INTRODUCTION

**INTRODUCTION**

The aim of our project ‘ServerZilla’ is to encourage transfer files between remote host and FTP server. And this project manages files in both remote host and FTP server. Currently remote administrator uses TFTP (Trivial File Transferring Protocol) to connect remote host to FTP server. To execute TFTP the administrator needs to spawn a shell using back connect or SSH (secured shell ).This application is based on Command Line Interface (CLI) and have several commands to encourage transfer files between remote host and FTP server which is very difficult for a normal user .Hence by doing this project reduce this problem. This project uses a web interface which will be user friendly

OBJECTIVES

**OBJECTIVES**

* Enable the administrator to connect the web server with FTP server.
* Provide impressive and user-friendly interface.
* Save time of administrator.
* Transfer files between web server and FTP server.
* Enable administrator to browse the files in web server and FTP server.
* Enable administrator to manage the files in web server and FTP server.

SYSTEM ANALYSIS

**INTRODUCTION**

Software systems analysis is a field in which analysts continually learn new techniques and approaches to properly capture, maintain, understand, and develop more efficient and effective software systems. We begin this research area overview by defining systems, systems analysis, and modeling. In subsequent sections, we focus on data and behavior representation of the system under study, prototyping, and formalism. Finally, we introduce some of the current work such as form-oriented analysis, fisheye views to support system analysis, and extreme programming and consider future work on software systems analysis such as extreme non-programming and new challenges for conceptual modeling.

The project ‘ServerZilla’ is aimed to establish connection between web server and FTP server, and enable administrator to manage the files in both web server and FTP server. To do this analyzed the existing systems like FileZilla and TFTP. But found that these existing system have some drawbacks. To overcome these drawbacks proposed the new system named ‘ServerZilla’.

**PRE LIMINARY STUDY**

Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. It is a problem solving activity that requires intensive communication between system users and system developers. It does various feasibility studies. In these studies a rough figure of the system activity can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be taken. Preliminary study also identifies the method collection to be followed.

As the preliminary study was conducted, an initial picture about the system working was obtained. From the information received from the study, data collection methods are identified. Even in the first investigation itself, the drawback of the existing system could be identified.

**REQUIREMENT GATHERING AND ANALYSIS**

The analyst starts the requirements gathering analysis activity by collecting all information from the customer, which could be used to develop the requirements of the system. He then analyzes the collected information to obtain a clear and thorough understanding of the product to be developed, with a view of removing all ambiguities and inconsistencies from the initial customer perception of the problem. This may sound like a simple task. However, in practice it is difficult to gather the necessary information and to form an unambiguous understanding problem. Now we elaborate the two main activities involved in the requirements gathering and analysis phase.

**REQUIREMENTS GATHERING**

This activity typically involves interviewing the end users and customers and studying the existing documents to collect all possible information regarding the system. If the project involves automating some existing procedure, the task of the system analyst becomes a little easier as we can immediately obtain the input and the output data formats and the details of the operational data procedures.

The required information related to develop an FTP client is gathered by analyzing existing system like FileZilla and TFTP. The features of FTP clients are also elicited from existing system. The other details of FTP client are gathered from Internet resources.

**ANALYSIS OF THE GATHERED REQUIREMENTS**

The main purpose of this activity is to clearly understand the exact requirements of the customer. The following basic questions pertaining to the project should be clearly understood by the analyst in order to obtain a good grasp of the problem:

* What is the problem?
* Why is it important to solve the problem?
* What are the possible solutions to the problem?
* If there is external software or hardware with which the developed Software has to interface, then what exactly would the data interchange Formats with the external system be?
* What are the likely complexities that might arise while solving the problem?
* What exactly are the data inputs to the system and what exactly are the data outputs to the system?

After the analyst has understood the exact customer requirements, he proceeds to identify and resolve the various requirement problems. The most important requirement problem that the analyst has to identify and eliminate is the problems of anomalies, inconsistencies and incompleteness

**SOFWARE REQUIREMENT SPECIFICATION (SRS)**

After the analyst has collected all the required information regarding the software to be developed, and has removed all completeness, inconsistencies and anomalies from the specification I start to systematically organize the requirements in the form of an SRS document. The SRS document usually contains all the user requirements in an informal form.

Among all the documents produced during a software development life cycle, writing SRS document is probably the toughest. One reason behind this difficulty is that the SRS document is expected to cater to the needs of a wide variety of audience. Different people need the SRS document for different purposes. Some of the important categories of users of the SRS document and their needs are as follows:

**Users, customers and marketing personnel:** The goal of this set of audience is to ensure that the system as described in the SRS document will cater to their needs.

**Software developers:** The software developers refer to the SRS document to make sure that they develop exactly what the customer requires.

**Test engineers:** Their goal is to ensure that the requirements are understandable from a functionality point of view, so that they can test the software and validate its working. They need the functionality to be clearly described, and input and output data to be identified precisely.

**User documentation writers:** Their goal in reading the SRS document is to ensure that they understand well enough to be able to write the user manuals.

**Project manager:** They want to ensure that they can estimate the cost easily by referring to the SRS document and that it contains all the information required planning the project well.

**Maintenance engineers:** The SRS document helps the maintenance engineers to understand the functionalities of the system. A clear knowledge of the functionality can help them to understand the design code. Also, the requirement knowledge would enable them to determine what modifications to the system’s functionality would be needed for a specific purpose.

An SRS document should clearly specify

* **Functional requirements**
* **Nonfunctional requirements**
* **Goals of implementation**

**SOFTWARE FEATURES**

**PLATFORM INDEPENDENT**

PHP stands for the PHP HyperText Preprocessor. PHP is server side scripting language. Server Side means that any user request a PHP page in the browser. Then the request is send to the server and the server then compile the code at the server side and convert is to html and return to the client or user. PHP is one of the most powerful server side scripting language. As it is freeware, Open Source, Platform Independent.

PHP is free to use as you have to pay nothing. It is open source language to so that you can use the PHP Script and make it more better. and one of the most important is that it is platform Independent.

**OPEN SOURCE**

PHP is freely available for use. The community of open source PHP developers provides technical support and is constantly improving updating the core PHP functionalities. PHP is available at free of cost under PHP General Public License and most of its associative required software’s like MySQL, Text Editors and Apache Server are also freely available, so it proves very cost effective for the developers.

**CROSS-PLATFORM**

PHP provides high compatibility with leading operating systems and web servers such as thereby enabling it to be easily deployed across several different platforms. PHP scripts can run across operating systems such as Linux, Windows, Solaris, OpenBSD, Mac OSX etc and also provide support for all major web servers such as Apache, IIS, iPlanet etc.

**SECURITY**

PHP offers security as well that helps prevent malicious attacks. These security levels can be adjusted in the .ini file.

FEASIBILITY STUDY

**SYSTEM FEASIBILITY**

Feasibility study is a test of a system proposal according to its workability, impact on the organization, ability to meet users need and effective use of resources. A feasibility study of the proposed system was carried out to see whether it was beneficial to the organization or not. The feasibility study involves. The study will decide whether the proposed system will be cost effective from a business point of view and if it can be developed using the given existing budgetary constraints.

All projects are feasible when given unlimited resources and infinite time. It is both necessary and prudent to evaluate the feasibility of a project at the earliest possible time. A feasibility study is not warranted for systems in which economic justification is obvious, technical risk is low, few legal problems are expected and no reasonable alternative exists. An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies.

The study will decide if the proposed system will be cost effective from the business point of view and if it can be developed in the given existing budgetary constraints. The feasibility study should be relatively cheap and quick. The result should inform the decision of whether to go ahead with a more detailed analysis.

Feasibility study may be documented as a separated report to higher officials of the top-level management and can be included as an appendix to the system specification. Feasibility and risk analysis is related in many ways. If there is more project risk then the feasibility of producing the quality software is reduced. The study is done in these phases

* **Operational feasibility**
* **Technical feasibility**
* **Economical feasibility**
* **Behavioral feasibility**
* **Software feasibility**
* **Hardware feasibility**

**OPERATIONAL FEASIBILITY:**

Proposed projects are beneficial only if they can be turned into information systems that will meet the organization's operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to implementation? Here are questions that will help test the operational feasibility of a project:

* Is there sufficient support for the project from management?
* Are current business methods acceptable to the users?
* Have the users been involved in the planning and development of the project?
* Will the proposed system cause harm?

The purpose of the operational feasibility study is to determine whether the new system will be used if it is developed and implemented. And whether there will be resistance from users that will undermine the possible application benefits. There was no difficulty in, implementing the system and the proposed system is so effective, user friendly and functionally reliable so that the users in the company will find that the new system reduce their hard-steps. If the users of the system are fully aware of the internal working of the system then the users will not be facing any problem in running the system.

**TECHNICAL FEASIBILITY:**

Technical feasibility study checks the new system is technically very efficient. Also used for know the development of the system in the new technology or not. This involves financial consideration to accommodate technical enhancement.

A study of function, performance and constraints may improve the ability to create an acceptable system. Technical feasibility is frequently the most difficult area to achieve at the stage of product engineering process.

Considering that are normally associated with the technical feasibility include

* Development risk
* Resource availability
* Technology

Technical feasibility study deals with the hardware as well as software requirements. The scope was whether the work for the project is done with the current equipments and the existing software technology has to be examined in the feasibility study. The outcome was found to be positive. In the proposed system, data can be easily stored and managed using database management system software. The reports and results for various queries can be generated easily. Therefore, the system is technically feasible.

**ECONOMICAL FEASIBILITY:**

The project is to be developed in PHP. It is free and open source. The project does not involve any special hardware other than standard specifications. So it is economically feasible to implement the project within the available budget.

**BEHAVIORAL FEASIBILITY:**

People are inherently resistant to changes and computer is known for facilitating the changes. An estimate should be made of how strongly the user staff reacts towards the developments of the computerized system. In the existing system more manpower is required and time factor is more. In the proposed system, both man power and time factors are reduced and also unnecessary burden is reduced. Thus, the remaining people are made to engage in some other important work. Therefore, the system is behaviorally feasible.

**SOFTWARE FEASIBILITY**:

Even though software is developed in a very high software environment, it will be supported by many other platforms and environments with minimum changes.

**HARDWARE FEASIBILITY:**

The software can be developed with resource already existing. Here the consideration is that the existing hardware resources support the technologies that are to be used by the new system. No hardware was newly bought for the project and hence. Software is said to achieve hardware feasibility.

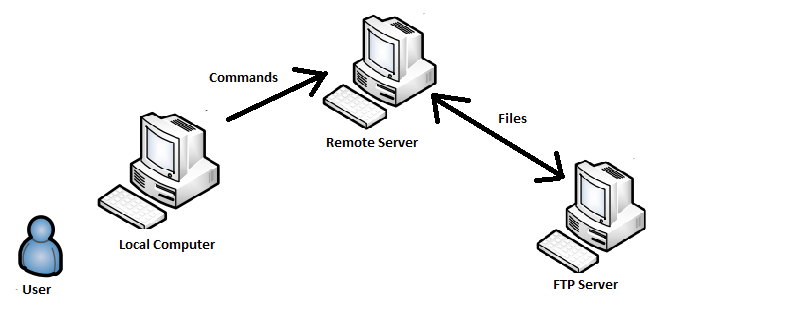
**EXISTING SYSTEM**

The current existing system is ‘FileZilla’ and ‘TFTP’. FileZilla is open source software. To connect a remote host to a FTP server is not possible using FileZilla .TFTP can connect a remote host and FTP server, but it is very hard to use.

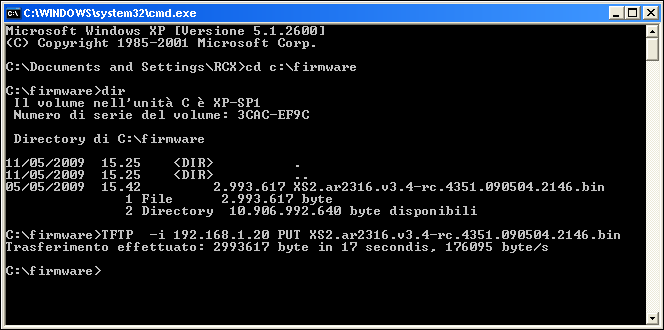
**LIMITATIONS**

1. Time consumption
2. Not user friendly
3. Command line interface
4. Hard to understand

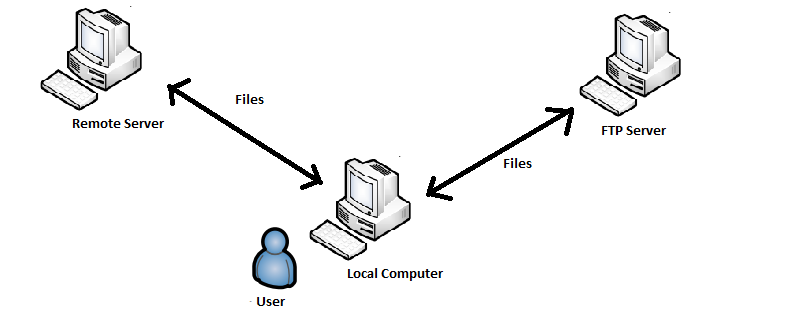
**TFTP Working**

****

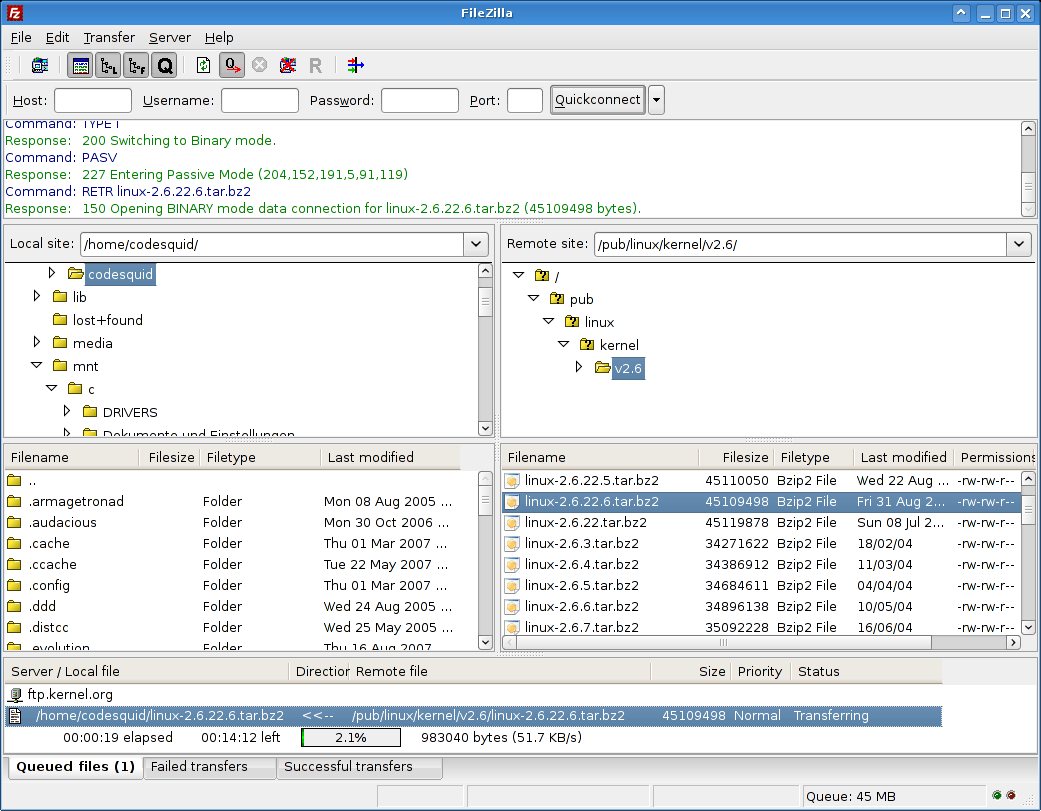
**TFTP Screenshot**



**FileZilla Working**

****

**FileZilla Screenshot**



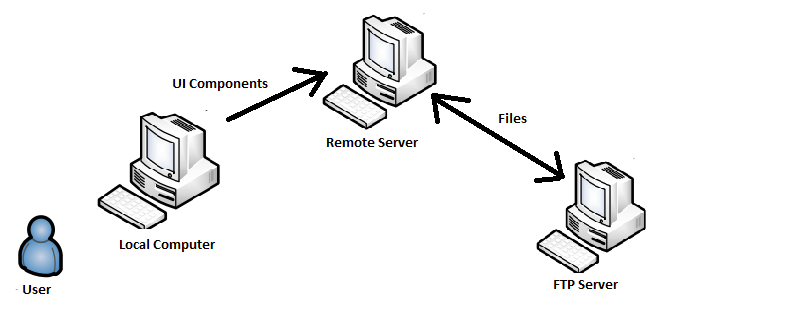
**PROPOSED SYSTEM**

The proposed is used to encourage file transfer between remote host and FTP server user friendly. This project reduces the limitations of existing system. The project uses a user friendly web interface which will be easy to understand and use.

**ADVANTAGES OF PROPOSED SYSTEM**

1. User friendly
2. Web interface
3. Easy to understand
4. Cross platform
5. Less time consumption

**Proposed System (ServerZilla) Working**



**OPERATING ENVIRONMENT**

Operating environment for the ‘ServerZilla’ is as listed below,

* Web Server
* PHP environment
* Database

**EXTERNAL INTERFACES REQUIRED**

* User Interfaces
  + Front-end: HTML, CSS, JavaScript, jQuery, Bootstrap
  + Back-end: MySQL
* Hardware Interfaces
  + Windows XP/7/8, Linux
  + Web browser which support CSS and JavaScript

***Software Interfaces***

|  |  |
| --- | --- |
| **Software Used** | **Description** |
| ***Operating System*** | It will support all operating systems which installed a web server and PHP. |
| ***Database*** | We have chosen MySQL. |
| ***PHP*** | We have implemented the project in PHP. |

SYSTEM SPECIFICATION

**HARDWARE DESCRIPTION**

The selection of hardware is very important in the existence and proper working of any software .When selecting hard ware, size and requirements are also important.

**MINIMUM REQUOIRMENTS:**

Processor : Pentium II class,450 MHz

RAM : 256 MB

Hard Disk Drive : PC with 3GB

**THE PROPOSED SYSTEM IS DEVELOPED ON**:

Operating System : Windows 8

Processor : Intel i5

Hard Disk Drive : 100GB

**SOFTWARWARE DESCRIPTION:**

Operating System : Windows XP SP3 or Later

Front-End : PHP, HTML, jQuery, CSS, JavaScript, Bootstrap.

Back-End : MySQL 5.0

Tools : Aptana studio 3

**TECHNOLOGIES USED**

Software selection is an important work in a project development cycle. Software must be selected in accordance with the application and the latest technology available. PHP is the best choice. A user can easily run this software. MySQL Server is used as the database.

**PHP**

PHP, which stands for “PHP: Hypertext Preprocessor” is widely-used Open Source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. Its syntax draws upon C, Java, and Perl, and is easy to learn. The main goal of the language is to allow web developers to write dynamically generated web pages quickly.

PHP is a powerful language and the interpreter, whether included in a web server as a module or executed as a separate CGI binary, is able to access files, execute commands and open network connections on the server. These properties make anything run on a web server insecure by default. PHP is designed specifically to be more secure language for writing CGI programs than Perl or C, and with correct selection of compile-time and runtime configuration options, and proper coding practices, it can give you exactly the combination of freedom and security you need.

**MYSQL**

MySQL is the world's most popular open source database. With its proven performance, reliability and ease-of-use, MySQL has become the leading database choice for web-based applications, used by high profile web properties including Facebook, Twitter, YouTube, Yahoo! and many more.

Oracle drives MySQL innovation, delivering new capabilities to power next generation web, cloud, mobile and embedded applications.

**BOOTSTRAP**

Bootstrap is a free and open-source collection of tools for creating websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. The bootstrap framework aims to ease web development. Bootstrap, originally named Twitter Blueprint, was developed by Mark Otto and Jacob Thornton at Twitter as a framework to encourage consistency across internal tools.

Bootstrap is a front end, that is an interface between the user and the server-side code which resides on the "back end" or server. And it is a web application framework, that is a software framework which is designed to support the development of dynamic websites and web applications.

**JQUERY**

JQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. jQuery is the most popular JavaScript library in use today. jQuery is free, open-source software licensed under the MIT License.

jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, theme-able widgets. The modular approach to the jQuery library allows the creation of powerful dynamic web pages and web applications.

**WEB BROWSERS**

A web browser is a software application that enables a user to display and interact with text, images, videos, music and other information typically located on a Web page at a website on the World Wide Web or local area network. Text and images on Web page can contain hyperlinks to other Web pages’ at the same or different website. Web browser allow a user too quickly and easily access information provided on many Web pages by traversing these links.

SYSTEM DESIGN AND MODELING

**INTRODUCTION**

Design is meaningful engineering and creative process; a good design is the key to effective system. The term “Design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. Various design features are followed to develop the system. Computer software design like engineering design approaches in other disciplines changes continually as new methods, better analysis, and broader understanding evolve. It refers to the technical specification that will be applied in implementing the candidate system. It includes as the construction of programs and program testing. The design specification describes the features of the system, the components or elements of the system and their appearance to end-users.

System designing is the most creative and challenging phase. Using one of a number of a design method the design step produces a data design, an architectural design and a procedural design. Preliminary design is concerned with transformation requirements to data and software architectures.

In system design high-end decisions are taken regarding the basic system architecture, platforms and tools to be used. The system design transform a logical representation of what a given system is required to be into the physical specification. Design starts with the systems requirements specification and covered it into a physical reality during the development. Detail design focus on refinements to architectural representation that lead to detailed data structure and algorithmic representation for software. Import design factors such, as reliability, response time, throughput of the system, maintainability, expandability, etc should be taken into account. The data design transforms the information domain model created during analysis into the data structures that will be required to implement the software.

**DESIGN PHASE**

Design phase covers following,

* Reviews the existing systems which used to transfer files between web server and FTP server.
* Determine functionalities that is to be done by proposed system.
* Design user interface.
* Prepare input specifications that determine the format content and most of the input functions.
* Prepare output specification.

Activities in design phase,

* **Allocation function:**

The dataflow diagrams or information oriented and process oriented system flow charts, prepared during the study phase and reviewed and expanded in order to allocate function between manual tasks, equipment functions, and computer program functions.

* **Reference manual identification:**

Reference manuals required by user personnel, programmers, and equipment operations are identified

* **Equipment functions definition:**

The functions to be performed by hardware (rather than by computer programs or manual operations) are defined.

* **Equipment specification:**

The hardware configuration used to convert input data to meaningful output information is described.

* **Computer program function definition:**

The specific functions of the computer program component of the overall system are defined, and design requirements for external system inputs are established.

* **Database design:**

The storage requirements of all the data elements on which the computer programs operate are calculated, taking into account the size and volume of the records to be stored and the methods of file organization and access.

* **Computer program design:**

The computer programs, which make up the overall computer program component, share the system database.

* **System test requirements definition:**

Requirements are established for the test necessary to verify the performance of the entire computer based system. This is accomplished in parallel with the activities associated with system design.

* **Computer program test requirement definition:**

Requirements also are determines for the tests necessary to verify the performance of the major computer programs. This is done after the definition of the system test requirements.

**OBJECT ORIENTED DESIGN**

The design of object-oriented software require the definition of multi-layered software architecture, the specification of subsystems that perform required functions and provide infrastructure support, a description of objects (classes) that form the building block of the system, and a description of the communication mechanisms that allow data to flow between layer, subsystems and objects. Object-oriented design accomplishes all the three things. The four layers of the object-oriented design pyramid are:

***The subsystem layer:*** This layer contains a representation of each of the subsystems that enable the software to achieve its customer-defines requirements and to implements the technical infrastructure that supports customer requirements.

***The class and the object laye*r:** This layer contains the class hierarchies and enables the system to be created using generalizations and increasingly more targeted specializations. This layer also contains representations of each object.

***The message layer:*** The layer contains the design details that enable each object to communicate with its collaborators. This layer establishes the external and internal interfaces for the system.

***The responsibility layer:*** This layer contains the data structure and algorithmic design for all attributes and operations for each object.

**SYSTEM DESIGN PROCESSES**

System design develops the architectural details required to build a system or products. The system design closes encompasses the following activities.

Partition the analysis model into subsystems.

* Identify the concurrency that is detected by the problem.
* Allocate subsystems to processors and tasks.
* Develop a design for the users interface.
* Choose a basic strategy for implementing data management.
* Identify global resources and control mechanisms required to access them.
* Design an appropriate control mechanism for the system. Including task management.
* Consider how boundary conditions should be handled.
* Review and consider trade-offs

**OBJECT DESIGN PROCESS**

During this process we focus on the description of data structures that implement class attributes, algorithms that implement operations and messages that enable collaboration and object relationships. A variety of representations contained in the analysis model and system design provide a specification for all operations and attributes. An algorithm is a simple computation or procedural sequence that can be implemented as a self contained software module. Data structures are designed concurrently with algorithms. Since operations invariably manipulated the attributes of a class, the design of the data structure that best reflect the attributes will have a strong bearing on the algorithmic design of the corresponding operations.

**MODULAR DESIGN**

This is the low-level design during which each module is designed in detail. Detailed design is concerned with specifying algorithmic detail and user interface specifications for the procedures in each module. The component level design transforms structural elements of the software architecture into procedural descriptions of software components. A design should lead to components that exhibit independent functional characteristic.

In this project detailed design of each module is done with component level design and these include mainly

* Administrator module

The application ServerZilla allows the administrator to connect the web server to FTP server and transfer files between them. Also the application allows file management like browse through directories, rename files and directories, delete files and directories, etc. Due to security problems the application allows only one user to use the application. So there exists only one module: Administrator.

**ADMIN**

This is the one and only one module in the project ‘ServerZilla’. He can manage the whole system. The administrator can transfer files between web server and FTP server. Also he can manage files in both web server and FTP server. The administrator can do,

* Transfer files between web server and FTP server
* Browse through directories
* Rename files in web server and FTP server
* Delete files and directories in web server and FTP server

**FUNDAMENTAL DESIGN CONCEPTS**

Fundamental design concepts provide the software designer with a foundation from which more sophisticated design methods can be applied. Fundamental design concepts provide the necessary framework for “getting it right”.

**ABSTRACTION**

Abstraction permits one to concentrated on a problem at some level of generalization without regard to irrelevant low level details, use of abstraction also permits one to work with concepts and terms that are familiar in the problem environment without having to transform them to an unfamiliar structure, Two types of abstractions are there, one is procedural abstraction and data abstraction. A procedural abstraction is a named sequence of instructions that as a specific and limited function. A data abstraction is a named collection of data that describe a data object.

**MODULARITY**

Modularity is the single attribute software that allows a program to be intellectually manageable. Software architecture embodies modularity, that is, software is divided into named and addressable components, called modules. Which are integrated to satisfy problem requirements?

**SOFTWARE ARCHITECTURE**

Software architecture alludes to “the overall structure of the software and the ways in which that structure provides conceptual integrity for the system”. Control hierarchy also called program structure”, represents the organization of control. The tree structure used to represent the control hierarchy.

**DATA STRUCTURE**

Data structure is a representation of logical relationship among individual elements of data because the structure of information will invariably effects the final procedural design, data structure is very important as the program structure to the representation of the software architecture, data structure detects the organization, methods of access degree of associatively, and processing alternatives for information. The organization and complexity of a data structure are limited only by the ingenuity of the designer. Scalar item array and linked list are some of the representations of the data structure.

**SOFTWARE PROCEDURAL**

Program structure defines control hierarchy without regard to the sequence of processing and decisions. Software procedural focuses on the processing details of each module individually. Procedural must provide a precise specification of processing, including sequence of events, exact, decision points, repeatitive operations and even data organization/structure. Information hiding suggests that modules be “charecterised by design decisions that hide from all others”.

**DATA FLOW DIAGRAM (DFD)**

A Data Flow Diagram (DFD) or bubble chart is a graphical tool for structured analysis, it was De Marco (1978) and Gane and Sarson (1979) who invented DFD. DFD models a system by using external entities from which data flows to a process, which transforms the data and creates, output-data-flows which go to other processes or external entities or files. Data in files may also flow to processes as inputs.

There are various symbols used in a DFD. Bubbles represent the processes. Named arrows indicate the data flow. External entities are represented by rectangles and are outside the system such as vendors or customers with whom the system interacts. They either supply or consume data. Entities supplying data are known as sources and those that consume data are called sinks. Data are stored in a data store by a process in the system. Each component in a DFD is labeled with a descriptive name. Process names are further identified with a number.

DFD’s can be hierarchically organized, which help in partitioning and analyzing large systems. As a first step, one dataflow diagram can depict an entire system which gives the system overview. It is called Context Diagram of level 0 DFD. The Context Diagram can be further expanded.

The successive expansion of a DFD from the context diagram to those giving more details is known as leveling of DFD. Thus a top down approach is used, starting with an overview and then working out the details.

The main merit of DFD is that it can provide an overview of what data a system would process, what transformation of data are done, what files are used, and where the results flow.

**BASIC DFD SYMBOLS:**

In the DFD, there are four symbols, they are as follows:

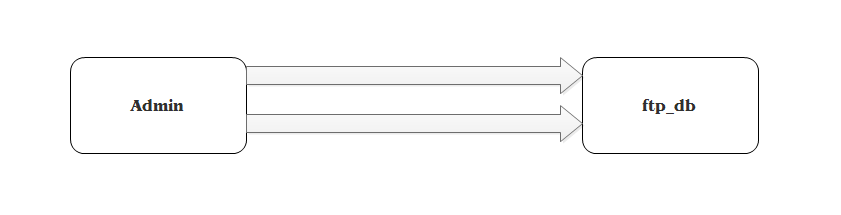
A data flow is route, which enables packets of data to travel from one point to another. Data may flow from a processor and from data store or process. An arrow line depicts the flow, with arrow head pointing in the direction of the flow.

A process represents transformation where incoming data flows are changed in to outgoing data flows.

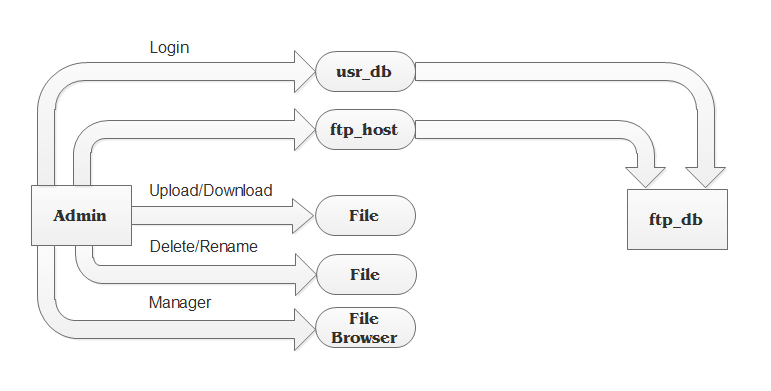
A data store is a repository of data that is to be stored for use by one or more processes. It may be as simple as buffer or queue or sophisticated as relational database. They should have clear names. If a process merely uses the content of store and does not alter it, the arrow head goes only from the store to the process. If a process alters the details in the store then a double-headed arrow is used.

A source or sink is a person or part of an organization which enter or receives information from the system. But it is considered to be outside of the contest of data flow model.

***DFD (Level 0)***



***DFD (Level 1)***



**ENTITY-RELATIONSHIP (E-R) DIAGRAM**

In software engineering, an entity–relationship model (ER model) is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as a relational database. The main components of ER models are entities (things) and the relationships that can exist among them.

Entity–relationship modeling was developed by Peter Chen and published in a 1976 paper. However, variants of the idea existed previously, and have been devised subsequently such as super type and subtype data entities and commonality relationships.

Introduction: An entity–relationship model is a systematic way of describing and defining a business process. The process is modeled as components (entities) that are linked with each other by relationships that express the dependencies and requirements between them, such as: one building may be divided into zero or more apartments, but one apartment can only be located in one building. Entities may have various properties (attributes) that characterize them. Diagrams created to represent these entities, attributes, and relationships graphically are called entity–relationship diagrams.

An ER model is typically implemented as a database. In the case of a relational database, which stores data in tables, every row of each table represents one instance of an entity. Some data fields in these tables point to indexes in other tables; such pointers represent the relationships.

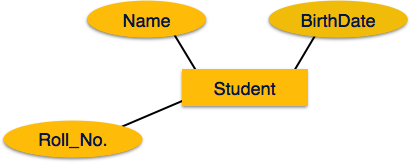
**Entity**

Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

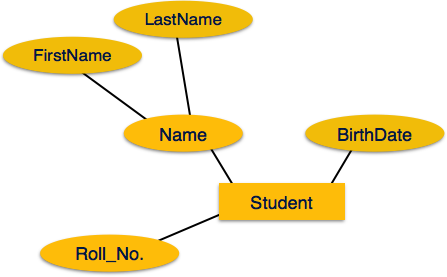
Entities in a school database

**Attributes**

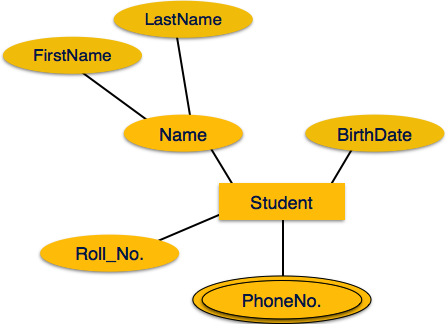
Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).



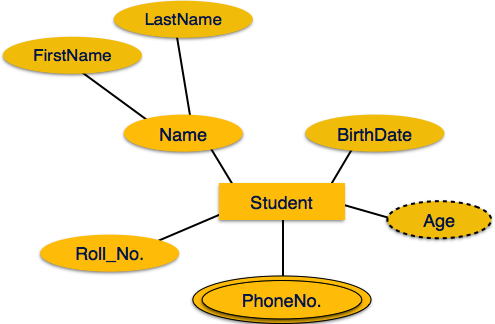
If the attributes are composite, they are further divided in a tree like structure. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse.



Multi valued attributes are depicted by double ellipse.



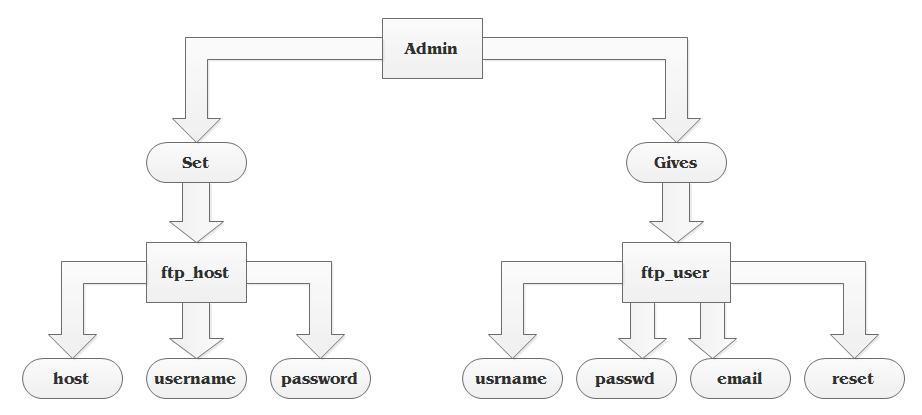
Derived attributes are depicted by dashed ellipse.



**Relationship**

Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line.

***E-R Diagram of ServerZilla***

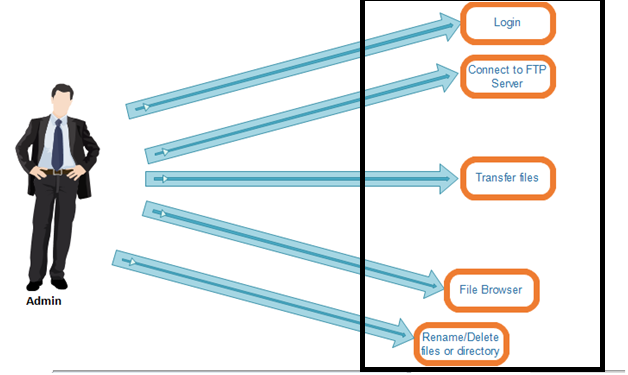


**USER-CASE VIEW**

In software and systems engineering, a use case is a list of steps, typically defining interactions between a role (known in Unified Modeling Language (UML) as an "actor") and a system, to achieve a goal. The actor can be a human, an external system, or time.

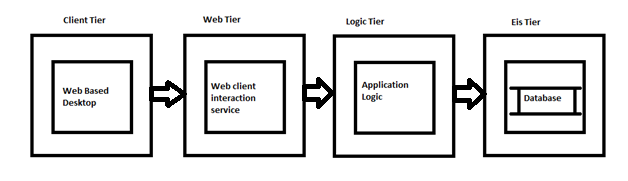
In systems engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals. The detailed requirements may then be captured in Systems Modeling Language (SysML) or as contractual statements.

Use Cases are an important requirement technique that has been widely used in modern software engineering since their formal introduction by Ivar Jacobson in 1992. Use case driven development is a key characteristic of process models and frameworks such as the Unified Process (UP), Rational Unified Process (RUP), and Oracle Unified Method (OUM). With its iterative and evolutionary nature, the use case is also a good fit for agile development.

****

**LOGICAL VIEW**

The ‘ServerZilla’ application is divided into layers based on the N-tier architecture.

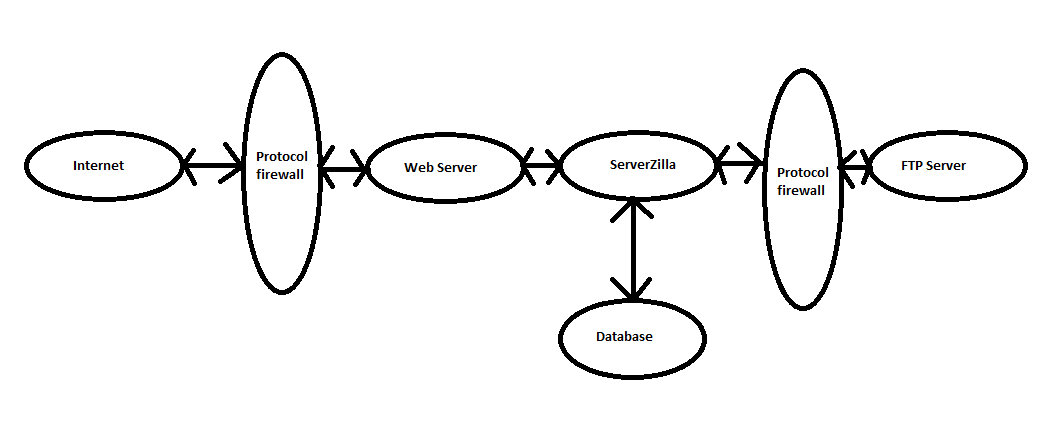


The layering approach is the mostly accepted solution for enterprise applications, which require scalability, modularity and easy maintenance.

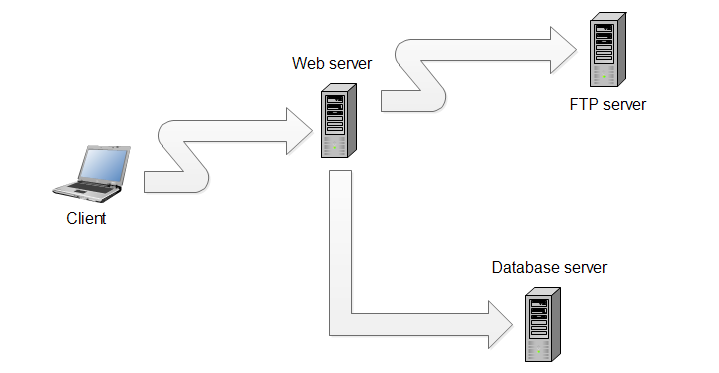
* The Web tier deals with the presentation logic and the pages rendering
* The Logic tier deals with the core functionalities of the system. (Search, post, match rides, manage profiles)
* The EIS tier is responsible for storing user profiles, and ride advertisements.

**DEPLOYMENT VIEW**

**LOGICAL STRUCTURE**



**PHYSICAL STRUCTURE**

****

**DETAILS**

* Runtime pattern is applied.
* Development is done on application server.
* The Web server used is Apache server.
* The technology used is PHP.
* The database server is MySQL.

SYSTEM DESIGN

**INTRODUCTION**

The system design involves three significant technical activities, such as designing, coding and testing. Each activity transforms information in a manner that ultimately results in validated computer software.

It is the phase in which the detailed design of the system selected in the study phase is accomplished, and the user oriented performance specification is converted in to a technical design specification. The principal activities performed during the design phase include the general system design, the design of all outputs, input design and the design of databases.

**DATA INTEGRITY AND CONSTRAINTS**

In database, database integrity is accuracy and compatibility of data. The database integrity is on the faith of many integrity constraints, so that is, designing complete database integrity is designing a complete database integrity constraint. Database integrity constraint can be achieved through the DBMS or application, and it is based on integrity constraint of DBMS as a part of mode to be stored in database. With achieving the integrity constraint of DBMS, we could design it according to the steps of designing database. And by the application software to achieve the integrity of the database are included in the application of software design.

Database integrity is a key for Database Application System, and its function is mainly reflected in the following areas:

1. Database integrity constraints can prevent legitimate users from adding non-semantic data using when using the database;
2. Using base on integrity control mechanism of DBMS to achieve the transaction rules, easy to define, easy to understand, and it can reduce the complexity of applications and improve the efficiency of operating efficiency. At the same time, base on integrity control mechanism of DBMS is centralized management, and therefore it can be achieved integrity of the database more easily than the application;
3. Reasonable designing in integrity of the database, designs are applied with consideration both the integrity of the database and system performance. For example, when loading a large amount of data, just need temporarily unavailable the constraint that is base on integrity control mechanism of DBMS before the loading of the database, after it, re-entry into force. We can guarantee the data will not affect the efficiency of loading but also ensures the integrity of the database;
4. In the software application testing, the good integrity of the database contributes to finding out the errors of application software earlier.

**Database Constraints of ‘ServerZilla’**

|  |  |
| --- | --- |
| ***CONSTRAINT*** | ***DESCRIPTION*** |
| NOT NULL | In MySQL NOT NULL constraint allows to specify that a column can not contain any NULL value. MySQL NOT NULL can be used to CREATE and ALTER a table. |
| UNIQUE | The UNIQUE constraint in MySQL does not allow to insert a duplicate value in a column. The UNIQUE constraint maintains the uniqueness of a column in a table. More than one UNIQUE column can be used in a table |
| PRIMARY KEY | A PRIMARY KEY constraint for a table enforces the table to accept unique data for a specific column and this constraint create a unique index for accessing the table faster. |
| FOREIGN KEY | A FOREIGN KEY in MySQL creates a link between two tables by one specific column of both table. The specified column in one table must be a PRIMARY KEY and referred by the column of another table known as FOREIGN KEY. |
| CHECK | A CHECK constraint controls the values in the associated column. The CHECK constraint determines whether the value is valid or not from a logical expression. |
| DEFAULT | In a MySQL table, each column must contain a value ( including a NULL). While inserting data into a table, if no value is supplied to a column, then the column gets the value set as DEFAULT. |

**Table**: ***ftp\_user***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Field*** | ***Type*** | ***Null*** | ***Key*** | ***Default*** |
| usrname | varchar(50) | Yes | Unique | NULL |
| passwd | varchar(50) | Yes |  | NULL |
| email | varchar(50) | No |  | NULL |
| reset | varchar(50) | Yes |  | NULL |

**Table**: ***ftp\_host***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Field*** | ***Type*** | ***Null*** | ***Key*** | ***Default*** |
| host | varchar(50) | Yes |  | NULL |
| username | varchar(50) | Yes |  | NULL |
| password | varchar(50) | Yes |  | NULL |

**DATABASE DESIGN**

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database.

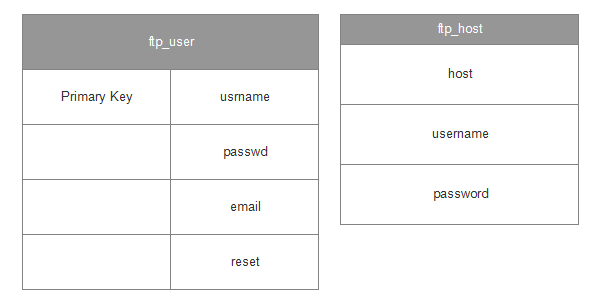
In database design several specific objectives are considered:

* Ease of learning and use.
* Controlled redundancy.
* Data independence.
* More information at low cost.
* Accuracy and integrity.
* Recovery from failure.
* Privacy and security
* Performance

The process of data base design is split into 2 distinct phases. They are

* Conceptual design.
* Logical design.

**Database Design of ServerZilla**



**INTERFACE DESIGN**

Interface provides the medium through which the users can interact with the system in an effective manner. Interfaces provide the way to either put data into the system or to retrieve data from the system, named input and output interfaces. The interface design is the process of making the layouts of interfaces. The interfaces should be simple, neat, and user friendly. The success of the system heavily depends on the quality of the interfaces.

**INPUT DESIGN**

Input design is a process of converting the data given by the user into computer acceptable form. The goal of the input design is to make the data entry easier, logical and free from errors. The system also checks the validity of user inputs.

**CODE DESIGN**

The main purpose of performing code design is to improve the efficiency of the coding and to provide ease of understanding. Proper documentation is provided to end-user for easy understanding. Further enhancement facilities are also possible.

**OUTPUT DESIGN**

Output design aims at communicating the results to user. They also used to provide a permanent copy of the results for later consultation. The generated output displays the alive, surviving and destroyed quarters of each generation with different colors for easy understanding.

PROGRAM SOURCE CODE

**PROGRAM SOURCE CODE**

***index.php***

<!DOCTYPE html>

<html xmlns**=**"http://www.w3.org/1999/xhtml" lang**=**"en">

<head>

<title>ServerZilla</title>

<link href**=**"img/icon.gif" rel**=**"icon" />

<meta name**=**"viewport" content**=**"width=device-width, initial-scale=1" />

<meta charset**=**"utf-8" />

<script src**=**"js/jquery-1.11.1.min.js"></script>

<script src**=**"js/misc.js"></script>

<script src**=**"js/jquery.confirm.min.js"></script>

<script src**=**"js/bootbox.min.js"></script>

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

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

<link rel**=**"stylesheet" href**=**"css/misc.css" />

**<?php**

@include **'**config.php**';**

@include **'**authentication.php**';**

@include **'**setcookie.php**';**

@include **'**delcookie.php**';**

@error\_reporting(1000000)**;**

set\_time\_limit(0)**;**

$usr = **"";**

$pass = **"";**

$host = **"";**

$conn = **"";**

$login = **"";**

$fpath = **"";**

$path = **"";**

$self = $\_SERVER[**'**PHP\_SELF**'**]**;**

$sep=**"**/**";**

if(strtolower(substr(PHP\_OS,0,3))==**"**win**"**)

{

$os=**"**win**";**

$sep=**"**\\**";**

$ox=**"**Windows**";**

}

else

{

$os=**"**nix**";**

$ox=**"**Linux**";**

}

if(isset($\_POST[**'**dbstore**'**]))

{

$var = $\_POST[**'**dbstore**'**]**;**

if($var==1)

{

$query = mysql\_query(**"**insert into ftp\_host values('**"**.$\_POST[**'**host**'**].**"**', '**"**.$\_POST[**'**usrname**'**].**"**', '**"**.$\_POST[**'**passwrd**'**].**"**');**"**)**;**

}

}

if(file\_exists(getcwd().$sep.**"**config.php**"**))

{

@include **'**config.php**';**

}

else {

header(**"**Location: install.php**"**)**;**

}

if(isset($\_COOKIE[**'**f\_usr**'**]) && isset($\_COOKIE[**'**f\_pass**'**]) && isset($\_COOKIE[**'**f\_host**'**])) {

$usr = $\_COOKIE[**'**f\_usr**'**]**;**

$pass = $\_COOKIE[**'**f\_pass**'**]**;**

$host = str\_replace(**"**ftp://**"**, **""**, $\_COOKIE[**'**f\_host**'**])**;**

$conn = ftp\_connect($host)**;**

if($conn)

{

$login = ftp\_login($conn, $usr, $pass)**;**

if($login)

{

$fpath = ftp\_pwd($conn)**;**

}

}

}

if(isset($\_GET[**'**path**'**]))

{

chdir($\_GET[**'**path**'**])**;**

}

if(isset($\_GET[**'**fpath**'**]))

{

$fpath = $\_GET[**'**fpath**'**]**;**

ftp\_chdir($conn, $fpath)**;**

}

if(isset($\_GET[**'**upload**'**]))

{

$file = $\_GET[**'**upload**'**]**;**

if($conn)

{

$upload = ftp\_put($conn, basename($file), basename($file), FTP\_BINARY)**;**

if($upload)

{

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Success",

message: "The file '**<?php** echo basename($file)**;** **?>**' uploaded successfully!!!.",

});

});

</script>

**<?php**

}

else{

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Error",

message: "The file '**<?php** echo basename($file)**;** **?>**' failed to upload!!!.",

});

});

</script>

**<?php**

}

}

else {

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Error",

message: "Connect to FTP server to upload file!!!.",

});

});

</script>

**<?php**

}

}

if(isset($\_GET[**'**rename**'**]))

{

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

$filer = $\_GET[**'**rename**'**]**;**

$path = $\_GET[**'**path**'**]**;**

$fpath = $\_GET[**'**fpath**'**]**;**

**?>**

<script type**=**"text/javascript">

$(**function()**{

$('#action**<?php** echo $id**;** **?>**').hide('slow');

$('#form**<?php** echo $id**;** **?>**').append("<form method='get' class='form-inline'><input type='hidden' name='fpath' value='**<?php** echo addslashes($fpath)**;** **?>**'><input type='hidden' name='path' value='**<?php** echo addslashes($path); ?>'><input type='hidden' name='oldname' value='<?php echo $filer; ?>'><input type='text' name='dorename' placeholder='<?php echo $filer; ?>' style='width='5%;' class='form-control input-small'><button class='btn btn-success btn-small' type='submit'> <span class='glyphicon glyphicon-ok' aria-hidden='true'></span> </button></form>");

});

</script>

**<?php**

}

if(isset($\_GET[**'**renamef**'**]))

{

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

$filer = $\_GET[**'**renamef**'**]**;**

$path = $\_GET[**'**path**'**]**;**

$fpath = $\_GET[**'**fpath**'**]**;**

**?>**

<script type**=**"text/javascript">

$(**function()**{

$('#action**<?php** echo $id**;** **?>**').hide('slow');

$('#form**<?php** echo $id**;** **?>**').append("<form method='get' class='form-inline'><input type='hidden' name='fpath' value='**<?php** echo addslashes($fpath)**;** **?>**'><input type='hidden' name='path' value='**<?php** echo addslashes($path); ?>'><input type='hidden' name='oldnamef' value='<?php echo $filer; ?>'><input type='text' name='dorenamef' placeholder='<?php echo $filer; ?>' style='width='5%;' class='form-control input-small'><button class='btn btn-success btn-small' type='submit'> <span class='glyphicon glyphicon-ok' aria-hidden='true'></span> </button></form>");

});

</script>

**<?php**

}

if(isset($\_GET[**'**oldname**'**]) && isset($\_GET[**'**dorename**'**]))

{

$oldname = $\_GET[**'**oldname**'**]**;**

$newname = $\_GET[**'**dorename**'**]**;**

if(rename($oldname, $newname))

{

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Success",

message: "The file '**<?php** echo $oldname**;** **?>**' renamed to '**<?php** echo $newname**;** **?>**' successfully!!!.",

});

});

</script>

**<?php**

}

else {

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Error",

message: "The file '**<?php** echo $oldname**;** **?>**' is not renamed to '**<?php** echo $newname**;** **?>**' successfully!!!.",

});

});

</script>

**<?php**

}

}

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

{

$file = $\_GET[**'**delete**'**]**;**

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

**?>**

<script type**=**"text/javascript">

$(**function()**{

$.confirm({

title:"Delete Confirmation",

text:"You are going to delete file '**<?php** echo $file**;** **?>**'. It may harm your remote host. Are you sure?",

confirm: **function(**button**)** {

document.location="**<?php** echo **"$**self?path=**"**.addslashes(getcwd()).**"**&fpath=**$**fpath&deletefile=**$**file**";** **?>**";

},

cancel: **function(**button**)** {

bootbox.alert({

title: "Cancelled",

message: "You have cancelled the job to delete file."

});

},

confirmButton: "Yes",

cancelButton: "No"

});

});

</script>

**<?php**

}

if(isset($\_GET[**'**deletefile**'**]))

{

$file = $\_GET[**'**deletefile**'**]**;**

if(is\_dir($file))

{

if(deleteDir($file))

{

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Success",

message: "You have deleted directory **<?php** echo $file**;** **?>**."

});

});

</script>

**<?php**

}

else {

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Failed",

message: "You cannot directory file **<?php** echo $file**;** **?>**."

});

});

</script>

**<?php**

}

}

else {

if(unlink($file))

{

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Success",

message: "You have deleted file **<?php** echo $file**;** **?>**."

});

});

</script>

**<?php**

}

else {

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Error",

message: "You cannot delete file **<?php** echo $file**;** **?>**."

});

});

</script>

**<?php**

}

}

}

if(isset($\_GET[**'**uploadd**'**]))

{

$odir = $\_GET[**'**uploadd**'**]**;**

$dirs = array()**;**

$fils = array()**;**

zip($odir)**;**

}

if(isset($\_GET[**'**oldnamef**'**]) && isset($\_GET[**'**dorenamef**'**]))

{

$oldname = $\_GET[**'**oldnamef**'**]**;**

$newname = $\_GET[**'**dorenamef**'**]**;**

if(ftp\_rename($conn, $oldname, $newname))

{

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Success",

message: "The file '**<?php** echo $oldname**;** **?>**' renamed to '**<?php** echo $newname**;** **?>**' successfully!!!.",

});

});

</script>

**<?php**

}

else {

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Error",

message: "The file '**<?php** echo $oldname**;** **?>**' is not renamed to '**<?php** echo $newname**;** **?>**' successfully!!!.",

});

});

</script>

**<?php**

}

}

if(isset($\_GET[**'**download**'**]))

{

$file = $\_GET[**'**download**'**]**;**

if(ftp\_get($conn, $file, $file, FTP\_BINARY))

{

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Success",

message: "The file '**<?php** echo $file**;** **?>**' downloaded to remote host successfully!!!"

})

});

</script>

**<?php**

}

else {

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Error",

message: "The file '**<?php** echo $file**;** **?>**' failed to download!!!"

})

});

</script>

**<?php**

}

}

if(isset($\_GET[**'**deletef**'**]))

{

$filed = $\_GET[**'**deletef**'**]**;**

**?>**

<script type**=**"text/javascript">

$(**function()**{

$.confirm({

title:"Delete Confirmation",

text:"You are going to delete file '**<?php** echo $filed**;** **?>**'. It may harm your FTP host. Are you sure?",

confirm: **function(**button**)** {

document.location="**<?php** echo **"$**self?path=**"**.addslashes(getcwd()).**"**&fpath=**$**fpath&deletefilef=**$**filed**";** **?>**";

},

cancel: **function(**button**)** {

bootbox.alert({

title: "Cancelled",

message: "You have cancelled the job to delete file."

});

},

confirmButton: "Yes",

cancelButton: "No"

});

});

</script>

**<?php**

}

if(isset($\_GET[**'**deletedf**'**]))

{

$filed = $\_GET[**'**deletedf**'**]**;**

**?>**

<script type**=**"text/javascript">

$(**function()**{

$.confirm({

title:"Delete Confirmation",

text:"You are going to delete file '**<?php** echo $filed**;** **?>**'. It may harm your FTP host. Are you sure?",

confirm: **function(**button**)** {

document.location="**<?php** echo **"$**self?path=**"**.addslashes(getcwd()).**"**&fpath=**$**fpath&deletefiledf=**$**filed**";** **?>**";

},

cancel: **function(**button**)** {

bootbox.alert({

title: "Cancelled",

message: "You have cancelled the job to delete file."

});

},

confirmButton: "Yes",

cancelButton: "No"

});

});

</script>

**<?php**

}

if(isset($\_GET[**'**deletefilef**'**]))

{

$file = $\_GET[**'**deletefilef**'**]**;**

if(ftp\_delete($conn, $file))

{

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Success",

message: "You have deleted the file '**<?php** echo $file**;** **?>**'."

});

});

</script>

**<?php**

}

else {

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Error",

message: "You cannot delete the file '**<?php** echo $file**;** **?>**'."

});

});

</script>

**<?php**

}

}

if(isset($\_GET[**'**deletefiledf**'**]))

{

$file = $\_GET[**'**deletefiledf**'**]**;**

if(deleteFDir($conn, $file))

{

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Success",

message: "You have deleted the directory '**<?php** echo $file**;** **?>**'."

});

});

</script>

**<?php**

}

else{

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Error",

message: "You cannot delete the directory '**<?php** echo $file**;** **?>**'."

});

});

</script>

**<?php**

}

}

if(isset($\_GET[**'**newdir**'**]))

{

$dir = $\_GET[**'**newdir**'**]**;**

if(mkdir($dir))

{

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Success",

message: "You have created the directory '**<?php** echo $dir**;** **?>**'."

});

});

</script>

**<?php**

}

else {

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Error",

message: "You cannot create the directory '**<?php** echo $dir**;** **?>**'."

});

});

</script>

**<?php**

}

}

if(isset($\_GET[**'**newdirf**'**]))

{

$dir = $\_GET[**'**newdirf**'**]**;**

if(ftp\_mkdir($conn, $dir))

{

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Success",

message: "You have created the directory '**<?php** echo $dir**;** **?>**'."

});

});

</script>

**<?php**

}

else {

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Error",

message: "You cannot create the directory '**<?php** echo $dir**;** **?>**'."

});

});

</script>

**<?php**

}

}

**function** deleteFDir**(**$conn, $path**)**

{

$list = getfiles(ftp\_pwd($conn).**'**/**'**.$path)**;**

foreach($list as $item)

{

if($item[**'**type**'**]==**'**Directory**'**)

{

deleteFDir($conn, $path.**'**/**'**.$item[**'**name**'**])**;**

}

else {

ftp\_delete($conn, $path.**'**/**'**.$item[**'**name**'**])**;**

}

}

if(ftp\_rmdir($conn, $path))

{

return true**;**

}

else {

return false**;**

}

}

$flg = false**;**

**function** deleteDir**(**$file**)**

{

**global** $sep, $flg**;**

$file = (substr($file,-1)==$sep)? $file:$file.$sep**;**

$flg = FALSE**;**

if($dh = opendir($file)){

while(($f = readdir($dh))!==false){

if($f != **'**.**'** && $f != **'**..**'**){

$f = $file.$f**;**

if(is\_dir($f)){

deleteDir($f)**;**

}

else{

unlink($f)**;**

}

}

}

closedir($dh)**;**

if(rmdir($file))

{

$flg = TRUE**;**

}

else {

$flg = FALSE**;**

}

}

return $flg**;**

}

**function** filesizes**(**$size**)**

{

if ($size>=1073741824)$size = round(($size/1073741824) ,2).**"** GB**";**

elseif ($size>=1048576)$size = round(($size/1048576),2).**"** MB**";**

elseif ($size>=1024)$size = round(($size/1024),2).**"** KB**";**

else $size .= **"** B**";**

return $size**;**

}

if($conn)

{

**?>**

<script type**=**"text/javascript">

$(**function()**{

$("#connect").text("Disconnect");

$('#connect').attr('href', '?disconnect');

$('#connect').removeClass('btn-primary').addClass('btn-danger');

});

</script>

**<?php**

}

else {

**?>**

<script type**=**"text/javascript">

$(**function()**{

$("#connect").text("Connect");

$('#connect').attr('href', '#loginbox');

});

</script>

**<?php**

}

**function** zip**(**$dir**)**

{

**global** $sep, $conn**;**

$zip = new ZipArchive**;**

$namef = getcwd().$sep.basename($dir).**"**.zip**";**

$zip->open($namef, ZipArchive::CREATE)**;**

$files = new RecursiveIteratorIterator(new RecursiveDirectoryIterator($dir), RecursiveIteratorIterator::SELF\_FIRST)**;**

foreach ($files as $name => $file)

{

if (substr($name,-1)==**"**.**"**) continue**;**

if (substr($name,-1)==**"**..**"**) continue**;**

$filePath = $file->getRealPath()**;**

if(filetype($filePath) == **'**dir**'**)

{

$zip->addEmptyDir($filePath, str\_replace($dir . $sep, **''**, $filePath . $sep))**;**

}

else {

$zip->addFile($filePath, str\_replace($dir . $sep, **''**, $filePath))**;**

}

}

if(!$zip->close())

{

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Failed",

message: "The directory **<?php** echo basename($dir)**;** **?>** failed to compress",

});

});

</script>

**<?php**

}

else {

if($conn)

{

$upload = ftp\_put($conn, basename($namef), basename($namef), FTP\_BINARY)**;**

if($upload)

{

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Success",

message: "The file '**<?php** echo basename($namef)**;** **?>**' uploaded successfully!!!.",

});

});

</script>

**<?php**

}

else{

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Error",

message: "The file '**<?php** echo basename($namef)**;** **?>**' failed to upload!!!.",

});

});

</script>

**<?php**

}

}

else {

**?>**

<script type**=**"text/javascript">

$(**function()**{

bootbox.alert({

title: "Error",

message: "Connect to FTP server to upload file!!!.",

});

});

</script>

**<?php**

}

}

}

**?>**

</head>

<body>

<div style**=**"background**:** black url(*'img/bg.jpg'*) no-repeat 50% 50%**;**color**:** yellow**;** height**:** 150px**;** width**:**100%**;**">

<div class**=**"row" style**=**"text-align**:** right**;**width**:**100%**;** ">

<a class**=**"btn btn-primary" data-toggle**=**"modal" href**=**"#loginbox" id**=**"connect">Connect</a>

<a class**=**"btn btn-danger" href**=**"?logout">Logout</a>

</div>

</div>

<div class**=**"container">

<div id**=**"loginbox" class**=**"modal" tabindex**=**"-1" role**=**"dialog" aria-hidden**=**"true">

<div class**=**"modal-dialog">

<div class**=**"modal-content">

<div class**=**"modal-header">

<button type**=**"button" class**=**"close" data-dismiss**=**"modal" aria-hidden**=**"true">×</button>

<h1 class**=**"text-center">Login FTP</h1>

</div>

<div class**=**"modal-body">

<form class**=**"form col-md-12 center-block" method**=**"post" action**=**"?">

<div class**=**"form-group has-property" id**=**"hostDiv">

<input type**=**"text" class**=**"form-control" name**=**"host" id**=**"host" placeholder**=**"FTP Host" list**=**"hosts">

<datalist id**=**"hosts">

**<?php**

$hostQuery = mysql\_query(**"**select \* from ftp\_host;**"**)**;**

while($row=mysql\_fetch\_array($hostQuery))

{

print **"**<option>**"**.$row[**'**host**'**].**"**</option>\n**";**

}

**?>**

</datalist>

</div>

<div class**=**"form-group has-property" id**=**"usrDiv">

<input type**=**"text" class**=**"form-control" name**=**"usrname" id**=**"usrname" placeholder**=**"Username">

</div>

<div class**=**"form-group has-property" id**=**"passwdDiv">

<input type**=**"password" class**=**"form-control" name**=**"passwrd" id**=**"passwrd" placeholder**=**"Password">

</div>

<div class**=**"checkbox">

<label>

<input type**=**"checkbox" name**=**"dbstore" value**=**"1"> Save to Database

</label>

</div>

<div class**=**"form-group">

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

</div>

</form>

</div>

<div class**=**"modal-footer">

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

<button class**=**"btn" data-dismiss**=**"modal" aria-hidden**=**"true">Cancel</button>

</div>

</div>

</div>

</div>

</div>

</div>

<div class**=**"container-fluid">

<div class**=**"row">

<div class**=**"col-md-6" style**=**"height**:**100%**;** border-right**:** 1px solid #ccc**;**">

**<?php**

$path = getcwd()**;**

$dirs = array()**;**

$fils = array()**;**

**?>**

<h2 style**=**"text-align**:** center**;**">Remote Host Files</h2><hr />

<form method**=**"get" action**=**"#"><div class**=**"form-inline"><input type**=**"hidden" value**=**"**<?php** echo $fpath**;** **?>**" name**=**"fpath"><input type**=**"hidden" name**=**"path" value**=**"**<?php** echo $path**;** **?>**" /><div><label for="path" class="required">Path: </label></div><input style="width: 80%;" value="<?php echo $path; ?>" class='input-large form-control search-query' name='path'><button class="btn btn-success" type="submit"> <span class='glyphicon glyphicon-ok' aria-hidden='true'></span> </button></div></form><hr />

<form method**=**"get" action**=**"#"><div class**=**"form-inline"><input type**=**"hidden" value**=**"**<?php** echo $fpath**;** **?>**" name**=**"fpath"><input type**=**"hidden" name**=**"path" value**=**"**<?php** echo $path**;** **?>**" /><div><label for="directory" class="required">New Directory: </label></div><input style="width: 80%;" placeholder="New Directory" class='input-large form-control search-query' name='newdir'><button class="btn btn-success" type="submit"> <span class='glyphicon glyphicon-ok' aria-hidden='true'></span> </button></div></form><hr />

<table class**=**"table table-hover">

**<?php**

if(is\_dir($path))

{

chdir($path)**;**

**?>**

<tr><th>Name</th><th>Size</th><th>Actions</th></tr>

**<?php**

$id = 0**;**

if($handle=opendir($path))

{

while(($item=readdir($handle))!==FALSE)

{

if($item==**'**.**'**){continue**;**}

elseif ($item==**'**..**'**) {continue**;**}

if(is\_dir($item))

{

array\_push($dirs, $path.$sep.$item)**;**

}

else {

array\_push($fils, $path.$sep.$item)**;**

}

}

foreach ($dirs as $dir) {

echo **"**<tr><td>[ <a href='**$**self?path=**$**dir&fpath=**$**fpath'>**"**.basename($dir).**"**</a> ]</td><td>**"**.filesizes(filesize($dir)).**"**</td><td style='width:40%;'><div id='action**$**id'>| <a href='**$**self?id=**$**id&path=$path&fpath=$fpath&rename=".basename($dir)."#action$id'><span class='glyphicon glyphicon-pencil' aria-hidden='true'></span></a> | <a id='delete$id' href='$self?id=$id&path=$path&fpath=$fpath&delete=".basename($dir)."'><span class='glyphicon glyphicon-remove' aria-hidden='true'></span></a> | <a href='$self?path=$path&uploadd=$dir&fpath=$fpath'><span class='glyphicon glyphicon-open' aria-hidden='true'></span></a> | </div><div style='text-align='right';' id='form$id'></div></td></tr>\n";

$id = $id+1**;**

}

foreach ($fils as $fil) {

echo **"**<tr><td><a href='**$**self?path=**$**path&upload=**$**fil&fpath=**$**fpath'>**"**.basename($fil).**"**</a></td><td>**"**.filesizes(filesize($fil)).**"**</td><td style='width:40%;'><div id='action**$**id'> | <a href='**$**self?id=$id&path=$path&fpath=$fpath&rename=".basename($fil)."#action$id'><span class='glyphicon glyphicon-pencil' aria-hidden='true'></span></a> | <a id='delete$id' href='$self?id=$id&path=$path&fpath=$fpath&delete=".basename($fil)."'><span class='glyphicon glyphicon-remove' aria-hidden='true'></span></a> | <a href='$self?path=$path&upload=$fil&fpath=$fpath'><span class='glyphicon glyphicon-open' aria-hidden='true'></span></a> | </div><div id='form$id'></div></td></tr>\n";

$id = $id+1**;**

}

}

}

**?>**

</table>

</div>

<div class**=**"col-md-6" style**=**"height**:**100%**;**border-left**:** 1px solid #ccc**;**">

<h2 style**=**"text-align**:** center">FTP Server Files</h2><hr />

<div class**=**"form-inline"><form method**=**"get" action**=**"#"><div><label for**=**"fpath" class**=**"required">Path: </label></div><input type**=**"hidden" value**=**"**<?php** echo $path**;** **?>**" name**=**"path"><input type**=**"hidden" name="fpath" value="<?php echo $fpath;?>"><input style="width: 80%;" value="<?php echo $fpath; ?>" class='input-large form-control search-query' name='fpath'><button class="btn btn-success" type="submit"> <span class='glyphicon glyphicon-ok' aria-hidden='true'></span> </button></div></form><hr />

<div class**=**"form-inline"><form method**=**"get" action**=**"#"><div><label for**=**"newdir" class**=**"required">New Directory: </label></div><input type**=**"hidden" value**=**"**<?php** echo $path**;** **?>**" name**=**"path"><input type="hidden" name="fpath" value="<?php echo $fpath;?>"><input style="width: 80%;" placeholder="New Directory" class='input-large form-control search-query' name='newdirf'><button class="btn btn-success" type="submit"> <span class='glyphicon glyphicon-ok' aria-hidden='true'></span> </button></div></form><hr />

<table class**=**"table table-hover">

<tr><th>Name</th><th>Size</th><th>Actions</th></tr>

**<?php**

if($conn)

{

$props = getfiles($fpath)**;**

$dirs = array()**;**

$fils = array()**;**

foreach($props as $prop) {

if($prop[**'**type**'**]==**'**Directory**'**)

{

array\_push($dirs, $prop)**;**

}

else {

array\_push($fils, $prop)**;**

}

}

foreach($dirs as $dir){

echo **"**<tr><td>[ <a href='**$**self?path=**$**path&fpath=**$**fpath**"**.$dir[**'**name**'**].**"**/'>**"**.$dir[**'**name**'**].**"**</a> ]</td><td>**"**.filesizes($dir[**'**size**'**]).**"**</td><td style='width:40%;'><div id='action**$**id'> | <a href='**$**self?id=$id&path=$path&fpath=$fpath&renamef=".$dir['name']."#action$id'><span class='glyphicon glyphicon-pencil' aria-hidden='true'></span></a> | <a href='$self?path=$path&fpath=$fpath&deletedf=".$dir['name']."'><span class='glyphicon glyphicon-remove' aria-hidden='true'></span></a> | </div><div id='form$id'></div></td></tr>";

$id = $id+1**;**

}

foreach ($fils as $fil) {

echo **"**<tr><td><a href='**$**self?path=**$**path&fpath=**$**fpath&download=**"**.$fil[**'**name**'**].**"**'>**"**.$fil[**'**name**'**].**"**</a></td><td>**"**.filesizes($fil[**'**size**'**]).**"**</td><td style='width:40%;'><div id='action**$**id'> | <a href='$self?id=$id&path=$path&fpath=$fpath&renamef=".$fil['name']."#action$id'><span class='glyphicon glyphicon-pencil' aria-hidden='true'></span></a> | <a href='$self?path=$path&fpath=$fpath&deletef=".$fil['name']."'><span class='glyphicon glyphicon-remove' aria-hidden='true'></span></a> | <a href='$self?path=$path&fpath=$fpath&download=".$fil['name']."'><span class='glyphicon glyphicon-save' aria-hidden='true'></span></a> | </div><div id='form$id'></div></td></tr>";

$id = $id+1**;**

}

}

**?>**

</table>

</div>

</div>

</div>

</body>

**<?php**

**function** getfiles**(**$directory=**"**.**"**)

{

**global** $conn**;**

if(is\_array($directs=ftp\_rawlist($conn, $directory)))

{

$prop = array()**;**

$props = array()**;**

foreach($directs as $dirx)

{

$chunks = preg\_split(**"**/[\s]+/**"**, $dirx, 9)**;**

list($prop[**'**perm**'**], $prop[**'**num**'**], $prop[**'**user**'**], $prop[**'**group**'**], $prop[**'**size**'**], $prop[**'**mon**'**], $prop[**'**day**'**], $prop[**'**time**'**]) = $chunks**;**

$prop[**'**type**'**]=$chunks[0]{0}===**'**d**'**?**'**Directory**'**:**'**File**';**

$prop[**'**name**'**]=$chunks[8]**;**

array\_splice($chunks, 0, 8)**;**

$props[implode(**"** **"**, $chunks)]=$prop**;**

}

return $props**;**

}

}

if(isset($\_COOKIE[**'**f\_usr**'**]) && isset($\_COOKIE[**'**f\_pass**'**]) && isset($\_COOKIE[**'**f\_host**'**])) {

if(!$conn) {

**?>**

<script>

**var** logBox = $("#loginbox");

logBox.fadeIn("slow");

$("#loginbox").modal("show");

$('#hostDiv').removeClass("has-success").addClass("has-error");

$(location).attr('href', "?#loginbox");

</script>

**<?php**

}

else {

if(!$login) {

**?>**

<script>

**var** logBox = $("#loginbox");

logBox.fadeIn("slow");

$("#loginbox").modal();

$('#usrDiv').removeClass("has-success").addClass("has-error");

$('#passwdDiv').removeClass("has-success").addClass("has-error");

$(location).attr('href', "?#loginbox");

</script>

**<?php**

}

}

}

**?>**

<hr />

<footer id**=**"header" style**=**"position**:** fixed**;** bottom**:** 0**;** width**:** 100%**;**">

<center><label>&copy; ServerZilla &copy;</label></center>

</footer>

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

</body></html>

SYSTEM TESTING

**SYSTEM TESTING**

The objective of system testing is to ensure that all individual programs are working as expected, that the programs link together to meet the requirements specified and to ensure that the computer system and the associated procedures work together.

Testing the newly developed or modified systems is one of the most important activities in the system development methodology. The goal of testing is to verify the logical and physical operation of the design blocks to determine that they operate as intended.

No system design is ever perfect. Communication problem, programmer’s negligence or time constraints create error that must be eliminated before the system is ready for user acceptance testing. A system is tested for on-line response, volume of transaction stress, recovery from failure. Following system testing is acceptance testing or live running the system with live data by the actual user.

Testing is the one step in the software engineering process that could be viewed as destructive rather than constructive. Testing requires that the developer discard the preconceived notions of the “correctness” of the software just developed and overcome the conflict of interest that occurs when errors are uncovered.

The initial phase of system testing is the responsibility of the analyst who determines what conditions are to be tested, generates test data, produced a schedule of expected results, runs the test and compares the computer produced the results with the expected results with the expected results.

The analyst may also be involved in the procedures for the testing. When the analyst is satisfied that the system is working properly, e hands it over to the user for testing. The importance of system testing is that the user must be stressed.

Method accepted for system testing is the stage of implementation, which is aimed at ensuring that the system works accurately & efficiently before live operation commences. Testing is vital to the success of the system. An elaborated testing of data prepared and the system is tested using the test data. While testing error noted and corrections are made.

Main Objectives of System testing are:

* To ensure during operation the system will perform as per specification.
* To make sure that the system meets user’s requirements during operation.
* To verify that the controls incorporated in the system function as intended.
* To see that when correct inputs are fed to the system the outputs are correct.
* To make sure that during operation, incorrect input and output will be deleted.

Testing generally removes all the residual bugs and improves the reliability of the program. The basic types of testing are,

* Unit Testing
* Integration Testing
* Validation Testing
* Output Testing

**UNIT TESTING**

Unit testing focuses verification efforts on the smallest unit of software design, the module. This is also known as “Module Testing”. This testing is carried out during programming stage itself. In this testing step each module is found to be working satisfactorily as regard to expected output from module.

In the project ‘ServerZilla’, tested the units like login of admin, upload files to FTP server, download files from FTP server to web server, rename of files and directories, deletion of files and directories, creation of directories, file browsing, cookie setting and deletion, resetting of password, etc. Then the errors found in the modules are eliminated.

**INTEGRATION TESTING**

Integration testing is systematic technique for constructing test to uncover errors associated within the interface. In this project, all the modules combined, and then entire program is tested as whole. Thus in the integration testing step, all the errors uncovered are corrected for next testing steps.

In project ‘ServerZilla’, tested file renaming and file browser, file uploading and file browser, file deletion and file browser at same time.

**VALIDATION TESTING**

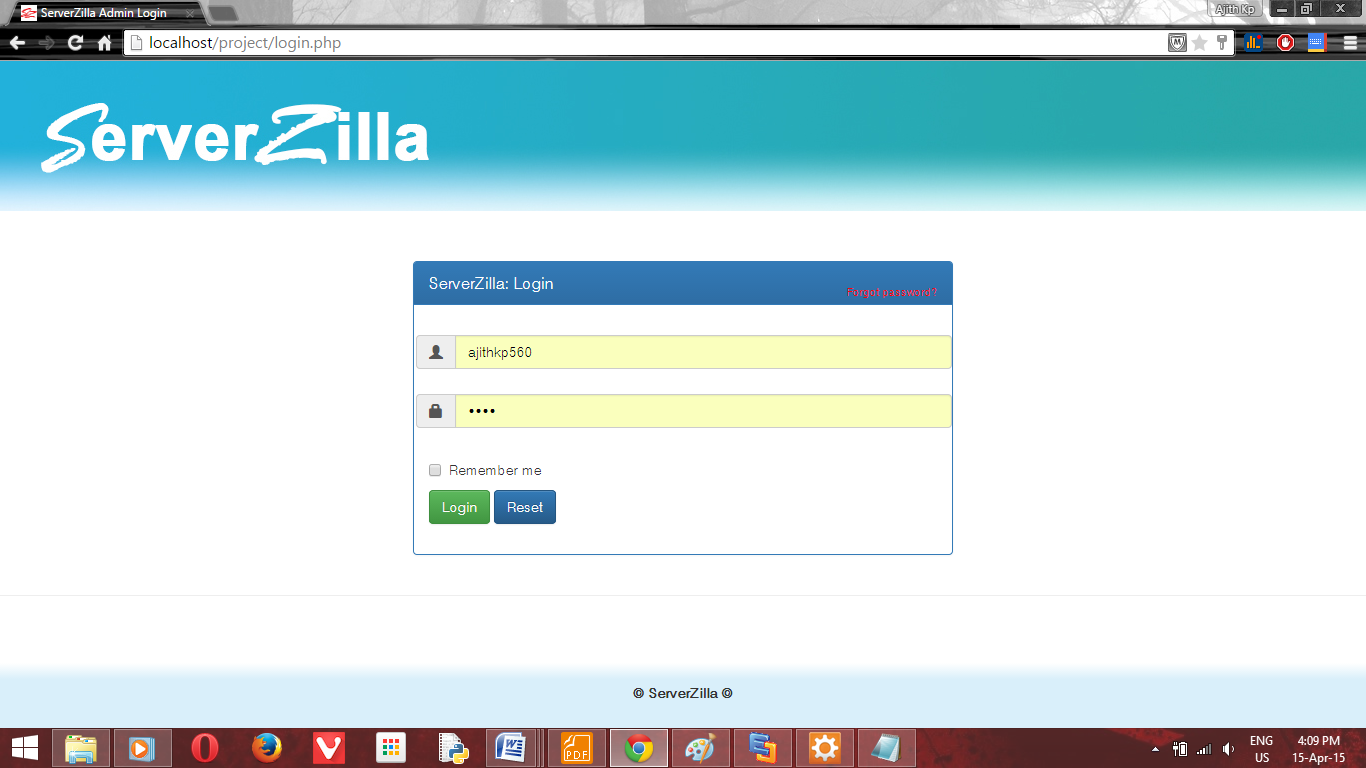
Validation testing is where requirements established as a part of software requirements analysis is validated against the software that has been constructed. This test provides the final assurance that the software meets all functional, behavioral and performance requirements. The errors, which are uncovered during integration testing, are corrected during this phase.

**OUTPUT TESTING**

After performing the validation testing, the next step is output testing of the proposed system since no system could be useful if it does not produce the required output in the specific format. The output generated or displayed by the system under consideration is tested asking the users about the format required by them. Here, the output is shown on screen as message box. The output message box is created using BootBox JavaScript code, which have better look than traditional JavaScript alert boxes.

SCREENSHOTS

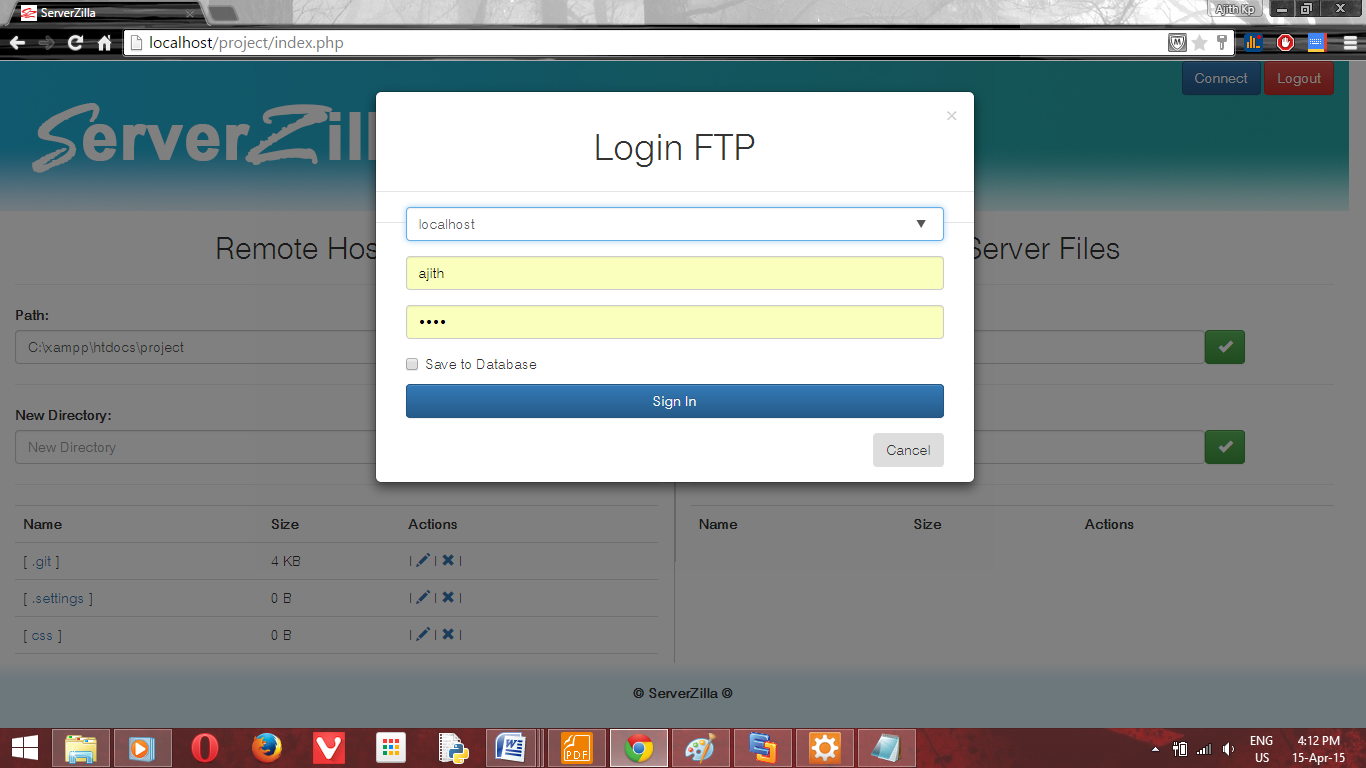
**SCREENSHOTS**



***Admin Login.***



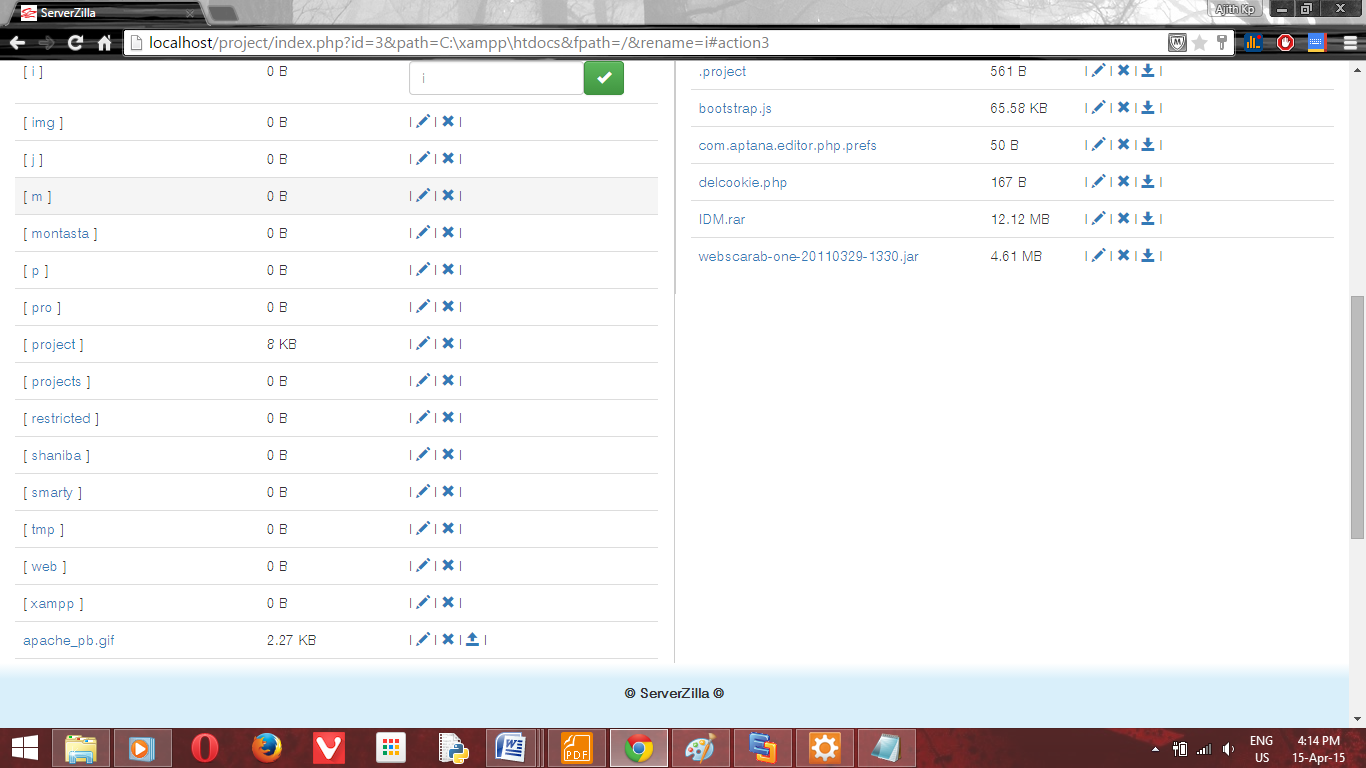
***Index Page.***



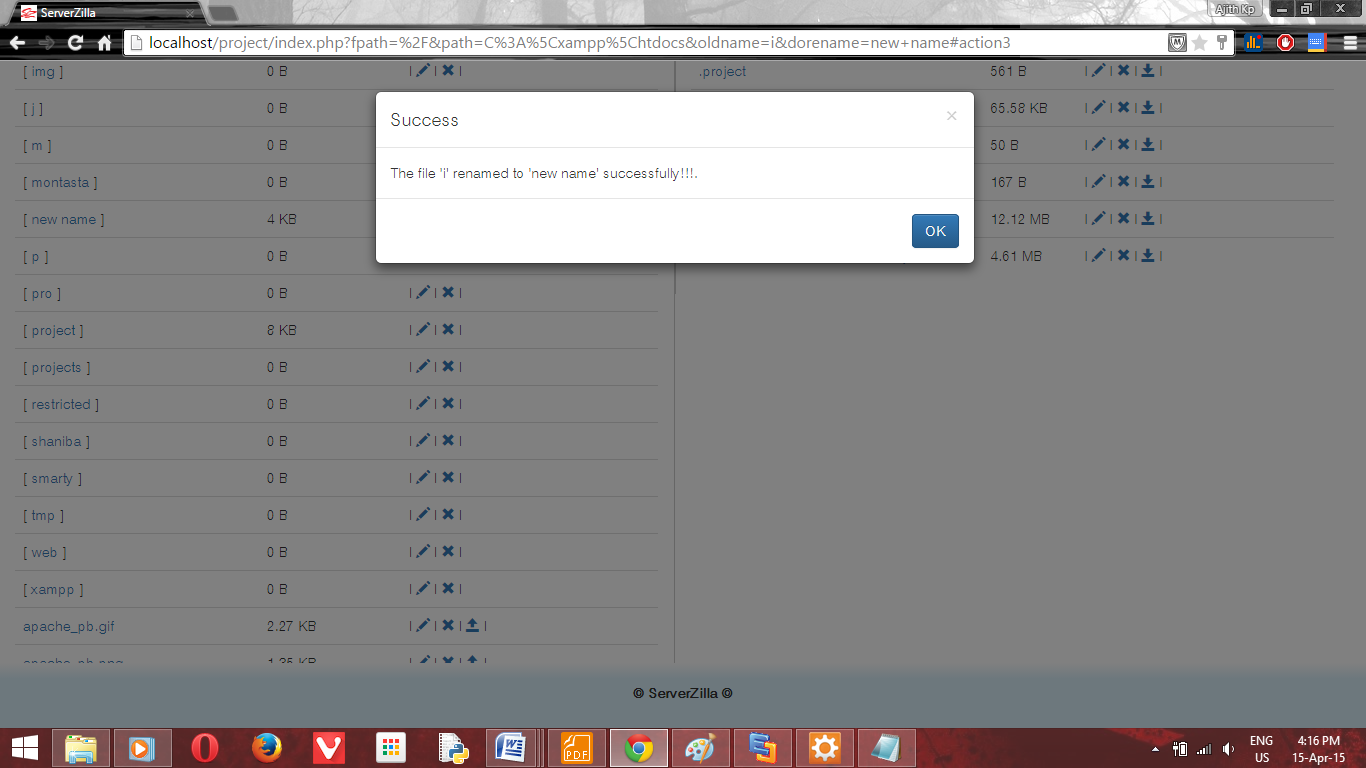
***FTP Login.***



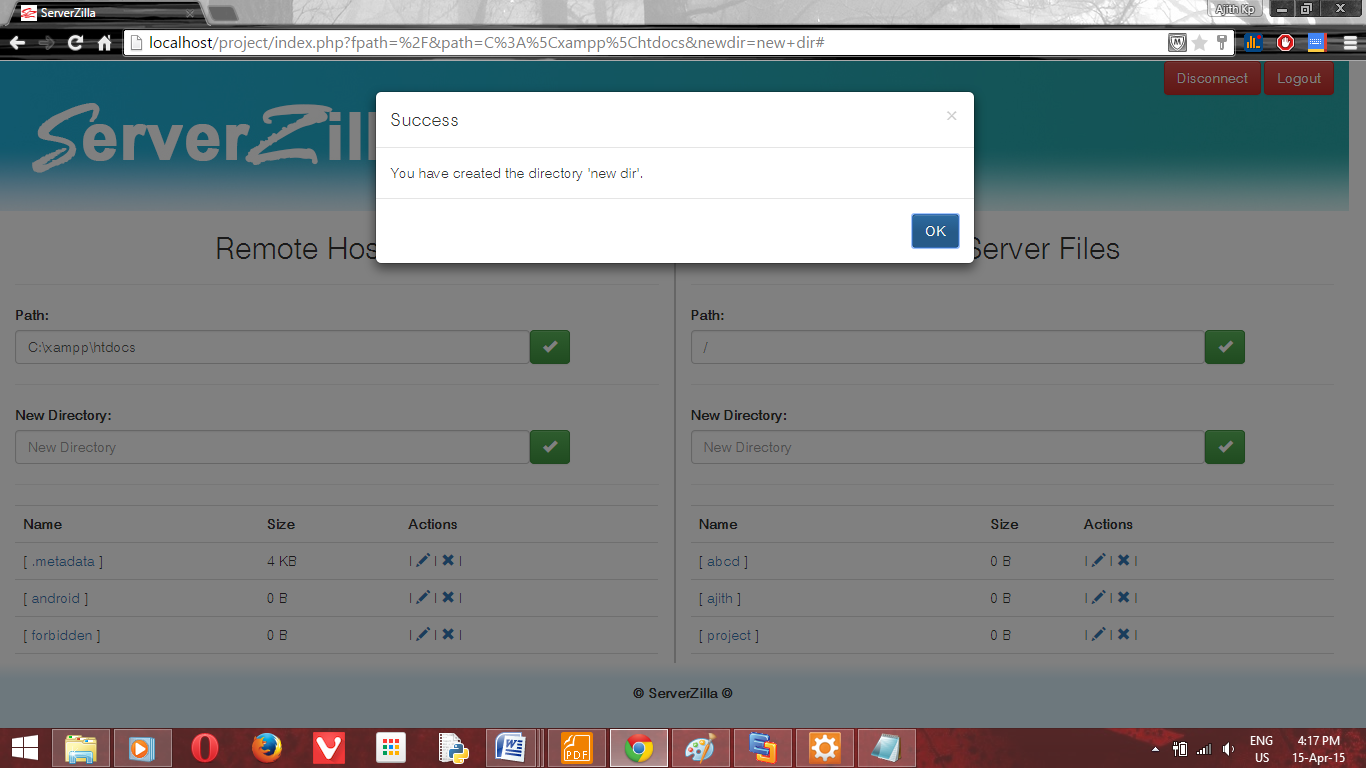
***After Login to FTP Server.***



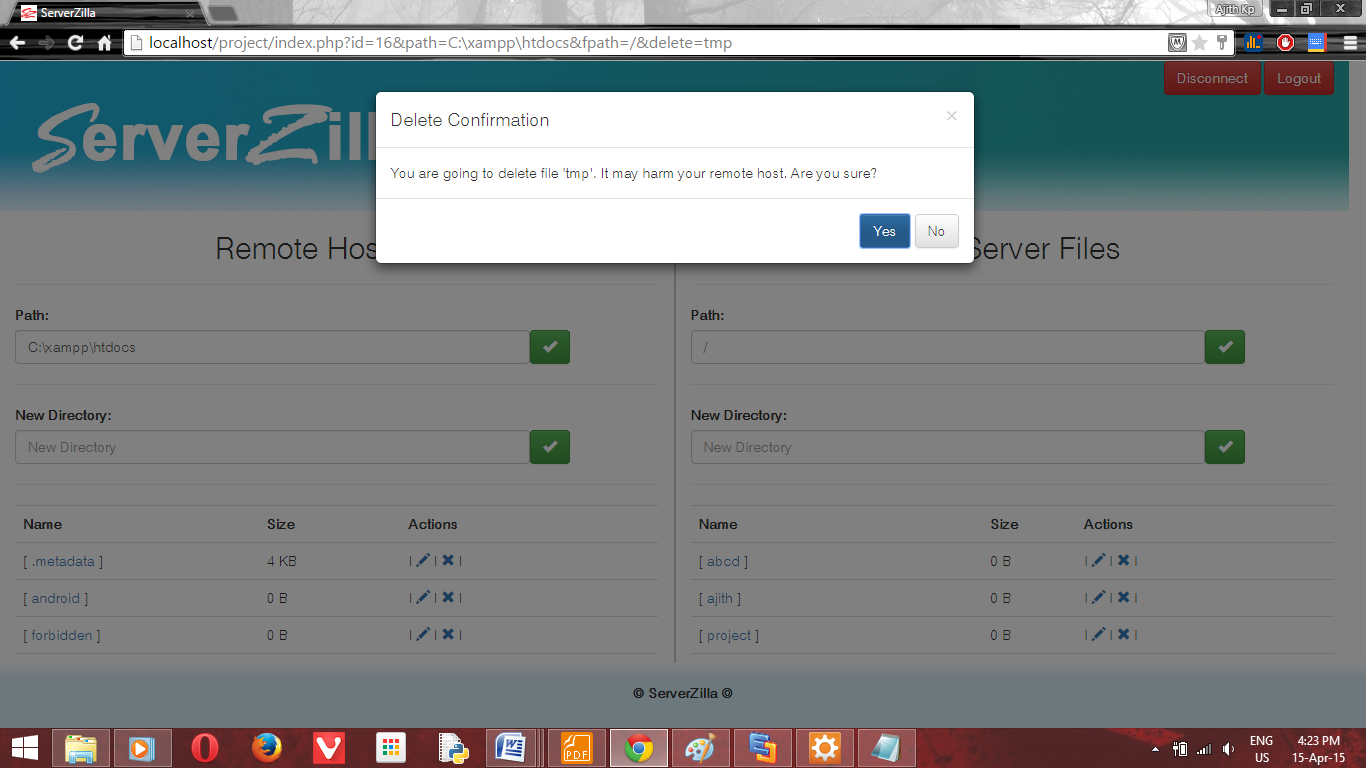
***Rename form.***



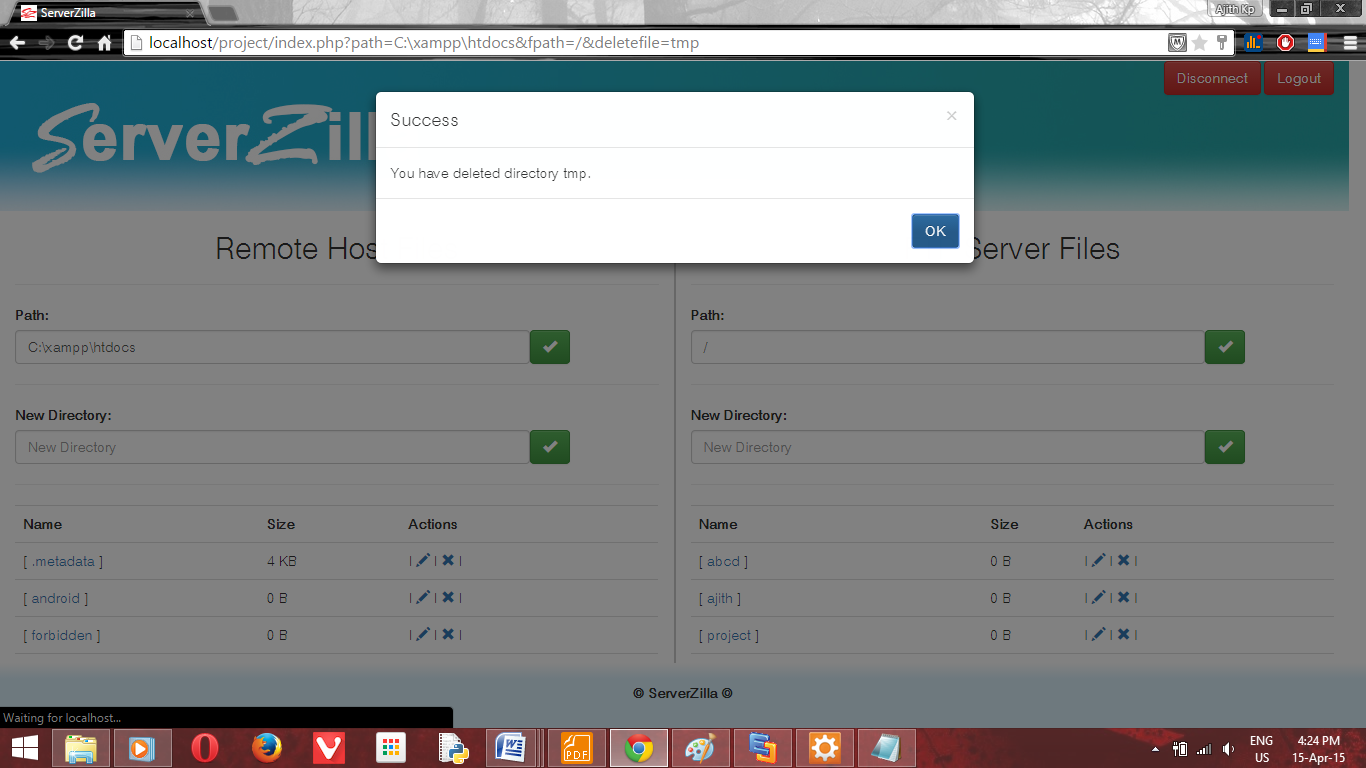
***After rename file successfully.***



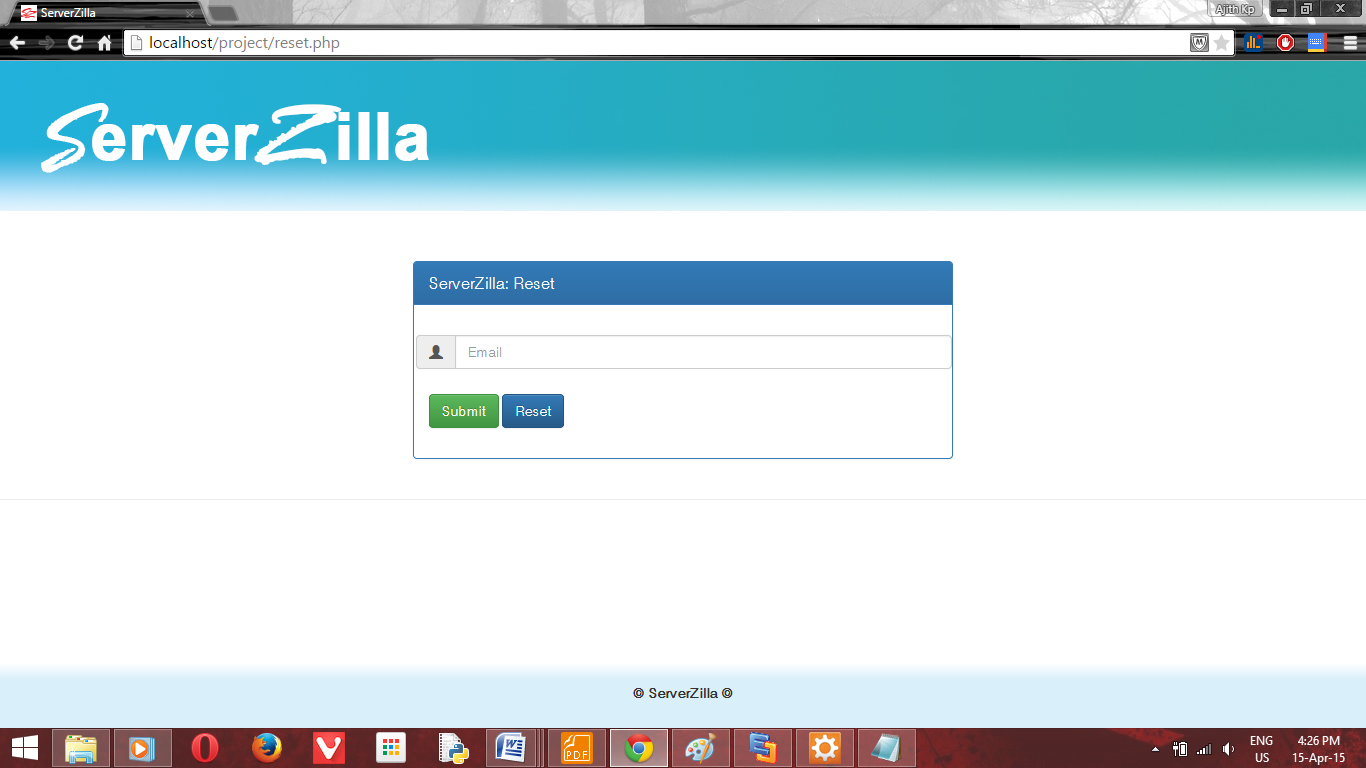
***After create new directory successfully.***



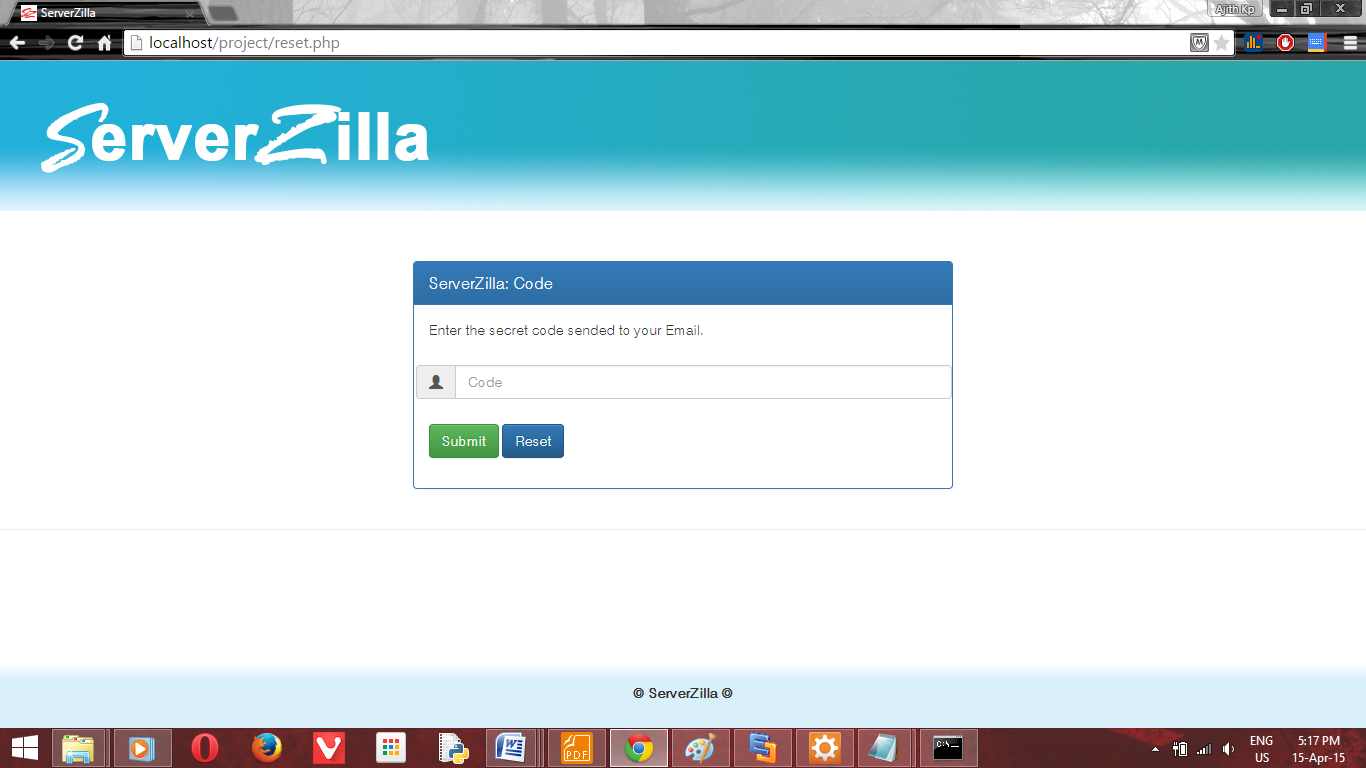
***Deletion confirm box.***



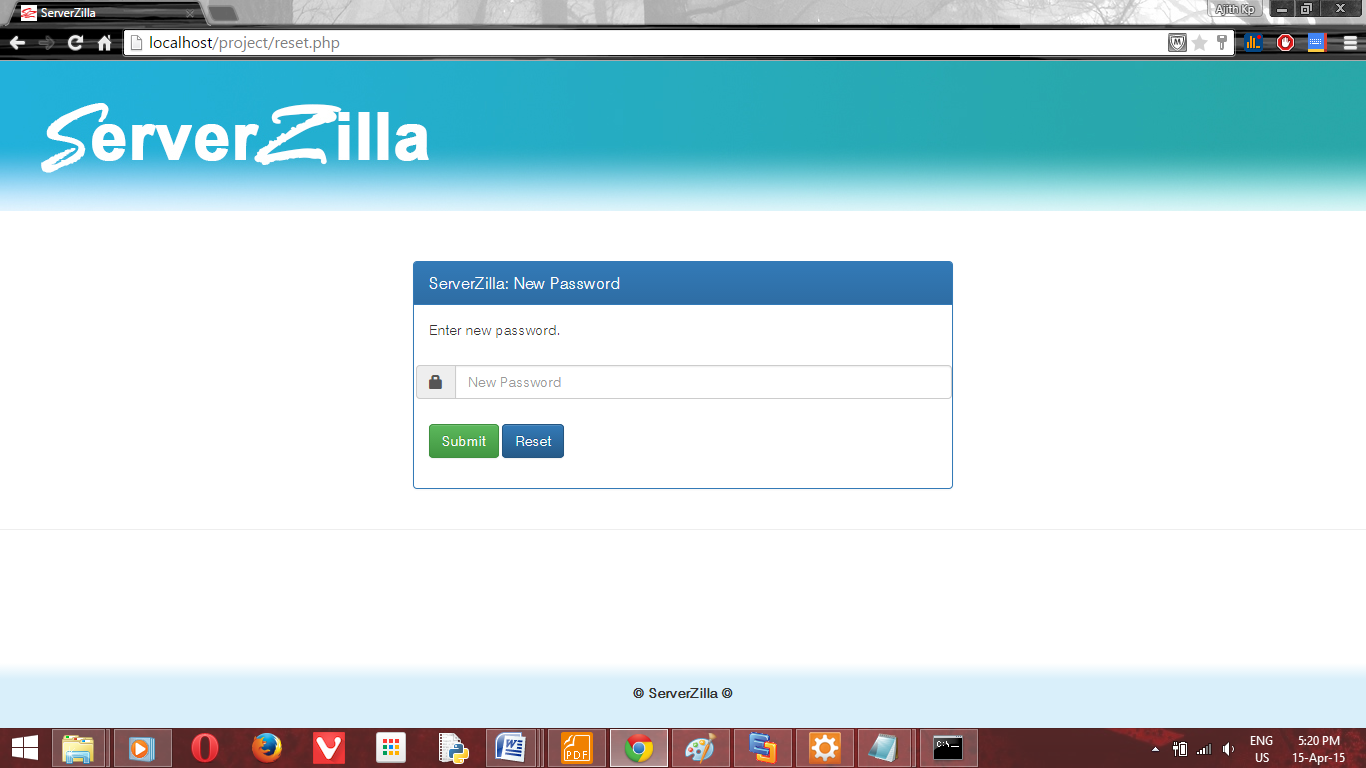
***After delete successfully.***



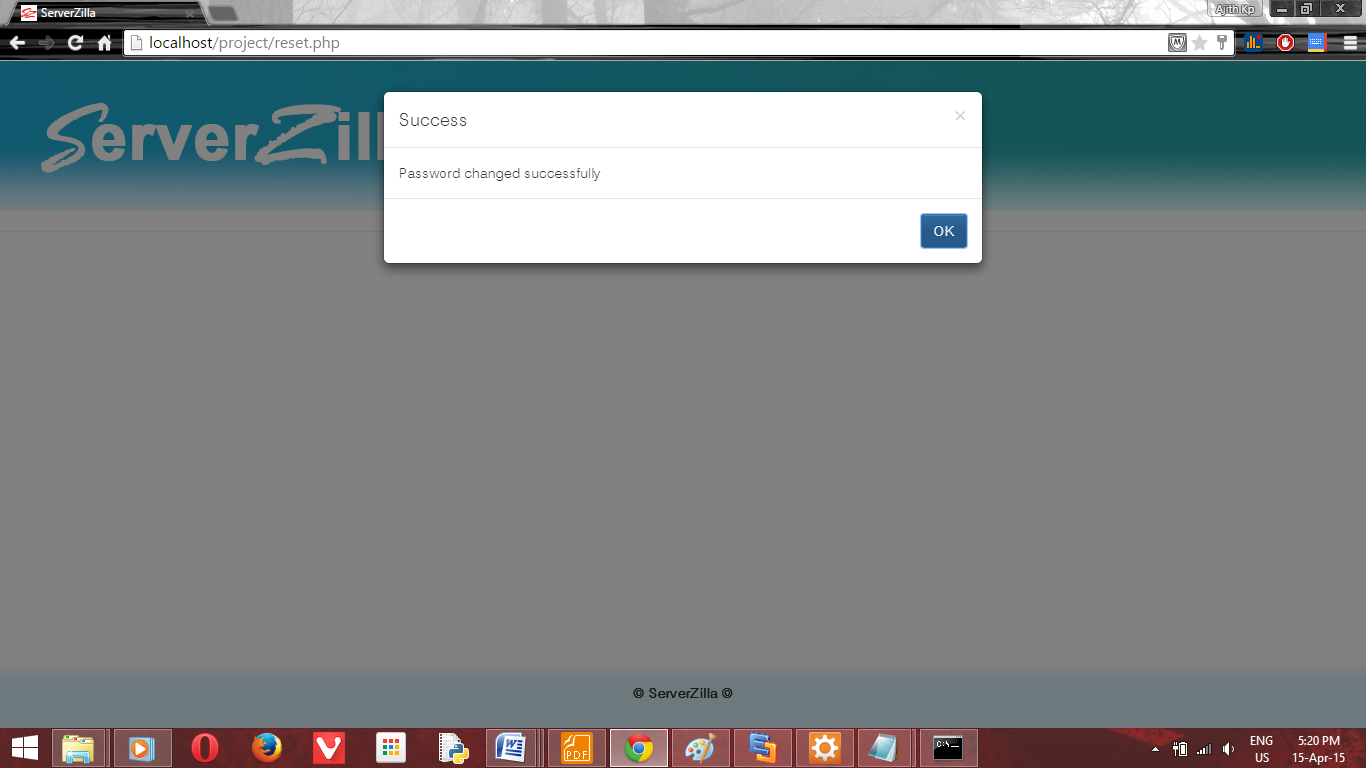
***Password reset form- Email address insertion form.***



***Secret code form.***



***New password form.***



***After change password.***

SECURITY

**SECURITY**

The project ‘ServerZilla’ is an administrating system. So the system must be protected from unauthorized accesses. Because unauthorized accesses may cause large destructions. Thus the security is implemented.

The security systems applied in the system are,

* Only one user is allowed; named administrator.
* Password is set for administrator. So when the administrator needs to login, he must input username and password.
* The password of administrator is encrypted in database. The encryption type is MD5 encryption. So if the hacker accessed the database by SQL injection, he do not get plain text password.
* Authentication in each page.
* Secret code is send to the administrator’s email address on resetting password. So the administrator only can reset the password. This is an extra security provided by system.

CONCLUSION

**CONCLUSION**

The project “ServerZilla” connects the web server with FTP server and allows the admin to transfer the files between them. Also it allows the admin to browse files, rename files and directories, and delete files and directories. The output or error messages are shown to admin using the BootBox script which helps to show alert boxes and confirm boxes which have better user interface than traditional JavaScript alert boxes. The main objective of project is to allow admin to connect the web server with FTP server in an easy and efficient way. The objective is achieved in this project.

LIMITATIONS

**LIMITATIONS**

* The project is single threaded.
* Due to single thread, it allows only one action at a time.
* It cannot upload more than one file at a time or more than one file download at a time.
* It cannot calculate the remaining time to complete the actions like upload or download.
* It cannot show the progress of upload or download.

FUTURE ENHANCEMENT

**FUTURE ENHANCEMENT**

* Enable multithreaded using PHP and JavaScript.
* Provide better interfaces.
* Enable AJAX to load the web pages, and it allow admin to reduce time to load the pages.
* Add addition features like file creating, editing, etc. which are lack now.

BIBLIOGRAPHY

**BIBLIOGRAPHY**

**Books referred**

* **The complete reference PHP**
  + Steven Holzner
  + Tata McGraw-Hill
* **Bootstrap**
  + Jake Spurlock
  + O’Reilly
* **jQuery 1.4 Animation Techniques**
  + Dan Wellman
  + PACKT

**Websites Referred**

* [www.bootstrap.com](http://www.bootstrap.com)
* [www.getbootstrap.com](http://www.getbootstrap.com)
* [www.stackoverflow.com](http://www.stackoverflow.com)

**Search Engines**

* Google ([www.google.co.in](http://www.google.co.in))