Software Requirements Specification

for

UCSB 360

Version 1.2

Prepared by

Group Name: Team Epsilon

Max Hinson	4426771	maxwellhinson@gmail.com
Jhon Faghih-Nassiri	4111274	jfaghihnassiri@gmail.com
Luke Buckland	4060893	bigduker20@yahoo.com
Andy Chou	4061123	andy168chipz@gmail.com
Jicheng Huang	4088779	jicheng0903@gmail.com

Instructor:	Chandra Krintz
Course:	CS 189A
Lab Section:	Friday 12 PM
Teaching Assistant:	Stratos Dimopoulos
Date:	02/12/13

Table of Contents

Revision History

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Team Epsilon	Creating initial draft of SRS as a description of our hopes for the final product	02/08/13
1.1	Team Epsilon	Updates based on feedback from Chad and Prince	02/12/13
1.2	Team Epsilon	Updates based on feedback from Chandra	03/07/13

1 Introduction

The goal of the UCSB 360 Application is to create a social augmented reality experience. In this section you will find information regarding this document, the product, our intended audience, and various conventions.

1.1 Document Purpose

This document is a software specification that describes the UCSB 360 application and defines its functionality. It covers user interfaces, performance considerations, expected operating environment, interfaces with other software and hardware systems, as well as use cases.

1.2 Product Scope

UCSB 360 is a social augmented reality application for Android-powered mobile devices. The application will perform three main functions: view, create, and share.

Users shall be able to view augmentations on user-created targets using cloud-recognition from an external database.

Users shall be able to create content. This includes making augmentations to existing targets, as well as creating new targets.

Users shall be able to share the content they find or create with Facebook friends and friends within the application.

These functions will create a social experience that allows users to interact with others in ways they never have before.

1.3 Intended Audience and Document Overview

This document is intended to formally convey the application's functionality for both users and the development team. For application users, it will be used as a specification of the software that will be delivered. For the development team, it will act as a guideline for how the software will be designed, as well as a basis for measuring progress. Finally, it will be used as a basis for unit testing and verification.

1.4 Definitions, Acronyms and Abbreviations

AWS - Amazon Web Services cloud-based solutions
ER Diagram - A model used to express relationships between data in a database
JDBC - Java DataBase Connection is used to connect to external databases
RDS - AWS's Relational Database Service that hosts a MySQL database
S3 - AWS's Simple Storage Service that hosts augmentation files
Vuforia - Qualcomm's augmented reality SDK that handles target recognition and tracking.
Vuforia Cloud Database - Qualcomm's cloud database that hosts image targets

2 Overall Description

2.1 Product Perspective

The UCSB 360 application will be a brand new Android based application. It will be implemented with Qualcomm's Vuforia SDK.
See figure 11 and 12 for more information.

2.2 Product Functionality

The product should have an easy to use main menu, from which the rest of the features or functions can be accessed. These main features as listed are: graffiti generating, basic viewing and Facebook login. See Figures 1 through 9 for a reference to the intended user interfaces for many of the pages. Also see Figure 10, the Flow Chart Diagram.

2.3 Users and Characteristics

Our application hopes to draw on three main user groups. The first, and by far the largest, is the group of UCSB students who attend classes on campus. These students often walk from location to location around campus, and almost all of them have smartphones. About half of those smartphones are Android, and more than half of those Android phones are fit to run UCSB 360. These users are obsessed with social media, and almost all of them actively uses at least one of the following: Facebook, Twitter, Instagram, Snapchat. Many of them are involved in some sort of club or organization, and spend time recruiting, fundraising, or trying to gain exposure. Many have meal plans.

The second group of users of UCSB are visitor to the campus. This group can consist of parents, prospective students, professionals, recruiters, or speakers. They are not familiar with many of the campus landmarks and buildings. Members of this group may be interested in learning more about campus history, or may need to navigate from one location to another on campus.

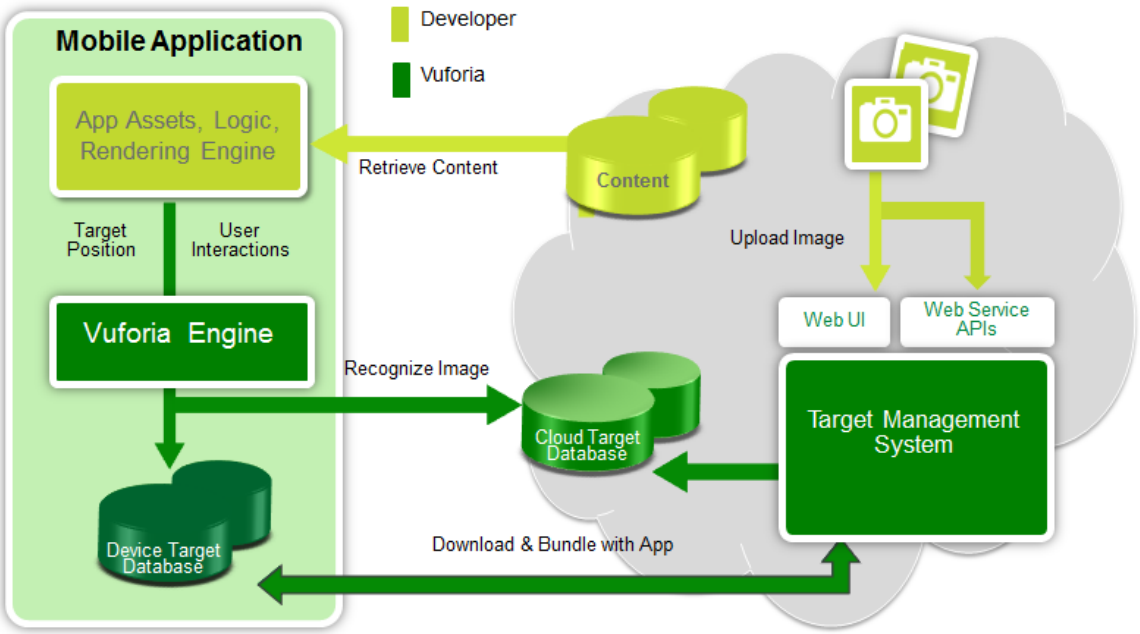
The final prospective group of users for the app are prospective advertisers who hope to gain exposure or sales for a commercial product or service. This group of users is most likely to be a restaurant in Isla Vista, in the UCEN, or at the Arbor. This group of users

would also include local companies hoping to recruit.

The group that is probably going to be using the app most, and thus the most important group, is the students. They will be on campus all the time able to use the app, as opposed to visitors and advertisers, who will probably only the use app while they are on campus for a limited time. Thus the features focused on the students such as graffiti and graffiti sharing will be what is focused on.

2.4 Operating Environment

The UCSB 360 application will only be available for the Android operating systems. The application shall only be used with compatible android devices, please read the hardware section for specific details. The user shall use this application on Android OS 2.2(API 8) or any later versions of the Android OS. The following diagram shows how the application shall interact with the Vuforia API and cloud database.



2.5 Design and Implementation Constraints

Implementation language restrictions

The programming language shall be Java for the main application.

The programming language shall be SQL for the cloud targets database.

Resource limits

The users' device shall have a working camera.
The users' device shall have a working data plan or wifi connection.
The users' device shall have sufficient memory storage to install the application.
The users' device shall have sufficient battery life to run the application.

Licensing

The developers shall use The MIT License(MIT).

Security Considerations

The user's login password shall be encrypted with OAuth 2.0 Protocol then stored on the device.

2.6 User Documentation

Tutorial

The user shall be able to see a tutorial the first time they start the application
The tutorial will be a series of splash screens that illustrates how to use each feature
This tutorial will not automatically show more than one time
There will always be a "help" button available in the menu if user needs revisit the tutorial

Online Help

There shall be contact information posted in "about" page
The user shall be able to contact the developers regarding any issues they encounter

2.7 Assumptions and Dependencies

Dependencies

The application shall be used with the assumption that the Android API and licensing agreement remains the same
The application shall be used with the assumption that the Facebook API remains the same
The application shall be used with the assumption that the Vuforia SDK and documentation are correct

Software Component Dependencies

The application shall integrate with Facebook's Android application
The application shall be used with the assumption that the built in camera application operates correctly

The application shall be used with the assumption that the device’s network interface card and driver are operating correctly

3 External Interfaces

3.1 External Interface Requirements

3.1.1 User Interfaces

The user shall interact with the application via a graphical user interface (GUI). The GUI shall consist of a menu structure and pop-up messages. The user will traverse the menu structure to access most features. User interface samples can be found in Appendix A (Figures 1-9). By default, the user will be in View mode. View mode allows the user to see the augmented environment around them through the device's camera. This mode shows three buttons: Graffiti mode, Share mode, and Menu.

Pressing the Menu button pulls the menu down from the top of the screen. The menu options are "Poster", "Friends", "Help", and "Logout". The menu structures for these items will be added later.

When the user selects Graffiti mode, the application will take a picture with the camera. As the still photo remains on the screen, the user will be presented with a selection of brushes and colors to edit the photo with. After the user is finished editing, he will be given the option to share photos via his Facebook wall or with his friends in the application. Finishing this step results in being taken back to View mode.

At any point in View mode, the user can press the Share button. The application uses the camera to take a picture. The user will then be prompted to share the photo via Facebook or with friends in the application. Finishing this step results in being taken back to View mode.

3.1.2 Hardware Interfaces

The UCSB 360 application shall function on mobile devices running the Android operating system. The application shall make extensive use of the device's camera to take pictures of the user's surroundings. It will also use the device's screen capture utility. Additionally, the application may utilize the device's GPS location data for tagging image targets. Finally, the application shall make use of the device's physical buttons, such as "back", "menu", and "home". All interactions with the hardware will be made through API calls to the Android SDK.

3.1.3 Software Interfaces

The UCSB 360 application shall make use of several different software interfaces to interact with external software systems.

Android - Android SDK (version 8)
The application will function on mobile phones and tablets running at least version 8 of the Android API.

Vuforia Cloud Database - Vuforia SDK (version 2.0.34)
The application will store and retrieve image targets from Qualcomm's Vuforia cloud database. Interactions with this cloud database will be made via function calls to the Vuforia SDK. Upon successful target recognition, the cloud database shall return a target identification number. This identifier will be used to retrieve that target's information from our MySQL database.

Amazon Web Services RDS - JDBC Connector/J Driver (version 5.1.23)
The application shall store and query user, target, and augmentation information in a MySQL database hosted on Amazon Web Services RDS. Queries will be made using Java Database Connection (JDBC) with the Connector/J driver for MySQL databases. For information that will be stored in this database, see the ER diagram in Appendix A (Figure 0).

Amazon Web Services S3 - Amazon S3 SDK (version 1.4.5)
The application shall store and retrieve augmentations from a virtual file system on Amazon Web Services S3. Augmentations will be retrieved via public URLs. In order to store new augmentations, the application must make use of the Amazon S3 SDK.

Facebook - Facebook SDK (version 3.0)
The application shall also make use of the Facebook API. Users will be prompted to login with their Facebook account. The application will then retrieve basic information about the user, including a list of their friends. Finally, users will be able to share augmentations via Facebook.

3.1.4 Communications Interfaces

The application shall communicate with the various databases and software services via API function calls. Because the application will be written in Java, Java functions will make these calls to the APIs. The exact formats and protocols for incoming and outgoing messages should be abstracted by the APIs.

3.2 Functional Requirements

Graffiti tagging

- Users can view an augmented world in real time through their camera, which shows the graffiti augmentations created by other users
- Allows users to find buildings, objects, and locations and then turn them into targets,

which are entities that can be recognized and then augmented via the Vuforia SDK
- Allows users to augment targets with their own drawings. This graffiti is then added to the augmented world

Facebook integration

- Users can log in via their Facebook account to create and share content
- Users may add friends ad hoc from menu, but also while selecting graffiti from View mode
- Allows users to post photos (or screenshots) of augmented world quickly and easily to their Facebook wall
- Allows users to post their newly made creation to their Facebook wall

View mode

- View mode should allow user to see augmented world through the camera
- Easy access to Graffiti mode, where user can create and add their own graffiti creations to world
- As well as graffiti, other augmentations should be viewable
- Building information should be displayed when camera is aimed at certain targets; dining common menus, building hours, historical and basic information about building
- Posters and ads from student groups and companies should be viewable

4 Other Non-functional Requirements

4.1 Performance Requirements

Speed

- The user shall be able to upload a target to the cloud database within 4 seconds.
- The user shall be able to locate a target from the device database within 2 seconds
- The user shall be able to locate a target from cloud database within 4 seconds
- The user shall be able to upload his graffiti to cloud within 10 seconds
- The system shall be able to render all graffiti on the screen with in 4 seconds

Response time

- After clicking any button, user shall receive a response within 1 second
- Graffiti
    - Application shall enter Graffiti mode no longer than 1 second after the user presses the Graffiti button
    - User shall be able to see his text in input window simultaneously
    - Delay time of recognizing user finger path shall be less than 1 second
    - User shall be able to see the position of his graffiti on the target according to his finger's position with a delay no more than 1 second.
- Advertising
    - After the user clicks the advertising button, the upload screen shall show up instantly

4.2 Safety and Security Requirements

- User needs to sign in with their facebook account to prove their identity before using.
- User shall not use our application while driving or biking.

4.3 Software Quality Attributes

- Adaptability
    - Any smartphone running an android system above version 2.2 shall be able to run our application
    - Any user with smartphone or tablet experience shall be able to use our application.
    - Learning curve for this app shall be very low since everything is one click access
- Availability
    - Maintain should be done on a daily bases, any inappropriate graffiti shall be removed as soon as possible.
    - Database shall be available to user 24 hours per day
    - Download and install application shall be available through Google App store.
- Testability
    - All available functionality shall be able to test separately from the application
    - User Interface shall be manually fully tested
    - Method within database section shall be fully tested by JUnit test
    - All available functionality shall be fully tested before release
- Portability
    - Not available on any platform other than Android

5 User Stories

5.1 Facebook Stories

1. As a UCSB student, I want to sign in with my facebook account because I don't like making and managing many accounts.
    a. Scenario: Login
        i. *Given* the user hasn't signed into the app yet
        ii. *And* has a facebook account
        iii. *When* the user tries to use the app
        iv. *Then* they will be prompted to login using facebook credentials
        v. *And* only have to do this once
        vi. *And* be told exactly what information we will be accessing
2. As a UCSB student, I want to share content I create, or find, to my facebook wall because I want to connect with my friends.
    a. Scenario: Create Post
        i. *Given* the user has just generated or found graffiti content
        ii. *And* wants to share it with facebook
        iii. *When* the user presses the "share to facebook" button
        iv. *Then* they will be taken to a generated post to facebook
        v. *And* be asked if they want to add a message, then post
3. As a UCSB student, I want to add my Facebook friends in the app because I want to share my content privately.
    a. Scenario: Create Share
        i. *Given* the user has just generated or found graffiti content
        ii. *And* wants to share it with certain friends
        iii. *When* the user presses the "share to friends" button
        iv. *Then* they will be prompted to choose which friends
        v. *And* be asked if they want to add a message, then send
4. As a UCSB student, I want the application to allow me to stay logged in even if I make the app crash.
    a. Scenario: App Crashes
        i. *Given* the user has crashed the app
        ii. *When* the user launches it again
        iii. *Then* they will bypass the login screen
        iv. *And* already be logged in

5.2 Graffiti Stories

1. As a UCSB student, I want to see student-made graffiti on campus because I want to see

interesting content during my down time and between classes.

a. Scenario: View Content
    i. *Given* the user is walking around the UCSB campus
    ii. *And* has their mobile phone
    iii. *And* is logged in
    iv. *When* the user points their phone at recognizable targets
    v. *Then* community-generated graffiti content is rendered
    vi. *And* is available in under four seconds
b. Scenario: View Content Crashes
    i. *Given* the users app crashed while they were using the viewer
    ii. *When* the user re-launches the app
    iii. *Then* any cached information, images, etc is lost
c. Scenario: Not Logged In
    i. *Given* the user is not logged in
    ii. *When* the user uses the camera view button to skip login
    iii. *Then* the user is shown the camera view
    iv. *And* any buttons related to friends, facebook, graffiti creation, posters, and sharing are grayed out

2. As a UCSB student, I want to create and submit my graffiti because I want to express myself in the digitally augmented world.
    a. Scenario: Create Content Basic
        i. *Given* the user is walking around the UCSB campus
        ii. *And* has their mobile phone
        iii. *And* is logged in
        iv. *And* has found a valid target to tag content on
        v. *When* the user selects the area
        vi. *Then* they can generate the content
        vii. *And* when they are done, they can save it to the AR world
    b. Scenario: Create Content Tools
        i. *Given* the user can create content (above)
        ii. *When* the user is creating content
        iii. *Then* they have a wide array of tools available to them
    c. Scenario: Create Content Time
        i. *Given* the user can create content (above)
        ii. *When* the user is done creating content
        iii. *Then* the size of the data to be uploaded from their device is minimal
    d. Scenario: Create Content Interups
        i. *Given* at some point create content (above) crashes
        ii. *When* the user launches the app again
        iii. *Then* they are logged in and taken to the camera view
        iv. *And* all of the previous steps they took are lost

5.3 Poster Stories

1. As a UCSB student, I want to see posters of clubs and events because I want to be up to date with campus activities.
    a. Scenario: View Posters
        i. *Given* the user is walking around the UCSB campus
        ii. *And* has their mobile phone
        iii. *And* is logged in
        iv. *When* the user points their phone at recognizable targets
        v. *Then* poster images are rendered as part of the AR world
        vi. *And* are rendered in under four seconds
2. As an approved user at UCSB, I want to put up posters of my organizations meetings and events because I want to increase my exposure.
    a. Scenario: Create Poster
        i. *Given* the user is on the UCSB campus
        ii. *And* has their mobile phone
        iii. *And* is logged in
        iv. *And* has approved poster privileges
        v. *When* the user points their phone at recognizable targets
        vi. *Then* they can select the area they want their poster displayed
        vii. *And* can upload the image for the poster right from their phone
    b. Scenario: Approved User Application
        i. *Given* the user has the desire to be an approved user
        ii. *When* the user submits a request through the app
        iii. *And the request is processed by the app owners*
        iv. *Then* they receive an email stating they have been approved
        v. *And* the poster button is not grayed out for them
    c. Scenario: Attempt to Post but Unapproved
        i. *Given* the user is not an approved user
        ii. *When the user pulls down the menu*
        iii. *Then* the poster button is grayed out for them

5.4 Future Feature Stories

5.4.1 Building Info Stories

1. As a visitor to the UCSB campus, I want to see hours, history, and information of the buildings I look at, because I'd like to learn more about the campus.
    a. Scenario: View Building Info
        i. *Given* the user is a visitor to the UCSB campus
        ii. *And* has their mobile phone
        iii. *And* is logged in

iv. *When* the user points their phone at recognizable targets
v. *Then* buttons to reach hours, history, and info are displayed
vi. *And* can be clicked on to expand into more information

2. As a UCSB student, I want to see events going on at a building because I want to conveniently enrich my campus experience
    a. Scenario: View Event Info
        i. *Given* the user is on the UCSB campus
        ii. *And* has their mobile phone
        iii. *And* is logged in
        iv. *When* the user points their phone at recognizable targets
        v. *Then* a button to reach event info is displayed
        vi. *And* can be clicked on to expand into more information
3. As a UCSB student, I want to see all the dining common menus when I point my app at any of them, because I want to easily decide where to eat.
    a. Scenario: View Menu Info
        i. *Given* the user is on the UCSB campus
        ii. *And* has their mobile phone
        iii. *And* is logged in
        iv. *And* has a meal plan
        v. *When* the user points their phone at a dining common
        vi. *Then* the menus for all the dining commons are displayed
        vii. *And* are rendered in under four seconds

5.4.2 Advertising Stories

1. As a local advertiser, I want to put up virtual ads because it is cost effective and extremely targeted.
    a. Scenario: Post Advertising Info - App
        i. *Given* the advertiser has an Android phone and web access
        ii. *And* has their mobile phone
        iii. *And* is logged in
        iv. *And* is aware of our ad program
        v. *When* the user wants to put up one of their ads
        vi. *Then* they submit a request to our team
        vii. *And* upload their ad image
        viii. *And* list their preferred location, run length
        ix. *And* can expect to be contacted with an estimate within 2 days
2. As a local advertiser, I want to put up my virtual ads from the web, because I don't necessarily have an Android phone or know how to use the app store.
    a. Scenario: Post Advertising Info - Web
        i. *Given* the advertiser has web access
        ii. *And* is logged into our website
        iii. *When* the advertiser wants to put up one of their ads

iv. *Then* they submit a request to our team
v. *And* upload their ad image
vi. *And* list their preferred location, run length
vii. *And* can expect to be contacted with an estimate within 2 days

Appendix A - Images

Figure 0 (below): Database ER Diagram

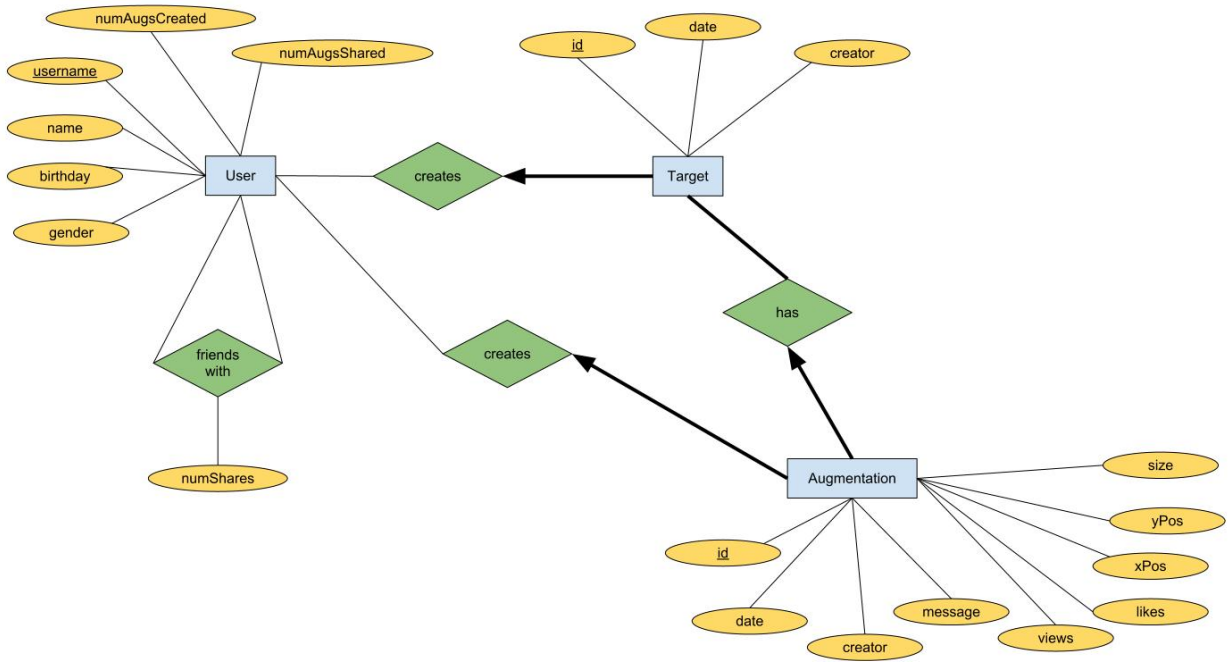


Figure 1(below): User Login Page

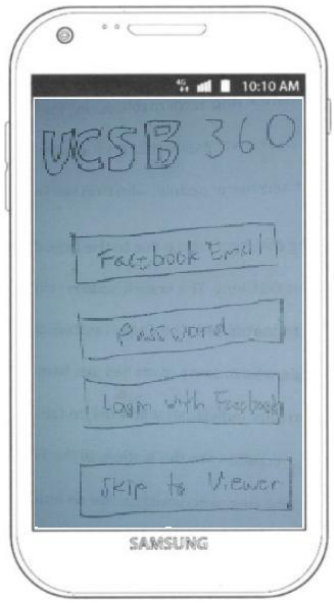


Figure 2: Main Menu Page
Note: Camera is background

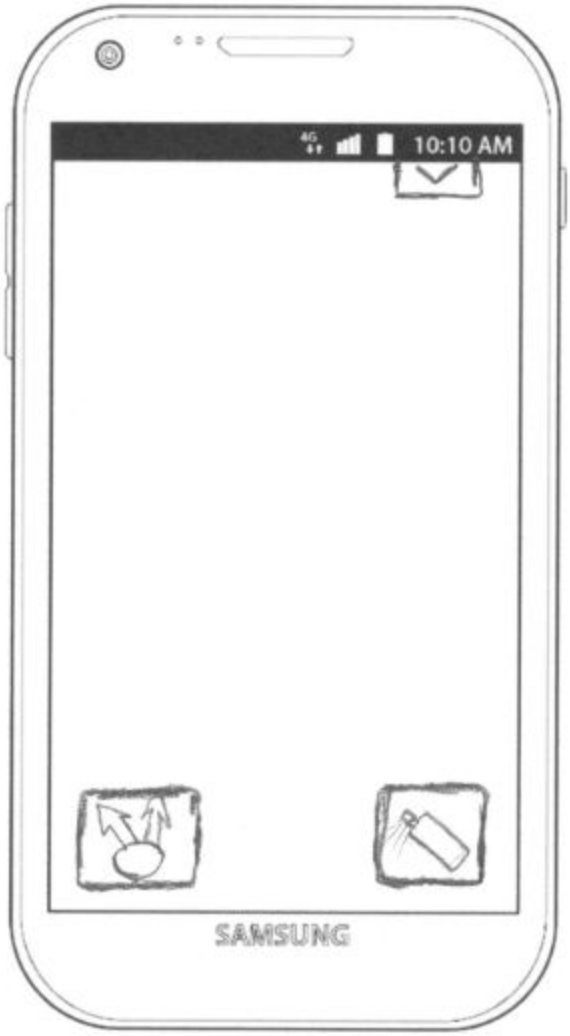


Figure 3: Menu Open Page
Note: Camera is background

Figure 4: Target Identification Page
Note: Camera is background

Figure 5: Target Error Page
Note: Camera is background

Figure 6: Graffiti Tools

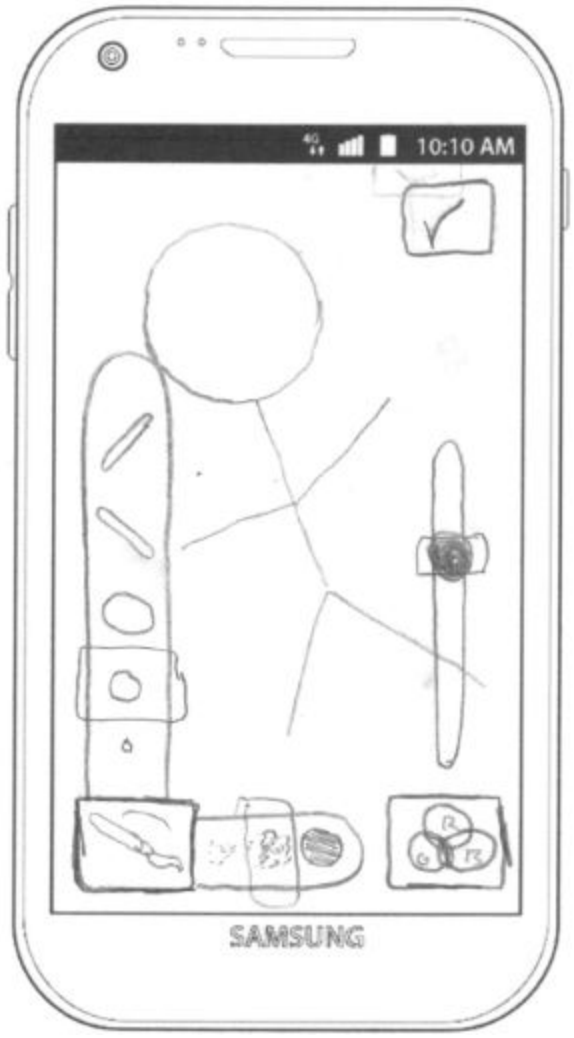


Figure 7: Graffiti Upload

Figure 8: Graffiti Share Page

Figure 9: Main Share Page

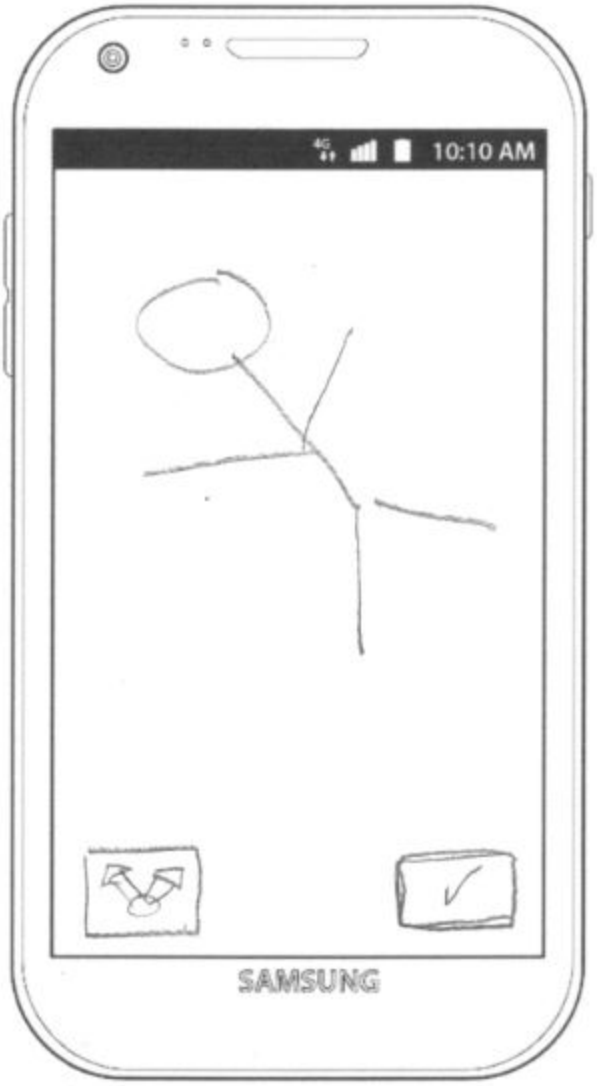


Figure 10: Flow Chart Diagram

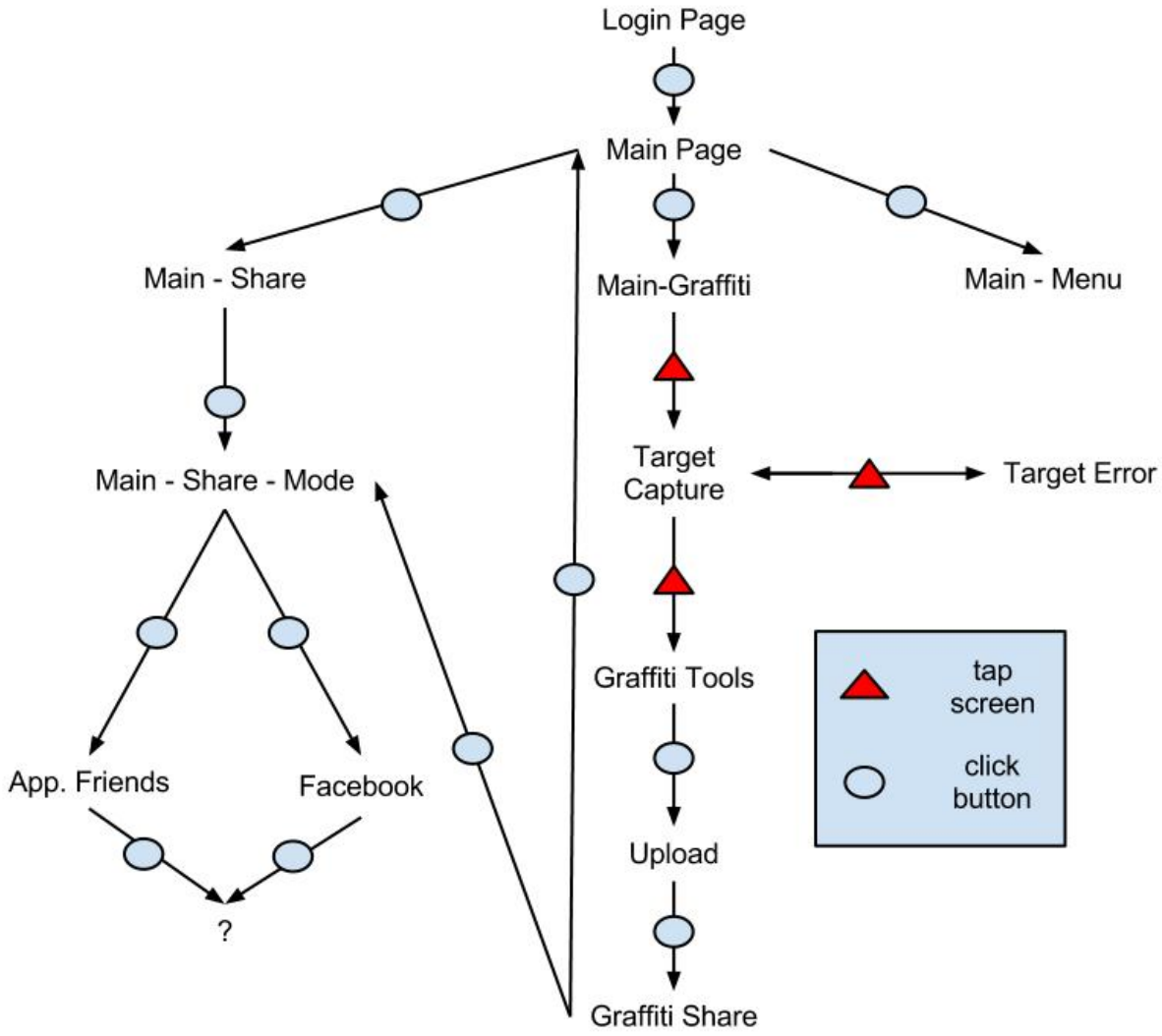


Figure 11: Software Stack

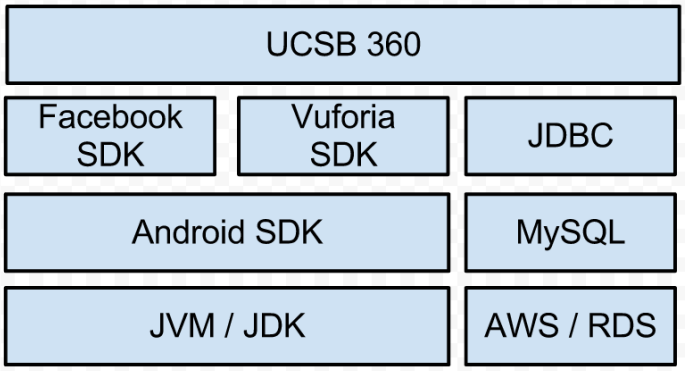


Figure 12: Relational Diagram