# DATABASE SYSTEMS
# (CS F212)
# ASSIGNMENT SUBMISSION

## TOPIC-18
## Password Manager and Vault

**Submitted By:**

**Atish**
**2020A7PS0107P**

**Swapnil Shivam**
**2020A7PS0040P**

## System Requirement Specification

This project is a password manager to store all usernames, passwords and memos for various platforms for any user. All the passwords are stored under AES encryption. Only after authentication, the password is accessible and can be directly copied to the clipboard. There is also an administrator account to reset the user password and decrypt all existing passwords.

Encryption workflow is as follows:

1. A secret key is generated if it does not exist. It is currently stored in the database itself, but can be stored in a secure location on the computer.
2. A master password is created by the admin user, which is encrypted using the secret key and stored in the database.
3. The login password for each user into the vault is encrypted using the master password and stored in the database.
4. The platform passwords for each user are encrypted using the login password for each user and stored in the database. This way, the administrator can decrypt the platform passwords for a particular user as they have access to login passwords of the user.

Functions would be required to encrypt and decrypt the various password types, and also to insert and delete the entries. Other functions will also be required to ensure consistency and non repetition of data.

This application stores all your passwords and important data securely and in an encrypted format, where internal operations are also not done on raw password data. This application supports a global decrypt implementation through admin and also supports multi-user authentication.

There will be tables required to store the masterpassword, secretkey, the relation between a username and their login passwords, and a final table to store the various data such as memo, platform name, platform account name and platform password.

## Function Descriptions

Encrypt: takes the input of the key and the data and encrypts the data according to the key and returns the encrypted data. The mode of encryption used is AES encryption.

Decrypt: takes the input of the key and the data and decrypts the passed data using the key. The mode decryption mode is complementary to the AES encryption.

Reset User Password function: is used to reset the password belonging to the username passed, all updates are done to the encrypted form, raw passwords are never used. Only the admin has this permission and this option is accessed through the admin page only. The function then also updates all the platform passwords belonging to the user.

Check Admin Login: authenticates the admin login by checking the admin password.

checkUserAlreadyExists: provides a check if already a user with the same username is accessing our application.

createAccount: this function takes in the username and the password and runs the query to fill the encryptedPassword table and inserts the user in our database as well as commit the changes.

addNewEntry: this function not only is used for adding an entry to the user's information vault but also provides a check if an entry exists with the same username and password. It also has validity checks for entries and if it passes all checks then it inserts in the vault table the various fields for the entry, else returns the corresponding error.

deleteEntry: this function takes in the parameters username, platform and account and deletes the entry from the vault table.

UpdatePassword: this function takes in the parameters username, platform and account and password and fetches the  entry and sets the new password to the passed password, in an encrypted form.

checkLogin: This function is used to check whether the login username and password are correct and returns true or false accordingly.

popUp: This function is used to generate popUp prompts for entries.

Launch: This function is used when the application is started for the first time, to create an administrator account.

signUp: This function is used to ask for username and password to create a new account.

loginScreen: This function displays the login screen with fields for username and password, as well as buttons to go to admin screen and signUp screen.

adminLogin: This function displays the administrator login screen.

adminDashboard: This function displays the administrator dashboard, with options to access the user vault or reset a user password.

AccessVault: This function is accessed by pressing the access user vault in the last function, and asks for username to access the vault.

ResetUser: This function is accessed by pressing the Reset User in the adminDashboard, and asks for username and password to which the user wants it to reset.

entryAddition: This function takes entries Platform Name, Platform Account Name, Platform Password and Memo to add into the table.

vaultScreen: This displays all the entries related to the user on the screen, with various buttons such as copy, delete, update, or store new entry.

## Normalization

| Username | User password | Platform username | Platform name | Platform password | Platform memo | Master password | Secret key |
|---|---|---|---|---|---|---|---|
| a | b | c | d | e | f | g | h |

G and H are the same for every single entry so in order to avoid redundancy G,H are extracted into another relational tables secretkey(H) and masterpassword(G).

| Username | User password | Platform username | Platform name | Platform password | Platform memo |
|---|---|---|---|---|---|
| a | b | c | d | e | f |

The functional dependencies in this new table are:

A->B
ACD->EF

Here ACD is a candidate key, and FD A->B violates the partial dependency rule necessary for 2NF as A is a prime attribute (part of candidate key) and B is a non-prime attribute.
So applying the 2NF decomposition algorithm we get 2 tables (ACDEF) (vault) and (AB)(encryptedPassword).

| Username | Platform username | Platform name | Platform password | Platform memo |
|---|---|---|---|---|
| a | c | d | e | f |

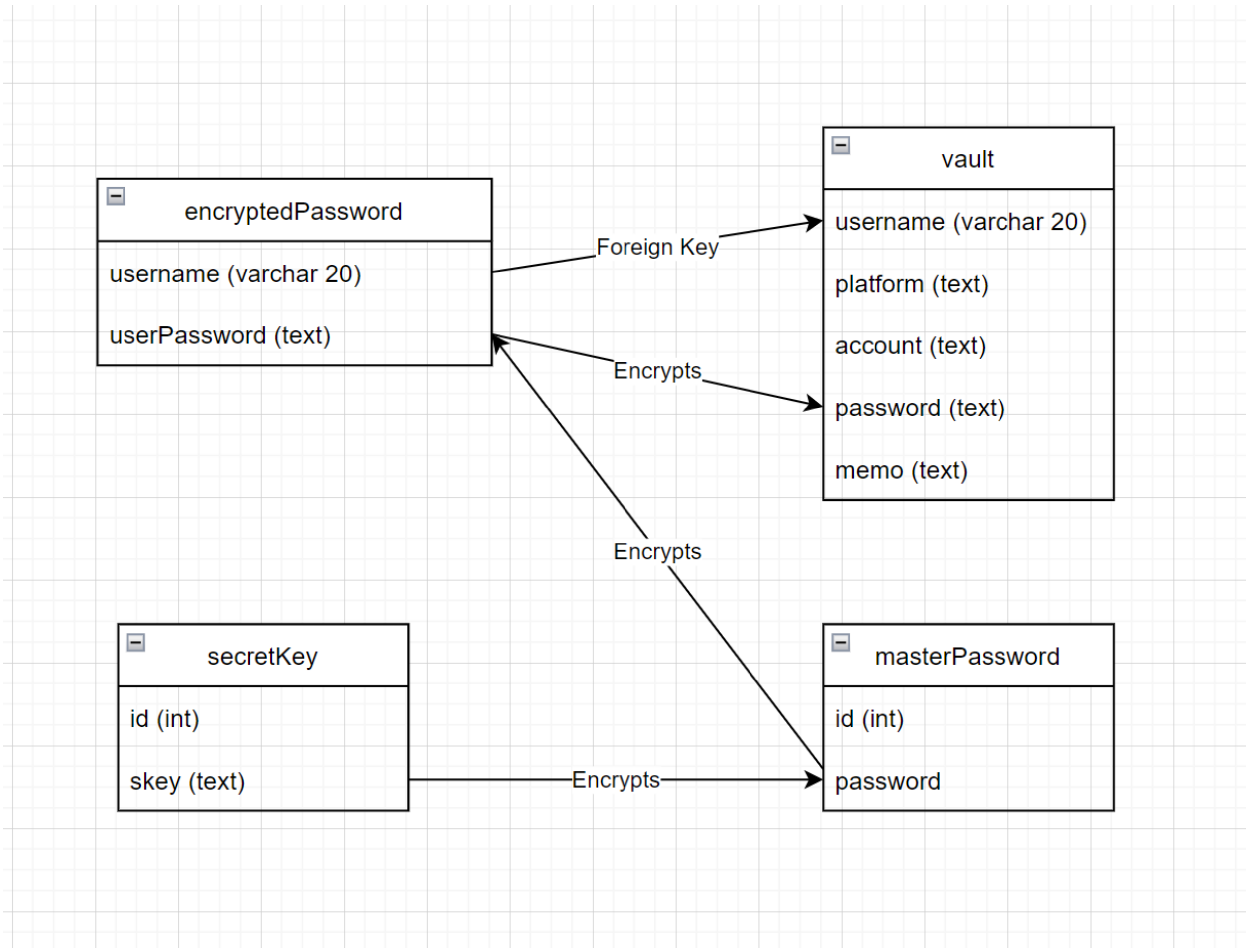
| Username | User Password |
|---|---|
| a | b |

This form is also in 3NF as no transitive dependency i.e.
Non Prime attribute->Non Prime attribute is absent.

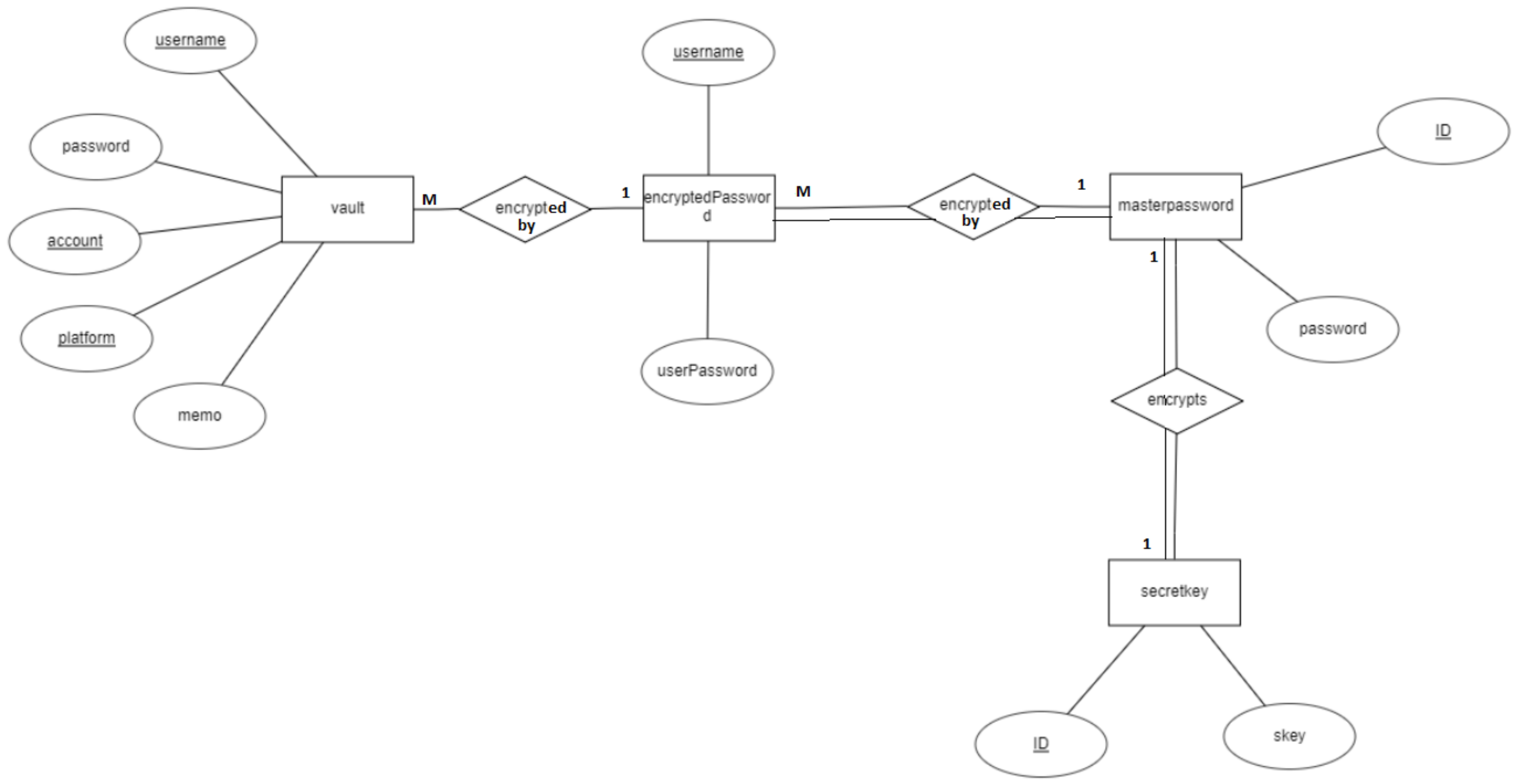
## Required Tables

1) **secretkey:** This table is being used to store the secretkey generated for encryption of the master password. This is not necessary to store in the database, and can be locally stored in a secure location on the PC itself.
2) **masterpassword:** This table is being used to store the masterpassword (or the admin password). It should consist of only one entry.
3) **encryptedpassword:** This table is used to store the username and password being used to login into the application. The username acts as the primary key in this table.
4) **vault:** This table is used to store the username, platform name, platform account, platform password (encrypted) and memo.

## Schema Design:



## ER Diagram



username
password
account
platform
memo
vault
M
encrypted by
1
username
encryptedPassword
userPassword
M
encrypted by
1
masterpassword
ID
password
1
encrypts
1
secretkey
ID
skey

## Walkthrough for I/O

### Setup:

1. Install MySQL Community 8.0.28.0.
2. In the given program, change the credentials in the MySQL connector code to your desired credentials. By default, the credentials are username=root and password=root.
3. Install the following python packages:
   a. pyaes
   b. mysql-connector-python

If using pip as python package manager, go into the folder and run
(pip install -r   requirements.txt)

### For using the program:

1. On the first startup, you will be prompted to create an admin account.
2. On the subsequent startups, you will be taken directly to the user login screen.
3. You can create new accounts using the new user button.
4. You can access the administrator dashboard by clicking the Admin button on the home page.
5. Users can sign in to their vault using their username and password.
6. In the vault screen, users can see their stored passwords, memos, usernames for a particular platform, as well as add new entries, update passwords, and have buttons to copy account name and passwords directly to the clipboard.
7. In the Administrator dashboard, you have options to reset a user password or to access their vault screen directly.

Note: The SQL file consists of a general test case for testing the functionality of the program. Otherwise, even if the database or tables do not exist, the python file handles the DDL statements for the database.

**Tested using:**
MySQL, mysql-connector-python==8.0.28
pyaes==1.6.1
protobuf==3.20.0 (automatically installed with mysql-connector-python)
Python 3.10.4
Windows 11 21H2

Link for GitHub Repository: https://github.com/swapshivam3/password-manager
Link for demo video: https://youtu.be/2OfDy_Q4NFk