ExpOCR! : Final Documentation

Description

Many people usually go for outings/restaurants/grocery shopping with friends and have to split the costs. Also, with limited earnings, it is preferable to maintain one's expense records and thus avoid splurging on unnecessary items. Currently, people have to use different apps for splitting expenses and maintaining their personal record, moreover, these apps do not do provide categorized summaries or bill scanning. This application, ExpOCR!, allows a user to track personal expenses, split the expenses, view summaries of transactions, a`nd scan items from the bills. With OCR people can Snap, assign, and be done!

Process Followed

This project followed the software development process called Extreme Programming (XP) which worked on a schedule that was partitioned into two week chunks called "sprints". Each sprint would start out with a meeting of the whole team. During this initial meeting the team discussed the tasks for the sprint that need to be completed. The team spent time taking each task and turning them into well defined partitions called user stories. Following the creation of the user stories, priorities were assigned to each of them. The team then made sure all goals were accounted for. In order to determine priorities, the team discussed the difficulty and expected time requirements for each user story. Tasks were considered complete once they had a user story, time requirement, and priority.

Once each user story was prioritized, the team was separated into four two-person groups. The workload was separated as evenly as possible between the groups. Generally speaking, the pairs were formed according to which team members were able to meet and work at the same time. The pairs were mixed in order to unify the team and to give everyone a chance to gain experience working in a different situation. If teammates were not constrained on being available, the pairs were mixed up so that the two same people would not be working together every sprint.

Once the pairs were formed, they each created a branch which let each pair make changes that would not affect the others. Following any the implementation of a feature, a pair would merge their branch as soon as possible. If any issues arose during the merging, they were handled by the pair that was attempting it.

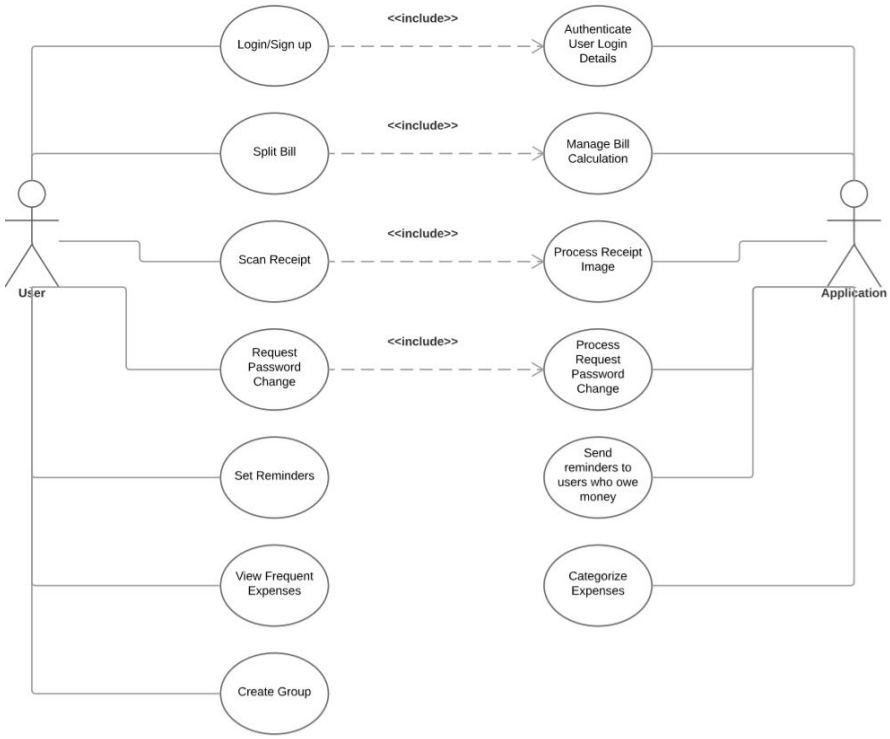
Before a pair even began coding the solution to their user stories, they needed to refactor the current code base. A pair finds code to be refactored by searching the code base for code smells. Code smells are poor uses of style and organization. Once these code smells were identified, they were then refactored, which involves re-writing the code but keeping the original functionality.

XP also uses what is called "test driven development". Test driven development involves writing unit tests before writing any code. Unit tests are the main way to verify, for the particular module, that code performs the actions that are intended. If changes caused any previously successful test to fail, then the pair would need to go back and make changes until all tests were successful, new and old.

After a week from the initial full team meeting, each pair completed their user stories on their branch. At this point, we had another full team meeting where we discussed, for the most part, any problems we came across. If, perhaps, there were no problems, we continued by going over the changes we made. This made sure all members of the team were on the same page.

Requirements And Specifications

Use Case Diagram



Actor GoalList

Actor	Task-Level Goal	Priority
User	Split Bill	1

Application	Categorize expenses	1
Application	Manage Bill Calculations	1
User	Login/sign up	2
User	View Frequent Expenses	2
Application	Authenticate User Login Details	2
User	Set Reminders	3
User	Request Password Change	3
User	Create Group	3
Application	Send reminders to users who owe money	3
Application	Process password reset request	3
User	Scan Receipt	4
Application	Process receipt image	4

Use Cases

Manage Bill Calculations:

Primary Actor	Application
Goal in Context	Calculate amounts to charge each person
Scope	Sub-System
Level	Lower-level
Stakeholders and Interest	The people splitting the bill
Precondition	The user has chosen who to share the bill with, as well as the total amount, and what the bill is for
Minimal Guarantees	The bill will be split evenly and charged to each person's account who is involved in the transaction

Login/Sign Up:

Primary Actor	User
Goal in Context	Create an account or sign into an existing one
Scope	System
Level	Summary
Stakeholders and Interest	The user who wants to create or login to an account
Precondition	If the user wants to login, he should already have an account. If the user does not have an existing account, there is no precondition.
Minimal Guarantees	The user will have an account that he/she can come back and log in to. The user cannot sign up with an email or Google or Facebook account if an account with those credentials already exist.

Authenticate User Login Details:

Primary Actor	Application (Server)
Goal in Context	Authentication of User's Login
Scope	System
Level	Subfunction
Stakeholders and Interest	User: wants to use the app
Precondition	The user is an authorized user. No guest account
Minimal Guarantees	User will be able to see a friendly message. Status - with error message.

View Frequent Expenses:

Primary Actor	User
Goal in Context	View frequent expenses
Scope	System
Level	Summary
Stakeholders and Interest	User: wants to view the frequent expenses and receive suggestions for cost-cutting
Precondition	If the user want to view the most frequent expenses, expense history should be available
Minimal Guarantees	User will be able to see the summary or be encouraged to use the application more

Split Bill:

Primary Actor	User
Goal in Context	Split bill among multiple parties.
Scope	System
Level	User Level
Stakeholders and Interest	User: Wants a bill split into parts. Application: Split bill correctly.
Precondition	User has an account.
Minimal Guarantees	Will properly split the bill.

Categorize Expenses:

Primary Actor	Application
Goal in Context	Tag expenses with a category and record into system.
Scope	System
Level	User Level
Stakeholders and Interest	Application: Wants to organize expenses for analysis. User: Wants to view past expenses by category.
Precondition	User has an account.
Minimal Guarantees	The expense will be recorded by default with a Misc. tag.

Set Reminders:

Primary Actor	User
Goal in Context	Set reminders for future financial plans
Scope	System-the mechanism to interact with the system to remind the users of important financial plans
Level	User
Stakeholders and Interest	User: Wants to be able to set and receive reminders of financial plans.
Precondition	User logged in
Minimal Guarantees	Set reminders only for logged in users. Users will only be reminded of plans set after the current date and before due date.

Request password change:

Primary Actor	User
Goal in Context	Send request to change user account password with email or password authentication
Scope	System - Part of the system storing user login information and account details
Level	User
Stakeholders and Interest	User - Wants to be able to control password on their account or recover password if they forget it and wants their account to be secure so others can't change their information
Precondition	User is logged in or user has an email authentication code
Minimal Guarantees	Minimal Guarantees: Changes password only if user can prove they are the owner of the account they are attempting to modify, new password must be reasonably secure (capitals, lowercase, numbers, 8 chars, etc.)

User Stories

4 Pairs * 12 Hrs/Iteration = 48Hrs/Iteration | 1 Hour/Unit | 48 Units/Iteration

Iteration 1 (Jan 23 - Feb 6)

actual	estimated	story description
5 units	3 units	Write initial proposal for project
2 units	2 units	Form the team
1 unit	1 unit	Project setup (Github group and Trello group)
2 unit	2 unit	Decide on coding style, testing framework, team meeting time
5 units	5 units	Get familiar with Language, framework, testing framework

Iteration 2 (Feb 6 - Feb 20)

actual	estimated	Pair assigned	story description
8 units	6 units	Mihika & Lanxiao	Create Login Page
6 units	6 units	Anthony & Eric	Create Main Menu Page
	6 units	Dan & Conner	Set up MySQL server and look into OCR
6 units	6 units	Brianna & Vimal	Simple Camera page
4 units * 4	5units * 4	Everyone	Set up Android Studio and follow the Hello World tutorial for Android Development (here)

Iteration 3 (Feb 20 - March 6)

actual	estimated	Pair assigned	story description
5 units	5 unit	Lanxiao Dan	Create server tables and test data
5 units	5 units	Conner Eric	Web server in python.
8 units	8 units	Everyone	Create application wireframe
4 units	4 units	Everyone	Discuss structure of server tables
5 units	5 units	Lanxiao Dan	Create Add transaction page
5 units	5 units	Anthony Vimal	Create expenses page
6 units	5 units	Mihika Brianna	Create groups page
4 units	3 Units	Anthony Vimal	Update Main Page
7 units	5 units	Mihika Brianna	Friends page
3 units	3 Units	Conner Eric	Try and read data with OCR

Iteration 4 (March 6 - 20)

actual	estimated	Pair assigned	story description
2 units	2 units	Dan Vimal	Parse sql output to json
4 units	4 units	Mihika Anthony	Request friend information from database (front end)
6 units	6 units	Mihika Anthony	Fix friends UI
4 units	4 units	Brianna Conner	Request friend information from database (back end)
4 units	4 units	Lanxiao Eric	User authentication with database (back end)
4 units	4 units	Lanxiao Eric	User authentication with database (front end)
4 units	4 units	Lanxiao Eric	Add user information to database (front end)
4 units	4 units	Dan Vimal	Add transaction information to database (front end)
1 unit	1 unit	Lanxiao Eric	Add user information to database (back end)
4 units	4 units	Brianna Conner	Add transaction information to database (back end)
4 units	4 units	Brianna Conner	Request transaction information from database (friend, your) (back end)
4 units	4 units	Dan Vimal	Request transaction information from database (friend, your) (front end)

Iteration 5 (March 20 - April 11)

actual	estimated	Pair assigned	story description
3 units	3 units	Eric & Dan	Create a group & add info to the database (back end)
4 units	3 units	Mihika & Conner	Create Recycler view for groups page (front end)
2 units	2 units	Mihika & Conner	Create Recycler view for groups page (back end)
2 units	2 units	Mihika & Conner	Layout for creating group/add members
4 units	4 units	Anthony & Brianna	Handle forgot password + Testing (back end)
4 units	4 units	Anthony & Brianna	Handle forgot password (front end)
9 units	6 units	Eric & Dan	Handle simplification of money owed (group)
2 units	2 units	Anthony & Brianna	Email verification for signing up (back end)
2 units	2 units	Anthony & Brianna	Email verification for forgot password
2 units	2 units	Lanxiao & Vimal	Design Summary Page
6 units	6 units	Lanxiao & Vimal	Create summary page including graphs (front end)
4 units	4 units	Lanxiao & Vimal	Create summary page (back end)
4 units	2 units	Anthony & Brianna	Send receipt image to server (front end)
4 units	2 units	Anthony & Brianna	Restore receipt image at server (back end)
3 units	4 units	Anthony & Brianna	Get text extraction of receipt image using OCR API
4 units	4 units	Anthony & Brianna	Set up Azure Server

Iteration 6 (March 20 - April 30)

actual	estimated	Pair Assigned	story description
2 units	2 units	Lanxiao & Vimal	Add groups transactions (back end)
4 units	4 units	Mihika & Eric	Create a group & add info to the database (front end)
4 units	4 units	Mihika & Eric	Update friends page recycler & individual balances (front & back end, net balance calculation)
2 units	2 units	Lanxiao & Vimal	Update groups page recycler & individual balances (back end net balance calculation)
4 units	6 units	Brianna & Dan	Add transaction for groups (front end)
2 units	2 units	Lanxiao & Vimal	Update group transactions table
4 units	2 units	Mihika & Eric	Add transaction for friends (front end)
4 units	4 units	Anthony (+ Eric?)	Implement settle up for groups
2 units	2 units	Dan & Brianna	Implement delete transaction feature (friends)
2 units	2 units	Dan & Brianna	Implement delete transaction feature (groups)
4 units	6 units	Lanxiao & Vimal	Implement delete groups and friends
4 units	4 units	Anthony & Conner	Connect Facebook Login and fix bugs

6 units	6 units	Conner, Mihika, Eric, Vimal, Lanxiao, Anthony, Dan, Brianna	Create manual test plan ● verify CreateData.sql and use as basis - Dan ● Groups ● Create Group - Mihika ● Add transaction - Brianna ● Settle Up - Eric ● Delete Group - Vimal ● Delete Transaction - Brianna ● Friends ● Add transaction - Mihika ● OCR - Anthony ● Settle Up - Brianna ● Delete Friend - Lanxiao ● Delete Transaction - Brianna ● Login/Logout - Conner ● Facebook Login ● Signup - Conner ● Forgot Password - Conner
6 units	4 units	Anthony & Conner	Finish OCR receipt recognition and validation
2 units	2 units	Mihika & Eric	Update Navigations Menu
2 units	2 units	Brianna & Dan	Fix friends settle up
4 units	4 units	Mihika & Eric	Update & Fix Expense Tab

Reflections and Lessons Learned:

Include personal reflections on the project and process by each team member. Describe what you learned.

Mihika Dave (mhdave2):

It has been an enriching experience working on this project, not only in terms of Software Engineering but also learning how to contribute to a team project. Before working on this project, I didn't have any experience with Software Development. This project has taught me a lot from proposing a project to be completed over a period of 3 months to actually being able to finish everything and reaching the publishing stage. I even plan to put ExpOCR! on the Google Play Store.

When we first proposed this project, it seemed to be an ambitious one, with lots of features and uncertainty of how feasible it is to implement OCR. However it was truly amazing to be able to work everything out and my learning curve has been exponential. I have learnt several new technologies, languages, frameworks and principles of software engineering. I learnt about UI/UX, frontend, backend, android, git, building UML diagrams. I learnt about XP which is one of the latest software engineering process and learnt how to implement features by breaking them into user stories.

Overall I think this project has honed my skills and prepared me to become a valuable software engineer in the outside industry. I consider the skills acquired extremely valuable for my upcoming internship in Android development at Facebook.

Brianna Ifft (bifft2):

Overall, I really enjoyed this project. Because I have never worked on a project quite this extensive before, I was a little overwhelmed at the beginning when I was trying to get all the components of the project up and running--Android Studio, a mysql server, the backend which eventually became django, and then mySQLWorkbench to interact with the sql tables. But, as a result of this project, I now feel like I have a much better understanding of all the moving parts of a holistic application, which I'm sure will help me out as I move on to a career.

Personally this project forced me to move out of my technical comfort zone. Although I had used java before, I had never created an Android application, so I was a little nervous about how I would adapt to this. However, I now feel very comfortable programming for android, and actually really enjoy doing it. I also had never really understood the connection between the front and back ends of an application before working on this project, but have learned so much about how the back and front ends of an application communicate through this project. Overall, I learned much more than I expected to by working on ExpOCR! this semester.

As far as the development process went, I thought it was pretty smooth. We used extreme programming, which is the same process I used for the group project in CS 427. I personally am a fan of this process so I really enjoyed using it. Our group broke the project into two week iterations and assigned pairs to work on user stories. Overall, I think that worked well to implement all the features. However, since this was such an extensive project, some group members went above and beyond their assigned user stories in order to implement all the features. All in all, I am thankful that I got to work on this project this semester.

Daniel Cummins (dpcummi2):

I found that this project was a good follow-up to the material I have been learning in Software Engineering I and II. I was able to apply many of the topics covered in the class in a genuine software development environment. Not every student gets this chance to work in a real life type situation. It was also helpful that the team was large because it encouraged a great deal of communication. Communication skills are difficult to teach in a traditional classroom setting and I found that was a particularly strong benefit of this project.

It was also good to reinforce the paradigm of using a software development process. There are many projects I have worked on in the past that either failed or did not meet expectations because they were not well structured. This course forced teams to follow a software development process, and while it felt like a hassle at times, I truly understand the value of such structure when dealing with a project this large. I will not necessarily say that Extreme Programming is my favorite software process, but just being exposed to it in action was a very valuable experience in my opinion.

Lanxiao Bai (lbai5):

This project provided me the first chance to work in group with people to develop a software that is somewhat valuable in real life. I think this is a good start as my career in the field of software engineering and make me more comfortable and confidence to move to my work in the summer.

And by strictly follow the agile software development process in this project, I start to feel that the software engineering methods theories are starting to make sense me, since I have experienced how this process can help our team to manage our project and make it better and better after each iteration.

Since I don't know android or django at all, I find this project challenging but rewarding. After all, a software engineer always have to learn my oneself in order to handle the needs one face in work. By working in groups, other team members also helped me a lot by showing how excellent their design and code are, from which there are a lot of things I can learn from. So I'm really appreciate to have the opportunity to work in this project and to have all these talented people to work with me on this project.

Eric Barron (ebarron2):

This project was unique compared to the project from Software Engineering 1 because we actually decided what project we were doing and how we would go about the iterations rather than being told what to do with our project. While this made the project more interesting, it also made the project a lot more difficult. There were a lot of challenges determining how we wanted the application to work and deciding what our goals and specifications would be compared to Software Engineering 1. This made it so we needed more time spent on deciding what we wanted to do with the project than in the previous class.

From this experience I learned more about how to make the software engineering process work. In Software Engineering 1, we were given a more strict specification on what to do and how to do it than in this project. In this class, I had to spend more effort thinking about and understanding how our project was going to work since it wasn't written out for us. Along with this, we were also using frameworks that I wasn't familiar with in Django and Android Studio. This made the project more difficult because I had to put more effort into making sure I understood how everything worked.

Because of the challenges I encountered with this project, I learned that I need to be more assertive in asking how we are implementing our project and having my ideas heard in the project. It was easy to get lost and have trouble understanding the implementation our specifications of our project since it wasn't directly written out for us and because I wasn't familiar with the frameworks we were using. If I was more adamant about understanding each new thing added to the project, the parts I worked on would have been easier since I would have had a better understanding of what I was supposed to do and how I could implement it.

Vimal Patel (vnpatel3):

This project was very beneficial to me because I think I learned more through this project than I have in most classes. I had programmed in Java before, but I still considered myself a beginner because I had never coded something nowhere near as complex as ExpOCR. I also had never used mySQL or Git or Django or even Android Studio so I increased my skill set by a lot. Another big part of the project for me was learning how the backend and frontend communicate and realizing just how much goes into creating a complete and polished package and how everything works together was such a relief for me because I had never really understood until now.

The software process wasn’t as tedious to me as I found it to be last semester in CS 427. I actually didn’t mind it this semester and I felt like it really kept us organized and able to accomplish what we wanted to in time. I wouldn’t have minded using a different process this semester though so that I would have something to compare the extreme programming process to. In the end though it all worked out perfectly because everyone in the group was familiar with the process and the process is a good one in my opinion because it holds us accountable for completing things on time and focuses a lot on testing.

Architecture and design:

Our project is split into 2 parts, a front end interface and a back end server. For the front end of our project, we used Android Studio, and for the back end we used a python server with Django.

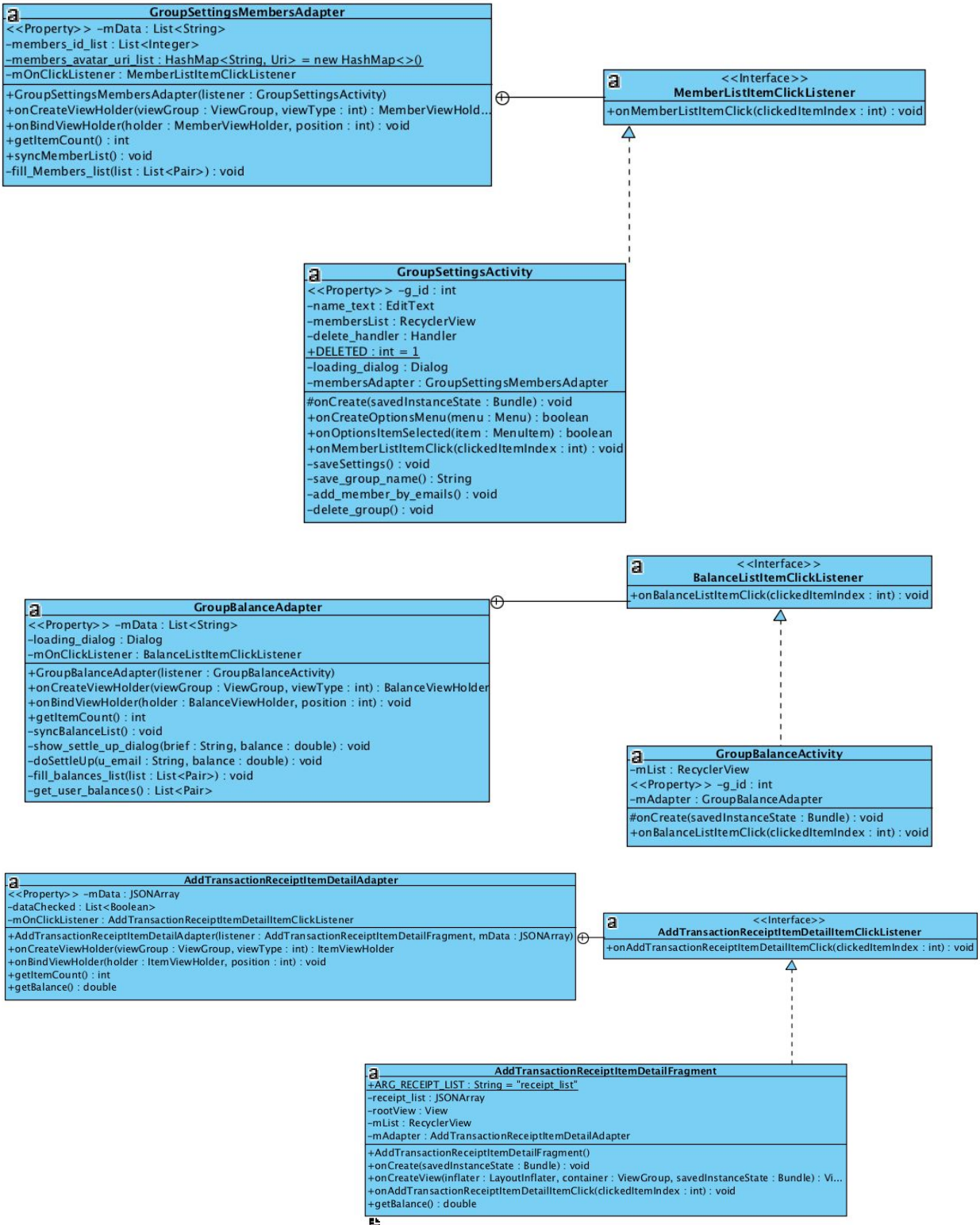
For our server, we use a sql database to store the information we receive from users. We have a table for users, transactions, groups, members (of a group), and group transactions. Our backend server is hosted on a Django server, which is broken up into multiple parts. We have a section for commands and a model having to do with user, which can be used to add users to the database and change information about users. We also have a section dedicated to transactions and adding and manipulating transactions on the server. Finally we have a section dedicated to groups which contains information on adding groups, group members, and group transactions to the server and changing information dealing with all of them. Both our transaction section and our group section use some information from the user section in order to get all the information about the users they are adding transactions or groups for. Our front end accesses the functions in each of these sections through separate urls contained in a urls.py file. Our backend also contains functions for parsing data from OCR.

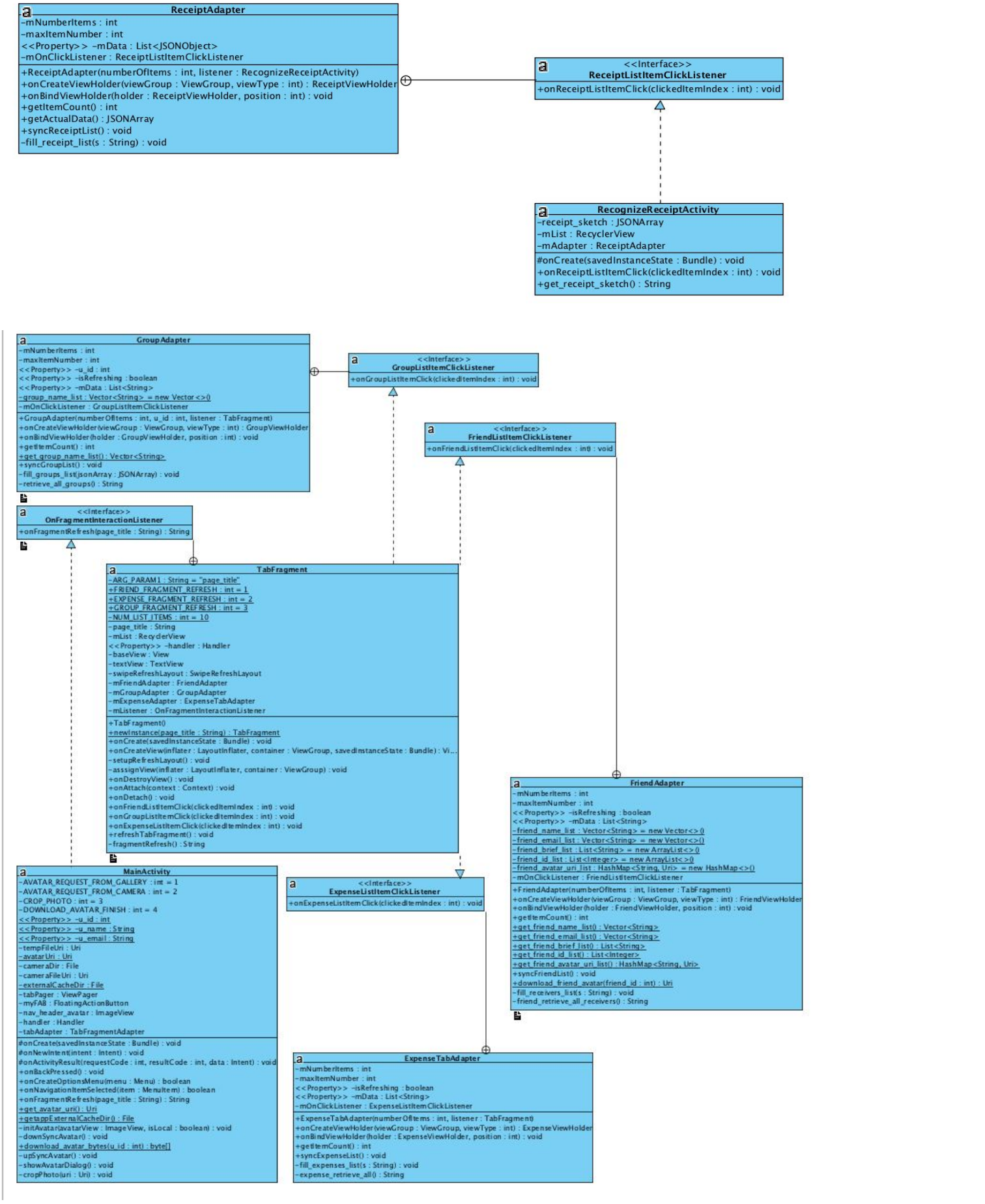
Our front end is an application on Android Studio. When the user first opens the app, they see the login screen. When the user logs in or creates an account, they will be signed in and the frond end will keep track of the user who is logged in. Our front end is separated into multiple pages. We have a group page for adding group members and viewing information relating to the group, a page for friends and adding or viewing transactions between friends, a page for when a user forgets their password, a page for receipt splitting using OCR, and a page for making payments. Our friends work on a system where whenever a user makes a transaction with another user, the other user becomes their friend. For our forgot password page, we send an email to the user with a code. This code is stored in our backend for a short period of time. If the user correctly inputs the code before the code expires then they are brought to a page to change their password.

Our front end connects with our back end using http requests. We write json strings to the back end. which parsers the strings, and returns a json string with whatever data the front end requests.

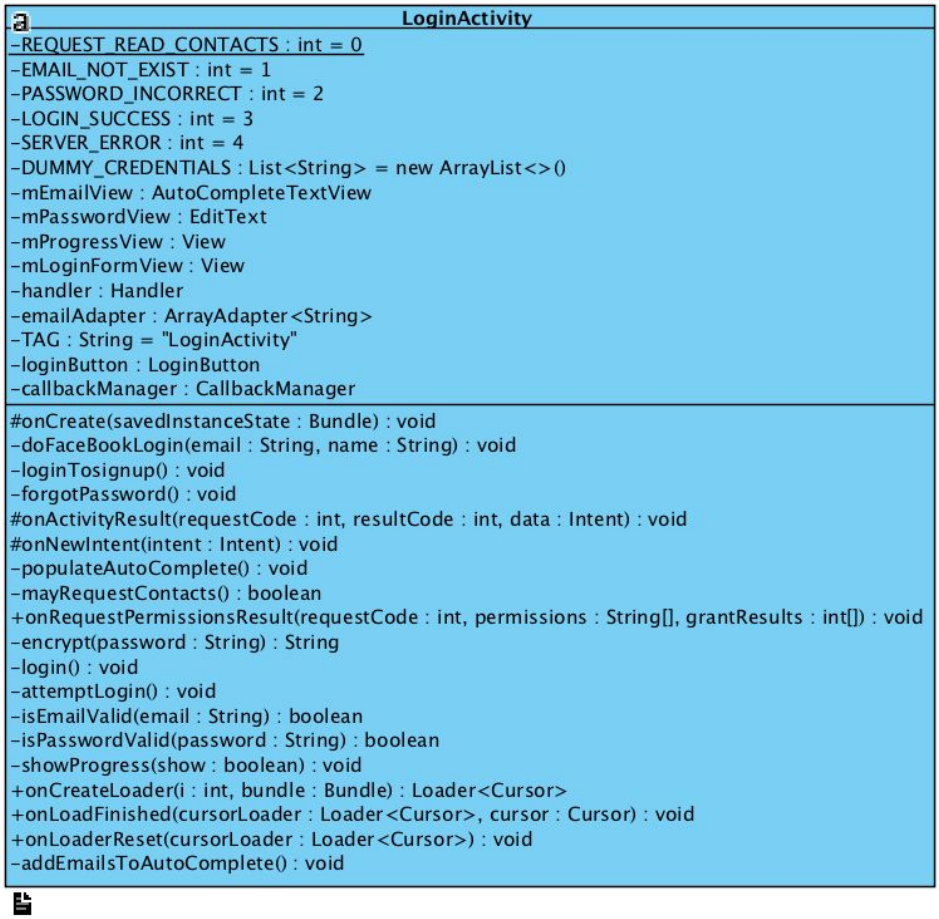
Our choice of using Django for our backend also influenced the way our backend was designed. It is because of our choice to use Django that our backend is split into multiple files and models for each part of our database, as these models are not necessary for a regular sql server. Similarly, our choice to use Android Studio meant that our project needed to be written in java.

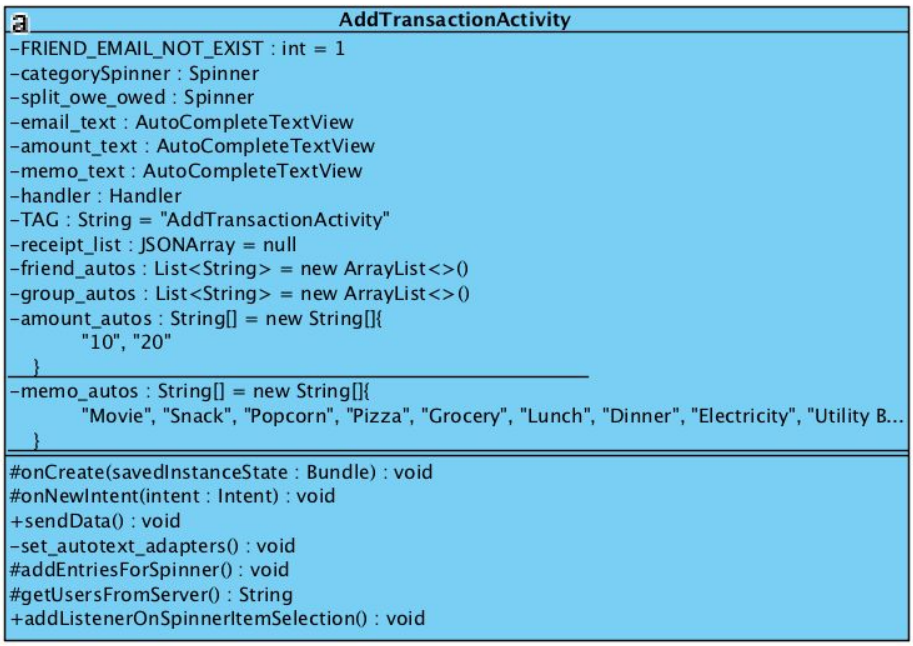
UML Diagrams

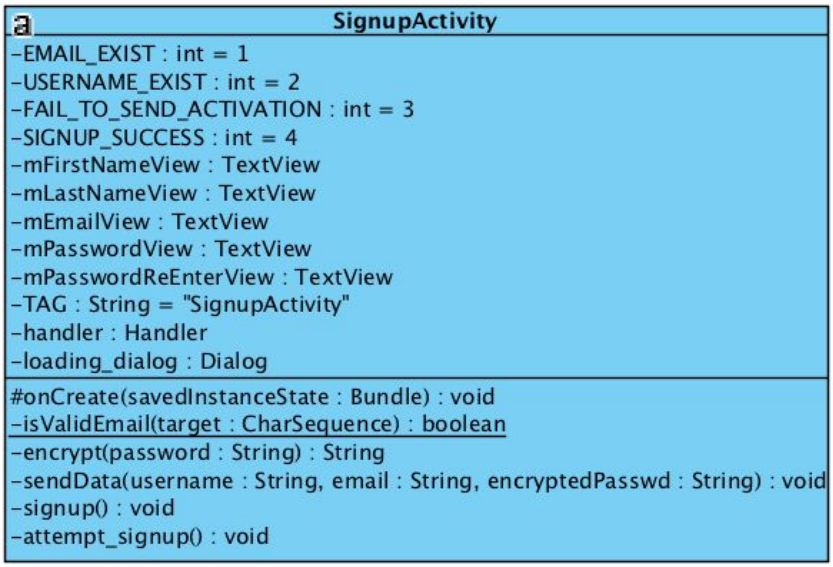

GroupSettingsMembersAdapter
<<Property>> -mData : List<String>
-members_id_list : List<Integer>
-members_avatar_uri_list : HashMap<String, Uri> = new HashMap<>()
-mOnClickListener : MemberListItemClickListener
+GroupSettingsMembersAdapter(listener : GroupSettingsActivity)
+onCreateViewHolder(viewGroup : ViewGroup, viewType : int) : MemberViewHold...
+onBindViewHolder(holder : MemberViewHolder, position : int) : void
+getItemCount() : int
+syncMemberList() : void
-fill_Members_list(list : List<Pair>) : void
<<Interface>>
MemberListItemClickListener
+onMemberListItemClick(clickedItemIndex : int) : void
GroupSettingsActivity
<<Property>> -g_id : int
-name_text : EditText
-membersList : RecyclerView
-delete_handler : Handler
+DELETED : int = 1
-loading_dialog : Dialog
-membersAdapter : GroupSettingsMembersAdapter
#onCreate(savedInstanceState : Bundle) : void
+onCreateOptionsMenu(menu : Menu) : boolean
+onOptionsItemSelected(item : MenuItem) : boolean
+onMemberListItemClick(clickedItemIndex : int) : void
-saveSettings() : void
-save_group_name() : String
-add_member_by_emails() : void
-delete_group() : void
<<Interface>>
BalanceListItemClickListener
+onBalanceListItemClick(clickedItemIndex : int) : void
GroupBalanceAdapter
<<Property>> -mData : List<String>
-loading_dialog : Dialog
-mOnClickListener : BalanceListItemClickListener
+GroupBalanceAdapter(listener : GroupBalanceActivity)
+onCreateViewHolder(viewGroup : ViewGroup, viewType : int) : BalanceViewHolder
+onBindViewHolder(holder : BalanceViewHolder, position : int) : void
+getItemCount() : int
-syncBalanceList() : void
-show_settle_up_dialog(brief : String, balance : double) : void
-doSettleUp(u_email : String, balance : double) : void
-fill_balances_list(list : List<Pair>) : void
-get_user_balances() : List<Pair>
GroupBalanceActivity
-mList : RecyclerView
<<Property>> -g_id : int
-mAdapter : GroupBalanceAdapter
#onCreate(savedInstanceState : Bundle) : void
+onBalanceListItemClick(clickedItemIndex : int) : void
AddTransactionReceiptItemDetailAdapter
<<Property>> -mData : JSONArray
-dataChecked : List<Boolean>
-mOnClickListener : AddTransactionReceiptItemDetailItemClickListener
+AddTransactionReceiptItemDetailAdapter(listener : AddTransactionReceiptItemDetailFragment, mData : JSONArray)
+onCreateViewHolder(viewGroup : ViewGroup, viewType : int) : ItemViewHolder
+onBindViewHolder(holder : ItemViewHolder, position : int) : void
+getItemCount() : int
+getBalance() : double
<<Interface>>
AddTransactionReceiptItemDetailItemClickListener
+onAddTransactionReceiptItemDetailItemClick(clickedItemIndex : int) : void
AddTransactionReceiptItemDetailFragment
+ARG_RECEIPT_LIST : String = "receipt_list"
-receipt_list : JSONArray
-rootView : View
-mList : RecyclerView
-mAdapter : AddTransactionReceiptItemDetailAdapter
+AddTransactionReceiptItemDetailFragment()
+onCreate(savedInstanceState : Bundle) : void
+onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : Vi...
+onAddTransactionReceiptItemDetailItemClick(clickedItemIndex : int) : void
+getBalance() : double

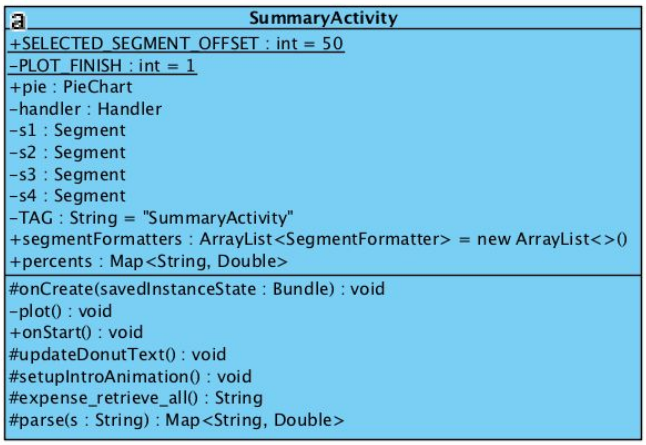

ReceiptAdapter
-mNumberItems : int
-maxItemNumber : int
<<Property>> -mData : List<JSONObject>
-mOnClickListener : ReceiptListItemClickListener
+ReceiptAdapter(numberOfItems : int, listener : RecognizeReceiptActivity)
+onCreateViewHolder(viewGroup : ViewGroup, viewType : int) : ReceiptViewHolder
+onBindViewHolder(holder : ReceiptViewHolder, position : int) : void
+getItemCount() : int
+getActualData() : JSONArray
+syncReceiptList() : void
-fill_receipt_list(s : String) : void
<<Interface>>
ReceiptListItemClickListener
+onReceiptListItemClick(clickedItemIndex : int) : void
RecognizeReceiptActivity
-receipt_sketch : JSONArray
-mList : RecyclerView
-mAdapter : ReceiptAdapter
#onCreate(savedInstanceState : Bundle) : void
+onReceiptListItemClick(clickedItemIndex : int) : void
+get_receipt_sketch() : String
GroupAdapter
-mNumberItems : int
-maxItemNumber : int
<<Property>> -u_id : int
<<Property>> -isRefreshing : boolean
<<Property>> -mData : List<String>
-group_name_list : Vector<String> = new Vector<>()
-mOnClickListener : GroupListItemClickListener
+GroupAdapter(numberOfItems : int, u_id : int, listener : TabFragment)
+onCreateViewHolder(viewGroup : ViewGroup, viewType : int) : GroupViewHolder
+onBindViewHolder(holder : GroupViewHolder, position : int) : void
+getItemCount() : int
+get_group_name_list() : Vector<String>
+syncGroupList() : void
-fill_groups_list(jsonArray : JSONArray) : void
-retrieve_all_groups() : String
<<Interface>>
GroupListItemClickListener
+onGroupListItemClick(clickedItemIndex : int) : void
<<Interface>>
FriendListItemClickListener
+onFriendListItemClick(clickedItemIndex : int) : void
<<Interface>>
OnFragmentInteractionListener
+onFragmentRefresh(page_title : String) : String
TabFragment
-ARG_PARAM1 : String = "page_title"
+FRIEND_FRAGMENT_REFRESH : int = 1
+EXPENSE_FRAGMENT_REFRESH : int = 2
+GROUP_FRAGMENT_REFRESH : int = 3
-NUM_LIST_ITEMS : int = 10
-page_title : String
-mList : RecyclerView
<<Property>> -handler : Handler
-baseView : View
-textView : TextView
-swipeRefreshLayout : SwipeRefreshLayout
-mFriendAdapter : FriendAdapter
-mGroupAdapter : GroupAdapter
-mExpenseAdapter : ExpenseTabAdapter
-mListener : OnFragmentInteractionListener
+TabFragment()
+newInstance(page_title : String) : TabFragment
+onCreate(savedInstanceState : Bundle) : void
+onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : Vi...
-setupRefreshLayout() : void
-asssignView(inflater : LayoutInflater, container : ViewGroup) : void
+onDestroyView() : void
+onAttach(context : Context) : void
+onDetach() : void
+onFriendListItemClick(clickedItemIndex : int) : void
+onGroupListItemClick(clickedItemIndex : int) : void
+onExpenseListItemClick(clickedItemIndex : int) : void
+refreshTabFragment() : void
-fragmentRefresh() : String
FriendAdapter
-mNumberItems : int
-maxItemNumber : int
<<Property>> -isRefreshing : boolean
<<Property>> -mData : List<String>
-friend_name_list : Vector<String> = new Vector<>()
-friend_email_list : Vector<String> = new Vector<>()
-friend_brief_list : List<String> = new ArrayList<>()
-friend_id_list : List<Integer> = new ArrayList<>()
-friend_avatar_uri_list : HashMap<String, Uri> = new HashMap<>()
-mOnClickListener : FriendListItemClickListener
+FriendAdapter(numberOfItems : int, listener : TabFragment)
+onCreateViewHolder(viewGroup : ViewGroup, viewType : int) : FriendViewHolder
+onBindViewHolder(holder : FriendViewHolder, position : int) : void
+getItemCount() : int
+get_friend_name_list() : Vector<String>
+get_friend_email_list() : Vector<String>
+get_friend_brief_list() : List<String>
+get_friend_id_list() : List<Integer>
+get_friend_avatar_uri_list() : HashMap<String, Uri>
+syncFriendList() : void
+download_friend_avatar(friend_id : int) : Uri
-fill_receivers_list(s : String) : void
-friend_retrieve_all_receivers() : String
MainActivity
-AVATAR_REQUEST_FROM_GALLERY : int = 1
-AVATAR_REQUEST_FROM_CAMERA : int = 2
-CROP_PHOTO : int = 3
-DOWNLOAD_AVATAR_FINISH : int = 4
<<Property>> -u_id : int
<<Property>> -u_name : String
<<Property>> -u_email : String
-tempFileUri : Uri
-avatarUri : Uri
-cameraDir : File
-cameraFileUri : Uri
-externalCacheDir : File
-tabPager : ViewPager
-myFAB : FloatingActionButton
-nav_header_avatar : ImageView
-handler : Handler
-tabAdapter : TabFragmentAdapter
#onCreate(savedInstanceState : Bundle) : void
#onNewIntent(intent : Intent) : void
#onActivityResult(requestCode : int, resultCode : int, data : Intent) : void
+onBackPressed() : void
+onCreateOptionsMenu(menu : Menu) : boolean
+onNavigationItemSelected(item : MenuItem) : boolean
+onFragmentRefresh(page_title : String) : String
+get_avatar_uri() : Uri
+getappExternalCacheDir() : File
-initAvatar(avatarView : ImageView, isLocal : boolean) : void
-downSyncAvatar() : void
+download_avatar_bytes(u_id : int) : byte[]
-upSyncAvatar() : void
-showAvatarDialog() : void
-cropPhoto(uri : Uri) : void
<<Interface>>
ExpenseListItemClickListener
+onExpenseListItemClick(clickedItemIndex : int) : void
ExpenseTabAdapter
-mNumberItems : int
-maxItemNumber : int
<<Property>> -isRefreshing : boolean
<<Property>> -mData : List<String>
-mOnClickListener : ExpenseListItemClickListener
+ExpenseTabAdapter(numberOfItems : int, listener : TabFragment)
+onCreateViewHolder(viewGroup : ViewGroup, viewType : int) : ExpenseViewHolder
+onBindViewHolder(holder : ExpenseViewHolder, position : int) : void
+getItemCount() : int
+syncExpenseList() : void
-fill_expenses_list(s : String) : void
-expense_retrieve_all() : String

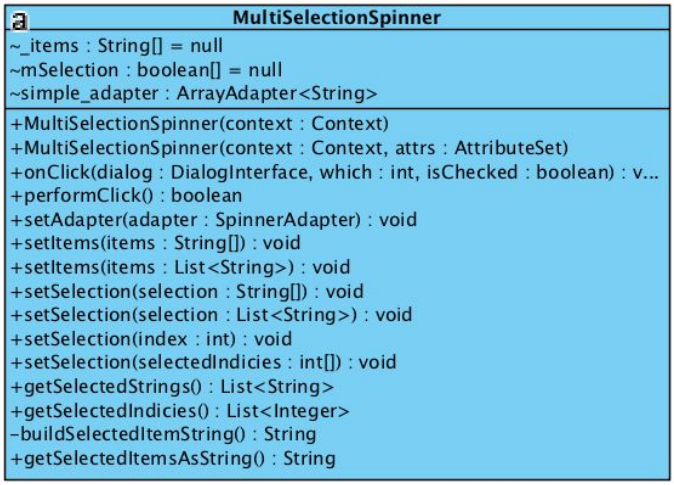
GroupTransaction
-amount : double
-date : Calendar
-memo : String
-category : String
-u_name : String
-u_email : String
-gid : int
-uid : int
+getGroupTransactionsFromServer(g_id : int) : List<GroupTransaction>
+getOwedAmounts(g_id : int, u_id : int) : List<Pair>
+getUserNetBalance(g_id : int, u_id : int) : double
+getUserNetBalances(g_id : int) : List<Pair>
+getMemberBriefs(mids : List<Integer>) : List<Pair>
+getMembers(g_id : int) : List<Integer>
+main(args : String[]) : void
ExpenseAdapter
-mContext : Context
-mInflater : LayoutInflater
-mDataSource : Expense
+ExpenseAdapter(context : Context, items : ArrayList<Expense>)
+getCount() : int
+getItem(position : int) : Object
+getItemId(position : int) : long
+getView(position : int, convertView : View, parent : ViewGroup) : Vi...
+getData() : List<Expense>
+getNetBalance() : double
IndividualGroupActivity
-mListView : ListView
-receiver_id : int
-g_id : int
-g_name : String
-mAddTransactionButton : Button
-currentPosition : int
~data : Expense
#onCreate(savedInstanceState : Bundle) : void
+onCreateContextMenu(menu : ContextMenu, v : View, menuInfo : ContextMenuInfo) : void
+onContextItemSelected(item : MenuItem) : boolean
-delete_transaction(transactionDate : String) : String
#onNewIntent(intent : Intent) : void
-group_get_transaction_list_for(g_id : String, u_id : int) : String
-group_get_transactions_by_t_id(t_id : int) : String
ChangePasswordActivity
-mEmailView : AutoCompleteTextView
-mPasswordView : EditText
-mReenterPasswordView : EditText
-TAG : String = "ChangePasswordActivity"
-handler : Handler
-CHANGE_PWD_SUCCESS : int = 1
-userEmail : String
-loading_dialog : Dialog
#onCreate(savedInstanceState : Bundle) : void
-encrypt(password : String) : String
-attemptChangePassword() : void
-changePassword() : void
Expense
<<Property>> -balance : Double
<<Property>> -date : Calendar
<<Property>> -id : int
<<Property>> -expense : String
+Expense()
+Expense(id : int, expense : String, balance : double, date : String)
+Expense(expense : String, balance : double, date : String)
+getRecipesFromFile(filename : String, context : Context) : ArrayList<Expense>
-loadJsonFromAsset(filename : String, context : Context) : String
AddTransactionReceiptItemListActivity
-mTwoPane : boolean
-receipt_list : JSONArray
-friend_list : List<String>
-balance_list : List<Double>
-selectedFriend : int = 0
-recyclerView : RecyclerView
#onCreate(savedInstanceState : Bundle) : void
#onNewIntent(intent : Intent) : void
-setupRecyclerView() : void
AddGroupMembersActivity
-group_name : String
-u_id : int = MainActivity.getU_id()
-group_members : MultiAutoCompleteTextView
-friend_autos : Vector = new Vector()
~add_button : Button
#onCreate(savedInstanceState : Bundle) : void
-set_autotext_adapters() : void
+sendData() : void
TabFragmentAdapter
-fm : FragmentManager
-tab_titles : String[]
-tab_fragments : Fragment[]
+TabFragmentAdapter(fm : FragmentManager)
+getItem(position : int) : Fragment
+getCount() : int
+getPageTitle(position : int) : CharSequence
+refreshTabs() : void
CreateGroupActivity
-group_name : EditText
-create_group_button : Button
-u_id : int
#onCreate(savedInstanceState : Bundle) : void
+attemptCreateGroup() : boolean
AddTransactionReceiptItemDetailActivity
-fragment : AddTransactionReceiptItemDetailFragment
#onCreate(savedInstanceState : Bundle) : void
+onOptionsItemSelected(item : MenuItem) : boolean

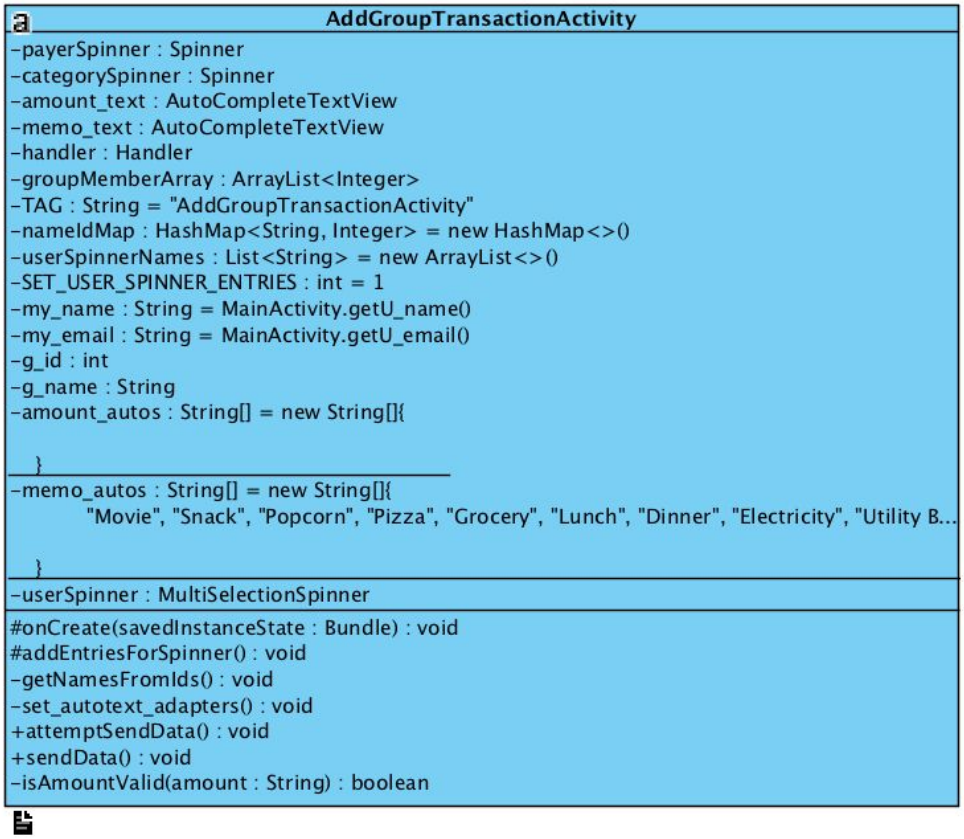

LoginActivity
-REQUEST_READ_CONTACTS : int = 0
-EMAIL_NOT_EXIST : int = 1
-PASSWORD_INCORRECT : int = 2
-LOGIN_SUCCESS : int = 3
-SERVER_ERROR : int = 4
-DUMMY_CREDENTIALS : List<String> = new ArrayList<>()
-mEmailView : AutoCompleteTextView
-mPasswordView : EditText
-mProgressView : View
-mLoginFormView : View
-handler : Handler
-emailAdapter : ArrayAdapter<String>
-TAG : String = "LoginActivity"
-loginButton : LoginButton
-callbackManager : CallbackManager
#onCreate(savedInstanceState : Bundle) : void
-doFaceBookLogin(email : String, name : String) : void
-loginTosignup() : void
-forgotPassword() : void
#onActivityResult(requestCode : int, resultCode : int, data : Intent) : void
#onNewIntent(intent : Intent) : void
-populateAutoComplete() : void
-mayRequestContacts() : boolean
+onRequestPermissionsResult(requestCode : int, permissions : String[], grantResults : int[]) : void
-encrypt(password : String) : String
-login() : void
-attemptLogin() : void
-isEmailValid(email : String) : boolean
-isPasswordValid(password : String) : boolean
-showProgress(show : boolean) : void
+onCreateLoader(i : int, bundle : Bundle) : Loader<Cursor>
+onLoadFinished(cursorLoader : Loader<Cursor>, cursor : Cursor) : void
+onLoaderReset(cursorLoader : Loader<Cursor>) : void
-addEmailsToAutoComplete() : void

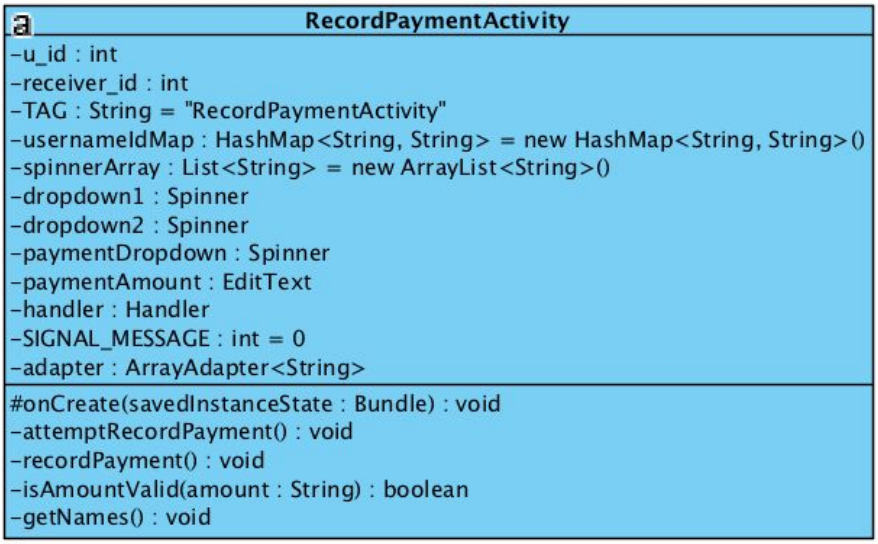

AddTransactionActivity
-FRIEND_EMAIL_NOT_EXIST : int = 1
-categorySpinner : Spinner
-split_owe_owed : Spinner
-email_text : AutoCompleteTextView
-amount_text : AutoCompleteTextView
-memo_text : AutoCompleteTextView
-handler : Handler
-TAG : String = "AddTransactionActivity"
-receipt_list : JSONArray = null
-friend_autos : List<String> = new ArrayList<>()
-group_autos : List<String> = new ArrayList<>()
-amount_autos : String[] = new String[]{
"10", "20"
}
-memo_autos : String[] = new String[]{
"Movie", "Snack", "Popcorn", "Pizza", "Grocery", "Lunch", "Dinner", "Electricity", "Utility B...
}
#onCreate(savedInstanceState : Bundle) : void
#onNewIntent(intent : Intent) : void
+sendData() : void
-set_autotext_adapters() : void
#addEntriesForSpinner() : void
#getUsersFromServer() : String
+addListenerOnSpinnerItemSelection() : void

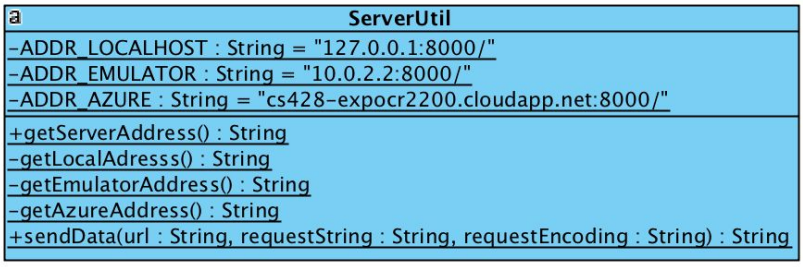

SignupActivity
-EMAIL_EXIST : int = 1
-USERNAME_EXIST : int = 2
-FAIL_TO_SEND_ACTIVATION : int = 3
-SIGNUP_SUCCESS : int = 4
-mFirstNameView : TextView
-mLastNameView : TextView
-mEmailView : TextView
-mPasswordView : TextView
-mPasswordReEnterView : TextView
-TAG : String = "SignupActivity"
-handler : Handler
-loading_dialog : Dialog
#onCreate(savedInstanceState : Bundle) : void
-isValidEmail(target : CharSequence) : boolean
-encrypt(password : String) : String
-sendData(username : String, email : String, encryptedPasswd : String) : void
-signup() : void
-attempt_signup() : void

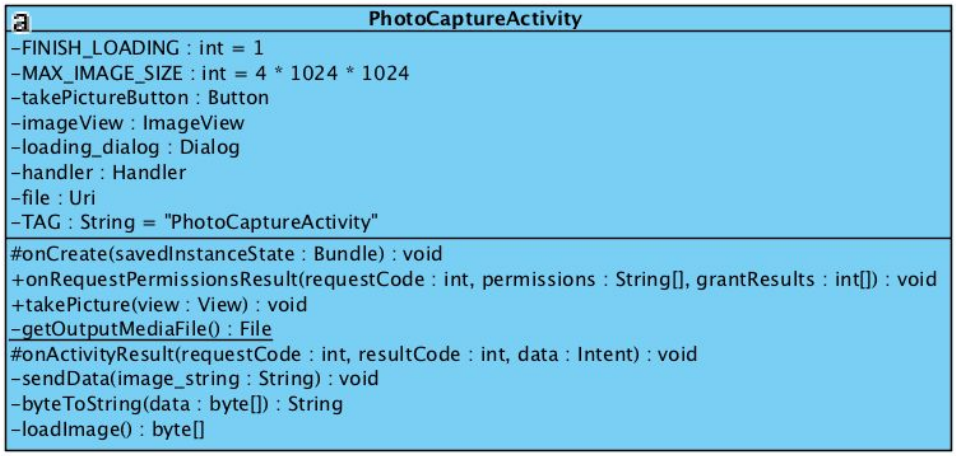

SummaryActivity
+SELECTED_SEGMENT_OFFSET : int = 50
-PLOT_FINISH : int = 1
+pie : PieChart
-handler : Handler
-s1 : Segment
-s2 : Segment
-s3 : Segment
-s4 : Segment
-TAG : String = "SummaryActivity"
+segmentFormatters : ArrayList<SegmentFormatter> = new ArrayList<>()
+percents : Map<String, Double>
#onCreate(savedInstanceState : Bundle) : void
-plot() : void
+onStart() : void
#updateDonutText() : void
#setupIntroAnimation() : void
#expense_retrieve_all() : String
#parse(s : String) : Map<String, Double>

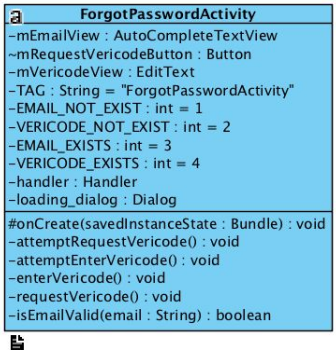

MultiSelectionSpinner
~_items : String[] = null
~mSelection : boolean[] = null
~simple_adapter : ArrayAdapter<String>
+MultiSelectionSpinner(context : Context)
+MultiSelectionSpinner(context : Context, attrs : AttributeSet)
+onClick(dialog : DialogInterface, which : int, isChecked : boolean) : v...
+performClick() : boolean
+setAdapter(adapter : SpinnerAdapter) : void
+setItems(items : String[]) : void
+setItems(items : List<String>) : void
+setSelection(selection : String[]) : void
+setSelection(selection : List<String>) : void
+setSelection(index : int) : void
+setSelection(selectedIndicies : int[]) : void
+getSelectedStrings() : List<String>
+getSelectedIndicies() : List<Integer>
-buildSelectedItemString() : String
+getSelectedItemsAsString() : String

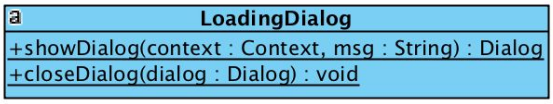

AddGroupTransactionActivity
-payerSpinner : Spinner
-categorySpinner : Spinner
-amount_text : AutoCompleteTextView
-memo_text : AutoCompleteTextView
-handler : Handler
-groupMemberArray : ArrayList<Integer>
-TAG : String = "AddGroupTransactionActivity"
-nameIdMap : HashMap<String, Integer> = new HashMap<>()
-userSpinnerNames : List<String> = new ArrayList<>()
-SET_USER_SPINNER_ENTRIES : int = 1
-my_name : String = MainActivity.getU_name()
-my_email : String = MainActivity.getU_email()
-g_id : int
-g_name : String
-amount_autos : String[] = new String[]{
}
-memo_autos : String[] = new String[]{
"Movie", "Snack", "Popcorn", "Pizza", "Grocery", "Lunch", "Dinner", "Electricity", "Utility B...
}
-userSpinner : MultiSelectionSpinner
#onCreate(savedInstanceState : Bundle) : void
#addEntriesForSpinner() : void
-getNamesFromIds() : void
-set_autotext_adapters() : void
+attemptSendData() : void
+sendData() : void
-isAmountValid(amount : String) : boolean

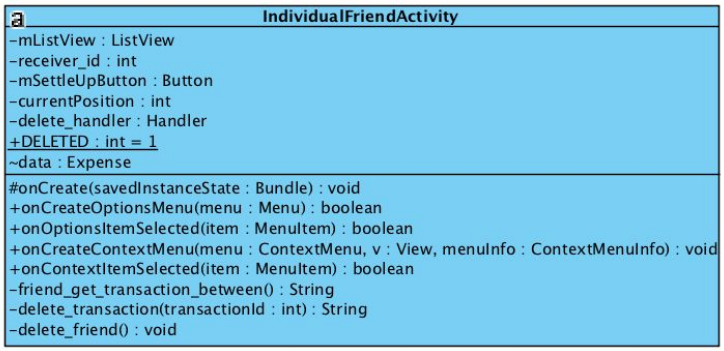

RecordPaymentActivity
-u_id : int
-receiver_id : int
-TAG : String = "RecordPaymentActivity"
-usernameIdMap : HashMap<String, String> = new HashMap<String, String>()
-spinnerArray : List<String> = new ArrayList<String>()
-dropdown1 : Spinner
-dropdown2 : Spinner
-paymentDropdown : Spinner
-paymentAmount : EditText
-handler : Handler
-SIGNAL_MESSAGE : int = 0
-adapter : ArrayAdapter<String>
#onCreate(savedInstanceState : Bundle) : void
-attemptRecordPayment() : void
-recordPayment() : void
-isAmountValid(amount : String) : boolean
-getNames() : void


ServerUtil
-ADDR_LOCALHOST : String = "127.0.0.1:8000/"
-ADDR_EMULATOR : String = "10.0.2.2:8000/"
-ADDR_AZURE : String = "cs428-expocr2200.cloudapp.net:8000/"
+getServerAddress() : String
-getLocalAdresss() : String
-getEmulatorAddress() : String
-getAzureAddress() : String
+sendData(url : String, requestString : String, requestEncoding : String) : String


PhotoCaptureActivity
-FINISH_LOADING : int = 1
-MAX_IMAGE_SIZE : int = 4 * 1024 * 1024
-takePictureButton : Button
-imageView : ImageView
-loading_dialog : Dialog
-handler : Handler
-file : Uri
-TAG : String = "PhotoCaptureActivity"
#onCreate(savedInstanceState : Bundle) : void
+onRequestPermissionsResult(requestCode : int, permissions : String[], grantResults : int[]) : void
+takePicture(view : View) : void
-getOutputMediaFile() : File
#onActivityResult(requestCode : int, resultCode : int, data : Intent) : void
-sendData(image_string : String) : void
-byteToString(data : byte[]) : String
-loadImage() : byte[]


ForgotPasswordActivity
-mEmailView : AutoCompleteTextView
~mRequestVericodeButton : Button
-mVericodeView : EditText
-TAG : String = "ForgotPasswordActivity"
-EMAIL_NOT_EXIST : int = 1
-VERICODE_NOT_EXIST : int = 2
-EMAIL_EXISTS : int = 3
-VERICODE_EXISTS : int = 4
-handler : Handler
-loading_dialog : Dialog
#onCreate(savedInstanceState : Bundle) : void
-attemptRequestVericode() : void
-attemptEnterVericode() : void
-enterVericode() : void
-requestVericode() : void
-isEmailValid(email : String) : boolean


LoadingDialog
+showDialog(context : Context, msg : String) : Dialog
+closeDialog(dialog : Dialog) : void


IndividualFriendActivity
-mListView : ListView
-receiver_id : int
-mSettleUpButton : Button
-currentPosition : int
-delete_handler : Handler
+DELETED : int = 1
~data : Expense
#onCreate(savedInstanceState : Bundle) : void
+onCreateOptionsMenu(menu : Menu) : boolean
+onOptionsItemSelected(item : MenuItem) : boolean
+onCreateContextMenu(menu : ContextMenu, v : View, menuInfo : ContextMenuInfo) : void
+onContextItemSelected(item : MenuItem) : boolean
-friend_get_transaction_between() : String
-delete_transaction(transactionId : int) : String
-delete_friend() : void