# Midterm Design Review (MDR)

## Cause and Effect


---

**By:**

**Keith Adler**

**Matthew Brannick**

**Tomin Kozhimala**

**William Vennes**

# Table of Contents

# Vision Statement

**Cause and Effect**: For people wanting to make their smart devices truly intelligent. Today, smart phones are overly complex and jammed pack with functionality. Despite having the power to do so many spectacular things, smart devices lack the ability for users to take control and put the phone to work for them. Cause and Effect is a software solution that let users easily program their smart devices to respond to a set of rules with a chosen effect. With Cause and Effect, smart devices will become more intelligent and make users' lives easier by intuitively automating tasks, gathering information, displaying information, etc. Unlike other rule based software, Cause and Effect will provide more functionality and a better user interface to allow any type of user to take control of their phone.

# Executive Summary

Today, smart phones are a part of our everyday lives. Users of smart phones are able to do a variety of functions, whether it is talking to friends on Facebook, listening to music, or sharing and watching videos on YouTube. However, smart phones currently do not have the ability to automatically perform user tasks based on changes that occur in the phone's environment.

Users often pay large amounts of money for their smart phones but never utilize their phone's potential. The typical smart phone user uses only functions such as the phone, the text messaging, email, and some fun games. There are so many features and APIs available for smart phone development but rarely are these integrated to enrich the life of the user by using the entire phone's brilliance.

The solution to this is Cause and Effect for Android. Cause and Effect allows users to take more control over their phones. Users of Cause and Effect will be able to set "rules" for their phone, which enables the phone to automatically respond to changes in the environment, such as a change in Wi-Fi connection or current phone location via GPS. Users will also be able to share their phone rules with other users of Cause and Effect, whether it is via email, text, or NFC between other Android devices. Cause and Effect will enable users to automate desired functionality and will increase the usability of their smart phone.

Partnered with our client, Samsung Mobile, we have been given the tools and direction to develop Cause and Effect for Android. We plan on revolutionizing the way how users see their phones and the smart phone development industry. Through much planning and hard work, we have a working prototype here today.

The application is simple and intuitive. It is so easy that your grandma could use it. The user can create rules that can be toggled on and off. These rules consist of Causes and Effects, which is where the name comes from. The Causes are actions from the phone's operating system or from an external source such as Facebook, Gmail, or Twitter. The operating system broadcasts can come from simple calls or messages, or many of the phones sensors and services. Users can choose causes with their own parameters to trigger effects. These effects are actions that the phone can execute. These effect actions allow users to really gain a lot of benefit from this application. With various types of causes and effects, users can create complex rule structures to fit their needs. Further variety can be attained using the logical AND and OR structure integrated into the causes chosen by the user. These causes are arranged in a tree structure so that users can apply user friendly Boolean logic to their own rules. Since you only have to setup a rule once, this functionality allows users to really take advantage of their phone from then on. Users can also share their rules and lists of rules with other Cause and Effect users so a community could form around rule creation and rule types that solve everyday problems.

# User Needs

We met with Samsung and outlined our requirements for the project. These requirements are organized into user stories, so that we can develop a project in a structured manner that abides by our agile development methodology. Some user stories can be considered "epic." Epic user stories are high level requirements and are intended to be broken into sub user stories and tasks when developed. For organizational purposes, the stories are organized into the initial user stories that we derived from our first meeting with Samsung, and the user stories that we assigned and completed for each iteration.

## Initial User Stories

These User Stories were the created before the first iteration based on our first meeting with Samsung. Although we may have added to or changed the user stories in other iterations, these were our initial guidelines for planning our designs.

### Client User Stories

- As a client, I want a rules engine, so that users can automate phone actions.
- As a client, I want the rule creation process to be as easy as possible.
- As a client, I want a fluid user interface, so that all users can use our application with ease.
- As a client, I want a simple way to add new causes and effects, so that third parties can add to the repository Extensible to Third Parties
- As a client, I want rules to work with Boolean algebra, so that users can create complex rules to tailor to their needs.
- As a client, I want rules to be sharable, so that users can easily distribute their rule frameworks to others
- As a client, I want the application to require a minimum of Android version 4.0 (Ice Cream Sandwich), so that we can utilize the next generation of Android devices.
- As a client, I want the application to run on all Android v4.0+ devices, so that we can have as much user compatibility as possible
- As a client, I want to include NFC based rules and sharing, so that the app can make use of Samsung's latest hardware sharing technology.
- As a client, I want rules that react to the user's location, so that users can create intricate rules using any given location.
- As a client, I want to include Wi-Fi based rules and sharing, so that the app can make use of local network capabilities.

- As a client, I want to be able to put many rules in (stress test) and have it work as quickly as with very few rules.
- As a client, I want the application to be optimized for battery efficient, so that the application will not drain the phone's battery life.
- As a client, I want an additional drag and drop rule creation interface, so that the user can choose to have a fluid look and feel while creating rules.
- As a client, I want the application to run quickly and not to slow down my phone's other processes.
- As a client, I want the application to not take up much space on my phone.
- As a client, I want to know the application is working correctly when closed.

## Developer User Stories

- As a developer, I want to be able to also share using email, MMS, Facebook, and Dropbox, so that users can have options on the sharing medium.
- As a developer, I want to create user notifications, so that users are aware of the application's activity and progress.
- As a developer, I want to include a user preferences activity, so that users can tweak and enable settings for the application.
- As a developer, I want to have text message based rules, so that I can debug rules using the phone's mobile network.
- As a developer, I want to have time-based rules, so that I can debug the application easily.
- As a developer, I want to have vibrate-based rules, so that I can debug the application easily.
- As a developer, I want to have "Toast" based rules, so that I can debug the application easily.
- As a developer, I want to have Facebook-based rules, so that I can test the account sync rules.
- As a developer, I want to have Google-based rules, so that I can test the account sync rules.
- As a developer, I want to have Dropbox-based rules, so that I can test the account sync rules.
- As a developer, I want an easily accessible database (such as SQLite).
- As a developer, I want to be able to decipher and encode rules with Boolean algebra.
- As a developer, I want to have screen animations, so that the application is fluid and aesthetically pleasing.

- As a developer, I want a cool logo, so that our application is memorable and aesthetically pleasing.
- As a developer, I want a cool application icon, so that our application is easy to recognize on the Android home screen.

# Iteration User Stories

Each iteration consisted of a set of user stories that were accomplished during that time. Some iterations had more user stories than others. This depended on the difficulty of certain user stories, and the amount of time that our group could put towards each iteration. Some user stories were entirely based on code, where others may be based on design or testing.

## Iteration 1

- As a developer, I want a Login screen design, so that we can implement a mock graphical user interface.
- As a developer, I want a Main Menu screen design, so that we can implement a mock graphical user interface.
- As a developer, I want a Rule Creation screen design, so that we can implement a mock graphical user interface.
- As a developer, I want a My Rules screen design, so that we can implement a mock graphical user interface.
- As a developer, I want a Rule Description screen design, so that we can implement a mock graphical user interface.
- As a developer, I want a Cause screen design, so that we can implement a mock graphical user interface.
- As a developer, I want a Effect screen design, so that we can implement a mock graphical user interface.
- As a developer, I want a Account Sync screen design, so that we can implement a mock graphical user interface.
- As a developer, I want a Cause and Effect Option screen design, so that we can implement a mock graphical user interface.
- As a developer, I want a Preferences screen design, so that we can implement a mock graphical user interface.
- As a developer, I want a General Preferences sub screen design, so that we can implement a mock graphical user interface.

- As a developer, I want a Account Sync Preferences sub screen design, so that we can implement a mock graphical user interface.
- As a developer, I want a Security Preferences sub screen design, so that we can implement a mock graphical user interface.
- As a developer, I want a About Page sub screen design, so that we can implement a mock graphical user interface.
- As a developer, I want a flow diagram, so that the activity screens can organized and have basic flow.
- As a developer, I want all of these designs to be implemented into a Mock GUI Application, so that we have a release our first iteration and can get feedback from Samsung.

## Iteration 2

- As a developer, I want an alphabetical list view for rules, causes, and effects, so that users can view rules in whichever method they prefer.
- As a developer, I want a category list view for rules, causes, and effects, so that users can view rules in whichever method they prefer.
- As a developer, I want a grid list view for rules, causes, and effects, so that users can view rules in whichever method they prefer.
- As a developer, I want buttons that allow the user to switch between different view types on any given list.
- As a developer, I want an action to occur when any item in a list is selected, so that the flow of the application can be improved and items can be selected.
- As a developer, I want a grid view for rules, causes, and effects, so that users can view rules in whichever method they prefer.
- As a developer, I want the help items menu to contain fields, so that we can simulate the future functionality for this activity.
- As a developer, I want menus on the Main Menu screen, so that users can navigate the application with ease.
- As a developer, I want menus on the New Rules screen, so that users can navigate the application with ease.
- As a developer, I want menus on the Help screen, so that users can navigate the application with ease.
- As a developer, I want menus on the Settings screen, so that users can navigate the application with ease.
- As a developer, I want divider lines for each list, so that the user can clearly see the distinction between list items.

- As a developer, I want to hide the password characters on the Account Sync screen, so that users can hide their personal login information.
- As a developer, I want a confirmation toast on the Account Sync screen, so that users know when the account was synched successfully.

## Iteration 3

- As a client, I want a Rules Engine design document, so that we can implement the Rules Engine backend in the next iteration.
- As a developer, I want a design for a Broadcast Receiver class, so that we can receive system calls from Android OS.
- As a developer, I want a design for an Event Handler class, so that other third party causes can be invoked.
- As a developer, I want a design for a Rules Database class, so that user data can be stored and accessed through simple and complex queries.
- As a developer, I want a design for a Rules Engine class, so that the events triggered by the Broadcast Receiver and the Event Handler can be evaluated as true or false.
- As a developer, I want a design for an Action Executer class, so that this can be called from the Rules Engine when the cause triggers are evaluated as true.
- As a developer, I want these rules features to be designed to run on separate threads, so that the main user thread is not slowed down.
- As a developer, I want the Rules Database class to consist of either SQLite, a B+ Tree, a Hash Table, or an XML/JSON structure, so that there can be structure to the rule data.
- As a developer, I want the Rules Engine class to contain a recursive logic tree function, so that rules can be evaluated from expressions including ANDs and ORs.
- As a developer, I want the Rules Engine class to contain a check cause function, so that the causes can be evaluated when the tree function reaches an individual cause.

## Iteration 4

- As a developer, I want an implementation of the Broadcast Receiver, so that we can start the rules engine flow.
- As a developer, I want an implementation of the Rules Database, so that we can manage rule data.
- As a developer, I want an implementation of the Rules Engine, so that we can process causes using the database.

- As a developer, I want an implementation of the Action Executer, so that we can display results of the Rules Engine.
- As a developer, I want styling on the buttons, so that the user will have a more aesthetically pleasing experience.
- As a developer, I want styling on the activity screens, so that the user will have a nice contrast that is clear and easy to look at it.
- As a developer, I want to use our application logo on the Main Menu, so that the user can remember our application.
- As a developer, I want to use our application icon on the application, so that the user can identify our application from the application menu on the Android OS.
- As a developer, I want to redesign the Edit Rule screen, so that it can allow for new causes and effects to be added to a rule.
- As a developer, I want three cause types, so that we can test the rules engine evaluation.
- As a developer, I want three effect types, so that we can test the rules engine results.

## Iteration 5

- As a developer, I want to connect the Broadcast Receiver and the Rules Engine classes, so that the causes are sent to be evaluated once they are triggered.
- As a developer, I want to connect the Rules Engine and the Rules Database classes, so that the Rules Engine can evaluate cause data stored in the database.
- As a developer, I want to connect the Rules Engine and the Action Executer classes, so that a cause tree evaluated to true will produce results that the user can see.
- As a developer, I want a test database function, so that the user will have a few sample rules available when they first install the application.
- As a developer, I want to reskin the Edit Rule screen, so that the user can see clear definition between different sections with clarity.
- As a developer, I want an Update Rule button, so that the user can save a new rule in the database.
- As a developer, I want the Update Rule button to handle rules that are already created, so that they can be updated in the database instead of being recreated.
- As a developer, I want a phone call Cause dialog, so that the user can choose a contact to be used in their rule.
- As a developer, I want a test message Cause dialog, so that the user can choose a contact to be used in their rule.
- As a developer, I want a time Cause dialog, so that the user can choose a time that they want the rule to be triggered.

- As a developer, I want a notification Effect dialog, so that the user can create a custom notification to be displayed in the notification bar.
- As a developer, I want a toast Effect dialog, so that the user can create a custom message to be displayed briefly.
- As a developer, I want a vibrate Effect dialog, so that the user can create custom vibrate tones and store this tone as an effect.
- As a developer, I want to update the dividers between items in the lists, so that they are more aesthetically pleasing.
- As a developer, I want to create a Preliminary Design Document, so that we can present documentation at the presentation at the end of the iteration.
- As a developer, I want to create a Preliminary Design Slideshow, so that we can have a basis for the presentation at the end of the week.

## Iteration 6

- As a developer, I want to be able to search causes by type, so that I can evaluate more than one rule at once if they trigger at the same time.
- As a developer, I want to redesign the cause tree, so that it will evaluate multiple causes properly.
- As a developer, I want to allow rules to be turned on and off, so that the user can decide which rules they want to be evaluated at a given time.
- As a developer, I want to provide a slide presentation for Samsung, so that we can update our clients on our progress since before winter break.
- As a developer, I want to add a flow mini-map to our presentation slides, so that the client and professors can follow along with our backend code design.
- As a developer, I want to update the contrast in our slides, so that the user can see our slides in different lighting conditions.
- As a developer, I want to design an AND and OR interface, so that the user can add their own ANDs and ORs to the causes.
- As a developer, I want to turn off screen rotation, so that the user doesn't get confused due to phone orientation.
- As a developer, I want to add "Not Implemented" toast messages, so that the user can see what features are completed, and which are not yet implemented.
- As a developer, I want to remove the grid view, so that the user can have a clear and simple interface for choosing between text items.
- As a developer, I want to be able to save a rule after any fields are updated, so that the user doesn't have to manually update the rule and the information is retained properly.

- As a developer, I want to fix the back stack, so that there aren't extraneous activities being created throughout the rule creation process.
- As a developer, I want a vertical separation of causes and effects on the Edit Rule screen, so that the user won't cross the divider lines for each list.
- As a developer, I want to limit ANDs and ORs to expressions with at most one level of nesting, so that users won't get confused by the complexity of the expressions for rules.
- As a developer, I want to create a cause tree structure that generates from a string of leaves, so that the algorithm is more efficient.
- As a developer, I want to start designing a system to implement the rules engine under a separate service so that the rules are evaluated in all of the edge cases.
- As a developer, I want to check causes and effects for duplicates, so that the interface is clear of clutter.

## Iteration 7

- As a developer, I want a function that will display a user friendly version of the cause tree, so that the user can see what the rule is supposed to do.
- As a developer, I want to be able to edit causes, so that the user can make quick changes or tweak parameter values.
- As a developer, I want to be able to edit effects, so that the user can tweak parameter values for effect messages.
- As a developer, I want to be able to delete causes, so that the user can remove unwanted causes from the cause list.
- As a developer, I want to be able to delete effects, so that the user can remove unwanted effects from the effect list.
- As a developer, I want to run the rules engine from a service, so that the rules evaluate in edge cases and the evaluation is more efficient and independent from the application.
- As a developer, I want a sound Effect type, so that the user can play sounds as an effect.
- As a developer, I want a ring mode Effect type, so that the user can change their ringer settings as an effect.
- As a developer, I want a location Effect type, so that the user can creates rules that are dependent on the phone's location.
- As a developer, I want the location type to account for multiple kinds of services, so the user can decide how accurate they want the location updates to be and save battery as need be.
- As a developer, I want to change the vibrate Effect so that it is a simple tone, so that the user doesn't get confused with the vibrate tone integer string during effect creation.

- As a developer, I want a Wi-Fi Effect type, so that the user can create rules based on current Wi-Fi settings.
- As a developer, I want to be able to share rules with another phone, so that users can share their created content as they see fit.
- As a developer, I want to be able to share rules using NFC, so that users can use this useful and cutting edge feature on Samsung devices.
- As a developer, I want to remove the account tables, so that we can provide a much more secure OAuth service to the user.

## Iteration 8

- As a developer, I want to add ANDs and ORs to the cause list, so that the user can see the expressions as they are being created.
- As a developer, I want to migrate all unresolved bugs to our Bitbucket tracker, so that our clients can view our progress on fixing errors and we can document them in a formal fashion.
- As a developer, I want to fix the stack on the Edit Rule screen, so that new causes, edited causes, and deleted causes don't create new activities that are not needed.
- As a developer, I want to redesign the background service, so that it will properly trigger all the causes from the Android OS and interact with application when necessary.
- As a developer, I want to implement actions on the action bar, so that the user can access different functionality at the touch of a button.
- As a developer, I want to implement contextual action bars, so that the user can have different action bar items available depending on the current item selection and screen.
- As a developer, I want to be able to handle duplicate rules, so that sharing will not create redundant instances of rules that are identical in every way.
- As a developer, I want a listener for Wi-Fi, so that Wi-Fi causes will be triggered whenever a change in Wi-Fi settings has occurred.
- As a developer, I want a listener for location, so that the location causes will be triggered whenever the phone receives an update that is better than the old location.
- As a developer, I want to update our slides for the Midterm Design Review, so that we can have an updated presentation representing our work since the PDR.
- As a developer, I want to update our Midterm Design Review document, so that we can update our documentation for our work since the PDR.
- As a developer, I want to spend additional time debugging, so that we have a clean iteration release, and we have a robust interface for user testing.
- As a developer, I want to format our code into one standard form, so that our code is legible, clear, and uniform.

- As a developer, I want to create Javadoc comments, so that our code is documented better internally.
- As a developer, I want to generate a Javadoc file, so that external developers (such as our clients after we hand it off) can understand all of the classes, functions, and members through code documentation.

## Phone/Physical Rules

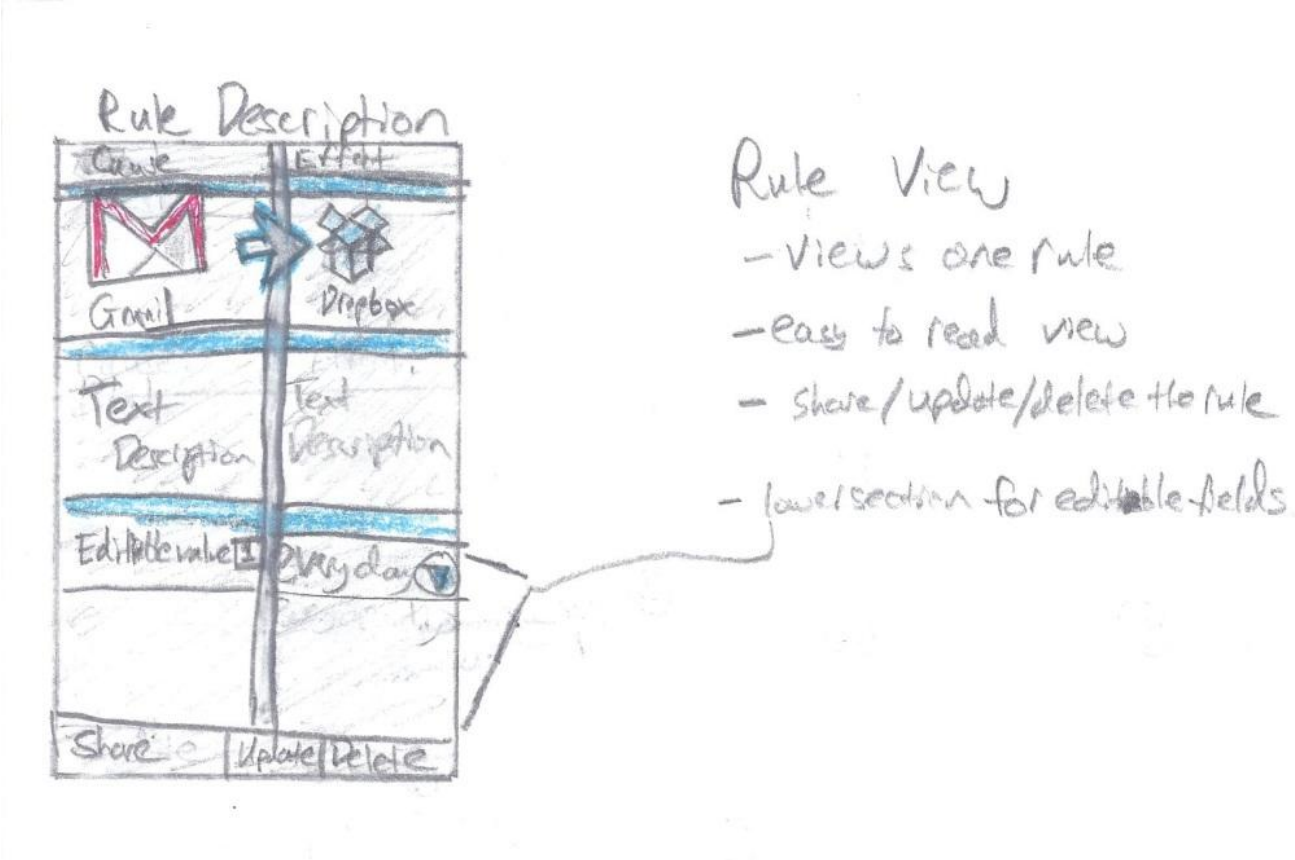
There are many phone features that will be included in the final product. Some of the rule types are considered more important, and there are listed above and are not included in this list. For the sake of space and organization, the list of phone-based rules is listed as the Backlog Appendix at the end of this document.

## API Based Rules

There are also numerous account types that will be integrated in the final product. The plethora of accounts and rule types even prompted the creation of a "Category View" that will be the default view for browsing rule options in a future release. For the sake of space and organization, the list of account types is listed as the Backlog Appendix at the end of this document.

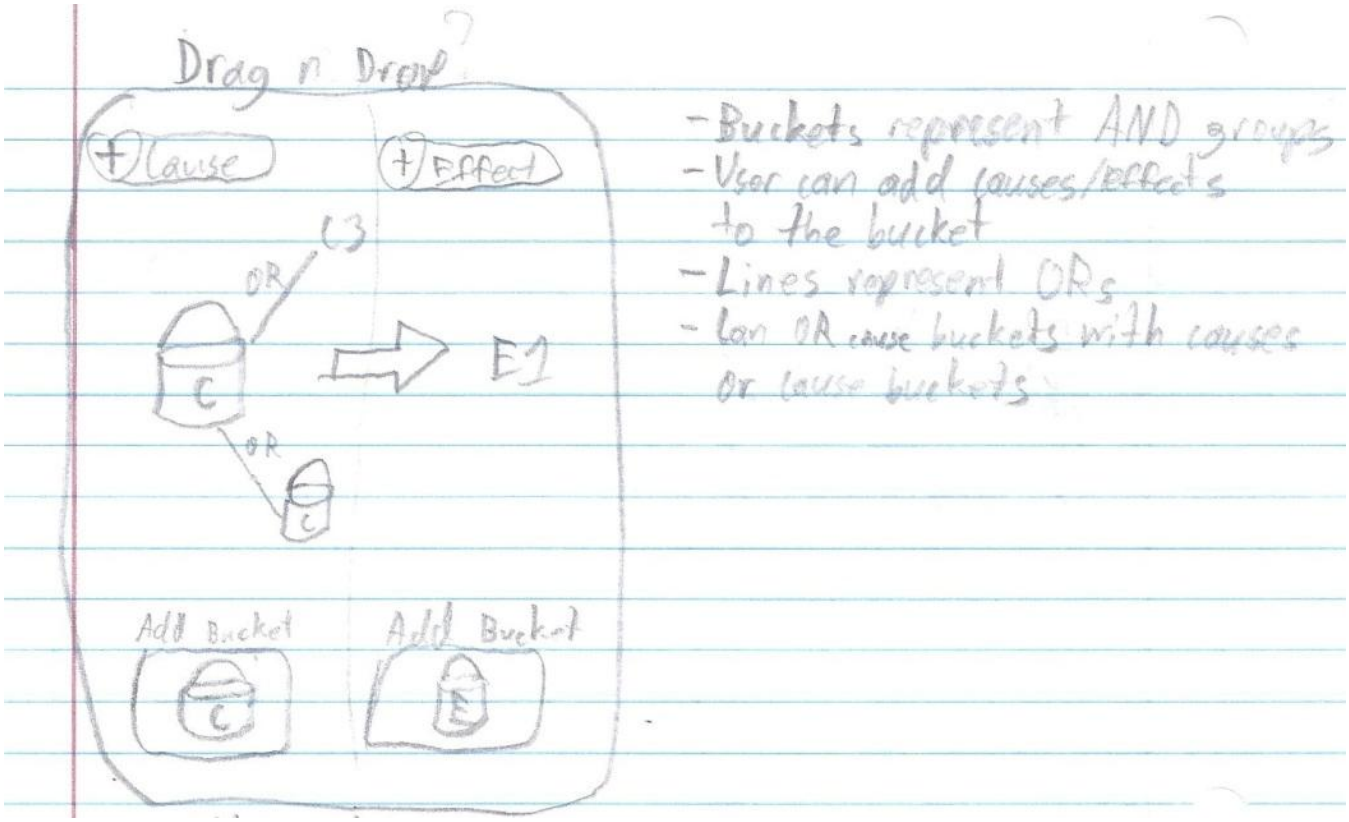
# Initial Design Ideas

## Rule View Activity



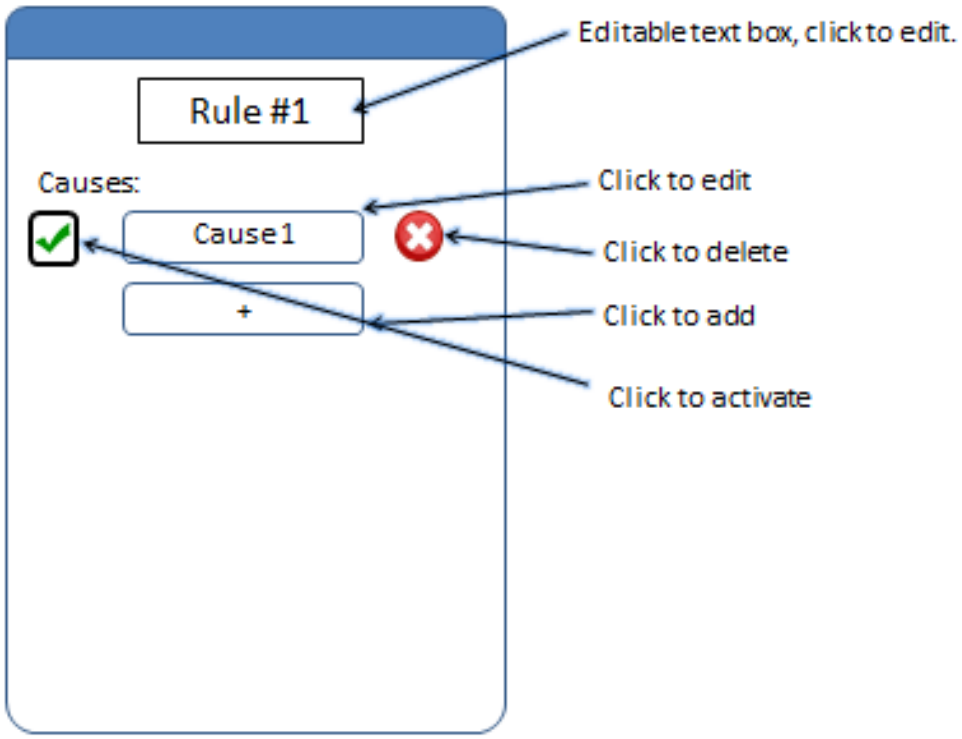

Here, the user will follow the rule description page as they create the rule. They click on the cause size to choose a cause, and the effect page to choose an effect. As the user adds causes and effects, they will return to this page to see how their rule has progressed as they create it.

# Drag and Drop View



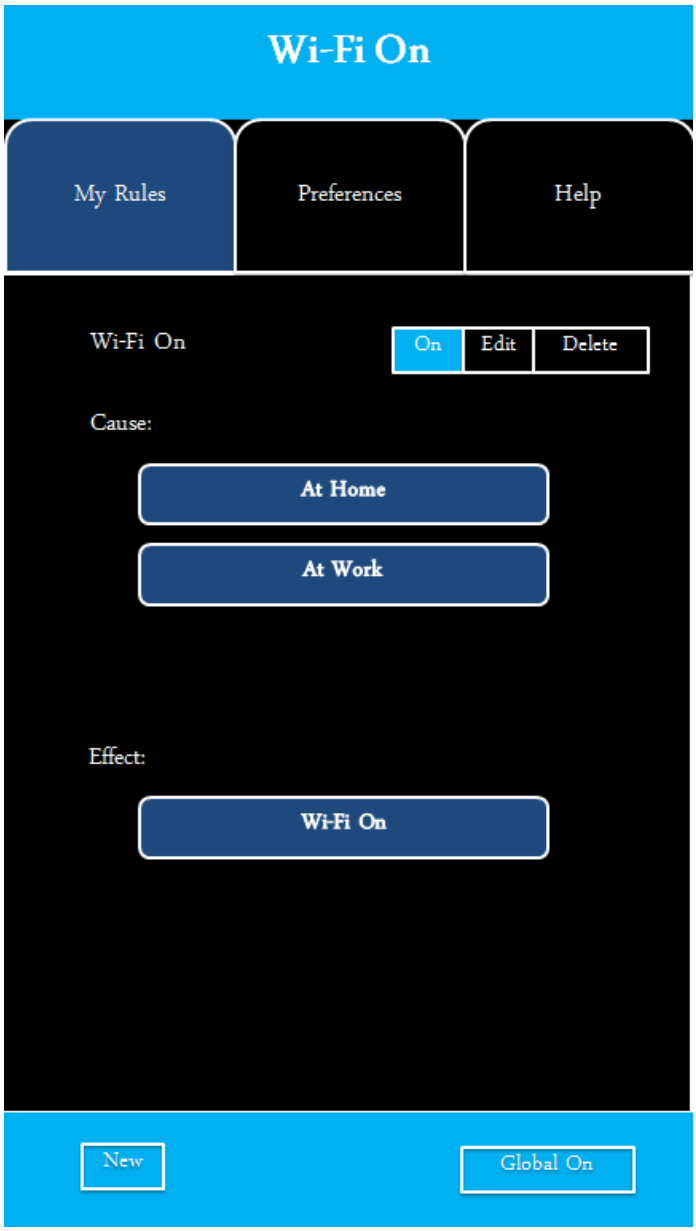

Here is an idea for a drag and drop implementation of the rule creation page, in which users place selections in "buckets" to represent groups of causes or effects, and link them to other buckets or individual items by an OR line. This would create a more abstract view for rule creation and could benefit visual-based users.

# Rule Creation Activity



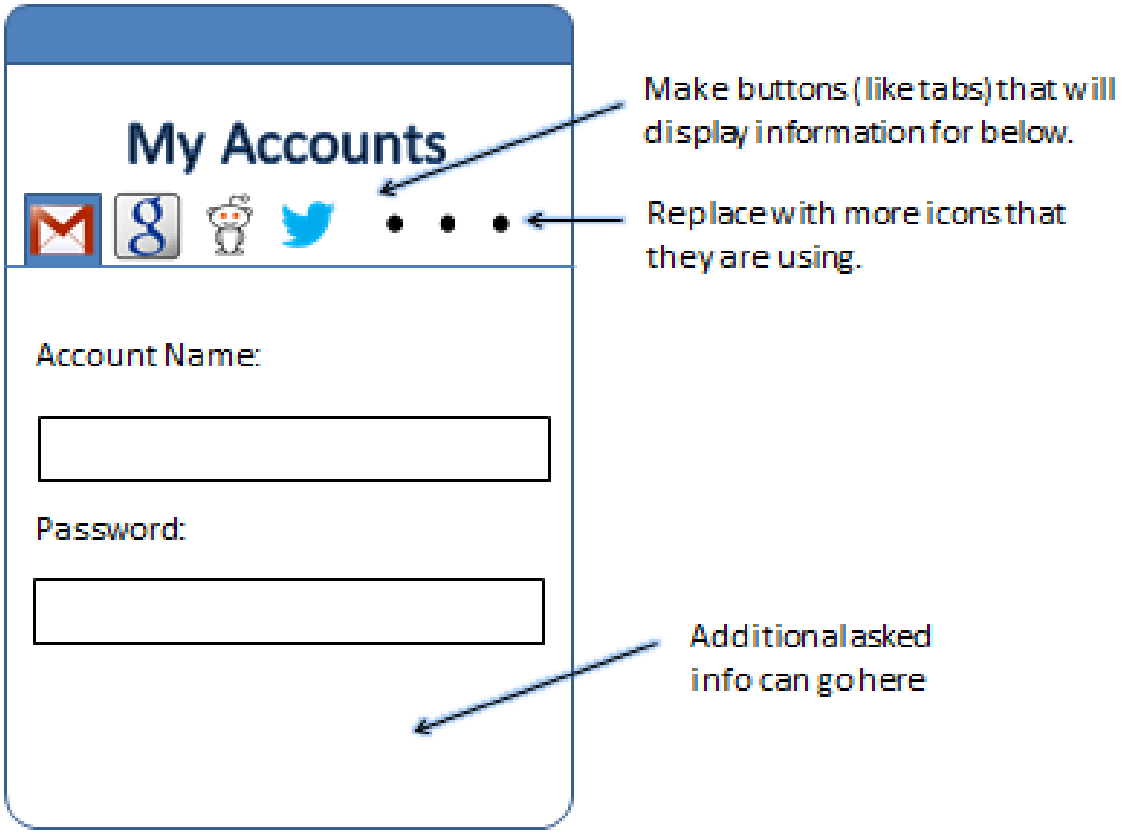

Here is a rule creation page in which causes and effects are separated by different pages and takes a more list view approach.

# My Rules Activity (Version 2)



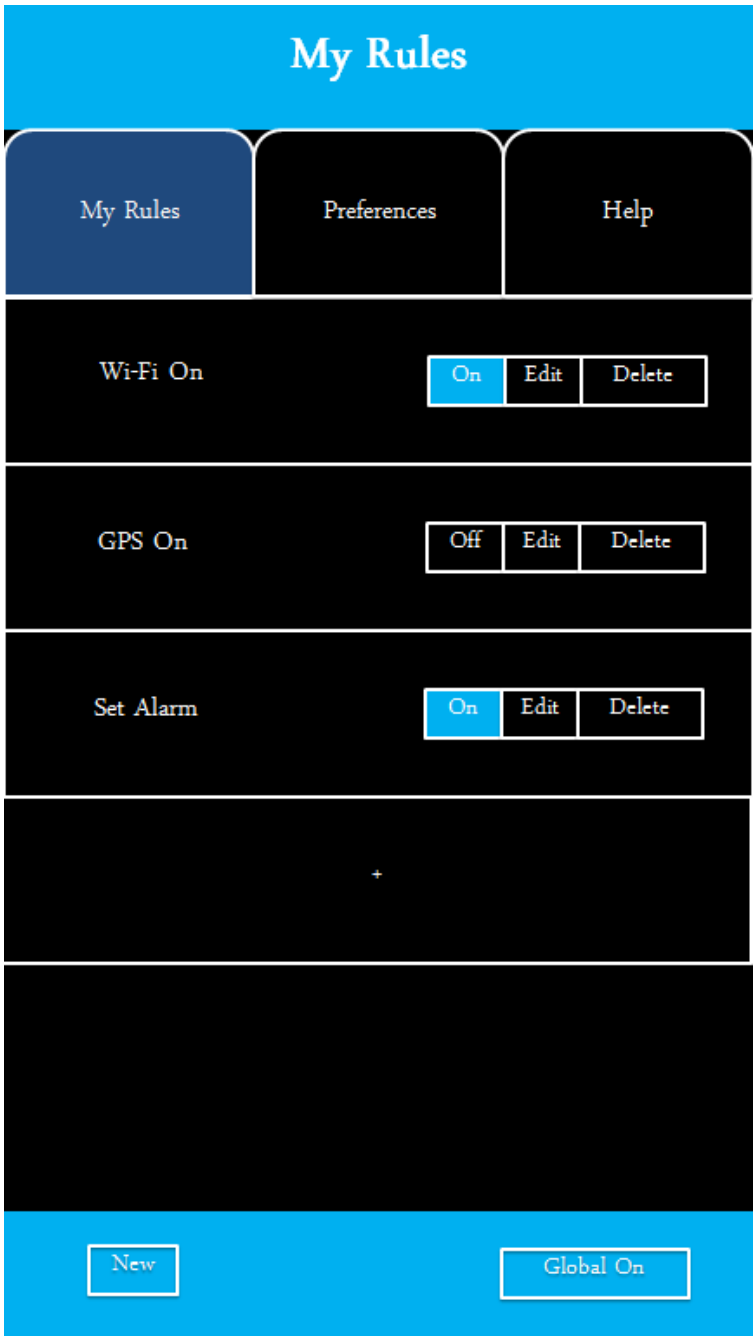

Another conceptual rule view where causes and effects are listed sequentially in a list format.

# Account Sync Activity



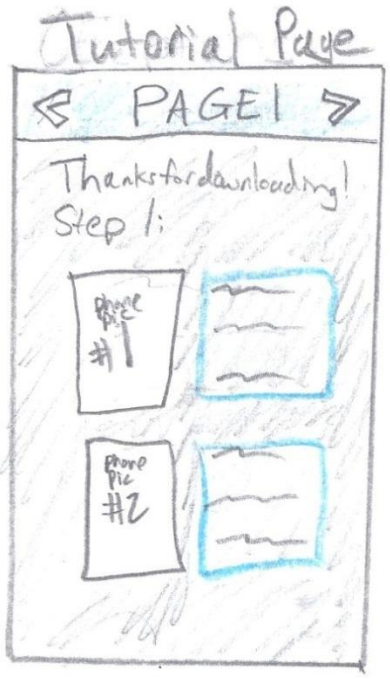

This is an accounts page in which the user syncs their Google, Facebook, etc. accounts for use with rule creation. Another idea was moving the list of accounts to a separate page accessed from preferences, rather than a slide at the top.

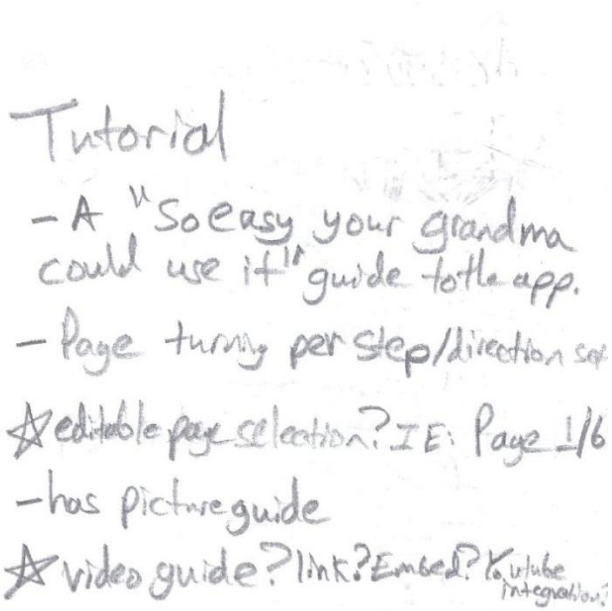
# My Rules Activity




This was a concept screen for the viewing of existing rules. From this screen, users can turn a rule on or off, go to the edit menu, or delete the rule.

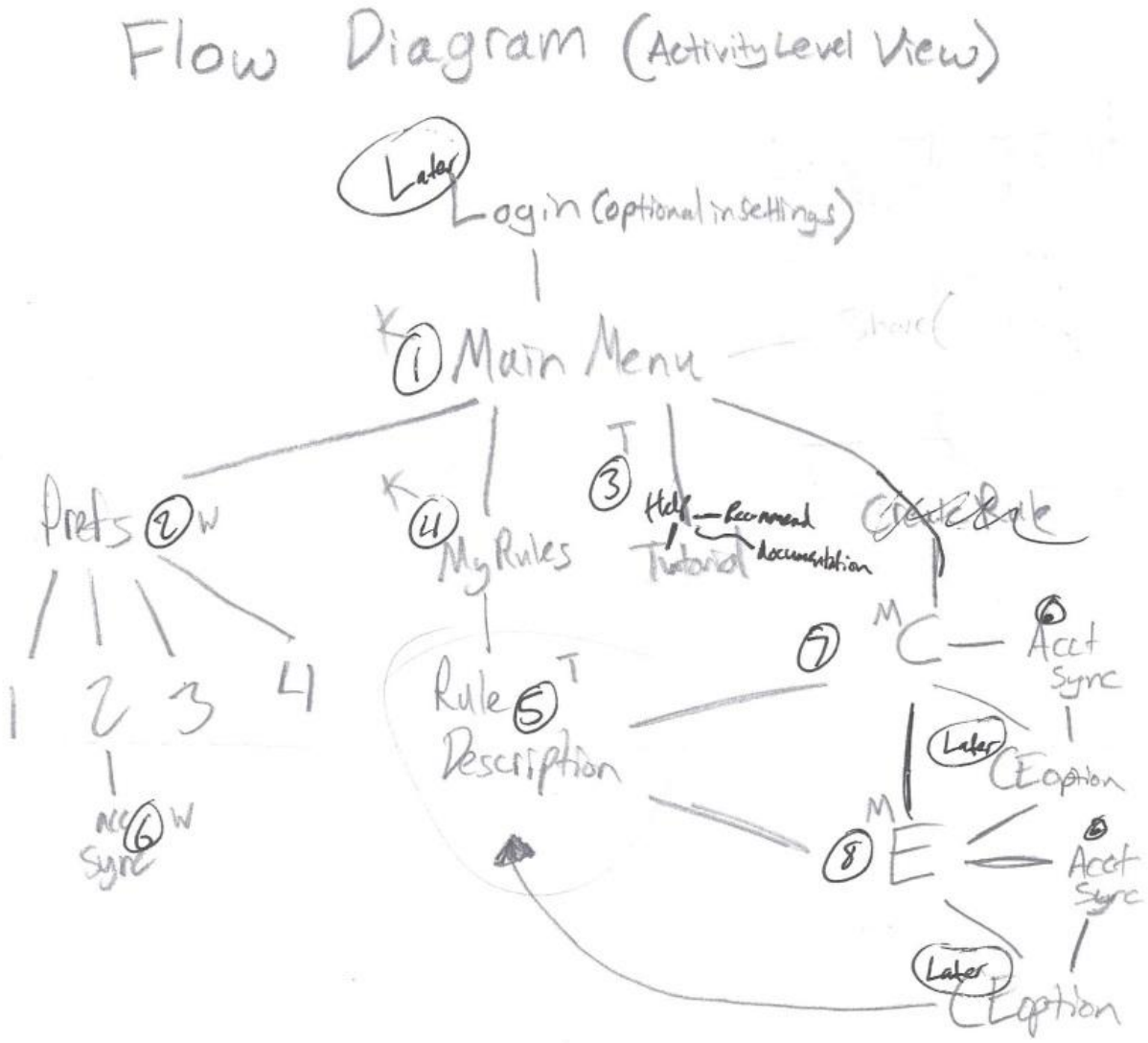
# Help Activity



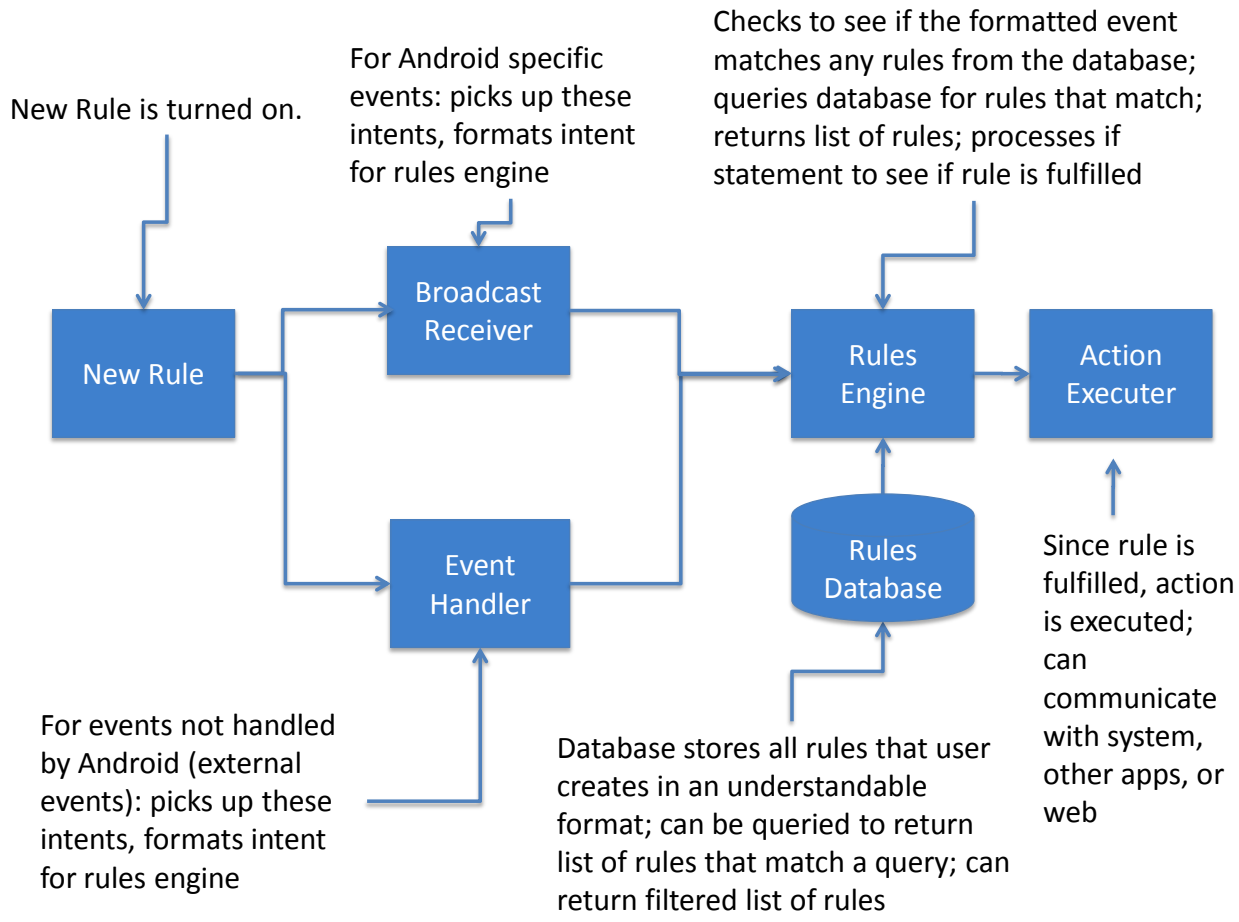

The tutorial page is an idea we had that would help first time users understand how to operate the app. There was also an idea for a tutorial overlay for the app that would direct users towards options and functions for app operation.

# Android Screen Flow Diagram




This Original flow diagram represents the screen flow for users operating the app

# Rule Handling Workflow (Backend)




This flow diagram represents the initial backend design for the app.

# Current Design

## User Interface

The user interface (UI) of the application is the portion of the application that the user directly interacts with. This is really the only portion of the app that the user will ever see. This portion of the application is critical, as it molds the application experience and determines the user's overall satisfaction with it. The interface is constantly changing from version to version, since it must cater to the user's need. The interface must be user centric, and a bad interface is one which is difficult for the user to complete tasks and user stories. We have specifically chosen to leave out any sort of tutorial for the first release of this senior design project since the user interface will constantly be changing in each revision during our proof of concept. In the long term, our client wishes to avoid the need for a tutorial to teach the user how to use the application, because we feel that the interface must be intuitive and centered on the user. If the user cannot figure out how to complete user stories without instruction, we must redesign to fix this rather than train the user to use our interface. We do however, include a help button to link to reference documents for the user to consult if necessary.

As new causes, effects, rules, and features are added to the application, the UI must be updated to reflect these changes. The initial screen shown to the user is the main menu. This menu is the first navigation to be interacted with, and lets the user continue to view a list of the rules, create a new rule, view help documents, change user preferences, and share rules with other devices.

Each screen was designed to be intuitive and have a natural flow while navigating from screen to screen. These activity screens were also designed to have a vertical orientation, to avoid the confusion of screen rotation. Certain features were important, so we used color contrast, size, font, and location to emphasize these elements of the design. Like many other Android features, we designed our application update as the user makes changes to rules. As an example, the user can press once in the Wi-Fi settings to turn Wi-Fi on. Our clients expressed interest in having our application have a similar instance gratification feel. All changes to be effective immediately could include all CRUD operations: Create, Read, Update, and Delete. As soon as a user makes a change, the changes are reflected on the GUI and in the database.

Continuing into the app, each new screen shown to the user is known as an activity in the Android operating system. The operating system keeps track of background activities in memory in a data structure known as the "back stack." The back stack is a LIFO collection of all activities created by the application. Each time the application creates a new activity, it is added to the top of the stack. Each time the user presses the back button on the device, the current

activity is “popped” from the top of the stack, destroyed, and the next most recent activity is brought into the foreground. It is important to ensure that the application does not create too many activities, because it can result in slower performance over an extended period of time. To combat this, each time a new instance of an activity is requesting to be created, a flag is set on the request to ensure that an instance of activity does not already exist in the back stack. If the activity does not already exist, a new one is created. If the activity is already existing and buried in the back stack, the application will locate the existing instance, pop all newer activities off it from the stack, and bring the instance into the foreground.

Navigation within the application is one of the most important aspects of the interface. A bad navigation scheme can result in confusion to the user and a bad overall user experience. As shown in Figure 1 (below), our application takes advantage of a hierarchy structure to center the navigation around. The user is able to organically navigate through typical workflows from within each activity and go deeper into the hierarchy to complete tasks. The “back” hardware button on the Android device will take the user back and activity based on temporal navigation, meaning the last activity seen. This can be analogous to the back button on an internet browser. The “up” button is located on the top left corner of the application and consists of the application logo and a back arrow. This button is used to ancestral navigation and will take the user up one level in the hierarchy regardless of the path the user took to get to the current activity. This is similar to navigating up one directory in a folder structure.

The application also takes advantage of the Android SDK’s “action bar” for further navigation. Except for home screen, each activity has a menu bar at the top of the screen. This contains the “up” button, in addition to a variety of additional button specific to the activity. Each activity will have a different workflow, so it is important for us to make sure this is tailored for the optimal experience for the user and does not get confusing. Most typical activities, contain a new rule button and a share rule button. These button navigate the user to the respective activities. The menu overflow button is includes in almost every activity and will contain navigation to tertiary actions, such as the help page or the settings page.

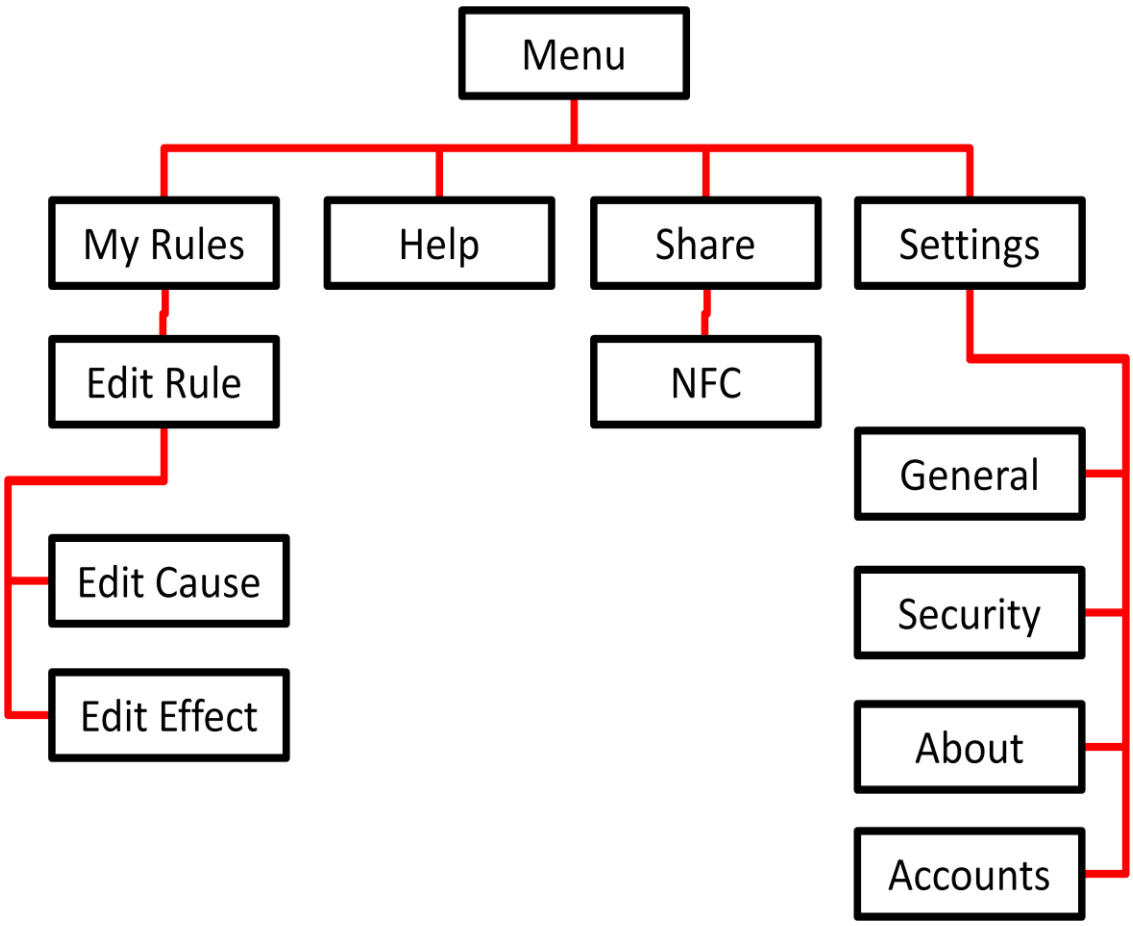
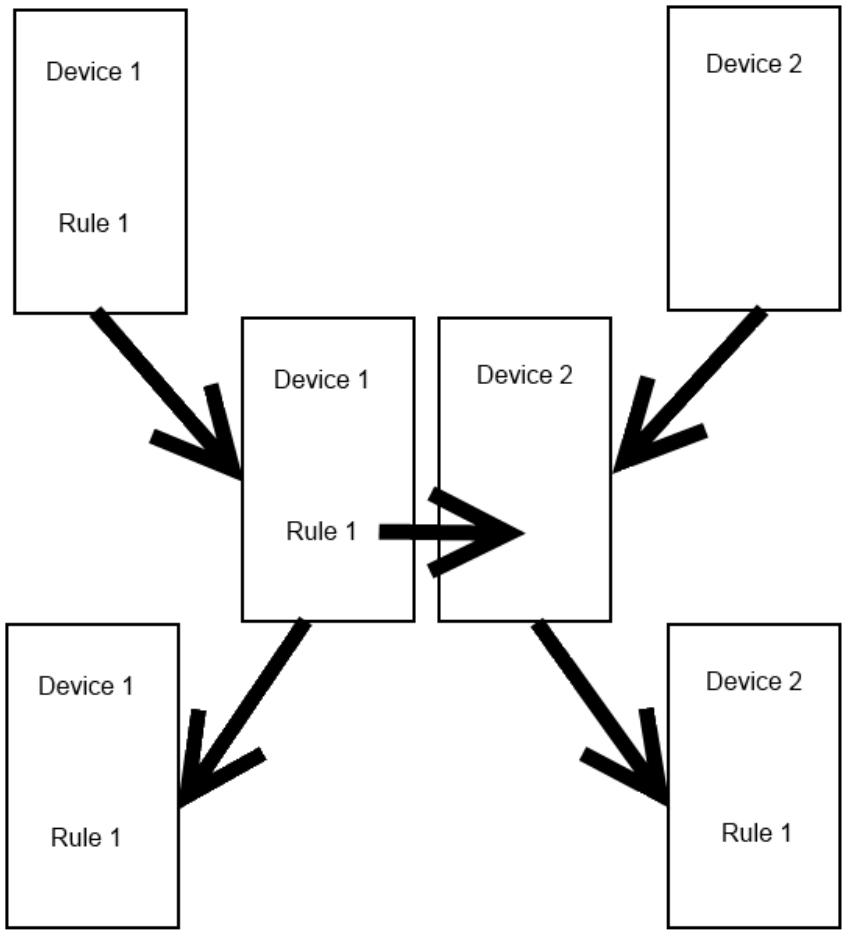



**Figure 1.** The Application Hierarchy

Future designs of the application will remove the initial menu screen from the application and jump straight into the "My Rules" activity. We felt that it was important to push the user directly to the content as quickly and effortlessly as possible, and the menu screen adds an additional level of friction for this. Although the application and client logo were initially a benefit of this "splash screen," they were mostly for branding purposes, and we were able to emulate good branding by following our client's color scheme. Furthermore, the next designs of the application will also take advantage of the contextual action bar. This is a modification of the existing action bar and will come into play when the user is executing specific tasks. For instance, the user will be able to select one or many rules by touching the checkboxes next to them. The original action bar will be temporarily replaced with the contextual action bar containing task-specific action buttons, such as edit, share, and delete.

# NFC Sharing

Users have the ability to share their rules with other users via NFC. To access this feature, the user must choose the “Share” button from the main page, choose a rule to share, and press their phone/android device to another and push the screen to share the rule with the other device. The sending device sends the selected rule in plain text form. The receiving device will parse the string and create a new rule in the database.



## The Application Service

The entire Rules Engine, consists of the Broadcast Receiver, moving into the Rules Engine class, and then finally to the Action Executer, if the causes are evaluated to be true. This entire engine is actually stored within a separate service that handles the application's backbone. The service is created when the application is first opened and has longevity compared to a simple Broadcast Receiver. The key feature of the service is that the Broadcast Receiver is registered to the service instead of the application itself, and therefore will run smoothly regardless of the application's current state. This fixed many problems with false positive causes and random data being evaluated for location and Wi-Fi based causes. Simply, the entire backbone is run in its own process, is much more permanent, and evaluates data from all types in the correct manner.

# Broadcast Receiver

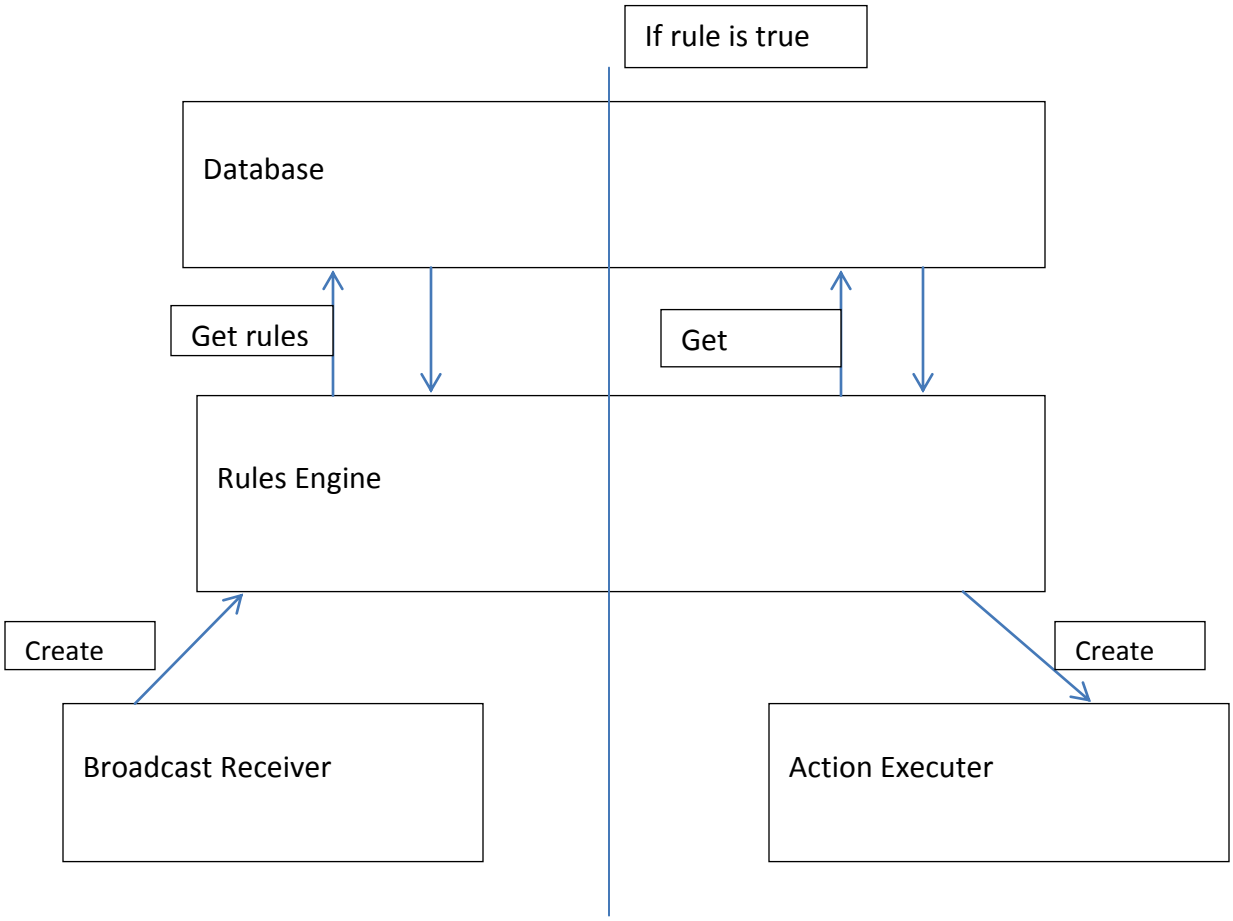
The first part of the rules engine is the broadcast receiver. The broadcaster receiver is a class built into the Android API that either sends or receives messages to/from the Android operation system. In our case, the broadcast receiver receives the different cause types and sends the data to the Rules Engine to process the causes if they exist in the database. The reason why the broadcast receiver is so light is because the broadcast receiver is only given 10 seconds to operate after it receives a broadcast. If the broadcaster receiver is still active after 10 seconds, the Android operating system will kill the process and the intent is lost. Therefore, we kept the broadcast receiver light and send off the commands with the minimal amount of processing. The broadcast receiver is also designed to work in the background so that it will be active even when the application is no longer focused.

# Rules Engine

The main function of the rules engine is the evaluation of rules. The rules engine object is what determines whether a rule is true or false. A rules engine object is created by the broadcast receiver once an event has been triggered. The broadcast receiver passes in the cause type, parameter and the context to the rules engine. The rules engine uses the type to send a request to the database for all rules of that type. Once the list of rules has been received, the rules engine evaluates each rule in separate threads using Android's AsyncTask.

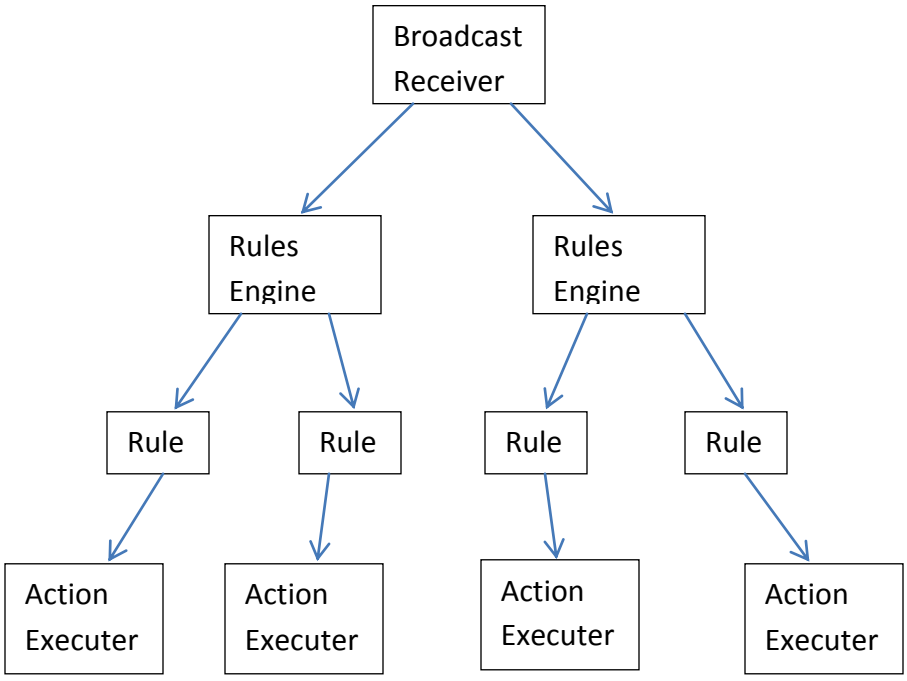
In each thread, an expression tree of rule causes is built using a stored character string in each rule. This expression tree is then evaluated recursively in a depth-first fashion. Each individual cause object within the tree has its own isTrue() method that returns either true or false depending on what value is evaluated for the cause. Using these returned values, the expression tree will return a final value, true or false, to the AsyncTask. If the tree of the rule returns true, then the task will request the effect list for the rule from the database, should it exist. This effect list is then sent to an ActionExecuter object for execution, along with the context initially passed from the broadcast receiver. If there are no effects, the rules engine will not create an ActionExecuter, since there are no actions.

## Structural View



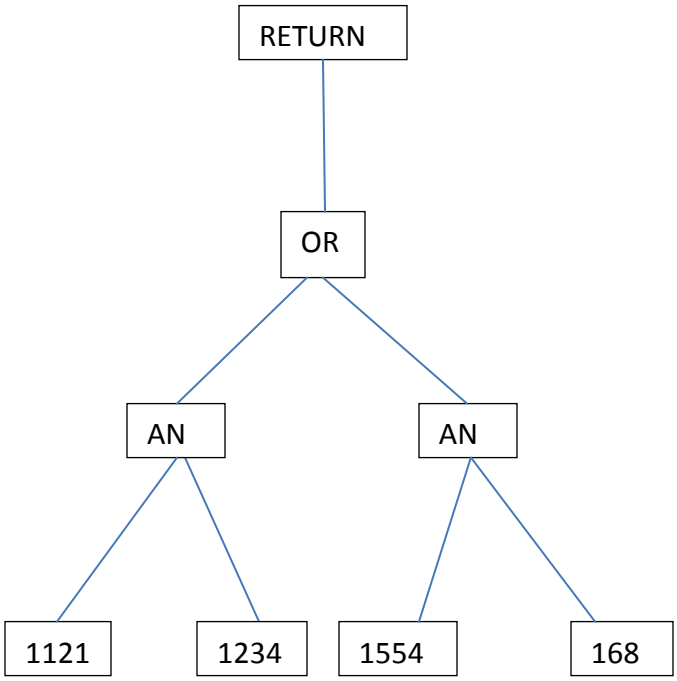

The rules engine interacts with the broadcast receiver, the database, and the action executer depending on if the rule cause tree returns true or not. If the rule evaluates to be false, then the right half of the view does not occur.

## Thread View




Each rule stemming from a rules engine object is evaluated in its own thread. Should the rule return true, then an effect list is fetched from the database and an action executer object is created, which takes in the effect list and executes each effect.

## Expression Tree




This is a view of the expression tree used to evaluate rule causes. This particular tree is the tree for a Boolean algebra equivalent of (1121 & 1234) + (1554 & 1685). The character string stored in the rule for this tree would be "1121,1234,&,1554,1685,&,+$". '+' indicates an OR node. '&' indicates an AND node. The expression tree string is stored in such a way that the tree is built from the leaves using a stack. Every '&' and '+' pushed to the stack causes two nodes to be popped as its children in the tree from the stack. The '$' symbol represents an end of line.

## Individual Cause Evaluation

Causes are evaluated using a function containing a switch statement that chooses evaluation methods based on the rule type.

Pseudo code for cause evaluation:

```
Bool isTrue(eventType, param)

{

// event type is given to use by the broadcast receiver

     Switch(eventType)

     {

// each case returns true or false depending on specific
//conditions of each case

          Case wifi: // do something

               Break;

          Case phoneCall: // do something

               Break;

          Case time: // do something

               Break;

          Default: return false;

     }

}
```

# Database

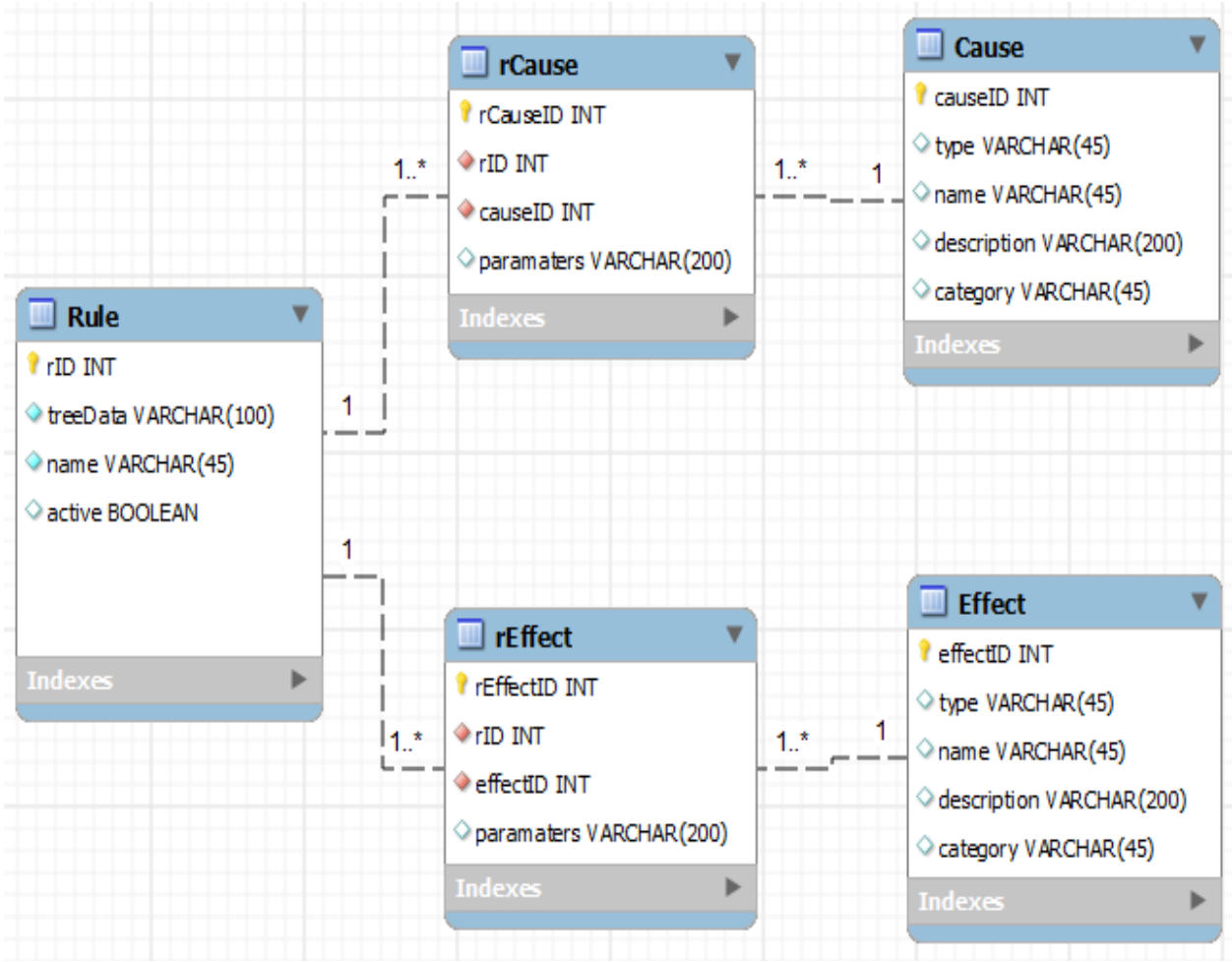
While developing the structure of the android application we decided that we needed a full database to hold all of the relevant information that we would need. This database would keep track of all causes, effects, and rules information so we needed it not only to be large but secure as well. We inquired with Samsung to see if they had any preferences on what we should use to accomplish this task and were presented with TouchDB (https://github.com/couchbaselabs/TouchDB-Android). We started playing around with it to try to get used to it but found the documentation for it to be subpar. After this setback we decided to go with SQLite since we needed to keep the database lightweight to keep our app from taking up too much memory on the phone as we remember it has to be constantly running in the background, but still contain all the search power of an SQL implementation.

We had to set up a structure for how we wanted the databases to look. This included how we would want our pulls to appear from the database. So we started with creating the objects that we would be using in our program to decide how many objects we would need to keep it lightweight while balancing that with how many variables were in each object. We also had to keep track of links between the objects to finally pull a full rule out of the database if we were to need that much information. There was a bit of a steep learning curve between not only learning SQL, but then to learn the limitations that were present with using SQLite as well as making the database so large with a number of tables and foreign keys.

Getting a little more specific into how the database was made, we have to first start by looking at how the objects were made that would basically be held by the tables. The cause object is one of the base objects so it needed to hold a lot of information including: cause id, type, name, description, category, and account id (which we will be implementing later). The cause id would of course be the primary key to keep track of which cause you would be talking about in the rCause table. It was at this time the auto-increment functionality became our friend as we discovered that it was already implemented in the tables and assigning a primary key actually renames the column to whatever you called it (led to many headaches and head scratching). The cause table would be kept separate from the rCause table since it needs to be an overarching list while the specifics that would needed to be implemented in each rule would be handled by the rCause table. The effect tables on the other side of the chart were basically mirror images of the cause tables.

When looking at the rules objects you realize that you need the string to keep track of how the rules interact (and/or) as well as a name to display and of course a primary key. A rule id is kept as well in the rCause/rEffect tables. Finally, the account table is not fully implemented yet in our application, but soon will be. The breakdown of the account (as we know right now) is that it must contain a (service) name, URL, username, and password. All of these are essential in

logging into certain other services such as Facebook, Twitter, etc. Below, you will see a display of how the tables talk to each other:




After that whole database is set up, you then need to be able to do multiple functions to accommodate for the database handler which is what the program needed to be able to get their information. Some of the most obvious ones were to insert causes and effects into the table. Whenever you start up the application for the first time you will be inserting all of the causes and effects from either a text file (to possibly be implemented later) or just being read in hard coded (current implementation). Now for inserting the rule you also need to be inserting the specific rCause and rEffect information since these are important parts of the rule. The application will be set to remember all of the rules from previous uses, because who really wants to have to input the rules every time they start up their phone.

Once you are done inserting all of the information for the application, there were the actual parts of the database that the other parts of the program needed to be able to access. The first ones I was told to implement were to return all cause, effect, and rule objects (separately). This was pretty easy as you just use an SQL statement to pull all of the information out of a certain

table and put the information into an ArrayList. After those methods got done, we started to get a little more complicated in how we wanted to call from the database. The first method was to return all cause objects where the category of that cause is equal to a given category. So we passed a string in, used it in a SQL query to create a result table that you would pull from to get the wanted information. So of course the same thing also needed to be done for effects and rules. The rules one was actually more complicated since you had to go into the rCause and rEffect tables with external columns from other tables using joins (since there is no category column in the rules table or the rCause/rEffect).

There also needed to be a function to return a full cause object given a cause id. We also needed a quick function that received all rule objects where the type of that rule is equal to the given type, making sure that the effect list and cause tree are null, but the cause tree string is still there. This was most important for early parts of the application when you are pulling up some information quickly to display to the user while it doesn't take up much processing power. And yet another method was to return an effect list given the rule id, which just made sure that we had all the method calls for the other parts of the program.

As we added more functionality, more functionality needed to be added to the database that included deletion of rules, rCauses, and rEffects. And as we got larger sets of rules, we needed to make sure there are no duplication of causes or effects in a rule. This quality of life update came as a change that made the program more efficient with not allowing the user to create basically a large rule that only really had a single cause or effect (conserves space on the phone and processing power).

# Action Executer

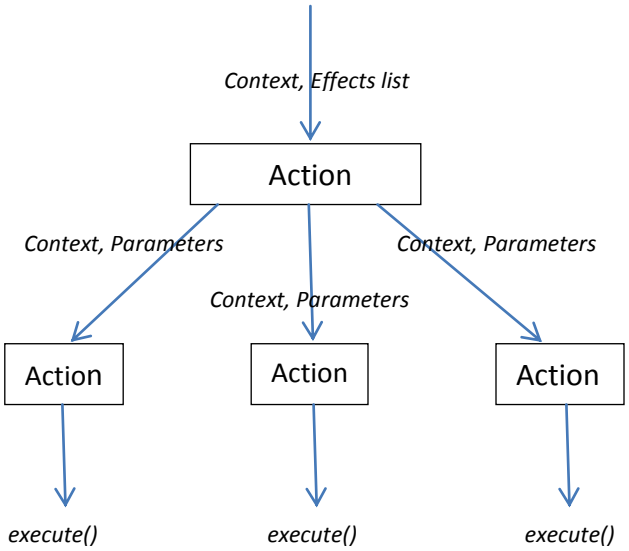
The main function of the action executer is the carrying out of effects of a positively evaluated rule. The action executer is what carries out the actual actions that are associated with a rule. The action executer object is created by the rules engine each time a rule has been evaluated as true. Many action executers can exist at one time as multiple rules have been triggered. The rules engine passes the context of the current state of the application as well as an array list of the rEffect objects from the rule. Once the object has been constructed, the action executer will loop through all the effects in the array list. In each loop, the "type" of the effect is enumerated and evaluated as the argument in a switch statement. This is more efficient than using many if statements. The individual case that is evaluated would create a new action object and execute the action.

Each rule type has a unique action class that inherits from AsyncTask. AsyncTasks are Android's native method of threading and allows for the parallelization of actions being executed. Each thread will be made of an action that can be executed. For instance, a separate thread is created to execute a "toast" (Android's most simple notification to the user). The individual action classes are passed both the context of the action and the parameters needed to execute that action in the form of a string. The constructor of the action will parse the parameters to make sense of one long string before executing the action. Each action type has a unique method for parsing the parameters because different rules require different parameters (some rules do not have any parameters). The "execute" function of the action class contains source code that will carry out the individual task, using both the application context and the parsed parameters.

The specifications of the action executer require it to be very lightweight and adaptable. The possibility of several action executers existing in memory and executing effects at one time is very probable. The action executer must also be able to execute any type of action. It must be universal enough to create an action object of any type needed and pass the necessary application context and parameters.

Our current version of the action executer remains relatively unchanged from the initial design. It has been stable and has not caused any problems at all with the application. Any changes to the action executer have been adding new effect types for execution.

## Thread Diagram




Each action object stemming from the action executer is executed on its own thread.

# Risk Assessment and Management Plan

| Risk | Type | Probability | Consequence | Strategy |
|---|---|---|---|---|
| The application does not adapt to all targeted Android devices. | Technology | High probability | Tolerable | We will program our application keeping in mind that our application must be dynamic and usable across a variety of devices, regardless of hardware, API level, and screen resolution. |
| The application leaks personal information to hackers. | Technology | Moderate probability | Serious | We will be using the highest available levels of security (OAuth 2.0) to encrypt user data and account information. |
| Team members are frequently sick or unavailable to work on the project or meet. | People | High probability | Tolerable | We will communicate frequently about our school work and upcoming assignments when assigning project tasks. We will set times and importance for our meetings, so that all team members will be available and accountable for each other. |
| Changes to requirements that require major design rework are proposed. | Requirements | Low probability | Serious | We will frequently be communicating with our client to be sure they are satisfied with our progress and results. Any changes to requirements will be up for discussion and source code will be distributed and dynamic enough to adapt. |

| Risk | Type | Probability | Consequence | Strategy |
|---|---|---|---|---|
| The time required to develop is underestimated. | Estimation | Moderate probability | Serious | Team members will regularly meet to discuss current status and estimate deadlines. As different tasks and components are completed, team members will take on new tasks to complete. |
| The source code is inefficient and the application is slow when executed. | Technology | Moderate probability | Insignificant | Our team will program efficiently and constantly review each other's source code to look for areas of improvement. |
| The test devices are not sufficient for testing. | Tools | Low probability | Insignificant | Members of the team have personal devices to test on. New devices may be purchased for the purpose of testing. |

# Agile Development Schedule

Agile Development has been quite an interesting trip so far. Agile development is essentially the waterfall cycle in short sprints called “iterations.” The entire requirements, design, implementation, testing/verification, and release process is including in a single sprint of agile development. The requirements are broken into development needs called “User Stories.” User stories one of the most unique parts about agile development. When a new iteration is started, user stories are chosen and ranked in a game called “Playing Poker” in the field. This allows developers to come to coordinate which tasks will be achieved during any given iteration by group decision and analysis of user story difficulty. The group ideally takes on an appropriate amount of user stories and begins the iteration with these user stories in mind. Each user story is very similar in concept to the change request forms idea from other development cycles. Each user story is its own change to the last release that has been decided upon, and ranked by the group. Typically (and ironically), there is little or no documentation for agile development projects. One feature of agile development is the SCRUM architecture. The typical agile development team meets every day in meetings called “Standup Meetings.” The daily standup meeting is critical to the success of the team. Since agile is such a fast moving development structure, these meetings are used to both check on the progress of development and discuss development issues, clarify user stories, clarify client’s needs, coordinate the group’s efforts, and unite the team members.

The SCRUM master (our team leader, Keith), is in charge of running the scrum meetings for our team. Normally, the SCRUM master is not a developer. However, all 4 members of our team are developers and are responsible for code. This has led to an interesting group dynamic. To spread out duties from the typical agile development team, the acceptance of user stories is handled by Matthew and William. The note taking for all of our meetings internally and with the client are handled by Tomin. Our team meets 4 times a week, at 12:30PM on Tuesday and Thursday, and at 3:00PM Monday and Friday.

Our iterations are 2 weeks in length. Iterations in agile development vary from 2-4 weeks but each team decides on a length and sticks to it. Our iterations start on Saturday and go through the Friday that is 13 days later. The finished code with all the iteration’s user stories is due by the Wednesday on day 12 of the iteration. Between day 12 and day 13, all acceptance testing is handled. If a user story is not accepted, the story is either patch fixed (if possible) or pushed into the next iteration’s user stories as a high priority user story. Between day 13 and day 14, the team members submit documentation on each user story they completed and on day 14, we plan the next iteration’s user stories and rank them using Playing Poker. The next iteration then starts immediately on the following Saturday.

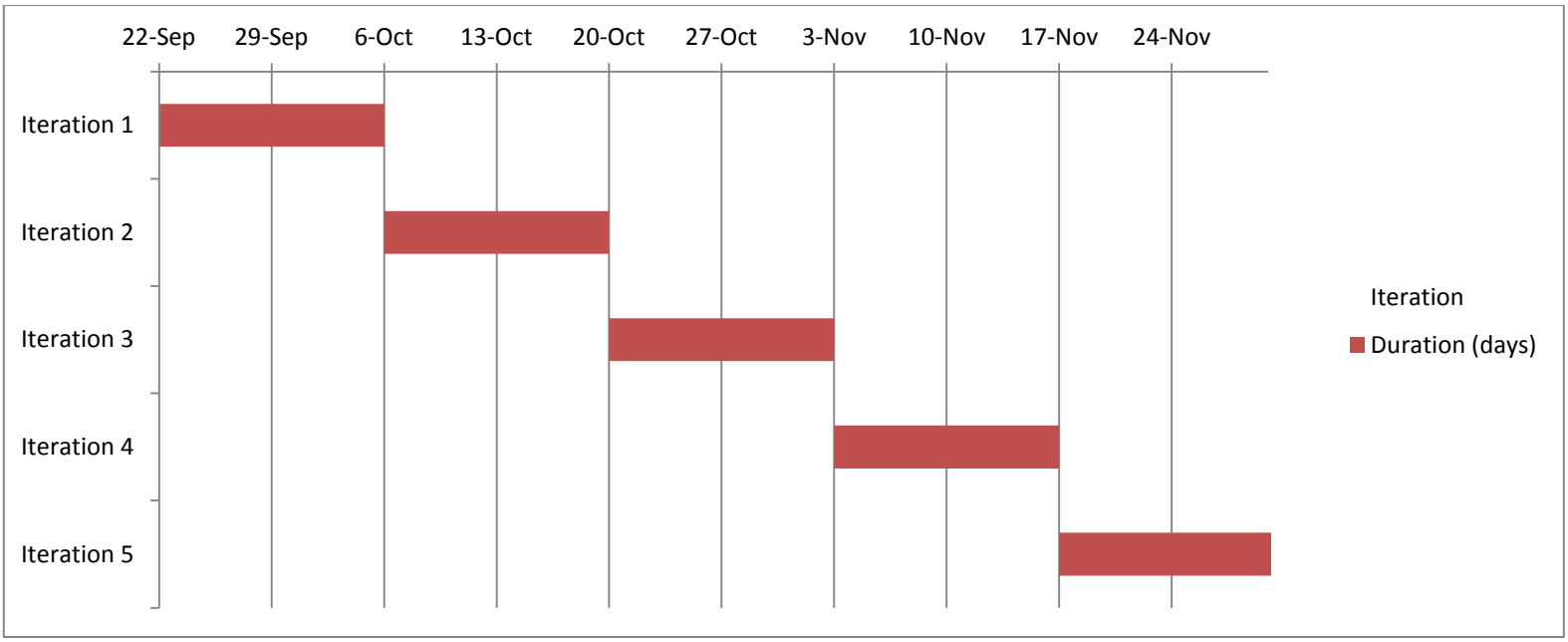
Our iteration plan is to have 5 iterations in fall and 6 in spring. This gives us a total of 11 iterations, so 22 weeks of formal development. During our first iteration, we defined most of our user stories and developed a mock GUI so we could begin testing functionality. During our second iteration, we were advised by our client to spend additional time to design the rules engine structure. Our release for iteration 2 was just documentation and a few cleaned up features on the mock GUI. This structure is the foundation of the functionality of our application and is complex to parallelize and integrate. To account for this, iterations three and four were designed to be the first and second halves of the rules engine. With school ramping up, iteration 5 was designed to merge, clean, test, and optimize the application before the PDR. Since we are doing agile development, this presentation will also include a demo of our iteration 5 prototype release. It just so happens (thanks to signing up early) that iteration 5 ends on the same day as the PDR.

We planned for a lighter iteration during Thanksgiving (to prepare for the PDR), and made the iteration over spring break a 3 week iteration to account for members out of town. The great thing about agile development is that all of the user stories that have not been assigned to an iteration live in the backlog. The backlog currently consists of a few functional user stories and the entire list of appendences at the end of this document. The benefit to this is that winter break doesn't have a formal structure encapsulating it (to respect the holidays and difficulty of meeting for SCRUM stand-ups), but instead the backlog will be updated with priorities so team members can work on user stories during the break and clear up that list. Our intention is to set a minimum amount of stories for each person to complete over break and let everyone develop when they have time. So yes, we are have a plan and are working over winter break.

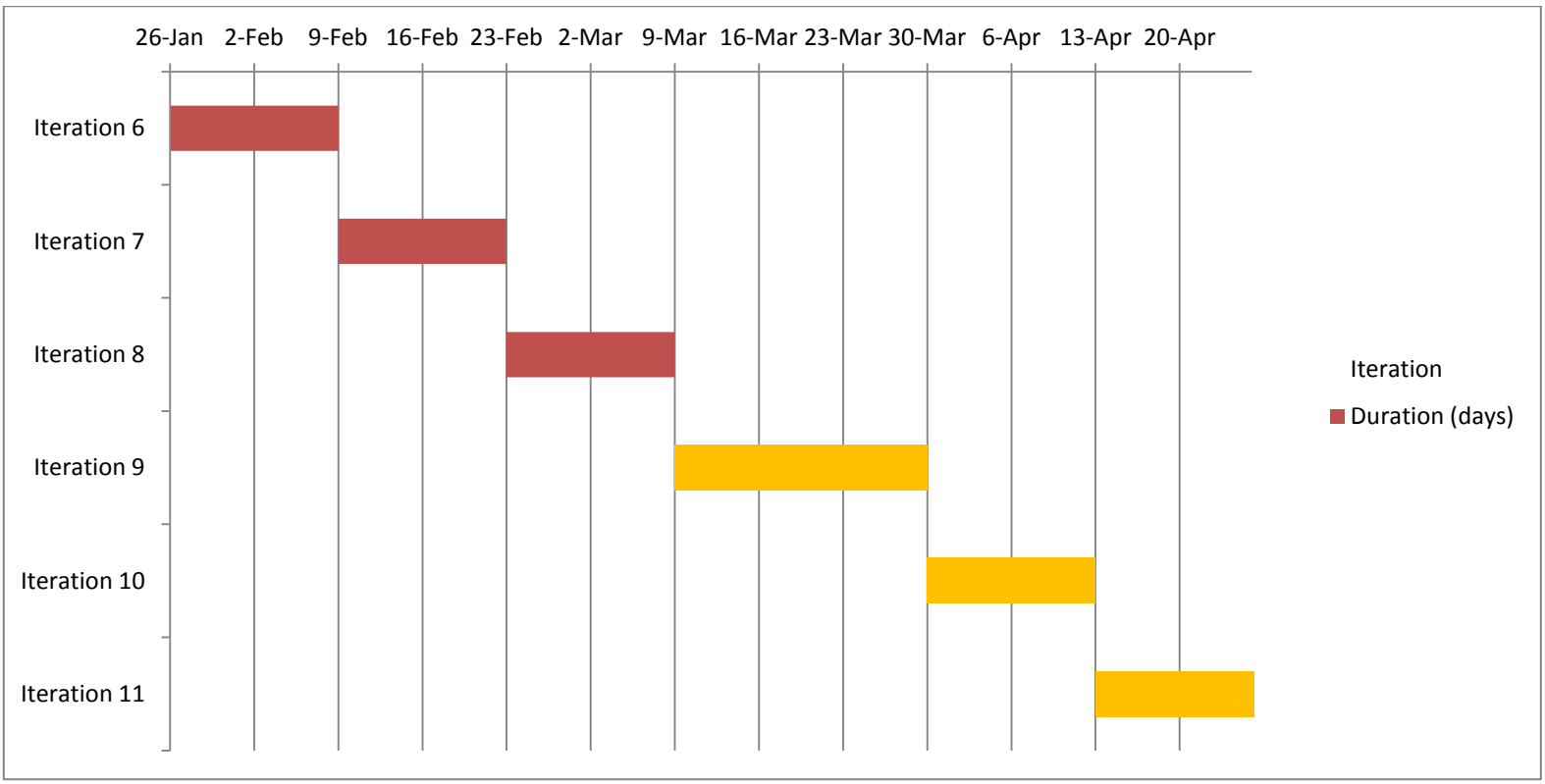
This spring, we he have worked really hard. We started off the semester with a presentation to Samsung the first week of Iteration 6. After receiving critical feedback, we planned the structure of our iteration schedule until the MDR. The past 6 weeks (3 iterations) have not allowed for a lot of sleep, but we have implemented all of the original feature requirements from Samsung. For the FDR, we will take more Samsung feedback and continue to clean, test, and debug the app to be as robust and user friendly as possible.

# Gantt Charts

## Fall




## Spring



# Budget

**Proposal Budget**

Project Name: Cause & Effect

**Period #1: 9/22/2012 - 4/20/2013**

| Description | Rate | Hours | Subtotal | Total |
|---|---|---|---|---|
| Employee Salary | | | | |
| Keith Adler | | 550 | $13,750.00 | $13,750.00 |
| Matthew Brannick | | 440 | $11,000.00 | $11,000.00 |
| Tomin Kozhimala | | 440 | $11,000.00 | $11,000.00 |
| William Vennes | | 440 | $11,000.00 | $11,000.00 |
| Employee Salaries Subtotal | | 1,870 | $46,750.00 | $46,750.00 |
| Employee Benefits | | | | |
| Emp Ben, FT Exempt Staff (42%) | | | $19,635.00 | $19,635.00 |
| Emp Ben, Non-Exempt Staff (42%) | | | $0.00 | $0.00 |
| Employee Benefits Subtotal | | | $19,635.00 | $19,635.00 |
| **Employee Salaries and Benefits Subtotal** | | **1,870** | | **$66,385.00** |
| Equipment | | | | |
| Samsung Nexus S (on Loan) | $0.00 | 2 | $0.00 | $0.00 |
| Samsung Epic 4G (on Loan) | $0.00 | 1 | $0.00 | $0.00 |
| Samsung Galaxy Tab 10.1 (on Loan) | $0.00 | 1 | $0.00 | $0.00 |
| Equipment Costs Subtotal | | 4 | $0.00 | $0.00 |

| | | | | |
|---|---|---|---|---|
| Software | | | | |
| Eclipse IDE | $0.00 | 4 | $0.00 | $0.00 |
| Android SDK | $0.00 | 4 | $0.00 | $0.00 |
| Bitbucket Repository | $0.00 | 4 | $0.00 | $0.00 |
| Mercurial Source Control | $0.00 | 4 | $0.00 | $0.00 |
| Android Open Source Project Code | $0.00 | 4 | $0.00 | $0.00 |
| Software Costs Subtotal | | | $0.00 | $0.00 |
| | | | | |
| Printing | | | | |
| PDR Printing | $75.00 | 4 | $75.00 | $75.00 |
| FDR Printing | $100.00 | 4 | $100.00 | $100.00 |
| Printing Costs Subtotal | | | $0.00 | $0.00 |
| | | | | |
| **Total Equipment, Software, and Printing Costs** | | | | **$175.00** |
| | | | | |
| Contingency Costs (20%) | | | | $35.00 |
| | | | | |
| **Project Costs Total** | | | | **$66,595.00** |

Our current proposed budget is $66,595. The budget above represents our total projected budget for the duration of the Cause & Effect Project. These figures for employee salary and benefits were estimated using an hourly wage of $25.00 per hour. This is the approximate starting salary for a Software Engineer at Samsung Mobile (Glassdoor). The group leader is estimated to work 25 hours per week, while the remaining group members are estimated to work 20 hours per week. Employee benefits were estimated to be close to 42% of the rate of wages and salaries (Bureau of Labor Statistics).

Our equipment consists of four devices given on loan from Samsung Mobile to assist us in testing our application. These include two Samsung Nexus S developer phones, a Samsung Epic 4G, and a Samsung Galaxy Tab 10.1.

In order to construct a software environment that was both robust and economical, our team took advantage of many free and open source software programs. Android Developers provide

the necessary Android Software Development Kit (SDK). We are currently using Eclipse as our Integrated Development Environment (IDE) with the Android Development Tools (ADT) plug-in, which provides us with a powerful integrated environment to develop out application, as well as build an app UI, debug, and export app packages (APK) to distribute and test. Our environment also includes Bitbucket hosting service and our team was able to take advantage of using our .edu email accounts to register with Bitbucket and create a free private repository to host our source code. We chose to use the free Mercurial distributed source control to manage our software code at our client's recommendation and our own interest. Finally, the Android Open Source Project provides us with a free developer's website with many resources for reference about the operating system (developer.android.com).

Finally, two of our project's review checkpoints will be accompanied by hard copy reports. The PDR and FDR will each have five copies of the final report which will be double-sided printed and bound. The PDR reports each had 61 pages and cost approximately $75. We anticipate our FDR report to contain 80 pages and cost approximately $100. We will be printing and binding these final reports at FedEx Kinko's.

# Source Code Repository

Our Repository can be found at:

**https://Talador12@bitbucket.org/Talador12/ceandroid**

Feel free to request read privileges and/or follow our project.

# References


[**Glassdoor**]

http://www.glassdoor.com/Salary/Samsung-Group-Software-Engineer-Salaries-E3363_D_KO14,31.htm

**[Bureau of Labor Statistics]**

http://www.bls.gov/news.release/ecec.nr0.htm

**[Android Developers]**

http://developer.android.com/guide/components/index.html

**[TouchDB-Android]**

https://github.com/couchbaselabs/TouchDB-Android

**[Mercurial Source Control]**

http://mercurial.selenic.com/

**[Bitbucket]**

https://bitbucket.org/

# Backlog Appendix

This is a list of all of the low priority user stories on our backlog. It includes all of the rule types implemented by our competitors and more.

## Phone/Physical Rule User Stories

- As a developer, I want to be able to integrate Bluetooth functionality, so that users can use rules and sharing that utilizes the phone's Bluetooth capabilities.
- As a developer, I want the app to feature media message based rules, so that the user can handle complex message types.
- As a developer, I want the application to access the display features, so that the display can be turned on or off as desired by the user.
- As a developer, I want the application to recognize when it is docked, so that the user can create rules based on docking status.
- As a developer, I want the application to be able to send fax messages, so that our users can use this functionality.
- As a developer, I want the application to recognize when headphones are plugged in, so that the user can create rules based on headphones.
- As a developer, I want the application to recognize when an external display is plugged in, so that the user can create rules that interact with the external display.
- As a developer, I want to include rules associated with the calendar, so that users can interact with their schedules using our application.
- As a developer, I want to take advantage of Android (and especially Android 4.1)'s voice commands, so that the user can interact with rules using their voice.
- As a developer, I want the application to be able to open or terminate other applications, so that the user can optimize application flow on their phone.
- As a developer, I want the application to be able to access the phone's memory and storage, so that the user can interact with the phone's data hardware.
- As a developer, I want rules that affect the battery usage, so that the user can optimize battery life using our application.
- As a developer, I want to access the phone's alarm system, so that the user can edit their alarm settings using rules.
- As a developer, I want to be able to toggle airplane mode, so that the user can access this feature using rules.
- As a developer, I want to be able to change the wallpaper using our app, so that users can have dynamically changing or updating wallpapers.

- As a developer, I want to be able to manage files on the device use our app, so that users can manage their file structure using rules.
- As a developer, I want to access the camera, so that users can create rules that can utilize the camera's features (new photo, flash, etc).
- As a developer, I want rules that can access the current time zone, so that the user can utilize different time zones as they change.
- As a developer, I want rules that access email, so that the user can manage their email using our rules engine.
- As a developer, I want rules that can access the web, so that the user can access web content using our rule system.
- As a developer, I want the access the state of the USB tethering, so that the user can build rules from this.
- As a developer, I want to be able to build rules for a wireless hotspot, so that the user can utilize their over expensive data plan or rooted phone.
- As a developer, I want to be able to access the phone's Radio, so that there can be rules that use the radio.
- As a developer, I want to be able to access different profiles, so that the user can enable or disable sets of rules at any time (even with rules).
- As a developer, I want to integrate rule types for the lock screen, so that the user can create rules using the lock functionality.
- As a developer, I want rules that access the motion sensors/accelerometers, so that the user can create rules based on the motion of the phone.
- As a developer, I want rules that access the gyroscope, so that the user can create rules based on the device's orientation.
- As a developer, I want to access when the phone is shutdown or restarted, so that the user can create rules based on these actions/events.
- As a developer, I want to know when the first boots up, so the user can respond using a set of rules.
- As a developer, I want language based rules, so that the application can be used internationally and for translation.
- As a developer, I want debug based rules, so that the user can create sets of rules for debugging their applications (including this one).
- As a developer, I want to be able to interact with a printer, so that the user can create rules that interact with the printer.
- As a developer, I want to allow the user to directly input JavaScript, so that a technical user can use the rules engine to its fullest.

- As a developer, I want to access the weather status, so that the user can use rules based on this.
- As a developer, I want to access emergency messages, so that the user is quickly informed of hazards.
- As a developer, I want to access the news, so that the user can create rules based on this.
- As a developer, I want to create rules based on traffic, so that the user can create rules for this.
- As a developer, I want to access Stocks, so that the user can integrate with any stock service.
- As a developer, I want to access Banking, so that the user can manage their banking using rules.
- As a developer, I want to store VPN settings, so that the user can manage VPN connections.
- As a developer, I want to react to Data roaming, so the user can include this state in rules.
- As a developer, I want to access Voicemail, so that the user can manage voicemail using rules.
- As a developer, I want to know when something is downloaded and uploaded, so the user can create rules for this.
- As a developer, I want to be able to access movie times, so that the user can create rules for this.

# API Based User Stories

- As a developer, I want to access Dropbox, so that the user can manage their files stored on the cloud.
- As a developer, I want to access Evernote, so that the user can manage their notes using this rules engine.
- As a developer, I want to access Flickr, so that the user can manage photos using this application.
- As a developer, I want to access Craigslist, so that the user can interact with this service using our application.
- As a developer, I want to access Blogger, so that the user can interact with blogs using rules.
- As a developer, I want to access RSS feeds, so that users can integrate their rules with the feeds.
- As a developer, I want to access Gmail, so that the user can use Google email as an email source for rules.
- As a developer, I want to access eBay, so that the user can interact with this service using our rules.
- As a developer, I want to access Instagram, so that the user can manage their sepia stained photos.
- As a developer, I want to access Twitter, so that the user can tweet using our rule system.
- As a developer, I want to access Steam, so that users can interact with their gaming communities using this application.
- As a developer, I want to access Foursquare, so that users can interact with this service and sync the user's location using this service.
- As a developer, I want to access Google+, so that users can interact with social media.
- As a developer, I want to access Pintrest, so users can manage their boards using our app.
- As a developer, I want to access Path, so users can stay connected with family & close friends.
- As a developer, I want to access last.fm, so users can manage music using our application.
- As a developer, I want to access LinkedIn, so users can manage their professional identities.
- As a developer, I want to access Google and Yahoo Finance, so users can manage their financial interests using our app.
- As a developer, I want to access YouTube, so they can manage their video accounts.
- As a developer, I want to access Word Press, so they can manage their blogs.
- As a developer, I want to access Tumblr, so that users can manage their Tumblr content.
- As a developer, I want to access StumbleUpon, so that users can stumble using our application.
- As a developer, I want to access Yahoo, so that users can manage their Yahoo accounts.

- As a developer, I want to access Windows live, so that users can integrate with their Window's live features.
- As a developer, I want to access ESPN, so that users can manage their sports.
- As a developer, I want to access GroupMe, so that users can manage group text messaging using our app.
- As a developer, I want to access Samsung Kies, so that users can update their hardware using this application.
- As a developer, I want to access Amazon, so that users access this service.
- As a developer, I want to access Newegg, so that users can interact with this online store.
- As a developer, I want to access Google play store, so that the user can create rules involving the Android marketplace.
- As a developer, I want to access Google voice, so that users can integrate with voice accounts.
- As a developer, I want to access AIM, so that users can interact with this messaging service.
- As a developer, I want to access Google Talk, so that users can interact with this service.
- As a developer, I want to access Skype, so that users can text/audio/video with other users.
- As a developer, I want to access Netflix, so that users can build rules for this type.
- As a developer, I want to access Hulu Plus, so that users can build rules for this type.
- As a developer, I want to access Kindles, so that users can interact with these accounts and devices.
- As a developer, I want to access Wikipedia, so that users can be well informed.
- As a developer, I want to access App.net, so that other developers can create rules based on this wonderful service.
- As a developer, I want to access Bit.ly, so that users can save, share, & discover links.
- As a developer, I want to access Buffer, so that users can have a smart way to share on social media.
- As a developer, I want to access Delicious.com, so that users can access cooking and recipe features.
- As a developer, I want to access Diigo, so that users can collect and organize using our app.
- As a developer, I want users to be able to ordering food (Example: Pizza Hut), so that users can automate this process.
- As a developer, I want to access Readability, so users can read comfortably if they prefer to.
- As a developer, I want to access Storify, so that users can create custom scenarios using social media.
- As a developer, I want to access Spotify, so that users can manage their media libraries.
- As a developer, I want to access Rdio, so that users can listen to music whenever they want.

- As a developer, I want to access Svpply, so that users can shop a wide variety of products.
- As a developer, I want to access Vimeo, so that users can have multiple options for viewing media.
- As a developer, I want to access ZooTool, so that users can bookmark and share content with ease.
- As a developer, I want to access Myspace, so that users can access music and video based social media.
- As a developer, I want to access Friendster, so that users can access this social media source.
- As a developer, I want to access Classmates.com, so that users can communicate with old classmates using our app.
- As a developer, I want to access Deviant art, so that users can manage their art portfolios using our application.
- As a developer, I want to access CrunchyRoll, so that users can be notified of new release dates.
- As a developer, I want to access Blogster, so that users can manage their blogs.
- As a developer, I want to access IGN, so that users can use this game database functionality.
- As a developer, I want to access GameFly, so that users can order games using our application.
- As a developer, I want to access IMDB, so that users can use this movie database functionality.
- As a developer, I want to access Live journal, so that users can manage their personal publishing.
- As a developer, I want to access Xanga, so that users can manage their own blogs.
- As a developer, I want to access Yelp, so that users can access local reviews and information.
- As a developer, I want to access Bebo, so that users can use this social media outlet.
- As a developer, I want to access Badoo, so that users can communicate with local people.
- As a developer, I want to access Stackoverflow, so that users can communicate on these developer forums.
- As a developer, I want to access XDA developers, so that developers can communicate on this medium.
- As a developer, I want to access the MSDN, so that developers can use this means of communication.
- As a developer, I want to access Digg, so that users can stay up to date on current events.
- As a developer, I want to access Orkut, so that users can social network and discuss using Google's engine.
- As a developer, I want to access PlayStation Network, so that we can accompany PlayStation users.

- As a developer, I want to access MSN, so that users can use access this functionality.
- As a developer, I want to access Bing, so that users can choose which search engine they wish to use.
- As a developer, I want to access AOL, so that we can address all user types.
- As a developer, I want to access Ask.com, so that users can have their questions answered.
- As a developer, I want to access Google's basic search, so that users can use this search engine.
- As a developer, I want to access CNN, so that users can receive updates about news.
- As a developer, I want to access FOX, so that users can receive updates about entertainment.
- As a developer, I want to access ABC, so that users can receive updates about news.
- As a developer, I want to access BBC, so that users can integrate with an international news source.
- As a developer, I want to access Wolfram Alpha, so that users can do complex math using our application.
- As a developer, I want to access Whitepages, so that users can quickly search this database.
- As a developer, I want to access Yellowpages, so that users can quickly search this database.
- As a developer, I want to access RememberTheMilk.com, so that users can update and receive reminder notifications.
- As a developer, I want to access iTunes, so that users can integrate their phone with this music application.
- As a developer, I want to access Apple accounts, so that users can integrate their Android device with Apple hardware.
- As a developer, I want to access PayPal, so that users can manage purchases using this application.
- As a developer, I want to access GoDaddy, so that users can manage website domains using our application.
- As a developer, I want to access 4shared, so that users can manage files on this service.
- As a developer, I want to access The Weather Channel, so that users can have more accurate weather on demand.
- As a developer, I want to access SourceForge, so that developers can manage their projects from their phone.
- As a developer, I want to access GitHub, so that developers can manage their code repositories.
- As a developer, I want to access Bitbucket, so that developers can manage their code repositories.
- As a developer, I want to access Walmart, so that users can be aware of updates and make purchases.
- As a developer, I want to access Target, so that users can utilize this department store's API.

- As a developer, I want to access Fedex, so that users can monitor shipping status.
- As a developer, I want to access UPS, so that users can monitor shipping status.
- As a developer, I want to access Dictionary.com, so that users can define words.
- As a developer, I want to access Urban dictionary, so that users can understand local colloquialisms.
- As a developer, I want to access Thesaurus.com, so that users can use this service.
- As a developer, I want to access Google Documents, so that users can manage their documents from our app.
- As a developer, I want to access Google Drive, so that users can manage their files on this service.
- As a developer, I want to access GroupOn, so that users can get deals from local businesses.
- As a developer, I want to access Living Social, so that users can receive deals on local services.
- As a developer, I want to access Urban Spoon, so that users can have information on cooking at hand.
- As a developer, I want to access Rhapsody, so that users can use this music player.
- As a developer, I want to access Imugr, so that users can manage their image files.
- As a developer, I want to access Heroku, so that users can manage their projects using this service.
- As a developer, I want to access Speedtest.net, so that users can check their current connection status with detail.
- As a developer, I want to access Photobucket, so that users can manage their image and video files.
- As a developer, I want to access Pandora, so that users can use this music service.
- As a developer, I want to access CNet, so that users can view technical reviews on products.
- As a developer, I want to access Battle.net, so that these users can manage their accounts.
- As a developer, I want to access Grooveshark, so that users can use this music service.
- As a developer, I want to access Monster, so that users can find the job that's right for them.
- As a developer, I want to access the NFL, so that users can receive updates from this feed.
- As a developer, I want to access Expedia, so that users can manage travel details.
- As a developer, I want to access Hotels.com, so that users can manage travel details.
- As a developer, I want to access Kayak, so that users can manage travel details.
- As a developer, I want to access Travelpedia, so that users can manage travel details.
- As a developer, I want to access Engadget, so that users can be up to date on technology news.
- As a developer, I want to access Slashdot, so that users can be up to date on technology news.
- As a developer, I want to access Fandango, so that users can manage movie tickets.

- As a developer, I want to access Ticketmaster, so that users can book and manage tickets.
- As a developer, I want to access StubHub, so that users can book and manage tickets.
- As a developer, I want to access Match.com, so that users can communicate with local individuals.
- As a developer, I want to access Rotten Tomatoes, so that users can use this movie review service.
- As a developer, I want to access Twilio, so that users can manage VoIP, Voice, and Text applications over the web.
- As a developer, I want to access Metro Lyrics, so that users can search for song lyrics.
- As a developer, I want to access Shazam, so that users can recognize songs using this service.
- As a developer, I want to access the NBA, so that users can receive updates from this service.
- As a developer, I want to access Dailymotion, so that users can upload, share, and embed their own videos.
- As a developer, I want to access Chacha, so that they can have their questions answered.
- As a developer, I want to access Best Buy, so that users can access this API service.
- As a developer, I want to access Quora, so that users can connect and share content using this service.
- As a developer, I want to access Voxeo, so that users can manage their cloud computing profiles.
- As a developer, I want to access "Let me Google that for you," so that users can share knowledge with peers.
- As a developer, I want to access About.com, so that users can solve their needs.
- As a developer, I want to access eHow, so they can use this service for information.
- As a developer, I want to access Mustang TRAK, so that users can manage their SMU job profiles.
- As a developer, I want to access to Access.smu.edu, so that users can manage their student accounts using our application.
- As a developer, I want to access Blackboard, so that users can manage their classes using this rules engine.

# Acceptance Test Plan

**Test A: The app starts up correctly.**

Test by asking users to start up the application by pressing the logo from the apps menu.

Process

Press the logo from the apps menu, see if it starts up.

Results

Able to start up (yes/no): _____
Comments:
_____
_____

[Passed/Failed]
_____

**Test B: Able to view all rules.**

Test by asking user to click My Rules from home page.

Process

Press the My Rules from home page button. Verify that all current rules show up (4 rules by default, plus any that may have been added).

Results

Able to view rules (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test C: Able to view details of a rule.**

Test by viewing the details of a rule.

Process

After following Test B, click on one of the rules. Verify that the name is correct at the top and that the rule's causes and effects appear. If there are multiple causes, an AND or OR should be between them.

Results

Able to bring up details of rule (yes/no):_____
Rule's name appears and is correct (yes/no):_____
"+" appears at bottom of both causes and effects (yes/no):_____
(Possible) AND or OR shows up between multiple causes (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test D: Arrive map location (GPS off).**

Test by having the user try to add the cause of arrival at destination and viewing the first screen.

Process

Before doing this test, make sure the GPS is off (in your phone settings). After getting past Test C, click the "+" below the causes list to create a new cause. Click "Arriving at a Location". The Google maps should show up with the last known location.

Results

"Arriving at a Location" is a cause listed (yes/no):_____
Last location known shows up on the screen (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test E: Pan the Google map.**

Test by having the user try to pan the map pulled up by Test D.

Process

As with most touch screens, attempt to hold and drag the screen to pan the view of the Google map that is pulled up from Test D.

Results

Able to pan map (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test F: Zoom in on the Google map.**

Test by having the user try to zoom in all the way on the map pulled up by Test D.

Process

As with most touch screens, put your two fingers together (preferably the thumb and index finger) on the screen. While pressing on the screen, slowly separate the two fingers to zoom in on the view of the Google map that is pulled up from Test D until you can no longer. Verify that app does not crash while doing so.

Results

Able to zoom in on map (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test G: Zoom out on the Google map.**

Test by having the user try to zoom out all the way on the map pulled up by Test D.

Process

As with most touch screens, put your two fingers apart (preferably the thumb and index finger) on the screen. While pressing on the screen, slowly close the two fingers to zoom out on the view of the Google map that is pulled up from Test D until you can no longer. Verify that app does not crash while doing so.

Results

Able to zoom out on map (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test H: My location button (GPS off).**

Test by having the user try to use the My Location button on Google maps (top right of screen).

Process

Before doing this test, make sure the GPS is off (in your phone settings). Press the My Location button in the top right of the screen. The screen should show a circle around your current location while also panning to your location.

Results

Circle shows up around a blue dot that includes your current location (yes/no):_____
Dot shows up in the middle of the screen (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test I: Arrive map location (GPS on).**

Test by having the user try to add the cause of arrival at destination and viewing the first screen.

Process

Before doing this test, make sure the GPS is on (in your phone settings). After getting past Test C, click the "+" below the causes list to create a new cause. Click "Arriving at a Location". The Google maps should show up with the last known location. This will be a lot more accurate than with the GPS off.

Results

Last location known shows up on the screen (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test J: My location button (GPS on).**

Test by having the user try to use the My Location button on Google maps (top right of screen).

Process

Before doing this test, make sure the GPS is on (in your phone settings). Press the My Location button in the top right of the screen. The screen should show a blue dot indicating your current location (if you are zoomed in a small circle will appear) while also panning to your location.

Results

Blue dot shows your current location (yes/no):_____
Dot shows up in the middle of the screen (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test K: Tap to add pin to Google custom map.**

Test by having the user tap the screen on the Google map to add a pin.

Process

While inside of Arriving at Location, tap on the screen to add a pin. A dialog should appear that asks for a name of the location. Leave the name blank and click cancel. The screen should clear and no pin visible anymore.

Results

Pin appears on screen after clicking (yes/no):_____
Pin disappears on screen after clicking Cancel (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test L: Tap to add multiple pins to Google custom map.**

Test by having the user tap the screen on the Google map to add a pin.

Process

While inside of Arriving at Location, tap on the screen to add a pin. A dialog should appear that asks for a name of the location. Leave the name blank and click OK. The pin should still be visible. Repeat this process a couple times to keep adding more pins. Multiple pins should be visible. Now add a new pin and instead of clicking OK, click cancel. All pins should now be gone.

Results

Pin appears on screen after clicking (yes/no):_____
Multiple pins appear after creating multiple (yes/no):_____
All pins disappear on screen after clicking Cancel (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test M: Tap to add pin to Google custom map (blank name).**

Test by having the user tap the screen on the Google map to add a pin.

Process

While inside of Arriving at Location, tap on the screen to add a pin. A dialog should appear that asks for a name of the location. Leave the name blank and click OK. Click and hold inside of the square to accept the location. Verify that this rule now appears in the cause list. Verify that the rule works by arriving at this location specified.

Results

Pin appears on screen after clicking (yes/no):_____
Able to add pin with blank name (yes/no):_____
Holding accepts the rule and stops the editing of the cause (yes/no):_____
New cause appears at the bottom of the cause list (yes/no):_____
Cause is triggered when arriving at the location (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test N: Tap to add pin to Google custom map (actual name).**

Test by having the user tap the screen on the Google map to add a pin.

Process

While inside of Arriving at Location, tap on the screen to add a pin. A dialog should appear that asks for a name of the location. Fill in the name and click OK. The screen should show the pin. Click and hold inside of the square to accept the location. Verify that this rule now appears in the cause list with the name visible after "Location: " in the cause. Verify that the cause works by arriving at the specified location.

Results

Pin appears on screen after clicking (yes/no):_____
Able to add pin with blank name (yes/no):_____
Holding accepts the rule and stops the editing of the cause (yes/no):_____
New cause appears at the bottom of the cause list with name (yes/no):_____
New cause triggers the rule effects (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test O: Tap to add pin to Google custom map (blank name).**

Test by having the user try to add the cause of departing a location and following the steps to create the cause.

Process

After getting past Test C, click the "+" below the causes list to create a new cause. Click "Departing a Location". The Google maps should show up with the last known location. Tap on the screen to add a pin. Leave the name blank and click OK. Click and hold inside the square to finalize the location. The new cause should show up at the bottom of the cause list. Finally, verify that leaving the designated location triggers the rule effects.

Results

"Departing at a Location" is a cause listed (yes/no):_____
Last location known shows up on the screen (yes/no):_____
Able to add blank pin (yes/no):_____
Holding accepts the rule and stops the editing of the cause (yes/no):_____
New cause appears at the bottom of the cause list (yes/no):_____
New cause triggers the rule effects (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test P: Tap to add pin to Google custom map (actual name).**

Test by having the user try to add the cause of departing a location and following the steps to create the cause.

Process

While inside of departing a location, tap on the screen to add a pin. Fill in the name and click OK. Click and hold inside the square to finalize the location. The new cause should show up at the bottom of the cause list. Verify that this rule now appears in the cause list with the name visible after "Location: " in the cause. Finally, verify that leaving the designated location triggers the rule effects.

Results

Able to add actual name pin (yes/no):_____
Holding accepts the rule and stops the editing of the cause (yes/no):_____
New cause appears at the bottom of the cause list with name (yes/no):_____
New cause triggers the rule effects (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test Q: Adding Phone Call cause.**

Test by having the user try to add the cause of phone call.

Process

After getting past Test C, click the "+" below the causes list to create a new cause. Click "Phone Call". Your contact list should be displayed. Click on one of the contacts (if you have no contacts, add one and restart test). The new cause should now be displayed at the bottom of the list with the correct name. Verify that the rule works by having that contact call the phone.

Results

"Phone Call" is a cause listed (yes/no):_____
Contact list displayed (yes/no):_____
New cause added to list with correct contact name (yes/no):_____
Rule effects triggered by new cause (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test R: Adding Text message cause.**

Test by having the user try to add the cause of text message.

Process

After getting past Test C, click the "+" below the causes list to create a new cause. Click "Text Message". Your contact list should be displayed. Click on one of the contacts (if you have no contacts, add one and restart test). The new cause should now be displayed at the bottom of the list with the correct name. Verify that the rule works by having that contact text the phone.

Results

"Text Message" is a cause listed (yes/no):_____
Contact list displayed (yes/no):_____
New cause added to list with correct contact name (yes/no):_____
Rule effects triggered by new cause (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test S: Adding Time cause.**

Test by having the user try to add the cause of time.

Process

After getting past Test C, click the "+" below the causes list to create a new cause. Click "Time". Input a time and click OK. The new cause should now be displayed at the bottom of the list with the correct time. Verify that the rule works by waiting for that time.

Results

"Time" is a cause listed (yes/no):_____
Time input dialog displayed (yes/no):_____
New cause added to list with correct time (military time) (yes/no):_____
Rule effects triggered by new cause (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test T: Adding Time cause (cancel).**

Test by having the user try to add the cause of time.

Process

After getting past Test C, click the “+” below the causes list to create a new cause. Click “Time”. Input a time and click cancel. No new cause should be added.

Results

Time input dialog displayed (yes/no):_____
No new cause added (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test U: Adding Wi-Fi SSID cause.**

Test by having the user try to add the cause of Wi-Fi SSID.

Process

After getting past Test C, click the "+" below the causes list to create a new cause. Click "Wi-Fi SSID". A create dialog should appear asking for text input. Input a name for the SSID (wireless network name) and hit Submit. Verify that the new cause appears at the bottom of the cause list and that the rule effects are triggered by entering the range of the SSID.

Results

"Wi-Fi SSID" is a cause listed (yes/no):_____
Create dialog appears (yes/no):_____
Able to create new cause with correct SSID name (yes/no):_____
Rule activates correctly (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test V: Adding Wi-Fi SSID cause (blank).**

Test by having the user try to add the cause of Wi-Fi SSID.

Process

After getting past Test C, click the "+" below the causes list to create a new cause. Click "Wi-Fi SSID". A create dialog should appear asking for text input. Leave the name blank and hit Submit. Verify that the new cause appears at the bottom of the cause list.

Results

"Wi-Fi SSID" is a cause listed (yes/no):_____
Create dialog appears (yes/no):_____
Able to create new cause with blank SSID name (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test W: Adding Wi-Fi Status cause (on).**

Test by having the user try to add the cause of Wi-Fi Status.

Process

After getting past Test C, click the "+" below the causes list to create a new cause. Click "Wi-Fi Status". A create dialog should appear with a slider (default in off position). Slide the slider into the one position and click Submit. Verify that the new cause appears at the bottom of the cause list and that the rule effects are triggered by turning on the Wi-Fi.

Results

"Wi-Fi Status" is a cause listed (yes/no):_____
Create dialog appears with slider (yes/no):_____
Able to create new cause with correct status (yes/no):_____
Rule activates correctly (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test X: Adding Wi-Fi Status cause (off).**

Test by having the user try to add the cause of Wi-Fi Status.

Process

After getting past Test C, click the "+" below the causes list to create a new cause. Click "Wi-Fi Status". A create dialog should appear with a slider (default in off position). Leave it in the off position and click Submit. Verify that the new cause appears at the bottom of the cause list and that the rule effects are triggered by turning off the Wi-Fi.

Results

Able to create new cause with correct status (yes/no):_____
Rule activates correctly (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test Y: Adding Notification effect (blank).**

Test by having the user try to add the effect of notification.

Process

After getting past Test C, click the "+" below the effects list to create a new effect. Click "Notification". A create dialog should appear waiting for input of text for Title and Subtext. Leave both blank and click Submit. Verify that the new effect appears at the bottom of the effect list and that the rule effect is triggered by activating the rule (making the causes true, whatever they may be).

Results

"Notification" is an effect listed (yes/no):_____
Create dialog appears with empty text slots (yes/no):_____
Able to create new effect with correct text (blank) (yes/no):_____
Notification effect added to list (yes/no):_____
Effect is displayed correctly (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test Z: Adding Notification effect (Title only).**

Test by having the user try to add the effect of notification.

Process

While inside creating a notification (Test Y), fill in the Title box but leave Subtext blank and click Submit. Verify that the new effect appears at the bottom of the effect list and that the rule effect is triggered by activating the rule (making the causes true, whatever they may be).

Results

Able to create new effect with correct text (yes/no):_____
Notification effect added to list (yes/no):_____
Effect is displayed correctly (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AA: Adding Notification effect (Message only).**

Test by having the user try to add the effect of notification.

Process

While inside creating a notification (Test Y), fill in the Subtext box but leave Title blank and click Submit. Verify that the new effect appears at the bottom of the effect list and that the rule effect is triggered by activating the rule (making the causes true, whatever they may be).

Results

Able to create new effect with correct text (yes/no):_____
Notification effect added to list (yes/no):_____
Effect is displayed correctly (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AB: Adding Notification effect (Full notification).**

Test by having the user try to add the effect of notification.

Process

While inside creating a notification (Test Y), fill in the Subtext box and the Title box and click Submit. Verify that the new effect appears at the bottom of the effect list and that the rule effect is triggered by activating the rule (making the causes true, whatever they may be).

Results

Able to create new effect with correct text (yes/no):_____
Notification effect added to list (yes/no):_____
Effect is displayed correctly (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AC: Adding Play Sound effect.**

Test by having the user try to add the effect of sound.

Process

After getting past Test C, click the "+" below the effects list to create a new effect. Click "Play Sound". A "Select a File to Play" dialog should appear and tell you to pick a way to choose your sound. Pick one of your choosing and use it's usability to select a sound to play. Verify that the new effect appears at the bottom of the effect list and that the rule effect is triggered by activating the rule (making the causes true, whatever they may be).

Results

"Play Sound" is an effect listed (yes/no):_____
"Select a File to Play" dialog appears (yes/no):_____
Able to create new effect with correct sound file displayed (yes/no):_____
Correct sound is played (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AD: Adding Ring Mode effect (Normal).**

Test by having the user try to add the effect of ring mode.

Process

After getting past Test C, click the "+" below the effects list to create a new effect. Click "Ring Mode". A "Ring Mode" dialog should appear with 3 options. Click on "Normal". Verify that the new effect appears at the bottom of the effect list and that the rule effect is triggered by activating the rule (making the causes true, whatever they may be).

Results

"Ring Mode" is an effect listed (yes/no):_____
"Ring Mode" dialog appears with 3 options (yes/no):_____
Able to create new effect with "Normal" for ring mode (yes/no):_____
Ring mode effect added to list (yes/no):_____
Effect is displayed correctly (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AE: Adding Ring Mode effect (Vibrate).**

Test by having the user try to add the effect of ring mode.

Process

Starting with dialog box from Test AD, click on “Vibrate”. Verify that the new effect appears at the bottom of the effect list and that the rule effect is triggered by activating the rule (making the causes true, whatever they may be).

Results

Able to create new effect with “Vibrate” for ring mode (yes/no):_____
Ring mode effect added to list (yes/no):_____
Effect is displayed correctly (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AF: Adding Ring Mode effect (Silent).**

Test by having the user try to add the effect of ring mode.

Process

Starting with dialog box from Test AD, click on “Silent”. Verify that the new effect appears at the bottom of the effect list and that the rule effect is triggered by activating the rule (making the causes true, whatever they may be).

Results

Able to create new effect with “Silent” for ring mode (yes/no):_____
Ring mode effect added to list (yes/no):_____
Effect is displayed correctly (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AG: Adding Toast effect (blank).**

Test by having the user try to add the effect of Toast.

Process

After getting past Test C, click the "+" below the effects list to create a new effect. Click "Toast". A "New Toast" dialog should appear with an empty Message text input. Leave the message blank and click Submit. Verify that the new effect appears at the bottom of the effect list and that the rule effect is triggered by activating the rule (making the causes true, whatever they may be).

Results

"Toast" is an effect listed (yes/no):_____
Able to create new effect with blank message (yes/no):_____
Toast effect added to list (yes/no):_____
Effect is displayed correctly (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AH: Adding Toast effect.**

Test by having the user try to add the effect of Toast.

Process

From the new Toast dialog, enter a message and click Submit. Verify that the new effect appears at the bottom of the effect list and that the rule effect is triggered by activating the rule (making the causes true, whatever they may be).

Results

Able to create new effect with correct message (yes/no):_____
Toast effect added to list (yes/no):_____
Effect is displayed correctly (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AI: Adding Vibrate effect.**

Test by having the user try to add the effect of Vibrate.

Process

After getting past Test C, click the "+" below the effects list to create a new effect. Click "Vibrate". Verify that the new effect appears at the bottom of the effect list and that the rule effect is triggered by activating the rule (making the causes true, whatever they may be).

Results

"Vibrate" is an effect listed (yes/no):_____
Vibrate is added to effect list (yes/no):_____
Effect is displayed correctly (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AJ: Editing Arrive cause.**

Test by having the user try to edit the Arrive cause.

Process

After adding an arrive cause (see earlier tests), click the cause. Verify that the map appears and allows you to set a new location. Input a new location (following the test for adding an arrive cause). The updated information should now appear in the cause list. Finally, verify that the rule now activates when this new cause is true.

Results

Map appears when you click on arrival cause (yes/no):_____
Able to make the new cause (yes/no):_____
Updated cause is displayed correctly (yes/no):_____
New cause triggers rule effects (yes/no):_____
Old cause does not trigger rule effects (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AK: Editing Depart cause.**

Test by having the user try to edit the Depart cause.

Process

After adding a depart cause (see earlier tests), click the cause. Verify that the map appears and allows you to set a new location. Input a new location (following the test for adding an arrive cause). The updated information should now appear in the cause list. Finally, verify that the rule now activates when this new cause is true.

Results

Map appears when you click on depart cause (yes/no):_____
Able to make the new cause (yes/no):_____
Updated cause is displayed correctly (yes/no):_____
New cause triggers rule effects (yes/no):_____
Old cause does not trigger rule effects (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AL: Editing Phone Call cause.**

Test by having the user try to edit the Phone Call cause.

Process

After adding a phone call cause (see earlier tests), click the cause. Verify that the contact list appears and allows you to set a new contact. Input a new contact (following the test for adding a phone call cause). The updated information should now appear in the cause list. Finally, verify that the rule now activates when this new cause is true.

Results

Contact list appears when you click on phone call cause (yes/no):_____
Able to make the new cause (yes/no):_____
Updated cause is displayed correctly (yes/no):_____
Old cause does not trigger rule effects (yes/no):_____
New cause triggers rule effects (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AM: Editing Text Message cause.**

Test by having the user try to edit the Text Message cause.

Process

After adding a text message cause (see earlier tests), click the cause. Verify that the contact list appears and allows you to set a new contact. Input a new contact (following the test for adding a text message cause). The updated information should now appear in the cause list. Finally, verify that the rule now activates when this new cause is true.

Results

Contact list appears when you click on text message cause (yes/no):_____
Able to make the new cause (yes/no):_____
Updated cause is displayed correctly (yes/no):_____
New cause triggers rule effects (yes/no):_____
Old cause does not trigger rule effects (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AN: Editing Time cause.**

Test by having the user try to edit the Time cause.

Process

After adding a time cause (see earlier tests), click the cause. Verify that the same dialog appears asking for a time. Input a new time and click OK. The updated information should now appear in the cause list. Finally, verify that the rule now activates when this new cause is true.

Results

Time dialog appears when you click on time cause (yes/no):_____
Updated cause is displayed correctly (yes/no):_____
New cause triggers rule effects (yes/no):_____
Old cause does not trigger rule effects (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AO: Editing Wi-Fi SSID cause.**

Test by having the user try to edit the Wi-Fi SSID cause.

Process

After adding a wifi ssid cause (see earlier tests), click the cause. Verify that the same dialog appears asking for a wifi ssid. Input a new ssid and click Submit. The updated information should now appear in the cause list. Finally, verify that the rule now activates when this new cause is true.

Results

Wi-Fi SSID dialog appears when you click on Wi-Fi SSID cause (yes/no):_____
Updated cause is displayed correctly (yes/no):_____
New cause triggers rule effects (yes/no):_____
Old cause does not trigger rule effects (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AP: Editing Wi-Fi Status cause.**

Test by having the user try to edit the Wi-Fi Status cause.

Process

After adding a wifi status cause (see earlier tests), click the cause. Verify that the same dialog appears with the slider. Switch the slider to the opposite of what is set and click Submit. The updated information should now appear in the cause list. Finally, verify that the rule now activates when this new cause is true.

Results

Wi-Fi Status dialog appears when you click on Wi-Fi Status cause (yes/no):_____
Updated cause is displayed correctly (yes/no):_____
New cause triggers rule effects (yes/no):_____
Old cause does not trigger rule effects (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AQ: Editing Notification effect.**

Test by having the user try to edit the Notification effect.

Process

After adding a notification effect (see earlier tests), click the effect. Verify that the same dialog appears asking for text for the notification. Input new text and click Submit. The updated information should now appear in the effect list. Finally, verify that when the rule is activated the new effect appears.

Results

Notification dialog appears when you click on Notification effect (yes/no):_____
Updated effect is displayed correctly (yes/no):_____
New effect is displayed (yes/no):_____
Old effect is not displayed (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AR: Editing Sound effect.**

Test by having the user try to edit the Sound effect.

Process

After adding a sound effect (see earlier tests), click the effect. Verify that the same dialog appears asking for which program to use to select the new sound. Use whichever input and select a new sound. The updated information should now appear in the effect list. Finally, verify that when the rule is activated the new effect appears.

Results

Sound dialog appears when you click on Sound effect (yes/no):_____
Updated effect is displayed correctly (yes/no):_____
New effect is played (yes/no):_____
Old effect is not played (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AS: Editing Ring Mode effect.**

Test by having the user try to edit the Ring Mode effect.

Process

After adding a ring mode effect (see earlier tests), click the effect. Verify that the same dialog appears asking for which ring mode to use. Select a different ring mode. The updated information should now appear in the effect list. Finally, verify that when the rule is activated the new effect appears.

Results

Ring Mode dialog appears when you click on Ring Mode effect (yes/no):_____
Updated effect is displayed correctly (yes/no):_____
New effect is used (yes/no):_____
Old effect is not used (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AT: Editing Toast effect.**

Test by having the user try to edit the Toast effect.

Process

After adding a toast effect (see earlier tests), click the effect. Verify that the same dialog appears asking for text for the toast. Input new text and click Submit. The updated information should now appear in the effect list. Finally, verify that when the rule is activated the new effect appears.

Results

Toast dialog appears when you click on Toast effect (yes/no):_____
Updated effect is displayed correctly (yes/no):_____
New effect is displayed (yes/no):_____
Old effect is not displayed (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AU: Editing Vibrate effect.**

Test by having the user try to edit the Notification effect.

Process

After adding a vibrate effect (see earlier tests), click the effect. Verify that no dialog appears.

Results

No dialog appears (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AV: Able to delete causes.**

Test by having user delete a cause.

Process

After adding a cause, long click on a cause to verify that delete dialog appears. Then test to see that if you click no, nothing happens. Also check to see if you click yes it disappears and the rule no longer is affected by that cause.

Results

Delete dialog appears for causes (yes/no):_____
Nothing happens if you click no (yes/no):_____
Deleted cause if you click yes (for cause) (yes/no):_____
Rule activates correctly after deletion of cause (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AW: Able to delete effects.**

Test by having user delete an effect.

Process

After adding an effect, long click on an effect to verify that delete dialog appears. Then test to see that if you click no, nothing happens. Also check to see if you click yes it disappears and the rule no longer is affected by that effect.

Results

Delete dialog appears for effects (yes/no):_____
Nothing happens if you click no (yes/no):_____
Deleted effect if you click yes (for effect) (yes/no):_____
Rule activates correctly after deletion of effect (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AX: Able to delete rules.**

Test by having user delete a rule.

Process

While looking at the rule list, long click on a rule to verify that delete dialog appears. Then test to see that if you click no, nothing happens. Also check to see if you click yes it disappears and the rule can no longer be activated.

Results

Delete dialog appears for rules (yes/no):_____
Nothing happens if you click no (yes/no):_____
Deleted rule if you click yes (yes/no):_____
Rule no longer appears and does not activate (yes/no):_____
Comments:
_____
_____

[Passed/Failed]
_____

**Test AY: Up button functionality works.**

Test by using integrated up button at top of screen while inside of app.

Process

While inside of an activity (any screen other than main menu), try using the integrated up button to return to previous major screen. Should eventually lead back to main screen without going into every specific screen you saw (e.g. edit rule -> rule list, rule list -> main menu, new rule -> rule list).

Note: the back button on phones is more of an undo, so using that is not applicable for this test, but can still be used.

Results

Up button is visible on all screens but Main Menu (yes/no):_____
Up button works as intended (listed in process) (yes/no):_____

[Passed/Failed]
_____

**Test AZ: Able to change name of rule.**

Test by changing the name of a rule and verifying it has updated.

Process

While in the editing page of a rule, click on the rule name and edit it (does not matter what you name it). Then use the back button (from Test AY) to go back to rule list. New name should appear. Click on the rule again to make sure the correct name appears in the edit rule page again.

Results

Able to edit name of rule (able to change text) (yes/no):_____
New name appears in rule list (yes/no):_____
New name appears again in edit rule page (yes/no):_____

[Passed/Failed]
_____

**Test AZ: Able to change rule active.**

Test by changing the rule active icon in edit rule page.

Process

While in edit rule page, change the slider at the top of the rule (either slide or click) to off (default is on usually, change to on if already off). Verify that the rule works now (or not). Then repeat after changing it back to the original state.

Results

Slider is visible (yes/no):_____
Able to change slider using slide (yes/no):_____
Able to change slider using click (yes/no):_____
Rule does not activate while in off position (yes/no):_____
Rule does activate while in on position (yes/no):_____

[Passed/Failed]
_____

**Test BA: Able to add rule.**

Test by adding a new rule.

Process

From main menu, click “New Rule” to start making a new rule. A blank rule should show up with “Untitled” as the name. Add a cause and/or an effect. This makes the rule permanent. Use the up button to go back to the rules list. Your new rule should appear there.

Results

Blank rule appears (yes/no):_____
New rule after instantiated appears (yes/no):_____

[Passed/Failed]
_____

**Test BB: Able to add rule (blank).**

Test by adding a new rule.

Process

From main menu, click “New Rule” to start making a new rule. A blank rule should show up with “Untitled” as the name. Do not add any causes or effects. Use the up button to go back to the rules list. The new rule should not appear in the list.

Results

New rule does not appear (yes/no):_____

[Passed/Failed]

_____

**Test BC: Test the sharing functions (NFC).**

Test by trying to share a rule through NFC.

Process

Using one of the rules on a phone and use the share service (found on the main menu) to try to share that rule. For NFC sharing, you click on the rule to share and click NFC. Then click the rule and touch the backs of the phones together and press on the screen of the sending phone to send (does not matter what screen the receiving phone is on). Verify that the rule shows up in the rule list and activates when appropriate.

Results

Able to share rule via NFC (yes/no): _____
Shared rule appears in rule list (yes/no):_____
Rule works as originally intended (yes/no):_____

[Passed/Failed]
_____

**Test BD: Test the sharing functions (Email).**

Test by trying to share a rule through Email.

Process

Using one of the rules on a phone and use the share service (found on the main menu) to try to share that rule. For email, click on the rule that is to be sent and select email and insert the email address. An email should be sent to an email that is on the phone that is receiving the rule. Simply open this email and follow the instructions from there. Verify that the rule shows up in the rule list and activates when appropriate.

Results

Able to share rule via Email (yes/no): _____
Shared rule appears in rule list (yes/no):_____
Rule works as originally intended (yes/no):_____

[Passed/Failed]
_____

**Test BE: Check extensibility to 3rd party programs.**

Test by making a rule that uses a 3rd party program.

Process

Before setting up the rule, make sure that the appropriate application is installed on your device. Then, set up one of the rules with a 3rd party program that you can easily activate, then activate it. Finally, go onto the 3rd party program through different means to check that it functioned correctly.

Results

Able to add rule (yes/no): _____
Rule is activated when appropriate (yes/no): _____
Rule functions correctly (yes/no): _____

[Passed/Failed]
_____

**Test BF: Make sure the Boolean algebra is added correctly.**

Test by adding a 2nd+ cause on any rule.

Process

Go to any rule that has at least 1 cause. Add any cause. An OR should be added between the causes. Then delete a cause and an OR/AND that was between it and the next cause (or the last one if it was the last cause) should disappear.

Results

OR is added between causes (yes/no):_____
OR/AND is deleted when cause is deleted (yes/no):_____

[Passed/Failed]
_____

**Test BG: Make sure the Boolean algebra works correctly (AND).**

Test by making a rule with Boolean algebra present and trying to activate.

Process

Set up one of the rules to easily activate that has multiple causes. Click on OR box to change to AND. Make the causes true to make the rule activate. Then test with only one of the causes true while the other false, then flip.

Note: with the AND rule, make sure both causes can be active at the same time (e.g. time and location, time and wifi are good examples).

Results

Able to change OR to AND between causes (yes/no):_____
Rule is activated when both causes true (yes/no): _____
Rule is not activated when first cause true and second cause false (yes/no):_____
Rule is not activated when second cause true and first cause false (yes/no):_____

[Passed/Failed]
_____

**Test BH: Make sure the Boolean algebra works correctly (OR).**

Test by making a rule with Boolean algebra present and trying to activate.

Process

Set up one of the rules to easily activate that has multiple causes. Click on AND box to change to OR (if necessary, change an OR to AND first). Make one of the causes true to make the rule activate. Then make the other cause true while making the first one false and make sure it activates.

Results

Able to change AND to OR between causes (yes/no):_____
Rule is activated when first cause is true and second one false (yes/no):_____
Rule is activated when second cause is true and first one false (yes/no):_____

[Passed/Failed]
_____

**Test BI: Make sure you cannot add duplicate causes.**

Test by trying to add a duplicate cause to a rule.

Process

Go into the edit page of any rule and try to add a duplicate of any of the causes. The new cause should not be added to the rule.

Results

New duplicate cause is not added to rule (yes/no):_____
Toast is shown notifying the user that it has found a duplicate cause (yes/no):_____

[Passed/Failed]
_____

**Test BJ: Make sure you cannot add duplicate effects.**

Test by trying to add a duplicate effect to a rule.

Process

Go into the edit page of any rule and try to add a duplicate of any of the effects. The new effect should not be added to the rule.

Results

New duplicate effect is not added to rule (yes/no):_____
Toast is shown notifying the user that it has found a duplicate effect (yes/no):_____

[Passed/Failed]
_____

**Test BK: Bring up menu in My Rules (rule list).**

Test by pressing the Menu key while inside of My Rules.

Process

Go into the My Rules page and press Menu. The Menu should pop up with Settings.

Results

Menu shows up with only Settings appearing (yes/no):_____

[Passed/Failed]

_____

**Test BL: Bring up menu in Edit Rule.**

Test by pressing the Menu key while inside of Edit Rule.

Process

Go into the Edit Rule page and press Menu. The Menu should pop up with Settings and Help.

Results

Menu shows up with Settings and Help appearing (yes/no):_____

[Passed/Failed]

_____

**Test BM: Bring up menu in Help.**

Test by pressing the Menu key while inside of Help page.

Process

Go into the Help page and press Menu. The Menu should pop up with Settings.

Results

Menu shows up with only Settings appearing (yes/no):_____

[Passed/Failed]

_____

**Test BN: Bring up menu in Settings.**

Test by pressing the Menu key while inside of Settings page.

Process

Go into the Settings page and press Menu. The Menu should pop up with Help.

Results

Menu shows up with only Help appearing (yes/no):_____

[Passed/Failed]

_____

**Test BO: Test Settings tabs.**

Test by pressing the tabs in the Settings page to make sure all the tabs work.

Process

Go into the Settings page and press the General tab, Accounts tab, Security tab, and About tab and make sure they work.

Results

General tab works (yes/no):_____
Accounts tab works (yes/no):_____
Security tab works (yes/no):_____
About tab works (yes/no):_____

[Passed/Failed]
_____

**Test BP: Test application service.**

Test by pressing backing out of the app and activate a rule.

Process

Make sure there is a relatively simple rule to activate and then exit out of the app by constantly pressing the android back button (or Home button). Then activate rule to make sure it works.

Results

Rule activates while app is closed (yes/no):_____

[Passed/Failed]

_____