1 Feature Design and Functional Requirements

We implemented all three features described in part A, that is Doggy Friends Matching, Dog-walking Event Exploration, Request to Walk with Dog. Furthermore, we realized additional functions, including different login methods, different kinds of notifications handler, user profile and pet profile manager.

Sign Up and Sign In: As a Dog owner, one can register with their email address, Google account, Facebook account or Apple ID. When typing in login page, we implemented an animation here that the dog icon will behave with typing actions. Later on, the user information will be returned to database, so that users don't need to log in again in a short time. After login, the dog owner can create multiple dog profiles for one's pets, which will be elaborated in details in the following sections.

Doggy Friends Matching: we implemented a swipe page showing images of recommended dogs. The user can swipe right to "like" this dog, and send a friend request to the pet owner at the same time. Or the user can swipe left to "dislike" this dog. We also added an animation with the swipe action, with "Woff" icon with "like" swipe and "Poop" icon with "dislike" swipe. Along with the dog image, the name and birthday will be displayed on matching page as well. One notification will be generated here and pushed to the related user via system notification.

Event Exploration: exploration feature covers two parts: exploring nearby events and initiate a new event. Compared with part A, we add the feature of supporting four different kinds of events: dog-walking, birthday ceremony, new-born celebration and pet race. In the main exploration page, a list of nearby events is displayed, and each with an event image, which can be tapped to deep dive for more detailed event information. And event type can be implied from the event icon displayed along with pet name. In the detailed page, the user can send a join request, which is a kind of notification we will see in the notification handler. Also a user can use the "plus" button on the right side of the main page create a new dog-walking event and submit.

Notification Handler: this feature deals with four different notifications: friends-owned events broadcast, join request, response to events invitation and upcoming events reminder. Friend-owned events broadcast shows all the events initiated by friends, via which the user can directly tap in to join the event or dismiss the notification. This is another way to join the events other than via exploration page. Join request is where event holder will accept or decline the request. And this response will be received and displayed in the response part. The reminder part will show all upcoming events that the user plans to participate, including the ones to hold and to join.

Profile Manager: profile manager provides the interface to manage user profile and dog profile. These data are post and get from database. We allow users to review and edit user profile, review, edit and create pet profile in this section. These updates can be displayed on the screen once submitted.

2 Non-Functional Requirements

We also conducted nonfunctional requirements (NFR) analysis to assess whether we build PupGo correctly. Specifically in the context of the PupGo app, we focus on the following categories of NFR:

Performance: To assess how fast does the app respond, we test with the time required to react to each button action. UI test returns exact app running time for us, seen in Fig. **??**. On average, it takes the app around 5 seconds to launch. For each button action, the app takes responding time varying from 0.1s to 6s. First button action requires longer time due to app idle state. Afterwards, the following interactions with users are quick and smooth.

Scalability: Thanks to the redeployment of service to AWS, we are able to dynamically expand the capacity of our database. AWS help us afford fluctuating traffic with its distributive algorithm.

Usability: User may upload pictures of any size, then front-end crop them into desirable size and return to database.

Security: Our back-end is not public, and only reply to calls that are encrypted with particular tokens. So that users' information is safely transmitted. Plus, we doesn't provide searching function-

ality, so users only inspect behaviors of nearby dogs and friends. In this way, stalking and offense from strangers are precluded. Similarly, we don't add chatting room or any text editing functionality, so our app can hardly be misused to spread any unhealthy information or organize any illegal events. However, there's room left for us to improve security. For instance, we don't have picture censorship, which demand more sophisticated computer vision API in the back-end.

Portability: Our backend projects are powered by the Docker [1] platform. We had tested our Docker image on different Operating Systems Like Ubuntu, Centos, and ArchLinux. With Docker, it is effortless to deploy our PupGo backend service and MySQL Database server in multiple machines. During the development period of our project, we have three development environments. Testing Environment: we host our service in our LAN. Staging Environment: a remote machine owned by one team member that is located in Taiwan. Production Environment: we rent a globally available EC2 instance on AWS. The app can't accept be used on device earlier than IPhone 8. Otherwise, the boundary will exceed device's screen.

[1] Docker Platform: `https://www.docker.com`