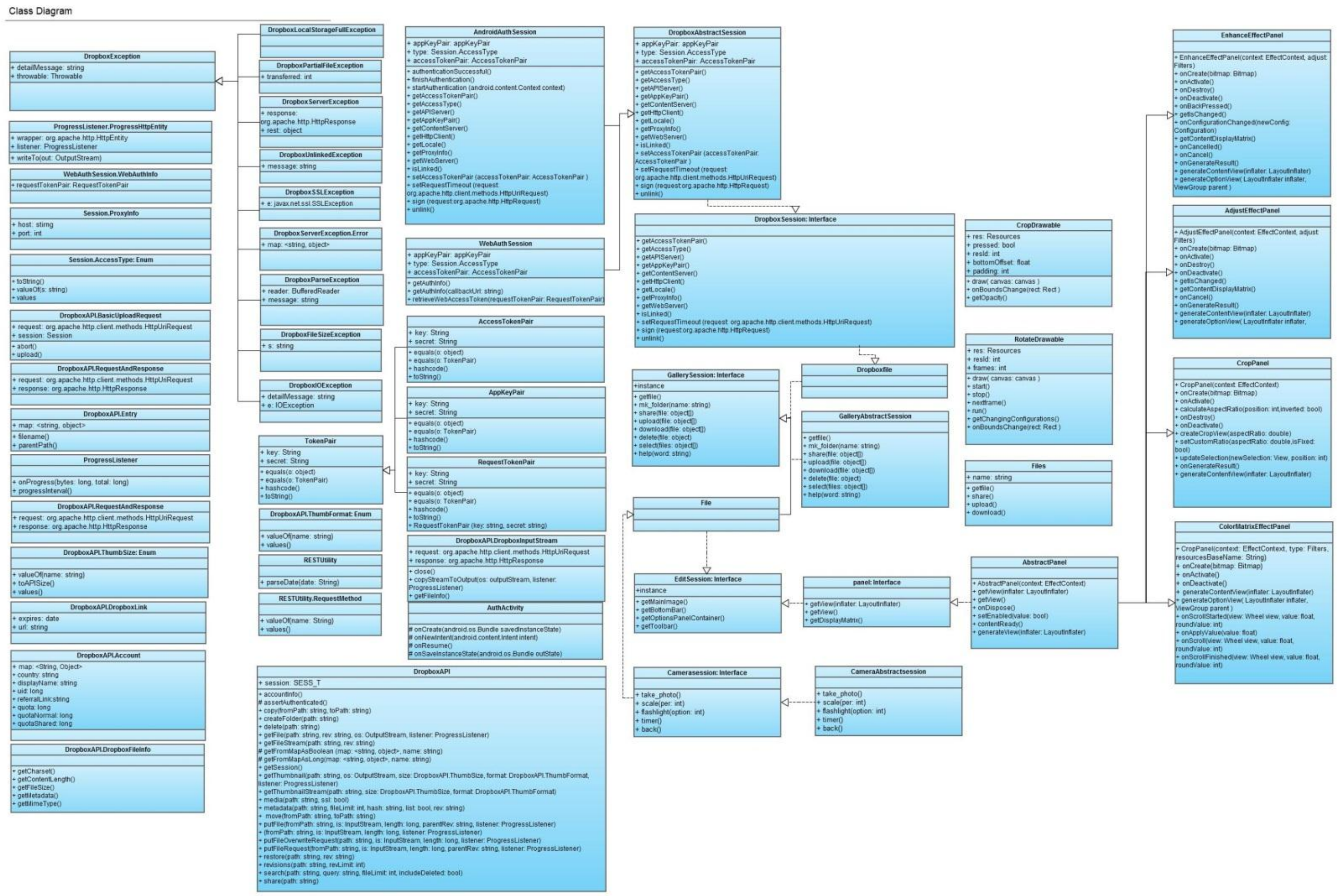
Detail Level Design

Detail Level Design documents based on your Phase A Architectural and High-Level Design a) Internal design of the classes you defined in your high-level design for Phase A, including non-public methods, fields (data representation).


Class Diagram
DropboxException
+ detailMessage: string
+ throwable: Throwable
ProgressListener.ProgressHttpEntity
+ wrapper: org.apache.http.HttpEntity
+ listener: ProgressListener
+ writeTo(out: OutputStream)
WebAuthSession.WebAuthInfo
+ requestTokenPair: RequestTokenPair
Session.ProxyInfo
+ host: stirng
+ port: int
Session.AccessType: Enum
+ toString()
+ valueOf(s: string)
+ values
DropboxAPI.BasicUploadRequest
+ request: org.apache.http.client.methods.HttpUriRequest
+ session: Session
+ abort()
+ upload()
DropboxAPI.RequestAndResponse
+ request: org.apache.http.client.methods.HttpUriRequest
+ response: org.apache.http.HttpResponse
DropboxAPI.Entry
+ map: <string, object>
+ filename()
+ parentPath()
ProgressListener
+ onProgress(bytes: long, total: long)
+ progressInterval()
DropboxAPI.RequestAndResponse
+ request: org.apache.http.client.methods.HttpUriRequest
+ response: org.apache.http.HttpResponse
DropboxAPI.ThumbSize: Enum
+ valueOf(name: string)
+ toAPISize()
+ values()
DropboxAPI.DropboxLink
+ expires: date
+ url: string
DropboxAPI.Account
+ map: <String, Object>
+ country: string
+ displayName: string
+ uid: long
+ referralLink:string
+ quota: long
+ quotaNormal: long
+ quotaShared: long
DropboxAPI.DropboxFileInfo
+ getCharset()
+ getContentLength()
+ getFileSize()
+ getMetadata()
+ getMimeType()
DropboxLocalStorageFullException
DropboxPartialFileException
+ transferred: int
DropboxServerException
+ response: org.apache.http.HttpResponse
+ rest: object
DropboxUnlinkedException
+ message: string
DropboxSSLException
+ e: javax.net.ssl.SSLException
DropboxServerException.Error
+ map: <string, object>
DropboxParseException
+ reader: BufferedReader
+ message: string
DropboxFileSizeException
+ s: string
DropboxIOException
+ detailMessage: string
+ e: IOException
TokenPair
+ key: String
+ secret: String
+ equals(o: object)
+ equals(o: TokenPair)
+ hashcode()
+ toString()
DropboxAPI.ThumbFormat: Enum
+ valueOf(name: string)
+ values()
RESTUtility
+ parseDate(date: String)
RESTUtility.RequestMethod
+ valueOf(name: String)
+ values()
AndroidAuthSession
+ appKeyPair: appKeyPair
+ type: Session.AccessType
+ accessTokenPair: AccessTokenPair
+ authenticationSuccessful()
+ finishAuthentication()
+ startAuthentication (android.content.Context context)
+ getAccessTokenPair()
+ getAccessType()
+ getAPIServer()
+ getAppKeyPair()
+ getContentServer()
+ getHttpClient()
+ getLocale()
+ getProxyInfo()
+ getWebServer()
+ isLinked()
+ setAccessTokenPair (accessTokenPair: AccessTokenPair)
+ setRequestTimeout (request: org.apache.http.client.methods.HttpUriRequest)
+ sign (request:org.apache.http.HttpRequest)
+ unlink()
WebAuth Session
+ appKeyPair: appKeyPair
+ type: Session.AccessType
+ accessTokenPair: AccessTokenPair
+ getAuthInfo()
+ getAuthInfo(callbackUrl: string)
+ retrieveWebAccessToken(requestTokenPair: RequestTokenPair)
AccessTokenPair
+ key: String
+ secret: String
+ equals(o: object)
+ equals(o: TokenPair)
+ hashcode()
+ toString()
AppKeyPair
+ key: String
+ secret: String
+ equals(o: object)
+ equals(o: TokenPair)
+ hashcode()
+ toString()
RequestTokenPair
+ key: String
+ secret: String
+ equals(o: object)
+ equals(o: TokenPair)
+ hashcode()
+ toString()
+ RequestTokenPair (key: string, secret: string)
DropboxAPI.DropboxInputStream
+ request: org.apache.http.client.methods.HttpUriRequest
+ response: org.apache.http.HttpResponse
+ close()
+ copyStreamToOutput(os: outputStream, listener: ProgressListener)
+ getFileInfo()
AuthActivity
# onCreate(android.os.Bundle savedInstanceState)
# onNewIntent(android.content.Intent intent)
# onResume()
# onSaveInstanceState(android.os.Bundle outState)
DropboxAPI
+ session: SESS_T
+ accountInfo()
# assertAuthenticated()
+ copy(fromPath: string, toPath: string)
+ createFolder(path: string)
+ delete(path: string)
+ getFile(path: string, rev: string, os: OutputStream, listener: ProgressListener)
+ getFileStream(path: string, rev: string)
# getFromMapAsBoolean (map: <string, object>, name: string)
# getFromMapAsLong(map: <string, object>, name: string)
+ getSession()
+ getThumbnail(path: string, os: OutputStream, size: DropboxAPI.ThumbSize, format: DropboxAPI.ThumbFormat, listener: ProgressListener)
+ getThumbnailStream(path: string, size: DropboxAPI.ThumbSize, format: DropboxAPI.ThumbFormat)
+ media(path: string, ssl: bool)
+ metadata(path: string, fileLimit: int, hash: string, list: bool, rev: string)
+ move(fromPath: string, toPath: string)
+ putFile(fromPath: string, is: InputStream, length: long, parentRev: string, listener: ProgressListener)
+ (fromPath: string, is: InputStream, length: long, listener: ProgressListener)
+ putFileOverwriteRequest(path: string, is: InputStream, length: long, listener: ProgressListener)
+ putFileRequest(fromPath: string, is: InputStream, length: long, parentRev: string, listener: ProgressListener)
+ restore(path: string, rev: string)
+ revisions(path: string, revLimit: int)
+ search(path: string, query: string, fileLimit: int, includeDeleted: bool)
+ share(path: string)
DropboxAbstractSession
+ appKeyPair: appKeyPair
+ type: Session.AccessType
+ accessTokenPair: AccessTokenPair
+ getAccessTokenPair()
+ getAccessType()
+ getAPIServer()
+ getAppKeyPair()
+ getContentServer()
+ getHttpClient()
+ getLocale()
+ getProxyInfo()
+ getWebServer()
+ isLinked()
+ setAccessTokenPair (accessTokenPair: AccessTokenPair)
+ setRequestTimeout (request: org.apache.http.client.methods.HttpUriRequest)
+ sign (request:org.apache.http.HttpRequest)
+ unlink()
Dropbox Session: Interface
+ getAccessTokenPair()
+ getAccessType()
+ getAPIServer()
+ getAppKeyPair()
+ getContentServer()
+ getHttpClient()
+ getLocale()
+ getProxyInfo()
+ getWebServer()
+ isLinked()
+ setRequestTimeout (request: org.apache.http.client.methods.HttpUriRequest)
+ sign (request:org.apache.http.HttpRequest)
+ unlink()
GallerySession: Interface
+instance
+ getfile()
+ mk_folder(name: string)
+ share(file: object[])
+ upload(file: object[])
+ download(file: object[])
+ delete(file: object)
+ select(files: object[])
+ help(word: string)
Dropboxfile
GalleryAbstractSession
+ getfile()
+ mk_folder(name: string)
+ share(file: object[])
+ upload(file: object[])
+ download(file: object[])
+ delete(file: object)
+ select(files: object[])
+ help(word: string)
File
EditSession: Interface
+instance
+ getMainImage()
+ getBottomBar()
+ getOptionsPanelContainer()
+ getToolbar()
panel: Interface
+ getView(inflater: LayoutInflater)
+ getView()
+ getDisplayMatrix()
Camerasession: Interface
+ take_photo()
+ scale(per: int)
+ flashlight(option: int)
+ timer()
+ back()
CameraAbstractsession
+ take_photo()
+ scale(per: int)
+ flashlight(option: int)
+ timer()
+ back()
CropDrawable
+ res: Resources
+ pressed: bool
+ resId: int
+ bottomOffset: float
+ padding: int
+ draw(canvas: canvas)
+ onBoundsChange(rect: Rect)
+ getOpacity()
RotateDrawable
+ res: Resources
+ resId: int
+ frames: int
+ draw(canvas: canvas)
+ start()
+ stop()
+ nextframe()
+ run()
+ getChangingConfigurations()
+ onBoundsChange(rect: Rect)
Files
+ name: string
+ getfile()
+ share()
+ upload()
+ download()
AbstractPanel
+ AbstractPanel(context: EffectContext)
+ getView(inflater: LayoutInflater)
+ getView()
+ onDispose()
+ setEnabled(value: bool)
+ contentReady()
+ generateView(inflater: LayoutInflater)
EnhanceEffectPanel
+ EnhanceEffectPanel(context: EffectContext, adjust: Filters)
+ onCreate(bitmap: Bitmap)
+ onActivate()
+ onDestroy()
+ onDeactivate()
+ onBackPressed()
+ getIsChanged()
+ onConfigurationChanged(newConfig: Configuration)
+ getContentDisplayMatrix()
+ onCancelled()
+ onCancel()
+ onGenerateResult()
+ generateContentView(inflater: LayoutInflater)
+ generateOptionView(LayoutInflater inflater, ViewGroup parent)
AdjustEffectPanel
+ AdjustEffectPanel(context: EffectContext, adjust: Filters)
+ onCreate(bitmap: Bitmap)
+ onActivate()
+ onDestroy()
+ onDeactivate()
+ getIsChanged()
+ getContentDisplayMatrix()
+ onCancel()
+ onGenerateResult()
+ generateContentView(inflater: LayoutInflater)
+ generateOptionView(LayoutInflater inflater,
CropPanel
+ CropPanel(context: EffectContext)
+ onCreate(bitmap: Bitmap)
+ onActivate()
+ calculateAspectRatio(position: int,inverted: bool)
+ onDestroy()
+ onDeactivate()
+ createCropView(aspectRatio: double)
+ setCustomRatio(aspectRatio: double,isFixed: bool)
+ updateSelection(newSelection: View, position: int)
+ onGenerateResult()
+ generateContentView(inflater: LayoutInflater)
ColorMatrixEffectPanel
+ CropPanel(context: EffectContext, type: Filters, resourcesBaseName: String)
+ onCreate(bitmap: Bitmap)
+ onActivate()
+ onDeactivate()
+ generateContentView(inflater: LayoutInflater)
+ generateOptionView(LayoutInflater inflater, ViewGroup parent)
+ onScrollStarted(view: Wheel view, value: float, roundValue: int)
+ onApplyValue(value: float)
+ onScroll(view: Wheel view, value: float, roundValue: int)
+ onScrollFinished(view: Wheel view, value: float, roundValue: int)

******The actual version will be sent along with all the other documents so you are able to see the image and details clearly.*****

b) Rationale for your design choices – brief explanation of why you chose to implement things in a certain way.

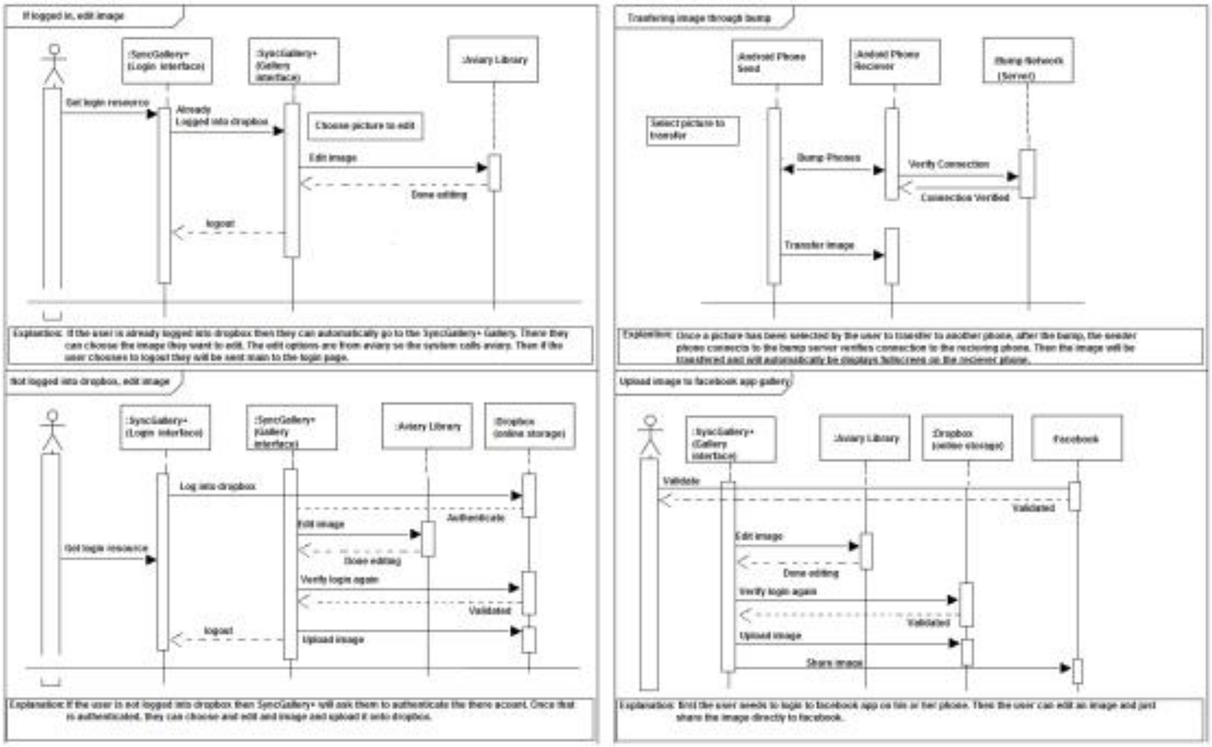
A neat, organized and detailed class diagram will accurately demonstrate the internal structure of the application and its processes. This is because the class diagram provides us with an initial set of notation elements that all other structure diagrams use. A class diagram describes the classes that make up a system and the static relationships between them. Classes are defined in terms of their name, attributes (or data), and behaviors (or methods). The static relationships are association, aggregation, and inheritance. This will help the developer get a clear understand of how the application should work like internally so he or she can start coding in the right direction faster and more efficiently.

Secondly, the sequence diagram is used for the design of the internal architecture of the application. This is because it is used primarily to show the interactions between objects in the sequential order that those interactions occur. The sequence diagram is very much so like the class diagram. With sequence diagrams developers can use it to force out the system's object interactions, thus fleshing out overall system design.

c) You may want to take into consideration the cohesion and coupling criteria in designing how your classes will work internally.

Basically, the Aviary source project is imported into the library where the developers will be implementing it while programming for the application. The developers will then import the dropbox API and implement the methods and functions etc. into their natural code.

d) If you used techniques such as sequence diagrams and robustness analysis in your design, you should document these.



If logged in, edit image

Explanation: If the user is already logged into dropbox then they can automatically go to the SyncGallery+ Gallery. There they can choose the image they want to edit. The edit options are from aviary so the system calls aviary. Then if the user chooses to logout they will be sent main to the login page.

Transfering image through bump

Explanation: Once a picture has been selected by the user to transfer to another phone, after the bump, the sender phone connects to the bump server verifies connection to the recioving phone. Then the image will be transfered and will automatically be displays fullscreen on the reciever phone.

Not logged into dropbox, edit image

Explanation: If the user is not logged into dropbox then SyncGallery+ will ask them to authenticate the there acount. Once that is authenticated, they can choose and edit and image and upload it onto dropbox.

Upload image to facebook app gallery

Explanation: first the user needs to login to facebook app on his or her phone. Then the user can edit an image and just share the image directly to facebook.

*****The actual version is almost submitted.**

In these sequence diagrams various different possible situations are posed and the way in which the SyncGallery+ application works with the integrated components and functions is illustration sequentially.