Term Project for Internet Application Design and Development CSC667, Spring 2022

Section 01, Team A: Spaghetti Factory

—

Nakulan Karthikeyan
(ID: 920198861)
Johnson Nguyen
(ID: 921066533)
Thien Pham
(ID: 921100229)
Brian Adams
(ID: 921039987)

—

MILESTONE 1

(14 March 2022)

https://github.com/sfsu-csc-667-spring-2022-roberts/term-project-spaghetti-factory

Table Of Contents

1 List of Main Data Entities and Functionalities

➔ **Card**

The game we have selected is an UNO card game. As such, the central element of our game would be the cards themselves, which have the properties of “color” and “value”. There are of course special cards such as the reverse card, skip card, draw two card, and the wild card. These “special” cards would have their functionality defined by unique functions that would be abstracted from a pre-defined abstract class. The card entity would be evaluated if it is playable or not, depending on the currently open card’s “color” and “value” properties in comparison to the card that we are trying to play. Cards will be stored in a class with variablename “value” of string and variablename “type” of string.

➔ **Deck**

Our deck entity should contain a set list of UNO cards, and when this set of cards are presented as a whole, we get the card deck. At the beginning of each game, we will initialize and shuffle the cards. The users should “draw” from the deck entity, and a card refill should also be implemented if the deck has run out of cards. This “draw” function is extremely important and versatile, as not only should users “draw” from the card deck at the beginning of each game, but different game situations (i.e., the draw two card being played or the user not having the right card to play based on the game situation) would demand this draw function be implemented correctly in order for our game to run smoothly. A deck will be an array of cards.

➔ **Player**

The player entity is another extremely important data entity, as it determines how the client would interact with our game. Any single player would need to have a unique name (to differentiate between different players that would be playing) and an initial hand of cards at the beginning of each game. Note that each hand of playable cards has certain properties that would need to perform different actions. This includes identifying the current state of the player and the playability/possible actions of the cards the player holds. As we continue developing our project, the list of functionalities the player entity would contain may/may not increase! Player class will consist of id: int and deck: array.

➔ **Turn**

Our UNO game needs to have the functionality of "turns". Essentially, the player's sequence of actions during a turn would either be that they draw a card or play a card. A nonfunctional requirement we may be able to implement would be making it so that our game should automatically draw the card(s) for the user during their turn if they have no playable cards in their current hand. Note that the turns should operate in a clockwise or counterclockwise manner, which we can accomplish by way of a queue to determine current player and the players that would go (in order) afterwards. Enum: CLOCKWISE, COUNTERCLOCKWISE.

➔ **Game**

Our card game function is essentially a series of turns (defined above). Our iteration of turns should be executed until one of the player's meets the winning condition of holding no cards in hand. After the card game ends, we should send the user back to the join lobby screen so that they can play again if they wish to do so. Game class will consist of id: int, chat: array of string, previous card: card class, players: array of players, cards: card class, and decks: array of cards.

➔ **Game Metadata Object**

Our game object would contain metadata (state about current player, references to game decks, winning player, etc.), which we would piece together in one unified game object. This information will not be revealed to the player.

➔ **Restrictions**

Our game should have certain rules/restrictions in play to ensure that the game does not break on the client. These rules/restrictions include making it so that the user should be able to draw playable cards from the card deck ONLY if they have no playable cards on hand, the game rules (i.e., players can only play cards either of the same color or value as the last played card, or both), and the card rules (i.e., wild card means the user can select what color they wish to be used from the following turn onwards).

➔ **Chat**

Our chat object should exist in the memory cache. The chat option should only be enabled upon player login in the game lobby and the game instance itself.

2 Initial List of Functional Requirements

➔ **Guest User**

1. A guest user shall be able to search for available lobbies.
2. A guest user shall be able to sign up and make a new account.

➔ **Account**

1. A guest user shall be able to make a new account.
2. An account shall have a unique id for each user.
3. An account shall require a username, which must pass the conditions of starting with a character (a-z or A-Z) and having the length of at least 3 characters.
4. An account shall require a password, which must pass the conditions of having the length be at least 8 characters long and must contain at least 1 upper case letter and 1 number and 1 of the following special character (/ * - + ! @ # $ ^ &).
5. An account registration shall require the user to accept the Terms of Service and Privacy Rules of the website.
6. An account registration shall offer users the option to upload a profile picture (optional for the users).

➔ **Registered User**

1. A registered user shall be able to login to the website.
2. A registered user shall be able to update their profile information.
3. A registered user shall be able to play the game.
4. A registered user shall be able to play a game against either random users or join a game lobby using a unique key that's given to each unique game lobby.

5. A registered user shall be able to create their own game lobby and invite other users to their lobby.
6. A registered user shall be able to change their username and profile picture.

➔ **Profile**

1. All registered users shall have a profile.
2. A profile shall contain a profile picture of the user.
3. Registered users can edit their profile picture and username.
4. The profile menu will act as a dropdown menu and will contain a logout button on the dropdown menu.

➔ **Chat**

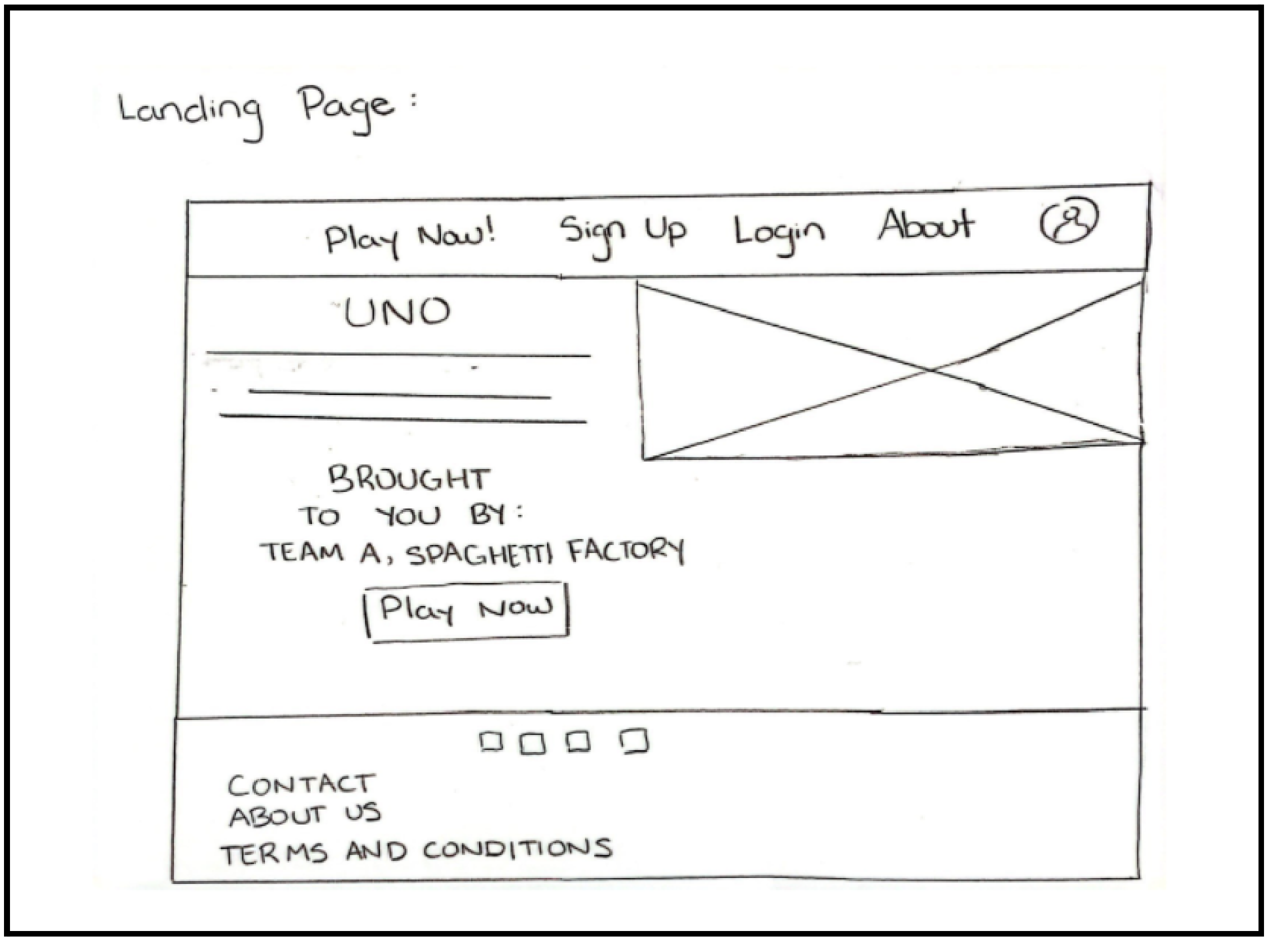
1. Registered users can chat with each other in game lobby.
2. Registered users can chat with each other in game instance.

3 Technologies Used

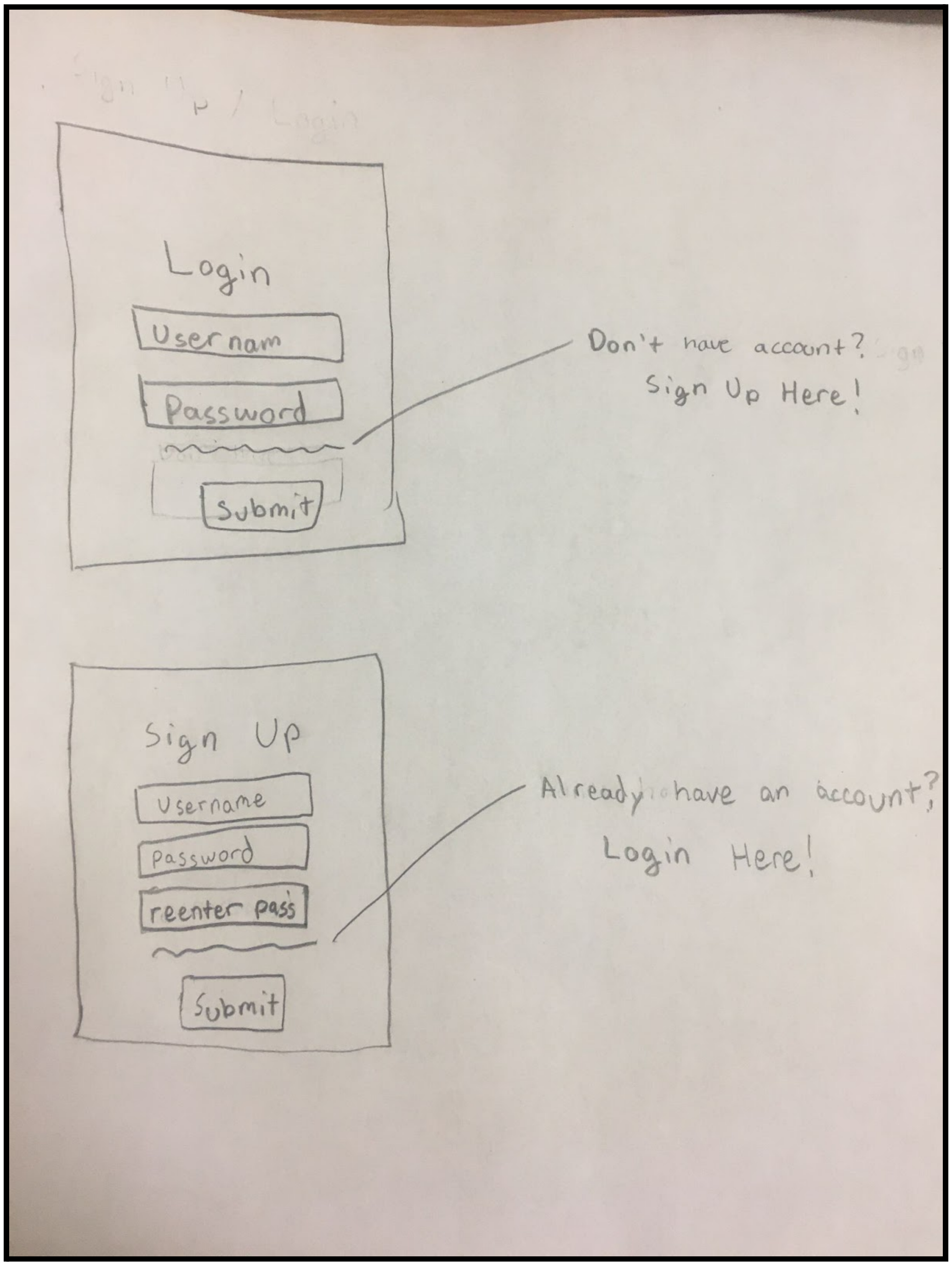
- ➔ Server Provider - Heroku
- ➔ Database - Postgres
- ➔ Backend - TypeScript, Express, Node 14.x
- ➔ Frontend - HTML, SCSS, TypeScript, HandleBars
- ➔ IDE - Visual Studio Code
- ➔ Version Control - Git, Repo hosted on GitHub
- ➔ Testing Library - Jest (Unit Tests), Cypress (End to End Tests)
- ➔ Node 14.x, TypeORM, Express, Express-Session, Jest, UUID

4 WireFrames

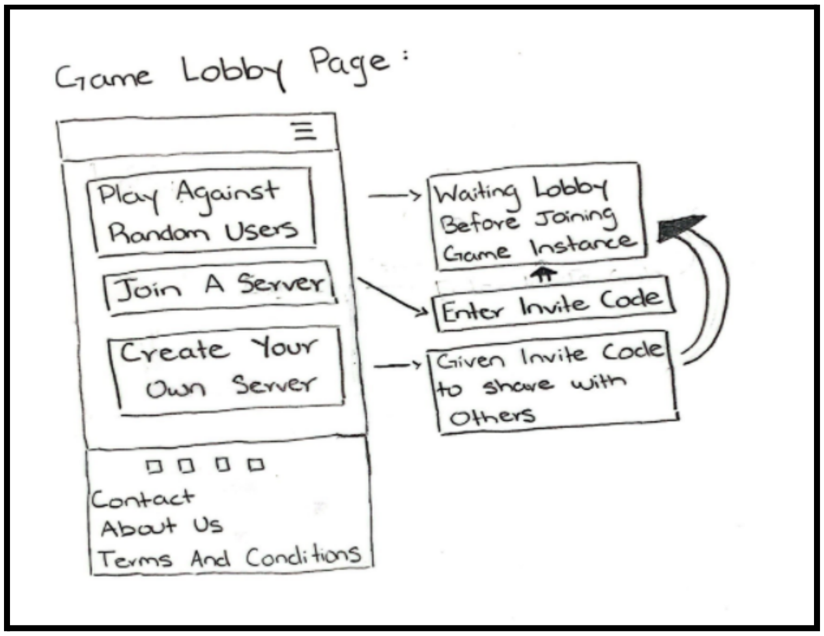
Landing Page:



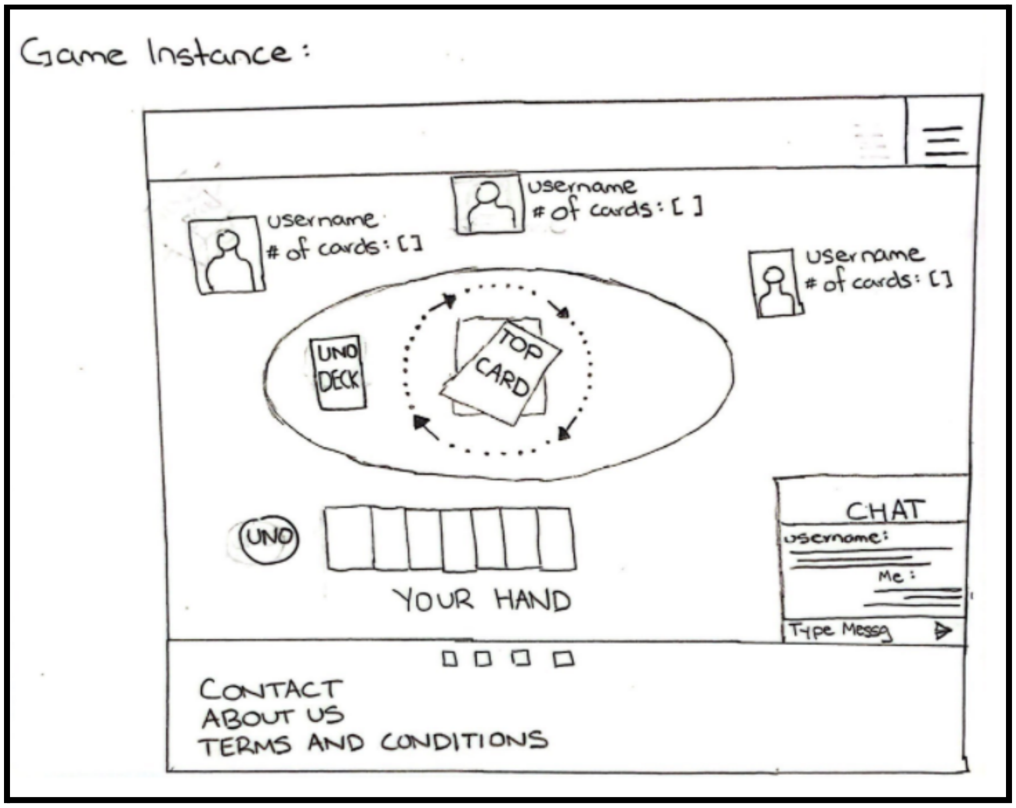
Login and SignUp Pages:


Login
Usernam
Password
Submit
Don't have account? Sign Up Here!
Sign Up
Username
Password
reenter pass
Submit
Already have an account? Login Here!

Game Lobby Page:


Game Lobby Page:
Play Against Random Users
Join A Server
Create Your Own Server
Waiting Lobby Before Joining Game Instance
Enter Invite Code
Given Invite Code to share with Others
Contact
About Us
Terms And Conditions

Game Instance Page:


Game Instance:
username
# of cards: [ ]
username
# of cards: [ ]
username
# of cards: [ ]
UNO DECK
TOP CARD
UNO
YOUR HAND
CHAT
username:
Me:
Type Messg
CONTACT
ABOUT US
TERMS AND CONDITIONS