Casino Simulator Final Report

Paige Johnson, Derek Gorthy, Michael Condon

May 2, 2017

Summary

A website that models a business environment by giving users the ability to play as a business owner or a business patron. Initially, the only business this project supports is a casino; the project supports the owners' accumulation of one or more businesses, the ability to add games to the casino, the modification of bet amounts of the games, and gives the patrons the ability to play the games and oversee their funds.

Project Requirements Met

Here we list out the requirements that were completed, and the ones that were not. We give a brief reason in the "incompleted" tables.

Table 1: Completed Non-Functional Requirements

ID	Requirement	Category	Priority
NFR-001	Longest login authentication will be 5 seconds	Authentication	Critical
NFR-006	Application will perform identically across Windows, Mac, Linux, iOS and Android	UI/Performance	High
NFR-008	Make the system as extensible as possible, especially thinking about the addition of new types of businesses	Extensibility	High

Table 2: Incomplete Non-Functional Requirements

ID	Requirement	Reason
NFR-002	System will support up to 1,000 casinos and 100,000 patrons	Untested
NFR-003	Databases for users and casinos will be independent	Incomplete Database
NFR-004	Databases will be easily replaceable with other databases	Incomplete Database
NFR-005	Users will only need to authenticate once every 24 hours	Not Implemented

Table 3: Completed User Requirements

ID	Requirement	Topic Area	Actor	Priority
UR-001	As a new User, I want to create an account, so that I can play as an Owner or Player	Startup	Program Driver	Critical
UR-002	As an Owner, I can create a casino, and chose what games to add into it	Casino functionality	Owner/Casino	Critical
UR-003	As an Owner, I want to be able to adjust the required bets of the games that I have already added to my casino	Casino functionality	Owner/Casino	Moderate
UR-004	As a User, I want to be able to log in to my account	Authentication	Owner	Critical
UR-005	As an Owner, I want to be able to view my list of businesses	Gameplay Backend	Player	Critical
UR-006	As an Owner, I want to be able to access an overview of my business.	Gameplay Backend	Owner	Critical
UR-008	As a Player, I want to be able to navigate to what game I want to play and be able to exit that game when I'm done, so that I can spend my funds where I want	UI/Gameplay	Player	Moderate
UR-010	As a Player, I want to be able to view a Casino's games, Net Worth, so that I can make a decision on what casino to gamble in	Player Capabilities	Player	Moderate
UR-011	As an Owner, I want to be able to sell a business, so that I can make money off of it and no longer moderate it	Owner Capabilities	Owner	Moderate

Table 4: Incomplete User Requirements

ID	Requirement	Topic Area	Reason
UR-007	As an Owner, I want to be to make my casino exclusive to the Players that I choose to invite	Gameplay Backend	Not Implemented
UR-009	As an Owner, I want to be able to see who is in my casino, so that I can moderate it	Gameplay Backend	Not Implemented

Table 5: Completed Functional Requirements

ID	Requirement	Priority
FR-001	Amount of money belonging to player shall be stored and player should be removed if money reaches 0	High
FR-002	When creating a Player profile, the "bankroll" for that Player will be set to $50,000	High
FR-003	When creating a Owner, the "initial capital" for the Owner will be set to $500,000	High
FR-004	When displaying the list of casinos for a Player to enter; total played, total paid out and win/loss ratio will be sorted and displayed	Moderate

Table 6: Incomplete Functional Requirements

ID	Requirement	Reason
FR-005	When selling a casino, the instance of the casino will be removed from the databases and the funds from scrapping the casino will be added to the Owner's balance	Incomplete Database

Original Class Diagram

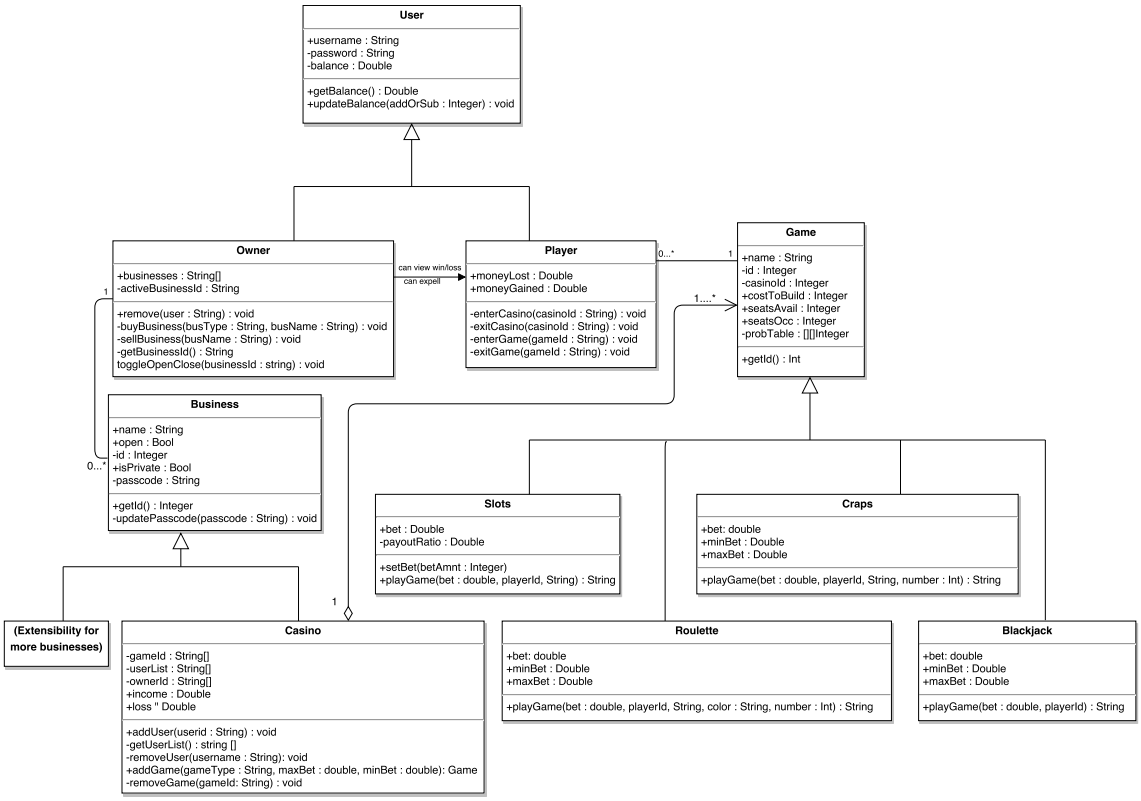

Figure 1: Class Diagram

Final Class Diagram

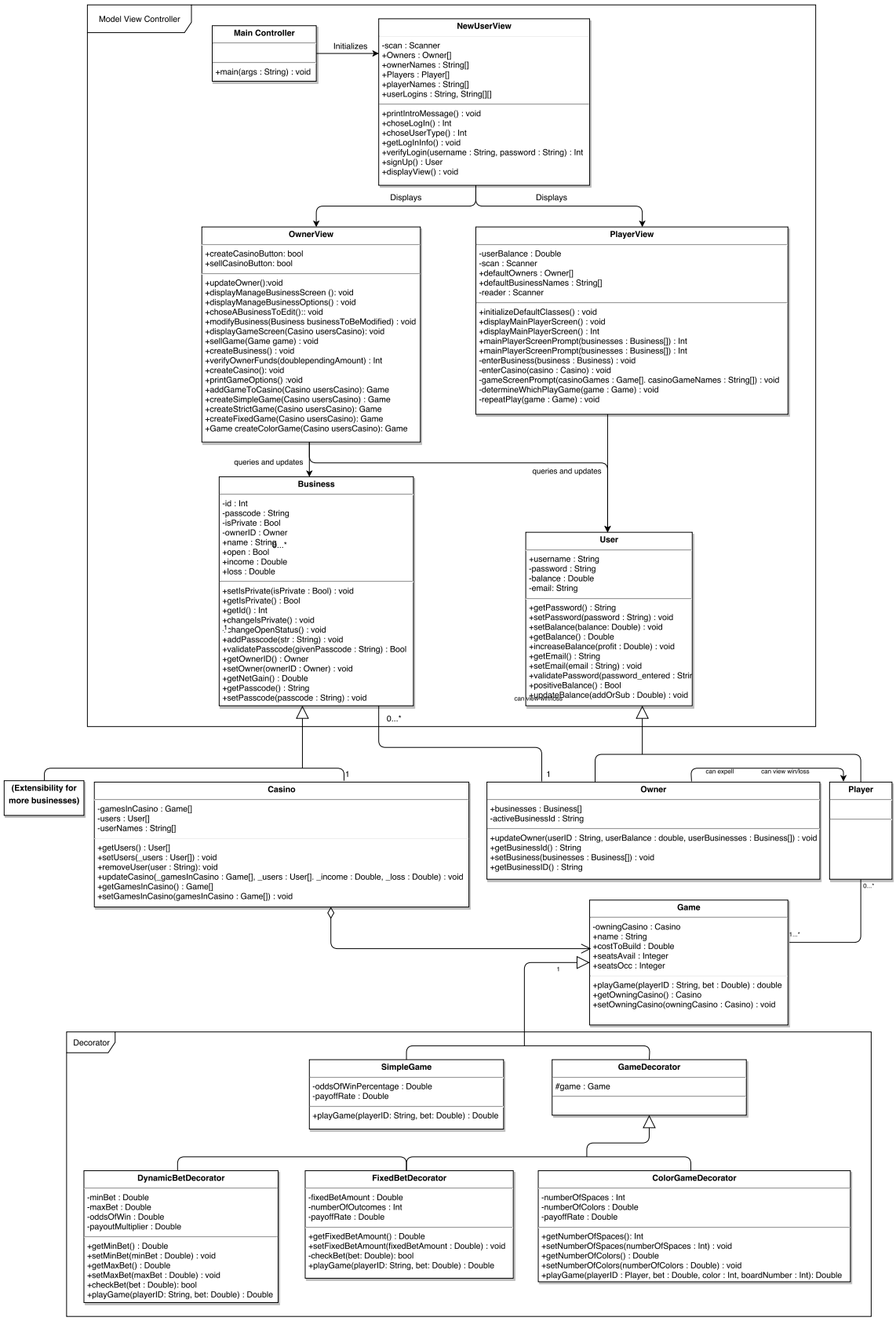

Figure 2: Final Class Diagram

Changes in the Class Diagram and Design Pattern

Our basic class structure worked well throughout this project. The relationship between businesses, casinos, owners, players and games remained fairly stable, although we ended up changing how these relationships are implemented.

5.1 Model-View-Controller Pattern

Initially, we were unsure on how to implement the visualization or the controller for the program. Each user lands on the initial Log-in/Sign-up page before being routed to either the Owner or Player view. Both views then interact with the rest of the classes. This implements the Model-View-Controller pattern because it gathers information from the user and passes it along to the rest of the classes.

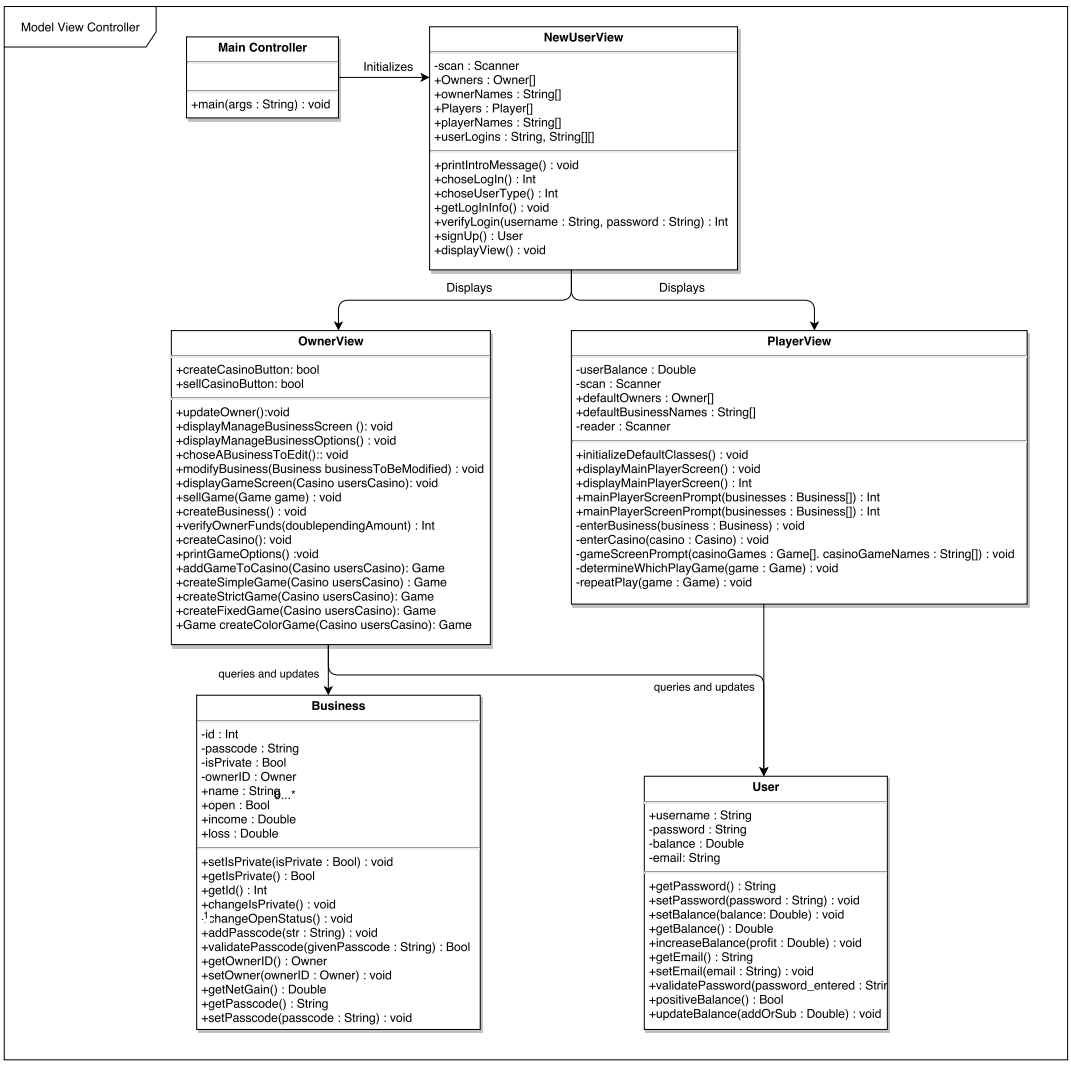

Figure 3: MVC Pattern Implementation

5.2 Decorator Pattern

The use of the Decorator design pattern simplified our system design more than any other improvement. In our initial diagram, we were only looking to implement a total of four games. This number was small because it was a realistic number of games for a project of this size, but did not actually represent the diversity of games in actual casinos.

Realizing that as the number of games scales to real-life quantities, the code would become bloated with repetitive code and classes with slight variations from each other. Instead, we chose to reduce games down to their core abstract concepts. For example, slot machines have a fixed bet and do not utilize the any cards or tables. On the other hand, a fixed-bet game of craps uses the same attributes and functions as a slot machine, but requires additional attributes and functions to reflect the craps table. By using the decorator design pattern, the fixed-bet code only needs to be written once.

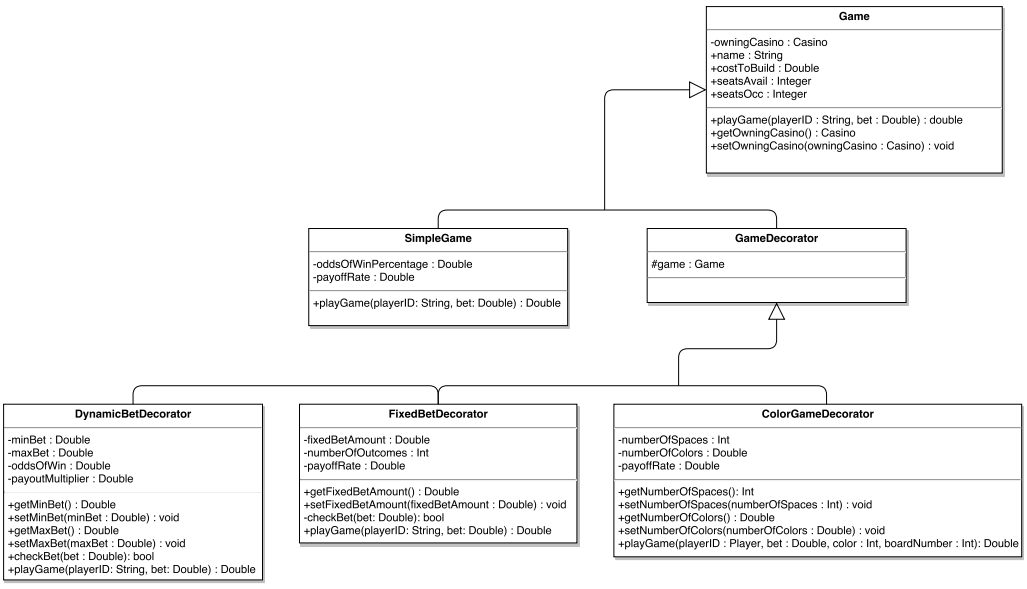

Figure 4: Decorator Pattern Implementation

5.3 Additional Functions

Another large change in our class diagram is the increase in the number of functions required by the controller. Unfortunately, these changes were not obvious until we began to implement the functionality. Then, we discovered a way that made the development of the project simpler in practice, even though it was not as clean on paper.

Retrospective

6.1 Up-front Planning/Refactoring Costs

If we hadn't changed our Game implementation, scaling the code to account for every type of gambling game would bloat the code and repeat sections of code dozens of times. This highlighted the importance of planning and refactoring; it is easy to be short-sighted and want to jump right into the development stage, but taking time to look into potential extensibility and plan around that.

6.2 Understanding the Advantages/Disadvantages of Design Patterns

Once it was clear that the design of our game design needed to improve, we struggled with choosing which pattern to use. Initially, we looked into using the Composite pattern. Had we gone with this pattern, we likely would have had to add much more code to force the Composite pattern to work. We failed to see the disadvantages of this pattern at first.

Eventually, we settled on the Decorator pattern. With this pattern, we considered both the advantages and disadvantages before deciding to implement it. With the sheer number of possible patterns, it is very important to consider all the advantages and disadvantages of a certain pattern before deciding to structures the design of the system around the pattern.

6.3 Revising Throughout the Development Process

While we did try to plan out every detail of the project before development, we largely figured out how to structure the controller after the development began. Additionally, we restructured the responsibilities of several classes after figuring out that the plan made a simple action much more difficult. This points to the conclusion that a simple class diagram can be revised after it is clear that a certain structure is more difficult that it looks. This reflected the concept of the agile development process – revising structure fluidly throughout the project.