Assignment 5

The manufacturing company wants a cloud backup for the inventory app. They also noticed that the app doesn't always save changes when it crashes. To fix the issue they want the app to save information after all actions that change the inventory or item.

Required Tasks

1. Assignment 5 builds upon the previous Assignment 4 and Assignment 3. The app is expected to work as previously stated. If not, fix any issues.
2. Data integrity is very important. Synchronize all changes to the web service instead of the local file system.
    a. If a change, deletion, or addition fails, undo the action and notify the user with a UIAlert error message.
        i. Display an appropriate error message for a non-technical user. It should be clear and concise.
        ii. The error message should give context, so that the user understands what failed, and should have an idea of what action they need to try again.
3. Add a test mode property to InventoryStore to test CloudMine failure conditions. If you don't test failures, its impossible to know if your app will respond appropriately when server errors occur.
    a. `@property (nonatomic, assign) BOOL isTestFailureMode;`
    b. If *isTestFailureMode* is YES, all actions (change, deletion, or addition) should fail and display the appropriate error messages for CloudMine network issues.
    c. If *isTestFailureMode* is NO, all actions should work assuming CloudMine and WiFi service are available.
4. Make the necessary changes to your *Inventory* and *Item* objects to subclass *CMObject* from CloudMine.
5. After starting the app, all data should be loaded from CloudMine instead of the local filesystem.
    a. Make sure to test opening and closing the app to make sure all actions work between sessions.
6. The following actions must be supported and should synchronize all changes.
    a. Delete - Swipe to remove rows in the Inventory and Item lists.
    b. Add - Create new items for the Inventory and Item lists with a modal pop-up.
    c. Change - Modifying *Item* attributes (name, unit cost, date, quantity) or *Inventory* attributes (name).
        i. You may group changes and sync after a user leaves a detail view.
        ii. Store a copy of the *Item*, so that you can roll back values. Conform your Item class to the NSCopying protocol.

iii. Provide an appropriate error message with context about the object being changed.

Hints

1. Create a new user account and application on CloudMine. https://cloudmine.me/
   a. Use Application ID and the API key in your app.
2. Make sure to invoke any super methods in the NSCoder methods when subclassing CMObject.
3. Test the *isTestFailureMode* flag in the CloudMine block after a server request.
4. Use globally unique identifiers for saving image data to CloudMine. The __bridge cast is required to convert from a non-ARC variable CFStringRef to an ARC NSString.

```
+ (NSString *)UniqueIdentifier {
    CFUUIDRef theUUID = CFUUIDCreate(NULL);
    CFStringRef value = CFUUIDCreateString(NULL, theUUID);
    CFRelease(theUUID);
    return (__bridge NSString *)value;
}
```

Evaluation (100 points)

Assignments will lose points for the following:

- Build errors.
- Build warnings.
- NSLog() print statements.
- Lack of Javadoc-style method comments.
  - Assume the reader is familiar with Objective-C. Explain what methods do in your header files and any complex logic in your implementation files.
- Incomplete or missing required specifications.
- Code style does not match “Coding Guidelines for Cocoa.”

Grading

- Style/Documentation 30
- Organization 20
- Functionality 50

Extra Credit

1. (5 points) Add a user login screen, UIViewController subclass, to support separate company accounts.
   a. Provide at least two test users in the DropBox submission notes.
   b. All actions should be associated with the currently logged in user.

   c. It should be possible to switch users with a login/logout button from the Inventory list view.

2. (8 points) Add images to the Inventory object and display them on the Inventory list and detail screen.
   a. Display an image on the InventoryDetailView and add a button to change the image.
   b. Display images on the UITableViewCells for the Inventory list.
   c. Use the UIImagePicker for the Photo Library or Camera (device only) to set an image.
   d. Save the image with a NSString unique identifier to CloudMine.
      i. The Inventory object needs a new property to be saved to associate the image with the Inventory object.
   e. Load images from CloudMine when viewing the cells on the Inventory list.
      i. It may be helpful to cache the images to prevent re-downloading images.
3. (2 points) Use the row's accessory button to edit the Inventory objects after creating them. Display the InventoryDetailViewController when pressed.
   a. `- (void)tableView:(UITableView *)tableView accessoryButtonTappedForRowWithIndexPath:(NSIndexPath *)indexPath`
   b. The standard row behavior should happen when tapping on the left side of the Inventory row cell. Show the Inventory's item list for the selected Inventory row.