**MoCCCA Specialization Capstone Project Overview and Requirements**

## Overview

The *Mobile Cloud Computing with Android* (MoCCA) Specialization comprises three content area MOOCs plus a final Capstone MOOC. Projects in the Capstone MOOC give Signature Track students who passed all previous MOOCs in the MoCCA Specialization “with Distinction” an opportunity to integrate and demonstrate the knowledge they've acquired across other MOOCs. To make the MoCCA Capstone MOOC as effective as possible, we are soliciting input from students, academics, and practitioners to help specify relevant Capstone projects. A successful Capstone project will involve building a mobile app that will leverage services running in a computing cloud accessed over the Internet to support multiple users.

The MoCCA Capstone MOOC will offer students multiple projects from which to choose. Each project will be described in a lightweight project specification. Each student will select one project to implement over roughly five weeks working roughly 8-12 hours per week (though students are free to devote as much time as they’d like to the project). Some project specifications will be defined by commercial organizations, which will also provide some nominal engineering support, e.g., by holding virtual office hours to answer project questions or by acting as reviewers for some project implementations.

As with the other MOOCs in the MoCCA Specialization, students can post questions about their project to project-specific sub-forums accessible from the Capstone course's online forum pages. The instructors and members of the MoCCA community will provide feedback on the questions as time permits.

## Basic Project Requirements

As mentioned above, potential projects must support multiple users and should leverage services running in the cloud. Each project's specification clearly outlines the app's intended high-level behavior, yet leaves substantial room for individual creativity. Students will need to flesh out many important design and implementation details. Requirements for all project specificiations in the Capstone MOOC include the following:

* Apps must support multiple users via individual user accounts. At least one user facing operation must be available only to authenticated users.
* App implementations must comprise at least one instance of at least two of the following four fundamental Android components: Activity, BroadcastReceiver, Service and ContentProvider.
* Apps must interact over the network via HTTP, with at least one remotely-hosted Java Spring-based service.
* At runtime apps must allow users to navigate between at least three different user interface screens.
  + e.g., a hypothetical email reader app might have multiple screens, such as (1) a ListView showing all emails, (2) a detail View showing a single email, (3) a compose view for creating new emails, and (4) a Settings view for providing information about the user's email account.
* Apps must use at least one advanced capability or API from the following list covered in the MoCCA Specialization: multimedia capture, multimedia playback, touch gestures, sensors, or animation. Experienced students are welcome to use other advanced capabilities not covered in the specialization, such as BlueTooth or Wifi-Direct networking, push notifications, or search. Moreover, projects specified by commercial organizations may require the use of additional organization-specific APIs or features not covered in the MoCCA Specialization. In these cases, relevant material will be provided by the specifying organization.
* Apps must support at least one operation that is performed off the UI Thread in one or more background Threads or a Thread pool.
* There may also be additional project-specific requirements (e.g., required use of a particular project-specific API or service).

## Capstone Execution

The Capstone will start the week of Oct. 5th, runs five weeks, and includes two Peer Assessment activities. The first Peer Assessment will occur the week of Oct. 19th, with the goal of ensuring that each student has an acceptable project design and implementation plan. The second Peer Assessment will start the week of Nov. 16th, with the goal of ensuring that each student has a working project, which they will demonstrate in several ways outlined below. Each student will select one Capstone project specification to implement from the list provided by the instructors.

**Peer Assessment #1**

At the beginning of the Capstone MOOC, the instructors will provide a rubric listing project requirements that will be evaluated by 5 Assessors using the Coursera Peer Assessment system. Students will submit an assessment package containing design documentation explaining how they will implement their selected specification. The assessment package will contain text and/or graphics explaining how the student's project implementation will meet each requirement listed in the rubric. The majority of Peer Assessors (i.e., 3 of 5 Assessors per project) must agree that the intended implementation meets each requirement listed in the rubric. This assessment will be provided as feedback to the student. Students whose projects do *not* meet each requirement must address the shortfall on their own. Failure to do so will result in them not passing the course, i.e., students failing Peer Assessment #1 proceed in the Capstone at their own risk.

**Peer Assessment #2.**

Each student will submit a final project deliverable containing the following:

* Design documentation explaining in detail how they implemented their selected specification. This documentation should be a refinement of their deliverable for Peer Assessment #1 described above.
* A screencast video of the working project.
* A Virtual Box virtual machine or URL to a remote server and accompanying documentation enabling the assessor to test the app's Internet-accessible service.
* The complete project source code and associated files needed to build the project. Note that all software will ultimately need to be released in open-source form to facilitate peer assessment. Students are responsible for not including proprietary software in their solutions. Students who aren't willing to release their solutions in open-source form can still participate in the Capstone project, but their solutions won't be evaluated and they won't receive credit for passing the Capstone.
* A completed webpage template that will be used to advertise the implemented project in a Capstone gallery.

The majority (i.e., 3 of 5) of Assessors must agree that the submission meets each requirement. Students whose projects don't meet this criterion will not pass the course.

**Expectations for Students**

To complete the Capstone project all students *must* have access to the following:

* Screencast software (e.g., see http://mashable.com/2008/02/21/screencasting-video-tutorials).
* For some projects students may need to have a webcam or microphone.
* A publicly-accessible website to upload their screencast (e.g., YouTube or some Coursera-managed site)
* A working Virtual Box 4.3.12 installation or a public URL for accessing the HTTP-based API of the cloud service hosted on infrastructure, such as Google AppEngine or Amazon EC2.
* A non-public repository for developing their solution so they don’t violate the Coursera Code of Conduct by allowing other students to copy their solutions.

**Outreach and Publicity**

Each student's project will be publicized in a Capstone gallery web site. We envision creating a web page template fashioned on existing app stores displaying key project information and an embedded video of the project. Instructors, students, and possibly the general public will be able to vote for their favorite projects. We will use this information as input when determining prize awards, possibly in conjunction with participating companies.

The following is an example specification for a project that will be used in the dry run of the Capstone project. Some variant of this specification may also be included in the full list of potential projects for the actual running of the Capstone project.

## Potlatch - A Sample Capstone Project

“Happiness is a gift and the trick is not to expect it, but to delight in it when it comes.”

- [Charles Dickens](http://www.goodreads.com/author/show/239579.Charles_Dickens), [Nicholas Nickleby](http://www.goodreads.com/work/quotes/4993095)

See a sunset that you just can't keep to yourself; a view that takes your breath away, an image that touches you personally? Then snap a picture, add some poetry to convey your thoughts, and then share it with Potlatch[[1]](#footnote-1).

**Functional Description and App Requirements:**

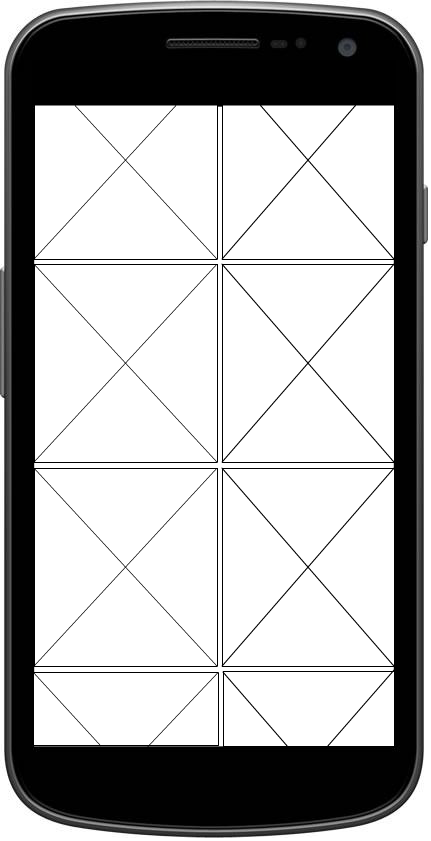
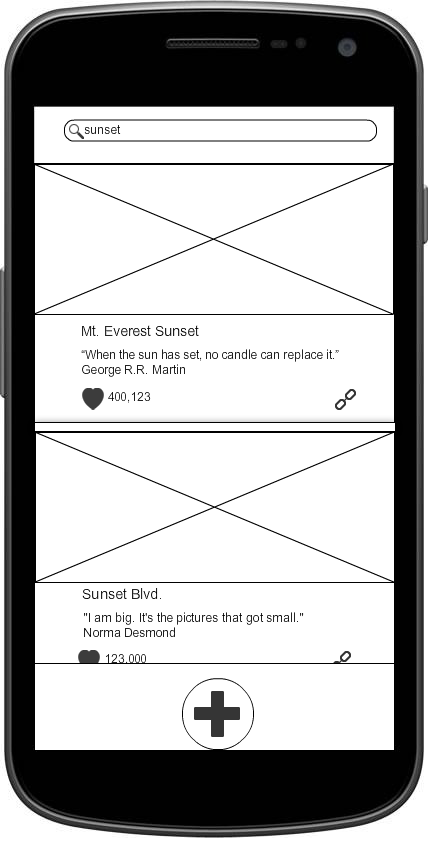
1. A *Gift* is a unit of data containing an image, a title, and optional accompanying text.
2. A *User* can create a Gift by taking a picture (or optionally by selecting an image already stored on the device), entering a title, and optionally typing in accompanying text.
3. Once the Gift is complete the User can post the Gift to a *Gift Chain* (which is one or more related Gifts). Gift data is stored to and retrieved from a web-based service accessible in the cloud. The post operation requires an authenticated user account.
4. Users can view Gifts that have been posted.
5. Users can do text searches for Gifts. The search is performed only on the Gift's title. Gifts matching the search criterion are returned for user viewing.
6. Users can indicate that they were *touched* by a Gift and can also flag Gifts as being obscene or inappropriate. Users can set a preference that prevents the display of Gifts flagged as obscene or inappropriate.
7. Touched counts are displayed with each Gift. These counts are periodically updated in accordance with a user-specified preference (e.g., Touched counts are updated every 1, 5 or 60 minutes).

**Implementation Considerations**

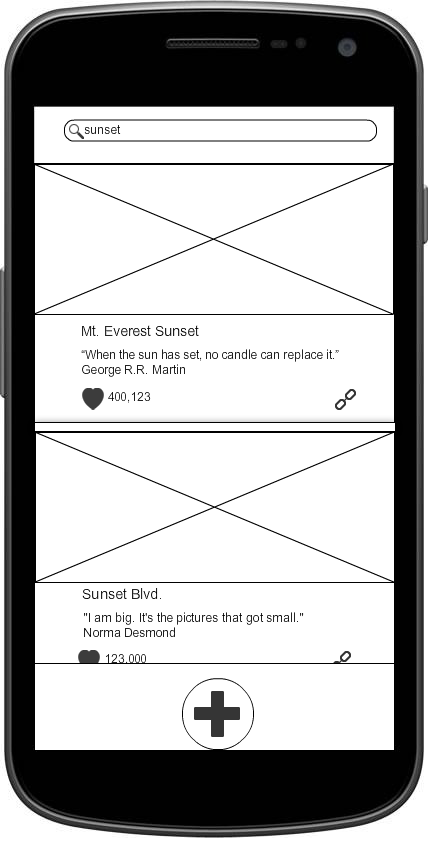
The Potlatch project specification outlined above is intentionally underspecified to maximize opportunities for student creativity. Students must therefore consider and choose between a number of design and implementation issues and solution alternatives to produce their final product solution. For example, students should consider at least the following issues for this project:

* How will Gift data be stored? In your device? In the remote service?
* What will the user interface screens look like? How will the user navigate between different screens?
* How and where will users attach Gifts to a Gift Chain?
* How, when, and how often will the user enter their user account information? For example, will the user enter this information each time they run the app? Will they specify the information as part of a preference screen?
* How will Gifts and Gift Chains be accessed? Will there be some initial default display? Will users have to enter search criteria? If so, how will the user get access to the search interface?
* Will all search results be displayed at the same time? Or will only a subset of Gifts be shown at any one time? Will the Gifts be sorted in some way, such as by their Touched counts, creation time, or latest update time?
* How will users indicate that they were touched by a Gift? Can they undo their decisions? How will they flag inappropriate or obscene Gifts?
* What user preferences can the user set? How will the app be informed of changes to these user preferences?
* How will the app handle concurrency issues, such as how will periodic updates occur - via server push or app pull? How will search queries and results be efficiently processed? Will the data be pulled from the server in multiple requests or all at one time? Will the server data include full-sized images, or thumbnails plus URLs pointing to the full-sized images?
* Will the app cache information on the local device, e.g., in a Content Provider?
* Will the app provide extensions and improvements that goes beyond the minimum requirements? For example, if an app collects location information for each Gift, queries by location, and then uses Maps to display Gifts at their locations, then it may be necessary to modify the various app databases and query facilities. Also, using Maps may require students to have access to an actual device. Also, how will these enhancements affect the rest of the app?
* Does your app idea really require two or more fundamental Android components? If so, which ones? For example, this app might benefit from using a ContentProvider or from using a background Service that synchronizes local and remote image data, only when the device is connected to a WiFi network.

**Wire Frames**

Some mocked-up screenshots suggesting what an implementation of this app might look like.

Click on chain icon to see entire Gift Chain

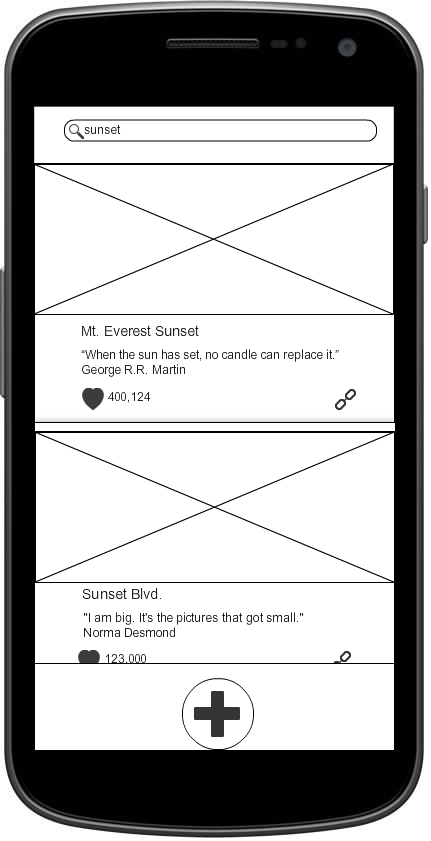
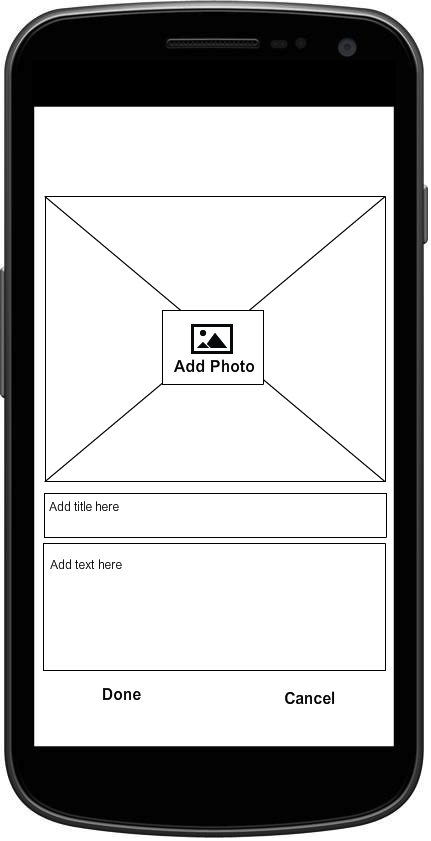


Click on heart icon to increment Touched count

Click on Gift image to view Gift

Click on + sign to add new Gift to Gift Chain

Click on + sign to add new Gift

****

1. The word [potlatch](http://en.wikipedia.org/wiki/Potlatch) comes from the Chinook jargon, meaning "to give away" or "a gift" and is a gift-giving feast practiced by [indigenous peoples of the Pacific Northwest Coast](http://en.wikipedia.org/wiki/Indigenous_peoples_of_the_Pacific_Northwest_Coast) of [Canada](http://en.wikipedia.org/wiki/Canada) and the [United States](http://en.wikipedia.org/wiki/United_States). [↑](#footnote-ref-1)