Team C0: Backpack Buddy

A smart inventory system for on-the-go students

Team Members: Joon Cha, Aaron Li, Janet Li

Presented by: Janet Li

Application Area

- Students need to carry a wide variety of items to their events
  - Classes
  - Meetings
  - Sports practices
- Current asset-tracking solutions like Tile only focus on locating individual items
- **Backpack Buddy allows students to manage collections of items in relation to their schedule** via
  - 1. BLE (Bluetooth Low Energy) tags on items to be tracked
  - 2. A backpack device (RPi Zero W) that tracks which items are in a backpack/bag
  - 3. A phone app (Android, Kotlin) that handles item registration, tracking, and assignment
- Additionally, Backpack Buddy now includes a learning system which learns the user's schedule of which items they bring when and where

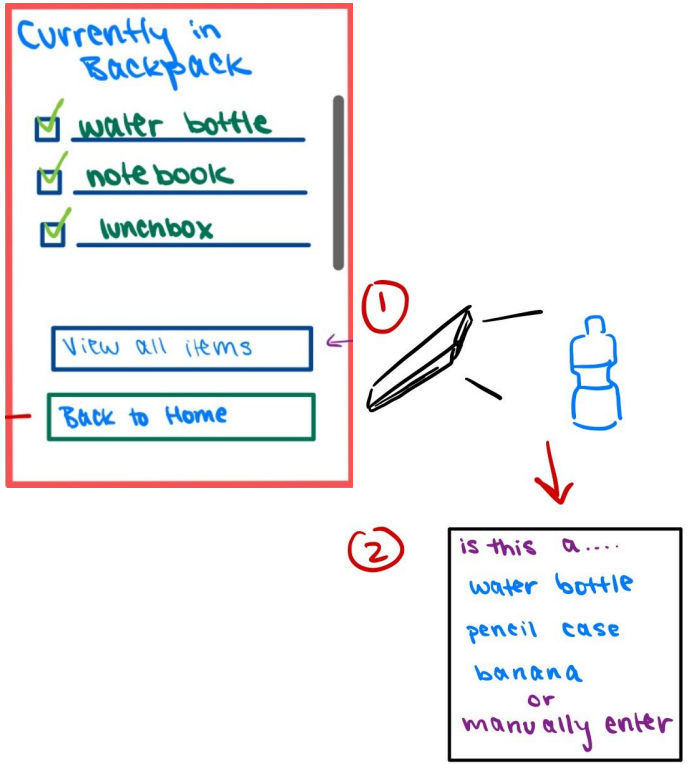
Solution Approach: Hardware

- Bluetooth Low Energy tags
  - Long battery life and easily scannable
- RPi Zero W
  - Cheap, with built-in Bluetooth module
- Distance-based item gating
  - Only include items within 0.5m and exclude items outside
- USB Li-Po Battery
  - Widely available

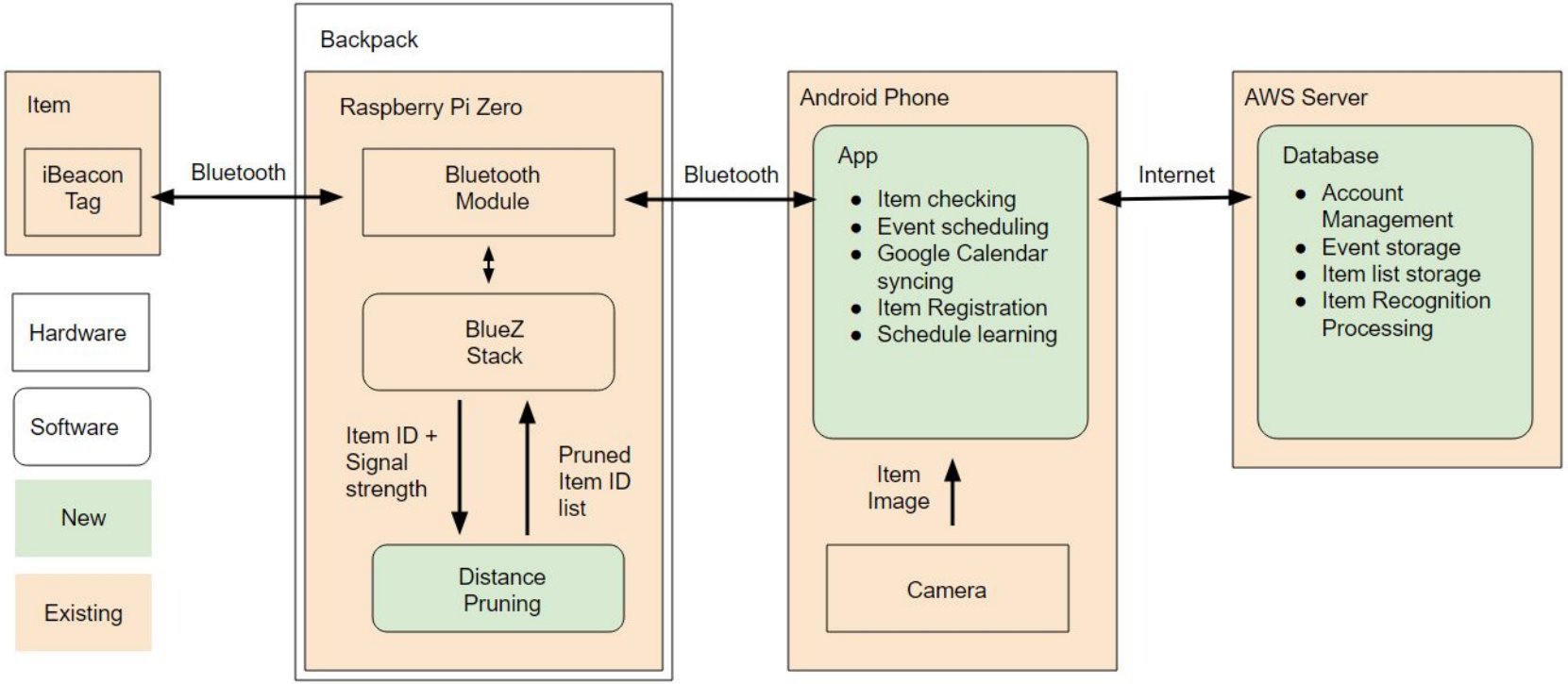
	Active RFID (RSSI LF)	Bluetooth Mesh (RSSI)	Ultra Wide Band (TDOA)	AOA/BLE
Accuracy	1 m	5 - 10 m	0.1 - 1 m	0.1 - 1 m
Refreshing period	<1 sec.	> 1min.	<1 sec.	<1 sec.
Tag battery consumption	Low	Medium	High	Low
Installation cost	Medium	Low	High	Medium/high*
Acquisition cost	Medium	Low	High	Medium/high*

Solution Approach: Software & Signals

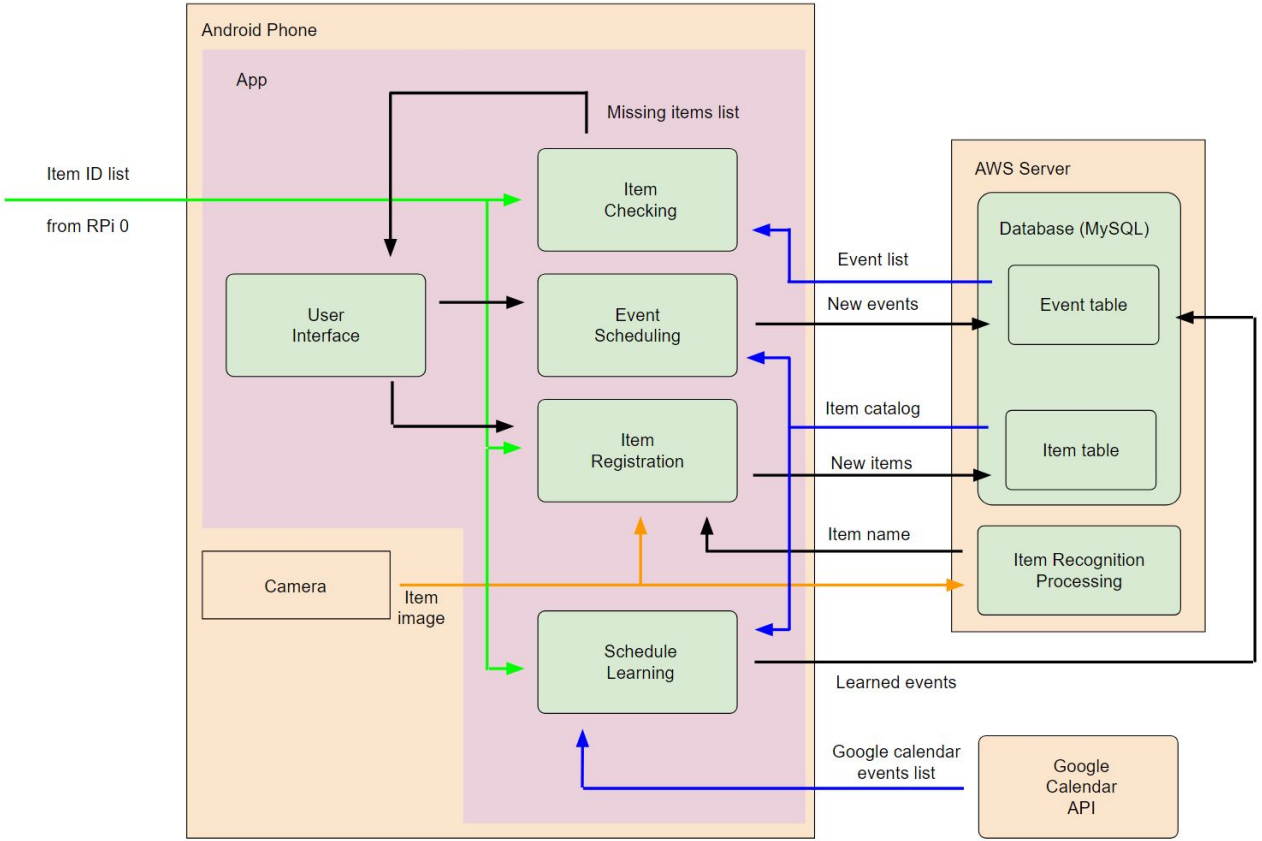
- Registration
  - Ask user to place then remove tag from backpack
    - ID is difference between before and after item lists
  - Suggest item name using computer vision item recognition
    - OpenCV, PyTorch, ImageNet dataset
- Scheduling
  - Database storage and retrieval
  - Associate items with every event
    - Ex. Soccer practice - soccer cleats
  - Can import user's calendar using GCal API
- Schedule learning
  - System can learn which items users bring to which events
    - Decreases user burden
- Item checklist and notification
  - "X" minutes before event time, notify user of missing items
    - Customizable timings



Overall System Specification



Software System Specification



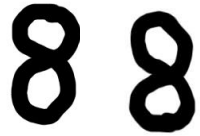
Hardware Implementation Plan

- Tags: BLE Tags bought from manufacturer
  - Cheap and reliable because BLE has existed since 2009
- Scanner: Built-in Bluetooth module on Raspberry Pi Zero
  - New - Distance pruning algorithm written by us
  - Existing - BlueZ Bluetooth development stack (open source)
- Battery: USB LiPo Power bank
  - New - Sleep mode algorithm on the RPi to conserve power
  - Existing battery technology has high power density and reliability

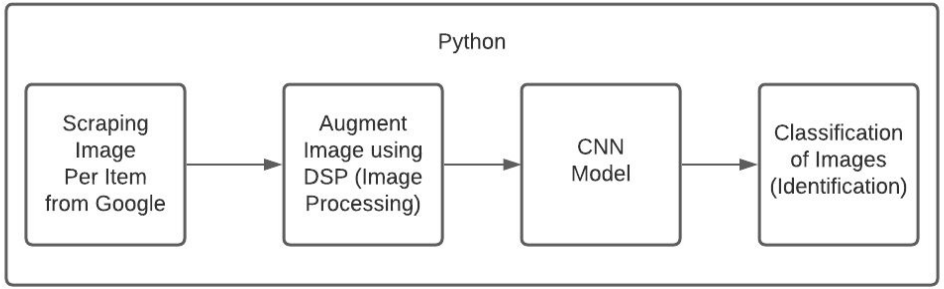
Software Implementation Plan

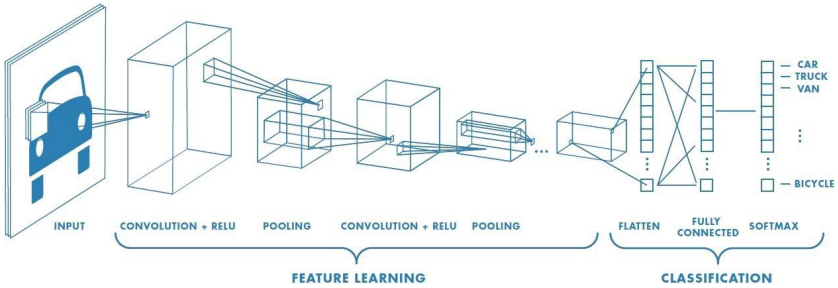
- Phone app built in Kotlin for Android phones using Android Studio
  - Kotlin > Java; coroutines, safer code, string templates, etc.
  - Our team members have Android phones
- User can assign tagged items to created events on the app or their Google Cal or…
  - Stored database of events created in-app
  - GCal API
- **Learning user's schedule**
  - Collect data about which items user brings on certain dates/times
  - Write algorithm to automatically assign items to user's events

CV Implementation Plan



- User uses phone camera to register tagged items
  - Decreases the user burden to manually type all the information when registering
- Training the Model: Web scraping images from Google
  - Collect ~250 images per item -> Augment the images up to ~1000 images per item
  - Image processing: image distortions & mirroring (augmentation), image resizing
- Creating the Model: PyTorch/Tensorflow + AWS
  - Create Convolutional Neural Networks (CNN)





Metrics & Validation

Functionality	Requirement	Testing
Tag communication & detection	< **1s** delay for tag detection	Input: Place item into backpack Output: Item list update within 1s
	Detect **10** items within a 0.5-meter range with **100%** accuracy	Input: 10 items placed into backpack Output: Item list displays all 10 correct ID's
Item visual recognition	**60%** item recognition accuracy	Input: Images from dataset of common items Output: Correct name of item
Event creation & notification	Handle **25** weekly events	Input: Create 25 events in app Output: All events appear in calendar
	Notification appears 5 minutes before event which indicates **exactly** which items are missing	Input: Subset of items for event placed in backpack Output: Notification with correct list of missing items
Interface display	< **3s** delay for interface update (when the item shows up or disappears on the phone app's checklist)	Input: Place item into backpack Output: App list updates within 3s
Backpack system battery life	Last **18 hours** w/o recharge	Input: Insert/remove item every 30 minutes Output: System still on after 18 hours

Metrics & Validation - Risk Management

Functionality	Requirement	Mitigation Plan
Tag communication & detection	< **1s** delay for tag detection	Upgrade Raspberry Pi Zero W to Raspberry Pi 4
	Detect **10** items within a 0.5-meter range with **100%** accuracy	Add additional scanners
Item visual recognition	**60%** item recognition accuracy	User can manually enter name of item
Event creation & notification	Handle **25** weekly events	Upgrade AWS storage
	Notification appears 5 minutes before event which indicates **exactly** which items are missing	Instead of notification, just display list of items currently inside backpack
Interface display	< **3s** delay for interface update (when the item shows up or disappears on the phone app's checklist)	Instead of phone display, use a display attached directly to the RPi to display items
Backpack system battery life	Last **18 hours** w/o recharge	Add additional batteries

Project Management

Aaron

- RPi scanner and tag setup
- Distance pruning algorithm
- Bluetooth communication protocol

Joon

- Item recognition CV algorithm
- Schedule learning
- Item registration

Janet

- Event scheduling
- Google Calendar syncing
- Item registration
- App GUI design