Documents(To export)

Meeting Log

Meeting #1 (May 31, 2024)

Meeting Date/Time/Agenda

- Date & Time: May 31, 2024 15:00
- Agenda: Implementation kick-off

Key Takeaway/Summary

- Front-end: Pierre, Jju
- Back-end: Eden, Lynn, Boonie
- Structural design

Meeting #2 (June 2, 2024)

Meeting Date/Time/Agenda

- Date & Time: June 2, 2024, 16:00
- Agenda: Implementation started

Key Takeaway/Summary

[Back-end]

- Server framework decided → Flask
- Protocol will be HTTP
- Implementation completed
    - Request for retrieving status window
    - Equipments swapping
    - Account creation

[Front-end]

- GitHub issue with .gitignore occurred

Meeting #3 (June 5, 2024)

Meeting Date/Time/Agenda

- Date & Time: June 5, 2024, 15:00
- Agenda: First communication between Server-Client

Key Takeaway/Summary

[Back-end]

- Code refactoring

[Front-end]

- GitHub issue solved

[Both]

- Server and Client communication completed
- Specify protocol between Server and Client

Meeting #4 (June 6, 2024)

Meeting Date/Time/Agenda

- Date & Time: June 6, 2024, 14:00
- Agenda: Unity CI/CD, Project close

> Key Takeaway/Summary

[Front-end]

- Using Unity DevOps tool for CI/CD
- Testing

[Front-end & Back-end]

- Project close agreement

Backlog

Project Backlog (1)

Aa Name	Assign	Status
Dungeon		Should do
Status		Must do
Login		Should do
Exchange		Could do
Store		Could do

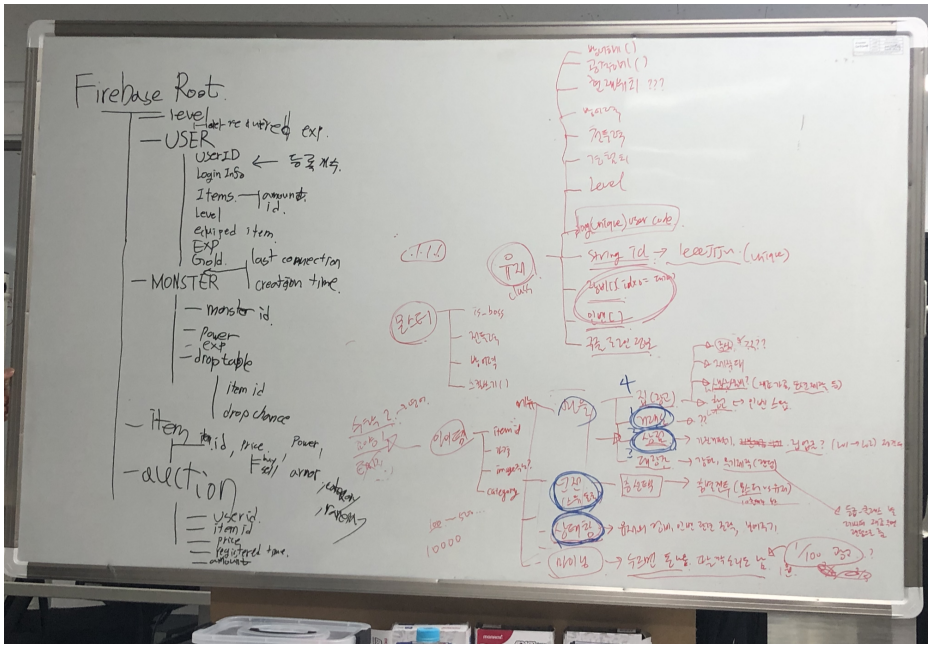
Sprint Board (1)

Aa Name	Assign	Status
[Contents] Story		Not started
[Contents] Music		Not started
[Back] Server Implementation		Done

Aa Name	Assign	Status
[Back-Front] Communication		Done
[Design] Icons		In progress

Brain Storming of Function

Structure



Status

- View inventory → Load item information from inventory from the Server
- View equipment → Load equipment information from the Server
- Change equipment → Send item ID from inventory and equipment slot ID to the server to request an exchange
- View stats → Fetch stats from the server
    - Must calculate with equipment values to display

Dungeon

- Floor(fighting area) selection
- Start combat → Load monster information & user stats
- Combat → Exchange information based on user action, exchange information based on monster action
- End combat → Request drop item (reward for winning the game) information from the server (proceed to the next combat start routine)

Login

- Google or KaKao API

Auction Market

- Open the item list → Get all item information registered in Server
- Register an item → Send the information of the item to Server
- Buy an item → Send the information of the item which Player wants to buy to Server
- Unregister a item which already be registered → Send the item information and get back the item
- Check the own item list which is registered

Store

- Open item list
- Buy items
- Sell items

Content Design

- Types of Monster
- Types of Weapon

- Types of Armor
- Types of Item & Resource & Equipment

Protocol

Network Protocol

- http

Communication Format

- Server to Client: JSON format
- Client to Server: URL

Model

- LEVEL
  - id
  - required_exp
- USER
  - UserID
  - LoginInfo
  - Registered amount
  - Items
    - amount
    - id
  - Level
  - equiped_item
  - EXP
  - Gold
- ITEM
  - item_id
  - price
    - buy_price
    - sell_price
  - power
  - armor
  - category
  - rarity
- AUCTION
  - user_id
  - item_id
  - price
  - registered_time

    - last_connection
    - creation_time
- MONSTER
    - monster_id
    - power
    - exp
    - drop_table
        - item_id
        - drop_chance
    - amount

Status

- Check inventory → Load item information stored in the inventory from Server

```
Route: status/inventory
Method: 'GET'
Client Send: # Add UserID in the GET frame (e.g. userID='123
Return:
{
    “item” = [
        {”amount”: 0, “id”:0},
        {”amount”: 0, “id”:0},
        {”amount”: 0, “id”:0}
    ],

    “equipped_item”: 0
}
```

- Swap equipment → Send (item ID stored in the inventory, slotted ID) to Server

```
Route: status/changeEquipment
Method: 'POST'
Client Send: # Add userID and equipID to HTTP frame
             # (e.g. userID='1234', equipID='1234')
Server Return:
{
    "status": 0
}
```

- Check stat → Load the stat from Server
  - Equipment figure should be calculated
  - Temporarily not in implementation

Login

- Create a account → Create a Object on Server

```
Route: login/create
Method: ‘GET’
Server Return: # Server generate the random number to identi
                              # and send it to Client
Return:
{
    “userID”: Random Number
}
```

Design Plan

Character Image

- Frontal exhalation, frontal inhalation
- Side exhale, side inhale
- Side view attack motion animation
- Side defense motion animation (optional)

Equipment Image

- Types of Sword
- Types of Armor

Background Image

- Village
- Exchange market
- Store
- Dungeon : when floor(fighting area) selection
- Dungeon : combat
- Status

Monster Image

- Boss
- Mop

BGM

- Village
- Dungeon
- Status