THE COOKIE CLUB – NEKO GAINS
SYSTEM DESIGN DOCUMENT

Roy · Arthur · Jeremy · Kana · Coco · Leila

TABLE OF CONTENTS

CLASSES – CRC CARDS

Class name: Cat

Parent class: Pet

Subclasses: N/A

Responsibilities:

- A different version of pet but with different visuals
- Pet class: stores the name, hunger, and level

Collaborators:

- Pet
- User

Class name: DatabaseHelper

Parent class: SQLiteOpenHelper

Subclasses: N/A

Responsibilities:

- Create database table for workouts
- Populate workout table available workouts along with their calorie burn

Collaborators:

- User
- Exercise
- UserInventory
- HomeFrag
- ProgressFrag
- SettingFrag
- StoreFrag

Class name: Exercise

Parent class: N/A

Subclasses: N/A

Responsibilities:

- Stores the name and calories burned

Collaborators:

- User
- DatabaseHelper

Class name: HomeFrag

Parent class: Fragment

Subclasses: N/A

Responsibilities:

- Runs the home page and updating the information in it

Collaborators:

- MainActivity
- User
- UserInventory

Class name: LoginActivity

Parent class: N/A

Subclasses: N/A

Responsibilities:

- Gets a user class from the database

Collaborators:

- MainActivity
- User
- UserInventory

Class name: MainActivity

Parent class: N/A

Subclasses: N/A

Responsibilities:

- Runs the main page and other pages

Collaborators:

- Login
- HomeFrag
- PreworkoutFrag
- ProgressFrag
- SettingFrag
- StoreFrag
- User
- UserInventory

Class name: Pet

Parent class: N/A

Subclasses: Cat

Responsibilities:

- Stores the name of the pet
- Stores the hunger of the pet
- All the interaction methods of the pet: feeding, petting

Collaborators:

- Cat
- User

Class name: PreworkoutFrag

Parent class: Fragment

Subclasses: N/A

Responsibilities:

- Runs the pre-workout page

Collaborators:

- MainActivity

Class name: ProgressFrag

Parent class: Fragment

Subclasses: N/A

Responsibilities:

- Runs the progress page
- Displays information taken from the database

Collaborators:

- MainActivity
- User
- UserInventory

Class name: Questionnaire

Parent class: AppCompatActivity

Subclasses: N/A

Responsibilities:

- Get’s information from the user when they sign up

Collaborators:

- User

Class name: SettingFrag

Parent class: Fragment

Subclasses: N/A

Responsibilities:

- Runs the setting page
- Updates the user information

Collaborators:

- MainActivity
- User
- UserInventory

Class name: StoreFrag

Parent class: Fragment

Subclasses: N/A

Responsibilities:

- Runs the store page

Collaborators:

- MainActivity

Class name: User

Parent class: N/A

Subclasses: N/A

Responsibilities:

- Stores the login information: username, password
- Stores physical information of the user: height, weight
- Stores a dictionary of Exercise Plans
- Stores amount of money in game the user has
- Stores the user's pet
- Will use the user's information to calculate BMI

Collaborators:

- Cat
- Pet
- Exercise
- Login Activity
- Registration
- UserInventory
- SettingFrag
- ProgressFrag
- HomeFrag
- Questionnaire

Class name: UserInventory

Parent class: N/A

Subclasses: N/A

Responsibilities:

- Stores the pet’s clothing, food, money

Collaborators:

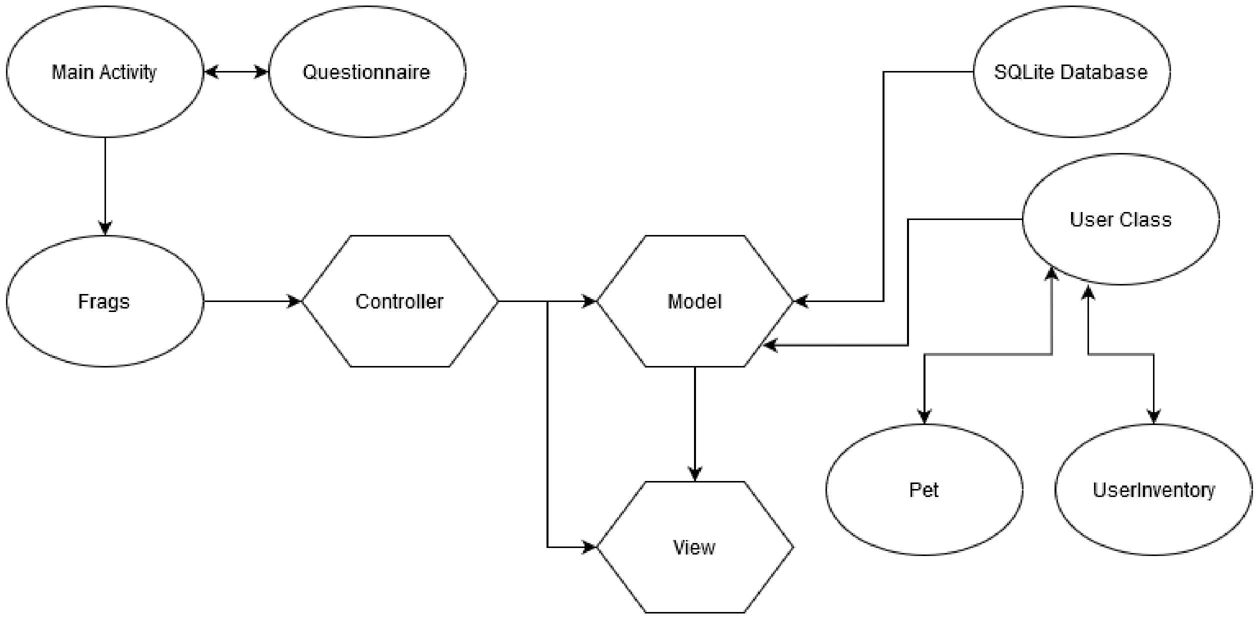
- Cat
- Pet
- Exercise
- Login Activity
- Registration
- User
- SettingFrag
- ProgressFrag
- HomeFrag

SYSTEM INTERACTION WITH ENVIRONMENT

Our app is strictly designed to be used in an Android environment, so we expect android touch devices such as phones and tablets to run our app. We assume that the version of android that will be running our app to be Android Q. The app will be using an SQLite3 database which will not be required to be ran externally but instead will be stored internally as a .db document. For the purposes of logging in and cloud file tracking, our app depends on Google Play Services. As this is an android studio project the app will be written and compiled using Java and Java compiling.

SYSTEM ARCHITECTURE

The architecture of our app will be using the MVC architectural pattern. Within the activities and frags contain the choosing of workouts and enacting workouts. Actions such as completing a workout will have the controller update the model user class and update the workout database which are required for the workout planning. The view comprises of the xml elements of the activities that will be updated accordingly based off the model and controller. For example, the stats of the player will be updated on the view as the player progresses. The User class contains a database helper that is used to talk to the database. The User then has all its getters and setters implemented to query the database as to have the data stored there as oppose to in the class itself.



There are multiple errors that our app will be required to handle. We require user input at multiple points during the use of our app including the filling out of the questionnaire and the naming of the virtual pet. We will be handling input through regular expressions and deny input that doesn't match. If there is little to no internet access, the app will allow basic use of the app without login requirements. The app will not be able to acquire profile information without information but will allow creation of workout plans. We will have error catchers for any exemptions that happen and upon exception, save the state of the app and important information and close the app with a relevant error message. The questionnaire also checks for whether there are empty fields that aren't filled in and stops it from submitting and erroring.