LockED

by

Marina Greben, Martin Manger, Matthew NeCamp

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Marina Greben — Date: 4-18-16

Martin Manger — Date: 4/18/16

Matthew NeCamp — Date: 4/18/16

Professor Robin Carew — Date: 4/18/16

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2016

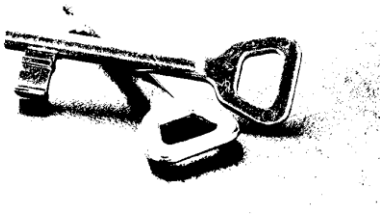


Table of Contents

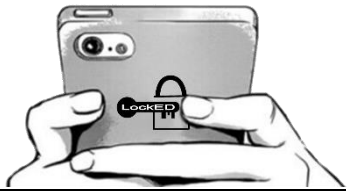
Tables and Figures

We would like to express our gratitude to the University of Cincinnati Senior Design Class of 2016. Particularly, the teachers who spend hours teaching, revising, reviewing, helping, and commenting on each and every one of our projects and proposals. Without these contributions and the help of the faculty, staff, peers, classmates, friends, our project wouldn't be what it is. After all, a great invention isn't created alone.

THANK YOU

ABSTRACT

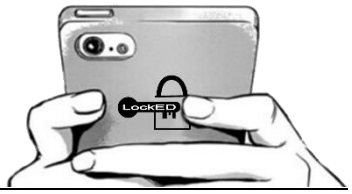
An average person's Smartphone contains contact information for friends and family, banking applications, even email capability in addition to the standard call and text capability of a regular cell phone. However, most of that information is protected by one, or two layers of passwords, and sometimes not even that. In the event that a person's phone is taken from their possession, one simple swipe of the screen can allow unfettered access to a person's life. We saw an opportunity to create LockED, an application that would limit a person's access to the phone based on individual passwords used. Our application can store any number of "guest" user accounts allowing differing levels of access to the phone in order to limit another user's access to sensitive data or applications. By limiting this access we better protect the sensitive information stored on smartphones.



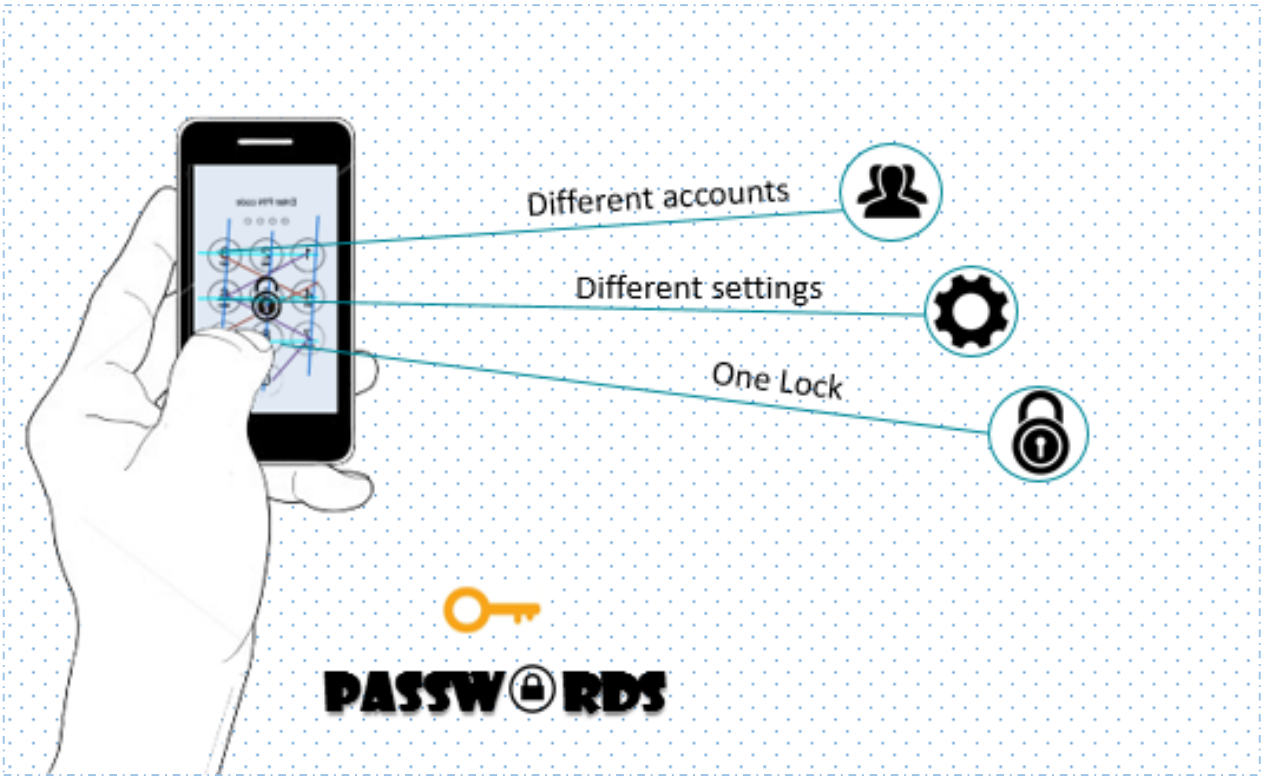
Smartphones are a brilliant invention but they lack the component of security that most modern computers already possess. Imagine how much information is on a cellphone. People store everything on their phone, be it the pictures they take, the contacts they save, and the information they store. Their whole lives could be on that phone and yet some people would still choose to lose their phone rather than their wallet.

The problem that we have seen is that an average user is not protecting their digital life as well as they are protecting their actual life. Such is the case with Mat Honan, who lost the videos and pictures of his daughter's birth when everything was erased from his phone. While hacking is the leading cause of data loss and information theft, simple access to a cellphone by a friend, colleague or even just a stranger can lead to just as significant of a loss. The loss of a cellphone, with all personal information stored, is just as bad as identity theft because most phones store easily accessible information such as bank details, personal information for contacts, email accounts connected to business accounts, and potentially incriminating photos or videos. This could lead to scenarios such as job loss and jail time if circulated to the wrong people. To lessen the risk of losing personal and sensitive information designed an application that would allow the user to limit the functionality and accessibility of the phone. This means that the owner can set different levels of accessibility depending on the user (i.e. child, friend, stranger, etc.). This would be similar to a guest account on a home computer, which also has different levels of access depending on the user.

Essentially, our application wants to ensure that the user has a good experience and no longer worries about their photos, videos, and information possibly getting erased, stolen, or accessed without permission. However, there are some problems that can arise from our application if the user doesn't use it effectively or as intended. Such as if they forget their password, give accounts the wrong level of accessibility, or simply decide not to use it.

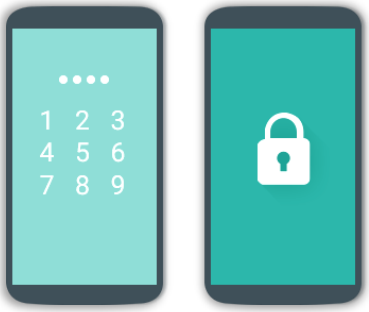


This project is based around an easy to use application that will allow the phone owner to set up different levels of access to personal information, settings, and features based on the '**password**' they enter.



Project Description Figure 1

Primarily, the application will record different user accounts containing different levels of access.

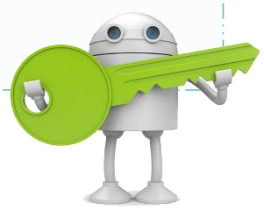


Design Objectives

This application should be able to run on any android device. However, for the purpose of testing everything will be coded and run in a virtual device using Android Studio. LockED has two major goals: *Security of Personal Information* and *Ensuring Phone Use is Limited*.

In order to accomplish this, we have used the idea of multiple "accounts" created by the phone owner providing different levels of access. Some examples are:

1. **Gaming**: no transactional data connection in order to minimize the risk of in-app purchases
2. **Emergency**: delete all contact information, prevent any use of the phones cell, data, or wireless capabilities, and prevent access to any app located on the device.



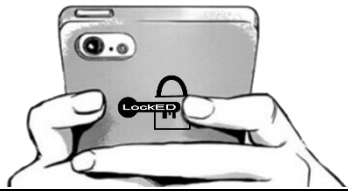
Methodology

We took the basic software release life-cycle (pre-alpha, alpha, and beta) approach in developing this application because it allowed us to work through each piece in a timely manner. During the *pre-alpha* phase we brainstormed and discussed what we wanted the purpose of the application to be. This allowed us to come up with a workable *alpha* design that could act as a proof of concept for our final version.

After creating the proof of concept, we used it as a foundation and began to improve on it while progressing through the *alpha* stage of development. We added more content, features, and abilities and eventually had a functional working application; though it was very plain.

Following the *alpha*, it was time to add any extra functions we wanted. This ranged from working on the functions of the application and making it more efficient to making a streamlined and improved graphical interface.

Testing was done after any significant change to the code, during the *alpha* and *beta* phases, and constantly throughout the preparation for the IT Expo in the spring.

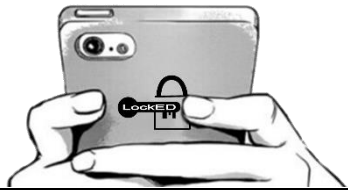


User Profile Form

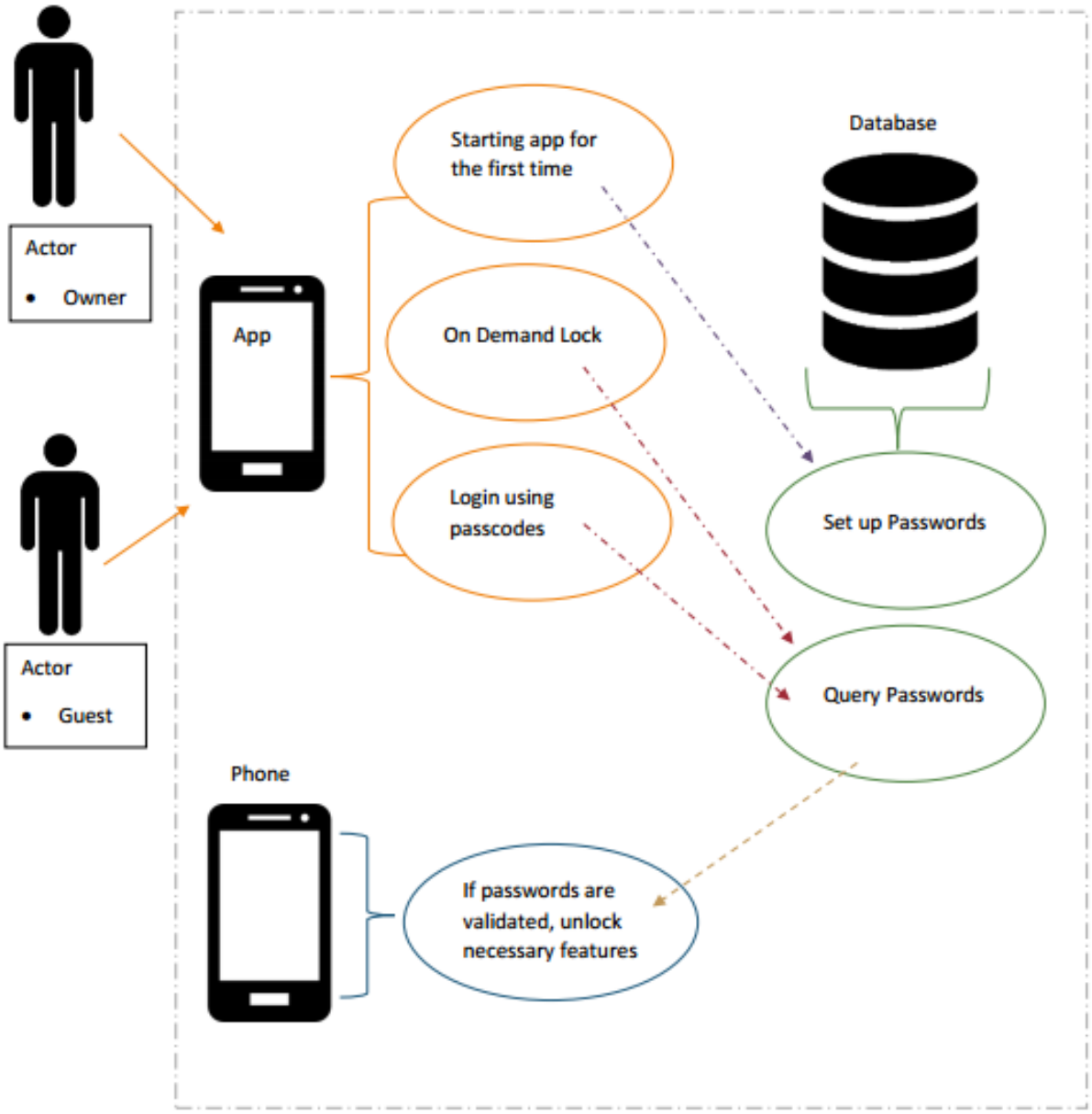
Application:	Potential Users:	Software and Interface Experience:
LockED Mobile Security Application LockED	**Primary Users, Phone owner, or Host** Anyone who is looking to add additional security features to their phone in order to protect their sensitive information from friends, family, or colleagues who might use the owner's phone. Parents or teachers who want to keep their children from accessing sensitive information when they lend their phone out to them. **Secondary Users, Guests, or Assigned phone user** Anyone who the owner wants limited functionality of their phone during temporary use. Business personnel who are assigned a phone with limited features that prevent distractions at work. Children who want to use their parent's phones who have special restrictions. Malicious people who are trying to tamper with the phone.	The android app will brief the user with different levels and layers of security and a general overview on what to expect with the security service. Primary Users will need to register one or multiple passwords depending on the features and layers of security they want to access. Custom settings include limiting native phone services (Wi-Fi, mobile data), disabling specific applications installed, hiding select folders or files. In addition, a user can register an Emergency password to thwart anyone who is attempting to steal the user phone. Once the settings are customized by the user, the application will be active and a new unlock screen is presented when someone tries to use the phone. Depending on which password is entered, the phone will behave according to the settings and features that are tied to the password. (*) If a malicious person who is attempting to access the phone, enters the *password during the unlock phase, the phone will shut down and delete any sensitive data that could be used for ID theft.

Experience with Similar Applications:	Task Experience:	Frequency of Use:	Key Interface Design Requirements
Target users can be anyone who wants additional privacy with their phones. This can range to users as young as high-school students to as old as tech savvy seniors and parents. People who have had their ID stolen in the past and want to feel safer knowing that their data is secure. If they can enter a password and adjust the basic settings of their phone without problems, they should easily be able to work with our app.	1. Initial app install with welcome screen and basic setup 2. Suggest advance options and custom settings 3. Finalize initial app install 4. User enters programmed password 5. User receives limited function of phone based off settings 6. User enters wrong password 7. User receives alerts 8. User enters wrong password multiple times 9. Phone enters data wipe and secures data on phone.	Depends on how often a user allows their phone to be used by other individuals. If a parent is very busy, app usage could be very frequent as they use mobile devices to entertain their children. Business could use it daily as they ensure quality within their work environment.	1. Easy to set up walkthrough guide 2. Clean and simple interface to emphasize functionality and navigation of service 3. Randomized password key-pad to avoid pattern recognition in attempts to steal password 4. Simple to understand description and feature list 5. Advanced features for business case Specific features aimed for children users 6. Protective measures, data wipes, and settings for harmful users and thieves

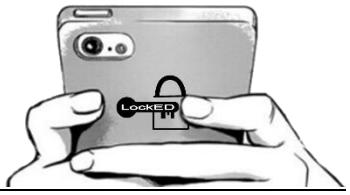
User Profile Figure 2



LOCKED



User Case Figure 3

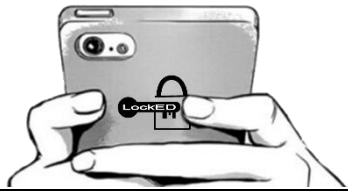


Everything our application is currently utilizing during production is open-source (Github and other such online solutions/platforms) or is currently owned—such as laptops, desktops, smartphones, etc. Other than labor and time allocated to programming and researching, the total money used to create this application is zero dollars. That being said, if the open-source platform we are currently using to develop our application falls through and does not contain the features we need, we would look into potentially purchasing another platform that does. Some other possibilities that could occur are: one of our devices breaking and/or needing other hardware/software pieces. As of right now though, our estimated cost is zero dollars.

As for the **labor costs**, we are currently not being compensated monetarily by any company to develop this application. However, if we were being paid a reasonable wage (about $18/hour) for 40 hours of work per week per person, each would earn about $18,000, leading to a total labor cost of $54,000. Unfortunately, we are not able to each put a full 40 hours a week worked per person into this project. An estimate of about 20 hours a week worked per person is much more realistic and would bring the potential labor cost to $27,000.

Materiel Costs (Laptops, Software, etc.)		
Item	**Expected Cost**	**Actual Cost**
Computer Hardware	$0.00 (Already Owned)	N/A
Coding Software	$0.00 (All Open Source or Free)	N/A
Testing Devices	$0.00 (Emulated Devices)	N/A
TOTAL COST	$0.00	N/A
Labor Costs		
Item	**Expected Cost**	**Actual Cost**
Actual Wages Cost	$0.00 (Not Supported by External Company)	N/A
Conference Costs	$0.00 (No Conferences Expected to be Attended)	N/A
Simulated Wages Cost	*$27,000 Simulated Wages (18$/hour, 20 hour/week, 25 weeks)*	N/A
TOTAL COST	$0 (Simulated $27,000 for wages)	N/A

Budget Chart Table 1



Timeline

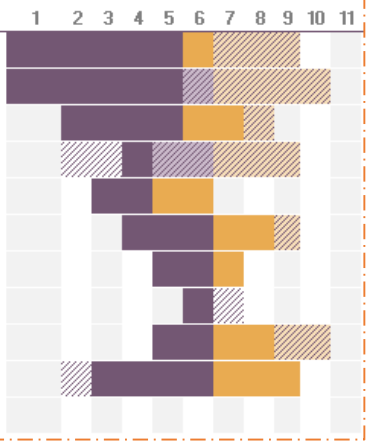
FIRST SEMESTER	
Week 1	• Analyze and agree on core/necessary features for prototype and final deliverable. • Begin looking at secondary features (wants not needs)
Week 2:	• Analyze and agree on secondary features. • Begin determining how primary features can be implemented. • Begin Usability study.
Week 3:	• Determine how to implement necessary features within our app. • Begin creating a "proof of concept" app that will be our beginning for the prototype. • Begin "Pre-Alpha" stage. • Determine programming languages and platforms.
Week 4:	• Proof of concept (Pre-Alpha) complete. • Basic core features up and running and very basic UI. • Begin morphing Pre-Alpha into Alpha.
Week 5:	• Continue developing the prototype (Alpha) from the proof of concept. • Begin considering preparing for presentation.
Week 6:	• Prototype complete. Interactive UI, not necessarily nice-looking, and core features included. • Discuss Potential Risks. • Prepare for presentations and bugfix as needed.
Week 7-10:	• Minor changes to the prototype as needed. • Prepare for presentation.
SECOND SEMESTER	
Week 1	• Prototype Cleanup • Update timeline
Week 2:	• Analyze prototype, how to improve, what works, what needs to be changed and what needs to be added • Update Deliverables
Week 3:	• Determine what features need implementing from the secondary features list. • Create test plan and Start the plan on how to implement • Update Abstract
Week 4:	• Implement security changes • Implement storage capacity • Implement data removal capacity
Week 5:	• Continue developing the prototype (Alpha) from the proof of concept. • Begin preparing for presentation. • Start the draft for the final paper and draft poster for the presentation
Week 6:	• Improve prototype (Alpha) to Beta version for the expo. • Test Beta • Continue preparing for presentation and finish up the final paper
Week 7-10:	• Polish Beta version into release version (1.0). • Finalize the Poster • Get the proper supplies for the expo ordered and begin preparation.

Timeline Chart Table 2

Gantt Chart

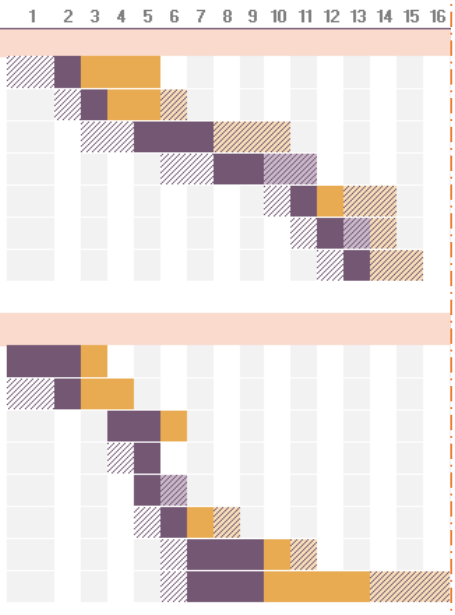
Fall Semester

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Core Feature Analysis & Design	1	5	1	9	75%
Extra Features Analysis & Design	1	6	1	10	50%
Research: How to enact Features	2	4	2	7	90%
Usability Study	2	5	4	6	30%
Proof of Concept App	3	2	3	4	100%
Improve PoC in Prototype App	4	3	4	6	85%
Prepare Prototype for Presentatic	5	2	5	3	100%
Prepare for Presentations	6	2	6	1	100%
BUGFIX	5	2	5	6	75%
Outreach	2	5	3	7	100%



Spring Semester

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Technology Changes					
Protype Cleanup	1	2	2	4	100%
Determine Security Changes	2	2	3	4	95%
Implement Security Changes	3	5	5	6	50%
Implement Storage Capacity	6	6	8	4	50%
Implement Data-Removal Capacity	10	2	11	4	50%
Test Beta Release	11	3	12	3	15%
Final Cleanup	12	2	13	3	5%
Paperwork					
Update Gantt Chart	1	2	1	3	100%
Update Deliverables	1	2	2	3	100%
Testing Plan	4	2	4	3	100%
Update Abstract	4	2	5	1	100%
Draft Paper	5	2	5	2	95%
Draft Poster	5	2	6	3	95%
Final Poster	6	4	7	5	80%
Final Paper	6	4	7	10	75%



Gant Chart Figure 4

Testing Plan

Overview

As with all products our application must be thoroughly tested. Due to the fact that our application is on an android phone and could be used by anyone we have decided that there are 3 types of demographics we need to have test our application. The three demographics are: Technologically skilled people, Non-Technologically skilled people, and random strangers who could be anywhere on the spectrum.

Testing setup

In order to run our tests we have decided to invite multiple people from the three demographics chosen to operate specific tests of our application on our emulator. There are a few tests that we will run ourselves as they are a simple testing of the layout and testing of the individual commands. Below is a list of the tests we will run ourselves.

Self-tests

- Will the app start? - *Start the app from the app menu.*
  - **Expectations:** The app will start.
- Verify layout on multiple screens. - *Ensure the layout is usable on multiple screens.*
  - **Expectations:** The layout will work on multiple screens.
- Limit Wireless access. - *Use a password with no wireless access to verify wireless connection.*
  - **Expectations:** No wireless
- Limit Cell access. - *Use a password with no cell access to verify no wireless connection.*
  - **Expectations:** No cell
- Limit Data access. - *Use a password with no data access to verify no wireless connection.*
  - **Expectations:** No data

Other user tests: 3 Demographics

- Adding an admin password: *Type in admin password and have it be stored.*
  - **Expectations:** The app will store the admin password.
- Unlock phone with Admin Password: *Type in admin password and unlock phone.*
  - **Expectations:** The phone unlocks.
- Adding new user account: *Use options to add new user.*
  - Sub test 1: Locate the options for adding new user?
    - **Expectations:** they can find the options
  - Sub test 2: will the new user be added?
    - **Expectations:** the new user is added
  - Sub test 3: can't use duplicate password?
    - **Expectations:** duplicate password is rejected
- Unlocking with new user: *Unlock the phone using new user information.*
  - **Expectations:** The phone unlocks
- Lock the phone: The user locks the phone.
  - **Expectations:** The phone locks.
- Unlock phone screen appears when phone is unlocked: *Phone unlocks & shows our app.*
  - **Expectations:** our app shows.
- False unlock password: *Attempt to unlock with false password.*
  - **Expectations**: Phone does not unlock.
- Editing user account: *edit the settings of an account that already exists with different features.*
  - **Expectations:** Access can change but passwords cannot.
- Deleting user account - *delete the saved account/features all together.*
  - **Expectations**: User account will be deleted.

Logging test reports:

As each test is run one of our group will assist the tester as they progress through the tests and will note any problems they run into. After testing is complete any problems noted will be explored in detail to solve and re-test after the initial round of testing.

Self-Tests						
Test	**Explanation**	**Expectation**	**Pass/Fail**	**Date**	**Reason**	**Logged By**
Will the app start	Start the app from the app menu	The app will start	Pass	Mar 22,24,26	Success	Martin
Verify layout on multiple screens	Ensure the layout is usable on multiple screens	The layout will work on multiple screens	Pass	Apr 2,3	Verified on 3 screens	Martin
Limit Wireless access	Use a password with no wireless access to verify no wireless connection	No wireless	Pass	Jan 22	Wireless access limited as needed	Martin
Limit Cell access	Use a password with no cell access to verify no wireless connection	No cell	Fail	April 6	Commands for Cell limitation not valid	Martin
Limit Data access	Use a password with no data access to verify no wireless connection	No data	Fail	April 6	Commands for Data limitation not valid	Martin
Remove Contacts	Use a password with no contacts access to verify deletion of contacts	No contacts	Pass	April 4	Contacts completely removed	Martin

Test Plan Table 3

Other User Tests						
The following tests must be run at least 3 times, one for each demographic.						
Adding an admin password	Type in admin password and have it be stored.	The app will store the admin password.	Pass	Jan 22, April 2	Admin passwords added every time	Martin, Marina
Unlock phone with Admin Password	Type in admin password and unlock phone.	The phone unlocks.	Pass	Jan 22, April 2	Phone unlocked every time	Martin, Marina
Adding new user account	Use options to add new user.	All three sub-tests work	Pass	April 2,3,4	New user accounts added	Martin, Marina
Sub test 1	Can they find the options for adding new user?	they can find the options	Pass	April 2,3,4	No issues finding new user dialog	Martin, Marina
Sub test 2	Will the new user be added?	the new user is added	Pass	April 2,3,4	New users added	Martin, Marina
Sub test 3	Can't use duplicate password?	duplicate password rejected	Pass	April 2,3,4	Duplicate passwords rejected	Martin, Marina
Unlocking with new user	Unlock the phone using new user information.	The phone unlocks	Pass	April 2,3,4	Phone unlocks with necessary access	Martin, Marina
Lock the phone	The user locks the phone	The phone locks.	Fail	April 2,3,4	Phone lock command not implemented	Martin
Unlocked phone screen appears	Phone unlocks and shows our app.	Our app shows.	Pass	April 2,3,4	Unlock screen shows	Martin, Marina
False unlock password	Attempt to unlock with false password.	Phone does not unlock.	Pass	April 2,3,4	Phone does not unlock	Martin, Marina
Editing user account	Edit the settings of an account that already exists with different features.	Access can change but passwords cannot.	Fail	April 2,3,4	Edit settings not implemented	Martin, Marina
Deleting user account	Delete the saved account/features all together.	User account will be deleted.	Pass	April 6	Removal of accounts successful	Martin
Creating a new user account	Using the credentials of the old user account that was deleted.	New account created without trouble.	Pass	April 2,3,4	Accounts created without trouble	Martin, Marina

Test Plan Table 4

Problems Encountered

Everyone wants a development cycle that goes smoothly and encounters minimal problems. As much as we would like to say that this is how our development cycle went, that was not the case—we ran into several scripting issues alone which took us quite some time to bypass.

Issue 1: *Taking control of the phone lock*

Though, we couldn't "take over" the unlocking mechanism we did find another way to force our app to run in front of the traditional unlock screen when the phone is woken up. If given the necessary privileges we can dismiss the lock screen with a command. However, this only works if the phone does not have any native security for their lock-screen active. To bypass that we would need to overwrite the currently stored password or pattern, dismiss the lock, and then re-instate the password or pattern

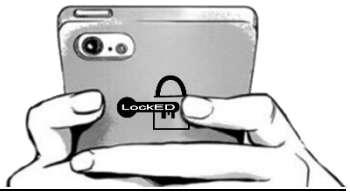
Issue 2: *Locking individual applications*

We could not find any *reliable* documentation explaining how to effectively lock individual applications on a phone. We knew it was possible, but all available code we researched was very poorly documented.

Issue 3: *Android Version*

We used the newest version of android, hoping for more features but ended up not being able to bypass the new android OS security

Version	Codename	API	Distribution
6.0	Marshmallow	23	2.3%

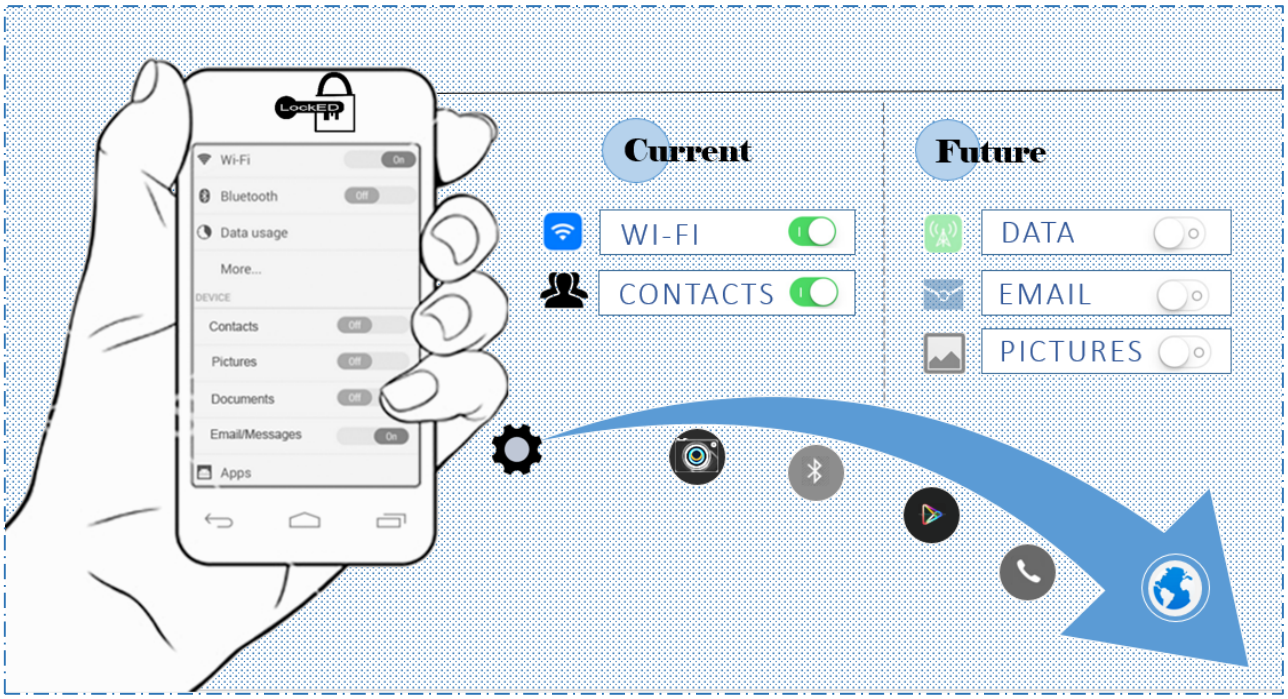


Throughout this process we learned multiple things such as:

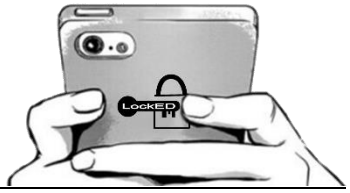
- Programming for an Android device—specifically: modifying the settings of a phone programmatically and replacing the lock screen security.
- That most available code (that is free source) from current developers for android is poorly documented.

We originally started with a very focused idea which broadened to a level we were not anticipating—leading to time being spent returning to the roots of the idea. Overall we learned that we should have spent more time researching on how to achieve the original purpose of our application—structure and locking specific applications.

For the future, we are unsure if we expect to update and continue this application post-graduation. That said, we still believe it's a useful idea and hope that the idea will reach user's mobile phones someday in the future; whether by us or others.



Present Model Figure 5



"Create an Android Lock Screen." *StackOverflow*. N.p., n.d. Web. 2016. <http://stackoverflow.com/questions/20943407/create-an-android-lock-screen>.

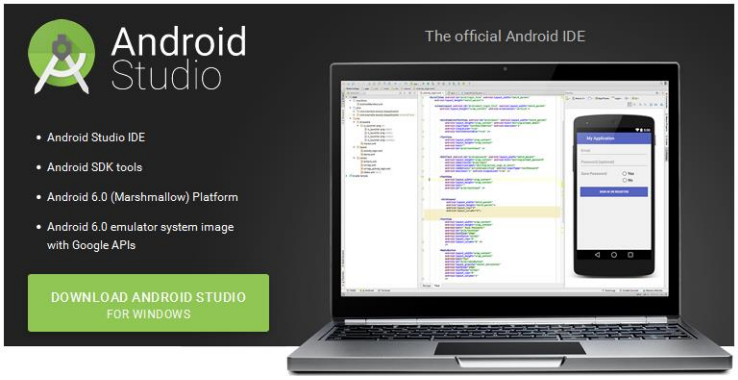
"Disable Cellular Radio in Android 4.4." *StackOverflow*. N.p., n.d. Web. 2016. <http://stackoverflow.com/questions/29595994/disable-cellular-radio-in-android-4-4>.

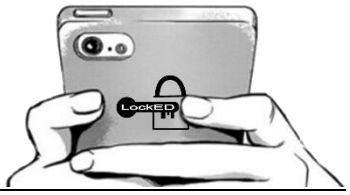
"How to programmatically turn of Wi-Fi on android device." *StackOverflow*. N.p., n.d. Web. 2016. <http://stackoverflow.com/questions/8863509/how-to-programmatically-turn-off-wifi-on-android-device>.

"How to remove a contact programmatically in Android." *StackOverflow*. N.p., n.d. Web. 2016. <http://stackoverflow.com/questions/527216/how-to-remove-a-contact-programmatically-in-android>.

“Show/hide password in a edit text view” CodeProject. N.p., n.d. Web. 2016. <http://www.codeproject.com/Tips/518641/Show-hide-password-in-a-edit-text-view-password-ty>.

"System Permissions." *Android Developers*. N.p., n.d. Web. 2016. <http://developer.android.com/guide/topics/security/permissions.html>.





TECHNICAL ARCHITECTURE DIAGRAM

•

FUNCTIONAL DIAGRAMS

•

DESIGN VERSIONS

•

ORIGINAL MOCKUP

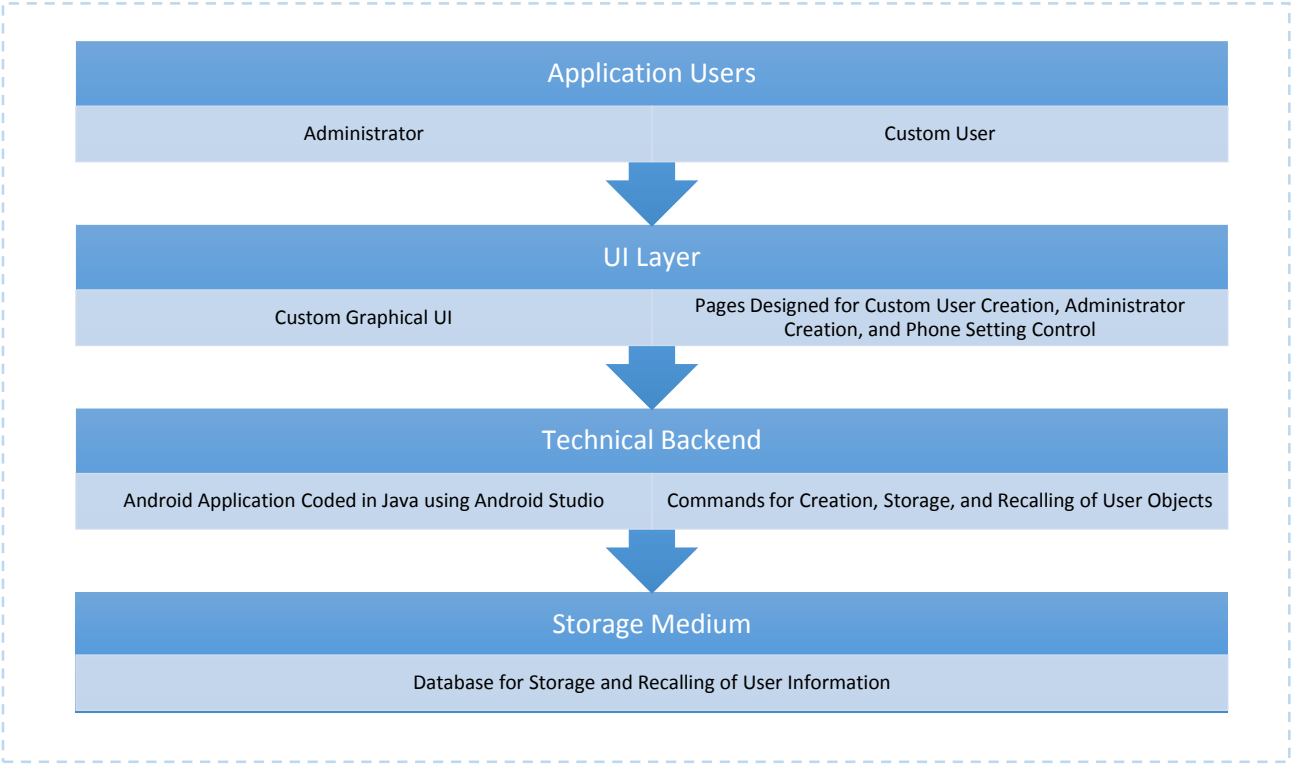
•

CODE SNIPPETS

•

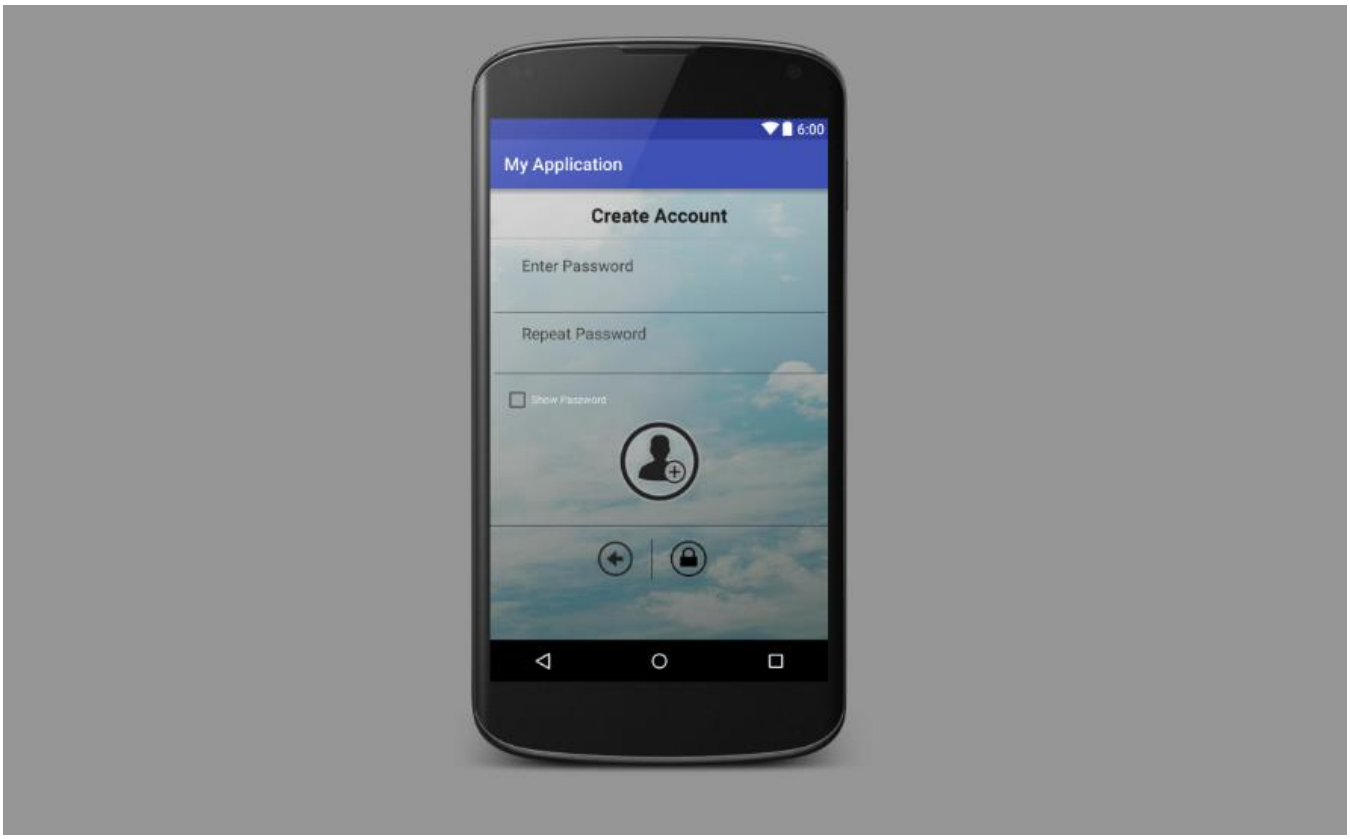
QUESTION

Technical Diagram



Functional Diagrams

Setting up Admin Account



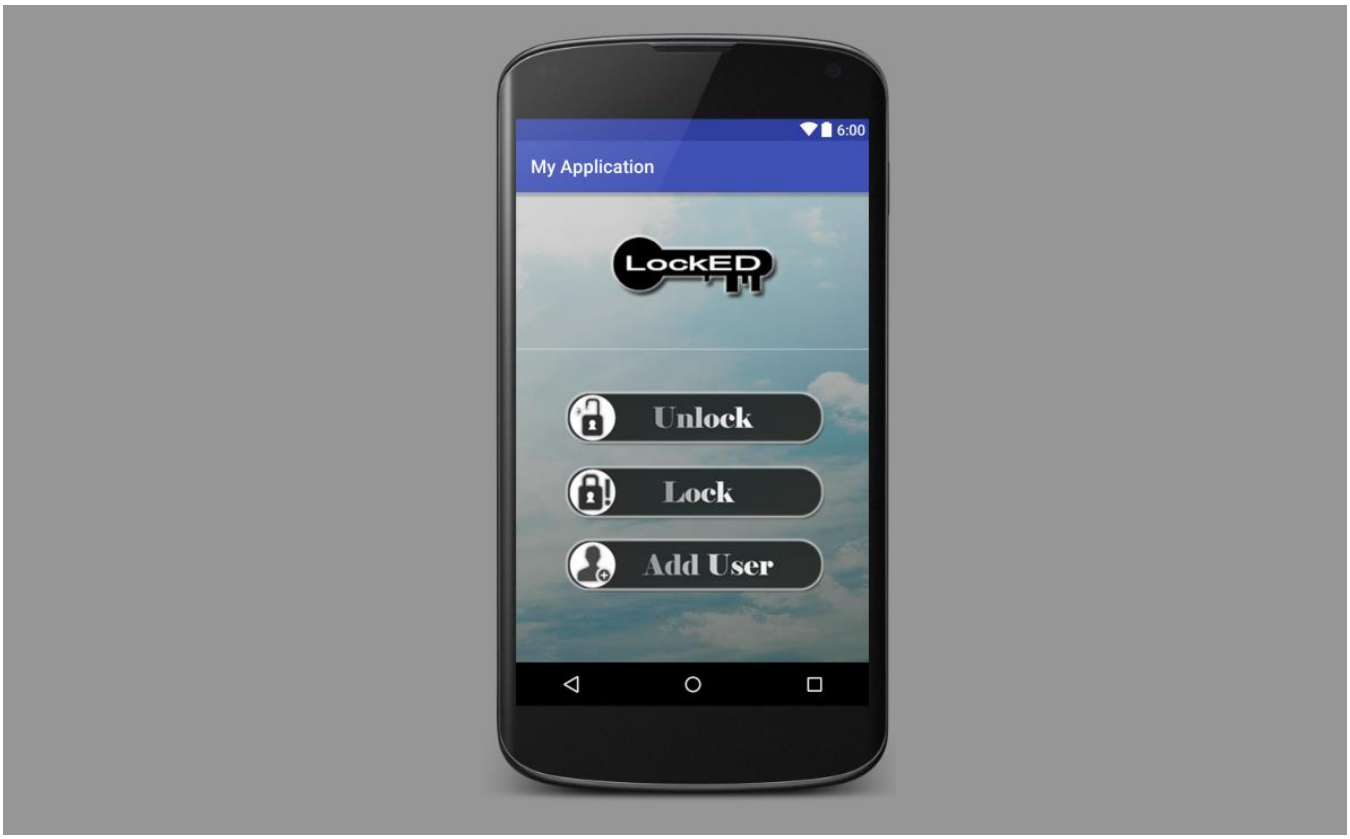
The first page of the application to add the Admin account. There are two text fields which act as password fields that the user types their desirable password into.

If user presses this button. The password fields are checked, if they match the application moves on to the main screen and adds the '**Admin**' user object to our database.

The Back button returns the view back to the main page without creating the Admin.

This button locks the screen without creating the Admin.

Main Page



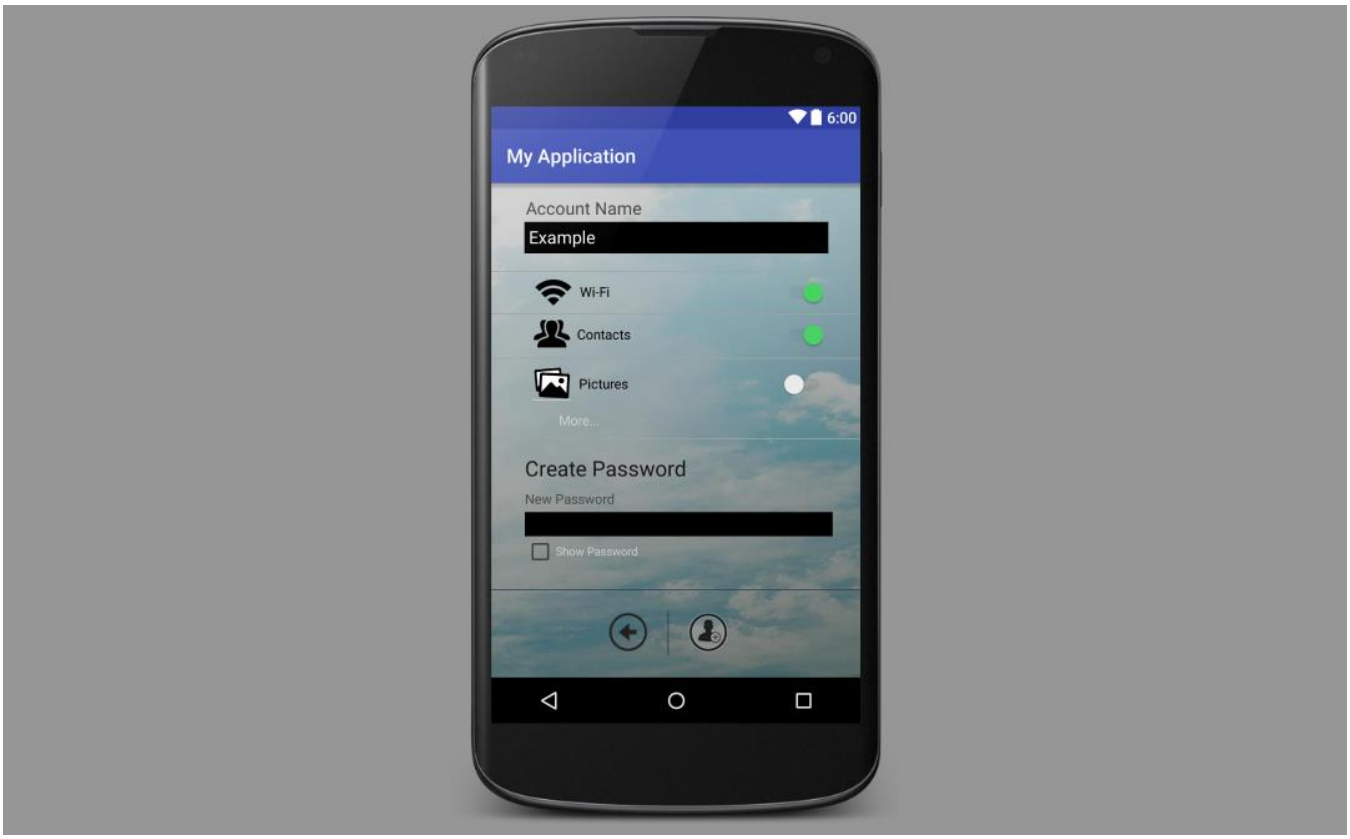
The Main page of the application

Primarily used for testing, it simulates how the actual unlocking mechanism will operate by popping up a dialog box that asks for the password. That password is checked against the database and once confirmed a message is displayed to welcome the user.

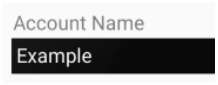
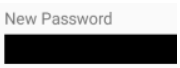
To lock the phone. For now all it does is set a variable false so that we can test our unlock system.

Takes the user to the add guest account screen.

Settings and Features



The page where the user creates '**guest**' accounts with their desired features

	Textfield for the userID. Serves as the account name.
	Are Boolean statements. By switching on or off, the user will allow these options
	Textfield for the password which will be used to access this particular account
	When pressed, it gathers all information from the text fields and creates a user object. It then returns to the main page and adds that user to the database.
	The Back button returns the view back to the main page without creating the guest account

Design Versions

Main Page

Settings and Features

Setting up Admin Account

Original Mockup

Locked LockED
LockED

LockED LockED

LockED LockED

LockED

Code Snippets

User class:

```
public class User {

        private String password;
        private boolean wifiAllowed;
        private boolean isSuperUser;
        private String username;
        private boolean removeContacts;

        public String getPassword() {
        return password;
        }

        public void setPassword(String password) {
        this.password = password;
        }

        public boolean isWifiAllowed() {
        return wifiAllowed;
```

Method for removing contacts:

```
public void removeContacts(){

        ContentResolver cr = getContentResolver();
        Cursor cur = cr.query(ContactsContract.Contacts.
CONTENT_URI, null, null, null, null);
        while (cur.moveToNext()) {
        try{

        String lookupKey = cur.getString(cur.getColumnIndex
(ContactsContract.Contacts.LOOKUP_KEY));
        Uri uri = Uri.withAppendedPath(ContactsContract.
Contacts.CONTENT_LOOKUP_URI, lookupKey);
        System.out.println("The uri is " + uri.toString());
        cr.delete(uri, null, null);
        }
        catch(Exception e)
        {
        System.out.println(e.getStackTrace());
        }
```

Create Admin Method:

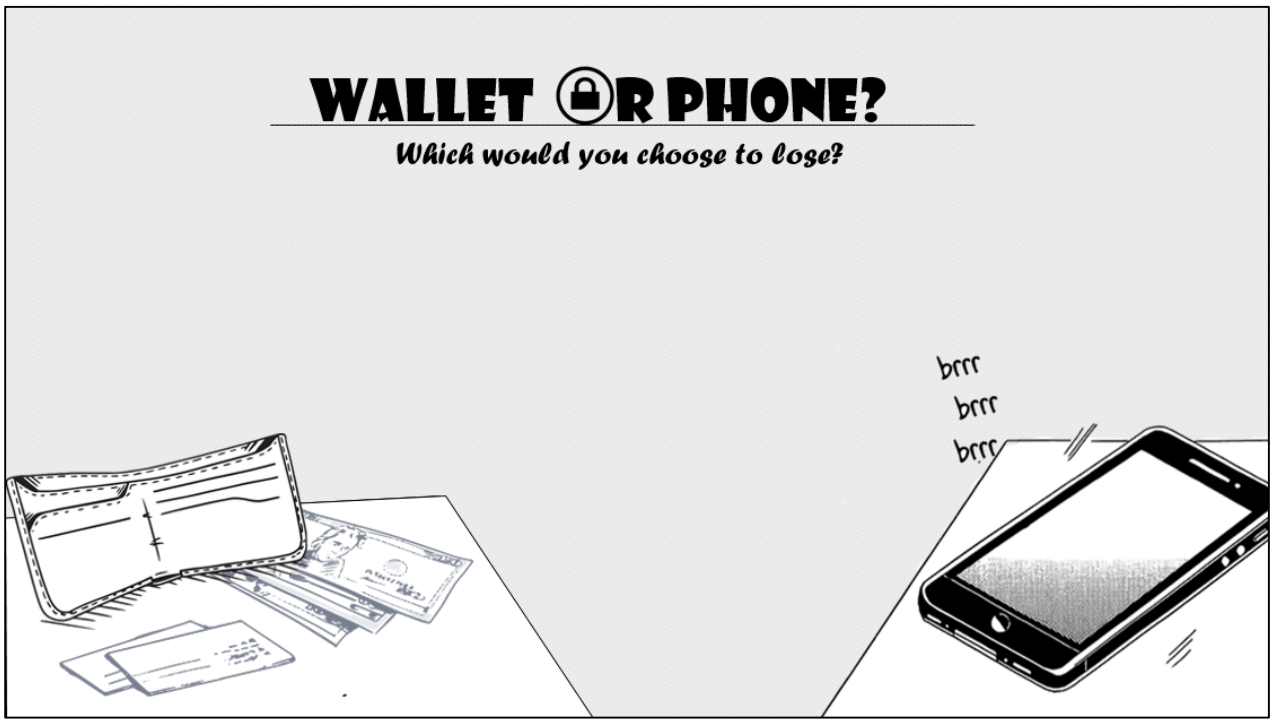
```
public void createAdmin(View v){
        //create an admin user
        EditText p1 = (EditText)findViewById(R.id.adminPass1);
        final EditText p2 = (EditText)findViewById(R.id.adminPass2);
        CheckBox showpass1 = (CheckBox)findViewById(R.id.showPass1);
        showpass1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (!isChecked) {    // show password
        p2.setTransformationMethod(PasswordTransformationMethod.getInstance());
        } else {    // hide password
        p2.setTransformationMethod(HideReturnsTransformationMethod.getInstance());
        }
```

Create custom user method:

```
        public void createUser(View v){
        EditText uname = (EditText)findViewById(R.id.userName);
        final EditText upass1 = (EditText)findViewById(R.id.UserPass);

        CheckBox showpass2 = (CheckBox)findViewById(R.id.showPass2);
        showpass2.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (!isChecked) { // show password
```


LockED


WALLET OR PHONE?
Which would you choose to lose?
brrr
brrr
brrr