Android v. Asthma

or, how I learned to stop worrying and love the handheld (sort of)

Brent Grossman
Advisor: Jason Leigh
Secondary: Robert Kenyon
December 2012

Table of Contents

Introduction

Although a manageable condition with medical intervention, asthma is a chronic respiratory disease known for its potentially life-threatening attacks that make breathing difficult.[i] Recently, cell phone technology, along with healthcare workforce shortages and the rising prevalence of chronic diseases, have converged and led to a number of mobile health interventions to manage chronic disease, including asthma.[1, 2]

Dr. Gisele S. Mosnaim and her colleagues have been working on asthma applications that specifically prioritize youth who live in the inner city and may experience higher incidences of asthma, as well as morbidity and mortality related to asthma.[1, 3] During the first ADEPT study, the intervention consisted of MP3 players with celebrity-delivered asthma adherence messages between music tracks. Those randomized to the intervention demonstrated significantly increased asthma knowledge compared to controls.[1]

The goal of this project is to develop an application that is avatar-based and will be a catalyst for and measure behavior change (i.e., improved asthma medication adherence compared to controls). An avatar as peer and coach has been demonstrated to serve as an effective motivator of health behavior changes.[4] Additionally, gaming, or apps that encourage peer collaboration and competition, have also been demonstrably effective.[5] Building on these evidence-based trajectories, this project aims to address the current gap in asthma management applications for youth.[1]

The members of the project team are:

- Joshua Albers: doser case design
- Paula Jo Belice: Clinical study design
- Jon Chambers: doser case design and app graphics
- Steve Conner: doser electronics design and implementation
- Brent Grossman: Android app implementation
- Robert Kenyon, PhD: Project design and oversight
- Jason Leigh, PhD: Project design and oversight
- Giselle Mosnaim, MD: Project and clinical study design and oversight
- Bala Rajan: Android app and server software implementation

Note that this project began as a class project for CS 594, Human Augmentics, in the Spring Semester of 2012 at UIC.

Project System Design and Structure

The project system consists of 3 primary components:

- The case and electronics attached to the inhalers used by asthmatic study participants (henceforth referred to as “doser”)
    - Two for each participant—one for the inhaler that’s used on a daily schedule (i.e., the inhaled corticosteroid, or “ICS” inhaler), one for the emergency inhaler (i.e., short-acting beta agonist or “SABA” inhaler) used during asthmatic distress situations
    - Records time and date of each inhaler use (the former henceforth referred to as “timedatestamp”, the latter “puff”) and sends that information to the Android app
- The Android app running on the cellular phone given to each participant
    - Main screen
        - Avatar on a basketball court. Avatar moves from the middle of the court towards the basket each time the participant uses their ICS inhaler
        - Scoreboard showing defined inhaler schedule adherence levels achieved for the current week, last week, and for the duration of the study
    - Trading card screen
        - One stats table showing the participant’s inhaler use counts for the current day, current week, and the duration of the study, alongside the numbers for the top performing participant in each category
        - A second stats table showing the number of weeks the participant has achieved each adherence level (as described above for the scoreboard section of the main screen)
    - Avatar customization screen—facial hair, glasses, and head hair can be added to the avatar here
        - Icons for selecting one of the three categories is at bottom of screen
        - Upon selecting a category, available items in that category appear in a scrolling list on the left side of the screen; touching on any item adds it to the avatar and removes any previously selected item in that category
    - Survey screen: used by clinical project personnel to set parameters used by app and transmitted to server for tracking
    - Sync mode screen: used by project personnel to set date and time of doser electronics
    - Background service
        - Receives, via Bluetooth, timedatestamp from inhaler component and stores that in file on phone
        - Transmits, over Internet, timedatestamp and other information to server each time timedatestamp is received
        - Causes main screen of app to come to foreground on phone and animation to be initiated when timedatestamp from ICS inhaler received
- The server to which data from the app will be sent for analysis and long-term storage

- From each participant's Android app, receives and stores time and date information sent each time the app receives this information from a doser, for analysis, and possible intervention with participant, by clinical members of project
- From among all participants, calculates top performing participant, based on adherence to expected inhaler usage, for current day, week, and duration of study, and transmits that back to the Android app on each participant's phone
- Most of the server programming was performed by Bala Rajan.

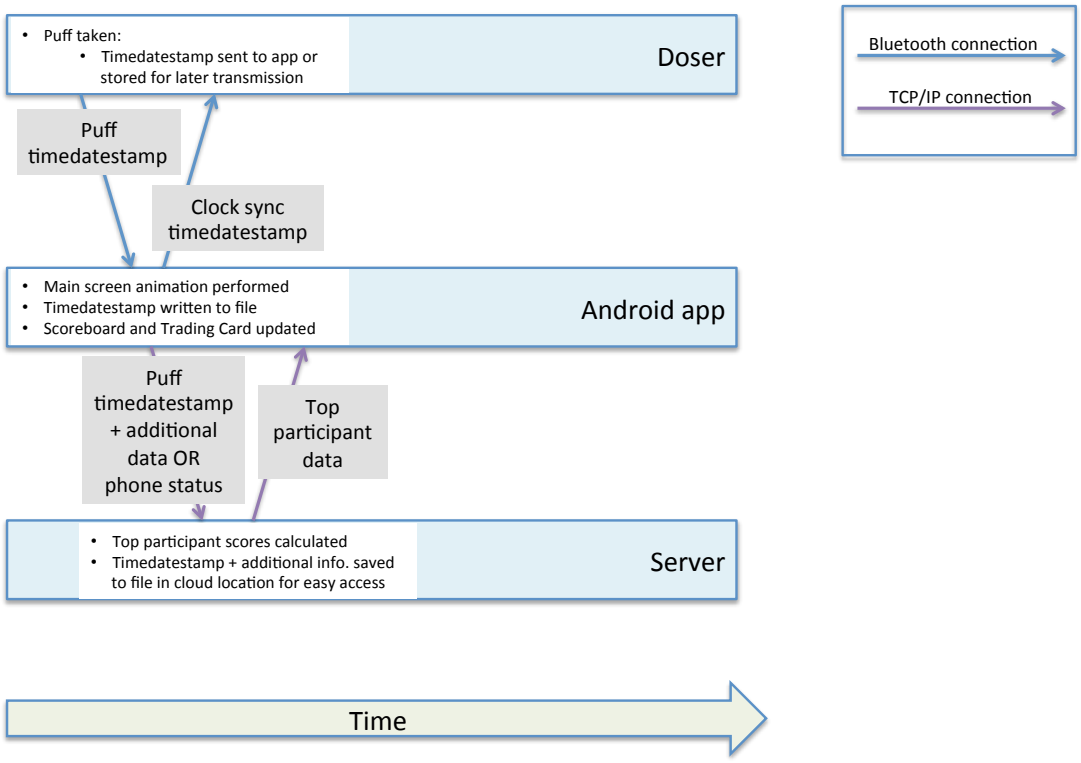


Figure 1: Overall Project System Interaction Diagram

App Implementation

The app was developed in the Eclipse integrated development environment with the Android Development Toolkit (ADT) and other Android Plug-ins, as well as the EGit plugin allowing for connection to and updating of Git repositories, installed in it, as well as the Android Software Development Kit (SDK) for Android 2.3. The app has been tested primarily on Samsung Galaxy Player 4.0 devices, running Android 2.3. All code is in a GitHub repository (https://github.com/hgross4/Asthma-Intervention).

The app consists of the following classes:

Accessory extends the Android ImageButton widget class. It's used to load and allow for the selection of accessories in the avatar customization screen. It was written entirely when this app was still part of a class project for CS 594.

AcraCrashReport extends the Android Application class, the latter being a base class that provides a means to maintain global application state information and other variables and methods that may be needed across classes. This class serves that function in this app with methods and variables that are used to keep track of whether the app is in sync mode, and also by providing a static instance of itself that can be used as an Android Context instance throughout the app in classes that don't have their own. This class is also responsible for sending crash report information to a Google Drive spreadsheet when the app crashes on a device that's connected to the Internet. This is accomplished through a library added to the project, acra-4.2.3.jar.[6]

AsthmaAppActivity extends the Android Activity class. It's associated with the avatarpage xml layout file, i.e., is the code behind the main activity (screen) of the app. It instantiates an AvatarView object which contains the avatar, basketball, and ball-shadow bitmaps. Through this object it initiates the animation of these bitmaps. This class receives an Android Intent from the BluetoothService class when it receives a timedatestamp from a doser, at which point it initiates the animation of the avatar, ball, etc. as well as code to determine the levels achieved and to display in the scoreboard area at the bottom of the screen. It also receives Android alarms that trigger it to display messages in dialog boxes, designed to encourage or remind the participant about his inhaler usage. This class also queries the AIRNow web site to determine and display air quality information at the top of the screen.[7] The air quality functionality was mostly developed as part of the class project. Touching on the dedicated Android menu button while viewing the main screen brings up a menu from which one can navigate to the app's survey or sync mode screens. Bala Rajan handled the drawing of the scorecard heading images.

AvatarCreator extends the Android Activity class. It's associated with the editor xml layout file, i.e., is the code behind the avatar customization activity (screen) of the app. This class also instantiates an AvatarView object that displays the bitmap of the avatar and selected accessories, as well as instantiations of Accessory objects to display the accessory bitmaps the participant can choose for his avatar. Most of the code for this class was created for the CS 594 class project, and modified very little since then.

AvatarView extends the Android SurfaceView class, which provides a canvas for drawing on the screen. The AvatarView class is used to generate the avatar and

associated images in three of the screens that make up the user interface: the main, trading card, and avatar customization screens. It also creates the animation of the avatar, ball, etc. on the main screen, through loops which draw the images in increments along a path of motion, and play sounds at different points in the animation, corresponding to the situation onscreen (e.g., a cheer when a basket is made). Bala Rajan did the initial investigation into how to animate the bitmaps, and collaborated in the creation of the state structure used to keep track of the positions of all bitmaps.

BluetoothService extends Android Service. It was adapted from the BluetoothChat sample app that accompanies the Android SDK. It will run on boot up of the phone or when the app is started, so that it's always running, unless the app is terminated. It is set to run as a foreground Service, which means that it's "survival" takes precedence over non-foreground processes when Android needs to kill processes to reclaim memory. Originally written to actively establish a connection with the doser, by looking for and attempting to connect to it every five minutes, it now acts as a passive server, waiting for a doser to establish connection and send a timedatestamp. When a timedatestamp is received from a doser, it writes that information, along with other information tracked by the app, to a file in the directory on the SD card designated for the app by Android. This class also sends this information to a server, tron.evl.uic.edu, which stores it in files that can be accessed and analyzed by project team members.
A complete description of the information stored and transmitted with each timedatestamp received is as follows:

- Participant ID (5 digit) (from app survey screen)
- 10-digit phone number (read from phone by app)
- Doser type: "ICS" or "SABA" (received from doser)
- Dose date-time: MM/DD/YYYY~hh:mm:ss (received from doser)
- Date-time dose received on phone: MM/DD/YYYY~hh:mm (read from phone by app)
- GPS location (location of phone when it receives data from doser, which may NOT be the location when the dose was taken, determined by app from network and GPS information on phone)
- Air quality information (from AIRNow.gov)
- 't' if dose taken within acceptable time window and does not exceed total doses expected to be taken within that window (calculated by app by comparing timedatestamp from doser to dose times entered in survey screen), 'f' if those criteria are not met, and 's' if dose received while app is in sync mode
- Number of doses taken so far today
- Number of doses taken so far this week
- Number of doses taken for the entire duration of the study so far
- Number of inhaler puffs per dose the participant is expected to take

Dose information transmission example:
12474~17735555555~ICS~11/15/2012~17:27:39~11/15/2012~17:27~41.87~-87.65~location unavailable~t~3~17~39~two
When there are no timedatestamps to communicate, phone status is instead sent to the server, so that project personnel can confirm that the phone is in the state it needs to be in to function properly for the project. This information consists of participant ID and whether Bluetooth is enabled on the phone.
The BluetoothService class also broadcasts timedatestamp information to the AsthmaAppActivity class for the initiation of the avatar animation, as well as the calculation of scores for the main and trading card screens. When the app is in sync

mode, timedatestamp information is sent to the SyncMode class for display there. BluetoothService also sends and acknowledgement back to the doser from which the timedatestamp initiated, in the form of date and time, which the doser uses to set its own clock. It also receives acknowledgment from the Tron server, in the form of top scorer information, which is displayed on the trading card screen.

BluetoothStartupReceiver extends Android BroadcastReceiver. It receives the boot broadcast Intent from Android when the phone boots up. It is also called by AsthmaAppActivity when it starts. This class starts the BluetoothService Service and also creates several AlarmManager alarms: one for each of the different messages displayed to the participant in AsthmaAppActivity (inhaler use reminders, etc.), one at midnight each night to trigger the reset of the avatar's position on the main screen and of the trading card scores for the day (and the week, when a new week begins), and one to trigger the communication from the app to the server (every hour to communicate dose information or phone status).

MessageService extends WakefulIntentService from the CWAC-WakefulIntentService.jar added to the project.[8] MessageService is called by BluetoothStartupReceiver when the latter receives the alarms it created (see the description for BluetoothStartupReceiver above). MessageService creates the Intents necessary for the action associated with each alarm, or, in the case of the hourly communication with the server, calls the responsible method directly. WakefulIntentService is used to ensure that the phone does not go to sleep before the work initiated by this class is completed.

ScrollButton extends Android ImageButton. It's used by AvatarCreator for the displaying and toggling of accessory images in the avatar customization screen. It has not been changed since it was created fro the CS 594 class project.

SurveyActivity extends Android Activity. It's associated with the survey xml layout file, i.e., is the code behind the survey activity (screen) of the app. The survey is where project personnel, principally clinical ones, will enter information about participants and their medications. Some of this information is used by the app to do calculations for scoring and other purposes, and some is just for identification and tracking purposes. The survey screen is password protected so that participants cannot access it. Information entered in the survey is saved in a text file and SharedPreferences, and reloaded into the screen fields upon revisit, so that what's currently stored can be reviewed.
The following fields are present in the survey:

- Participant ID
- Participant Age
- Avatar Name
- Study Start Date
- ICS Medication Name
- ICS Medication Dosage
- ICS Puffs per Dose
- ICS First Dose Time
- ICS Second Dose Time
- SABA Medication Name
- SABA Medication Dosage

- SABA Puffs per Dose
- SABA Frequency
- Shot Probability % (allows project personnel to set the likelihood that a basketball shot on the main screen will go in the basket or not)

SyncMode extends Android Activity. It's associated with the syncmode xml layout file, i.e., is the code behind the syncmode activity (screen) of the app. Sync mode is used for setting the clock on the dosers, as indicated above. When in sync mode, a timedatestamp from a doser causes the app to transmit the phone's time and date back to the doser, for it to use to set its clock. Sync mode has a very minimal interface, with just a large field for displaying the timedatestamp from the doser and an exit button.

TradingCard extends Android Activity. It's associated with the tradingcard xml layout file, i.e., is the code behind the trading card activity (screen) of the app. TradingCard does much of the calculating for the scores displayed on the trading card screen (though some are done in AsthmaAppActivity) and is responsible for the display of all stats on that screen. It also has a BroadcastReceiver for the Intent for resetting scores at midnight each night. Bala Rajan did the programming for some of the images on this screen.

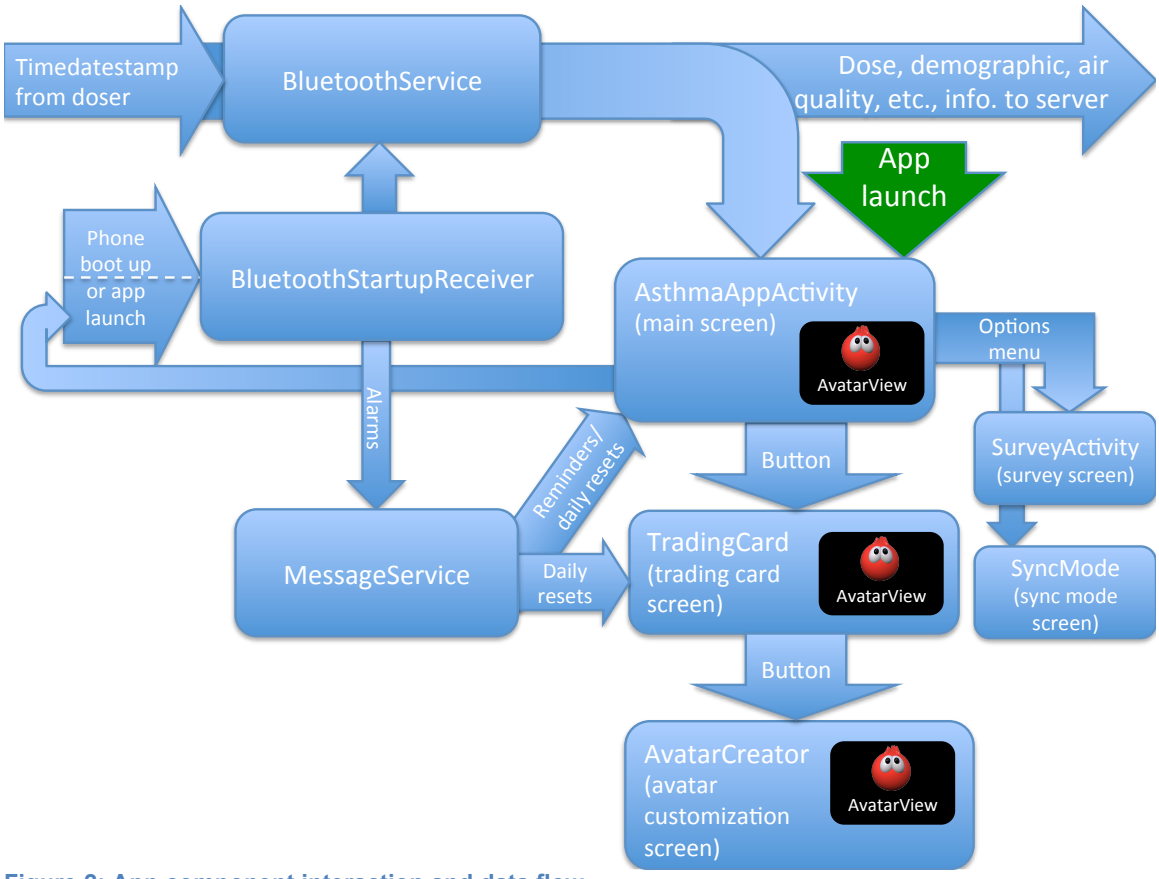
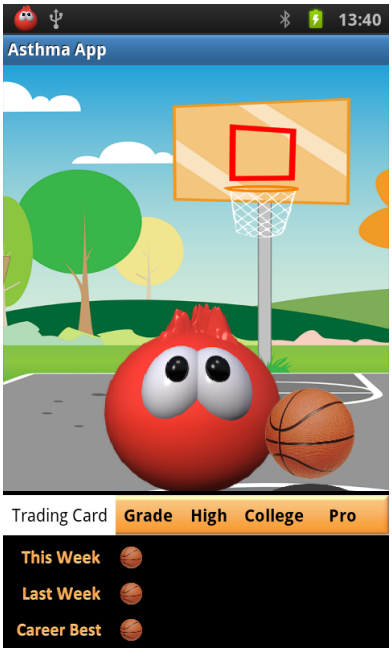


Figure 2: App component interaction and data flow

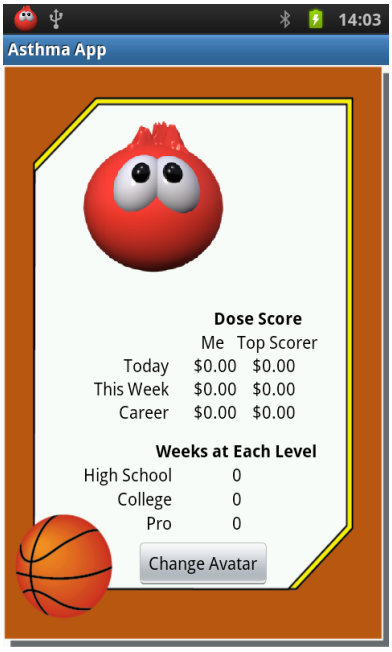
Results

The following is a detailed description of the functioning of the Asthma Intervention Android app. When the app is launched from the phone home screen via its icon, its main screen (avatarpage.xml layout and the program code behind it, AsthmaAppActivity) is launched and presented. The main screen program code causes the launch of BluetoothStartupReceiver, which sets several alarms for time-based functions of the app, and also starts up BluetoothService, the Service that receives data from the dosers, initiates animation in the main screen, updates the scoreboard on the main screen and the trading card, and transmits data to the server.

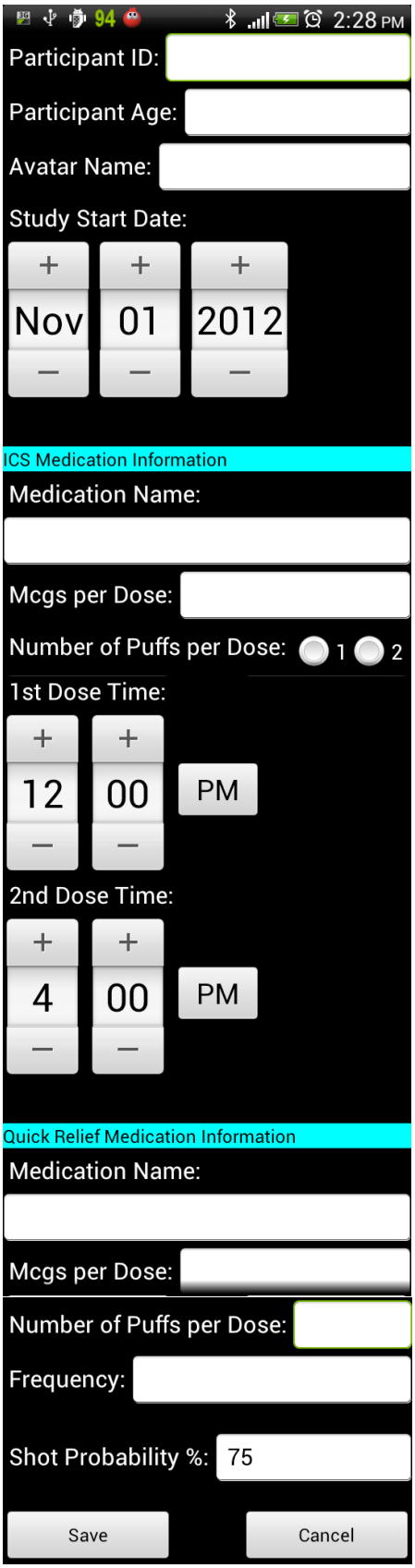
Before any timedatestamps have been sent to from a doser, the app main screen is in its initial state, with the avatar in center of the screen horizontally, at the closest point on the basketball court, and not customized in any way.



This is how the trading card initially looks.

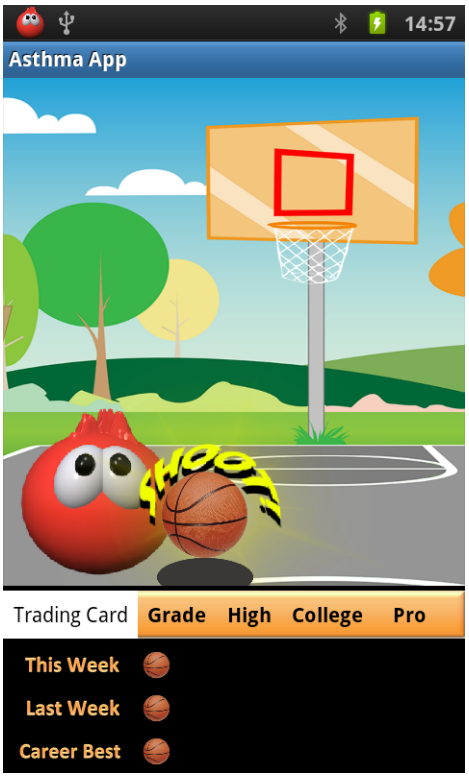


This is the survey screen in its initial state. The key fields for the functioning of the app are “Number of Puffs per Dose”, which affects the animation on the main screen, determines when a participant has taken a full dose of medication, and has to be taken into account when calculating scores on the trading card screen; and the two “Dose Time” fields, which are used to determine whether puffs taken are within the accepted time frames, according to the schedule determined by the participant’s physician.

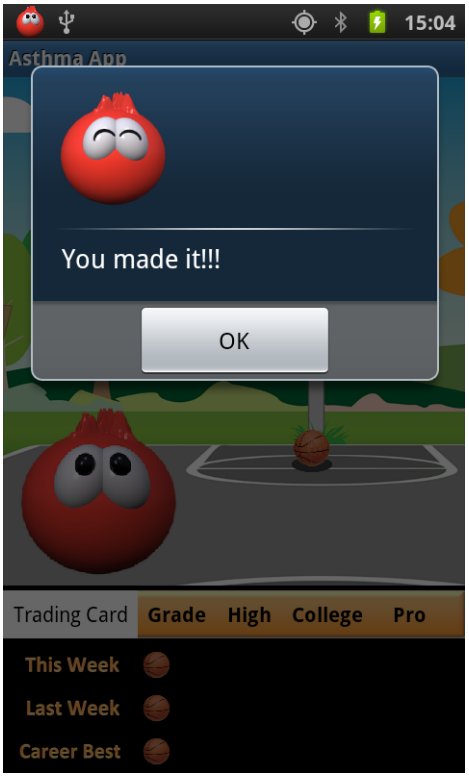


When the participant takes a puff from his ICS inhaler, the avatar moves to the next position on the court and the participant is prompted to shoot the basketball by touching it. Use of the SABA inhaler does not initiate the animation nor contribute to the scoring in the app in any way.

There are a total of four positions from which the avatar can shoot, each one closer to the basket than the last. A participant who takes two puffs per dose will shoot from all four positions (two in the morning and two at night). A participant who takes one puff per dose will only shoot from the second and fourth positions, with the avatar moving through the first and third positions after the first and second puffs of the day, respectively. Participants who take three or four puffs per day (planned though not implemented as of the writing of this report) will take additional shots at the fourth position. At midnight the avatar returns to the initial position; any shots not shot in the previous day cannot be shot in the new one.

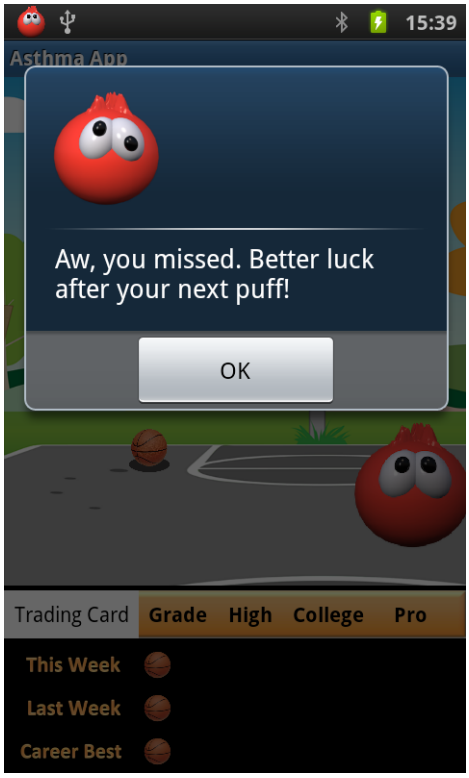


The probability of a made shot is set in the survey screen (not depicted above), and is not under the control of the participant, nor does it affect any of the scoring in the app (which is all determined by inhaler use). If the shot is made, the screen looks as follows. A cheering sound is also played.



A missed shot looks as follows. It's accompanied by a sound of sympathy from a crowd (i.e., an "aw" sound).

If the participant has taken a second puff before launching the basketball shot, the app will wait for the first shot to be taken, and then, once the dialog box indicating that the shot has been made or missed has been dismissed by touching "OK", the avatar will immediately move to the next position on the court and wait for the participant to touch the ball to shoot again.



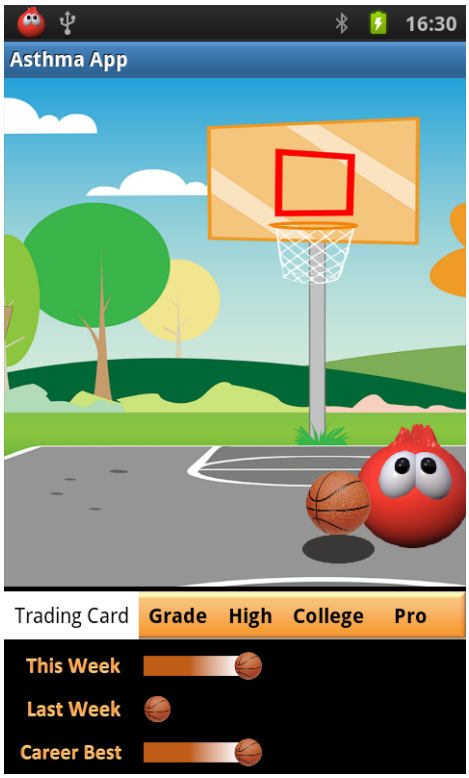
For each full dose (determined by the number of puffs per dose entered in the survey) the participant takes, he is awarded $.50, which is added to his score on the trading card. Thus, after 1 full dose, the trading card screen will look as depicted at right.

The Top Scorer column displays stats from the server to which all participant phones send their data. Each time a puff is received by the app, or every hour in the absence of puffs, the app communicates with the server. The server responds by calculating which of all recipients has the highest score in each category, and sends that information back to the app, which then displays that information in the Top Scorer column. If a given participant has the top score in any category, an icon indicating this displays in the upper right corner of the screen.

The scoreboard at the bottom of the main screen shows the participant's adherence to his ICS inhaler use schedule for the current week, past week, and his best week. It's calculated by dividing the number of puffs (not doses) taken during the week in question by the number expected for that entire week, then comparing the resulting number to ranges set for each level (high school, college, and pro) by the project team. The ranges are as follows:

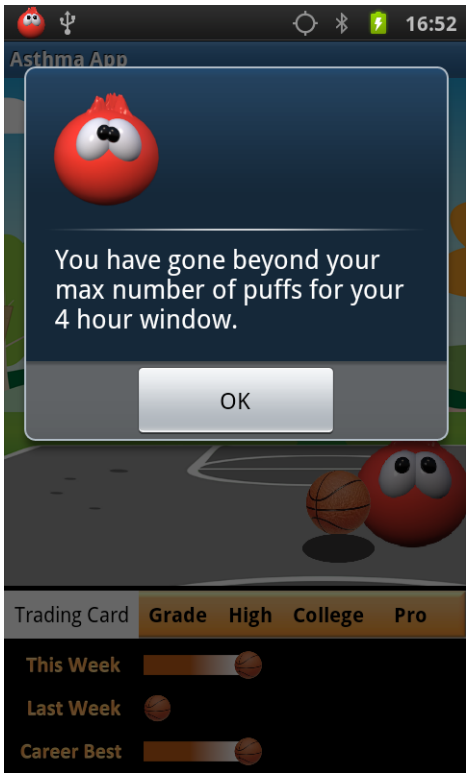
High school: >= 60% and < 70%
College: >= 70% and < 90%
Pro: >= 90%

As an example, a participant who takes two puffs per dose is expected to take 28 puffs in a week (2 puffs per dose X 2 doses per day X 7 days in the week). If, by the end of the day on Friday of the current week, the participant has taken all but four of his puffs, for a total of 20 puffs, he will have taken approximately 71% of his puffs for the week, and will thus have reached the college level for the week.
For a participant who has reached the high school level for the week, and whose best week is the high school level, the scoreboard looks as depicted at right.

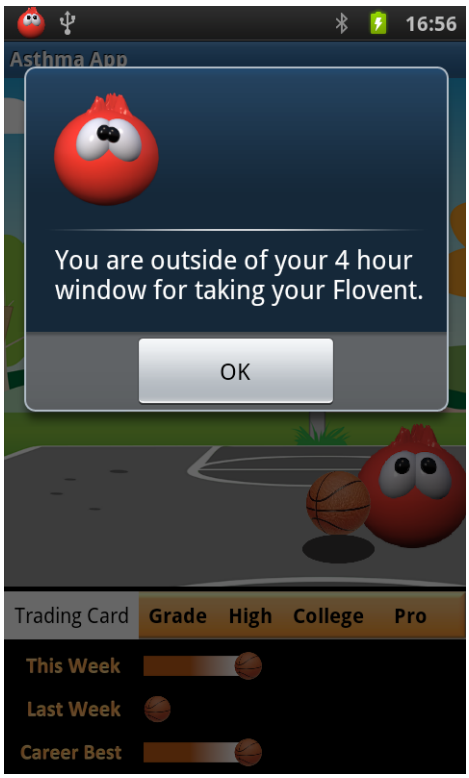


The trading card also displays information about weekly level attainment, showing how many weeks the participant has performed at each level.

If a participant attempts to take more than his prescribed number of puffs for his dose time window (defined as two hours before and two hours after the two dose times entered in the survey), those puffs are not counted toward his level on the main screen scoreboard nor in the Dose Score on the trading card and a message appears on the main screen.



If the participant attempts to take any puffs outside of his dose time window, these puffs are also not toward his scoreboard level or Dose Score, and he sees the following message on the main screen.



The avatar customization screen is accessed via a button on the trading card. Before any customizations have been applied to the avatar, it looks as follows. The three icons at the bottom of the screen are used to select between the three categories of accessories, and accessories are applied by touching on an item in the column on the left side of the screen. Items can be removed by touching on the blank space at the top of that column. Only one accessory from each category can be applied at a time. (Touching on an accessory in a category causes that accessory to replace any that was already on the avatar.)

Here's how the avatar looks after applying an arbitrary combination of accessories from all three categories. The avatar will appear thus accessorized in all three screens on which it appears.

The sync mode screen is accessed from the main screen via the dedicated device menu button. This menu also provides access to the survey (via “Accounting stuff”, a phrase chosen because the participants are expected to find it uninteresting), both of which are protected by the same password, so that participants do not access these screens. Also showing at right in the menu is “Dev param”, which is used to change variable values in the app without having to change program code and reinstall the app, for testing purposes. This menu item will be removed from the app before the participants begin to use it.

The sync mode screen is shown at right. Puffs received while in this mode are recorded in the file on the phone with all other puff entries and sent to the server, but these puffs have no effect on scoring or animation within the app. As with puffs received outside of this screen, the app responds to the doser by sending the phone’s date and time.

Conclusion

There is still work to be done before this app can be called complete. A few elements, as noted above, have yet to be implemented.

Though some informal testing has been done as part of the construction of the app, more testing still needs to be done. Within a week or two of the writing of this report, this testing is scheduled to commence. The plan is to try to simulate the in-study use of the app by having members of the project team and some of their associates in the Electronic Visualization Laboratory carry dosers and phones with them and take puffs throughout the day as the study participants are expected to do. This will inevitably result in the discovery of bugs, and likely design features of the app that will be deemed less than satisfactory and thus may need to be changed.

Also, the phone that will be used for the study has not been chosen yet, so problems as a result of the way the app behaves on the device on which it will ultimately be used may manifest. Related to this, a major implementation flaw that's been discovered is that the interface of the app does not properly display on phones that have different resolutions than the Galaxy Player. For that reason, candidates for the phones for the study have mostly been limited to those that have the same resolution as the Galaxy Player, but this is a limitation that will likely need to be addressed in the future.

The robustness of the app is a concern. Crashes have occurred during the informal testing of the app, and while many of the causes of these have been addressed (e.g., memory overuse due to the size of the bitmaps in the app), some (e.g., exceptions caused by some sort of problem with the Samsung Player's Bluetooth) may yet prove to be a problem.

Much of the challenge of the implementation of the app likely arose from the lack of knowledge of the Android platform on the part of the author of this report and the other project members. The platform is still relatively young, and good resources for those trying to learn it are fairly scarce. Though the programming language used is Java and XML, and thus not unfamiliar to the author and other team members, the platform does have some notably unique characteristics and constructs. Some of the ones at issue in this project were:

- The Activity lifecycle, i.e., how the different interface components of an app behave over time, in response to navigation by the end user, and as a result of the constraints that the operating system imposes (e.g., memory reclamation). An example of this is the difference in how Android responds to the use of the dedicated device back button vs. the dedicated home button. Pressing the first from the main screen of an app causes the main Activity to be destroyed (and thus results in it being recreated when the user switches back to it); pressing the latter does not.
- Related to the first item are the means and resources for the management and recreation of app states in response to lifecycle changes. For this project in particular, SharedPreferences were used to maintain necessary state information, but this was a choice made in ignorance of an Activity feature called the savedInstanceState, which is specifically designed for this purpose.
- The use of Intents for communication between different program classes. There are many flavors of Intent, and which to use and why is a bit of an art. Also

knowing what happens to an Intent when it's sent from one component to another is important—Intents can tend to stick around and be acted on more than once by an activity, but this is generally not desired. Particularly challenging for this project was getting the main screen to come to the foreground whenever a puff was received from the doser. This behavior is actually frowned upon generally in Android app design, because the assumption is that people do not want their apps popping up no matter what they happen to be doing on their phones—this can be an annoying interruption. The desire for this project, though, is for the app to be strongly persuasive and thus be aggressive in presenting itself to the participants. Bringing the main screen to the foreground whenever a puff is received was not a huge technical challenge, but what was difficult was realizing that every time this happened the main screen was getting recreated, without the previous instances of it being destroyed, thus causing the phone to run out of memory. The solution was a change to the Android manifest file to tell Android to try to limit the main screen (also ultimately used for the other screens that contain the avatar bitmap, too) to one instance at a time, and an addition to the "bundle" accompanying the Intent that launches the main screen.

- Animating the avatar, his ball, and the ball's shadow. Though animation tools exist in the Android API[9], animating everything with loops in which in each iteration bitmap positions were incrementally updated and the bitmaps redrawn, was ultimately selected as the animation approach.[10]

Other ways in which the app could probably be improved include:

- The breaking up of some of the larger classes. AvatarView and BluetoothService in particular have a lot of code in them, and figuring out which parts of the code are doing what would likely be a challenge for an outsider (though attempts were made to include copious comments). In addition to making collaboration more difficult (because fewer classes mean a greater likelihood that two developers with different tasks would still have to work in the same class), the shear intimidation factor of touching these large and complex classes may be an issue, at least for novice programmers. AvatarView, in addition to serving to provide the means to draw the avatar and other bitmaps in different screens, also handles all of the animation on the main screen. Though how to separate these functions admittedly doesn't seem obvious the author at this point, still seems appropriate or at least worthy of consideration.
- The handling of the "business logic" of the app. There are a lot of dependencies within, and timing expected of, the app (e.g., alarms to go off at certain times, certain actions on the part of the user to have certain results at some times but different results at others). Unfortunately, as this logic was built, its implementation was scattered throughout several classes, mostly AsthmaAppActivity, BluetoothService, BluetoothStartupReceiver, MessageService, SurveyActivity, and TradingCard. The means of recording and accessing the different conditions and states also varies quite a bit; in some cases information is stored in SharedPreferences, in some cases in text files, in others in variables, and sometimes a mixture of more than one of these. Though bitmap positions corresponding to progress of inhaler use throughout the day are saved in a fairly well-organized state structure, other app state information is not, as management for different conditions has been developed in a fairly ad hoc way for each one, with the strategy for handling each chosen on the basis of what seemed easiest for each, not on how a coherent whole could be built and maintained.

It should be noted for those to follow on this project that the author found the following resources most useful for learning Android programming and for sample code:

- The Busy Coder's Guide to Android Development[11]
- Android Developers[12]
- Stack Overflow[13,14]

The author would like to thank the other members of the project team for their collaboration, patience with the author (especially relative to his whining and interrupting :), for pushing to make the software more than was perhaps feared it could be, and for general advice and counsel.

---

[1] J. Albers, J. Chambers, B. Grossman, J. Lancaster, M. Matulyauskas, and H. Thompson. An Avatar-Centric Smartphone App to Promote Adherence to Asthma Treatment for Inner-City Youths. Project report for Human Augmentics course, University of Illinois at Chicago, May, 2012.

[2] Boland P. The emerging role of cell phone technology in ambulatory care. *The Journal of ambulatory care management*. 30(2):126-33. Available at: http://www.ncbi.nlm.nih.gov/pubmed/17495681.

[3] Mosnaim GS, Cohen MS, Rhoads CH, Rittner SS, Powell LH. Use of MP3 players to increase asthma knowledge in inner-city African-American adolescents. International journal of behavioral medicine. 2008; 15(4): 341-6. Available at: http://www.ncbi.nlm.nih.gov/pubmed/19005935. Accessed April 22, 2012.

[4] Ruiz JG, Andrade AD, Roos BA, Twain M. Medical Avatars—An Innovative Approach to Fostering Health Promotion and Lifestyle Change. *Federal Practitioner*. 2012;(April): 1-7.

[5] McGonigal J. *Reality is Broken: Why Games Make Us Better and How They Can Change the World*. London: Penguin Books; 2011.

[6] http://acra.ch/

[7] U.S. Environmental Protection Agency Office of Air Quality Planning and Standards. AIRNow Local Air Quality Condtions. Available at: http://airnow.gov/.

[8] https://github.com/commonsguy/cwac-wakeful/downloads and http://commonsware.com

[9] http://developer.android.com/guide/topics/graphics/index.html

[10] Deitel P., Deitel A., Morgano M., *Android for Programmers: An App-Driven Approach,* First Edition, Prentice Hall, 2012, Chapter 7.

[11] Murphy, M., *The Busy Coder's Guide to Android Development,* CommonsWare, 2012.

[12] http://developer.android.com/index.html

[13] http://stackoverflow.com/

[14] http://stackoverflow.com/questions/12607395/alertdialog-persisting-on-screen-too-long