Bounty Hunter App - Project Documentation

Gutter-Bacsi Zsombor

April 20, 2025

Contents

1 Project Introduction

Gutter-Bacsi Zsombor Git Hub `https://github.com/GutterZsombor/OOP-project`
002368012
Zsombor.Gutter-Bacsi@student.lut.fi

My Project is a Star Wars Bounty Hunters themed simple "trading cards" game-like Android application. The application is designed to manage, train, and simulate LOCAL and ONLINE battles between bounty hunters. The app uses minimalistic style user interface with many activities like:

- Hiring hunters,
- Training hunters,
- Displaying statistics,
- Online and Local battles

Data is stored and fetched from JSON files.
Network functionality is used for multiplayer battles

2 Overview + UML

The Documentation and the code explanation videos got a little too long I went into too much detail. Sorry for that.

Detailed code explanation videos can be found on page 10. They are very Long.
Don't need to watch them all!.
Please watch this 4 demonstration videos.
Intro*
Link INTRO (2 min)
`https://1drv.ms/v/c/3e8ee7900513c457/EfPBauxDTPlDi98EwaZZDZUB5Jo7q7J7hJldl0xMI0VsSA?e=PC08Fv`
Run on Emulator*
Part one clean run on emulator Link Part One (10 min)
`https://1drv.ms/v/c/3e8ee7900513c457/ETBQeumwX4pDjchp3FE87iQBZn290XkpthvIAhQaYRolfQ?e=f4eoyz`
Part two played for a few minutes generated some Statistical data: Link Part Two (3 min)
`https://1drv.ms/v/c/3e8ee7900513c457/EfuNamTWdIRAvEqJnRuw7k0Buthm7KHgBl168SVMYXVMAw?e=0KMjxK`
Run on Physical Device Online Battle*
Link Physical Device Online (5 min)
`https://1drv.ms/v/c/3e8ee7900513c457/EfYlHAF6diZCiwvDGvgnU9EB6uZGeLbwG56ouKLNzzXCDA?e=61cD0x`

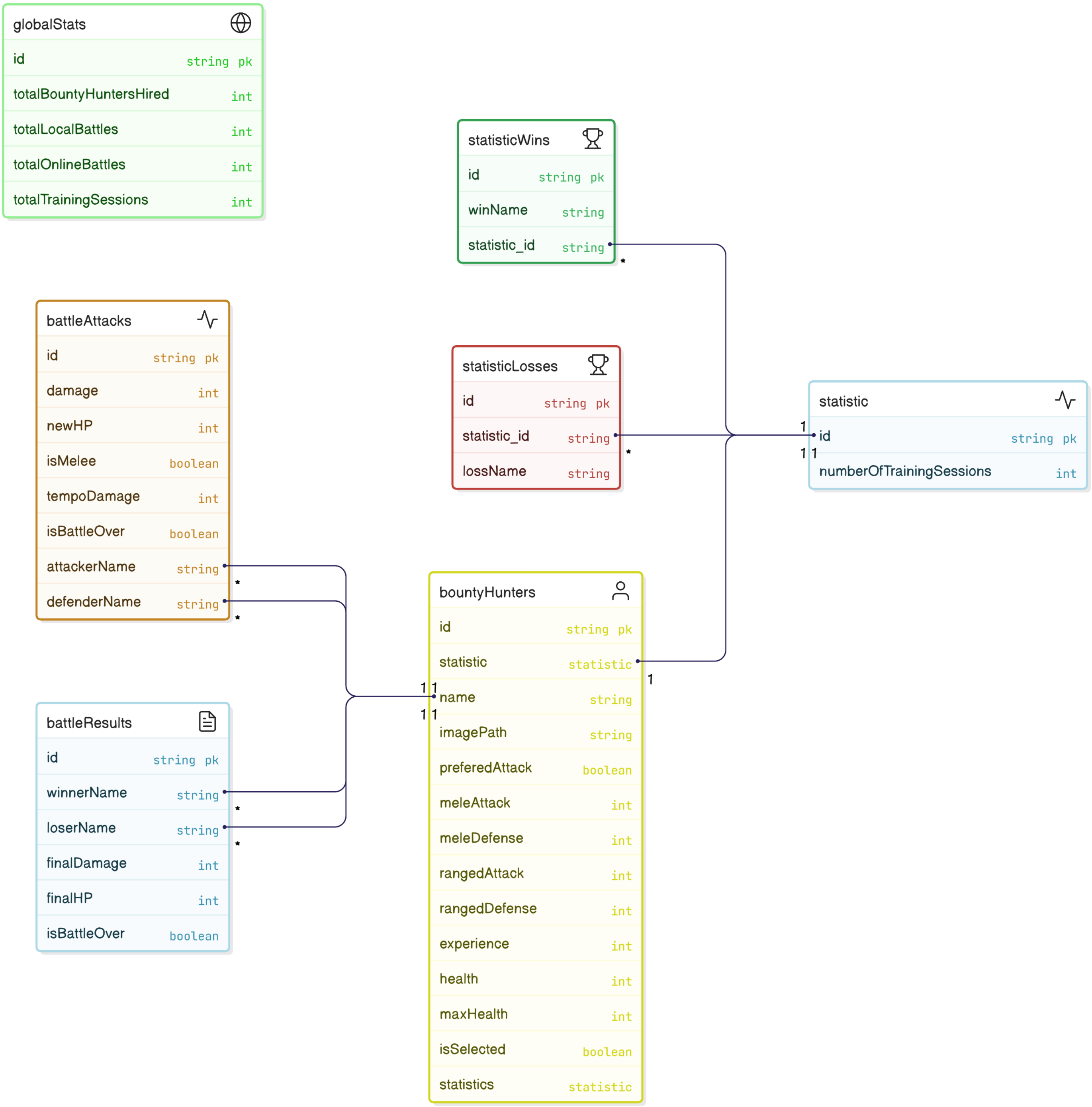
If something didn't make sense in the demonstration video please check it in this document. I tried to go trough everything on a not technical surface level. If something still doesn't make sense Check the detailed videos i go over the code.
Code is commented.

Check List on all the implemented Bonus feature:

RecyclerView	Implemeted: Home Activity, Hire Activity, Statistics Activity
Bounty Hunters Have Images	Implemeted
Battle Visualizaton	Attacking hunter has White background, Defending Grey, Winner Green, Loser Red
Turn-based combat	Attack Button needs to be Pressed For every attack
Statistics	Implemeted: Statistics Activity
Randomness in Battles	Implemented: deciding melee/ranged attack (60 percent chance for preferred)
Fragments	Implemented: all the different card view items
Data Storage Loading	Implemeted: Json files and JsonHelper
Statistics Visualization	Implemented: Charts
Custom Feature X - Network	Implemented: Multiplayer Battle mode --> Grade: +2

UML: (1. Class Diagram, 2. Activity Diagram, 3. Old UML)

OOP Class Diagram



globalStats	
id	string pk
totalBountyHuntersHired	int
totalLocalBattles	int
totalOnlineBattles	int
totalTrainingSessions	int

statisticWins	
id	string pk
winName	string
statistic_id	string

battleAttacks	
id	string pk
damage	int
newHP	int
isMelee	boolean
tempoDamage	int
isBattleOver	boolean
attackerName	string
defenderName	string

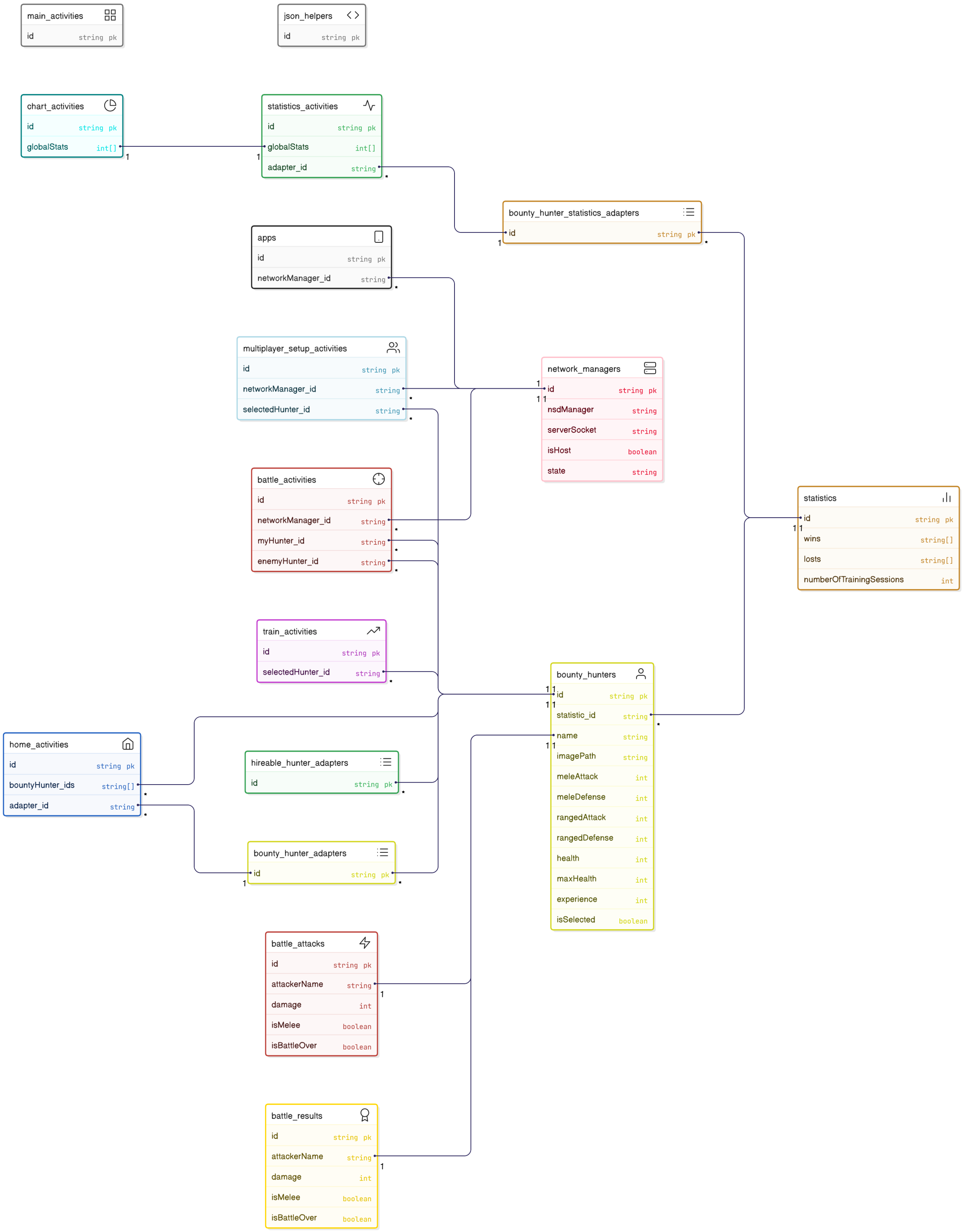
statisticLosses	
id	string pk
statistic_id	string
lossName	string

statistic	
id	string pk
numberOfTrainingSessions	int

bountyHunters	
id	string pk
statistic	statistic
name	string
imagePath	string
preferedAttack	boolean
meleAttack	int
meleDefense	int
rangedAttack	int
rangedDefense	int
experience	int
health	int
maxHealth	int
isSelected	boolean
statistics	statistic

battleResults	
id	string pk
winnerName	string
loserName	string
finalDamage	int
finalHP	int
isBattleOver	boolean

Bounty Hunter Activity diagram



main_activities		
id	string	pk

json_helpers		
id	string	pk

chart_activities		
id	string	pk
globalStats	int[]	

statistics_activities		
id	string	pk
globalStats	int[]	
adapter_id	string	

bounty_hunter_statistics_adapters		
id	string	pk

apps		
id	string	pk
networkManager_id	string	

multiplayer_setup_activities		
id	string	pk
networkManager_id	string	
selectedHunter_id	string	

network_managers		
id	string	pk
nsdManager	string	
serverSocket	string	
isHost	boolean	
state	string	

battle_activities		
id	string	pk
networkManager_id	string	
myHunter_id	string	
enemyHunter_id	string	

statistics		
id	string	pk
wins	string[]	
losts	string[]	
numberOfTrainingSessions	int	

train_activities		
id	string	pk
selectedHunter_id	string	

bounty_hunters		
id	string	pk
statistic_id	string	
name	string	
imagePath	string	
meleAttack	int	
meleDefense	int	
rangedAttack	int	
rangedDefense	int	
health	int	
maxHealth	int	
experience	int	
isSelected	boolean	

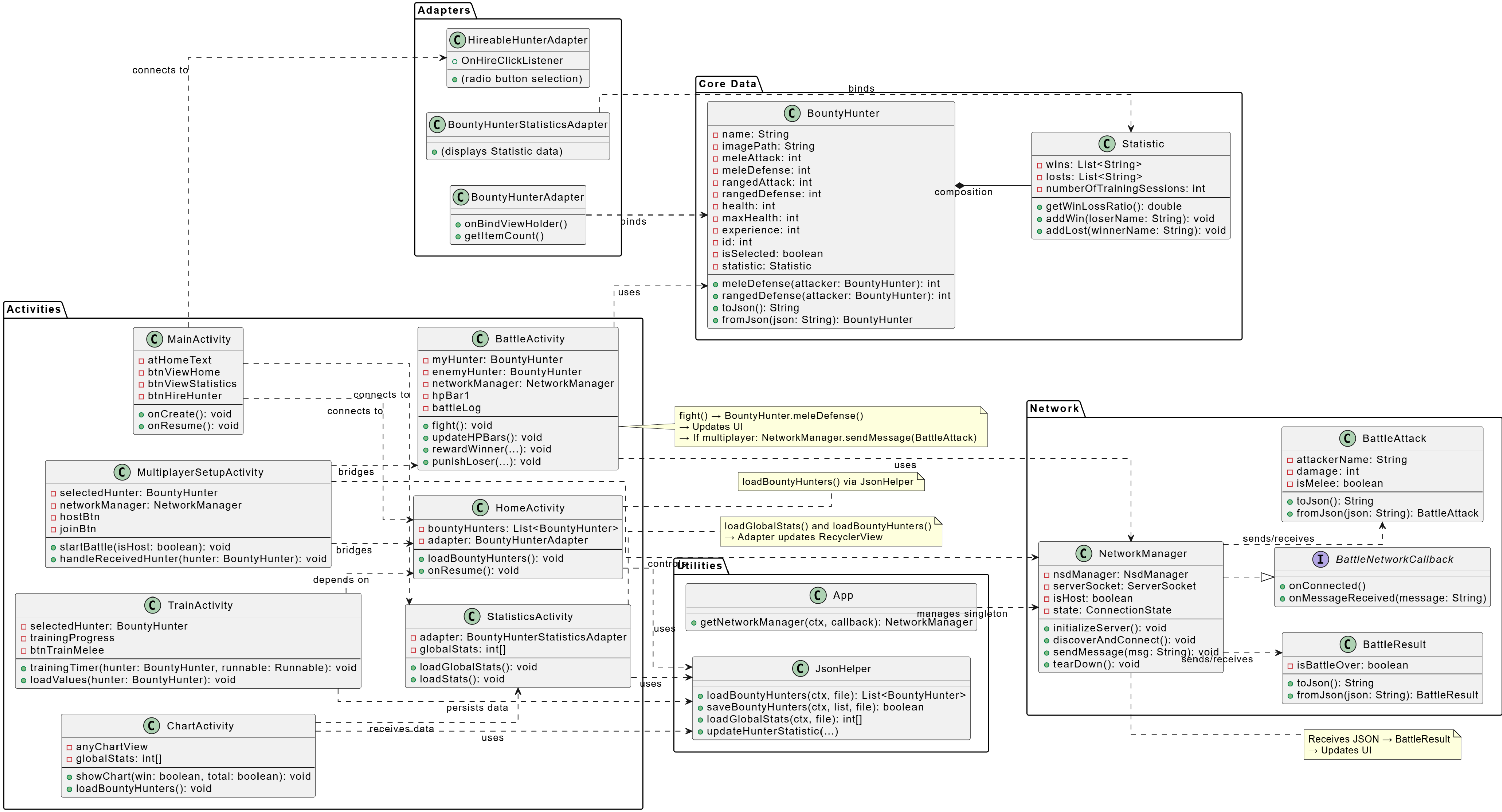
home_activities		
id	string	pk
bountyHunter_ids	string[]	
adapter_id	string	

hireable_hunter_adapters		
id	string	pk

bounty_hunter_adapters		
id	string	pk

battle_attacks		
id	string	pk
attackerName	string	
damage	int	
isMelee	boolean	
isBattleOver	boolean	

battle_results		
id	string	pk
attackerName	string	
damage	int	
isMelee	boolean	
isBattleOver	boolean	


Adapters
HireableHunterAdapter
OnHireClickListener
(radio button selection)
BountyHunterStatisticsAdapter
(displays Statistic data)
BountyHunterAdapter
onBindViewHolder()
getItemCount()
connects to
binds
binds
Core Data
BountyHunter
name: String
imagePath: String
meleAttack: int
meleDefense: int
rangedAttack: int
rangedDefense: int
health: int
maxHealth: int
experience: int
id: int
isSelected: boolean
statistic: Statistic
meleDefense(attacker: BountyHunter): int
rangedDefense(attacker: BountyHunter): int
toJson(): String
fromJson(json: String): BountyHunter
composition
Statistic
wins: List<String>
losts: List<String>
numberOfTrainingSessions: int
getWinLossRatio(): double
addWin(loserName: String): void
addLost(winnerName: String): void
uses
Activities
MainActivity
atHomeText
btnViewHome
btnViewStatistics
btnHireHunter
onCreate(): void
onResume(): void
connects to
connects to
BattleActivity
myHunter: BountyHunter
enemyHunter: BountyHunter
networkManager: NetworkManager
hpBar1
battleLog
fight(): void
updateHPBars(): void
rewardWinner(...): void
punishLoser(...): void
fight() → BountyHunter.meleDefense()
→ Updates UI
→ If multiplayer: NetworkManager.sendMessage(BattleAttack)
uses
MultiplayerSetupActivity
selectedHunter: BountyHunter
networkManager: NetworkManager
hostBtn
joinBtn
startBattle(isHost: boolean): void
handleReceivedHunter(hunter: BountyHunter): void
bridges
bridges
HomeActivity
bountyHunters: List<BountyHunter>
adapter: BountyHunterAdapter
loadBountyHunters(): void
onResume(): void
loadBountyHunters() via JsonHelper
loadGlobalStats() and loadBountyHunters()
→ Adapter updates RecyclerView
depends on
TrainActivity
selectedHunter: BountyHunter
trainingProgress
btnTrainMelee
trainingTimer(hunter: BountyHunter, runnable: Runnable): void
loadValues(hunter: BountyHunter): void
StatisticsActivity
adapter: BountyHunterStatisticsAdapter
globalStats: int[]
loadGlobalStats(): void
loadStats(): void
controls
uses
uses
persists data
ChartActivity
anyChartView
globalStats: int[]
showChart(win: boolean, total: boolean): void
loadBountyHunters(): void
receives data
uses
Utilities
App
getNetworkManager(ctx, callback): NetworkManager
manages singleton
JsonHelper
loadBountyHunters(ctx, file): List<BountyHunter>
saveBountyHunters(ctx, list, file): boolean
loadGlobalStats(ctx, file): int[]
updateHunterStatistic(...)
Network
BattleAttack
attackerName: String
damage: int
isMelee: boolean
toJson(): String
fromJson(json: String): BattleAttack
sends/receives
NetworkManager
nsdManager: NsdManager
serverSocket: ServerSocket
isHost: boolean
state: ConnectionState
initializeServer(): void
discoverAndConnect(): void
sendMessage(msg: String): void
tearDown(): void
BattleNetworkCallback
onConnected()
onMessageReceived(message: String)
sends/receives
BattleResult
isBattleOver: boolean
toJson(): String
fromJson(json: String): BattleResult
Receives JSON → BattleResult
→ Updates UI

3 Individual Section

3.1 Class: BountyHunter and Statistics

`BountyHunter.java` is the most basic object in the application. It stores all the properties of a single hunter, including Statistics. Defense against incoming attacks is also calculated here.

Statistics.java contains individual statistics about a hunter that are only required in specific cases so its only added to the hunter when needed.

3.2 Adapters

In the Application there are tree Adapters for the tree different CardView item that are presented in RecyclerView components.

- BountyHunterAdapter.,
- HireableHunterAdapter,
- BountyHunterStatisticsAdapter

They are very similar in functionality but they are required to be separet because each Card View has different layout and they show different values.

3.3 Data Storage: Json files + JsonHelper

To ensure data doesn't get lost and can be saved the application uses JSON files.

- bountyhunters.json,
- mybountyhunters.json,
- nothiredbountyhunters.json,
- Statistics.json

Why 4 instead of one big? :
Sadly, this question came to mind when i was writing this document. The idea behind the 4 json is they all represent different data and this way it felt better separated in my head.
Of course a single JSON would work also.

`JsonHelper.java` is used for internal data loading, saving, updating. Contains methods and functions.
e.g.: loading Json, saving Json, updating,and of course phrasing the raw-data, returning the desired BountyHunter or Statistics Object.

Methods `copyJsonIfNotExists and DONOTUSEcopyJson`:
These methods are required when Android applications use Json files especially when they need to be modified. It copies the Json file from assets folder to the app's internall storage.
Files in assets are read-only.
`DONOTUSEcopyJson` is not in use but i left it there in case of a "clean run" is required and i want to override all existing files in internal storage.

Why not Gson?:
Why phrasing Json manualy instead of using Gson wich phrases object to Json and back? I didn't use it on the course. And manualy phrasing shows the inner working visually rather than a single line like: "`Gson().toJson(this);`"

Why is it used than?:
For example in BountyHunter.java there are two extra functions toJson() and fromJson(...) they appear in other classes that were not mentioned yet. Simple answer because of network. JsonHelper.java is responible for inner Json related matters (offline). When I started to work on Multiplayer funtionality I had to add additional methods to convert objects into Json and back quickly and simply in order to send them to other devices.

3.4 Activities

Main Activity

This is the Page that Loads when application is launched. Here we have 3 options.

- View Home `-->` We can see our already hired hunters and manage them.
- View Statistics `-->` We can see Statistical data of our hired hunters
- Hire New Bounty Hunter `-->` We can hire new Bounty Hunters

Hire Hunter Activity

On this Page we are presented with not hired bounty hunters their name, base stats are presented in a CardView and with the help of radio buttons we can select a single hunter at a time and hire them, meaning they now work for us we can train them and send them into battle.

`HireableHunterAdapter.java` is used as a RecyclerView adapter.

Hunters are loaded from nothiredbountyhunters.json. And uppon hiring they are removed from this file and moved to mybountyhunter.json.

Statistics Activity

On this Page we are presented with two type of data that is loaded in from `Statistics.json`.

Global Statistics this is the top section where we can see: How many hunters were hired, Number of Local / Online Battles, Number of training Sessions of all the hunters.

Hunter Statistics in this section we have a RecyclerView containing CardView items of Hired hunters we can see thier individual Statistics including wich hunter they defeted and got defeted by. Not hired hunters are not presented here.

`BountyHunterStatisticsAdapter.java` is used as a RecyclerView adapter.

Hunters are loaded from mybountyhunter.json and Statistics.json. So the Bonty Hunter object's Statistic field also gets matched and loaded in. (`loadBountyHunters(), loadStats(), loadGlobalStats()`)

Charts

Charts can be viewed from Statistics Activity. It shows game statistics using AnyChart-Android.
There are 3 Pie-charts in connection with Battle Statistics containing information about Online and Local battles. On these charts we can see all bounty hunters (not only the hired) if they have the necessary data:

- Win Chart `-->` This chart shows out of all the O/L Battles which hunter won what percentage of them.
- Lose Chart `-->` This chart shows out of all the O/L Battles which hunter lost what percentage of them.
- Battle Chart `-->` This chart shows out of all the O/L Battles which hunter participated at what percentage.

Home Activity

This Page is the responsible page to manage our hired hunters. From this point we only deal with hired hunters, loaded from `mybountyhunter.json`

`BountyHunterAdapter.java` is used as a RecyclerView adapter.
In this RecyclerView we can see bounty hunter items. Their picture, name and stats inclduting xp. These stats here now not just their basic stat but their constantly updatated stat wich might come from winning a O/L battle or from training.

Check boxes on the Right Upper corner allowes to select one or multiple hunter.
After one or two hunter selected we can send it/them to train or into Battle.

Train Activity

If a single Hunter is selected we can send it to Train. On this page we are presented with our selected hunter and tree option:

- Train Melee: `-->` This adds +1 experience and +3 to melee attack and defense stats

- Train Range: `-->` This adds +1 experience and +3 to ranged attack and defense stats

- Train XP: `-->` This adds +5 experience and +1 to melee and ranged attack, defense stats

On this Page we also have a custom Progress bar (`drawable/greenprogressba.xml` its also used as hp bars) which shows training time (`trainingTimier(...)`) in in order to protect against "spam" training. Training time depends on experience level in a way higher experience level means longer training sessions.

After each Training session the hunters stats are increased and saved as mentioned also the number of training session gets saved in `Statistics.json` both the total and the individual level number of training sessions.
When we return to Home Activity these updated stats are shown.

BattleActivityOLD

Finally we arrive to the Part which around the application is designed: The Battle.

Why OLD?:

When I developed the app I already knew I want Multiplayer functionality, but i had a different idea (unnecessarily complicated) in mind. This way Online and Local Battle would be handled separately thus the need for two Battle Activity. During implemantation i realized handling both Online and Local Battle is essentially the same so no need for two Activity.
So the Old version is here to explain the how the battle works without Multiplayer mod. It is not used but i kept because its easier to explain battle without multiplayer. So when i explain the Actual in use Battle Activity the basics are already explained and I can focus only the added Multiplayer part.

How Does Simple Battle System works?:

If two Hunters are selected in Home Activity we can select Battle.
When Page is loaded in we are shown the two selected hunter with their respective Hp bars these are same custom bars from Training Activity.

`fight()`

The Battle is turnbase when the single Attack button is pressed `fight()` is called. The fighting algoridm figures out whose turn is it (first turn is the hunter with lower id). It designates the "attacker" and "defender" according to whose tun is it. gives a 60 percent chance for the "preferred attack" (melee or ranged) than calculates the damage according to `BountyHunter.(melee/ranged)Defense(BountyHuner)`.

Tempo Damage is if the attacker manages to damage the opponent above a certain threshold (12 or 16) it adds additional (30 or 20 percent) damage.

After Total Damage is calculated the battle log gets created and added to the Battle Log Text view. Also the Hp bars and the hp text gets updated according to the damage. At the and of round the turns change attacker becomes defender and vice versa.

In case of defeat: defeat means one of the hunters hp is or goes below zero. THE HUNTER DOES NOT DIE! The defeated gets punished and winner gets rewarded.

- `punishLoser(...) -->` hunter gets removed from `mybountyhunter.json` "becomes unhired" returns to `nothiredbountyhunter.json` with its original stats meaning it loses all of its pervious training gains and xp.
  Also in it's statistics into the Defeted by section the winner get's appended and that stays there forever.

- `rewardWinner(...) -->` hunter get's all of it's stats incrased by +10 and gains +3 xp.
  Also in it's statistics into the Defeted section the loser get's appended and that stays there forever.

4 Network

Custom Feature X
I wanted to play this game with my friends so i decided as a custom feature I should add multiplayer mod.
I was debating over 3 options:

- Bluetooth connection
- A cloud platform like firebase
- Or direct socket connections (playing over same wi-fi network)

Eventually I choose direct socket connection because im already familiar with the two other method I wanted to learn something new.

Network Manager

`NetworkManager.java` is the base of everything that is related to multiplayer functionality, including service discovery, socket communication, connection status handling, and sending or receiving both game data and `BountyHunter` objects between devices.
Its a fairly complicated Class so the vide explanation will focus on this more.

`NetworkManager` allows two devices to connect locally over the same network using Android's NSD (Network Service Discovery).
It sets up a server if the device is the host (`initializeServer()`), or discovers and connects to a host if it's a client (`discoverAndConnect(...)`).
Once connected, it manages the exchange of messages and hunters (`sendMessage(...), sendHunter(...), reciveHunterandMessage()`) using sockets, enabling real-time interaction between players during a battle.

App

What is it Why is it needed?:

It extends Android's `Application.` `NetworkManager` is placed here so different activities can still access the same instance. This is relevant when we are transitioning from `MultiPlayerSetupActivity` to `BattaleActivity` If we would declare different `NtworkManager` instances in both activities that would lead to inconsistent network connection and duplicate instances. `setNetworkManager(...)` and `getNetworkManager(...)` makes this doesn't happen.

I had to create this class because the mentioned transition always broke the connection and `BattleActivity` loaded but the connection was lost so the actual battle couldn't start.This also need's to be declared in `AndroidManifest.xml` on the application level rather than on activity or other level.

BattleAttack.java + BattleResult.java

These classes are not very important and the game could easily function without them.
They exist to eas the communication between devices during battle. They are phrased into or from Json messages and sent between devices using `NetworkManager.sendMessage(...)`. They are the mentioned classes in section **3.3**. To convert classes into Json and back I use Gson.

`BattleAttack.java` is responsible to send the damage, newHp, info about attack type and the names to the other device. Because of randomness in attack type (melee/ranged) I can't let the devices botrh calculate locally I need to calculate only on one device than send the message over.

`BattleResult.java` is only sent if one of the hunters got defeated. It contains winner and losers name the last damage and of course the info that the battle is over.

This just like the Json files could have been handled by a single message class but this way its nice and separated. And later I want to add Spectator mod meaning I can view ongoing battles if I'm on same network. Than it makes sense that attack and result is separated.

Multiplayer Setup Activity

When in Home activity a single hunter is selected and Online Battle option is pressed, `MultiplayerSetupActivity.java` is loaded. `NetworkManager` instance gets declared on the Application level. When the page loads we are shown our selected hunter and presented with two option:

- Host Battle
- Join Battle

Host Battle in this case the device becomes the host of course and initializes the server. Lets the system choose a port, registers it, stars "broadcasting" so the client can find it.
It waits for connections. And stars listening for incoming messages.

Join Battle in this case the device becomes the client and starts to look for possible connections using Android's NSD. Also starts listening for incoming messages.

If connection is made Both the client and the host send their respective hunters. Up on reciving a hunter `BattleActivity` is launched (The "in use" version not the OLD).
In case any problem arise error messages are shown

Battle Activity (in use)

Finally the actual Battle. `BattleActivity.java` handles both Online and Local Battle when the Intent is called an extra checker is passed `isMultiplayer` a simple if-else checks weather this is true.

In case of ***False*** – ***Local*** it is the exact same as `BattleActivityOLD` this is why I kept it so in this section i can focus only on Multiplayer mode.

In case of ***True*** – ***Online*** we get the already declared `NetworkManager` instance from App, rather than declaring a new one or somehow passing it true the Intent. In the OLD version who stars the battle depended on the id of the hunter, here that is problematic because the id's can be same. So the first turn is the Host.

`fight()`

The algorithm is again same as in the OLD version (calculate damage, update UI...), but now as an extra step an instance of `BattleAttack` is sent, containing all relevant information.

Up on reciving the `BattleAttack` the opponent "handlws it" by setting healt according to damage updating Hp bars and adding the necessary battle log.

In case of defeat `BattleResult` instance gets sent out. Up on reciving it the algoritm handles it. the same way as in local battle. Battle log shows the result, and `rewardWinner(...) or punishLoser(...)` (depending on weather player won or lost) gets called the same way as in Local Battle.

5 Layout files and UI

Layout .xml files can be found in layout folder. To kinds layout files:

- Activity layout `-->` they start with activity... usage is self explanatory
- Card's `-->` they end with card or cardview. They usage is also in the name.

Recycler View components were used and Card view items were presented in them.
As I mentioned the app is very minimalistic I tried uniform colors (mainly Gray). I'm not a very artistic person.

6 Videos

Please watch the intro and Demostration videos, the rest is code explanation in case something didn't make sense in GitHub, description or demonstration.

Watch them in the intended order.

6.1 Intro*

6.2 Run on Emulator*

Part one clean run on emulator Link Part One (10 min)
https://1drv.ms/v/c/3e8ee7900513c457/ETBQeumwX4pDjchp3FE87iQBZn290XkpthvIAhQaYRolfQ?e=f4eoyz
Part two played for a few minutes generated some Statistical data: Link Part Two (3 min)
https://1drv.ms/v/c/3e8ee7900513c457/EfuNamTWdIRAvEqJnRuw7k0Buthm7KHgBl168SVMYXVMAw?e=0KMjxK

6.3 Run on Physical Device Online Battle*

Link Physical Device Online (5 min)
https://1drv.ms/v/c/3e8ee7900513c457/EfYlHAF6diZCiwvDGvgnU9EB6uZGeLbwG56ouKLNzzXCDA?e=61cD0x

Code Explaining Videos about 20 min each ONLY watch if something DIDN'T make sense in GitHub, in Demostrational Videos, or in The description

6.4 Package: hunter

Link
https://1drv.ms/v/c/3e8ee7900513c457/ERMj4VLcyS1Cs5H_DhCLYqABZfGC_zH1J-LjsIHgwideEw?e=m4EYu7

6.5 Data Storage: Json files + JsonHelper

Link
https://1drv.ms/v/c/3e8ee7900513c457/EW22rFs9LtVJlecmGiSNQIwBdfAM6zjkbQ2I_ylP-eaHEg?e=Nh6JYd

6.6 Main Activity + Hire Hunter + Home

Link
https://1drv.ms/v/c/3e8ee7900513c457/ESRKwekWYQ5EqMJ-mXfM9rUBEnNXIo-E9yej8LaXEhRRLQ?e=IjumIq

6.7 Training + Statistics + Charts

Link
https://1drv.ms/v/c/3e8ee7900513c457/EbrNcS7B45F0nxcAvyKcF4UBSWQUoCPG3zWMnK2-AvSn1Q?e=A4mvJb

6.8 Battle Activity OLD

Link
https://1drv.ms/v/c/3e8ee7900513c457/EW_iDPR3yQJAk3d4x9x0U4IB4NyAtixZb7e3wlhEeaVJ1g?e=BEy88g

6.9 Network + Multi Player Setup

Link
https://1drv.ms/v/c/3e8ee7900513c457/EUk-0vUiB9NDrn4qfJixS-sBd0PpSGPMylxw6Cf0Wd5pLQ?e=P5RhK5

6.10 Battle Activity (in use)

Link
https://1drv.ms/v/c/3e8ee7900513c457/ERSKY8pwo5BHlnLZGj3VFIIBPks18qHmk0tCD6Tc-wluIQ?e=qWxC39