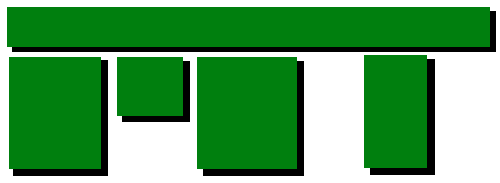
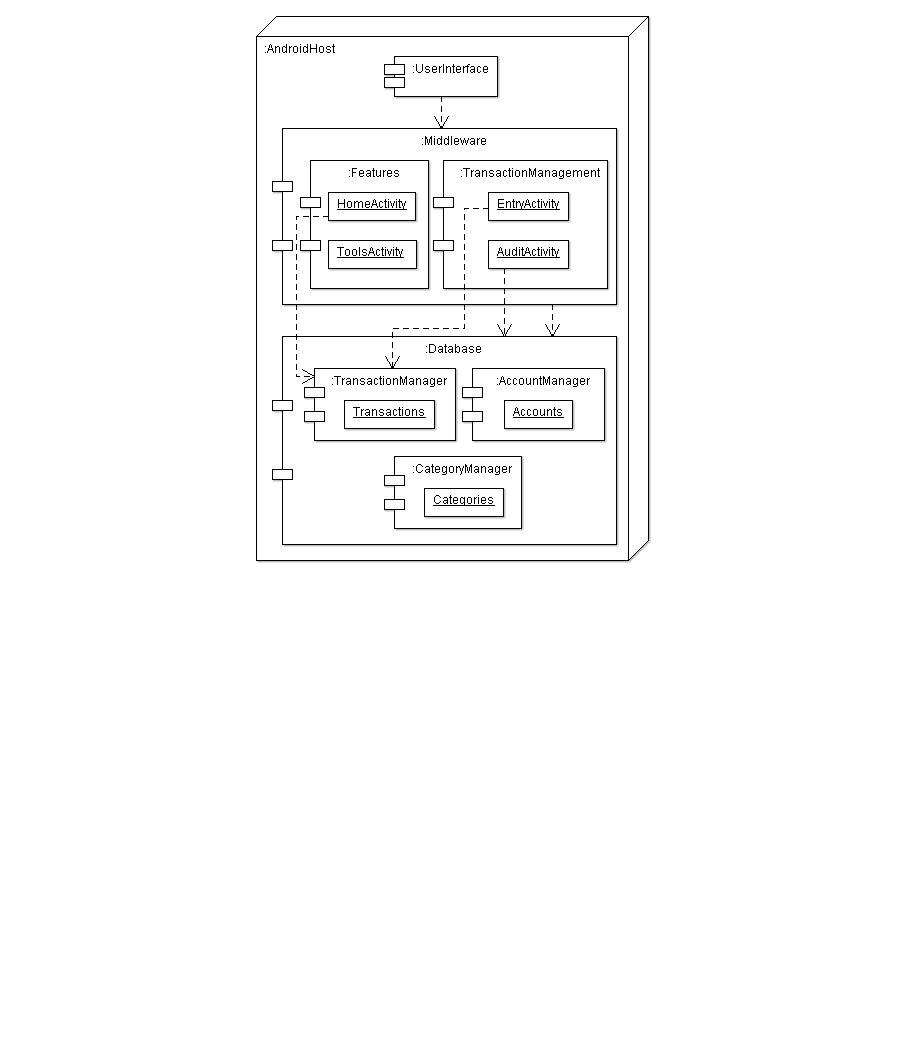
**MintTrack – System Design Document (SDD)**

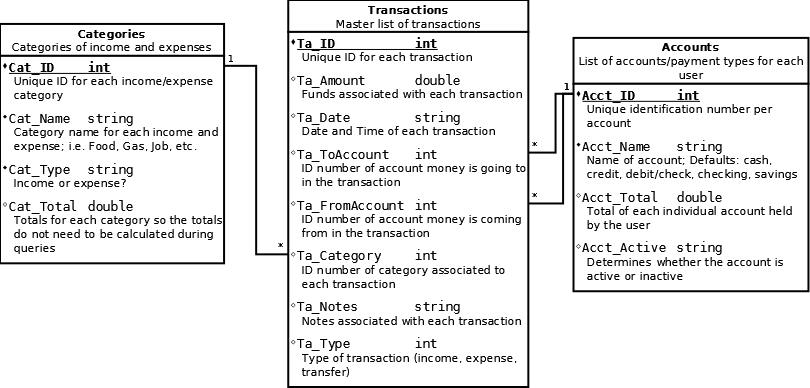
1. Introduction
   1. Purpose of the system - The purpose of this system is to allow users to keep track of their income and expenses through an application on their Android powered cell phones. This application promotes making smart purchases and budgeting funds to make the most of money on hand and in the bank.
   2. Design goals
      1. Usability: Design with the user in mind – It is our goal to make the application as easy as possible to use and configure. The tabbed interface and pop-up menu of the phone allows users to see all possible interactive options.
      2. Portability: For physical portability, seeing as the application is being built to work on a cellular device, users will have the ability to carry the system around with them wherever they go. For software portability, it is built specifically for devices with the Android OS, so it is only portable between different versions of that software.
      3. Encapsulation: The data types, methods, and database objects are hidden from the user beneath a user interface (UI). The information in the database’s transaction table can only be viewed when the user queries his or her transactions for custom reports. Additionally, methods and variables will only be viewable by their classes as private data members unless making a method or variable public is absolutely necessary (such is the case with the *onCreate* method for each tab).
      4. Data Abstraction: Each tab in the application and each table in the database has its own class in the application package resembling individual entities of the software.
      5. Availability: Constant, as long as there is a charge to the phone’s battery.
      6. Memory: Very little memory is required for the application to run. The database is a single flat file and the program itself only uses, at most, one activity at a time.
      7. Robustness: Error checking is performed on all user input.
   3. Definitions, acronyms, and abbreviations-
      1. Android: Google's mobile device operating system
      2. API: Application Programming Interface – a way for the programmer to interact with the system hardware
      3. APK: Stands for “Android Package.” The file format of Android applications.
      4. ChartDroid: An API for using graphing functionality in Android application development.
      5. Cursor: An iterator in a SQLite database. Used to access data in a table based on what column and row that data resides in. Since the SQLite database is a single flat file, a cursor may only move up and down single entries or between the first and last entries of a table.
      6. DBMS: Database Management System
      7. GUI: Graphical User Interface
      8. SD: Secure Digital – Protocol for an SD card reader
      9. SDK: “Software Development Kit”
      10. Spinner: a drop-down menu in the Android OS
      11. SQLite: A 'lite' version of database languages such as SQL and MySQL. SQLite database functionality is built into the Android operating system and consists of a single file that holds information and can be used much like a regular database.
      12. YATC: Yet Another Tip Calculator
   4. References
      1. <http://developer.android.com> – Android API documentation
      2. MintTrack Requirements Analysis Document (RAD)
   5. Overview – Through use of the Android API and SDK, MintTrack will be designed to be a useful, reliable, portable, and robust cell phone application. Users will find the program to be not only quick to respond, but also very easy to use to track their purchases and acquisitions.
2. Current software architecture – None available to reference.
3. Proposed System architecture
   1. Overview – MintTrack presents a fairly simple architecture in that it only consists of the application with a built-in database, and therefore uses no external services regularly. Upon exporting data, MintTrack will communicate with an SD card or Internet connection to e-mail a created file to a specified address.
   2. Subsystem decomposition - The proposed software architecture for this project closely resembles a 3-tier system. The top tier is the graphical user interface developed with XML. Secondly, the bottom tier is the database, developed through the use of SQLite that will hold all transaction, account and category information. Finally, the middle layer, or middleware, is developed using Java, and issues functionality between the user interface and the database such as adding, removing, and editing transactions, accounts and categories.



* 1. Hardware/software mapping – MintTrack will be an application on Google’s Android OS for the Motorola and HTC Droid cellular devices. Android API 1.5 is being used to develop the application, therefore any phone version 1.5 and higher will be able to support it. The UI will be implemented using XML and the functionality will be implemented using Java. SQLite will be used as the DBMS.

Package and deployment diagram for MintTrack:

* 1. Persistent data management – Persistent data will be stored in a SQLite database object provided by the Android OS inside the MintTrack application. The SQLite database is built to take up as little space as possible in the phone’s memory – it is a single, flat file that works nearly the same way as a relational database. When querying for information, a Cursor object is used to access columns and rows of each table. Below is the database design:



* 1. Access control and security – MintTrack is an Android based application with no network access and is only meant to be used by a single person per phone. The Android OS has a locking and password mechanism and thus the application does not require one. If multiple profiles were to be implemented in the program, then there would be need for access control and security, however, that is not the case. MintTrack does not keep track of important personal information such as credit card numbers or bank account numbers and therefore does not require an encryption mechanism.

*Access matrix for main MintTrack objects:*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Objects 🡪  Actors | Categories Table | Accounts Table | Transactions Table | YATC | Home  Activity | Entry  Activity | Audit  Activity | Tools  Activity |
| User | AddCategory  EditCategoryName  EditCategoryType | AddAccount  EditAccountName  DeactivateAccount  ReactivateAccount | AddTransfer  AddExpense  AddIncome  EditTransaction  DeleteTransaction | <<  instantiate  >> | <<instantiate>>  UpdateDisplay | <<instantiate>>  Choose Date  Enter Transaction | <<instantiate>>  Query  Transactions  ExportQuery | <<instantiate>>  Initialize YATC or Graph Analysis |

* 1. Global software control – MintTrack’s control flow will be event-driven due to its reliability on user input in most of its activities. For example, the user must select one of three transaction types in the entry tab and then enter information for that transaction before submitting it to the database. The system does nothing until the user commands it to perform some task. In the audit tab, a similar functionality is found, but the transaction list begins automatically generated and does not change until the user enters query information and submit the query to the application. The user has full control of the system and therefore MintTrack is procedure-driven.
  2. Boundary conditions –
     1. Start-up behavior: Upon start-up of the application, the home tab loads and displays the user’s income total, expense total, grand total, and four most recent transactions. Upon clicking on the other tabs, their respective features will be loaded. The database (persistent data) is created upon the very first execution of the program.
     2. Shut-down behavior: Upon shut-down of the application the Android OS will return to its main screen, or desktop. The persistent data will remain stored in the application’s SQLite database until the application is removed from the device.
     3. Error behavior:
        1. All undefined errors on the Android platform are unfortunately handled by a Force Close, in which the application unexpectedly shuts down.
        2. Errors caused by incomplete information in forms on the entry tab or in the YATC are to be handled by the application with an error message instructing the user to enter the required information before continuing.
        3. If the user enters information on a form and then navigates away from the form, information will stay on the form until the user returns or until the Android OS gets rid of the instance of the form during garbage collection. A clear button will be provided to the user on all forms to get rid of unwanted information.
        4. Error situations will be updated as they appear.

1. Subsystem services
   1. User Interface Subsystem (XML) – Provides the user with an easy way to navigate through and utilize the program to its full potential.
   2. Middleware Subsystem (Java) – Provides data entry error checking and database manipulation functionality.
      1. Features Subsystem – Contains the HomeActivity class and the ToolsActivity class:
         1. HomeActivity class– Provides the user with an overview of transactions and current balances including income and expense totals, a grand total of all income/expense transactions, and a short recent transactions list.

Methods:

* *displayIncomeTotal*
* *displayExpenseTotal*
* *displayGrandTotal*
* *displayRecentTransactions*
* *updateDisplay*
  + - 1. ToolsActivity class– Displays a list of extra features the user can execute. Contains the executable button for the tip calculator.

Methods:

* *onClick*
  + 1. Transaction Management Subsystem – Contains the EntryActivity and AuditActivity classes:
       1. Entry Activity Subsystem – Gives the user the ability to choose a transaction type and enter all information for a transaction of chosen type.

Methods:

* + - * *updateDisplay*
      * *fillCatDropDown*
      * *fillAccountDropDown*
      1. Audit Activity Subsystem – Allows the user to view recent transactions as well as query the database for a set of transactions matching criteria entered by the user.

Methods:

* + - * *showEvents*

\* All activities have an *onCreate* method.

* 1. Database Subsystem (SQLite) – Stores all transactions, accounts, and categories to be easily accessed by the middleware.
     1. Transaction Manager Subsystem – Contains a class to handle functionality of the transaction table and the table itself. The table holds a master list of all transactions and transaction information entered by the user.

Fields (for table):

* TOACCOUNT
* FROMACCOUNT
* AMOUNT
* DATE
* NOTE
* TYPE

Methods (for class):

* *getTransactions*
* *createIncome*
* *createExpense*
* *createTransfer*
  + 1. Account Manager Subsystem – Contains a class to handle functionality of the account table and the table itself. The table holds a list of accounts and their respective total values. Also describes whether the account is active or inactive.

Fields (for table):

* NAME
* TOTAL
* ACTIVE

Methods (for class):

* *getAccount*
* *addAccount*
* *deactivateAccount*
* *editAccountName*
* *editAccountTotal*
* *reactivateAccount*
  + 1. Category Manager Subsystem – Contains a class to handle functionality of the categories table and the table itself. The table holds a list of categories and their respective total values. Also describes the category type (income, expense, transfer).

Fields (for table):

* NAME
* TOTAL
* TYPE

Methods (for class):

* *getCategory*
* *addCategory*
* *editCategoryType*
* *editCategoryName*
* *updateCategory()*