Fireproof

One password to rule them all

Annie Kelly, Tyler Tafoya, Kara James, Tori Augustine

AESCipher module

class `AESCipher.AESCipher`

static `decryptCredentials`(*key*, *iv*, *text*)

Creates an AES128 decryption suite that includes the symmetric key and iv for the particular master
account. We use the decryption suite to decrypt the given string of text.

Parameters:
- **key** – symmetric key used in the AES128 suite
- **iv** – random 16 bit iv used in the AES128 suite
- **text** – an encrypted string that is a multiple of 16 bits

Returns: returns the decrypted text

static `encryptCredentials`(*key*, *iv*, *text*)

Creates an AES128 encryption suite that includes a symmetric key and an initialization vector (iv).
The key is created by hashing the master password using sha256 and the iv is created using os.random(16). Using the encryption suite we can encrypt the given string of text.

Parameters:
- **key** – symmetric key used in the AES128 suite
- **iv** – random 16 bit iv used in the AES128 suite
- **text** – a string of text

Returns: returns an encrypted string of text that is a multiple of 16 bits

static `hashPassword`(*password*)

Hashes the given password string using sha256.

Parameters: **password** – a string of at least 8 characters

Returns: returns a hash of 256 bits

static `pad`(*text*)

Takes a string of text as input and pads it to make its length a multiple of 16

static `unpad`(*text*)

Takes padded text as input and strips the extra padding from it

FireproofProgram module

class `FireproofProgram.AddNewServicePage`(*parent*, *controller*)

Bases: `Tkinter.Frame`

class `FireproofProgram.CreateAccountPage`(*parent*, *controller*)

Bases: `Tkinter.Frame`

class `FireproofProgram.EditPage`(*parent*, *controller*)

Bases: `Tkinter.Frame`

class `FireproofProgram.Fireproof`(**args*, ***kwargs*)

Bases: `Tkinter.Tk`

`current_account` = *None*

`show_frame`(*c*)

This function changes frame (c) so it is visible to the user.

Parameters: **c** – Name of frame

class `FireproofProgram.LoginPage`(*parent*, *controller*)

Bases: `Tkinter.Frame`

class `FireproofProgram.RemoveServicePage`(*parent*, *controller*)

Bases: `Tkinter.Frame`

class `FireproofProgram.ServiceInfoPage`(*parent*, *controller*)

Bases: `Tkinter.Frame`

class `FireproofProgram.ServicesPage`(*parent*, *controller*)

Bases: `Tkinter.Frame`

class `FireproofProgram.SettingsPage`(*parent*, *controller*)

Bases: `Tkinter.Frame`

LoginFunctions module

class `LoginFunctions.LoginFunctions`

static `Login`(*master_username*, *master_password*)

Checks a user’s Username and Password against the master account database. The user enters his/her credentials which are checked against the database to see if a matching Username & Password combination exist. If a username or password is not provided, an error message appears asking the user to enter one. If the credentials don’t match any existing master accounts in the database, an error message asks the user to create an account, otherwise the user successfully logs into their Fireproof account.

Parameters:

- **master_username** – the main username used to login to Fireproof (checked against database)
- **master_password** – the main password used to login to Fireproof (checked against database)

Returns: returns the current account based on the username and password provided by the user.

`accounts` = *[]*

static `createLoginInfo`(*master_username*, *master_password*, *confirm_master_password*)

Creates a new master account in the database when a new user signs up for Fireproof. The Master Username and Password then uses the insertMasterAccount function to be inserted as a new set in the database of master account. Error messages handle malformed inputs for either username, password, or confirm password. Upon successful input of credentials, a new master account is created and the user returns to the Login function.

Parameters:

- **master_username** – the main username used to login to Fireproof (checked against database)
- **master_password** – the main password used to login to Fireproof (checked against database)

MasterAccount module

class `MasterAccount.MasterAccount`(*username*, *password*, *service_name_list=[]*)

static `changeMasterPassword`(*account*, *new_password*, *confirm_password*)

Decrypts and reencrypts everything in the database using the user’s new hashed password as the key

Parameters:
- **account** – MasterAccount object
- **new_password –** The new password the user chooses

`count` = *0*

note:: a global count of how many accounts we have in our database

`insertMasterAccount`()

Inserts the account’s encrypted username and password into our database

static `retrieveMasterAccountId`(*username_enc*, *password_enc*)

Retrieves the account’s primary id from our database

Parameters:
- **username_enc** – The account’s encrypted username
- **password_enc** – The account’s encrypted password

Returns: primary id

Return type: int

Service module

class `Service.Service`(*service_name*, *account_owner*, *service_accounts=[]*)

`ServiceCount` *= 1*

static `changeService`(*account*, *service*)

Allows the user to update their existing services

static `createServiceTable`()

Initializes the FireproofServices table in our database

static `insertServiceName`(*account*, *service*)

Inserts the encrypted Service into the database by matching it with the master account's primary id

Parameters: **account** – The master account who owns this service

static `retrieveServiceNameId`(*account*, *service*)

Retrieves the primary id of the service from the database

Parameters:	**account** – The master account who owns this service
Returns:	Primary id of the service
Return type:	int

ServiceAccount module

class `ServiceAccount.ServiceAccount`(*username*, *password*, *account_owner*)

static `createServiceAccountsTable`()

Initializes the FireproofServicesAccounts table in our database

static `insertServiceAccount`(*account*, *service*, *serviceaccount*)

Inserts the encrypted ServiceAccount into the database by matching it with the master account's and service account's primary id

Parameters:

- **account** – the master account who owns this service
- **service** – the service that this account is associated with