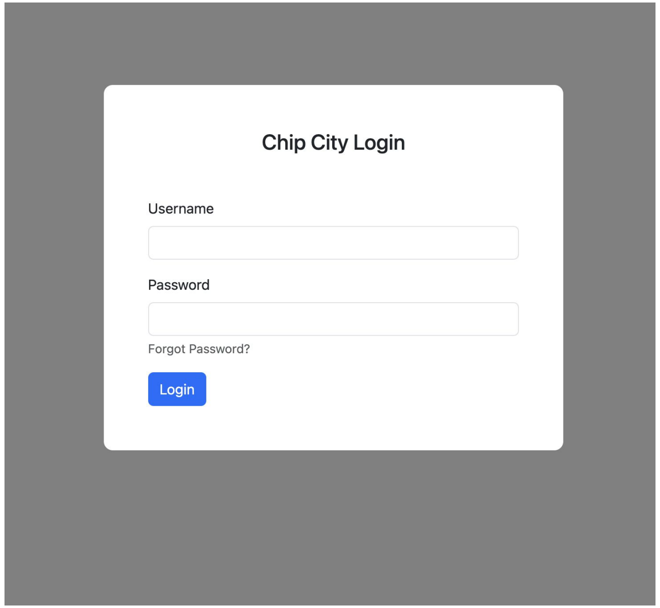
ChipCity: Poker Game implemented using WebSockets

GitHub: https://github.com/cmu-webapps/s24_team_39

Tedd Jung, Aditri Gupta, Eddie Zhang, Lucy Wang

Overview of Project

UI Overview

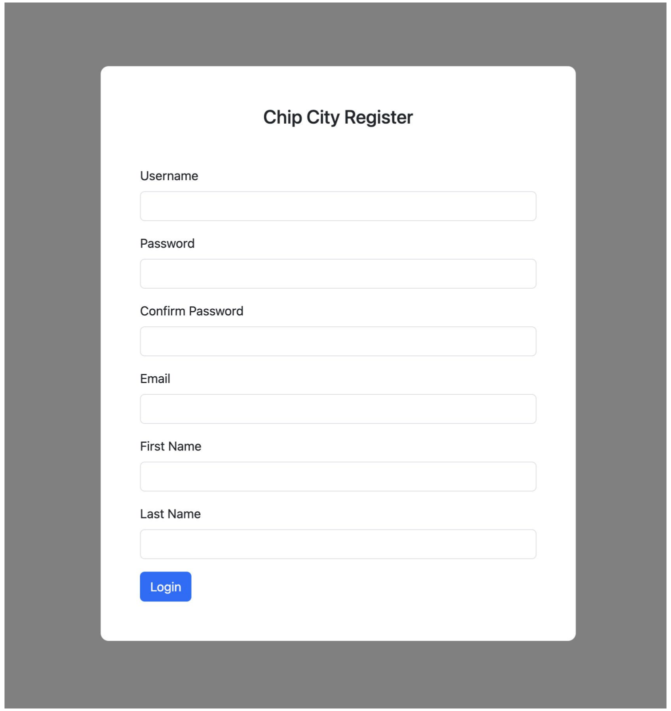

Chip City Login

Username

Password

Forgot Password?

Login


Chip City Register

Username

Password

Confirm Password

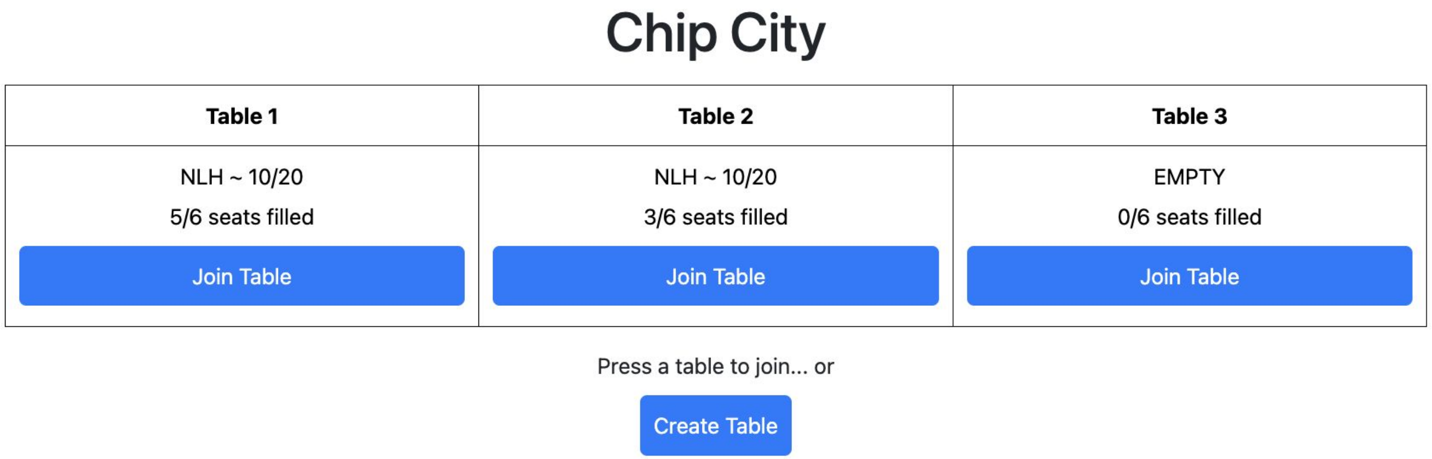
Email

First Name

Last Name

Login

UI Overview (continued)



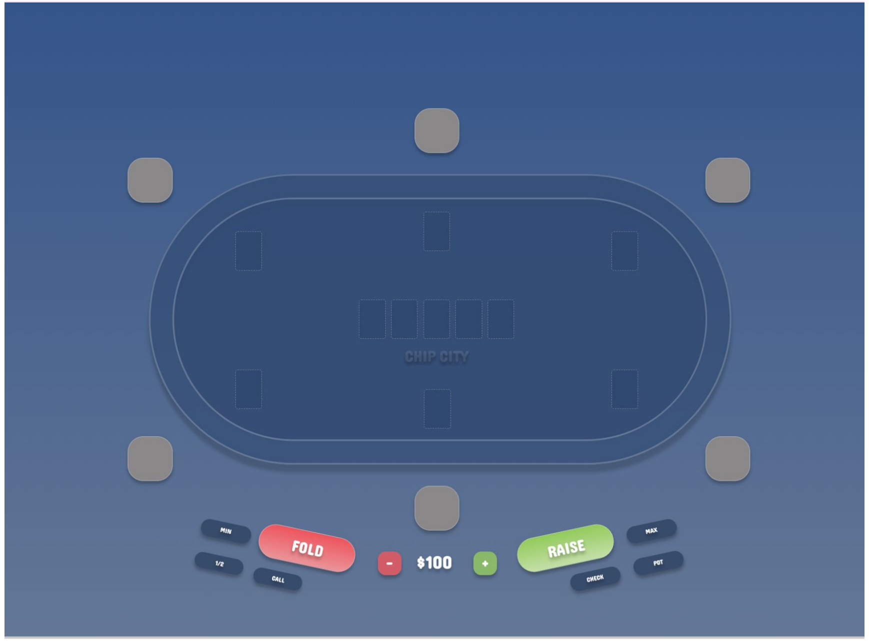
Chip City

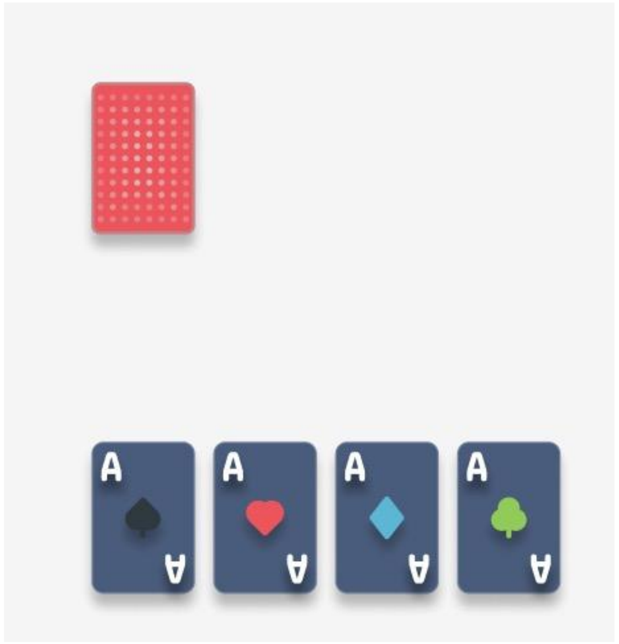
Table 1	Table 2	Table 3
NLH ~ 10/20	NLH ~ 10/20	EMPTY
5/6 seats filled	3/6 seats filled	0/6 seats filled
Join Table	Join Table	Join Table

Press a table to join... or

Create Table

UI Overview (continued)





Original Goals vs. What We've Completed

- Finish UI Design
- Complete models.py
- Implement OAuth
- Implement one person poker game
    - Create new game each time create table is clicked
    - Working game table UI

- Finish UI Design
- Completed models.py
- Implemented OAuth
- Created new game object each time create table is clicked

New Goal

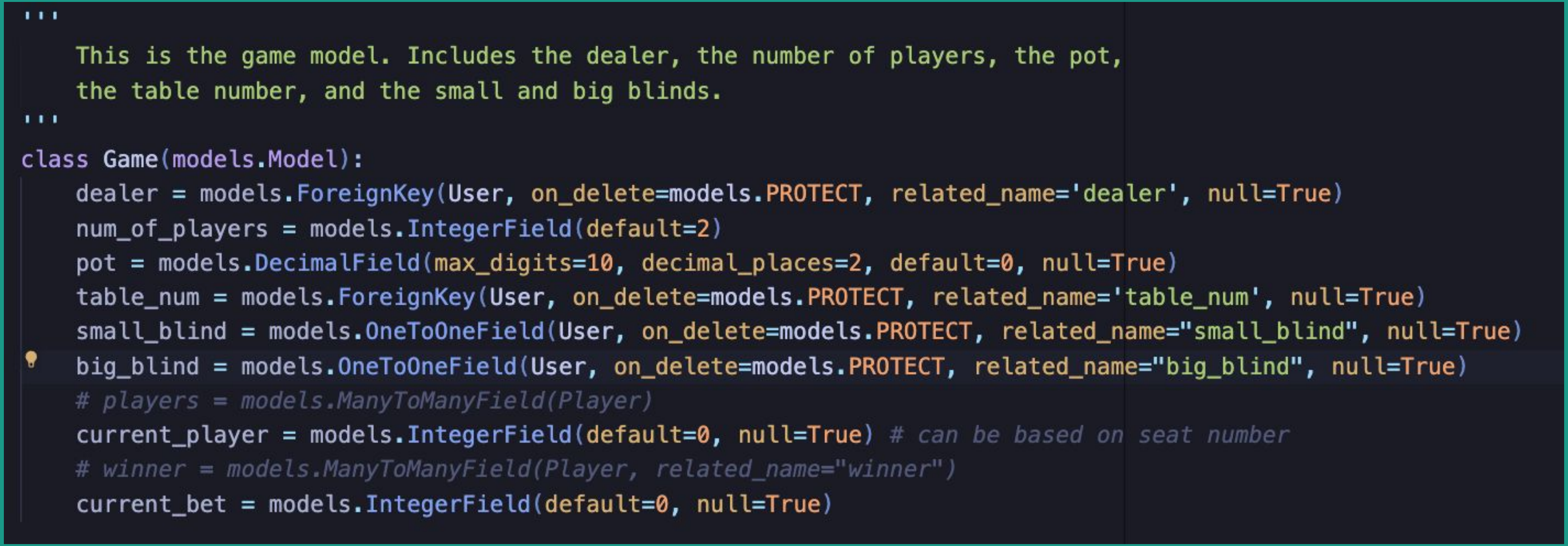
- Working Demo by Sprint 2
  - Implement the turn based mechanics
  - Implement a rudimentary version of betting/calculating card hands

Problems

- Not sure how to implement the game, especially tying everything together

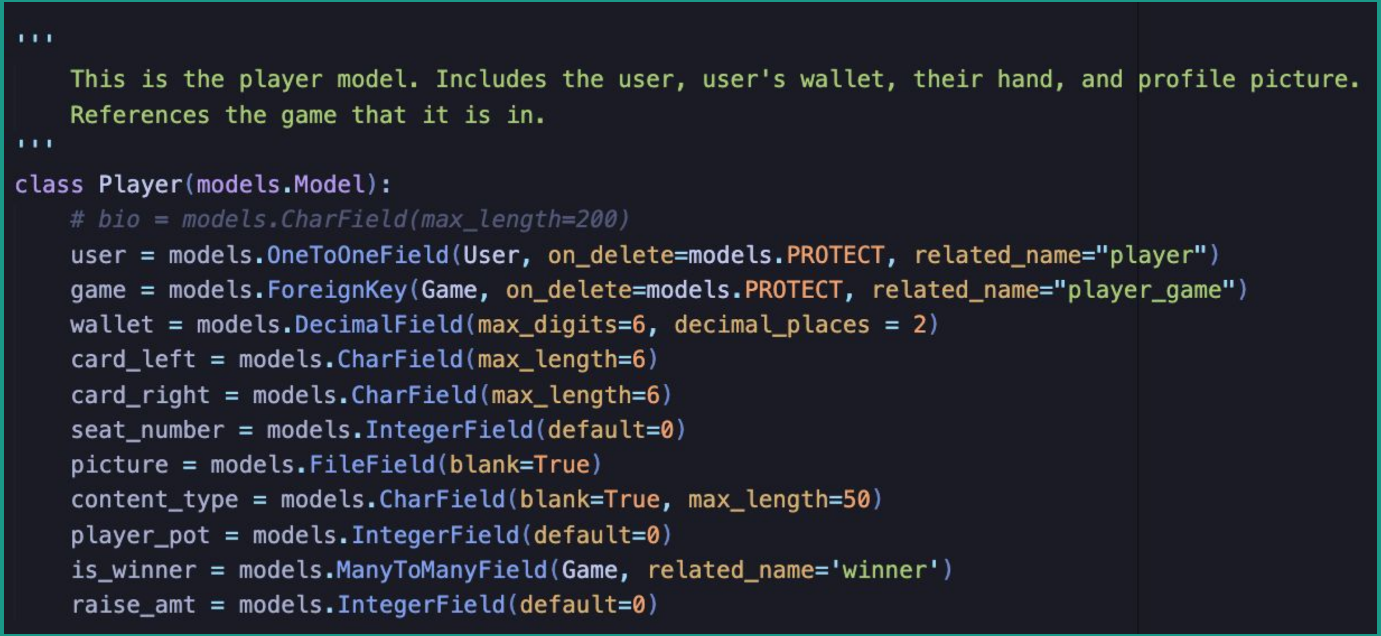
Models

Game Model



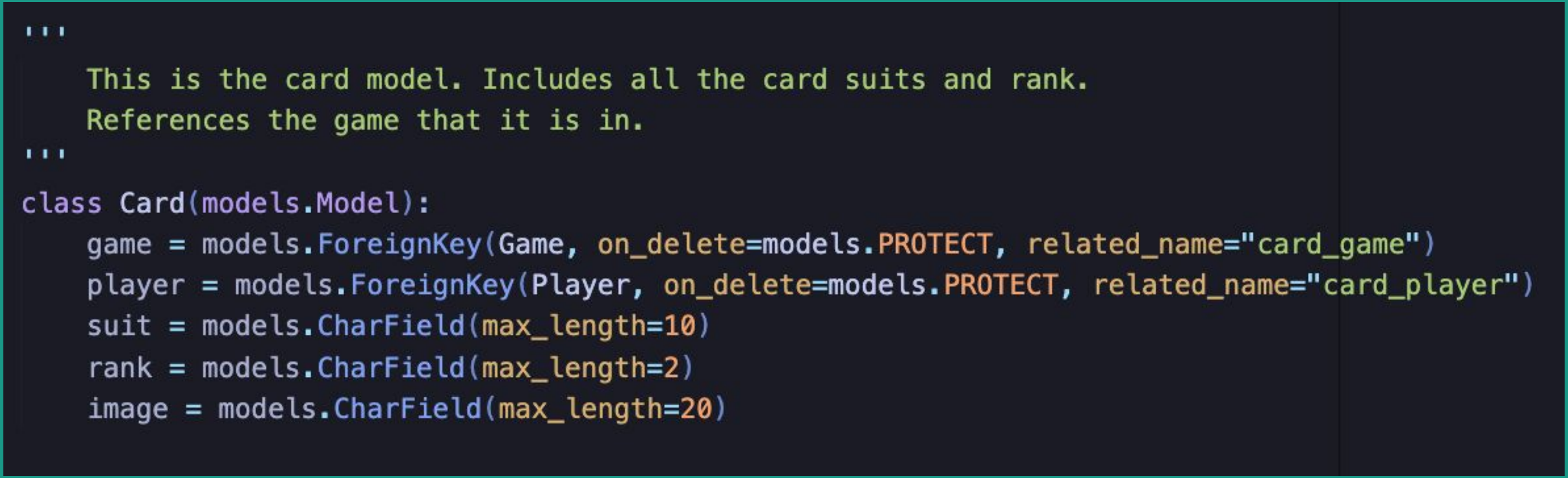
```
'''
    This is the game model. Includes the dealer, the number of players, the pot,
    the table number, and the small and big blinds.
'''
class Game(models.Model):
    dealer = models.ForeignKey(User, on_delete=models.PROTECT, related_name='dealer', null=True)
    num_of_players = models.IntegerField(default=2)
    pot = models.DecimalField(max_digits=10, decimal_places=2, default=0, null=True)
    table_num = models.ForeignKey(User, on_delete=models.PROTECT, related_name='table_num', null=True)
    small_blind = models.OneToOneField(User, on_delete=models.PROTECT, related_name="small_blind", null=True)
    big_blind = models.OneToOneField(User, on_delete=models.PROTECT, related_name="big_blind", null=True)
    # players = models.ManyToManyField(Player)
    current_player = models.IntegerField(default=0, null=True) # can be based on seat number
    # winner = models.ManyToManyField(Player, related_name="winner")
    current_bet = models.IntegerField(default=0, null=True)
```

Player Model



```python
'''
    This is the player model. Includes the user, user's wallet, their hand, and profile picture.
    References the game that it is in.
'''
class Player(models.Model):
    # bio = models.CharField(max_length=200)
    user = models.OneToOneField(User, on_delete=models.PROTECT, related_name="player")
    game = models.ForeignKey(Game, on_delete=models.PROTECT, related_name="player_game")
    wallet = models.DecimalField(max_digits=6, decimal_places = 2)
    card_left = models.CharField(max_length=6)
    card_right = models.CharField(max_length=6)
    seat_number = models.IntegerField(default=0)
    picture = models.FileField(blank=True)
    content_type = models.CharField(blank=True, max_length=50)
    player_pot = models.IntegerField(default=0)
    is_winner = models.ManyToManyField(Game, related_name='winner')
    raise_amt = models.IntegerField(default=0)
```

Card Model



```
'''
    This is the card model. Includes all the card suits and rank.
    References the game that it is in.
'''
class Card(models.Model):
    game = models.ForeignKey(Game, on_delete=models.PROTECT, related_name="card_game")
    player = models.ForeignKey(Player, on_delete=models.PROTECT, related_name="card_player")
    suit = models.CharField(max_length=10)
    rank = models.CharField(max_length=2)
    image = models.CharField(max_length=20)
```

Demo