



Online Voting System Using Python and Django

Vempala Durga Maheshwari, Kalyani Ashwini, Byathala Balraj
and Monapati Sai Pavan Krishna

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

December 29, 2020

ONLINE VOTING SYSTEM USING PYTHON AND DJANGO

DURGA MAHESHWARI,KALYANI ASHWINI,BYATHALA BALRAJ,

M.SAI PAVAN KRISHNA

MALLA REDDY COLLEGE OF ENGINEERING

ABSTRACT:

Django poll application is a full featured polling application. This application Features are the user have to register in this application to show the polls and to vote. If user already voted, he cannot vote again.Only the owner of a poll can add poll, edit poll, update poll, delete poll, and choice, update choice, delete choice and end a poll. Ended poll only shows user the final result of

the poll. There is a search option for polls to search particular poll. User can filter polls by name, publish date, and by number of voted pagination will work even after applying filter.

INTRODUCTION:

We will create a pollster (voting system) web application using Django. This application will conduct a series of questions along with many choices. A user will be allowed to give



voting for that question by selecting a choice. Based on the answer the total votes will be calculated and it will be displayed to the user. Users can also check the result of the total votes for specific questions on the website directly. We will also build the admin part of this project. Admin user will be allowed to add questions and manage questions in the application

Pre-requisite: Knowledge of Python and basics of Django Framework. Python should be installed in the system. Visual studio code or any code editor to work on the application.

Technologies used in the project: Django framework and

SQLite database which comes by default with Django.

PYTHON:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** – You can actually sit at a Python prompt and



interact with the interpreter directly to write your programs.

- **Python is Object-Oriented** – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python:

Python was developed by Guido van Rossum in the late eighties and early nineties at

the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features:

Python's features include:

- **Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language



quickly.

- **Easy-to-read** – Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** – Python's source code is fairly easy-to-maintain.
- **A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can

add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

- **Databases** – Python provides interfaces to all major commercial databases.
- **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good



features, few are listed below –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

Local Environment

Setup:

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- Unix (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS
- BeOS
- Amiga
- VMS/OpenVMS



- QNX
- VxWorks
- Psion
- Python has also been ported to the Java and .NET virtual machines

Getting Python:

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of

Python <https://www.python.org/>

You can download Python documentation from <https://www.python.org/doc/>. The documentation is available in HTML, PDF, and PostScript formats.

Installing Python:

Python distribution is available for a wide variety of platforms.

You need to download only the binary code applicable for your platform and install Python.

If the binary code for your platform is not available, you need a C compiler to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features that you require in your installation.

Here is a quick overview of installing Python on various platforms

Windows Installation:

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to <https://www.python.org/downloads/>.
- Follow the link for the Windows installer *python-XYZ.msi* file where XYZ is the version you need to install.



- To use this installer *python-XYZ.msi*, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

Setting up PATH:

Programs and other executable files can be in many directories, so operating systems provide a search path that lists the directories that the OS searches for executables.

The path is stored in an

environment variable, which is a named string maintained by the operating system. This variable contains information available to the command shell and other programs.

The path variable is named as PATH in Unix or Path in Windows (Unix is case sensitive; Windows is not).

In Mac OS, the installer handles the path details. To invoke the Python interpreter from any particular directory, you must add the Python directory to your path.

Setting path at Windows:

To add the Python directory to the path for a particular session in Windows -

At the command prompt - type `path %path%;C:\Python` and press Enter.

Note - C:\Python is the path of the Python directory

IDLE[Integrated

Development Learning



Environment]:

Every Python installation comes with an **Integrated Development and Learning Environment**, which you'll see shortened to IDLE or even IDE. These are a class of applications that help you write code more efficiently. While there are many [IDEs](#) for you to choose from, Python IDLE is very bare-bones, which makes it the perfect tool for a beginning programmer.

Python IDLE comes included in Python installations on Windows and Mac. If you're a Linux user, then you should be able to find and download Python IDLE using your package manager. Once you've installed it, you can then use Python IDLE as an interactive interpreter or as a file editor.

An Interactive Interpreter:

The best place to experiment with Python code is in the [interactive interpreter](#), otherwise known as a **shell**. The shell is a basic [Read-Eval-Print Loop \(REPL\)](#). It reads a Python statement, evaluates the result of that statement, and then prints the result on the screen. Then, it loops back to read the next statement.

The Python shell is an excellent place to experiment with small code snippets. You can access it through the terminal or command line app on your machine. You can simplify your workflow with Python IDLE, which will immediately start a Python shell when you open it.

How to Use the Python IDLE Shell:

The shell is the default mode of operation for Python IDLE.

When you click on the icon to open the program, the shell is the first thing that you see:



```
Python 3.7.1 (v3.7.1:260ec2c36a, Oct
[Clang 6.0 (clang-600.0.57)] on darv
Type "help", "copyright", "credits"
>>> |
```

This is a blank Python interpreter window. You can use it to start interacting with Python immediately. You can test it out with a short line of code:

```
Python 3.7.1 (v3.7.1:260ec2c36a, Oct
[Clang 6.0 (clang-600.0.57)] on darv
Type "help", "copyright", "credits"
>>> print("Hello, from IDLE!")
Hello, from IDLE!
>>> |
```

Here, you used `print()` to output the string "Hello, from IDLE!" to your screen. This is the most basic way to interact with Python IDLE. You type in commands one at a time and Python responds with the result of each command.

GUI [graphical user

interface]:

A GUI (graphical user interface) is a system of interactive visual components for computer [software](#). A GUI displays objects that convey information, and represent actions that can be taken by the user. The objects change color, size, or visibility when the user interacts with them.

GUI objects include [icons](#), [cursors](#), and [buttons](#). These graphical elements are sometimes enhanced with sounds, or visual effects like [transparency](#) and [drop shadows](#).

A GUI is considered to be more [user-friendly](#) than a text-based [command-line interface](#), such as [MS-DOS](#), or the [shell](#) of [Unix-like](#) operating systems.

The GUI was first developed at [Xerox](#) [PARC](#) by [Alan](#)



[Kay, Douglas Engelbart](#), and a group of other researchers in [1981](#).

Later, [Apple](#) introduced the [Lisa computer](#) with a GUI on January 19, [1983](#).

How do you pronounce GUI

GUI is pronounced by saying each letter (*G-U-I*). It sometimes is also pronounced as "*gooey*."

How does a GUI work?

A GUI uses [windows](#), [icons](#), and [menus](#) to carry out commands, such as opening, deleting, and moving files. Although a GUI operating system is primarily navigated using a [mouse](#), a keyboard can also be used via [keyboard shortcuts](#) or the [arrow keys](#).

As an example, if you wanted to open a [program](#) on a GUI system, you would move the [mouse pointer](#) to the

program's [icon](#) and [double-click](#) it.

What are the benefits of GUI?

Unlike a [command-line operating system](#) or [CUI](#), like [Unix](#) or [MS-DOS](#), GUI operating systems are easier to learn and use because commands do not need to be memorized. Additionally, users do not need to know any [programming languages](#). Because of their ease of use and more modern appearance, GUI operating systems have come to dominate today's market.

What are examples of a GUI operating system?

- [Mac OS](#)

[i](#)

[C](#)



r
o
s
o
f
t
W
i
n
d
o
w
s

• A

p
p
l
e

S
y
s
t
e
m
7

a
n
d

m
a
c
O
S

• C

h



r
o
m
e
O
S

• L

i
n
u
x

v
a
r
i
a
n
t

s
l
i
k
e

U
b
u
n
t
u

u
s
i
n
g
a



G
U
I
i
n
t
e
r
f
a
c
e.

versions of [Linux](#) today have no GUI interface.

What are examples of a GUI interface?

1.

**G
N
O
M
E**

2. **K**

**D
E**

3. **A**

n
y
M
i
c
r
o
s
o
f
t
p
r
o
g
r
a
m,
i
n
c
l
u
d
i
n

Are all operating systems GUI?

No. Early command line operating systems like [MS-DOS](#) and even some



g
W
o
r
d,
E
x
c
e
l,
a
n
d
O
u
t
l
o
o
k.

4. I
n
t
e
r
n
e
t
b
r
o
w
s
e
r
s,
s
u
c
h
a

s
I
n
t
e
r
n
e
t
E
x
p
l
o
r
e,
C
h
r
o
m
e,
a
n
d
F
i
r
e
f
o
x.

How does the user interact with a GUI?

A pointing device, such as the mouse, is used to interact with nearly all aspects of the GUI. More modern (and



mobile) devices also utilize a [touch screen](#). However, as stated in previous sections, it is also possible to navigate a GUI using a [keyboard](#).

Does a GUI require a mouse?

No. Nearly all GUI interfaces, including Microsoft Windows, have options for navigating the interface with a keyboard only.

GUI WITH TKINTER:

Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below.

- **Tkinter** – Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would

look this option in this chapter.

- **wxPython** – This is an open-source Python interface for wxWindows <http://wxpython.org>.
- **JPython** – JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine <http://www.jython.org>

There are many other interfaces available, which you can find them on the net.

Tkinter Programming:

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.



Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Example:

```
#!/usr/bin/python
import Tkinter
top = Tkinter.Tk()
# Code to add widgets will go
here...
top.mainloop()
```

Tkinter Widgets:

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

SQLITE3:

SQLite is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It's also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle.

The `sqlite3` module was written by Gerhard Häring. It provides a SQL interface compliant with the DB-API 2.0 specification described by [PEP 249](#).

Module functions and constants:

`sqlite3.version`

The version number of this module, as a string. This is not the version of



the SQLite library.

`sqlite3.version_info`

The version number of this module, as a tuple of integers. This is not the version of the SQLite library.

`sqlite3.sqlite_version`

The version number of the run-time SQLite library, as a string.

`sqlite3.sqlite_version_info`

The version number of the run-time SQLite library, as a tuple of integers.

`sqlite3.PARSE_DECLTYPES`

This constant is meant to be used with the *detect_types* parameter of the `connect()` function.

Setting it makes the `sqlite3` module parse the declared type for each column it returns. It will parse out the first word of the declared type, i. e. for “integer primary key”, it will parse out

“integer”, or for “number(10)” it will parse out “number”. Then for that column, it will look into the converters dictionary and use the converter function registered for that type there.

`sqlite3.PARSE_COLNAMES`

This constant is meant to be used with the *detect_types* parameter of the `connect()` function.

Setting this makes the SQLite interface parse the column name for each column it returns. It will look for a string formed [mytype] in there, and then decide that ‘mytype’ is the type of the column. It will try to find an entry of ‘mytype’ in the converters dictionary and then use the converter



function found there to return the value. The column name found in `Cursor.description` does not include the type, i.e. if you use something like 'as "Expiration date [datetime]" in your SQL, then we will parse out everything until the first '[' for the column name and strip the preceding space: the column name would simply be "Expiration date".

```
sqlite3.connect(database[, timeout, detect_types, isolation_level, check_same_thread, factory, cached_statements, uri])
```

Opens a connection to the SQLite database file *database*. By default returns a `Connection` object, unless a

custom *factory* is given.

database is a `path-like object` giving the path name (absolute or relative to the current working directory) of the database file to be opened. You can use `":memory:"` to open a database connection to a database that resides in RAM instead of on disk.

When a database is accessed by multiple connections, and one of the processes modifies the database, the SQLite database is locked until that transaction is committed.

The *timeout* parameter specifies how long the connection should wait for the lock to go away until raising an exception. The default for the timeout parameter is 5.0 (five seconds).



`sqlite3.register_converter(type name, callable)`

Registers a callable to convert a byte string from the database into a custom Python type. The callable will be invoked for all database values that are of the type *type name*. Confer the parameter *detect_types* of the `connect()` function for how the type detection works. Note that *type name* and the name of the type in your query are matched in case-insensitive manner.

`sqlite3.register_adapter(type, callable)`

Registers a callable to convert the custom Python type *type* into one of SQLite's supported types. The callable *callable* accepts as single parameter the Python value, and must

return a value of the following types: int, float, str or bytes.

`sqlite3.complete_statement(SQL)`

Returns `True` if the string SQL contains one or more complete SQL statements terminated by semicolons. It does not verify that the SQL is syntactically correct, only that there are no unclosed string literals and the statement is terminated by a semicolon.

SYSTEM REQUIREMENTS:

HARDWARE REQUIREMENTS:

- ▶ System : Pentium Dual Core.
- ▶ Hard Disk : 500 GB.
- ▶ Monitor : 15" LED



- ▶ Input Devices : total no.of
Keyboard, Mouse
- ▶ Ram : votes of every party. There is a
1GB. DATABASE which is maintained
by the

SOFTWARE REQUIREMENTS:

- ▶ Operating system : with complete information is
Windows 7. stored. In this user who is above
18 year's
- ▶ Coding Language : register his/her information on
Python the database and when he/she
want to vote
- ▶ Framework : he/she has to login by his id
Django Framework and password and can vote to
any party only
- ▶ Tool : single time. Voting details
python IDLE, store in database and the result
Pycharm is displayed by
- ▶ Database : calculation. By online voting
SQLITE3 system percentage of voting is
increases. It

CONCLUSION:

This Online Voting system will manage the Voter's

information by which voter can login and use his voting rights. The

system will incorporate all features of Voting system. It provides the tools for maintaining voters vote to every party and it count

is vary less time consuming. It is very easy to debug.

