Josiah Yeow 26923408

PocketLint
Mobile Application Design Specification

Application Concept

Introduction

PocketLint is a mobile application that aims to become your virtual pocket by providing a place to quickly capture and store photos of information you find throughout your day. PocketLint hopes to free up your physical pockets leaving only 'lint' left in them.

Throughout our day we may take a lot of "throwaway photos" which are images that we quickly take to save some information but don't want to keep in our phone's gallery indefinitely. Examples of these "throwaway photos" are things like brochures, business cards, advertisements, event posters etc... We would throw these images away after we've acted upon the information inside them but currently they become lost in our phone's gallery mixed in between our precious photos of our friends and family.

The focus of PocketLint is to build a mobile application that improves the process of capturing, storing, finding and managing these "throwaway photos". Users will be able to quickly take a photo and PocketLint will extract its content highlighting important information such as URLs, email addresses, phone numbers and more. By storing these photos in PocketLint, users will also be encouraged to delete these images once they are no longer needed which allows users to not only free up their physical pockets but also their phone's internal storage.

Target Audience

The intended target audience for PocketLint are busy individuals such as professionals and students who use their smartphone cameras to capture information throughout their day. According to statistics, 41% of 18-24 year-olds take photos daily[1] so this age range will be the target demographic. This demographic are also more familiar with using photo capture apps such as Snapchat and Instagram which will make adoption of PocketLint feel more familiar.

With the large number of photos that are taken daily, it has become increasingly difficult for these users to manage and locate them. Currently, users have to remember when they took the photo they're looking for and scroll to that particular location in their photo gallery app. Users may also use a note-taking app such as Google Keep or Apple Notes but these solutions require the user to create an individual note for each photo. PocketLint focuses on photo capture rather than note taking which puts it above the solutions from Apple and Google for this purpose and adds image specific functionality such as image recognition and text detection.

[1] http://landing.deloitte.com.au/rs/761-IBL-328/images/tmt-mobile-consumer-2016-final-report-101116.pdf

Application Functionality

There are several main features of PocketLint, these include:

Photo Capture

Users need to be able to quickly take photos of the information they want to capture. The button to open up the camera will be easily accessible from the main screen of the app and will use iOS/Android's native camera interface so it's familiar to the users. Once the user takes a photo, they will be given a preview of it where they can either and a caption and save or retake the photo. Upon saving, the image will be added to the main screen at the top of the list.

Photo Viewing

As the purpose of the app is to allow users to quickly locate the images they want to see, photos taken in PocketLint will be displayed in reverse chronological order so the latest image can be found first. The images will be presented in a large thumbnail format so the contents of the image can be recognised without the need of opening them.

When an image is opened, it will be displayed with more information below it (gathered from image recognition or manually added).

Photo Management

One of the purposes of the app is to reduce photo clutter to save storage space. PocketLint will persistently display the number of photos that are stored to encourage the user to delete unused images. The metaphor of a physical pocket that has limited space is the point of the app here.

Users can easily batch delete photos that have already been opened after capture. Users will also be able to delete individual photos from the main feed with a swipe to the left. The option to undo deletions will also be present to avoid accidental data loss.

Cloud Photo Storage

Photos taken in the PocketLint app will also be uploaded to the cloud and synchronised across devices. This will allow users to easily access their information no matter what device they are on. It also provides the security that their photos are not lost if they delete the app or lose their device.

Image recognition (Extra)

Due to the busy nature of the target audience, users may not have the time to add titles and descriptions of the photos they are taking. To solve this, PocketLint will use image recognition technology to figure out the contents of each photo and extract important information such as URLs, addresses and phone numbers which then can be acted upon. (e.g. calling a phone number or adding it to the Contacts app).

Document Scanning (Extra)

In order to make capturing items with text clearer, document scanning will detect the edges of the item being captured and automatically crop the image as such.

Innovation via Mobile Technology

While camera and file storage applications are common, PocketLint is able to use the unique functions of a mobile device to achieve its purpose. An outline of these include:

- High-resolution cameras allow users to quickly and easily capture high fidelity photos of items no matter where a user is. The high fidelity of these photos allows users to read the information found on the items they are capturing.
- The capacitive touch screen and accelerometer allow the use of touch interaction and physical gestures which allow users to naturally view and manage their photos. Gestures such as:
  - pinch to zoom to view details in photos,
  - swiping to scroll and delete photos,
  - push and pop to quickly preview photos
  - shaking the phone to perform an undo.

  work together to enhance this experience.
- GPS sensors allow the geolocations of photos to be captured which then can be presented on a map to give better contextual information for the photos taken.
- Internet connection allows the photos captured to be backed up online so that they can be accessed on other devices. High-speed mobile data internet connections allow for this backup to happen frequently so that no photos are lost in the event that the device it was taken on became unavailable. This fast connection also allows for the use of web API services which can be used for features such as image recognition.

The main advantage that PocketLint has above other apps is the ability to quickly take and save photos without the need to create new individual notes (as seen in Apples Notes, Google Keep, Evernote and OneNote) and keeping the user's photo gallery clean from these photos (consequence of using the native Camera app.

Platform Technology Considerations

General considerations

The Firebase Realtime Database and Firebase Auth APIs will be used to handle account creation and cloud photo storage for both iOS and Android platforms. The UI for sign up and login will be created using the FirebaseUI [2] Library. Each photo taken will be stored as an object with attributes including the photo file itself, geotag, title, text content and additional attributes for detected URLs, addresses and phone numbers. These photo objects will be stored in an array.

Google's Cloud Vision API[3] can be used for the image recognition features of the application. It's Optical Character Recognition can be used to extract text and It's attributes could also be used to assist with document scanning.

[2] https://firebase.google.com/docs/auth/ios/firebaseui

[3] http://www.codepool.biz/use-dynamsoft-ios-camera-sdk.html

iOS specific

- User settings will be stored in UserDefaults.
- Core Data will be used to locally store photos and their details.
- Photo capture will be handled using iOS' AVFoundation Camera and Media Capture subsystem[4].
- The photos will be listed using the iOS 11 style card views recreated by PaoloCuscela[5] code on Github.
- Document scanning capabilities can be implemented using Dynamsoft iOS Camera SDK[6].
- Photo geolocation and map views will be implemented using the MapKit Framework[7]

A strength of iOS is that development doesn't have to account for a large range of devices and firmware versions. If the application functions correctly on a device running iOS11, the majority of devices should run the application fine as well.

A limitation of iOS is that there are stricter restrictions and guidelines in place which could make implementing certain features or getting the app approved for the App Store more difficult. Though this limitation could be seen to improve the overall security and quality of the application.

Android specific

- User settings will be stored in Shared Preferences.
- SQLite database will be used to locally store photos and their details.
- Photo capture will be handled using Android's Camera API[8]
- The photos will be listed using Android's CardView API[9]
- Photo geolocation and will map will be implemented using Android's Location Strategies[10] and Google Maps Android API[11]

A strength of Android it is open which allows applications to access the file system and other sensors of the device more easily. This could make implementing certain features less difficult. The guidelines in place to get the app onto the Play Store is also less strict compared to Apple which could make publishing the app easier as well.

A limitation of Android is the ability to test that the application will function correctly on all devices. This is because of the software fragmentation and wide range of device hardware that runs on Android. An example of this could be the difficulty of ensuring that the app was optimised to use the camera on all devices.

---

[4] https://developer.apple.com/documentation/avfoundation/cameras_and_media_capture

[5] https://github.com/PaoloCuscela/Cards

[6] http://www.codepool.biz/use-dynamsoft-ios-camera-sdk.html

[7] https://developer.apple.com/documentation/mapkit

[8] https://developer.android.com/guide/topics/media/camera.html

[9] https://developer.android.com/reference/android/support/v7/widget/CardView.html

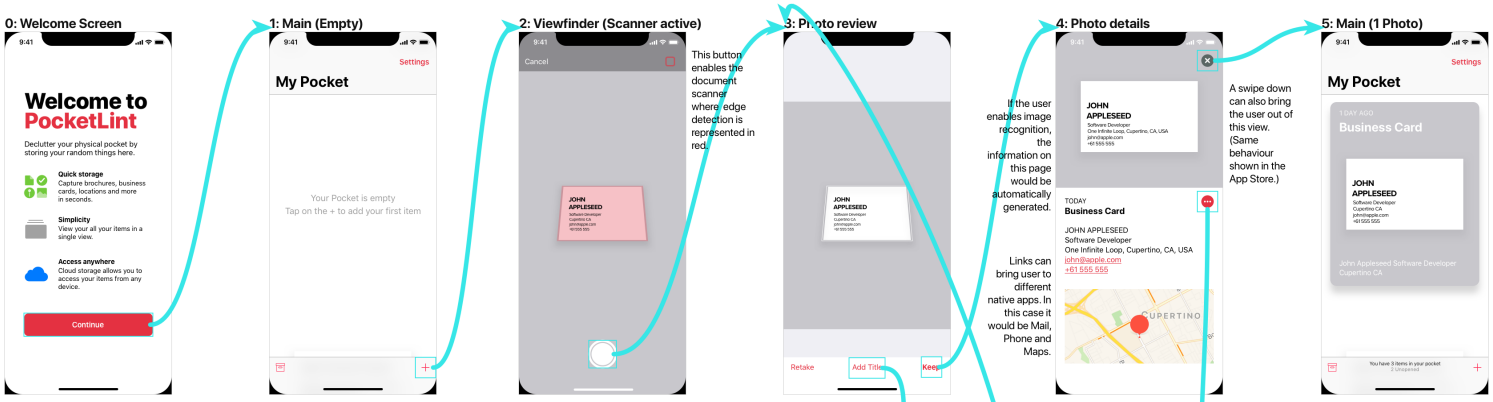
[10] https://developer.android.com/guide/topics/location/strategies.html

[11] https://developers.google.com/maps/documentation/android-api/

User Interface Design and Navigation (Storyboards)

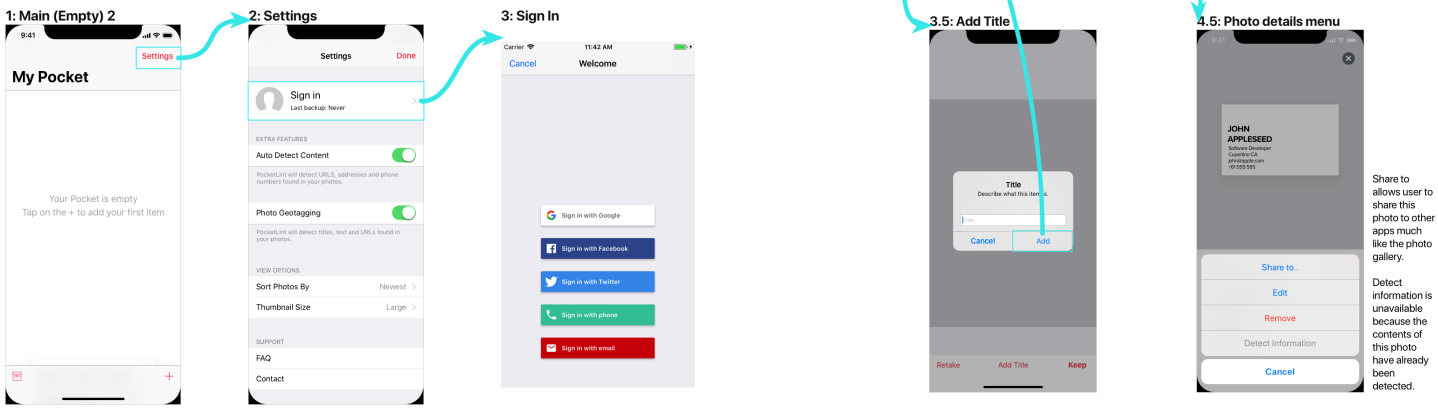
Add item to pocket.

From the welcome screen, a user is introduced to the app, taken to the My Pocket screen and then adds an item to their pocket which is then added to their My Pocket screen.



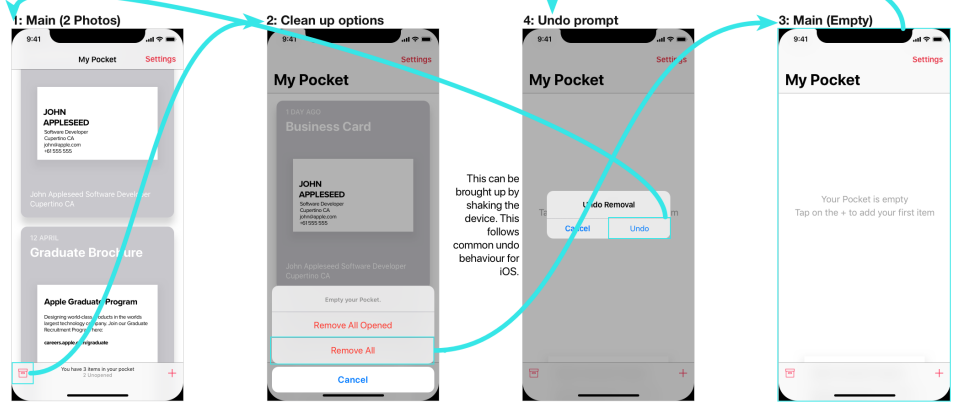
Sign In.

Go to settings page from My Pocket screen and Sign In .



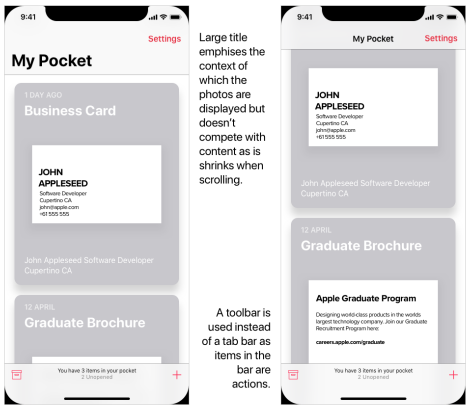
Remove items from pocket.

Remove all items from pocket. Shake device to undo removal.



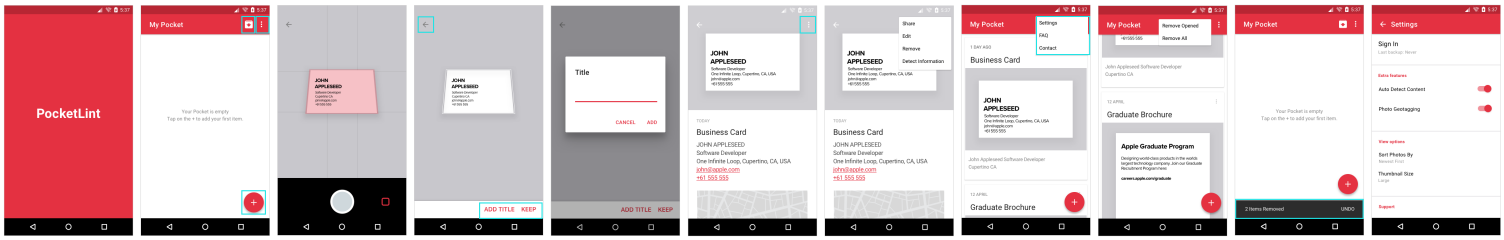
Human Interface Guidelines

Point out where Apple's Human Interface Guidelines have been adhered to.



Android Style Screens.

The android version of the PocketLint app. The user flows are the same as iOS but the cards, buttons and menus have been adapted to fit into Google's Material Design. The key difference have been highlighted below.



Positioning of + button, remove buttn and settings have be relocated to make use of the Floating Action button and Toolbar.

Toolbar back button is used and other buttons are right aligned.

More options button relocated into Toolbar Menu

Settings is relocated into the Toolbar Menu

Undo is presented as a Toast Message

Scope and Limitations

The minimum functionality required for PocketLint is the ability to capture, view and back up photos online with the option to add relevant information to each photo. Additional functions include implementing a document scanner in the camera viewfinder, image recognition to identify important information such as URLs, email addresses and phone numbers and the ability to act on these by adding integration with the Contacts, Phone and Web browser applications. The research involved to develop these additional features include learning about the iOS SDK for document scanning, connecting the application with other native apps and research about Google's Vision API.

Project Timeline

Milestone	
Prototype 1	• Create Views for all screens • Set up local storage of data • Add photo capture functionality (camera view) • Add photo viewing functionality (card view) • Connect app to firebase for account creation, sign in and online photo backup
Prototype 2	• Add ability to manually enter photo information • Add photo geotagging and map • Connect app to Google Vision API • Add ability to identify URLs, addresses and phone numbers • Add extra interaction gestures (shake to undo, swipe to go back)
Final Submission	• Add ability to open URLs, addresses and phone numbers in other applications • Add document scanner to the camera viewfinder • Ensure application is free from errors

Potential Setbacks

- Libraries may not contain certain required UI elements so they would have to be re-engineered.
- Google Vision API doesn't extract data that is useful or reliable enough for the application to use.
- Document scanner SDK may not be compatible so would have to be manually done.