**LITIGATION INFORMATION GATEWAY**

**ABSTRACT**

Repository of departmental court cases Design and prepare a court case management software that has facility to record information like adding a case, adding lawyers (have facility select from existing list of lawyers), add invoice for each hearing and for different heads under which lawyers charge the clients. In short the system should provide end to end management of court case from client perspective and should be easy to use.

In this project develop a website for court cases and lawyers and their service. Customers can register and search for lawyers basing their requirement. In this project the user can search the lawyers based on category. Info related to lawyers will be there in website which customers can browse through and view their profile before contacting them. Customers can book a schedule for meeting with lawyer. We also had a discussion in regard to few clarifications and inputs from your side specifically to understand the project flow.

1. **INTRODUCTION**

**1.1 PROBLEM DEFINATION**

Court case can be take advantage of existing system. Use the meeting to determine whether the attorney is honest and forthcoming. Finding a good attorney may be the most important step you can take toward winning a legal case and it doesn't have to be a difficult task. You will, however, need to take your time with the search. Typically, you should look for an attorney with at least three to five years of experience practicing the type of law you need help with. Additionally, you should choose an attorney that currently practices in the area you need help with. It is always preferable to locate an attorney who has specialized expertise in the practice area that your case involves (e.g., malpractice law, bankruptcy law, etc.). It’s also a good idea to find attorney familiar with the courts and laws of the area where you live. We need to design and develop a website for lawyers and their service. Customers can register and search for lawyers basing their requirement. Info related to lawyers will be there in website which customers can browse through and view their profile before contacting them. Customers can book a schedule for meeting with lawyer.

**1.2COMPANY PROFILE**

**SAMCORE SOLUTION**

SamCore Solution as a leading IT solution and service provider provides innovative information technology - enabled solutions and services to meet the demands arising from social transformation, shaping new life styles for individuals and creating values for the society. Focusing on software technology, SamCore solution provides industry solutions and product engineering solutions, related software products & platforms, and services, through seamless integration of software and services, software and manufacturing, as well as technology and industrial management capacity.

SamCore Solution helps industry customers establish best practices in business development and management. The SamCore solution serves include real time projects, web designing, web hosting, software development and training etc, in many of which, has a leading market share. Notably, SamCore Solution has participated in the formulation of many national IT standards and specifications.

SamCore solution has the world’s leading product engineering capabilities, ranging from consultation, design, R&D, and integration to testing of embedded software, in the fields of automotive electronics, smart devices, digital home products, and IT products. The software provided by SamCore solution runs in a number of globally renowned brands.

**Our services:**

In this ever-changing environment, keeping a competitive edge means being able to anticipate and respond quickly to changing business conditions. SamCore solution is a global software development company providing IT solutions to enterprises worldwide. Combining proven expertise in technology, and an understanding of emerging business trends, SamCore delivers a range of software development solutions that includes e-business solutions, computer telephony, enterprise applications, professional web site design and development, product engineering, Electronic Health Records, CMS Software’s, Payment Gateway solutions, Time and attendance tracking software’s, Debt collection software’s, Appointment Reminder Solutions, Medical Transcription Services etc. We study, design, develop, enhance, customize, implement, maintain and support various aspects of information technology.

We are a professionally recognized software development company having huge experience in developing custom software development and application development best match to your need and requirements. We have expertise in working with a variety of customers from companies to individuals. Our successful assignments with client companies have established our reputation as superior providers of IT solutions. Services offered by SamCore include Software and Hardware development in Electronics Circuit Design, Hardware Design, Embedded System and IoT Services. We aim to carve a position in the forefront, and it is our continuing goal to gain the trust of our clients. Our Motto is to serve the purpose of our clients with perfection.

**Company Workers**

In SamCore Solution, more than 50 employees are working in the company. And it has good atmosphere. Company manger and employees have more knowledgeable in the field of software developing, designing and testing departments. In marketing team put full effort to developing the company’s growth and environment.

**Mission**

SamCore Solutions’ mission includes:

1. Providing high quality software development services, professional consulting and development outsourcing that would improve our customers’ operations;
2. Making access to information easier and securer (Enterprise Business);
3. Improving communication and data exchange (Business to Business);
4. Providing our customers with a Value for Money and
5. Providing our employees with meaningful work and advancement opportunities.

**Vision**

SamCore Solutions is a leading IT company for Consulting Services and Deployment of best of breed Business Solutions to top tier domestic and international customers.

**2. PROBLEM DESCRIPTION**

**MODULES**

* Admin
  + Login
  + Add lawyers Categories
  + View Lawyers Details
  + View User Details
* Lawyer
  + Register
  + Login
  + View Booking Details
  + Accept / Reject Booking
* User
  + Register
  + Login
  + View Lawyers Details
  + Book Lawyer
  + Make Payment
* Court module
  + Login
  + Add Case History
  + Add /Update Hearing Date
  + View Information

**MODULE DESCRIPTION**

**Admin**

* **Login**

Admin has to login before he/she can access the admin dashboard. In this module admin has unique user name and password. Admin has to view the overall information about this system.

* **View Lawyers Details**

In this module used to view the lawyer details such as lawyers name, address, contact number etc,.

* **Add Lawyers Categories**

In this module used to add the lawyers categories such as criminal lawyer, civil lawyer, family lawyer etc.

* **View User Details**

In this module helps admin to know user information. User details such as user name, location, case details, contact number etc.,

**User:**

* **Register**

In this module used to register the details, to database. After the Registration the database ask the password for an authentication. The details such as name, address, and phone no, email id, gender, location etc,

* **Login**

User has to login before he/she can access the user dashboard. The user can view all the lawyers and can search for a particular lawyer using name or ID.

* **View Lawyers Details**

In this module used to view the lawyer details such as lawyers name, address, contact number etc,.

* **Book Lawyer**

In this module used to book the particular lawyer. The user can view the lawyer’s information. The user can not book the already booked slot; he/she booked only available slots.

* **Make Payment**

In this module used to make payment. This module contains users card details like name, card no etc,.

**Lawyer:**

* **Register**

In this module used to register the details to database. After the Registration the database ask the password for an authentication. The details such as name, address, phone no, email id, gender, location etc,

* **Login**

Lawyers has to login before he/she can access the lawyer dashboard. After login the system lawyer can access the system

* **View Booking Details**

In this module used to view the booking details. The booking details contain user information’s and date.

* **Accept / Reject Booking**

In this module is used to accept or reject the booking information. In this module the lawyer can send the notification message to the user.

**Court module**

* **Login**

Court can be login to the system and view the case history details. Then update the case history details

* **Add Case History**

In this module, user adds case details which includes case name, hearing date and time details. Based on these case histories, court response the case details

* **Add /Update Hearing Date**

This module, we can add or update the existing case history for future verification

* **View Information**

In this module, view the overall information about the case details, lawyer details and user details

**3. SYSTEM STUDY**

**3.1 EXISTING SYSTEM**

Instinctively, some people have the ability to determine an individual's character within a few minutes of interacting with the person; however, there are a few personality traits that can also tip you off.Naturally, to cover such high overhead, fees rise and clients (like you) get stuck with the bill. Moreover, law firms operate at partnerships--a century old model that creates armies of associates to subsidize overhead and maximize partner profit. In existing system locating a good lawyer who can efficiently help with your particular problem may not be easy. Don't expect to locate a good lawyer by simply looking in the phone book or reading an advertisement. There's not enough information in these sources does not help you make a valid judgment.

**DISADVANTAGES OF EXISTING SYSTEM**

* In existing System so many possible to meet fraud lawyers.
* Lack of awareness that choose experienced lawyer.
* Lack Of security.
* Reduction in sharing information and customer services.
* Time consuming and costly to produce reports.

**3.2 PROPOSED SYSTEM**

The proposed system of EV

Charging mobile app to provide EV car owner the convenience of locating

charging stations on Google map, vacancy of charging slots, getting status updates

on charging. Help to easy way of charging of EV station and ensure smooth

journeys long distance

The proposed system of EV

Charging mobile app to provide EV car owner the convenience of locating

charging stations on Google map, vacancy of charging slots, getting status updates

on charging. Help to easy way of charging of EV station and ensure smooth

journeys long distance

The proposed system of EV

Charging mobile app to provide EV car owner the convenience of locating

charging stations on Google map, vacancy of charging slots, getting status updates

on charging. Help to easy way of charging of EV station and ensure smooth

journeys long distance

In this application helps user to find lawyer based on their location. Many attorneys offer an initial consultation free of charge. Focus your efforts on finding a lawyer that has dealt with your specific legal issue in the past and that you get along with personally. The proposed system helps to reduce the time to find the right lawyer will be worth it, as they are more likely to help you win your case. This System will provide information about each attorney working for the firm. This system helps to look at each attorney’s educational background and work history. It’s also a good idea to find attorney familiar with the courts and laws of the area where you live.

**ADVANTAGES OF PROPOSED SYSTEM**

* Reduce Manual Work.
* Avoiding the wasting time to search good lawyer.
* Increase Trust and Quality.
* It is a great resource for information about lawyers.

**3.4DATA FLOW DIAGRAMS**

A two-dimensional diagram explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow diagram must identify external inputs and outputs, determine how the inputs and outputs relate to each other, and explain with graphics how these connections relate and what they result in. This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects.

**Data flow Symbols:**

|  |  |
| --- | --- |
| **Symbol** | **Description** |
| http://cpanel.stpaulsscience.org/gceict/specifications/ocr/unit3/sdlc/dfd/entity.jpg | An **entity**. A source of data or a destination for data. |
| http://cpanel.stpaulsscience.org/gceict/specifications/ocr/unit3/sdlc/dfd/process.jpg | A **process** or task that is performed by the system. |
| http://cpanel.stpaulsscience.org/gceict/specifications/ocr/unit3/sdlc/dfd/store.jpg | A **data store**, a place where data is held between processes. |
| http://cpanel.stpaulsscience.org/gceict/specifications/ocr/unit3/sdlc/dfd/flow.jpg | A **data flow**. |

**LEVEL 0**

The Level 0 DFD shows how the system is divided into 'sub-systems' (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.

Database

Admin

Lawyer

User

Court

**LEVEL-1**

The next stage is to create the Level 1 Data Flow Diagram. This highlights the main functions carried out by the system. As a rule, to describe the system was using between two and seven functions - two being a simple system and seven being a complicated system. This enables us to keep the model manageable on screen or paper.

Admin

View Lawyer Details

View User Details

Database

Login

**3.5 DATA DICTIONARY**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **TYPE** | **DESCRIPTION** | **SAMPLE VALUES** |
| uname | varchar(50) | Specifies the user name | Ram |
| password | varchar(50) | Specifies the password | \*\*\*\* |
| ***id*** | int(50) | Specifies the id | 01 |
| lid | varchar(50) | Specifies the lawyer id | 20 |
| uid | varchar(50) | Specifies the user id | 10 |
| status | varchar(50) | Specifies the status | action |
| data | varchar(50) | Specifies the data | crime |
| cno | varchar(50) | Specifies the complaint no | 25 |
| categories | varchar(100) | Specifies the cetegories | crime |
| lname | varchar(50) | Specifies the lawyer name | Raman |
| hdate | varchar(50) | Specifies the hearing date | 10/02/2020 |
| history | varchar(1000) | Specifies the history | processing |
| gender | varchar(100) | Specifies the gender | Male |
| address | varchar(100) | Specifies the address | Trichy |
| city | varchar(100) | Specifies the city | Trichy |
| email | varchar(50) | Specifies the email | [ahj@gmail.com](mailto:ahj@gmail.com) |
| image | varchar(100) | Specifies the image | Uj.png |
| accno | varchar(50) | Specifies the account no | 5698741230256 |
| amount | varchar(50) | Specifies the amount | 1500 |

**4. SYSTEM CONFIGURATION**

**4.1 HARDWARE REQUIREMENTS**

* CPU Type : Any Intel or AMD x64 with 2 Core processor
* Clock speed  : 2 GHz and Above
* RAM size : 4 GB Min / 8 GB Recommended
* Disk Space : 20 GB
* Display / Resolution : Min 14” / 1024 by 768 or Higher Display Resolution
* Keyboard & Mouse : USB or Internal

**4.2 SOFTWARE REQUIREMENTS**

* Operating System : Windows 7 / 10 / 11
* Frond End : Python
* Back End : MYSQL (WAMP Server)
* Tool : Python 3.7
* IDE : PyCharm 2019.3

1. **OVERVIEW OF THE SOFTWARE**

**Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation. 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, which espoused the opposite approach. While offering choice in coding methodology, the Python philosophy rejects exuberant syntax (such as that of Perl) in favor of a simpler, less-cluttered grammar. As Alex Martelli put it: "To describe something as 'clever' is not considered a compliment in the Python culture."Python's philosophy rejects the Perl "there is more than one way to do it" approach to language design in favour of "there should be one—and preferably only one—obvious way to do it".

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of CPython that would offer marginal increases in speed at the cost of clarity.[ When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. CPython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard for and bar.

A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic. Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as Pythonists, Pythonistas, and Pythoneers. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python’s initial development was spearheaded by Guido van Rossum in the late 1980s. Today, it is developed by the Python Software Foundation. Because Python is a multiparadigm language, Python programmers can accomplish their tasks using different styles of programming: object oriented, imperative, functional or reflective. Python can be used in Web development, numeric programming, game development, serial port access and more.

There are two attributes that make development time in Python faster than in other programming languages:

1. Python is an interpreted language, which precludes the need to compile code before executing a program because Python does the compilation in the background. Because Python is a high-level programming language, it abstracts many sophisticated details from the programming code. Python focuses so much on this abstraction that its code can be understood by most novice programmers.
2. Python code tends to be shorter than comparable codes. Although Python offers fast development times, it lags slightly in terms of execution time. Compared to fully compiling languages like C and C++, Python programs execute slower. Of course, with the processing speeds of computers these days, the speed differences are usually only observed in benchmarking tests, not in real-world operations. In most cases, Python is already included in Linux distributions and Mac OS X machines.

**BACK END**

**MY SQL**

MySQL is the world's most used open source [relational database management system](http://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS) as of 2008 that run as a server providing multi-user access to a number of databases. The MySQL development project has made its [source code](http://en.wikipedia.org/wiki/Source_code) available under the terms of the [GNU General Public License](http://en.wikipedia.org/wiki/GNU_General_Public_License), as well as under a variety of [proprietary](http://en.wikipedia.org/wiki/Proprietary_software) agreements. MySQL was owned and sponsored by a single [for-profit](http://en.wikipedia.org/wiki/Business) firm, the [Swedish](http://en.wikipedia.org/wiki/Sweden) company [MySQL AB](http://en.wikipedia.org/wiki/MySQL_AB), now owned by [Oracle Corporation](http://en.wikipedia.org/wiki/Oracle_Corporation).

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack—LAMP is an acronym for "[Linux](http://en.wikipedia.org/wiki/Linux), [Apache](http://en.wikipedia.org/wiki/Apache_HTTP_Server), MySQL, [Perl](http://en.wikipedia.org/wiki/Perl)/[PHP](http://en.wikipedia.org/wiki/PHP)/[Python](http://en.wikipedia.org/wiki/Python_%28programming_language%29)." [Free-software](http://en.wikipedia.org/wiki/Free_software)-open source projects that require a full-featured database management system often use MySQL.

For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: [TYPO3](http://en.wikipedia.org/wiki/TYPO3), [Joomla](http://en.wikipedia.org/wiki/Joomla), [Word Press](http://en.wikipedia.org/wiki/WordPress), [phpBB](http://en.wikipedia.org/wiki/PhpBB), [MyBB](http://en.wikipedia.org/wiki/MyBB), [Drupal](http://en.wikipedia.org/wiki/Drupal) and other software built on the [LAMP](http://en.wikipedia.org/wiki/LAMP_%28software_bundle%29) software stack. MySQL is also used in many high-profile, large-scale [World Wide Web](http://en.wikipedia.org/wiki/World_Wide_Web) products, including Wikipedia, Google(though not for searches), [Imagebook](http://en.wikipedia.org/wiki/Facebook)[Twitter](http://en.wikipedia.org/wiki/Twitter), [Flickr](http://en.wikipedia.org/wiki/Flickr), [Nokia.com](http://en.wikipedia.org/wiki/Nokia), and [YouTube](http://en.wikipedia.org/wiki/YouTube).

Interimages

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

Graphical

The official MySQL Workbench is a free integrated environment developed by MySQL AB, that enables users to graphically administer MySQL databases and visually design database structures. MySQL Workbench replaces the previous package of software, [MySQL GUI Tools](http://en.wikipedia.org/wiki/MySQL_GUI_Tools). Similar to other third-party packages, but still considered the authoritative MySQL frontend, MySQL Workbench lets users manage database design & modeling, SQL development (replacing MySQL Query Browser) and Database administration (replacing MySQL Administrator).

MySQL Workbench is available in two editions, the regular free and open source Community Edition which may be downloaded from the MySQL website, and the proprietary Standard Edition which extends and improves the feature set of the Community Edition.

**Command line**

`MySQL ships with some command line tools. Third-parties have also developed tools to manage a MySQL server, some listed below.

* Maatkit - a cross-platform toolkit for MySQL, [PostgreSQL](http://en.wikipedia.org/wiki/PostgreSQL) and [Memcached](http://en.wikipedia.org/wiki/Memcached), developed in Perl Maatkit can be used to prove replication is working correctly, fix corrupted data, automate repetitive tasks, and speed up servers. Maatkit is included with several GNU/Linux distributions such as [CentOS](http://en.wikipedia.org/wiki/CentOS) and [Debian](http://en.wikipedia.org/wiki/Debian) and packages are available for Programming

MySQL works on many different [system platforms](http://en.wikipedia.org/wiki/System_platform), including [AIX](http://en.wikipedia.org/wiki/AIX_operating_system), [BSDi](http://en.wikipedia.org/wiki/BSD/OS), [FreeBSD](http://en.wikipedia.org/wiki/FreeBSD), [HP-UX](http://en.wikipedia.org/wiki/HP-UX), [eComStation](http://en.wikipedia.org/wiki/EComStation), [i5/OS](http://en.wikipedia.org/wiki/IBM_i5/OS), [IRIX](http://en.wikipedia.org/wiki/IRIX), Linux, [Mac OS X](http://en.wikipedia.org/wiki/Mac_OS_X), [Microsoft Windows](http://en.wikipedia.org/wiki/Microsoft_Windows), [NetBSD](http://en.wikipedia.org/wiki/NetBSD), [Novell NetWare](http://en.wikipedia.org/wiki/Novell_NetWare), [OpenBSD](http://en.wikipedia.org/wiki/OpenBSD), [OpenSolaris](http://en.wikipedia.org/wiki/OpenSolaris), [OS/2](http://en.wikipedia.org/wiki/OS/2) Warp, [QNX](http://en.wikipedia.org/wiki/QNX), [Solaris](http://en.wikipedia.org/wiki/Solaris_%28operating_system%29), [Symbian](http://en.wikipedia.org/wiki/Symbian), [SunOS](http://en.wikipedia.org/wiki/SunOS), [SCO OpenServer](http://en.wikipedia.org/wiki/SCO_OpenServer), SCO [UnixWare](http://en.wikipedia.org/wiki/UnixWare), [Sanos](http://en.wikipedia.org/wiki/Sanos) and [Tru64](http://en.wikipedia.org/wiki/Tru64). A port of MySQL to [OpenVMS](http://en.wikipedia.org/wiki/OpenVMS) also exists.[[32]](http://en.wikipedia.org/wiki/MySQL#cite_note-31)

MySQL is written in [C](http://en.wikipedia.org/wiki/C_%28programming_language%29) and [C++](http://en.wikipedia.org/wiki/C%2B%2B). Its SQL parser is written in [yacc](http://en.wikipedia.org/wiki/Yacc), and a home-brewed [lexical analyzer](http://en.wikipedia.org/wiki/Lexical_analysis). Many [programming languages](http://en.wikipedia.org/wiki/Programming_language) with language-specific [APIs](http://en.wikipedia.org/wiki/Application_programming_interface) include [libraries](http://en.wikipedia.org/wiki/Library_%28computing%29) for accessing MySQL databases. These include MySQL Connector/Net for integration with Microsoft's [Visual Studio](http://en.wikipedia.org/wiki/Visual_Studio) (languages such as [C#](http://en.wikipedia.org/wiki/C_Sharp_%28programming_language%29) and [VB](http://en.wikipedia.org/wiki/Visual_Basic) are most commonly used) and the JDBC driver for Java. In addition, an [ODBC](http://en.wikipedia.org/wiki/ODBC)interimage called [MyODBC](http://en.wikipedia.org/wiki/MyODBC) allows additional programming languages that support the ODBC interimage to communicate with a MySQL database, such as [ASP](http://en.wikipedia.org/wiki/Active_Server_Pages) or [ColdFusion](http://en.wikipedia.org/wiki/Adobe_ColdFusion). The [HTSQL](http://en.wikipedia.org/wiki/HTSQL) - [URL](http://en.wikipedia.org/wiki/Uniform_resource_locator)-based query method also ships with a MySQL adapter, allowing direct interaction between a MySQL database and any web client via structured URLs.

**Features**

As of April 2009, MySQL offered MySQL 5.1 in two different variants: the open source MySQL Community Server and the commercial [Enterprise Server](http://en.wikipedia.org/wiki/MySQL_Enterprise). MySQL 5.5 is offered under the same licences. They have a common code base and include the following features:

* A broad subset of [ANSI SQL 99](http://en.wikipedia.org/wiki/SQL:1999), as well as extensions
* Cross-platform support
* [Stored procedures](http://en.wikipedia.org/wiki/Stored_procedure)
* [Triggers](http://en.wikipedia.org/wiki/Database_trigger)
* [Cursors](http://en.wikipedia.org/wiki/Cursor_%28databases%29)
* Updatable [Views](http://en.wikipedia.org/wiki/View_%28database%29)
* [Information schema](http://en.wikipedia.org/wiki/Information_schema)
* Strict mode (ensures MySQL does not truncate or otherwise modify data to conform to an underlying data type, when an incompatible value is inserted into that type)
* [X/Open XA](http://en.wikipedia.org/wiki/X/Open_XA)[distributed transaction processing](http://en.wikipedia.org/wiki/Distributed_transaction_processing) (DTP) support; [two phase commit](http://en.wikipedia.org/wiki/Two-phase-commit_protocol) as part of this, using Oracle's [InnoDB](http://en.wikipedia.org/wiki/InnoDB) engine
* Independent [storage engines](http://en.wikipedia.org/wiki/Storage_engine) ([MyISAM](http://en.wikipedia.org/wiki/MyISAM" \o "MyISAM) for read speed, InnoDB for transactions and [referential integrity](http://en.wikipedia.org/wiki/Referential_integrity), [MySQL Archive](http://en.wikipedia.org/wiki/MySQL_Archive) for storing historical data in little space)
* Transactions with the InnoDB, and Cluster storage engines; savepoints with InnoDB
* [SSL](http://en.wikipedia.org/wiki/Secure_Sockets_Layer) support
* Query [caching](http://en.wikipedia.org/wiki/Cache_%28computing%29)
* Sub-[SELECTs](http://en.wikipedia.org/wiki/Select_%28SQL%29) (i.e. nested SELECTs)
* Replication support (i.e. Master-Master Replication & Master-Slave Replication) with one master per slave, many slaves per master, no automatic support for multiple masters per slave.
* Full-text [indexing](http://en.wikipedia.org/wiki/Index_%28database%29) and searching using MyISAM engine
* Embedded database library
* [Unicode](http://en.wikipedia.org/wiki/Unicode) support (however prior to 5.5.3 [UTF-8](http://en.wikipedia.org/wiki/UTF-8) and [UCS-2](http://en.wikipedia.org/wiki/UTF-16/UCS-2) encoded strings are limited to the [BMP](http://en.wikipedia.org/wiki/Basic_Multilingual_Plane), in 5.5.3 and later use utf8mb4 for full unicode support)
* [ACID](http://en.wikipedia.org/wiki/Atomicity,_consistency,_isolation,_durability) compliance when using transaction capable storage engines (InnoDB and Cluster)
* Partititoned tables with pruning of partitions in optimiser
* [Shared-nothing](http://en.wikipedia.org/wiki/Shared-nothing) clustering through [MySQL Cluster](http://en.wikipedia.org/wiki/MySQL_Cluster)
* Hot backup (via mysqlhotcopy) under certain conditions
* Multiple storage engines, allowing one to choose the one that is most effective for each table in the application (in MySQL 5.0, storage engines must be compiled in; in MySQL 5.1, storage engines can be dynamically loaded at [run time](http://en.wikipedia.org/wiki/Run_time_%28program_lifecycle_phase%29)): Native storage engines (MyISAM, [Falcon](http://en.wikipedia.org/wiki/Falcon_%28storage_engine%29), Merge, Memory (heap), [Federated](http://en.wikipedia.org/wiki/MySQL_Federated), Archive, [CSV](http://en.wikipedia.org/wiki/Comma-separated_values), Blackhole, Cluster, EXAMPLE, [Maria](http://en.wikipedia.org/wiki/Maria_%28storage_engine%29), and InnoDB, which was made the default as of 5.5). Partner-developed storage engines ([solidDB](http://en.wikipedia.org/wiki/SolidDB" \o "SolidDB), NitroEDB, [ScaleDB](http://en.wikipedia.org/w/index.php?title=User:MPH007&action=edit&redlink=1), TokuDB, [Infobright](http://en.wikipedia.org/wiki/Infobright) (formerly Brighthouse), [Kickfire](http://en.wikipedia.org/wiki/Kickfire), XtraDB, [IBM DB2](http://en.wikipedia.org/wiki/IBM_DB2)). InnoDB used to be a partner-developed storage engine, but with recent acquisitions, [Oracle](http://en.wikipedia.org/wiki/Oracle_Corporation) now owns both MySQL core and InnoDB.

**6. DESIGN AND DEVELOPMENT**

**6.1 ARCHITECTURAL DESIGN**

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. System architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called architecture description languages (ADLs).

**Various organizations define systems architecture in different ways, including:**

* An allocated arrangement of physical elements which provides the design solution for a consumer product or life-cycle process intended to satisfy the requirements of the functional architecture and the requirements baseline.
* Architecture comprises the most important, pervasive, top-level, strategic inventions, decisions, and their associated rationales about the overall structure (i.e., essential elements and their relationships) and associated characteristics and behavior.
* If documented, it may include information such as a detailed inventory of current hardware, software and networking capabilities; a description of long-range plans and priorities for future purchases, and a plan for upgrading and/or replacing dated equipment and software
* The composite of the design architectures for products and their life-cycle processes.

**LITIGATION INFORMATION GATEWAY**

User

Admin

Registration

Login

View lawyer details

Login

Add lawyers

View lawyers

View user details

Make payment

Lawyer

Registration

Login

View booking details

Book Lawyer

Accept/ Reject booking

Court

Login

Add case history

Add/ update hearing date

View information

**6.2 DATABASE DESIGN**

A table is a data structure that organizes information into rows and columns. It can be used to both store and display data in a structured format. For example, databases store data in tables so that information can be quickly accessed from specific rows. Websites often use tables to display multiple rows of data on page. Spreadsheets combine both purposes of a table by storing and displaying data in a structured format.

Databases often contain multiple tables, with each one designed for a specific purpose. For example, a company database may contain separate tables for employees, clients, and suppliers. Each table may include its own set of fields, based on what data the table needs to store. In database tables, each field is considered a column, while each entry (or record), is considered a row. A specific value can be accessed from the table by requesting data from an individual column and row.

## Table Name: admin

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Default** |
| uname | varchar(50) | Yes | NULL |
| password | varchar(50) | Yes | NULL |

## Table Name: book

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Default** |
| ***id*** | int(50) | Yes | NULL |
| lid | varchar(50) | Yes | NULL |
| uid | varchar(50) | Yes | NULL |
| status | varchar(50) | Yes | NULL |
| data | varchar(50) | Yes | NULL |

## Table Name: casehistory

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Default** |
| ***id*** | int(50) | Yes | NULL |
| cno | varchar(50) | Yes | NULL |
| status | varchar(50) | Yes | NULL |
| categories | varchar(100) | Yes | NULL |
| uname | varchar(50) | Yes | NULL |
| lname | varchar(50) | Yes | NULL |
| lname1 | varchar(50) | Yes | NULL |
| bench | varchar(50) | Yes | NULL |
| hdate | varchar(50) | Yes | NULL |
| ctitle | varchar(100) | Yes | NULL |
| history | varchar(1000) | Yes | NULL |

## Table Name: categories

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Default** |
| ***id*** | int(50) | Yes | NULL |
| slno | varchar(50) | Yes | NULL |
| categories | varchar(50) | Yes | NULL |

## Table Name: lreg

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Default** |
| ***id*** | int(50) | Yes | NULL |
| name | varchar(100) | Yes | NULL |
| gender | varchar(100) | Yes | NULL |
| address | varchar(100) | Yes | NULL |
| city | varchar(100) | Yes | NULL |
| pnumber | varchar(10) | Yes | NULL |
| email | varchar(50) | Yes | NULL |
| categories | varchar(50) | Yes | NULL |
| image | varchar(100) | Yes | NULL |
| uname | varchar(50) | Yes | NULL |
| password | varchar(50) | Yes | NULL |

## Table Name: pay

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Default** |
| ***id*** | int(50) | Yes | NULL |
| lname | varchar(50) | Yes | NULL |
| uname | varchar(50) | Yes | NULL |
| accno | varchar(50) | Yes | NULL |
| amount | varchar(50) | Yes | NULL |
| status | varchar(50) | Yes | NULL |

## Table Name: user

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Default** |
| ***id*** | int(50) | Yes | NULL |
| name | varchar(100) | Yes | NULL |
| gender | varchar(50) | Yes | NULL |
| address | varchar(100) | Yes | NULL |
| pnumber | varchar(10) | Yes | NULL |
| email | varchar(50) | Yes | NULL |
| image | varchar(50) | Yes | NULL |
| uname | varchar(50) | Yes | NULL |
| password | varchar(50) | Yes | NULL |

**6.3 INPUT AND OUTPUT DESIGN**

**INPUT DESIGN**

In an information system, input is the raw data that is processed to produce output. During the

input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

Therefore, the quality of system input determines the quality of system output. Well designed

input forms and screens have following properties −

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

### Objectives for Input Design

The objectives of input design are −

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

**OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user.

Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

**7. IMPLEMENTATION AND TESTING**

**IMPLEMENTATION**

Implementation includes all those activities that take place to convert from the old system to the new. The old system consists of a manual process, which operates differently from the proposed system. A proper implementation is essential to provide a reliable system to meet the requirement of the organization—the project developed by using PHP front end MYSQL as back end. The following MYSQL functions are used in this project for database related activities.

**1) MYSQL\_CONNECT()** query are used to make a connections to the sql database.

**2) MYSQL\_SELECT\_DB()** query are used to select a database.

**3) MYSQL\_QUERY()** query is used to create or delete a mysql databse.

**4) MYSQL\_CLOSE()** query are used to close the database connections.

**5) MYSQL\_FETCH\_ARRAY()** can be used to fetch all the selected data. This

function return row as an associative array, a numeric array, or both.

**6) MYSQL\_FETCH\_ASSOC()** which return the row as an associative array.

**SYSTEM TESTING**

**Testing Methods**

Testing is a series of different tests that whose primary purpose is to fully exercise the computer based system. Although each test has a different purpose, all work should verify that all system element have been properly integrated and performed allocated function. Testing is the process of checking whether the developed system works according to the actual requirement and objectives of the system. The philosophy behind testing is to find the errors. A good test is one that has a high probability of finding an undiscovered error. A successful test is one that uncovers the undiscovered error. Test cases are devised with this purpose in mind. A test case is a set of data that the system will process as an input.

**Types of Testing:**

* **System testing**

After a system has been verified, it needs to be thoroughly tested to ensure that every component of the system is performing in accordance with the specific requirements and that it is operating as it should including when the wrong functions are requested or the wrong data is introduced.  Testing measures consist of developing a set of test criteria either for the entire system or for specific hardware, software and communications components. For an important and sensitive system such as an electronic voting system, a structured system testing program may be established to ensure that all aspects of the system are thoroughly tested.

 Testing measures that could be followed include:

* Applying functional tests to determine whether the test criteria have been met
* Applying qualitative assessments to determine whether the test criteria have been met.
* Conducting tests in “laboratory” conditions and conducting tests in a variety of “real life” conditions.
* Conducting tests over an extended period of time to ensure systems can perform consistently.
* Conducting “load tests”, simulating as close as possible likely conditions while using or exceeding the amounts of data that can be expected to be handled in an actual situation.

Test measures for hardware may include:

* Applying “non-operating” tests to ensure that equipment can stand up to expected levels of physical handling.
* Testing “hard wired” code in hardware (firmware) to ensure its logical correctness and that appropriate standards are followed.
* Tests for software components also include:
* Testing all programs to ensure its logical correctness and that appropriate design, development and implementation standards have been followed.
* Conducting “load tests”, simulating as close as possible a variety of “real life” conditions using or exceeding the amounts of data that could be expected in an actual situation.
* Verifying that integrity of data is maintained throughout its required manipulation.

**Unit Testing**

The first test in the development process is the unit test. The source code is normally divided into modules, which in turn are divided into smaller units called units. These units have specific behavior. The test done on these units of code is called unit test. Unit test depends upon the language on which the project is developed. Unit tests ensure that each unique path of the project performs accurately to the documented specifications and contains clearly defined inputs and expected results. Functional and reliability testing in an Engineering environment. Producing tests for the behavior of components (nodes and vertices) of a product to ensure their correct behavior prior to system integration.

**Integration Testing**

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

**Validation Testing**

Valid and invalid data should be created and the program should be made to process this data to catch errors. When the user of each module wants to enter into the page by the login page using the use rid and password .If the user gives the wrong password or use rid then the information is provided to the user like “you must enter user id and password”. Here the inputs given by the user are validated. That is password validation, format of date are correct, textbox validation. Changes that need to be done after result of this testing.

**7. CONCLUSION**

User no needs to go anywhere to see the lawyer. They can directly see information on site. The Client can easily search the records of any lawyer through the system based on the location. Lawyer can easily get the information about the client and case details. In this project to help the user to book the particular lawyer on time and one more advantage of this project is online payment. This System will provide information about each attorney working for the firm. This system helps to look at each attorney’s educational background and work history. It’s also a good idea to find attorney familiar with the courts and laws of the area where you live.

**FUTURE ENHANCEMENT**

In future we can develop this project in android application with extra features like all case process like user case history also stored by user.

**8. BIBLIOGRAPHY**

**BOOK REFERENCE**

* Van Rossum, Guido, and Fred L. Drake. The python language reference manual. Network Theory Ltd., 2011.
* Van Rossum, Guido, and Fred L. Drake. The python language reference manual. Network Theory Ltd., 2011.
* Dierbach, Charles. Introduction to Computer Science using Python: A Computational Problem-Solving Focus. Wiley Publishing, 2012.
* James, Mike. Programmer's Python: Everything is an Object Something Completely Different. I/O Press, 2018.
* Reges, Stuart, Marty Stepp, and Allison Obourn. Building Python Programs. Pearson, 2018.

**WEB REFERENCES**

* <https://www.programiz.com/python-programming>
* <https://www.learnpython.org/>
* <https://docs.python.org/3/tutorial/>
* <https://www.w3schools.com/python/>

**9. APPENDIX**

**SOURCE CODE**

#from Scripts.bottle import request

from flask import Flask, render\_template, flash, request,session,send\_file

from wtforms import Form, TextField, TextAreaField, validators, StringField, SubmitField

from werkzeug import secure\_filename

import mysql.connector

#import string

#import hashlib

#import base64

#import os

#from io import BytesIO

#import datetime

import tkinter.messagebox

#import os, shutil

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

app.config.from\_object(\_\_name\_\_)

app.config['SECRET\_KEY'] = '7d441f27d441f27567d441f2b6176a'

class ReusableForm(Form):

name = TextField('Name:', validators=[validators.required()])

@app.route("/")

def home():

return render\_template('index.html')

@app.route("/admin")

def admin():

return render\_template('admin.html')

@app.route("/lawyer")

def lawyer():

return render\_template('lawyer.html')

@app.route("/user")

def user():

return render\_template('user.html')

@app.route("/acceptlawyer")

def acceptlawyer():

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM lreg where status='0'")

data = cur.fetchall()

#return render\_template('lawyerdetails.html', data=data)

return render\_template('acceptlawyer.html',data=data)

@app.route("/court")

def court():

return render\_template('court.html')

@app.route("/lreg")

def lreg():

return render\_template('lreg.html')

@app.route("/ureg")

def ureg():

return render\_template('ureg.html')

@app.route("/search1")

def search1():

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM categories")

data = cur.fetchall()

return render\_template('userhome1.html', data=data)

#return render\_template('userhome1.html')

#-----------------------------------------------------------Admin Log Code----------------------------------------

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

def adminlog():

error = None

if request.method == 'POST':

if request.form['uname'] == 'admin' or request.form['password'] == 'admin':

error = 'Invalid Credentials. Please try again.'

return render\_template('adminhome.html')

#return render\_template('adminhome.html', error=error)

else:

return render\_template('index.html')

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

def addcat():

error = None

if request.method == 'POST':

slno = request.form['slno']

categories = request.form['categories']

cno = request.form['cno']

details = request.form['details']

conn = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

cursor = conn.cursor()

cursor.execute(

"INSERT INTO categories VALUES ('','" + slno + "','" + categories + "','"+cno+"','"+details+"')")

conn.commit()

conn.close()

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM categories")

data = cur.fetchall()

return render\_template('adminhome.html',data=data)

@app.route("/adminhome")

def adminhome():

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM categories")

data = cur.fetchall()

return render\_template('adminhome.html',data=data)

@app.route("/lawyerdetails")

def lawyerdetails():

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM lreg")

data = cur.fetchall()

return render\_template('lawyerdetails.html', data=data)

@app.route("/userdetails")

def userdetails():

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM user")

data = cur.fetchall()

return render\_template('userdetails.html', data=data)

#----------------------------------------------------------------------end admin-----------------

#----------------------------------------------------------------------lawyer module-------------------------------------

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

def lawyerregister():

error = None

if request.method == 'POST':

name = request.form['name']

gender = request.form['gender']

age = request.form['age']

qualification = request.form['qualification']

address=request.form['address']

city = request.form['city']

pnumber = request.form['pnumber']

email = request.form['email']

categories = request.form['specialist']

uname = request.form['uname']

password = request.form['password']

f = request.files['file']

f1 = request.files['file1']

f.save("static/images/" + secure\_filename(f.filename))

f1.save("static/images/" + secure\_filename(f1.filename))

conn = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

cursor = conn.cursor()

cursor.execute(

"INSERT INTO lreg VALUES ('','" + name + "','" + gender + "','"+age+"','"+qualification+"','" + address + "','" + city + "','" + pnumber + "','" + email + "','" + categories + "','" + f.filename + "','"+f1.filename+"','" + uname+ "','" + password + "','0')")

conn.commit()

conn.close()

return "Register Success"

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

def lawyerlogin():

error = None

if request.method == 'POST':

uname = request.form['uname']

password = request.form['password']

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM lreg where uname='"+uname+"' and password='"+password+"'")

data = cur.fetchone()

if data is None:

return 'Username or Password is wrong'

else:

cur1 = conn1.cursor()

cur1.execute("SELECT \* FROM lreg where uname='" + uname + "' and password='" + password + "'")

data1 = cur1.fetchall()

for item in data1:

session['lname'] = item[0]

lid=item[0]

print(lid)

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

cur11 = conn1.cursor()

cur11.execute("SELECT \* FROM booking1 where lid='" + str(lid) + "' and status='0'")

data11 = cur11.fetchall()

return render\_template('lawyerhome.html',data=data11)

#return render\_template('lawyerhome.html', data=data)

@app.route("/lawyerhome")

def lawyerhome():

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

lid=session['lname']

print(lid)

cur11 = conn1.cursor()

cur11.execute("SELECT \* FROM booking1 where lid='" + str(lid) + "' and status='0'")

data11 = cur11.fetchall()

return render\_template('lawyerhome.html', data=data11)

@app.route("/lawyeruserdetails")

def lawyeruserdetails():

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

lid=session['lname']

cur11 = conn1.cursor()

cur11.execute("SELECT \* FROM booking1 where lid='" + str(lid) + "' and status='Accept'")

data4=cur11.fetchall()

return render\_template('lawyeruserdetails.html', data=data4)

@app.route("/status")

def status():

#categories=request.form['id']

id=request.args.get('id')

act = request.args.get('act')

conn = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

cursor = conn.cursor()

cursor.execute("update booking1 set status='"+str(act)+"' where id='"+str(id)+"'")

conn.commit()

conn.close()

return render\_template('lawyerhome.html')

@app.route("/status1")

def status1():

id = request.args.get('id')

act = request.args.get('act')

conn = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

cursor = conn.cursor()

cursor.execute("update booking1 set status='" + str(act) + "' where id='" + str(id) + "'")

conn.commit()

conn.close()

return render\_template('lawyerhome.html')

#-----------------------------------------------User-------------------------------------------------

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

def userregister():

error = None

if request.method == 'POST':

name = request.form['name']

gender = request.form['gender']

address=request.form['address']

city = request.form['city']

pnumber = request.form['pnumber']

email = request.form['email']

#categories = request.form['specialist']

uname = request.form['uname']

password = request.form['password']

f = request.files['file']

f.save("static/images/" + secure\_filename(f.filename))

conn = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

cursor = conn.cursor()

cursor.execute(

"INSERT INTO user VALUES ('','" + name + "','" + gender + "','" + address + "','" + pnumber + "','" + email + "','" + f.filename + "','" + uname+ "','" + password + "')")

conn.commit()

conn.close()

return "Register Success"

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

def userlogin():

error = None

if request.method == 'POST':

uname = request.form['uname']

password = request.form['password']

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM user where uname='"+uname+"' and password='"+password+"'")

data = cur.fetchone()

if data is None:

return 'Username or Password is wrong'

else:

cur1 = conn1.cursor()

cur1.execute("SELECT \* FROM user where uname='" + uname + "' and password='" + password + "'")

data1=cur1.fetchall()

for item in data1:

session['uname'] = item[0]

cur2 = conn1.cursor()

cur2.execute("SELECT \* FROM categories")

data2 = cur2.fetchall()

print(data2)

return render\_template('userhome.html',data=data2)

#return render\_template('lawyerhome.html', data=data)

@app.route("/userhome")

def userhome():

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM categories")

data = cur.fetchall()

return render\_template('userhome.html',data=data)

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

def search():

error = None

if request.method == 'POST':

categories=request.form['select']

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM lreg where categories='"+categories+"'")

data2 = cur.fetchall()

cur1 = conn1.cursor()

cur1.execute("SELECT \* FROM categories")

data = cur1.fetchall()

return render\_template('userhome1.html',data2=data2,data=data)

@app.route("/book")

def book():

#categories=request.form['id']

id=request.args.get('id')

lid=str(id)

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM lreg where id='"+lid+"'")

d1 = cur.fetchone()

print(d1)

lname=d1[1]

laddress=d1[5]

lphone=d1[7]

ldetails=d1[9]

print(id)

uid=session['uname']

print(uid)

conn11 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur1 = conn11.cursor()

cur1.execute("SELECT \* FROM user where id='" + str(uid) + "'")

d11 = cur1.fetchone()

name = d11[1]

address = d11[3]

phone = d11[4]

details = d11[5]

conn = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

cursor = conn.cursor()

cursor.execute(

"INSERT INTO booking1 VALUES ('','" + str(lid) + "','" + str(lname) + "','"+str(laddress)+"','"+str(lphone)+"','"+str(ldetails)+"','"+str(uid)+"','"+str(name)+"','"+str(address)+"','"+str(phone)+"','"+str(details)+"','0')")

conn.commit()

conn.close()

return render\_template('userhome.html')

@app.route("/lawyerbooking")

def lawyerbooking():

uid=session['uname']

print(uid)

conn = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

cursor = conn.cursor()

cursor.execute("select \* from booking1 where uid='"+str(uid)+"'")

data1 = cursor.fetchall()

return render\_template('ldetails.html',data1=data1)

@app.route("/lawyercasedetails")

def lawyercasedetails():

lid = session['lname']

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM lreg where id='" + str(lid) + "'")

data2 = cur.fetchall()

for item in data2:

name=item[1]

cur1 = conn1.cursor()

cur1.execute("SELECT \* FROM casehistory where lname='"+name+"'")

data = cur1.fetchall()

return render\_template('lawyercasedetails.html', data=data)

@app.route("/usercasedetails")

def usercasedetails():

lid = session['uname']

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM user where id='" + str(lid) + "'")

data2 = cur.fetchall()

for item in data2:

name=item[1]

cur1 = conn1.cursor()

cur1.execute("SELECT \* FROM casehistory where uname='"+name+"'")

data = cur1.fetchall()

return render\_template('usercasedetails.html', data=data)

#----------------------------------------------------------court---------------------------------

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

def courtlogin():

error = None

if request.method == 'POST':

if request.form['uname'] == 'admin' or request.form['password'] == 'admin':

error = 'Invalid Credentials. Please try again.'

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM lreg")

data2 = cur.fetchall()

cur1 = conn1.cursor()

cur1.execute("SELECT \* FROM user")

data = cur1.fetchall()

cur11 = conn1.cursor()

cur11.execute("SELECT \* FROM categories")

data1 = cur11.fetchall()

return render\_template('courthome.html',data=data2,data1=data,data2=data1)

#return render\_template('adminhome.html', error=error)

else:

return render\_template('index.html')

@app.route("/courthome")

def courthome():

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM lreg")

data2 = cur.fetchall()

cur1 = conn1.cursor()

cur1.execute("SELECT \* FROM user")

data = cur1.fetchall()

cur11 = conn1.cursor()

cur11.execute("SELECT \* FROM categories")

data1 = cur11.fetchall()

return render\_template('courthome.html',data=data2,data1=data,data2=data1)

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

def addcase():

error = None

if request.method == 'POST':

cno=request.form['cno'];

status = request.form['status'];

categories = request.form['categories'];

uname = request.form['uname'];

lname = request.form['lname'];

lname1 = request.form['lname1'];

bench = request.form['bench'];

hdate = request.form['hdate'];

ctitle = request.form['ctitle'];

history = request.form['history'];

conn = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

cursor = conn.cursor()

cursor.execute(

"INSERT INTO casehistory VALUES ('','" + cno + "','" + status + "','" + categories + "','" + uname + "','" + lname + "','" + lname1 + "','" + bench + "','" + hdate + "','" + ctitle + "','" + history + "')")

conn.commit()

conn.close()

return "Case Added Success"

@app.route("/update")

def update():

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM casehistory")

data2 = cur.fetchall()

return render\_template('update.html',data=data2)

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

def addupdate():

error = None

if request.method == 'POST':

cno=request.form['select'];

hdate = request.form['hdate'];

conn = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

cursor = conn.cursor()

cursor.execute(

"update casehistory set hdate='"+hdate+"' where cno='"+cno+"'")

conn.commit()

conn.close()

return render\_template('courthome.html')

@app.route("/view")

def view():

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM casehistory")

data2 = cur.fetchall()

return render\_template('view.html', data=data2)

@app.route("/view1", methods = ['GET', 'POST'])

def view1():

error = None

if request.method == 'POST':

cno=request.form['select'];

#hdate = request.form['hdate'];

conn = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

cur = conn.cursor()

cur.execute("SELECT \* FROM casehistory where cno='"+cno+"'")

data2 = cur.fetchall()

return render\_template('view.html', data1=data2)

@app.route("/payment")

def payment():

lid = session['uname']

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM user where id='" + str(lid) + "'")

data2 = cur.fetchall()

for item in data2:

name = item[1]

cur1 = conn1.cursor()

cur1.execute("SELECT \* FROM casehistory where uname='" + name + "'")

data = cur1.fetchall()

return render\_template('payment.html', data=data)

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

def pay():

error = None

if request.method == 'POST':

#categories=request.form['id']

lname=request.form['select']

accno = request.form['accno']

amount = request.form['amount']

conn = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

uid=session['uname']

print(uid)

cur = conn.cursor()

cur.execute("SELECT \* FROM user where id='" + str(uid) + "'")

data2 = cur.fetchall()

for item in data2:

name = item[1]

cursor = conn.cursor()

cursor.execute(

"INSERT INTO pay VALUES ('','" + lname + "','" + str(name) + "','"+accno+"','"+amount+"','Card Pay')")

conn.commit()

conn.close()

return render\_template('userhome.html')

@app.route("/payview")

def payview():

lid = session['uname']

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM user where id='" + str(lid) + "'")

data2 = cur.fetchall()

for item in data2:

name = item[1]

cur1 = conn1.cursor()

cur1.execute("SELECT \* FROM pay where uname='" + name + "'")

data = cur1.fetchall()

return render\_template('paymentview.html', data=data)

@app.route("/payview1")

def payview1():

lid = session['lname']

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM lreg where id='" + str(lid) + "'")

data2 = cur.fetchall()

for item in data2:

name = item[1]

cur1 = conn1.cursor()

cur1.execute("SELECT \* FROM pay where lname='" + name + "'")

data = cur1.fetchall()

return render\_template('paymentview1.html', data=data)

@app.route("/search2", methods=['GET','POST'])

def search2():

error = None

if request.method == 'POST':

categories=request.form['select']

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM categories where details='"+categories+"'")

data = cur.fetchall()

return render\_template('userhome.html',data=data)

@app.route("/acpt")

def acpt():

#categories=request.form['id']

id=request.args.get('id')

conn = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

cursor = conn.cursor()

cursor.execute("update lreg set status='1' where id='"+str(id)+"'")

conn.commit()

conn.close()

conn1 = mysql.connector.connect(user='root', password='', host='localhost', database='ccmanagement1')

# cursor = conn.cursor()

cur = conn1.cursor()

cur.execute("SELECT \* FROM lreg where status='1'")

data = cur.fetchall()

return render\_template('acceptlawyer.html',data=data)

@app.route("/dwnd")

def dwnd():

#categories=request.form['id']

id=request.args.get('name')

path = 'Static/images/' + id

return send\_file(path, as\_attachment=True)

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

app.run(debug=True, use\_reloader=True)

**SCREEN SHOTS**























