University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 1

Date: July 14, 2024

Group Number: 51

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Perry Zhu	32653826	t9n60	perryz@students.ubc.ca
Yuchen Gu	37280534	i6m6h	guyuchen999@gmail.com
Ch Muhammad Daud Virk	26838482	e1h8m	daudvirk@student.ubc.ca

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Note: This refers to an Uno Game Application. As such you may say that this is an app where players can play against each other in a virtual environment.

Note: Some of the sub-entities (subentities of Action and Card) do not have extra attributes, as nothing distinguishes them apart from them being a separate entity. This is implemented because they are needed to satisfy the total constraint coverage, to ensure cohesion, and allow for appropriate relationships to be depicted (e.g. Look at the "Draw" and "Play" sub-entities for "Action"). Furthermore, there are, in total, six types of cards in UNO, and only Wild cards and Wild Draw 4 cards don't have color. Although these two cards do not have any extra attributes, we still include them to make sure we have covered all types of cards in the game and to satisfy total constraint coverage. Their primary keys are inherited from their respective super-entities.

1. **[Cover Letter Above.]**

2. A brief project description answering these questions:
    a. What is the domain of the application? Describe it.

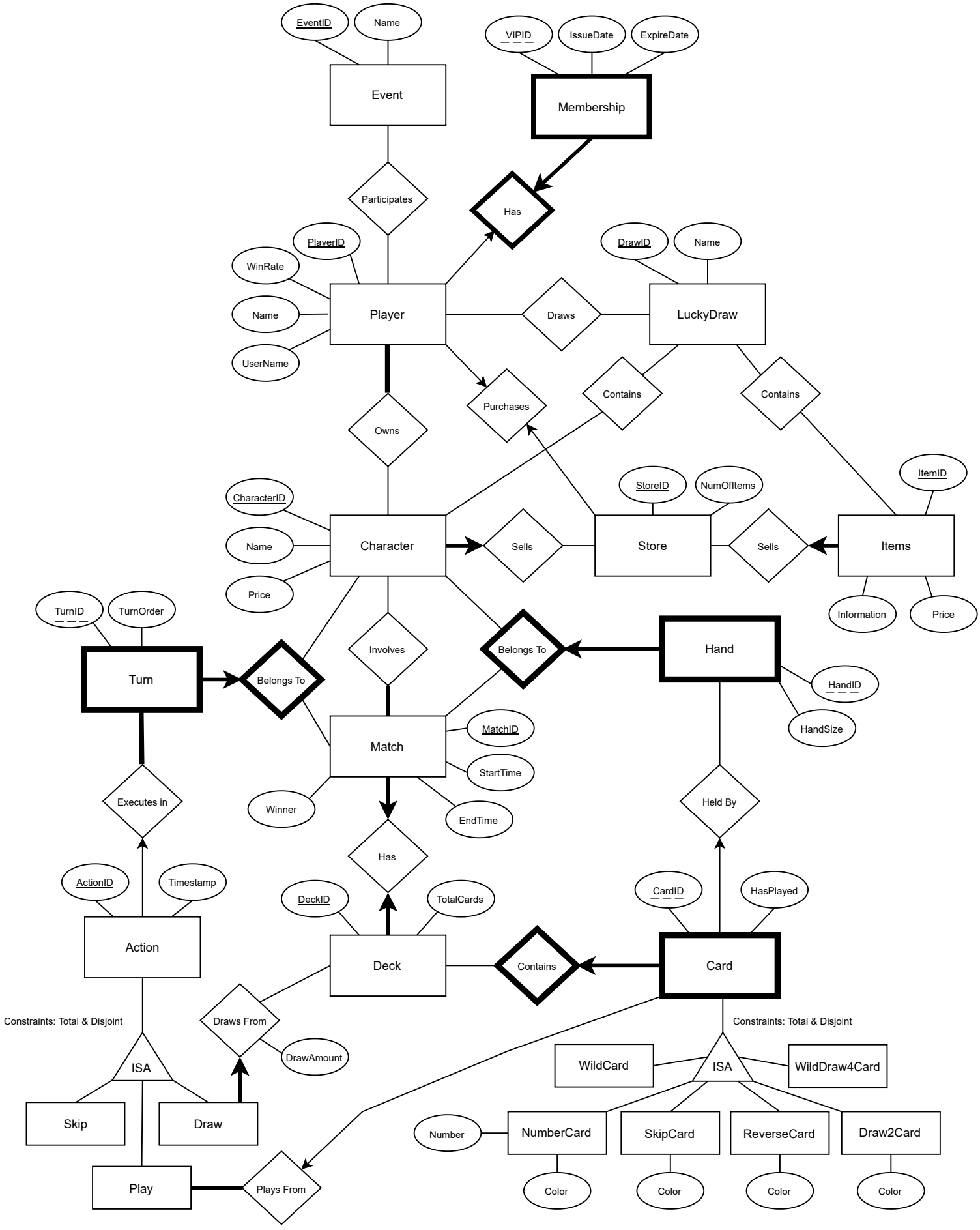
        - The domain is Game State Management and Logistics for Game App Users' information. The domain can be combined into one: Game Application Management. To be more specific, domain-wise, we use UNO, and game state refers to keeping track of the cards held by each player during a game, their turns et cetera. Similarly, for logistics we refer to keeping track of player information inside an Uno application (a game app) which includes information about virtual items and characters they have, and in-app transactions et cetera.

    b. What aspects of the domain are modeled by the database?
        - The Game State Management encompasses the entire game state of an Uno Game. For example, the cards held by each player, the type of cards held by each player, the cards remaining in the deck. and whose turn it is. A real-life situation would of course be a group of individuals engaged in a (virtual) game of Uno.This addresses the issue of tracking game states. For "Logistics for Game App Users' information," we keep track of virtual items that players have linked to their accounts e.g. badges, avatars, and virtual characters. A real-life situation to which this can be assimilated, is an online platform for a card game where players have accounts with virtual characters that represent them inside the game, and have items (badges etc.) linked to their account. This addresses the issue of account management. Furthermore, this database could also be modified briefly to apply in any online card game environment such as "Exploding Kittens".

3. Database specifications
    a. What functionality will the database provide?
        - This Database provides many comprehensive features that allow Players to interact with the gaming platform and their profile information in a

productive way. Players can create and update their profiles, view their game histories, and statistics (WinRate), and resume paused games. This will be for the Players, using the app. At the backend, the people using the Database will be allowed to manage user information as well (in the manner described above). The database will also allow for the management of active game sessions (via the "Game State"), tracking each player's hand, the state of the deck, and the current turn, ensuring the game can be played accurately. So the functionalities can be grouped into the management of User Information, the management/overseeing of current Game States, and record-keeping for previous Game Instances.

4. Description of the application platform
    a. What database will your project use?
        - We will use the department-provided Oracle (DB) Database Management System (on the remote server).
    b. What is your expected application technology stack?
        - We will use Hypertext Markup Language, Cascading Style Sheet, and JavaScript for the interface we will use to interact with the implemented Database (this will be the "frontend"). We will use Javascript with the Node.js runtime environment and the Express Node.js web framework for the "backend" API (that directly communicates with the database). Express will also facilitate communication with the database.
5. The Entity-Relationship Model [Next Page]:


EventID
Name
Event
VIPID
IssueDate
ExpireDate
Membership
Participates
Has
PlayerID
WinRate
Name
UserName
Player
DrawID
Name
LuckyDraw
Draws
Purchases
Contains
Contains
Owns
ItemID
StoreID
NumOfItems
CharacterID
Name
Price
Character
Sells
Store
Sells
Items
Information
Price
TurnID
TurnOrder
Turn
Belongs To
Involves
Belongs To
Hand
HandID
HandSize
Match
MatchID
StartTime
EndTime
Winner
Executes in
Held By
Has
ActionID
Timestamp
Action
DeckID
TotalCards
Deck
Contains
Card
CardID
HasPlayed
Constraints: Total & Disjoint
Draws From
DrawAmount
ISA
Skip
Draw
Play
Plays From
Constraints: Total & Disjoint
WildCard
ISA
WildDraw4Card
Number
NumberCard
SkipCard
ReverseCard
Draw2Card
Color
Color
Color
Color

(Other/) Explanatory Comments (for the diagram, if they are considered necessary):

- A membership cannot exist without a player, and each player only has one membership, and each membership belongs to one player.
- A player **must** have a character to play a game. (Upon account creation a default character is assigned to the player.)
- The Hand Entity refers to the hand of a Character (being used by a player).
- The Turn Entity keeps track of which Character's turn it is.
- Both of the above entities are weak entities and are identifiable via a Match and a Character entity. There is a ternary relationship for both of them.
- Each Match has only one associated deck and one vice versa. Each Match **must** have one associated deck and vice versa.
- A card that is played **must** belong to the card entity set. And you can play more than one card per turn.