**Abstract**

Online Accreditation System is a web based accreditation of application system that allows public and private higher education providers to submit applications for the accreditation of programmers online. Institutions can submit programmed-accreditation applications throughout the year, taking into account the dates of the Accreditation committee meetings. We Grants accreditation of the schools based on the application form and after following the various stages of scrutiny. The process of accreditation includes Submission, Digitization, SMS alerts and mail. It provides certificate and seal for your institute can be used to promote on several mediums include your website, official documents and marketing. It is very simple and hassle free.

**Introduction**

Accreditation is an official recognition that a school is credible–that is, that the school maintains educational standards well enough that other accredited schools will recognize the professed level of education claimed by its students and faculty. Accreditation is a process that many schools, colleges and universities wanting government support, as well as recognition by industry and the public, regularly undergo. Accredited schools are then held to strict standards which must be maintained with each accreditation renewal. The process of accreditation involves a brief school inspection to assess the level of conformity to set accreditation standards.

1. Interested schools fill out a formal request for accreditation from one of the accreditation bodies and pay an application fee. The application must be completed about 3 months before a school visit is scheduled. Visits are scheduled only during the second semester of operation for new schools.
2. After approval, the accreditation organization schedules a preliminary one-day or two-day visit by a small committee. The committee prepares a report back to the accreditation body, including recommendations about how the school can improve.

**1. System Specification**

* 1. **Hardware Configuration**

Hard Disk : 115GB

Keyboard : 104 Keys.

Monitor : Lenovo-PC

Processor : Intel(R) Pentium(R) processor.

RAM Capacity : 2GB.

Speed : 4GHZ.

System Bus : 32bit.

* + 1. **Software Specification**

Operating System : windows 8.1

Software Used : Geany, Notepad++.

Server Used : Pg Admin4.

Front End : HTML, CSS, Python.

Back End : PostgreSql.

**1.2 Software Features:**

**HTML, CSS and JS**

**Hypertext Markup Language (HTML)** is the standard [markup language](https://en.wikipedia.org/wiki/Markup_language) for creating [web pages](https://en.wikipedia.org/wiki/Web_page) and [web applications](https://en.wikipedia.org/wiki/Web_application). With [**Cascading Style Sheets**](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) (CSS) and [**JavaScript**](https://en.wikipedia.org/wiki/JavaScript) (JS) it forms a triad of cornerstone technologies for the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web).[Web browsers](https://en.wikipedia.org/wiki/Web_browser) receive HTML

Documents from a [web server](https://en.wikipedia.org/wiki/Web_server) or from local storage and render them into multimedia web pages. HTML describes the structure of a web page [semantically](https://en.wikipedia.org/wiki/Semantic_Web) and originally included cues for the appearance of the document.

**Cascading Style Sheets** (**CSS**) is a [style sheet language](https://en.wikipedia.org/wiki/Style_sheet_language) used for describing the [presentation](https://en.wikipedia.org/wiki/Presentation_semantics) of a document written in a [markup language](https://en.wikipedia.org/wiki/Markup_language). Although most often used to set the visual style of [web pages](https://en.wikipedia.org/wiki/Web_page) and user interfaces written in [HTML](https://en.wikipedia.org/wiki/HTML) and [XHTML](https://en.wikipedia.org/wiki/XHTML), the language can be applied to any [XML](https://en.wikipedia.org/wiki/XML) document, including [plain XML](https://en.wikipedia.org/wiki/Plain_Old_XML), [SVG](https://en.wikipedia.org/wiki/Scalable_Vector_Graphics) and [XUL](https://en.wikipedia.org/wiki/XUL), and is applicable to rendering in [speech](https://en.wikipedia.org/wiki/Speech_synthesis), or on other media. Along with HTML and [JavaScript](https://en.wikipedia.org/wiki/JavaScript), CSS is a cornerstone technology used by most websites to create visually engaging web pages, user interfaces for [web applications](https://en.wikipedia.org/wiki/Web_applications), and user interfaces for many mobile applications. CSS is designed primarily to enable the separation of presentation and content, including aspects such as the [layout](https://en.wikipedia.org/wiki/Page_layout), [colors](https://en.wikipedia.org/wiki/Color), and [fonts](https://en.wikipedia.org/wiki/Typeface).

**JavaScript** often abbreviated as **JS**, is a [high-level](https://en.wikipedia.org/wiki/High-level_programming_language), [dynamic](https://en.wikipedia.org/wiki/Dynamic_programming_language), [weakly typed](https://en.wikipedia.org/wiki/Weak_typing), [prototype-based](https://en.wikipedia.org/wiki/Prototype-based_programming), [multi-paradigm](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language), and [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [programming language](https://en.wikipedia.org/wiki/Programming_language). Alongside [HTML](https://en.wikipedia.org/wiki/HTML) and [CSS](https://en.wikipedia.org/wiki/CSS), JavaScript is one of the three core technologies of Web content. It is used to make web pages interactive and provide online programs, including video games. The majority of [websites](https://en.wikipedia.org/wiki/Website) employ it, and all modern [web browsers](https://en.wikipedia.org/wiki/Web_browser) support it without the need for [plug-ins](https://en.wikipedia.org/wiki/Browser_extension) by means of a built-in [JavaScript engine](https://en.wikipedia.org/wiki/JavaScript_engine). Each of the many JavaScript engines represent a different implementation of JavaScript, all based on the [ECMA Script](https://en.wikipedia.org/wiki/ECMAScript) specification, with some engines not supporting the spec fully, and with many engines supporting additional features beyond ECMA.

**Python**

Python is a [multi-paradigm programming language](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language). [Object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) and [structured programming](https://en.wikipedia.org/wiki/Structured_programming) are fully supported, and many of its features support [functional programming](https://en.wikipedia.org/wiki/Functional_programming) and [aspect-oriented programming](https://en.wikipedia.org/wiki/Aspect-oriented_programming) (including by [meta programming](https://en.wikipedia.org/wiki/Metaprogramming) and [meta objects](https://en.wikipedia.org/wiki/Metaobject) (magic methods)). Many other paradigms are supported via extensions, including [design by contract](https://en.wikipedia.org/wiki/Design_by_contract) and [logic programming](https://en.wikipedia.org/wiki/Logic_programming).

The language's core philosophy is summarized in the document *The*[*Zen of Python*](https://en.wikipedia.org/wiki/Zen_of_Python) (*PEP 20*), which includes [aphorisms](https://en.wikipedia.org/wiki/Aphorism) such as:

* Beautiful is better than ugly
* Explicit is better than implicit
* Simple is better than complex
* Complex is better than complicated
* Readability counts

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with [ABC](https://en.wikipedia.org/wiki/ABC_(programming_language)), which espoused the opposite approach.

While offering choice in coding methodology, the Python philosophy rejects exuberant syntax (such as that of [Perl](https://en.wikipedia.org/wiki/Perl)) in favor of a simpler, less-cluttered grammar.

**PostgreSQL**

**PostgreSQL**, often simply **Postgres**, is an [object-relational database management system](https://en.wikipedia.org/wiki/Object-relational_database_management_system) (ORDBMS) with an emphasis on extensibility and standards compliance. As a database server, its primary functions are to store data securely and return that data in response to requests from other software applications. It can handle workloads ranging from small single-machine applications to large [Internet-facing applications](https://en.wikipedia.org/wiki/Web_service) (or for [data warehousing](https://en.wikipedia.org/wiki/Data_warehousing)) with many [concurrent users](https://en.wikipedia.org/wiki/Concurrent_user); on [macOS Server](https://en.wikipedia.org/wiki/MacOS_Server), PostgreSQL is the default database and it is also available for [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows) and [Linux](https://en.wikipedia.org/wiki/Linux) (supplied in most distributions). PostgreSQL is [ACID](https://en.wikipedia.org/wiki/ACID)-compliant and [transactional](https://en.wikipedia.org/wiki/Transaction_processing). PostgreSQL has updatable [views](https://en.wikipedia.org/wiki/View_(SQL)) and [materialized views](https://en.wikipedia.org/wiki/Materialized_view), [triggers](https://en.wikipedia.org/wiki/Database_trigger), [foreign keys](https://en.wikipedia.org/wiki/Foreign_key); supports functions and stored procedures, and other expandability.

Procedural languages allow developers to extend the database with custom [subroutines](https://en.wikipedia.org/wiki/Subroutines) (functions), often called [stored procedures](https://en.wikipedia.org/wiki/Stored_procedure). These functions can be used to build [triggers](https://en.wikipedia.org/wiki/Database_trigger) (functions invoked upon modification of certain data) and custom [aggregate functions](https://en.wikipedia.org/wiki/Aggregate_function). Procedural languages can also be invoked without defining a function, using the "DO" command at SQL level.

Languages are divided into two groups: "Safe" languages are [sandboxed](https://en.wikipedia.org/wiki/Sandbox_(computer_security)) and can be safely used by any user. Procedures written in "unsafe" languages can only be created by [super users](https://en.wikipedia.org/wiki/Superuser), because they allow bypassing the database's security restrictions, but can also access sources external to the database. Some languages like Perl provide both safe and unsafe versions.

PostgreSQL has built-in support for three procedural languages:

* Plain SQL (safe). Simpler SQL functions can get [expanded inline](https://en.wikipedia.org/wiki/Inline_expansion) into the calling (SQL) query, which saves function call overhead and allows the query optimizer to "see inside" the function.
* [PL/pgSQL](https://en.wikipedia.org/wiki/PL/pgSQL) (safe), which resembles Oracle's [PL/SQL](https://en.wikipedia.org/wiki/PL/SQL) procedural language and [SQL/PSM](https://en.wikipedia.org/wiki/SQL/PSM).
* [C](https://en.wikipedia.org/wiki/C_(programming_language)) (unsafe), which allows loading custom [shared libraries](https://en.wikipedia.org/wiki/Shared_library) into the database. Functions written in C offer the best performance, but bugs in code can crash and potentially corrupt the database. Most built-in functions are written in C.

In addition, PostgreSQL allows procedural languages to be loaded into the database through extensions. Three language extensions are included with PostgreSQL to support [Perl](https://en.wikipedia.org/wiki/Perl), [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) and [Tcl](https://en.wikipedia.org/wiki/Tcl). There are external projects to add support for many other languages.

PostgreSQL is available for the following operating systems: [Linux](https://en.wikipedia.org/wiki/Linux) (all recent distributions), [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows) (Windows 2000 SP4 and later; compilable by e.g. [Visual Studio](https://en.wikipedia.org/wiki/Visual_Studio), now with up to most recent 2015 version), [FreeBSD](https://en.wikipedia.org/wiki/FreeBSD), [OpenBSD](https://en.wikipedia.org/wiki/OpenBSD), [NetBSD](https://en.wikipedia.org/wiki/NetBSD), OS X ([macOS](https://en.wikipedia.org/wiki/MacOS)), [AIX](https://en.wikipedia.org/wiki/IBM_AIX), [HP-UX](https://en.wikipedia.org/wiki/HP-UX), [Solaris](https://en.wikipedia.org/wiki/Solaris_(operating_system)), and [UnixWare](https://en.wikipedia.org/wiki/UnixWare); and not officially tested: [DragonFly BSD](https://en.wikipedia.org/wiki/DragonFly_BSD), [BSD/OS](https://en.wikipedia.org/wiki/BSD/OS), [IRIX](https://en.wikipedia.org/wiki/IRIX), [OpenIndiana](https://en.wikipedia.org/wiki/OpenIndiana), [OpenSolaris](https://en.wikipedia.org/wiki/OpenSolaris), [OpenServer](https://en.wikipedia.org/wiki/OpenServer), and [Tru64 Unix](https://en.wikipedia.org/wiki/Tru64_UNIX). Most other Unix-like systems could also work; most modern do support.

PostgreSQL works on any of the following [instruction set architectures](https://en.wikipedia.org/wiki/Instruction_set_architecture):  [x86](https://en.wikipedia.org/wiki/X86) and [x86-64](https://en.wikipedia.org/wiki/X86-64) on Windows and other operating systems; these are supported on other than Windows: IA-64 [Itanium](https://en.wikipedia.org/wiki/Itanium) (external support for HP-UX), [PowerPC](https://en.wikipedia.org/wiki/PowerPC), PowerPC 64, [S/390](https://en.wikipedia.org/wiki/S/390), [S/390x](https://en.wikipedia.org/wiki/IBM_System_z), [SPARC](https://en.wikipedia.org/wiki/SPARC), SPARC 64, [ARMv8](https://en.wikipedia.org/wiki/ARMv8)-A ([64-bit](https://en.wikipedia.org/wiki/64-bit)) and older [ARM](https://en.wikipedia.org/wiki/ARM_architecture) ([32-bit](https://en.wikipedia.org/wiki/32-bit), including older such as [ARMv6](https://en.wikipedia.org/wiki/ARMv6) in [Raspberry Pi](https://en.wikipedia.org/wiki/Raspberry_Pi)), [MIPS](https://en.wikipedia.org/wiki/MIPS_architecture), [MIPSel](https://en.wikipedia.org/wiki/MIPSel), and [PA-RISC](https://en.wikipedia.org/wiki/PA-RISC). It is also known to work on [Alpha](https://en.wikipedia.org/wiki/DEC_Alpha) (dropped in 9.5), [M68k](https://en.wikipedia.org/wiki/Motorola_68000_family), [M32R](https://en.wikipedia.org/wiki/M32R), [NS32k](https://en.wikipedia.org/wiki/NS320xx), and [VAX](https://en.wikipedia.org/wiki/VAX). In addition to these, it is possible to build PostgreSQL for an unsupported CPU by disabling [spinlocks](https://en.wikipedia.org/wiki/Spinlock).

The pgAdmin package is a free and open source [graphical user interface](https://en.wikipedia.org/wiki/Graphical_user_interface) administration tool for PostgreSQL, which is supported on many computer platforms. The first prototype, named pgManager, was written for PostgreSQL 6.3.2 from 1998, and rewritten and released as pgAdmin under the GNU General Public License (GPL) in later months. The third version, pgAdmin III, was originally released under the [Artistic License](https://en.wikipedia.org/wiki/Artistic_License) and then released under the same license as PostgreSQL. The query tool includes a scripting language called pgScript for supporting admin and development tasks. In 2016, pgAdmin 4 was released.

**2. System Study**

**2.1 Existing System**

It is so tedious and time consuming. The problem is written in paper and submitted at the service centre. Then the complaint will be registered or updated by the admin who is handling computer. The process is so difficult when the number of schools increases. The system should provide an interface to payroll for deductions. Several inquiry facilities should also be provided to view the schools.

After enquire by the admin and allocate the problem to specified person in the department. In some software, complaints are being recorded by visiting the office, but complaint registering takes time.

**2.1.1 Drawbacks**

The drawbacks of the existing system are,

* Customer satisfaction is difficult.
* There is no possibility for knowing details about the institutions or certifications.
* There are no categories to apply in existing systems.
* Proper approval is not maintained.

**2.2 Proposed System**

The proposed system will maintain customers effectively. When the client applies for the certificate, admin can be easily checked for the purpose of application.

Proposed system overcomes the limitations of existing systems. The system maintains the basic details, school details, extra details separately. It also maintains requirements separately and efficiently.

**2.2.1 Features**

The features of the proposed system are,

* Takes personal care for each client.
* Analysis for the future development of accreditations
* Client can apply it from anywhere.
* Can search client information efficiently and easily.
* Availability of customer can be searched easily.
* Using the system, user can also know about the certifications.
* The proposed system manages database without redundancy thereby following all normalization rules.

**3. System Design and Development**

* 1. **File Design**

The file design is one of the important features which mainly depend on the performance of the system. The project is designed with the 19 files. User can access the files like Register.html file, Login.html file, Home.html file, Category.html file, Aboutus.html file, Contact us.html, Application.html file.

Admin accesses files like Login.html file, Dashboard.html file, Home.html file, Category.html file, Aboutus.html file, Contact us.html, Application.html file.

* Register.html file:

Register.html file has been set as an optional page of the project. The page is designed with text boxes and buttons like register and sign in button. New users can be start with this form and by clicking log in button they can log in to the project.

* Login.html file:

Login.html file has been set as a start page of the project. The page is designed with user name and password fields and sign in button is situated to enter into home page. Admin can login with admin-id and password. Remember me is present to remember the password.

* Home.html file:

Home.html file is the main page for both user and admin. This will be displayed next to the moment of login. For admin the drop down menus such as Dashboard, Category, Contact us, about us, Application form will be visible. For users who login this website the menus such as Category, Application, Contact us, about us will be displayed in this home.html file.

* Elementary.html file:

Elementary.html file is a drop down menu of category field. It includes Pre-kg, LKG and UKG. It has text boxes of standard, staff, classes, ground and washrooms. A submit button is presented to store the data’s which is given by the users.

* Primary.html file:

Primary.html file is a drop down menu of category field. It includes standard from I to V. It has text boxes of standard, staff, classes, ground and washrooms. A submit button is presented to store the data’s which is given by the users.

* Secondary.html file:

Secondary.html file is a drop down menu of category field. It includes standard from VI to X. It has text boxes of standard, staff, classes, ground, wash rooms and laboratories for the subjects of computer and science. A submit button is presented to store the data’s which is given by the users.

* Highersecondary.html file:

Highersecondary.html file is a drop down menu of category field. It includes standard of XI and XII. It has text boxes of standard, staff, classes, ground, wash rooms and laboratories for the subjects of computer, physics, chemistry and biology. A submit button is presented to store the data’s which is given by the users.

* About us.html file:

About us.html is a file which views information about the certification and about the work of the accreditation by online.

* Contact.html file:

Contact.html file is used to provide the feedback and to contact the certification.

**3.1.1Input Design**

Input is any data or instructions entered into the memory of a computer. Two types of input are data and instructions. Data is a collection of unorganized items that can include words, numbers, pictures, sounds and video. A computer processes data into information, which is organized, meaningful, and useful. Instructions can be in the form of programs, commands, or user responses. A program is a series of instructions that tells a computer how to perform the tasks necessary to process data into information. A command is an instruction given to a computer program.

* User registration
* Application form
* Contact us
* **User registration**

New users will be registered to this registration page. After registration the user will be logging in to the web page. Many fields includes in this form such as user name, display name, email and password.

* **Application form**

This is used to apply to the certificate. Basic detail, school details and small description form is followed by the application form. The categories and about the school includes this fields.

* **Contact us**

This form is used to provide the feedback about the certifications with the required fields.

Input design features can ensure the reliability of the system and produce result from accurate data or they can result in the production of erroneous information.

**3.1.2Output Design**

Output design is very important concept in the computerized system, without reliable output the user may feel the entire system is unnecessary and avoids using it. The proper output design is important in any system and facilitates effective decision-making. The output design of this system includes various reports.

Output requirements are designed during system analysis. A good starting point for the output design is the data flow diagram. Human factors reduce issues for design involved addressing internal controls to ensure readability

An application is successful only when it can provide efficient and effective reports. Reports are actually presentable form of the data. The report generation should be useful to the management for future reference. The reports are the main source of information for user’s operators and management. Report generated are a permanent record of the transaction occurred. After any valid transactions have commenced the report of the same are generations and filled for future reference. Great care has been taken when designation the report as it plays an important role in decision-making.

* View application
* Approval or rejection of certification
* Generate certificate
* **View application**

Viewing the application details present in the application table in database. It displays the categories and basic details of the schools that are applied

* **Approval or rejection of certification**

Admin can accept or reject the request from the user if the details are not sufficient. This takes place in dashboard.

* **Generate certificate**

The certificate will be generated after the verification by admin. SMS and notifications will be sending to the user.

**3.1.3 DATABASE DESIGN**

An efficient database should fulfill certain conditions such as controlled redundancy for that purpose constraints have to be included in the tables in the project three constraints have been used namely primary key constraint not null constraint and foreign key constraint database is designed with 7 tables they are as follows

* Registration
* Login
* Elementary
* Primary
* Secondary
* Higher Secondary
* Application
* Contact

**Registration**

Database has a primary key “student \_ id” in the register table. Student \_ id field in the database design has been assigned primary key, so the values to be stored in the student \_ id field should be unique. Not null constraints are used in the columns like email \_ address, password and display \_ name.

**Login**

Login table has columns like display \_ name and password. Display \_ name has been assigned as primary Key, password column has been assigned with not null constraint

**Elementary**

Elementary table has six columns. They are student \_ id, standard, staff, classes, ground, Washrooms. Primary key constraint is being set to the column student \_ id. Other columns require value, so not null constraint has been set.

**Primary**

Primary table has six columns. They are student \_ id, standard, staff, classes, ground, Washrooms. Primary key constraint is being set to the column student \_ id. Other columns require value, so not null constraint has been set.

**Secondary**

Secondary table has eight columns. They are student \_ id, standard, staff, classes and ground labs like computer and science. Primary key constraint is being set to the column student \_id. Other columns require value, so not null constraint has been set.

**Higher Secondary**

Higher Secondary has ten columns. They are student \_id, standard, staff, classes and ground. Labs like computer, chemistry, physics and biology. Primary key constraint is being set to the column student \_id. Other columns require value, so not null constraint has been set.

**Contact**

Contact table has five columns. They are student \_id, name, email \_ address, message and subject. Primary key constraint is being set to the column student \_id. Other columns require value, so not null constraint has been set.

**3.2 System Design and Developmentation**

**3.5.1 Description of modules**

**Registration Module**

Inputs given through registration module are first name, last name, display name, email address, password, confirm password. User enters the values and selects registration button. When the registration button is pressed values entered in the registration form will be stored in the register table. Once the record is inserted the system will display the message box showing the information record added.

**Login Module**

Inputs given in the login module are display name and password. User presses sign in button as an input after entering values. If the matching record is found in the database, system will generate login successful message as an output and will redirect to home page, else the system will return to same page. The system will also validate whether the required columns are entered or not. If the required values are left empty, system will generate message like all fields are required.

**Elementary Module**

Inputs given in the elementary module are standard, staff, classes, ground and washrooms. User or admin presses submit it stores the data’s in the database, system will return to same page.

**Primary Module**

Inputs given in the elementary module are standard, staff, classes, ground and washrooms. User or admin presses submit it stores the data’s in the database, system will return to same page.

**Secondary Module**

Inputs given in the elementary module are standard, staff, classes, ground labs like computer, science and washrooms. User or admin presses submit it stores the data’s in the database, system will return to same page.

**Higher Secondary Module**

Inputs given in the elementary module are standard, staff, classes, ground labs like computer, chemistry, physics, biology and washrooms. User or admin presses submit it stores the data’s in the database, system will return to same page.

**Application Module**

Inputs given in the application module are category, address of the school, name of the school, chairman, secretary, manager, principle, no. of staffs, no. of buses and vans, contact, sports, extracurricular, library and class room for primary, secondary, elementary and higher secondary. User presses next button and finish button it stores the data in the database, system will return to same page.

**Contact Module**

Inputs given in the contact module are name, email address, subject and message. User presses send message button it stores the data in the database, system will return to same page.

**4. Testing and Implementation**

A test case is a set of conditions or variables under which a tester will determine if a requirement upon an application is partially or fully satisfied. The sample test cases for the project as follows.

**Screen name: Admin login page**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test id** | **Test case description** | **Action** | **Expected result** | **Actual result** | **Status** |
| TC001 | To check if the username and password matches with the database | Click submit button | If the username and password matches with the database home page will be redirected | Same as expected | **Pass** |
| TC002 | To check if the username and password not matches with the database | Click submit button | If the username and password matches with the database home page will be redirected | Where there is error message displayed | Fail |

**Screen name: User registration page**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test id** | **Test case description** | **Action** | **Expected result** | **Actual result** | **Status** |
| TC001 | To check if the username and password already exist in the database | Press register button | If user details already exist error message should be displayed | Message box is displayed | PASS |
| TC002 | To check if the username and password already exist in the database | Press register button | If user details already exist error message should be displayed | Message box is not displayed | FAIL |
| TC003 | To check if the username and password already exist in the database | Press register button | If user details does not exist new account should be created | Account created successfully | PASS |
| TC004 | To check if the username and password already exist in the database | Press register button | If the user details does not exist new account should be created | Account not created | FAIL |

**5. Conclusion**

The **“Online Accreditation System”** has been developed to satisfy all proposed requirements. The system is highly scalable and user friendly. Almost all the system objectives have been met. The system has been tested under all criteria. The system minimizes the problem arising in the existing system and it eliminates human error to zero level.

The software executes successfully by fulfilling the subjects of the project. Further extensions to the system can be made required with minor modification. The invention can be implemented in cryptography, or in computer hardware, firmware, and software or in combinations of them. Apparatus of the invention can be implemented in the computer program product tangibly embodied in a machine readable storage device for execution by a programmable processor executing a program of instructions to perform functions of the intention by operating an input data and generating output.

**SCOPE FOR FUTURE ENHANCEMENT**

This application avoids the manual work and the problems concern with it. It is an easy way to obtain information regarding the various certificates information that is present in the online accreditation system.

This project is an improved website better than the existing one’s regarding the information about the various activities. Still, the project can be done in a better way. Primarily, user can request information about their schools.

The enhancement that admin can add the searching option. The user can directly search to the particular online certifications which is approved and rejected in the given duration. These are the enhancements at present.

**APPENDIX**

1. **Data Flow Diagram**

A data flow diagram is graphical tool used to describe and analyze movement of data through a system these are the central tool and the basis from which the other components are developed the transformation of data from input to output though processed may be described logically and independently of physical components associated with the system these are known as the logical data flow diagrams.

The physical data flow diagrams show the actual implements and movement of data between people departments and workstations the idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater at the next level this is done until further explosion and an adequate amount of detail is describe d for analyst to understand the process

Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from this lead to the modular design a DFD is also known as a bubble chart has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design so it is the starting point of the design to the lowest level of detail a DFD consists of a series of bubbles joined by data flows in the system several rules of thumb are used in drawing DFD S they are as

Process should be numbered for an easy reference. Each name should be representative of the process

The direction of flow is from top to bottom and from left to right Data traditionally flow from source to the destination although they may flow back to the source One way to indicate this is to draw long flow line back to a source An alternative way is to draw long flow line back to a source An alternative way is to repeat the source symbol as Since it is used more than once in the DFD it is marked with a short diagonal.

When a process is exploded into lower details they are numbered.

The names of data stores and destinations are written capital letters. Process and dataflow names have the first letter of each work capitalized.

**Data flow diagram symbols**

Source or destination of data -

Data flow -

Process -

Views -

Level 0: Context level diagram

User

Admin

Database

Level 1: User

Register

User

After Login

Application

Level 2: Certification process

Admin

Display

Application

Saves & updates

Rejected

Accepted

Display

Level 3: Admin Login details Stores & Retrieves

Admin

Login Login

After Login Views & updates

Application

Display

Apply

Views Views

Views the users who are applied

Rejected request

Approved request

1. **Table Structure**

Table name : register

Primary key : Display\_name

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| First\_name | Varchar | 15 | Not null | First name of the customer |
| Last\_name | Varchar | 15 | Not null | Last name of the customer |
| Display\_name | Varchar | 25 | Primary key | Display name of customer |
| Email\_address | Varchar |  | Not null | Email address of the customer |
| Password | Varchar | 20 | Not null | Password for their account |

Table name: Admin\_login

Primary key: Display\_name

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Size** | **Constraint** | **Description** |
| Display\_name | Varchar | 20 | Primary key | Display name of the administrator |
| Password | Varchar | 20 | Not null | Password of the administrator |

Table name: Elementary

Primary key: Standard

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field name** | **Data Type** | **Size** | **Constraint** | **Description** |
| Standard | Varchar | 15 | Primary key | Enter the standard of the category |
| Staff | Int | 5 | Not null | Enter the count of staff |
| Classes | Int | 5 | Not null | Number of classes |
| Ground | Int | 8 |  | Size of the ground |
| Washrooms | Int | 10 | Not null | Number of washrooms |

Table name: Primary

Primary key: Standard

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field name** | **Data Type** | **Size** | **Constraint** | **Description** |
| Standard | Varchar | 15 | Primary key | Enter the standard of the category |
| Staff | Int | 5 | Not null | Enter the count of staff |
| Classes | Int | 5 | Not null | Number of classes |
| Ground | Int | 8 |  | Size of the ground |
| Washrooms | Int | 10 | Not null | Number of washrooms |

Table name: Secondary

Primary key: Standard

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fieldname** | **Data Type** | **Size** | **Constraint** | **Description** |
| Standard | Varchar | 15 | Primary key | Enter the standard of the category |
| Staff | Int | 5 | Not null | Enter the count of staff |
| Classes | Int | 5 | Not null | Number of classes |
| Ground | Int | 8 |  | Size of the ground |
| Computer | Int | 4 | Not null | Number of computer labs |
| Science | Int | 4 | Not null | Number of Science labs |

Table name: Higher\_secondary

Primary key: Standard

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fieldname** | **Data Type** | **Size** | **Constraint** | **Description** |
| Standard | Varchar | 15 | Primary key | Enter the standard of the category |
| Staff | Int | 5 | Not null | Enter the count of staff |
| Classes | Int | 5 | Not null | Number of classes |
| Ground | Int | 8 |  | Size of the ground |
| Computer | Int | 2 | Not null | Number of computer labs |
| Physics | Int | 2 | Not null | Number of physics labs |
| Biology | Int | 2 | Not null | Number of biology labs |
| Chemistry | Int | 2 | Not null | Number of biology labs |

Table name: Contact

Primary key: Email\_address

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field name** | **Data Type** | **Size** | **Constraint** | **Description** |
| Name | Varchar | 10 | Not null | Enter the name |
| Email\_address | Varchar | 20 | Not null | Enter the email what they are provided |
| Subject | Varchar | 15 | Not null | Select the subject |
| Message | Varchar | 50 | Not null | Write the message to send |

**C. Sample Coding**

**App.py**

from flask import \*

from flask\_sqlalchemy import SQLAlchemy

from sqlalchemy.orm import sessionmaker

from sqlalchemy import create\_engine

engine=create\_engine('postgresql://postgres:kgisl@localhost/onlineacc')

app=Flask(\_\_name\_\_)

app.config['SQLALCHEMY\_DATABASE\_URI'] = ` 'postgresql://postgres:kgisl@localhost/onlineacc'

app.config['SECRET\_KEY'] = "random string"

db = SQLAlchemy(app)

@app.route("/")

def index():

return render\_template("loginonline.html")

@app.route("/application")

def application():

return render\_template("application.html")

@app.route("/home")

def home():

return render\_template("home.html")

@app.route("/login")

def login():

return render\_template("loginonline.html")

@app.route("/register")

def register():

return render\_template("register.html")

@app.route("/elementary")

def elementary():

return render\_template("elementary.html")

@app.route("/primary")

def primary():

return render\_template("primary.html")

@app.route("/secondary")

def secondary():

return render\_template("secondary.html")

@app.route("/higher\_secondary")

def higher\_secondary():

return render\_template("higher\_secondary.html")

@app.route("/contact")

def contact():

return render\_template("contact.html")

@app.route("/dashboard")

def dashboard():

return render\_template("dashboard.html")

@app.route("/about")

def about():

return render\_template("about.html")

@app.route("/display")

def display():

return render\_template("display.html",application=application.query.all())

@app.route("/approved")

def approved():

return render\_template("approved.html",application=application.query.filter\_by(status=0).all())

@app.route("/rejected")

def rejected():

return render\_template("rejected.html",application=application.query.filter\_by(status=1).all())

### login validation ###

@app.route('/login\_check',methods= ["GET", "POST"])

def login\_check():

POST\_DISPLAY\_NAME=str(request.form['display\_name'])

POST\_PASSWORD=str(request.form['password'])

display\_name = request.form['display\_name']

session['display\_name']=display\_name

Session = sessionmaker(bind = engine)

s = Session()

query = s.query(register).filter(register.display\_name.in\_([POST\_DISPLAY\_NAME]),register.password.in\_([POST\_PASSWORD]))

result=query.first()

if request.form['password']=="admin" and request.form['display\_name']=="admin":

session['loggin\_in'] = True

return redirect(url\_for('home'))

return render\_template("home.html")

if result:

session['loggin\_in'] = True

return redirect(url\_for('home'))

return render\_template("home.html")

else:

flash('wrong password')

return redirect(url\_for('login'))

return render\_template('loginonline.html')

@app.route('/status\_1/<id>', methods=['POST', 'GET'])

def status\_1(id):

student\_id=application.query.get(id)

student\_id.status=0

db.session.commit()

return redirect(url\_for('approved'))

@app.route('/status\_2/<id>', methods=['POST', 'GET'])

def status\_2(id):

student\_id=application.query.get(id)

student\_id.status=1

db.session.commit()

return redirect(url\_for('rejected'))

#### database for register"""

class register(db.Model):

id=db.Column('student\_id',db.Integer,primary\_key=True)

first\_name=db.Column(db.String)

last\_name=db.Column(db.String)

display\_name=db.Column(db.String)

email\_address=db.Column(db.String)

password=db.Column(db.String)

confirm\_password=db.Column(db.String)

def \_\_init\_\_(self,first\_name,last\_name,display\_name,email\_address,password,confirm\_password):

self.first\_name=first\_name

self.last\_name=last\_name

self.display\_name=display\_name

self.email\_address=email\_address

self.password=password

self.confirm\_password=confirm\_password

@app.route("/register\_db",methods=["GET","POST"])

def register\_db():

if request.method == 'POST':

if not request.form['first\_name'] or not request.form['last\_name'] or not request.form['display\_name'] or not request.form['email\_address'] or not request.form['password'] or not request.form['confirm\_password']:

flash("Error")

else:

student=register(request.form['first\_name'],request.form['last\_name'],request.form['display\_name'],request.form['email\_address'],request.form['password'],request.form['confirm\_password'])

db.session.add(student)

db.session.commit()

return redirect(url\_for('register'))

return render\_template("register.html")

# """ db for elementary"""

class elementary(db.Model):

id=db.Column('student\_id',db.Integer,primary\_key=True)

standard=db.Column(db.String)

staff=db.Column(db.String)

classes=db.Column(db.String)

ground=db.Column(db.String)

washrooms=db.Column(db.String)

def \_\_init\_\_(self,standard,staff,classes,ground,washrooms):

self.standard=standard

self.staff=staff

self.classes=classes

self.ground=ground

self.washrooms=washrooms

@app.route("/elementary\_db",methods=["GET","POST"])

def elementary\_db():

if request.method == 'POST':

if not request.form['standard'] or not request.form['staff'] or not request.form['classes'] or not request.form['ground'] or not request.form['washrooms']:

flash("Error")

else:

student=elementary(request.form['standard'],request.form['staff'],request.form['classes'],request.form['ground'],request.form['washrooms'])

db.session.add(student)

db.session.commit()

return redirect(url\_for('elementary'))

return render\_template("elementary.html")

# """db for primary"""

class primary(db.Model):

id=db.Column('student\_id',db.Integer,primary\_key=True)

standard=db.Column(db.String)

staff=db.Column(db.String)

classes=db.Column(db.String)

ground=db.Column(db.String)

washrooms=db.Column(db.String)

def \_\_init\_\_(self,standard,staff,classes,ground,washrooms):

self.standard=standard

self.staff=staff

self.classes=classes

self.ground=ground

self.washrooms=washrooms

@app.route("/primary\_db",methods=["GET","POST"])

def primary\_db():

if request.method == 'POST':

if not request.form['standard'] or not request.form['staff'] or not request.form['classes'] or not request.form['ground'] or not request.form['washrooms']:

flash("Error")

else:

student=primary(request.form['standard'],request.form['staff'],request.form['classes'],request.form['ground'],request.form['washrooms'])

db.session.add(student)

db.session.commit()

return redirect(url\_for('primary'))

return render\_template("primary.html")

# """db for secondary"""

class secondary(db.Model):

id=db.Column('student\_id',db.Integer,primary\_key=True)

Standard=db.Column(db.String)

Staff=db.Column(db.String)

Classes=db.Column(db.String)

Ground=db.Column(db.String)

Computer=db.Column(db.String)

Science=db.Column(db.String)

wash\_rooms=db.Column(db.String)

def \_\_init\_\_(self,Standard,Staff,Classes,Ground,Computer,Science,wash\_rooms):

self.Standard=Standard

self.Staff=Staff

self.Classes=Classes

self.Ground=Ground

self.Computer=Computer

self.Science=Science

self.wash\_rooms=wash\_rooms

@app.route("/secondary\_db",methods=["GET","POST"])

def secondary\_db():

if request.method == 'POST':

if not request.form['Standard'] or not request.form['Staff'] or not request.form['Classes'] or not request.form['Ground'] or not request.form['Computer'] or not request.form['Science'] or not request.form['wash\_rooms']:

flash("Error")

else:

student=secondary(request.form['Standard'],request.form['Staff'],request.form['Classes'],request.form['Ground'],request.form['Computer'],request.form['Science'],request.form['wash\_rooms'])

db.session.add(student)

db.session.commit()

return redirect(url\_for('secondary'))

return render\_template("secondary.html")

# """db for higher secondary"""

class higher\_secondary(db.Model):

id=db.Column('student\_id',db.Integer,primary\_key=True)

Standard=db.Column(db.String)

Staff=db.Column(db.String)

Classes=db.Column(db.String)

Ground=db.Column(db.String)

Computer=db.Column(db.String)

physics=db.Column(db.String)

Science=db.Column(db.String)

wash\_rooms=db.Column(db.String)

def \_\_init\_\_(self,Standard,Staff,Classes,Ground,Computer,physics,Biology,chemistry,wash\_rooms):

self.Standard=Standard

self.Staff=Staff

self.Classes=Classes

self.Ground=Ground

self.Computer=Computer

self.physics=physics

self.Biology=Biology

self.chemistry=chemistry

self.wash\_rooms=wash\_rooms

@app.route("/higher\_secondary\_db",methods=["GET","POST"])

def higher\_secondary\_db():

if request.method == 'POST':

if not request.form['Standard'] or not request.form['Staff'] or not request.form['Classes'] or not request.form['Ground'] or not request.form['Computer'] or not request.form['physics'] or not request.form['Biology'] or not request.form['chemistry'] or not request.form['wash\_rooms']:

flash("Error")

else:

student=higher\_secondary(request.form['Standard'],request.form['Staff'],request.form['Classes'],request.form['Ground'],request.form['Computer'],request.form['physics'],request.form['Biology'],request.form['chemistry'],request.form['wash\_rooms'])

db.session.add(student)

db.session.commit()

return redirect(url\_for('higher\_secondary'))

return render\_template("higher\_secondary.html")

# """db for contact"""

class contact(db.Model):

id=db.Column('student\_id',db.Integer,primary\_key=True)

Name=db.Column(db.String)

Email\_Address=db.Column(db.String)

Message=db.Column(db.String)

Subject=db.Column(db.String)

def \_\_init\_\_(self,Name,Email\_Address,Message,Subject):

self.Name=Name

self.Email\_Address=Email\_Address

self.Message=Message

self.Subject=Subject

@app.route("/contact\_db",methods=["GET","POST"])

def contact\_db():

if request.method == 'POST':

if not request.form['Name'] or not request.form['Email\_Address'] or not request.form['Message'] or not request.form['Subject']:

flash("Error")

else:

student=contact(request.form['Name'],request.form['Email\_Address'],request.form['Message'],request.form['Subject'])

db.session.add(student)

db.session.commit()

return redirect(url\_for('contact'))

return render\_template("contact.html")

# """db for application"""

class application(db.Model):

id=db.Column('student\_id',db.Integer,primary\_key=True)

Category=db.Column(db.String)

Address=db.Column(db.String)

Secretray=db.Column(db.String)

Manager=db.Column(db.String)

buses\_vans=db.Column(db.String)

driving\_licence=db.Column(db.String)

School\_name=db.Column(db.String)

Chairman=db.Column(db.String)

Principal=db.Column(db.String)

staffs=db.Column(db.String)

form2\_I=db.Column(db.String)

form2\_II=db.Column(db.String)

form2\_III=db.Column(db.String)

form2\_IV=db.Column(db.String)

form2\_V=db.Column(db.String)

form2\_VI=db.Column(db.String)

form2\_VII=db.Column(db.String)

form2\_VIII=db.Column(db.String)

form2\_IX=db.Column(db.String)

form2\_X=db.Column(db.String)

form2\_XI=db.Column(db.String)

Room\_description=db.Column(db.String)

status=db.Column(db.Integer)

def \_\_init\_\_(self,Category,Address,Secretray,Manager,buses\_vans,driving\_licence,School\_name,Chairman,Principal,staffs,form2\_I,form2\_II,form2\_III,form2\_IV,form2\_V,form2\_VI,form2\_VII,form2\_VIII,form2\_IX,form2\_X,form2\_XI,Room\_description,status):

self.Category=Category

self.Address=Address

self.Secretray=Secretray

self.Manager = Manager

self.buses\_vans=buses\_vans

self.driving\_licence=driving\_licence

self.School\_name=School\_name

self.Chairman=Chairman

self.Principal=Principal

self.staffs=staffs

self.form2\_I=form2\_I

self.form2\_II=form2\_II

self.form2\_III=form2\_III

self.Room\_description=Room\_description

self.status=status

@app.route("/application\_db",methods=["GET","POST"])

def application\_db():

if request.method == 'POST':

if not request.form['Category'] or not request.form['Address'] or not request.form['Secretray'] or not request.form['Manager'] or not request.form['buses\_vans'] or not request.form['driving\_licence'] or not request.form['School\_name'] or not request.form['Chairman'] or not request.form['Principal'] or not request.form['staffs'] or not request.form['form2\_I'] or not request.form['form2\_II'] or not request.form['form2\_III'] or not request.form['form2\_IV'] or not request.form['form2\_V'] or not request.form['form2\_VI'] or not request.form['form2\_VII'] or not request.form['form2\_VIII'] or not request.form['form2\_IX'] or not request.form['form2\_X'] or not request.form['form2\_XI'] or not request.form['Room\_description'] or not request.form['status']:

pass

else:

student=application(request.form['Category'],request.form['Address'],request.form['Secretray'],request.form['Manager'],request.form['buses\_vans'],request.form['driving\_licence'],request.form['School\_name'],request.form['Chairman'],request.form['Principal'],request.form['staffs'],request.form['form2\_I'],request.form['form2\_II'],request.form['form2\_III'],request.form['form2\_IV'],request.form['form2\_V'],request.form['form2\_VI'],request.form['form2\_VII'],request.form['form2\_VIII'],request.form['form2\_IX'],request.form['form2\_X'],request.form['form2\_XI'],request.form['Room\_description'],request.form['status'])

db.session.add(student)

db.session.commit()

return redirect(url\_for('application'))

return render\_template("application.html")

if \_\_name\_\_=='\_\_main\_\_':

db.create\_all()

app.run(debug = True)

**home.html**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Home Page</title>

<meta charset="utf-8">

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

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<link rel="stylesheet" href="../static/homepage.css">

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

<style>

.jumbotron{

padding-top: 14px;

padding-bottom: 14px;

}

</style>

</head>

<body>

<nav class="navbar navbar-default sidebar" role="navigation">

<div class="container-fluid">

<div class="navbar-header">

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

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

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

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

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

</button>

</div>

<form class="form-horizontal" method="post" action="/login\_check">

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

<ul class="nav navbar-nav">

<h1>Welcome {{session.display\_name}}</h1>

{% with name=session.display\_name %}

{% if name=="admin" %}

<li class="active"><a href="/dashboard">Dashboard<span style="font-size:16px;" class="pull-right hidden-xs showopacity glyphicon glyphicon-home"></span></a></li>

{% endif %}

{% endwith %}

<li class="dropdown">

<a href="#" class="dropdown-toggle" data-toggle="dropdown">Category <span class="caret"></span><span style="font-size:16px;" class="pull-right hidden-xs showopacity glyphicon glyphicon-user"></span></a>

<ul class="dropdown-menu forAnimate" role="menu">

<li><a href="/elementary">Elementary</a></li>

<li><a href="/primary">Primary</a></li>

<li><a href="/secondary">Secondary</a></li>

<li><a href="/higher\_secondary">Higher Secondary</a></li>

</ul>

</li>

<li ><a href="/about">About Us<span style="font-size:16px;" class="pull-right hidden-xs showopacity glyphicon glyphicon-th-list"></span></a></li>

<li ><a href="/contact">Contact<span style="font-size:16px;" class="pull-right hidden-xs showopacity glyphicon glyphicon-phone"></span></a></li>

</ul>

</div>

</div>

</nav>

<div class="jumbotron text-center">

<p style="margin-bottom:3px;font-size:25px";>Welcome To PLPC Certifications</p>

</div>

<div class="container">

<div class="row">

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

<p style="margin-top: 90px;margin-left: 350px; font-size:200%";>

Applying for the certificate with specified requirements

</p>

</div>

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

<p style="margin-top: 150px;margin-left: 370px; font-size:180%";>

<p style="margin-top: 190px;margin-left: 35px; font-size:180%";>Click here..</p>

<!--form action="{{url\_for('application')}}">

<button>TO APPLY</button>

</form-->

<a href="{{url\_for('application')}}" class="announcement-text"> TO APPLY</a>

</a></p>

</div>

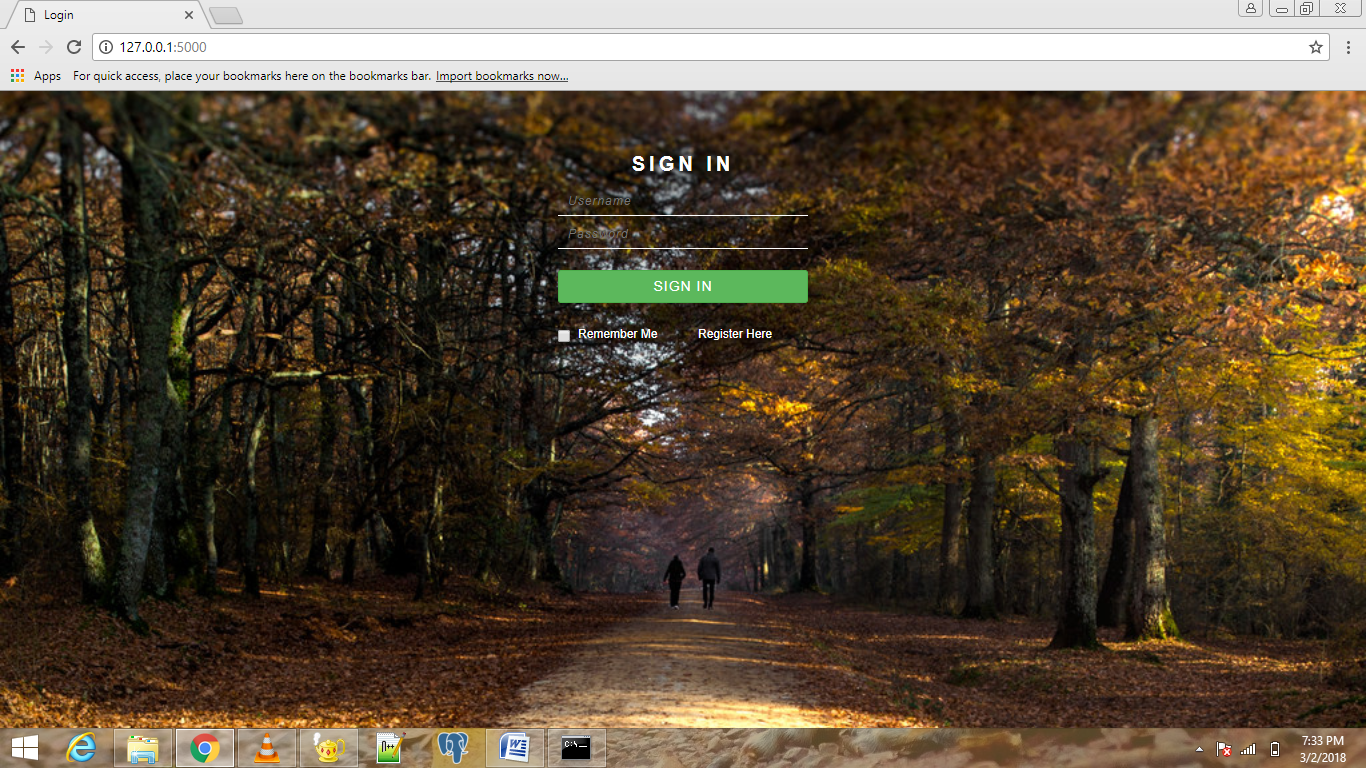
</div>

</div>

</body>

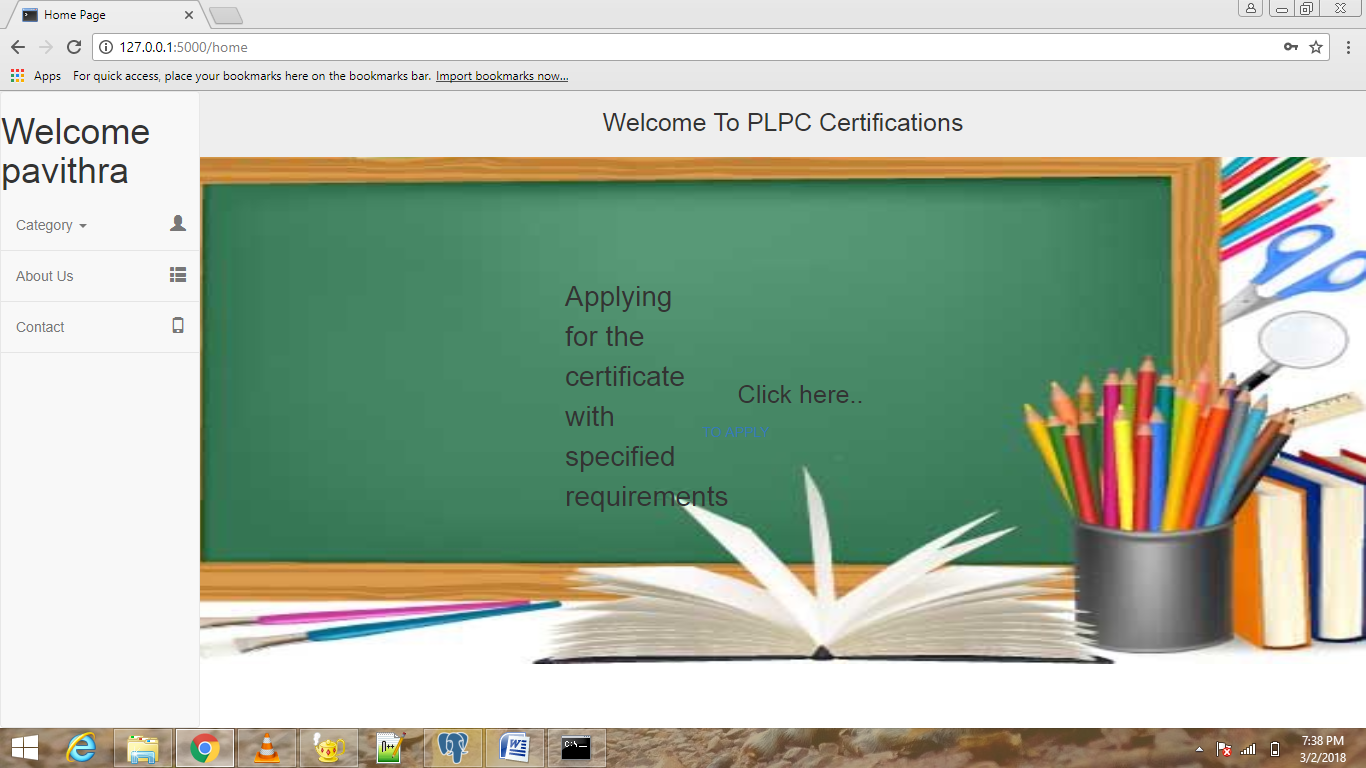
</html>

* 1. **Screen shots**



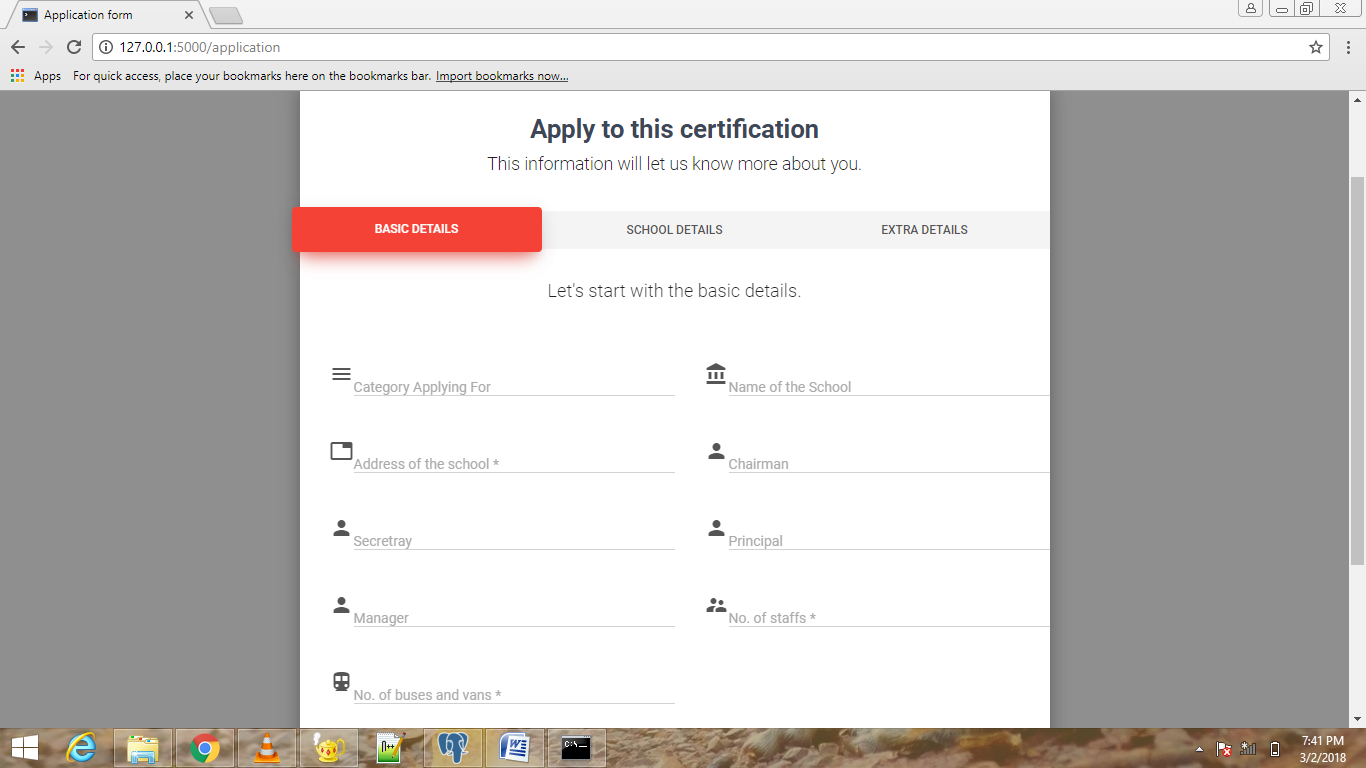
**Login form**

This form used to login for the user and admin. This form includes the display name and password. In this page register option is provided for new users.



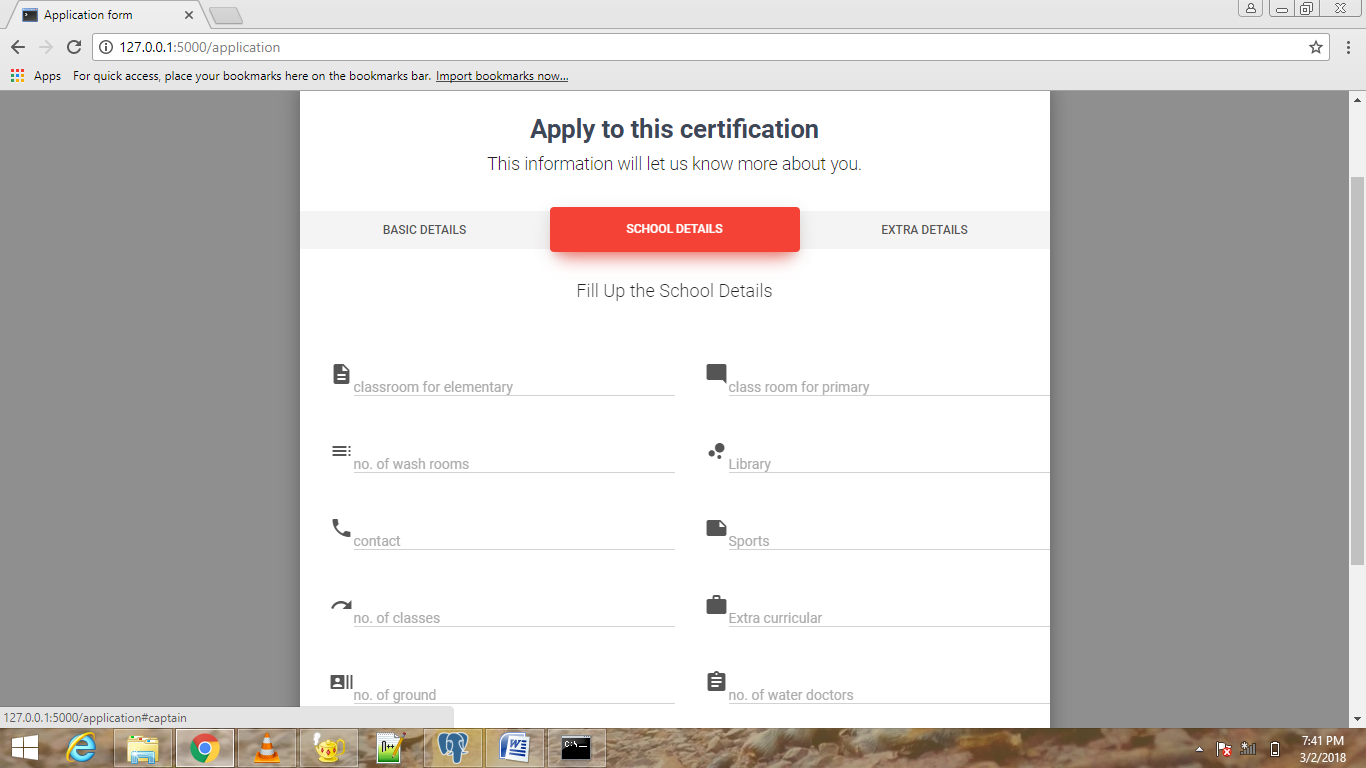
**Home Page**

This is a home page for the user login and application form is provided with the hyperlink. Click on to apply link and register with the required details. In these form categories, about us and contact form is present in the home page.

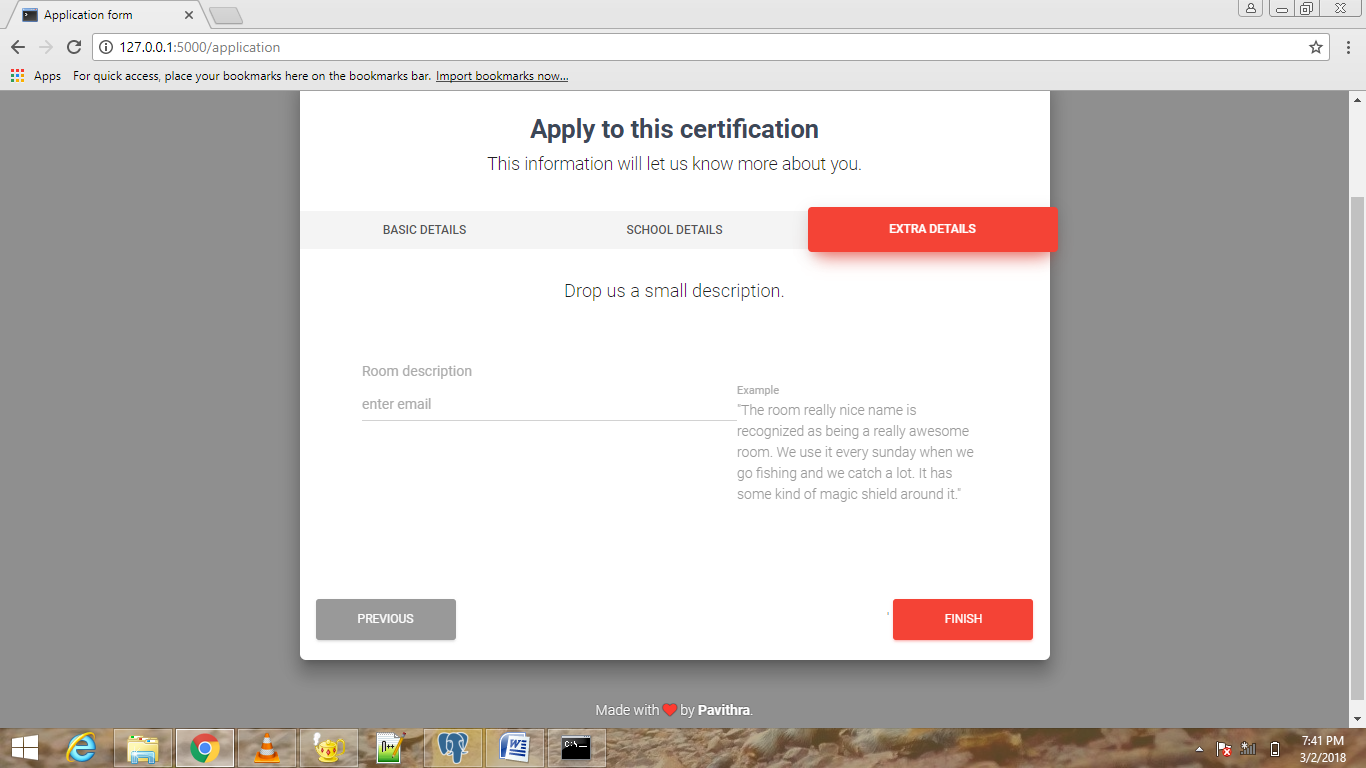


**Application form**

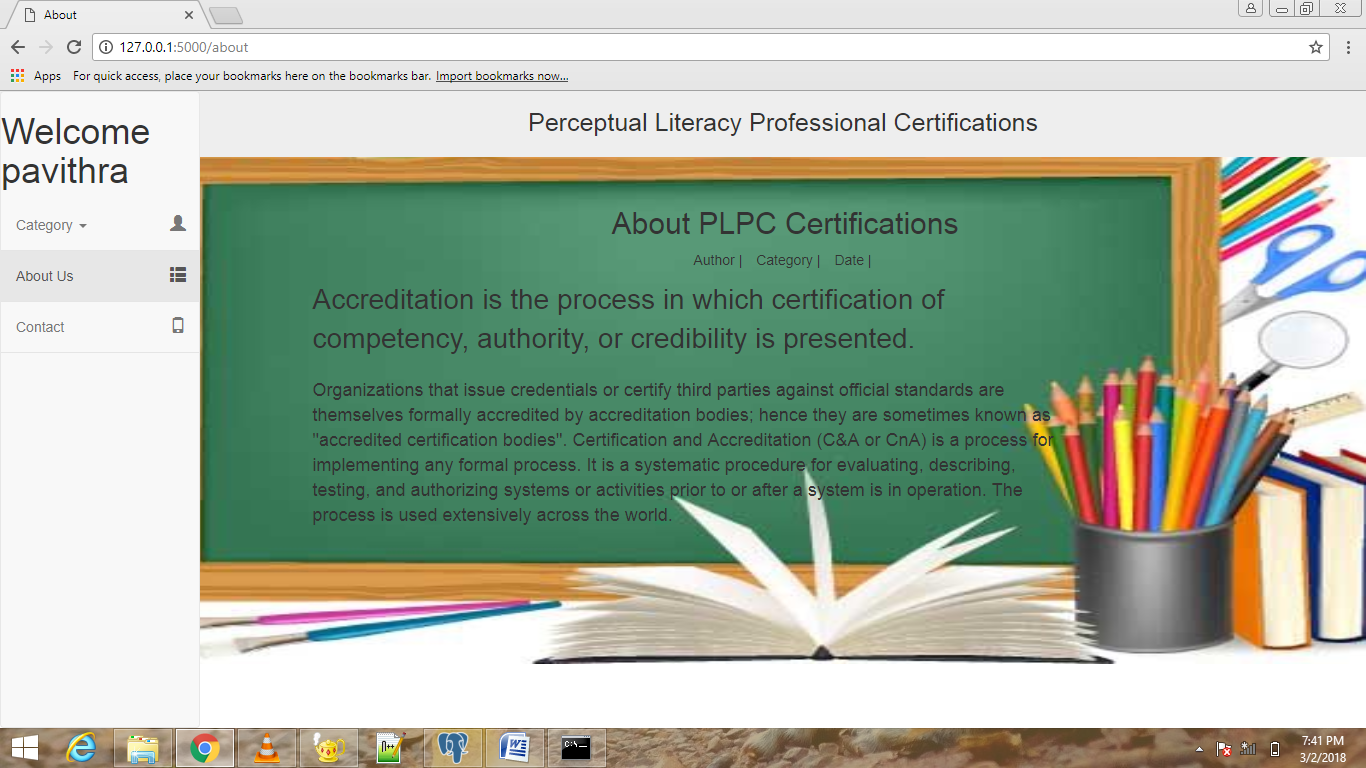
In this application form the basic details should be provided by the user. Some of the fields such as category, name of the school, chairman, secretary, principal, manager, no. of staffs are present in this form.



About this form school details should be given to apply no. of classes, wash rooms, sports, contact, no. of water doctors are some of the field that are mandatory to this application form.

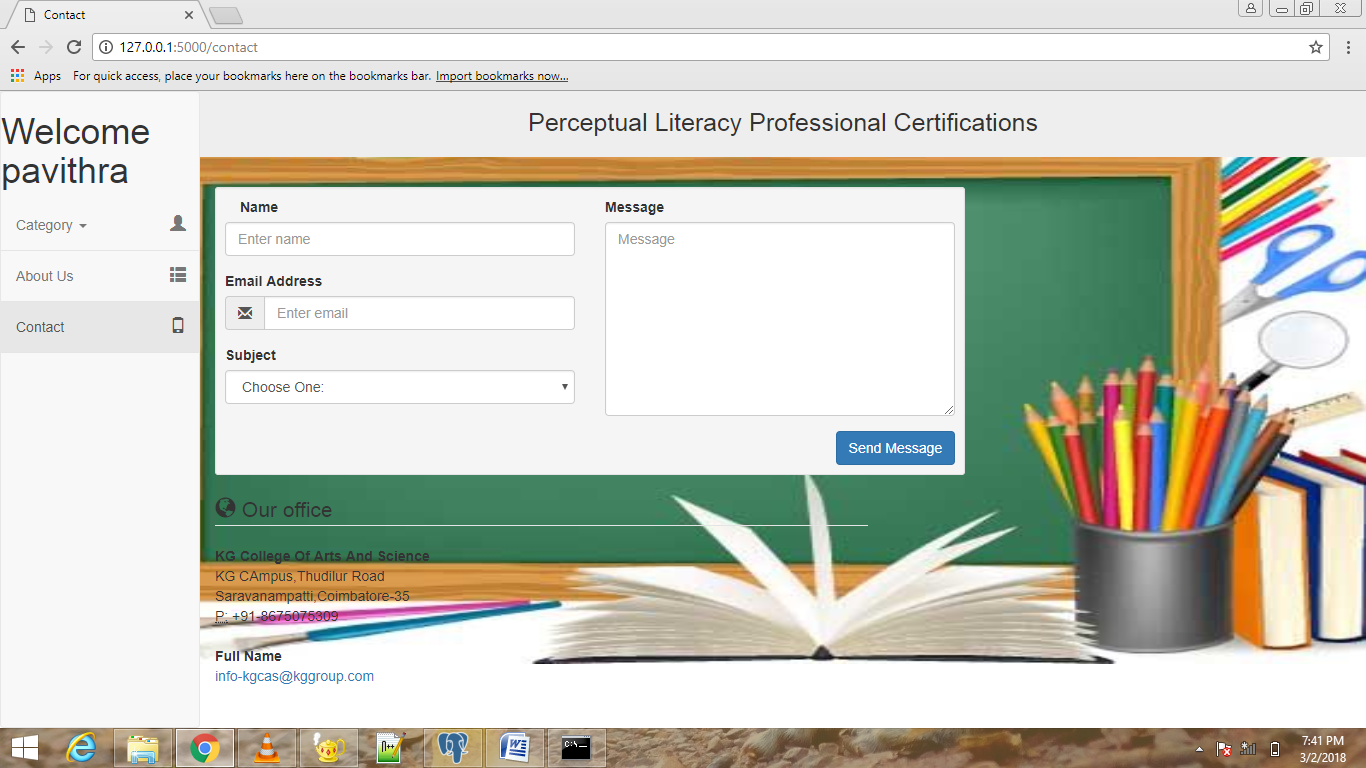


Small description column is provided to give the official email address to the admin. Press finishes button to stores the data’s.



**About us**

About the certifications are provided in this form. It establishes the certification process that is taking place.



It is a contact us form that gives about the feedback about this certification. Common details should be provided to this form. This will be viewed to the admin.