2020

Neko Gains System Design Document

THE COOKIE CLUB

Roy Lu Roy · Arthur · Jeremy · Kana · Coco · Leila

Contents

Classes – CRC Cards

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:

- Pet
- Exercise
- LoginActivity
- CalculateUserScore
- Questionnaire
- UserInventory
- WorkoutActivity

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:

- User
- Cat
- UserInventory

Class name: Exercise

Parent class: N/A

Subclasses: N/A

Responsibilities:

- Stores the name of the exercise and general information
- Stores the number of calories burned in one rep and recommended reps

Collaborators:

- 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: N/A

Class name: MainActivity

Parent class: N/A

Subclasses: N/A

Responsibilities:

- Runs the main page and other pages

Collaborators:

- LoginActivity
- HomeFrag
- PreworkoutFrag
- ProgressFrag
- SettingFrag
- StoreFrag

Class name: HomeFrag

Parent class: Fragment

Subclasses: N/A

Responsibilities:

- Runs the home page

Collaborators:

- MainActivity

Class name: PreworkoutFrag

Parent class: Fragment

Subclasses: N/A

Responsibilities:

- Runs the pre-workout page

Collaborators:

- MainActivity
- preworkout_item

Class name: ProgressFrag

Parent class: Fragment

Subclasses: N/A

Responsibilities:

- Runs the progress page

Collaborators:

- MainActivity

Class name: SettingFrag

Parent class: Fragment

Subclasses: N/A

Responsibilities:

- Runs the setting page

Collaborators:

- MainActivity

Class name: StoreFrag

Parent class: Fragment

Subclasses: N/A

Responsibilities:

- Runs the store page

Collaborators:

- MainActivity

Class name: LoginActivity

Parent class: N/A

Subclasses: N/A

Responsibilities:

- Gets a user class from the database

Collaborators:

- User
- MainActivity

Class name: CalculateUserScore

Parent class: N/A

Subclasses: N/A

Responsibilities:

- User score decides intensity of the workout plan for the user

Collaborators:

- User

Class name: Cat

Parent class: Pet

Subclasses: N/A

Responsibilities:

- A type of pet

Collaborators:

- Pet

Class name: Questionnaire

Parent class: AppCompatActivity

Subclasses: N/A

Responsibilities:

- Gets new user's information and puts them into the database

Collaborators:

- User

Class name: UserInventory

Parent class: N/A

Subclasses: N/A

Responsibilities:

- Stores the inventory of the pet and user

Collaborators:

- Pet
- User

Class name: WorkoutActivity

Parent class: AppCompatActivity

Subclasses: N/A

Responsibilities:

- Where workouts will be run

Collaborators:

- User
- CounterFrag
- CustomWorkoutFrag

Class name: CounterFrag

Parent class: Fragment

Subclasses: N/A

Responsibilities:

- Is counting the number of reps

Collaborators:

- WorkoutActivity

Class name: CustomWorkoutFrag

Parent class: Fragment

Subclasses: N/A

Responsibilities:

- Is for creating your own custom workouts

Collaborators:

- WorkoutActivity

Class name: EndWorkoutFrag

Parent class: Fragment

Subclasses: N/A

Responsibilities:

- For the end of the workout
- Shows how much money and xp earned

Collaborators: N/A

Class name: preworkout_item

Parent class: N/A

Subclasses: N/A

Responsibilities:

- Is an xml that shows the layout of the card views on the PreworkoutFrag

Collaborators:

- PreworkoutFrag

Class name: preworkoutAdapter

Parent class: RecyclerView.Adapter<preworkoutAdapter.preworkoutViewHolder>

Subclasses: N/A

Responsibilities:

- Java code to create the card views with the workout name, sets and reps

Collaborators: N/A

Class name: TimerFrag

Parent class: Fragment

Subclasses: N/A

Responsibilities:

- Is for workouts that require a timer

Collaborators: N/A

Class name: TutorialFrag

Parent class: Fragment

Subclasses: N/A

Responsibilities:

- Is where the tutorials go

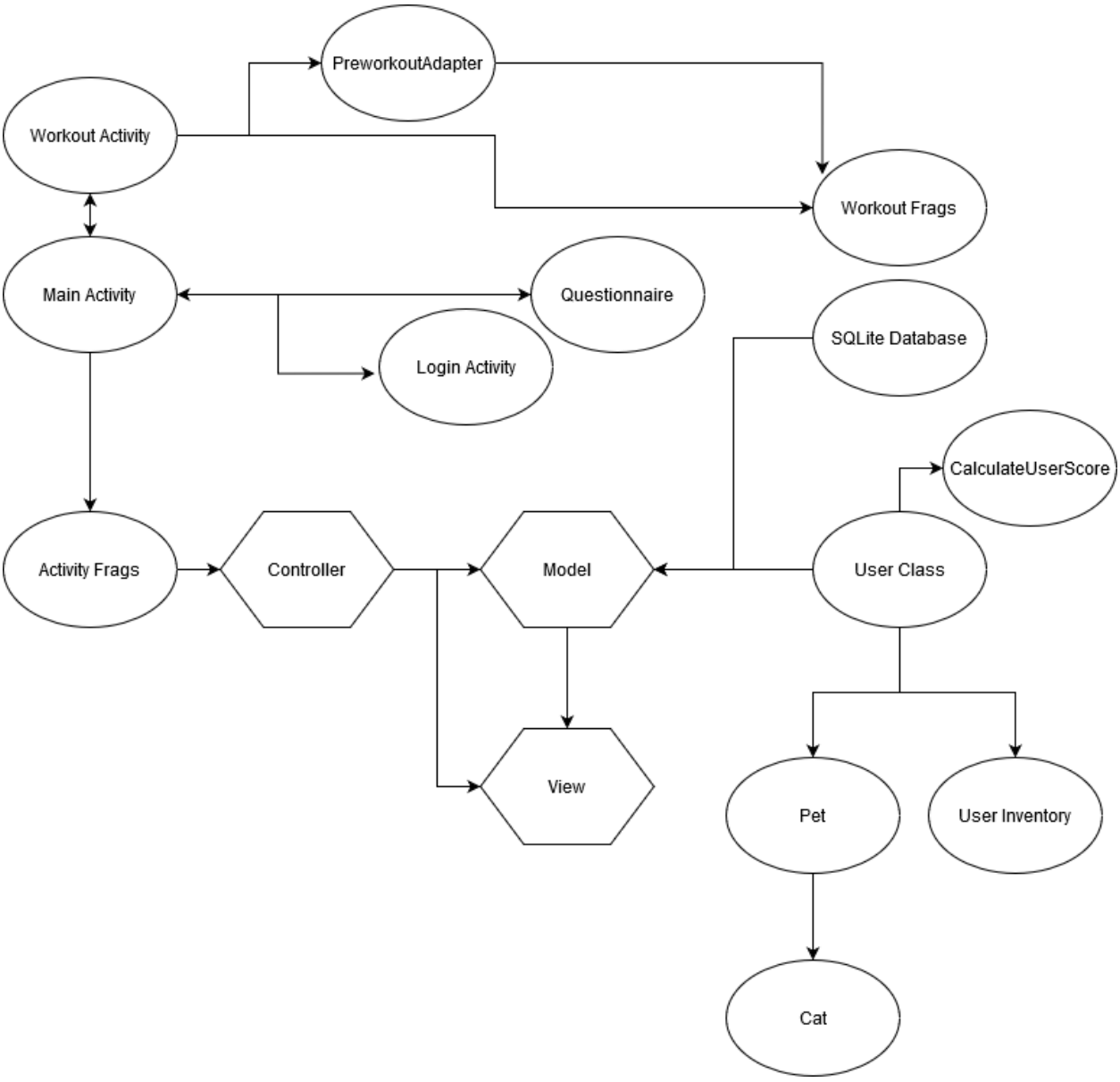
Collaborators: N/A

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. The main windows within the app (Home Page, Settings Page, Workout Page, Stats Page) are modules of the main activity screen so they are set up as frags of the main activity as they are closely related. The workout activity is a titular component of the app and is therefore set up as its own activity with its own frags that serve for the tutorial and rep count pages.



System Decomposition

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 and settings page include text fields where the user can enter any text. There is error checking and constraints placed on these text fields so that they don't crash the app trying to parse an empty field or if the fields contain unexpected characters. There are also possible errors that can occur when there are inconsistencies within the database. There are foreign keys in place to keep up the database integrity and error checking that rejects invalid queries to the database.