User Identification System

Release 0.0.91

Ayaan Imran

Jan 04, 2023

TABLE OF CONTENTS:

User Identifier System is a tool you can use to create a powerful login and signup system. This package contains many helpful tools in order to create a robust login and signup system. It encrypts the user's important crendentials and stores them in a database.

CHAPTER

ONE

BASIC LOGIN AND SIGNUP SYSTEM

The `UserIdentificationSystem` package has a class that allows users to register, login, and signup using a username and 1 password.

This class is called `Basic()`

1.1 Parameters `Basic()`

Parameter	Default value	Description	Data Type
filename	REQUIRED PARAMETER	The name of the database the user's credentials will be stored. The Basic() class will automatically create and initialise a new database with this name.	string
log	False	If set to True, the system will automatically log user's actions to a "log.txt" file.	boolean

1.2 Methods `Basic()`

1.2.1 Simplified version

Method	Short description
get_log()	This function gets all the logs recorded by the system.
get_usernames()	This function gets all the usernames that have been registered in the system.
username_is_valid()	This function is to check if a username is valid or not.
signup()	This function will register a new user in the database.
login()	This function will compare the username and password with the system.
deluser()	This function will delete a user from the system.
secure()	This function is vital to be present at the end of your code to securely close the database connection.

1.2.2 get_log()

If the `self.log` class variable is set to true during the setup, a `log.txt` file will automatically be generated along with the database file. *This can also be done optionally using the create_log() method.*

Returns:
This method returns a list that contains each log (represented as a single dictionary).

```
controller = Basic("mydatabase.db", log=True) # Log must be True
logs = controller.get_log() # Returns the logs from the log file.
```

1.2.3 get_usernames()

The `get_usernames()` method is a useful feature that provides you with all the registered and valid users in the system.

Returns:
This method returns a list containing all the usernames registered in the system.

```
controller = Basic("mydatabase.db")
log = controller.get_usernames() # Returns a list with all the registered usernames.
```

1.2.4 username_is_valid()

The `username_is_valid()` method checks whether or not a username is valid. The username is valid if it is not already taken by another user.

Parameter	Data type
username	String

Returns:
Returns a boolean depending on if the username provided is valid or not.

```
controller = Basic("mydatabase.db")

# Returns a boolean depending on if the username is valid or not.
log = controller.username_is_valid("uis learner")
```

1.2.5 signup()

The `signup()` method allows you to register users into the system.

Parameter	Description	Data type
username	A unique name the user will be represented by.	String
password	The authentication key the user has selected.	String

Returns:
This method returns a boolean depending on if the process was executed successfully or not.

If this method returns a `False`, it could occur because:

1. The username was invalid (It was already in use by another user).
2. An error occurred with the database (Rare case).

```
controller = Basic("mydatabase.db", log=True) # Log is optional

# If log is set to True...
# ...it will automatically log a signup statement (both if it failed or if it was␣
↪successful.
success = controller.signup("uis learner", "password123") # Will return True if process␣
↪was successful
```

1.2.6 login()

The `login()` method compares the user's credentials with the credentials stored in the database.

Parameter	Description	Data type
username	The unique name the user chose while signing up.	String
password	The password assigned to that specific username.	String

Returns:

This method returns a boolean depending on the validity of the user's credentials.

If this method returns `False`, it could occur because of the following reasons:

1. The username passed in does not exist.
2. The user's credentials are invalid.
3. There is an error with the database (Rare case).

```
controller = Basic("mydatabase.db", log=True) # Log is optional

# If log is set to True...
# ...it will automatically log a login statement (both if it failed or if it was␣
↪successful).
success = controller.login("uis_learner", "password123")
```

1.2.7 deluser()

The `deluser()` method deletes a user from the database, after confirming the validity of the user.

Parameter	Description	Data type
username	The unique name the user chose while signing up.	String
password	The password assigned to that specific username.	String

Returns:

This method returns a boolean depending on the validity of the user's credentials.

If this method returns `False`, it could occur because of the following reasons:

1. The username passed in does not exist.
2. The user's credentials are invalid.
3. There is an error with the database (Rare case).

Note: If `log=True` during the setup of the system, then a delete statement along with a login statement will be logged into the `log.txt` file.

```
controller = Basic("mydatabase.md", log=True) # Log is optional
success = controller.deluser("uis learner", "password123")
```

1.2.8 secure()

The `secure()` method is essential to be present at the end of your code. It is responsible to close the connection of the databse. If the databse is not closed, it stays open until it goes out of scope.

Returns:
This method returns `True` if the database was successfully closed.

```
controller = Basic("mydatabase.db") # This method does not log
# login(), signup(), deluser(), get_usernames(), username_is_valid()...
# ...can be called here.
controller.secure()
```

1.3 Class variables `Basic()`

1.3.1 Simple overview

Class variable	Description	Data type
self.log	Configuration for `log` at setup of the system.	Boolean
self.filename	The filename of the database.	String
self.username	The latest username used in the system.	String

1.3.2 self.username

The `username` class variable contains the latest username used in the system. The `username` is updated in the following cases:

1. If the `signup()` method returns `True`.
2. If the username passed in the `login()` method is valid.
3. If the username passed in the `deluser()` method is valid.

CHAPTER

TWO

EXTRA LOGIN AND SIGNUP SYSTEM

The `UserIdentificationSystem` package has a class that allows users to register, login, and signup using a username and 2 passwords.

This class is called `ExtraPass()`

2.1 Parameters `ExtraPass()`

Parameter	Default value	Description	Data Type
filename	REQUIRED PARAMETER	The name of the database the user's credentials will be stored. The Basic() class will automatically create and initialise a new database with this name.	string
log	False	If set to True, the system will automatically log user's actions to a "log.txt" file.	boolean

2.2 Methods `ExtraPass()`

2.2.1 Simplified version

Method	Short description
get_log()	This function gets all the logs recorded by the system.
get_usernames()	This function gets all the usernames that have been registered in the system.
username_is_valid()	This function is to check if a username is valid or not.
signup()	This function will register a new user in the database.
login()	This function will compare the username and password with the system.
deluser()	This function will delete a user from the system.
secure()	This function is vital to be present at the end of your code to securely close the database connection.

2.2.2 get_log()

If the `self.log` class variable is set to true during the setup, a `log.txt` file will automatically be generated along with the database file. *This can also be done optionally using the create_log() method.*

Returns:
This method returns a list that contains each log (represented as a single dictionary).

```
controller = ExtraPass("mydatabase.db", log=True) # Log must be True
logs = controller.get_log() # Returns the logs from the log file.
```

2.2.3 get_usernames()

The `get_usernames()` method is a useful feature that provides you with all the registered and valid users in the system.

Returns:
This method returns a list containing all the usernames registered in the system.

```
controller = ExtraPass("mydatabase.db")
log = controller.get_usernames() # Returns a list with all the registered usernames.
```

2.2.4 username_is_valid()

The `username_is_valid()` method checks whether or not a username is valid. The username is valid if it is not already taken by another user.

Parameter	Data type
username	String

Returns:
Returns a boolean depending on if the username provided is valid or not.

```
controller = ExtraPass("mydatabase.db")

# Returns a boolean depending on if the username is valid or not.
log = controller.username_is_valid("uis learner")
```

2.2.5 signup()

The `signup()` method allows you to register users into the system.

Parameter	Description | Data type	
username	A unique name the user will be represented by. | String	
password	The authentication key the user has selected.	String
extra	The second authentication key the user has selected	String

Returns:
This method returns a boolean depending on if the process was executed successfully or not.

If this method returns a `False`, it could occur because:

1. The username was invalid (It was already in use by another user).

2. An error occurred with the database (Rare case).

```
controller = ExtraPass("mydatabase.db", log=True) # Log is optional

# If log is set to True...
# ...it will automatically log a signup statement (both if it failed or if it was␣
↪successful.
success = controller.signup("uis learner", "password123", "extra123") # Will return True␣
↪if process was successful
```

2.2.6 login()

The `login()` method compares the user's credentials with the credentials stored in the database.

Parameter	Description	Data type
username	The unique name the user chose while signing up.	String
password	The password assigned to that specific username.	String
extra	The second password to that specific username	String

Returns:

This method returns a boolean depending on the validity of the user's credentials.

If this method returns `False`, it could occur because of the following reasons:

1. The username passed in does not exist.
2. The user's credentials are invalid.
3. There is an error with the database (Rare case).

```
controller = ExtraPass("mydatabase.db", log=True) # Log is optional

# If log is set to True...
# ...it will automatically log a login statement (both if it failed or if it was␣
↪successful).
success = controller.login("uis_learner", "password123", "extra123")
```

2.2.7 deluser()

The `deluser()` method deletes a user from the database, after confirming the validity of the user.

Parameter	Description	Data type
username	The unique name the user chose while signing up.	String
password	The password assigned to that specific username.	String
extra	The second password to that specific username	String

Returns:

This method returns a boolean depending on the validity of the user's credentials.

If this method returns `False`, it could occur because of the following reasons:

1. The username passed in does not exist.
2. The user's credentials are invalid.

3. There is an error with the database (Rare case).

Note: If `log=True` during the setup of the system, then a delete statement along with a login statement will be logged into the `log.txt` file.

```
controller = ExtraPass("mydatabase.md", log=True) # Log is optional
success = controller.deluser("uis learner", "password123", "extra123")
```

2.2.8 secure()

The `secure()` method is essential to be present at the end of your code. It is responsible to close the connection of the databse. If the databse is not closed, it stays open until it goes out of scope.

Returns:
This method returns `True` if the database was successfully closed.

```
controller = ExtraPass("mydatabase.db") # This method does not log
# login(), signup(), deluser(), get_usernames(), username_is_valid()...
# ...can be called here.
controller.secure()
```

2.3 Class variables `ExtraPass()`

2.3.1 Simple overview

Class variable	Description	Data type
self.log	Configuration for `log` at setup of the system.	Boolean
self.filename	The filename of the database.	String
self.username	The latest username used in the system.	String

2.3.2 self.username

The `username` class variable contains the latest username used in the system. The `username` is updated in the following cases:

1. If the `signup()` method returns `True`.
2. If the username passed in the `login()` method is valid.
3. If the username passed in the `deluser()` method is valid.

CHAPTER

THREE

INSTALLATION

Note: This package requires a python version of >= 3.6.

Follow this link to view the pypi repository: https://pypi.org/project/user-Identification-System/

The package also required a module that handles and encrypts passwords ==> *pypasstools*[1]. See its pypi repository: https://pypi.org/project/pypasstools/

Use the following command to install the package along with its dependencies:

```
pip install user-Identification-System
pip install pypasstools
```

To import `user-Identification-System` in your python code, use the following code-snippet:

```
import UserIdentificationSystem as uis
```

[1] If the module `pypasstools` is not installed, the program will be end, displaying an error message that instructs the user to install pypasstools.

CHAPTER

FOUR

VERSION DETAILS

Current version: 0.0.91

1. *autotask* feature was removed.
2. System can now log user's actions and save them in a *log.txt* file.
3. Contains important bugs and error fixes.

CHAPTER

FIVE

LINKS AND REFERENCES

- Github
- Pypi repository
- Pypasstools Pypi repo
- miskiacuberayaan2509@gmail.com