Final Presentation:
HealthBuzz
A health reminder

By Team 2:
Yang Hyeonseo
Song UGyeong
Jang Donghae
Jasmine Abtahi

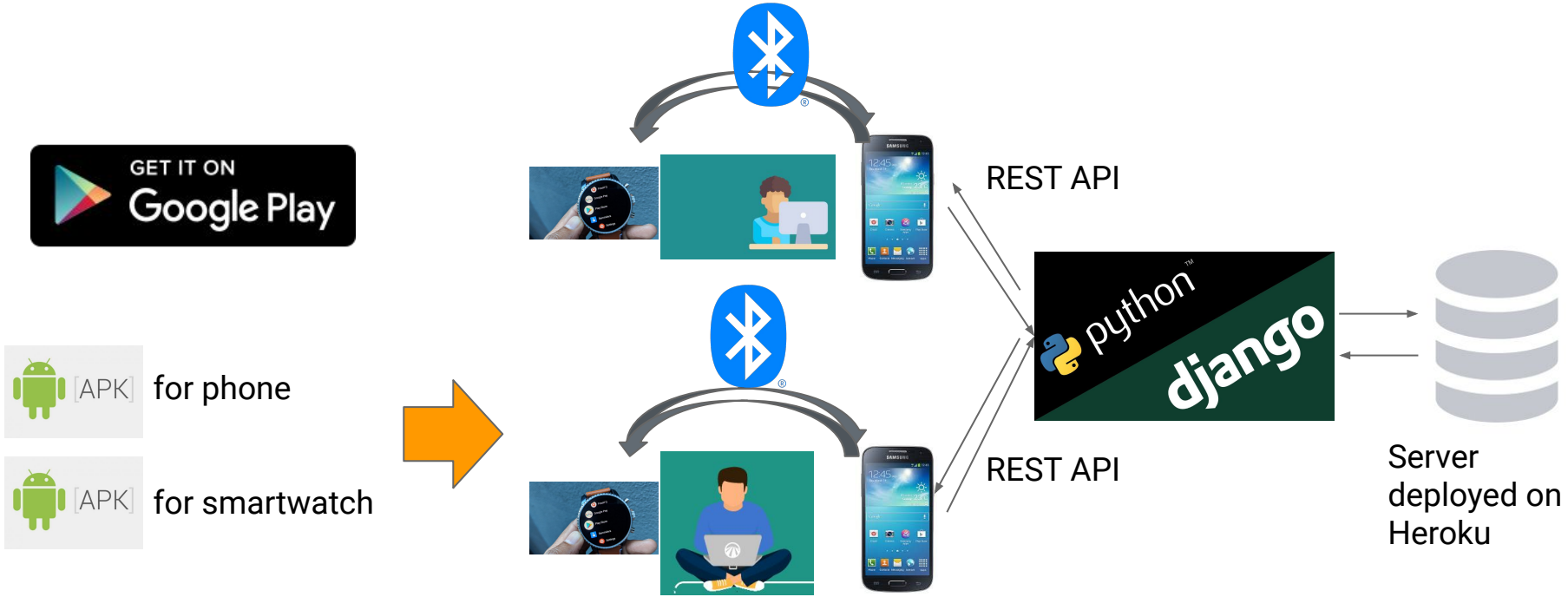
Quick recap on the project

- **Motivation**
  - People who sit in front of their desk for long time need reminder to stretch (customized interval)
  - People who forget to drink enough water for health(8 cups)
- **Proposed Idea**
  - By using sensor data, we build model to detect cup grabbing and inactive state and remind user automatically to drink water
- **Novelty**
  - We provide different service by checking user's smartwatch availability
  - We alarm user to drink water at appropriate time using ML.
  - We recommend YouTube videos based on the detected state (Stretch for sitting, Cool down, Running)

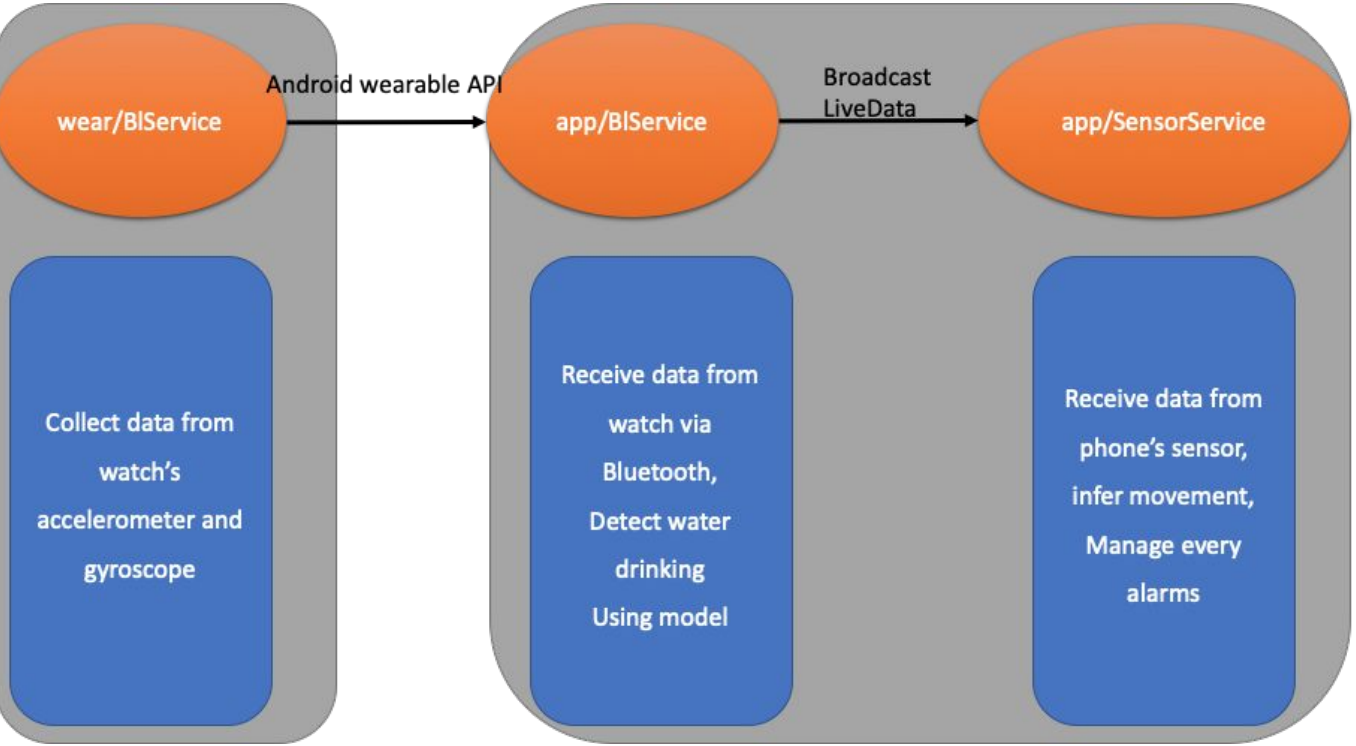
Demonstration

Login -> go to detail, and show stretching, water drinking motion

Health Buzz's overall architecture



Health Buzz's overall architecture

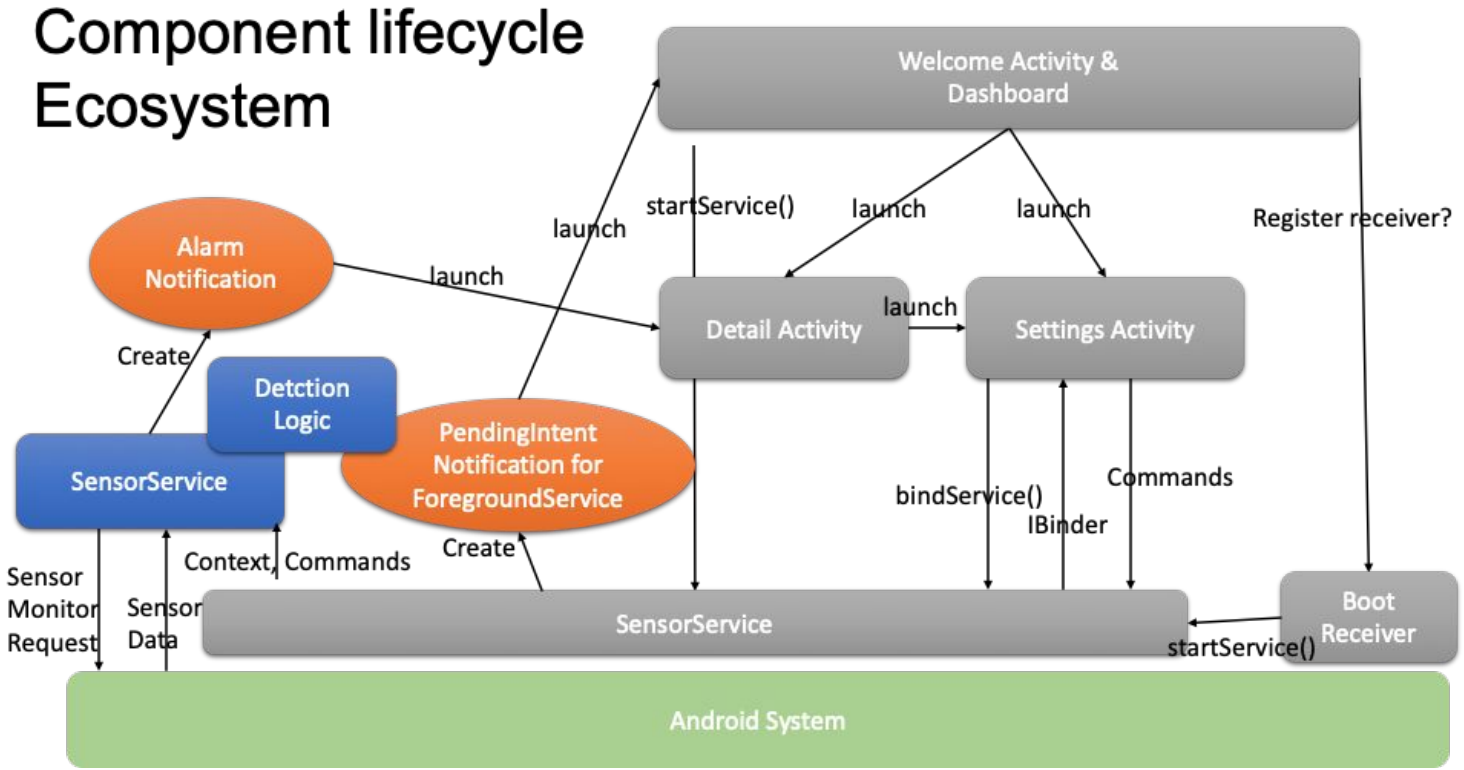


BroadcastReceivers

Activities

LiveData holder object

Deeper view of architecture



The system architecture overview

1. Two apps provided: one for android phone and the other for the android smartwatch.
2. There is a service for getting gyroscope and accelerometer data from smartwatch, and there are two services on android phone: one for listening for the phone's sensor data and one for communicating with the watch.
3. Each services of the phone own their AI models, and they infer whether the user moved for a long time, or detect if the user drank water.
4. Every notifications are managed by SensorService.
5. Communications between all these components are by using Intent, Broadcast, and LiveData.

Key technical challenges & solutions

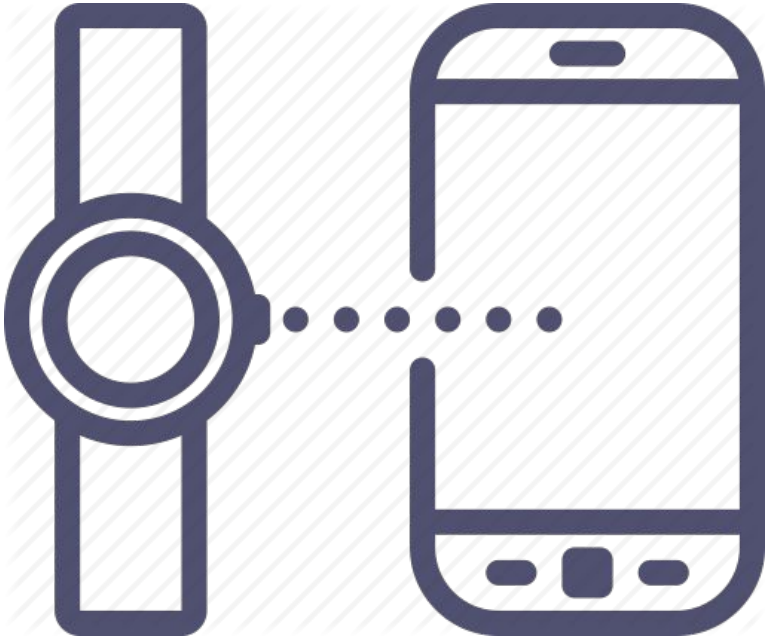
1. The smartwatch and the phone should communicate with each other.
    a. We use google wear api over bluetooth.
2. We have to infer whether the user is moving or sitting, and whether the user drank
    a. Solved by using Weka library
3. We need to provide users' rankings
    a. We created backends
4. The app has to monitor the sensor even when the user is not using the app foreground.
    a. Solved by creating a service
5. The service needs to be started automatically when the phone booting is finished.
    a. Solved by registering a BroadcastReceiver
6. The UI needs to be updated in real time, based on settings and the service's detection result.
    a. Solved by adopting LiveData pattern
7. The app should communicate with user in background with low cost.
    a. We use action-added notifications
8. Latency, Power, Accuracy
    a. Latency: Fine enough
    b. Power consumption: Low (two stage not needed

Solution approach 1 : Bluetooth

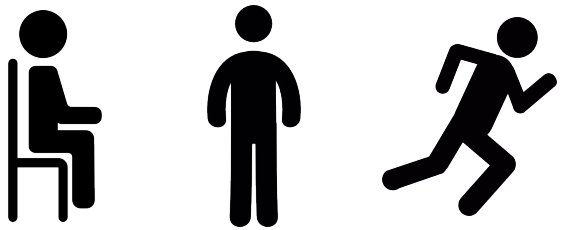
- **Connectivity issues**: Such as any other wireless system, Bluetooth can't be totally reliable. We had to make sure the connection stays continues. To solve this problem, we added parts of code that detects disconnectivity and resends the command to target to re-establish channels.

- **Delay**: We faced delay issues when we asked the smart watch to send many sensor datas all at once. To fixed this issue, we added threads that run in the background to deliver the onSensorChanged messages to the host (mobile)

- **Detect the right target**: Our app cannot connect just to any smart watch. It has to be a paired device that has our wear app installed. When we can't detect a watch as described, we provide the option to add the count manually.



Solution approach 2 : ML, DL model for inference



- Activity detection model
  => Check whether user doesn't move for long time(ex. 60 min)
  => Notify user to stretch

- Drink water detection model
  => Check user drinked or not
  => Give notification to add drink amount when user drank
  => Notify when user doesn't drink water for long time

Solution approach 2 : data collection

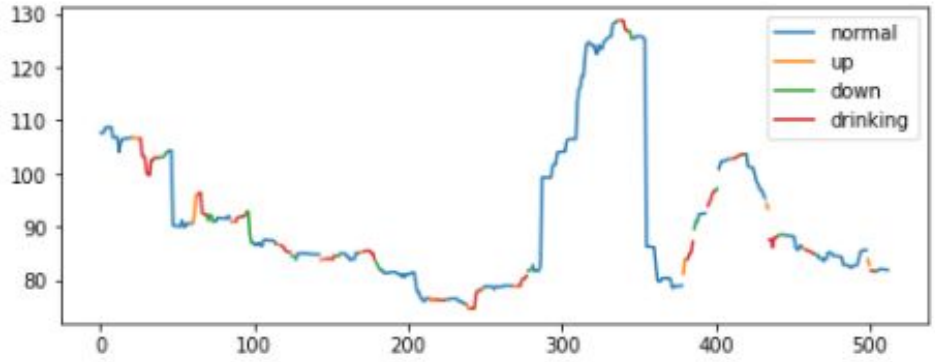
Activity Detection data	
Sensor	Accelerometer, Gyroscope
Label	Sitting, Walking, Running
Raw data dimension	10000 * 6
Total time	15 minute

Water drinking Detection data	
Sensor	Accelerometer, Gyroscope, Heartrate
Label	Normal, up, drinking, down
Raw data dimension	200000 * 6
Total time	40 minute
Participant	4 people(both right, left hand)

Solution approach 2 : Activity Detection model

1. Classifier candidate : NaiveBayes, DecisionStump, RandomForest
2. Find best model
   a. Train/Test split : 7:3 split
   b. Train the classifier candidates.
   c. Check the test accuracy
3. RandomForest was best model
   a. 98% accuracy on test set
   b. Also work fine for the realtime user test
4. Heursitic added
   a. When user is classified as sitting, timer is counting time.
   b. When user classified as walking or running more than 6 second, timer is reset
   c. When timer counts all the left time, stretch notification is given to user

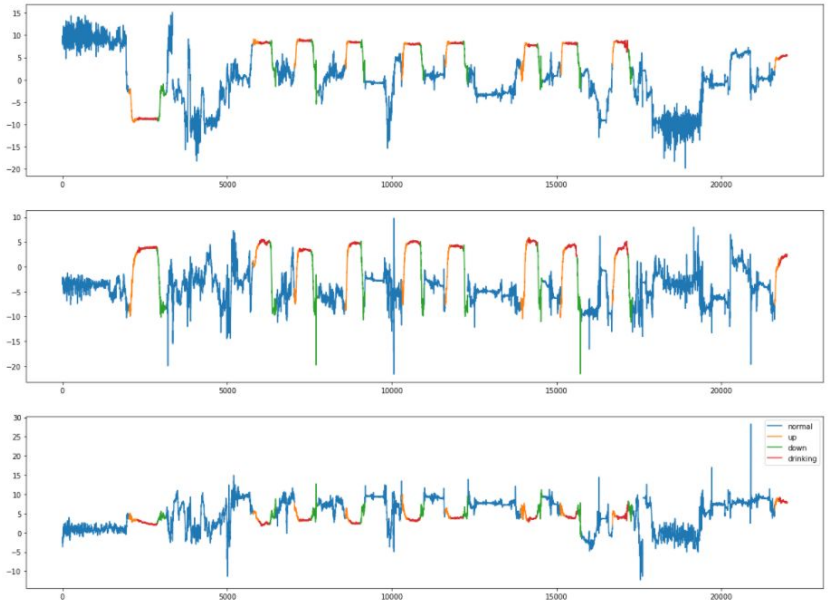
Solution approach 2 : Water drink data EDA



< Heartrate Graph>

- Heartrate didn’t show any pattern for different labels
- It is hard to use on realtime inference, because it is sampled in low frequency
- It also didn’t give accuracy gain on our model

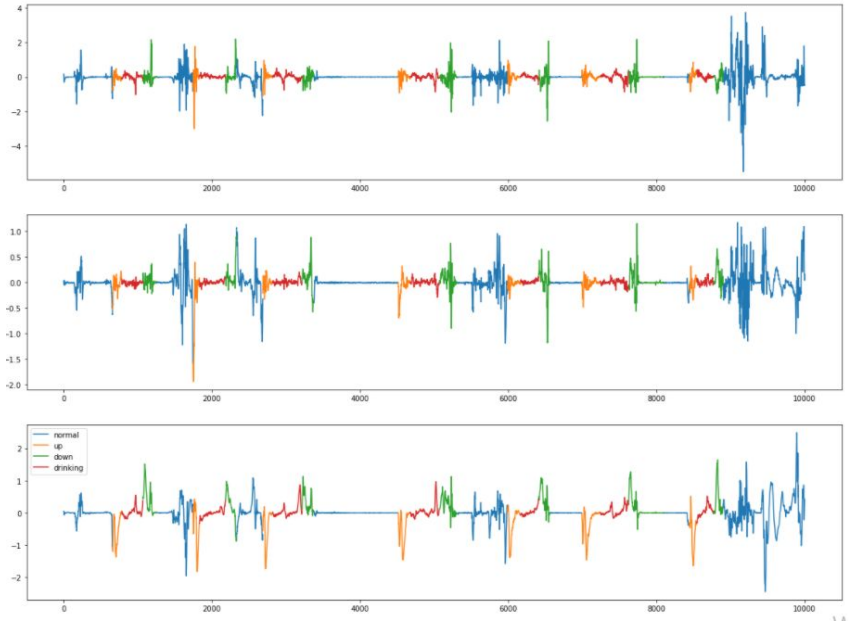
Solution approach 2 : Water drink data EDA



< Accelerometer Graph>

- Accelerometer show clear pattern on different motion
- It uses low battery, with high sampling rate
- It also give lot's of accuracy gain on our model

Solution approach 2 : Water drink data EDA

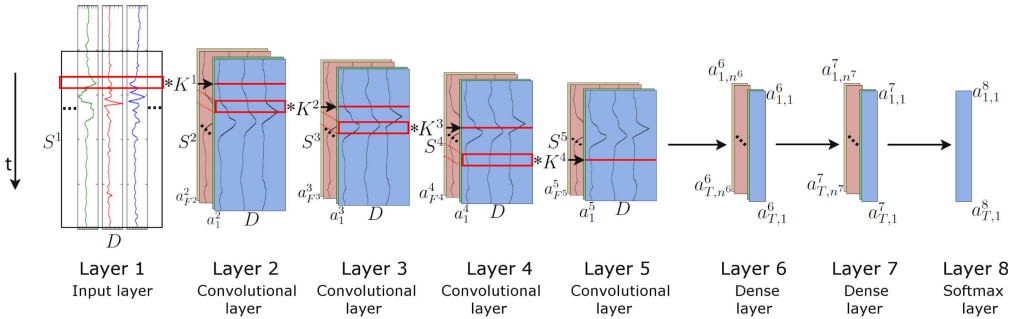


< Gyroscope Graph>

- Gyroscope show some pattern but it is little bit ambiguous with normal pattern
- It uses high battery, with high sampling rate
- It also give small accuracy gain on our model

Solution approach 2 : Water drinking model

	Model candidates	**Accuracy / F1-score**	**note**
ML	Randomforest	88.2% / 82.0	Feature extraction needed
DL	LSTM	73.4% / 66.6	
	DeepConvLSTM	76.5% / 68.3	



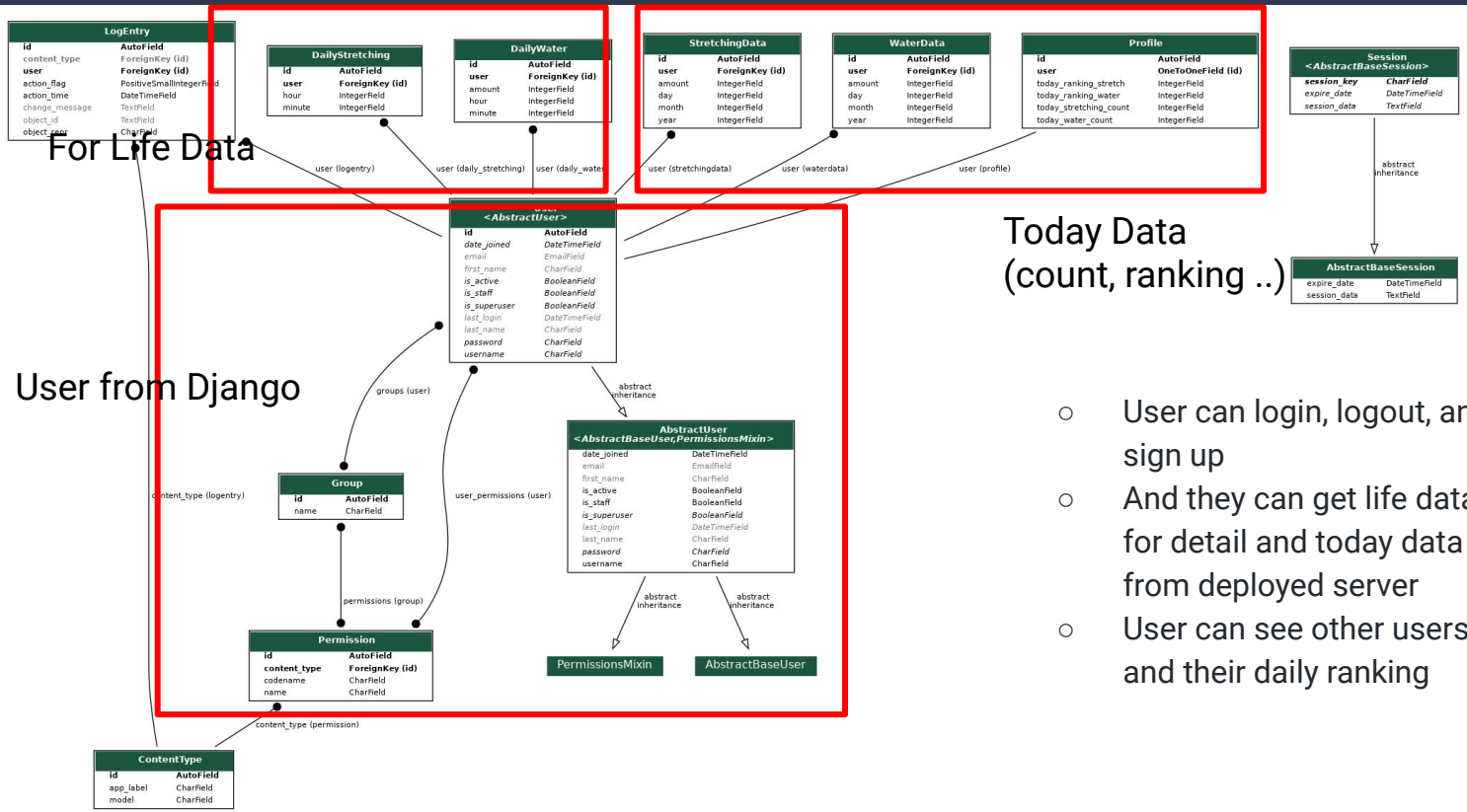
Solution approach 2 : Water drinking model

- Heuristic to use model to detect water drinking
  - Detect up => change state to up
  - Detect drink when we are on state up => state change to drink
  - Detect down when we are on state drink => state change to down
  - Detect normal when we are on state down => water drink detected!
- State is maintained when that motion have more than 50% portion
- By using Good Heuristic, final accuracy for water detection was almost 100%

Evaluation Results2 : Inference

Model : RandomForest (with feature engineering)	
Model Accuracy	Gyroscope + Accel : 88.2% / 82.0 Only Accel : 87.1 % / 81.1
Drinking detection Accuracy	Test set : 100% Real time test : about 90%
Latency	About 1 second (Bluetooth : 1 second?, Inference : almost 0)
Power	Approx 0.5% per Hour (Only accel)
Usability	It is little bit unstable on real time

Solution approach 3 : Backend design



- User can login, logout, and sign up
- And they can get life data for detail and today data from deployed server
- User can see other users and their daily ranking

Solution approach 4: Notification logic

Initial thought: Chain Observables from the sensor data to notification!

Low experience in reactiveX-tive way of thinking

Use traditional way like callbacks to run tasks based on timing

Final deliverable and success criteria

o Our Final APP

Using sensors of smart phone and watch, without setting fixed time people can get alarmed for their stretching and water drinking at their desired time interval.

To give some sort of accomplishment sense, they will get success batch when they do stretching and drinking.

People can also record and share their health history about stretching and water drinking.

o Our Success Criteria are

If our model for moving gets upper 99% acc, one for water drinking gets upper 95% acc(19 of 20) and other functions(data recording and competing with others) are implemented, we will judge our project succeeds.

Scope of the project

- **Scope is same with Mid,**
- **All our plans were done!**

- ***What are to be done (in Mid)***
  - *Implement bluetooth communication between smartphone and watch*
  - *Improve model accuracy on detecting water drinking by using DL model and feature engineering*
  - *Develop proper backend*
  - *Add community feature like ranking or competition*
  - *Improve UI/UX*
  - *Deploy server*
  - *Testing*

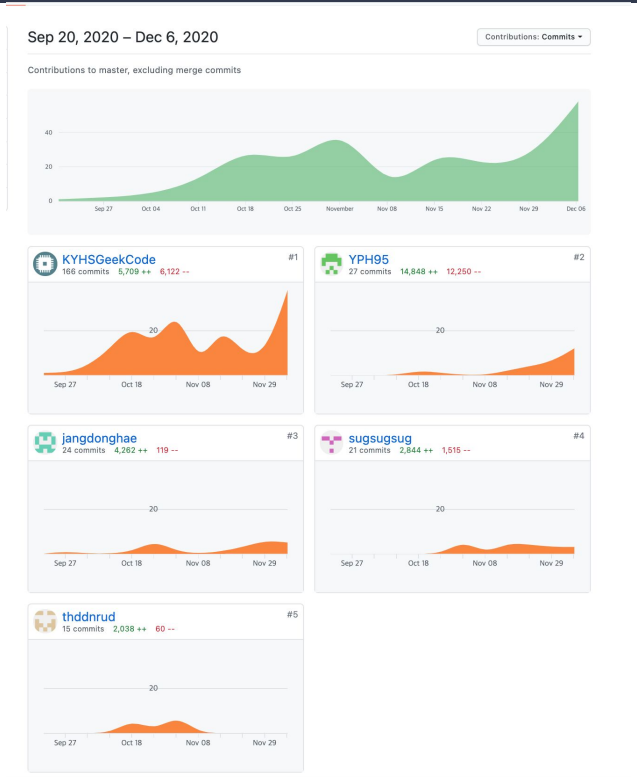
Project schedule - Done list before Mid

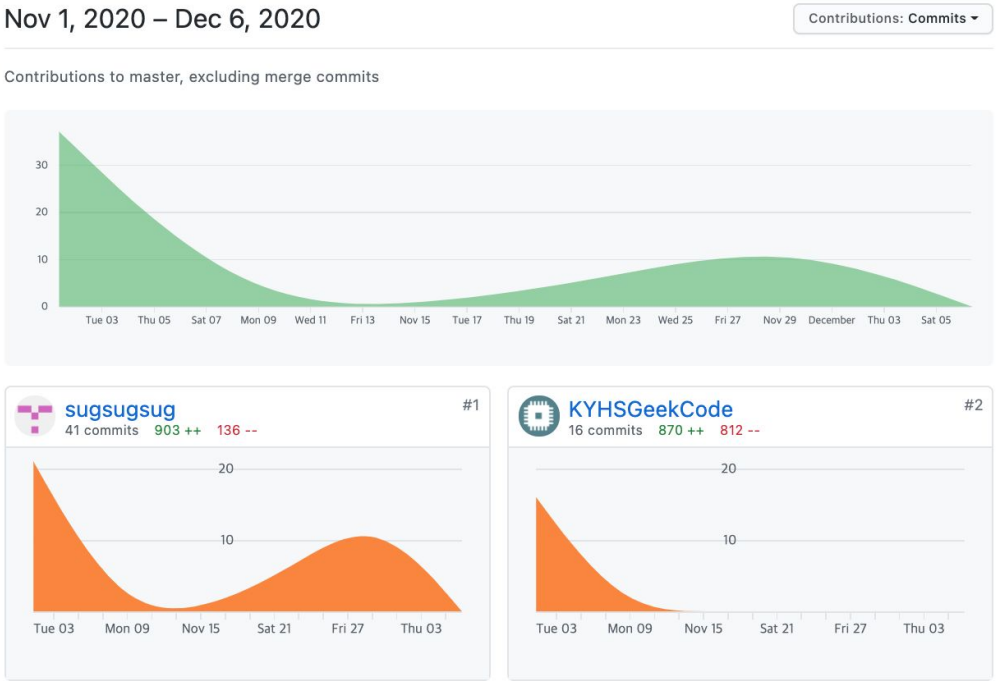
Feature	Who did
Build skeleton app	Everyone
Get friendly with smartwatch and phone sensors	Everyone
Add additional app component(service, notification, broadcast)	Yang
Study ML,DL model	Everyone(Jang lead)
Collect data from sensors and Train model and deployment	Jang & Song
Improve UI(adding badge, icon)	Jasmine
Detail page design	Song
Main dashboard	Yang

Project schedule - Timeline(mid ~ final)

11~13	Study and implement Bluetooth between smartwatch and mobile phone	Jasmine, Yang, Jang	
11~13	Develop proper backend	Song, Yang	User account, user info, Daily, Life Stretching Water Data
12~14	Collecting data for water drinking from smartwatch	ALL	
12~14	Improve accuracy of water drinking detecting model by using ML/DL technique, feature engineering	Jang	Dealing with inaccurate classified samples => Heuristic used
12~13	deploy server	Yang, Song	
12~14	Work on detail page(UI, connect with backend)	Song	
12~14	Work on main page dashboard(UI, connect with setting)	Yang	
12~14	Implement app's background service part for giving notification	Yang	
12~14	Improve UI(weather, etc)	Jasmine, Yang	
12~14	Improve UX	Jang	Deal with users's potential uncomfort
13~14	Add community feature like ranking or competition	Song	
13~14	Sync smartwatch and phone app	Jasmine	
13~14	Add user guide	Jasmine	
13~14	App testing	ALL	All functions, backend need to work well without error
15	**Project Final Presentation and Demo **	ALL	

Git usage stats





Lessons learnt and reflections

- We read a medical paper that diver's HR becomes lower than 95% of its significance level, and tried HeartRate to detect water drinking motion more accurately but failed(sadly..) -> We learned paper and real situations can be quite different.
- Watches memory and computation strength is not good enough -> need to wisely decide what parts of implementation needs to stay on watch and what should be on mobile.
- Make our work smarter -> collecting user's data and add predictions, user's own model
- And lastly, after A LOT of testing, we realized this couldn't be more true: **"No amount of testing can prove a software right, a single test can prove a software wrong."**

Thank you!