A Project Report On

**College Admission Cell Management System**

Submitted for Partial Fulfilment of the Requirements for the Degree of

**BACHELOR OF SCIENCE(IT)**

**BY**

**Tabish jamal (20204521015)**

Under the guidance of

**Mr. Arvinder Singh Hira**

**Department of BSc. (IT)**



**BFIT Groups Of Institutions, Suddhowala, Dehradun.**

****

**Hemawati Nandan Bahuguna Garwhal University, Srinagar**

**CERTIFICATE**

I am Tabish Jamal hereby certify that the work which is being presented in the Bsc.IT

(Bachelor of Science. Information Technology) Project Report entitled **“College Admission Cell Management System”,**

in partial fulfillment of the requirements for the award of the **Bachelor of Science**

**Information Technology** and submitted to the Department of Bsc.IT of BFIT Group Of Institutionsis an under the guidance and supervision of **Mr. Arvinder Singh Hira**.

The result embodied in this report has not been submitted to any other university or

institute for the award of any degree.

**Tabish Jamal Mr. Arvinder Singh Hira**

**(20204521015)**

#### 

**DECLARATION**

I hereby declare that the major Project report entitled: - **“College Admission Cell Management System”** is the result of the project work done by me under the supervision of **Mr. Arvinder Singh Hira**.

The matter embodied in this report has not been submitted by me for the award of any other degree.

Date: - \_ \_/\_ \_/\_ \_ \_ \_ **Signature of Student**

**Tabish Jamal**

**(Bsc.IT)**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

**Signature of HOD Signature of Supervisor**

**(Mr. Arvinder Singh Hira) (Mr. Arvinder Singh Hira)**

**(Department Of BSc. IT) (Department of BSc.IT)**

Date: -\_ \_/\_ \_/\_ \_ \_ \_

**ACKNOWLEDGEMENT**

I take this opportunity to express my sincere thanks and deep gratitude to

all those people who extended their wholehearted co-operation and have

helped me in completing this project successfully. Special thanks to **Mr.**

**Arvinder Singh Hira**, Project Head for all the help and guidance extended

to me by him in every stage during my training. His inspiring suggestions

and timely guidance enabled me to perceive the various aspects of the

project in a new light. I am highly indebted and graceful for his strict

supervision, constant encouragement, inspiration and guidance, which

ensure the worthiness of my work. Working under them was an enrich

experience. I express my sincere thanks to my senior of software

development department for their encouragement and valuable suggestion.

Name: - Tabish Jamal

BSc.IT: - 6th Sem

Roll No: - **20204521015**

* **Table of Contents: -**
* Chapter1: - Introduction
* Chapter2: - Title of the Project

* Chapter3: - Objectives
* Chapter 4: - System Requirement Specification
* Chapter 5: - System Analysis
* Chapter 6: - Integrated Development Environments
* Chapter 7: - Design

* Chapter 8: - Introduction to MySQL

* Chapter 9: - Coding Implementation
* Chapter10: - Programming

* Chapter 11: - Input/output from (Screen layout)
* Chapter 12: - Limitation and drawback
* Chapter 13: - Conclusion

**INTRODUCTION**

This project report to is submitted in partial fulfillment of the requirement for **BSC.IT**. The course includes a topic **“College Admission Cell Management System”**.

* Welcome to the **College Admission Cell Management System** This project aims to streamline and automate the admission process for colleges, ensuring a seamless and efficient experience for both students and administrators. By leveraging modern technology, this system revolutionizes the way college admissions are managed, eliminating the cumbersome paperwork and simplifying the entire process.
* The registration page serves as the gateway for students to provide their essential information, enabling the system to generate a table number for them to proceed with the counseling process. This intuitive system offers a comprehensive set of information fields, tailored to capture all the necessary details from prospective students.
* By implementing this College Admission Cell Management System, colleges can alleviate the administrative burden associated with admissions, optimize resource allocation, and enhance the overall efficiency of the process. Students, on the other hand, benefit from a user-friendly interface that guides them through the registration process, making it easier and more convenient to apply for admission.

Key Features:

* Seamless Registration Process: The system offers a user-friendly and intuitive registration page where prospective students can provide their personal details, academic qualifications, and other relevant information required for admission.
* Customizable Information Fields: The system provides a comprehensive set of information fields that can be customized to meet the specific requirements of each college or university. This ensures that all the necessary information is captured accurately and efficiently.
* Automated Table Number Generation: Once the registration form is submitted, the system automatically generates a unique table number for each student. This table number serves as a reference for the counseling process, allowing students to easily proceed to their designated counseling sessions.
* Streamlined Counseling Process: The College Admission Cell Management System facilitates a smooth and organized counseling process. The generated table numbers help administrators allocate students to their respective counseling sessions, reducing confusion and ensuring a systematic approach to the admissions process.
* Efficient Data Management: All the data provided by students during the registration process is securely stored in a centralized database. This enables easy access and retrieval of information whenever required, eliminating the need for manual paperwork and reducing the chances of data loss or duplication.
* In conclusion, the College Admission Cell Management System revolutionizes the traditional admission process by introducing a seamless and efficient digital solution. By leveraging modern technology, this system optimizes the workflow, enhances data management, and provides a convenient experience for both students and administrators. With this innovative system in place, colleges can efficiently manage their admissions while ensuring a smoother journey for aspiring students.

**ABSTRACT OF THE PROJECT**

An online college admission management system project that holds online admission of various colleges on a web application and serves student and colleges accordingly. The system is designed to allow colleges to set up admission to various courses in the college online and students to register in the portal and apply for various courses in different colleges.

**Online college admission management system project consists of the following parts:**

The project aims to streamline and automate the college admission process, specifically focusing on the registration and counseling phases. By utilizing modern technologies and database management systems, this system revolutionizes the traditional manual approach to admissions, offering an efficient and user-friendly experience for both students and administrators.

The system features a registration page that allows prospective students to provide their personal information, academic qualifications, and other relevant details required for admission. The registration form consists of various information fields tailored to capture all the necessary information accurately. Once the form is submitted, the system validates the data and generates a unique table number for each student.

The table number serves as a reference for students to proceed with the counseling process. It is automatically assigned by the system based on the information provided during registration. The table number ensures an organized and systematic approach to the counseling sessions, allowing administrators to allocate students to their designated sessions efficiently.

The system utilizes a backend built with PHP and MySQL to securely store and manage the student data. The data is stored in a centralized database, ensuring easy retrieval and access whenever required. This eliminates the need for manual paperwork, reduces data duplication, and enhances data management efficiency.

The College Admission Cell Management System offers several benefits. It simplifies the registration process for students, providing them with a user-friendly interface and reducing the time and effort required to apply for admission. For administrators, the system optimizes resource allocation, reduces paperwork, and provides a centralized database for efficient data management.

Overall, this project presents a comprehensive solution for managing college admissions, streamlining the registration and counseling process, and improving the overall efficiency of the college admission cell. By leveraging HTML, CSS, JavaScript, PHP, and MySQL, the system provides an intuitive and reliable platform for colleges and universities to handle admissions seamlessly.

**TITLE OF THE PROJECT**

The Project Title **“College Admission Cell Management System”** is a Software which Has the work of backend and frontend. Which allows the user to interact the software and check out the features. Here user can see the see of the given information about themselves which is given by themselves and facilities which is provided by this system. User can contact and query about the question.

* User can login by the user id/emergency id and check the extra features.
* Basically, this software represents the given information of user by which user can get overview to think about the system that is it useful or not.
* They can also register themselves only and give information another day.
* It has attractive design which is very user friendly can easy access by any user.

**Benefits of College Admission Cell Management System**

**College Admission Cell Management System** provide you the significant benefits. They are as follow:

* Time savings.
* Improved Control.
* Efficient use of resources.
* Minimum Human Source Use.

**Functions that is available for User: -**

* Registration
* Login
* Find the Course
* Find a Counselor Details

**Objectives of the Project**

The College Admission Cell Management System requires a system that will handle all the necessary and minor details easily and proper database security accordingly to the user. They require software, which will store data about user, their family members, their nearest police stations, their nearest power stations, their nearest fire stations, their nearest hospitals.

* Streamline the Admission Process: The primary objective of the project is to streamline the college admission process by providing a user-friendly registration page. The system aims to simplify the collection of student information and ensure a smooth flow of data for efficient admission processing.
* Automate Table Number Generation: The project aims to automate the generation of a unique table number for each student based on the information provided during registration. This objective eliminates the manual assignment of table numbers and ensures a systematic approach to the counseling process.
* Enhance Data Accuracy: By implementing the College Admission Cell Management System, the project aims to improve the accuracy of student data collection. The registration page includes various information fields that capture essential details, reducing the chances of errors and discrepancies.
* Improve Resource Allocation: The system intends to optimize resource allocation by providing administrators with an organized and centralized database. This objective helps in efficiently assigning students to their designated counseling sessions, reducing confusion and saving time for both students and staff.
* Ensure Data Security: The project focuses on maintaining the security and confidentiality of student information. By utilizing PHP and MySQL, the system ensures secure storage and access control measures, safeguarding sensitive data from unauthorized access or manipulation.
* Enhance User Experience: The College Admission Cell Management System aims to provide a user-friendly interface for both students and administrators. The objective is to create an intuitive and visually appealing registration page that guides students through the process and offers a seamless experience.
* Simplify Administrative Tasks: The project aims to simplify administrative tasks associated with the admission process. By automating data management, generating table numbers, and providing easy access to student information, the system reduces the administrative burden and enables staff to focus on other critical aspects of admissions.
* Facilitate Decision-Making: The system aims to provide administrators with efficient data retrieval and reporting capabilities. This objective helps in analyzing admission trends, tracking student data, and generating insightful reports, facilitating informed decision-making and strategic planning.

Overall, the objectives of the College Admission Cell Management System project revolve around streamlining the admission process, automating tasks, improving data accuracy, and enhancing the overall experience for students and administrators. By leveraging HTML, CSS, JavaScript, PHP, and MySQL, the project aims to provide a robust and efficient solution for managing college admissions.

**System Requirements Specification**

**Hardware Requirements**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Description** | **Components** |
| **1** | Processor | Intel Dual Core or above |
| 2 | Ram | 1GB or above |
| 3 | Hard disk/ SSD | 50GB or above |
| 4 | Processor Speed | 1.0GHZ or above |

**Software Requirements**

|  |  |  |
| --- | --- | --- |
| **S. No** | **Description** | **TYPE** |
| 1 | Operation system | Windows |
| 2 | Languages | PHP, JavaScript, Html, CSS |
| 3 | Data Base | MySQL |
| 4 | Web hosting | Apache |

**Platform Requirements**

|  |  |  |
| --- | --- | --- |
| **S. No** | **Description** | **Type** |
| 1 | Tool | Apache Lamp Stack |
| 2 | Coding Platform | Visual Studio |
| 3 | Execution Platform | CMD/visual studio IDE |

**SOFTWARE REQUIREMENTS**

* Operating system- Windows 10 is used as the operating system as it is stable and supports more features and is more user friendly.
* Development tools and Programming language- PHP is used to write the whole backend code and develop software. HTML, CSS and JavaScript is used for frontend code.

**HARDWARE REQUIREMENTS**

* Intel core i5 5th generation is used as a processor because it is fast than other processors and provide reliable and stable and we can run our pc for longtime. By using this processor, we can keep on developing our project without any worries.
* Ram 1 GB is used as it will provide fast reading and writing capabilities and will in turn support in processing.

**System Analysis**

System Analysis and design are very important aspects of any system development process. These two are the steps of SDLC.

SDLC stands for system development life cycle. Which represents the actual processes happened during the development of any system.

**Feasibility Study**

* The first study aspect is whether the current project is technically feasible i.e. whether the project be carried out with the current equipment, existing software and available personnel. If new technology is required than what is the likelihood that it can be developed?
* The second study aspect is whether the project is economically feasible i.e. are

there sufficient benefits in creating the system to make the cost acceptable. Are the costs of not creating the system so great that the project must be undertaken?

* The third study aspect is whether the project is operationally feasible or not i.e. whether the system will be used if it is developed and implemented? Project is worth developing only if it can meet institutions operating requirements.
* The feasibility study proposes one or more conceptual solutions to the problem set for the project. The objective in assessing feasibility is to determine whether a development project has a reasonable chance of success. It helps us to determine the input & output of the system. The following are the criteria that are considered to confirm the project feasibility.

The following feasibility study was undertaken for the proposed system:

* **Technical feasibility: -**

At first, it’s necessary to check that the proposed system is technically feasible or not and to determine the technology and skill necessary to carry out the project. If they are not available, then find out the solution to obtain them. Hardware is already available in the collage.

* **Economical feasibility: -**

While considering economic feasibility, it is checked in points like performance, information and outputs from the system. MS Access is available in one package of the windows operating system & does not require additional software cost for the client tools. The cost incurred to develop the system is freeware & does not incur the cost to the project. Backend database technology is a freeware. This justifies economical feasibility of the system. Social feasibility: Although generally there is always resistance, initially to any change in the system is aimed at reliving the work load of the users to extent the system is going to facilitate user to perform operations like calculating salary amounts and deductions, generating reports with less possible errors. Thus there is no reason to make system socially unfeasible

* **Operational feasibility: -**

The operational feasibility is obtained by consulting with the system users. Check that proposed solution satisfies the user needs or not. There is no resistance from employee since new system is helpful. The existing system is manual system, while the new system is computerized and extremely user friendly.

**Phased development process: -**

A development process consists of various phases, each phase ending with a defined output. The main reason for having a phased process is that it breaks the problem of developing software into successfully performing a set of phases, each handling a different concern of software development.

**Requirement Analysis: -**

* Requirements analysis is done in order to understand the problem the software system is to solve. The goal of the requirements activity is to document the requirements in a software requirements specification document.
* There are two major activities in this phase: Problem Understanding or Analysis and Requirement Specification. In problem analysis, the aim is to understand the problem and its context, and the requirements of the new system that is to be developed.
* Once the problem is analyzed and essentials understood, the requirements must be specified in the requirements specification document. The requirements specification document. The requirement document must specify all functional and performance requirements; the formats of inputs and output; and all design constraints that exist due to political, economic, environmental, and security reasons.

**Problem Analysis**

In my project main object was to improve the user interface, making it user friendly. During problem analysis, we have listed”

* + - Manually working and much of paper work.
    - Redundancy, lack of integrity and many other problems of manual system.

**Coding: -**

* The goal of the coding phase is to translate the design of the system into code in a given programming language. For a given design, the aim in this phase is to implement the design in the best possible way.
* The coding phase affects both testing and maintenance profoundly. Well-written code can reduce the testing and maintenance effort. The testing and maintenance costs of software are much higher than coding cost. Hence during coding the focus should be developing programs that are easy to read and understand, and not simply on developing programs that are easy to write. Simplicity and clarity should be strived for during the coding phase.
* PHP is used for backend query from MySQL database.

**Language: -**

* We used PHP technology to create this system.
* PHP is a popular server-side scripting language that is widely used for web development. It's known for its ease of use and integration with HTML, making it a great choice for creating dynamic web pages.

**PHP: -**

PHP (Hypertext Preprocessor) is a server-side scripting language used for web development. It is embedded within HTML code and is executed on the web server, generating dynamic web pages. Here are some key points to understand about PHP:

* Syntax: PHP code is enclosed within <?php ?> tags. It can be embedded within HTML or written as standalone PHP scripts.
* Variables: In PHP, variables start with a dollar sign ($) followed by the variable name. Variables are dynamically typed, meaning you don't need to declare the data type explicitly.
* Data Types: PHP supports various data types, including strings, integers, floats, booleans, arrays, objects, and more.
* Control Structures: PHP provides control structures such as if-else statements, switch statements, loops (for, while, do-while), and more to control the flow of execution.
* Functions: PHP has a wide range of built-in functions for performing common tasks. You can also create your own functions to modularize your code and reuse it.
* Form Handling: PHP is commonly used for processing form data. You can retrieve form input using the $\_POST or $\_GET superglobals and perform validations or database operations.
* Database Connectivity: PHP has built-in functions to connect to databases like MySQL, PostgreSQL, and others. You can execute queries, fetch results, and manipulate data using PHP.
* Object-Oriented Programming (OOP): PHP supports object-oriented programming concepts like classes, objects, inheritance, and polymorphism.
* File Handling: PHP provides functions for reading from and writing to files, creating directories, and performing file-related operations.
* Error Handling: PHP has error reporting and exception handling mechanisms to handle errors and exceptions gracefully.

### Automatic memory management

### Automatic memory management

In PHP, automatic memory management is handled by a technique called "garbage collection." Here are key points about automatic memory management in PHP:

* Garbage Collection: PHP uses a garbage collector to automatically manage memory. The garbage collector identifies and frees up memory that is no longer in use, allowing developers to focus on writing code without explicitly deallocating memory.
* Reference Counting: PHP's garbage collector primarily uses a reference counting mechanism to track memory usage. Each variable in PHP is associated with a reference count that keeps track of the number of references to that variable. When the reference count reaches zero, indicating that no references exist, the memory occupied by the variable is automatically released.
* Cyclic References: PHP's garbage collector also handles cyclic references, which occur when objects reference each other in a circular manner. The garbage collector identifies and breaks these cycles, ensuring that memory is properly deallocated even in such scenarios.
* Mark-and-Sweep Algorithm: In addition to reference counting, PHP's garbage collector employs a mark-and-sweep algorithm to detect and collect unreachable objects. This algorithm traverses the object graph, marking objects that are still reachable and sweeping away those that are not.
* Performance Implications: While automatic memory management in PHP simplifies memory handling for developers, it can introduce some performance overhead. The garbage collector needs to periodically run to reclaim unused memory, which can result in slight performance fluctuations during garbage collection cycles.
* Configuration Options: PHP provides configuration options to fine-tune garbage collection behavior, such as setting the frequency of garbage collection cycles and memory thresholds for triggering collection. These options allow developers to optimize memory management based on the specific requirements of their applications.

Overall, automatic memory management in PHP through garbage collection helps alleviate the burden of manual memory allocation and deallocation. It ensures efficient memory usage and reduces the risk of memory leaks and memory-related errors in PHP applications.

## Class libraries

PHP is a popular programming language for web development, and it has a wide range of class libraries available for various purposes. Class libraries in PHP are collections of reusable code and classes that provide pre-built functionality to simplify and speed up the development process.

Here are some commonly used PHP class libraries:

1. Laravel: Laravel is a powerful and widely-used PHP framework that provides a comprehensive set of tools and libraries for web application development. It includes features such as routing, database ORM (Object-Relational Mapping), authentication, caching, and more.
2. Symfony: Symfony is another popular PHP framework that follows the MVC (Model-View-Controller) architectural pattern. It offers a rich set of components and libraries for building robust and scalable applications. Symfony components can also be used independently in non-Symfony projects.
3. PHPUnit: PHPUnit is a unit testing framework for PHP. It provides a set of classes and methods for writing and running tests to ensure the correctness of your code. PHPUnit supports various testing features such as assertions, test suites, fixtures, and mocking.
4. Guzzle: Guzzle is a powerful HTTP client library for PHP. It simplifies the process of making HTTP requests and handling responses. Guzzle supports features like handling cookies, sending JSON data, uploading files, and more.
5. Doctrine: Doctrine is an ORM (Object-Relational Mapping) library for PHP. It provides a convenient way to work with databases by mapping database tables to PHP objects. Doctrine supports query building, entity relationships, caching, and other database-related tasks.
6. Twig: Twig is a templating engine for PHP that provides a clean and easy-to-use syntax for creating dynamic templates. It separates the presentation logic from the application logic and offers features like template inheritance, filters, and macros.

* Monolog: Monolog is a logging library for PHP. It allows you to log messages and events from your application to various output channels, such as files, databases, or even remote services. Monolog supports different log levels, formatting options, and log handlers.

# Integrated Development Environment

## Overview

* An **integrated development environment (IDE)** is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools, and a debugger. Most modern IDEs have intelligent code completion.
* Some IDEs contain a compiler, interpreter, or both. The boundary between an integrated development environment and other parts of the broader software development environment is not well-defined. Sometimes a version control system, or various tools to simplify the construction of a graphical user interface (GUI), are integrated. Many modern IDEs also have a class browser, an object browser, and a class hierarchy diagram, for use in object-oriented software development.

## Discussion

The development of a College Admission Cell Management System utilizing HTML, CSS, JavaScript, PHP, and MySQL brings several advantages and opportunities for improving the college admission process. Let's delve into some key points for discussion:

* Streamlined Registration Process: The registration page with multiple information fields allows students to provide comprehensive details required for admission. This streamlines the registration process by capturing all the necessary information in a structured manner, reducing the chance of missing or incomplete data.
* Automation for Efficient Counseling: The automated generation of table numbers based on the provided information simplifies the counseling process. It eliminates the manual assignment of table numbers and ensures a systematic approach to counseling sessions. This automation saves time and minimizes the chances of errors or discrepancies in table number assignments.
* Data Accuracy and Validation: By implementing proper validation techniques, the system ensures that the entered data is accurate and consistent. Error handling mechanisms can prompt users to correct any invalid or missing information, reducing data entry errors and enhancing data accuracy.
* Centralized Database Management: The use of MySQL as the database management system allows for efficient storage and retrieval of student information. Storing data in a centralized database enables easy access, searchability, and reduces the need for physical paperwork. It also facilitates data integrity and security through proper backup and access control mechanisms.
* User-Friendly Interface: The registration page's user-friendly interface, designed using HTML, CSS, and JavaScript, enhances the overall user experience. It guides students through the registration process and ensures a smooth and intuitive interaction. Clear instructions, proper labeling of fields, and visual cues contribute to a seamless user journey.
* Security Measures: The implementation of security measures, such as user authentication, password encryption, and protection against SQL injection attacks, helps safeguard student data. These security measures are crucial in maintaining the confidentiality and integrity of sensitive information.
* Administrative Efficiency: The system empowers administrators with features to efficiently manage student registrations and counseling sessions. They can easily view, edit, and update student information, assign students to counseling sessions, and generate reports for analysis and decision-making. This improves administrative efficiency and reduces manual paperwork.
* Potential for Scalability and Adaptability: The modular structure of the system allows for future scalability and adaptability. Additional features, data fields, or integration with other systems can be implemented as per evolving requirements. This flexibility ensures that the system can accommodate future growth and changes in the admission process.
* Collaboration and Communication: The system can facilitate effective collaboration and communication between students and administrators. Students can easily submit their information, while administrators can efficiently manage and communicate important updates or changes in the admission process. This enhances transparency and improves communication channels.
* Continuous Improvement: The development of the College Admission Cell Management System is an iterative process. Feedback from users and stakeholders can be collected to identify areas for improvement. Regular updates, bug fixes, and feature enhancements can be implemented to refine the system's functionality and address emerging needs.

**Integrated Development Environment or IDE**

* Upon starting the IDE software, the programmer usually indicates the file he or she wants to open for editing as source code. As they make changes they might either do a “save as” or “save”. When they have finished entering the source code, they usually direct the IDE to “compile & run” the program. The IDE does the following steps:

1.If there are any unsaved changes to the source code file it has the **test editor** save the changes.

2.The **compiler** opens the source code file and does its **first step** which is executing the **pre-processor** compiler directives and other steps needed to get the file ready for the second step. The #include will insert header files into the code at this point. If it encounters an error, it stops the process and returns the user to the source code file within the text editor with an error message. If no problems encountered it saves the source code to a temporary file called a translation unit.

3.The **compiler** opens the translation unit file and does its **second step** which is **converting** the programming language code to machine instructions for the CPU, a data area, and a list of items to be resolved by the linker. Any problems encountered (usually a syntax or violation of the programming language rules) stops the process and returns the user to the source code file within the text editor with an error message. If no problems encountered it saves the machine instructions, data area, and linker resolution list as an object file.

4.The **linker** opens the program object file and links it with the library object files as needed. Unless all linker items are resolved, the process stops and returns the user to the source code file within the text editor with an error message. If no problems encountered it saves the linked objects as an executable file.

5.The IDE directs the operating system’s program called the **loader** to load the executable file into the computer’s memory and have the Central Processing Unit (CPU) start processing the instructions. As the user interacts with the program, entering test data, he or she might discover that the outputs are not correct. These types of errors are called logic errors and would require the user to return to the source code to change the algorithm.

### Resolving Errors

Despite our best efforts at becoming perfect programmers, we will create errors. Solving these errors is known as **debugging** your program. The three types of errors in the order that they occur are:

* Compiler
* Linker
* Logic

There are two types of compiler errors; pre-processor (1st step) and conversion (2nd step). A review of Figure 1 above shows the four arrows returning to the source code so that the programmer can correct the mistake.

During the conversion (2nd step) the compiler might give a **warning** message which in some cases may not be a problem to worry about. For example: Data type demotion may be exactly what you want your program to do, but most compilers give a warning message. Warnings don’t stop the compiling process but as their name implies, they should be reviewed.

## Key Terms

* **Compiler**
* Converts source code to object code.
* **Debugging**
* The process of removing errors from a program. 1) compiler 2) linker 3) logic.
* **Linker**
* Connects or links object files into an executable file.
* **Loader**
* Part of the operating system that loads executable files into memory and directs the CPU to start running the program.
* **Pre-processor**
* The first step the compiler does in converting source code to object code.
* **Text editor**
* A software program for creating and editing ASCII text files.
* **Warning**
* A compiler alert that there might be a problem.

**Source Code Editor**

A **source-code editor** is a [text editor](https://en.wikipedia.org/wiki/Text_editor) program designed specifically for editing [source code](https://en.wikipedia.org/wiki/Source_code) of [computer programs](https://en.wikipedia.org/wiki/Computer_program). It may be a standalone application or it may be built into an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) or [web browser](https://en.wikipedia.org/wiki/Web_browser). Source-code editors are a fundamental [programming tool](https://en.wikipedia.org/wiki/Programming_tool), as the fundamental job of programmers is to write and edit source code.

* **Features**
* Source-code editors have features specifically designed to simplify and speed up typing of source code, such as [syntax highlighting](https://en.wikipedia.org/wiki/Syntax_highlighting), [indentation](https://en.wikipedia.org/wiki/Indentation_style), [autocomplete](https://en.wikipedia.org/wiki/Autocomplete) and [brace matching](https://en.wikipedia.org/wiki/Brace_matching) functionality.
* These editors also provide a convenient way to run a [compiler](https://en.wikipedia.org/wiki/Compiler), [interpreter](https://en.wikipedia.org/wiki/Interpreter_(computing)), [debugger](https://en.wikipedia.org/wiki/Debugger), or other program relevant for the [software-development process](https://en.wikipedia.org/wiki/Software-development_process).
* So, while many text editors like [Notepad](https://en.wikipedia.org/wiki/Microsoft_Notepad) can be used to edit source code, if they don't enhance, automate or ease the editing of code, they are not *source-code editors*.
* [Structure editors](https://en.wikipedia.org/wiki/Structure_editor) are a different form of source-code editor, where instead of editing raw text, one manipulates the code's structure, generally the [abstract syntax tree](https://en.wikipedia.org/wiki/Abstract_syntax_tree).
* In this case features such as syntax highlighting, validation, and code formatting are easily and efficiently implemented from the [concrete syntax tree](https://en.wikipedia.org/wiki/Concrete_syntax_tree) or abstract syntax tree, but editing is often more rigid than free-form text.
* Structure editors also require extensive support for each language, and thus are harder to extend to new languages than text editors, where basic support only requires supporting syntax highlighting or indentation.
* For this reason, strict structure editors are not popular for source code editing, though some IDEs provide similar functionality.
* A source-code editor can check syntax while code is being entered and immediately warn of syntax problems.
* A few source-code editors compress source code, typically converting common keywords into single-byte tokens, removing unnecessary whitespace, and converting numbers to a binary form.
* Such tokenizing editors later uncompress the source code when viewing it, possibly [prettyprinting](https://en.wikipedia.org/wiki/Prettyprint" \o "Prettyprint) it with consistent capitalizing and spacing. A few source-code editors do both.
* **Compiler**
* In [computing](https://en.wikipedia.org/wiki/Computing), a **compiler** is a [computer program](https://en.wikipedia.org/wiki/Computer_program) that [translates](https://en.wikipedia.org/wiki/Translator_(computing)) computer code written in one [programming language](https://en.wikipedia.org/wiki/Programming_language) (the *source* language) into another language (the *target* language).
* The name "compiler" is primarily used for programs that translate [source code](https://en.wikipedia.org/wiki/Source_code) from a [high-level programming language](https://en.wikipedia.org/wiki/High-level_programming_language) to a [lower level language](https://en.wikipedia.org/wiki/Lower_level_language) (e.g. [assembly language](https://en.wikipedia.org/wiki/Assembly_language), [object code](https://en.wikipedia.org/wiki/Object_code), or [machine code](https://en.wikipedia.org/wiki/Machine_code)) to create an [executable](https://en.wikipedia.org/wiki/Executable) program.

* There are many different types of compilers which produce output in different useful forms. A [*cross-compiler*](https://en.wikipedia.org/wiki/Cross-compiler) produces code for a different [CPU](https://en.wikipedia.org/wiki/Central_processing_unit) or [operating system](https://en.wikipedia.org/wiki/Operating_system) than the one on which the cross-compiler itself runs.
* A [*bootstrap compiler*](https://en.wikipedia.org/wiki/Bootstrap_compiler) is written in the language that it intends to compile. A program that translates from a low-level language to a higher level one is a *[decompiler](https://en.wikipedia.org/wiki/Decompiler" \o "Decompiler)*.
* A program that translates between high-level languages is usually called a [*source-to-source compiler*](https://en.wikipedia.org/wiki/Source-to-source_compiler) or *transpiler*.
* A language [*rewriter*](https://en.wikipedia.org/wiki/Rewriting) is usually a program that translates the form of [expressions](https://en.wikipedia.org/wiki/Expression_(computer_science)) without a change of language. A [*compiler-compiler*](https://en.wikipedia.org/wiki/Compiler-compiler) is a compiler that produces a compiler (or part of one).
* Some early milestones in the development of compiler technology:
* **1952**: An [Autocode](https://en.wikipedia.org/wiki/Autocode" \o "Autocode) compiler developed by [Alick Glennie](https://en.wikipedia.org/wiki/Alick_Glennie) for the [Manchester Mark I](https://en.wikipedia.org/wiki/Manchester_Mark_I) computer at the University of Manchester is considered by some to be the first compiled programming language.
* **1952**: [Grace Hopper](https://en.wikipedia.org/wiki/Grace_Hopper)'s team at [Remington Rand](https://en.wikipedia.org/wiki/Remington_Rand) wrote the compiler for the [A-0](https://en.wikipedia.org/wiki/A-0_System) programming language (and coined the term *compiler* to describe it), although the A-0 compiler functioned more as a loader or linker than the modern notion of a full compiler.
* **1954–1957**: A team led by [John Backus](https://en.wikipedia.org/wiki/John_Backus) at [IBM](https://en.wikipedia.org/wiki/IBM) developed [FORTRAN](https://en.wikipedia.org/wiki/Fortran) which is usually considered the first high-level language. In 1957, they completed a FORTRAN compiler that is generally credited as having introduced the first unambiguously complete compiler.
* **1959**: The Conference on Data Systems Language (CODASYL) initiated development of [COBOL](https://en.wikipedia.org/wiki/COBOL). The COBOL design drew on A-0 and FLOW-MATIC. By the early 1960s COBOL was compiled on multiple architectures.
* **1958–1962**: [John McCarthy](https://en.wikipedia.org/wiki/John_McCarthy_(computer_scientist)) at [MIT](https://en.wikipedia.org/wiki/MIT) designed [LISP](https://en.wikipedia.org/wiki/Lisp_(programming_language)). The symbol processing capabilities provided useful features for artificial intelligence research. In 1962, LISP 1.5 release noted some tools: an interpreter written by Stephen Russell and Daniel J. Edwards, a compiler and assembler written by Tim Hart and Mike Levin.
* **Interpreter**
* In [computer science](https://en.wikipedia.org/wiki/Computer_science), an **interpreter** is a translator ([computer program](https://en.wikipedia.org/wiki/Computer_program)) that repeatedly reads instructions (one at a time) and translates them to machine code.

It then [executes](https://en.wikipedia.org/wiki/Execution_(computers)) instructions written in a [programming](https://en.wikipedia.org/wiki/Programming_language) or [scripting language](https://en.wikipedia.org/wiki/Scripting_language), without requiring them previously to have been [compiled](https://en.wikipedia.org/wiki/Compiler) into a [machine language](https://en.wikipedia.org/wiki/Machine_language) program.

* An interpreter generally uses one of the following strategies for program execution:
* [Parse](https://en.wikipedia.org/wiki/Parse) the [source code](https://en.wikipedia.org/wiki/Source_code) and perform its behavior directly.
* [Translate](https://en.wikipedia.org/wiki/Translator_(computing)) source code into some efficient [intermediate representation](https://en.wikipedia.org/wiki/Intermediate_representation) or [object code](https://en.wikipedia.org/wiki/Object_code) and immediately execute this.
* Explicitly execute stored precompiled codemade by a [compiler](https://en.wikipedia.org/wiki/Compiler) which is part of the interpreter system.
* Early versions of [Lisp programming language](https://en.wikipedia.org/wiki/Lisp_programming_language) and [minicomputer and microcomputer BASIC dialects](https://en.wikipedia.org/wiki/BASIC_interpreter) would be examples of the first type.
* [Perl](https://en.wikipedia.org/wiki/Perl), [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [MATLAB](https://en.wikipedia.org/wiki/MATLAB), and [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)) are examples of the second, while [UCSD Pascal](https://en.wikipedia.org/wiki/UCSD_Pascal) is an example of the third type.
* Source programs are compiled ahead of time and stored as machine independent code, which is then [linked](https://en.wikipedia.org/wiki/Linker_(computing)) at run-time and executed by an interpreter and/or compiler (for [JIT](https://en.wikipedia.org/wiki/Just-in-time_compilation) systems).
* Some systems, such as [Smalltalk](https://en.wikipedia.org/wiki/Smalltalk) and contemporary versions of [BASIC](https://en.wikipedia.org/wiki/BASIC) and [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) may also combine two and three.
* Interpreters of various types have also been constructed for many languages traditionally associated with compilation, such as [Algol](https://en.wikipedia.org/wiki/ALGOL), [Fortran](https://en.wikipedia.org/wiki/Fortran), [Cobol](https://en.wikipedia.org/wiki/COBOL), [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B).
* While interpretation and compilation are the two main means by which programming languages are implemented, they are not mutually exclusive, as most interpreting systems also perform some translation work, just like compilers.
* The terms "[interpreted language](https://en.wikipedia.org/wiki/Interpreted_language)" or "[compiled language](https://en.wikipedia.org/wiki/Compiled_language)" signify that the canonical implementation of that language is an interpreter or a compiler, respectively.
* A [high-level language](https://en.wikipedia.org/wiki/High-level_programming_language) is ideally an [abstraction](https://en.wikipedia.org/wiki/Abstraction_(computer_science)) independent of particular implementations.
* **Debugger**
* A **debugger** or **debugging tool** is a [computer program](https://en.wikipedia.org/wiki/Computer_program) used to [test](https://en.wikipedia.org/wiki/Software_testing) and [debug](https://en.wikipedia.org/wiki/Debugging) other programs (the "target" program). The main use of a debugger is to run the target program under controlled conditions that permit the programmer to track its operations in progress and monitor changes in computer resources (most often memory areas used by the target program or the computer's operating system) that may indicate malfunctioning code.
* Typical debugging facilities include the ability to run or halt the target program at specific points, display the contents of memory, CPU registers or storage devices (such as disk drives), and modify memory or register contents in order to enter selected test data that might be a cause of faulty program execution.
* The code to be examined might alternatively be running on an [*instruction set simulator*](https://en.wikipedia.org/wiki/Instruction_set_simulator) (ISS), a technique that allows great power in its ability to halt when specific conditions are encountered, but which will typically be somewhat slower than executing the code directly on the appropriate (or the same) processor. Some debuggers offer two modes of operation, full or partial simulation, to limit this impact.
* A "[trap](https://en.wikipedia.org/wiki/Trap_(computing))" occurs when the program cannot normally continue because of a [programming bug](https://en.wikipedia.org/wiki/Software_bug) or invalid data. For example, the program might have tried to use an instruction not available on the current version of the [CPU](https://en.wikipedia.org/wiki/Central_processing_unit) or attempted to access unavailable or [protected](https://en.wikipedia.org/wiki/Memory_protection) [memory](https://en.wikipedia.org/wiki/Computer_memory).
* When the program "traps" or reaches a preset condition, the debugger typically shows the location in the original code if it is a **source-level debugger** or **symbolic debugger**, commonly now seen in [integrated development environments](https://en.wikipedia.org/wiki/Integrated_development_environment).
* If it is a **low-level debugger** or a **machine-language debugger** it shows the line in the [disassembly](https://en.wikipedia.org/wiki/Disassembly) (unless it also has online access to the original source code and can display the appropriate section of code from the assembly or compilation).

**Debugging in pHp**

* PHP provides several debugging tools and techniques that can be used to diagnose and fix issues in your PHP code. Here are some commonly used debugging tools for PHP:
* var\_dump(): This function is a basic debugging tool that outputs the structure and values of variables or expressions. It can be used to inspect the contents of variables at specific points in your code.
* print\_r(): Similar to var\_dump(), print\_r() is used to print human-readable information about a variable or expression. It is particularly useful for displaying arrays and objects in a more readable format.
* error\_reporting(): This function sets the error reporting level at runtime. By adjusting the error reporting level, you can control which types of errors and warnings are displayed. For debugging purposes, you can set it to E\_ALL to report all errors and warnings.
* error\_log(): With error\_log(), you can write custom error messages or debug information to the server's error log file instead of displaying them on the web page. This is helpful when you want to log and track specific information without interrupting the user's experience.
* Xdebug: Xdebug is a powerful debugging and profiling tool for PHP. It provides features like stack traces, code coverage analysis, and remote debugging capabilities. Xdebug can be integrated with IDEs such as PhpStorm, NetBeans, and Visual Studio Code, allowing you to step through your code, set breakpoints, and inspect variables during runtime.
* Debugging Extensions: There are also various debugging extensions available for PHP, such as "phpdbg" and "Zend Debugger". These extensions provide additional debugging functionality, including interactive debugging consoles, step-by-step debugging, and profiling.
* Remember to disable or remove any debugging statements or tools in production environments, as they can expose sensitive information and impact performance.
* **Features**
* Typically, debuggers offer a query processor, a symbol resolver, an expression interpreter, and a debug support interface at its top level.
* Debuggers also offer more sophisticated functions such as running a program [step by step](https://en.wikipedia.org/wiki/Stepping_(debugging)) (**single-stepping** or [program animation](https://en.wikipedia.org/wiki/Program_animation)), stopping (**breaking**) (pausing the program to examine the current state) at some event or specified instruction by means of a [breakpoint](https://en.wikipedia.org/wiki/Breakpoint), and tracking the values of variables.
* Some debuggers have the ability to modify program state while it is running. It may also be possible to continue execution at a different location in the program to bypass a crash or logical error.
* The same functionality which makes a debugger useful for correcting bugs allows it to be used as a [software cracking](https://en.wikipedia.org/wiki/Software_cracking) tool to evade [copy protection](https://en.wikipedia.org/wiki/Copy_protection), [digital rights management](https://en.wikipedia.org/wiki/Digital_rights_management), and other software protection features.
* It often also makes it useful as a general verification tool, [fault coverage](https://en.wikipedia.org/wiki/Fault_coverage), and [performance analyzer](https://en.wikipedia.org/wiki/Profiling_(computer_programming)), especially if [instruction path lengths](https://en.wikipedia.org/wiki/Instruction_path_length) are shown.[]](https://en.wikipedia.org/wiki/Debugger#cite_note-3) Early microcomputers with disk-based storage often benefitted from the ability to diagnose and recover corrupted directory or registry data records, to "undelete" files marked as deleted, or to crack file password protection.
* Most mainstream debugging engines, such as [gdb](https://en.wikipedia.org/wiki/Gdb" \o "Gdb) and [dbx](https://en.wikipedia.org/wiki/Dbx_(debugger)), provide console-based [command line interfaces](https://en.wikipedia.org/wiki/Command_line_interface). [Debugger front-ends](https://en.wikipedia.org/wiki/Debugger_front-end) are popular extensions to debugger engines that provide [IDE](https://en.wikipedia.org/wiki/Integrated_Developer_Environment) integration, [program animation](https://en.wikipedia.org/wiki/Program_animation), and visualization features.
* **Hardware Support For Debugging**
* When it comes to debugging a PHP website, the hardware requirements are typically the same as those needed for running a PHP website in a production environment. The debugging process primarily relies on software tools and configurations rather than specific hardware requirements. However, there are a few considerations that can enhance your debugging experience:
* Sufficient RAM: Ensure that your computer has sufficient RAM to handle the PHP website and the debugging tools simultaneously. This will prevent slowdowns or crashes due to memory constraints.
* Multiple Displays: Working with multiple displays can be beneficial for debugging. It allows you to have the PHP code, debugging tools, and browser open simultaneously, making it easier to analyze and troubleshoot issues.
* Fast Storage: Having a solid-state drive (SSD) or a fast storage solution can improve the overall performance of your development environment, including the speed of loading and saving files during debugging.
* Ample CPU Power: While PHP debugging does not generally require high CPU power, having a capable processor can help ensure smooth execution of your PHP code and debugging tools.
* Network Connectivity: If your PHP website interacts with external services or APIs, ensure that your network connectivity is stable to avoid any potential network-related issues during debugging.
* Remember that while hardware can contribute to a more efficient debugging experience, it is not the sole determining factor. The choice of debugging tools, software configurations, and your familiarity with the debugging process are equally important.
* Most modern microprocessors have at least one of these features in their [CPU design](https://en.wikipedia.org/wiki/CPU_design) to make debugging easier:
* Hardware support for single-stepping a program, such as the [trap flag](https://en.wikipedia.org/wiki/Trap_flag).
* An instruction set that meets the [Popek and Goldberg virtualization requirements](https://en.wikipedia.org/wiki/Popek_and_Goldberg_virtualization_requirements" \o "Popek and Goldberg virtualization requirements) makes it easier to write debugger software that runs on the same CPU as the software being debugged; such a CPU can execute the inner loops of the program under test at full speed, and still remain under debugger control.
* [In-system programming](https://en.wikipedia.org/wiki/In-system_programming) allows an external hardware debugger to reprogram a system under test (for example, adding or removing instruction breakpoints). Many systems with such ISP support also have other hardware debug support.
* Hardware support for code and data [breakpoints](https://en.wikipedia.org/wiki/Breakpoint), such as address comparators and data value comparators or, with considerably more work involved, [page fault](https://en.wikipedia.org/wiki/Page_fault) hardware.

* [JTAG](https://en.wikipedia.org/wiki/Joint_Test_Action_Group#Example:_ARM11_debug_TAP) access to hardware debug interfaces such as those on [ARM architecture](https://en.wikipedia.org/wiki/ARM_architecture) processors or using the [Nexus](https://en.wikipedia.org/wiki/Nexus_(standard)) command set. Processors used in embedded systems typically have extensive JTAG debug support.
* JTAG (Joint Test Action Group) is a standard interface that provides a method for testing and debugging integrated circuits, including microcontrollers, processors, and other digital components on a printed circuit board (PCB). It allows for in-circuit testing, debugging, and programming of these devices.
* The JTAG interface consists of several signals that enable communication and control between a JTAG-compliant device (such as a microcontroller) and a JTAG debugger or programmer. The main signals in the JTAG interface are:
* TCK (Test Clock): This signal provides the clock for JTAG communication. It synchronizes the data transfer between the device and the debugger/programmer.
* TMS (Test Mode Select): TMS controls the state transitions and mode selection of the JTAG interface. It determines the operation mode of the device, such as shifting data, entering/exiting test modes, and initiating boundary scan testing.
* TDI (Test Data Input): TDI is the data input line used to shift data into the device for testing or programming purposes.
* TDO (Test Data Output): TDO is the data output line used to read data from the device during testing or programming. It provides the test results or other information from the device.
* TRST (Test Reset): TRST is an optional signal that resets the JTAG state machine and sets the device to a known state.
* JTAG is widely used in the electronics industry for hardware testing, manufacturing testing, and low-level debugging of embedded systems. It allows developers to perform tasks such as boundary scan testing, real-time debugging, firmware programming, and flash memory programming.
* To utilize the JTAG interface, you typically need a JTAG debugger or programmer that supports the specific JTAG protocol and the target device you are working with. Various hardware tools and software environments, such as OpenOCD, J-Link, and ST-Link, provide JTAG debugging and programming capabilities for different microcontrollers and processors.
* Micro controllers with as few as six pins need to use low pin-count substitutes for JTAG, such as [BDM](https://en.wikipedia.org/wiki/Background_debug_mode_interface), [Spy-Bi-Wire](https://en.wikipedia.org/wiki/Spy-Bi-Wire), or [debugWIRE](https://en.wikipedia.org/wiki/DebugWIRE" \o "DebugWIRE) on the [Atmel AVR](https://en.wikipedia.org/wiki/Atmel_AVR). DebugWIRE, for example, uses bidirectional signaling on the RESET pin.
* **Some widely used debuggers are:**
* [Arm DTT](https://en.wikipedia.org/wiki/Allinea_DDT), formerly known as Allinea DDT
* [Eclipse](https://en.wikipedia.org/wiki/Eclipse_(software)) debugger API used in a range of IDEs:
* Eclipse IDE (Java) [Nodeclipse](https://en.wikipedia.org/wiki/Nodeclipse" \o "Nodeclipse) (JavaScript)
* [Firefox](https://en.wikipedia.org/wiki/Firefox) [JavaScript](https://en.wikipedia.org/wiki/JavaScript) debugger
* [GDB](https://en.wikipedia.org/wiki/GNU_Debugger) - the GNU debugger
* [LLDB](https://en.wikipedia.org/wiki/LLDB_(debugger))
* [Microsoft Visual Studio Debugger](https://en.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger)
* [Radare2](https://en.wikipedia.org/wiki/Radare2)
* [TotalView](https://en.wikipedia.org/w/index.php?title=TotalView_(debugger)&action=edit&redlink=1)
* [Valgrind](https://en.wikipedia.org/wiki/Valgrind)
* WDW, the [OpenWatcom](https://en.wikipedia.org/wiki/Watcom_C/C%2B%2B_compiler" \o "Watcom C/C++ compiler) debugger
* [WinDbg](https://en.wikipedia.org/wiki/WinDbg)
* Earlier [minicomputer](https://en.wikipedia.org/wiki/Minicomputer) debuggers include:
* [Dynamic debugging technique](https://en.wikipedia.org/wiki/Dynamic_debugging_technique) (DDT)
* [On-line Debugging Tool](https://en.wikipedia.org/wiki/On-line_Debugging_Tool) (ODT)
* Earlier [Mainframe](https://en.wikipedia.org/wiki/Mainframe) debuggers include (in date of release order):
* 1974 OLIVER CICS TEST/DEBUG
* 1980 SIMON BATCH TEST/DEBUG
* 1985 [CA/EZTEST](https://en.wikipedia.org/wiki/CA/EZTEST)
* 1990 [XPEDITER](https://en.wikipedia.org/wiki/XPEDITER) and Expediter [CICS](https://en.wikipedia.org/wiki/CICS)
* Current mainframe debuggers:
* Debug Tool for z/OS
* [XPEDITER](https://en.wikipedia.org/wiki/XPEDITER) and Expediter [CICS](https://en.wikipedia.org/wiki/CICS)
* z/XDC

**Design:-**

The next and most important step is to design a system. The system is designed to be user friendly and interactive. It is designed by strictly following the UID (user interface design) concepts. The user can interact with the system with minimal computer knowledge. For this system should satisfy then following requirements.

* Single window services.
* Integrated on-line modules easily customized to specific requirements with minimum time.
* Site implementation & fine-tuning of software for procedures and facilities controlled by user’s definable parameters.
* Software options fully menu-driven with popup windows for ease of operations by user’s highly integrated linkage and auto-posting setup definable by database administrator.
* Flexible for phased installation and integration of modules.
* Incorporation of special features and hot-key button for help.
* **Data Flow Diagram: -**

A data flow (DFD) is a graphical system model that shows all of the main requirements for an information system in one datagram: inputs and outputs, processes, and data storage. A DFD describes what data flows rather than how it is processed. Everyone working on a development project can see all aspects of the system working together at once with DFD. That is one reason for its popularity. The DFD is also easy to read because it is graphical model.

* **DFD SYMBOLS:-**

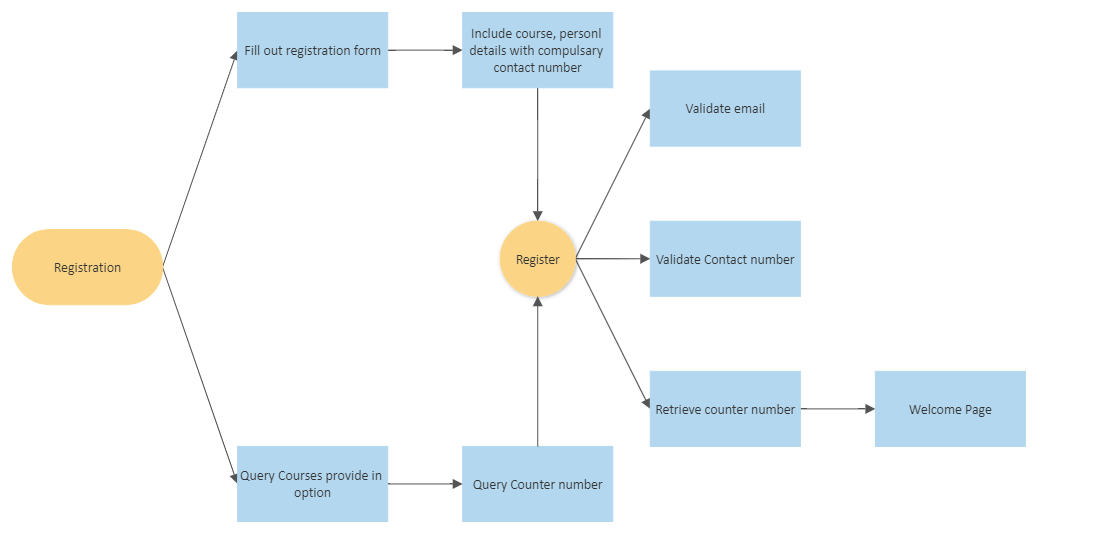
1. Process

2. Data Flow

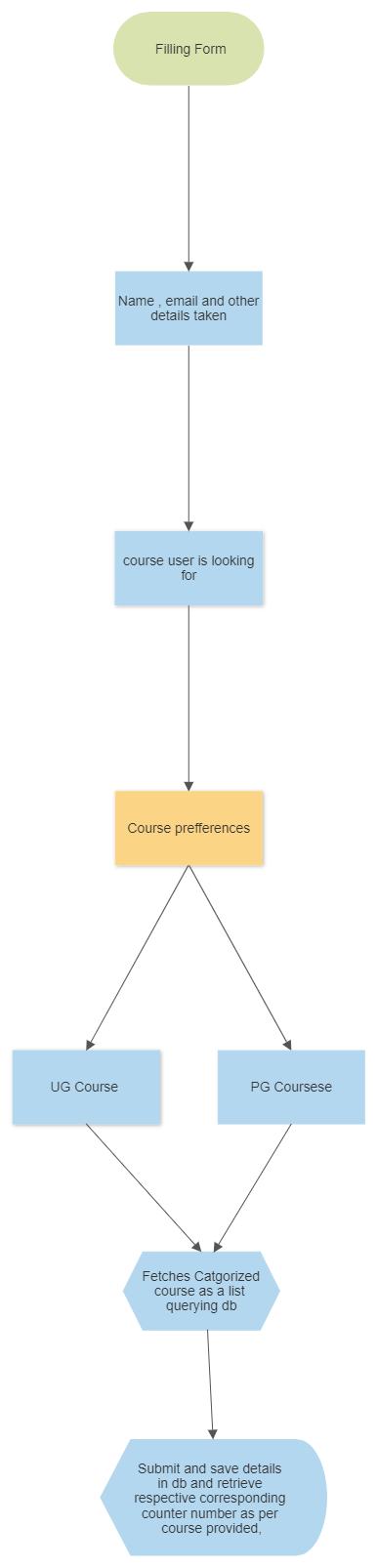
3.External Entity

**Data Flow Diagram**

**Registration Page flow Diagram**



**Backend Flow Diagram**

****

* **Entity Relationship Diagram: -**

The entity-relationship (ER) data model allows us to describe the data involved in a real world enterprise in terms of object and their relationships and is widely used to develop an initial database design.

The ER model is important primarily for its role in database design. It provides useful concepts that allow us to move from an informal description of what users want from their database to a more detailed and precise description that can be implemented in a DBMS.

* **Entity: -**

An entity is an object in the real world that is distinguishable from other objects. Examples include the following: The address of the manager of the institution, a Person with unique name etc. It is often useful to identify a collection of similar entities. Such a collection is called as “Entity set”. Note that entity set need not be disjoint.

* **Attributes: -**

An entity is described using a set of attributes. All entities in a given entity set have the same attributes; this essentially what we mean by similar. Our choice of attributed reflects the level of detail at which we wish to represent information in crisis.

For e.g. The admission entity set would use the name, age, and qualification of the students as the attributes. In this case we will store the name, the registry no, the course enrolled of the student and not his/her address or the gender.

* **Domain: -**

For each attribute associated with an entity set, we must identify a domain of possible values. For e.g. the domain associated with the attribute name of the student might be of the set of 20-character string.

Another example would be the ranking of the students in the institute would be on the scale of 1-6, the associated domain consists of integers 1 through 6.

* **Key:-**

Further, for each entity set we choose a key. A key is a minimal set of attributed whose values uniquely identify an entity in the set. There could be more than one candidate; if so we designate one of them as primary key. For now we will assume that each entity set contains at least one set of attributes that uniquely identify an entity in the entity set; that is the set of attributes contains a key.

**Introduction to MySQL**

MySQL is a popular open-source relational database management system (RDBMS) that is widely used for managing and storing structured data. It is known for its scalability, reliability, and ease of use. MySQL is a key component of the LAMP (Linux, Apache, MySQL, PHP/Python/Perl) stack and is widely used in web development and various other applications.

Here are some key aspects and features of MySQL:

* Relational Database: MySQL is a relational database, meaning it organizes data into tables with predefined schemas. It supports relationships between tables through the use of primary keys, foreign keys, and joins.
* Structured Query Language (SQL): MySQL utilizes the SQL language for querying and manipulating data. SQL provides a standardized syntax and set of operations for interacting with the database, including creating, reading, updating, and deleting data (CRUD operations).
* Client-Server Architecture: MySQL follows a client-server architecture, where clients (such as applications or web servers) interact with the MySQL server to perform database operations. The server manages the database, processes queries, and handles data storage and retrieval.
* Scalability and Performance: MySQL is designed to handle large datasets and high traffic loads. It offers various performance optimization techniques, including indexing, caching, and query optimization, to ensure efficient data retrieval and processing.
* Data Security: MySQL provides robust security features to protect data integrity and confidentiality. It supports user authentication, access control, and encryption to safeguard sensitive information.
* Storage Engines: MySQL offers multiple storage engines that govern how data is stored and accessed. The default storage engine is InnoDB, which provides transactional capabilities and ensures data integrity. Other engines, such as MyISAM and Memory, offer different trade-offs in terms of performance and features.
* Community and Ecosystem: MySQL has a large and active community of users, developers, and contributors. It is supported by extensive documentation, forums, and online resources, making it easy to find help and share knowledge.
* MySQL can be installed on various operating systems, including Linux, Windows, and macOS. It integrates well with popular programming languages like PHP, Python, Java, and more, making it a versatile choice for web development and application backends.
* Overall, MySQL is a reliable and widely adopted database management system, suitable for a wide range of applications, from small-scale projects to enterprise-level solutions.
* **Establishment of MySQL database:-**

To establish a MySQL database, you need to follow these general steps:

1. **Install MySQL**: Begin by installing the MySQL server software on your computer. MySQL provides installation packages for various operating systems. Choose the appropriate package for your system and follow the installation instructions provided by MySQL.
2. **Configure MySQL**: After the installation, you may need to perform some initial configuration of the MySQL server. This includes setting up the root password, defining server options, and securing the server. Refer to the MySQL documentation for detailed instructions on configuration.
3. **Connect to MySQL**: Once the MySQL server is installed and running, you can connect to it using a MySQL client. MySQL provides a command-line client called "mysql" or you can use graphical tools such as phpMyAdmin, MySQL Workbench, or Navicat. These clients allow you to interact with the MySQL server and execute SQL commands.
4. **Create a Database**: Using the MySQL client, connect to the MySQL server and execute SQL commands to create a new database. For example, you can use the following command to create a database named "mydatabase":

CREATE DATABASE mydatabase;

1. **Create Tables**: After creating the database, you can create tables to organize and store your data. Tables define the structure of your data with columns and specify data types and constraints. Use the CREATE TABLE statement to create tables. Here's an example:

CREATE TABLE students ( id INT PRIMARY KEY AUTO\_INCREMENT, name VARCHAR(50) NOT NULL, email VARCHAR(100) UNIQUE, age INT );

1. **Insert Data**: Once you have created the tables, you can insert data into them using the INSERT INTO statement. Here's an example:

INSERT INTO students (name, email, age) VALUES ('John Doe', 'john@example.com', 25);

1. **Query and Manipulate Data**: You can perform various operations on the data stored in the database, such as querying data, updating records, deleting records, and more. Use SQL statements like SELECT, UPDATE, DELETE, and others to retrieve and modify data as needed.

These are the basic steps to establish a MySQL database. Depending on your specific requirements, you may need to perform additional tasks like defining relationships between tables, creating indexes, setting up user accounts with appropriate privileges, and ensuring data security. The MySQL documentation and resources provide detailed information on these topics to help you further explore and optimize your database setup.

* **Creating a table:-**

To create a table in MySQL, you can use the **CREATE TABLE** statement. Here's an example syntax for creating a table:

CREATE TABLE datatype constraints, column2 datatype constraints, ... columnN datatype constraints );

Let's break down the components of the **CREATE TABLE** statement:

* **table\_name**: This is the name of the table you want to create. Choose a meaningful name that represents the type of data the table will store.
* **column1**, **column2**, ..., **columnN**: These are the column names of the table. Each column represents a specific attribute of the data you want to store. You can specify multiple columns separated by commas.
* **datatype**: This specifies the data type of the column, such as **INT** for integer values, **VARCHAR(n)** for variable-length character strings (with a maximum length of **n** characters), **DATE** for dates, and so on. Choose the appropriate data type based on the nature of the data to be stored.
* **constraints**: You can add optional constraints to enforce specific rules or conditions on the columns. For example, you can define a column as **PRIMARY KEY** to ensure its values are unique and used as the primary identifier for each row. Other common constraints include **NOT NULL** (to ensure the column cannot contain NULL values), **UNIQUE** (to enforce uniqueness of values within the column), and **FOREIGN KEY** (to establish relationships with other tables).

Here's an example of creating a simple "students" table with three columns: **id**, **name**, and **age**:

CREATE TABLE students (id INT PRIMARY KEY, name VARCHAR(50) NOT NULL, age INT );

In this example, the "students" table has three columns: **id**, **name**, and **age**. The **id** column is defined as the primary key, ensuring its values are unique. The **name** column is a required field (**NOT NULL** constraint), and the **age** column is optional.

You can execute the **CREATE TABLE** statement using a MySQL client or any tool that allows executing SQL commands, such as the MySQL command-line client, phpMyAdmin, MySQL Workbench, or other similar tools.

**Creating data entry forms: -**

* A form is a type of a database object that is primarily used to enter or display data in a database. Most forms are bound to one or more tables and queries in the database. A form's record source refers to the fields in the underlying tables and queries.
* A form: -
* Focuses on one record at a time
* Can display fields from more than one table
* Can also display pictures and other objects
* Can contain a button that prints, opens other objects, or otherwise automates tasks.
* Data entry forms can be created either using a form wizard or in the design view. Once a table with fields is available, it is easy to create a form using the wizard:
* Click on the forms tab
* Double click on the create form by using a wizard.

* This will lead you through a series of steps until you finish creating the form. The form created is linked to the table and information entered in the fields of the form is stored in the table.
* A form need not contain all the fields from each of the tables or queries that it is based on.
* When you open a form, Microsoft Access retrieves the data from one or more tables, and displays it on the screen with the chosen layout in the Form Wizard, or with the layout that you created on your own in design view.
* **Queries: -**
* A query is a derived item in the database meant to answer specific questions that relate to the information in the database. Queries are handy during data processing.
* To find and retrieve just the data that meets conditions that you specify, including data from multiple tables, create a query. A query can also update or delete multiple records at the same time, and perform predefined or custom calculations on your data.
* A query requests data from the database. At its simplest, a query merely fetches all data from a single table. But as you create more complex (and more typical) queries, you can assemble exactly the data you want (i.e. unique sets of data that you require at any given time). Queries can also be used to execute mathematical and logical functions to obtain certain information in the database.
* Queries are derived from and linked to tables or other queries. (Due to these linkages, they tend to largely inflate the size of the database and should thus only be used to execute the intended functions, and stored only if updated information is to be retrieved)
* There are various types of queries for different uses:
* Select queries- used for extracting specific information from a large multi-information table. They can also be helpful in merging related information from different tables.
* Make-Table queries- used for making sub tables from the main table(s) and queries.
* Update queries- important in adding information in the fields of a Table. Append queries- used to copy records from one table/ query to another.
* Delete query- to PERMANENTLY remove unwanted content from the table.
* NOTE: delete query should not be used unless one surely will not require the information to be deleted.
* To run a simple Select query: In the Database window, click Queries under Objects.

* Click the query you want to open. Click Open on the Database window toolbar.

**Caution: -**

* It's a good idea to make a copy of the data you are changing or moving in an action query, in case you need to restore the data to its original state after running the action query
* In Design view,
* Choose the tables or existing queries that contain the fields to use
* Select and drag those fields to a grid (The fields can come from just one table, or from multiple tables, the fields specified for a query control the data that the query retrieves)
* Specify criteria and other settings, such as whether to sort the results ( )
* To test a new query in Design view. Just click the Run button on the Query Design toolbar. This will switch you to the results view.
* If you don't see the results you want, click the Design button to return to Design view.
* Note: To stop a query in progress (after you start it), press “CTRL+BREAK”
* A query result can include data from multiple tables. To combine data from tables, you use the Join operation
* The Join operation matches rows of one table with rows of another table, based on values in those rows.

**Relational Databases**

* A relational database is one whose components (tables, forms, queries etc) are related (linked). The linkages between database components are created by making relationship links between them. The relationship can be between:
* One component and another (one-to-one relationship),
* One component related to several other components (one-to-many)
* Several database components (many-to-many).
* Creation of relationships between database components reduces data redundancy and enhances ease of access of the information.

**Creation of relationships between database components**

* When you create a relationship between tables, the related fields don't necessarily have to have the same names.
* However, related fields must have the same data type.
* UNLESS the Primary key field is an AutoNumber field.
* An AutoNumber field can be matched with a Number field only if the FieldSize property of both of the matching fields is the same.
* For example, you can match an AutoNumber field and a Number field if the FieldSize property of both fields is Long Integer.
* Even when both matching fields are Number fields, they must have exactly the same FieldSize property setting.

**Defining a one-to-many or a one-to-one relationship**

* Close any tables you have open. You can't create or modify relationships between open tables.
* Press F11 to switch to the Database Window.
* Click Relationships on the toolbar.
* If you haven't yet defined any relationships in your database, the Show Table dialog box is automatically displayed. If you need to add the tables you want to relate and the Show Table dialog box isn't displayed, click Show Table on the toolbar.
* Double-click the names of the tables you want to relate, and then close the Show Table dialog box. To create a relationship between a table and itself, add that table twice.
* Drag the field that you want to relate from one table to the related field in the other table.
* The Edit Relationships dialog box is displayed. Check the field names displayed in the two columns to ensure they are correct.
* You can change them if necessary. Set the relationship options if necessary.
* Click the Create button to create the relationship.
* Repeat steps 5 through 8 for each pair of tables you want to relate.

* When you close the Relationships window, Microsoft Access asks if you want to save the layout. Whether you save the layout or not, the relationships you create are saved in the database.
* Note: You can create relationships using QUERIES as well as TABLES. However, referential integrity isn't enforced with queries.

**Reports**

* Reports provide a means of organizing and summarizing data. Reports are often used to present an overview highlighting main points and trends. A report can be a simple list, a status report or a monthly production report. With reports, one can prepare:
* Mailing labels for various producers
* Produce a directory
* Prepare invoices
* Present data summaries.
* A report is made from the data available. There are several ways of preparing a report:

• AutoReport: the quickest way to create a report, but gives least control over the report's structure and appearance. This automatically contains all the fields in its data source, whether table or query.

• Report Wizard: asks you questions and creates a report based on your answers. The Report Wizard asks which tables or queries the report will be based on, and which fields to use from those data sources. It also asks whether the data is to be grouped, and how it should be sorted and summarized.

• Creating a Report in Design view: Gives one control, right from the start. Design view provides you with a toolbox from which you drag selected controls and arrange them on a grid. To see the report as it will appear when printed, it should be viewed in Print Preview.

* Different properties may be set for a report, to change how it is viewed or printed.

**Database Design**

**Database Name: -** registration

**Table Name:-** courses

**Description:-**Use for contain course detail

|  |  |
| --- | --- |
| **Field Name** | **Data Type** |
| Id (primary Key) NOT NULL | Bigint |
| Prog NOT NULL | Varchar |
| Course NOT NULL | Varchar |
| tableNo NOT NULL | Varchar |
| course\_code NOT NULL | Varchar |

**Table Name:-** users

**Description:-**used to store users detail from the registration form

|  |  |
| --- | --- |
| **Field Name** | **Data Type** |
| Id (primary key) NOT NULL | Bigint |
| Fname NOT NULL | Varchar |
| Lname NOT NULL | Varchar |
| email NOT NULL | Varchar |
| mobno NOT NULL | Varchar |
| programme NOT NULL | Varchar |
| courses NOT NULL | Varchar |

**Programs:-**

* **Header Coding:**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Registration Page</title>

<!-- Mobile Specific Metas -->

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

<!-- Font-->

<link rel="stylesheet" type="text/css" href="css/opensans-font.css">

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

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

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

<link rel="stylesheet" type="text/css" href="fonts/line-awesome/css/line-awesome.min.css">

<!-- Jquery -->

<link rel="stylesheet" href="css/demo.css">

<!-- Main Style Css -->

<link rel="stylesheet" href="css/style.css"/>

</head>

<body class="form-v4">

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<div class="container-fluid">

<div class="nav-item">

<img class="bfit-logo" src="css/img/bfit-logo.jpg" title="bfit logo" alt="best college in dehradun- BFIT logo" loading="lazy"> </div>

</div>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a class="nav-link active" aria-current="page" href="home.php">Home</a>

</li>

<li class="nav-item">

<a class="nav-link" href="#">Link</a>

</li>

</div>

</div>

</nav>

* **Footer Coding:**

<?php

if (isset($pageJS)){

foreach ($pageJS as $key => $value){

echo '<script type="text/javascript" src="js/'.$value.'?'.filemtime('js/'.$value).'"></script>';

}

}

?>

<script src="https://code.jquery.com/jquery-1.11.1.min.js"></script>

<script src="https://cdn.jsdelivr.net/jquery.validation/1.16.0/jquery.validate.min.js"></script>

<script src="https://cdn.jsdelivr.net/jquery.validation/1.16.0/additional-methods.min.js"></script>

<!-- <script src="js/bootstrap/bootstrap.bundle.min.js"></script> -->

<script>

function func(){

// var prog = this.value;

var prog = document.getElementById("prog").value;

opt = $.ajax({

type: "POST",

url: "api/getdropdown.php",

data: {'prog':prog},

async: false,

dataType: "json",

});

console.log(opt) ;

var element = document.getElementById("moreopt");

element.classList.remove("dn");

document.getElementById('moreopt').innerHTML = opt.responseText;

}

// function sub(){

// var formData = {

// name: $("#first\_name").val(),

// last: $("#last\_name").val(),

// email: $("#your\_email").val(),

// prog: $("#prog").val(),

// phone: $("#phone").val(),

// };

// $.ajax({

// type: "POST",

// url: "api/register.php",

// data: formData,

// dataType: "json",

// encode: true,

// }).done(function (data) {

// console.log(data);

// $( ".page-content" ).load( "welcome.php" );

// });

// }

</script>

<script>

// $(document).ready(function(){

// $("form").submit(function(){

// debugger;

// e.preventDefault();

// var course = $("#courses").val();

// var formData = {

// name: $("#first\_name").val(),

// last: $("#last\_name").val(),

// email: $("#your\_email").val(),

// prog: $("#prog").val(),

// courses: $("#courses").val(),

// phone: $("#phone").val()

// };

// out = $.ajax({

// type: "POST",

// url: "api/register.php",

// data: formData,

// dataType: "json",

// async: false,

// });

// console.log(out) ;

// console.log(out.responseText) ;

// document.getElementById("pagecontent").classList.add("dn");

// document.getElementById('outputtable').innerHTML = out.responseText;

// document.getElementById("outputtable").classList.remove("dn");

// });

// });

function myFunction(){

// var prog = this.value;

var course = $("#courses").val();

var formData = {

name: $("#first\_name").val(),

last: $("#last\_name").val(),

email: $("#your\_email").val(),

prog: $("#prog").val(),

courses: $("#courses").val(),

phone: $("#phone").val()

};

out = $.ajax({

type: "POST",

url: "api/register.php",

data: formData,

dataType: "json",

async: false,

});

// console.log(out) ;

// console.log(out.responseText) ;

alert('Go to'+out.responseText);

window.location.reload();

}

jQuery.validator.setDefaults({

debug: true,

success: function(label){

label.attr('id', 'valid');

},

});

$( "#myform" ).validate({

messages: {

first\_name: {

required: "Please enter a firstname"

},

last\_name: {

required: "Please enter a lastname"

},

your\_email: {

required: "Please provide an email"

},

prog: {

required: "Please enter a password"

},

phone: {

required: "Please enter 10 digit phone number"

}

}

}

);

</script>

</body>

</html>

* **Header Home Coding:**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Registration Page</title>

<!-- Mobile Specific Metas -->

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

<!-- Font-->

<link rel="stylesheet" type="text/css" href="css/opensans-font.css">

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

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

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

<link rel="stylesheet" type="text/css" href="fonts/line-awesome/css/line-awesome.min.css">

<!-- Jquery -->

<link rel="stylesheet" href="css/demo.css">

<!-- Main Style Css -->

<link rel="stylesheet" href="css/style.css"/>

<link rel="stylesheet" href="home.css">

</head>

<body class="form-v4">

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<div class="container-fluid">

<div class="nav-item">

<img class="bfit-logo" src="css/img/bfit-logo.jpg" title="bfit logo" alt="best college in dehradun- BFIT logo" loading="lazy"> </div>

</div>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a class="nav-link active" aria-current="page" href="index.php">Register</a>

</li>

<li class="nav-item">

<a class="nav-link" href="#">Link</a>

</li>

</div>

</div>

</nav>

* **Registration Page Coding:**

<?php

$pageJs = ["main.js"];

include('header.php');

$page = 'loginview';

$pageTitle= 'BFIT Registration';

?>

<div class="page-content" id ="pagecontent" >

<div class="form-v4-content">

<form class="form-detail" method="post" action='welcome.php' id="myform" >

<h2>REGISTRATION FORM</h2>

<div class="form-group">

<div class="form-row form-row-1">

<label class="form-label" for="first\_name">First Name</label>

<input type="text" name="first\_name" id="first\_name" class="input-text">

</div>

<div class="form-row form-row-1">

<label class="form-label" for="last\_name">Last Name</label>

<input type="text" name="last\_name" id="last\_name" class="input-text">

</div>

</div>

<div class="form-group">

<div class="form-row form-row-1">

<label class="form-label" for="phone">Mobile No:</label>

<input class="input-text" type="tel" id="phone" name="phone" placeholder="9876543210" pattern="[0-9]{10}" required><br><br>

</div>

<div class="form-row form-row-1">

<label class="form-label " for="prog">Programmes:</label>

<select class="form-select input-text" id="prog" required onchange="func()" >

<option value="" disabled selected >Programmes Available</option>

<option value="bachelor">BFIT Bachelor Programme</option>

<option value="master">BFIT Master Programme</option>

<option value="diploma">Diploma in Engineering</option>

<option value="certdiploma">Certificate Diplomas</option>

</select>

</div>

<div class="form-row form-row-1 dn" id= 'moreopt'>

</div>

</div>

<div class="form-row form-row form-row-1">

<label class="form-label" required for="your\_email">Your Email</label>

<input type="text" name="your\_email" id="your\_email" class="input-text" required pattern="[^@]+@[^@]+.[a-zA-Z]{2,6}">

</div>

<div class="form-row-last">

<input type="button" onclick="myFunction()" class ="register" value="Register">

<!-- <input type="submit" name="register" class="register" value="Register"> -->

</div>

</form>

</div>

</div>

<div class="outputtable dn container" id="outputtable" style="padding-top: 100px; padding-bottom: 100px">

</div>

<?php

include('footer.php');

?>

* **Welcome Page Coding:**

<?php

$pageJs = ["main.js"];

include('header\_home.php');

?>

<div class="main-heading">

<h1>Welcome to BFIT</h1>

</div>

<div class="btn">

<button id="register"><a href="index.php">Register</a></button>

</div>

* **Welcome Page CSS Coding:**

.main-heading{

font-size: 90px;

text-align: center;

margin: 10% auto;

}

.btn {

width: 300px;

height: 50px;

margin: 0px 38%;

}

#register{

color: white;

background-color: steelblue;

width: 300px;

height: 50px;

outline: none;

border: none;

}

#register a{

color: white;

text-decoration: none;

font-size: 30px;

}

* **CSS Coding:**

#field { margin-left: .5em; float: left; }

#field, label { float: left; font-family: Arial, Helvetica, sans-serif; font-size: small; }

br { clear: both; }

input { border: 1px solid black; margin-bottom: .5em; }

input.error { border: 1px solid red; }

label.error {

background: url('images/unchecked.gif') no-repeat;

padding-left: 16px;

margin-left: .3em;

}

label.valid {

background: url('images/checked.gif') no-repeat;

display: block;

width: 16px;

height: 16px;

}

@font-face {

font-family: 'Open Sans';

src: url('../fonts/Open\_Sans/OpenSans-Bold.ttf') format('truetype');

font-weight: bold;

font-style: normal;

}

@font-face {

font-family: 'Open Sans';

src: url('../fonts/Open\_Sans/OpenSans-Italic.ttf') format('truetype');

font-weight: normal;

font-style: italic;

}

@font-face {

font-family: 'Open Sans';

src: url('../fonts/Open\_Sans/OpenSans-Regular.ttf') format('truetype');

font-weight: normal;

font-style: normal;

}

@font-face {

font-family: 'Open Sans';

src: url('../fonts/Open\_Sans/OpenSans-LightItalic.ttf') format('truetype');

font-weight: 300;

font-style: italic;

}

@font-face {

font-family: 'Open Sans';

src: url('../fonts/Open\_Sans/OpenSans-Light.ttf') format('truetype');

font-weight: 300;

font-style: normal;

}

@font-face {

font-family: 'Open Sans';

src: url('../fonts/Open\_Sans/OpenSans-BoldItalic.ttf') format('truetype');

font-weight: bold;

font-style: italic;

}

@font-face {

font-family: 'Open Sans';

src: url('../fonts/Open\_Sans/OpenSans-SemiBold.ttf') format('truetype');

font-weight: 600;

font-style: normal;

}

@font-face {

font-family: 'Open Sans';

src: url('../fonts/Open\_Sans/OpenSans-ExtraBoldItalic.ttf') format('truetype');

font-weight: 800;

font-style: italic;

}

@font-face {

font-family: 'Open Sans';

src: url('../fonts/Open\_Sans/OpenSans-ExtraBold.ttf') format('truetype');

font-weight: 800;

font-style: normal;

}

@font-face {

font-family: 'Open Sans';

src: url('../fonts/Open\_Sans/OpenSans-SemiBoldItalic.ttf') format('truetype');

font-weight: 600;

font-style: italic;

}

body {

margin: 0;

}

.page-content {

width: 100%;

height: 75%;

margin: 0 auto;

background: #75e2e9;

display: flex;

display: -webkit-flex;

justify-content: center;

-o-justify-content: center;

-ms-justify-content: center;

-moz-justify-content: center;

-webkit-justify-content: center;

align-items: center;

-o-align-items: center;

-ms-align-items: center;

-moz-align-items: center;

-webkit-align-items: center;

}

.form-v4-content {

background: #fff;

width: 80%;

border-radius: 10px;

-o-border-radius: 10px;

-ms-border-radius: 10px;

-moz-border-radius: 10px;

-webkit-border-radius: 10px;

box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

-o-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

-ms-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

-moz-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

-webkit-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

margin: 10px 0;

position: relative;

display: flex;

display: -webkit-flex;

font-family: 'Open Sans', sans-serif;

}

.form-v4-content h2 {

font-weight: 700;

font-size: 30px;

padding: 6px 0 0;

margin-bottom: 2px;

}

.form-v4-content .form-left {

background: #3786bd;

border-top-left-radius: 10px;

border-bottom-left-radius: 10px;

padding: 2px 4px;

position: relative;

width: 100%;

color: #fff;

}

.form-v4-content .form-left p {

font-size: 15px;

font-weight: 300;

line-height: 1.5;

}

.form-v4-content .form-left span {

font-weight: 700;

}

.form-v4-content .form-left .text-2 {

margin: 20px 0 25px;

}

.form-v4-content .form-left .account {

background: #fff;

border-top-left-radius: 5px;

border-bottom-right-radius: 5px;

width: 180px;

border: none;

margin: 15px 0 50px 0px;

cursor: pointer;

color: #333;

font-weight: 700;

font-size: 15px;

font-family: 'Open Sans', sans-serif;

appearance: unset;

-moz-appearance: unset;

-webkit-appearance: unset;

-o-appearance: unset;

-ms-appearance: unset;

outline: none;

-moz-outline: none;

-webkit-outline: none;

-o-outline: none;

-ms-outline: none;

}

.form-v4-content .form-left .account:hover {

background: #e5e5e5;

}

.form-v4-content .form-left .form-left-last input {

padding: 15px;

}

.form-v4-content .form-detail {

padding: 20px 40px;

position: relative;

width: 100%;

}

.form-v4-content .form-detail h2 {

color: #3786bd;

}

.form-v4-content .form-detail .form-group {

display: flex;

display: -webkit-flex;

margin: 0px;

}

.form-v4-content .form-detail .form-row {

width: 100%;

position: relative;

}

.form-v4-content .form-detail .form-group .form-row.form-row-1 {

width: 50%;

padding: 0 8px;

}

.form-v4-content .form-detail label {

font-weight: 600;

font-size: 15px;

color: #666;

display: block;

margin-bottom: 2px;

}

.form-v4-content .form-detail .form-row label#valid {

position: absolute;

right: 20px;

top: 50%;

transform: translateY(-50%);

-o-transform: translateY(-50%);

-moz-transform: translateY(-50%);

-ms-transform: translateY(-50%);

-webkit-transform: translateY(-50%);

width: 14px;

height: 14px;

border-radius: 50%;

-o-border-radius: 50%;

-ms-border-radius: 50%;

-moz-border-radius: 50%;

-webkit-border-radius: 50%;

background: #53c83c;

}

.form-v4-content .form-detail .form-row label#valid::after {

content: "";

position: absolute;

left: 5px;

top: 1px;

width: 3px;

height: 8px;

border: 1px solid #fff;

border-width: 0 2px 2px 0;

-webkit-transform: rotate(45deg);

-ms-transform: rotate(45deg);

-o-transform: rotate(45deg);

-moz-transform: rotate(45deg);

transform: rotate(45deg);

}

.form-v4-content .form-detail .form-row label.error {

padding-left: 0;

margin-left: 0;

display: block;

position: absolute;

bottom: -5px;

width: 100%;

background: none;

color: red;

}

.form-v4-content .form-detail .form-row label.error::after {

content: "\f343";

font-family: "LineAwesome";

position: absolute;

transform: translate(-50%, -50%);

-webkit-transform: translate(-50%, -50%);

-ms-transform: translate(-50%, -50%);

-o-transform: translate(-50%, -50%);

-moz-transform: translate(-50%, -50%);

right: 10px;

top: -31px;

color: red;

font-size: 18px;

font-weight: 900;

}

.form-v4-content .form-detail .input-text {

margin-bottom: 27px;

}

.form-v4-content .form-detail input {

width: 100%;

padding: 11.5px 15px;

border: 1px solid #e5e5e5;

border-top-left-radius: 5px;

border-bottom-right-radius: 5px;

appearance: unset;

-moz-appearance: unset;

-webkit-appearance: unset;

-o-appearance: unset;

-ms-appearance: unset;

outline: none;

-moz-outline: none;

-webkit-outline: none;

-o-outline: none;

-ms-outline: none;

font-family: 'Open Sans', sans-serif;

font-size: 15px;

color: #333;

box-sizing: border-box;

-moz-box-sizing: border-box;

-webkit-box-sizing: border-box;

-o-box-sizing: border-box;

-ms-box-sizing: border-box;

}

.form-v4-content .form-detail .form-row input:focus {

border: 1px solid #53c83c;

}

.form-v4-content .form-detail .form-checkbox {

margin-top: 1px;

position: relative;

}

.form-v4-content .form-detail .form-checkbox input {

position: absolute;

opacity: 0;

}

.form-v4-content .form-detail .form-checkbox .checkmark {

position: absolute;

top: 13px;

left: 0;

height: 15px;

width: 15px;

border: 1px solid #ccc;

cursor: pointer;

}

.form-v4-content .form-detail .form-checkbox .checkmark::after {

content: "";

position: absolute;

left: 5px;

top: 1px;

width: 3px;

height: 8px;

border: 1px solid #3786bd;

border-width: 0 2px 2px 0;

-webkit-transform: rotate(45deg);

-ms-transform: rotate(45deg);

-o-transform: rotate(45deg);

-moz-transform: rotate(45deg);

transform: rotate(45deg);

display: none;

}

.form-v4-content .form-detail .form-checkbox input:checked ~ .checkmark::after {

display: block;

}

.form-v4-content .form-detail .form-checkbox p {

margin-left: 34px;

color: #333;

font-size: 14px;

font-weight: 600;

}

.form-v4-content .form-detail .form-checkbox .text {

font-weight: 700;

color: #3786bd;

text-decoration: underline;

}

.form-v4-content .form-detail .register {

background: #3786bd;

border-top-left-radius: 5px;

border-bottom-right-radius: 5px;

width: 130px;

border: none;

margin: 6px 0 50px 0px;

cursor: pointer;

color: #fff;

font-weight: 700;

font-size: 15px;

}

.form-v4-content .form-detail .register:hover {

background: #2f73a3;

}

.form-v4-content .form-detail .form-row-last input {

padding: 12.5px;

}

/\* Responsive \*/

@media screen and (max-width: 991px) {

.form-v4-content {

margin: 180px 20px;

flex-direction: column;

-o-flex-direction: column;

-ms-flex-direction: column;

-moz-flex-direction: column;

-webkit-flex-direction: column;

}

.form-v4-content .form-left {

width: auto;

border-top-right-radius: 10px;

border-bottom-left-radius: 0;

}

.form-v4-content .form-detail {

padding: 30px 20px 30px 20px;

width: auto;

}

}

@media screen and (max-width: 575px) {

.form-v4-content .form-detail .form-group {

flex-direction: column;

-o-flex-direction: column;

-ms-flex-direction: column;

-moz-flex-direction: column;

-webkit-flex-direction: column;

margin: 0;

}

.form-v4-content .form-detail .form-group .form-row.form-row-1 {

width: 100%;

padding: 0;

}

}

.bfit-logo{

width: 100px !important;

}

.container-fluid{

margin: 0px;

padding: 0px;

}

.navbar{

--bs-navbar-padding-y: 0 !important;

background-color: #3786bd !important;

}

.dn{

display:none;

}

.table-card {

background: white;

border-radius: 2px;

box-shadow: 0 2px 5px 0 rgba(0, 0, 0, 0.16), 0 2px 10px 0 rgba(0, 0, 0, 0.12);

}

.table-card .header {

line-height: 64px;

padding-left: 24px;

padding-right: 14px;

}

.table-card .header .title {

font-size: 20px;

}

.table-card .header .actions {

float: right;

}

.table-card .header .actions a {

color: rgba(0, 0, 0, 0.54);

font-size: 24px;

padding-left: 24px;

}

.table-card .table {

padding: 0;

margin: 0;

}

.table-card .table table,

.table-card .table table tr,

.table-card .table table > thead > tr > th,

.table-card .table table > tbody > tr > td {

padding: 0;

margin: 0;

}

.table-card .table table > thead > tr > th {

line-height: 56px;

color: rgba(0, 0, 0, 0.54)

}

.table-card .table table > tbody > tr > td {

line-height: 48px;

word-wrap: break-word;

}

.table-card .table table > thead > tr > th:first-child,

.table-card .table table > tbody > tr > td:first-child {

padding-left: 24px;

}

.table-card .table table > thead > tr > th:last-child,

.table-card .table table > tbody > tr > td:last-child {

padding-right: 14px;

}

.table-card .footer {

line-height: 56px;

font-size: 12px;

color: rgba(0, 0, 0, 0.54);

text-align: right;

padding-right: 14px;

}

* **JavaScript Coding:**

// just for the demos, avoids form submit

jQuery.validator.setDefaults({

debug: true,

success: function(label){

label.attr('id', 'valid');

},

});

$( "#myform" ).validate({

messages: {

first\_name: {

required: "Please enter a firstname"

},

last\_name: {

required: "Please enter a lastname"

},

your\_email: {

required: "Please provide an email"

},

prog: {

required: "Please enter a password"

},

phone: {

required: "Please enter 10 digit phone number"

}

}

}

);

* **Dropdown API Coding:**

<?php

$errors = [];

$data = [];

if (isset($\_POST['prog'])){

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

}

else{

$prog = 'bachelor';

}

$server\_name = 'localhost';

$user\_name = 'root';

$password= 'root';

$dbname = 'registration';

$con = mysqli\_connect($server\_name, $user\_name, $password, $dbname);

if (!$con) {

die("Failed ". mysqli\_connect\_error());

}

$query = "select distinct course,course\_code from courses where prog='$prog';";

// Perform query

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

if (!$result) {

trigger\_error(mysqli\_error($con), E\_USER\_ERROR);

}

$str = '<label class="form-label " for="courses">Courses:</label><select class="form-select input-text " id="courses" required ><option value="" disabled selected >Select Courses</option>';

while ($row = mysqli\_fetch\_array($result)) {

$str .="<option value=".$row[1].">".$row[0]."</option>\n";

}

$str .= '</select>';

echo $str;

mysqli\_close($con);

return $str;

?>

* **Registration API Coding:**

<?php

$errors = [];

$data = [];

// print\_r($\_POST);

if (isset($\_POST)){

$fname = $\_POST['name'];

$lname = $\_POST['last'];

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

$mobno = $\_POST['phone'];

$programme = $\_POST['prog'];

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

$server\_name = 'localhost';

$user\_name = 'root';

$password= 'root';

$dbname = 'registration';

$con = mysqli\_connect($server\_name, $user\_name, $password, $dbname);

if (!$con) {

die("Failed ". mysqli\_connect\_error());

}

// echo "Connection established successfully";

#

#INSERT INTO users (fname,lname,email,mobno,programme,courses) VALUES('Tabish','jamal','jamaltab24@gmail.com','0897947539','bachelor','BBA') ON DUPLICATE KEY UPDATE mobno='0897947539';

$query = "INSERT INTO users (fname,lname,email,mobno,programme,courses) VALUES('$fname','$lname','$email','$mobno','$programme','$courses') ON DUPLICATE KEY UPDATE mobno='$mobno';";

// echo "$query";

// Perform query

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

$query2 = "select tableno from courses where course = '$courses';";

$str = '';

$result2 = mysqli\_query($con, $query2);

while ($row = mysqli\_fetch\_array($result2)) {

$str .= $row[0];

}

echo $str;

mysqli\_close($con);

return $str;

}

else{

print\_r($POST);

echo "Please Enter Proper values to start";

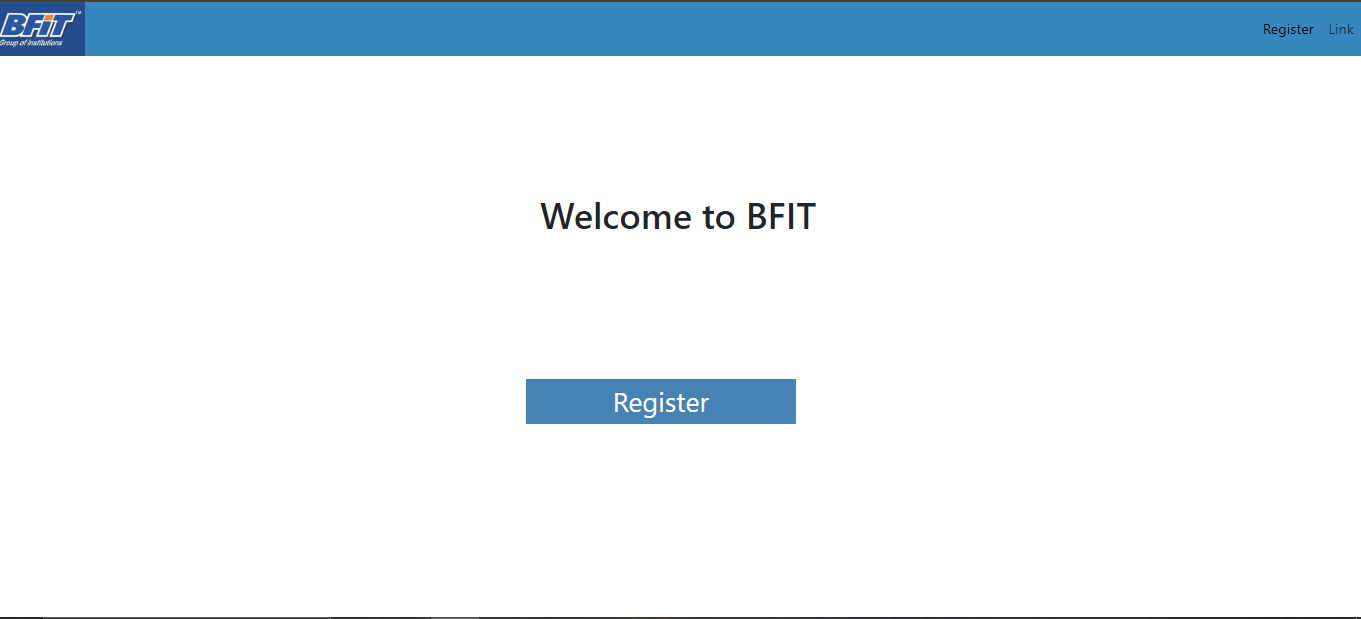
}

// echo json\_encode($data);

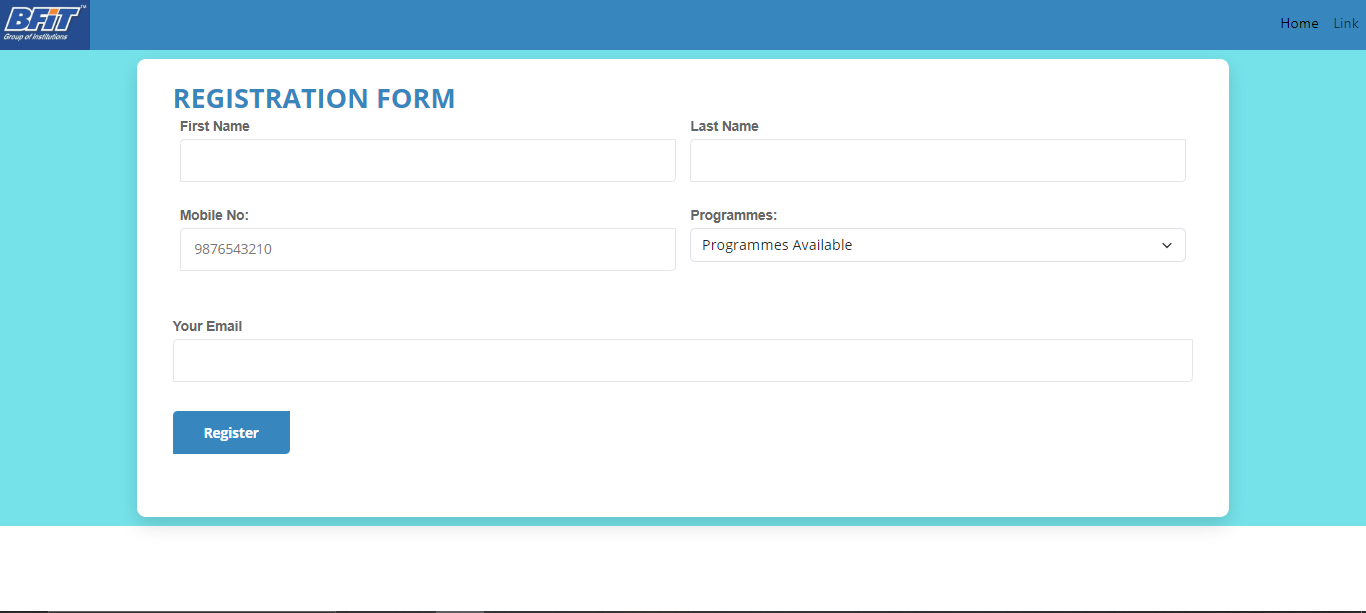
?>

**Input/output from (Screen layout)**

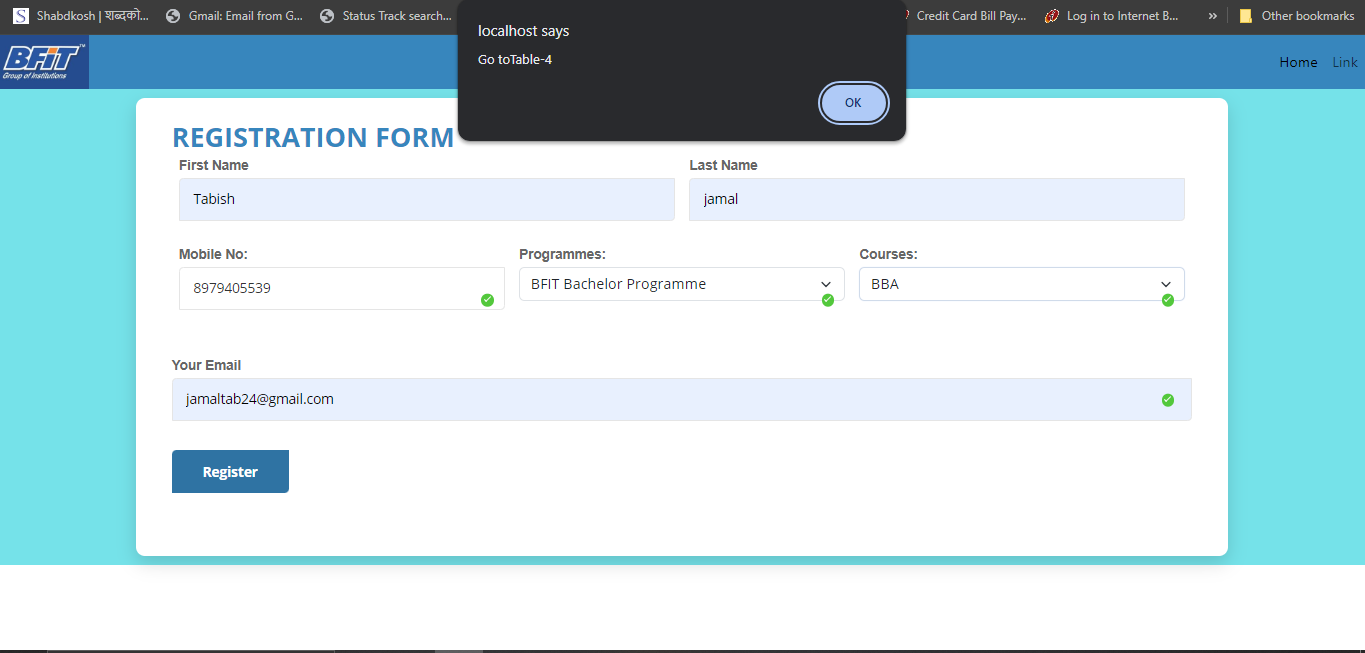
* **Welcome**

****

* **Registration Form**



* **After Submission Details**

****

**Testing**

* **Software Testing** is a method to check whether the actual software product matches expected requirements and to ensure that software product is[Defect](https://www.guru99.com/defect-management-process.html)free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.
* Some prefer saying Software testing definition as a [White Box](https://www.guru99.com/white-box-testing.html) and [Black Box Testing](https://www.guru99.com/black-box-testing.html). In simple terms, Software Testing means the Verification of Application Under Test (AUT). This Software Testing course introduces testing software to the audience and justifies the importance of software testing

**What is the need of Testing?**

Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss, and history is full of such examples.

* In April 2015, Bloomberg terminal in London crashed due to software glitch affected more than 300,000 traders on financial markets. It forced the government to postpone a 3bn pound debt sale.
* Nissan cars recalled over 1 million cars from the market due to software failure in the airbag sensory detectors. There has been reported two accident due to this software failure.
* Starbucks was forced to close about 60 percent of stores in the U.S and Canada due to software failure in its POS system. At one point, the store served coffee for free as they were unable to process the transaction.
* Some of Amazon's third-party retailers saw their product price is reduced to 1p due to a software glitch. They were left with heavy losses.
* Vulnerability in Windows 10. This bug enables users to escape from security sandboxes through a flaw in the win32k system.
* In 2015 fighter plane F-35 fell victim to a software bug, making it unable to detect targets correctly.
* China Airlines Airbus A300 crashed due to a software bug on April 26, 1994, killing 264 innocents live
* In 1985, Canada's Therac-25 radiation therapy machine malfunctioned due to software bug and delivered lethal radiation doses to patients, leaving 3 people dead and critically injuring 3 others.
* In April of 1999, a software bug caused the failure of a $1.2 billion military satellite launch, the costliest accident in history
* In May of 1996, a software bug caused the bank accounts of 823 customers of a major U.S. bank to be credited with 920 million US dollars.

## Types of Software Testing

Here are the software testing types:

Typically Testing is classified into three categories.

* Functional Testing
* Non-Functional Testing or [Performance Testing](https://www.guru99.com/performance-testing.html)
* Maintenance (Regression and Maintenance)

## Testing Strategies in Software Engineering

Here are important strategies in software engineering:

* **Unit Testing:**This software testing basic approach is followed by the programmer to test the unit of the program. It helps developers to know whether the individual unit of the code is working properly or not.
* **Integration testing:**It focuses on the construction and design of the software. You need to see that the integrated units are working without errors or not.
* **System testing:**In this method, your software is compiled as a whole and then tested as a whole. This testing strategy checks the functionality, security, portability, amongst others.

## Program Testing

* **Program Testing** in software testing is a method of executing an actual software program with the aim of testing program behavior and finding errors. The software program is executed with test case data to analyse the program behavior or response to the test data. A good program testing is one which has high chances of finding bugs.

**Limitations and drawback**

* Through it was planned, that the system developed will absolutely error free but there certain limitations in the system.

Following may be the drawback of this system.

* Since every time interaction is the database so the load is more. So the system tends to be bit show.
* Through the system is developed as a multi-user system but it is real time system.
* Limited Customization: While the system provides information fields that can be customized, there may be limitations in terms of the extent of customization. It may not fully accommodate unique requirements or additional data fields specific to certain colleges or universities without additional development effort.
* Lack of Real-Time Updates: The system may not provide real-time updates to students regarding the status of their applications or changes in counseling schedules. Students may need to rely on other communication channels or manual notifications for timely information.

**Conclusion**

The College Admission Cell Management System developed using HTML, CSS, JavaScript, PHP, and MySQL presents a comprehensive solution for streamlining the college admission process. By implementing a user-friendly registration page and automating the table number generation, the system simplifies the application and counseling phases, improving efficiency and accuracy.

With the registration page's various information fields, students can easily provide their personal details, contact information, academic qualifications, and any other necessary information required for admission. The system ensures data accuracy through proper validation and error handling, reducing the chances of erroneous data entry.

The automated table number generation feature eliminates manual table assignment, providing students with a unique identifier associated with their registration details. This table number facilitates the counseling process by allowing administrators to allocate students to their designated counseling sessions systematically.

The system leverages PHP and MySQL to securely store and manage student data in a centralized database. This ensures efficient data retrieval, eliminates paperwork, and enhances data management. The system's security measures protect student information, employing authentication and authorization to restrict access and encryption techniques to safeguard sensitive data.

Administrative functionalities empower administrators to manage student registrations and counseling sessions effectively. They can view, edit, and update student information, as well as generate statistical reports and admission trends to aid decision-making.

By meeting the system requirements and considering factors like scalability and compatibility, the project ensures a seamless user experience across different devices and web browsers. Ongoing maintenance, documentation, and technical support contribute to the system's longevity and usability.

In conclusion, the College Admission Cell Management System offers a user-friendly, automated, and secure platform for managing the college admission process. By leveraging HTML, CSS, JavaScript, PHP, and MySQL, the system enhances efficiency, data accuracy, and decision-making, making it an invaluable tool for colleges and universities.