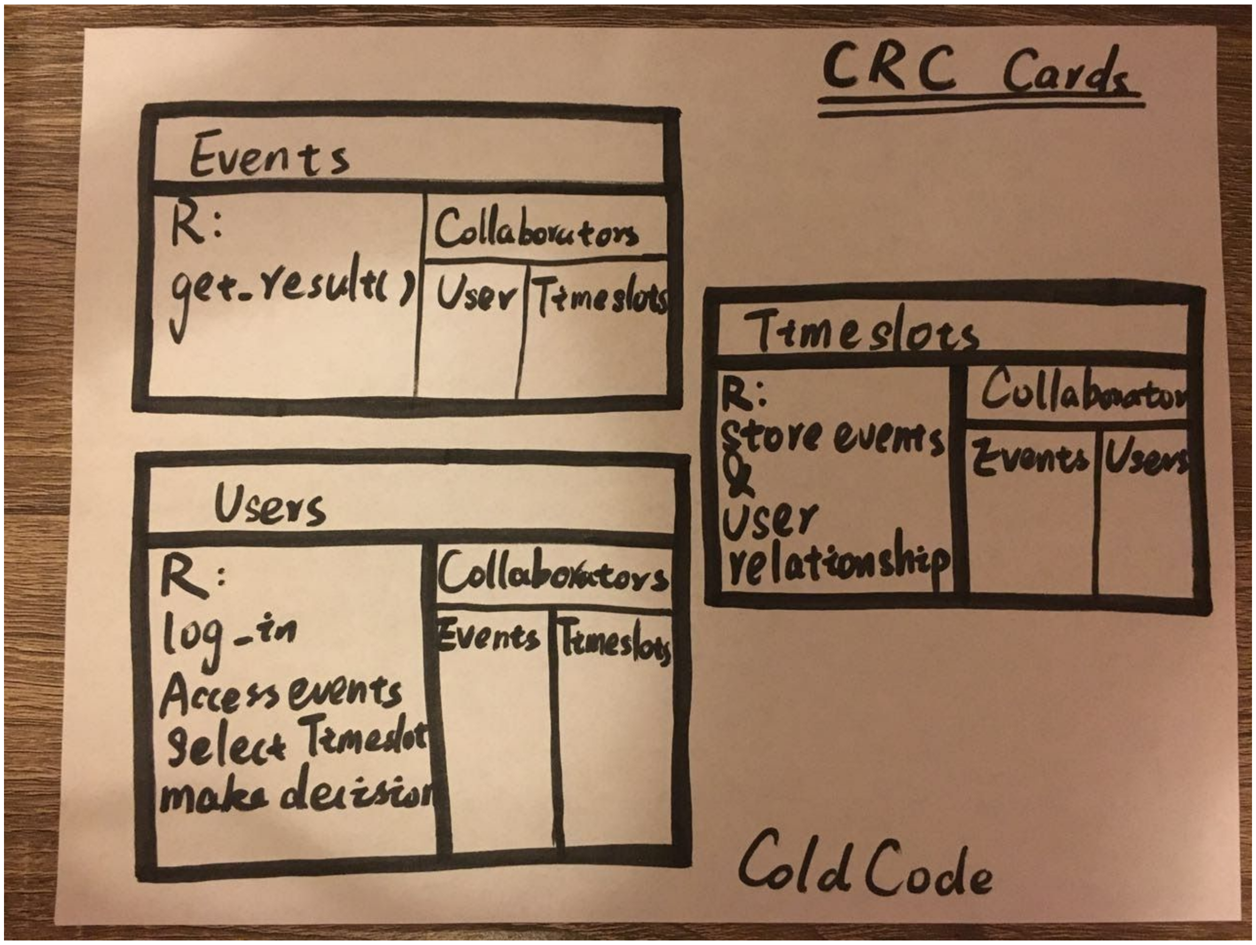
Design: Team Assignment 2



CRC Cards & User Stories:

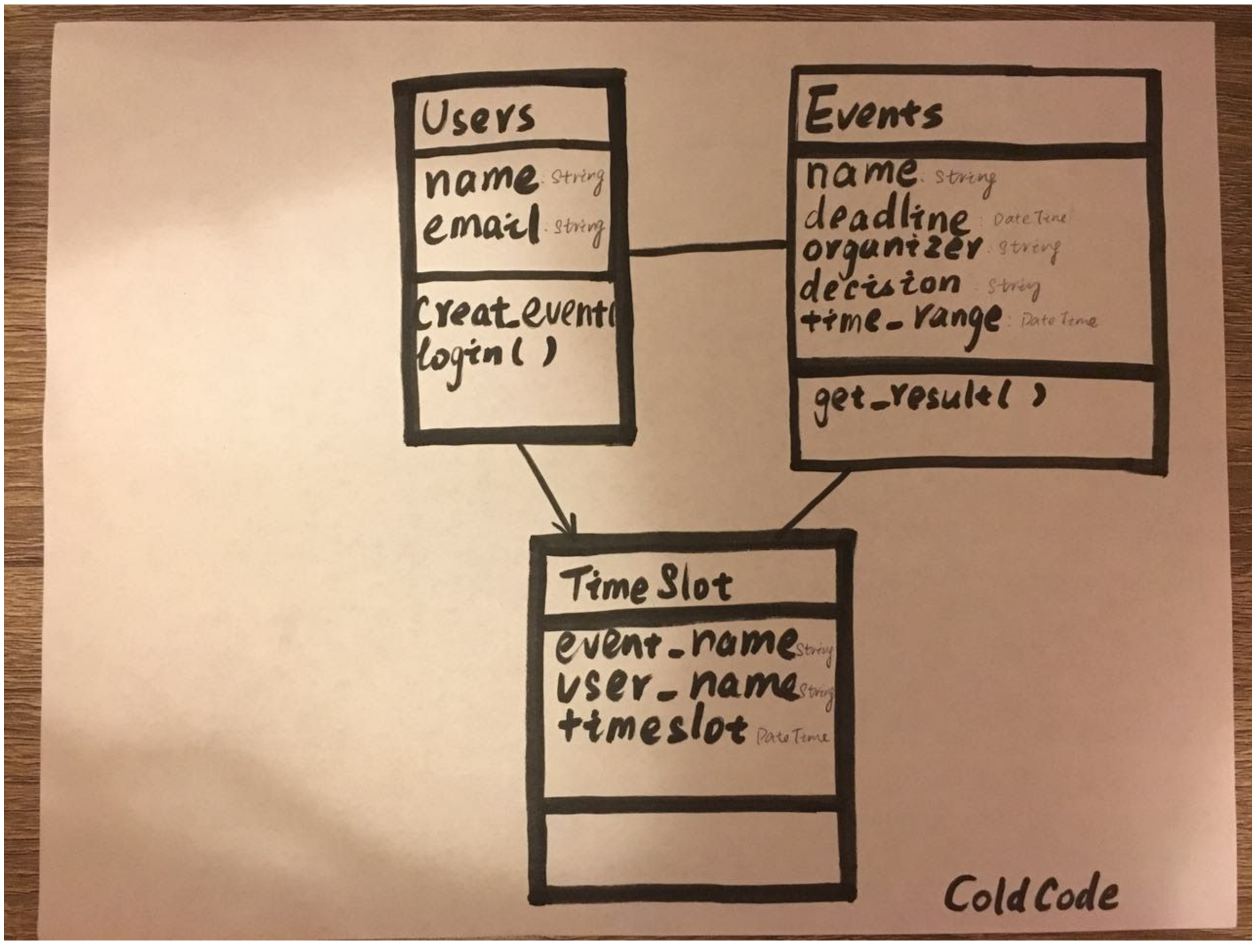
User Card:
- “As a meeting organizer, I want to create a meeting within a time range."
- “As a common user, I want to keep track of the meeting which I am involved.”

Timeslot Card:
- “As a common user, I want to choose my available time slots for a meeting I am invited.”

Event Card:
- “As a meeting organizer, I want to get the result of available time slots for every member, and make the final decision accordingly.”

Class diagrams:

According to the above CRC cards, we create these diagrams. There are mainly 3 associations among these classes. Each user can attend several events, users can input multiple timeslots for each event and each events can have several timeslots.



Here are the code we used for each class:

Models

There are in total three classes in the model file, which including the Events class; User class and Timeslot Class. Each class has its own attributes and according to the needs.

```
Class Events (model.Models):
        Event_name = model.CharField(30)
        Organizer = model.CharField(30)
        Finish = model.BooleanField()
        Time_range = model.DateTimeField()
        Deadline = model.DateTimeField()
        Final_decision = model.booleanFiled()
        Final_decision_timeslot = model.DateTimeField()

        Def __str__():

        Def isFinished():
                # modify the value of Finish according to the deadline and number of
        timeslots

        Def generate_url():
                Return url

        # def __init__(self.event_name, organizer):
        #       Self.event_name = event_name
        #       Self.organizer = organizer

Class Users (model.Models):
        # how to check user's login state?
        user_name = model.CharField(30)
        user_email = model.CharField(30)
        def.__str__():

Class Timeslot (model.Models):
        user_name = ForeignKey(Users)
        event_email = ForeignKey(Events)
        Timeslots = model.datatimeField()
        def.__str__():

Views


Def login(request):
        Return render(result, ……, context = next)

Def creat_event(request, event_name, user_name):
        event = Events()
        event. Event_name = …
        Event. Organizer = …
        //don’t forget to save the model object whenever you modify the object
        Return HttpResponse(url = …)

Def delete_event(request, event_name, user_name):

class detail_view(generic.Detailview):
        Model = events
        Def get_request(request):
                Return Events.objects.filter()

Def select_timeslots(request, user_name, event_name):
        # add new timeslot to event with specified user_name

        Return HttpResponse(url = ‘finished’)

Def make_decision(request, user_name, event_name):
        Result = get_result(request, event_name)
        If request == “abort”:
                Return (url=..., content = …)
        If request == “decide”
                # update Final_decision
        Return HttpResponse(url = , …)

Def get_result(request, event_name)
        # get timeslot and comput
        Return result
```

Work Division:

Pair One: Zhidong Liu & Yi Zhang
Events class and Users class

Pair Two: Yufeng Zhang & Qimeng Han
Timeslot class and UI impletation