuanced semantics of PHP, defined by how Zend works, it is difficult for competing implementations to offer complete compatibility.

PHP's single-request-per-script-execution model, and the fact the Zend Engine is an interpreter, lead to inefficiency. As a result, various products have been developed to help improve PHP performance. In order to speed up execution time and not have to compile the PHP source code every time the web page is accessed, PHP scripts can also be deployed in the PHP engine's internal format by using an [opcode](http://en.wikipedia.org/wiki/Opcode) cache, which works by [caching](http://en.wikipedia.org/wiki/Cache_%28computing%29) the compiled form of a PHP script (opcodes) in [shared memory](http://en.wikipedia.org/wiki/Shared_memory) to avoid the overhead of [parsing](http://en.wikipedia.org/wiki/Parsing) and [compiling](http://en.wikipedia.org/wiki/Compiling) the code every time the script runs. An opcode cache, [Zend Opcache](http://en.wikipedia.org/wiki/Zend_Opcache), is built into PHP since version 5.5. Another example of a widely used opcode cache is the [Alternative PHP Cache](http://en.wikipedia.org/wiki/Alternative_PHP_Cache) (APC), which is available as a [PECL](http://en.wikipedia.org/wiki/PHP_Extension_Community_Library) extension.

While Zend PHP is still the most popular implementation, several other implementations have been developed. Some of these are [compilers](http://en.wikipedia.org/wiki/Compiler) or support [JIT compilation](http://en.wikipedia.org/wiki/JIT_compilation), and hence offer performance benefits over Zend PHP at the expense of lacking full PHP compatibility. Alternative implementations include the following:

* [HipHop Virtual Machine](http://en.wikipedia.org/wiki/HipHop_Virtual_Machine) (HHVM) – developed at Facebook and available as open source, it converts PHP code into a high-level bytecode (commonly known as an [intermediate language](http://en.wikipedia.org/wiki/Intermediate_language)), which is then translated into x86-64 machine code dynamically at runtime by a [just-in-time](http://en.wikipedia.org/wiki/Just-in-time_compiler) (JIT) compiler, resulting in up to 6× performance improvements.
* [Parrot](http://en.wikipedia.org/wiki/Parrot_virtual_machine) – a virtual machine designed to run dynamic languages efficiently; Pipp transforms the PHP source code into the [Parrot intermediate representation](http://en.wikipedia.org/wiki/Parrot_intermediate_representation), which is then translated into the Parrot's bytecode and executed by the virtual machine.
* [Phalanger](http://en.wikipedia.org/wiki/Phalanger_%28compiler%29) – compiles PHP into [Common Intermediate Language](http://en.wikipedia.org/wiki/Common_Intermediate_Language) (CIL) bytecode
* [HipHop](http://en.wikipedia.org/wiki/HipHop_for_PHP) – developed at Facebook and available as open source, it transforms the PHP scripts into [C++](http://en.wikipedia.org/wiki/C%2B%2B) code and then compiles the resulting code, reducing the server load up to 50%. In early 2013, Facebook deprecated it in favor of HHVM due to multiple reasons, including deployment difficulties and lack of support for the whole PHP language, including the create\_function() and eval() constructs.

## Licensing

PHP is [free software](http://en.wikipedia.org/wiki/Free_software) released under the [PHP License](http://en.wikipedia.org/wiki/PHP_License), which stipulates that:

Products derived from this software may not be called "PHP", nor may "PHP" appear in their name, without prior written permission from group@php.net. You may indicate that your software works in conjunction with PHP by saying "[Foo](http://en.wikipedia.org/wiki/Foo) for PHP" instead of calling it "PHP Foo" or "phpfoo".

This restriction on use of the name *PHP* makes the PHP License incompatible with the [GNU General Public License](http://en.wikipedia.org/wiki/GNU_General_Public_License) (GPL), while the Zend License is incompatible due to an advertising clause similar to that of the original license of BSD.

## Development and community

PHP includes various [free and open-source libraries](http://en.wikipedia.org/wiki/List_of_PHP_libraries) in its source distribution, or uses them in resulting PHP binary builds. PHP is fundamentally an [Internet](http://en.wikipedia.org/wiki/Internet)-aware system with built-in modules for accessing [File Transfer Protocol](http://en.wikipedia.org/wiki/File_Transfer_Protocol) (FTP) servers and many database servers, including [PostgreSQL](http://en.wikipedia.org/wiki/PostgreSQL), [MySQL](http://en.wikipedia.org/wiki/MySQL), [Microsoft SQL Server](http://en.wikipedia.org/wiki/Microsoft_SQL_Server) and [SQLite](http://en.wikipedia.org/wiki/SQLite) (which is an embedded database), [LDAP](http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol) servers, and others. Numerous functions familiar to C programmers, such as those in the [stdio](http://en.wikipedia.org/wiki/Stdio.h) family, are available in standard PHP builds.

PHP allows developers to write [extensions](http://en.wikipedia.org/wiki/Software_extension) in [C](http://en.wikipedia.org/wiki/C_%28programming_language%29) to add functionality to the PHP language. PHP extensions can be compiled statically into PHP or loaded dynamically at runtime. Numerous extensions have been written to add support for the [Windows API](http://en.wikipedia.org/wiki/Windows_API), process management on [Unix-like](http://en.wikipedia.org/wiki/Unix-like) [operating systems](http://en.wikipedia.org/wiki/Operating_system), multibyte strings ([Unicode](http://en.wikipedia.org/wiki/Unicode)), [cURL](http://en.wikipedia.org/wiki/CURL), and several popular [compression formats](http://en.wikipedia.org/wiki/Archive_format). Other PHP features made available through extensions include integration with [IRC](http://en.wikipedia.org/wiki/Internet_Relay_Chat), dynamic generation of images and [Adobe Flash](http://en.wikipedia.org/wiki/Adobe_Flash) content, *PHP Data Objects* (PDO) as an abstraction layer used for accessing databases, and even [speech synthesis](http://en.wikipedia.org/wiki/Speech_synthesis). Some of the language's core functions, such as those dealing with strings and arrays, are also implemented as extensions. The [PHP Extension Community Library](http://en.wikipedia.org/wiki/PHP_Extension_Community_Library) (PECL) project is a repository for extensions to the PHP language.

Some other projects, such as *Zephir*, provide the ability for PHP extensions to be created in a high-level language and compiled into native PHP extensions. Such an approach, instead of writing PHP extensions directly in C, simplifies the development of extensions and reduces the time required for programming and testing.

[Zend Technologies](http://en.wikipedia.org/wiki/Zend_Technologies) provides a [certification](http://en.wikipedia.org/wiki/Zend_Certified_Engineer) exam for programmers to become certified PHP developers.

## Installation and configuration

There are two primary ways for adding support for PHP to a web server – as a native web server module, or as a CGI executable. PHP has a direct module interface called [Server Application Programming Interface](http://en.wikipedia.org/wiki/Server_Application_Programming_Interface) (SAPI), which is supported by many web servers including [Apache HTTP Server](http://en.wikipedia.org/wiki/Apache_HTTP_Server), [Microsoft IIS](http://en.wikipedia.org/wiki/Microsoft_Internet_Information_Server), [Netscape](http://en.wikipedia.org/wiki/Netscape) (now defunct) and [iPlanet](http://en.wikipedia.org/wiki/IPlanet). Some other web servers, such as OmniHTTPd, support the [Internet Server Application Programming Interface](http://en.wikipedia.org/wiki/Internet_Server_Application_Programming_Interface) (ISAPI), which is a [Microsoft](http://en.wikipedia.org/wiki/Microsoft)'s web server module interface. If PHP has no module support for a web server, it can always be used as a [Common Gateway Interface](http://en.wikipedia.org/wiki/Common_Gateway_Interface) (CGI) or [FastCGI](http://en.wikipedia.org/wiki/FastCGI) processor; in that case, the web server is configured to use PHP's CGI executable to process all requests to PHP files.

PHP-FPM (FastCGI Process Manager) is an alternative FastCGI implementation for PHP, bundled with the official PHP distribution since version 5.3.3. When compared to the older FastCGI implementation, it contains some additional features, mostly useful for heavily loaded web servers.

When using PHP for command-line scripting, a PHP [command-line interface](http://en.wikipedia.org/wiki/Command-line_interface) (CLI) executable is needed. PHP supports a CLI SAPI as of PHP 4.3.0. The main focus of this SAPI is developing [shell](http://en.wikipedia.org/wiki/Command_line_interface) applications using PHP. There are quite a few differences between the CLI SAPI and other SAPIs, although they do share many of the same behaviors.

PHP can also be used for writing desktop [graphical user interface](http://en.wikipedia.org/wiki/Graphical_user_interface) (GUI) applications, by using the [PHP-GTK](http://en.wikipedia.org/wiki/PHP-GTK) extension. PHP-GTK is not included in the official PHP distribution, and as an extension it can be used only with PHP versions 5.1.0 and newer. The most common way of installing PHP-GTK is compiling it from the source code.

When PHP is installed and used in [cloud](http://en.wikipedia.org/wiki/Cloud_computing) environments, [software development kits](http://en.wikipedia.org/wiki/Software_development_kit) (SDKs) are provided for using cloud-specific features. For example:

* [Amazon Web Services](http://en.wikipedia.org/wiki/Amazon_Web_Services) provides the AWS SDK for PHP
* [Windows Azure](http://en.wikipedia.org/wiki/Windows_Azure) can be used with the Windows Azure SDK for PHP.

Numerous configuration options are supported, affecting both core PHP features and extensions. Configuration file php.ini is searched for in different locations, depending on the way PHP is used. The configuration file is split into various sections, while some of the configuration options can be also set within the web server configuration.

## [http://upload.wikimedia.org/wikipedia/commons/thumb/8/82/LAMP_software_bundle.svg/400px-LAMP_software_bundle.svg.png](http://en.wikipedia.org/wiki/File:LAMP_software_bundle.svg)

## Use

A broad overview of the LAMP software bundle, displayed here together with [Squid](http://en.wikipedia.org/wiki/Squid_%28software%29).

PHP is a general-purpose scripting language that is especially suited to [server-side](http://en.wikipedia.org/wiki/Server-side_scripting) [web development](http://en.wikipedia.org/wiki/Web_development), in which case PHP generally runs on a [web server](http://en.wikipedia.org/wiki/Web_server). Any PHP code in a requested file is [executed](http://en.wikipedia.org/wiki/Execution_%28computing%29) by the PHP runtime, usually to create [dynamic web page](http://en.wikipedia.org/wiki/Dynamic_web_page) content or dynamic images used on websites or elsewhere. It can also be used for [command-line](http://en.wikipedia.org/wiki/Command-line) scripting and [client-side](http://en.wikipedia.org/wiki/Client-side) [graphical user interface](http://en.wikipedia.org/wiki/Graphical_user_interface) (GUI) applications. PHP can be deployed on most web servers, many [operating systems](http://en.wikipedia.org/wiki/Operating_system) and [platforms](http://en.wikipedia.org/wiki/Computing_platform), and can be used with many [relational database management systems](http://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS). Most [web hosting](http://en.wikipedia.org/wiki/Web_hosting) providers support PHP for use by their clients. It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use.

PHP acts primarily as a [filter](http://en.wikipedia.org/wiki/Filter_%28software%29), taking input from a file or stream containing text and/or PHP instructions and outputting another stream of data. Most commonly the output will be HTML, although it could be [JSON](http://en.wikipedia.org/wiki/JSON), [XML](http://en.wikipedia.org/wiki/XML) or [binary data](http://en.wikipedia.org/wiki/Binary_data) such as image or audio formats. Since PHP 4, the PHP [parser](http://en.wikipedia.org/wiki/Parser) [compiles](http://en.wikipedia.org/wiki/Compiler) input to produce [bytecode](http://en.wikipedia.org/wiki/Bytecode) for processing by the [Zend Engine](http://en.wikipedia.org/wiki/Zend_Engine), giving improved performance over its [interpreter](http://en.wikipedia.org/wiki/Interpreter_%28computing%29) predecessor.

Originally designed to create dynamic [web pages](http://en.wikipedia.org/wiki/Web_page), PHP now focuses mainly on [server-side scripting](http://en.wikipedia.org/wiki/Server-side_scripting),[[126]](http://en.wikipedia.org/wiki/PHP#cite_note-126) and it is similar to other server-side scripting languages that provide dynamic content from a web server to a [client](http://en.wikipedia.org/wiki/Client_%28computing%29), such as [Microsoft](http://en.wikipedia.org/wiki/Microsoft)'s [ASP.NET](http://en.wikipedia.org/wiki/ASP.NET), [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems)' [JavaServer Pages](http://en.wikipedia.org/wiki/JavaServer_Pages), and [mod\_perl](http://en.wikipedia.org/wiki/Mod_perl). PHP has also attracted the development of many [software frameworks](http://en.wikipedia.org/wiki/Software_framework) that provide building blocks and a design structure to promote [rapid application development](http://en.wikipedia.org/wiki/Rapid_application_development) (RAD). Some of these include [PRADO](http://en.wikipedia.org/wiki/PRADO_%28framework%29), [CakePHP](http://en.wikipedia.org/wiki/CakePHP), [Symfony](http://en.wikipedia.org/wiki/Symfony), [CodeIgniter](http://en.wikipedia.org/wiki/CodeIgniter), [Laravel](http://en.wikipedia.org/wiki/Laravel), [Yii Framework](http://en.wikipedia.org/wiki/Yii_Framework), and [Zend Framework](http://en.wikipedia.org/wiki/Zend_Framework), offering features similar to other [web application frameworks](http://en.wikipedia.org/wiki/List_of_web_application_frameworks).

The [LAMP architecture](http://en.wikipedia.org/wiki/LAMP_architecture) has become popular in the web industry as a way of deploying web applications. PHP is commonly used as the *P* in this bundle alongside [Linux](http://en.wikipedia.org/wiki/Linux), [Apache](http://en.wikipedia.org/wiki/Apache_HTTP_Server) and [MySQL](http://en.wikipedia.org/wiki/MySQL), although the *P* may also refer to [Python](http://en.wikipedia.org/wiki/Python_%28programming_language%29), [Perl](http://en.wikipedia.org/wiki/Perl), or some mix of the three. Similar packages, [WAMP](http://en.wikipedia.org/wiki/WAMP_%28software_bundle%29) and [MAMP](http://en.wikipedia.org/wiki/MAMP), are also available for [Windows](http://en.wikipedia.org/wiki/Microsoft_Windows) and [OS X](http://en.wikipedia.org/wiki/OS_X), with the first letter standing for the respective operating system. Although both PHP and Apache are provided as part of the Mac OS X base install, users of these packages seek a simpler installation mechanism that can be more easily kept up to date.

As of April 2007, over 20 million Internet domains had web services hosted on servers with PHP installed and mod\_php was recorded as the most popular [Apache HTTP Server](http://en.wikipedia.org/wiki/Apache_HTTP_Server) module. As of October 2010, PHP was used as the server-side programming language on 75% of all websites whose server-side programming language was known (as of February 2014, the percentage had reached 82%), and PHP was the most-used open source software within enterprises. [Web content management systems](http://en.wikipedia.org/wiki/Web_content_management_system) written in PHP include [MediaWiki](http://en.wikipedia.org/wiki/MediaWiki), [Joomla](http://en.wikipedia.org/wiki/Joomla), [eZ Publish](http://en.wikipedia.org/wiki/EZ_Publish), [SilverStripe](http://en.wikipedia.org/wiki/SilverStripe), [WordPress](http://en.wikipedia.org/wiki/WordPress), [Drupal](http://en.wikipedia.org/wiki/Drupal), [Moodle](http://en.wikipedia.org/wiki/Moodle), the user-facing portion of [Facebook](http://en.wikipedia.org/wiki/Facebook), and [Digg](http://en.wikipedia.org/wiki/Digg).

For specific and more advanced usage scenarios, PHP offers a well defined and documented way for writing custom extensions in [C](http://en.wikipedia.org/wiki/C_%28programming_language%29) or [C++](http://en.wikipedia.org/wiki/C%2B%2B). Besides extending the language itself in form of additional [libraries](http://en.wikipedia.org/wiki/Library_%28computing%29), extensions are providing a way for improving execution speed where it is critical and there is room for improvements by using a true [compiled language](http://en.wikipedia.org/wiki/Compiled_language). PHP also offers well defined ways for embedding itself into other software projects. That way PHP can be easily used as an internal [scripting language](http://en.wikipedia.org/wiki/Scripting_language) for another project, also providing tight interfacing with the project's specific internal [data structures](http://en.wikipedia.org/wiki/Data_structure).

PHP received mixed reviews due to lacking support for [multithreading](http://en.wikipedia.org/wiki/Multithreading_%28software%29) at the core language level, though using threads is made possible by the "pthreads" [PECL](http://en.wikipedia.org/wiki/PHP_Extension_Community_Library) extension.

## Security

In 2013, 9% of all vulnerabilities listed by the [National Vulnerability Database](http://en.wikipedia.org/wiki/National_Vulnerability_Database) were linked to PHP; historically, about 30% of all vulnerabilities listed since 1996 in this database are linked to PHP. Technical security flaws of the language itself or of its core libraries are not frequent (22 in 2009, about 1% of the total although PHP applies to about 20% of programs listed). Recognizing that programmers make mistakes, some languages include [taint checking](http://en.wikipedia.org/wiki/Taint_checking) to automatically detect the lack of [input validation](http://en.wikipedia.org/wiki/Data_validation) which induces many issues. Such a feature is being developed for PHP, but its inclusion into a release has been rejected several times in the past.

There are advanced protection patches such as [Suhosin](http://en.wikipedia.org/wiki/Suhosin) and [Hardening](http://en.wikipedia.org/wiki/Hardening_%28computing%29)-Patch, especially designed for web hosting environments.

There are certain language features and configuration parameters (primarily the default values for such runtime settings) that make PHP prone to security issues. Among these, [magic\_quotes\_gpc](http://en.wikipedia.org/wiki/Magic_quotes) and register\_globals configuration directives are the best known; the latter made any URL parameters become PHP variables, opening a path for serious security vulnerabilities by allowing an attacker to set the value of any uninitialized global variable and interfere with the execution of a PHP script. Support for "[magic quotes](http://en.wikipedia.org/wiki/Magic_quotes)" and "register globals" has been deprecated as of PHP 5.3.0, and removed as of PHP 5.4.0.

Another example for the runtime settings vulnerability comes from failing to disable PHP execution (via engine configuration directive) for the directory where uploaded images are stored; leaving the default settings can result in execution of malicious PHP code embedded within the uploaded images. Also, leaving enabled the dynamic loading of PHP extensions (via enable\_dl configuration directive) in a [shared web hosting](http://en.wikipedia.org/wiki/Shared_web_hosting) environment can lead to security issues.

Also, implied [type conversions](http://en.wikipedia.org/wiki/Type_conversion) that result in incompatible values being treated as identical against the programmer's intent can lead to security issues. For example, the result of "0e1234" == "0" comparison will be true because the first compared value will be treated as [scientific notation](http://en.wikipedia.org/wiki/Scientific_notation) of a number (0×101234) with value of zero. This feature resulted in authentication vulnerabilities in [Simple Machines Forum](http://en.wikipedia.org/wiki/Simple_Machines_Forum), [Typo3](http://en.wikipedia.org/wiki/Typo3) and [phpBB](http://en.wikipedia.org/wiki/PhpBB) when [MD5](http://en.wikipedia.org/wiki/MD5) [password hashes](http://en.wikipedia.org/wiki/Password_hashing) were compared. Instead, identity operator (===) should be used; "0e1234" === "0" results in false.

**SYSTEM DESIGN AND DEVELOPMENT**

**FILE DESIGN**

At the highest level a ﬁle system is a way to organize, store, retrieve, and manage information on a permanent storage medium such as a disk. File systems manage permanent storage and form an integral part of all operating systems. There are many different approaches to the task of managing permanent storage. At one end of the spectrum are simple ﬁle systems that impose enough restrictions to inconvenience users and make using the ﬁle system difficult. At the other end of the spectrum are persistent object stores and object-oriented databases that abstract the whole notion of permanent storage so that the user and programmer never even need to be aware of it. The problem of storing, retrieving, and manipulating information on a computer is of a general-enough nature that there are many solutions to the problem.

The primary functionality that all ﬁle systems must provide is a way to store a named piece of data and to later retrieve that data using the name given to it. We often refer to a named piece of data as a ﬁle. A ﬁle provides only the most basic level of functionality in a ﬁle system. A ﬁle is where a program stores data permanently. In its simplest form a ﬁle stores a single piece of information. A piece of information can be a bit of text (e.g., a letter, program source code, etc.), a graphic image, a database, or any collection of bytes a user wishes to store permanently. The size of data stored may range from only a few bytes to the entire capacity of a volume. A ﬁle system should be able to hold a large number of ﬁles, where “large” ranges from tens of thousands to millions.

The name of a ﬁle is metadata because it is a piece of information about the ﬁle that is not in the stream of bytes that make up the ﬁle. There are several other pieces of metadata about a ﬁle as well—for example, the owner, security access controls, date of last modiﬁcation, creation time, and size. The ﬁle system needs a place to store this metadata in addition to storing the ﬁle contents.

INPUT DESIGN

Input design is the process of converting user-originated inputs to a computer-based format. Input design is one of the most expensive phases of the operation of computerized system and is often the major problem of a system.

In the project, the input design is made in various web forms with various methods. For example, in the user creation form, the empty username and password is not allowed. The username if exists in the database, the input is considered to be invalid and is not accepted. Likewise, during the login process, the username is a must and must be available in the user list in the database. Then only login is allowed.

**OUTPUT DESIGN**

Output design generally refers to the results and information that are generated by the system for many end-users; output is the main reason for developing the system and the basis on which they evaluate the usefulness of the application.

DATABASE DESIGN

The database design is a must for any application developed especially more for the data store projects. Since the Logistic automation method involves retrieving the information in the table and produced to the administrator, proper handling of the table is a must. In the project, login table is designed to be unique in accepting the username and the length of the username and password should be greater than zero. The different users view the data in different format according to the privileges given.

**SYSTEM DEVELOPEMENT**

A Systems Development Life Cycle (SDLC) adheres to important phases that are essential for developers, such as [planning](http://en.wikipedia.org/wiki/Planning), [analysis](http://en.wikipedia.org/wiki/Analysis), [design](http://en.wikipedia.org/wiki/Design), and [implementation](http://en.wikipedia.org/wiki/Implementation), and are explained in the section below. A number of system development life cycle (SDLC) models have been created: waterfall, fountain, spiral, build and fix, rapid prototyping, incremental, and synchronize and stabilize. The oldest of these, and the best known, is the [waterfall model](http://en.wikipedia.org/wiki/Waterfall_model): a sequence of stages in which the output of each stage becomes the input for the next.

The waterfall model is a popular version of the systems development life cycle model for software engineering. Often considered the classic approach to the systems development life cycle, the waterfall model describes a development method that is linear and sequential. Waterfall development has distinct goals for each phase of development. Imagine a waterfall on the cliff of a steep mountain. Once the water has flowed over the edge of the cliff and has begun its journey down the side of the mountain, it cannot turn back. It is the same with waterfall development. Once a phase of development is completed, the development proceeds to the next phase and there is no turning back.

The advantage of waterfall development is that it allows for departmentalization and managerial control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process like a car in a carwash, and theoretically, be delivered on time. Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order, without any overlapping.

**TYPES OF TESTING**

**Unit Testing**

This is the first level of testing. The different modules are tested against the specifications produced during the integration. This is done to test the internal logic of each module. Those resulting from the interaction between modules are initially avoided. The input received and output generated is also tested to see whether it falls in the expected range of values. Unit testing is performed from the bottom up, starting with the smallest and lowest modules and proceeding one at a time.

The units in a system are the modules and routines that are assembled and integrated to perform a specific function. The programs are tested for correctness of logic applied and detection of errors in coding. Each of the modules was tested and errors are rectified. They were then found to function properly.

**Integration Testing**

In integration testing, the tested modules are combined into sub-systems, which are then tested. The goal of integration testing to check whether the modules can be integrated properly emphasizing on the interfaces between modules. The different modules were linked together and integration testing done on them.

**Validation Testing**

The objective of the validation test is to tell the user about the validity and reliability of the system. It verifies whether the system operates as specified and the integrity of important data is maintained. User motivation is very important for the successful performance of the system.

All the modules were tested individually using both test data and live data. After each module was ascertained that it was working correctly and it had been "integrated" with the system. Again the system was tested as a whole. We hold the system tested with different types of users. The System Design, Data Flow Diagrams, procedures etc. were well documented so that the system can be easily maintained and upgraded by any computer professional at a later

**System Testing**

The integration of each module in the system is checked during this level of testing. The objective of system testing is to check if the software meets its requirements. System testing is done to uncover errors that were not found in earlier tests. This includes forced system failures and validation of total system as the user in the operational environment implements it. Under this testing, low volumes of transactions are generally based on live data. This volume is increased until the maximum level for each transactions type is reached. The total system is also tested for recovery after various major failures to ensure that no data are lost during the breakdown.

**SYSTEM Implementation**

Implementation is the process that actually yields the lowest-level system elements in the system hierarchy (system breakdown structure). The system elements are made, bought, or reused. Production involves the hardware fabrication processes of forming, removing, joining, and finishing; or the software realization processes of coding and testing; or the operational procedures development processes for operators' roles. If implementation involves a production process, a manufacturing system which uses the established technical and management processes may be required.

The purpose of the implementation process is to design and create (or fabricate) a system element conforming to that element’s design properties and/or requirements. The element is constructed employing appropriate technologies and industry practices. This process bridges the system definition processes and the integration process.

System Implementation is the stage in the project where the theoretical design is turned into a working system. The most critical stage is achieving a successful system and in giving confidence on the new system for the user that it will work efficiently and effectively. The existing system was long time process.

The proposed system was developed using PHP. The existing system caused long time transmission process but the system developed now has a very good user-friendly tool, which has a menu-based interface, graphical interface for the end user. After coding and testing, the project is to be installed on the necessary system. The executable file is to be created and loaded in the system. Again the code is tested in the installed system. Installing the developed code in system in the form of executable file is implementation.

**CONCLUSION**

In conclusion, the Electrical Unit Tracking System Project provides an effective solution for monitoring and managing electricity consumption in residential and commercial settings. Through the integration of PHP with MySQL backend, the system offers a seamless experience for users to track their usage in real-time. By implementing automated email notifications when usage exceeds predefined thresholds, the project empowers users to take timely actions to control their consumption and potentially reduce costs.

Furthermore, the project's user-friendly interface allows for easy customization of threshold limits and notification preferences, catering to the specific requirements of individual users. This flexibility enhances user engagement and promotes active participation in energy management efforts.

Overall, the Electrical Unit Tracking System Project not only addresses the immediate need for efficient electricity usage monitoring but also contributes to broader goals of energy efficiency, cost savings, and sustainability. As society continues to prioritize environmental responsibility and resource conservation, projects like this play a vital role in promoting positive change and fostering a more sustainable future.

**SCOPE FOR FUTURE ENHANCEMENT**

Every application has its own merits and demerits. The project's modular coding structure facilitates seamless future enhancements and improvements to the system. Transitioning into a web-based application would enhance accessibility and usability, while refining the user interface for intuitiveness and aesthetics would improve user experience. Advanced monitoring and analytics capabilities, integration with smart devices, and personalized customization options represent promising avenues for future development, enabling users to gain deeper insights, automate processes, and tailor the application to their specific needs. By continually refining and expanding upon its functionality, the application stands to become an indispensable tool for monitoring, managing, and optimizing electricity consumption in both residential and commercial contexts.

**Bibliography**

Textual Reference

* **PHP book** by Vasvani (TMH publications).
* **Beginning PHP5** by WROX.
* **Informatics practices** by Sumita Arora.
* **Head First PHP & MySQL** by Lynn Beighley and Michael Morrison(O’Reilly)
* [**PHP Objects, Patterns, and Practice**](https://www.goodreads.com/book/show/1569843.PHP_Objects_Patterns_and_Practice) by [Matt Zandstra](https://www.goodreads.com/author/show/79192.Matt_Zandstra)
* [**PHP and MySQL Web Development**](https://www.goodreads.com/book/show/40540.PHP_and_MySQL_Web_Development) by [Luke Welling](https://www.goodreads.com/author/show/22734.Luke_Welling)
* **PHP Object-Oriented Solutions** by [David Powers](https://www.goodreads.com/author/show/17995.David_Powers)
* **Modern PHP: New Features and Good Practice** by [Josh Lockhart](https://www.goodreads.com/author/show/7819917.Josh_Lockhart)
* **Learning PHP Design Patterns** by [William Sanders](https://www.goodreads.com/author/show/8086513.William_Sanders)
* [**PHP 5 Objects, Patterns, and Practice**](https://www.goodreads.com/book/show/136867.PHP_5_Objects_Patterns_and_Practice) by [Matt Zandstra](https://www.goodreads.com/author/show/79192.Matt_Zandstra)
* [**PHP for the Web: Visual QuickStart Guide**](https://www.goodreads.com/book/show/9683472-php-for-the-web) by [Larry Ullman](https://www.goodreads.com/author/show/22735.Larry_Ullman)

Online Reference:

* [www.wikipedia.com](http://www.wikipedia.com)
* [www.w3schools.com](http://www.w3schools.com)
* <http://www.phpreferencebook.com/>
* <http://www.tizag.com/phpT/>
* <http://www.phpbuddy.com/>
* <http://www.php.net/>

**DATA FLOW DIAGRAM**

**LEVEL 0**

**voter**

uname

Admin

password

**LEVEL 1**

Admin

user

candidate

User

user

vote

**LEVEL 2**

**ER:**

user

View

View candidate

Admin

Login

Add candidate

view

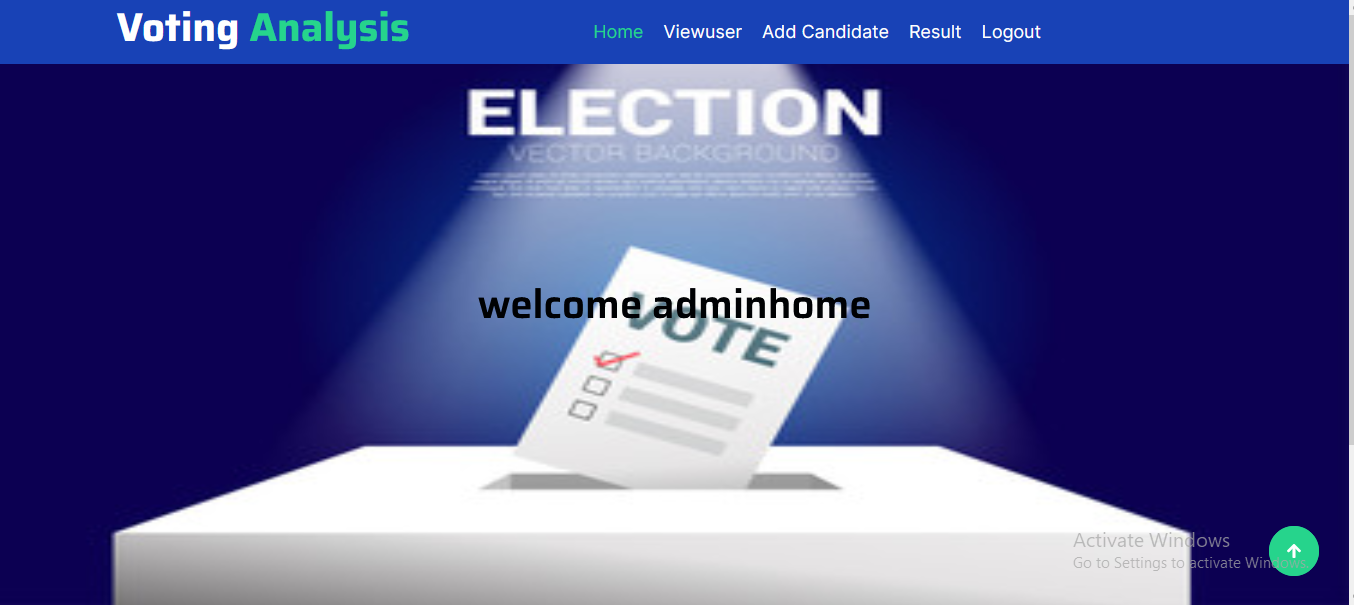
Result

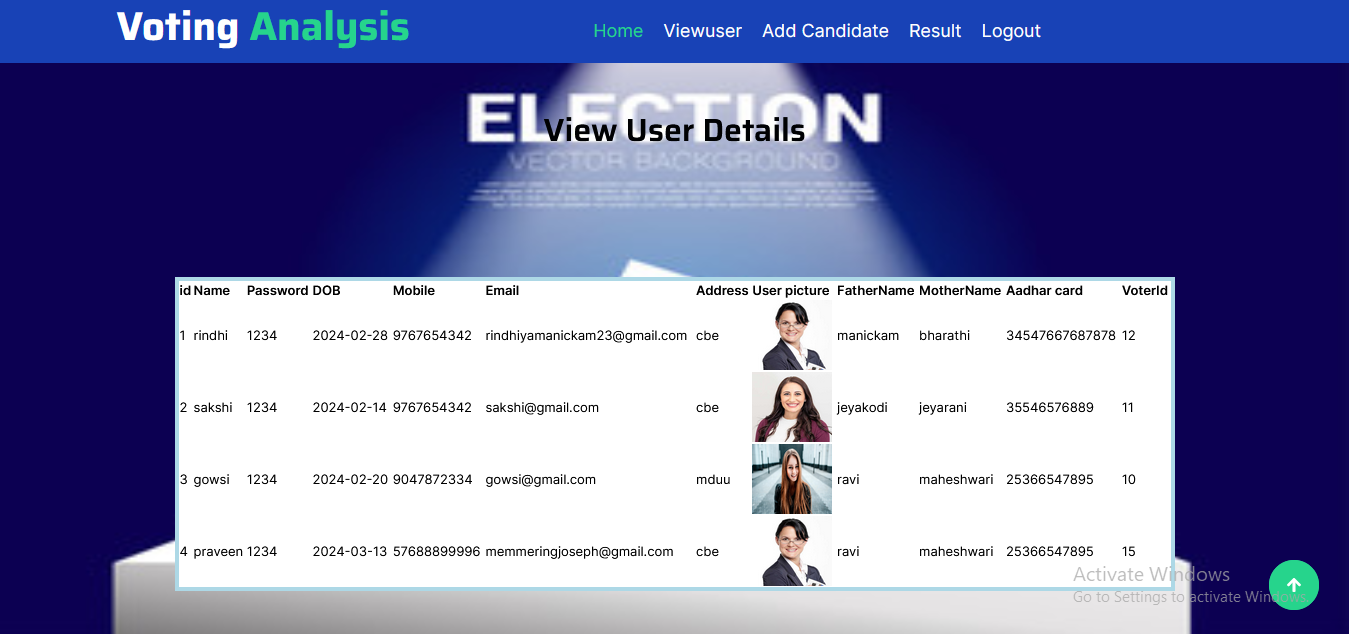
Login

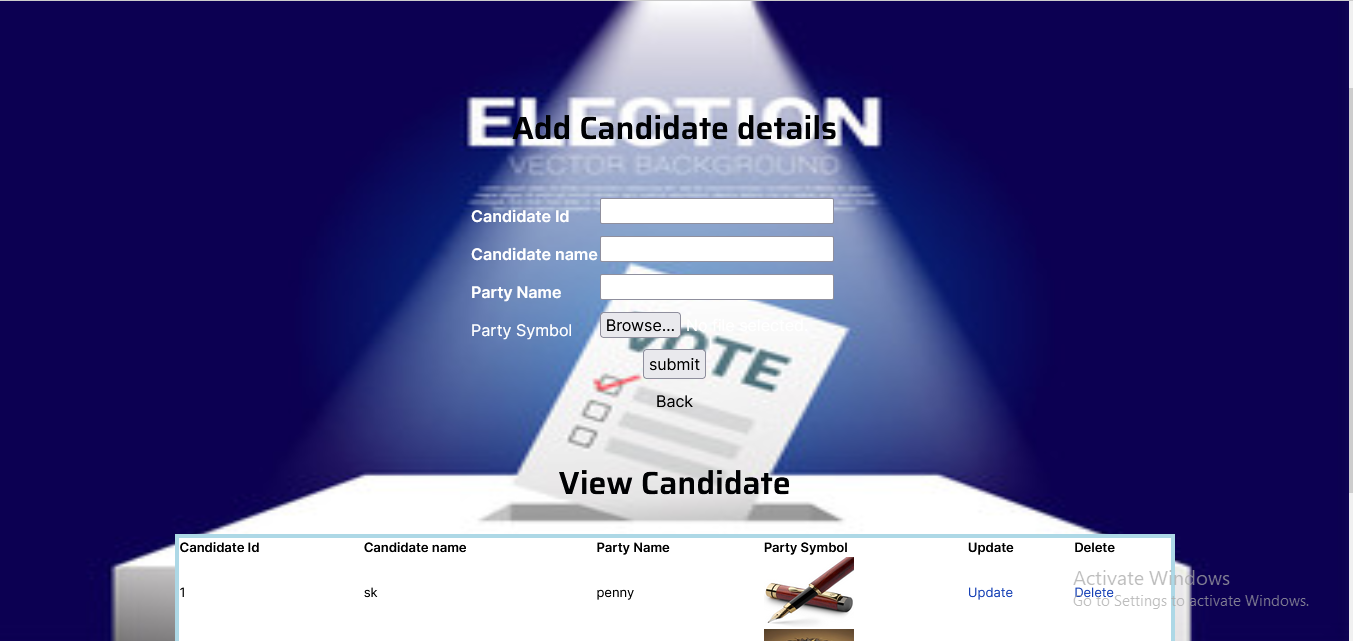
**ScreenShots:**

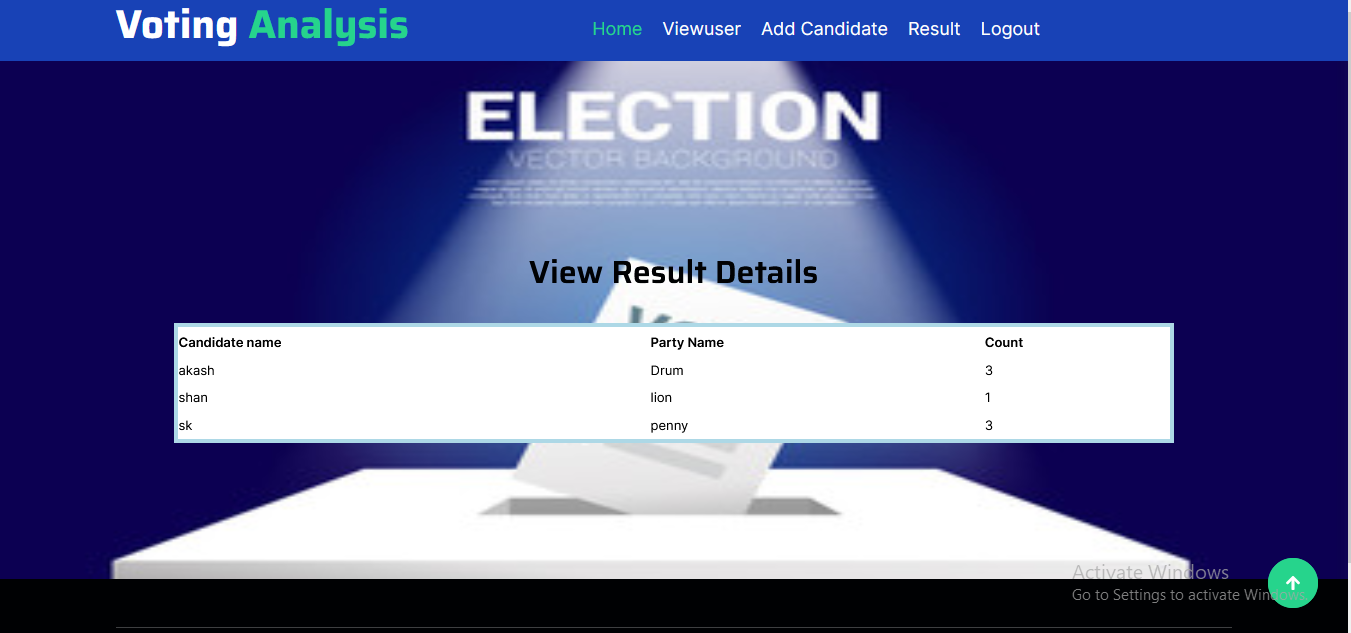


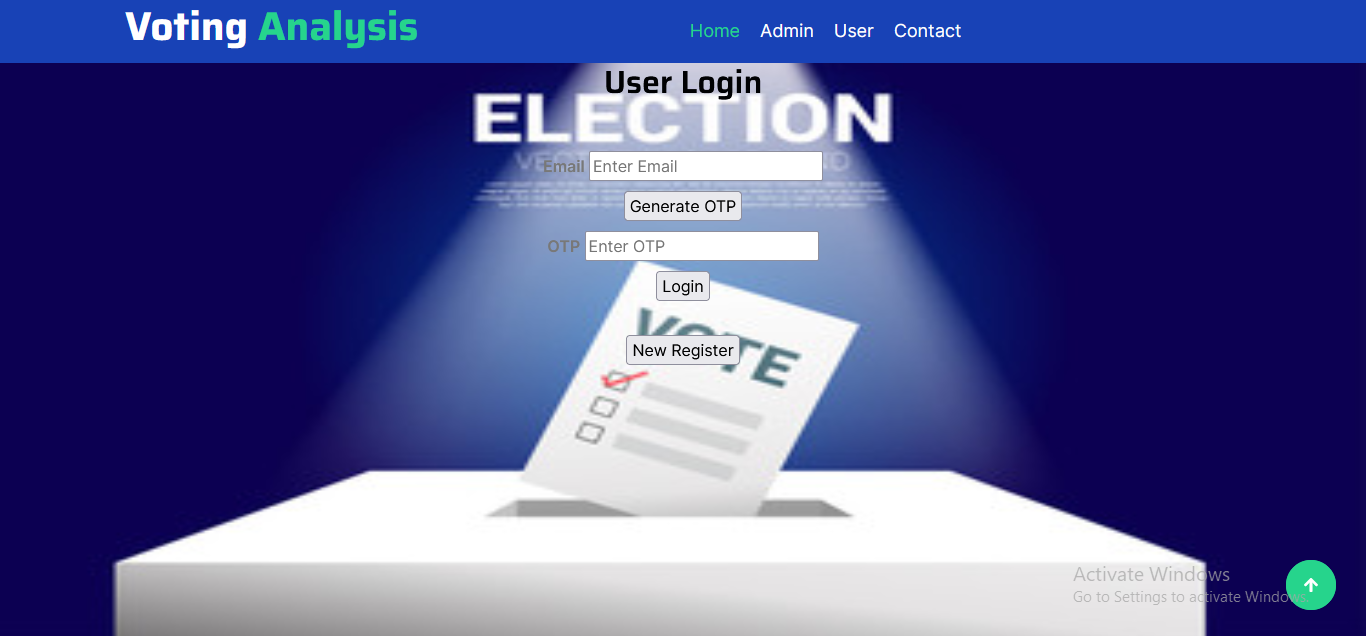












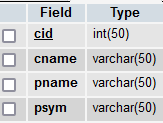


**Database: voter**

**Tables:**

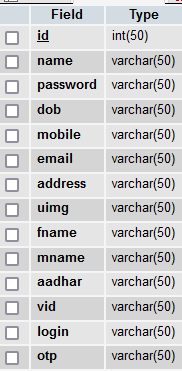
**Candidate:**

**Primary key:cid**

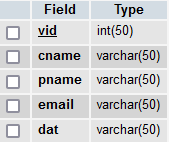
****

**User:**

**Primary key:id**



**Vote**:



**Code:**

**Addcandidate.php**

<?php

include('adminheader.php');

include('style.php');

include('config.php');

?>

<?php

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

{

$query = "INSERT INTO `candidate` VALUES (null,'".$\_POST['cname']."','".$\_POST['pname']."', '".$\_POST['psym']."')";

if(mysql\_query($query)){

echo ' SUCCESSFULLY';

echo '<script> alert("REGISTERED SUCCESSFULLY");</script>';

}

else{

echo 'NOT REGISTERD';

}

header("location:addcandidate.php");

// exit;

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Add Candidate details</title>

<style>

body {

background-image: url('vote.jpg');

background-repeat: no-repeat;

background-attachment: fixed;

background-size: 100% 100%;

color:white;

}

h2 {

text-align: center;

}

form {

text-align: center;

margin-top: 50px;

}

table {

margin: 0 auto;

}

label, input {

display: inline-block;

margin-bottom: 10px;

}

label {

width: 120px;

text-align: right;

margin-right: 10px;

}

</style>

</head>

<body>

<br>

<br>

<br>

<br>

<br>

<br>

<h2>Add Candidate details</h2>

<form method="post" action="">

<table>

<tr>

<td><b>Candidate Id</b></td>

<td><input type="text" name="cid" required></td>

</tr>

<tr>

<td><b>Candidate name</b></td>

<td><input type="text" name="cname" required></td>

</tr>

<tr>

<td><b>Party Name</b></td>

<td><input type="text" name="pname" required></td>

</tr>

<tr><td>Party Symbol</td>

<td><input type="file" name="psym" required></td></tr>

</table>

<input type="submit" name="submit" value="submit">

<br>

<a href="adminhome.php" style="color: black;">Back</a>

</form>

<form method="post">

<div >

<h2 align="center">View Candidate</h2>

<table border="2" cellspacing="6" class="displaycontent" width="1000" height="120" style="border:4px solid lightblue;" align="center">

<tr>

<th bgcolor=white><font color=black size=2>Candidate Id</font></th>

<th bgcolor=white><font color=black size=2> Candidate name</font></th>

<th bgcolor=white><font color=black size=2>Party Name</font></th>

<th bgcolor=white><font color=black size=2>Party Symbol</font></th>

<th bgcolor=white><font color=black size=2>Update</font></th>

<th bgcolor=white><font color=black size=2>Delete</font></th>

</tr>

<br>

<?php

$query = "select \* from candidate";

$result = mysql\_query($query) or die(mysql\_error());

while($row = mysql\_fetch\_assoc($result))

{

?>

<tr>

<td bgcolor=white><font color=black size=2><?php echo $row['cid']; ?></font></td>

<td bgcolor=white><font color=black size=2><?php echo $row['cname']; ?></font></td>

<td bgcolor=white><font color=black size=2><?php echo $row['pname']; ?></font></td>

<td bgcolor=white><font color=black size=2><img src=<?php echo $row['psym'];?> width=90 height=70></td>

<td bgcolor=white><font color=black size=2><a href="update.php?select=<?php echo $row['cid'];?>">Update</a></font></td>

<td bgcolor=white><font color=black size=2><a href="?delete=<?php echo $row['cid'];?>">Delete</a></font></td>

</tr>

<?php } ?>

</table>

</div>

</form>

<?php

if(isset($\_GET['delete']))

{

$query = "delete from candidate where cid='".$\_GET['delete']."'";

mysql\_query($query);

echo '<script>alert("Deleted");</script>';

header("location:addcandidate.php");

exit;

}

?>

<?php

include('footer.php');

?>

</body>

</html>

Adminhome.php

<?php

include('adminheader.php');

include('style.php');

?>

<!DOCTYPE html>

<html lang="en">

<style>

body {

background-image: url('vote.jpg');

background-repeat: no-repeat;

background-attachment: fixed;

background-size: 100% 100%;

}

h1 {

text-align: center;

}

h3 {

text-align: center;

}

</style>

<br>

<br>

<br>

<br>

<br>

<br>

<br>

<br>

<br>

<h1>welcome adminhome</h1>

<br>

<br>

<br>

<br>

<br>

<br>

<br>

<br>

<br>

<br>

<br>

<br>

<br>

<br>

</head>

</body>

<?php

include('footer.php');

?>

</html>