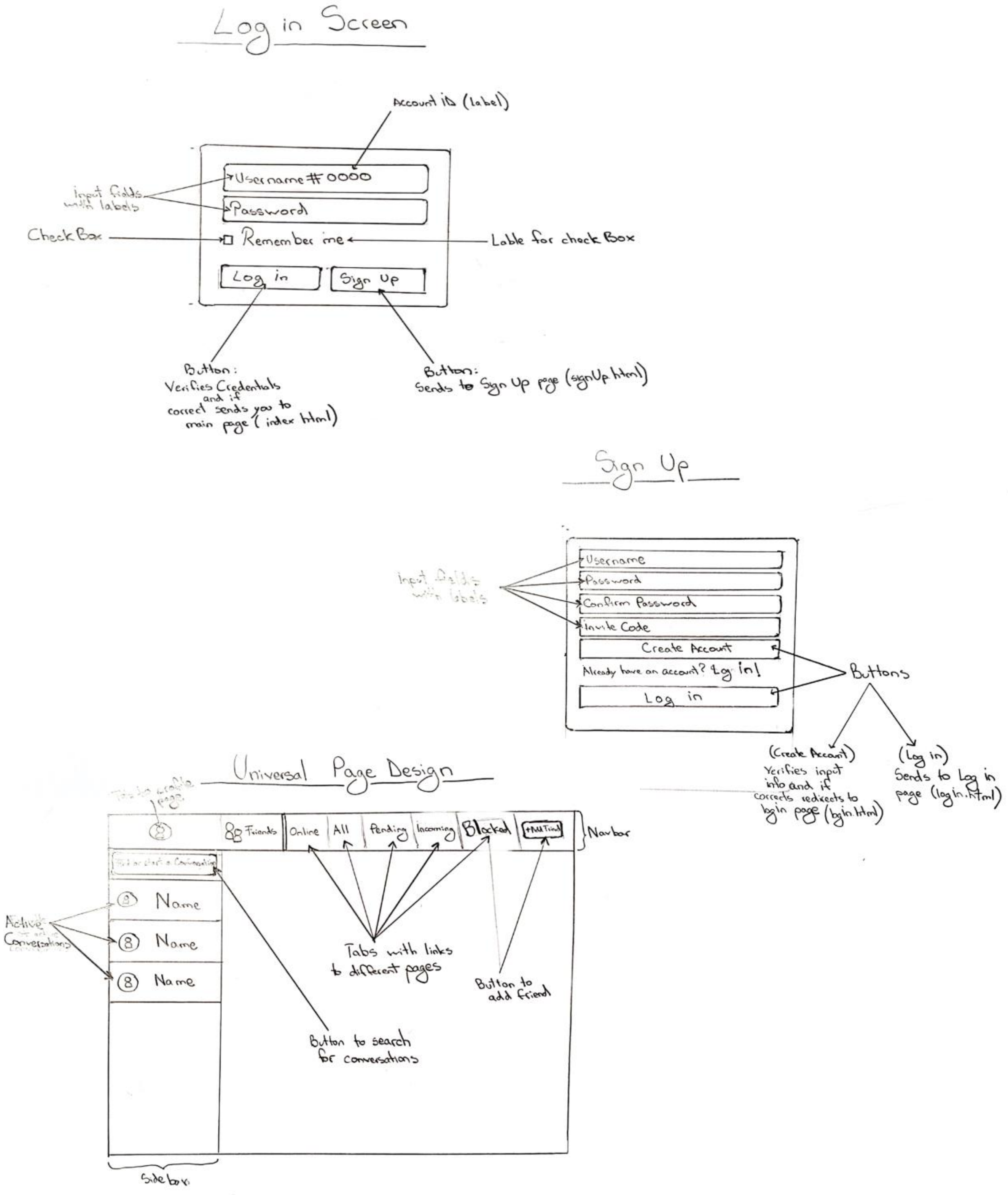
Personal Code: jfg073

Criterion B: Design

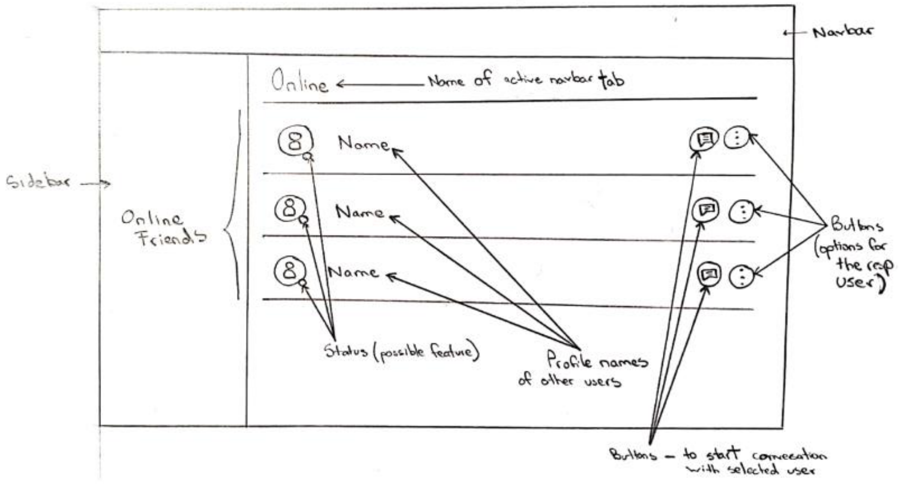
Design of the application:

Initial Paper Designs:



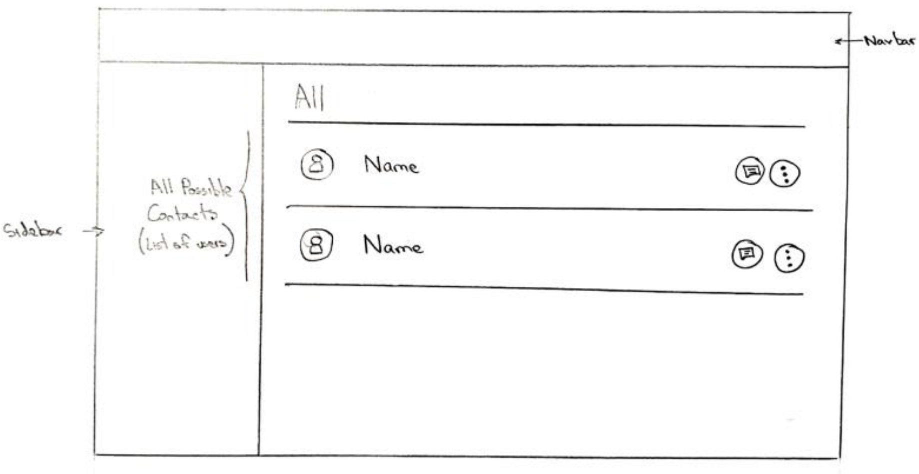
Personal Code: jfg073

Home Screen

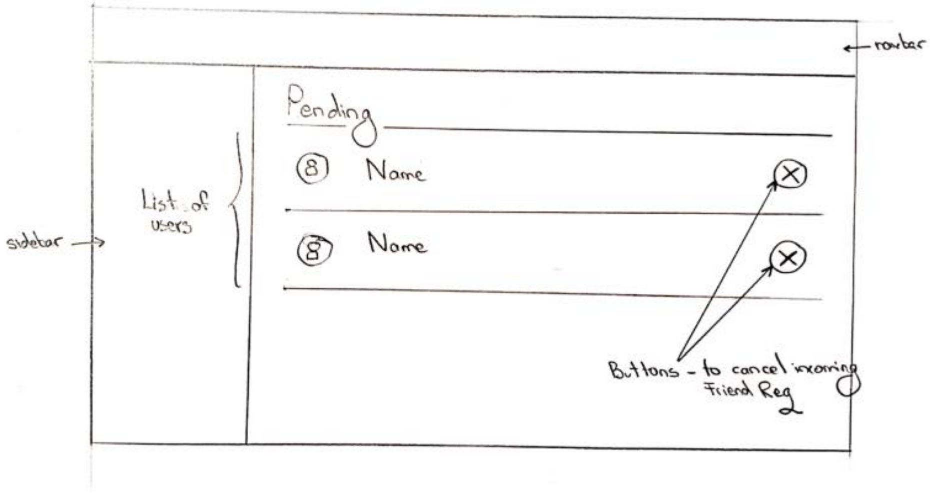


Navbar
Online
Name of active navbar tab
Sidebar
Online Friends
Name
Name
Name
Buttons (options for the resp user)
Status (possible feature)
Profile names of other users
Buttons - to start conversation with selected user

All Page

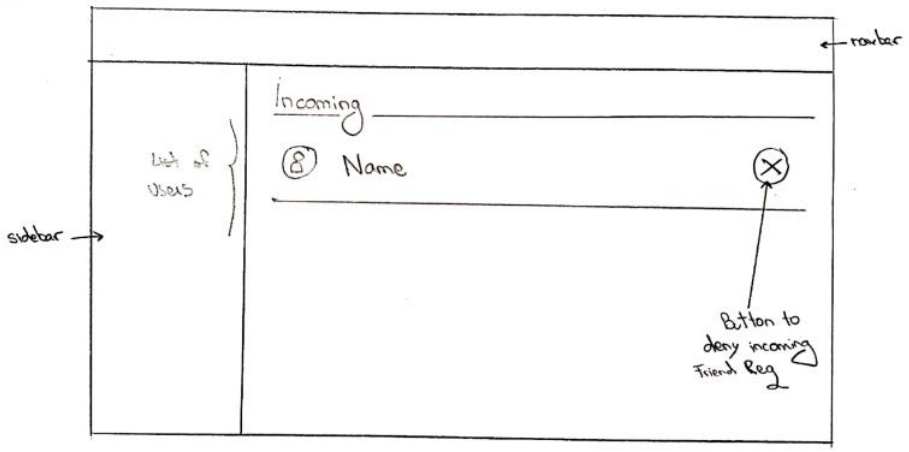


Pending Page

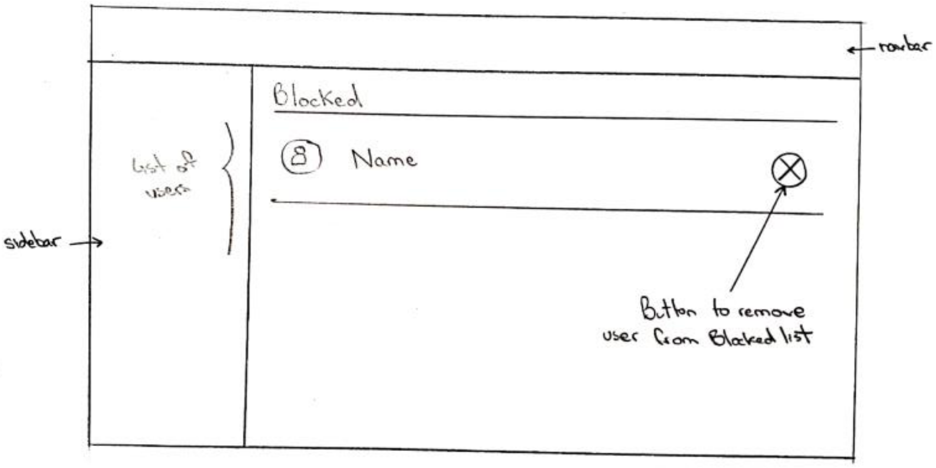


Personal Code: jfg073

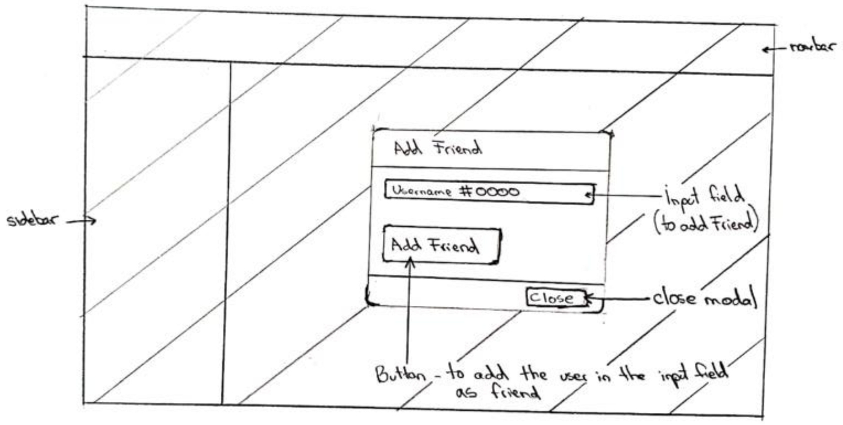
Incoming Page



Blocked Page

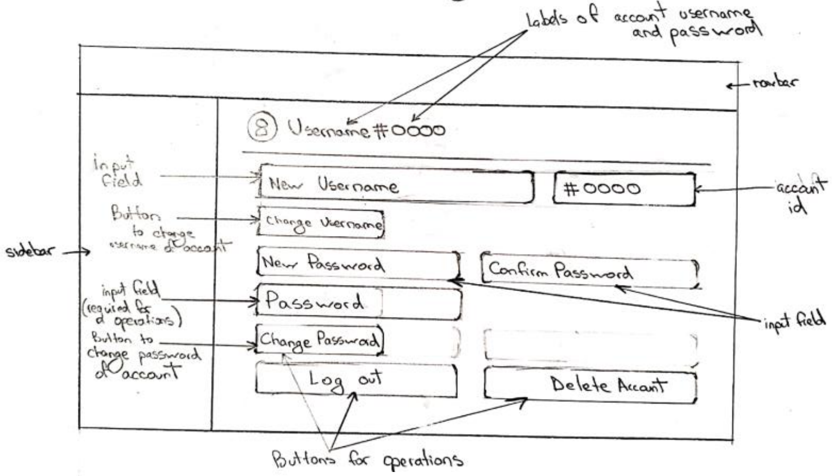


Add Friend Modal

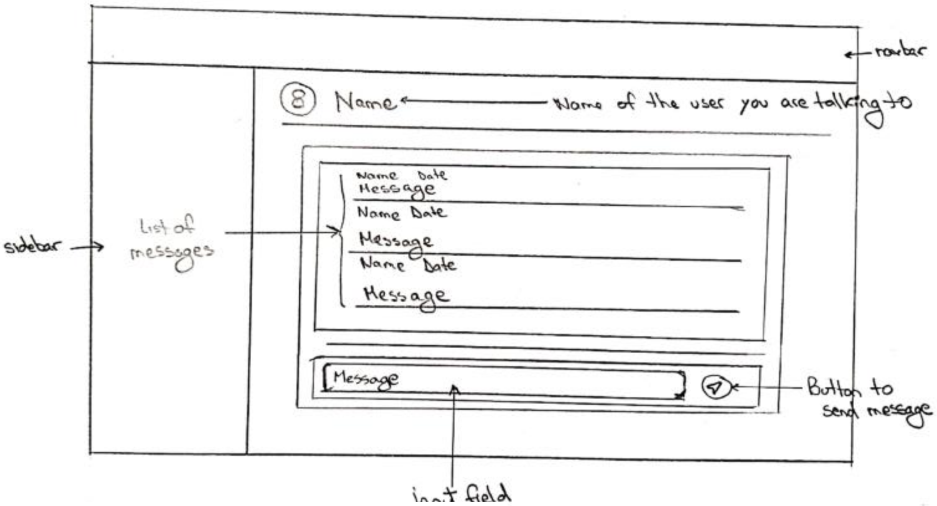

navbar
Add Friend
Username #0000
Input field (to add Friend)
Add Friend
sidebar
Close
close modal
Button - to add the user in the input field as friend

Personal Code: jfg073

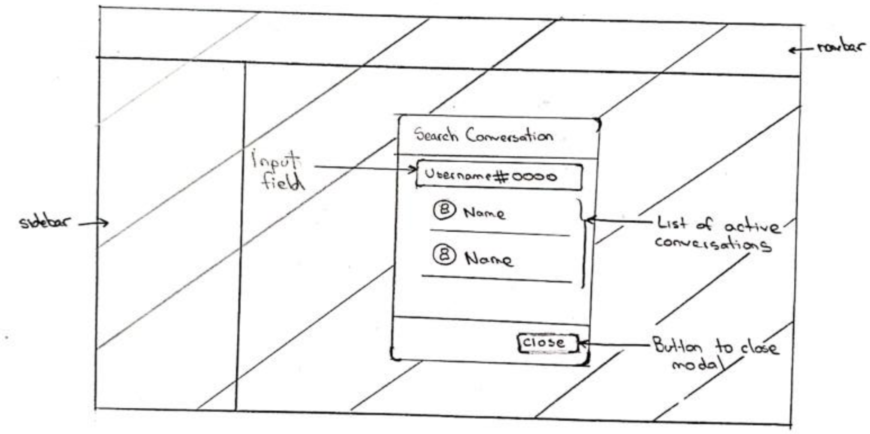
Profile Page



Conversation Page



Search Conversation Modal



Personal Code: jfg073

Electronic Designs:

After using (Draw.io, n.d.) to create a much more detailed design of the web application (Image. 1), I received feedback from my client. I then accounted for the changes proposed to make sure the product fulfills the client expectations.

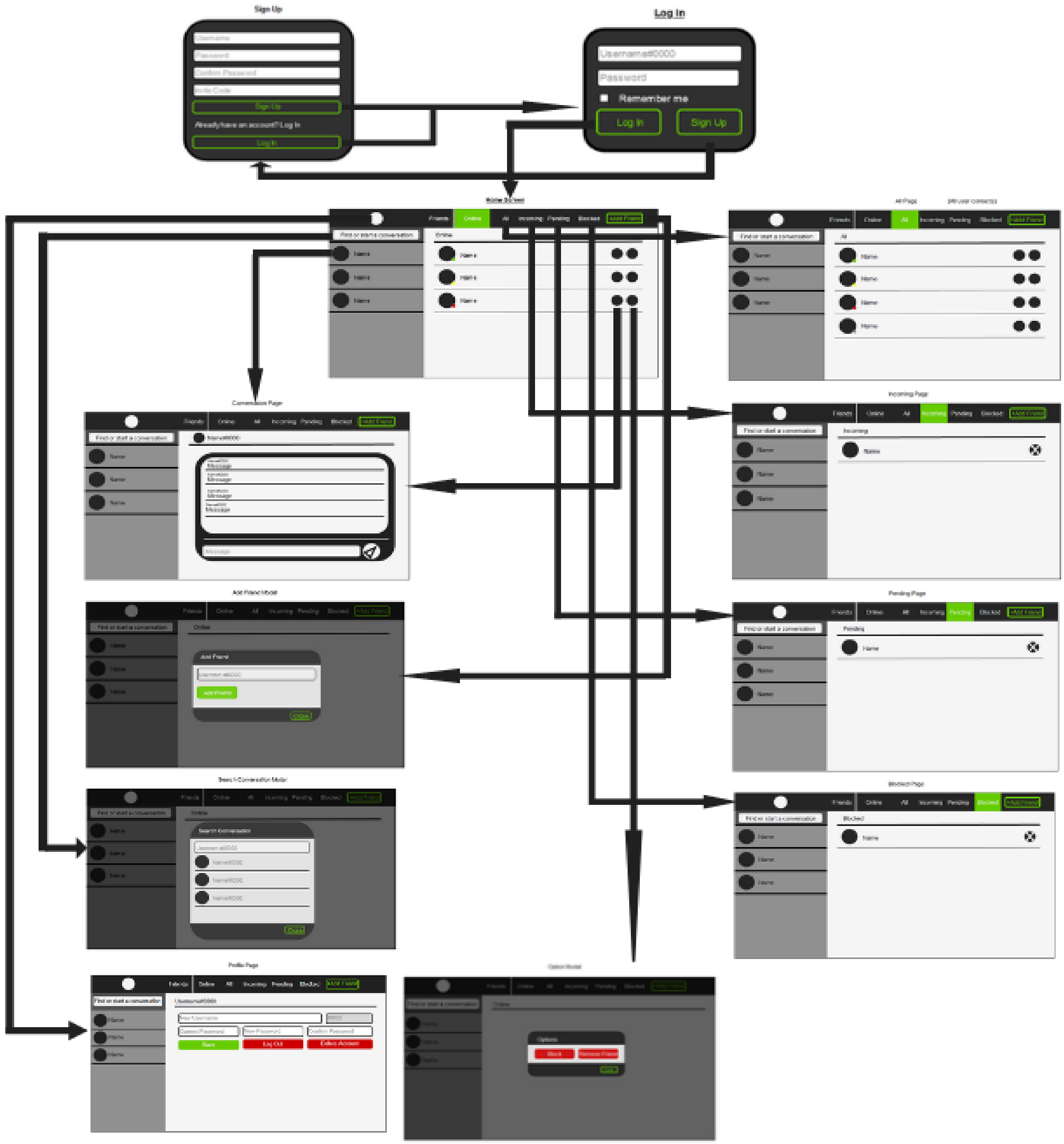
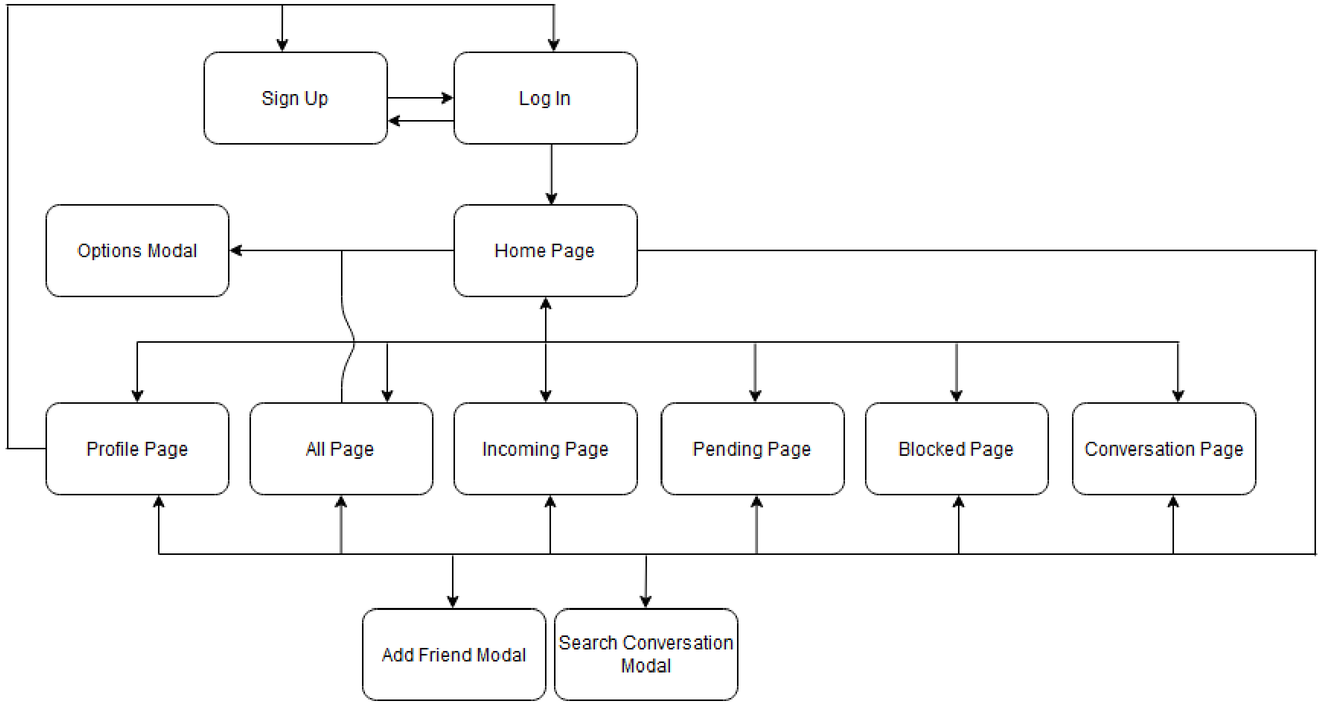


Image 1: Electronic design of application

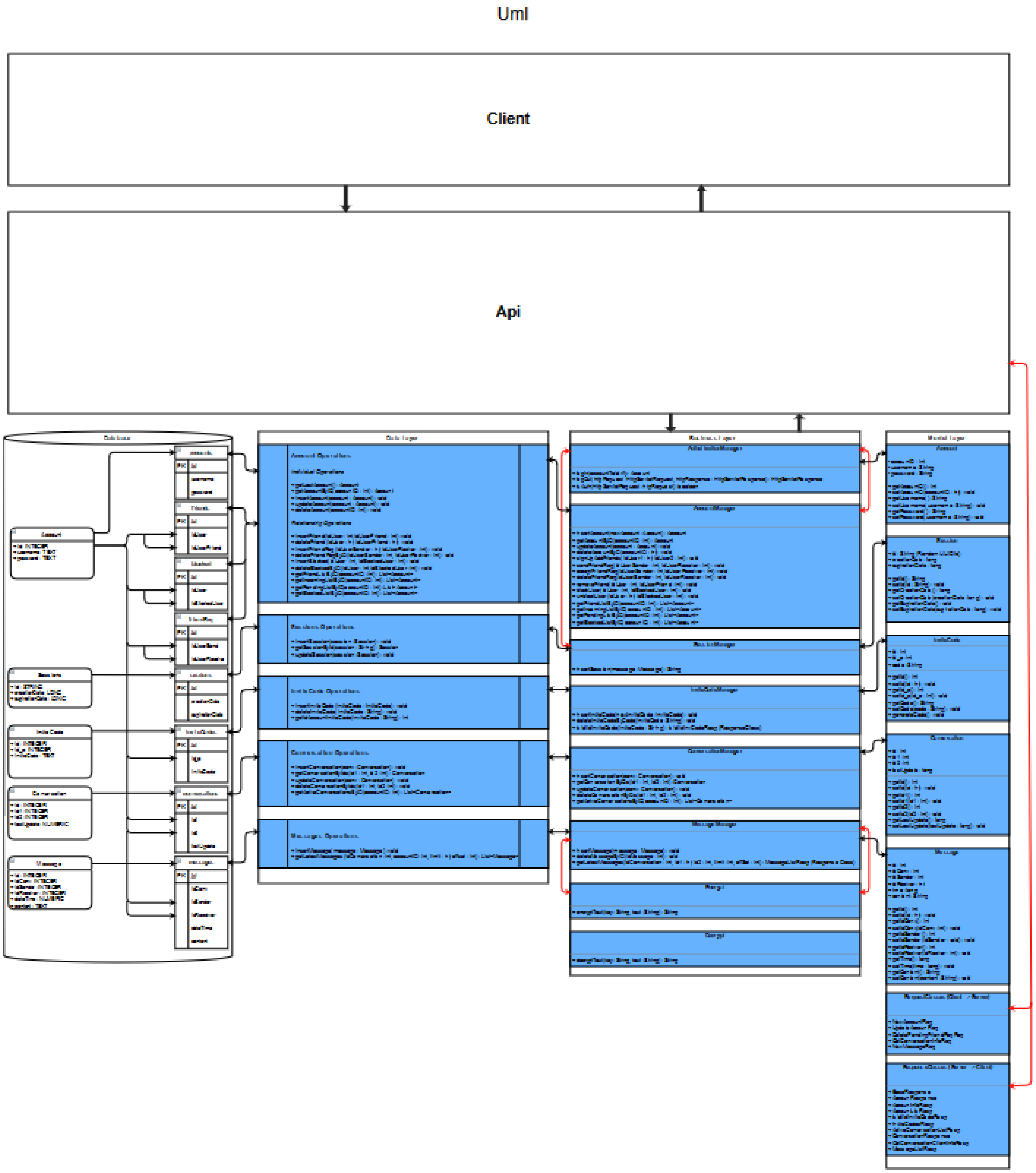
Personal Code: jfg073

System Storyboard:



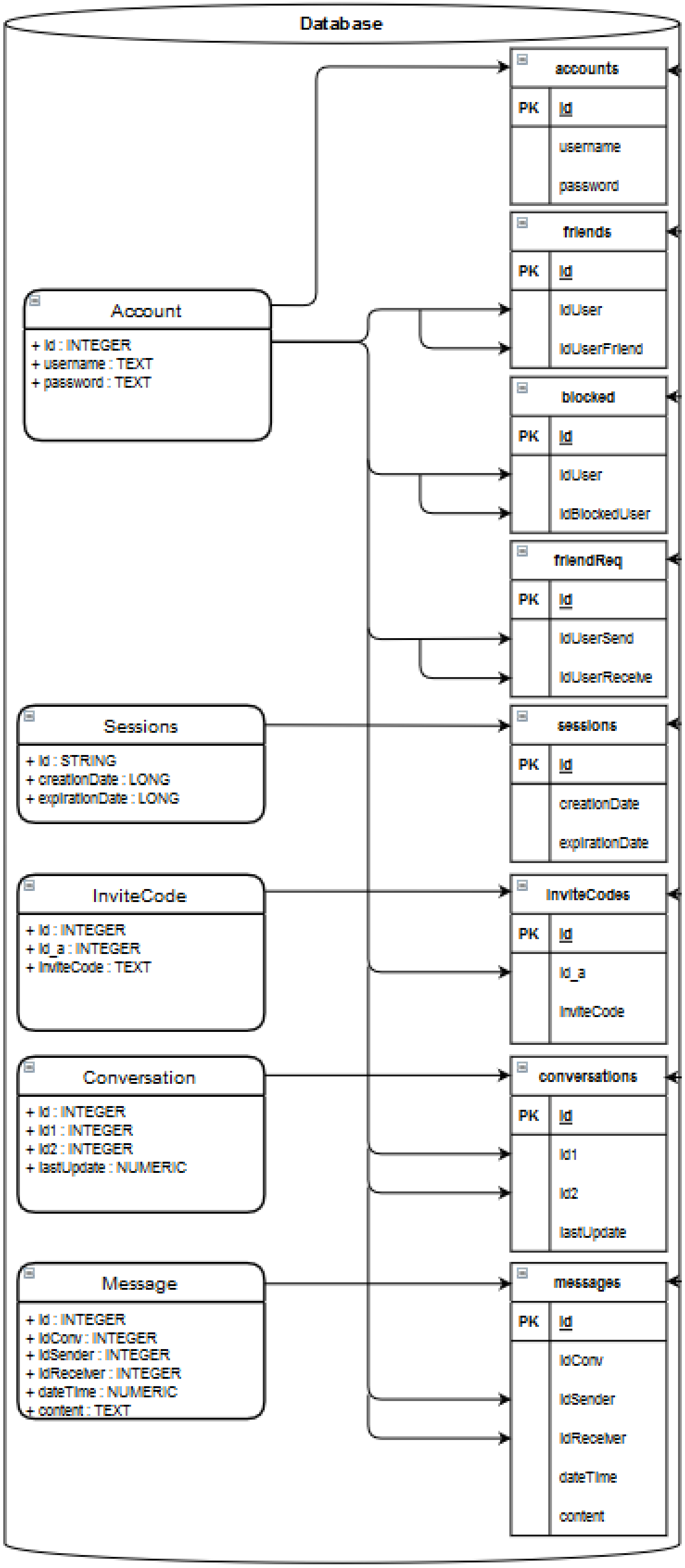
Personal Code: jfg073

UML Class Interaction Diagram:



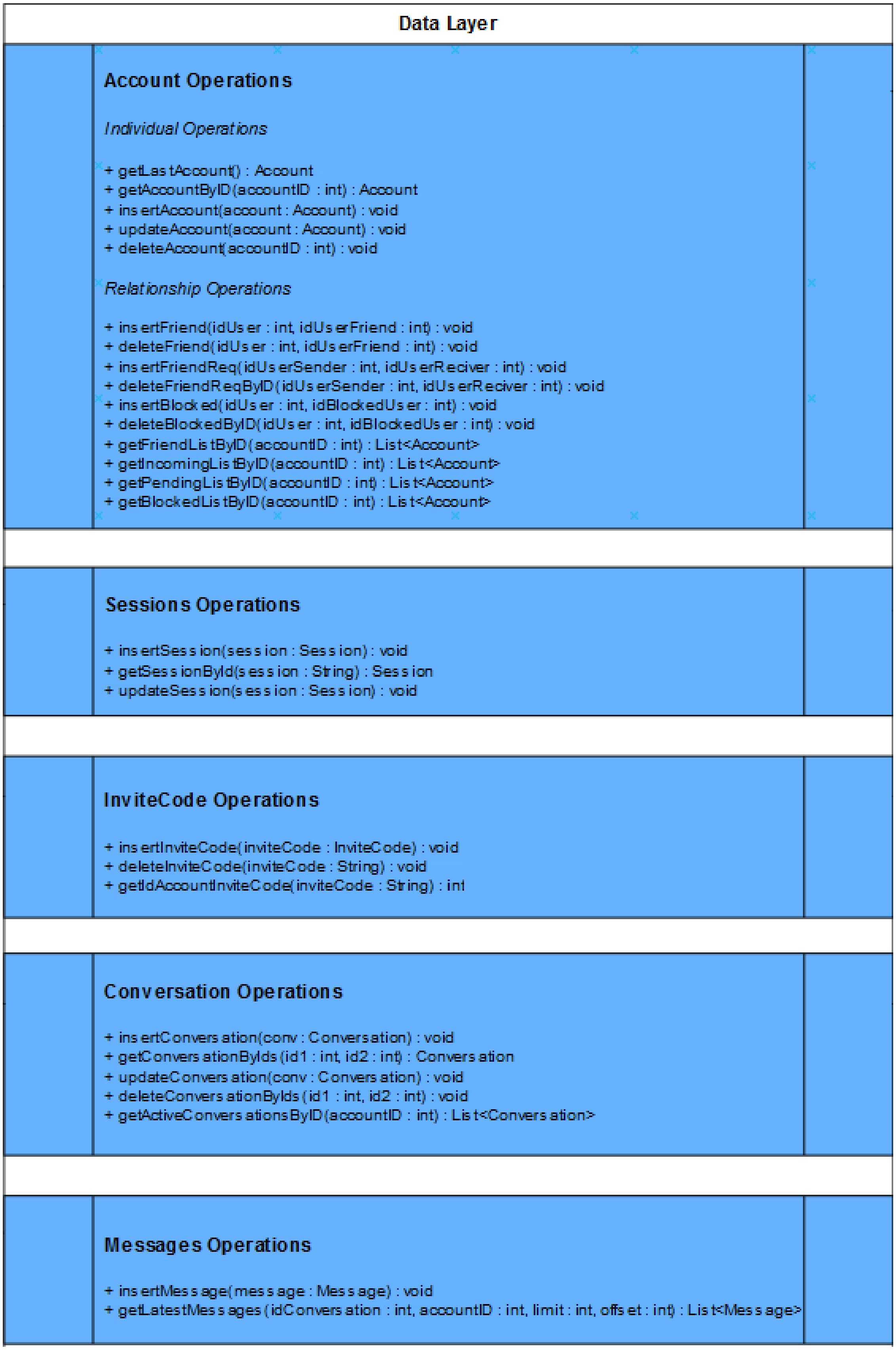
Personal Code: jfg073

Database:



Personal Code: jfg073

Data Layer:



Data Layer

Account Operations

Individual Operations

+ getLastAccount() : Account
+ getAccountByID(accountID : int) : Account
+ insertAccount(account : Account) : void
+ updateAccount(account : Account) : void
+ deleteAccount(accountID : int) : void

Relationship Operations

+ insertFriend(idUser : int, idUserFriend : int) : void
+ deleteFriend(idUser : int, idUserFriend : int) : void
+ insertFriendReq(idUserSender : int, idUserReciver : int) : void
+ deleteFriendReqByID(idUserSender : int, idUserReciver : int) : void
+ insertBlocked(idUser : int, idBlockedUser : int) : void
+ deleteBlockedByID(idUser : int, idBlockedUser : int) : void
+ getFriendListByID(accountID : int) : List<Account>
+ getIncomingListByID(accountID : int) : List<Account>
+ getPendingListByID(accountID : int) : List<Account>
+ getBlockedListByID(accountID : int) : List<Account>

Sessions Operations

+ insertSession(session : Session) : void
+ getSessionById(session : String) : Session
+ updateSession(session : Session) : void

InviteCode Operations

+ insertInviteCode(inviteCode : InviteCode) : void
+ deleteInviteCode(inviteCode : String) : void
+ getIdAccountInviteCode(inviteCode : String) : int

Conversation Operations

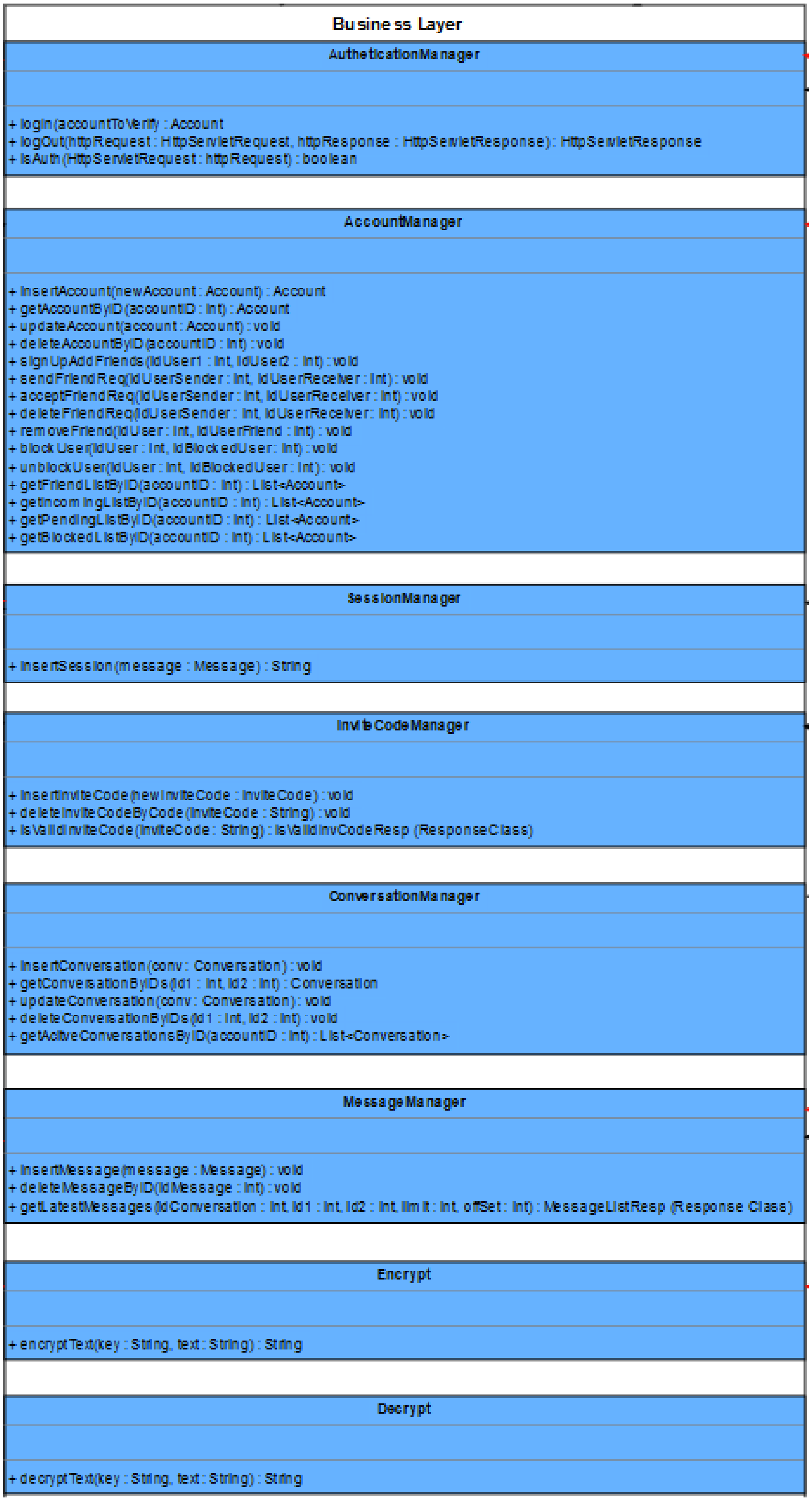
+ insertConversation(conv : Conversation) : void
+ getConversationByIds(id1 : int, id2 : int) : Conversation
+ updateConversation(conv : Conversation) : void
+ deleteConversationByIds(id1 : int, id2 : int) : void
+ getActiveConversationsByID(accountID : int) : List<Conversation>

Messages Operations

+ insertMessage(message : Message) : void
+ getLatestMessages(idConversation : int, accountID : int, limit : int, offset : int) : List<Message>

Personal Code: jfg073

Business Layer:



Business Layer

AutheticationManager

+ login(accountToVerify : Account
+ logOut(httpRequest : HttpServletRequest, httpResponse : HttpServletResponse) : HttpServletResponse
+ isAuth(HttpServletRequest : httpRequest) : boolean

AccountManager

+ insertAccount(newAccount : Account) : Account
+ getAccountByID(accountID : int) : Account
+ updateAccount(account : Account) : void
+ deleteAccountByID(accountID : int) : void
+ signUpAddFriends(idUser1 : int, idUser2 : int) : void
+ sendFriendReq(idUserSender : int, idUserReceiver : int) : void
+ acceptFriendReq(idUserSender : int, idUserReceiver : int) : void
+ deleteFriendReq(idUserSender : int, idUserReceiver : int) : void
+ removeFriend(idUser : int, idUserFriend : int) : void
+ blockUser(idUser : int, idBlockedUser : int) : void
+ unblockUser(idUser : int, idBlockedUser : int) : void
+ getFriendListByID(accountID : int) : List<Account>
+ getIncomingListByID(accountID : int) : List<Account>
+ getPendingListByID(accountID : int) : List<Account>
+ getBlockedListByID(accountID : int) : List<Account>

SessionManager

+ insertSession(message : Message) : String

InviteCodeManager

+ insertInviteCode(newInviteCode : InviteCode) : void
+ deleteInviteCodeByCode(inviteCode : String) : void
+ isValidInviteCode(inviteCode : String) : isValidInvCodeResp (ResponseClass)

ConversationManager

+ insertConversation(conv : Conversation) : void
+ getConversationByIDs(id1 : int, id2 : int) : Conversation
+ updateConversation(conv : Conversation) : void
+ deleteConversationByIDs(id1 : int, id2 : int) : void
+ getActiveConversationsByID(accountID : int) : List<Conversation>

MessageManager

+ insertMessage(message : Message) : void
+ deleteMessageByID(idMessage : int) : void
+ getLatestMessages(idConversation : int, id1 : int, id2 : int, limit : int, offSet : int) : MessageListResp (Response Class)

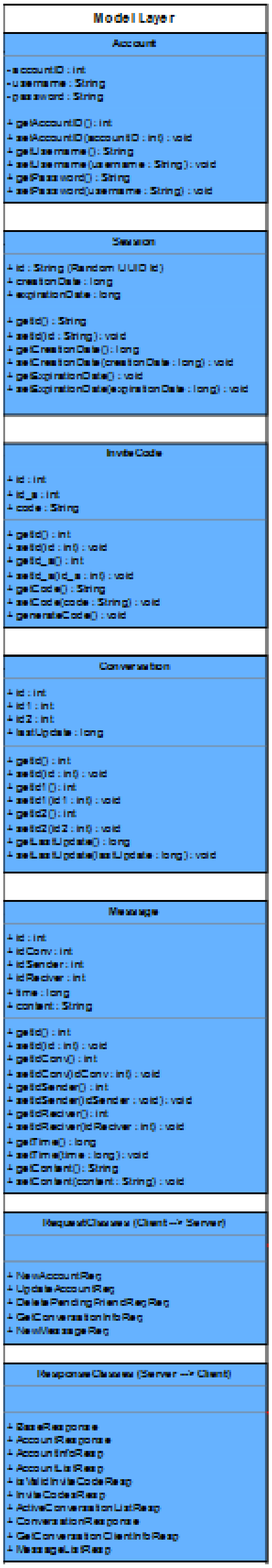
Encrypt

+ encryptText(key : String, text : String) : String

Decrypt

+ decryptText(key : String, text : String) : String

Model Layer:



Model Layer

Account

- accountID : int
- username : String
- password : String

+ getAccountID() : int
+ setAccountID(accountID : int) : void
+ getUsername() : String
+ setUsername(username : String) : void
+ getPassword() : String
+ setPassword(username : String) : void

Session

+ id : String (Random UUID id)
+ creationDate : long
+ expirationDate : long

+ getId() : String
+ setId(id : String) : void
+ getCreationDate() : long
+ setCreationDate(creationDate : long) : void
+ getExpirationDate() : void
+ setExpirationDate(expirationDate : long) : void

InviteCode

+ id : int
+ id_a : int
+ code : String

+ getId() : int
+ setId(id : int) : void
+ getId_a() : int
+ setId_a(id_a : int) : void
+ getCode() : String
+ setCode(code : String) : void
+ generateCode() : void

Conversation

+ id : int
+ id1 : int
+ id2 : int
+ lastUpdate : long

+ getId() : int
+ setId(id : int) : void
+ getId1() : int
+ setId1(id1 : int) : void
+ getId2() : int
+ setId2(id2 : int) : void
+ getLastUpdate() : long
+ setLastUpdate(lastUpdate : long) : void

Message

+ id : int
+ idConv : int
+ idSender : int
+ idReciver : int
+ time : long
+ content : String

+ getId() : int
+ setId(id : int) : void
+ getIdConv() : int
+ setIdConv(idConv : int) : void
+ getIdSender() : int
+ setIdSender(idSender : void) : void
+ getIdReciver() : int
+ setIdReciver(idReciver : int) : void
+ getTime() : long
+ setTime(time : long) : void
+ getContent() : String
+ setContent(content : String) : void

RequestClasses (Client --> Server)

+ NewAccountReq
+ UpdateAccountReq
+ DeletePendingFriendReqReq
+ GetConversationInfoReq
+ NewMessageReq

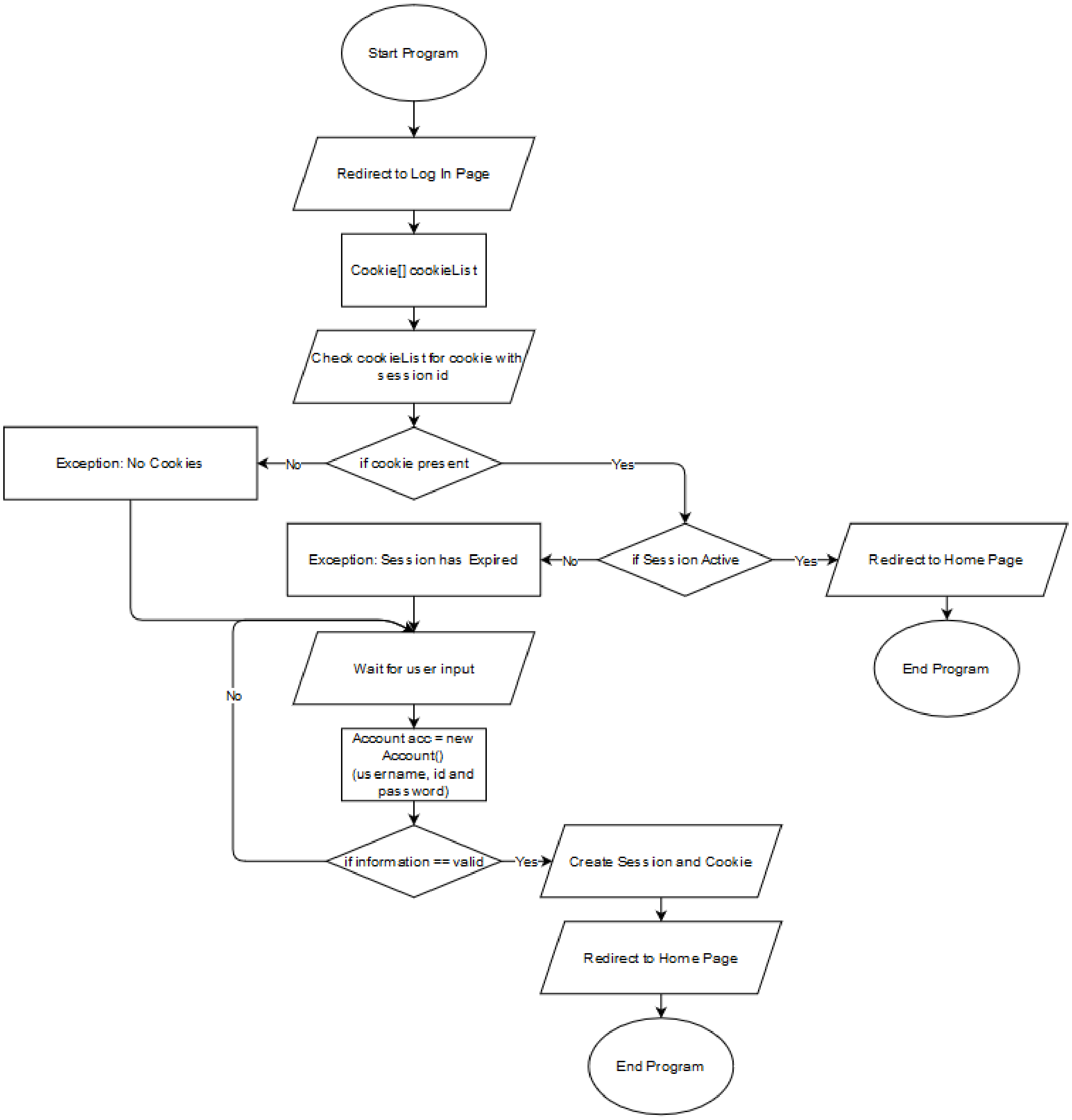
ResponseClasses (Server --> Client)

+ BaseResponse
+ AccountResponse
+ AccountInfoResp
+ AccountListResp
+ isValidInviteCodeResp
+ InviteCodesResp
+ ActiveConversationListResp
+ ConversationResponse
+ GetConversationClientInfoResp
+ MessageListResp

Personal Code: jfg073

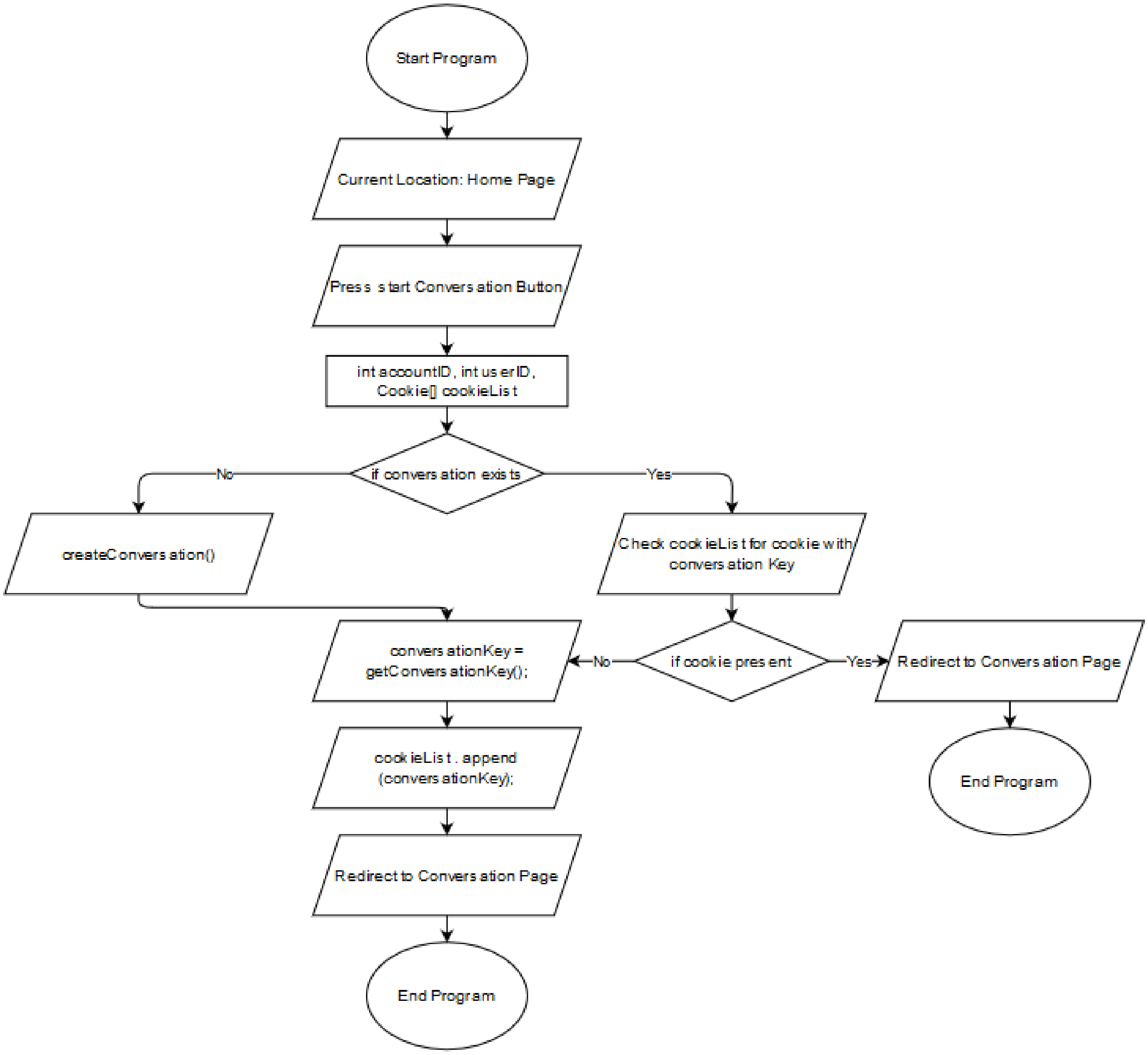
Algorithmic Flow Charts:

Logging in with Cookie and Sessions



Personal Code: jfg073

Starting or entering a conversation



Personal Code: jfg073

Justification of data structures:

The main data structure that will be used for this Web Application will be lists. There will be multiple type of lists that operate with different entities, such as list of Accounts (ex: to be able to see through your contacts), list of Messages, list of InviteCodes (ex: to verify upon a user trying to create an account whether he introduced a correct invite code from a user within the application or not), list of Conversations (ex: to see active conversations of a user) etc.

Most of the instances where a list will be implemented, I will use the Java ArrayList class, however for certain requests from the client such as requesting their messages from a certain conversation, I will created my own personal type of List, where I pass onto the client from the server multiple information such as the list of messages between two accounts (the ones that are taking part in the conversation), and the two accounts in question, in order to send additional useful information to the client such as the username and id of the person who sent a message.

Test Plan:

No	Action to test	Method of testing - result	Success Criteria	Tested	Result
1	Create Account	Enter a desired username, valid password (min. Password Security Criteria) with confirmation and valid invite code.	1		
2	Log In	Enter correct details of an account and enter application	2		
3	Add Friend	Enter a valid Account id in the add friend modal and accept friend request from secondary account.	3		
4	Remove Friend	Click the option button of a user and press remove friend button	3		
5	Block User	Click the option button of a user and press block button. Observe the blocked user in the blocked tab	3		
6	Search User	Enter	4		
7	Change account username	On the profile page, enter a username and the valid password of the account to confirm identity.	5		
8	Change account password	On the profile page, enter new password, confirm it, and enter the	5		

		valid password of the account to confirm identity.			
9	Start Conversation	Press the start conversation button of a user in the account contact list	6		
10	Send Message	Write a messages and press enter or send message button	7		
11	Receive Message	Send message with one account and receive message on the primary tab	7		
12	Look in message history	Scroll upwards in conversation with a lot of messages	8		
13	Delete an account		9		

Word Count: 222

Bibliography

Draw.io. (n.d.). *Draw.io*. Retrieved from Draw.io: https://app.diagrams.net/#