CDOT ANDROID APP Developer Manual

V 0.1.0

2016.04.09

Contents

[Vision Statement 0](#_Toc448517340)

[Scope 0](#_Toc448517341)

[Key Technical and Operational Requirement 1](#_Toc448517342)

[System Constraints 1](#_Toc448517343)

[Google Map API 1](#_Toc448517344)

[Storage Space 1](#_Toc448517345)

[Android Version 1](#_Toc448517346)

[Android Framework SDK 1](#_Toc448517347)

[System Dependency 2](#_Toc448517348)

[Drupal Services 2](#_Toc448517349)

[Enabled Resources 2](#_Toc448517350)

[User Permission 2](#_Toc448517351)

[Dependent Modules 3](#_Toc448517352)

[Services View for Mobile 3](#_Toc448517353)

[System Acceptance Test 5](#_Toc448517354)

[Project Structure 6](#_Toc448517355)

[Introduction of Packages 6](#_Toc448517356)

[CRC Diagrams 6](#_Toc448517357)

[Adapter Package: 7](#_Toc448517358)

[DBContract Package: 8](#_Toc448517359)

[DBHelper Package: 8](#_Toc448517360)

[DrupalForAndroidSDK Package: 9](#_Toc448517361)

[HelperClass Package: 11](#_Toc448517362)

[ViewAndFragmentClass Package: 12](#_Toc448517363)

[ca.zhuoliupei.observationapp Package: 13](#_Toc448517364)

[Use Case Diagram 19](#_Toc448517365)

# Vision Statement

This is part of the CDOT Observation App project. The whole project contains 2 parts: website app and mobile app. The website app is the primary part which provides full set of services to user for both viewing and recording observation records. The purpose of mobile app is to provide a more convenient way to record observation and also provide a different user experience of viewing observations. The website app also provides a RESTful API for mobile app to interact with database.

# Scope

The Android app contains the following basic functions:

1. User login, logout, register and reset password
   1. Observation app uses Session Authorization to keep user login
   2. User could login with password and user name
   3. When user logout, the session would be expired
   4. User could only send a request through Observation app to register/reset password and a validation email would be sent to the user specified email. The real register/reset password procedure is done on web app.
2. User profile with edit function
   1. User could see and edit username, profile image, address l1 and l2. User could also view email but can’t change it
3. Search, Upload, Delete Observation records
   1. User could search by Category, Record, Area with a user specified radius, and Date.
   2. User could upload observation just like web app but with an even more convenient UI
   3. User could delete his own post
4. My Posts, Newest Observations
   1. User able to see his own post in with My Posts function and able to click in any of them to view detail of it or delete it
   2. User able to see the Newest Observations that are just uploaded.
   3. This two function would cache the downloaded record so user won’t see an empty screen when offline

Since both mobile and web app share the same database, important data should be synced to show correctly in both web and mobile app. Like web app, we also want to constrain the function entries provides to anonymous users. One feature of mobile app is the high requirement of memory and storage control. So in the development, we have to following questions to answer:

1. How could we detect if the data is sync correctly
2. What should we do if the data is failed to sync because of network error or user operation error
3. What data needs to be stored locally, how much data needs to be stored, and how to manage them
4. What functions are opened for anonymous user and what’s for authorized user

# Key Technical and Operational Requirement

Related technique includes: Android Framework, Java, Google Map API, Drupal7, PHP5 and RESTful services. This project uses the Services module to provide RESTful services to mobile end. But some functions is not included in Services module so some modules in written to help data synchronization which requires the knowledge of Drupal framework and PHP5 to write your own module. This app also used Google Map API to show geo information. To use Google Map API, we need a license from Google which would be discussed later.

Operational requirement is that all code should follow Java coding standard and follow the feature in Android framework. Security is considered to be an important task so coding security principle should also be applied to prevent attacks from input.

# System Constraints

## Google Map API

Google Map API is used in this app for picking and showing location. **Developer needs to apply an API key in Manifest file in order to use API services, this key is not provided in source code.** The initial limit of request is 1000/24 hours. This could be increased free of charge, see here: <https://developers.google.com/places/android-api/usage#enable-billing>

## Storage Space

To install the app, it requires at least 30 MB storage space.

Both Newest Observations Activity and My Posts Activity would cache the title and photo of observations in local storage. The max cache count of Newest Observations Activity is 300 and My Posts Activity is 2000. So additional cache space is around 150 MB.

## Android Version

The minimum version is Android 4.0 and suggested version is above Android 5.0.

## Android Framework SDK

The minimum SDK version is 15 and the target SDK version is 23.

# System Dependency

## Drupal Services

This app relies heavily on the RESTful services provided by Drupal site. Drupal website provides services through two modules: Services module and Service View module. Besides installing these two modules, we also need other configuration to provide correct data to mobile app.

### Enabled Resources

The following resources need to be enabled for mobile end to access (more details in attached photo “services resource screenshot.png”):

* File( Create, Retrieve, Delete, Index, Create Raw)
* Geocoder(Retrieve, Index)
* My-Posts(Index)
* Newest-Observations-Mobile(Index)
* Node(Create, Retrieve, Delete, Update, Index, files, comments, attach\_file)
* Observation-record-autocomplete-mobile( Index)
* search-mobile(Index)
* single-node-detail-mobile(Index)
* System(connect)
* User (Create, Retrieve, Delete, Update, Index, login, logout, token, request\_new\_password, register, cancel, password\_reset, resend\_welcome\_email)
* User-mobile(Index)
* Views(retrieve)

### User Permission

The following permissions are also required:

* Anonymous User:
  + View comment
  + View published content
  + View Terms and Conditions
  + Use search
  + Use advanced search
* Authenticated User:
  + View comments
  + Post comments
  + Skip comment approval
  + Use the Filtered HTML text format
  + View Terms and Conditions
  + View published content
  + Climate Diary Entry: Create new content
  + Climate Diary Entry: Edit own content
  + Climate Diary Entry: Delete own content
  + Use search
  + Use advanced search
  + Save file information
  + Get own binary files
  + View user profiles
  + Change own username

### Dependent Modules

The following modules are required:

1. Services: Get a specific resource(a node, a user, or get token)
2. Services View: Get a group of filtered resources by passing in filter arguments
3. Date View: when add a date filter criteria in a services view, you could set the date value greater/less than a specific value.
4. GeofieldUpdateSaveServices: A customized module used to force store lat/lng info to database. By default, if the lat/lng is set to geocoded from address, when you upload the lat/lng info from http request, it would be ignored.
5. ProfileImageSaveService: A customized module to save user profile image. By default, we can’t change user’s profile image file id through RESTful services, this module would help change the fid in the database;

Ps. GeofieldUpdateSaveServices and ProfileImageSaveService module are in the “Drupal Customized Modules.zip” file in the “Documents” folder.

### Services View for Mobile

Several views are created only for providing services for mobile end. These views don’t have page and are separated from the views used in web app. Some of them are pretty similar to the views used by web app. They are all named with “mobile” at the end. To use these views, it requires **Services View** module. Here is a list of them:

1. Newest\_Mobile(path: newest-observations-mobile)
   1. Function: Get the newest observations, sorted by date of observation
   2. Fields returned:
      1. Content: Title
      2. Content: Nid (nid)
      3. Content: Image (image)
      4. Content: Date Observed
      5. (author) User: Name (author\_name)
      6. Content: Author uid (uid)
   3. Filter criteria:
      1. Content: Published (Yes)
      2. Content: Type (= Climate Diary Entry)
      3. Content: Date Observed (Exposed Select Day)
   4. Sort Criteria
      1. Content: Date Observed (desc)
2. Observation Record Autocomplete Mobile(path: observation-record-autocomplete-mobile)
   1. Function: Filter record by record name and return to mobile app for record name autocomplete
   2. Return fields:
      1. Content: Title
      2. Content: Nid (Nid)
   3. Filter criteria
      1. Content: Published (Yes)
      2. Content: Type (= Climate Diary Record)
      3. Content: Name (exposed)
   4. Sort criteria
      1. Content: Name (asc)
3. Personal Post Mobile(path: my-posts)
   1. Function: Return a user’s own posts
   2. Return fields:
      1. Content: Title
      2. Content: Image (Image)
      3. Content: Date Observed (Date Observed)
      4. Content: Nid (Nid)
   3. Filter criteria
      1. Content: Published (Yes)
      2. Content: Type (= Climate Diary Entry)
      3. Content: Author uid (exposed)
   4. Sort criteria
      1. Content: Post date (desc)
4. Single Node Detail View Mobile(path: single-node-detail-mobile)
   1. Function: Retrieve the detail of a post given a node id
   2. Return Fields:
      1. Content: Title
      2. Content: Description (Description)
      3. Content: Date Observed (Date Observed)
      4. Content: Image (Image)
      5. Content: Location lat/long (Location lat/long)
      6. Content: Location Observed (Location Observed)
      7. Content: Climate Diary Record (Climate Diary Record)
      8. (author) User: Picture (user\_picture)
      9. (author) User: Name (author\_name)
   3. Filter criteria:
      1. Content: Published (Yes)
      2. Content: Nid (exposed)
5. search\_mobile(path: search-mobile)
   1. Function: Provide a searching function to mobile app
   2. Return Fields:
      1. Content: Title
      2. Content: Climate Diary Record (Climate Diary Record)
      3. Content: Date Observed (Date Observed)
      4. Content: Image (Image)
      5. Content: Nid (Nid)
      6. (Content entity referenced from field\_climate\_diary\_record) Content: Category (Category)
   3. Filter criteria:
      1. Content: Published (Yes)
      2. Content: Type (= Climate Diary Entry)
      3. Content: Location lat/long (field\_location\_lat\_long) - proximity (exposed)
      4. (Content entity referenced from field\_climate\_diary\_record) Content: Category (exposed)
      5. Content: Date Observed (Exposed Select Day) | Settings
      6. Content: Climate Diary Record (exposed)
      7. Content: Title (exposed)
   4. Sorted criteria:
      1. Content: Date Observed (desc)

# System Acceptance Test

The completed Android app should have the following functions working properly:

* Login, logout, register and reset password
* Newest observations
* View and edit user profile
* View user’s own posts
* Search and upload observations
* Input validations and error handling

# Project Structure

## Introduction of Packages

* Manifests
  + The manifests file contains the settings of each activities. It also contains the uses-permission and Google Map API license key.
* Adapter
  + This package contains all the adapters which is used to display items in List View or Grid View.
* Const
  + This package contains several classes which store some app-wide commonly used strings. These strings would not change and are final variables. Such as “StatusCode”, “ResponseBody”.
* DBContract
  + Contains the Database Contract classes which defines the structure of local database tables.
* DBHelper
  + Contains the database helper classes which help read/write to local database.
* DrupalForAndroidSDK
  + Contains classes that communicate with Drupal website in a RESTful way. Dealing with the http requests.
* HelperClass
  + Contains several helper classes with different functions.
* Interface
  + Contains Interface files.
* Model
  + Contains data structure classes that used to wrap data into objects.
* ViewAndFragmentClass
  + Contains customized views and fragments
* ca.zhuoliupei.observationapp
  + Contains activities classes

## CRC Diagrams

Here we just include some important classes, for some helper classes which provide common utilities are not included since it’s so widely used by many classes.

### Adapter Package:

|  |  |
| --- | --- |
| **Class** MyObservationAdapter | |
| **Responsibilities**:  Build List View items for My Post Activity from a list of ObservationEntryObject objects | **Collaborators**:  MyPostActivity  ObservationEntryObject |
| **Class** NewestObservationAdapter | |
| **Responsibilities**:  Build Grid View items for Newest Observation Activity from a list of ObservationEntryObject objects | **Collaborators**:  NewestObservationActivity  ObservationEntryObject |

|  |  |
| --- | --- |
| **Class** RecordAutoCompleteAdapter | |
| **Responsibilities**:  Build List View items for Record Autocomplete listview from a list of RecordAutoCompleteItem  objects | **Collaborators**:  SearchObservationActivity  UploadObservationActivity  RecordAutoCompleteItem  ObservationRecordFinder |

|  |  |
| --- | --- |
| **Class** SearchResultAdapter | |
| **Responsibilities**:  Build List View items for Search Result Activity listview from a list of SearchResultObservationEntryObject  objects | **Collaborators**:  SearchResultActivity  SearchResultObservationEntryObject |

|  |  |
| --- | --- |
| **Class** SlidingMenuAdapter | |
| **Responsibilities**:  Build listview items for Sliding Menu in Newest Observation Activity from a list of SlidingMenuItem  objects | **Collaborators**:  NewestObservationActivity  SlidingMenuItem |

### DBContract Package:

|  |  |
| --- | --- |
| **Class** ObservationContract | |
| **Responsibilities**:  Provides database table schema | **Collaborators**:  NewestObservationCacheManager  MyObservationCacheManager  MyObservationDBHelper  NewestObservationDBHelper |

### DBHelper Package:

|  |  |
| --- | --- |
| **Class** MyObservationDBHelper | |
| **Responsibilities**:  Provides Read/Write MyObservation table services | **Collaborators**:  MyObservationCacheManager |

|  |  |
| --- | --- |
| **Class** NewestObservationDBHelper | |
| **Responsibilities**:  Provides Read/Write NewestObservation table services | **Collaborators**:  NewestObservationCacheManager |

### DrupalForAndroidSDK Package:

|  |  |
| --- | --- |
| **Class** DrupalAuth | |
| **Responsibilities**:  Define an interface for any authorization class | **Collaborators**:  DrupalAuthSession  DrupalAuthBase |

|  |  |
| --- | --- |
| **Class** DrupalAuthSession | |
| **Responsibilities**:  Given a HttpRequestBase Object, add cookies and CSRF token to its header. | **Collaborators**:  LoginActivity  MyPostActivity  NewestObservationActivity  ObservationDetailActivity  ObservationRecordFinder  RegisterActivity  ResetPasswordActivity  SearchResultActivity  UploadActivity  UserProfileActivity |

|  |  |
| --- | --- |
| **Class** DrupalServicesBase | |
| **Responsibilities**:  The base class of other DrupalServices class.  Get/Set resource url  Send GET/POST/PUT/DELETE http requests to the resource url | **Collaborators**:  DrupalServicesFile  DrupalServicesNode  DrupalServicesUser  DrupalServicesView |

|  |  |
| --- | --- |
| **Class** DrupalServicesResource | |
| **Responsibilities**:  Define an interface to provide create, retrieve, update, delete and index operations on resources.  Provides a better way to control resources, wrap the http requests under these methods. | **Collaborators**:  DrupalServicesFile  DrupalServicesNode  DrupalServicesUser  DrupalServicesView |

|  |  |
| --- | --- |
| **Class** DrupalServicesFile | |
| **Responsibilities:**  Implements DrupalServicesResource interface.  Extends DrupalServicesBase class.  Create, Retrieve, Update, Delete and Index operations on file resources. | **Collaborators:**  UploadActivity  UserProfileActivity  DrupalServicesBase  DrupalServicesResource |

|  |  |
| --- | --- |
| **Class** DrupalServicesNode | |
| **Responsibilities:**  Implements DrupalServicesResource interface.  Extends DrupalServicesBase class.  Create, Retrieve, Update, Delete and Index operations on Node resources. | **Collaborators:**  UploadActivity  MyPostActivity  ObservationDetailActivity  DrupalServicesBase  DrupalServicesResource |

|  |  |
| --- | --- |
| **Class** DrupalServicesUser | |
| **Responsibilities:**  Extends DrupalServicesBase class.  Login, logout user  Send reset password and register request  Get current session info  Get user profile detail  Update user profile info | **Collaborators:**  LoginActivity  NewestObservationActivity  RegisterActivity  ResetPasswordActivity  UserProfileActivity  DrupalServicesBase |

|  |  |
| --- | --- |
| **Class** DrupalServicesView | |
| **Responsibilities:**  Extends DrupalServicesBase class.  Retrieve data from the Drupal view services.  User could specify the view they want to use. | **Collaborators:**  NewestObservationActivity  MyPostActivity  ObservationDetailActivity  ObservationRecordFinder  SearchResultActivity |

### HelperClass Package:

|  |  |
| --- | --- |
| **Class** MyObservationCacheManager | |
| **Responsibilities:**  Given a list of observationEntryObjects, cache them to local database  Control the total amount of records stored locally according to First In First Out principal  Retrieve a list of observationEntryObjects, the amount could be specified.  Clear the cache data  Update cached data.  Delete a cached node by specifying the node id | **Collaborators:**  MyPostActivity  ObservationDetailActivity  UserProfileActivity |

|  |  |
| --- | --- |
| **Class** NewestObservationCacheManager | |
| **Responsibilities:**  Given a list of observationEntryObjects, cache them to local database.  Control the total amount of records stored locally according to First In First Out principal  Retrieve a list of observationEntryObjects, the amount could be specified.  Clear the cached data  Update cached data.  Delete a cached node by specifying the node id | **Collaborators:**  NewestObservationActivity  SplashActivity |

|  |  |
| --- | --- |
| **Class** ObservationRecordFinder | |
| **Responsibilities:**  Given a record name, retrieve a list of similar name from Drupal services view | **Collaborators:**  RecordAutoCompleteAdapter |

### ViewAndFragmentClass Package:

|  |  |
| --- | --- |
| **Class** DatePickerFragment | |
| **Responsibilities:**  Provide a date picker user interface  Call the context activity to handle to selected date when date is set | **Collaborators:**  SearchObservationActivity  UploadActivity |

|  |  |
| --- | --- |
| **Class** DelayAutoCompleteTextView | |
| **Responsibilities:**  Extends TextView  Provide a textview user interface with autocomplete function | **Collaborators:**  SearchObservationActivity  UploadActivity |

|  |  |
| --- | --- |
| **Class** GridViewWithHeaderAndFooter | |
| **Responsibilities:**  Extends GridView  Provide a customized Grid View with header and footer view item | **Collaborators:**  NewestObservationActivity |

|  |  |
| --- | --- |
| **Class** MLRoundedImageView | |
| **Responsibilities:**  Extends ImageView  Provide a customized ImageView which is a circle but not a square | **Collaborators:**  None( Used directly in layout files) |

### ca.zhuoliupei.observationapp Package:

|  |  |
| --- | --- |
| **Class** ChangeTextFieldActivity | |
| **Responsibilities:**  Show a simple UI for getting a single text/number input.  Return the result to caller activity | **Collaborators:**  None(No customized class is used) |

|  |  |
| --- | --- |
| **Class** ChooseUploadPhotoMethodActivity | |
| **Responsibilities:**  provide a simple UI for picking a method to upload image  Return the selected method to caller activity | **Collaborators:**  PhotoUtil |

|  |  |
| --- | --- |
| **Class** LoginActivity | |
| **Responsibilities:**  provide a UI for getting user name and password to login  provide entry to Register Activity and Newest Observation Activity  login and store session info, use profile info to local  download user profile picture | **Collaborators:**  ToolBarStyler  DrupalAuthSession  DrupalServicesUser  PreferenceUtil  RegexValidator |

|  |  |
| --- | --- |
| **Class** MyPostActivity | |
| **Responsibilities:**  Fetch user's own post from network, show it and store it locally  If network is not available, fetch from local database and show to user  Detect if the cached post still exists in cloud, if not, delete it  Provide entry to Observation Detail Activity  Monitor when user scroll to bottom, automatically download more    Control the amount of observations downloaded, stop load more when amount exceeds a max value. | **Collaborators:**  ToolBarStyler  DrupalAuthSession  DrupalServicesNode  DrupalServicesView  PreferenceUtil  DownLoadUtil  MyObservationCacheManager  MyObservationAdapter  AnimationUtil  ObservationEntryObject |

|  |  |
| --- | --- |
| **Class** NewestObservationsActivity | |
| **Responsibilities:**  Fetch newest observations from cloud, show them and cache them to local database  If network not available, fetch from local database and show them to user    Start an asynctask to detect if user's session expired, show a notification did  Start an asynctask to detect if the cached observation has an image with it, if not, download it from cloud    Provides entries to other activities includes UserProfile, Search, Upload, My Posts depends on if user login  Monitor when user scroll to bottom, automatically download more    Control the amount of observations downloaded,stop load more when amount exceeds a max value. | **Collaborators:**  ToolBarStyler  DrupalAuthSession  DrupalServicesUser  DrupalServicesView  PreferenceUtil  DownLoadUtil  NotificationUtil  SlidingMenuItem  NewestObservationCacheManager  NewestObservationAdapter  SlidingMenuAdapter  AnimationUtil  GridViewWithHeaderAndFooter  ObservationEntryObject |

|  |  |
| --- | --- |
| **Class** RegisterActivity | |
| **Responsibilities:**  Send register request to cloud and return if it's success or not | **Collaborators:**  DrupalAuthSession  DrupalServicesUser  RegexValidator |

|  |  |
| --- | --- |
| **Class** ResetPasswordActivity | |
| **Responsibilities:**  Provide UI to get the user's email  Send a reset password request to server for the entered email  Return if the request is success or not | **Collaborators:**  DrupalAuthSession  DrupalServicesUser  RegexValidator  ToolBarStyler |

|  |  |
| --- | --- |
| **Class** SearchObservationActivity | |
| **Responsibilities:**  Provide a UI to collect all search criteria  Pass these criteria to SearchResultActivity to filter from cloud | **Collaborators:**  RecordAutoCompleteAdapter;  ToolBarStyler;  DatePickerCaller;  SerializableNameValuePair;  DatePickerFragment;  DelayAutoCompleteTextView; |

|  |  |
| --- | --- |
| **Class** SearchResultActivity | |
| **Responsibilities:**  Get search criteria from intent and send them as params to cloud to download observations  Monitor when user scroll to bottom, automatically download more  Control the amount of observations downloaded, stop loading more when amount exceeds a max value. | **Collaborators:**  SearchResultAdapter;  SearchResultObservationEntryObject;  DrupalAuthSession;  DrupalServicesView;  AnimationUtil;  PhotoUtil;  PreferenceUtil;  ToolBarStyler;  SerializableNameValuePair; |

|  |  |
| --- | --- |
| **Class** SplashActivity | |
| **Responsibilities:**  Detect if there's a user already login    If a current user exists, fetch observations from local database and pass it to Newest Observation Activity  Otherwise jump to Login Activity. | **Collaborators:**  NewestObservationCacheManager;  PreferenceUtil;  ObservationEntryObject; |

|  |  |
| --- | --- |
| **Class** UploadActivity | |
| **Responsibilities:**  Provide UI for uploading observation | **Collaborators:**  RecordAutoCompleteAdapter;  DrupalAuthSession;  DrupalServicesFile;  DrupalServicesNode;  DatePickerFragment;  DelayAutoCompleteTextView; |

|  |  |
| --- | --- |
| **Class** UploadActivity | |
| **Responsibilities:**  Showing user profile  Editing user profile includes image, username, address 1 and 2  Provide entries to My Posts Activity, Reset Password Activity, Logout | **Collaborators:**  DrupalAuthSession;  DrupalServicesFile;  DrupalServicesUser;  MyObservationCacheManager; |

## Use Case Diagram

