December 08, 2017

Design Studio 3 - A Doll's World

Team C8

79209224 Vales, Darlene Cesicar

79369208 Nunez, Alicia

11671379 Li, Zhaomin

30882088 Espinosa, Sarah

17251265 Sattar, Suha

Table of Contents

Introduction

Our team is focused on designing the plan for “A Doll’s World” which is a project intended on revamping the basic doll house into a virtual experience. The application is a desktop-based game intended to ultimately promote the company’s current doll collection. The dollhouse is intended to be collaborative so kids ages 4-7 can have fun playing with their friends. Based off the specifications provided, we have made design decisions to implement application, interaction, architectural, and implementation design. The logic behind our design decisions is detailed within our design.

Software Summary (Application)

Stakeholders

- Children
- Parents
- Toy Company
- Developers
- UI/UX Designers
- Tech Support

Goals

- Kids from multiple locations can play together on the application.
    - Can meet other dolls
    - Can visit other doll houses
    - Can add friends, have a list of friends.
- The game is intuitive -- simple and easy to use
- The game is fun and engaging for the kids to play.
    - Have levels that they can unlock, giving them incentives to play.
- The game can be easily maintained and accommodate changes for the future.

Assumptions

- Kids have decent knowledge and experience on how to use a computer.
- Computers will meet minimum requirements to run the game.
- Servers will be able to handle online traffic.
- This is a desktop game.
- The company makes gender-neutral dolls. User chooses the skin color, eyes, and hair.

- There is a default body type for all the dolls. We think kids would enjoy changing the outfit for the dolls more, rather than changing the height and body type. And it helps the company to make the dolls in the same size.

Constraints

- Target age range is kids 4-7 years old.
- Username duplicates are not allowed. Kids will be able to name their dolls separately.
- Some features will only be accessible with internet access.
- You must download the game online in order to play. There will be no web-version of the game.
- For security purposes, there will be limited online chat features.
- No in-app purchases.
- Kids can only have one dollhouse, chosen from predefined dollhouse templates. They will have the option to choose the placement of the rooms

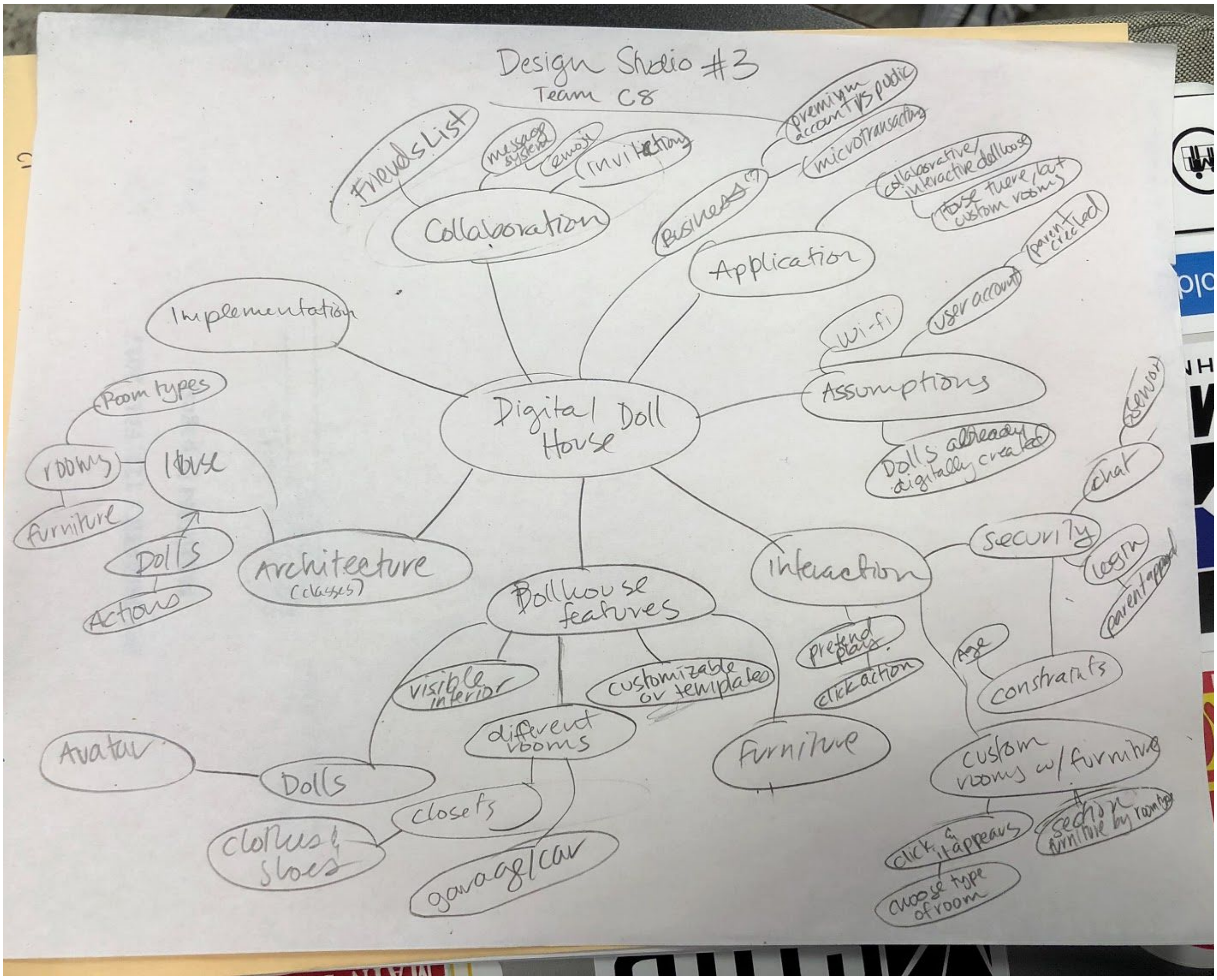
Decisions

- User must have internet connection when they first download and create their username.
    - An online database will be checked to make sure the username is unique.
- Have a set of established furniture
- Doll actions are determined by furniture. Different types of rooms have different types of furnitures and various interactions with the furnitures. Dolls cannot do anything in an empty room.
- Minimum of one room, one kitchen, one bathroom, and one garage for each house.
- The system sets the number of rooms in a house and the arrangement of the rooms. There are four empty rooms and the kids can choose the types of the rooms.This decision eliminates the mis-arrangement of the house (in shape or in functionality), and eases the kids' work.
- There is level system for each user.
    - The users have daily a task to complete, so they can level-up. It motivates the kids to play games on a daily basis.
    - By leveling up, users can unlock new furniture in the game.
- Limited chat services. Users cannot message each other.
    - Dolls have 'chat bubbles' with a preset list of sayings to say to one another

- Chat services can be considered unsafe, especially for age group we are designing for. By using chat bubbles instead of messaging, we prevent any type of inappropriate behavior on the game.

Application Design

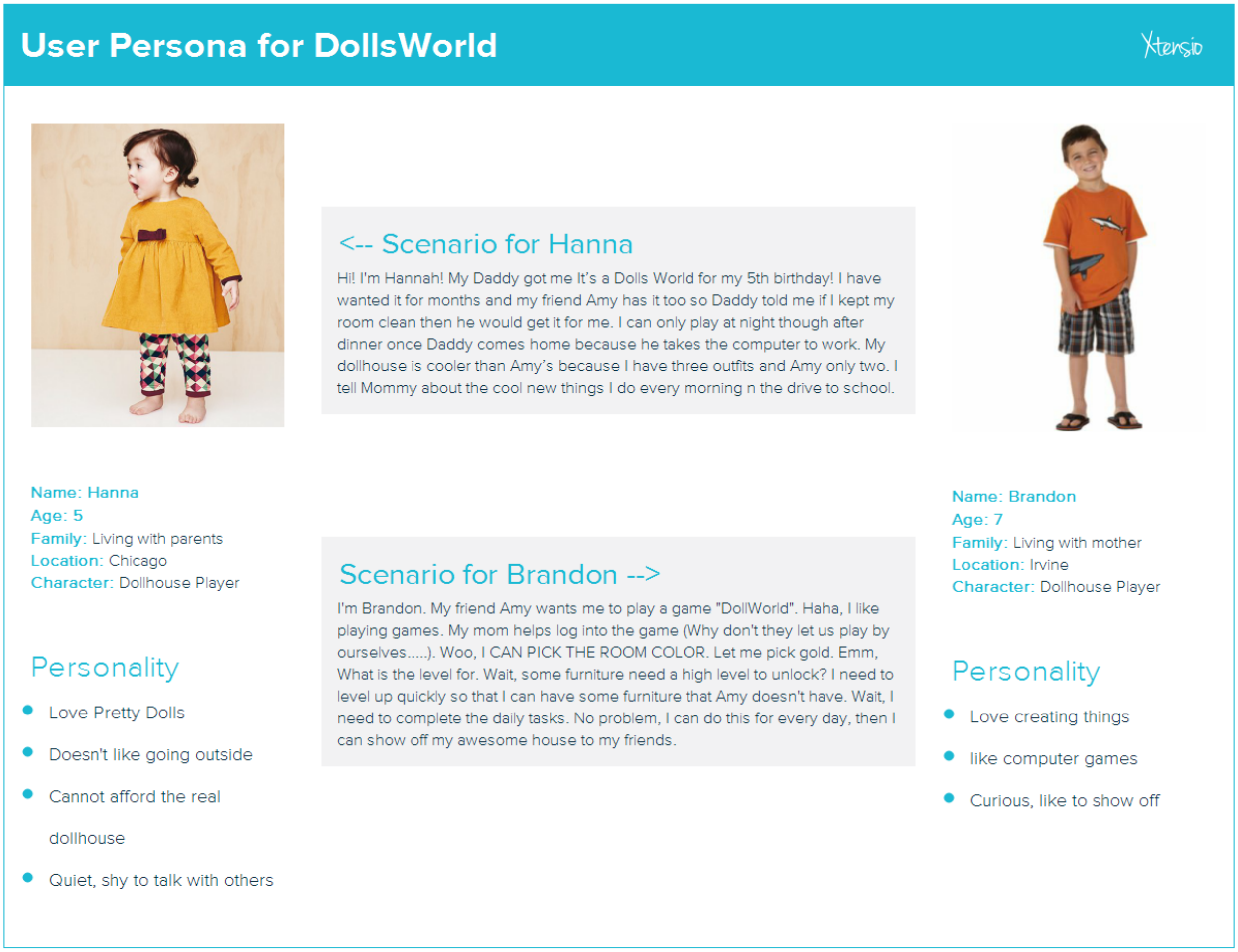
Mind mapping



We performed the mind mapping on the first day that we met up. In this practice, we start from the Doll House Software, which is the essence of our project. We wanted to analyze it from four aspect: application design, interaction design, architecture design, and implementation design. As we have more and more nodes in the map, we also concern some of other features such as the physical features of the dollhouse, the collaboration among users, and the company's business.

Interaction Design

Persona



User Persona for DollsWorld

Xtensio

<-- Scenario for Hanna

Hi! I'm Hannah! My Daddy got me It's a Dolls World for my 5th birthday! I have wanted it for months and my friend Amy has it too so Daddy told me if I kept my room clean then he would get it for me. I can only play at night though after dinner once Daddy comes home because he takes the computer to work. My dollhouse is cooler than Amy's because I have three outfits and Amy only two. I tell Mommy about the cool new things I do every morning n the drive to school.

Name: Hanna
Age: 5
Family: Living with parents
Location: Chicago
Character: Dollhouse Player

Personality

- Love Pretty Dolls
- Doesn't like going outside
- Cannot afford the real dollhouse
- Quiet, shy to talk with others

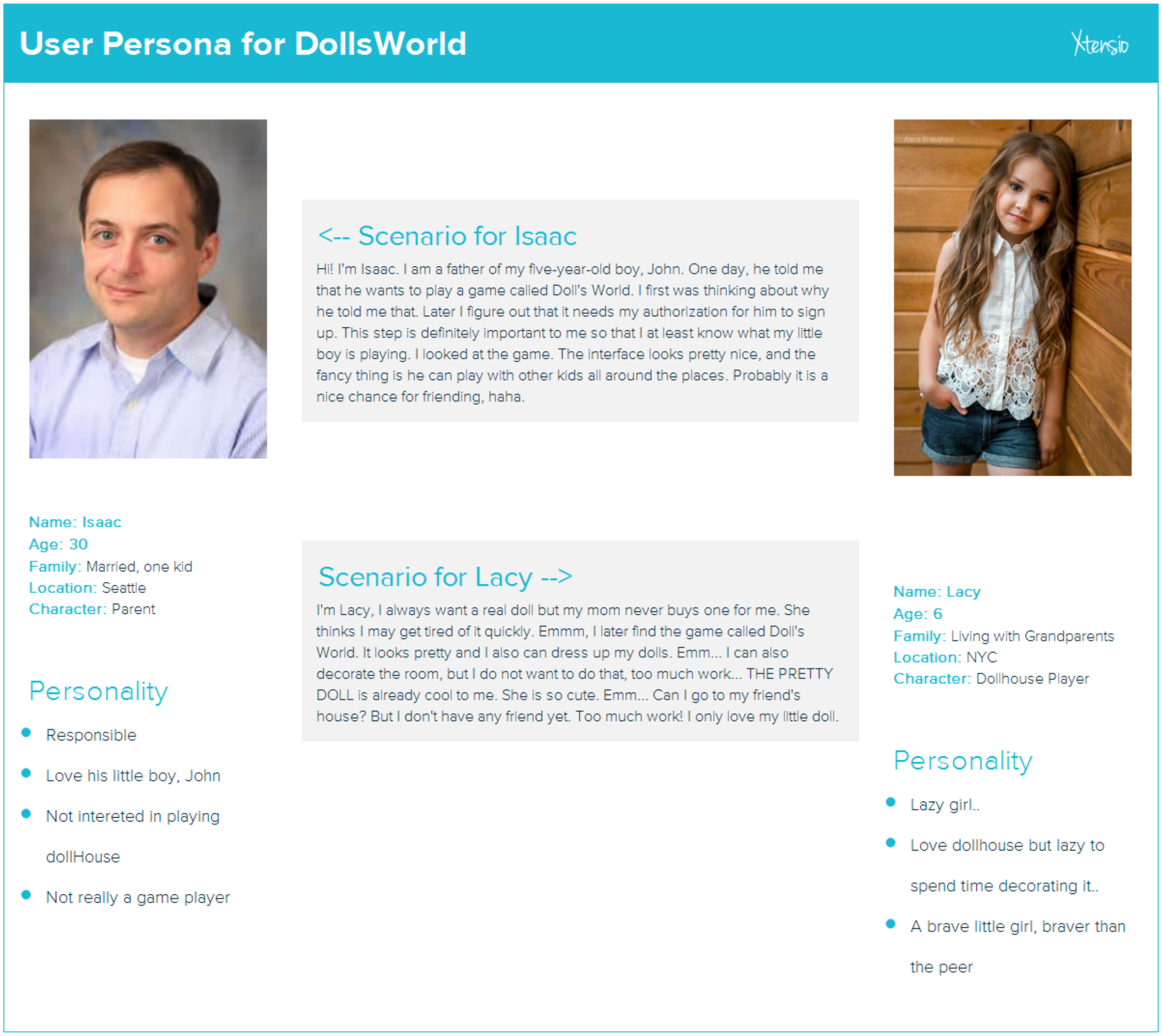
Scenario for Brandon -->

I'm Brandon. My friend Amy wants me to play a game "DollWorld". Haha, I like playing games. My mom helps log into the game (Why don't they let us play by ourselves.....). Woo, I CAN PICK THE ROOM COLOR. Let me pick gold. Emm, What is the level for. Wait, some furniture need a high level to unlock? I need to level up quickly so that I can have some furniture that Amy doesn't have. Wait, I need to complete the daily tasks. No problem, I can do this for every day, then I can show off my awesome house to my friends.

Name: Brandon
Age: 7
Family: Living with mother
Location: Irvine
Character: Dollhouse Player

Personality

- Love creating things
- like computer games
- Curious, like to show off



User Persona for DollsWorld

<-- Scenario for Isaac

Hi! I'm Isaac. I am a father of my five-year-old boy, John. One day, he told me that he wants to play a game called Doll's World. I first was thinking about why he told me that. Later I figure out that it needs my authorization for him to sign up. This step is definitely important to me so that I at least know what my little boy is playing. I looked at the game. The interface looks pretty nice, and the fancy thing is he can play with other kids all around the places. Probably it is a nice chance for friending, haha.

Name: Isaac
Age: 30
Family: Married, one kid
Location: Seattle
Character: Parent

Personality

- Responsible
- Love his little boy, John
- Not intereted in playing dollHouse
- Not really a game player

Scenario for Lacy -->

I'm Lacy, I always want a real doll but my mom never buys one for me. She thinks I may get tired of it quickly. Emmm, I later find the game called Doll's World. It looks pretty and I also can dress up my dolls. Emm... I can also decorate the room, but I do not want to do that, too much work... THE PRETTY DOLL is already cool to me. She is so cute. Emm... Can I go to my friend's house? But I don't have any friend yet. Too much work! I only love my little doll.

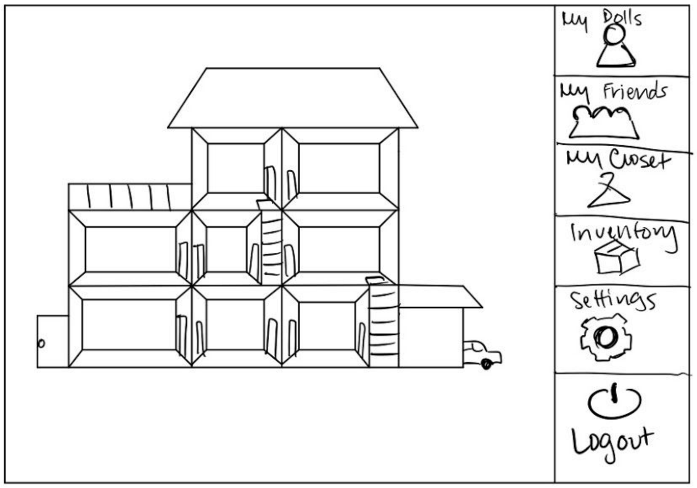
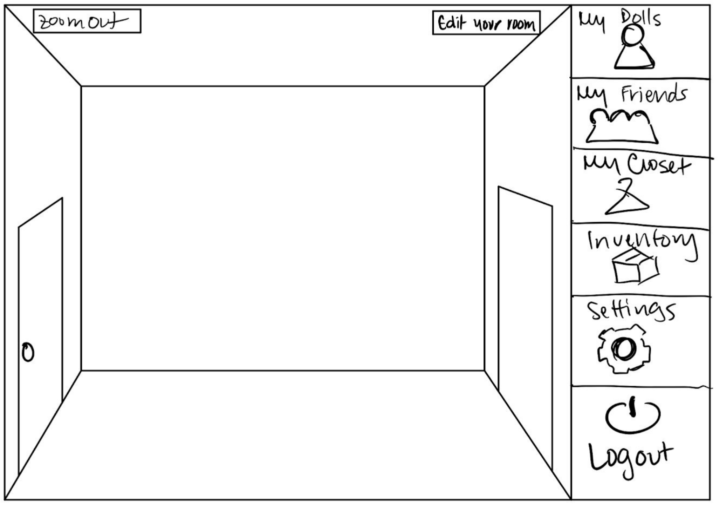
Name: Lacy
Age: 6
Family: Living with Grandparents
Location: NYC
Character: Dollhouse Player

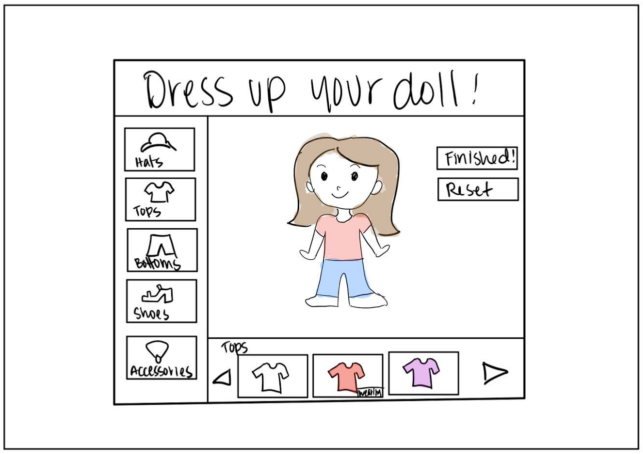
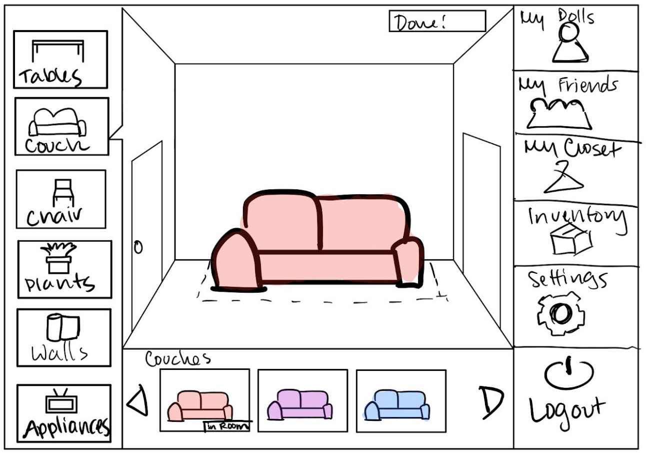
Personality

- Lazy girl..
- Love dollhouse but lazy to spend time decorating it..
- A brave little girl, braver than the peer

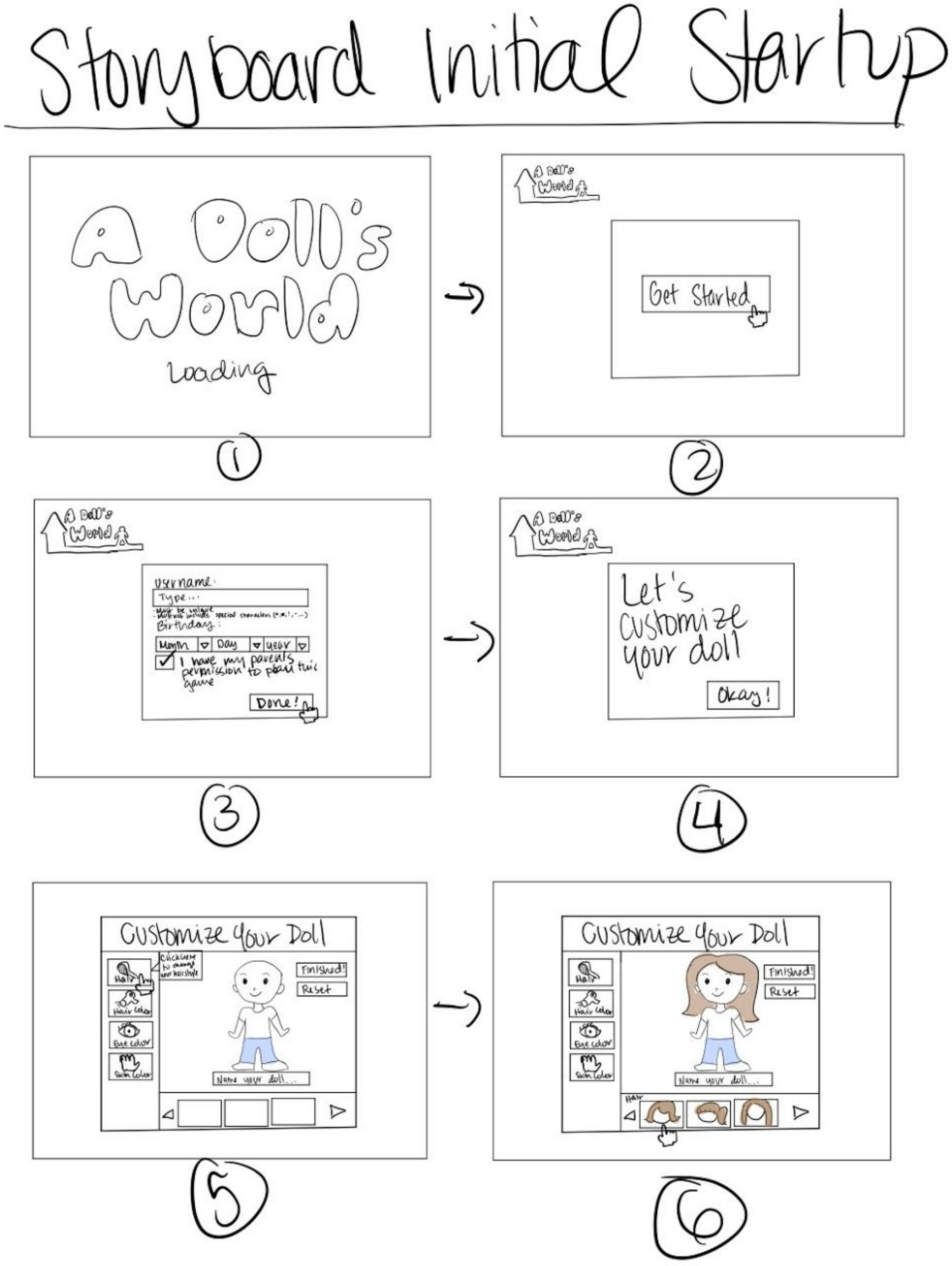
We have four personas, one for a 7-year-old boy, one for a 6-year-old girl, one for a 6-year-old girl, and one for a 5-year-old boy’s father. They have different characteristics. The persona can show how different types of users interact with our doll’s World game.

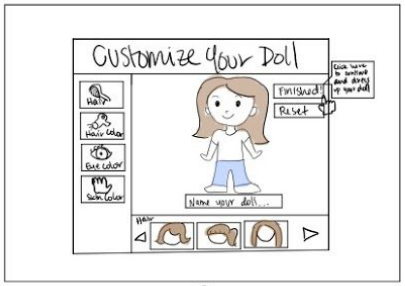
Mockup User Interface

 My Dolls My Friends My Closet Inventory Settings Logout	This is the interface for viewing the whole dollHouse. This is the first interface for users after they log in. It show the arrangement of the rooms and the right sidebar contains the information or functions the users want to access. Users are able to click the room to zoom in. The operation leads to the room interface
 zoom out Edit your room My Dolls My Friends My Closet Inventory Settings Logout	This is the interface for an individual room. This mockup in particular is empty, but typically, a room will have furniture in it that go in set places. Users are able to zoom out with the top left button to see the whole dollhouse. With the top right button, they can be transferred to an interface that lets them customize the room that they are in. When Furniture is inside a room, users can click on a piece of furniture and be presented with a list of actions that their dolls can do with it.

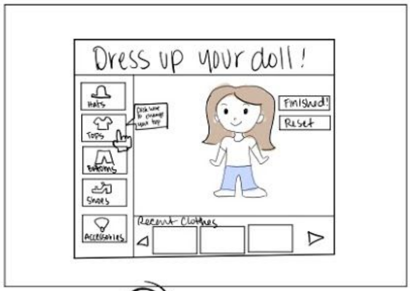
 Dress up your doll! Hats Tops Bottoms Shoes Accessories Finished! Reset Tops Wearing	This is the interface for the users to decorate their dolls. It can be accessed by clicking the “My Dolls” on the sidebar. In this interface, the users can change the outfits of the by selecting a type of clothing from the left side bar. From the bottom bar, they can then choose an item of clothing. When an item of clothing is clicked on, it will appear on the doll. A user can reset their doll to its default outfit with the “Reset” button or finish and save their doll with the “Finished” button. They will the. Be taken back to the dollhouse interface.
 Tables Couch Chair Plants Walls Appliances Done! Couches In Room My Dolls My Friends My Closet Inventory Settings Logout	This is the interface for the users to decorate their dolls. It can be accessed by clicking the “Edit Room” on the up right side of the room interface. In this interface, the users can drag or remove the unlocked furnitures in the certain room. There is a grid (coordinations) inside the room, which is invisible to the user. But the allocation of the furniture relates to the coordinates and not two furnitures can overlap with each other.

Storyboards

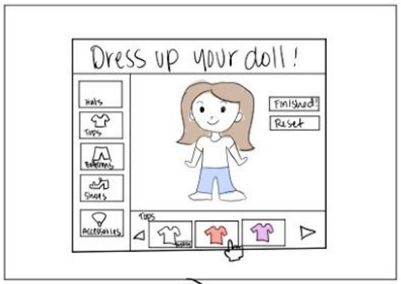



Customize Your Doll
Hair
Hair Color
Eye Color
Skin Color
Finished!
Reset
Click here to continue and dress up your doll
Name your doll...
Hair

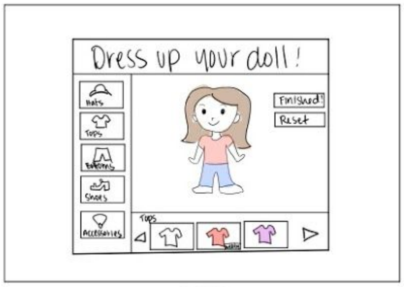
7


Dress up your doll!
Hats
Tops
Bottoms
Shoes
Accessories
Click here to change your top
Finished!
Reset
Recent Clothes

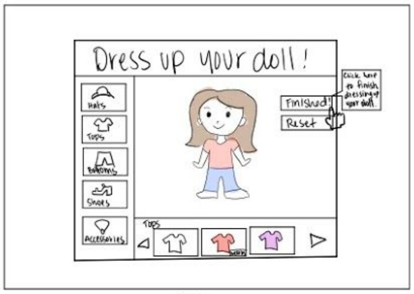
8


Dress up your doll!
Hats
Tops
Bottoms
Shoes
Accessories
Finished!
Reset
Tops

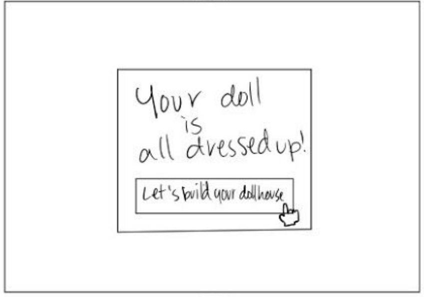
9


Dress up your doll!
Hats
Tops
Bottoms
Shoes
Accessories
Finished!
Reset
Tops

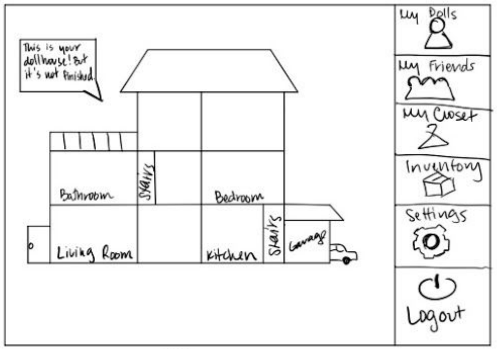
10


Dress up your doll!
Hats
Tops
Bottoms
Shoes
Accessories
Finished!
Reset
Click here to finish dressing up your doll.
Tops

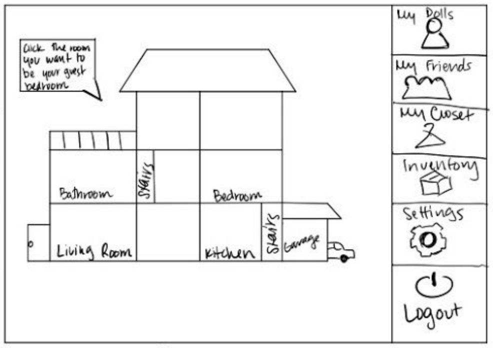
11


Your doll
is
all dressed up!
Let's build your dollhouse

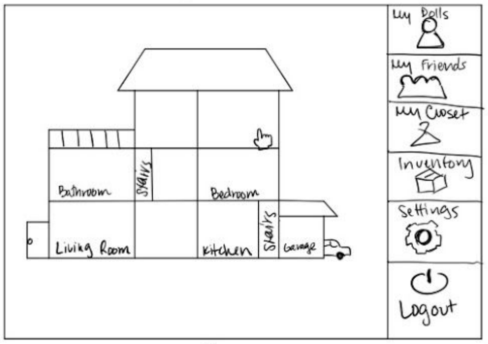
12



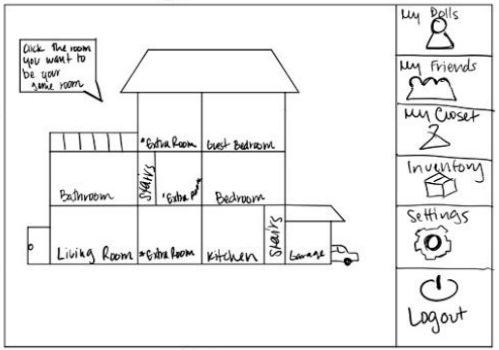
13



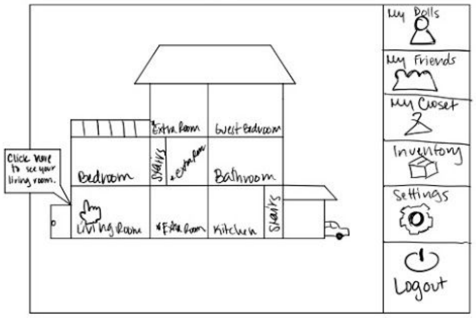
14



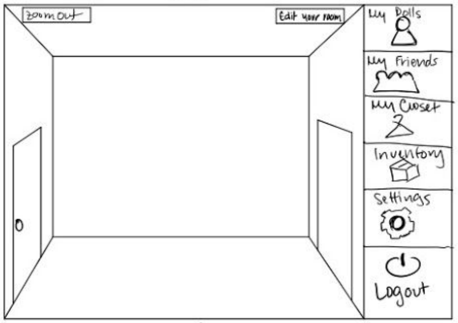
15



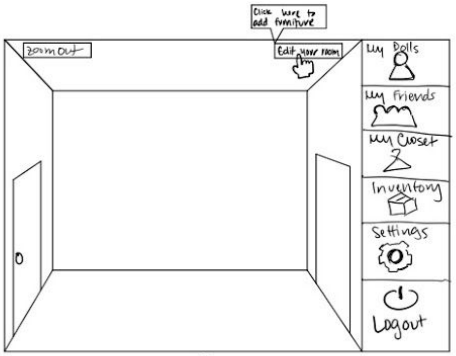
16



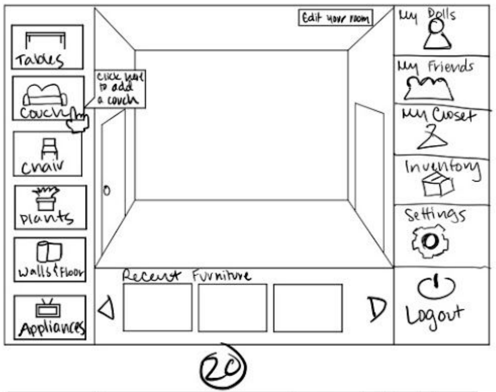
17



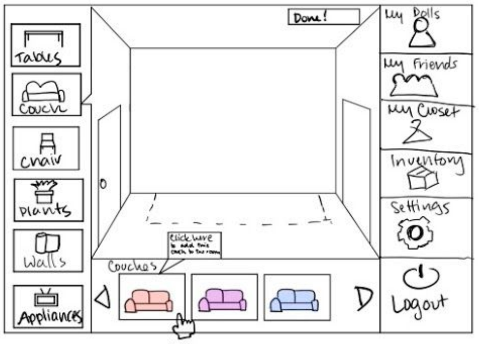
18



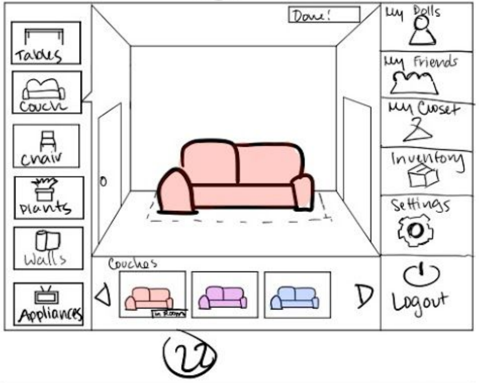
19



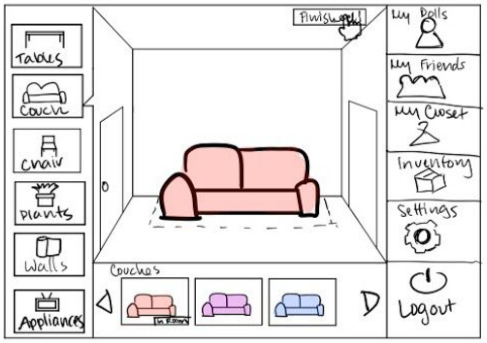
20



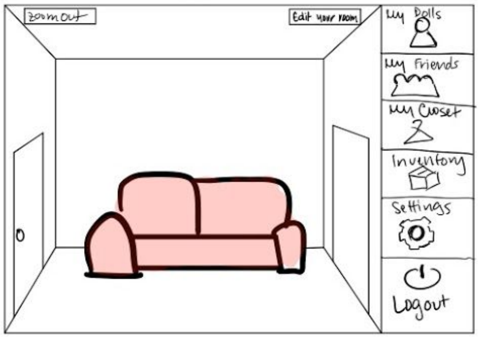
21



22



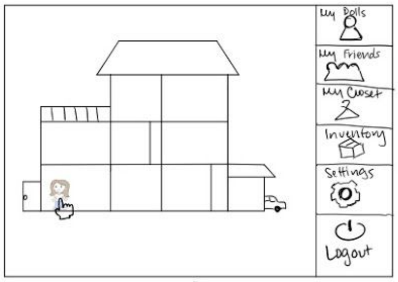
23



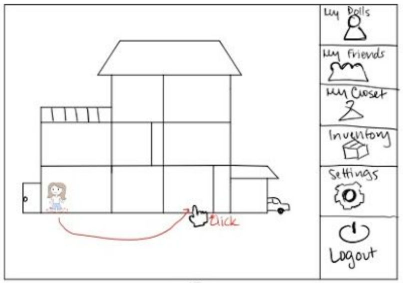
24

2 Storyboards For Moving a Doll

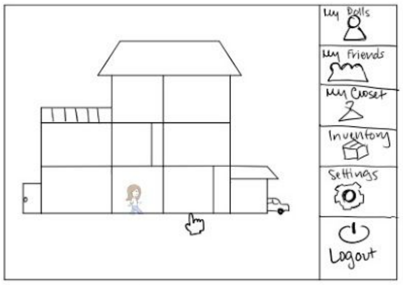
Moving a Doll when zoomed out



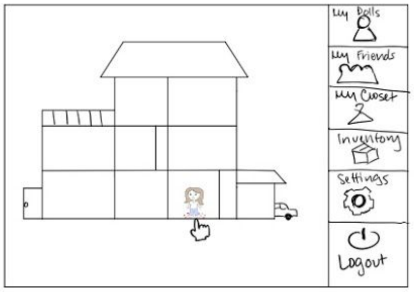
1



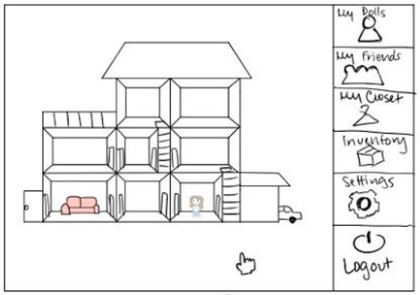
2



3

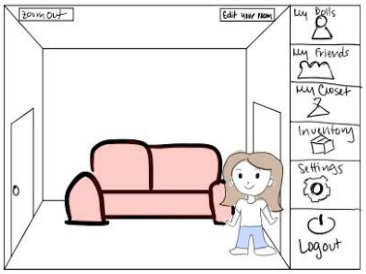


4

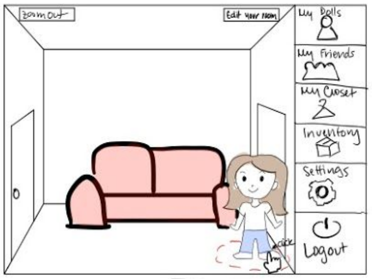


7

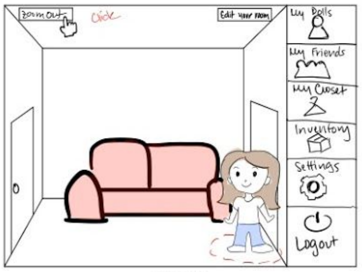
Moving a Doll when zoomedin



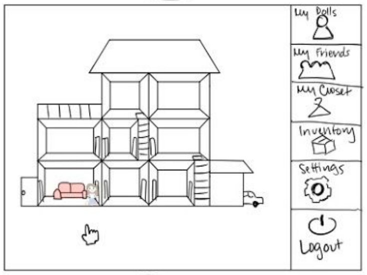
1



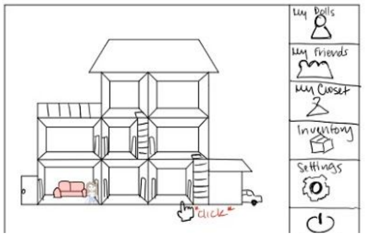
2

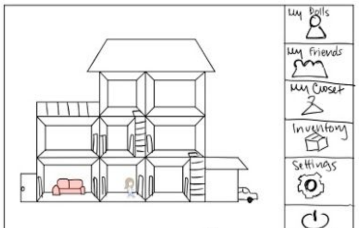


3

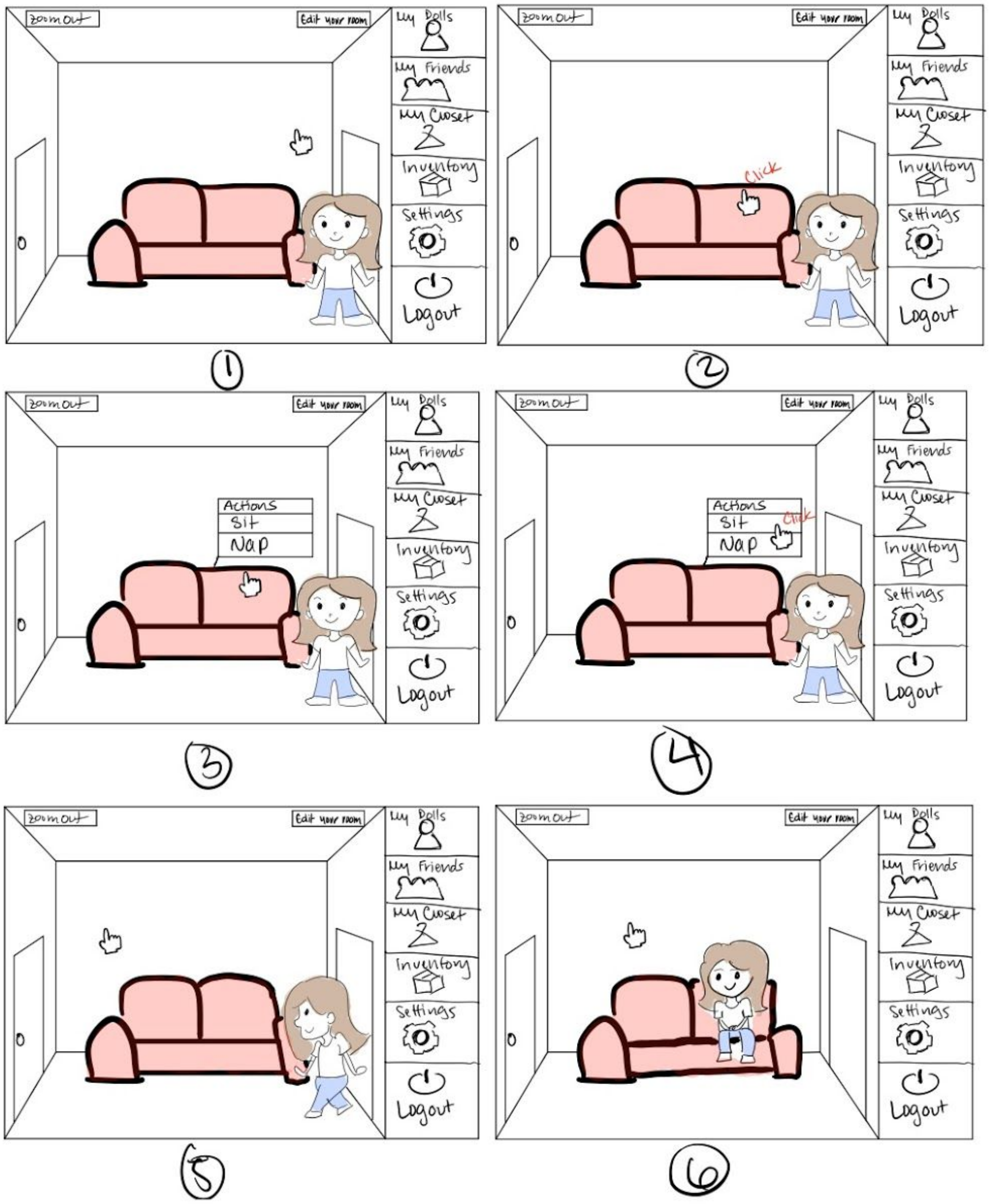


4



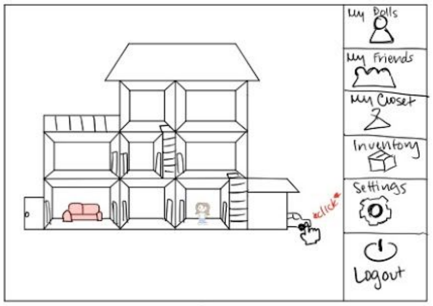


Storyboard of Interacting w/ Furniture

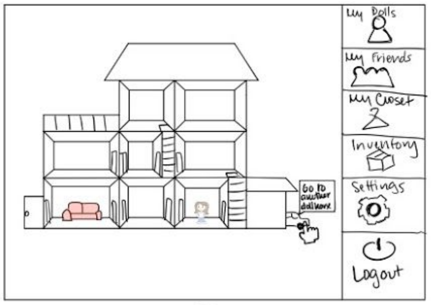


Storyboard For Going to Another Doll House

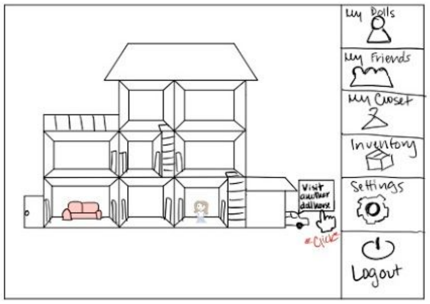
(from User1's perspective)



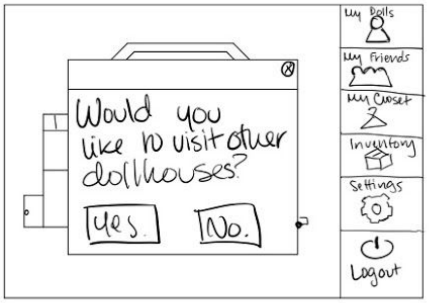
①



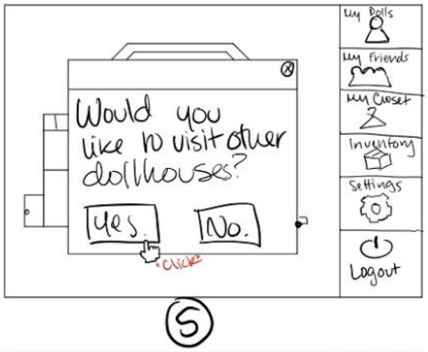
②



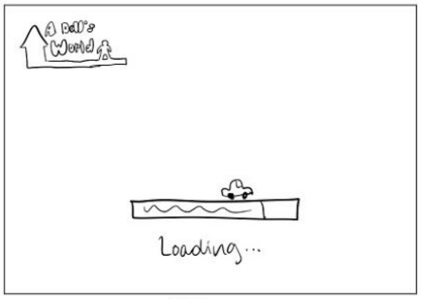
③



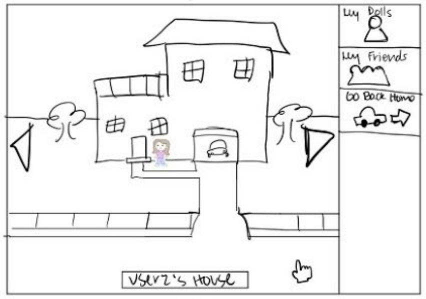
④


Would you like to visit other dollhouses?
Yes.
No.
Click
My Dolls
My Friends
My Closet
Inventory
Settings
Logout

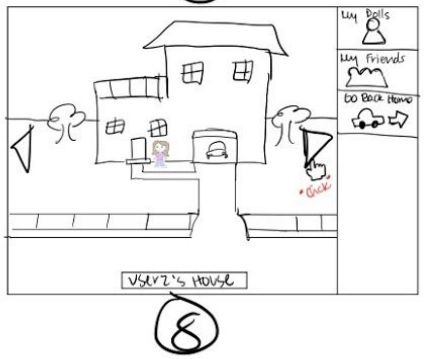
5


A Doll's World
Loading...

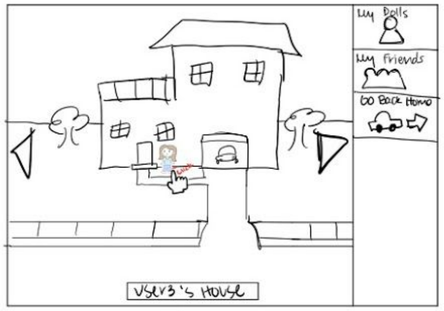
6


My Dolls
My Friends
Go Back Home
User2's House

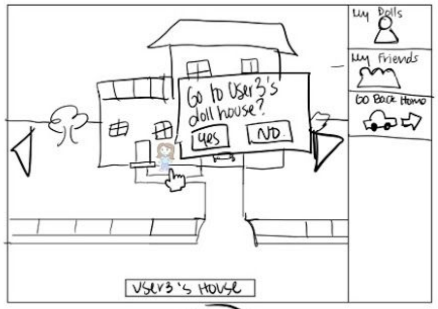
7


My Dolls
My Friends
Go Back Home
Click
User2's House

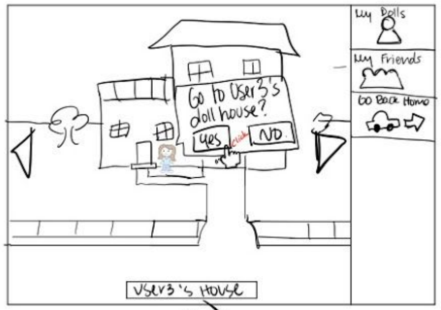
8


My Dolls
My Friends
Go Back Home
User3's House

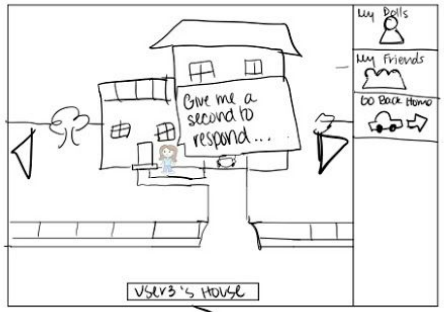
9


Go to User3's doll house?
Yes
No
My Dolls
My Friends
Go Back Home
User3's House

10

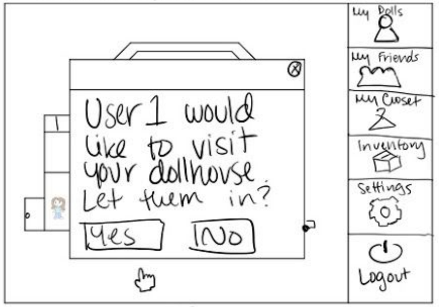


11

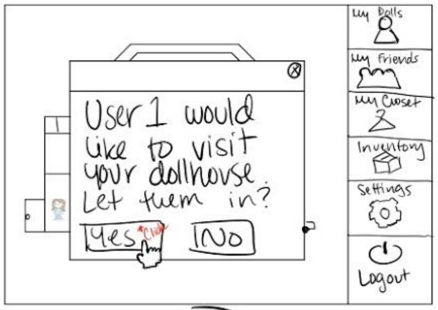


12

User 3's Perspective

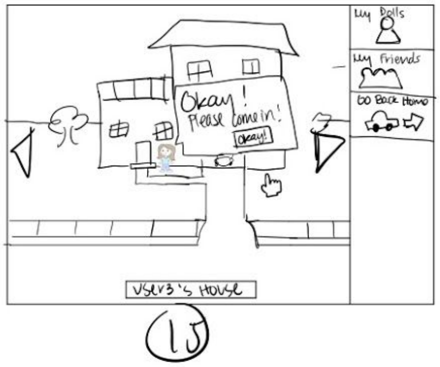


13

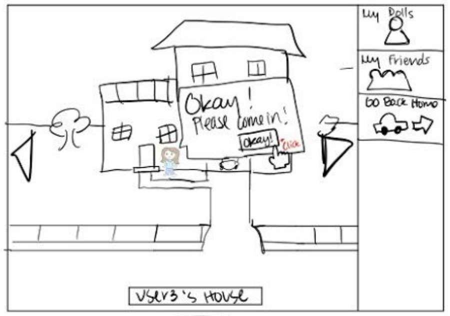


14

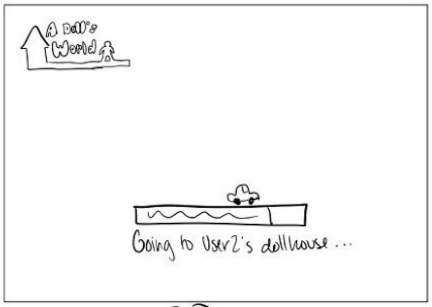
User 1's Perspective



15

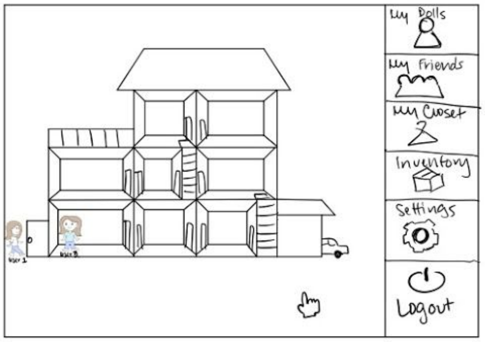


16

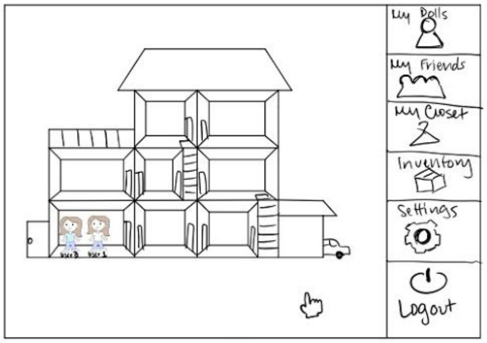


17

On Both User1 & User3's Screens



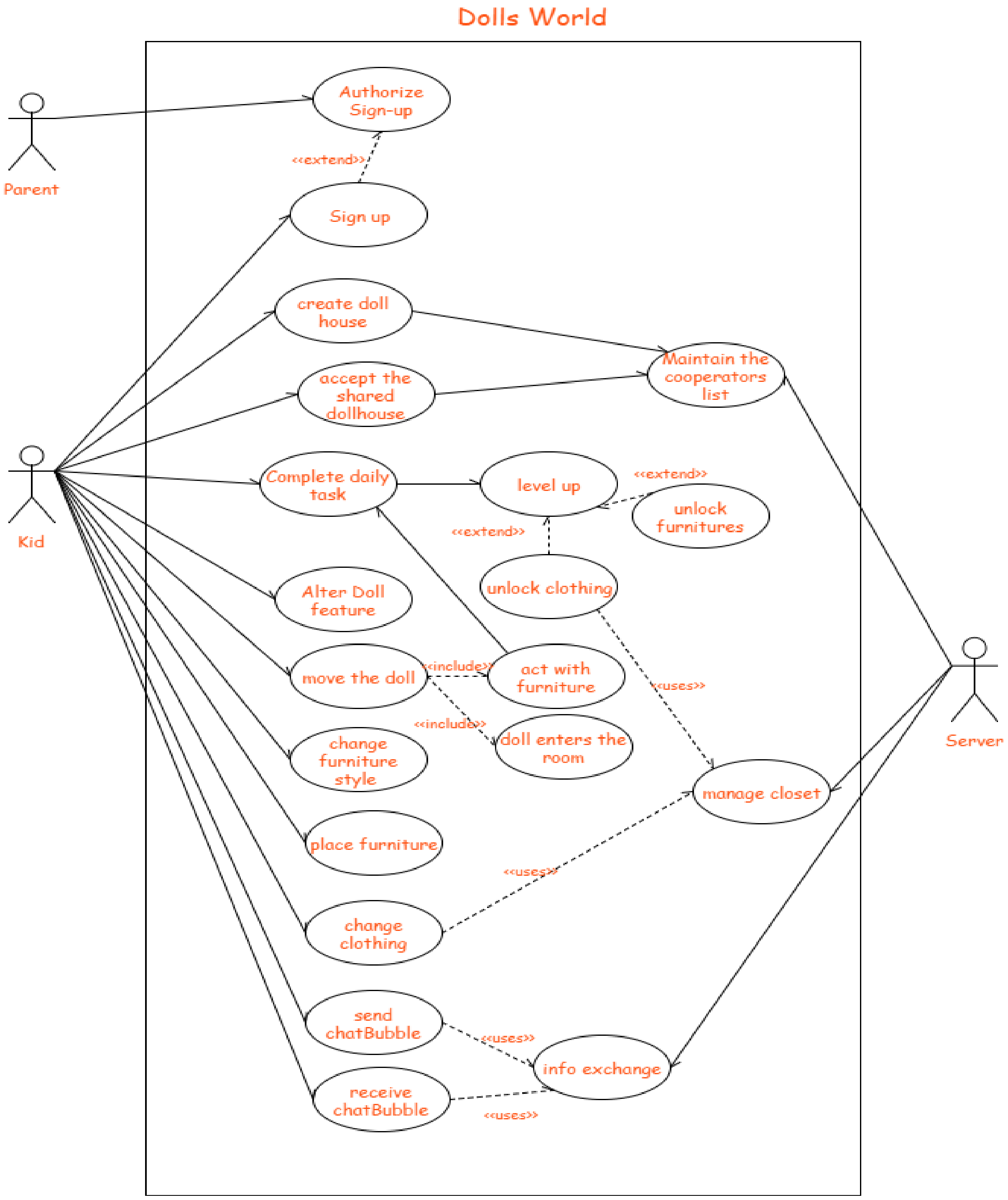
18



19

Architecture Design

User Case Diagram



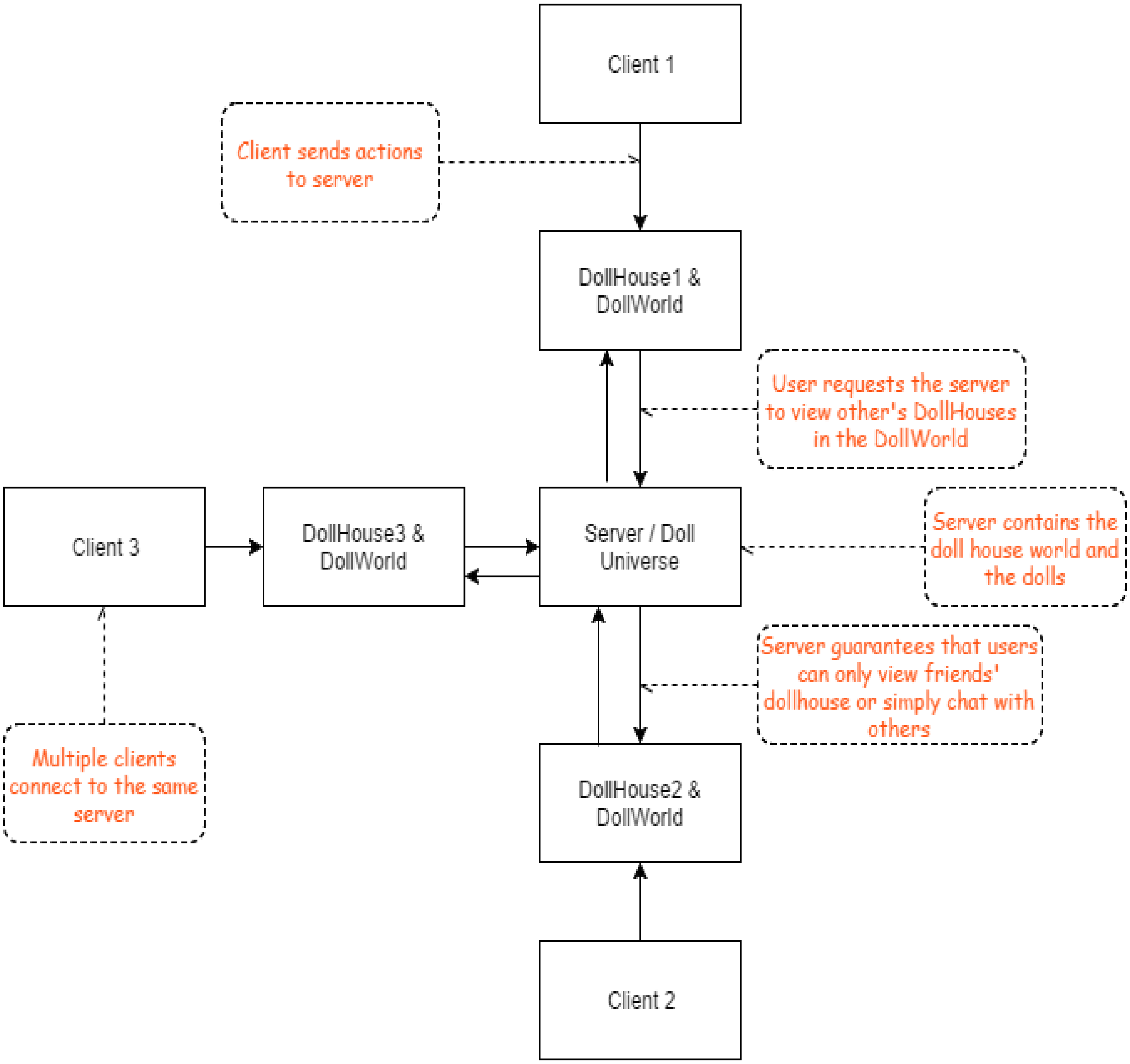
Actor Description:

Kid: The main players of the dollhouse. They can enjoy the most functionalities of the game. Various user cases relating to kids can help them gain the great gaming experience.

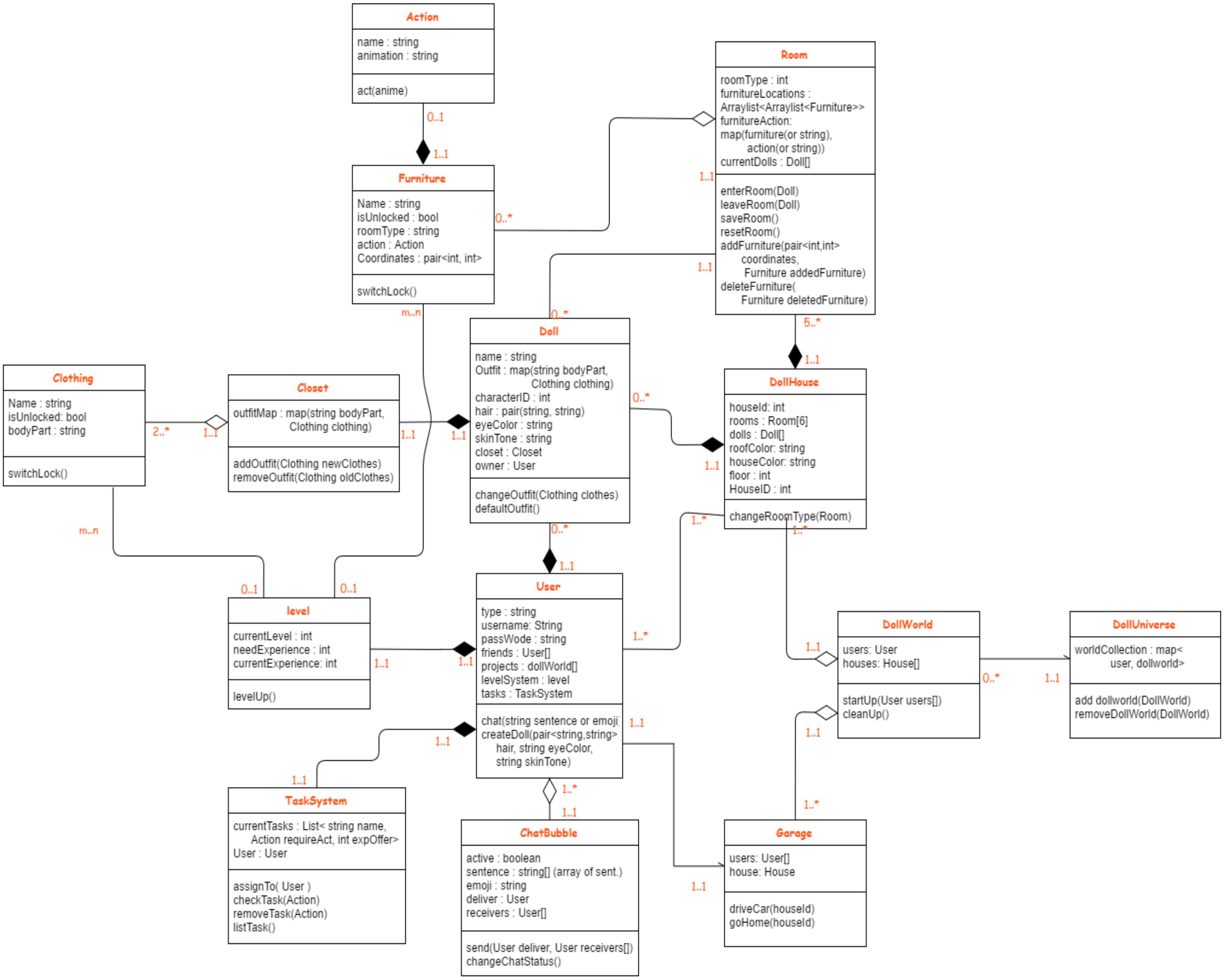
Parent: Their main job is to supervise the kids while the kids are playing the game, or to confirm while the kids start to play the game. When the kids sign up the accounts, the parents need to authorize the sign-up request.

Server: Server is able to handle the communication and collaboration between different players on the same dollhouse project. It supports the collaborative functionalities, guaranteeing the connection between users on the same project.

Architectural Sequential Diagram


Client 1
Client sends actions to server
DollHouse1 & DollWorld
User requests the server to view other's DollHouses in the DollWorld
Client 3
DollHouse3 & DollWorld
Server / Doll Universe
Server contains the doll house world and the dolls
Server guarantees that users can only view friends' dollhouse or simply chat with others
Multiple clients connect to the same server
DollHouse2 & DollWorld
Client 2

UML Diagram


Action
name : string
animation : string
act(anime)
0..1
1..1
Furniture
Name : string
isUnlocked : bool
roomType : string
action : Action
Coordinates : pair<int, int>
switchLock()
0..*
m..n
Room
roomType : int
furnitureLocations :
Arraylist<Arraylist<Furniture>>
furnitureAction:
map(furniture(or string),
action(or string))
currentDolls : Doll[]
enterRoom(Doll)
leaveRoom(Doll)
saveRoom()
resetRoom()
addFurniture(pair<int,int>
coordinates,
Furniture addedFurniture)
deleteFurniture(
Furniture deletedFurniture)
1..1
1..1
5..*
1..1
0..*
Doll
name : string
Outfit : map(string bodyPart,
Clothing clothing)
characterID : int
hair : pair(string, string)
eyeColor : string
skinTone : string
closet : Closet
owner : User
changeOutfit(Clothing clothes)
defaultOutfit()
0..*
DollHouse
houseId: int
rooms : Room[6]
dolls : Doll[]
roofColor: string
houseColor: string
floor : int
HouseID : int
changeRoomType(Room)
1..1
1..*
1..*
Clothing
Name : string
isUnlocked: bool
bodyPart : string
switchLock()
2..*
1..1
Closet
outfitMap : map(string bodyPart,
Clothing clothing)
addOutfit(Clothing newClothes)
removeOutfit(Clothing oldClothes)
1..1
1..1
m..n
0..1
0..1
level
currentLevel : int
needExperience : int
currentExperience: int
levelUp()
1..1
1..1
0..*
1..1
User
type : string
username: String
passWode : string
friends : User[]
projects : dollWorld[]
levelSystem : level
tasks : TaskSystem
chat(string sentence or emoji)
createDoll(pair<string,string>
hair, string eyeColor,
string skinTone)
1..*
1..1
1..1
1..1
TaskSystem
currentTasks : List< string name,
Action requireAct, int expOffer>
User : User
assignTo(User)
checkTask(Action)
removeTask(Action)
listTask()
1..*
1..1
ChatBubble
active : boolean
sentence : string[] (array of sent.)
emoji : string
deliver : User
receivers : User[]
send(User deliver, User receivers[])
changeChatStatus()
1..1
DollWorld
users: User
houses: House[]
startUp(User users[])
cleanUp()
0..*
1..1
1..1
1..*
1..1
DollUniverse
worldCollection : map<
user, dollworld>
add dollworld(DollWorld)
removeDollWorld(DollWorld)
Garage
users: User[]
house: House
driveCar(houseId)
goHome(houseId)

Implementation Design

Class Description

- **User :** The main players of the doll's world. Most of the users are kids. Each user can create the doll houses and control the decoration of the house. The user keep a list of friends so that the user can visit the friends' dollhouses. Each user can create doll(s) in the dollhouse. Each user has a level which can help him/her unlock the clothing and the furnitures. Each user has a task system. Completing the task can offer the user experience for leveling up.
    - Attributes:
        - Type : string in {'kid', 'guardian'} → indicating the condition of users, mostly kids, and parents can be the supervisors.
        - userName : string → the account name of the user, unique for one user
        - passWord : string → the passcode for logging in
        - friends : User[] → list of friends
        - projects : dollHouse → the dollhouse that is created and owned by the user
        - levelSystem : level → corresponding level for the user
        - Tasks : TaskSystem → record all the tasks for the user
    - Methods:
        - chat(string sentence or emoji) → start a chatBubble when they are visiting others' houses
        - createDoll(pair<string, string> hair, string eyeColor, string skinTone) → It creates Doll in the corresponding dollhouse. The doll's user attribute is setted by the user.
- **Level:** The level is assigned to each user. While playing the dollHouse, user can increase the level by completing the task and gain the experience. Some clothing and furnitures require a certain level to be unlocked. Whenever the user reaches the needed experience, the level increase by one and the needed experience for next level is increased by 50%.
    - Attribute:
        - currentLevel : Int → the current level of the user
        - needExperience : Int → the need experience to level up
        - currentExperience : Int → the current experience of the user

    - Methods:
        - Void levelUp(): the user levels up after reaching the needExperience. Everytime the needExperience is updated by multiplying 1.5
- **Dollhouse :** A collection of rooms. User can create a dollhouse and modify it. The dollhouse relates to the dollWorld. For the collaborative functionality, the user can visit other players' dollhouses by accessing the friend dollhouses list in the DollWorld. DollHouse has the garage, which is for visiting other's houses.
    - Attributes:
        - rooms : Room[6] → list of rooms in the house. The house would have 6 rooms in total (4 empty, 1 bathroom, 1 kitchen).
        - dolls : Doll[] → list of dolls in the house
        - garage: Garage → a garage is how a user will connect to other users. In dollhouse, it has a garage for the interfacing purpose. And user connects to one garage so the user can send request to enter other people's houses.
        - roofColor : string → the roof color, map to the global color reference
        - houseColor : string → the house wall color, map to the global color reference
        - floor : int → record the number of floors for a house
        - HouseID : int → a unique ID for the dollHouse so that user can request to go to the house with the specific ID.
    - Methods:
        - changeRoomType(Room) : written in the decisions, the house can have four empty rooms and user can assign the type to the empty rooms
- **Doll :** User can create doll(s) in the dollhouse. Each doll can interact with the furnitures and move in/between rooms following the user's orders. A doll has a closet containing all the clothing. And doll has different body parts for different clothing.
    - Attributes:
        - Name : string → user can make a nickname for the doll
        - Outfit : map(string bodyPart, Clothing clothing) → each doll is supposed to have a list of clothing mapping to each part of body. The clothing are also in the Closet.
        - Closet : Closet → all the clothing owned by the doll
        - characterID : Int → a number (or set of numbers) that represents a doll's ID

        - Hair : pair(string, string) → the first string is color (red, yellow...), and the second string is style (short, curly...)
        - eyeColor : string → the eye color of the doll, map to the global color reference
        - skinTone : string → the skin tone of the doll, map to the global color reference
        - Owner : User → the user who owns this doll
    - Methods:
        - changeOutfit(Clothing clothes) : search the clothing in the closet, and replace the clothing for corresponding body parts.
        - defaultOutfit() : generate the default top and the default pants, may be white vest and shorts.
- **ChatBubble :** Each person has a chat bubble, used to show when user want to talk with friends. It is normally used when the user is in the friends' houses, they can talk with friends and send emojis. That functionality follows the idea of collaboration. And, considering, the age of the players, the chatBubble can only send predefined text or emojis, but not a long sentence. When a user is in other's house by accessing the dollWorld, the user can use the chat bubbles by sending the server a request. The server would deliver the message the friends.
    - Attribute:
        - Active : Bool→ kid can activates or inactivate the chat function
        - Sentence : string → simple greeting. Long sentences are not allowed
        - Emoji : picture → or string which records the source of picture
        - deliver : User → the user who delivers the message
receiver : User[] → all the co-workers who receive the message
    - Methods
        - send(User deliver, User receiver[]) : user send a request to server to let the server pass the information to the receiver(s).
        - changeChatStatus: active = !active → chatBubble is not allowed when the status is inactive.
- **Room :** Room is the place to allocate all the furnitures and to make interactions between the dolls and the furnitures. Room remains a map the between the furniture and the corresponding actions. The types of furniture can differ, based on the type of the room (bath, bed..). To locate the furniture, the room has a coordinating grid in it (on floor/on wall). Furniture can be place on a empty grid or replace the former furnitures.
    - Attributes:

        - roomType : int → an ID representing the type of room.
        - furnitureLocations : Arraylist<Arraylist<Furniture>> → a 2-D array representing the coordinates in a room, each furniture is located on one specific coordinates and if two furnitures' coordinates are overlapped, the old one would be replaced
        - furnitureAction : map(furniture(or string), action(or string)) → when a user clicks a furniture in a specific room, the system finds the corresponding action to the specific furniture, and then plays the animations.
        - currentDolls : Doll[] → list of the dolls that are currently in the room
    - Methods:
        - enterRoom(Doll) - Doll enters the room. The doll is added into the list of currentDoll.
        - leaveRoom(Doll) - Doll leaves the room. The doll is removed from the list of currentDoll.
        - saveRoom() - saves the state of the room, storing the furnitureLocation in the memory.
        - resetRoom() - Resets the room so it is empty, cleaning up the furnitureLocation.
        - addFurniture(pair<int,int> coordinates, Furniture addedFurniture) - Adds furniture into the room based on the coordinators. The furniture is add on to the certain location in the room and replace the former furnitures if they are overlapped.
        - deleteFurniture(Furniture deletedFurniture) - Deletes the furniture, remove from the furnitureLocation
        -

- **Closet :** Collection of clothing that a user has unlocked and belongs to a single doll.
    - Attributes:
        - outfitMap : map(string bodyPart, Clothing clothing) → map the body part with the clothing
    - Methods:
        - addOutfit(Clothing newClothes) : add clothing into the map
        - removeOutfit(Clothing oldClothes) : remove the clothing key from the map.
        -
- **Clothing :** The clothing which can be worn by the dolls. Each clothing has a type for a certain body part. There is a lock for the level system.
    - Attributes:

        - Name : string → the name the clothing
        - isUnlocked: bool → determines if the clothing is locked. It would unlock if the user reach a certain level.
        - bodyPart : string (including top, pant, head, hand, foot, bag) → it show the which bodyPart the clothing belongs to. When picturing the doll, each clothing should go to the right body part.
    - Methods:
        - switchLock(): unlocks or lock the clothing item, isUnlocked != isUnlocked
- **Furniture :** Furniture has a roomType, indicating in which room the furniture can be placed. There is a lock imposed by the level system. The coordinators is used to match the grid in each room. Each furniture has a specific interaction and relates to the task system.
    - Attributes:
        - Name : string → the name of the clothing
        - Coordinates : pair<int, int> → pair of integers representing the coordinates of where the furniture is located in the room.
        - roomType: string → can be {livingroom, kitchen, bathroom, diningroom, bedroom, garage, garden…...}, the room type shows in which room the furniture should be placed
        - isUnlocked : Bool → determines if the furniture is locked. It would unlock if the user reach a certain level.
        - action : Action[] → the corresponding actions to the furniture. User use the certain Action class to interact with the furniture.
    - Methods:
        - switchLock(): unlocks or lock the clothing item, isUnlocked != isUnlocked
- **dollWorld :** The class maintains the list of the dollhouses that the user can visit. It contains at least one house which is the user's dollHouse. When garage sends a request, the software search the friends' list of dollHouses to find the destination so that the user can go to other's dollhouse.
    - Attributes:
        - users : User → player who owns the doll house. The dollWorld object can relate this user to all his/her friends' houses.
        - Houses : house[] → the houses that the user can visit, must check the friend list stored inside the User class.
    - Methods:
        - startUp(User users[]): set the default house and the initial users.
        - cleanUp(): clean up the whole house in Project object.

- **Action** : User can perform certain action on certain furniture. Action can be taking shower in the tub, sitting on the sofa, or watching the television. The action is presented as a shot anime for the doll-furniture interaction. Anime is stored in the system and can be imported (matched up) by using a string-type address.
    - Attribute:
        - Name : string → the name of the action
        - Animation : string → the string indicating the address of the action animation stored in the system back-storage
    - Method:

      act(string animation) : perform the anime, showing the interaction between the dolls and the furnitures.
- **TaskSystem :** The task system that the user can play on daily basis. The is a task list, and each task is a node. After user completing the action, the task tried to match if it's reaction may be expected.
    - Attribute:
        - currentTasks : List< string name, Action requireAct, int expOffer> → the task name and the related Action and the experience offered to the users
        - user : User → certain user complete the task.
    - Method:

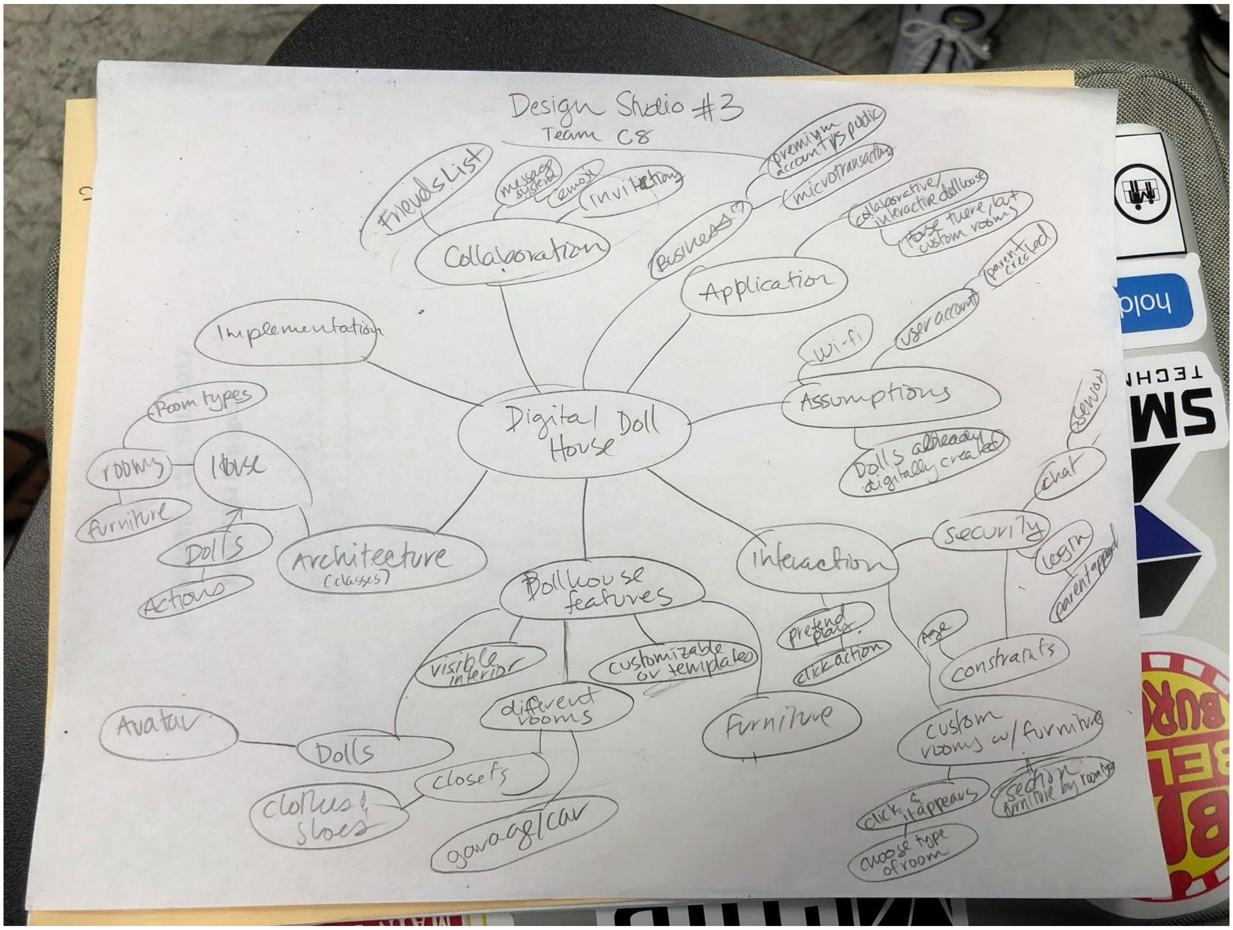
      Void AssignTo(User) : assign the experience to the users. It would change the currentExperience in the levelSystem to the corresponding user, and may trigger the levelUp method.
        - listTask() : presenting all the tasks to the user
        - checkTask(Action) : if a certain action is completed, check if the action is a task. Perform every time when a new action is completed.
        - removeTask(Action) : If a user have done a action which is a task, remove that certain task from the currentTasks list.
- **DollUniverse** : The class of collection. Collecting the map containing all the users and corresponding dollWorld. The map is continuously updated as the number of the user (#of the dollhouse) increases.
    - Attribute:
        - worldCollection : map< user, dollworld> → a map that relating the users with the doolwords
    - Methods:
        - add dollworld(DollWorld) : adding the New dollhouse into the doll universe.
        - removeDollWorld(DollWorld)

- **Garage:** The class that enables users to visit their friend's dollHouse. The driveCar method actually sends the users to the friend's house they request. The request will go to this user's dollWorld and find the requested friend's house in the house list. In another word, Garage class is like a portal for the users to access others' houses (we animate the visit-friend-dollHouse process as the user drives a car to the friend's house). This class is important in the collaborative design.
    - Attribute:
        - users: User[] → list of friends
        - house: House → current dollHouse that the user is in, showing the dollhouse that the garage is attached to
        - D
    - Methods:
        - driveCar(houseId): brings the user to their friend's dollHouse
        - goHome(houseId): brings the user back to their own dollHouse

Design Log

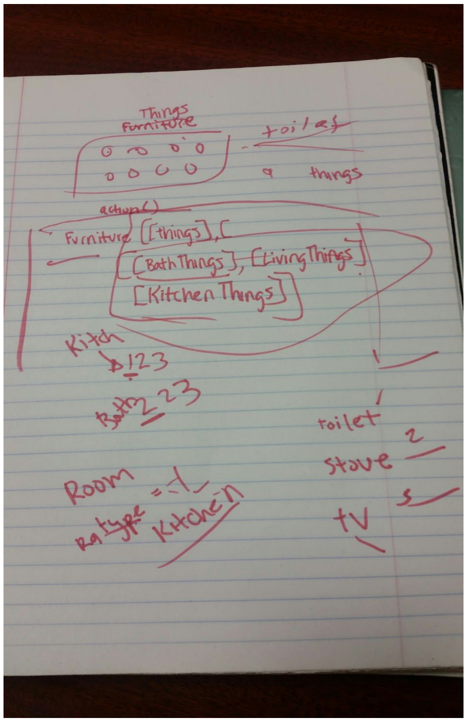
November 17, 2017

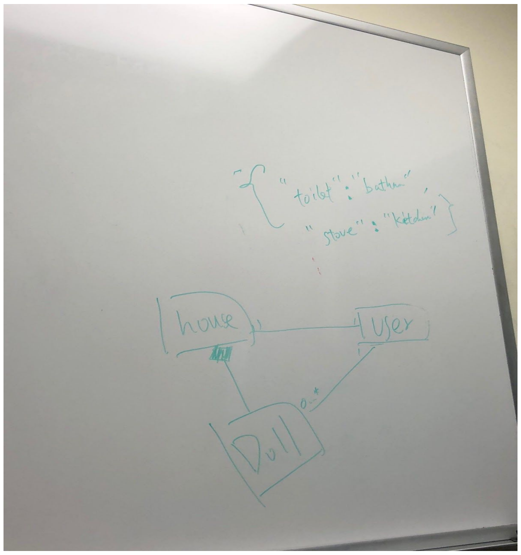
- The First Meeting
- Group is assembled; everyone is present
- We created a mind map (pictured below), containing all of our ideas for the dollhouse, planning out possible classes and looking at different requirements necessary for the dollhouse.
- Created a brainstorming document containing ideas that we have.
- Created a notes document, containing requirements specified from the assignment, which we will later add on to for any more requirements specified through email or in class.



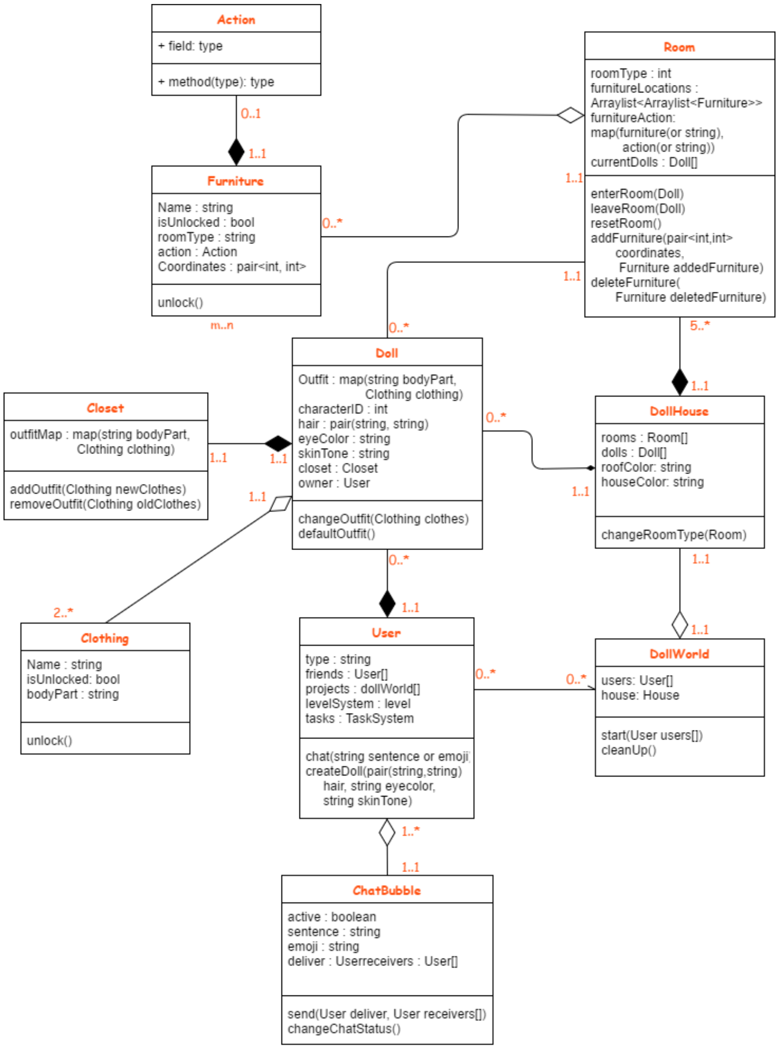
November 28, 2017

- First meeting outside of class
- Figured out most of the class in the UML design
- Figure out some interesting features including the level system
- Add goals, decisions, ideas, assumptions



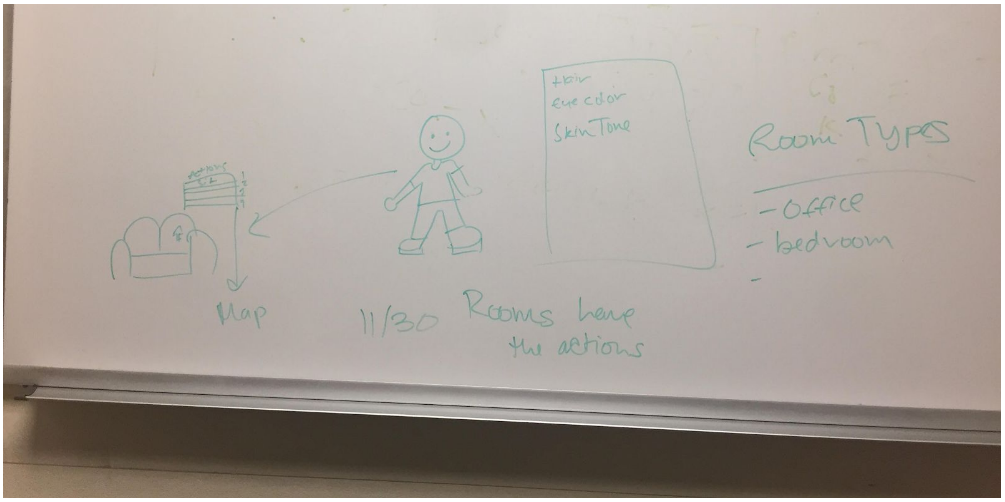


First UML draft - it doesn't have the task system and the level system embedded

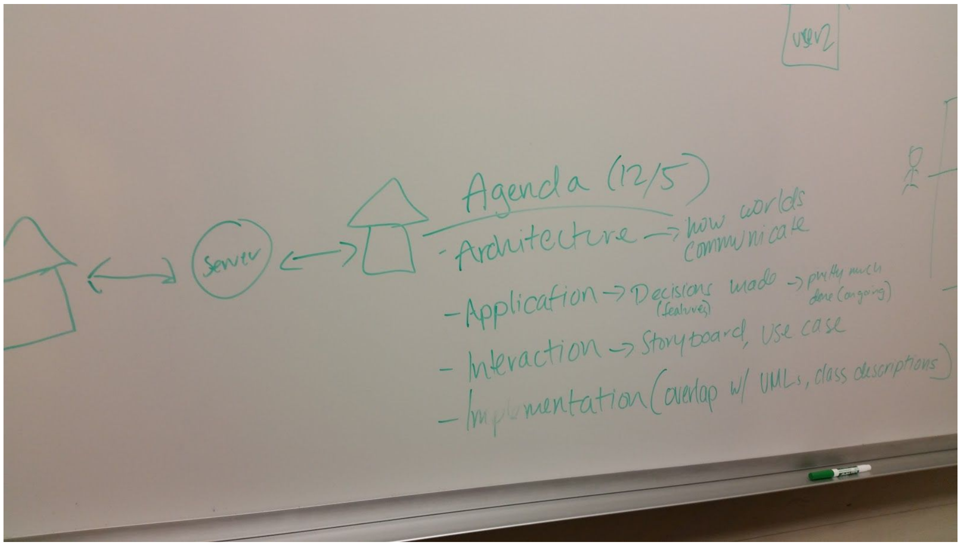


November 30, 2017

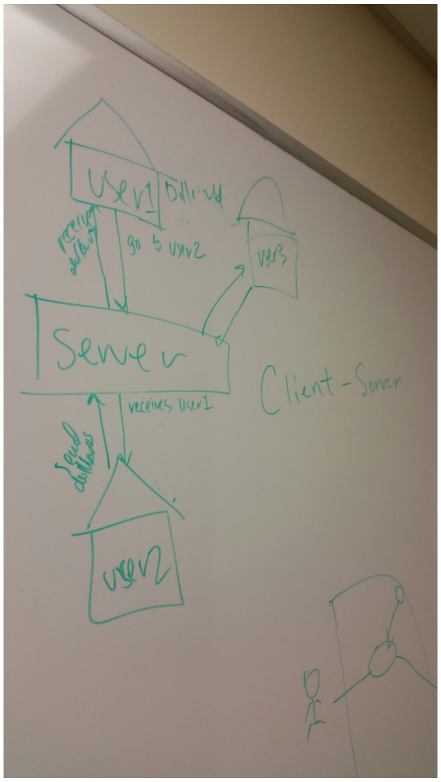
- Second meeting outside of class
- Add goals, decisions, ideas, assumptions
- Review the UI designs
- Design the specific methods of each class
- Design the user interaction with the dollhouse and how to implement them in system



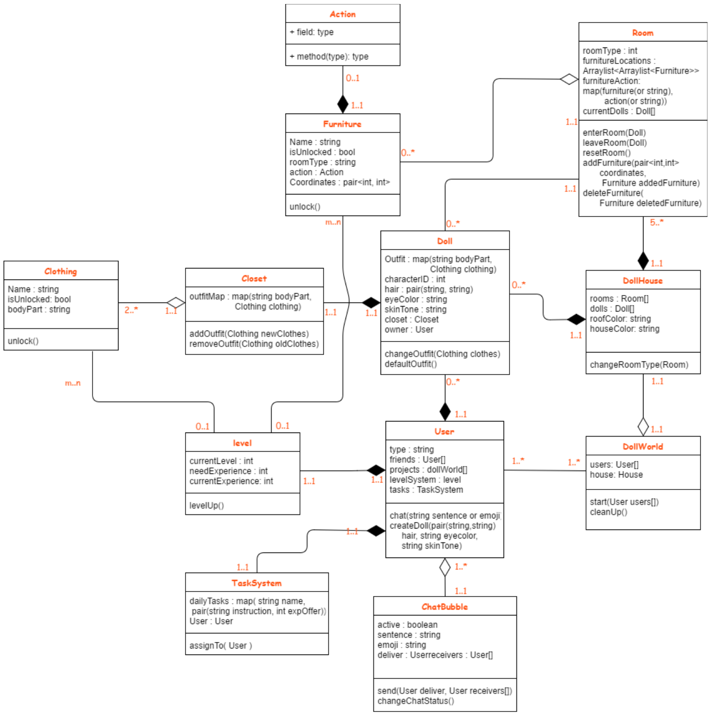
December 5, 2017

- Today’s meeting
- We decide all the contents that we will present in the final Design Studio 3
- Do the persona
- Deep into the class description
- 

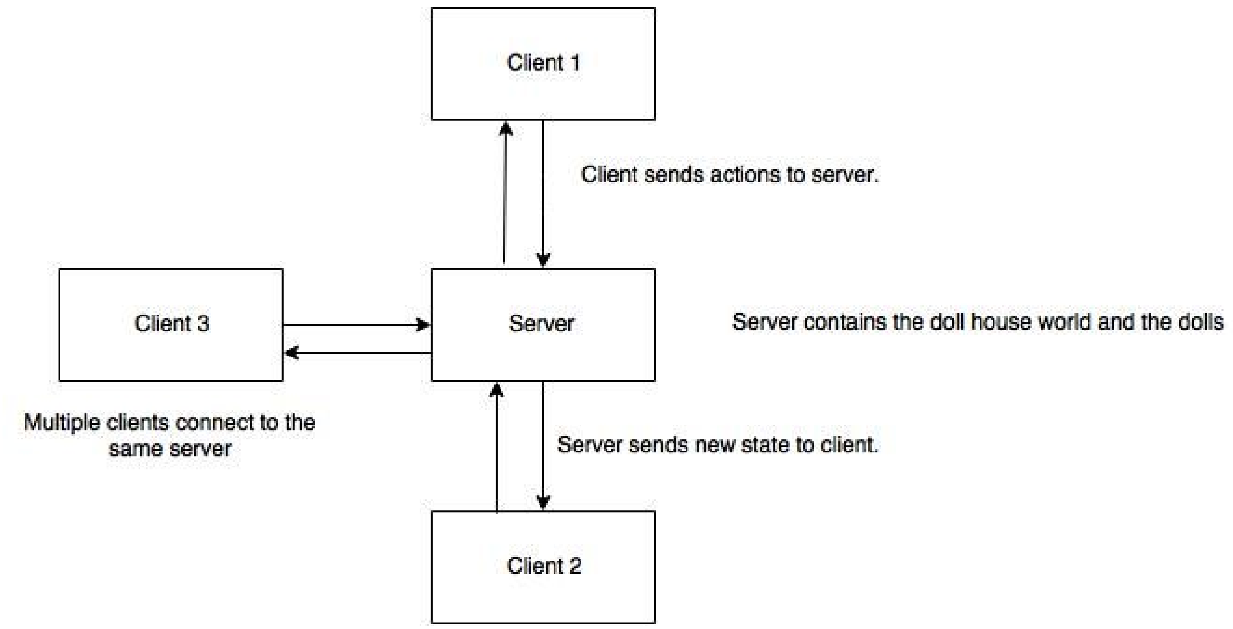
user2
Server
Agenda (12/5)
- Architecture → how worlds communicate
- Application → Decisions made (features) → pretty much done (ongoing)
- Interaction → Storyboard, Use case
- Implementation (overlap w/ UMLs, class descriptions)

- 

- Second UML diagram draft - can not apply the functionality that user can go to friends house



- First architectural sequence diagram draft


Client 1
Client sends actions to server.
Client 3
Server
Server contains the doll house world and the dolls
Multiple clients connect to the same server
Server sends new state to client.
Client 2

Design Methods

Application Design:

- *Feature Comparison* - We looked up different online dollhouse games to see what our "competitors" were doing. We saw that a lot of the games were almost the same: They allowed users to pick furniture items from a set of items (varying by colors) and the furniture would be placed in specific place. Many of the games did not have a doll, or allow the user to create a doll, it was mainly interior design. However, we liked many of the ideas we saw online and were influenced by them when deciding the features on our game.
- *Mind Mapping* - The very first thing we did as a group was create a mind map. The mind map allowed us to focus on the heart of the game (the dollhouse) and then delve into subcategories that the dollhouse would need (rooms, the doll, furniture). Once we had our subcategories, we needed to decide what we would allow the user to do with these subcategories. Could the user decide what color furniture they wanted? Yes, they could choose from a list of different items that were different colors. Could the user decide where to put their furniture? No, furniture location is already pre-established. Mind mapping allowed us to come to decisions like these.
- *Role Playing* - We took part in quite a bit of role playing when we began to evaluate whether the actions we've decided to allow for the user was something

they would want. Initially, we were only going to allow the user to create female dolls, but after doing roleplaying, we realized that the user may want to play a male doll. This led to us allowing the user to create a doll, starting with a gender-neutral template.

Interaction Design: For our interaction design, we ended up using three different design methods. We choose these methods because they allowed us to delve into what we believe the user will do/think when interacting with the user interface.

- *Personas* - We made personas for different users: a five-year-old named Hanna, a seven-year-old named Brandon, a thirty-year-old parent named Isaac, and a six-year-old named Lacy. By creating personas, we were able to think of what type of user would be using the dollhouse game. Brandon, Hanna, and Lacy are younger aged kids who love creating things and playing with dolls. By taking this into account, we knew that we had to design an interface that is easy for children to use, and create a game that peaks their interests (allowing them to create and design their rooms and allowing them to create dolls that can interact with other dolls). Additionally, Isaac is a typical parent who wants to ensure that their kids are playing a fun and safe game. By taking this into account, we knew we had to design an interface that requires parent authorization and have limited online chat features.
- *Storyboarding* - Creating a storyboard allowed us to have an idea of how a user would interact with the system. We drew what we believed the user should do when using our game. For example, when they start creating their doll, we plan that they will click a category (shirt), and then they will select an option from that category (blue shirt, red shirt, pink shirt). Storyboarding helped us see if we were missing any implementation or design logic of the game.
- *Think-aloud protocol* - While we did not have children to partake in our think-aloud protocol, we did decide to do it ourselves. By talking out loud, we were able to hear the user's thought process and what they would consider doing as the next step. Doing this also allowed us to realize any flaws in our system. For example, when we were having our doll do actions, we were unsure if the doll will do these actions by clicking on the doll and then choose an action, or if we will click a piece of furniture and then choose an action based off of the furniture. By thinking-aloud we were able to make decisions on things we had not thought about previously.

Architecture Design:

- *Model-driven engineering* - In order to keep track of the game's architecture, we decided to make a UML diagram. With a complex game like A Doll's World, it is easy to miss key information or design logic. By having the UML diagram, we were able to see the whole picture of the game and see if we were missing anything. We were able to focus on the different domains of the game and see how they are related with each other.

- *Decomposition* - After we determined the subcategories of the game during our mind mapping exercise, we were able to break down the assignment into smaller, doable tasks. As a result, we were able to address each task during our meetings and make a decision as a group. This increased our work productivity a lot, since we knew exactly what to work on next after we finished a task. We were also able to work on the assignment remotely, since each person was assigned a task to be finished.
- *Architectural style* - In order to implement the collaboration aspect of the game, we decided to have a dollUniverse that contains all the dollHouses of the users who play the game. Each user will have their own dollWorld, where they can visit the dollHouse of their friends. This functionality of the game will only be available with internet connection, since the online server must display each doll's action in real-time.

Implementation Design:

- *Summarization* - After the application, interaction, and architecture design, we were able to summarize our progress and see what else needs to be worked on. This is the part of the assignment where we looked closely at our design to see if there are any design flaws and missing class/attribute/method.
- *Visualization* - Having the storyboard helped us visualize how we want the GUI to look like, and how users might interact with the game.This helped us see if we were missing any important functionalities that the user might need.
- *Inspections/Reviews* - After all the deliverables for Design Studio 3 have been integrated into one document, each team member reviewed it to ensure that everything is complete and satisfies the team's standards. This guarantees that everyone in the team is happy with the final product of our assignment.

Final Report

<u>Design Decisions We've Made</u>

- For our implementation we chose to go with a version of the doll house in which users should pick rooms from template and while a minimum is in place for number of rooms the maximum is preset. The reasoning behind this decision is based off of real-life doll house's that come pre-made and you can't add and remove rooms.

- For our overall game scope we chose to take a top-down approach where there is a system-level universe which houses a collection of doll-worlds'. Each world is a personalized view of the all the doll house's based off user's and who their friends are. The doll world's consist of individual doll houses that are associated to users. Each user has ownership of their own dollhouse as well as dolls for the dollhouse.
- In order to afford more universal usability, taking into consideration who our target audience is, we decided to forgo any sort of login to access the game. The game must be downloaded onto to the desktop upon which initial user setup will occur. Users must create a unique username which will be the unique identification for the user account. This eliminates the need for young children to have to remember a login and password every time they want to play the game. Also, since the game must be downloaded in order to be played it is redundant to also have login.
- Another design decision we made was that every doll house comes equipped with its own car and it is the means for dolls to get to their friends houses. The car's aren't customizable because we felt that was outside of the scope of this application, but we choose doll's to have cars because it is a realistic method of getting to other friends houses.
- We allowed users to have multiple dolls, but not multiple dollhouses modeling the real-world doll-house. We chose this in order to limit the scope of the application and simplify design decisions in regards to displaying the doll's worlds.
- For security purposes, we decided to limit the online chat features of the game. There will be predefined text and emojis that the user can choose from to communicate instead of a chat box. Since there is no way to verify that the user is within the target range of 4-7 years old, we want to limit the chances of a kid interacting with a pedophile.
- For our interaction design, we wanted to keep things simple, since the targeted audience ranges from 4 - 7 year olds. We decided that moving a doll through the dollhouse would be limited to clicking on the doll and clicking on the desired rooms. Virtually, dolls will remain in one spot of the room. When a user wants to move them to a room, they simply click their doll and then click the desired room. This is the same with interacting with furniture or household items. In a room, a user simply has to click a piece of furniture and choose an action from a pop up list. They will then see the animation of the action.