GambleMania

Collin Winstead, Diego Arriaga

Abstract

GambleMania allows the user to play the popular casino game of blackjack against a computer-controlled dealer. Blackjack uses a custom design for the deck of cards the game uses. GambleMania does not require any real money to play; all money is virtual and has no real monetary value. Users will have their own accounts which will contain login information and the account's balance. This information will be managed with a database.

1. Introduction

Our project is called GambleMania. It allows you to play the blackjack casino game. The player's balance will be saved in a database along with their login information. We decided to do this project because we thought it would be fun and require a good bit of research and work to finish. As for why we chose blackjack as the game to implement, it's because blackjack is a fairly simple game to learn and play.

As blackjack is fairly simple, it is easier to program than more complex games like poker. Our target audience is adults 18 years old and older because blackjack is a game that is typically played in casinos. We expect our audience to use our app simply for fun or to practice blackjack.

1.1. Background

Minimally, we expect the user to know how to play blackjack and be familiar with the terms used in blackjack. The goal of blackjack is to get the point value of your hand as close to 21 as possible without going over. The terms you should know are defined below.

- Ante - the amount of money you bet before you play the game.
- Blackjack - when your hand has a point value of 21.
- Bust - when your hand has a point value greater than 21.
- Push - when your hand and the dealer's hand have the same point value at the end of the dealer's turn.
- Hit - deal another card to your hand.
- Stand - end your turn.
- Double down - double your ante bet and be dealt one, and only one, card to your hand.
- Split - double your ante bet and split your hand into two hands. Both hands are dealt another card.

1.2. Impacts

Although our project is very small and limited, if distributed the project could possibly have a cultural impact by getting more people into blackjack.

1.3. Challenges

Our biggest challenge was the logic behind blackjack. In blackjack, there are many conditions which you have to check for such as getting a blackjack or busting your hand. You have to check these conditions for both the user and the dealer, which further complicated the development of the game. This challenge was overcome by applying conditionals in places where they would only need to be checked once.

Before we began developing Gamblemania, we thought that programming the dealer's functions would be the most difficult challenge. However, this turned out to be almost trivial for blackjack because the dealer must play according to a strict guideline: hit until you get at least 17 or your hand is greater than the player's, then stand. This rule was handled by some simple conditionals and a loop.

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Log In	User	Easy	1
2	Sign Up	User	Med	1
3	Play Blackjack	User	Hard	3
4	Navigate the Main Menu	User	Easy	2
5	Exit the Program	User	Easy	4

TABLE 1. USE CASES

Another challenge we faced was the "split" functionality. Giving the user another hand doubles the amount of logic the program needs to run the game. By adding another hand, we now have to compare the user's hands seperately against the dealer's hand. This challenge has not been completely solved yet, as there are still some bugs with splitting. However, splitting works normally for the most part.

2. Scope

Firstly, the user should be able to navigate between pages by clicking a button on the page they are currently on. Users should be able to create accounts and log into them. The main menu should minimally allow the user to navigate between pages by clicking a button and return to desktop by clicking an exit button. The user must be able to play a game of blackjack with the rules of dealer stands on 17 and blackjack pays 3:2. Finally, all of the cards that you would use in blackjack must be designed.
Below is the future work we could do to expand GambleMania:

1) Add more games to the app.
2) Create a tutorial for each game for users who do not know how to play.
3) Make the main menu and game environments look better.
4) Networked gameplay

2.1. Requirements

The basis of the base requirements stems from the rules of play for blackjack. There are also requirements that involve the UI design, some of which are simply required for a functional app and others are for a good-looking app. The non-functional requirements are based on how the user will interact with our app. These requirements are not necessarily required for the app to function, but are important for the users of our app.

2.1.1. Functional.

- A button that exits the program in the main menu and in each of the games.
- Buttons to navigate from the main menu to each of the games.
- A back button in each game that navigates the user back to the main menu.
- Game logic for each game.
- A Microsoft Access database that stores user login information and user balances.

2.1.2. Non-Functional.

- User login data is hashed.
- The app should be responsive and perform well without any major dips in framerate.
- When the user's balance is modified, it should be updated in the database as soon as possible.
- The app should be flexible enough so that it is possible to scale it up in the future.

2.2. Use Cases

This section defines and ranks the use cases for our project. A table of use cases is shown in Table 1.

2.2.1. Use Case 1: Log In. See Figure 1

1) User enters their username and password into the provided text fields.
2) User clicks submit.
3) If the username and password are found in the database, the user proceeds to the main menu. Otherwise, the user must keep entering the username and password until the username and password matches.

2.2.2. Use Case 2: Sign Up. See Figure 1

1) User enters the username and password for their new account.
2) User clicks submit.
3) Account is registered to the database and the user proceeds to the main menu with the new account. The new account will have a starting balance of $1000.

2.2.3. Use Case 3: Play Blackjack. See Figure 3

1) User places an ante bet.
2) User and dealer are dealt two randomly chosen cards.
3) User is shown a maximum of four options: Hit, Stand, Double Down, and Split. Double Down is only shown on the first turn. Split is only shown on the first turn when the user has two cards of the same face.
4) If the user chooses the Hit option, the user will be dealt another card.
5) If the user chooses the Stand option, the dealer will begin their turn.
6) If the user chooses the Double Down option, the user will place a bet with the same value as the ante bet and be dealt another card. After this, the dealer's turn begins.
7) If the user chooses the Split option, the user's two cards will be split into separate hands. Each hand will be dealt another card, giving the user two hands with two cards each. The user can play each hand separately.
8) After the user's turn is over, the dealer's turn begins. The dealer uses the same options the user has.
9) After the dealer's turn is over the user's hand and the dealer's hand are compared. The hand with the closest point value to 21 is the winner.
10) After the game is over, the user may enter another bet or exit to the main menu.

2.2.4. Use Case 4: Navigate the Main Menu. See Figure 2

1) The user can click the "Blackjack" button to go to the blackjack game screen.
2) The user can click the "Log Out" button to log out of their account and go back to the log in/sign up screen.
3) The user can click the "X" button at the top right corner of the screen to exit the program.

2.2.5. Use Case 5: Exit the Program. See Figure 2

1) The user can click the "X" button at the top right corner of the main menu screen to exit the program.

2.3. Interface Screenshots

Figure 1. Login/sign-up GUI

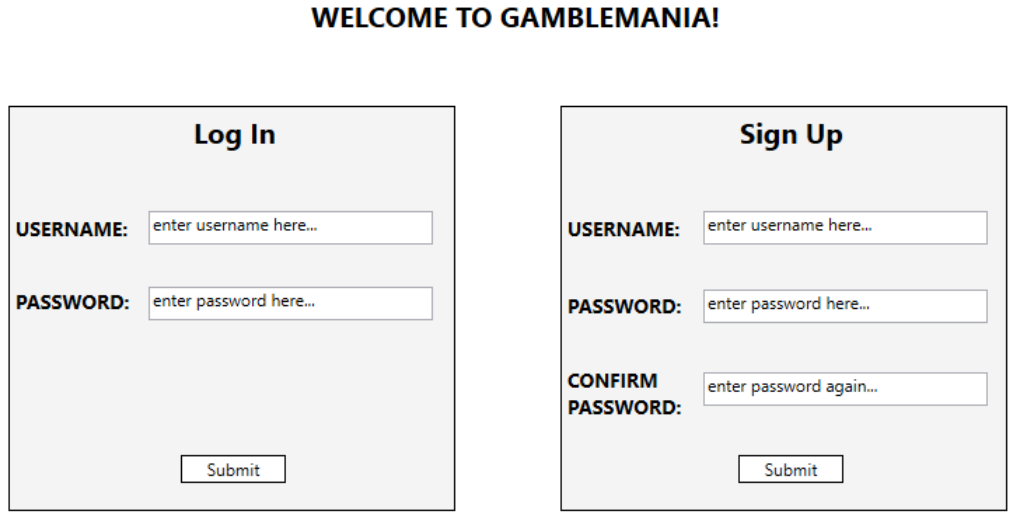


Figure 2. Main menu GUI

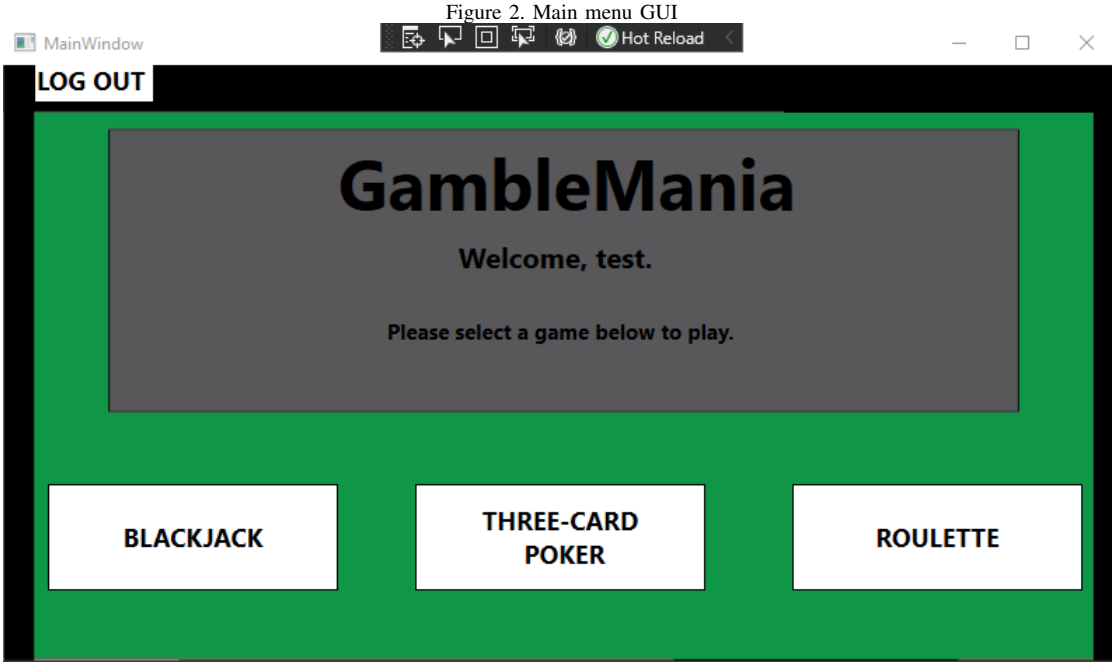
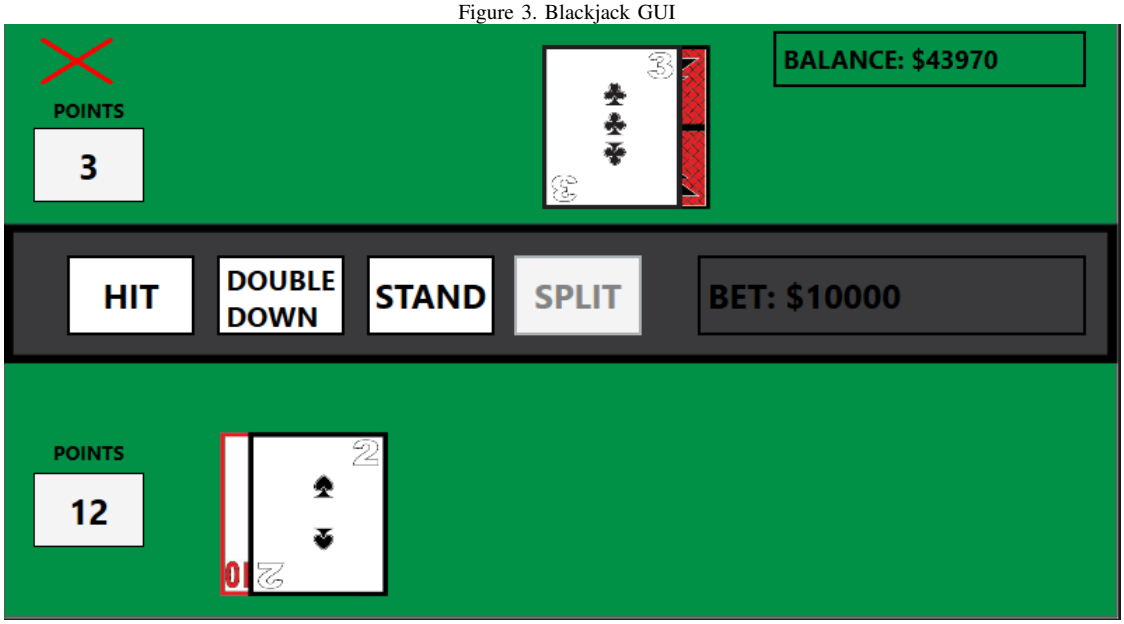


Figure 3. Blackjack GUI



3. Timeline

Below is the timeline of the project. This timeline is accurate up until 11/11. We were unfortunately delayed this week due to other events and could not work on the project. From then on, we were behind schedule and were not able to implement three-card poker and roulette as a result.

8/27 - Finding partners.
9/2 - Formed a team.
9/9 - Created a rough draft of the project proposal.
9/16 - Designed a plan of action for the project.
9/23 - Worked on the GUI mockups and the Requirements and Use Cases sections of the project proposal.
9/30 - Updated the GUI mockups.
10/7 - Researched potential design patterns for the project.
10/14 - Modified the Use Cases section of the project proposal.
10/21 - Midterms
10/28 - Added the Timeline and Project Structure & UML Outline sections. Begin implementing the login/sign-up screen and database.
11/4 - Begin work on blackjack and three-card poker. Login/sign-up should be done by this point.
11/11 - We should be about halfway done with blackjack and three-card poker by this point.
11/18 - Begin work on roulette. Blackjack and three-card poker should be done by this point.
11/25 - Begin work on the presentation and work on stretch goals such as the tutorials.
12/2 - Demo the project. The project should be finished by this point.

4. Project Structure

This section covers the general structure of our app. We designed our project's structure to be flexible and expandable. We made our project modular by using pages, which can be swapped in and out as needed. This means new pages can be added and removed very easily, and one page does not necessarily depend on another. Another way we made our project modular is that some of the classes we used in the Blackjack_Page can be used in another card game. These classes are the CardFactory class, the card struct, and the CardType enum. All of these classes can be applied to a different card game.

4.1. UML Outline

This section provides a UML outline of the project (see Figure 5 and Figure **??**). The outline was too big for one picture, so we had to separate it into two parts.

The project's structure hinges on a MainWindow object. The window has an object called a Frame that holds the page the user is currently on. At launch, the frame contains the log in/sign up page. To navigate between pages, we simply set the frame's content to a new instance of the page we want to navigate to. For each of the following pages, a reference to this frame is passed.

The log in/sign up page connects to the MS Access database through OleDb to log a user in or create an account for them. This page has two main methods: LoginSubmit_Click and SignUpSubmit_Click. As their names imply, these methods handle log ins and sign ups respectively. LoginSubmit_Click hashes whatever the user entered into the username and password fields on the page and compares them to every account in the database. If a match is found, then the user is logged in with that account. If not, the user is prompted to try again. SignUpSubmit_Click checks the validity of the username and password the user submitted and creates a new account in the database for the user if they are valid. "Valid" in this case is when the password is at least 10 characters long and contains at least one uppercase letter, symbol, and number. The password must also match what the user entered in the "Confirm password" text box. The username must not already be registered to another account.

The main menu page serves as a hub for navigation to other pages. Currently you can only navigate to the blackjack page and log out, but there is room for expansion when we get to that point. The two methods of this class, Black_Jack_Button_Click and log_out_btn_Click serve this purpose. They siimply navigate the user to the appropriate page.

The bet page does exactly what it implies: it allows the user to place a bet before they get into the game. Whenever the user enters a bet, the an event is raised that calls bet_input_TextChanged. This method validates the bet the user typed

into the textbox. If the bet is valid, the user may click the submit button and navigate to their chosen game. The functionality for this is handled by submit_btn_Click.

The blackjack page is where the blackjack game is played. This class has several methods. There are four methods that represent actions the user can take in the game. These actions are hit, stand, double down, and split. There are four buttons that handle the logic behind these actions. Double down and split are only available on the player's first turn after they are dealt their starting hand. There are five methods in this class that deal with the functionality of the game. The first is checkHand, which checks the state of the given hand and returns that state as a string. So if the user has a blackjack, it will return "blackjack." It will return the point value of the hand if the hand has no special state. A similar type of method is the insertCard method. This method simply adds a card to the given hand. The next method is Deal. This method initializes the game by dealing two random cards to the player and dealer. This method also handles blackjacks when needed. When the player clicks the stand button, dealerPlay is called. This method handles the logic behind the dealer's turn. This is made easy by the fact that the dealer must follow certain rules. The final functionality method is the play_btn_Click method. This method reveals the page content and calls Deal, which initializes the game. The last method in this class is the exit_btn_Click method, which simply closes the application.

Inside of the blackjack page class are three other classes: CardFactory, a card struct, and the CardType enum. The card struct defines what a card is. It contains three primitive values: "value", "suit", and "face." "value" is the amount of points a card is worth. "suit" is the suit of the card (e.g. hearts). "face" is what is displayed on the card (e.g. 'K' for a King). CardType defines every card in a standard deck of cards (52 cards in total). CardFactory is an implementation of the Factory Method design pattern. Its role is to create a card object given a CardType.

Figure 4. The Blackjack Page UML

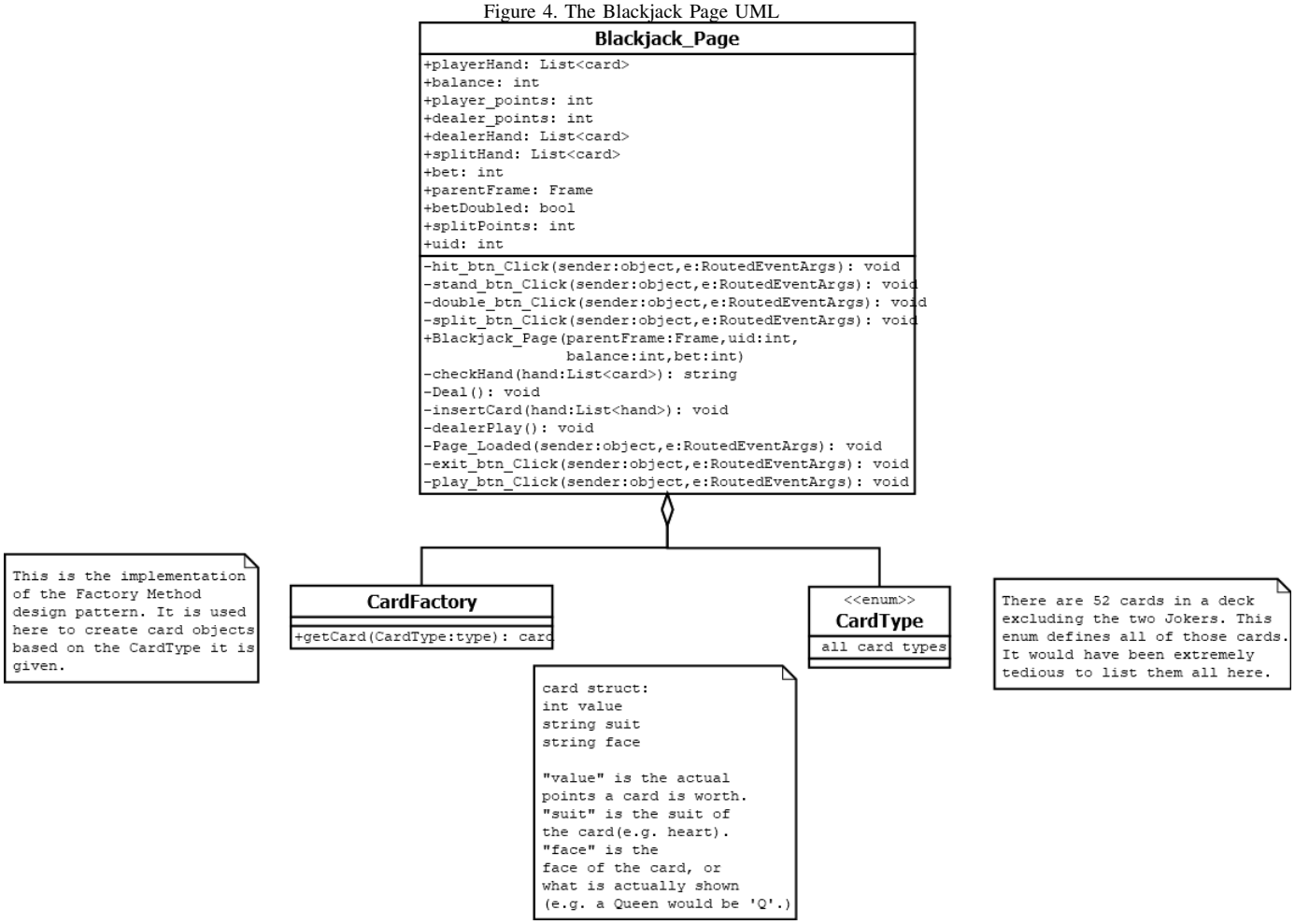
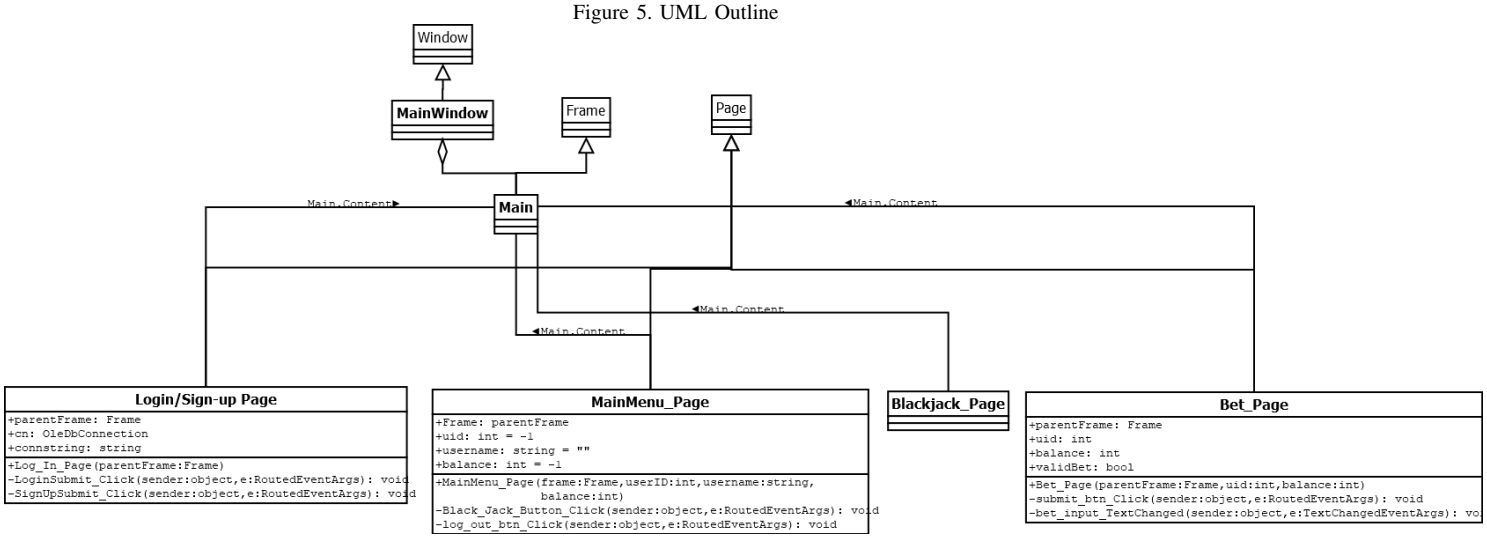


Figure 5. UML Outline



4.2. Design Patterns Used

We used the Factory Method creational pattern and the Façade structural pattern for this project. We used the Factory Method to create objects that represent cards in our game. We chose to use this pattern because we needed a method of generating many different variations of the same object, and the Factory Method does that well. We used Façade to make playing the game much easier for the user. All the user needs to do to control the game's subsystems is click some buttons.

5. Results

Looking back on the project, we managed to create a functional app with the limited time we had. As we developed the app, we discovered that our structure would definitely not work, so we had to quickly adapt to the changes in the structure. For example, we had planned to use a NavigationWindow as our method of navigation, but we discovered that it did not support the UI elements we needed to implement. Although there was probably a way around this, we found Pages to be easier to use. There is definitely room for improvement and expansion in the app, which is described in the Future Work section. As for our accomplishments, firstly we created a functional, albeit simple, user authentitication system using a Microsoft Access database. Second, we implemented user balances in the database. Finally, we created a functional blackjack game. Given more time, we could make many more accomplishments.

6. Future Work

In the future, we would like to expand the scope of our app to include the other games we originally planned to include: three-card poker and roulette. Another game we could potentially implement is Texas Hold'em style poker game that can be played over the internet. Unfortunatley, due to time constraints we were only able to implement blackjack into the app. Finally, we would like to implement a tutorial page that contains tutorials for each game.