AnyMemo Release One Summary

Team members

Name and Student id	GitHub id	Authored Story Points
Brendan McGarry	@brendanmcgarry	5
Paul Richard	@prich28	2
Dylan Fernandes	@dylanfernandes	5
Hiu Tung Lam (Emily)	@Ereimei	5
Adam Galati	@aagalati	3
Matthew Teolis	@MatthewTeolis	5
Olivier Nourry	@ONourry	5

Table of Content

Mobile App Summary

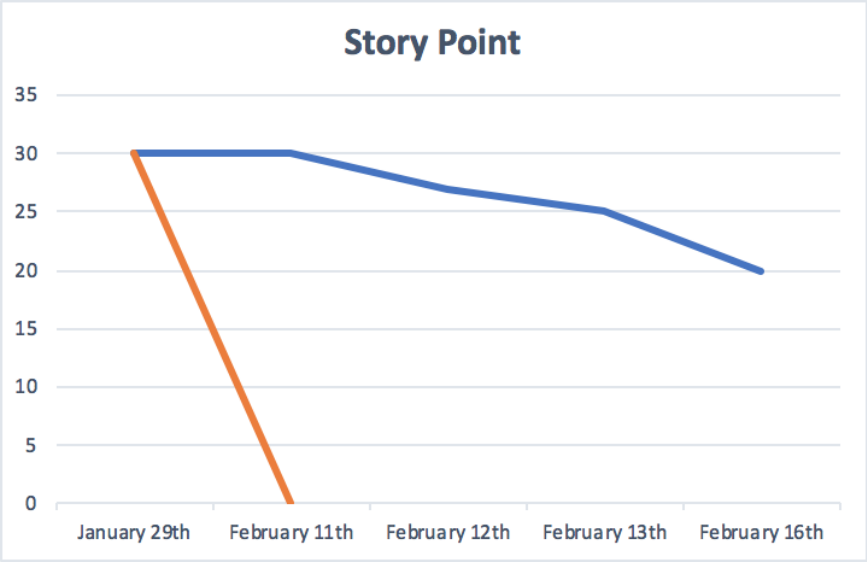
The AnyMemo application is principally a flashcard application where decks of cards with questions can be downloaded or created and used for studying. The application allows for making new decks, editing cards, studying and self-quizzing.

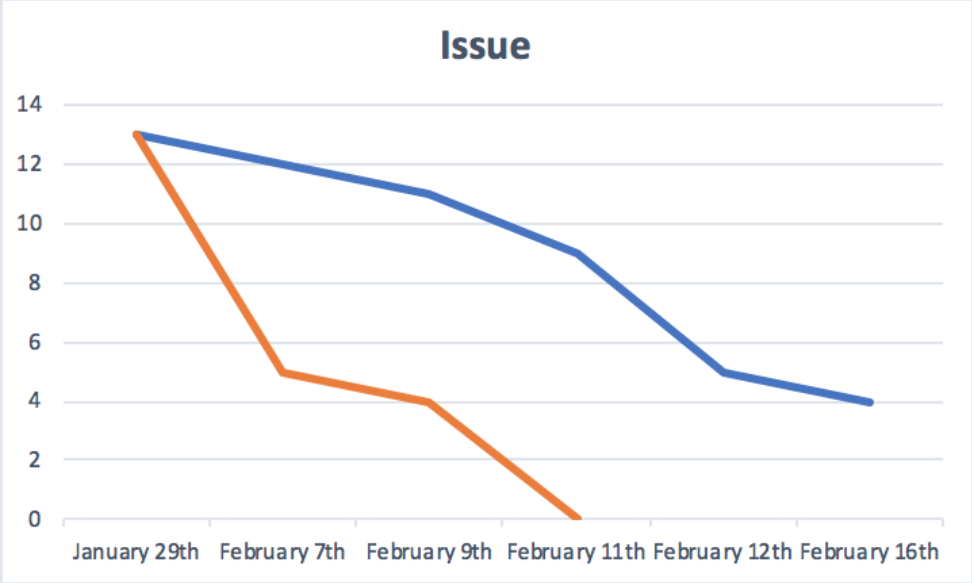
Velocity

Our velocity for the first iteration was 10 points.

A large story worth 20 points this sprint was half completed and push to next sprint, so the velocity is expected to see a jump between sprint 2 and 3.

Burndown Charts





Plan up to next release

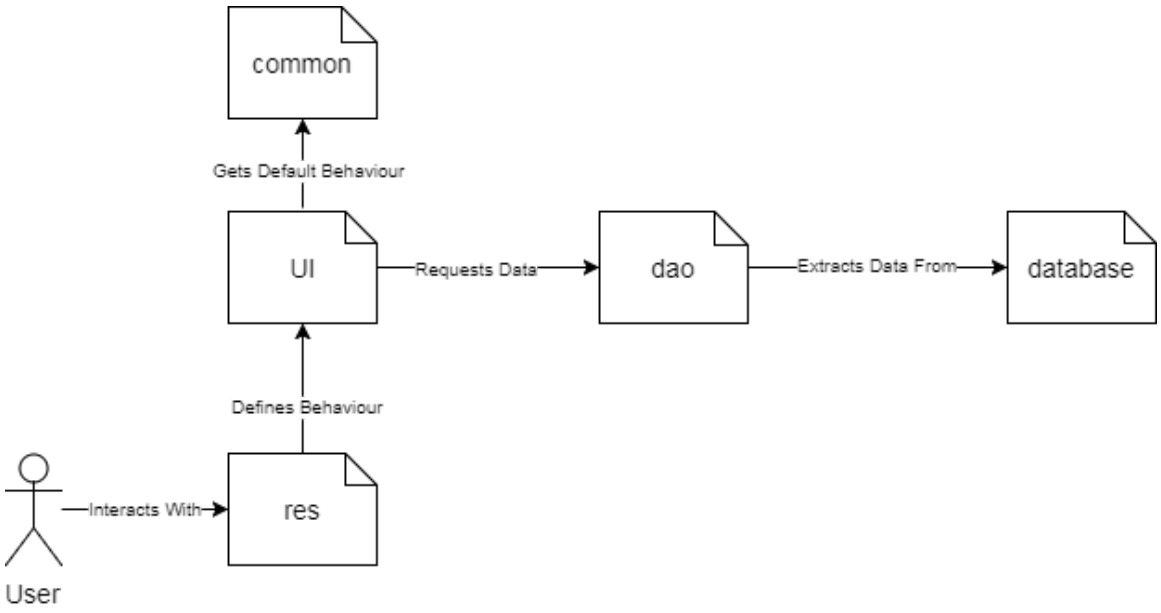
Total: 7 stories, 80 points, over 4 weeks
Iteration 3 (3 stories, 46 points)
Iteration 4, Release 2 (4 stories, 34 points)

Overall Architecture and Class diagram

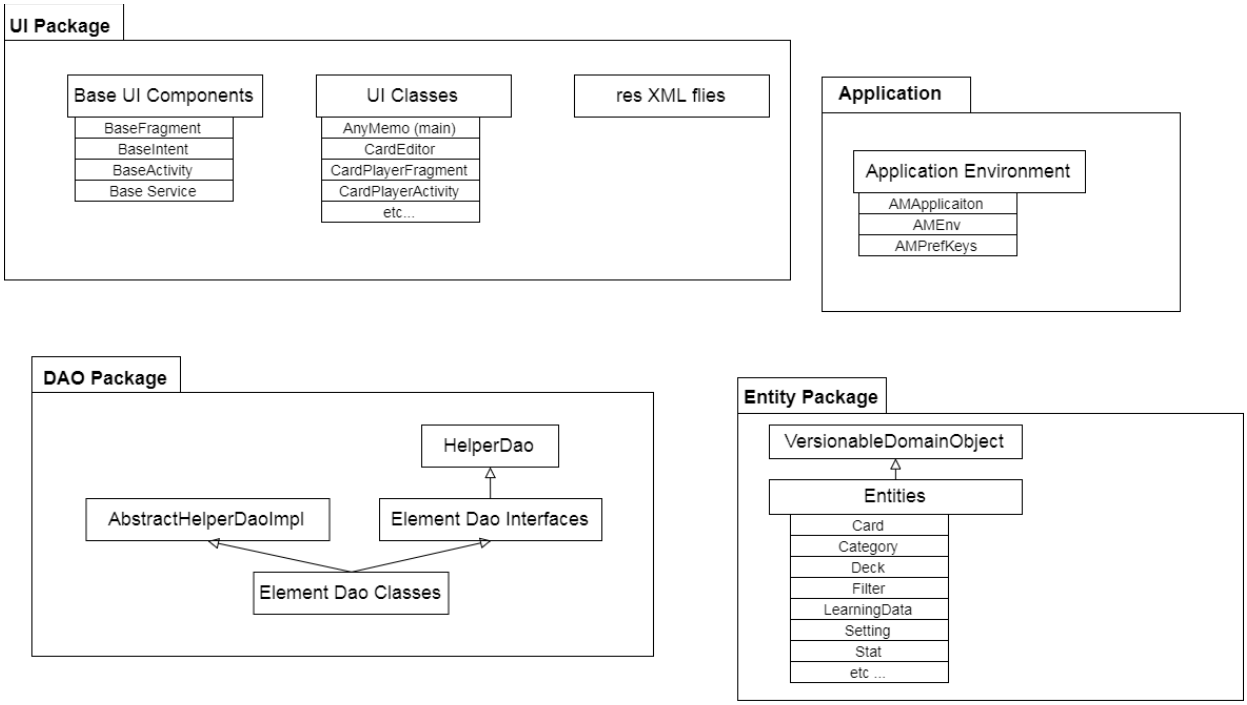
The initial recovery process of the application was difficult due to the structure of the project. The main classes of the application, such as Card and LearningData, are generated and handled via OrmLite's Dao (Data Access Object) interfaces and are very abstracted.

However, once it was determined that the Dao implementations for key classes such as Card and LearningData were handling the bulk of the domain logic along with the Android activities in the UI package, the behavior of the application became clearer.

Interaction Diagram

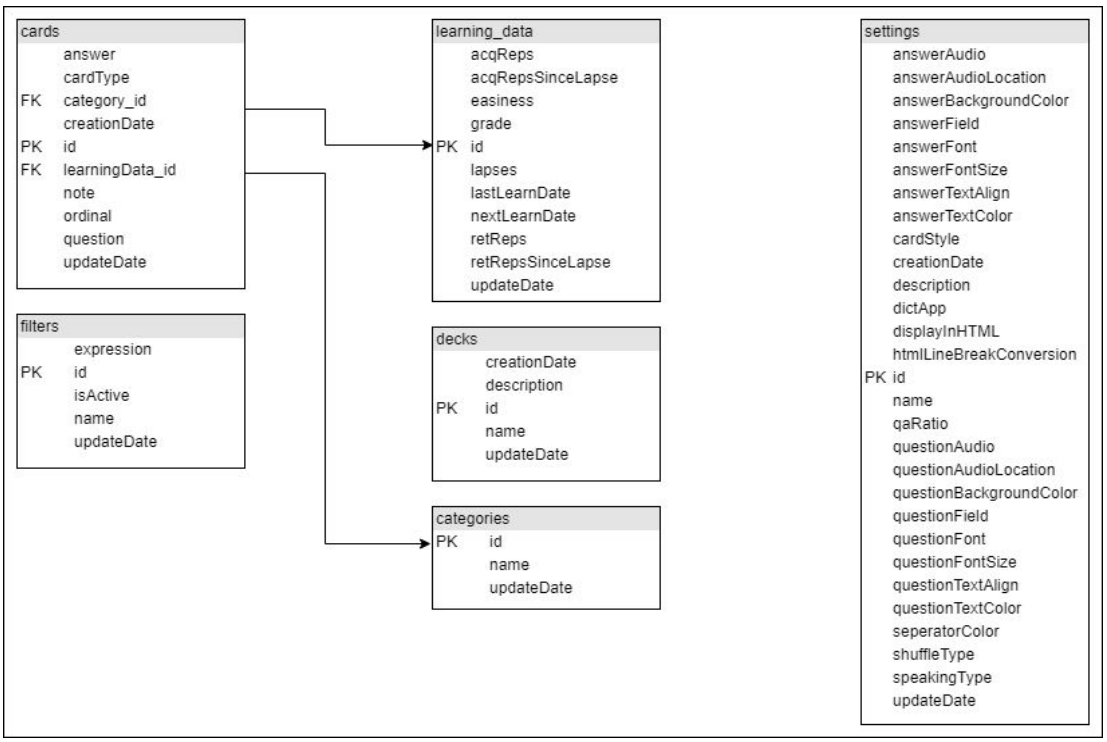


Package Diagram



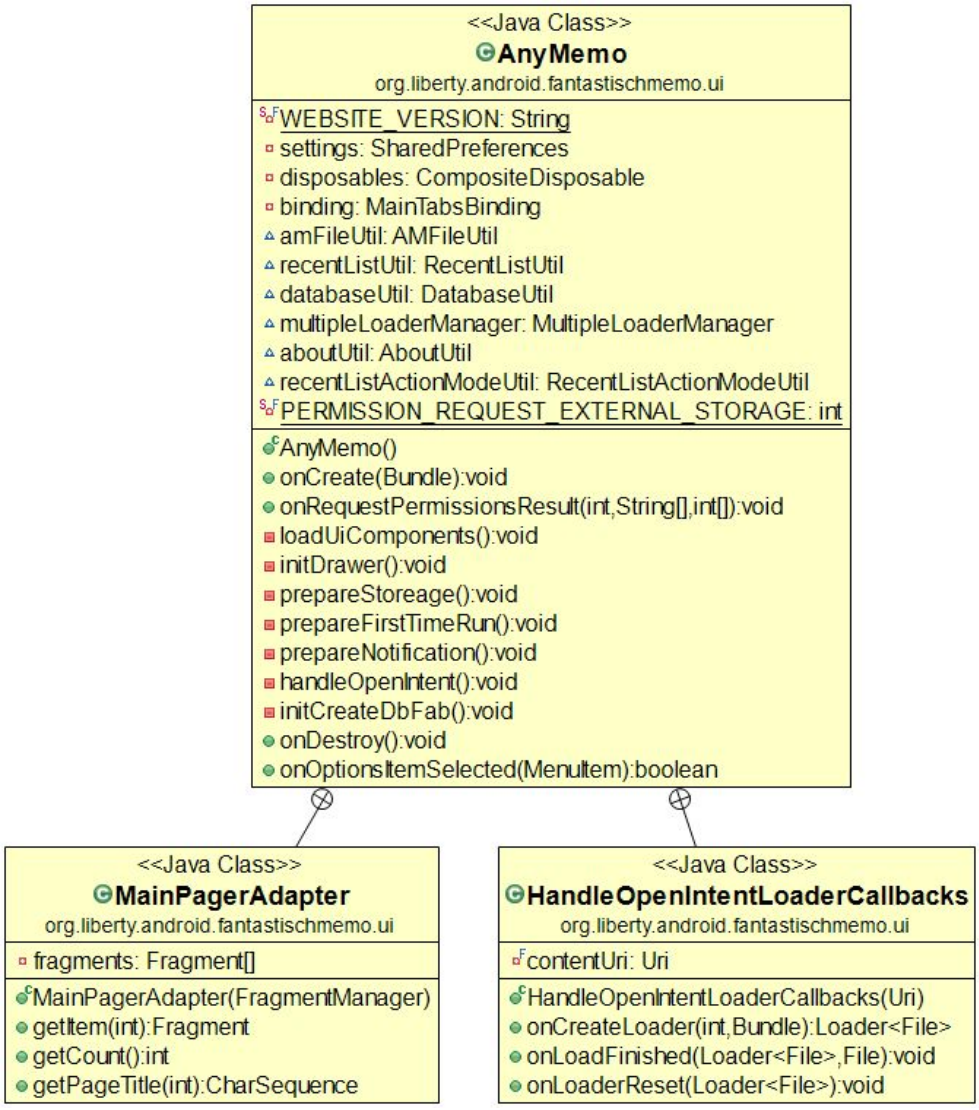
Database Architecture Diagram

The application uses individual databases for each deck. The databases are stored in the devices memory using the SQLite database management system. The entity package is stored in the database files using the corresponding Data Access Object in the DAO package; this functionality comes from the ORMlite package.

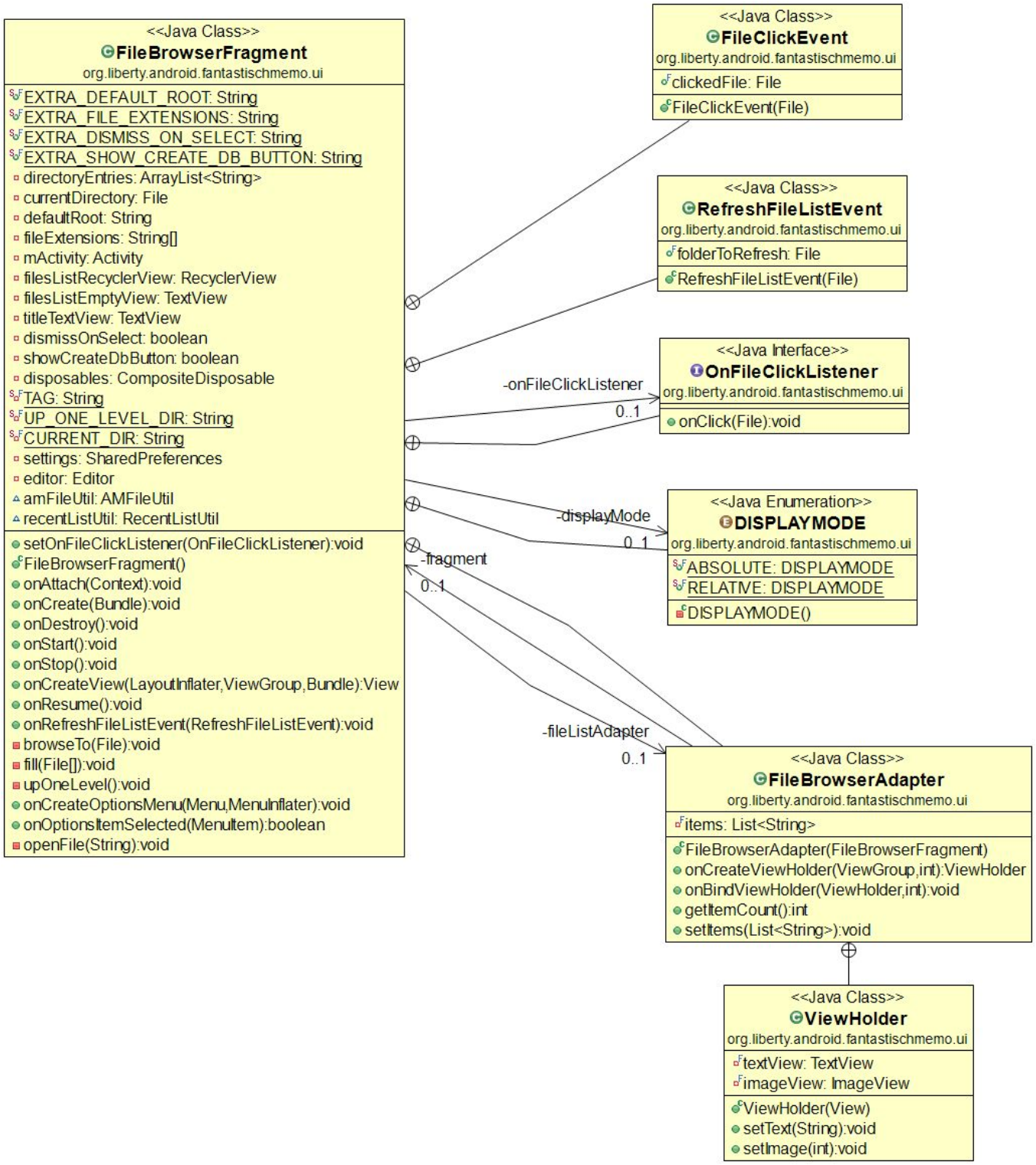


Class Diagram

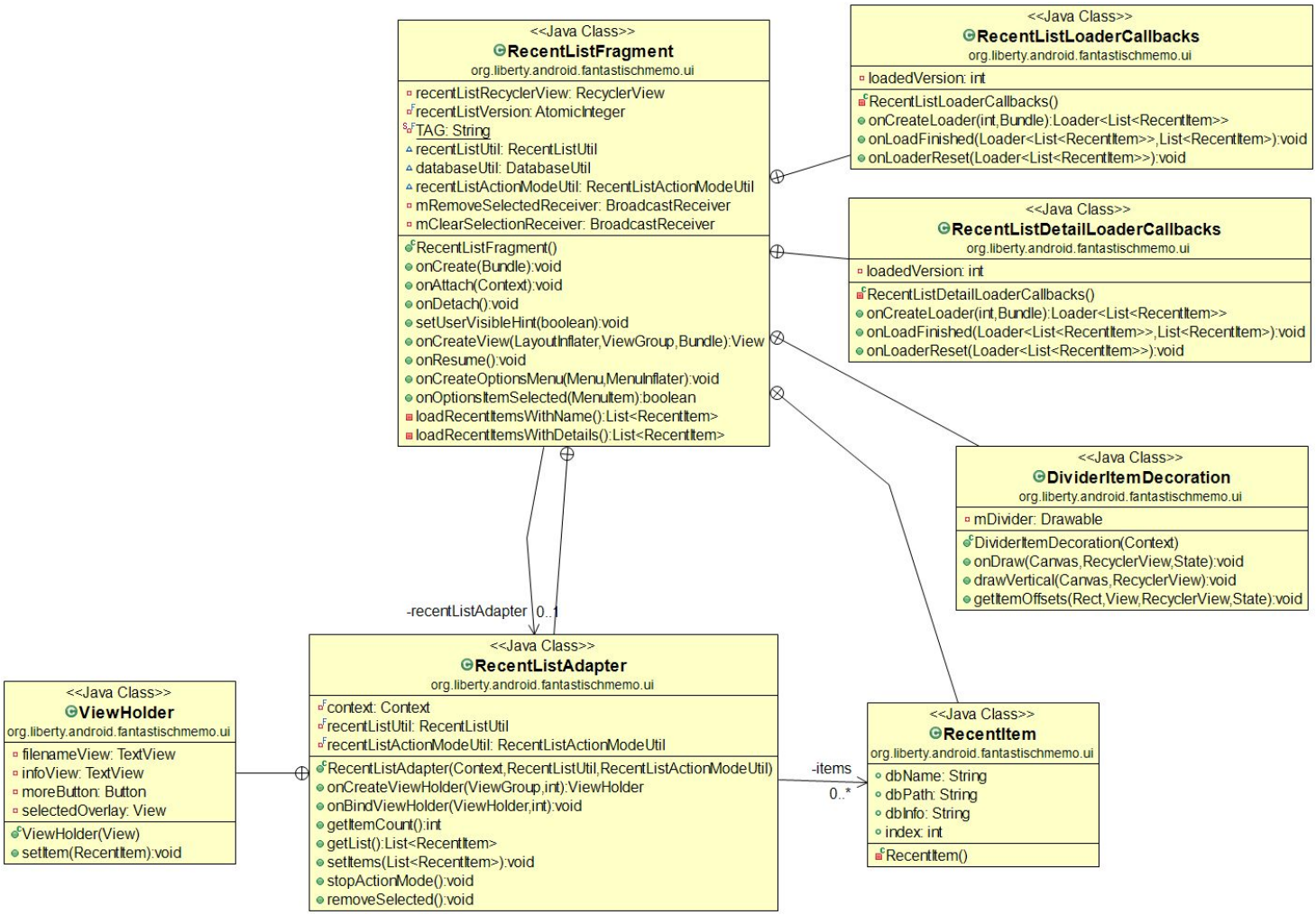
AnyMemo



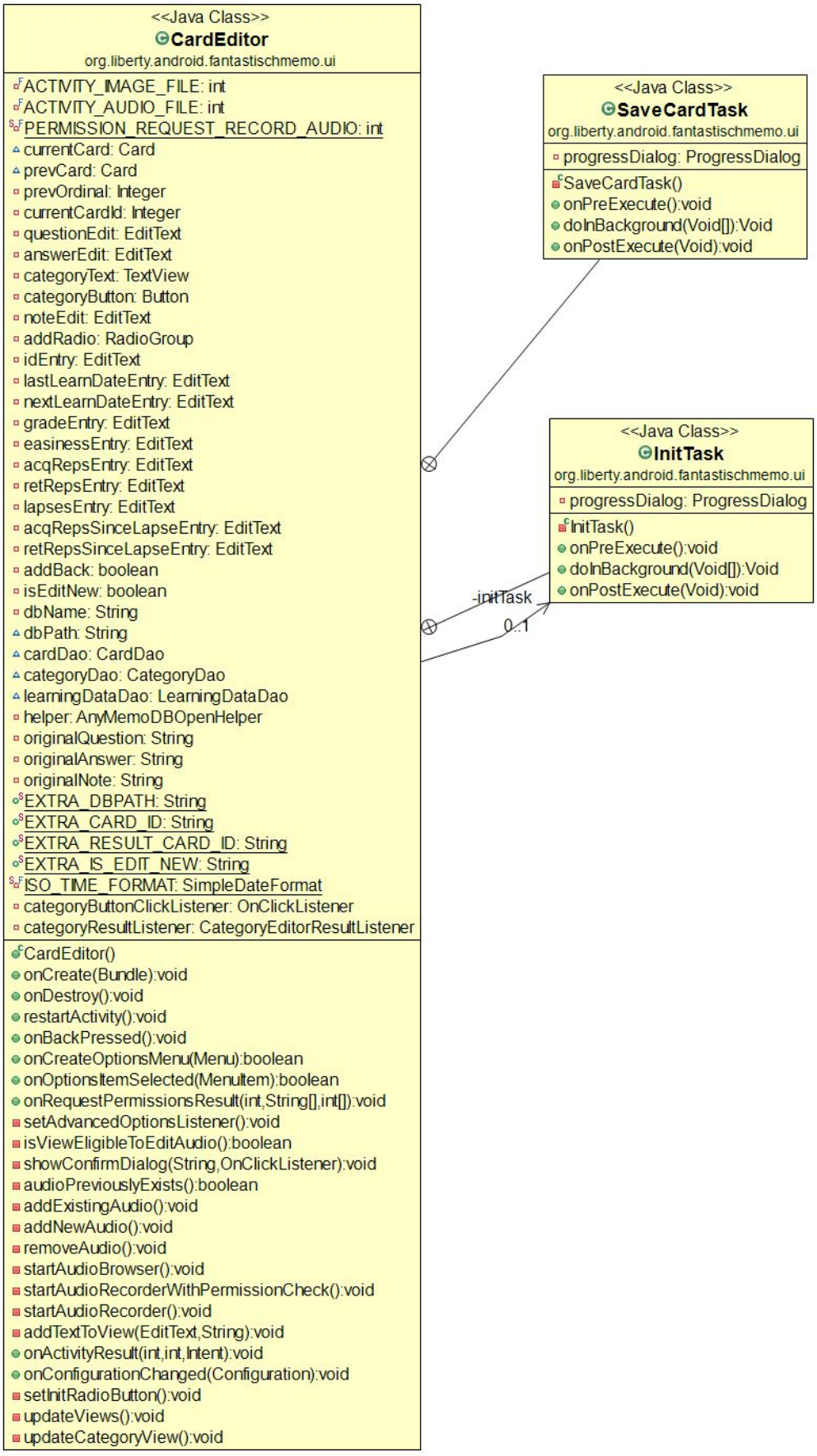
File Browser



Recent List


<<Java Class>>
RecentListFragment
org.liberty.android.fantastischmemo.ui
recentListRecyclerView: RecyclerView
recentListVersion: AtomicInteger
TAG: String
recentListUtil: RecentListUtil
databaseUtil: DatabaseUtil
recentListActionModeUtil: RecentListActionModeUtil
mRemoveSelectedReceiver: BroadcastReceiver
mClearSelectionReceiver: BroadcastReceiver
RecentListFragment()
onCreate(Bundle):void
onAttach(Context):void
onDetach():void
setUserVisibleHint(boolean):void
onCreateView(LayoutInflater,ViewGroup,Bundle):View
onResume():void
onCreateOptionsMenu(Menu,MenuInflater):void
onOptionsItemSelected(MenuItem):boolean
loadRecentItemsWithName():List<RecentItem>
loadRecentItemsWithDetails():List<RecentItem>
<<Java Class>>
RecentListLoaderCallbacks
org.liberty.android.fantastischmemo.ui
loadedVersion: int
RecentListLoaderCallbacks()
onCreateLoader(int,Bundle):Loader<List<RecentItem>>
onLoadFinished(Loader<List<RecentItem>>,List<RecentItem>):void
onLoaderReset(Loader<List<RecentItem>>):void
<<Java Class>>
RecentListDetailLoaderCallbacks
org.liberty.android.fantastischmemo.ui
loadedVersion: int
RecentListDetailLoaderCallbacks()
onCreateLoader(int,Bundle):Loader<List<RecentItem>>
onLoadFinished(Loader<List<RecentItem>>,List<RecentItem>):void
onLoaderReset(Loader<List<RecentItem>>):void
<<Java Class>>
DividerItemDecoration
org.liberty.android.fantastischmemo.ui
mDivider: Drawable
DividerItemDecoration(Context)
onDraw(Canvas,RecyclerView,State):void
drawVertical(Canvas,RecyclerView):void
getItemOffsets(Rect,View,RecyclerView,State):void
-recentListAdapter
0..1
<<Java Class>>
ViewHolder
org.liberty.android.fantastischmemo.ui
filenameView: TextView
infoView: TextView
moreButton: Button
selectedOverlay: View
ViewHolder(View)
setItem(RecentItem):void
<<Java Class>>
RecentListAdapter
org.liberty.android.fantastischmemo.ui
context: Context
recentListUtil: RecentListUtil
recentListActionModeUtil: RecentListActionModeUtil
RecentListAdapter(Context,RecentListUtil,RecentListActionModeUtil)
onCreateViewHolder(ViewGroup,int):ViewHolder
onBindViewHolder(ViewHolder,int):void
getItemCount():int
getList():List<RecentItem>
setItems(List<RecentItem>):void
stopActionMode():void
removeSelected():void
-items
0..*
<<Java Class>>
RecentItem
org.liberty.android.fantastischmemo.ui
dbName: String
dbPath: String
dbInfo: String
index: int
RecentItem()

Card Editor


<<Java Class>>
CardEditor
org.liberty.android.fantastischmemo.ui
ACTIVITY_IMAGE_FILE: int
ACTIVITY_AUDIO_FILE: int
PERMISSION_REQUEST_RECORD_AUDIO: int
currentCard: Card
prevCard: Card
prevOrdinal: Integer
currentCardId: Integer
questionEdit: EditText
answerEdit: EditText
categoryText: TextView
categoryButton: Button
noteEdit: EditText
addRadio: RadioGroup
idEntry: EditText
lastLearnDateEntry: EditText
nextLearnDateEntry: EditText
gradeEntry: EditText
easinessEntry: EditText
acqRepsEntry: EditText
retRepsEntry: EditText
lapsesEntry: EditText
acqRepsSinceLapseEntry: EditText
retRepsSinceLapseEntry: EditText
addBack: boolean
isEditNew: boolean
dbName: String
dbPath: String
cardDao: CardDao
categoryDao: CategoryDao
learningDataDao: LearningDataDao
helper: AnyMemoDBOpenHelper
originalQuestion: String
originalAnswer: String
originalNote: String
EXTRA_DBPATH: String
EXTRA_CARD_ID: String
EXTRA_RESULT_CARD_ID: String
EXTRA_IS_EDIT_NEW: String
ISO_TIME_FORMAT: SimpleDateFormat
categoryButtonClickListener: OnClickListener
categoryResultListener: CategoryEditorResultListener
CardEditor()
onCreate(Bundle):void
onDestroy():void
restartActivity():void
onBackPressed():void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
onRequestPermissionsResult(int,String[],int[]):void
setAdvancedOptionsListener():void
isViewEligibleToEditAudio():boolean
showConfirmDialog(String,OnClickListener):void
audioPreviouslyExists():boolean
addExistingAudio():void
addNewAudio():void
removeAudio():void
startAudioBrowser():void
startAudioRecorderWithPermissionCheck():void
startAudioRecorder():void
addTextToView(EditText,String):void
onActivityResult(int,int,Intent):void
onConfigurationChanged(Configuration):void
setInitRadioButton():void
updateViews():void
updateCategoryView():void
<<Java Class>>
SaveCardTask
org.liberty.android.fantastischmemo.ui
progressDialog: ProgressDialog
SaveCardTask()
onPreExecute():void
doInBackground(Void[]):Void
onPostExecute(Void):void
<<Java Class>>
InitTask
org.liberty.android.fantastischmemo.ui
progressDialog: ProgressDialog
InitTask()
onPreExecute():void
doInBackground(Void[]):Void
onPostExecute(Void):void
-initTask
0..1

Infrastructure

For release one, no additional libraries or packages were added. The only libraries or packages used were already in use by the application. The main package that was made use of for development was OrmLite for the Dao (data access object) package. However, this was for the large story of the release which was pushed back to release two.

Code

File path with clickable GitHub link	Purpose (1 line description)
app/src/main/java/org/liberty/android/fantastischmemo/ui/CardEditor.java	Logic to merge Edit and Detail views and to show hidden Advanced Options.
app/src/main/res/layout/card_editor_layout.xml	Merge of edit and detail layout

Testing and Continuous Integration

Test File path	What is it testing (1 line description)
/app/src/androidTest/java/org/liberty/android/fantastischmemo/test/Filter/FilterTest.java	Unit tests for the DeckMap class methods: getAllTags(), getTagByName(), filterDecksByTags()
Manual Test for Issue #11	Testing to make sure a user can select the preview option in card list.

For the continuous integration environment, we are using Jenkins. The server is running Jenkins in its own Docker container. The server is a dedicated home server running Linux, specifically Ubuntu Server 16.04.3 LTS. The technical specifications of the server are: Intel-I3 processor, 8GB of ram, 500GB Hard Drive with a 750GB Hard Drive for backups. Since the server is running on a dynamic IP Address assigned by my Internet Service Provider, I am running another custom application in another Docker container to update my DNS. The application would detect an IP Address change and update the DNS point to the new IP Address. The URL to the Jenkins server can be found here: http://jenkins.matthewteolis.com:8080, with the Jenkins job configuration here.

Our continuous integration job runs on every new pull request, and whenever that pull request is updated with a new commit, from GitHub. From there, then Jenkins job will pull the code and run the gradle commands to assemble the binaries for the project and run the tests in an emulator. If all the tests pass, then Jenkins will return a pass message to the GitHub pull request which would allow the branch to be merged. If there are failing tests, it will return a failing message to the GitHub pull request and would not allow a merge to be done.