Group: Bohdan Yevdokymov
Alex Voytovich
Marcus Giarrusso
Class: Csc380 Section 800
Date: Sunday March 12, 2017

Password Manager Project Description

I'm looking for a program to manage passwords, and pins for all my accounts. I often find myself using the same password for everting, forgetting passwords, or writing them down. I want something to manage everything, and keep it secure.

For this program, I want it to be locked down when I'm not using it, and quickly open when needed. This means, I want to be able to log in and log off at my own leisure. I would also like the passwords to be secured, and not made plainly visible. I wouldn't want someone to be looking over my shoulder, and see the passwords in plain text. This means they need to be individually secured.

Another feature I would like to see is being able to update account information easily, and quickly, just in case I need to change my password. As well as update my profile for the actual program, in case and changes need to be made.

Identifier	Priority	Requirement
REQ1	2	Storing passwords encrypted in local file
REQ2	1	To create DB file ask for name and password
REQ3	3	This password is then used to encrypt and decrypt file
REQ4	2	To log in only select database and enter the password
REQ5	5	Try to update the DB file at every login also when logged in have a button for manual updating
REQ6	5	(Maybe)Update using JDBC, for every db file created new table in mysql is created on server, name of the table randomly selected (hash key) and saved in local file, table on the server contain rows like (login/password/comment) encrypted using personal key entered by user when created db file. (get passwords - encrypt - upload, or download - decrypt - output) OR Simply sync whole .db file
REQ7	4	When updating first save file then take “last modified” time of file and upload it to the server table timestamp with all of the logins, passwords… OR Write each time in .db file write date when saved in format yyyy:mm:dd:hh:mm:ss
REQ8	3	At logging in get date from file and timestamp from server and compare them, if version on a server is newer - replace existing data, if local version is newer upload and replace table on the server.
REQ9	4	Also update information before closing a program or automatically when changes are made.
REQ10	3	Simply have “copy login”, “copy password” buttons (clear clipboard after 30 seconds)
REQ11	5	? have “generate password” button, selecting the length/ letters/digits and give random

Identifier	User Story	Size
St_1	As a user, I have a simple, easy and user-friendly interface with little clutter and a modern look.	8 Points
St_2	As a user, my passwords are organized neatly alphabetically.	6 Points
St_3	As a user, I have to only remember 1 main password to login.	3 Points
St_4	As a user, I have a ‘hint’ for the main password in case I forget.	2 Points
St_5	As a user, I have all passwords encrypted and safe.	9 Points
St_6	As a user, I my application locks after a defined period of time of my choosing	3 Points
St_7	As a user, I can also lock the application at any given time.	3 Points
St_8	As a user, I have the ability to have auto-generated passwords when creating new logins.	6 Points
St_9	As a user, I can synchronize my passwords between multiple computers.	7 Points
St_10	As a user, I’m able to edit my usernames and passwords as needed.	4 Points
St_11	As a user, I can add notes to my logins	2 Points
St_12	As a user, I can copy and paste my usernames and passwords.	3 Points
St_13	As a user, I have a ‘help’ button that offers help and guidance	5 Points

Actor	Actor's Goals	Use Case Name
Initiating User	To either choose a pre-existing database or create new database and login.	Create-Database (UC-1) Log-In (UC-2) Decrypt (UC-4)
Initiating User	To save and securely store specified passwords.	Encrypt (UC-3)
Initiating User	To access passwords to copy & paste specified username / passwords / login items.	Decrypt (UC-4)
Initiating User	To create a new user account database and allow access to database.	Add-Account (UC-5)
Initiating User	To remove / delete given user account and disable access to database.	Remove-Account (UC-6)
Initiating User	To update given database / user account when changes are made	Update (UC-7)
Initiating User	To configure preferences of account (i.e. type of account, URL, 'other' notes)	Preferences (UC-8)
Initiating User	To generate password for given account.	Generate-Password (UC-9)
Initiating User	To lock / disable access and log-out	Lock (UC-10) Encrypt (UC -1)
Initiating User	To disable lock & log-in to user account	Unlock (UC-11) Decrypt (UC-2)
Initiating User	To auto-lock database when left unlocked for a given time interval.	Lock (UC-8) Encrypt (UC -1)

Use Case UC-1:	Create-Database
Related Requirements	REQ2, REQ5, REQ6, REQ7
Initiating Actor:	Any of: End-User
Actor's Goal:	To create new database at login.
Participating Actors	Login, SavePassword, SaveDatabase, Unlock, Decrypt
Preconditions:	- Available database - Available storage
Postconditions:	- A Database file is created and available to hold user accounts

Flow of Events for Main Success Scenario:

→ 1. End user prompted to create new database
2. include : *Add-Account* (UC-5)
← 3. System (a) sends new database to 'Add-Account' (b) signals to 'Log-in' (c) signals to 'Lock' to disarm the lock
← 4. End user enters passwords for newly created database and logs in
→ 5. End user granted access to database, auto-lock timer begins countdown

Use Case UC-2:	Log-In
Related Requirements	REQ4, REQ5, REQ6, REQ7, REQ8 and REQ10
Initiating Actor:	Any of: End-User
Actor's Goal:	To disarm the lock and enter database
Participating Actors	UnLock, AutoTimer
Preconditions:	- A specified database is selected - Matching password
Postconditions:	The auto-lock timer has started countdown

Flow of Events for Main Success Scenario:

→ 1. End user prompted to select database and enter valid password
2. include : Unlock (UC-11)
← 3. System (a) checks to see if password matches (b) signals to database to 'Unlock'
← 4. System signals to start 'Auto-Lock' timer
→ 5. End user is granted access to given database

Use Case UC-3:	Encrypt
Related Requirements	REQ1, REQ3 and REQ6
Initiating Actor:	Any of: End User
Actor's Goal:	To save and securely store specified passwords.
Participating Actors	Lock
Preconditions:	- A valid user account and password are created and saved
Postconditions:	- User account is encrypted and saved

Flow of Events for Main Success Scenario:

→ 1. End user enters a valid user account and password
2. include : Generate-Password (UC-9)
← 3. System (a) 'Generates-Password' if user wishes (b) system generates password or enters valid password and saves clicks "save" button
← 4. System signals to encrypt user information
→ 5. End users information is saved and encrypted

Use Case UC-4:	Decrypt
Related Requirements	REQ3 and REQ6
Initiating Actor:	Any of: End User
Actor's Goal:	To access passwords to copy & paste specified username / passwords / login items.
Participating Actors	CopyPassword, CopyUserName
Preconditions:	- A valid username and password is saved and encrypted - User clicks "Copy Username" or "Copy Password"
Postconditions:	- Username and password are copied to clipboard to be used - Timer begins to erase clipboard after specific time

Flow of Events for Main Success Scenario:

→ 1. End user wishes to use username and password to login
2. End user clicks "Copy Username" or "Copy Password":
← 3. System (a) locates valid encrypted username and password and (b) signals to decrypt them
← 4. Username and password are decrypted and copied to clipboard
→ 5. End user can now paste username and password to destination of choice

Use Case UC-5:	Add-Account
Related Requirements	REQ1, REQ2 and REQ6
Initiating Actor:	Any of: End User
Actor's Goal:	To create a new user account database and allow access to database.
Participating Actors	Update, Encrypt, Preferences, Generate-Password, LogIn
Preconditions:	- User must be logged-in to valid database - Available space within database
Postconditions:	- Database saves newly added account

Flow of Events for Main Success Scenario:

→ 1. End user wishes to add new account to database

2. include : *Update* (UC-7) *Preferences (UC-8) Encrypt (UC-3)*

← 3. System (a) signals to current database to allow new account (b) signals to 'Preferences' to add specified data (c) signals to encrypt to safely store new data and (d) signals to 'Update' to update newly added data

← 4. System adds newly encrypted account to current database

→ 5. End user is able to utilize new account

Use Case UC-6:	Remove-Account
Related Requirements	REQ1, REQ2 and REQ6
Initiating Actor:	Any of: End User
Actor's Goal:	To remove / delete given user account and disable access to database.
Participating Actors	Update, LogIn
Preconditions:	- User must be logged-in to valid database - Must be a valid account
Postconditions:	- Database erases specified account from storage

Flow of Events for Main Success Scenario:

→ 1. End user wishes to delete old / un-used account from database storage & clicks "delete account" button

2. include : *Update* (UC-7)

← 3. System (a) signals to current database the specified account (b) deletes that account

← 4. System erases specified account from its data storage

→ 5. End user can no longer use or see deleted account information

Use Case UC-7:	Update
Related Requirements	REQ5, REQ6 and REQ9
Initiating Actor:	Any of: End User
Actor's Goal:	To update given database / user account when changes are made
Participating Actors	(All Actors)
Preconditions:	- User is within a valid database - User alters / changes any data or information within database
Postconditions:	- Any and all data altered or changed is updated and saved

Flow of Events for Main Success Scenario:

→ 1. Tenant/Landlord arrives at the door and selects the menu item "Unlock"

2. include : *(All Use Cases)*

← 3. System (a) signals to region in which changes were made (b) signals to discard old information / data

← 4. System signals to save all newly updated data

→ 5. End user's database / accounts are up-to-date to utilize

Use Case UC-8:	Preferences
Related Requirements	REQ1, REQ2 and REQ6
Initiating Actor:	Any of: End User
Actor's Goal:	To configure preferences of account (i.e. type of account, URL, 'other' notes)
Participating Actors	AddAccount, Update
Preconditions:	- User must be within a valid account within a valid database
Postconditions:	- Users account's preferences are saved and seen by user

Flow of Events for Main Success Scenario:

→ 1. End user wishes to add account info and notes to specified account

2. include : *Add-Account (UC-5) Update (UC-7)*

← 3. System (a) signals to database and selects specified account which (c) allows user to input preferred information such as account type, URL, & notes

← 4. System saves user preferred information

→ 5. User can now see and use inputted information for later use

Use Case UC-9:	Generate-Password
Related Requirements	REQ1, REQ3 and REQ11
Initiating Actor:	Any of: End User
Actor's Goal:	To generate password for given account.
Participating Actors	AddAccount, Encrypt, Update
Preconditions:	- User must be within a valid database - User must be adding a new account - System displays the available function: "generate-password" button
Postconditions:	- A password is generated, saved, & encrypted for given account

Flow of Events for Main Success Scenario:

→ 1. End user prefers more secure generated password rather than thinking of one & selects button to generate one

2. include : *Add-Account (UC-5) Encrypt (UC- 3) Update (UC-7)*

← 3. System (a) signals to "Generate-Password" to (b) generate pseudo-random password

← 4. System saves and encrypts generated password for current account

→ 5. End user is able to view, copy, paste & utilize generated password for account

Use Case UC-10:	Lock
Related Requirements	REQ1, REQ3 and REQ6
Initiating Actor:	Any of: End User
Actor's Goal:	To lock / disable access and log-out
Participating Actors	Encrypt, LogOut
Preconditions:	- User is within an unlocked database
Postconditions:	- Database is enabled for user

Flow of Events for Main Success Scenario:

→ 1. End user is finished with using application and presses "Lock" button

2. include : *Encrypt (UC-3)*

← 3. System (a) signals to database to arm the 'Lock' (b) encrypt the database's content (c) and lock database

← 4. System changes to Lock screen & prompts user to re-enter specified database

→ 5. End user is able to walk away knowing their data is safely stored from other users

Use Case UC-11:	Unlock
Related Requirements	REQ3 and REQ6
Initiating Actor:	Any of: End User
Actor's Goal:	To disable lock & log-in to user account
Participating Actors	LogIn, Decrypt, Update
Preconditions:	- A valid specified database is selected - Entered password matches database password
Postconditions:	- Database is unlocked and user is able to view data

Flow of Events for Main Success Scenario:

→ 1. Tenant/Landlord arrives at the door and selects the menu item "Unlock"

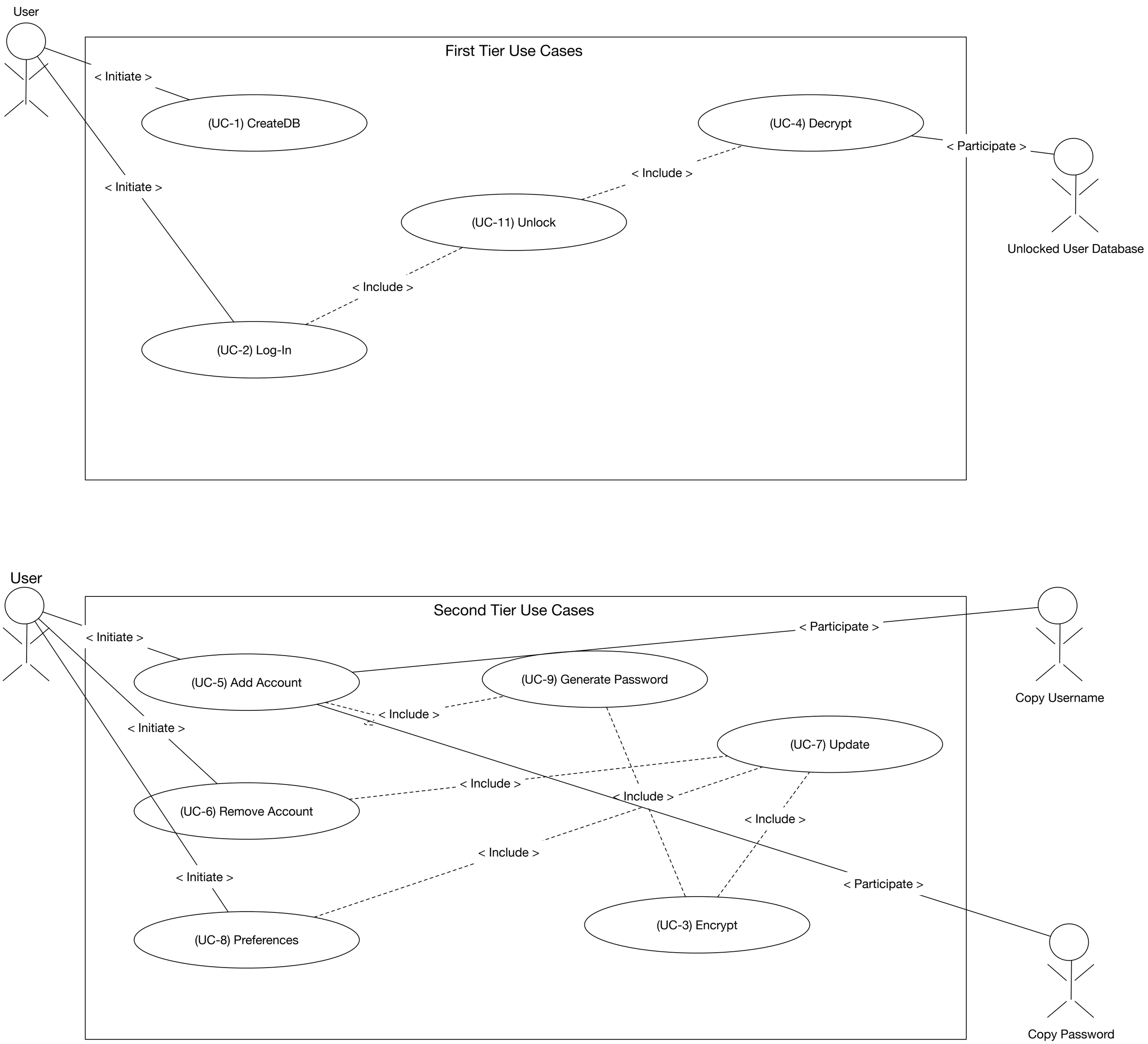
2. include : *Log-In (UC-2) Decrypt (UC- 4) Update (UC-7)*

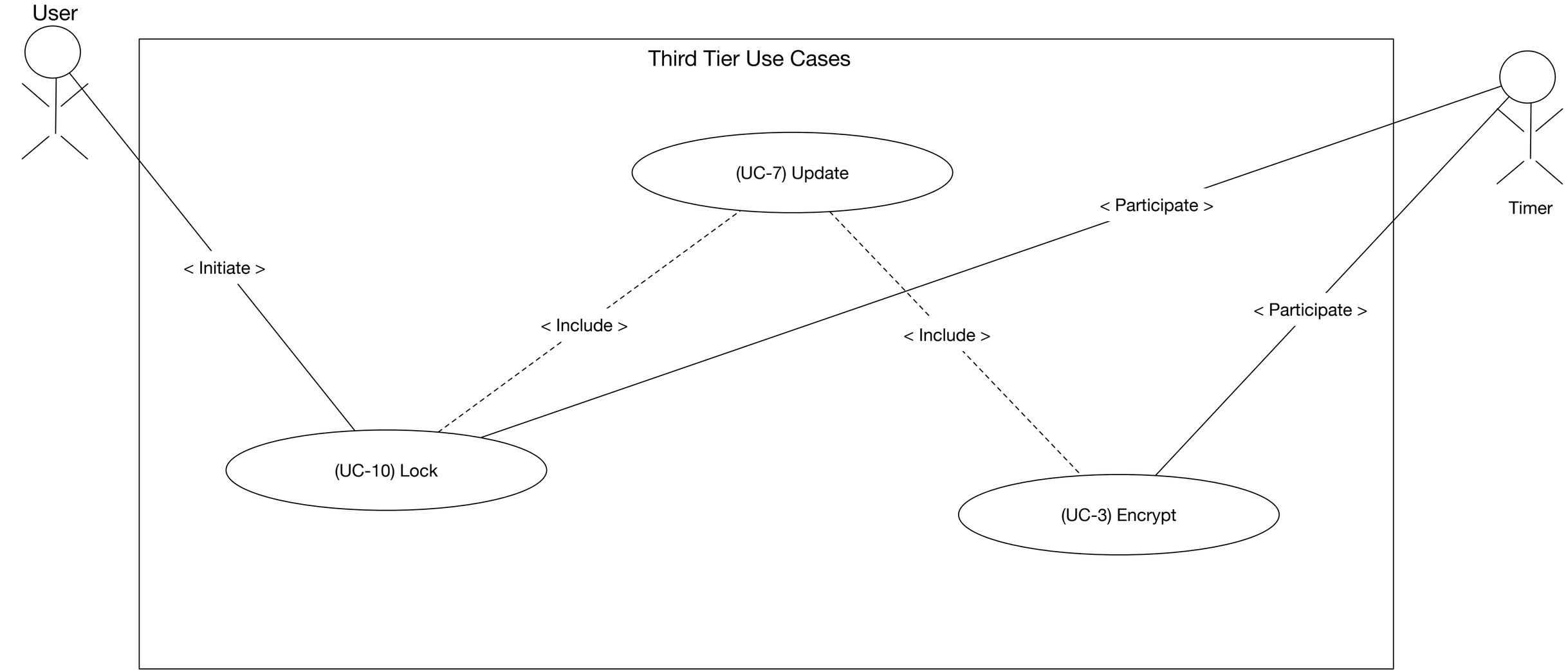
← 3. System (a) signals to database that entered password matches (b) signals to 'UnLock' to disarm the lock (c) signals 'Update' to update database content

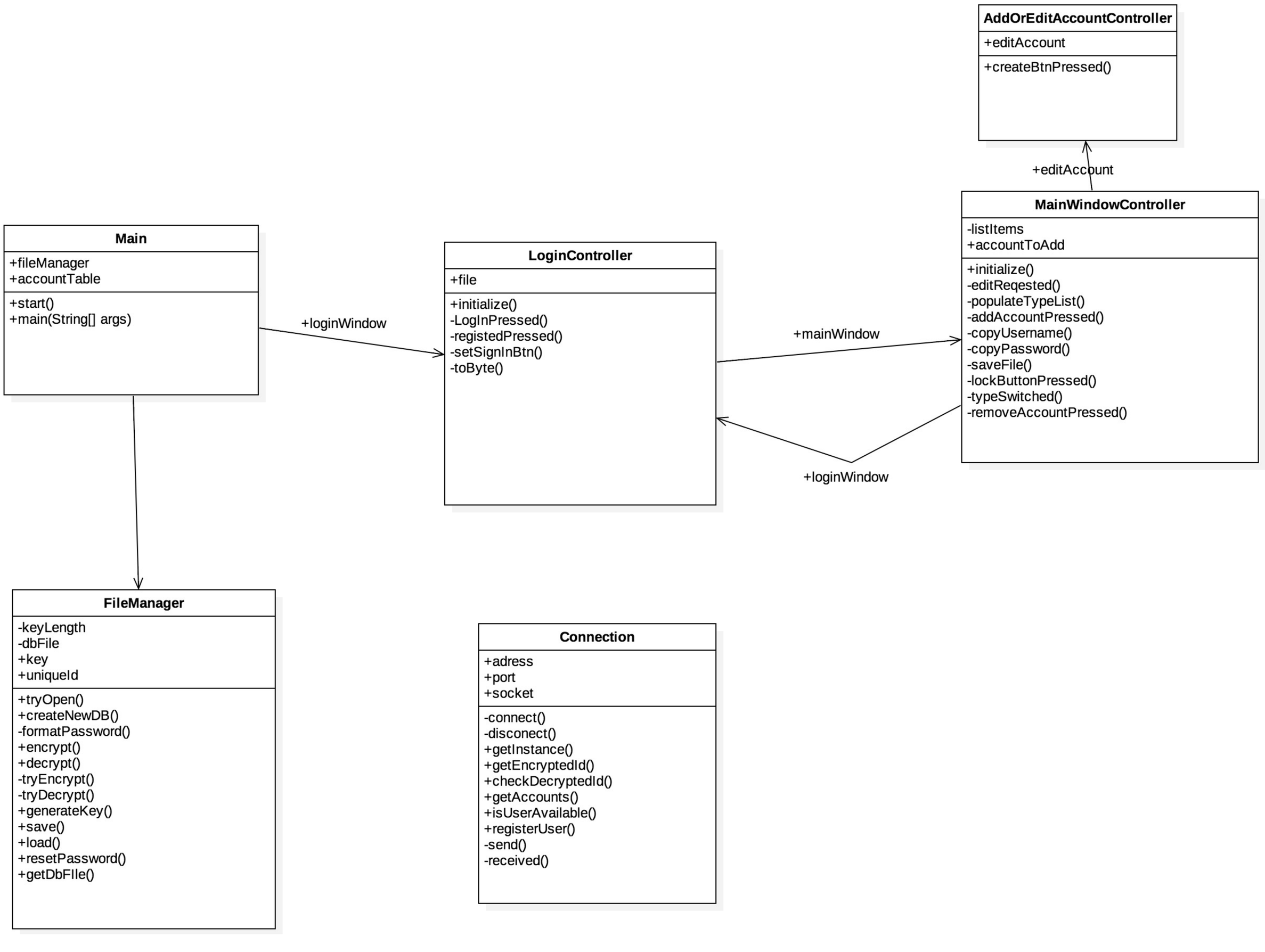
← 4. System signals 'Auto-Lock' to start timer

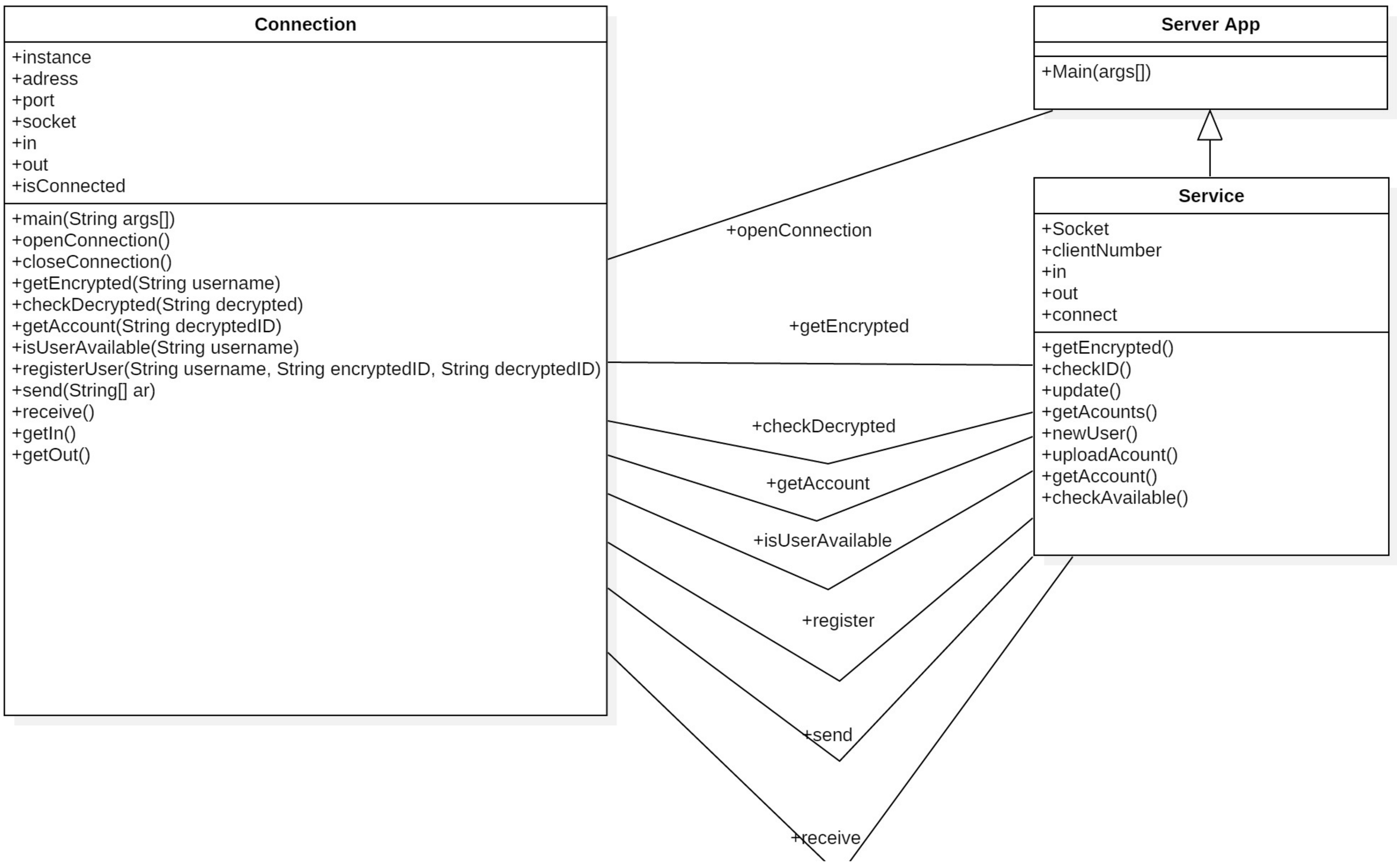
→ 5. End user enters database and is able to utilize its content

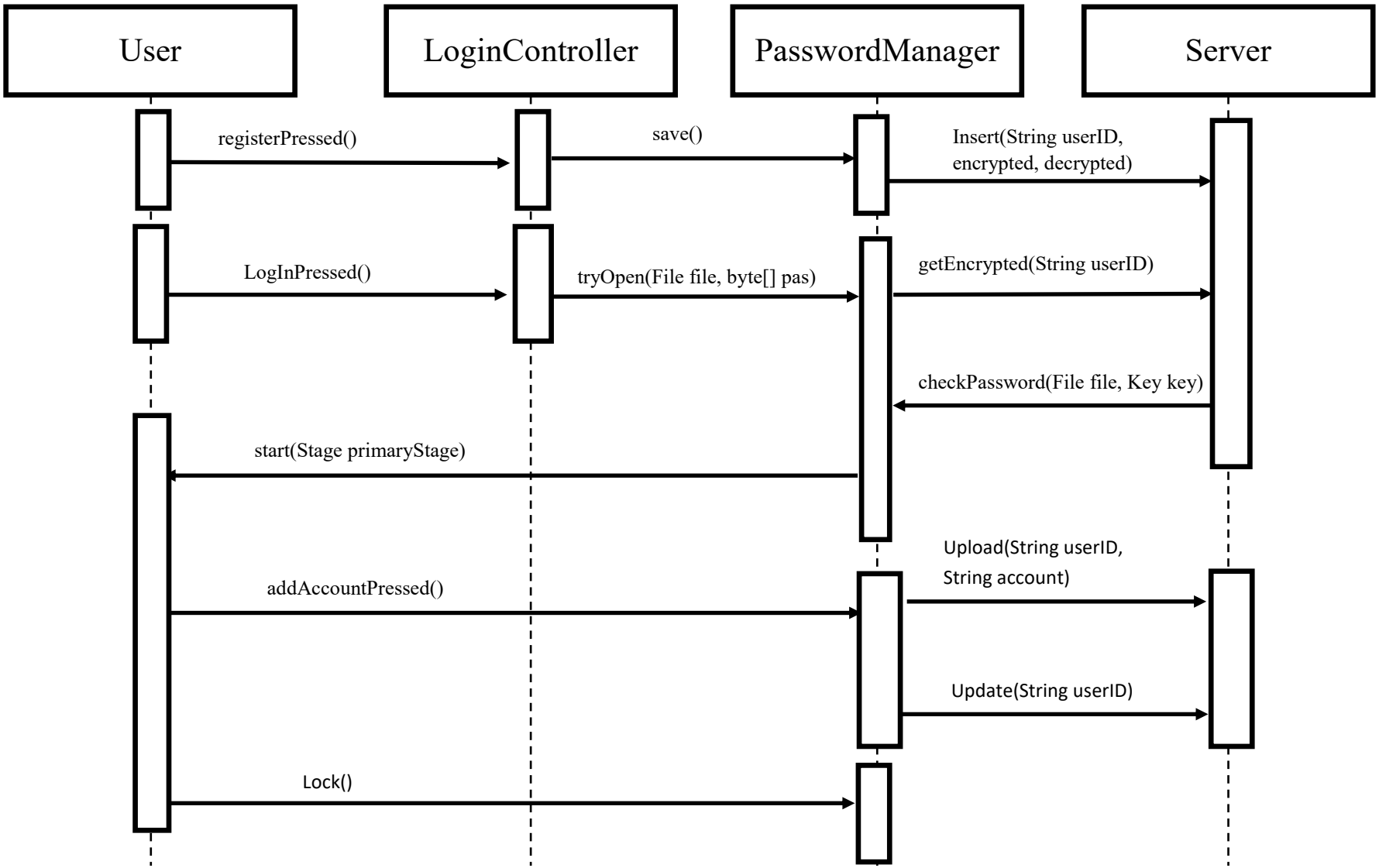
Requirement	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10	UC-11
REQ1			X		X	X		X	X	X	
REQ2	X				X	X		X			
REQ3			X	X					X	X	X
REQ4		X									
REQ5	X	X					X				
REQ6	X	X	X	X	X	X	X	X		X	X
REQ7	X	X									
REQ8		X									
REQ9							X				
REQ10		X									
REQ11									X		


First Tier Use Cases
User
< Initiate >
(UC-1) CreateDB
(UC-4) Decrypt
< Participate >
Unlocked User Database
< Include >
< Initiate >
(UC-11) Unlock
< Include >
(UC-2) Log-In
Second Tier Use Cases
User
< Initiate >
< Participate >
(UC-5) Add Account
(UC-9) Generate Password
Copy Username
< Include >
< Initiate >
(UC-7) Update
< Include >
(UC-6) Remove Account
< Include >
< Include >
< Include >
< Initiate >
< Participate >
(UC-8) Preferences
(UC-3) Encrypt
Copy Password


User
Third Tier Use Cases
(UC-7) Update
< Participate >
Timer
< Initiate >
< Include >
< Include >
< Participate >
(UC-10) Lock
(UC-3) Encrypt


AddOrEditAccountController
+editAccount
+createBtnPressed()
+editAccount
MainWindowController
-listItems
+accountToAdd
+initialize()
-editReqested()
-populateTypeList()
-addAccountPressed()
-copyUsername()
-copyPassword()
-saveFile()
-lockButtonPressed()
-typeSwitched()
-removeAccountPressed()
Main
+fileManager
+accountTable
+start()
+main(String[] args)
+loginWindow
LoginController
+file
+initialize()
-LogInPressed()
-registedPressed()
-setSignInBtn()
-toByte()
+mainWindow
+loginWindow
FileManager
-keyLength
-dbFile
+key
+uniqueId
+tryOpen()
+createNewDB()
-formatPassword()
+encrypt()
+decrypt()
-tryEncrypt()
-tryDecrypt()
+generateKey()
+save()
+load()
+resetPassword()
+getDbFIle()
Connection
+adress
+port
+socket
-connect()
-disconect()
+getInstance()
+getEncryptedId()
+checkDecryptedId()
+getAccounts()
+isUserAvailable()
+registerUser()
-send()
-received()


Connection
+instance
+adress
+port
+socket
+in
+out
+isConnected
+main(String args[])
+openConnection()
+closeConnection()
+getEncrypted(String username)
+checkDecrypted(String decrypted)
+getAccount(String decryptedID)
+isUserAvailable(String username)
+registerUser(String username, String encryptedID, String decryptedID)
+send(String[] ar)
+receive()
+getIn()
+getOut()
Server App
+Main(args[])
Service
+Socket
+clientNumber
+in
+out
+connect
+getEncrypted()
+checkID()
+update()
+getAcounts()
+newUser()
+uploadAcount()
+getAccount()
+checkAvailable()
+openConnection
+getEncrypted
+checkDecrypted
+getAccount
+isUserAvailable
+register
+send
+receive


User
LoginController
PasswordManager
Server
registerPressed()
save()
Insert(String userID,
encrypted, decrypted)
LogInPressed()
tryOpen(File file, byte[] pas)
getEncrypted(String userID)
checkPassword(File file, Key key)
start(Stage primaryStage)
Upload(String userID,
String account)
addAccountPressed()
Update(String userID)
Lock()