SDP22: Team 20

JACKBLACK

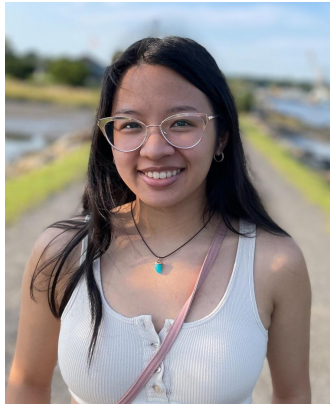
Ray Castillo, Alina Daraphet,
Marvin Nguyen, Lamisa Sheikh

Advisor: Bill Leonard

Meet the Team

Professor Leonard
Advisor

Ray Castillo
Computer Engineer



Alina Daraphet
Computer Engineer

Lamisa Sheikh
Electrical Engineer

Marvin Nguyen
Computer Engineer

Problem Statement

When playing Blackjack among friends, the issue of who will be the dealer always arises. The job of the dealer includes:

- Dealing and shuffling cards
- Collecting and counting chips

These are tedious tasks that most players do not want to take on.

Our Solution

We are proposing an automated Blackjack system that has the following features:

- Automated shuffling and dispensing system
  - Users input the deck into the dispenser and the cards will shuffle and dispense themselves to each player/dealing system
- No chips (non-physical betting; using play chips in system)
  - Incorporate the betting system in a virtual environment

System Specifications [pt.1]

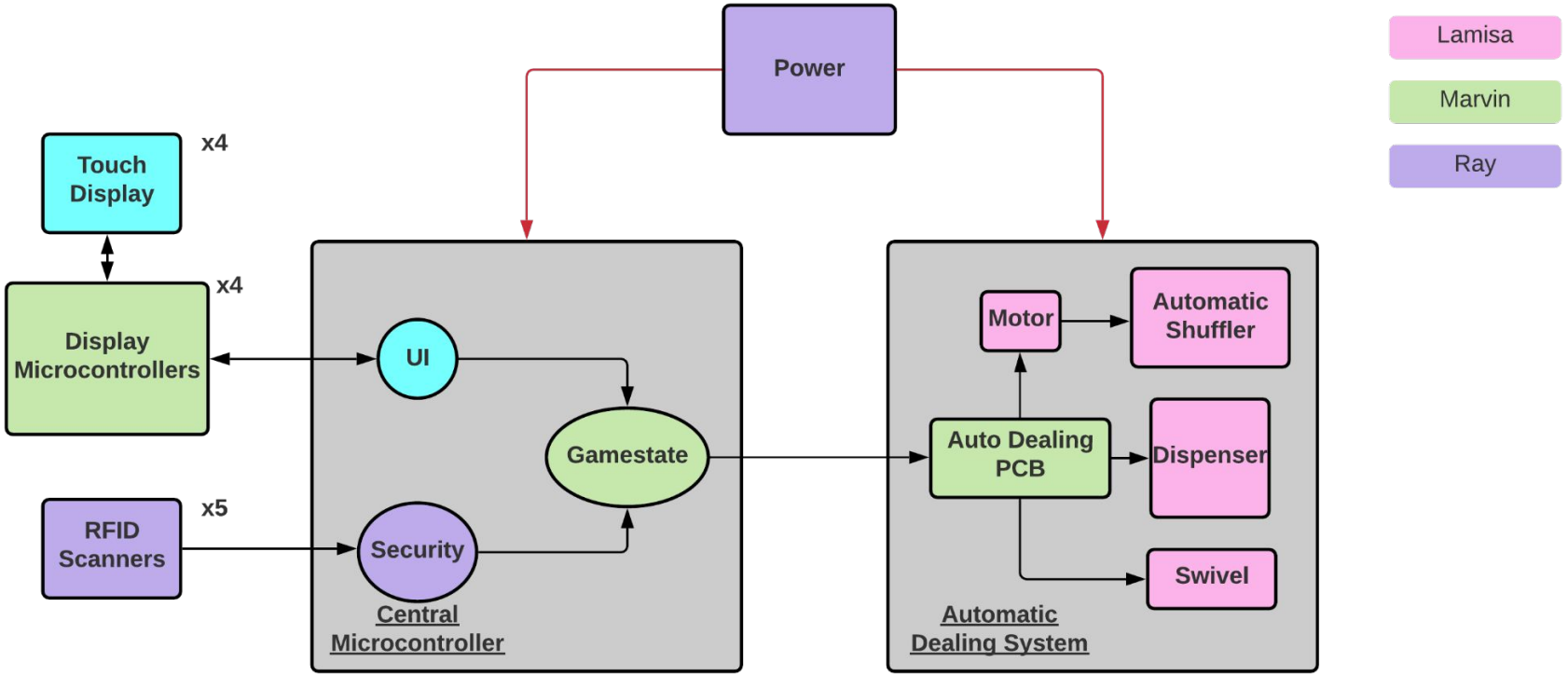
1. Shuffle all cards loaded into the system when prompted by the game.
2. Transfer cards from shuffling system to dispenser without dropping any cards.
3. Dispense cards to community/dealer as well as dispensing cards across the table to the player's designated section (within reach of the player's screen/designated area, without having cards fall off the table, and allocating the correct number of cards to each player).
4. System will stop once an "end" state is reached.
    a. "End" state can be reached by:
        i. A button on the UI, where a majority of players can vote to end the game
        ii. When a player's total winnings reaches the "winning amount"
        iii. After a certain amount of games (player inputs number of games)
        iv. Players can input duration of play before end game (e.g. 1 hour of playtime)
    b. On game end, display on UI: Total winnings, "play again", "exit"

continued....

System Specifications [pt.2]

5. System will identify cards by their number/letter and suit with 99% accuracy of recognition.
6. System will engage with the user by the following:
    a. Display wins/player money
    b. Player input (Betting / Player Moves)
    c. “Help Guide” to teach new players
    d. Asking the number of players (1-4 players)
    e. Asking for house rules
7. Players will not be able to substitute cards from another deck into the system deck.
8. System will organize betting virtually and display values through UI: Each player will have their own touch display.

System Block Diagram

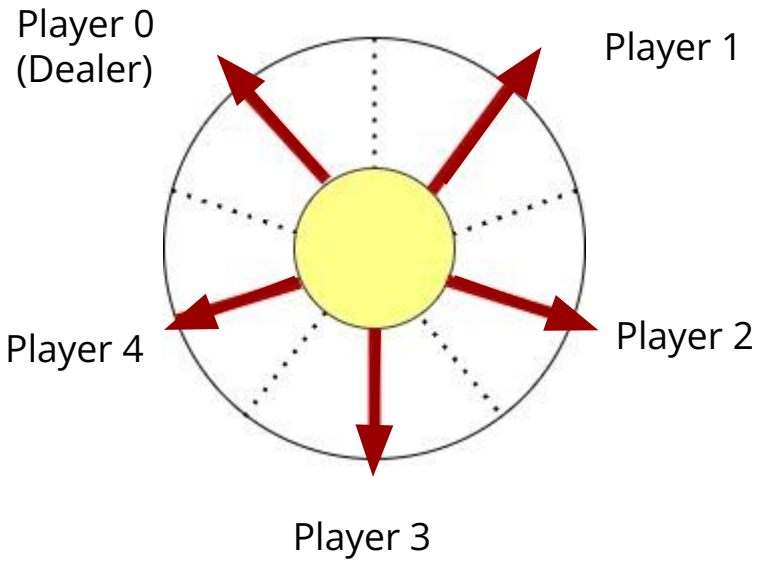


Proposed MDR Deliverables (Checklist)

- Functioning card dispenser integrated with rotation mechanism able to dispense cards
- Functioning Shuffler
  - Components of Dealer should be controlled by input
- Card identification able to get specific card number and suit
- Game algorithm for Blackjack
- GUI prototype for touch display

Card Dealer (shuffling, dispensing, and rotating as a whole)

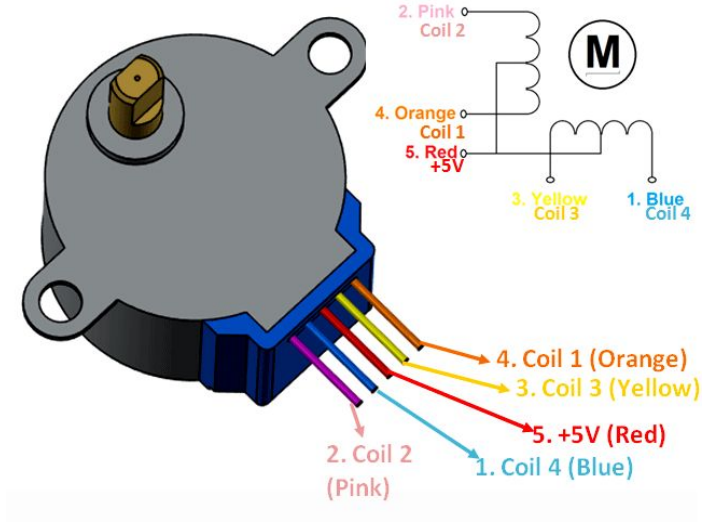
Dealer indicated by yellow circle. Red arrows show stopping points/trajectories for cards being dispensed, and dotted lines show section for each player



Stepper Motor

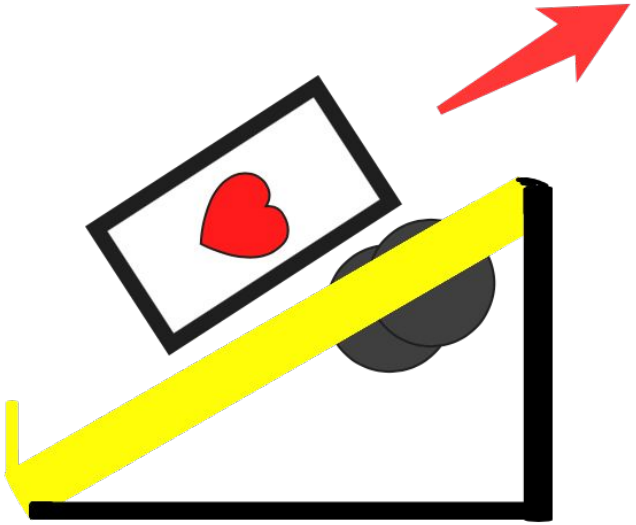
- This stepper motor takes 2048 steps for one full rotation, but ours is programmed for 2045 steps before turning counterclockwise all the way back to the starting point.
    - This is because 2045 is divisible by 5 (for each player).
- It rotates back counterclockwise as opposed to a continuous clockwise rotation so that the wires do not get tangled.
- The system is programmed to do this complete cycle twice so that each player is dealt two cards.
- It is also programmed to travel at 5 rotations per minute when it is rotating. The stepper motor has the capability to travel at 10 rotations per minute, but we chose 5 as this seemed like the most reasonable pace
- This type of stepper motor consumes high current therefore a driver is needed.

- Rotation Calculation:
  (2pi [rads] / 5 [players]) * number of segments from origin
  (2045/5) * input -1



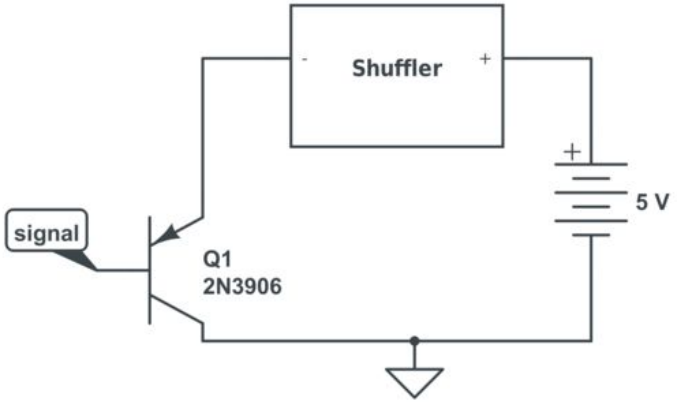
Dispenser

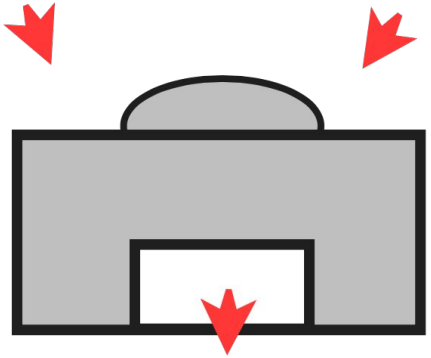
- The dispenser part of the dealing system a whole is an old UNO card dispenser that we hacked into, or repurposed
- Uses a transistor as a switch
- Requires 5V and is powered by Arduino
- pulses for 0.115 seconds (115 ms)
- Mechanical Aspect: Two rubber tracks to push one card out at a time. A weight is required on top of the cards to keep them in place and only dispense one card at a time.



Shuffler

- Set gears to rotate for 5 seconds to allow one deck to shuffle through fully
- Requires 5V
- Uses Transistor as switch





- Option for multiple shuffles
- Must be stacked on top of dispenser. Considering using 3D printing to aid in this

Power

- System consumes too much power being run solely using Arduino 5V
  - Issues we encounter: Motor consuming too much power, not enough to power both motor and dispenser
- System will be powered through a wall outlet (5 volts, 1 amp, 5 watts)
- Adafruit PN532 (for RFID Card reading) requires 3.3V
  - We will need to step 5V down to 3.3V potentially using a level shifter / opamp

Proposed MDR Deliverables (Checklist)

- Functioning card dispenser integrated with rotation mechanism able to dispense cards
- Functioning Shuffler
  - Components of Dealer should be controlled by input
- Card identification able to get specific card number and suit
- Game algorithm for Blackjack
- GUI prototype for touch display

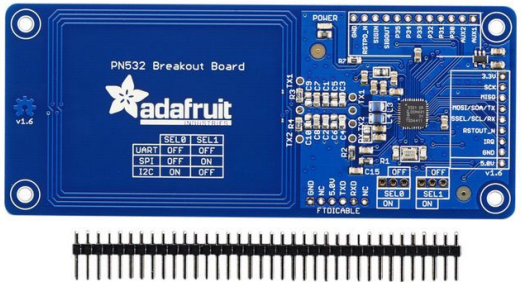
RFID Card Reading

Adafruit PN532

3.3V

Gets a uid len of 7bytes

Verifies UID against stored uids to get a value



Reads every .5 seconds (can change that time)

2 Cards at once (working on allowing for multiple cards at once)

Z axis (4.5 inches height)

XY Plain (1 inches from center face down)(3 inches from center sideways)

Proposed MDR Deliverables (Checklist)

- Functioning card dispenser integrated with rotation mechanism able to dispense cards
- Functioning Shuffler
  - Components of Dealer should be controlled by input
- Game algorithm for Blackjack
- GUI prototype for touch display

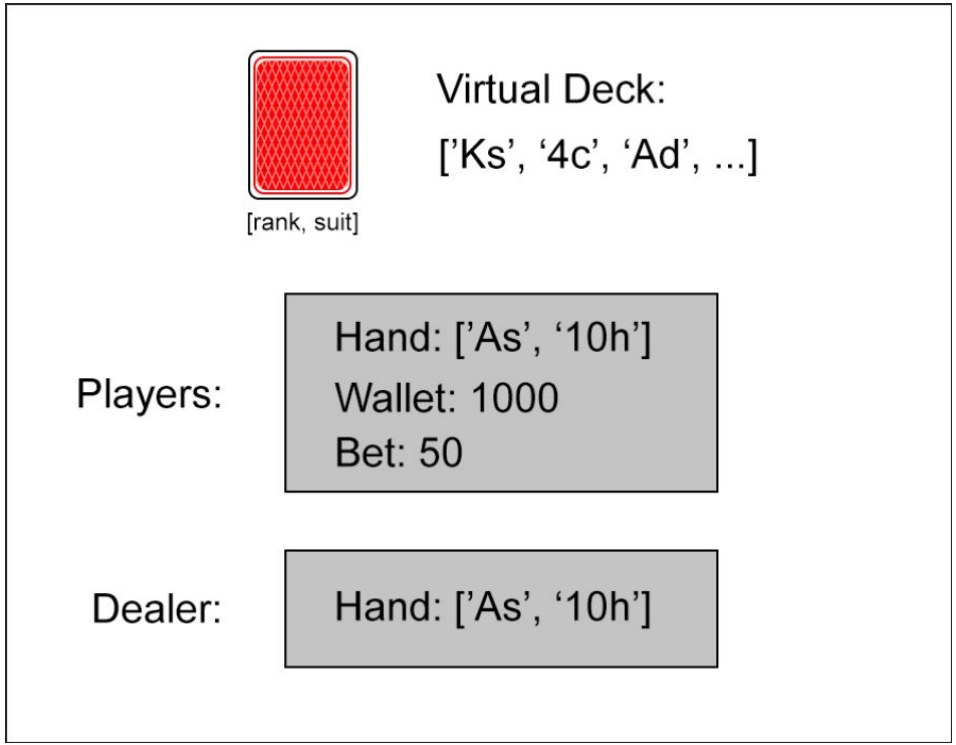
Game Algorithm

- Algorithm follows a standard Blackjack game
- Main Game Loop
  - Build deck, deal cards
  - Loop through player turns
    - Bet, Hit/Stand
  - Dealer's turn
  - Settle scores and bets
    - Pay players out 2 to 1
  - Ask to play again

```
gs = gameState(1) #initialize gamestate
while(1):   #main game loop
    totals = [] #list of hand totals
    gs.setPlayers((int)(input("How many people are playing?\n")))
    gs.resetHands()
    gs.deck.build()
    gs.deck.shuffle()
    gs.dealCards(2) #arg = num of cards

    totals.append(0) #temp dealer score; needs to be calculated after players
    for x in range(1, gs.numPlay): #Player turns
        totals.append(playerTurn(gs.players[x], gs.deck))
    totals[0] = dealerTurn(gs.players[0], gs.deck) #dealer turn
    score(gs.players, totals)

    gs.showWallets() #debugging purposes

    play_again = input("Play again (y/n)?\n").lower()
    if play_again == "y":
        continue
    elif play_again == "n":
        quit() #temporary, should go back to menu screen
    else:
        print("\n Invalid answer, quitting.")
        quit() #same as above
```

Local Game State

- Gamestate is kept to keep track of the cards/money
- Cards are taken from virtual deck and given to players

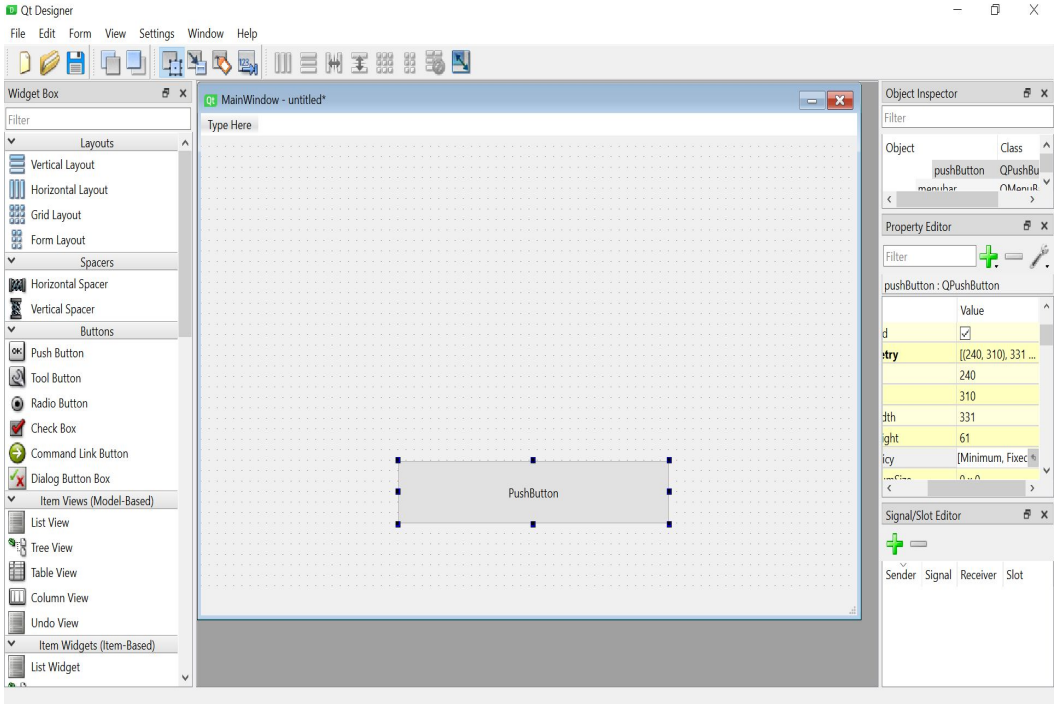


Proposed MDR Deliverables (Checklist)

- Functioning card dispenser integrated with rotation mechanism able to dispense cards
- Functioning Shuffler
  - Components of Dealer should be controlled by input
- Card identification able to get specific card number and suit
- Game algorithm for Blackjack
- GUI prototype for touch display

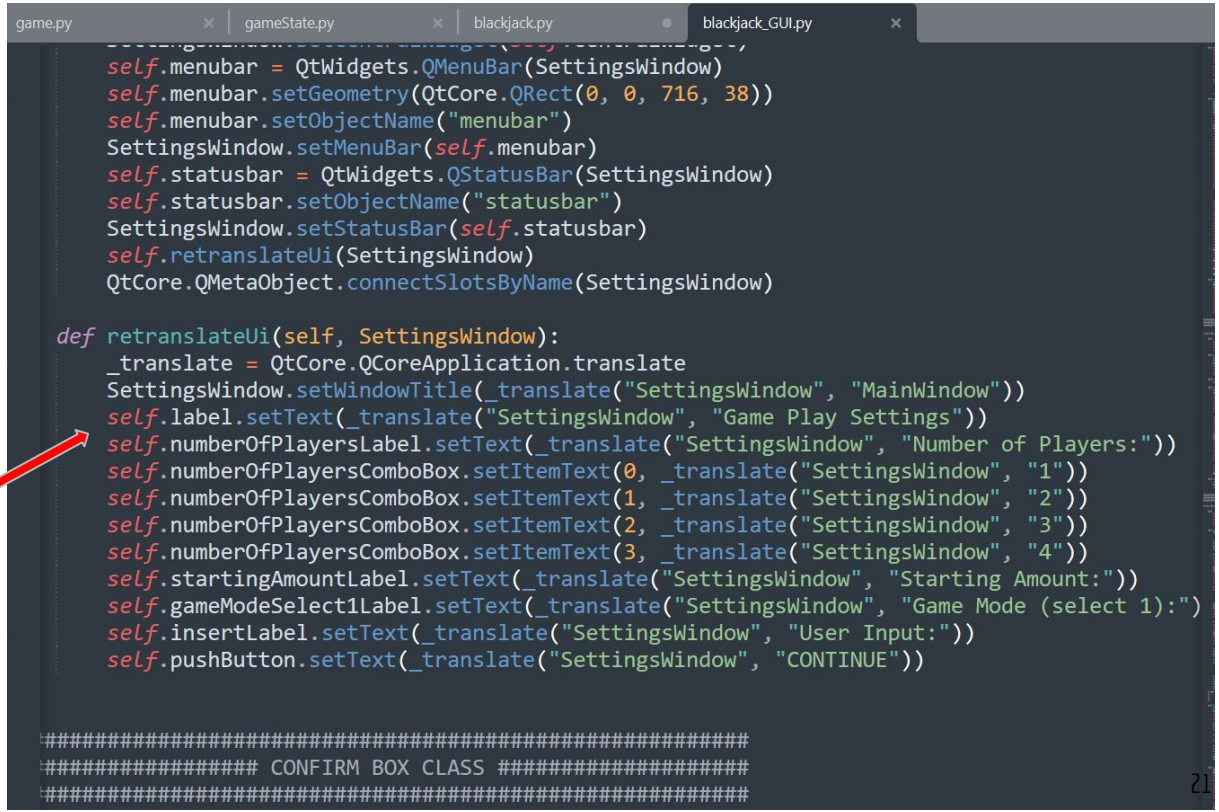
Graphical User Interface (GUI)

- PyQt/Qt Designer
  - Widgets
  - Buttons/Labels
  - Windows
  - Connections
- Pyuic
  - Convert UI forms to .py



PyQt Code [pt.1]

- Qt Designer allows for easy initial designing purposes
- Pyuic allows conversion for easier coding as well
- Once converted, changes made to .py does not affect .ui



PyQt Code [pt. 2]

- Additional coding needed for more detailed information/functions
- Ex: setting values, “clicked” functionality, etc.
- Each window is separate
  - Needs to be coded to function together

```
self.formLayout.setWidget(1, QtWidgets.QFormLayout.LabelRole, self.startingAmountLabel)
self.startingAmountSpinBox = QtWidgets.QSpinBox(self.formLayoutWidget,
    maximum=5000,
    minimum=0,
    value=1000,
    singleStep=100)
self.startingAmountSpinBox.setObjectName("startingAmountSpinBox")
self.formLayout.setWidget(1, QtWidgets.QFormLayout.FieldRole, self.startingAmountSpinBox
self.gameModeSelect1Label = QtWidgets.QLabel(self.formLayoutWidget)
self.gameModeSelect1Label.setObjectName("gameModeSelect1Label")
self.formLayout.setWidget(2, QtWidgets.QFormLayout.LabelRole, self.gameModeSelect1Label)
self.gameModeSelect1ComboBox = QtWidgets.QComboBox(self.formLayoutWidget)
self.gameModeSelect1ComboBox.addItems(["Winning Amount", "Number of Wins", "Total Games"
self.gameModeSelect1ComboBox.setObjectName("gameModeSelect1ComboBox")
self.formLayout.setWidget(2, QtWidgets.QFormLayout.FieldRole, self.gameModeSelect1ComboB
self.insertLabel = QtWidgets.QLabel(self.formLayoutWidget)
self.insertLabel.setObjectName("insertLabel")
self.formLayout.setWidget(3, QtWidgets.QFormLayout.LabelRole, self.insertLabel)
self.insert = QtWidgets.QLineEdit(self.formLayoutWidget)
self.insert.setObjectName("insert")
self.formLayout.setWidget(3, QtWidgets.QFormLayout.FieldRole, self.insert)
self.pushButton = QtWidgets.QPushButton(self.formLayoutWidget, clicked=lambda: self.open
self.pushButton.setObjectName("pushButton")
self.formLayout.setWidget(4, QtWidgets.QFormLayout.SpanningRole, self.pushButton)
SettingsWindow.setCentralWidget(self.centralwidget)
self.menubar = QtWidgets.QMenuBar(SettingsWindow)
self.menubar.setGeometry(QtCore.QRect(0, 0, 716, 38))
```

Demo

PCB Design

- Microcontroller: Arduino
- Drivers: Motor Driver
- Power: view slide on power

Hardware Components/Justification

- 4 Raspberry Pi’s
- 4 Touch Displays
- 2 Sparkfun RedBoards
- Adafruit PN532 (RFID Scanning)
- Stepper motor
- Motor driver (currently using L298N dual bridge DC stepper controller board, would like to switch to a ULN2003 Driver)

Software Components/Justification

- GUI:
  - PyQt/Qt Designer
  - Python
- Gaming Algorithm:
  - Python
- RFID code:
  - Arduino → C/C++
- Use a professional IDE for dealing system as opposed to Arduino

Cost Estimates

Item	Quantity	Cost/item	Total
Table	1	$50	$50
RFID Cards	1	$118	$118
RFID Scanners	4	$40	$120
Card Shuffling	1	$11	$11
Card Dealing	1	$25	$25
RaspberryPi	5	35	175
PCB	1	$100	$100
Total			479

Proposed CDR Deliverables

1. Completely functional GUI embedded with gaming algorithm and RFIDs.
    - Completely functional GUI defined as having all necessity features included, as well as the game play and RFIDs working together.
2. System will be powered through a wall outlet using AC-DC converter
3. An integrated table with motor and dealing system.
    - Physical table will be constructed, combining the motor and dealing system.
4. System able to dispense cards to each player with reasonable timing.

Technical Responsibilities

Ray

- Budget Lead
- RFID Systems
- Assembly of table

Lamisa

- Team Coordinator
- 3D Printing Designs
- Dealing unit

Marvin

- PCB Lead
- Event Handlings
- Integration/Communication

Alina

- GUI Interface
  - Improve GUI visuals
  - Additional GUI features
- Project Website

Gantt Chart

Overarching category	Task	Team Member	Dec 5 - Dec 11	Dec 12 - Dec 18	Dec 19 - Jan 24	Jan 25 - Jan 29	Jan 30 - Feb 5	Feb 6 - Feb 12	Feb 13 - Feb 19	Feb 20 - Feb 26	Feb 27 - Mar 5	Mar 6 - Mar 12
Integration/ Communication	Complete Integration Game & GUI	**Marvin**	X	X	X							
Communication	GUI working on Multiple Touch Displays	**Marvin/Alina**			X	X	X	X				
Software	Convert code: Python to C/C++	**Marvin**				X	X	X				
Display/Touch Interface	Additional ft. to GUI	**Alina**	X	X	X	X						
Software	Project Website	**Alina/Ray**			X		X	X				
Construction	Build Table	**Ray/Lamisa**					X	X				
Design	3D Print Platform for Stepper Motor	**Lamisa**				X	X					
Hardware	Add Multiple Shuffles to System	**Ray/Lamisa**					X					
Hardware Assembly	5 RFID Systems	**Ray**				X	X					
Design/Hardware	PCB Design	**ALL**	X	X	X	X						
Integration/ Assembly	Integrate Hardware and Software	**ALL**							X	X	X	X

Questions?

THE END!