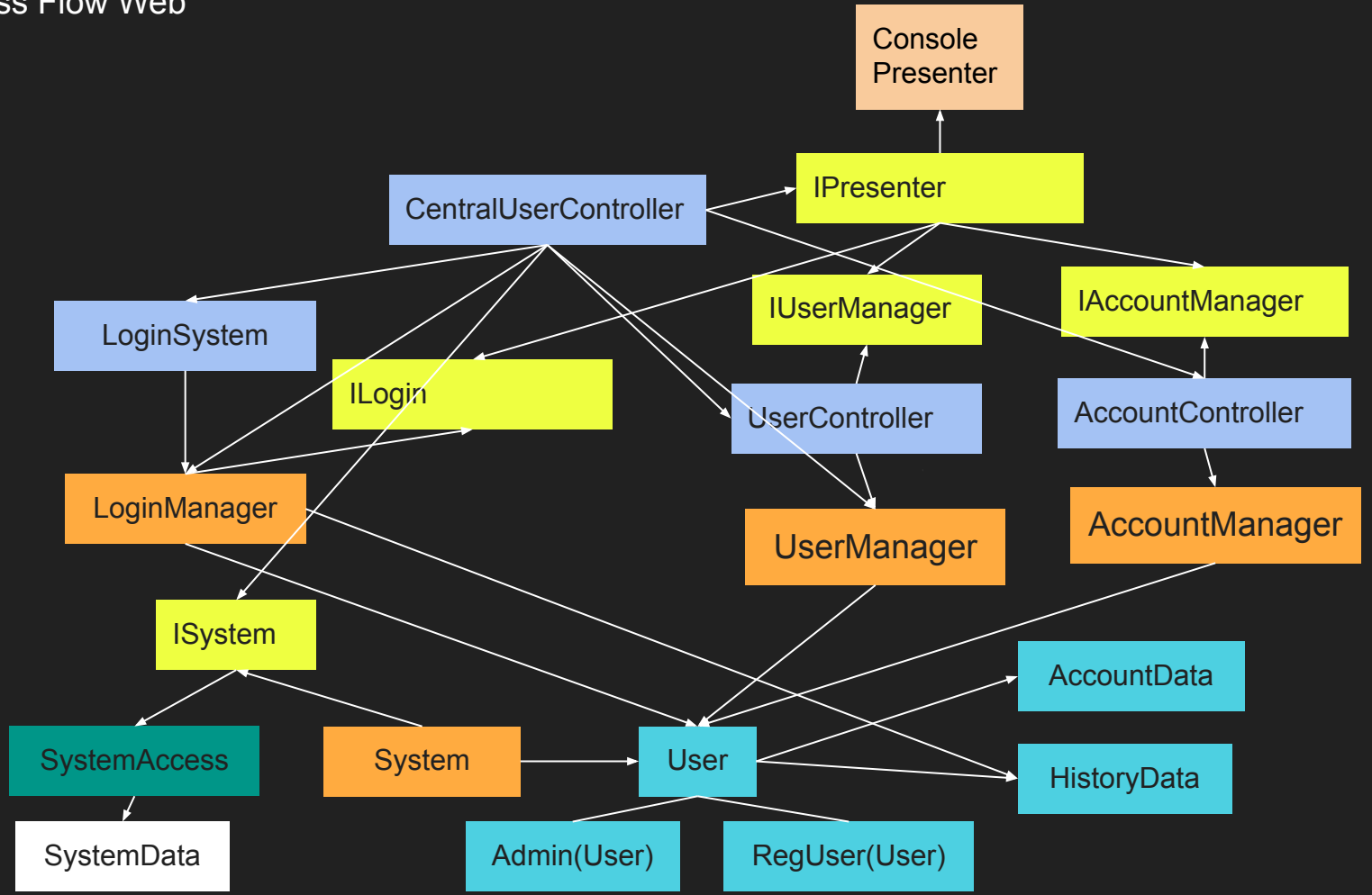
Task 1 Goals

1. User must be able to log-in and see account
2. Accounts must track previous login dates and times
3. New users can create accounts (no more than 1 account per user?)
4. Admin Users may ban and delete non-admin users. Also create new admin users
5. Implements enough code such that a user can log in and see the history of their account. At the moment, the only thing their history needs to track is their previous login times and dates.

Task 2 Goals

1. can compile and run
2. contains at least one unit test (ideally many more!)
3. implements enough code such that a user can log in and see the history of their account. At the moment, the only thing their history needs to track is their previous login times and dates.
4. allows new users to be able to create accounts
5. allows the admin user to be able to create new admin users and, if required, delete or temporarily ban non-admin users.
6. has no style warnings in IntelliJ — we recommend that you fix them as you find them. There is a difference between IntelliJ guessing what you are trying to do and making suggestions and IntelliJ complaining that you could improve something you already did. If you are unsure which is which, please ask your TA.

Class Flow Web

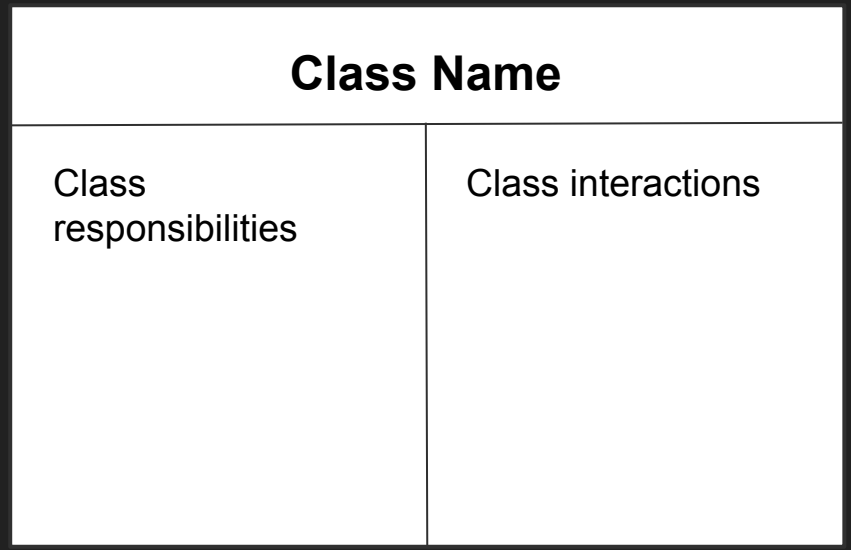


Legend

- Entity Class
- controller
- Use Case
- Gateway
- Presenters
- Database
- Interface

Card Assignment

Name	CRC Card Creation Assignment	Card Implementation Assignment
Angela Niu	LoginSystem, Login	User, RegUser, Admin
Nairi Kelian	UserController, AccountController	LoginSystem, Login
Ysha Santos	AccountManager, Presenter	UserController, AccountController
Eswar	User, RegUser, AdminUser	AccountManager
ij	UserManager, AccountManager, IUserManager, IAccountManager	GetAccountData, Account, HistoryData
Paulo Guerero	AccountData, System	UserManager, AccountManager, IUserManager, IAccountManager, ConsolePresenter, ILoginManager
Francisco	GetAccountData, Account, HistoryData	AccountData, system
Trevor Foote	Unresponsive	



Comment Here

Example CRC Card

Entities

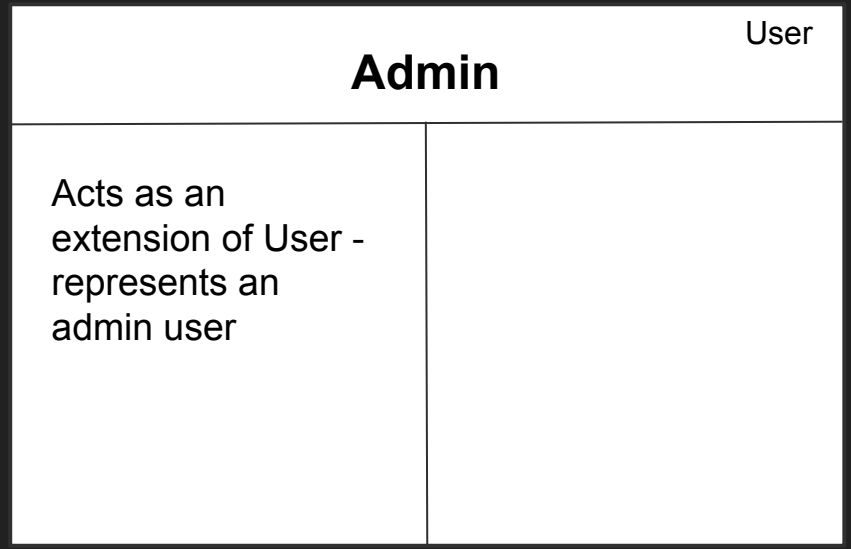
abstract

User

regUser, admin

Username: String Password: String UserID: int (reference to an account) Login status: Boolean TotalUsers: int ban status: Boolean accountData: AccountData historyData: HistoryData getAccountData() getHistoryData() getUsername getPassword setPassword getAccountId getLoginStatus setLoginStatus getBanStatus setBanStatus getHistoryData setHistoryData	accountData historyData

This is an abstract entity class which represents a program user. It contains all user data. It contains a link to historyData.



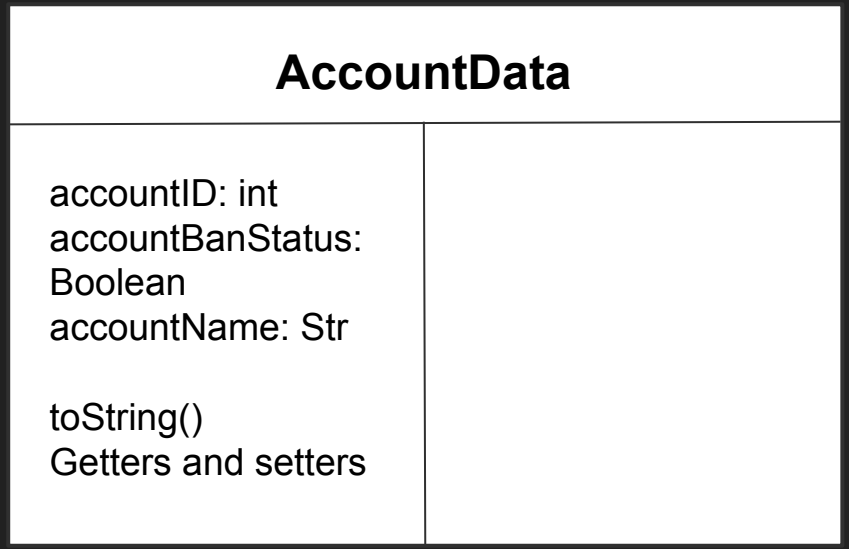
No extra attributes or methods that are not implemented by User for now. BanUser and DeleteUser are methods in the UserManager class.

User

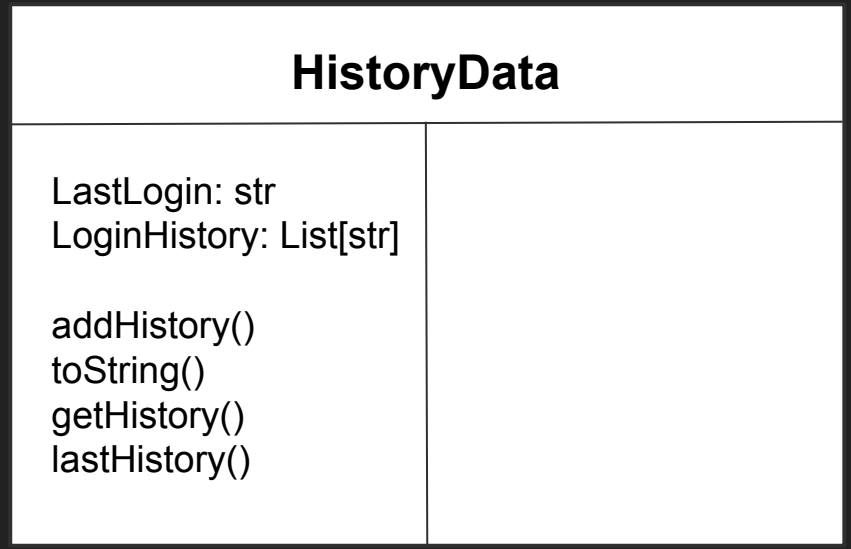
RegUser

Acts as an extension of User which represents a regular user

No extra attributes or methods that are not implemented by User for now. BanUser and DeleteUser are methods in the UserManager class.



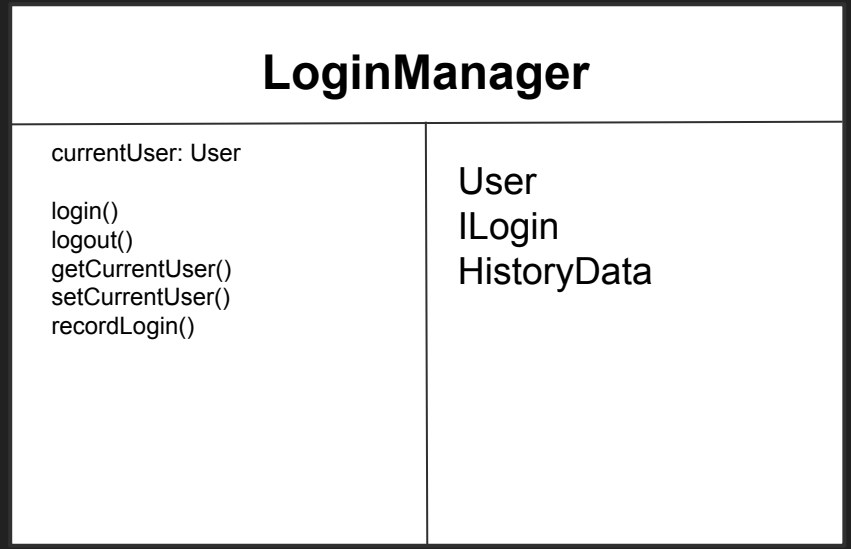
Entity that represents account data. It contains a unique id linking the account to a user which is set based on the quantity of accounts in the system on creation. Has a boolean which updates based on the account ban status.



An Entity class that stores the account's history information upon login to the database

For the future:
Store failed login attempt data

Use Case CRC Cards



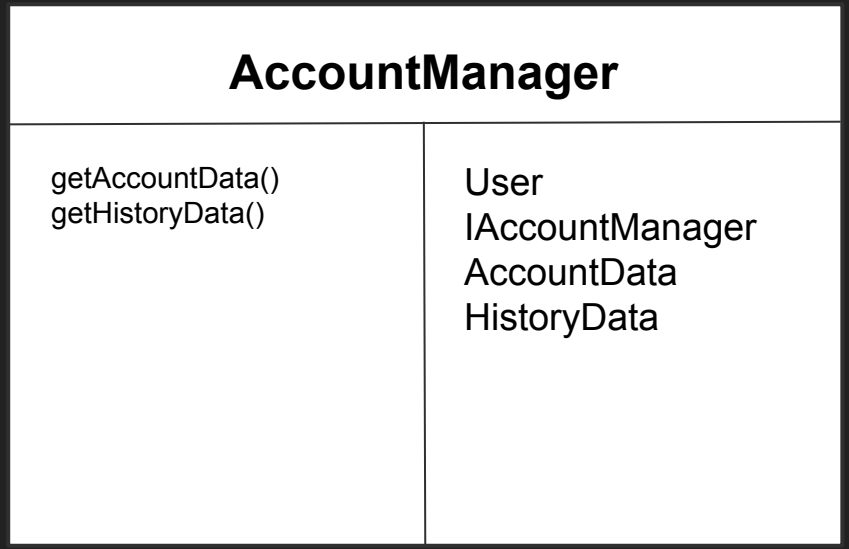
LoginManager prompts HistoryData to record ever login by a specified user.

UserManager	
Users: A mapping of ID's to users createUserBase deleteUser banUser addUser createUser createAdmin getUserData getAllUsers toString	User IUserManager

Most of what it is able to do should only be accessed by an Admin user, aside from perhaps GetUserData.

Example implementation of BanUser:

- Parameters(User other, IUserManager presenter)
- Verify this user is an admin through getter
- Set other ban status to True through setter
- Call response on presenter from IUserManager interface which returns the operation response



This use-case class retrieves account and history data

Controller

LoginSystem

UserManager
LoginManager
ConsolePresenter
ISystem

login()
signUp()
logout()
adminLoggedIn()

UserManager
LoginManager
ConsolePresenter
ISystem

AccountController

AccountManager
UserManager
ConsolePresenter

showHistoryData()
showAccountData()

AccountManager
UserManager
IPresenter

CentralConsoleController

filePath: Str
UserManager
LoginManager
ConsolePresenter
ISystem
LoginSystem
AccountController
UserController
run()
loginMenu()
adminMenu()
regularMenu()
login()
signup()
logout()
exitProgram()
createAdminUser()
deleteUser()
banUser()

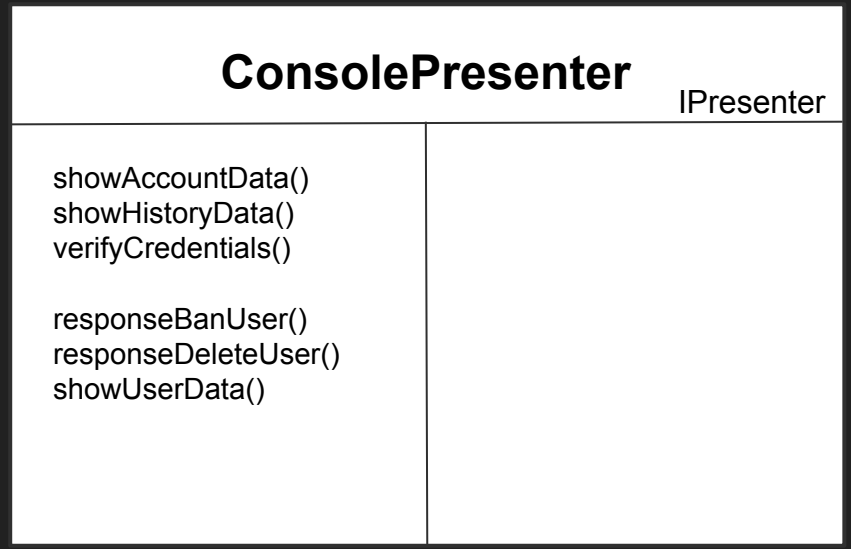
UserManager
LoginManager
IPresenter
ISystem
LoginSystem
AccountController
UserController

UserController

UserManager
ISystem
ConsolePresenter

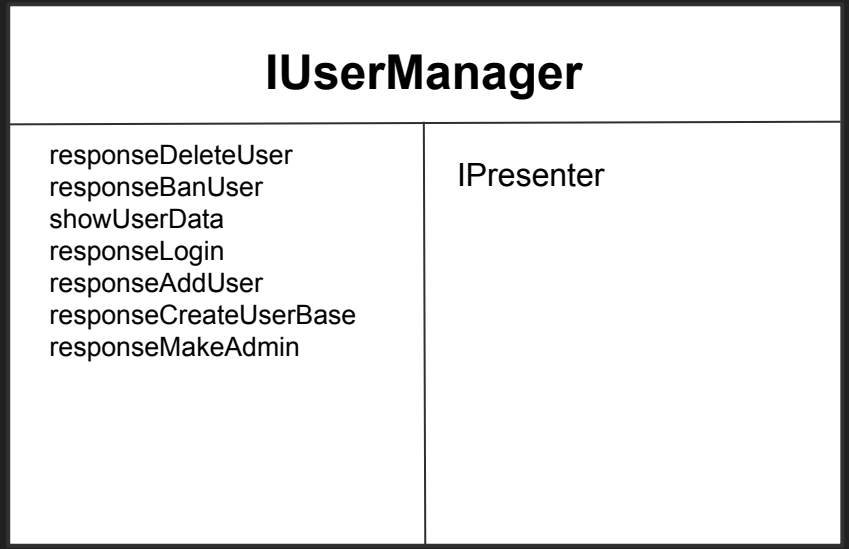
banUser()
deleteUser()
createRegUser()
createAdminUser()

Presenter



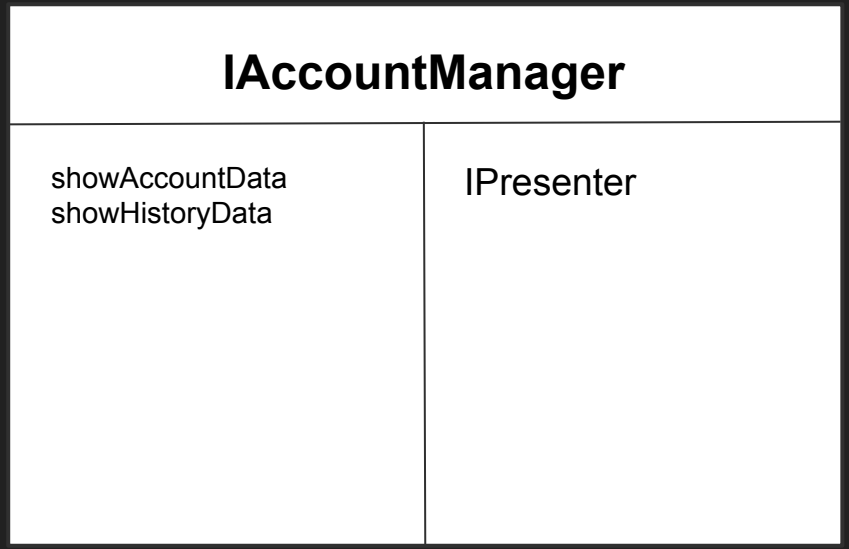
Implements methods in IPresenter (used for dependency inversion) to display data to user in console

Interface



An interface that is implemented by presenters to take in data from the UserManager use-case.

NOTE* THIS IS AN INTERFACE. DO NOT IMPLEMENT THESE METHODS IN THE USERMANAGER CLASS



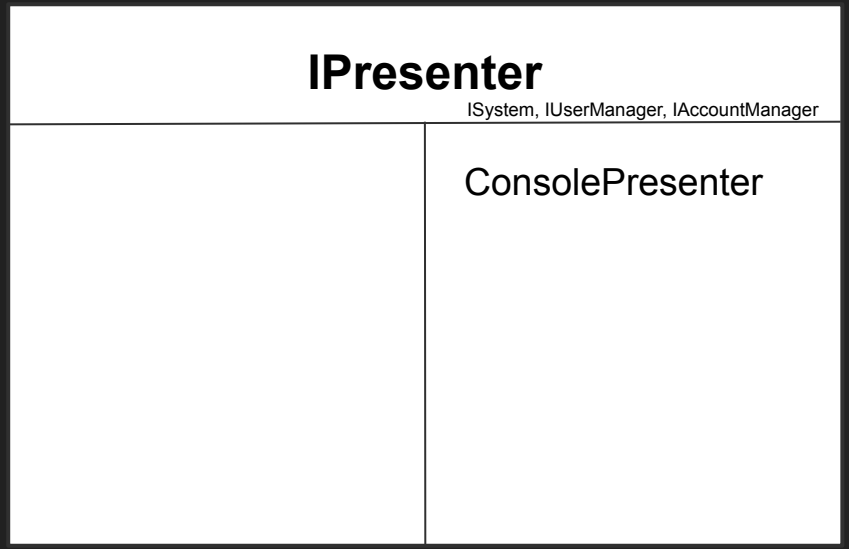
Takes the output of AccountManager method's GetAccountData and GetHistoryData and makes it presentable to the presenter

NOTE* THIS IS AN INTERFACE. DO NOT IMPLEMENT THESE METHODS IN THE ACCOUNTMANAGER CLASS

ILoginManager	
responseLogout	IPresenter

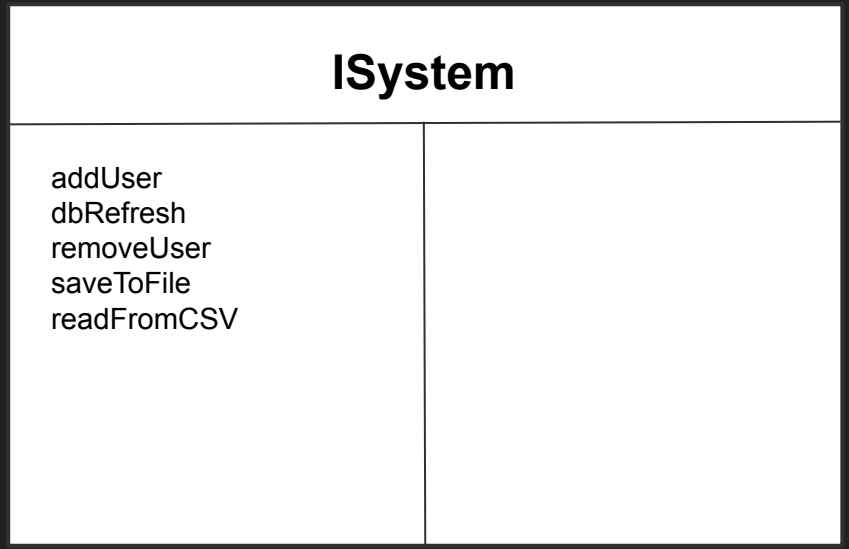
Sends the response (taken as a parameter) from the login and sign up methods from Login use case

NOTE* THIS IS AN INTERFACE. DO NOT IMPLEMENT THESE METHODS IN THE IACCOUNTMANAGER CLASS



Sends the response (taken as a parameter) from the login and sign up methods from Login use case

NOTE* THIS IS AN INTERFACE. DO NOT IMPLEMENT THESE METHODS IN THE IACCOUNTMANAGER CLASS



This Interface is a system gateway. This was included so that systemAccess can be swapped out for any database.

NOTE* THIS IS AN INTERFACE. DO NOT IMPLEMENT THESE METHODS IN THE IACCOUNTMANAGER CLASS