6.104 · Software Design · MIT · Fall 2025

designing behavior

Daniel Jackson

your goals for today's class

know how to model behavior
states (aka data model) and actions
a key computer science skill!

understand trace view
behavior as history of actions

know how to use invariants in design
integrity constrains that define good states

on details

The details are not details. They make the design. *Charles Eames*

Charles Eames Ray Eames
13721
Eames Contract Storage
1507 Folding bed unit; light shelf, bed platform, reading light. Accessories: mattress.
HERMAN MILLER INC.
ES 103

what kind of behavioral details?

	for online bookstore, eg
details to include	
steps the user takes	buy a book
system responses to the user	book gets delivered
data the user gives & gets	address, arrival estimate
details to exclude	
coding & algorithmic details	order id has checksum
distribution, replication, etc	orders on separate server
internal steps	request to warehouse

also UI independent
layout & styling of pages
navigation between pages
“micro-steps”

UI-dependent questions: important but not conceptual

Terra - Eataly Boston

4.5 (3940) • $31 to $50 • Contemporary Italian

Overview Experiences Popular dishes Photos

About this restaurant

Charming Lively Good for special occasions

Located on the third floor of Eataly Boston, Terra is a unique restaurant inspired by earth and fire. The dining room centers around a wood-burning Italian grill, where the Terra culinary team cooks raw ingredients over burning flames, allowing the...

Read more

Experiences

Brunch at Terra

Aug 22, 2024 - Jan 28, 2026

Every Saturday and Sunday from 11AM-4PM, indulge in our brunch menu featuring all your favorites...with an Italian...

Make a reservation

2 people

Jun 13, 2025 7:00 PM

Select a time

6:00 PM* 6:15 PM* 8:00 PM* 9:00 PM*

+1,000 pts +1,000 pts

Notify me

Booked 110 times today

You're in luck! We still have 4 timeslots left

Experiences are available. See details

Additional seating options

why postpone UI-dependent details?

they're a lot of work
we need to tend to
more basic things first

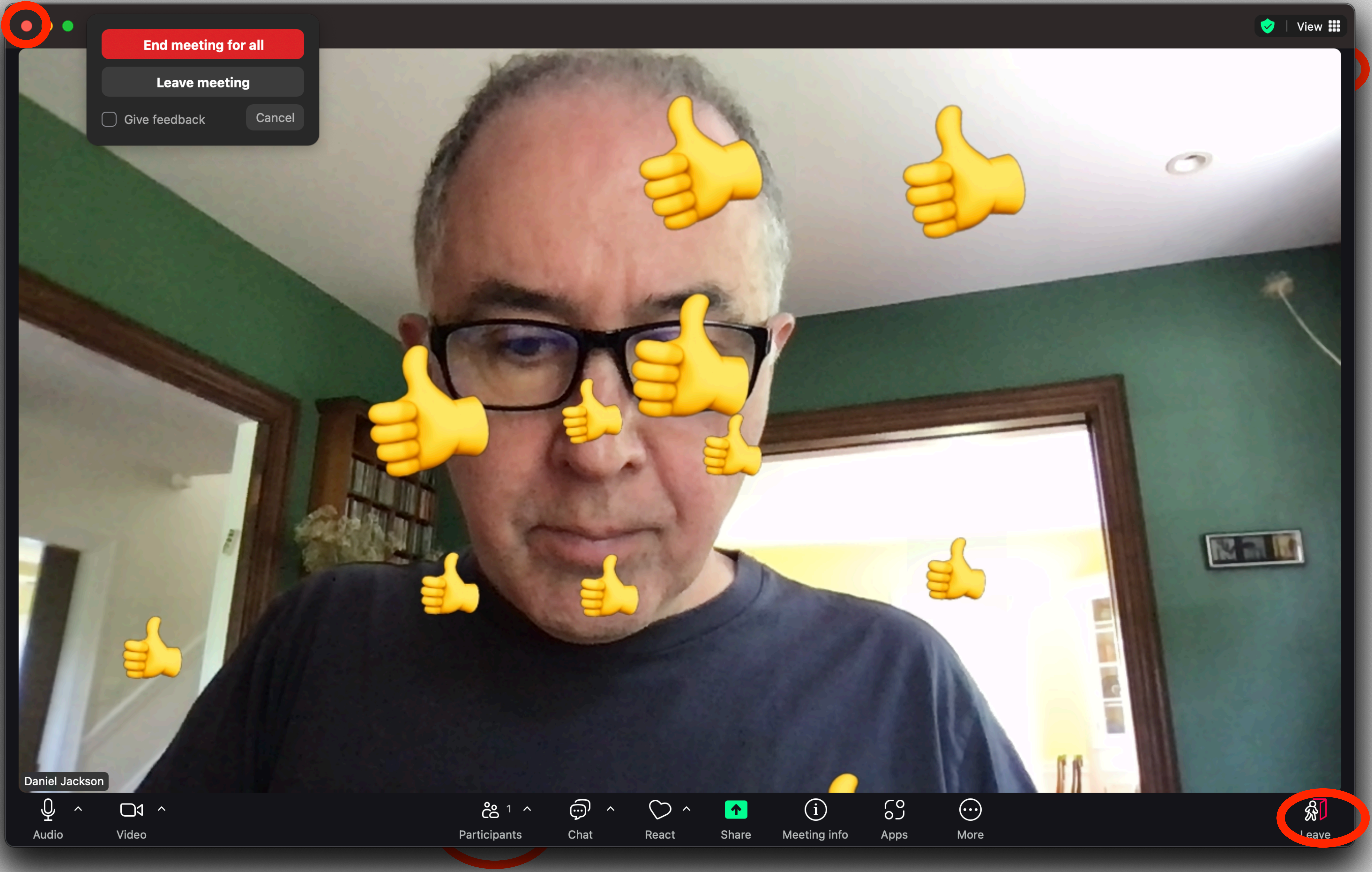
they can be a distraction
color of slots before we've
decided that we have slots?

want to judge a UI
projects concepts well?
then need pure concepts

shared understanding
between UX & engineering
capturing the overlap

what this doesn't mean
can't sketch UI ideas
during concept design
often helpful to concretize

which steps are concept actions?



a full example

a reservation concept

how to design a concept

pick a name
specific to function
but for general use

describe purpose
why design or use it?
value to stakeholders

tell story
a simple scenario
of how it's used

list actions
by user or system
key steps, not UI

specify state
what's remembered
enough for actions

picking a name

pick a name
specific to function
but general enough

Restaurant

RestaurantReservation ✓

OpenTableReservation

Reservation

describing a purpose

describe purpose
why design or use it?
value to stakeholders

reducing wait time for tables ✓

maximizing use of available tables

making money for reservation service

tracking occupancy patterns

telling the story

the restaurant makes slots available at various times; a diner reserves for a particular time, and then can be assured of being seated at that time

✓

tell story
a simple scenario of how it's used

listing actions

list actions
by user or system
key steps, not UI

select date
select time
click reserve

no! these are
all low-level
UI interactions

login
search for restaurant
review restaurant

no! these belong
to other concepts

let's return to our
story for hints:

the restaurant makes slots available at various times; a diner reserves for a particular slot, and then can be assured of being seated at that time

createSlot
reserve
seat

✓

what other actions
might be needed?

cancel

noShow

deleteSlot

defining action arguments

createSlot	createSlot (t: Time)
reserve	reserve (u: User, t: Time): Reservation
seat	seat (r: Reservation)
cancel	cancel (r: Reservation)
noShow	noShow (r: Reservation)
deleteSlot	deleteSlot (s: Slot)

devising the state

specify state
what's remembered
enough for actions

```
a set of Slots with
  a Time
a set of Reservations with
  a User
  a Slot
```

defining the actions

state

a set of Slots with
  a Time
a set of Reservations with
  a User
  a Slot
  a seated Flag

actions

createSlot (t: Time)
effect
  creates a fresh slot
  associates it with time t

reserve (u: User, t: Time): Reservation
requires
  some slot at time t not yet reserved
effect
  creates & returns a fresh reservation
  associates it with user u and the slot

"precondition"
what's true of state before

"postcondition"
relates state after to before

seat (r: Reservation)
requires
  r is a reservation for about now
effect
mark r as seated

state
a set of Slots with
  a Time
a set of Reservations with
  a User
  a Slot
  a seated Flag

actions

createSlot (t: Time)
effect creates a fresh slot
  associates it with time t

reserve (u: User, t: Time): Reservation
requires some slot at time t not yet reserved
effect creates & returns a fresh reservation
  associates it with user u and the slot

seat (r: Reservation)
requires r is a reservation for about now
effect mark r as seated

initially

slot	time

res	user	slot	seated

createSlot (July 4, 2025 at 7pm)

slot	time
s0	July 4, 2025 at 7:00pm

res	user	slot	seated

reserve (u1, July 4… 7pm): r0

slot	time
s0	July 4, 2025 at 7:00pm

res	user	slot	seated
r0	u1	s0	FALSE

seat (r0)

slot	time
s0	July 4, 2025 at 7:00pm

res	user	slot	seated
r0	u1	s0	TRUE

putting it all together

pick a name
specific to function
but for general use

describe purpose
why design or use it?
value to stakeholders

tell story
a simple scenario
of how it's used
including setup

list actions
by user or system
key steps, not UI

specify state
what's remembered
enough for actions

concept RestaurantReservation

purpose reducing wait time for tables

principle the restaurant makes slots available at various times; a diner reserves for a particular time, and then can be assured of being seated at that time

state
a set of Slots with
  a Time
a set of Reservations with
  a User
  a Slot
  a seated Flag

actions
 createSlot (t: Time)
  effect creates a fresh slot & associates with time t
 reserve (u: User, t: Time): Reservation
  requires some slot at time t not yet reserved
  effect creates & returns a fresh reservation
   associates it with user u and the slot
 seat (r: Reservation)
  requires r is a reservation for about now
  effect mark r as seated

your turn:
state & actions

```
concept RestaurantReservation [User]

state
a set of Slots with
  a Time
a set of Reservations with
  a User
  a Slot

actions
  createSlot (t: Time)
  reserve (u: User, t: Time): Reservation
```

extend the concept in these ways
add an action for canceling
add an action for deleting a slot
support multiple restaurants
include party size

https://yellkey.com/movement

a possible solution

concept RestaurantReservation [User, Restaurant]

state
a set of Slots with
  a Restaurant
  a Time
  a max Number
  a min Number
a set of Reservations with
  a User
  a Slot
  a partySize Number

actions
  createSlot (r: Restaurant, t: Time, max, min: Number)
  deleteSlot (s: Slot)
    requires no reservations for this slot
    effects remove s from set of slots
  reserve (u: User, t: Time, party: Number, r: Restaurant): Reservation
    requires some unreserved slot for r at t with min <= party <= max
    effects add new reservation for slot with user and party size
  cancel (r: Reservation)
    requires r is an existing reservation
    effects remove r from reservations

traces

action histories

a trace of the reservation system

createSlot (July 4, 2025 at 7pm)

register (“Daniel”, “foo”): u1

login (“Daniel”, “foo”): u1

reserve (u1, July 4... 7pm): r0

notify (u1, “reserved”)

cancel (r0)

notify (u1, “canceled”)

register (“Mitchell”, “bar”): u2

login (“Mitchell”, “bar”): u2

reserve (u2, July 4... 7pm): r1

seat (r1)

projecting system trace into concept traces

concept SessionAuth
concept Notification
concept RestaurantReservation
createSlot (July 4, 2025 at 7pm)
register (“Daniel”, “foo”): u1
setEmail (u1, “dnj@mit.edu")
login (“Daniel”, “foo”): u1
current user is u1
notify (u1, “reserved”)
reserve (u1, July 4... 7pm): r0
notify (u1, “canceled”)
cancel (r0)
register (“Mitchell”, “bar”): u2
login (“Mitchell”, “bar”): u2
current user is u1
reserve (u2, July 4... 7pm): r1
seat (r1)

state invariants

aka integrity constraints

designing invariants for concepts

concept PasswordSession

state
a set of Users with
  a username String
  a password String

concept RestaurantReservation

state
a set of Slots with
  a Time
a set of Reservations with
  a User
  a Slot
  a Restaurant

invariants?

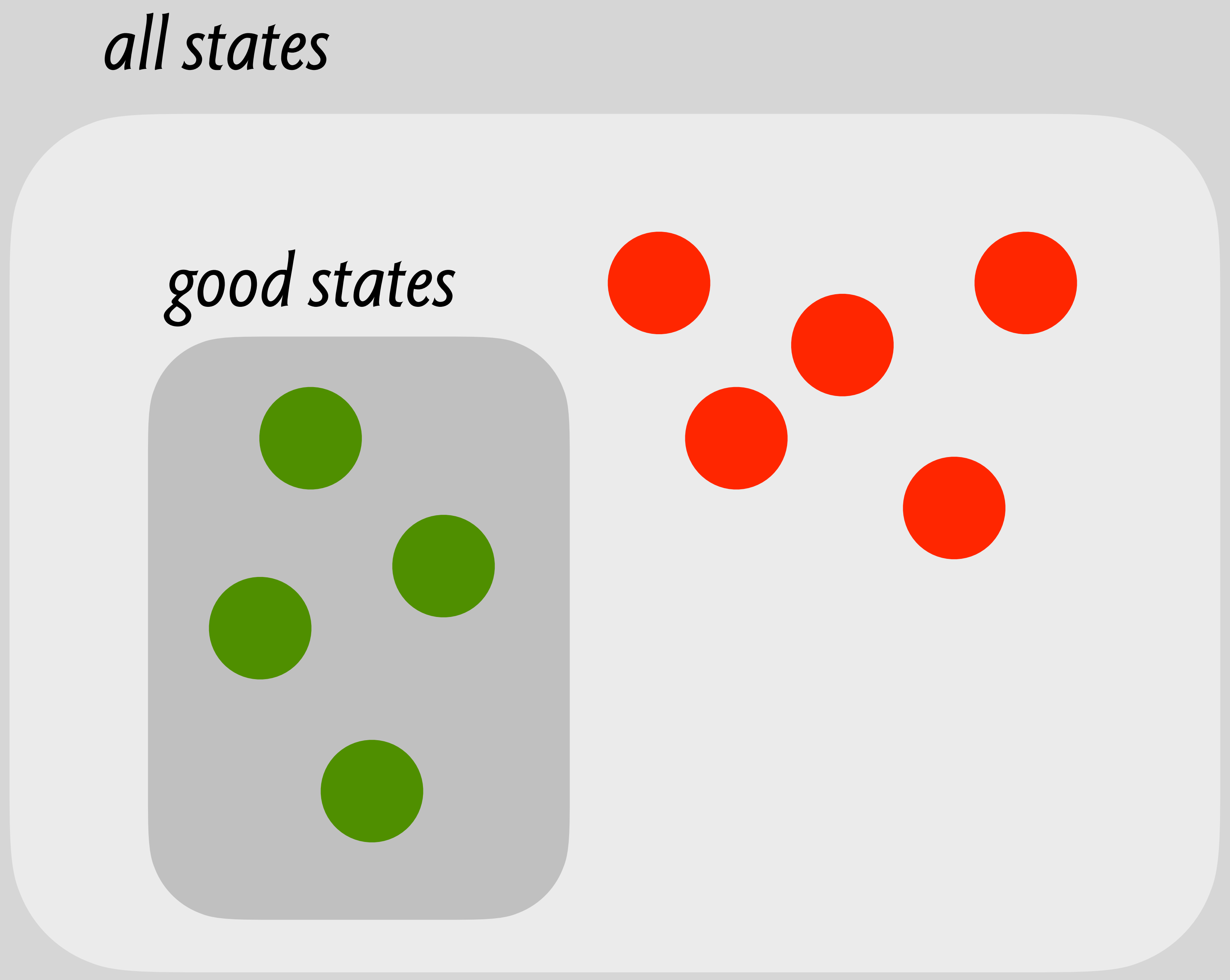
at most one user with a given username

at most one reservation for a given slot

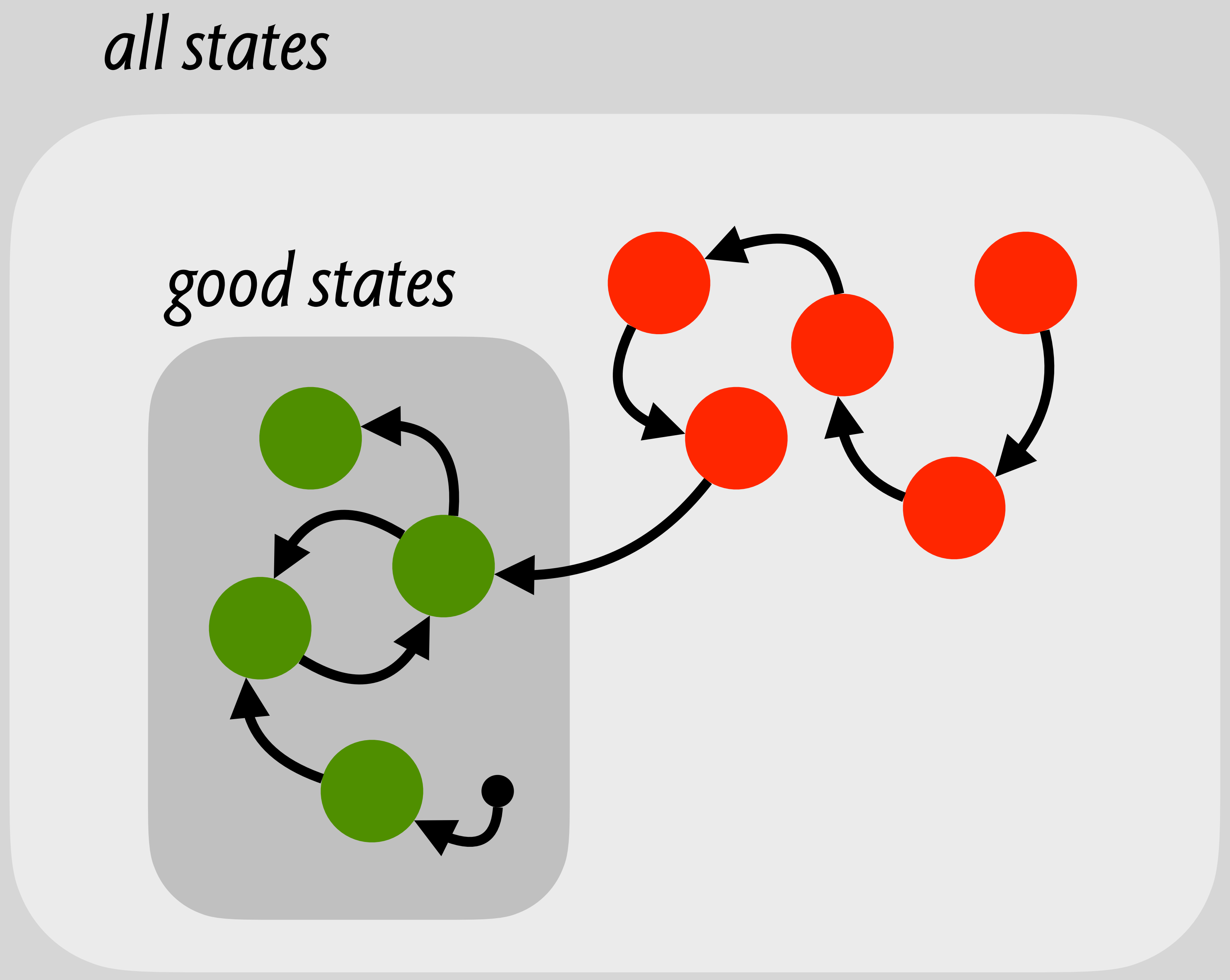
? at most one reservation for a given user at a given time?

what goes wrong if violated?

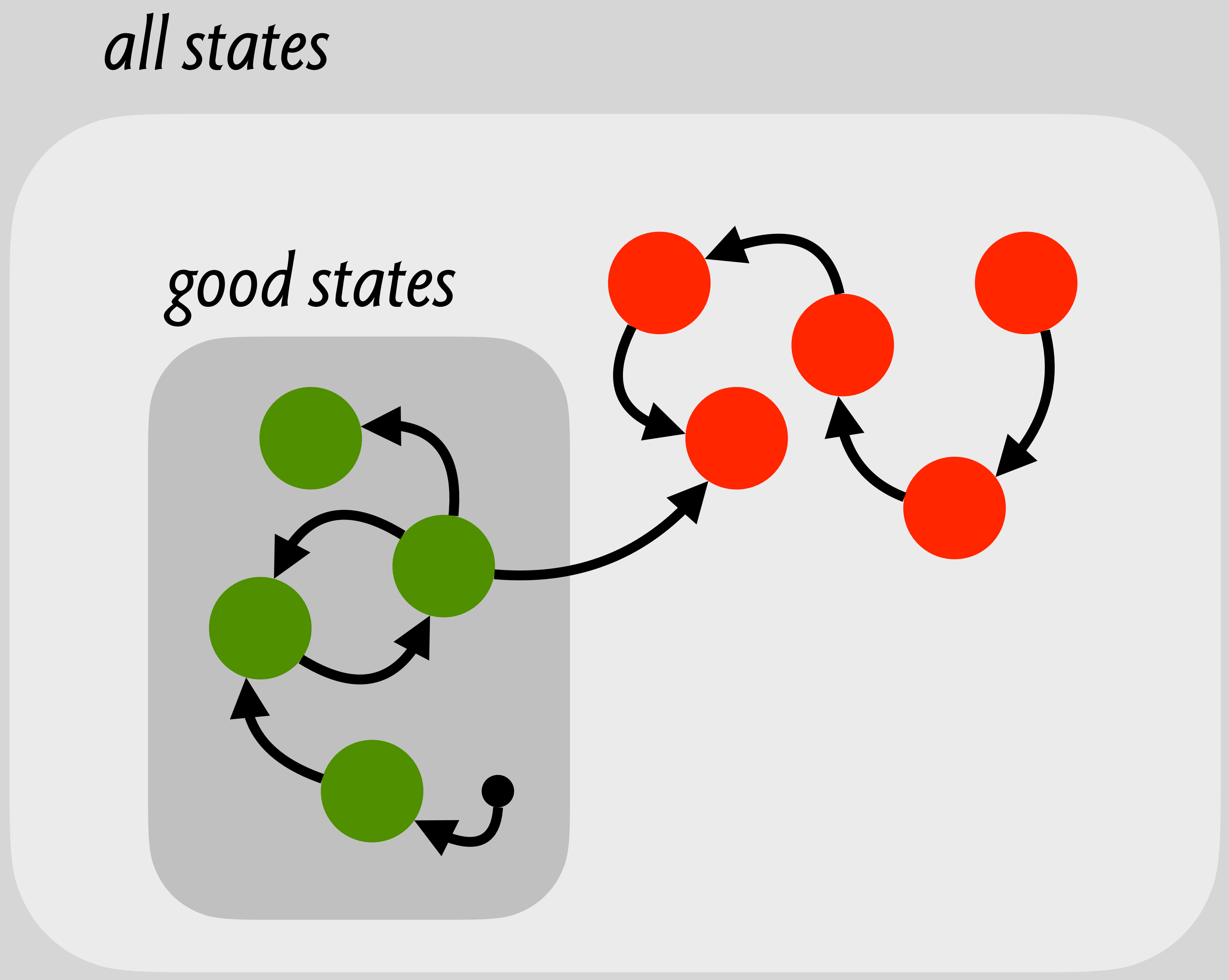
classifying states



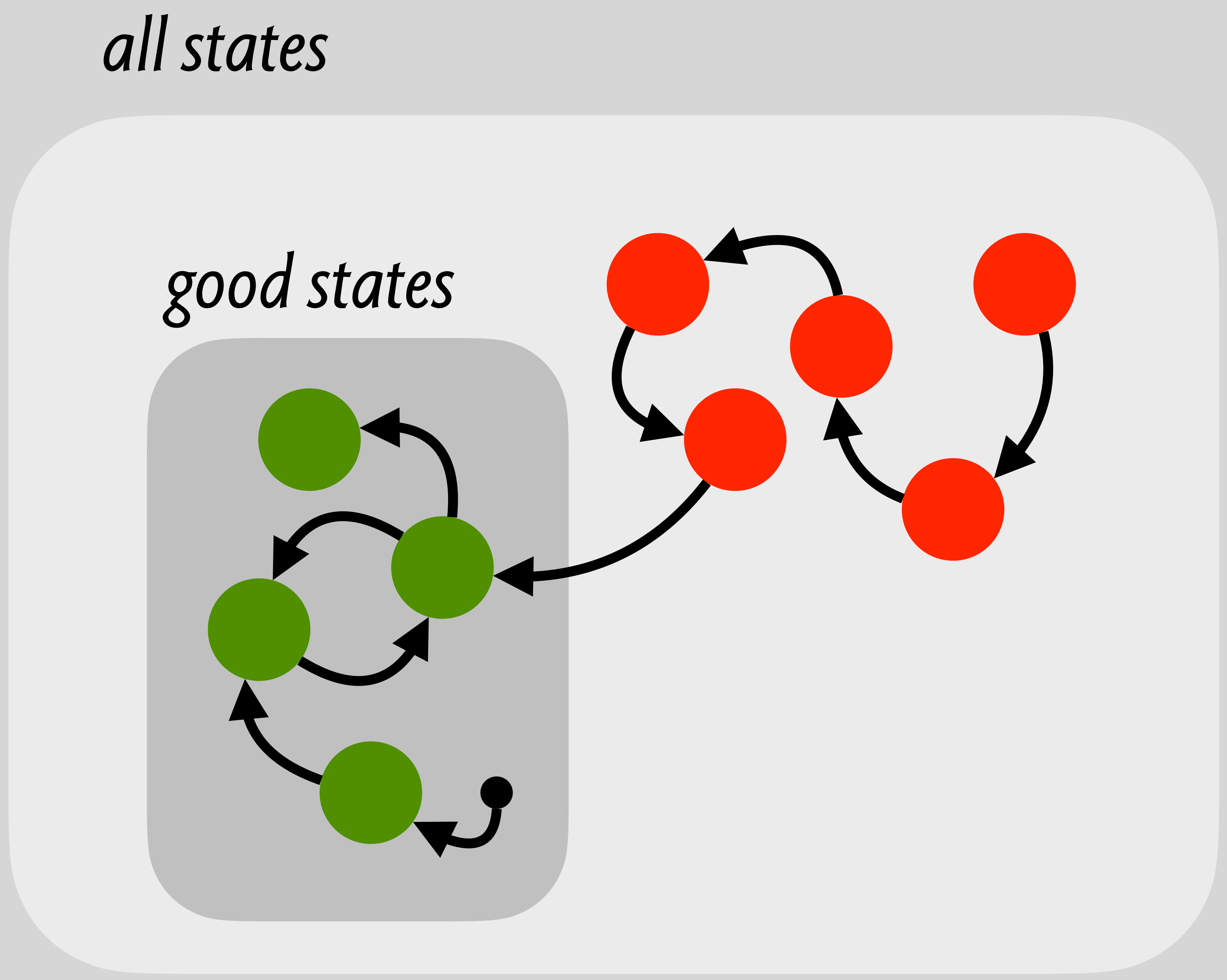
a safe design



an unsafe design



inductive reasoning strategy



what we want to avoid
reasoning about all scenarios
complicated and tedious!

a better approach
reasoning about steps taken by actions
(1) check that the initial state is good
(2) and no action goes from a good to a bad state

applying inductive reasoning to reservation concept

concept RestaurantReservation

```
state
a set of Slots with
  a Time
a set of Reservations with
  a User
  a Slot
```

```
actions
  createSlot (time: Time)
    effects creates a fresh slot for the time
  reserve (user: User, time: Time): Reservation
    requires some slot at this time not yet reserved
    effects creates & returns a fresh reservation
        associates it with user and slot
```

the invariant we want to check

at most one reservation for a given slot

check that the invariant holds in initial state

✓ initially, no reservations

check each action preserves invariant

✓ only the reserve action modifies set of reservations

reserve action ensures slot is not reserved

states & data models
getting more precise

simplifying the state

concept RestaurantReservation

state
a set of Slots with
  a Time
a set of Reservations with
  a User
  a Slot

before, we represented like this

slot	time
s0	July 4, 2025 at 7:00pm

res	user	slot
r0	u1	s0

here's a simpler, more atomized representation

Slot
s0

Reservation
r0

User
u1

these are SETS

time

s0	Ju..

user

r0	u1

slot

r0	s0

these are BINARY RELATIONS

a diagrammatic form

Slot	Reservation	User
s0	r0	u1

these are SETS

time		user		slot	
s0	Ju..	r0	u1	r0	s0

these are BINARY RELATIONS

$$\begin{tikzcd}
& \text{Reservation} \arrow[dl, "\text{slot}"'] \arrow[dr, "\text{user}"] & \\
\text{Slot} \arrow[d, "\text{time}"'] & & \text{User} \\
\text{DateTime} & &
\end{tikzcd}$$

why kind of set is DateTime?
a set of built-in values

what are the values of Slot, eg?
they're identifiers

concept does not expose composite objects!

concept RestaurantReservation

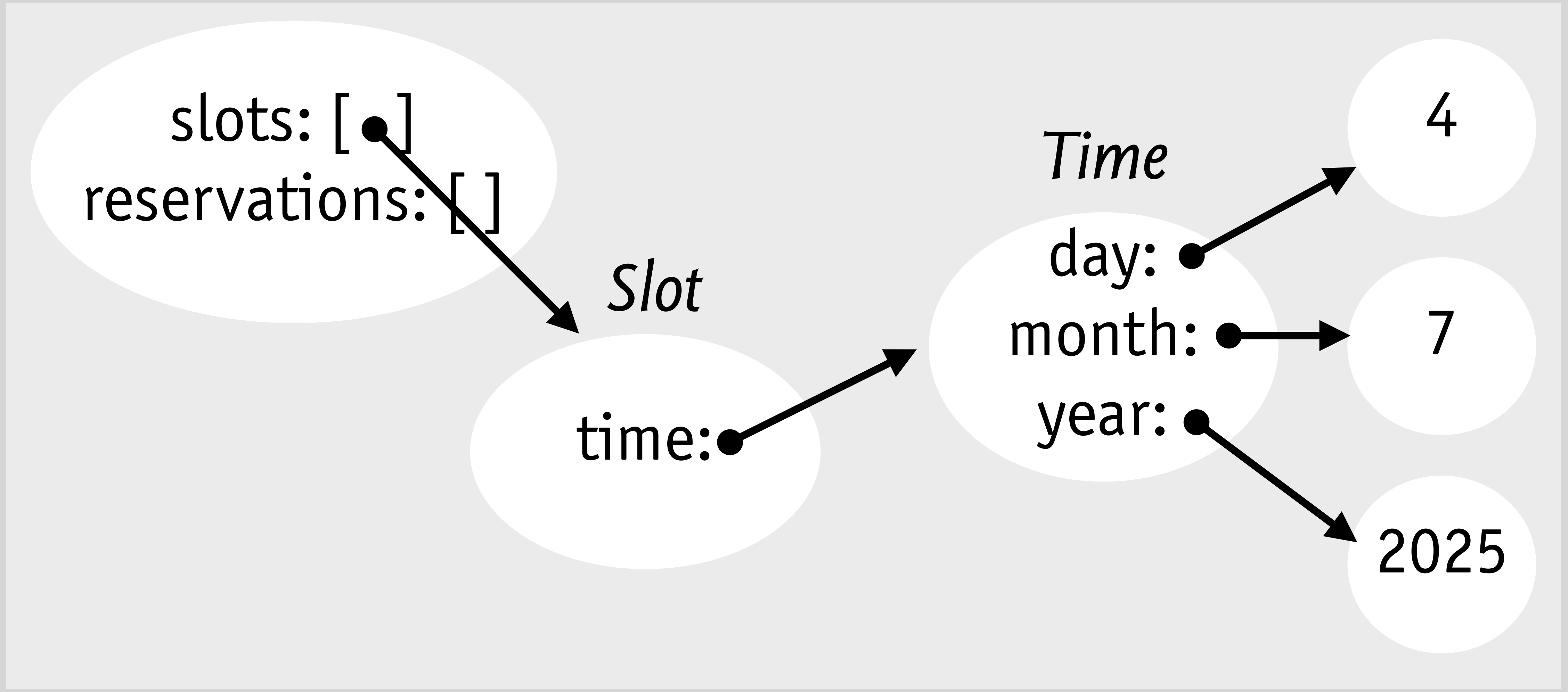
state
a set of Slots with
  a Time
a set of Reservations with
  a User
  a Slot

actions
  ~~addSlot (s: Slot) // BE CAREFUL!~~
    ~~**effect** adds the slot s to the set of slots wit[illegible]me~~

A relational view of the state

slot	time
s0	July 4, 2025 at 7:00pm

An object-oriented view of the state



summary

states can be represented as just sets & binary relations
never need tables with more than two columns

this allows a nice diagrammatic representation
this is the “entity relationship diagram”

no composite objects are visible
a slot is just an identifier associated with a time etc
not a composite object (but could be implemented as one)

why this model helps
succinct and precise, brings clarity during design
easily translated into code (and database schemas etc)

your turn:
a design challenge

how to reduce no-shows

what's a no-show?
diner makes reservation, doesn't cancel but doesn't show up

how might you design to reduce no-shows?
consider concept design interventions

which concepts might you add?
and how would their actions be sync'd with reservation actions?

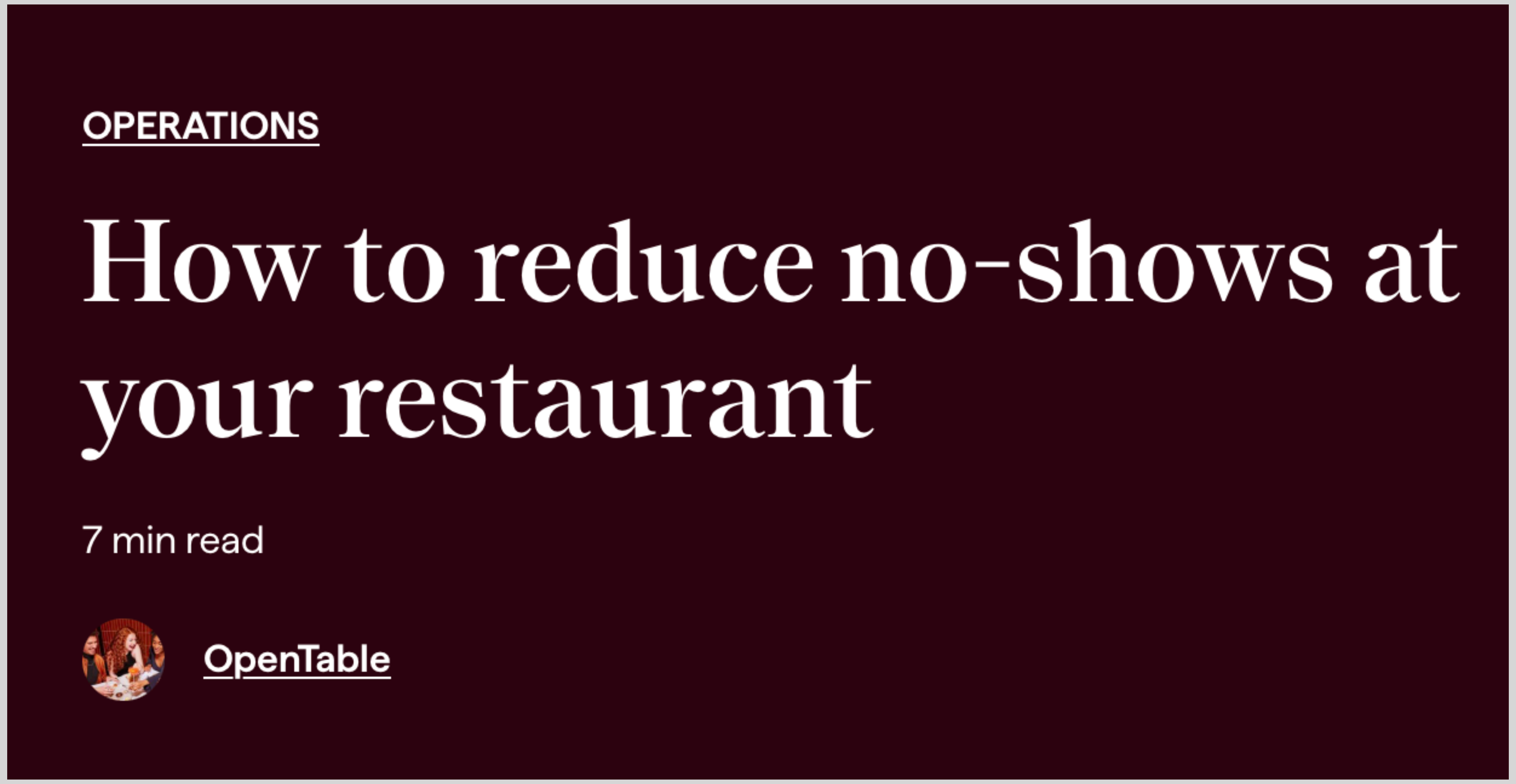
some possible solutions

modifying reservation concept
add invariant: at most one reservation at a given time
require confirmations of reservations

adding a payment concept
require deposit for make a reservation

adding a reminder concept
remind diners when they have an upcoming reservation

adding a karma concept
track no-shows and ban repeat offenders
share no-show data with restaurants

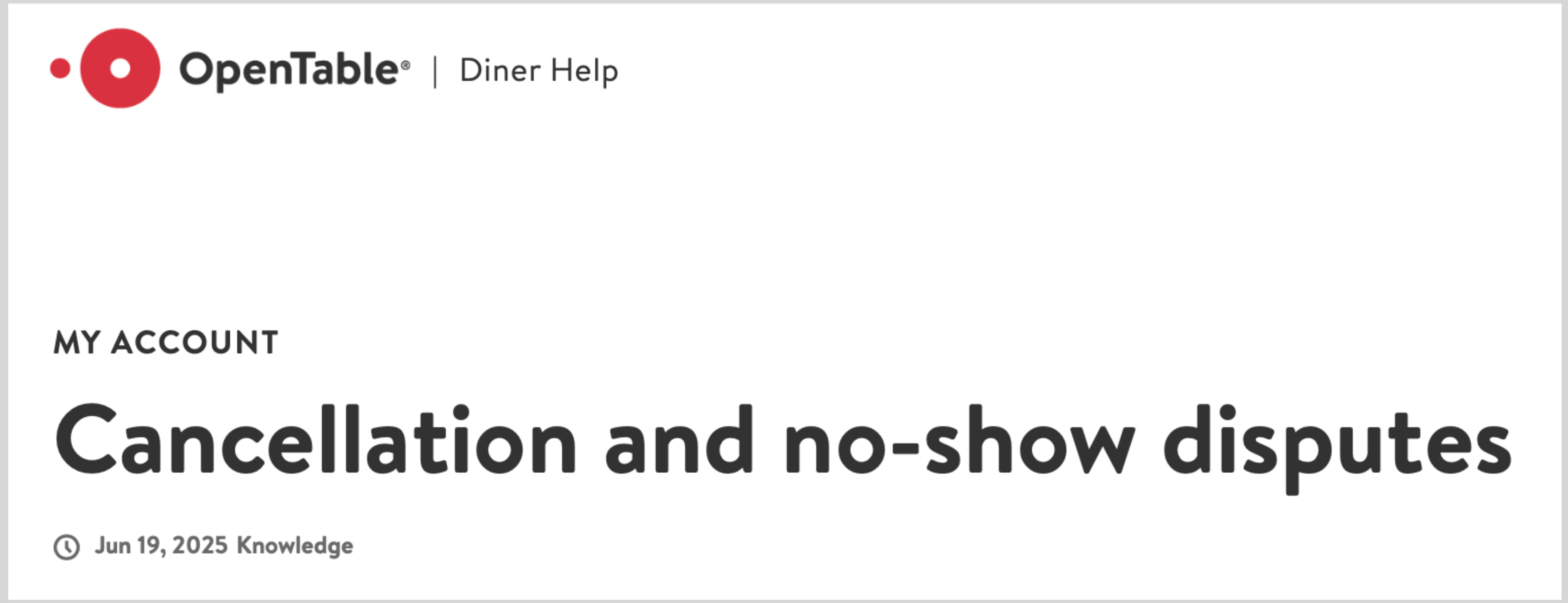


OPERATIONS

How to reduce no-shows at your restaurant

7 min read

OpenTable



OpenTable | Diner Help

MY ACCOUNT

Cancellation and no-show disputes

Jun 19, 2025 Knowledge

heuristics
for states & actions

do you have enough actions?

is purpose/value delivered?
note that have info in state may be enough

have you covered the whole life cycle?
is there an initial setup? a winding down?

are there ways to undo previous actions?
or to compensate if erroneous?

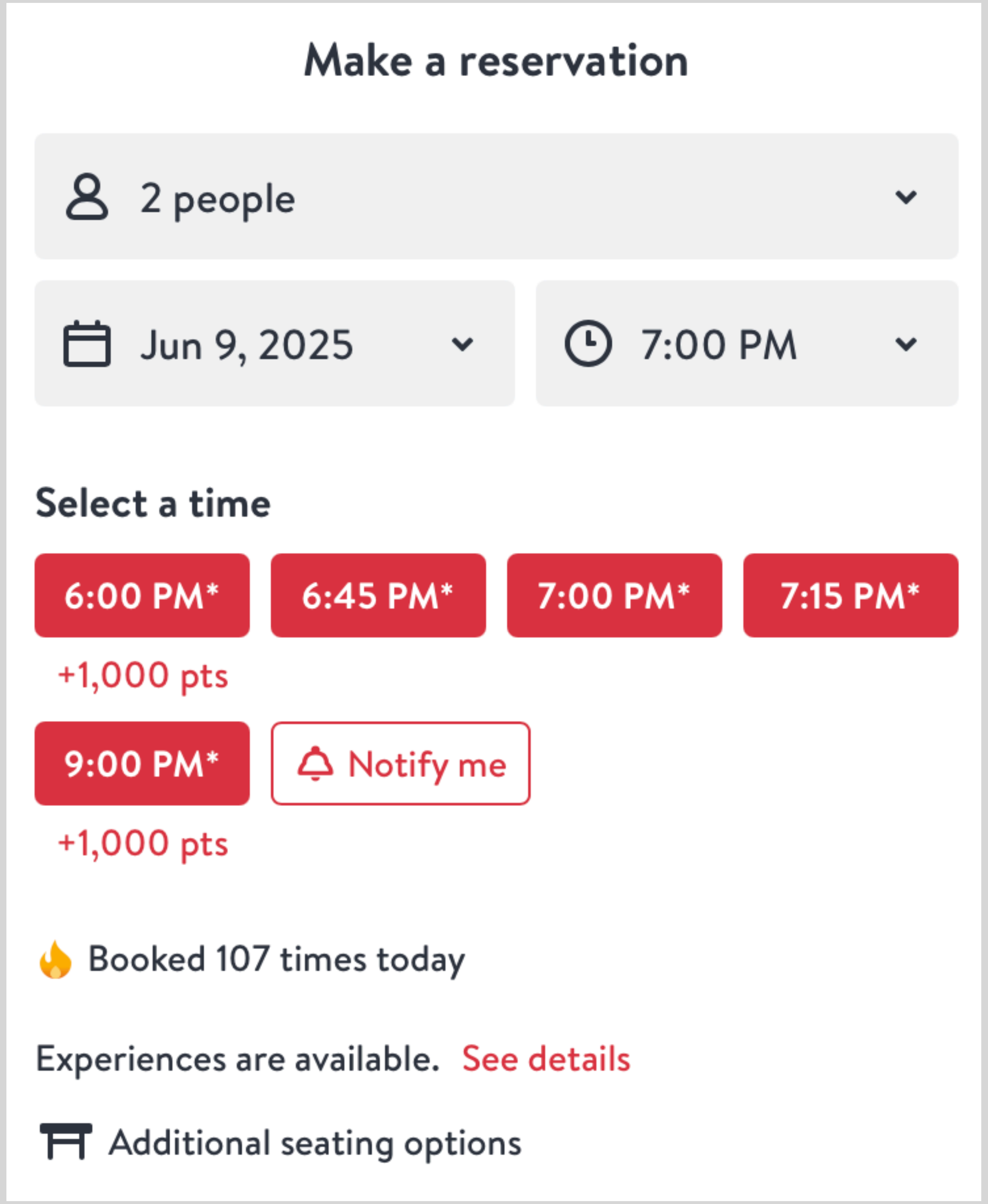
do all nouns have create, update, delete?
for associated state?

seat action?

create slots?

unseat?
cancel reservation?

change reservation?


Make a reservation

2 people

Jun 9, 2025 | 7:00 PM

Select a time

6:00 PM* +1,000 pts | 6:45 PM* | 7:00 PM* | 7:15 PM*

9:00 PM* +1,000 pts | Notify me

Booked 107 times today

Experiences are available. See details

Additional seating options

concept Reservation
actions reserve...

do you have a rich enough state?

can you support all your actions?
determine if allowed, and generate results

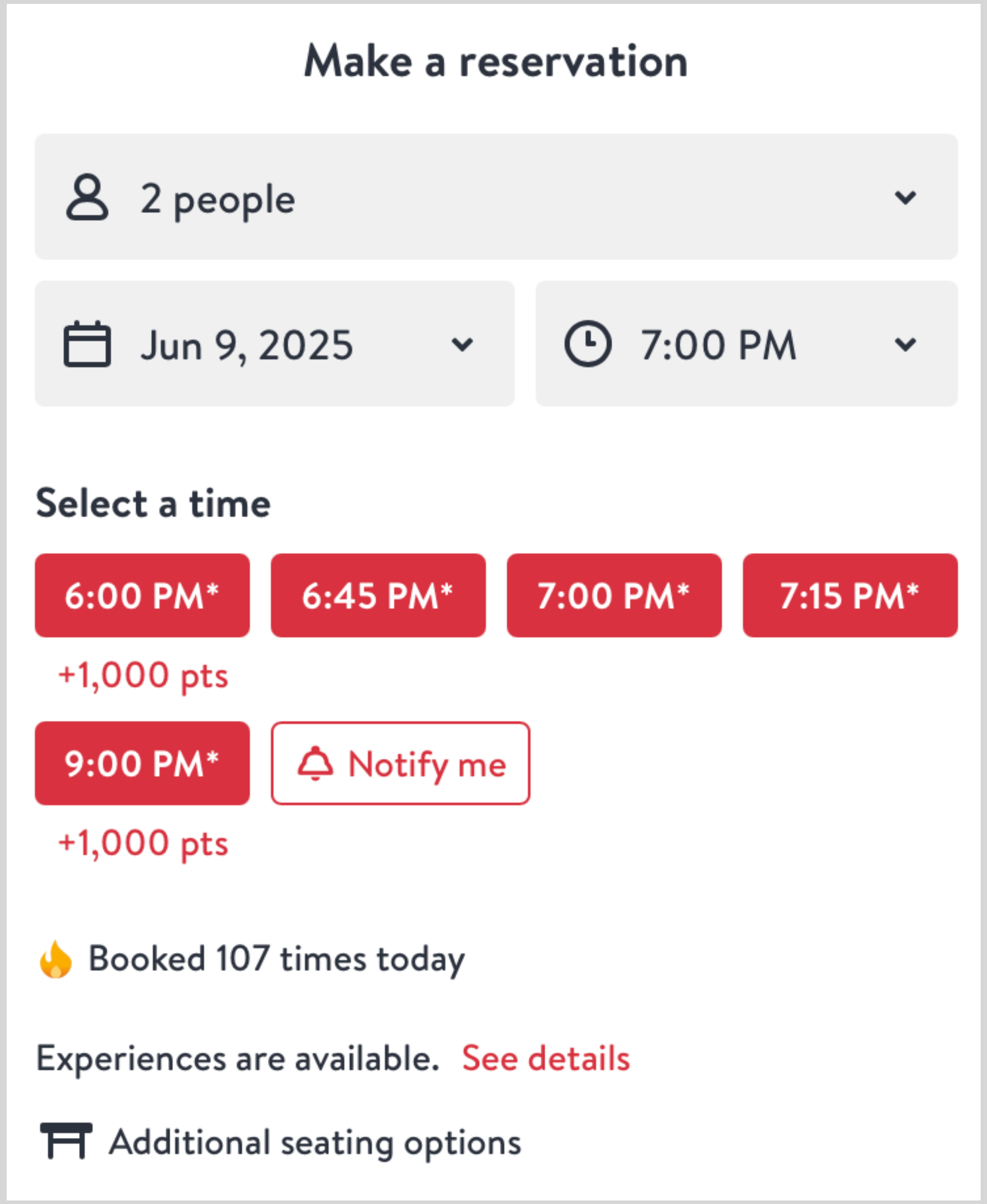
should you track history?
remember completions, deletions, undos?

what info about action occurrence?
maybe also who did it? when?

table sizes?

retain after seat?

by vs. for?
time of reservation?



Make a reservation

2 people

Jun 9, 2025

7:00 PM

Select a time

6:00 PM* +1,000 pts

6:45 PM*

7:00 PM*

7:15 PM*

9:00 PM* +1,000 pts

Notify me

Booked 107 times today

Experiences are available. See details

Additional seating options

concept Reservation
actions createSlot, reserve, cancel, seat, unseat, no-show, ...

takeaways

details matter

big impact on flexibility & complexity

states & data models

a simpler relational view, not composite objects

behavior = traces

actions & visible states; can project onto individual concepts