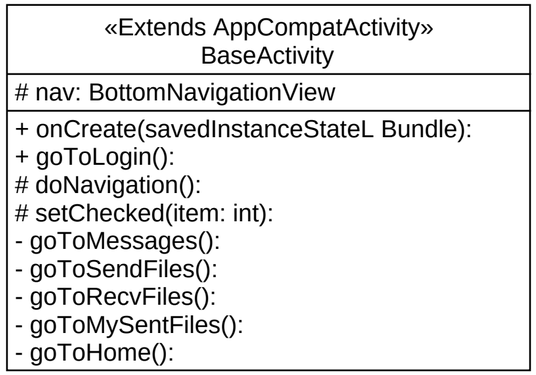
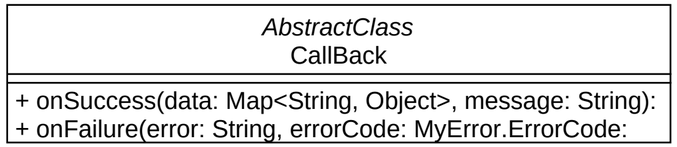
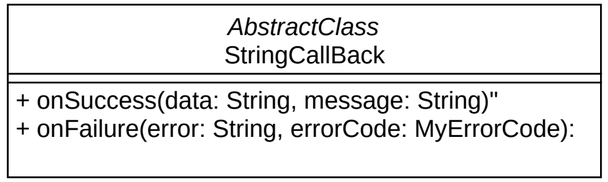
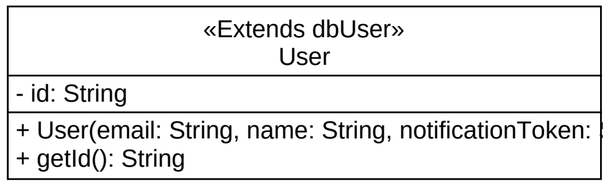
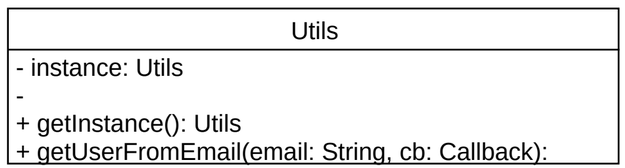
3Klein_release_1_general.pdf

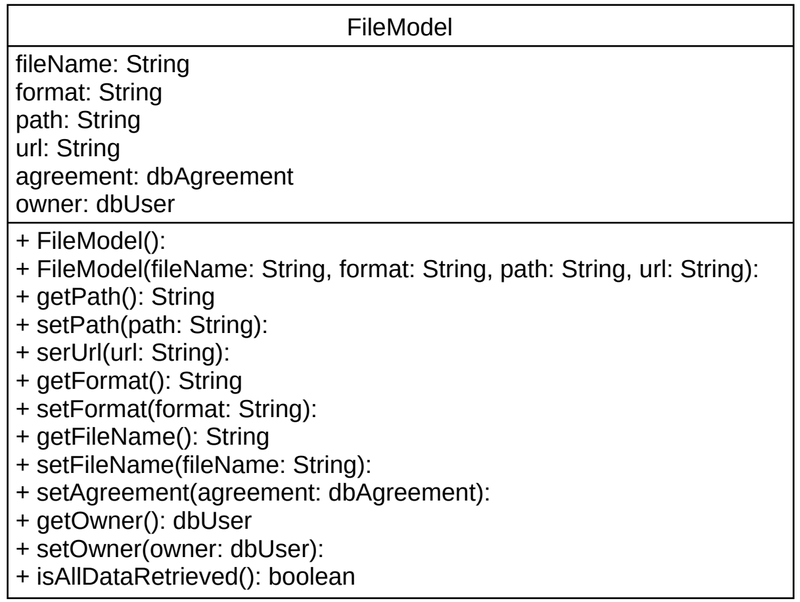

«Extends AppCompatActivity»
BaseActivity
nav: BottomNavigationView
+ onCreate(savedInstanceStateL Bundle):
+ goToLogin():
# doNavigation():
# setChecked(item: int):
- goToMessages():
- goToSendFiles():
- goToRecvFiles():
- goToMySentFiles():
- goToHome():

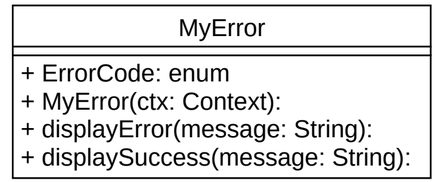

AbstractClass
CallBack
+ onSuccess(data: Map<String, Object>, message: String):
+ onFailure(error: String, errorCode: MyError.ErrorCode:

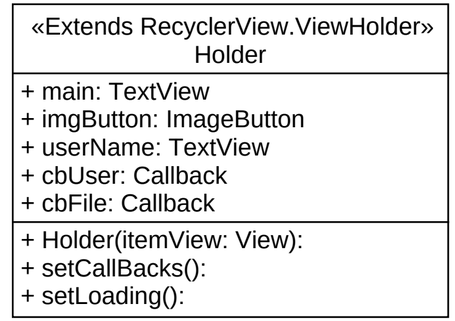

AbstractClass
StringCallBack
+ onSuccess(data: String, message: String)"
+ onFailure(error: String, errorCode: MyErrorCode):

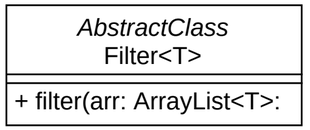

«Extends dbUser»
User
- id: String
+ User(email: String, name: String, notificationToken:
+ getId(): String

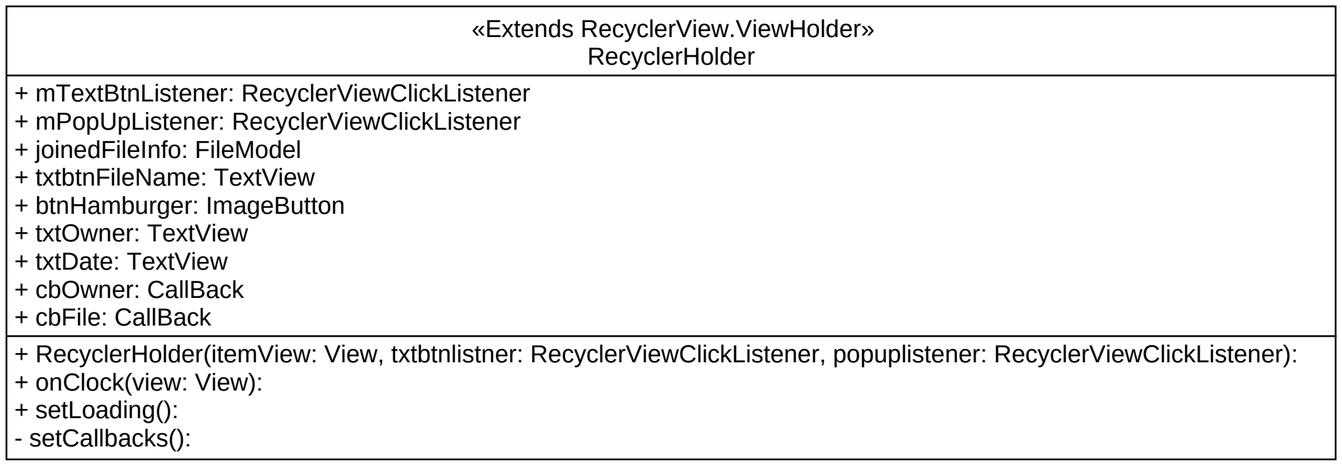

Utils
- instance: Utils
-
+ getInstance(): Utils
+ getUserFromEmail(email: String, cb: Callback):

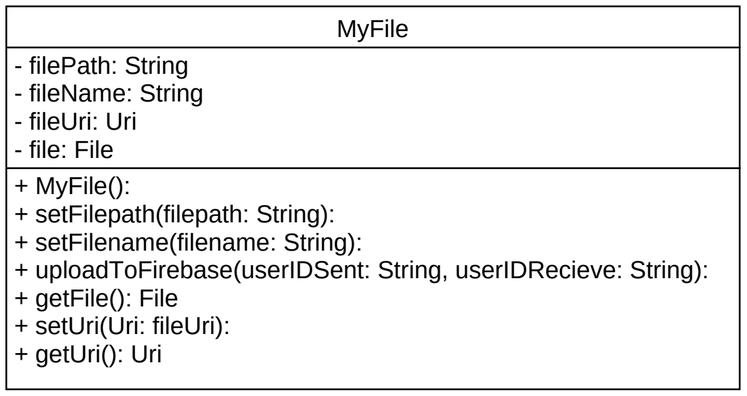

FileModel
fileName: String
format: String
path: String
url: String
agreement: dbAgreement
owner: dbUser
+ FileModel():
+ FileModel(fileName: String, format: String, path: String, url: String):
+ getPath(): String
+ setPath(path: String):
+ serUrl(url: String):
+ getFormat(): String
+ setFormat(format: String):
+ getFileName(): String
+ setFileName(fileName: String):
+ setAgreement(agreement: dbAgreement):
+ getOwner(): dbUser
+ setOwner(owner: dbUser):
+ isAllDataRetrieved(): boolean

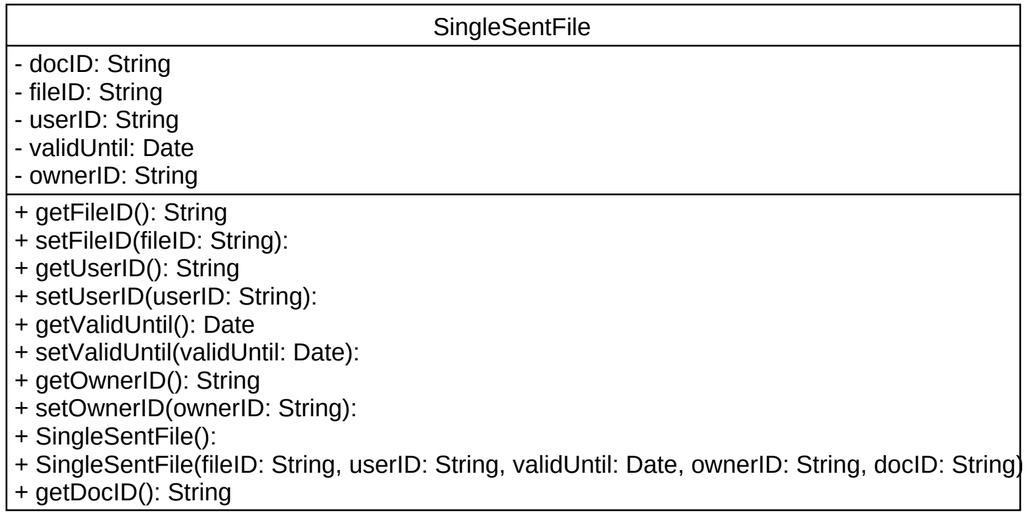

MyError
+ ErrorCode: enum
+ MyError(ctx: Context):
+ displayError(message: String):
+ displaySuccess(message: String):


«Extends RecyclerView.ViewHolder»
Holder
+ main: TextView
+ imgButton: ImageButton
+ userName: TextView
+ cbUser: Callback
+ cbFile: Callback
+ Holder(itemView: View):
+ setCallBacks():
+ setLoading():


AbstractClass
Filter<T>
+ filter(arr: ArrayList<T>:


«Extends RecyclerView.ViewHolder»
RecyclerHolder
+ mTextBtnListener: RecyclerViewClickListener
+ mPopUpListener: RecyclerViewClickListener
+ joinedFileInfo: FileModel
+ txtbtnFileName: TextView
+ btnHamburger: ImageButton
+ txtOwner: TextView
+ txtDate: TextView
+ cbOwner: CallBack
+ cbFile: CallBack
+ RecyclerHolder(itemView: View, txtbtnlistner: RecyclerViewClickListener, popuplistener: RecyclerViewClickListener):
+ onClock(view: View):
+ setLoading():
- setCallbacks():


MyFile
- filePath: String
- fileName: String
- fileUri: Uri
- file: File
+ MyFile():
+ setFilepath(filepath: String):
+ setFilename(filename: String):
+ uploadToFirebase(userIDSent: String, userIDRecieve: String):
+ getFile(): File
+ setUri(Uri: fileUri):
+ getUri(): Uri


SingleSentFile
- docID: String
- fileID: String
- userID: String
- validUntil: Date
- ownerID: String
+ getFileID(): String
+ setFileID(fileID: String):
+ getUserID(): String
+ setUserID(userID: String):
+ getValidUntil(): Date
+ setValidUntil(validUntil: Date):
+ getOwnerID(): String
+ setOwnerID(ownerID: String):
+ SingleSentFile():
+ SingleSentFile(fileID: String, userID: String, validUntil: Date, ownerID: String, docID: String)
+ getDocID(): String

3Klein_release_1_activities.pdf

«Extends AppCompatActivity»
LoginActivity

- signInButton: SignInButton
- mGoogleSignInClient: GoogleSignInClient
- TAG: String
- mAuth: FirebaseAuth
- btnSignOut: Button
- RC_SIGN_IN: int

+ signIn():
# onCreate(savedInstances Bundle):
# onActivityResult(requestCode: int, data: Intent):
- handleSignInResult(completedTask: Task<GoogleSignInAccount>):
- FirebaseGoogleAuth(acct: GoogleSignInAccount):
- updateUI(fUser: FirebaseUser):

«Extends BaseActivity»
SendFileActivity

- «final» FILE_RESULT_SUCCESS: int
- «final» FILE_REQUEST_CODE: int
- «final» LOG_TAG: String
- storage: FirebaseStorage
- auth: FirebaseAuth
- btnChooseFile: Button
- btnChooseRecievedFile: Button
- btnSend: Button
- txtEmail: EditText
- txtFilename: EditText
- lblFilename: TextView
- progressBar: ProgressBar
- errorHandler: MyError
- file: MyFile
- isSendFileLocalStorage: boolean
- fileToSend: dbFile
- userThatSentFile: dbUser
- determineCurrentUser()

«Implements View.OnClickListener»
chooseReceivedFile

+ «override» onClick(view: View):
- makeDialogBox():

«Implements View.onClickListener»
«private» sendFile

- filePathFirebase: String
- filename: String
- user: FirebaseUser
- userToReceiveID: String
- fileRef: StorageReference
-
+ «override» onSuccess(data: Map<String, Object>, message: String):
+ «override» onFailure(error: String, errorCode: MyError.ErrorCode):
- afterGetEmail(data: Map<String, Object>, message: String):
- afterUploadFile(taskSnapShot: UploadTask.TaskSnapshot, key: Strin
- afterUpdateDB():
- afterFailGetEmail(error: String, errorCode: MyError.ErrorCode):
- afterFailUpload(exception: Exception):
- afterFailUpdateDB(e: Exception):
- cleanUp():

«Implements TextWatcher»
FileNameChange

+ «override» beforeTextChanged(charSequence: CharSequence, i: int
+ «override» onTextChanged(charSequence: CharSequence, i: int, i1:

«Extends AppCompatActivity»
MainActivity

+ onCreate(savedInstanceState: Bundle, persistentState: PersistableBundle):

SayHello

+ hello(name: String): String

«Extends BaseActivity»
MySentFiles

«final» USER_COLLECTION_NAME: String
«final» FILE_COLLECTION_NAME: String
- LOG_TAG: String
- currentFileSelected: SingleSentFile
- errorHandler: MyError
+ btnApproved: Button
+ btnAll: Button
+ btnPending:Button
+ adapter: MyRecyclerViewAdapter
- details: HashMap<String, HashMap<String, Object»>()

getAsync(collectionName: String, docID: String, cb: Callback):
+ «override» onCreate(savedInstanceState: Bundle):
+ recyclerViewStuff():
+ showPopup(v: View):
+ colourButtons(clicked: Button):
+ doButtons():
+ changePermissions(d: Date, file: SingleSentFile):
+ revoke(file: SingleSentFile):
+ approveTilTomorrow(file: SingleSentFile):
+ approveIndefinitely(file: SingleSentFile:

«Extends RecyclerView.Adapter<Holder»
MyRecyclerViewAdapter

+ MyRecyclerVIewAdapter(data: QuerySnapshot)
-«final» LOG_TAG: String
- mDataSet: ArrayList<SingleSentFile>
- filtered: ArrayList<SingleSentFile>
- currentFilter: Filter<SingleSentFile>
+ «override» onCreateViewHolder(group: ViewGroup, i: int): Holder
+ «override» onBindViewHolder(holder: Holderm position: int):
+ «ovveride» getItemCount(): int
+ filter(filter: Filter<SingleSentFile>):
+ filterWithCurrentFilter():

«Extends BaseActivity»
RecieveFilesActivity

+ «final» USER_COLLECTION_NAME: String
+ «final» FILE_COLLECTION_NAME: String
+ btnSort: Button
+ editFIlter: EditText
+ db: FirebaseFirestore
+ myAdapter: RecyclerAdapter
+ user: FirebaseUser
+ mFirebaseFunctions: FirebaseFunctions
+ «final» LOG_TAG: String
+ «override» onCreate(savedInstanceState Bundle):
+ setEvents():
+ setElements():
- determineCurrentUser():
getAsync(collectionName: String, docID: String, cb: Callback):
- setUpFireStore():
- setUpRV():
- sendNotificationRequestingPermission(fName: String, ownerToken: String):
- showHamburgerPopUp(v: View, position: int, file: FileModel)
getAsync(collectionName: String, docID: String, cb: Callback):
- setUpFireStore():
- setUpRV():
- showHamburgerPopUp(v: View, position: int, file: FileModel)

«Implements View.OnClickListener»
Sorter

+ «override» onClick(v: View):
+ showSortPopup(v: View):

«Implements TextWatcher»
Filter

+ «override» beforeTextChanged(charSequence: CharSequence, i: int, i1: int, i2:
+ «override» onTextChanged(charSequence: CharSequence, i: int, i1: int, i2: int):
+ «override» afterTextChanged(editable: Editable):

«Extends RecyclerView.Adapter<RecyclerHolder> »
RecyclerAdapter

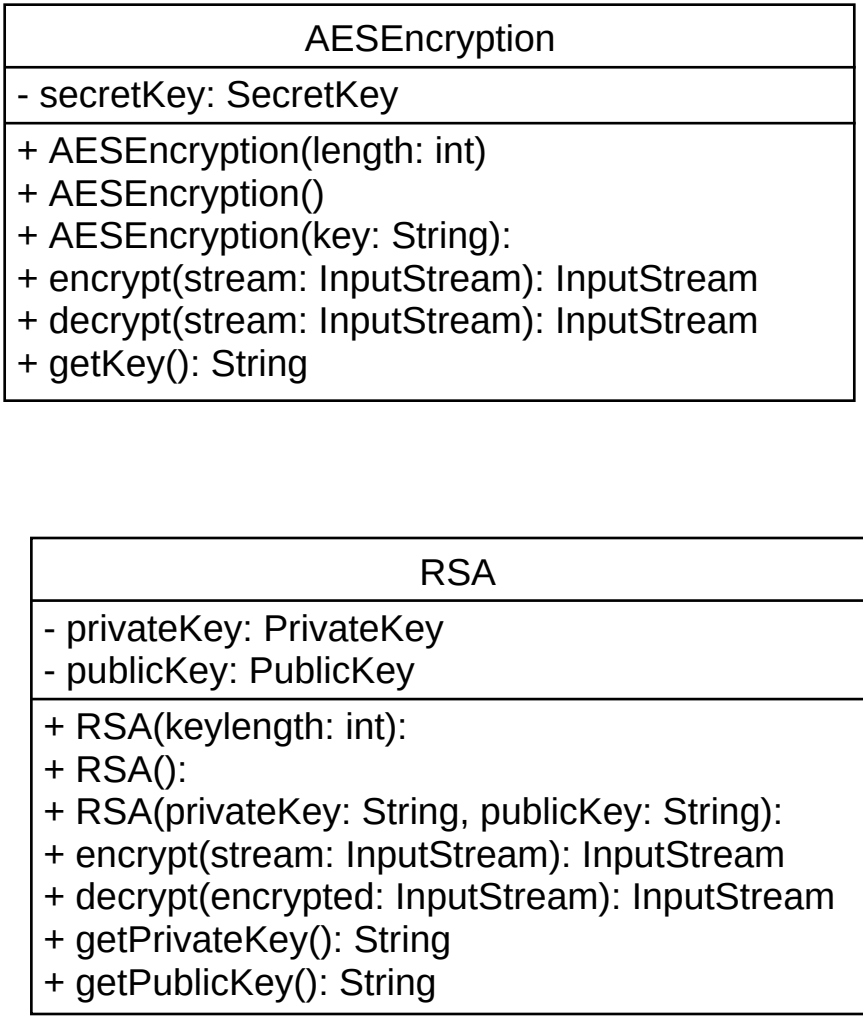
+ «override» onCreateViewHolder(group: ViewGroup, i: int): Holder
+ «override» onBindViewHolder(holder: Holderm position: int):

«Extends RecyckerView.Adapter<RecyclerHolder»
RecyclerAdapter

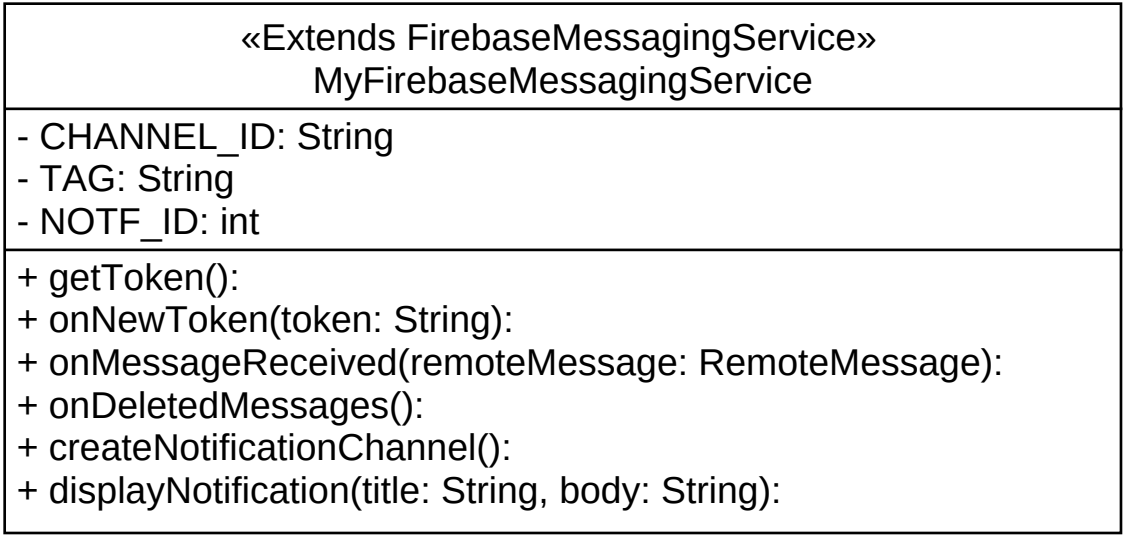
(Variables)

(Functions)

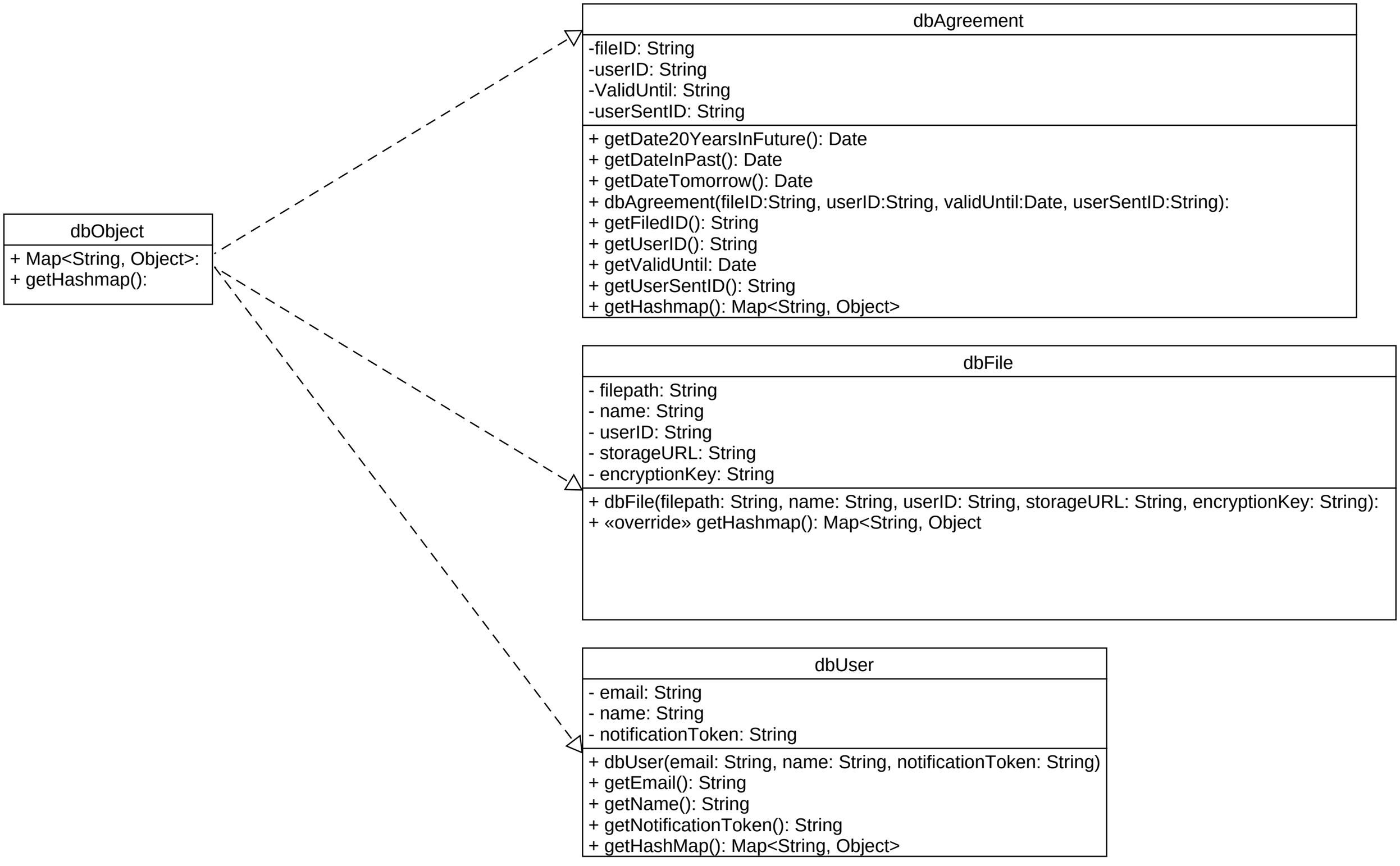
3Klein_release_1_Encryptions.pdf


AESEncryption
- secretKey: SecretKey
+ AESEncryption(length: int)
+ AESEncryption()
+ AESEncryption(key: String):
+ encrypt(stream: InputStream): InputStream
+ decrypt(stream: InputStream): InputStream
+ getKey(): String
RSA
- privateKey: PrivateKey
- publicKey: PublicKey
+ RSA(keylength: int):
+ RSA():
+ RSA(privateKey: String, publicKey: String):
+ encrypt(stream: InputStream): InputStream
+ decrypt(encrypted: InputStream): InputStream
+ getPrivateKey(): String
+ getPublicKey(): String

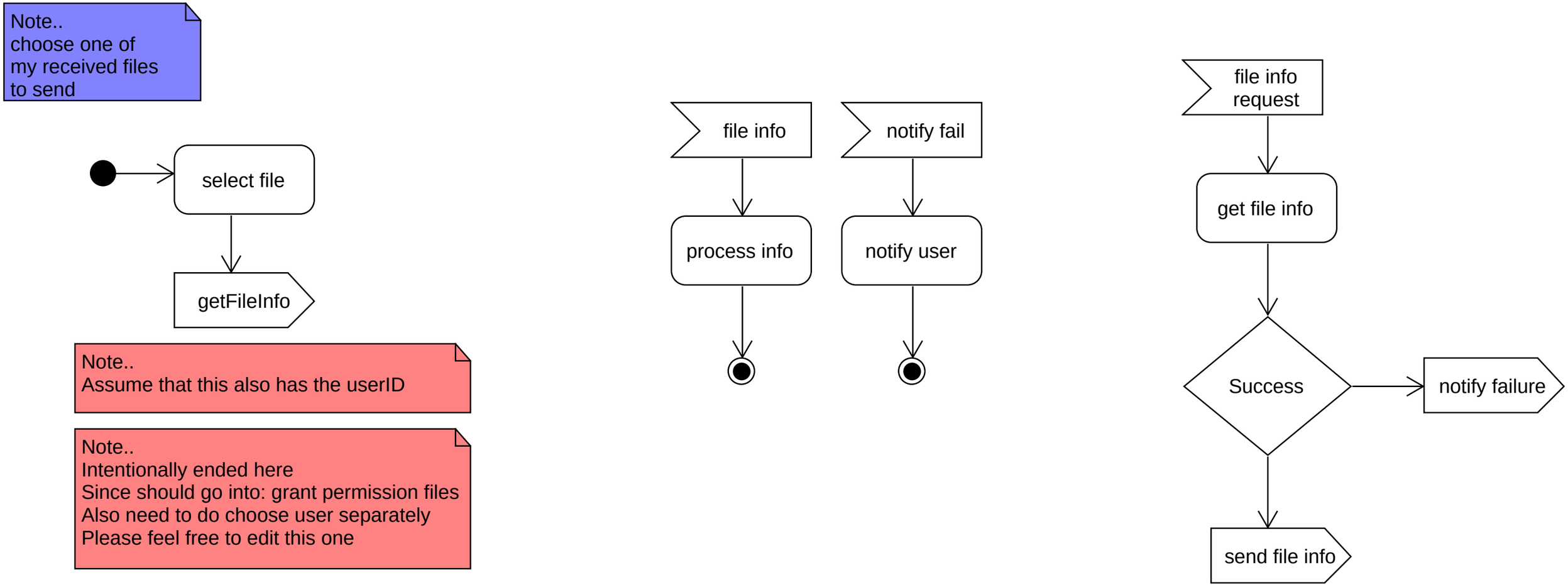
3Klein_release_1_MyFirebaseMessagingService.pdf


«Extends FirebaseMessagingService»
MyFirebaseMessagingService
- CHANNEL_ID: String
- TAG: String
- NOTF_ID: int
+ getToken():
+ onNewToken(token: String):
+ onMessageReceived(remoteMessage: RemoteMessage):
+ onDeletedMessages():
+ createNotificationChannel():
+ displayNotification(title: String, body: String):

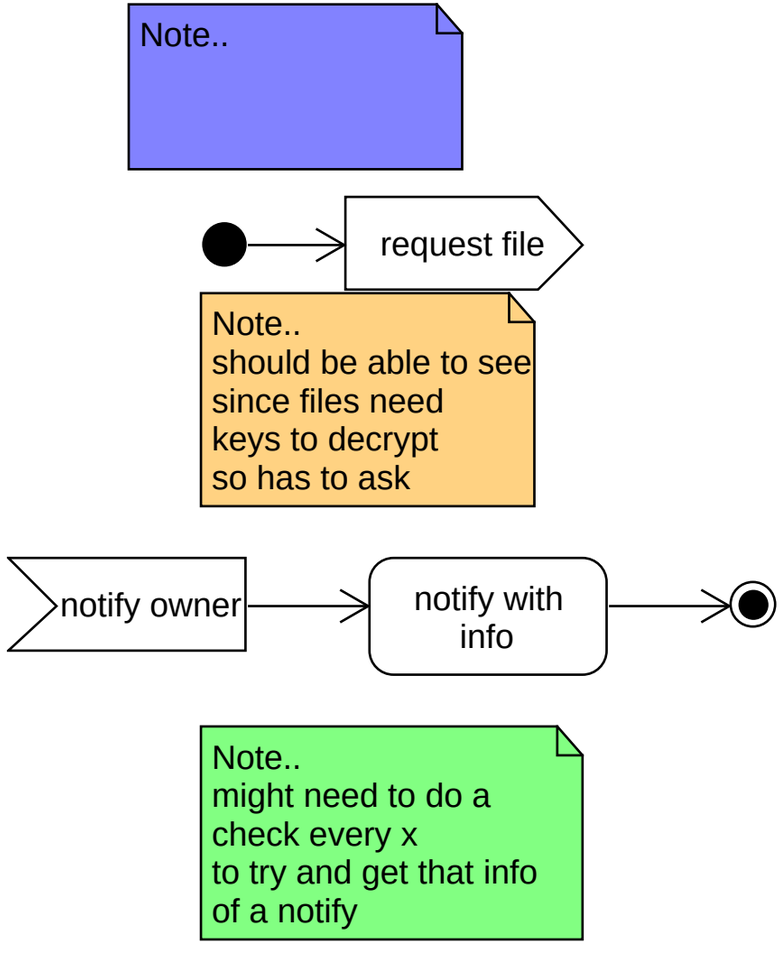
3Klein_release_1_databases.pdf

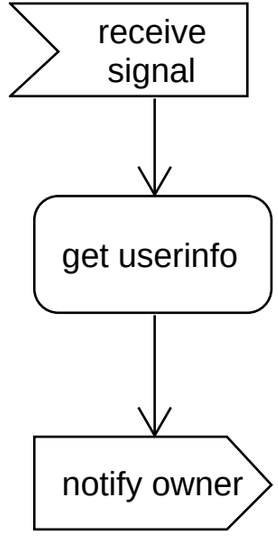

dbObject
+ Map<String, Object>:
+ getHashmap():
dbAgreement
-fileID: String
-userID: String
-ValidUntil: String
-userSentID: String
+ getDate20YearsInFuture(): Date
+ getDateInPast(): Date
+ getDateTomorrow(): Date
+ dbAgreement(fileID:String, userID:String, validUntil:Date, userSentID:String):
+ getFiledID(): String
+ getUserID(): String
+ getValidUntil: Date
+ getUserSentID(): String
+ getHashmap(): Map<String, Object>
dbFile
- filepath: String
- name: String
- userID: String
- storageURL: String
- encryptionKey: String
+ dbFile(filepath: String, name: String, userID: String, storageURL: String, encryptionKey: String):
+ «override» getHashmap(): Map<String, Object
dbUser
- email: String
- name: String
- notificationToken: String
+ dbUser(email: String, name: String, notificationToken: String)
+ getEmail(): String
+ getName(): String
+ getNotificationToken(): String
+ getHashMap(): Map<String, Object>

choose one of my received files to send.pdf

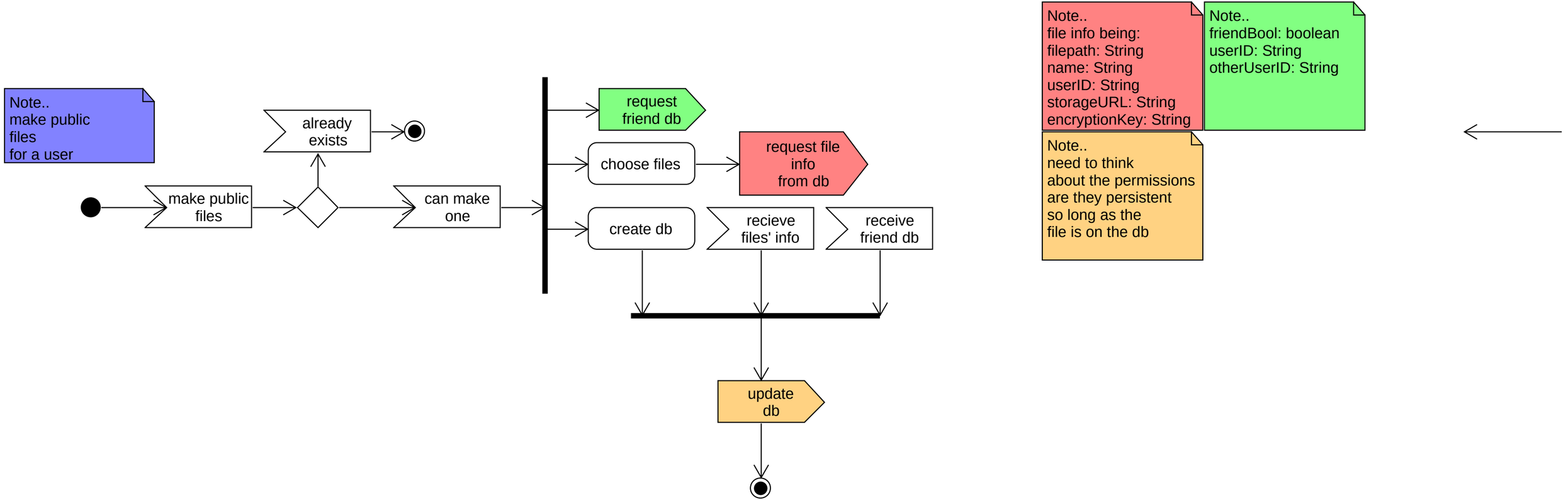

Note..
choose one of
my received files
to send
select file
getFileInfo
Note..
Assume that this also has the userID
Note..
Intentionally ended here
Since should go into: grant permission files
Also need to do choose user separately
Please feel free to edit this one
file info
process info
notify fail
notify user
file info
request
get file info
Success
notify failure
send file info

requested my public files.pdf

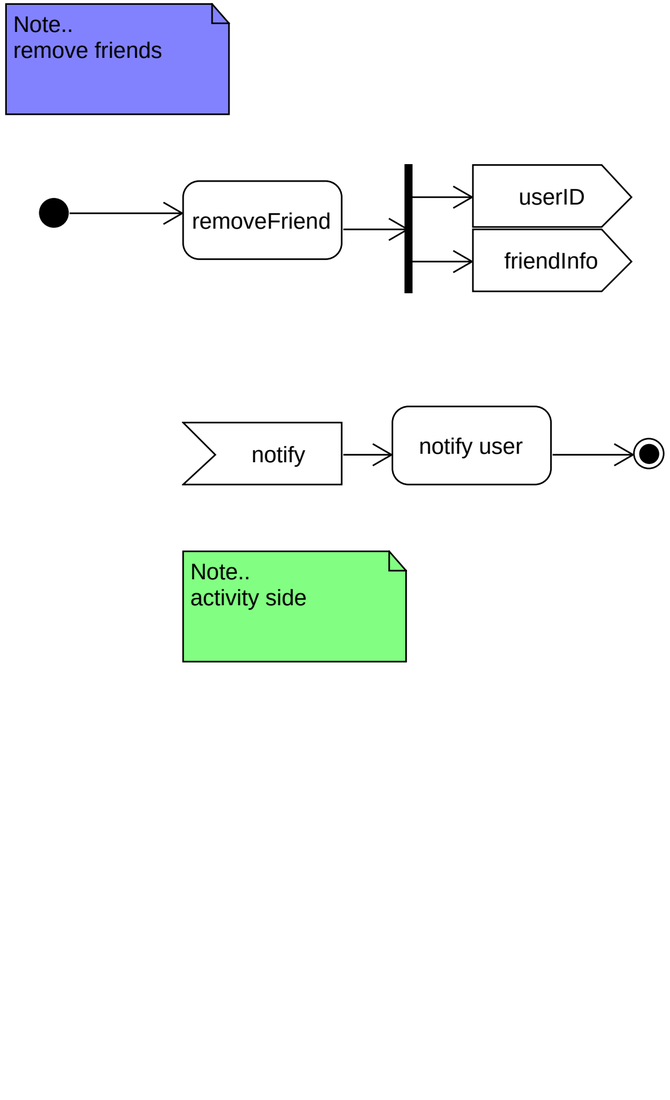

Note..
request file
Note..
should be able to see
since files need
keys to decrypt
so has to ask
notify owner
notify with
info
Note..
might need to do a
check every x
to try and get that info
of a notify


receive
signal
get userinfo
notify owner

make public files.pdf

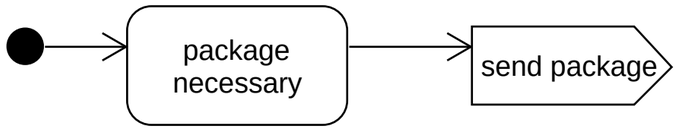

Note..
make public
files
for a user
already
exists
make public
files
can make
one
request
friend db
choose files
request file
info
from db
create db
recieve
files' info
receive
friend db
update
db
Note..
file info being:
filepath: String
name: String
userID: String
storageURL: String
encryptionKey: String
Note..
friendBool: boolean
userID: String
otherUserID: String
Note..
need to think
about the permissions
are they persistent
so long as the
file is on the db

remove friends.pdf


Note..
remove friends
removeFriend
userID
friendInfo
notify
notify user
Note..
activity side
state
userID
friendInfo
Note..
db side
state
are friends
notify not
friends
delete friend
Success
notify fail
notify
success

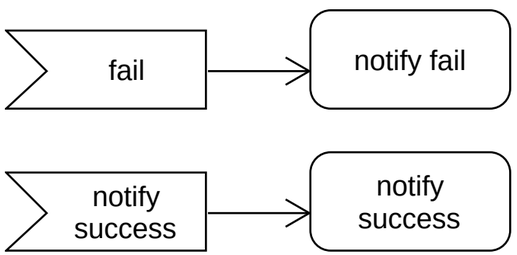
uploading a file only upload encrypted part.pdf

Note..
When uploading a file, upload
only the encrypted part

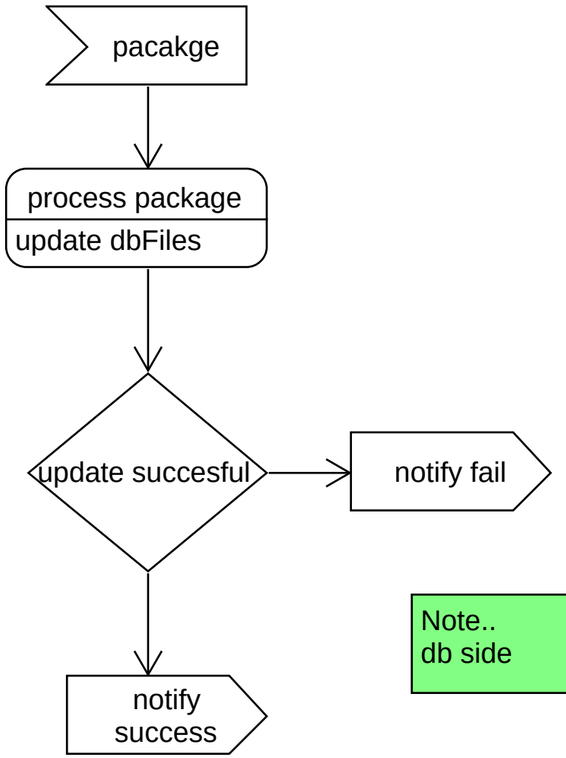


Note..
the package should just be
your userID, encryptedDoc and storageKey.

This should be after you have encrypted it
so you have done the necessaying RSA/AES
Locally



Note..
user side

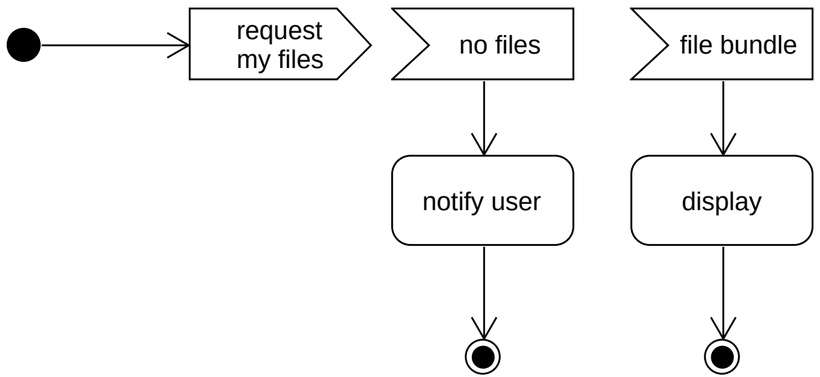


Note..
db side

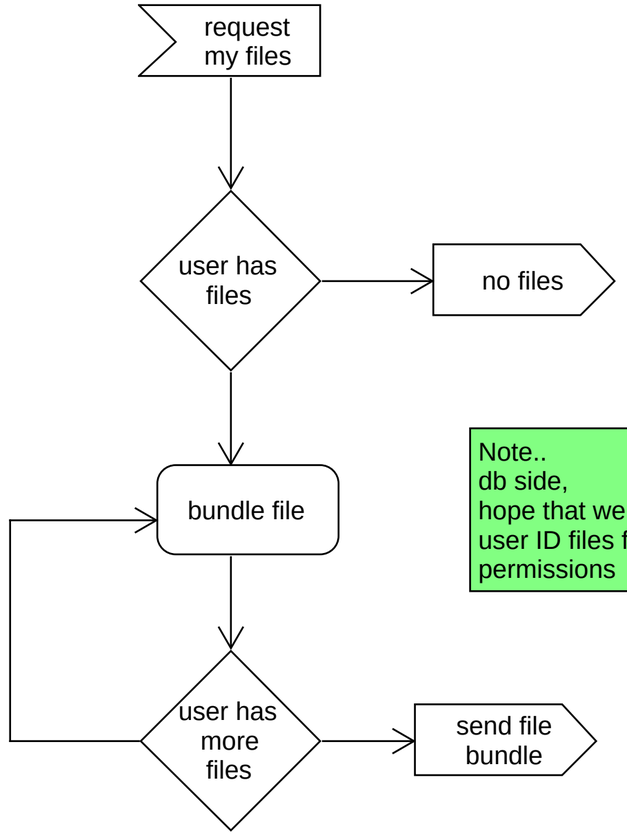
«Michael»
I combined this with the uploading of the encryption key
Please if you feel they should be separated keep them
Separated. You will know what is better to do,
I just assume that it is possible to send both at once.

view all my sent files and include usernames of recipients.pdf

Note..
View all of my files that have been sent to others - including usernames of recipients

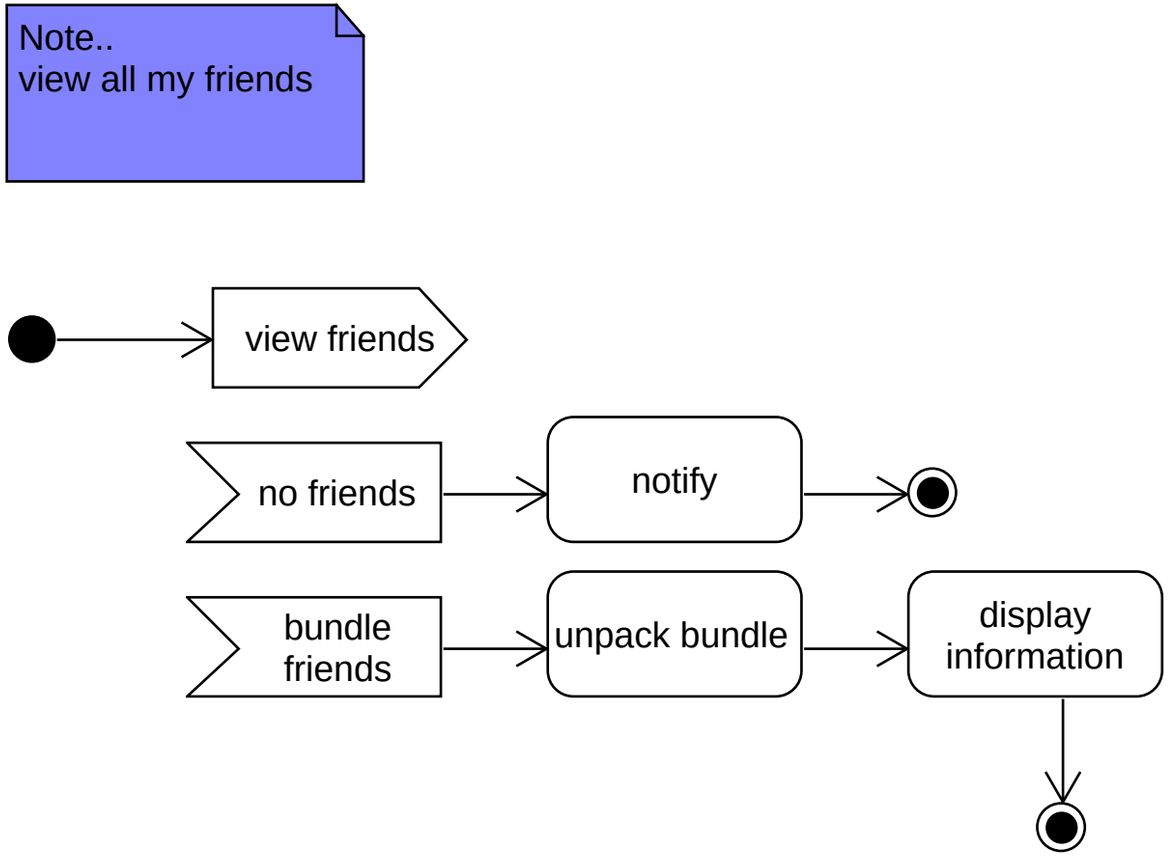


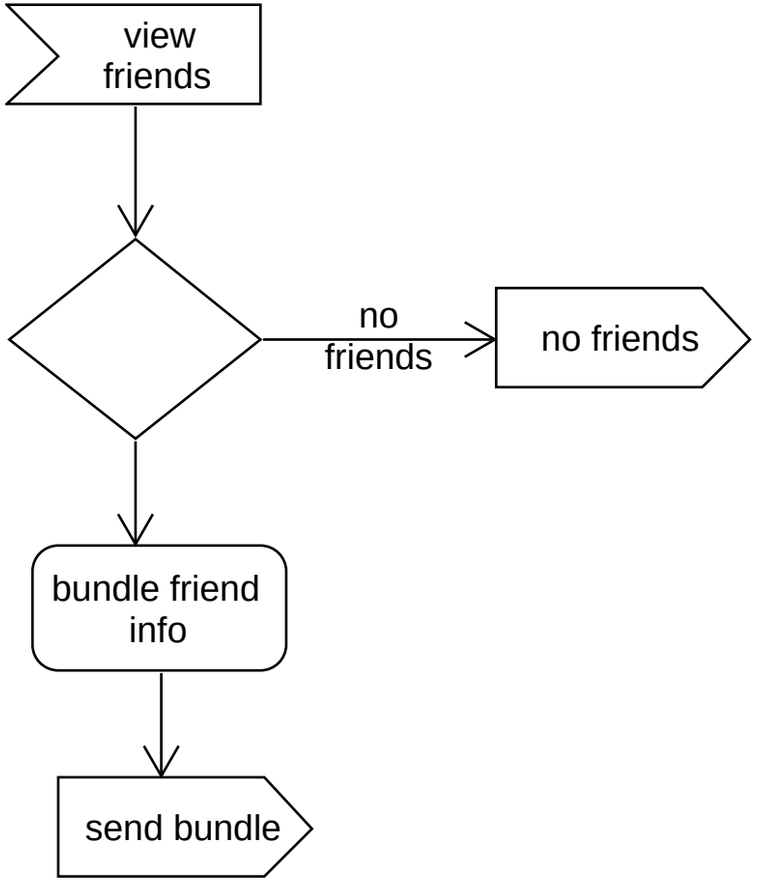
Note..
user side



Note..
db side,
hope that we just go through user ID files for those with permissions

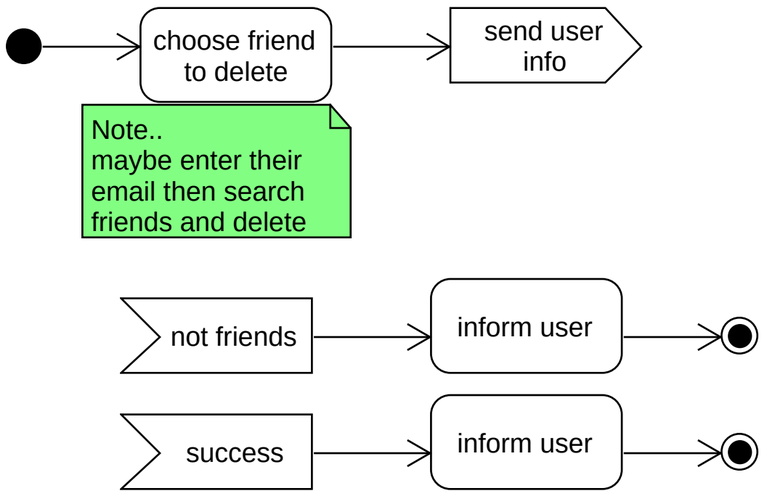
view all my friends.pdf

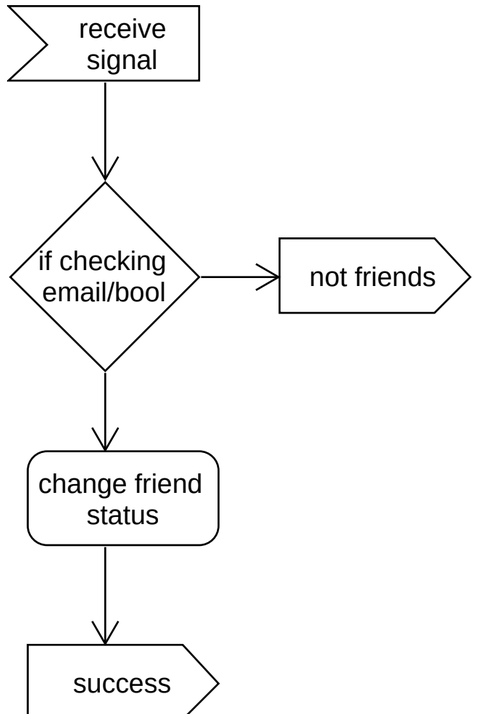

Note..
view all my friends
view friends
no friends
notify
bundle
friends
unpack bundle
display
information


view
friends
no
friends
no friends
bundle friend
info
send bundle

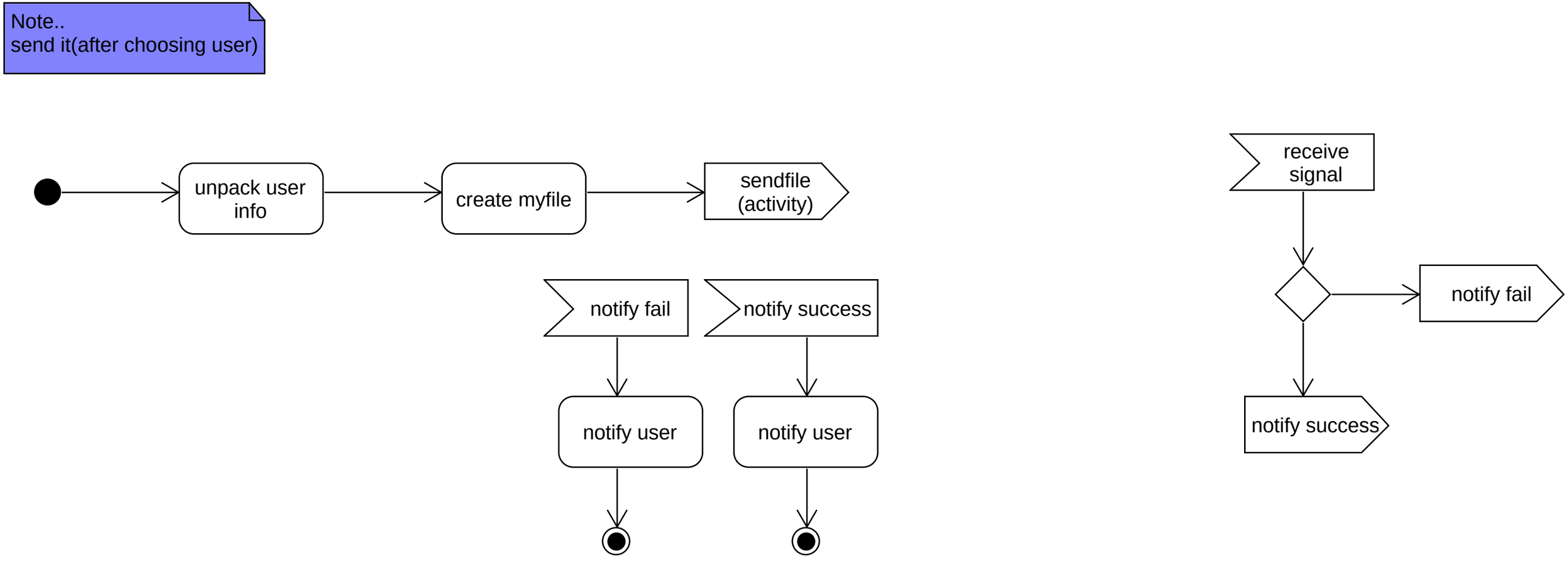
delete friends.pdf

Note..
delete a friend

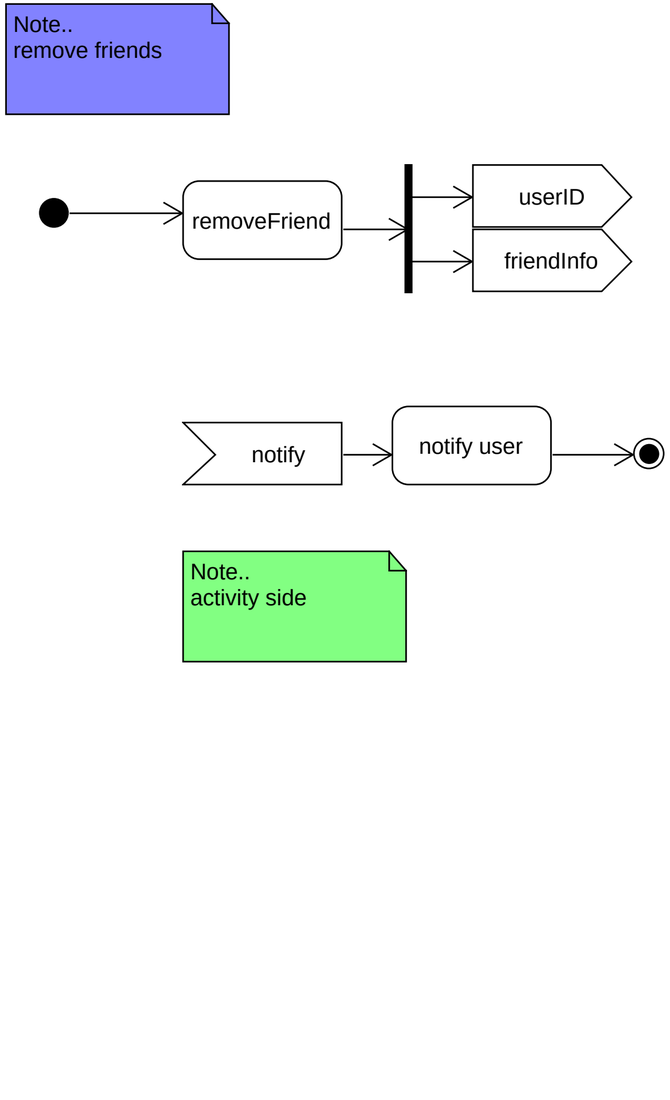




send it (after choosing user).pdf

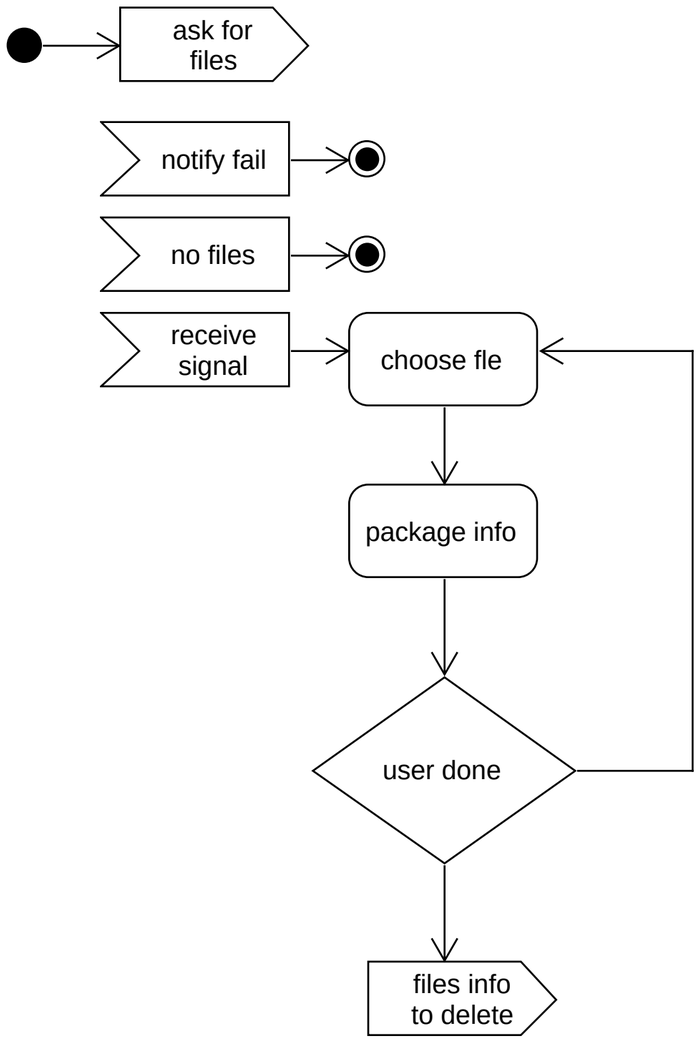

Note..
send it(after choosing user)
unpack user info
create myfile
sendfile (activity)
notify fail
notify success
notify user
notify user
receive signal
notify fail
notify success

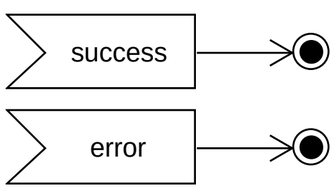
add friends.pdf

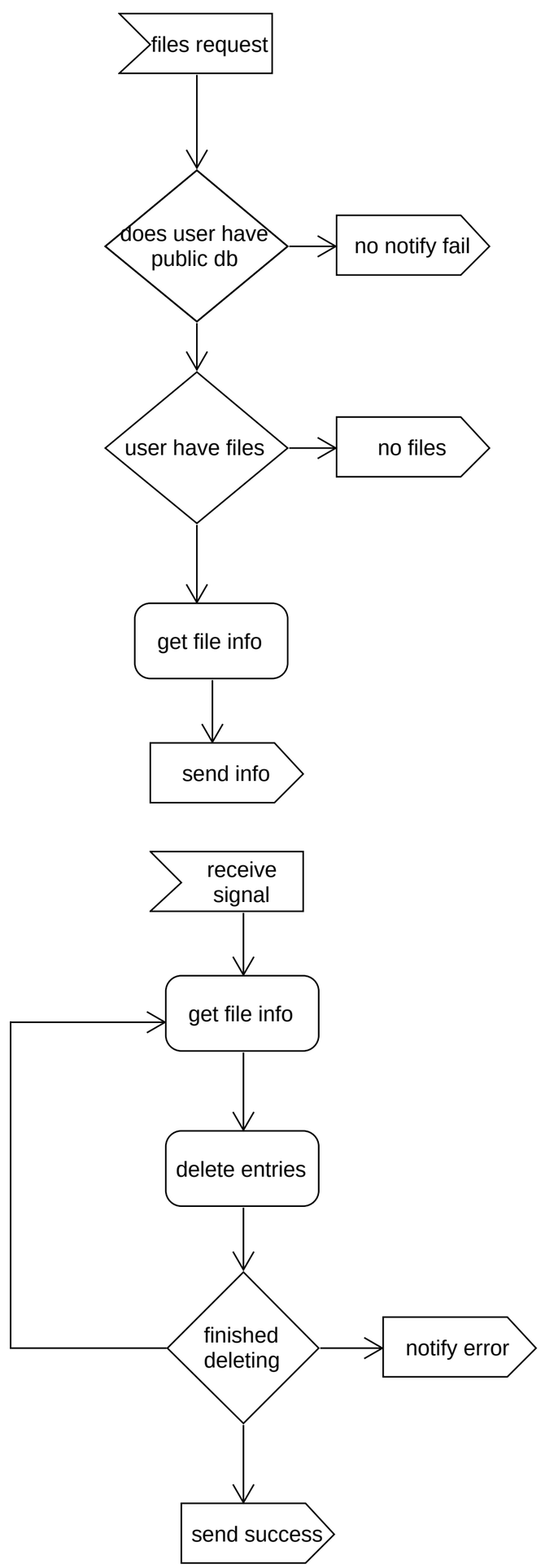

Note..
remove friends
removeFriend
userID
friendInfo
notify
notify user
Note..
activity side
state
userID
friendInfo
Note..
db side
state
are friends
notify not
friends
delete friend
Success
notify fail
notify
success

Remove files from public space.pdf

Note..

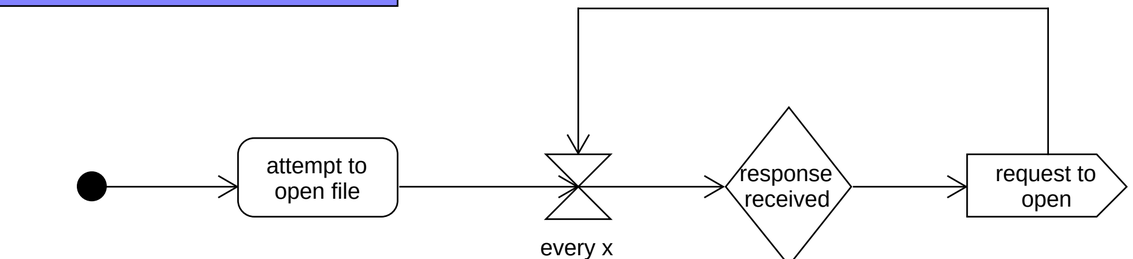

ask for files
notify fail
no files
receive signal
choose fle
package info
user done
files info to delete

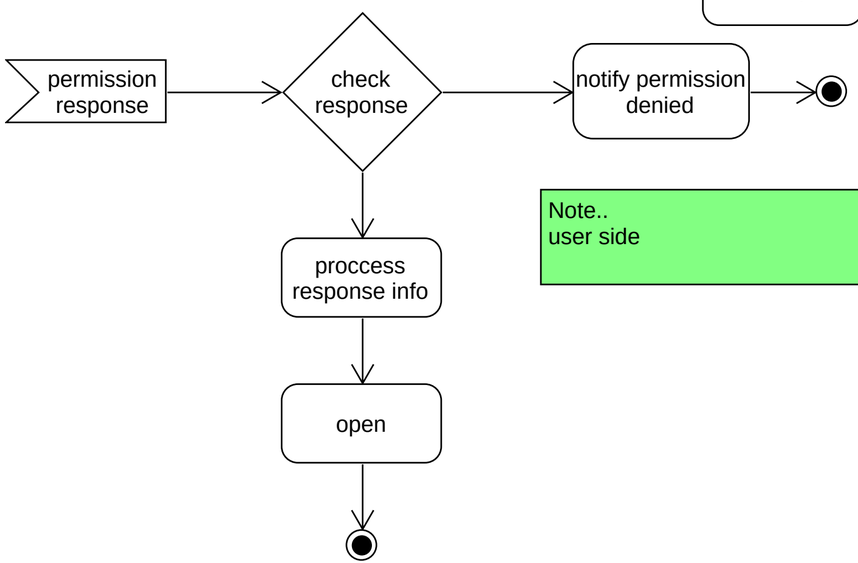

success
error


files request
does user have public db
no notify fail
user have files
no files
get file info
send info
receive signal
get file info
delete entries
finished deleting
notify error
send success

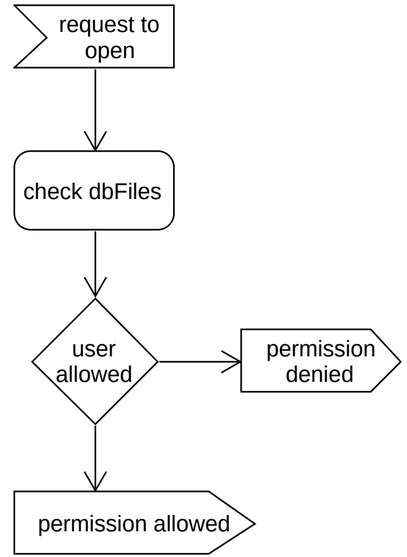
request permission to original owner.pdf

Note..
When opening a file, the permission request needs to go to the original owner.





Note..
user side

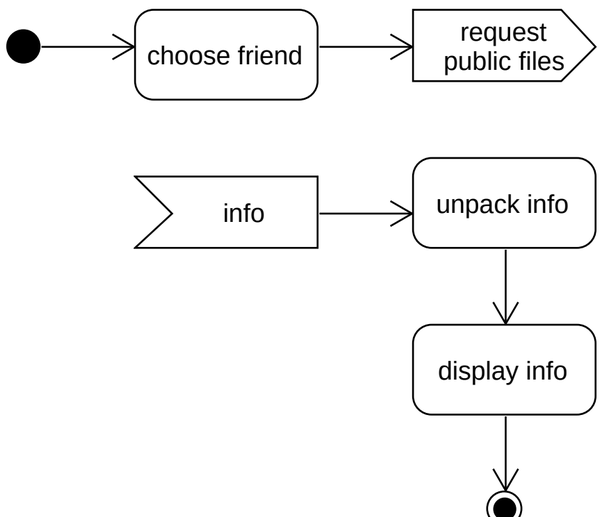


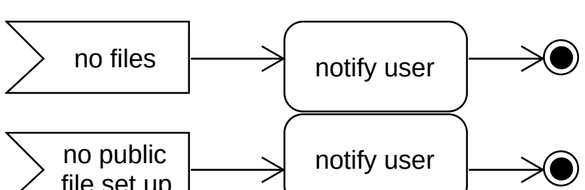
state

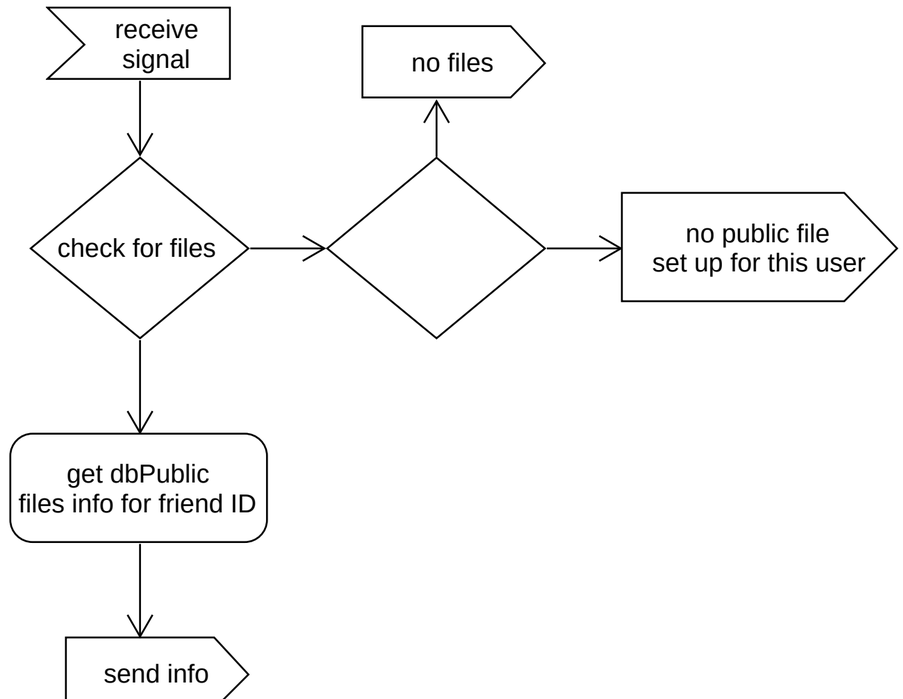
Note..
db side
check to see if user allowed
(valid date and permissions)

view a single friend and their public files.pdf

Note..
view one friend and th







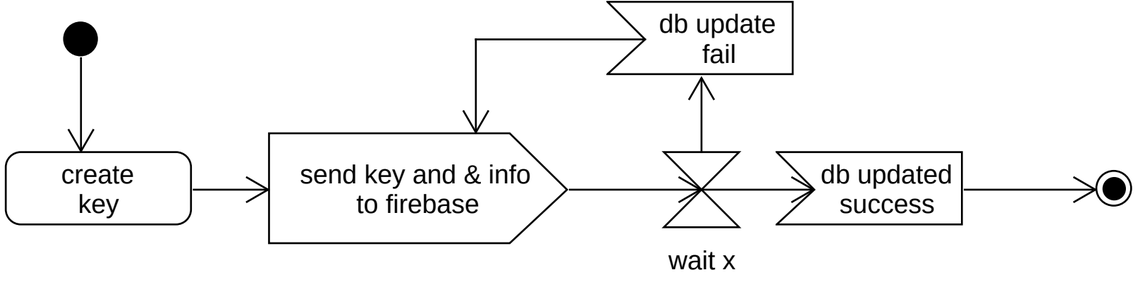
receive(with perm) decrypt and open unencrypted one.pdf

«PLEASE READ»

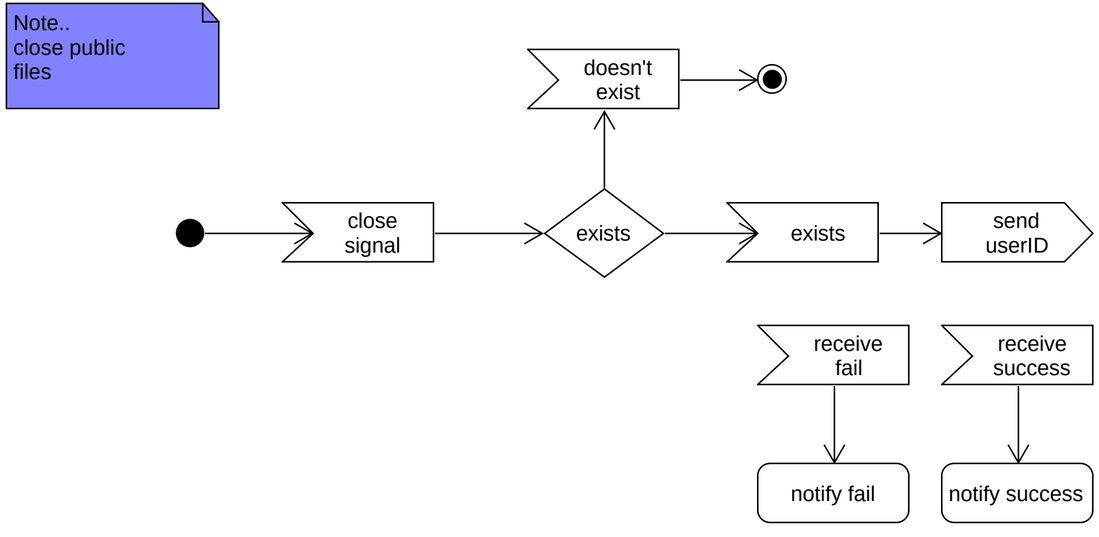
Not too sure how would do currently
If you do, please just add stuff to this diagram and export the pdf

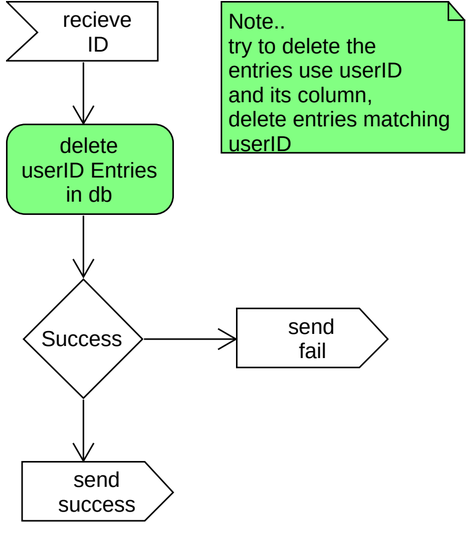
store keys.pdf

Note..
store keys



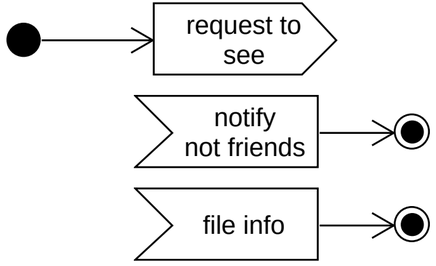
close public files.pdf

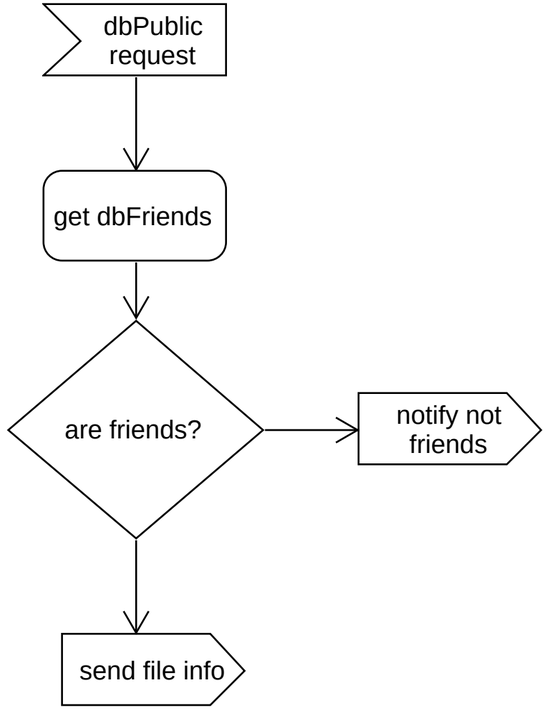

Note..
close public
files
doesn't
exist
close
signal
exists
exists
send
userID
receive
fail
receive
success
notify fail
notify success


recieve
ID
Note..
try to delete the
entries use userID
and its column,
delete entries matching
userID
delete
userID Entries
in db
Success
send
fail
send
success

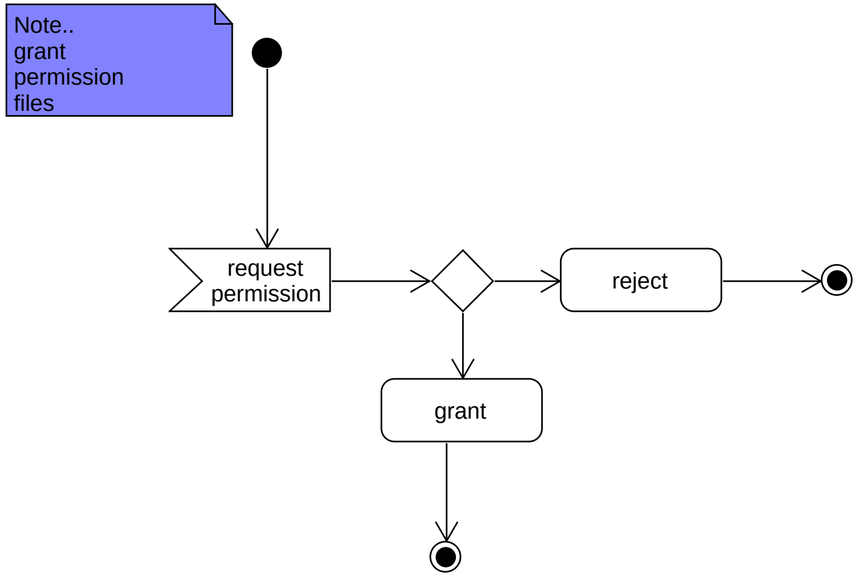
any friend should be able to see my public files.pdf

Note..

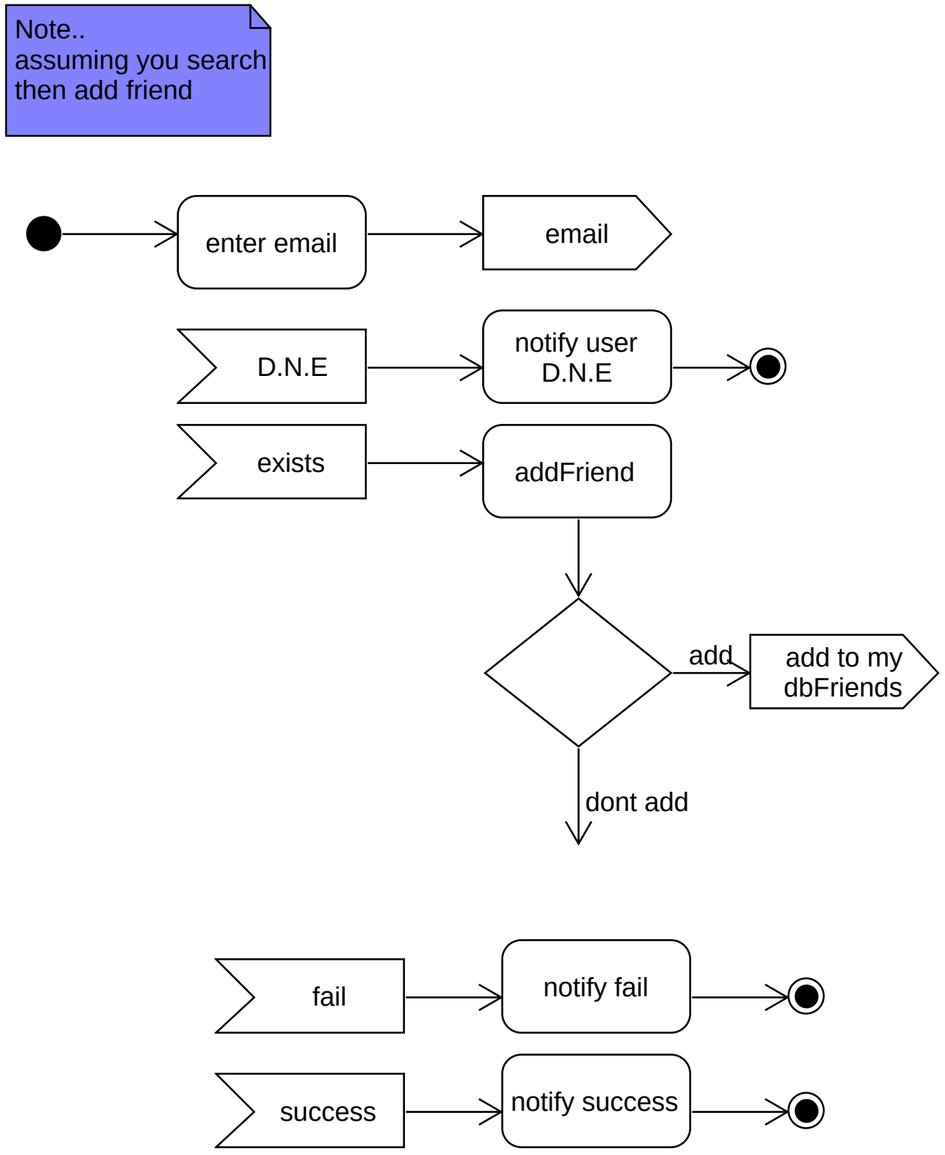

request to see
notify not friends
file info

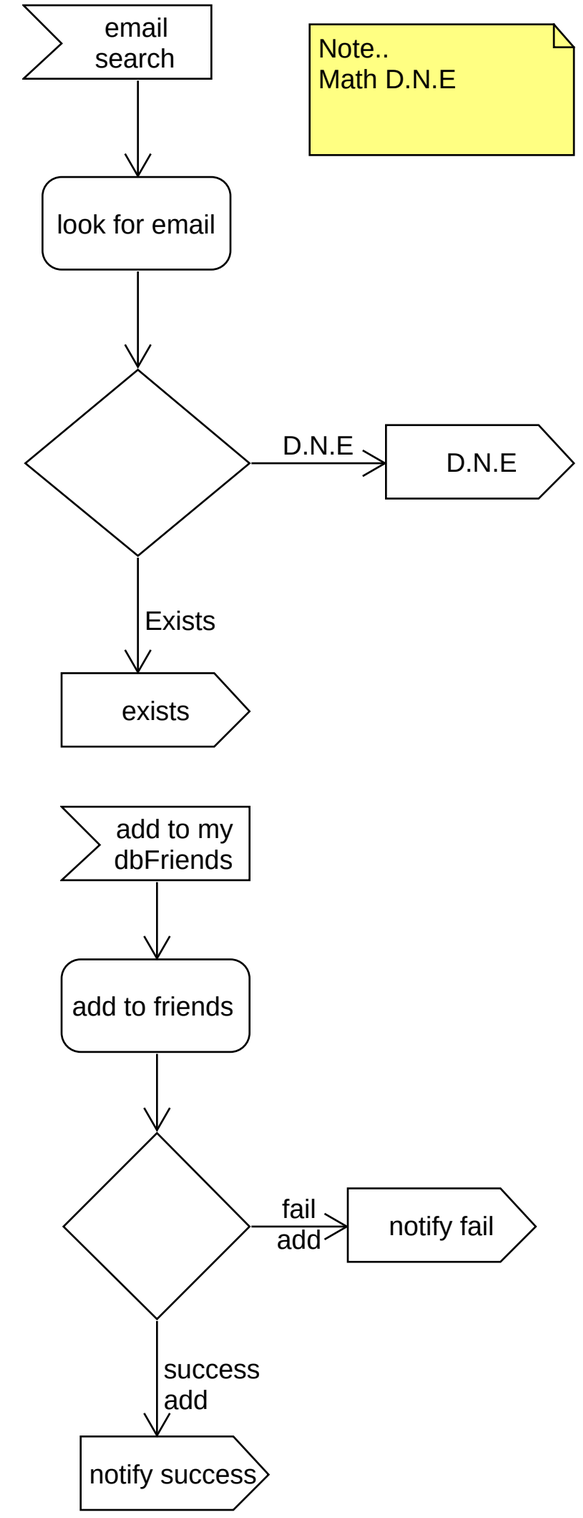

dbPublic request
get dbFriends
are friends?
notify not friends
send file info

grant permisision files.pdf


Note..
grant
permission
files
request
permission
reject
grant

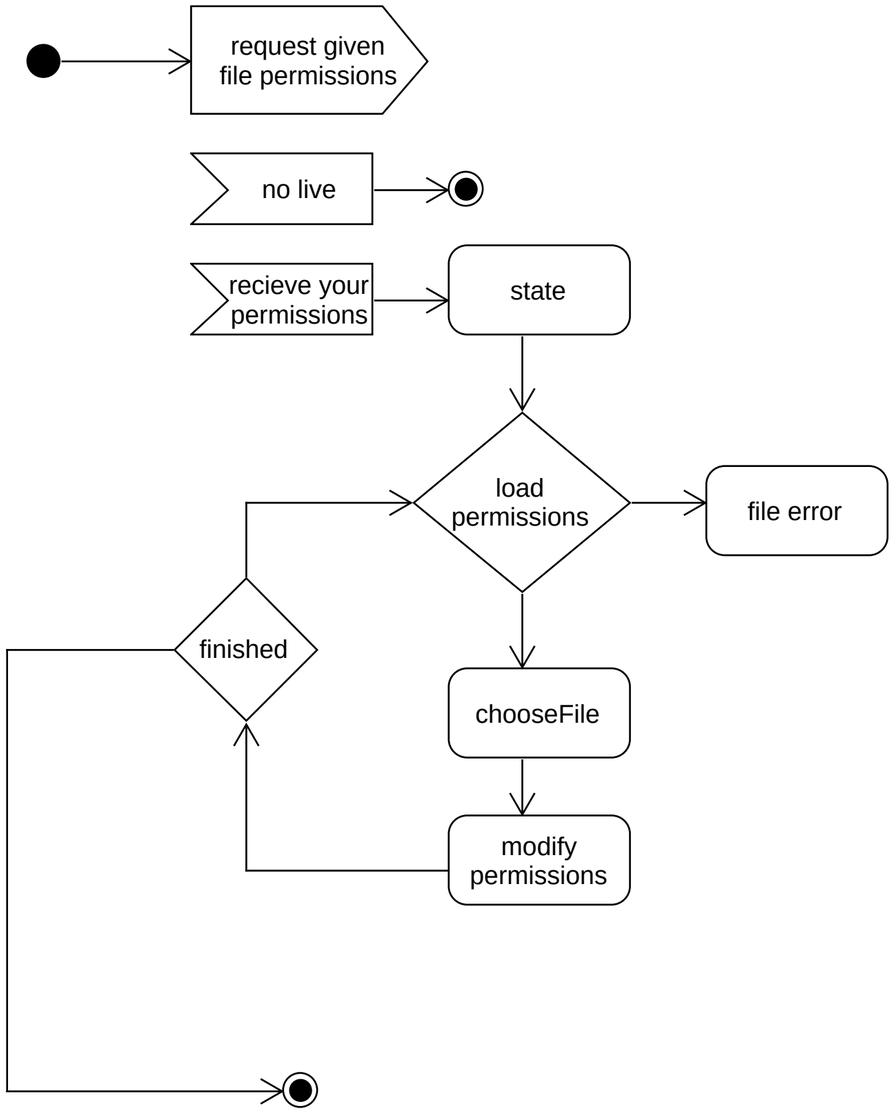
be able to search for emails and add friends .pdf

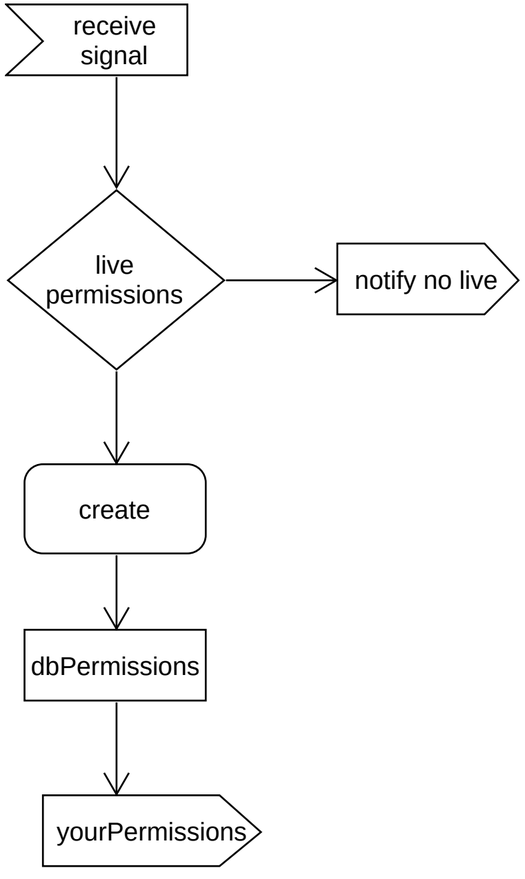

Note..
assuming you search
then add friend
enter email
email
D.N.E
notify user
D.N.E
exists
addFriend
add
add to my
dbFriends
dont add
fail
notify fail
success
notify success


email
search
Note..
Math D.N.E
look for email
D.N.E
D.N.E
Exists
exists
add to my
dbFriends
add to friends
fail
add
notify fail
success
add
notify success

give or revoke permission as if I sent file.pdf

Note..
Be able to give/revoke permission
as if I sent the file

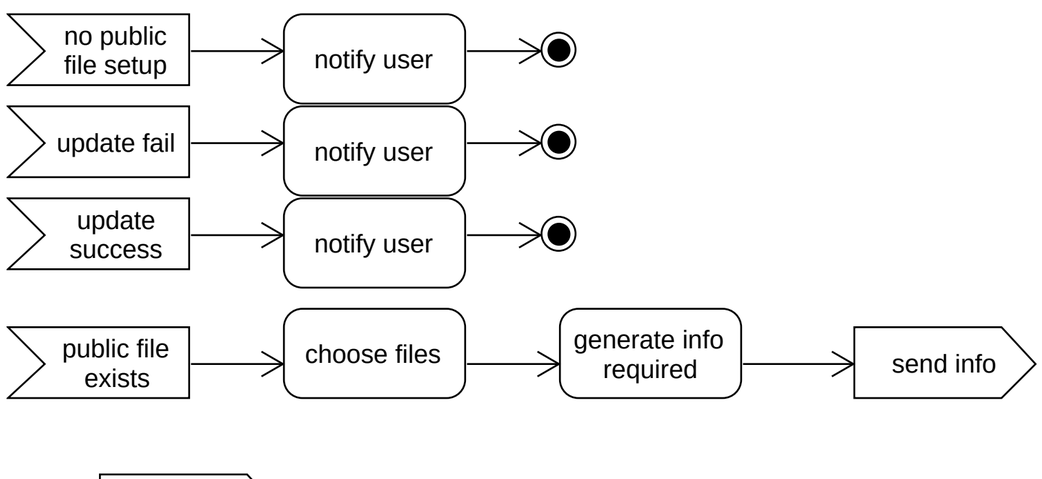


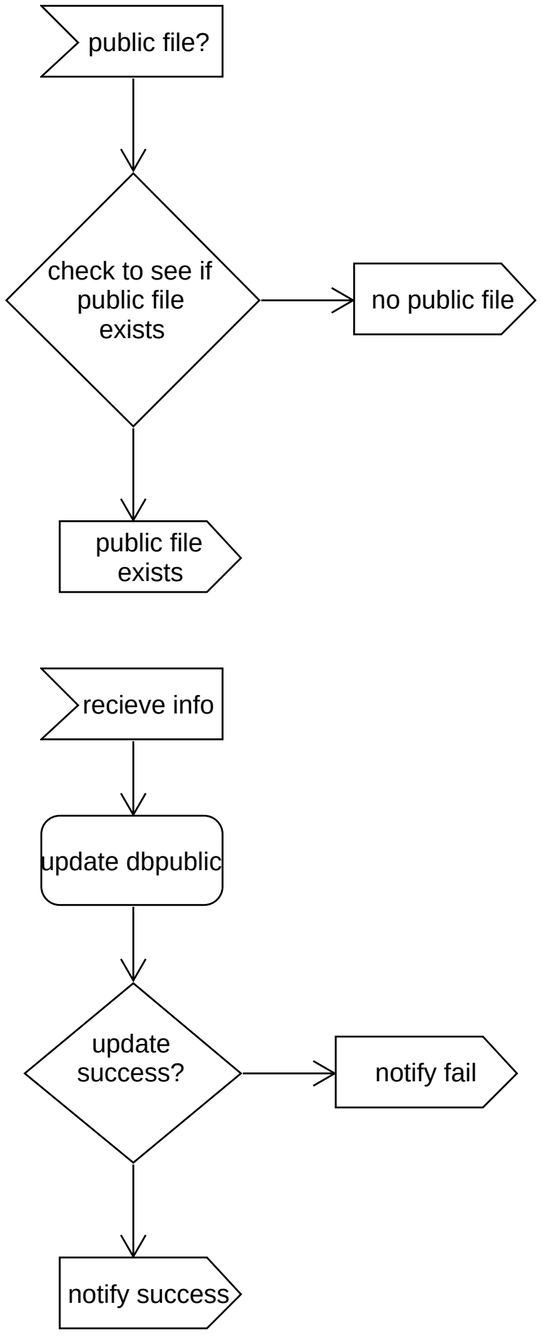


Note..
Assuming that we want to send back some form of files
and their permissions so that you can tweak them.
object dbPermissions most likely unecessary as could be
packaged differently

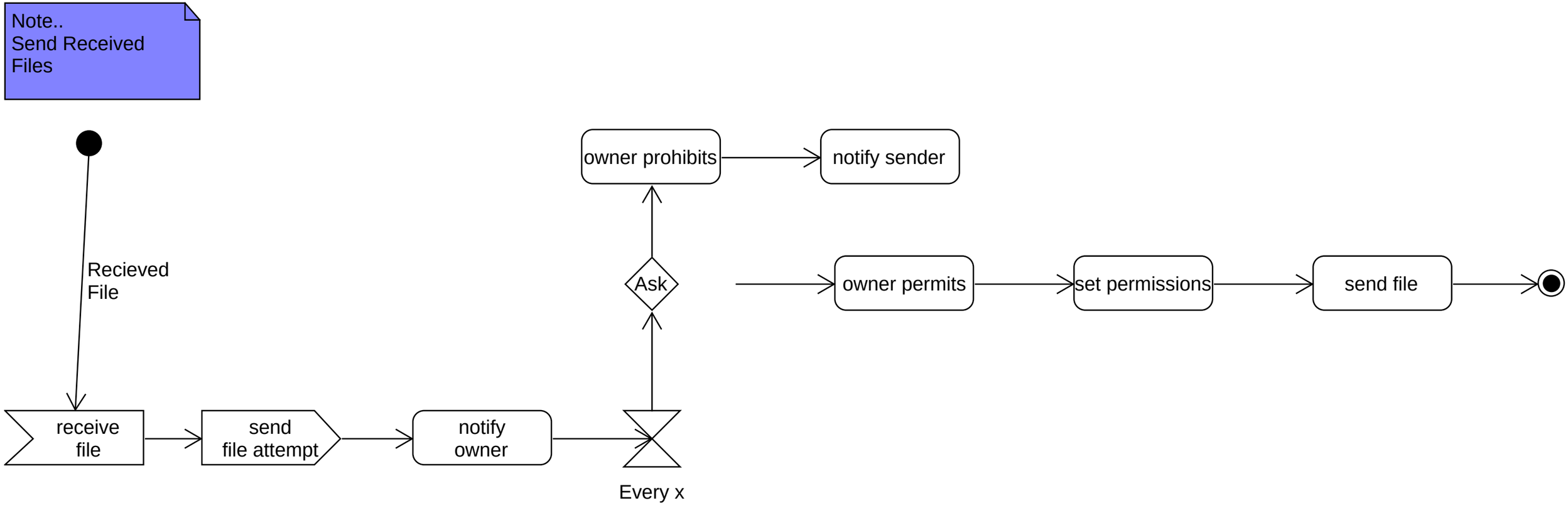
add files to my public shared space.pdf

Note..


no public file setup
notify user
update fail
notify user
update success
notify user
public file exists
choose files
generate info required
send info
public file created?


public file?
check to see if public file exists
no public file
public file exists
recieve info
update dbpublic
update success?
notify fail
notify success

send recieved file.pdf


Note..
Send Received
Files
Recieved
File
receive
file
send
file attempt
notify
owner
Every x
Ask
owner prohibits
notify sender
owner permits
set permissions
send file

when viewing my friends i should be able to see their public files.pdf

Note..
This kind of feels like it goes with the
any friend should be able to see my public file
so long as you are friends yours should be available
and theirs should be available.
Will rather think of as a view dbs and select a friend
if they have on load files otherwise notify empty.