MET CS683: Mobile Application Development Project Proposal

Project Title

Tiny Paws

Github Repository

https://github.com/CS683/project-agam-purohit-jipeng-liu

Team Profile

Team Member	Experience & Qualifications & Strengths
Jipeng	Web application back end, Java, C#,

Overview

For Animal lovers Who want to record their pets' body information and medical records. And Someone who wants to connect with others within the community to learn about pet care tips, pet friendly locations. Also, For anyone who likes to help homeless animals, this application will meet their needs.The Pet Management(PT) is a Mobile Android application that allows users to manage their pets on a virtual platform, connecting other owners. More importantly, this application(PT) will help users save stray animals. Users can post homeless animals' whereabouts or post an adoption request for them. Any users who live close to posted animals can leave them food at a certain location on the street or adopt them home as a new family member.

Related Work

UNLIKE other social media applications, such as Facebook, twitter, tiktok, they are focused on human life sharing. This mobile application (TP) will pay more attention to animals, everything posted on this app must be animal related. It aims to improve people's recognition of scientific pet breeding, disseminate useful pet breeding knowledge, establish a good pet social circle, track and save stray animals.
UNlike other pet applications, like YOUCHONG PET, BELL PET, these applications focus on online interactions. They encourage users to have discussions with other people through app chats online. TP will have those online functions, additionally, we

like to encourage users to have more off-line activities. We will have more features that rely on maps and hope users can enjoy both indoor and outdoor animal time. Users can share pet friendly location flags on map, then more users can meet together here in person.

Virtual user persona

User A:Emma	New cat owner
Currently	Emma just adopted a cat from a shelter. She knows basic knowledge about taking care of cats, she wants to give the cat a better life. Then Emma did a lot of research on google. Sometimes it works, but sometimes two topics have opposite opinions. Emma knows different cats have different situations, she is not sure about what kind of cat she has. She is hesitant to take the next move.
With PM	Emma can record new cat's body information in PM. PM will calculate the cat's daily calorie needs, and give her a suggested food amount/weight based on current cat food. Also, Emma can check the posts to find cats of the same variety and ask owners what they should pay attention to.

User B: Mr.Lee	Large dog owners
Currently	Mr. Lee's family has two large dogs, and he has not found a place for large dogs to gather and play near his home. He want s to have a walk after dinner, but the sidewalk is not wide enough for two large dogs. Mr. Lee has to walk off the sidewalk if facing other passers-by, which brings inconvenience. Then, he decides to take the walk in the early morning and late afternoon. This led to insufficient daytime exercise for large dogs, and also reduced the quality of Mr. Lee's sleep.
With PM	Mr. Lee found the nearest dog playground by checking the green pushpin marked by PM, and finally he could take the dog out for a walk at the prime time after dinner. People and dogs have made new friends, and he has learned more about dog breeding from PM. Finally Mr.Lee could have a good night.

User C: Jason	College student, cat lover but can't have one at dorm

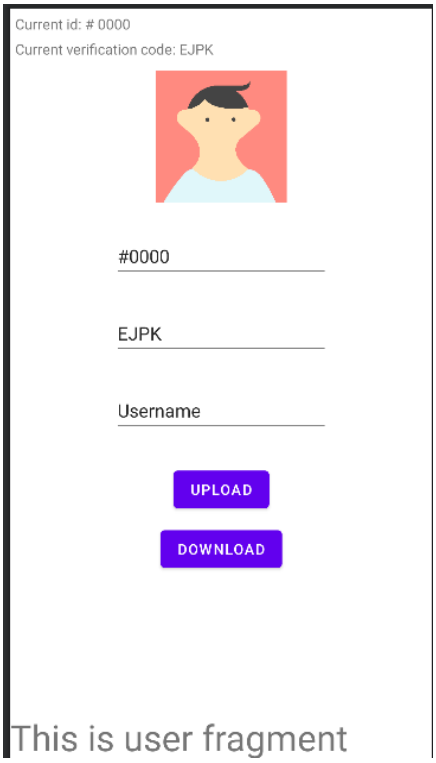
Currently	Jason really loves cats, but he is a college student who lives in college dorm. Manager doesn’t allow students to have pets in their room. Jason has to feed stray cats in the closest green way. One day, he tried to save an injured kitten. However, he doesn't have lots of twitter fans, and most facebook friends are students who can’t do much help as well. Jason can only do a simple first-aid, and give the kitten an amazon box with some food and water. Hope someone can save the kitten . After that, Jason never saw the kitten again, he was always worried.
With PM	Jason posts stray cats documents in PM, Let more people know and understand the physical status (photos, physical characteristics, sterilization status), habits, habits of getting along with people and food preferences of stray cats in the community. The incidents of stray cats and people frightening each other were successfully reduced, more people participated in the interaction with stray cats, and some even adopted community stray cats. Posting on TP when encountering emergencies, the response time is greatly shortened, and pets in dangerous situations get effective help.

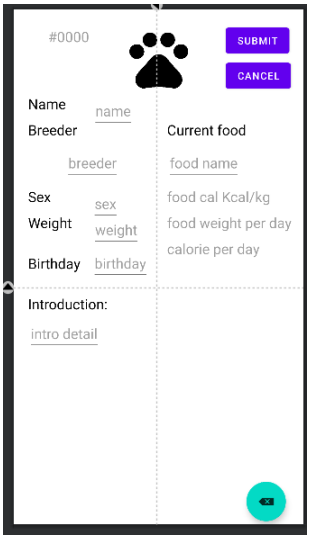
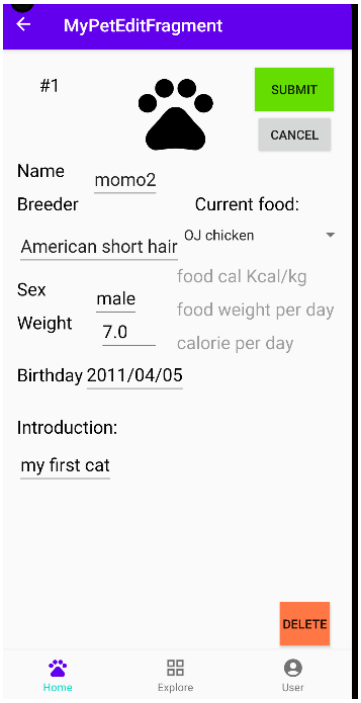
Requirement Analysis and Testing

Essential goal

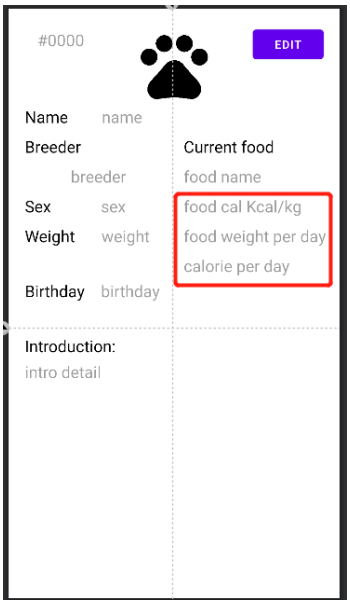
- helping pet owners record pet’s body information, medical records and give some tips based on those recorded information, such as suggested weight, length, food & water intake, activities and sleep time.

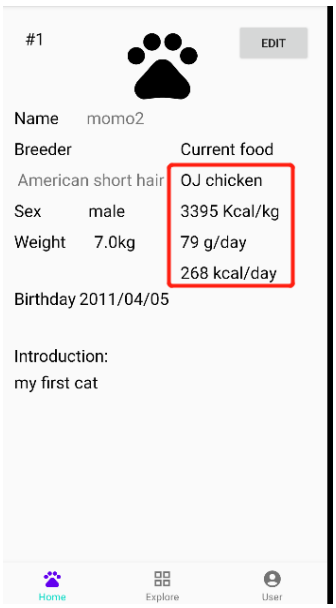
Title	Account system
Description	As a user, I want to have an online account & password account to Synchronize data with cloud.

Mockups	 
Acceptance tests	User type in user_id and user_verification_code. If they match the data in the cloud. App will show current account information at the left top corner, if not math, id = #0000. After that, user can sync local data with the cloud, which includes “pets”, “foods”, “notes” and “username”. Upload button will upload local data to cloud, download will rewrite local data with cloud data.
Test Result	
Status	Iteration 0: implemented the project mind mapping. Iteration 1: ● Created mockups, fragment and navigation designs ● Created ERD ver0.1 Iteration 2: ● Redesign sync logic and ui ● Coded xml file Iteration 3: ● Nothing new

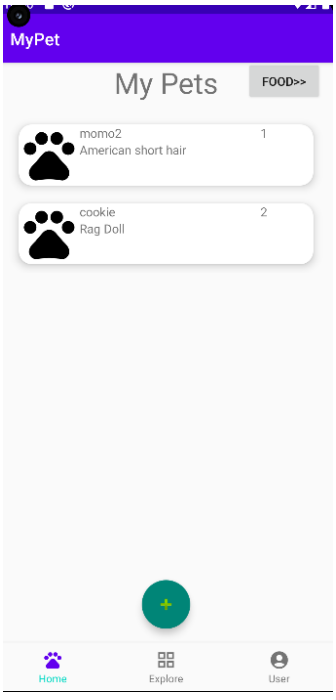
Title	Health management
Description	As a user, I want to have a page to record and track my pet's information.
Mockups	
Acceptance tests	Users can type in any data in, click submit the database will update the new changes. There is a remove float fab to remove the current pet when users click on it.
Test Result	
Status	Iteration 1: • Created mockups, fragment and navigation designs

	• Created ERD ver0.1 • Created homepage bottom navigation bar Iteration 2: • Coded the MyPetEditFragment and related xml • Coded pet data class, the database to store pet data, and the repository, adapter, viewmodel to manage data. Iteration 3: • Added a spinner to choose food • Update out looks for buttons • Update food data class, make it contains more attruibutes

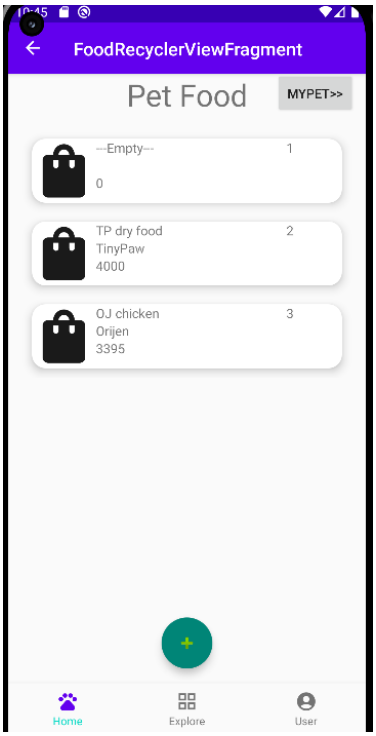
Title	Diet calculator
Description	As a user, I want to know how much food I should feed my pet daily based on their body information.
Mockups	
Acceptance tests	User added a pet with necessary body information. App can calculate and show the pet's daily calorie intake.

Test Result	
Status	Iteration 1: ● Created ERD ver0.1 Iteration 2: ● Coded the pet data class, food data class. ● Coded the PetDetailfragment and xml Iteration 3: ● Can get food information from database ● Can calculate food weight and calorie intake based on pet weight and food unit calorie

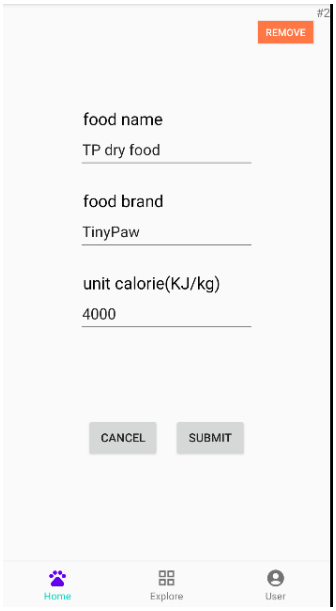
Title	Multiple animals
Description	I have multiple pets, I want this app can store more than one pet information

Mockups	
Acceptance tests	This is a recycler list view, containing a list of pet data class card views. Users can click on the card view to check the pet detail page. Users can click the float fab to add a new pet.
Test Result	

Status	Iteration 1: ● Created mockups, fragment and navigation designs ● Created myPet fragment, plan to hold multiple pets cardviews Iteration 2: ● Coded the pet data class, related cardview, view model, database, adapter, and viewholder. ● Coded the recyclerview list xml file ● Created the page navigation graph Iteration 3: ● Nothing new

Title	Multiple foods
Description	I have multiple pets, and they eat different food. I want this app can store more than one kind of food
Mockups	
Acceptance tests	This is a recycler list view, containing a list of food data class card views. Users can click on the card view to check the food detail page. Users can click the float fab to add a new pet.
Test Result	

Status	Iteration 3: ● Coded the recyclerview fragment, food cardview holder, food card adapter. ● Created the xml

Title	Pet foods management
Description	I want to record the food I purchased for pets, record its calorie information and write notes to record if this food is good or not.
Mockups	
Acceptance tests	This is a food detail and edit page. Users can change the food name, food brand, and unit calorie. Click submit to save data into database, click remove to delete current food from database, click cancel to void any changes.
Test Result	
Status	Iteration 3: ● Coded the food edit fragment ● Created the xml

Desirable goal

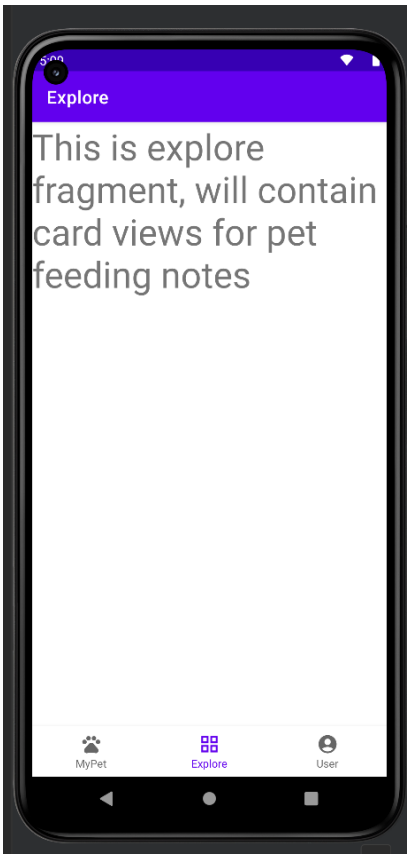
- building a pets social circle. Owner's can make posts to share pets' photos, pet friendly locations and host some offline pet events.

Title	Text Post
Description	As a user, I want to make title + text posts.
Mockups	
Acceptance tests	Users click the ‘new post’ button, a new post editing page will come up. Users can type in the title and context. Then click the post. The database should have those texts on record, and the user's post can be seen by others.
Test Result	
Status	Iteration 1: ● Created ERD ver0.1 Iteration 2: ● Nothing new Iteration 3: ● Nothing new

Title	Photo Post
Description	As a user, I want to make a photo + title + text post.
Mockups	
Acceptance tests	Users click the ‘new post’ button, a new post editing page will come up. Users can type in the title, context, and a photo. Then click the post. The database should have that context on record, and the user's post can be seen by others.
Test Result	
Status	Iteration 1: ● Created ERD ver0.1 Iteration 2: ● Nothing new Iteration 3: ● Nothing new

Title	Video Post
Description	As a user, I want to make a short film + title + text post.

Mockups	
Acceptance tests	Users click the ‘new post’ button, a new post editing page will come up. Users can type in the title, context, and a short film. Then click the post. The database should have that context on record, and the user's post can be seen by others.
Test Result	
Status	Iteration 1: ● Created ERD ver0.1 Iteration 2: ● Nothing new Iteration 3: ● Nothing new

Title	Notes page
Description	As a user, I want to have a new page to record feeding notes. And explore others’ notes.
Mockups	

Acceptance tests	Users can click the ‘explore’ button at the navigation bar. A new page for notes will show up..
Test Result	
Status	Iteration 1: ● Created mockups, fragment and navigation designs ● Created explore fragment, plan to hold multiple notes cardiews. Iteration 2: ● Relocated navigation graph for explore Iteration 3: ● Nothing new

Title	Location mark
Description	As a user, I want to mark my notes with location
Mockups	
Acceptance tests	When users make posts, users can put a pin flag on the map.
Test Result	
Status	Iteration 1: ● Created a map fragment , should connected with google map. Iteration 2: ● Nothing new Iteration 3: ● Nothing new

Title	Post tag system
Description	As a user, I want every post has its tag, then I can use filter to quick find useful information
Mockups	
Acceptance tests	When a user makes a post, the user can choose their post’s tag.
Test Result	
Status	Iteration 1:

	● Nothing new Iteration 2: ● Nothing new Iteration 3: ● Nothing new

Optional goal

- animal rescue, it's an extended function based on a secondary goal. It's a special post which shares the homeless animal locations, pet abuser alert.

Title	Authenticity Check
Description	As a user, I want every emergency post to be real.
Mockups	
Acceptance tests	Don't have a detailed idea yet
Test Result	
Status	Iteration 1: ● Nothing new Iteration 2: ● Nothing new Iteration 3: ● Nothing new

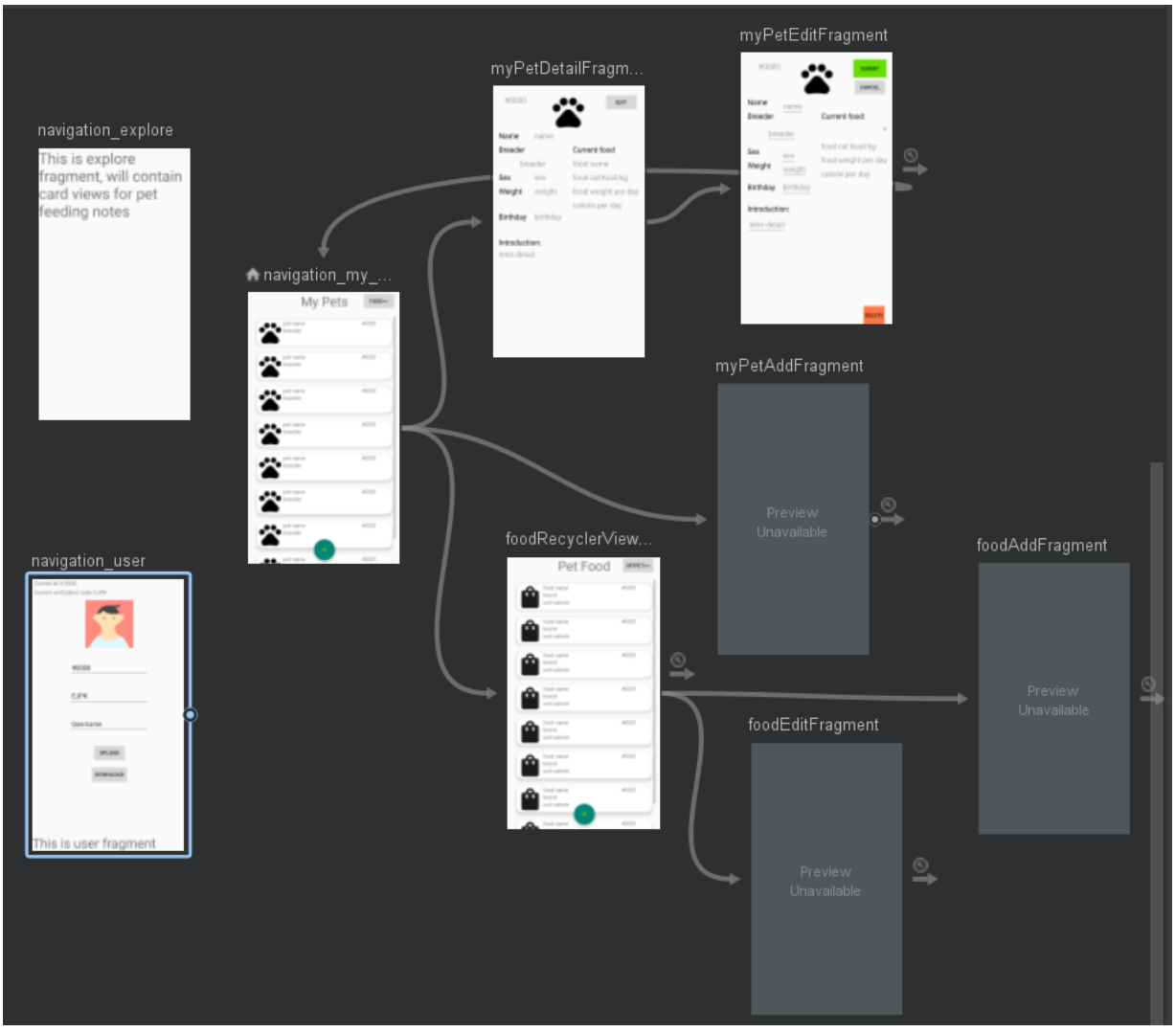
Title	Post folder
Description	As a user, I want to find all posts related to one specific stray animal in a folder or an easy way.
Mockups	
Acceptance tests	Don't have a detailed idea yet
Test Result	
Status	Iteration 1: ● Nothing new Iteration 2: ● Nothing new Iteration 3: ● Nothing new

Title	Post follower
Description	As a user, I want to make a new post for a specific stray animal to update its information.
Mockups	
Acceptance tests	Don’t have a detailed idea yet
Test Result	
Status	Iteration 1: ● Nothing new Iteration 2: ● Nothing new Iteration 3: ● Nothing new

Design and Implementation

This project is a pet management application, it has two major targets and a minor target.
1.[Major] Record pet body information, food information and calculate the daily calories and food intake amount based on body weight and food.
2.[Major] Record owner's feeding notes.
3.[Minor] sync data with cloud

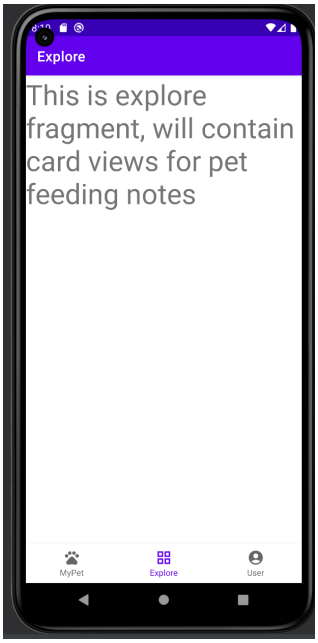
This application uses MVVM structure to manage data. The navigation graph of this project is shown below **NAV-1**.



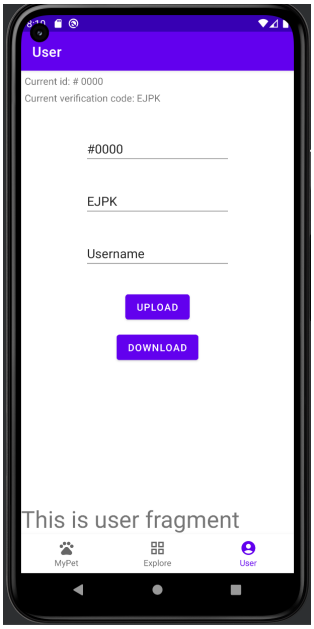
NAV-1

To manage the two targets in the app, create a bottom navigation in the main activity. The bottom navigation contains three tabs, as shown in **IMG-1.1 ~ IMG-1.3.**

IMG-1.1



IMG-1.2



IMG-1.3

The bottom navigation code in MainActivity is displayed in **IMG-1.4**. Using binding to binding the mainActivity xml and mainActivity, created a navigation controller in mainActivity, and set the three tabs.

```
class MainActivity : AppCompatActivity() {


    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val navView: BottomNavigationView = binding.bottomNavBar

        val mainNavController = findNavController(R.id.nav_host_fragment_activity_main)

        val appBarConfiguration = AppBarConfiguration(
            setOf(
                R.id.navigation_my_pet, R.id.navigation_explore, R.id.navigation_user
            )
        )
        setupActionBarWithNavController(mainNavController, appBarConfiguration)
        navView.setupWithNavController(mainNavController)
    }

}
```

IMG-1.4

Then build the pet list recycler view. It needs to list the pet information card view. To create the card view, build a data class Pet, which contains pet body information. Created a database , related application, repository dao, adapter and card view holder class to store and manage the pet data into the database. The key related codes are displayed in the following table.

IMG-2.1 ~ IMG-2.6

<table>
<tr><td>

```
@Entity(tableName="myPets")
data class Pet(
    @PrimaryKey(autoGenerate = true) val id: Int,
    var name: String,
    var photo: String,//for now
    var breeder: String,
    var sex: String,
    var weight: String,
    var birthday: String,
//    var currentFood: Food,
    var intro: String,
//    var sugFoodWeight: Int,
//    var sugCal: Int,
){
```

</td><td>

```
@Database(
    entities = [Pet::class],
    version = 1,
    exportSchema = false
)

abstract class TinyPawDatabase:RoomDatabase() {
    abstract fun myPetDao(): MyPetDao
}
```

</td></tr>
<tr><td>

IMG-2.1 Pet data class

</td><td>

IMG-2.2 Tiny Paw database class

</td></tr>
<tr><td>

```
import ...

class TinyPawApplication:Application() {
    lateinit var tinyPawDatabase: TinyPawDatabase
    lateinit var myPetRepository: MyPetRepository
    override fun onCreate() {
        super.onCreate()
        tinyPawDatabase = Room.databaseBuilder(
            applicationContext, TinyPawDatabase::class.java,
            name: "tinyPaw-db"
        ).addCallback(object: RoomDatabase.Callback(){  //a callback to m
            override fun onCreate(db: SupportSQLiteDatabase){
                super.onCreate(db)
                defaultMyPets()
            }
        }
        ).build()

        myPetRepository = MyPetRepository(tinyPawDatabase.myPetDao())

    }
```

</td><td>

```
@Dao
interface MyPetDao {

    @Insert( onConflict = OnConflictStrategy.REPLACE)
    fun addPet(myPet: Pet)

    @Delete
    fun removePet(pet:Pet)

    @Update
    fun updatePet(pet:Pet)

    @Query("SELECT * FROM myPets")
    fun getAllMyPets(): LiveData<List<Pet>>

    @Query("SELECT count(*) From myPets")
    fun count(): LiveData<Int>

    @Query("SELECT * FROM myPets where id =:petId")
    fun getPetById(petId: Long): LiveData<Pet>
}
```

</td></tr>
<tr><td>

IMG-2.3 Application class

</td><td>

IMG-2.4 Pet Dao class

</td></tr>
<tr><td>

```
class MyPetCardViewHolder(
    private val cardMyPetBinding: CardMyPetListBinding
): RecyclerView.ViewHolder(cardMyPetBinding.root) {

    val cardView: CardView = cardMyPetBinding.myPetCard
//    val idView : TextView = cardMyPetBinding.cardMyPetPetId
////    val photoView : ImageView         TODO("set photo as string for now")
//    val nameView : TextView = cardMyPetBinding.cardMyPetName
//    val breederView: TextView = cardMyPetBinding.cardMyPetBreeder

    fun bindPet(pet:Pet){
        cardMyPetBinding.cardMyPetPetId.text = pet.id.toString()
//        TODO("set photo as string for now")
//        photoView.setImageResource(pet.photo)
        cardMyPetBinding.cardMyPetName.text = pet.name
        cardMyPetBinding.cardMyPetBreeder.text = pet.breeder

    }
```

</td><td>

</td></tr>
<tr><td>

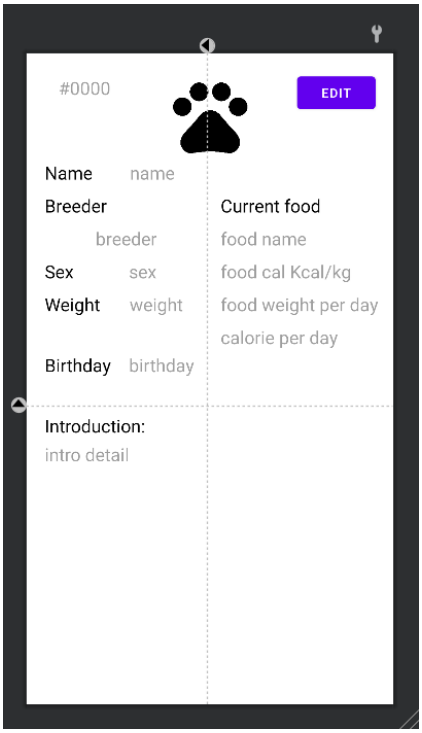
IMG-2.5 Card Viewholder class

</td><td>

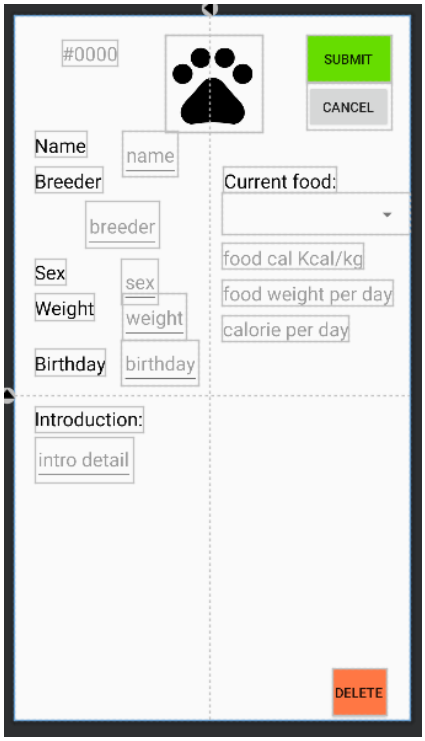
IMG-2.6 card view xml result

</td></tr>
</table>

For user add and edit pet information, the MVVM structure is used to manage data. There is an adapter to link data between database and viewmodel. Users can click the “+” fab and get navigated to add a new pet page. Coded a pet detail view to pop up when the user clicks the pet card view in the recycler list. Detail view has the basic pet information users have added, and an Edit button for navigating to a pet edit page. Users can edit pet information or remove current pets from the database here. Users can choose current feeding food from the current food spinner, which takes data from the database food table. The detail page and edit page xml view and viewmodel codes are displayed in the following table. **IMG-3.1 ~ IMG-3.4**



IMG-3.1 Pet Detail View



IMG-3.2 Pet Edit View

```
class CurPetViewModel(application: Application): AndroidViewModel(application) {
    private val _curPet: MutableLiveData<Pet> = MutableLiveData()
    val curPet: LiveData<Pet>
        get()=_curPet

    val MyPetRepository = (application as TinyPawApplication).myPetRepository

    fun initCurPet(pet:Pet){
        if(_curPet.value == null)
            _curPet.value == pet
    }

    fun setCurPet( pet: Pet){
        _curPet.value = pet
    }

    fun getCurPet(): Pet{
        return _curPet.value!!
    }

    fun isCurPet(pet:Pet):Boolean{
        return _curPet.value?.id == pet.id
    }


    fun updateCurPet(name: String, photo: String, breeder: String, sex: String,
        _curPet.value = _curPet.value?.apply {  this: Pet
            this.name = name
            this.photo = photo
            this.breeder = breeder
            this.sex = sex
            this. weight = weight
            this.birthday = birthday
            this.intro = intro
        }
        MyPetRepository.updatePet(_curPet.value!!)
```

IMG-3.3 Current Pet ViewModel code

```
    override fun onViewCreated(view:View, savedInstanceState: Bundle?)
    {
        super.onViewCreated(view,savedInstanceState)

        curPetViewModel =
            ViewModelProvider(requireActivity()).get(CurPetViewModel::class.java)
        petListViewModel =
            ViewModelProvider( owner: this).get(MyPetRecyclerViewViewModel::class.java)

        binding.myPetList.apply {  this: RecyclerView
            layoutManager = LinearLayoutManager(context) //layout management

            petAdapter = MyPetCardAdapter(
                object: MyPetCardAdapter.OnPetClickListener{
                    override fun onPetClick(pet: Pet) {
                        curPetViewModel.setCurPet(pet)
                        view.findNavController().navigate(R.id.action_myPetRecyclerViewFragm
//                          onPetClickListener?.onPetClick(pet)    //TODO("failed to used slide
                    }
                }
            )
            this.adapter = petAdapter //attach adapter to the recyclerView

            //view model initial pass
            petListViewModel.petList.observe(viewLifecycleOwner) {  it: List<Pet>!
                petAdapter.updatePets(it)
                curPetViewModel.initCurPet(petAdapter.getPet( pos: 0))
            }

            //pass list
            petListViewModel.petList.observe(viewLifecycleOwner) {  it: List<Pet>!
                petAdapter.updatePets(it)
                curPetViewModel.initCurPet(petAdapter.getPet( pos: 0))
            }

            curPetViewModel.curPet.observe(viewLifecycleOwner) {  it: Pet!
                petAdapter.notifyDataSetChanged()
            }

        }
    }
}
```

IMG-3.4 Using ViewModel in list fragment

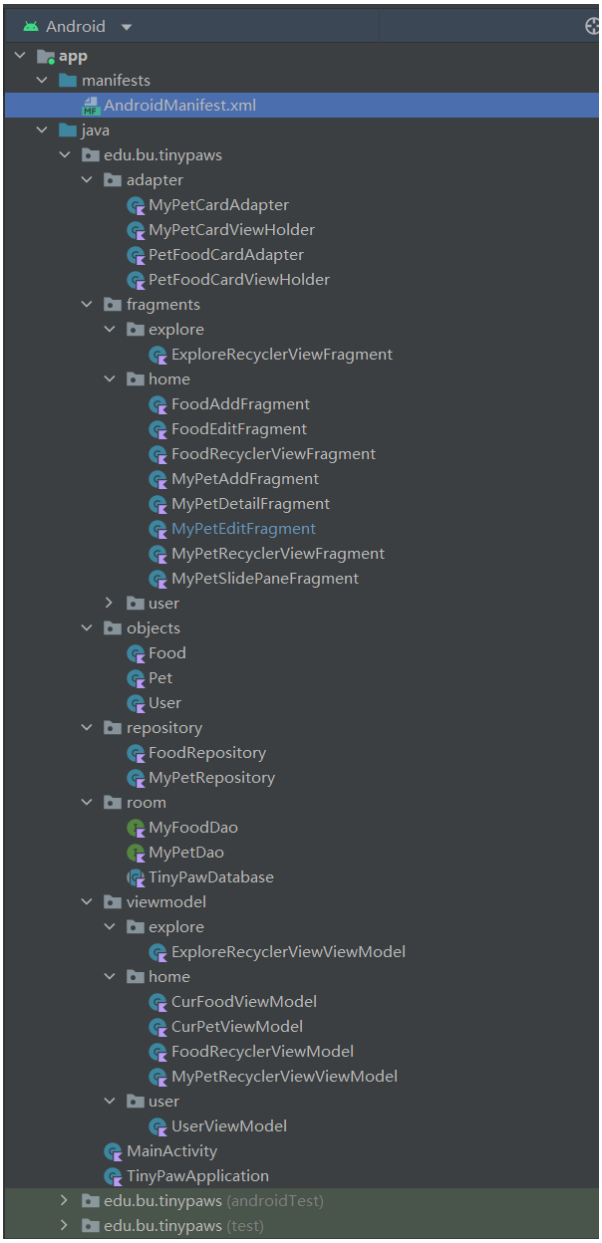
```
//initialize spinner
private fun initSpinner() {

    CoroutineScope(Dispatchers.Default).launch {  this: CoroutineScope
        val foodOptions : List<String> = foodListViewModel.getAllMyFoodsName()
        val foodAdapter: ArrayAdapter<String>? =
            context?.let {  it: Context
                ArrayAdapter(it,android.R.layout.simple_spinner_item,foodOptions) }
        foodAdapter?.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
        foodAdapter?.notifyDataSetChanged()
        foodSpinner.adapter = foodAdapter
        defaultSpinnerValue(foodOptions.size)
    }
    binding.myPetEditSpinner.onItemSelectedListener = this
}
```

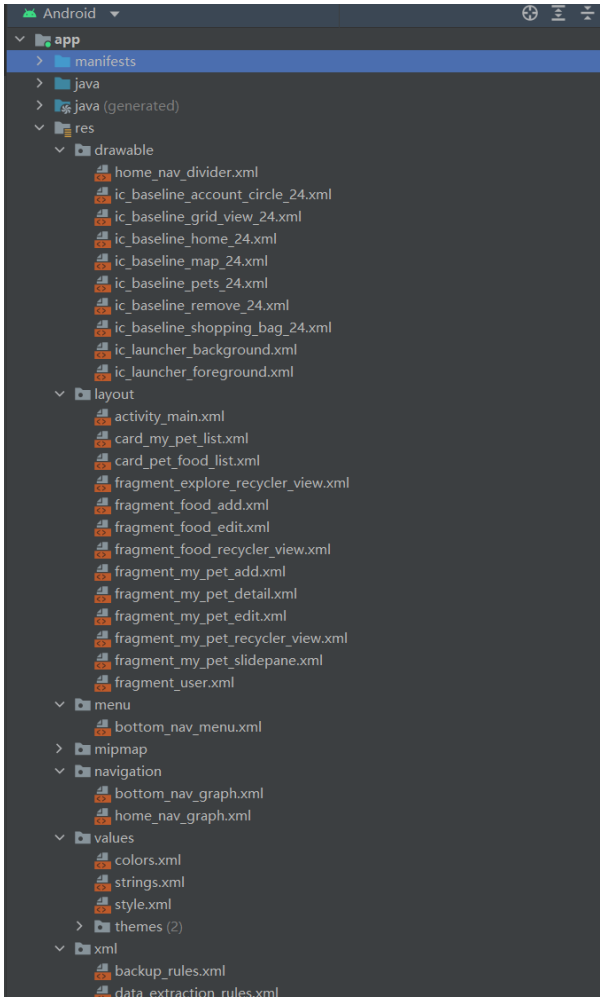
IMG-3.5 Initialize spinner in fragment edit pet

```
    foodListViewModel.foodList.observe(viewLifecycleOwner){  it: List<Food>!
        for(i in 0 until it.count()){
            if(it[i].name.equals(foodName)){
                binding.petDetailFoodCalContext.setText(it[i].calPerUnit.toString()+" Kcal/kg")
                var calIntake: Double = 1.2*(weight*30+70)*0.8          //calculate calorie intake per day
                var foodIntake: Double = 1000*(calIntake)/it[i].calPerUnit //calculate food weight per day
                binding.petDetailFoodWeightContext.setText(foodIntake.toInt().toString()+" g/day")
                binding.petDetailCalIntakeTitle.setText(calIntake.toInt().toString()+" kcal/day")
            }
        }
    }
```

IMG-3.6 Get food information and Calculate feeding suggestion in PetDetailFragment

Project Structure





Project structure -1	Project structure -2

TimeLine

Iteration	Application Requirements	Android Components and Features	Agam Purohit contribution/tasks	Jipeng Liu contribution/tasks
0	Project proposal Ver1.0			Come up the project idea and write project proposal
1	Project proposal Ver1.1			Add another part of Related work. Re-do the Essential goal tables.
1	Mind mapping			Created the mind mapping of the project. Briefly show the project features, and basic idea of ERD
1	UI mockups			Created the UI mockups, shows the UI outlook, and interactions between different pages.
1	Activity/fragment mockups			Designed the project activities and fragment mockups based on the UI mockups.
2	Health management	Bottom navigation bar to navigate three fragments: my pet, explore, user		Coded the bottom_nav.xml Set home as the default fragment, and be able to switch screen between home, map, user.
2	Multiple pets	Database, repository, adapter and view holder,		Developed database , data, repository,

		recycler view, and view model		adapter, viewholder, viewmodel, and recycler view fragment classes for pets. Coded the pet card view, pet recycler list view, Created the navigation graph for these pages.
2	Account system			Re-designed the account logic and coded the user.xml
2	User profile editing			Combined into one page, user can only edit username
2	Pets profile add/ edit/ delete	Pet information card view, pet recycler view, pet list view model, Pet view model, Pet detail fragment, Pet edit fragment.		Developed pet detail fragment, pet edit fragment, Coded individual pet detail, individual pet edit xml files, and created the related navigation graph.
3	Food profile add/ edit/ delete	Food information card view, food recycler view, food list view model, food view model, food edit fragment.		Developed food edit fragment, Coded individual food edit xml files, and created the related navigation graph.
3	Get food information and Set a food in pet edit page	Current food spinner		Developed a sniper in pet edit page, which can get food list from database through viewmodel.
3	Calculate feed suggestion	A view model observer		Developed a food VM observer in

				pet detail page, which can pass data from DB, and calculate it with math formula
3	Fixed known bugs from Iteration 2			Fixed navigation crash bugs. Fixed submit empty data makes the app crash.
3	Text Post			
3	Photo Post			

Future Work

In the next work duration, I will focus on making the Mypet part more functional.

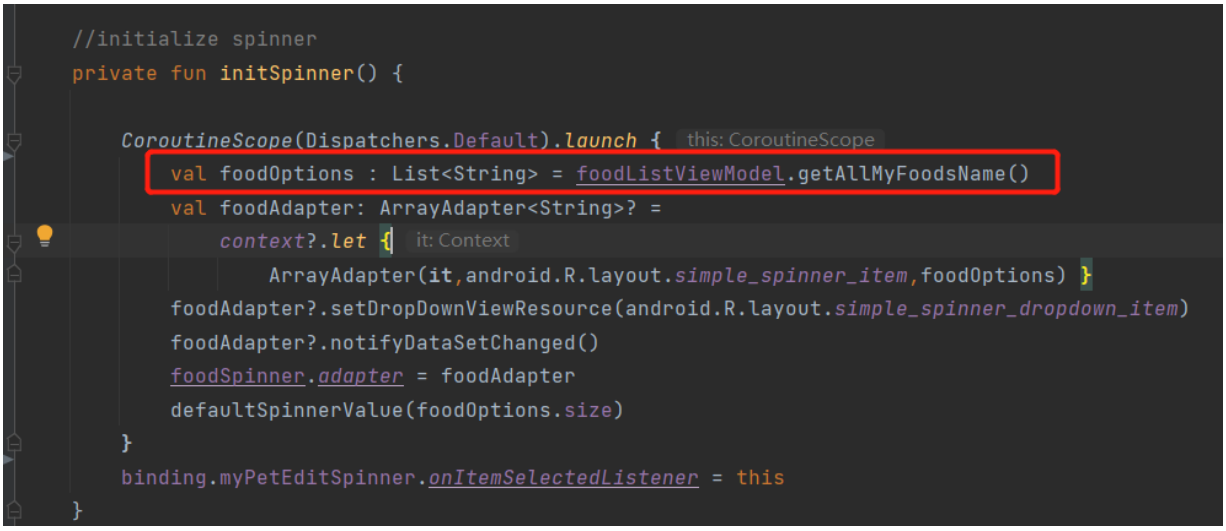
1. Keep updating attributes data type in data class
2. Link two database tables with foreigner key and relation classes.
3. Work on notes part and cloud synchronize part.

Questions and bug

1.**Some questions for viewmodel.**

I feel I was sort of trapped in the class code sample. I want to know do we have to use an observer to observe the data in the viewmodel. As the code shown in **Q-1.** For this project I planned to make the data class `Pet` has an attribute `food :Food`. I read the android developer web but I don't have enough time to try the foreigner key and database relation. So, I use `food:String` for now. I have to get the food String from Pet first, then use the string and find the food from the database. The problem is, the string value only exists in the pet viewmodel observer, and I can't create another observer for the food viewmodel in the current viewmodel. So, I just created a series of methods from viewmodel to DAO to get the List<String>.

Is that ok if I just get data from a viewmodel without using an observer? It works at this moment, but I'm not sure if this meets the android development rules.



Q-1

References

Class lab