MintTrack Test Plan

# Introduction

The MintTrack android application is a tool to keep tabs on personal finances. It allows daily financial events to be organized into individual transactions to later be recalled to reference. Transactions are organized into categories to help group like types of transactions in an ordered way. Accounts are also provided to represent different financial entities such as a bank account.

# System Overview

1. **Hardware**: The hardware includes any Android powered mobile device running the Android SDK 1.5 and up. Any mobile device should be considered an embedded device and thus software efficiency must be considered in all tests.
   1. Input Considerations:
      1. Touch Input: Tests will need to involve testing the interaction between the application and the touch input.
      2. Keypad Input: Tests will need to involve testing the interaction between the application and the keyboard input.
   2. Performance consideration:
      1. Due to the devices embedded nature performance testing a different device environment is vital. The Android device requires a 10 second response time thus if an application takes any longer it will be closed by the operating system.
      2. While examining the application it is vital to analyze the application’s memory footprint. Due to the embedded nature of the device, the amount of memory used by an application must be as minimal as possible.
2. **Database**: All financial information will be stored in a SQLite database. Due the SQLite's dynamic typing testing must take into consideration test cases that verify the accuracy of data being stored. The accuracy and stability of the data being stored in the local database must be as accurate as possible thus it is vital to verify the accuracy of all input interactions.
3. **Android Framework**: While the Android framework has matured enough to be trusted in most cases it should not be left out of test cases. Verifying important interactions with the android framework are vital to verifying low level interactions with the mobile device. Test cases must be also created to test that the framework has been properly used through the MintTrack application. Due to the fact the android platform is so young the likelihood of developers using the framework incorrectly is rather likely.
4. **MintTrack Layered Architecture**
   1. Graphical Interface Layer
      1. The MintTrack GUI provides a four tabs interface for interacting with the overall application.
         1. Home Tab - Provides summarized and total information about overall financial status. The most recent transactions are also displayed on the home tab to act as a helpful reminder of recent activity.
         2. Entry Tab - Allows the entry of financial transactions. This includes income, expense, and transfer transactions. The entry tab queries multiple tables to provide drop downs displaying available accounts and categories in the database system.
         3. Audit Tab - Introduces the ability to query transaction history based on a user provided data range. The audit tab also provides buttons for deleting and editing specific transactions. If edit is pressed information is passed to the entry tab to allow it to modify a specific transaction based on row ID.
         4. Tools Tab
            1. Manage Accounts: Used to add, edit, and delete accounts
            2. Manage Categories: Used to add, edit and delete categories
            3. Tip Calculator: Used to calculate the amount of a tip based on a specific dollar amount entered by a user.
   2. Middleware Interface Layer: the middleware interface is a consistent way of accessing the database. The middleware attempts to hide the complexity of the database from those using the interface classes. It also handles the overall management of the database without any visibility to its user.
   3. Database Layer: Low level classes that provide direct interaction with the different tables in the database. These layers should be hidden from the higher level developer to help reduce complexity.

# Features to be Tested

1. **Home Tab Features** - Provides a summary of overall transaction history
   1. Recent Transaction - By default the home tab displays the four more recent transactions to occur in the system
   2. Totals - The totals on the home screen help to summarize all transactions introduced into the application. The totals do not represent the totals located in the actual accounts but is instead an accumulation of transaction amounts.
2. **Entry Tab Features** - Used to enter transactions into the application
   1. Transaction Types - A transaction is any new financial event that is input via the user
      1. Expense - A transaction that reduces the amount of money in a specific account. The account that an expense reduces is considered the "From Account". An expense is any type of financial event that decreases a person’s money on hand, for example, buying something would be an expense.
      2. Income - A transaction that increases the amount of money in a specific account. It is used in the event of any financial event that the user receives income. For example, a user may enter their paycheck as income into the MintTrack application
      3. Transfer - A transaction that is used to take currency from one account and add it to another account. A transfer would occur in the event that a user moves money around within financial accounts. An example of a transfer is moving money from a checking account into a savings account.
   2. Edit Transaction - The entry tab is also used to modify existing transactions. When a user presses edit under the audit tab on a specific transaction its row ID is passed to the entry tab to allow it to be populated into the entry tab. The entry tab then modifies its default display to make it to appear as though the user is using a new feature. The interface provides the ability to modify all data elements, save any modifications, or cancel the modification.
   3. Dynamic Dropdowns: Drop downs are provided for specific data elements required for a transaction. The drop downs include categories and accounts. The drop downs are dynamically populated from the database tables.
3. **Audit Tab Features** - Used to review historical transaction data
   1. Querying: The audit tab provides the ability to query the entire transaction history based on a date range. A user has complete control of the granularity of what will be displayed by searching based on any range of time.
   2. Support Transactions - The audit tab supports the display of all transactions created via the entry tab. All information entered for a specific transaction type is displayed row by row per transaction.
   3. Transaction Interaction- After querying the database, if a transaction is pressed it displays actions that may be performed on that specific transaction.
      1. Edit Action: Pressing the edit action redirects the user to a special version of the entry tab that allows them to make modifications to the selected transaction. From the entry tab they can either save or cancel the interaction.
      2. Delete Action: The delete action will remove the existence of the transaction from the database. It will also update the account and category tables accordingly based on what is being deleted.
   4. Scroll - When too many transactions are to be displayed on the screen the audit tab provides the ability to scroll vertically.
4. **Tool Tab Features** - Used to interact with secondary features of the overall MintTrack application
   1. Account Manager: Provides all the functionality to manage accounts including add, remove, and editing actions.
      1. Activate/Deactivate Actions: A user is able to activate and deactivate accounts once they are added. If a user is no longer using a specific account it must still be included in the database for past transactional referencing. Deactivating an account will hide it from the account drop downs under the entry tab.
      2. Add Action: Allows the creation of a new account. This includes a name, and current balance.
      3. Edit Action: Used to correct the naming or balance of an existing account.
   2. Category Manager: Provides all the functionality to manage categories including add, remove, and editing actions.
      1. Activate/Deactivate Actions: A user is able to activate and deactivate categories once they are added. If a user is no longer using a specific category it must still be included in the database for past transactional referencing. Deactivating a category will hide it from the category drop downs under the entry tab.
      2. Add Action: Allows the creation of a new category. This includes a name, and current balance.
      3. Edit Action: Used to correct the naming or balance of an existing category.
   3. Tip Calculator: Allows the calculation of tips based on a purchase amount and specific percentage provided by the user.

# Approach

1. Unit Testing
   1. Each method must be tested based on its specific purpose. Each unit test should verify the output of a method based on a range of valid and invalid input. Prior to the implementation of a method and unit test expected outputs for specific input events must be defined.
2. Device Testing
   1. While development may primarily run preliminary testing within the Android Emulator all testing done by quality assurance must commence on the actual device. This is vital to identify conflicts between how MintTrack acts and displays on the actual hardware versus an emulator.
3. Input Considerations:
   1. Valid considerations - Based on the valid ranges identified during the creation of unit tests a wide range of inputs should be tested within that range.
   2. Invalid considerations - As a quality assurance agent, possible forms of bad input must be thorough. Input points include the entry tab, category manager, and account manager. Each context must be analyzed and possible inputs must be generated that could possibly lead to bad data in the database or a crash in the device.
4. Performance Testing:
   1. Due to the embedded nature of all Android mobile devices it is important to check how MintTrack performs in many different situations. Failure points that must be considered include:
      1. How does the application run with many transactions?
      2. Can MintTrack perform when other applications are also in memory?
      3. Are response times throughout the device fast enough to please the user base?
5. Redundancy Checks:
   1. Throughout the application it is vital to verify that redundancy cannot occur. This involves testing the creation of duplicate accounts, categories, and empty transactions. From a data accuracy standpoint it is vital that all possible redundancies be prevented.

# Testing Materials

1. Hardware
   1. Android Mobile Device - Cell phones powered by the Android SDK 1.5. The device is necessary to provide an actual working environment that is exactly as the consumer’s.
   2. PC - A personal computer may be used to execute tests within the Android emulator or to execute JUnit tests on the code base.
2. Software
   1. Android Emulator version 1.5 - 2.1: An emulator provided by Google provides a strong environment for testing before going to a mobile device.
   2. JUnit: The Java unit testing framework is used to create usability tests.

# Test Cases

*Please note that the test cases provided below are not the only test cases necessary for effectively checking the stability of MintTrack. It is instead of subset of useful test cases to represent an understanding of what a test case exactly is.*

**Name**: Create expense transaction  
**Input**: This test case should be run using valid and invalid input for an expense transaction. The use of bad input will help to identify any inconsistencies in the form’s validation process. Input includes:

* Amount: Decimal numerical value provided by the user. Is the storage of this data element accurate?
* Category: This is constrained by the drop down and database.
* From Account: This is constrained by the drop down and database.
* Note: Notes are a string representations of a textual reminder. A user would possibly do an SQL injection through this field.

**Output**: The expected output is a transaction representing the valid input provided by the user. This includes:

* Proper transaction amount has been stored in the database
* No truncation has occurred with the transaction amount
* Category and Account ID has been stored correctly in the transaction based on that actual user’s input
* Note has been correctly stored in the database

**Actions:**

1. Execute MintTrack from the menu of the device
2. Press the entry tab
3. Select the transaction type, Expense.
4. Enter the amount, account, category, and note for the transaction
5. Press save

**Special Conditions:**

In the event of bad input, i.e. user enters out of bounds amount, the entry tab should prompt the user with a friendly error. If any invalid input gets through to the database this should be considered a high priority bug as it can lead to inconsistency in the database.

**Name:** Execute date range query

**Input:** Input should exam all possibilities of transactional output. This includes:

* No transactions existing for date range
* Large amounts of transactions existing for a date range
* A single transaction existing for a date range
* Date ranges that go into the future
* Start date later then the end date

Many different queries must be tested on different sets of transactional data. This data should include all transaction types including expense, income, and transfer transactions.

**Output:**

* **Valid Transactions:** Output must represent the available transactions in the particular date range stored on the database. Valid information should be displayed accurately and correctly in the ListActivity. Each transaction should be interactive and allow the user to click edit or delete actions to change a transaction.
* **Invalid Transactions:** Even when a valid date range is provided it is possible that bad data could make it to the database. It is important to test with premade transactions to see how the audit tab will respond to bad data.

**Actions:**

1. Execute MintTrack from the menu on the device
2. Press the Audit tab
3. Choose the starting date
4. Choose the to end date
5. Press the query button
6. Review transactions output to screen

**Special Conditions:**

In the event of bad input the audit tab should attempt to display the information stored in the database to the screen. In the event that a transaction has in fact been corrupted it should be labeled as so and the user should be allowed to modify or delete it.

**Name**: Create transfer transaction  
**Input**: This test case should be tested using valid and invalid input for a transfer transaction. The use of bad input will help to identify any inconsistencies in the form’s validation process. Input includes:

* Amount: Decimal numerical value provided by the user. Is the storage of this data element accurate?
* To Account: This is constrained by the drop down and database.
* From Account: This is constrained by the drop down and database.
* Note: Notes are a string representations of a textual reminder. A user would possibly do an SQL injection through this field.

**Output**: The expect output is a transaction representing the valid input provided by the user. This includes:

* Proper transaction amount has been stored in the database
* No truncation has occurred with the transaction amount
* Account identification numbers have been stored correctly in the transaction based on that actual user’s input
* Note has been correctly stored in the database

**Actions:**

1. Execute MintTrack from the menu of your device
2. Press the Entry tab
3. Select the transaction type, Transfer.
4. Enter the amount, choose a source and destination account with dropdowns, and enter a note for the transaction
5. Press save

**Special Conditions:**

In the event of bad input, i.e. enter out of bounds amount, the entry tab should prompt the user with a friendly error. If any invalid input gets through to the database this should be considered a high priority bug as it can lead to inconsistency in the database.