**Virtual Tourist Code Review**

**Core Data**

1. The app uses a managed object model created in the Xcode Model Editor. A .xcdatamodeld model file is present.
2. The object model contains both Pin and Photo entities.
3. The object model contains a one-to-many relationship between the Pin and Photo entities, with an appropriate inverse.

**Travel Locations Map View**

1. The app contains a map view that allows users to drop pins with a touch and hold gesture.
2. When a pin is tapped, the app transitions to the photo album associated with the pin.
3. When pins are dropped on the map, the pins are persisted as Pin instances in Core Data and the context is saved.

**Photo Album View**

1. When a Photo Album View is opened for a pin that does not yet have any photos, it initiates a download from Flickr.
2. The code for downloading photos is in its own class, separate from the PhotoAlbumViewController.
3. Images display as they are downloaded. They are shown with placeholders in a collection view while they download, and displayed as soon as possible.
4. The specifics of storing an image is left to Core Data by activating the “Allows External Storage” option.
5. Once all images have been downloaded, the user can remove photos from the album by tapping the image in the collection view. Tapping the imageremoves it from the photo album, the booth in the collection view, and Core Data.
6. The Photo Album view has a button that initiates the download of a new album, replacing the images in the photo album with a new set from Flickr. The new set should contain *different* images (if available) from the ones previously displayed. One way this can be achieved is by specifying a random value for the page parameter when making the request.
7. If the Photo Album view is opened for a pin that previously had photos assigned, they are immediately displayed. No new download is needed.