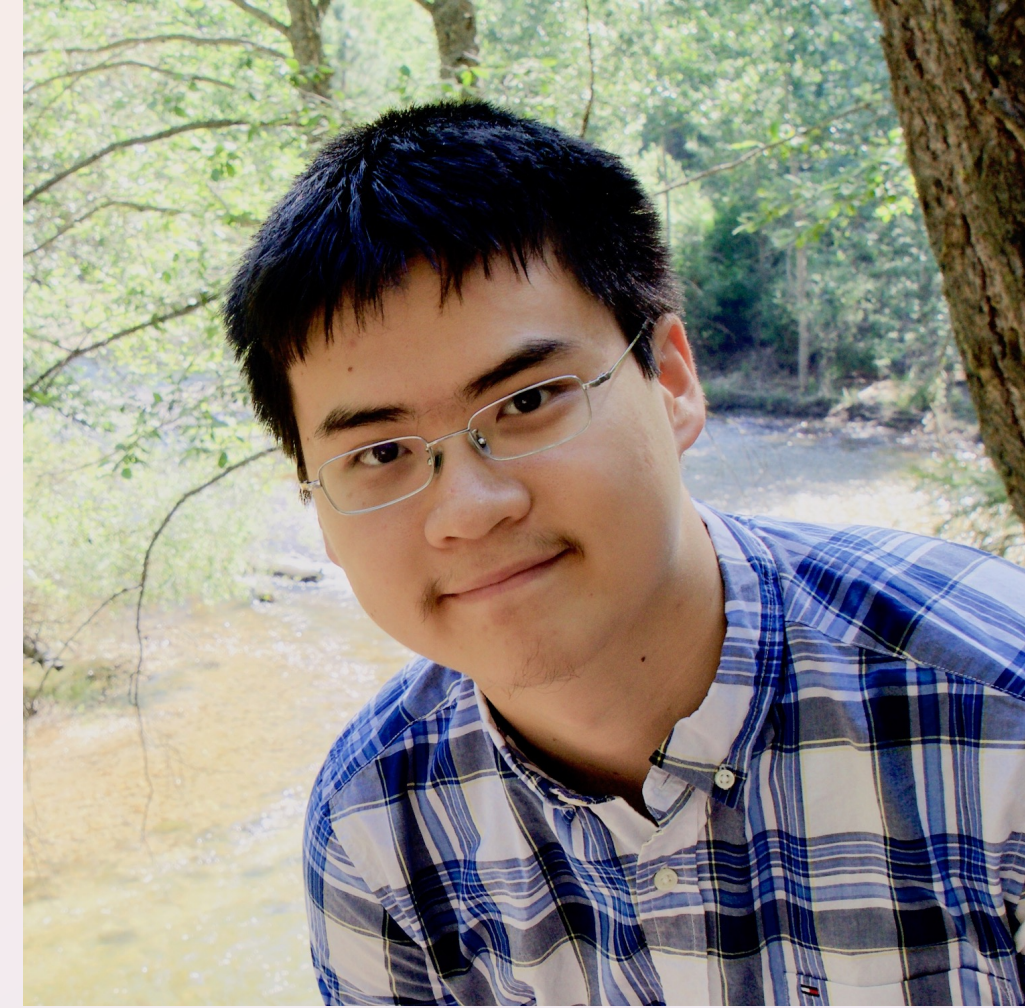


Multimodal Applications

Jackie (Junrui) Yang & Monica Lam

A bit about myself - Jackie Yang

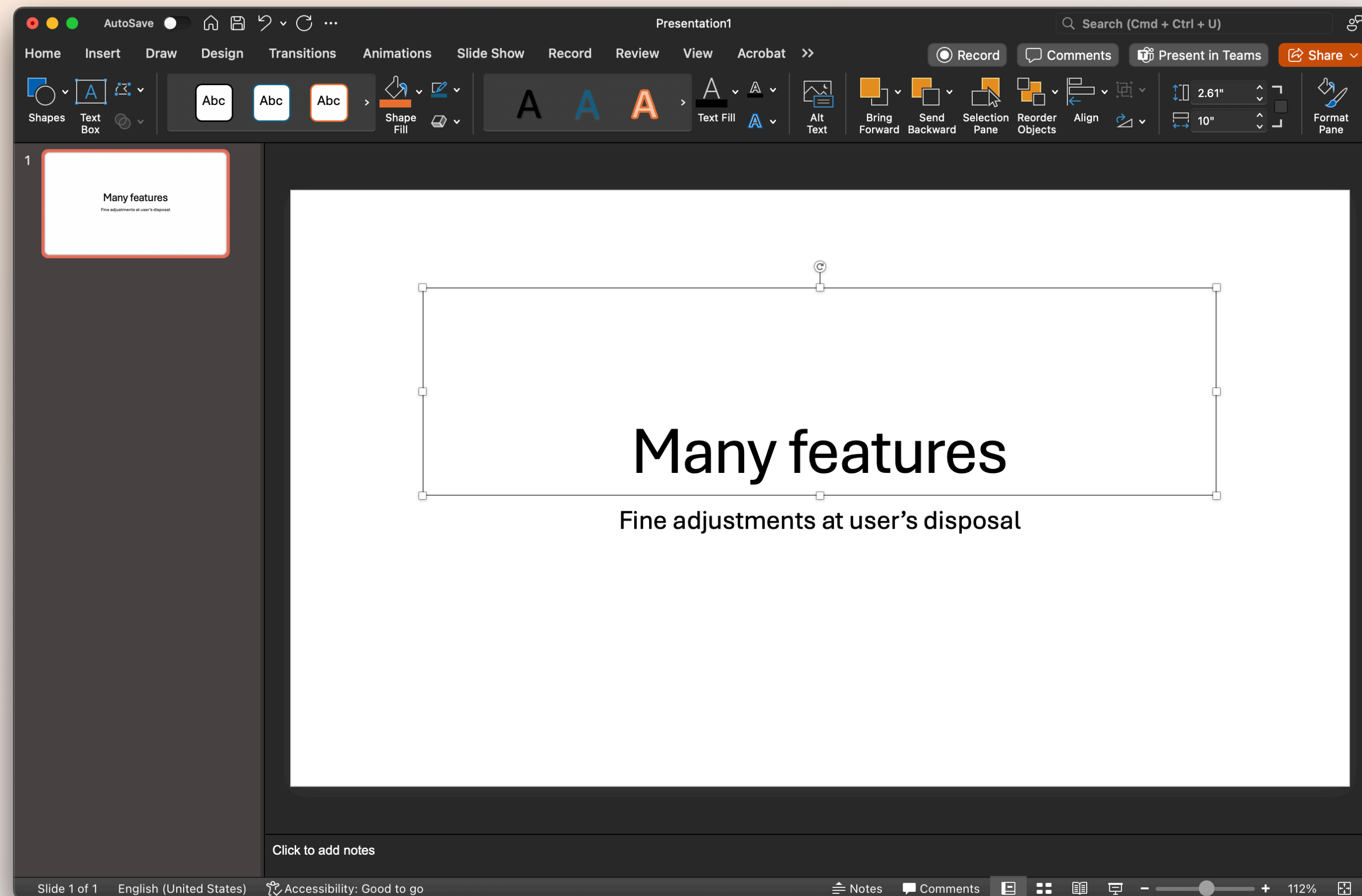


- PhD - Dissertation with Monica Lam and James Landay
- Thesis title: Multimodal AI-augmented User Interface Development Architecture
- Research topics: Multimodal interactions
- Now:
 - Working on my startup: VP of Software and ML @ Skywalk Inc.
 - Vision: *Always-available private* voice interfaces

Lecture Goals

- Why do we need multimodal interactions?
- Three problems for multimodal app development
- ReactGenie: a multimodal app development framework
- GenieWizard: addressing the multimodal app's feature discovery problem

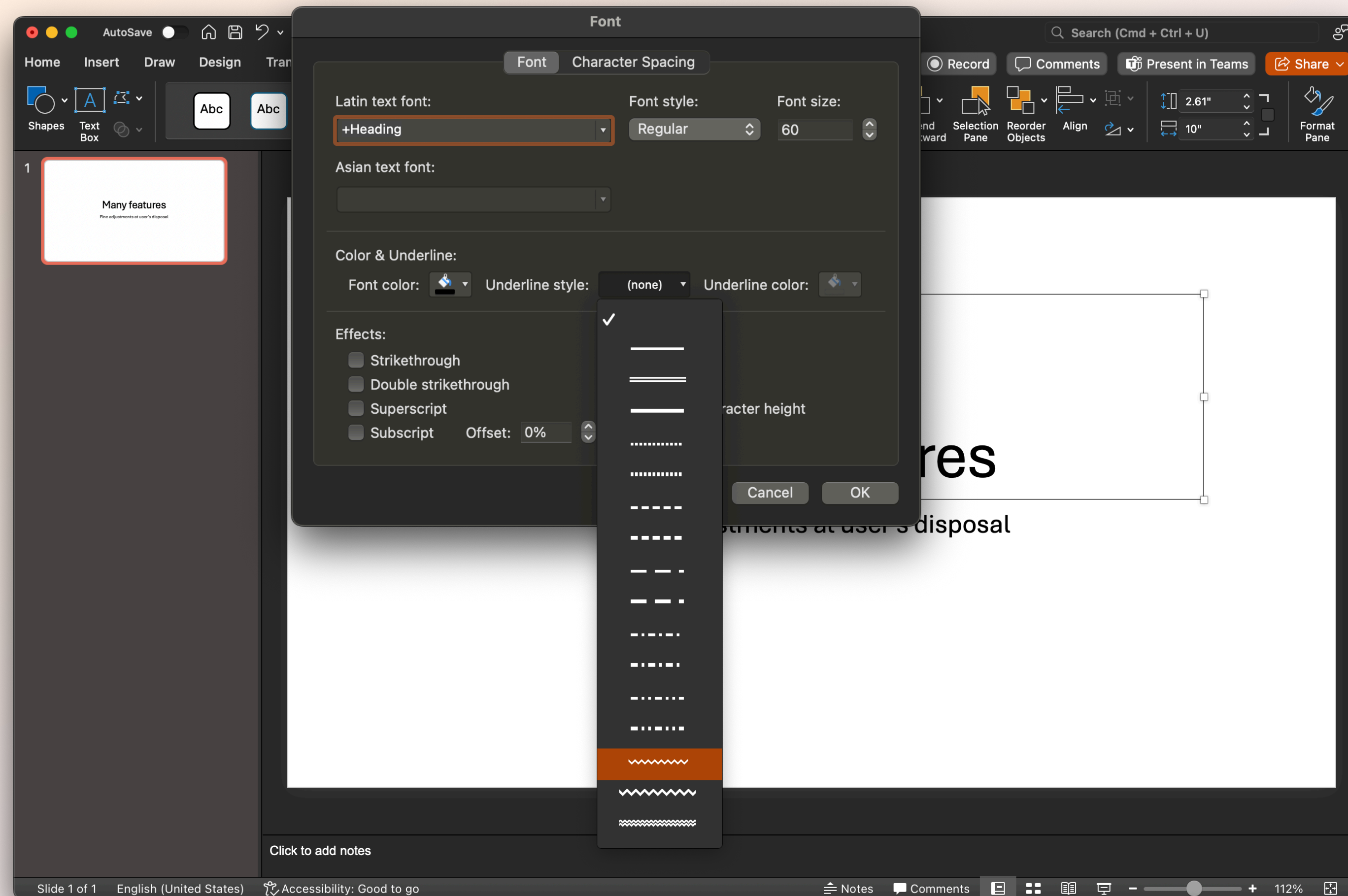
Interacting with Touch + Graphics: Powerful and accurate



PowerPoint:

- Textboxes, pictures, shapes
- Fonts, colors, line styles
- Adjust everything accurately via GUI

Interacting with Touch + Graphics: But sometimes inefficient and repetitive

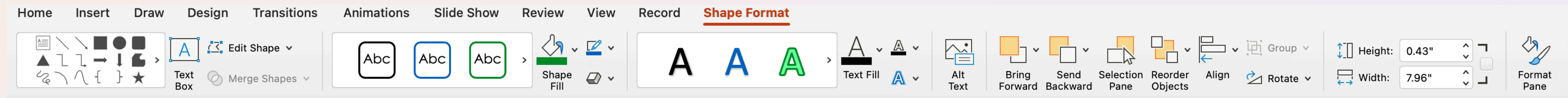
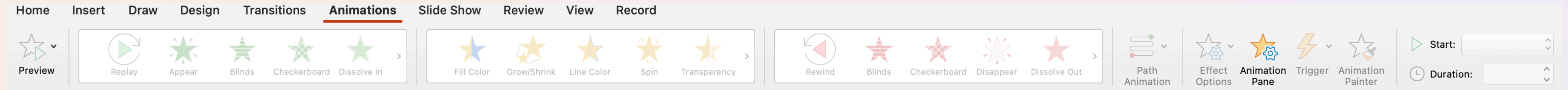
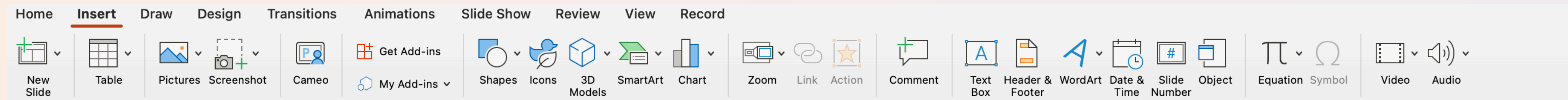
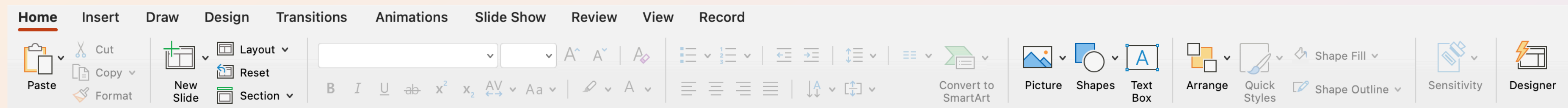


Slow and tedious for

- Apply actions to multiple objects
- Less common features

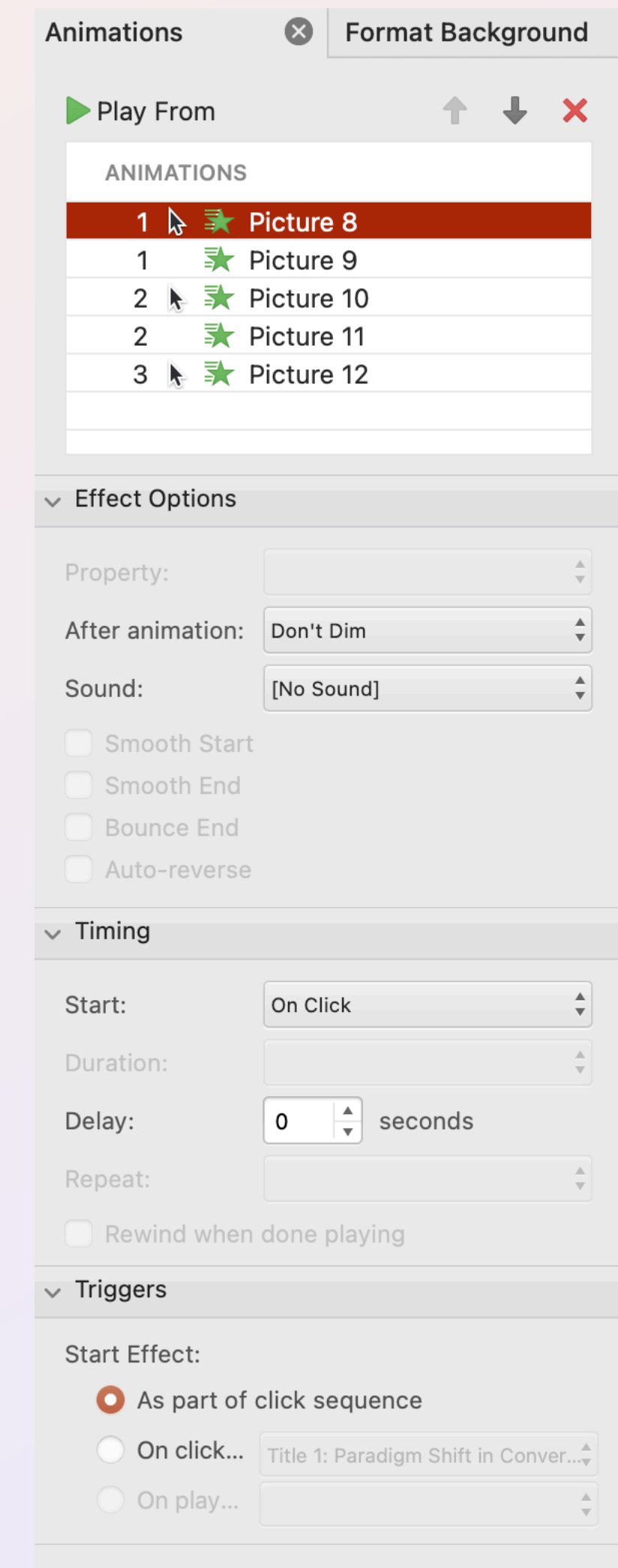
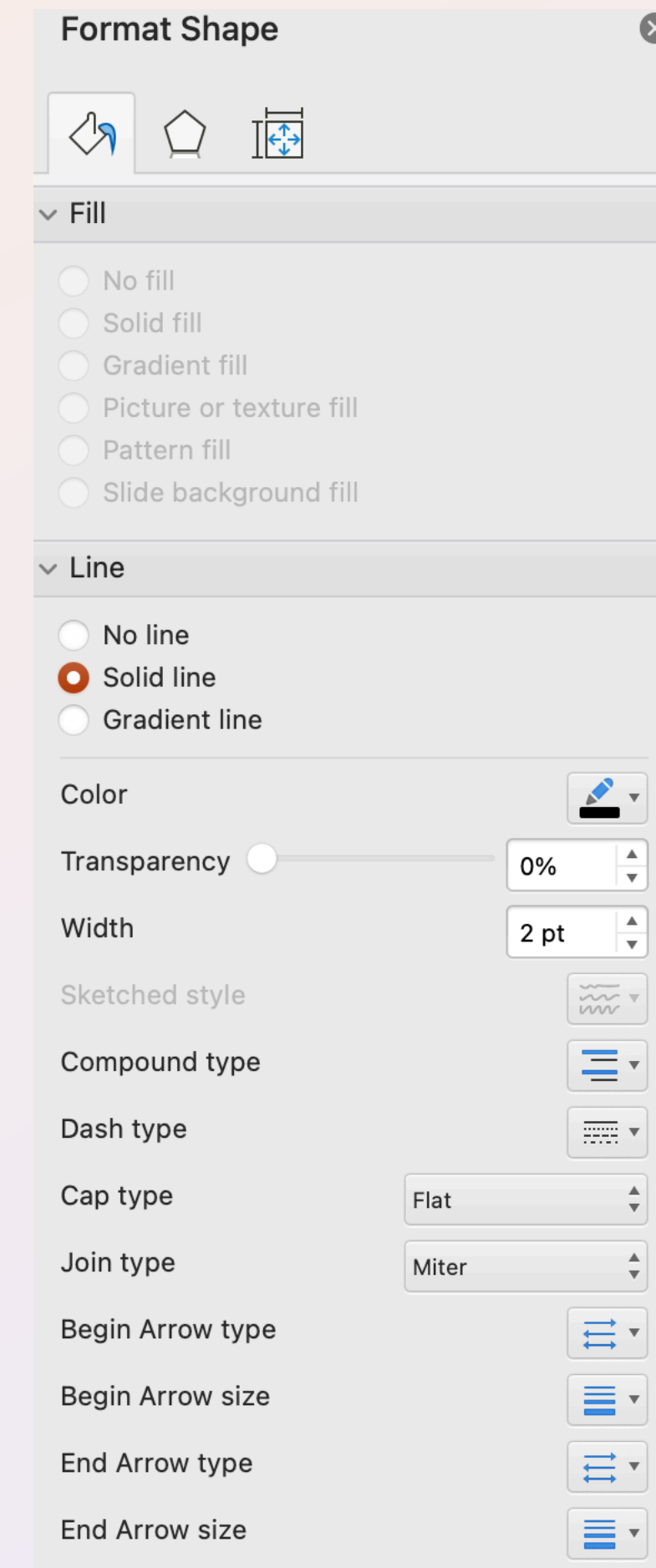
Example: Powerpoint

- Lots and lots of nested menus



Example: Powerpoint

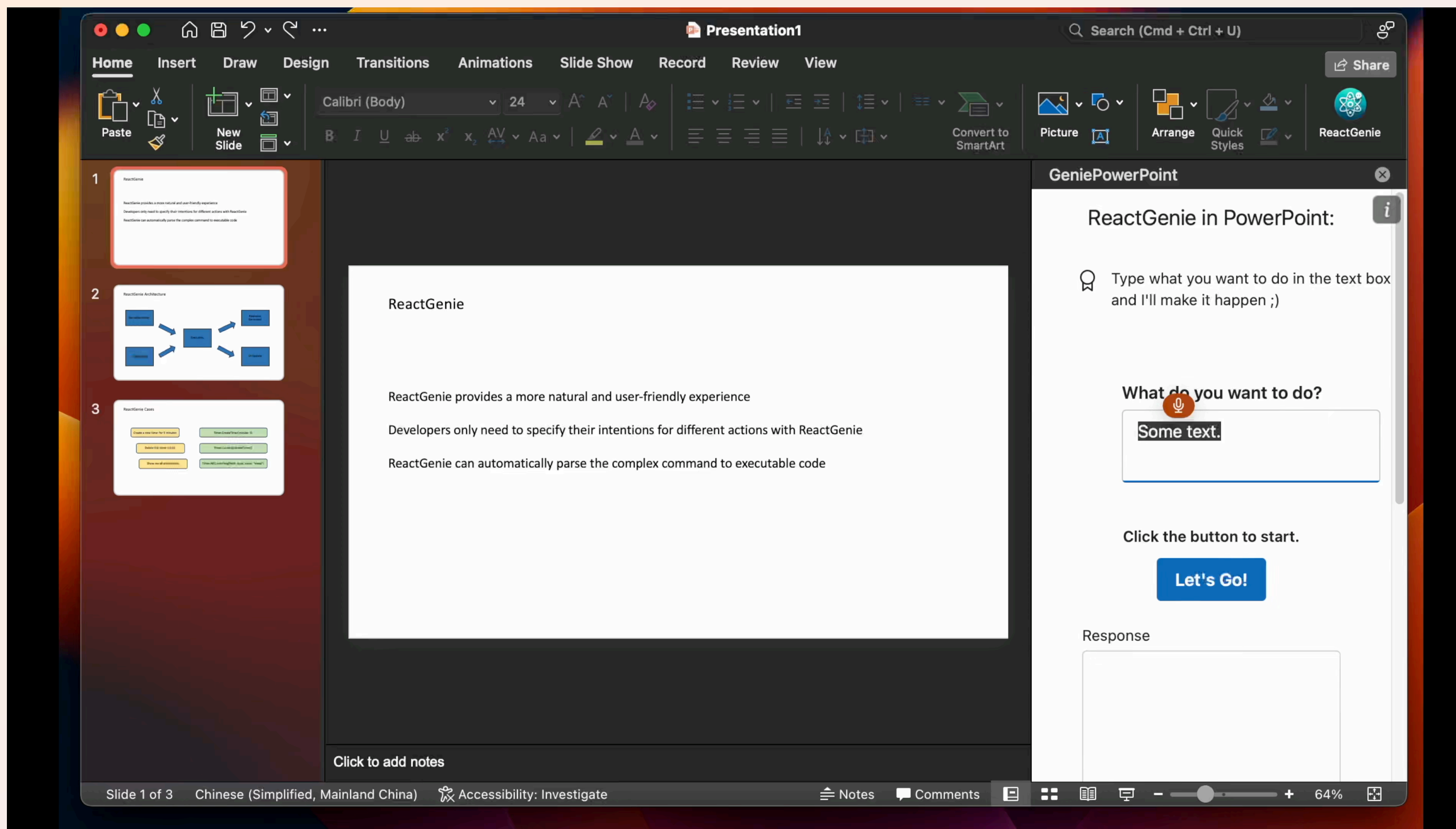
- Lots and lots of nested menus
 - Horizontal ones and vertical ones
- Takes a long time to make slides (even if you know what functions are available)



Here are Some Examples:

- Make this text box bold in the slide master.
- Make the border of this shape with little dots.
- Make everything right aligned on this slide.
- Make every shape on this slide above this yellow.

Multimodal Interaction: High-level goals rather than low-level actions

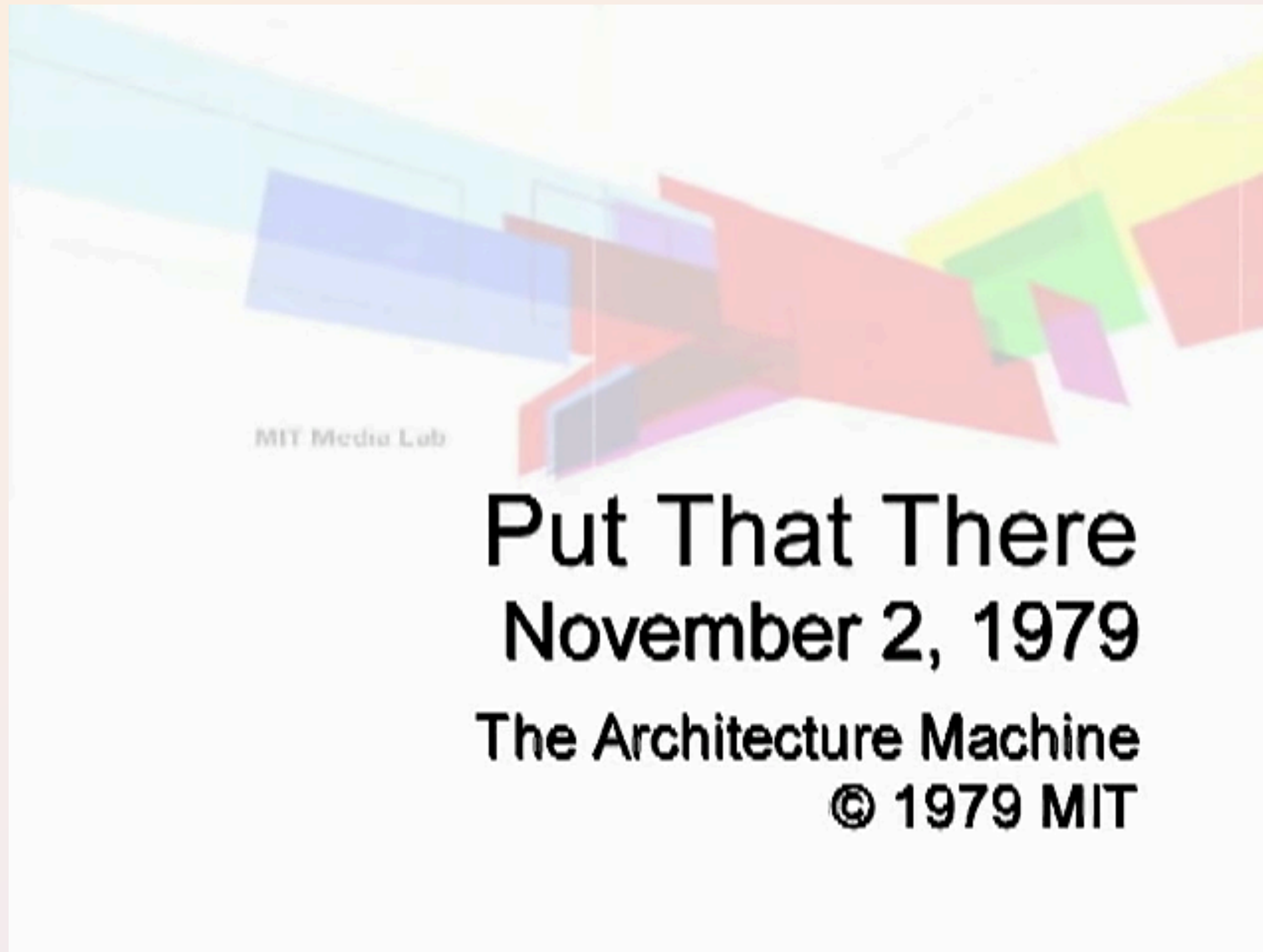


High-level goals
*“Make all the word ‘ReactGenie’
red and bold”*

Instead of

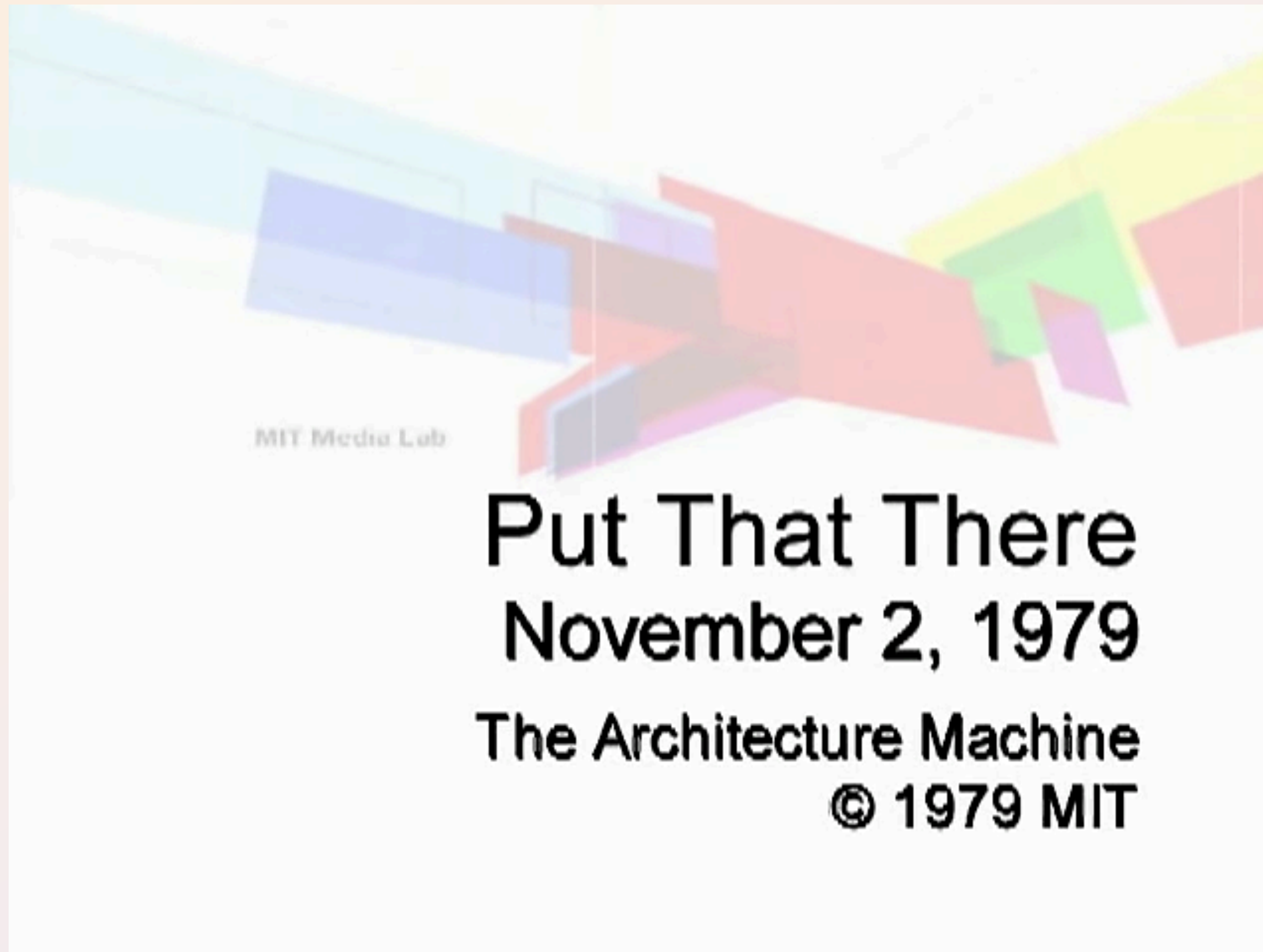
Low-level actions:
[select] -> [change color] ->
[make bold] ... x 3

Prior Work on Multimodal Interaction



Put-That-There
1982 Bolt et al.

Limited adoption



Why?

Lecture Goals

- Why do we need multimodal interactions?
- Three problems for multimodal app development
 - Compositionality of multimodal commands
 - Expose diverse actions/APIs from a GUI app
 - Allows for interchangeable and simultaneous multimodal interactions
- ReactGenie: a multimodal app development framework
- GenieWizard: addressing the multimodal app's feature discovery problem

Recall the examples earlier

Make this text box bold in the slide master.

Make the border of this shape with little dots.

Make everything right aligned on this slide.

Make every shape on this slide above this yellow.

Problem 1: Compositionality of multimodal commands

Make this text box bold in the slide master.

`MakeSlideMasterTextBold()`

Make the border of this shape with little dots.

`MakeShapeBorder(borderType:"littleDots")`

Make everything right aligned on this slide.

`SetEverythingAlignment(alignment:"Right")`

Make every shape on this slide above this yellow.

`SetShapeAboveColor(color:"Yellow")`

This is not scaleable with simple function calling!!!

Problem 1: Compositionality of multimodal commands

NLPL (natural-language
programming language)

Make this text box bold in the slide master.

`Slide.Current().getSlideMaster().matching(field:.id,value:Shape.Current().id)
.textFrame.textRange.font.setBold(bold: true)`

Make the border of this shape with little dots.

`Shape.Current().lineFormat.setDashStyle(dashSyle:"RoundDot")`

Make everything right aligned on this slide.

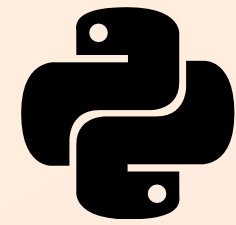
`Slide.Current().getShapes().textFrame.textRange.paragraphFormat.
setHorizontalAlignment(horizontalAlignment:"Right")`

Make every shape on this slide above this yellow.

`Slide.Current().getShapes().between(field:.top,to:Shape.Current().top).
fill.setForegroundColor(color:"yellow")`

Solve Compositionality with compound function calls

Solution to compositionally: NLPL



Python

```
Slide.Current().  
findShape(  
  above=Shape.Current())
```

Weak type, more errors



TypeScript

```
Slide.Current().findShape(  
  Shape.Current().top  
)>.forEach((x)=> x.delete())
```

No param names, more errors



Swift

```
Slide.Current().findShape(  
  above: Shape.Current().top  
)>.forEach{$0.delete()}
```

Ambiguous query

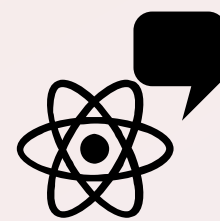


SQL

```
select * from shape  
where top>current_top
```

No Action

Existing language



```
Slide.Current().getShapes()  
>.between(  
  field: .top,  
  to: Shape.Current().top  
)>.delete()
```

Easy to generate
Versatile query
Fewer errors
No Lambda Expression

NLPL

Delete Shapes above this

Automatically
distributed to each
element

Expressiveness of NLPL

C h a n g e b a c k g r o u n d c o l o r f o r a l l t h e y e l l o w s h a p e s t o o r a n g e

VerbObject ModifierObjectVerb Modifier

Shape.All().equals(field: .shapeFill.foregroundColor, value: "yellow").shapeFill.setForegroundColor(color: "orange")

Feature of NLPL	English grammar	Food Ordering	PowerPoint	Social Network
Call function of a object	Singular Object + Verb	Order.Current().place()	TextRange.Current().setBold(bold: true)	Post.Current().like()
Distribute action to an array of objects	Plural Object + Verb	Order.All()[-1].foods.like()	Slide.Current().getShapes().textFrame.setText("")	User.Me().posts.delete()
Specify function parameters	Object + Verb + Verb Modifier	Order.Current().addFoods(foods: [Food.GetFood(name:"Burger")])	Shape.Current().textFrame.setText(text : "12345")	Post.Current().comment(comment: "Nice Photo")
Select objects to do actions	Object Modifier + Object + Verb	Restaurant.Current().foods.sort(field: .price)[0].order()	Shape.All().matching(field: .textFrame.text, value: "yellow").delete()	Post.All().equals(field: .like, value: true)

Presentation1

Search (Cmd + Ctrl + U)

Share

Home

Insert

Draw

Design

Transitions

Animations

Slide Show

Record

Review

View

Paste

New Slide

Calibri (Body) 24

B I U

x^2 x_2 ΔV Aa

≡ ≡ ≡ ≡

↕

↔

Picture

Arrange

Quick Styles

Convert to SmartArt

ReactGenie

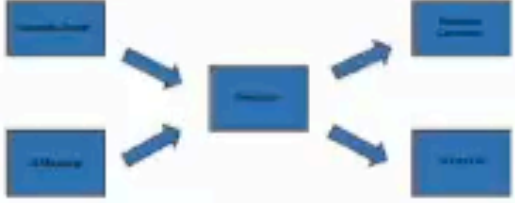
1

ReactGenie

ReactGenie provides a more natural and user-friendly experience
Developers only need to specify their intentions for different actions with ReactGenie
ReactGenie can automatically parse the complex command to executable code

2

ReactGenie Architecture



3

ReactGenie Cases

There is a new error in the code

Insert ReactGenie into the code

Remove the error (if any)

Save the code (and the error)

Generate all the code

Generate the code (and the error)

Click to add notes

GeniePowerPoint

ReactGenie in PowerPoint:

Type what you want to do in the text box and I'll make it happen ;)

What do you want to do?

Some text.

Click the button to start.

Let's Go!

Response

Slide 1 of 3

Chinese (Simplified, Mainland China)

Accessibility: Investigate

Notes

Comments

64%

Genie PowerPoint

- 110 APIs: Slide, Shape, SlideMaster, TextFrame, TextRange, ...
 - That's everything from MS PowerPoint JS API.
 - Yet, it still does not have common APIs like Animation, Font color, etc.
- It is super hard to expose API outside of the development cycle of GUI
- With the advancement of AI, developers need to build two interfaces:
 - A human interface, and
 - An AI interface
- That's double the work on developers!

Problem 2: Expose diverse actions/APIs from a GUI app

- 110 APIs: Slide, Shape, SlideMaster, TextFrame, TextRange, ...
 - That's everything from MS PowerPoint JS API.
 - Yet, it still does not have common APIs like Animation, Font color, etc.
- It is super hard to expose API outside of the development cycle of GUI
- With the advancement of AI, developers need to build two interfaces:
 - A human interface, and
 - An AI interface
- That's double the work on developers!

Problem 2: Expose diverse actions/APIs from a GUI app

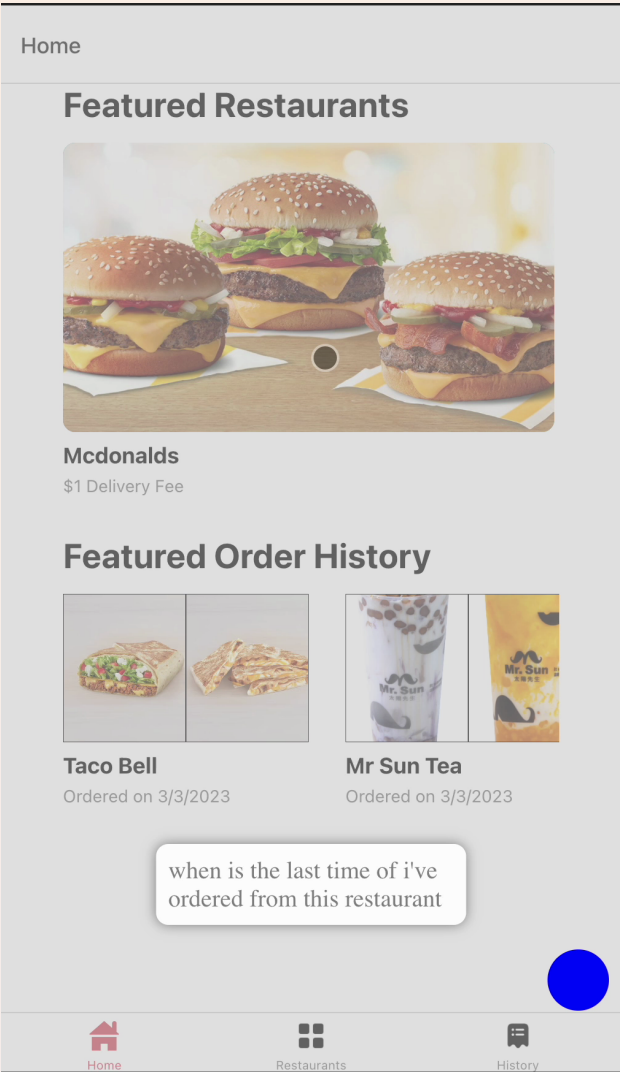
ReactGenie
Annotations

```
@GenieClass("Past order or a shopping cart")
class Order extends DataClass {
  @GenieKey()
  public orderId: string;
  @GenieProperty("Items in the order")
  public orderItems: FoodItem[];
  constructor({orderId, orderItems}: {orderId: string, orderItems: FoodItem[]}) {
    super({orderId, orderItems}); this.orderId = orderId; this.orderItems = orderItems;
  }
  @GenieFunction()
  static All(): Order[] {
    return fetchOrdersFromServer();
  }
  @GenieFunction("Create a new order")
  static CreateOrder(): Order {
    return new Order({orderId: randomId(), orderItems: []});
  }
  @GenieFunction("Add an item to the order")
  addItem({foodItem}: {foodItem: FoodItem}) {
    this.orderItems.push(foodItem); updateServer();
  }
}
```

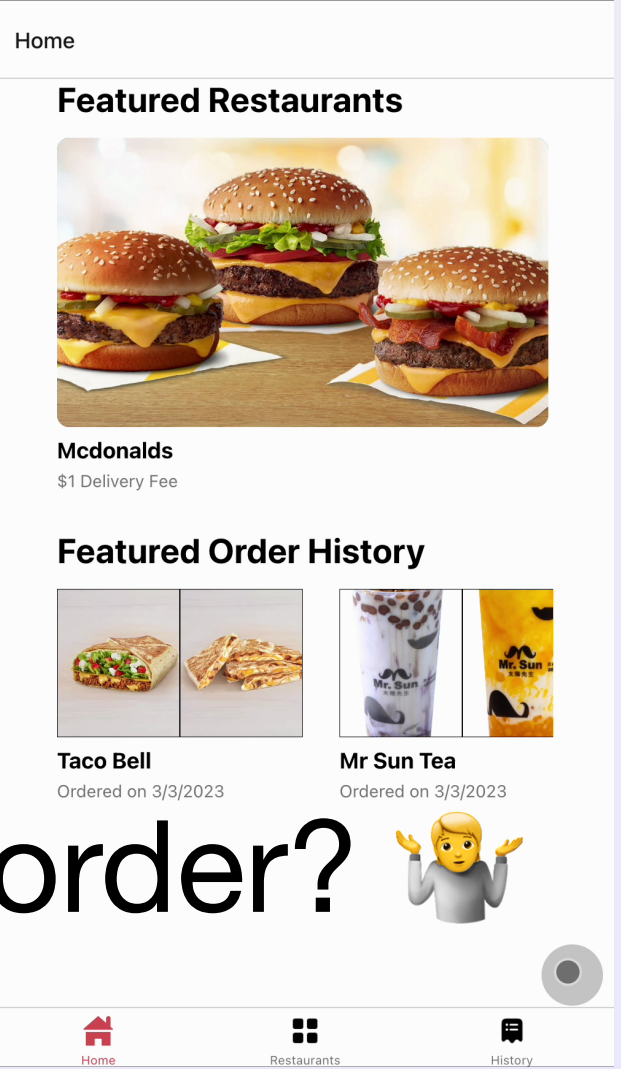
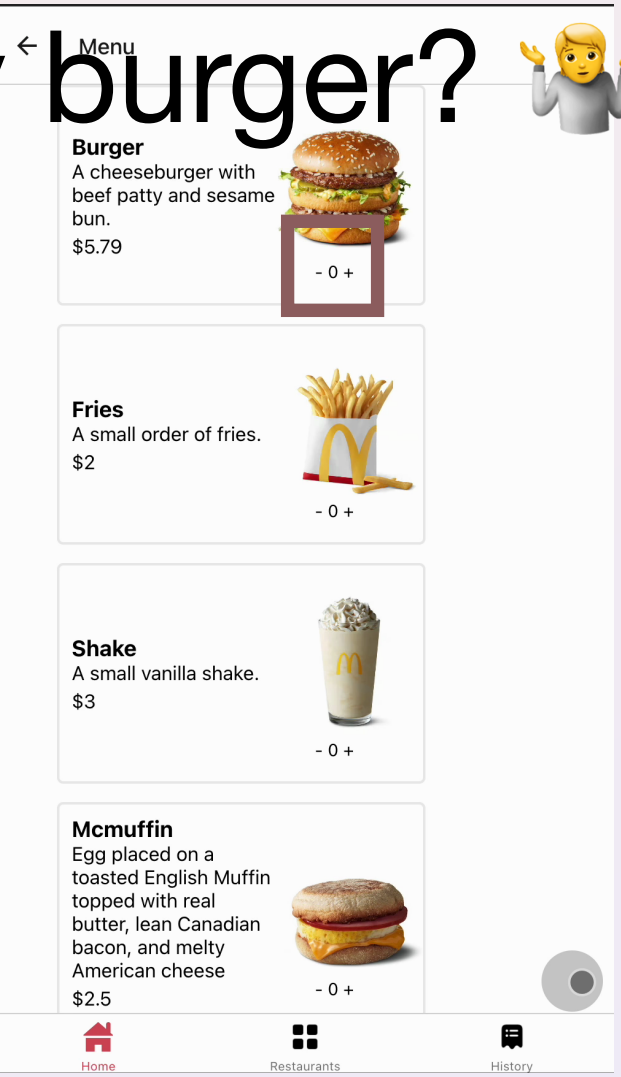
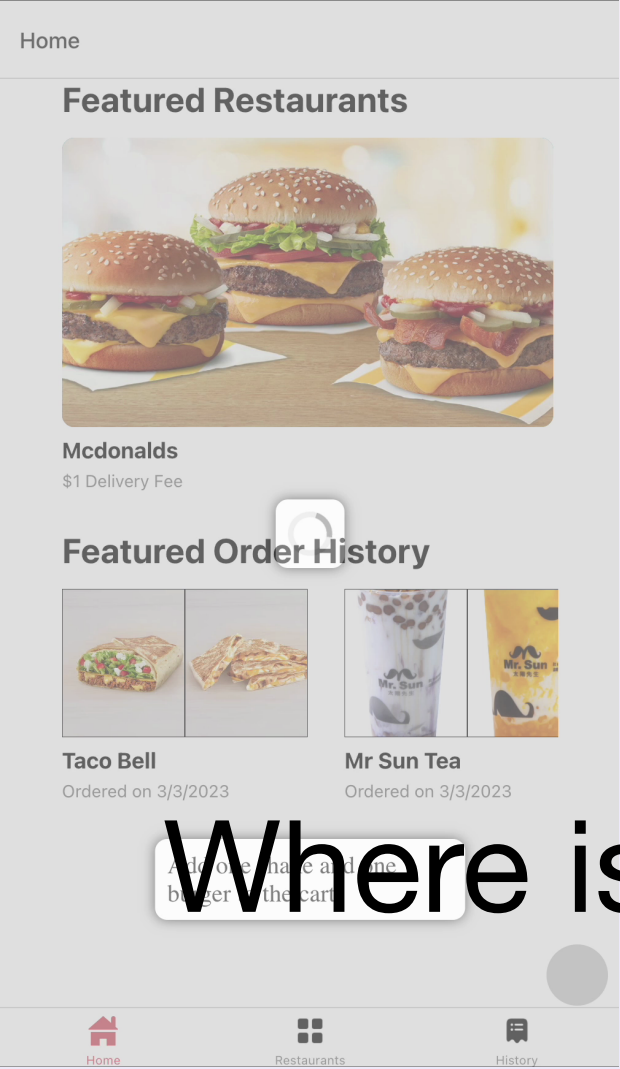
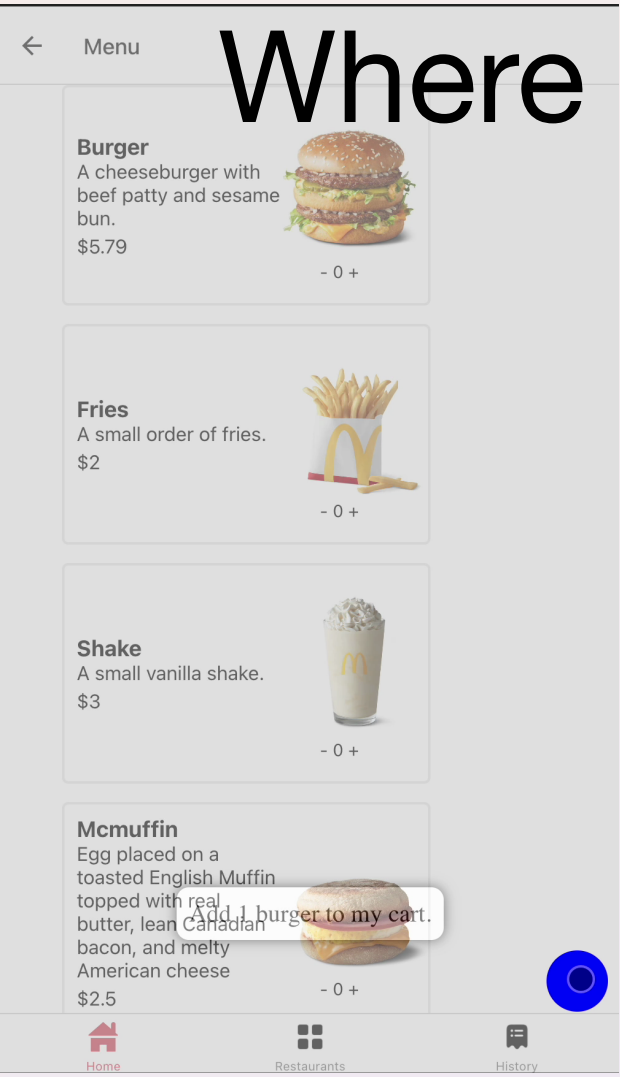
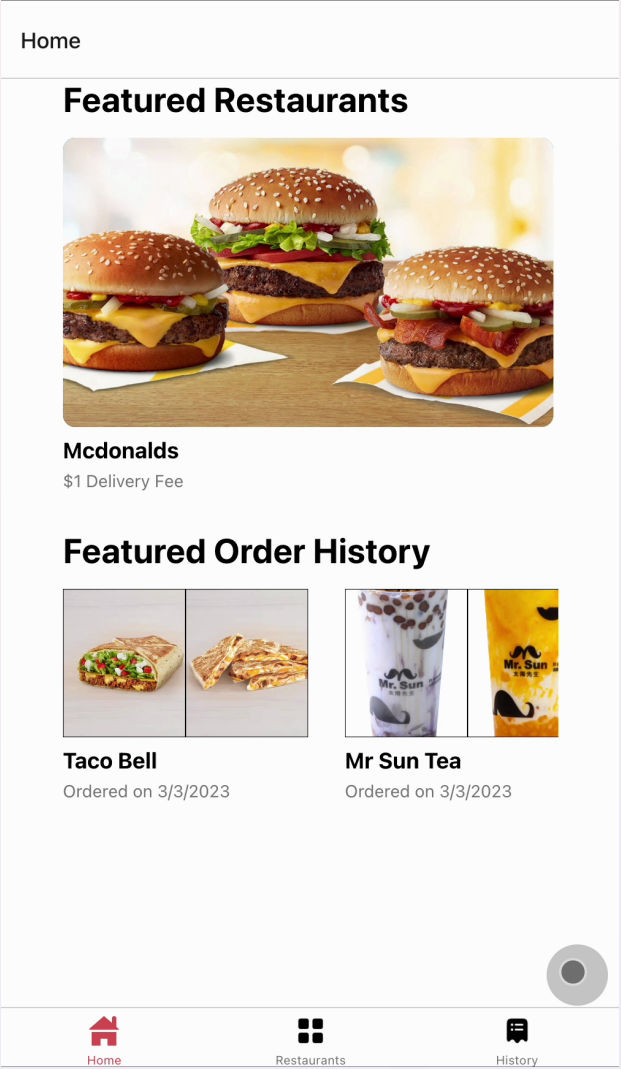
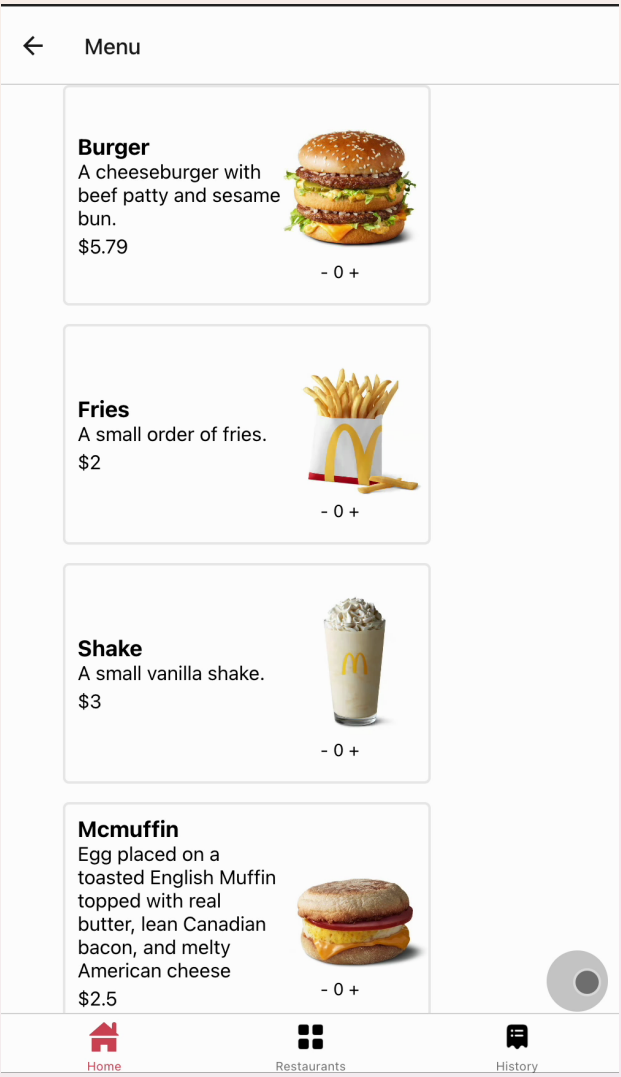
React App Logic Code

When using a ReactGenie app

How can the user know what happened? 🧑



Which restaurant?

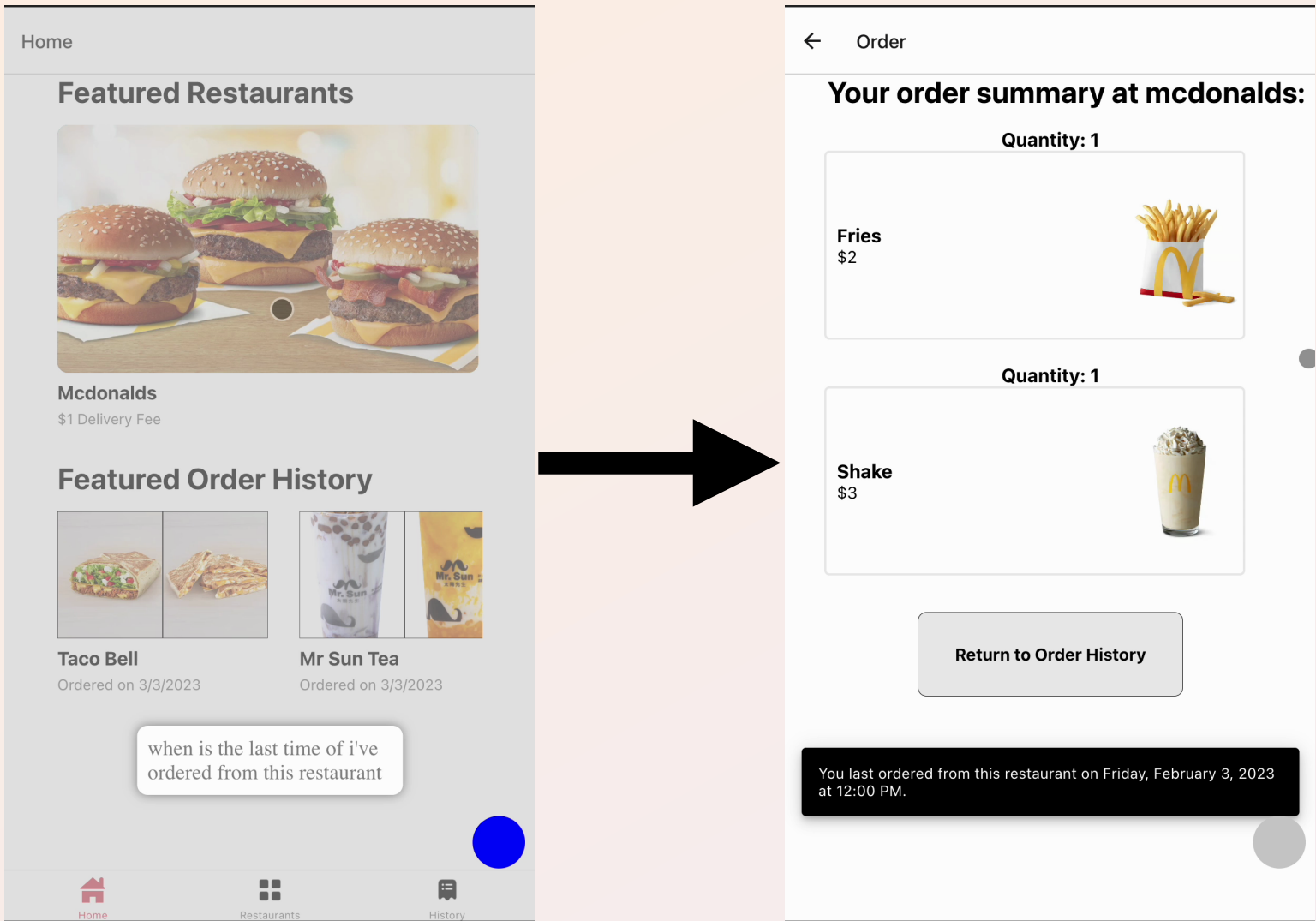


Where is my burger? 🧑

Where is my order? 🧑

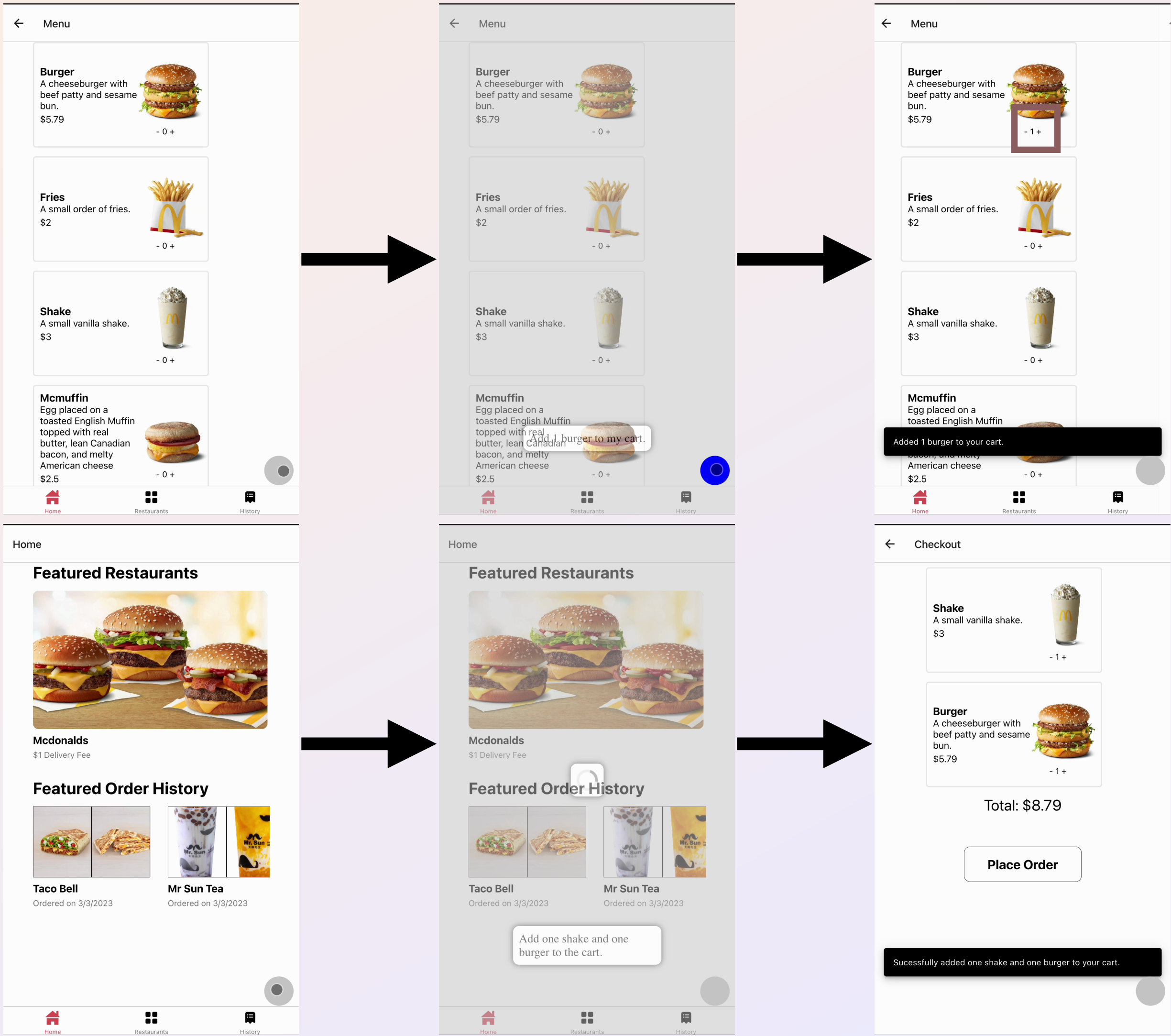
Problem 3: Allows for interchangeable and simultaneous multimodal interactions

Changes on screen: UI updates



UI input mapping

Changes off-screen: Navigation



Problem 3: Allows for interchangeable and simultaneous multimodal interactions

- Multimodal Input
 - UI Input Mapping: Getting what object that I'm touching
- Multimodal Output
 - Resulting objects are on-screen — UI Updates: Update the values on screen
 - Resulting objects are off-screen — Navigation: Navigate to the page with results

Lecture Goals

- Why do we need multimodal interactions?
- Three problems for multimodal app development
- **ReactGenie: a multimodal app development framework**
- GenieWizard: addressing the multimodal app's feature discovery problem

What is ReactGenie?

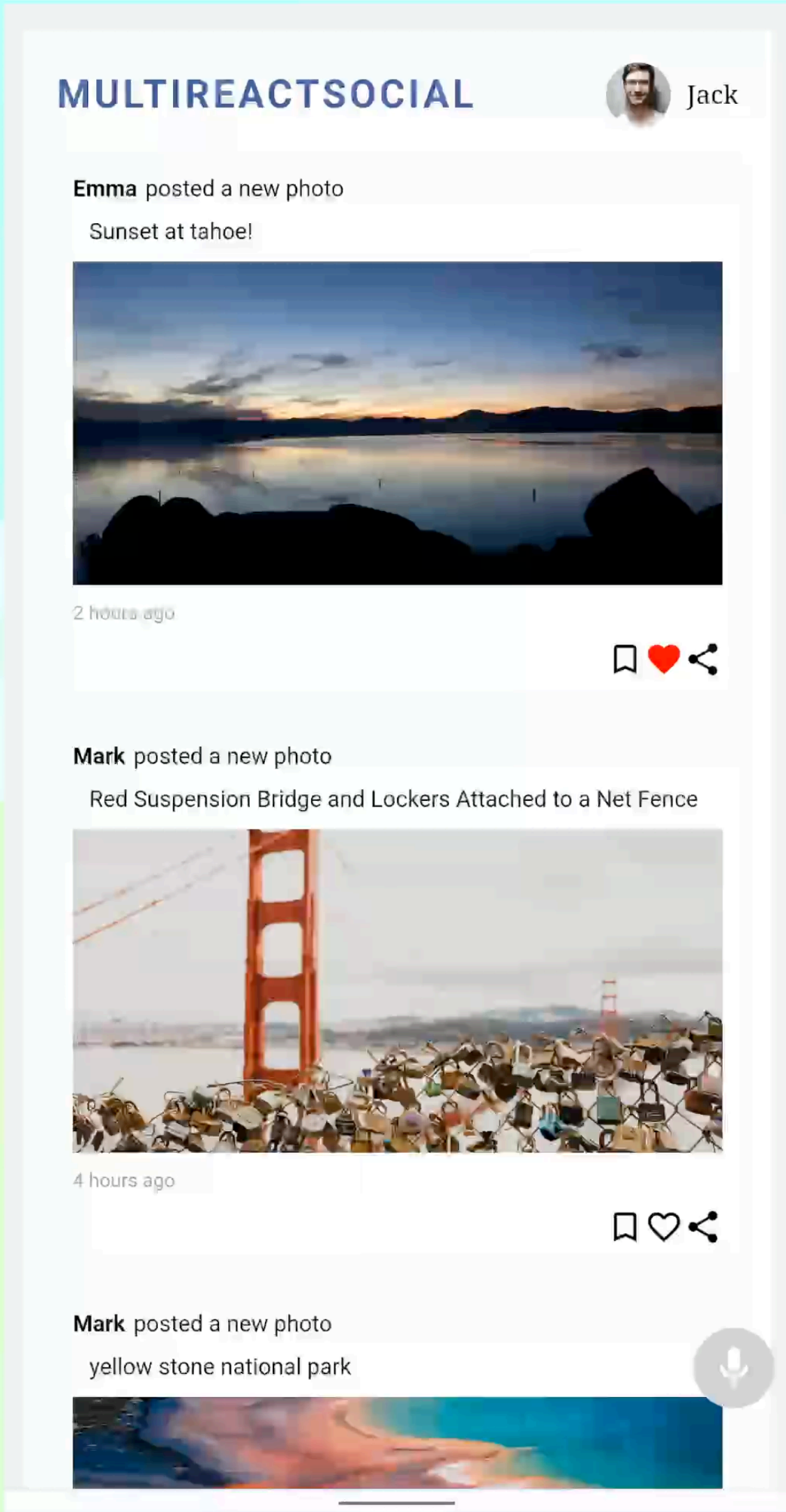
Ease of Development



Rich Multimodal Functionality



ReactGenie Demo



Modern GUI 101: State + Components

```
class Recipe {  
  name: String;  
  img: Image;  
  loved: boolean;  
  
  love(): void {  
    this.loved = true;  
  }  
}
```

State Code: Implements Features

```
RecipeViewImpl = (recipe: Recipe) => {  
  return (  
    <div>  
      <img image={recipe.img}/>  
      <love loved={recipe.loved}  
        onClick={()=>recipe.love()}/>  
      <div> {recipe.name} </div>  
    </div>  
  )  
}
```

Components: Describe GUI

ReactGenie = React + Annotations

```
@DataClass()
class Recipe: GenieClass {
    @GenieProperty()
    name: String;

    img: Image;
    @GenieProperty()
    loved: boolean;

    @GenieFunction()
    love(): void {
        this.loved = true;
    }
}
```

State Annotations:
Which class/property/function
can be accessed with voice

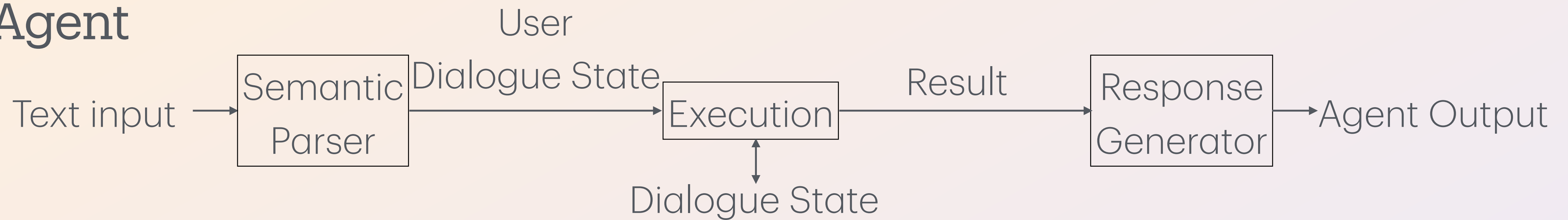
```
RecipeViewImpl = (recipe: Recipe) => {
    return (
        <div>
            <img image={recipe.img}/>
            <love loved={recipe.loved}
                onClick={()=>recipe.love()}/>
            <div> {recipe.name} </div>
        </div>
    )
}

export RecipeView = GenieInterface("Recipe",
    RecipeViewImpl)
```

Components Annotations:
Which components represent
which state classes

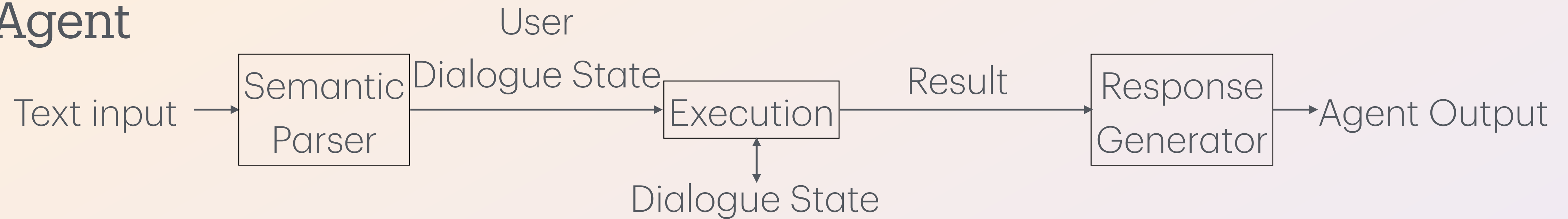
Recall the Agent Architecture (No Policies)

Agent

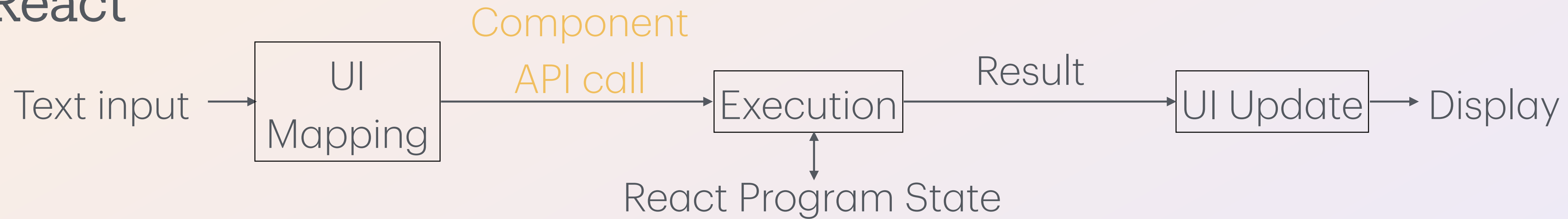


React has a Similar Architecture

Agent

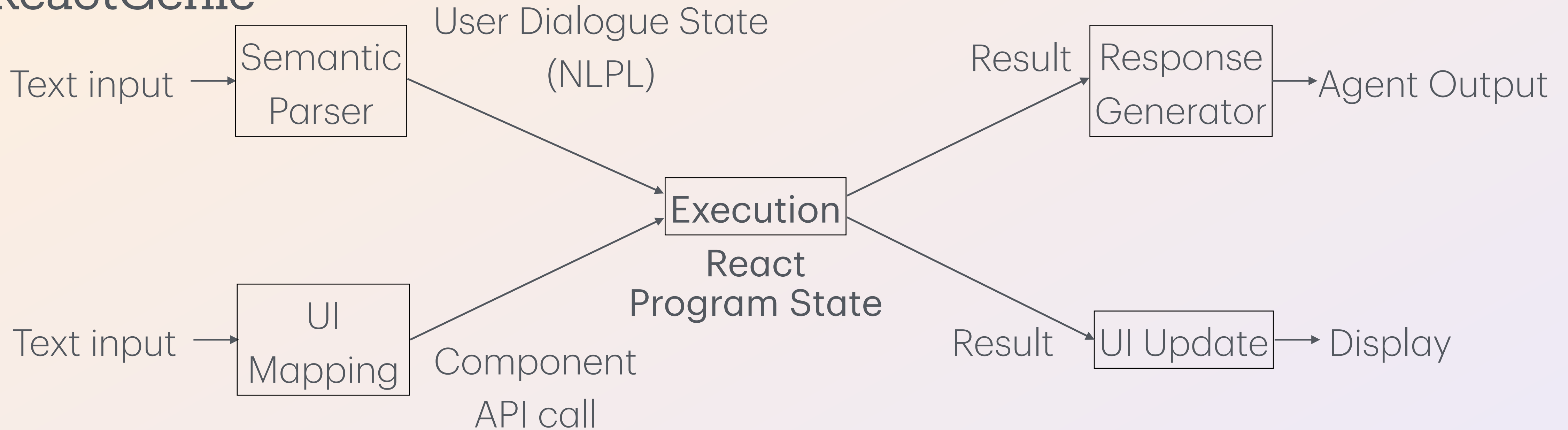


React



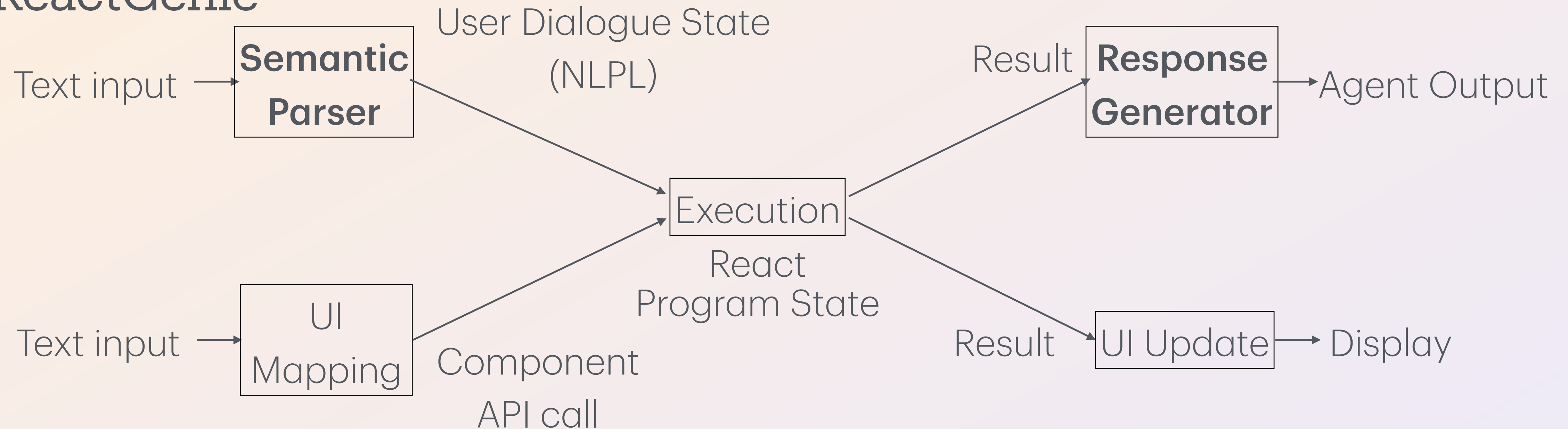
Multimodal Agent Architecture

ReactGenie



Semantic Parser + Response Generator

ReactGenie



Revisiting NLPL

NLPL (natural-language
programming language)



Make this text box bold in the slide master.

```
Slide.Current().getSlideMaster().matching(field:.id,value:Shape.Current().id)  
    .textFrame.textRange.font.setBold(bold: true)
```

Make the border of this shape with little dots.

```
Shape.Current().lineFormat.setDashStyle(dashStyle:"RoundDot")
```

Make everything right aligned on this slide.

```
Slide.Current().getShapes().textFrame.textRange.paragraphFormat.  
    setHorizontalAlignment(horizontalAlignment:"Right")
```

Make every shape on this slide above this yellow.

```
Slide.Current().getShapes().between(field:.top,to:Shape.Current().top).  
    fill.setForegroundColor(color:"yellow")
```


LLM-Based Semantic Parser

// Here are all the functions that we have

```
class Restaurant {
  string name;
  string address;
  string cuisine;
  float rating;

  // All active restaurants
  static Restaurant[] all();

  // The current restaurants
  static Restaurant current();

  // Get a list of foods representing the menu from a restaurant
  Food[] menu;

  // Book reservations on date
  Reservation get_reservation(date: DateTime)
}
```

Declaration

// Examples:

```
user: get me the best restaurant in palo alto
agent: Restaurant.all().matching(field: .address, value: "palo alto")...
```

Few-shot examples

```
// User interaction
user: order the same burger that I ordered at mcDonald last time
```

Current interaction

```
parsed: Order.current.addFoods(foods: Order.all().matching...
```

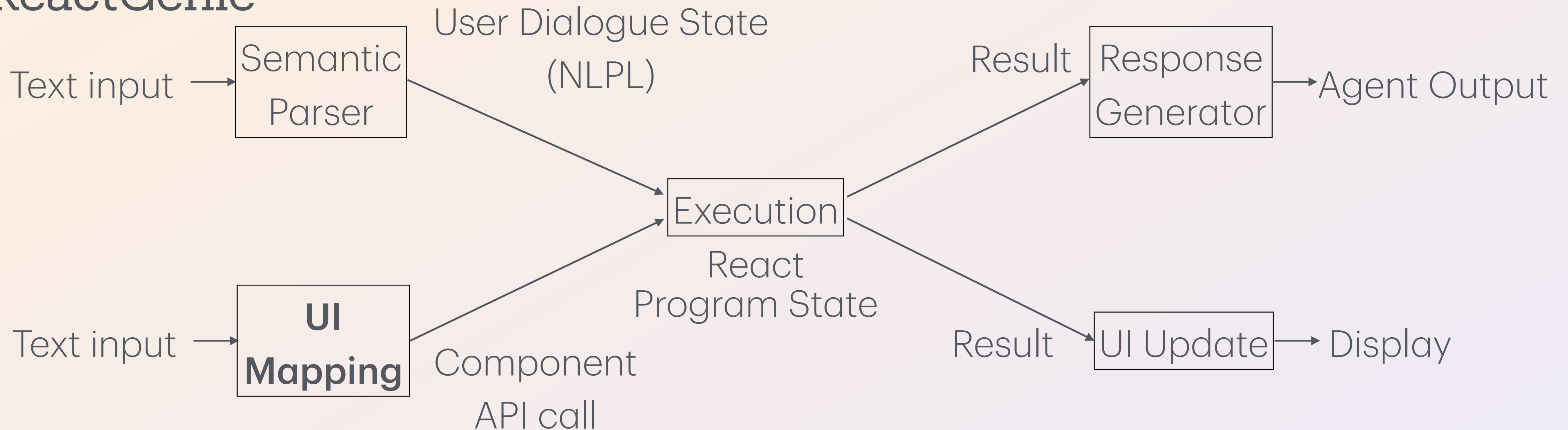
Parsed result

LLM-Based Response Generator

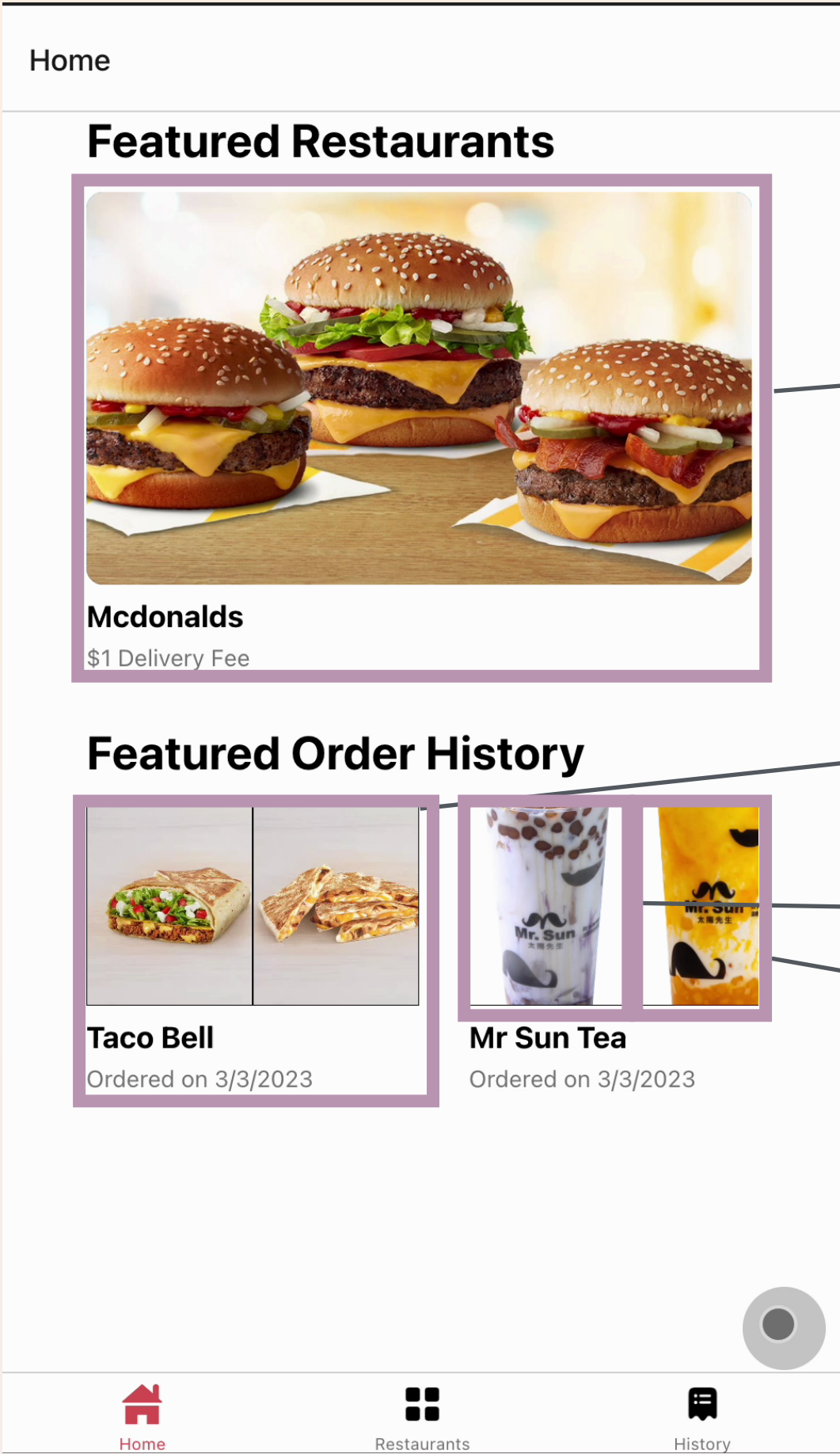
<pre>// Here are all the functions that we have class Restaurant { string name; string address; string cuisine; float rating; // All active restaurants static Restaurant[] all(); // The current restaurants static Restaurant current(); // Get a list of foods representing the menu from a restaurant Food[] menu; // Book reservations on date Reservation get_reservation(date: DateTime) }</pre>	Declaration
<pre>// Generate concise voice feedback for the user's command</pre>	Instructions
<pre>// User interaction user: order the same burger that I ordered at mcDonald last time parsed: Order.current.addFoods(foods: Order.all().matching... execution_result: {"type": "Order", "items": [{"type": "FoodItem", "name": "Hamburger"}, {"type": "FoodItem", "name": "Fries"}]}</pre>	Current interaction
<pre>response: Your order with a hamburger and fries has been placed.</pre>	Generated Response

How to Handle Hybrid Inputs

ReactGenie



UI Mapping



Restaurant
(name: "Mcdonald")

Order(date: "3/3/2023")

FoodItem
(name: "boba tea")

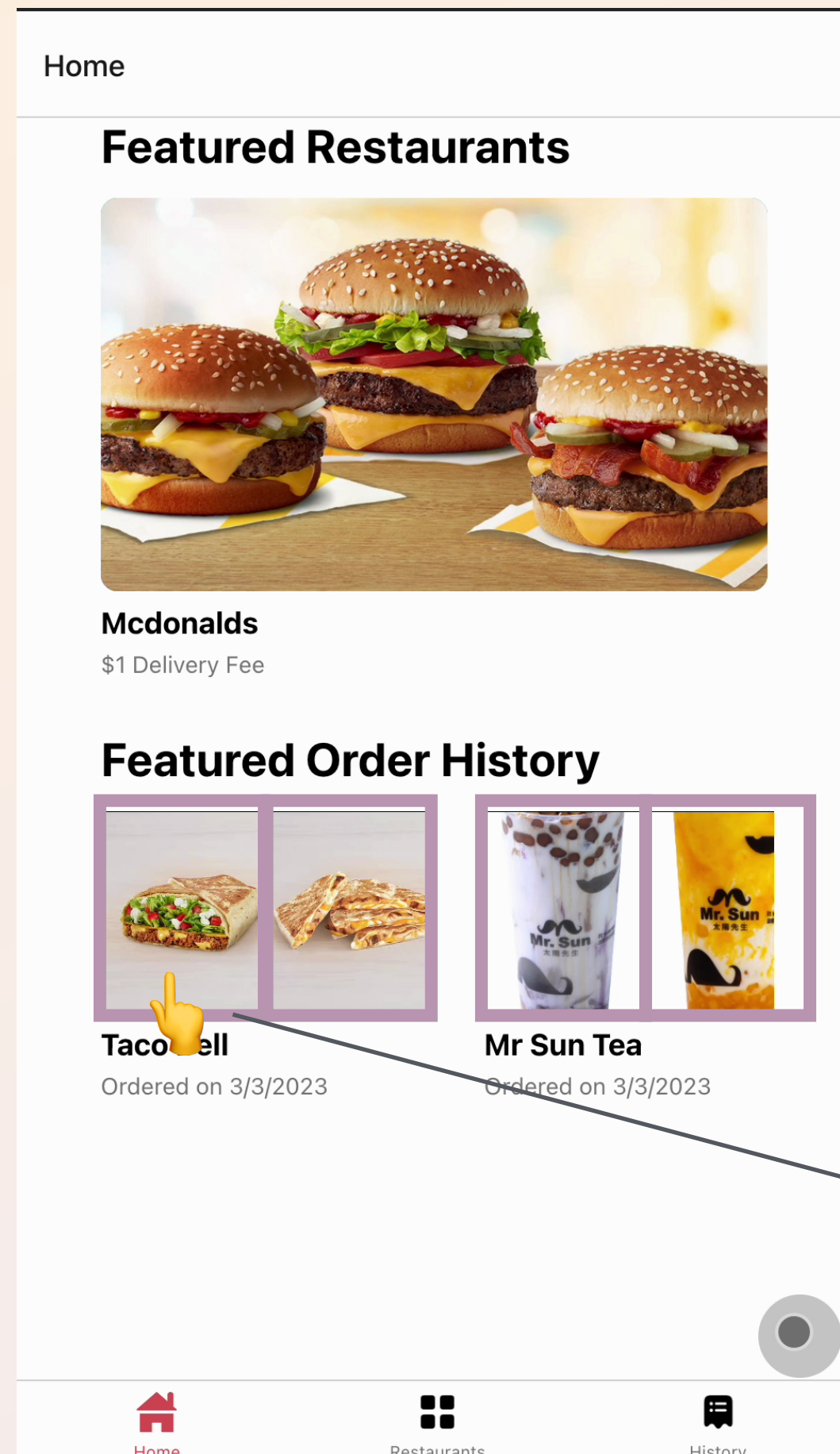
FoodItem
(name: "mango drink")

Programming
Objects

View

ReactGenie

How to Handle Hybrid Inputs



User: "Reorder this food"



```
Order.GetActiveOrder().addFood(food:[FoodItem.Current()])
```



Unresolved UI reference `FoodItem.Current()`



UI Mapping: Find `FoodItem` closest to the click point



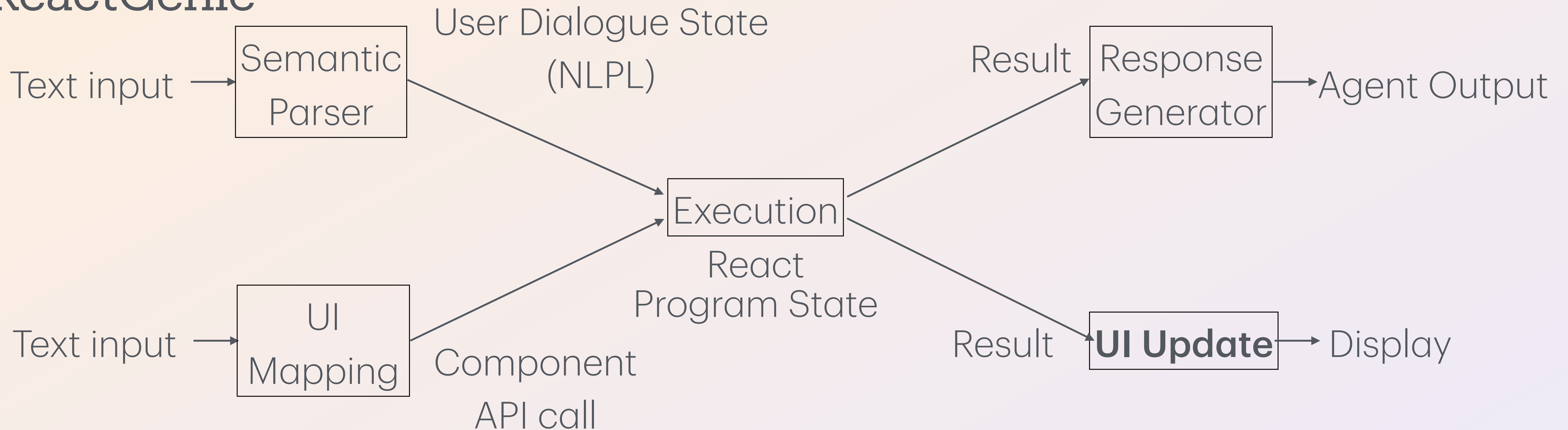
```
FoodItem(name:"CrunchWrap")
```



Continue execution

How to Navigate with Voice Commands

ReactGenie



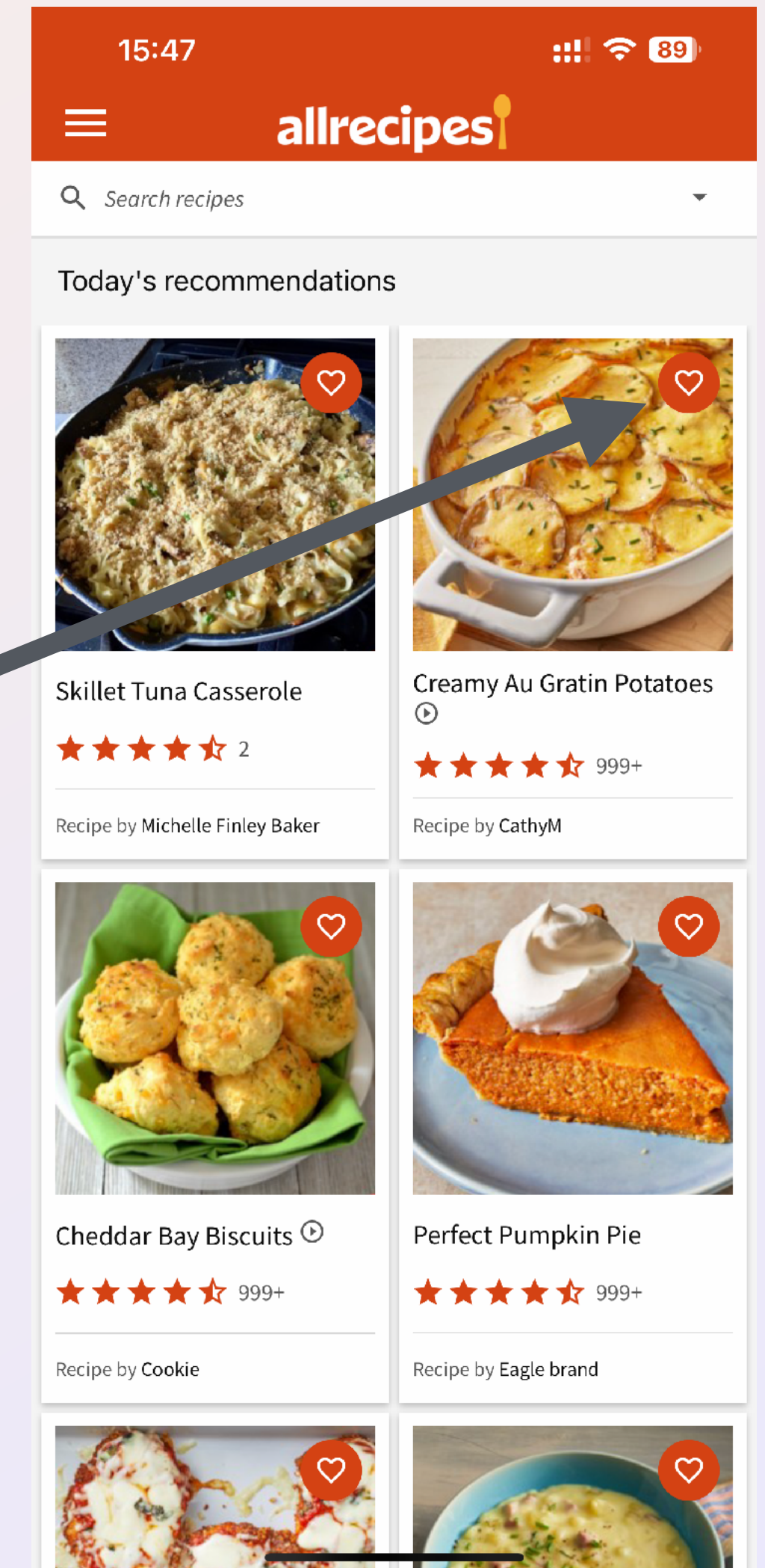
UI Update 1: Object is on UI → React takes care of it

"I love the Creamy Potatoes recipe!"

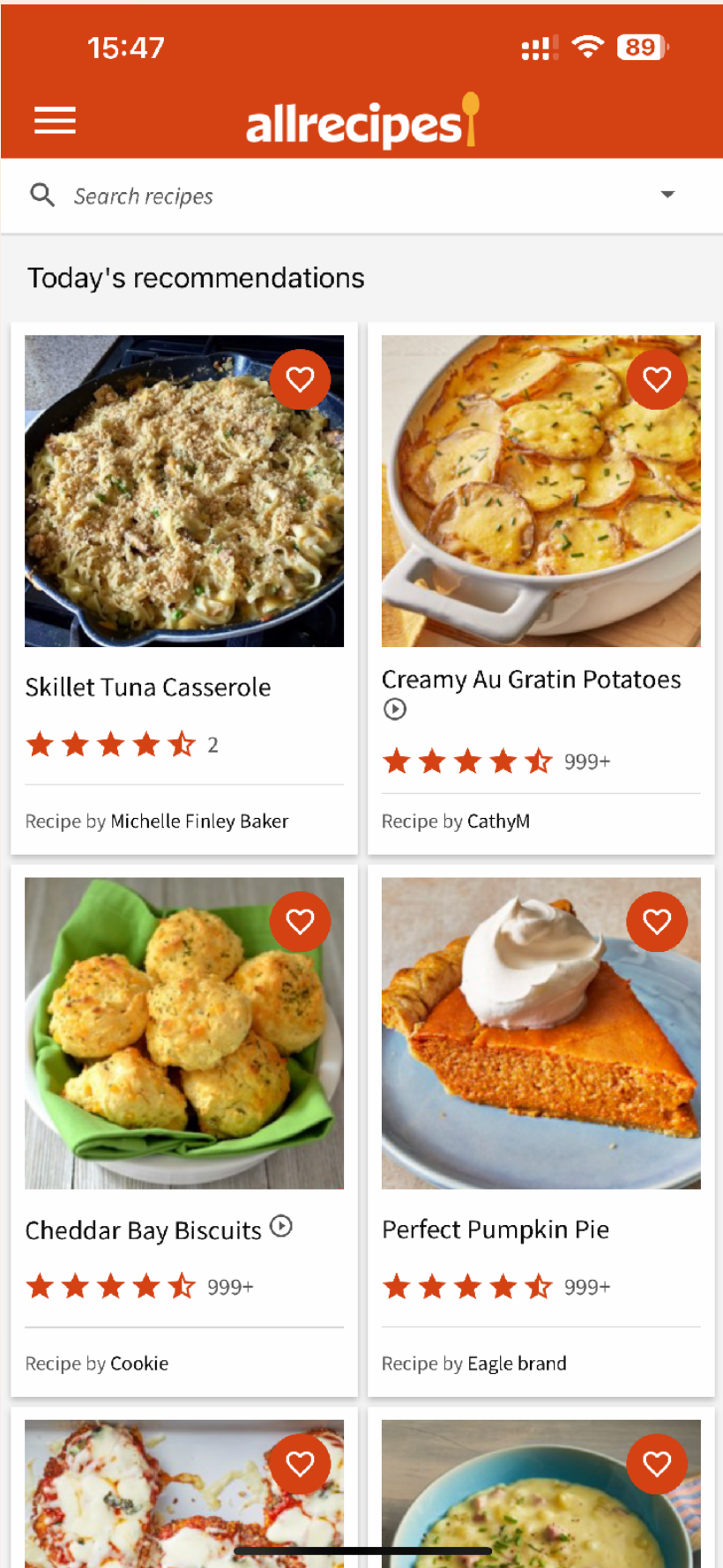
Recipe(name: "Creamy"). love()

recipe.loved = true

```
RecipeViewImpl = (recipe: Recipe) => {  
  return (  
    <div>  
      <img image={recipe.img}/>  
      <love loved={recipe.loved}  
        onClick={()=>recipe.love()}/>  
      <div> {recipe.name} </div>  
    </div>  
  )  
}
```

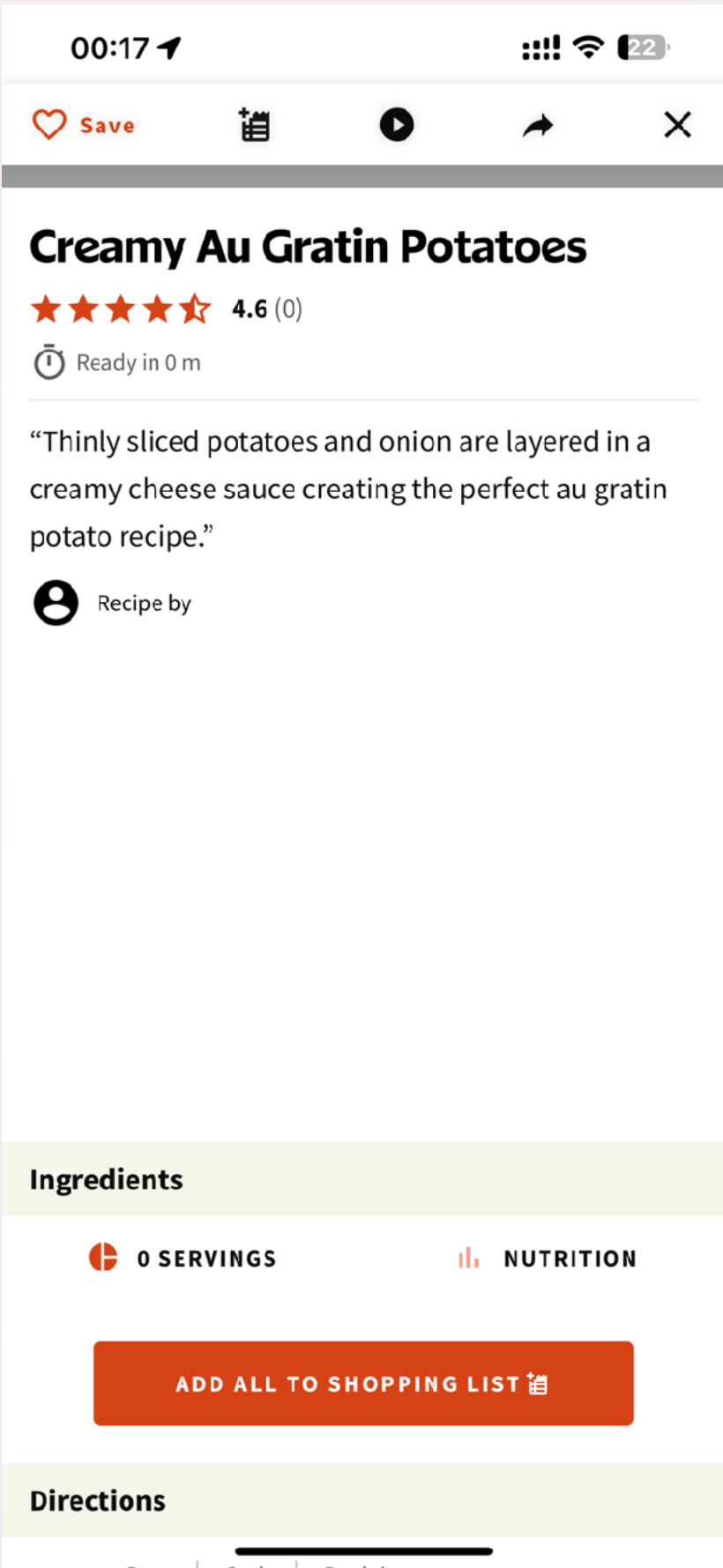


UI Update 2: Object not on UI → Navigate to the page



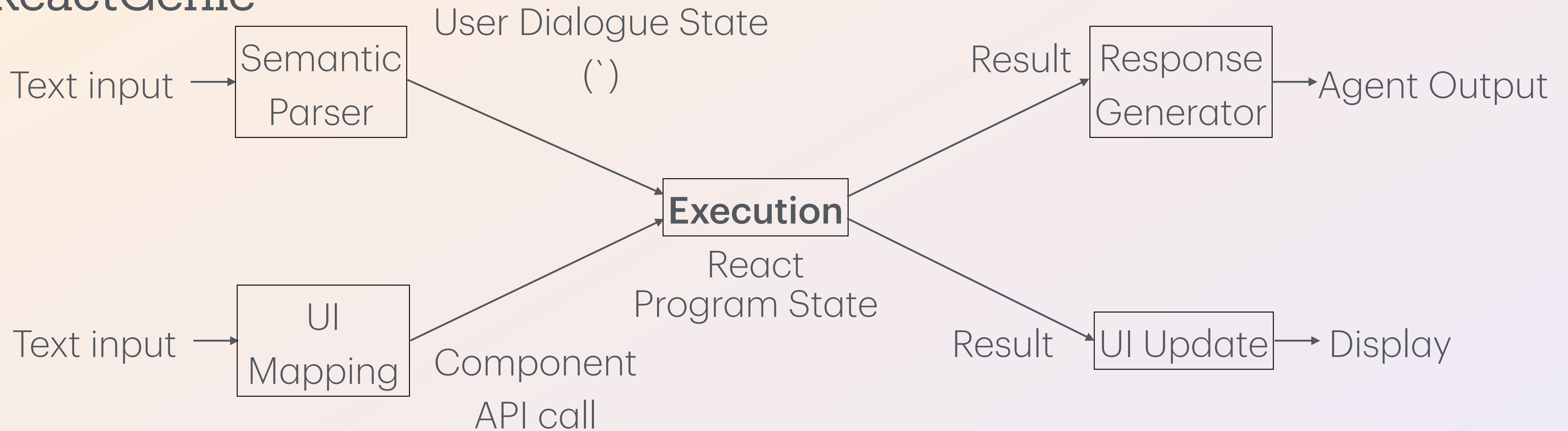
“Show me the Creamy Potato recipe!”

Recipe(name: “Creamy Potato”)



Execution

ReactGenie

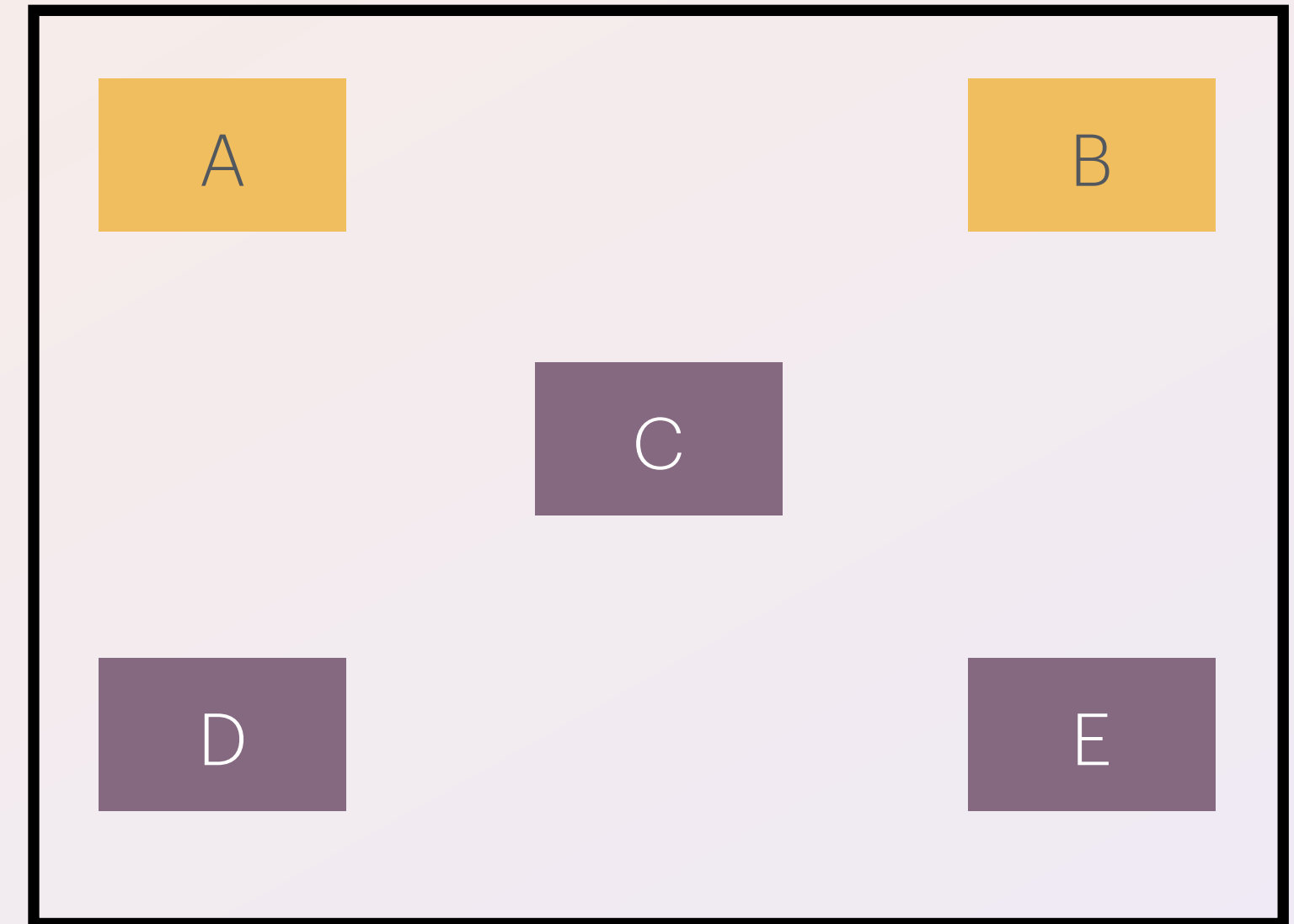


Execution

“Make everything above this yellow”

```
Slide.Current().getShapes().  
between(field:.top,to:Shape.Current().top).  
fill.setForegroundColor(color:"yellow")
```

- Slide.Current()
 - Slide(id: 1)
- Slide.Current().getShapes()
 - [Shape(text: "A"), Shape(text: "B"), Shape(text: "C"), Shape(text: "D"), Shape(text: "E")]
- Slide.Current().getShapes().between(field:.top, to:Shape.Current().top)
 - 100
- Slide.Current().getShapes().between(field:.top, to:Shape.Current().top)
 - [Shape(text: "A"), Shape(text: "B")]
- Slide.Current().getShapes().between(field:.top, to:Shape.Current().top).fill.setForegroundColor(color:"yellow")
 - [Fill(), Fill()]



ReactGenie = React + Annotations

Semantic
Parser

Execution

Response
Generator

UI
Mapping

UI Update

```
@DataClass()
class Recipe: GenieClass {
    @GenieProperty()
    name: String;
    img: Image;
    @GenieProperty()
    loved: boolean;

    @GenieFunction()
    love(): void {
        this.loved = true;
    }
}
```

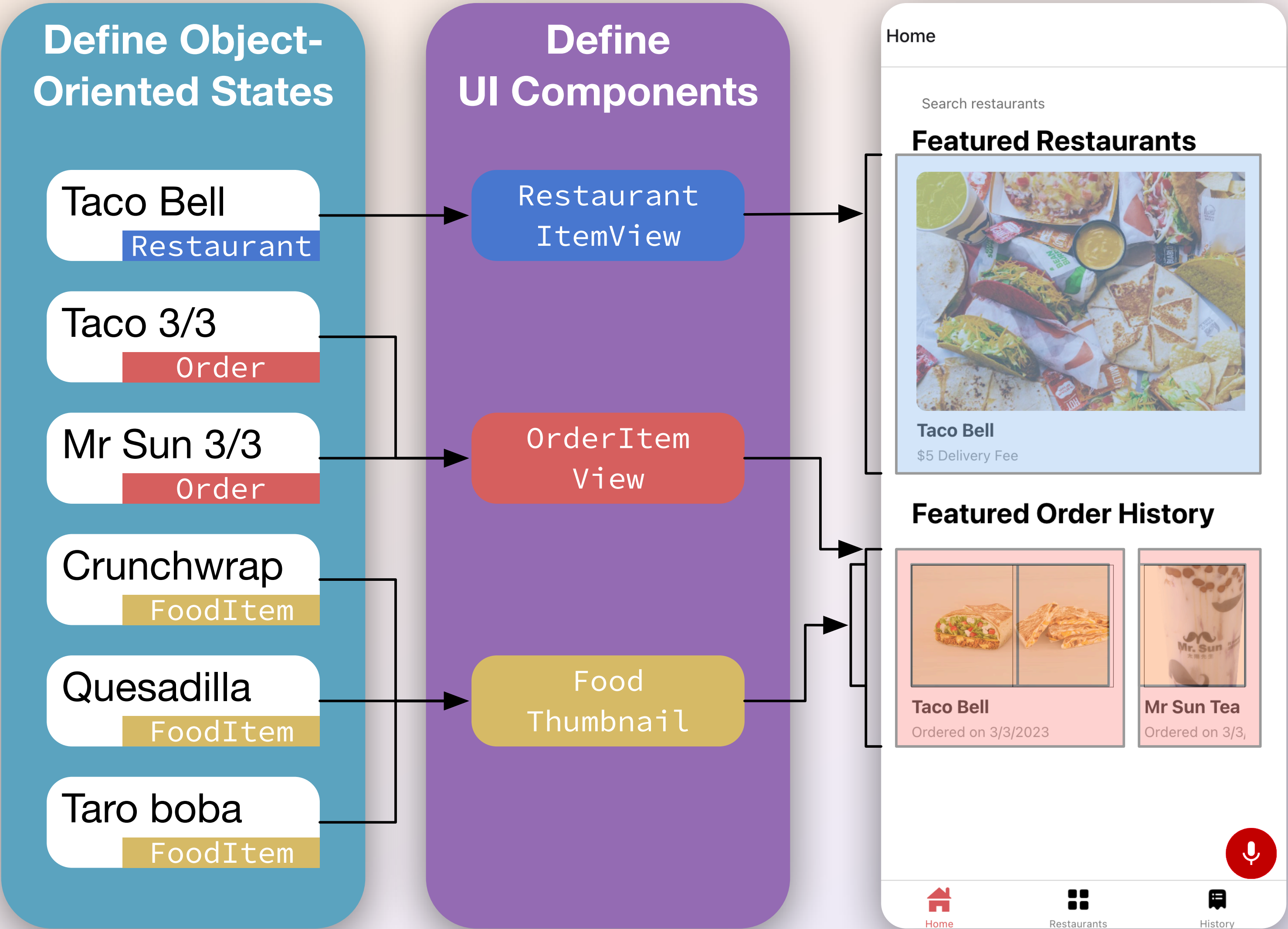
State Annotations:
Which class/property/function
can be accessed with voice

```
RecipeViewImpl = (recipe: Recipe) => {
    return (
        <div>
            <img image={recipe.img}/>
            <love loved={recipe.loved}/>
            <div> {recipe.name} </div>
        </div>
    )
}
```

```
RecipeView = GenieInterface("Recipe",
    RecipeViewImpl)
```

Components Annotations:
Which components represent
which state classes

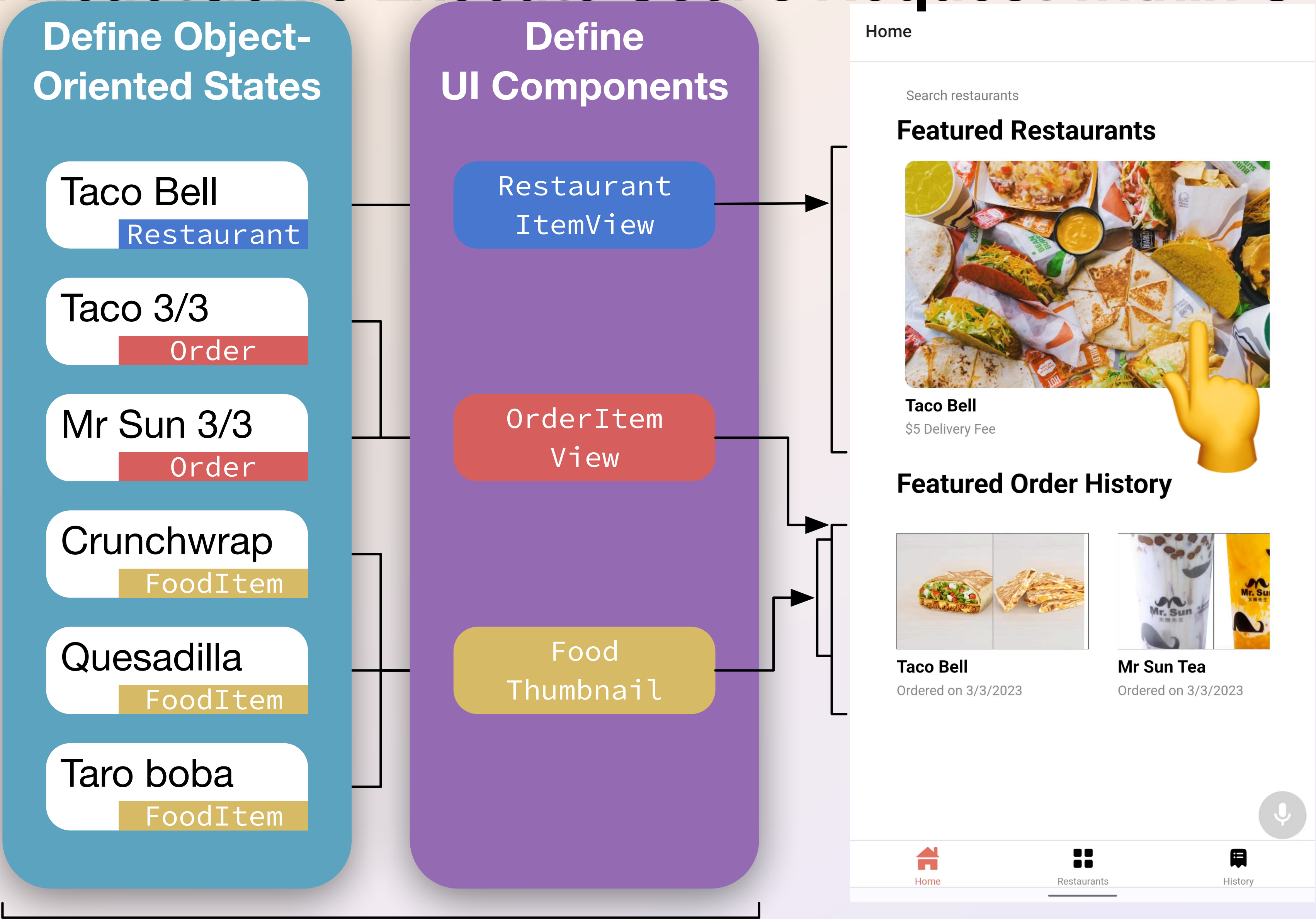
Recap: ReactGenie Uses Declarative UI Architecture for Ease of Development



Recap: ReactGenie Use Annotations for Multimodal Integration

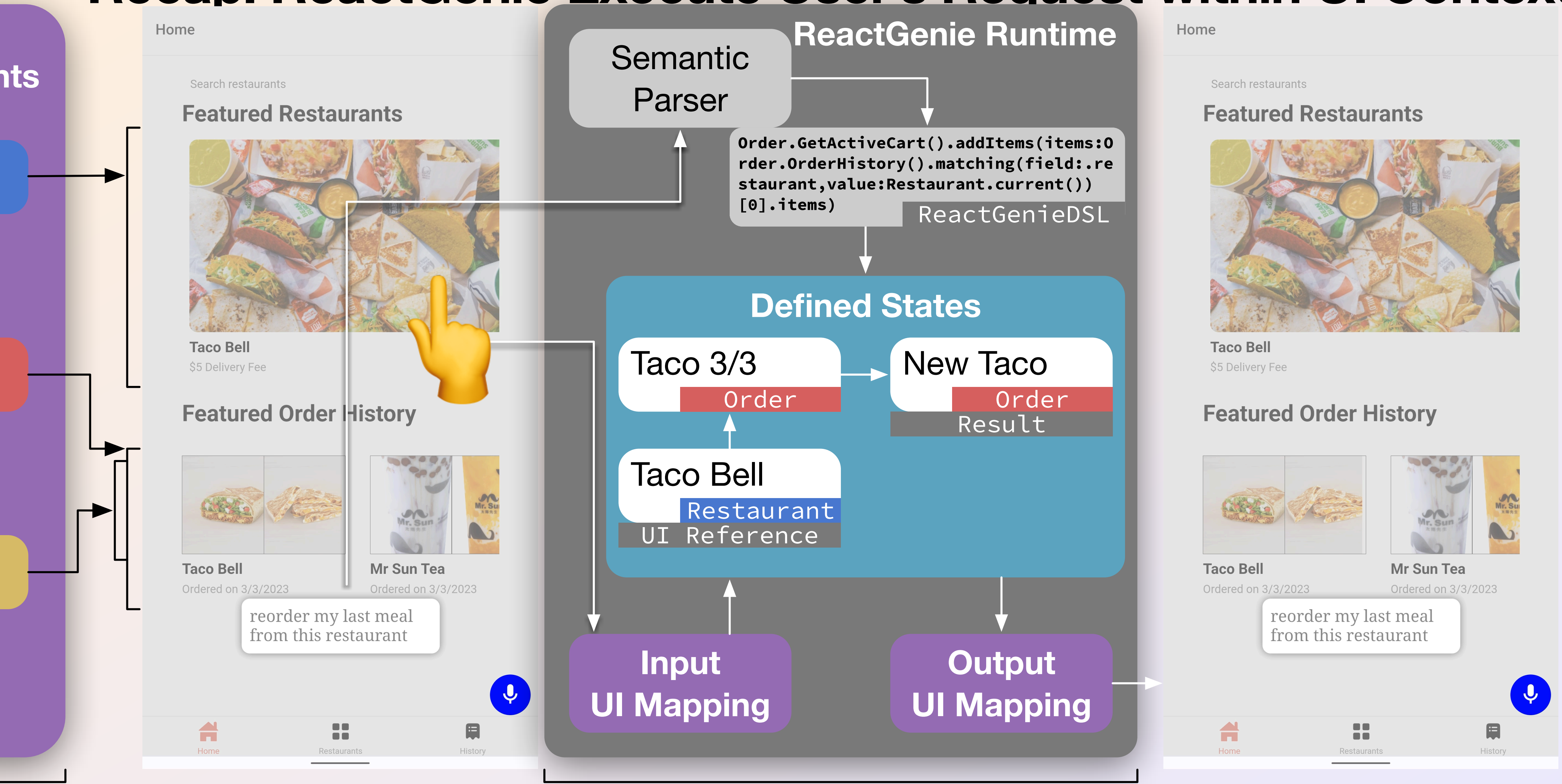
```
@GenieClass("Past order or a shopping cart")
class Order extends DataClass {
  @GenieKey()
  public orderId: string;
  @GenieProperty("Items in the order")
  public orderItems: FoodItem[];
  constructor({orderId, orderItems}: {orderId: string, orderItems: FoodItem[]}) {
    super({orderId, orderItems}); this.orderId = orderId; this.orderItems = orderItems;
  }
  @GenieFunction()
  static All(): Order[] {
    return fetchOrdersFromServer();
  }
  @GenieFunction("Create a new order")
  static CreateOrder(): Order {
    return new Order({orderId: randomId(), orderItems: []});
  }
  @GenieFunction("Add an item to the order")
  addItem({foodItem}: {foodItem: FoodItem}) {
    this.orderItems.push(foodItem); updateServer();
  }
}
```


Recap: ReactGenie Execute User's Request within UI Context



Developer-Coded GUI

Recap: ReactGenie Execute User's Request within UI Context

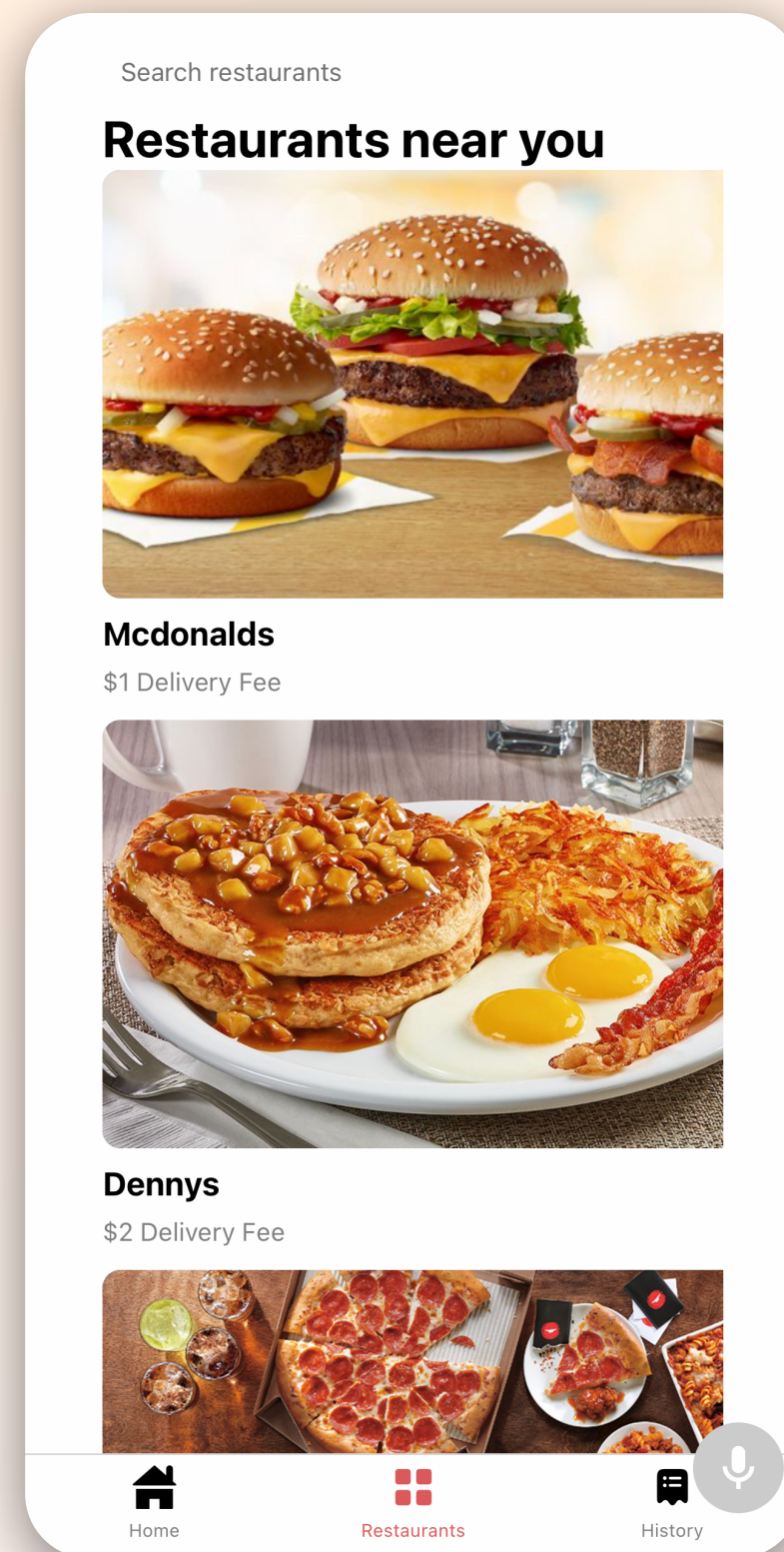


Generated Multimodal UI

How good is ReactGenie as a framework?

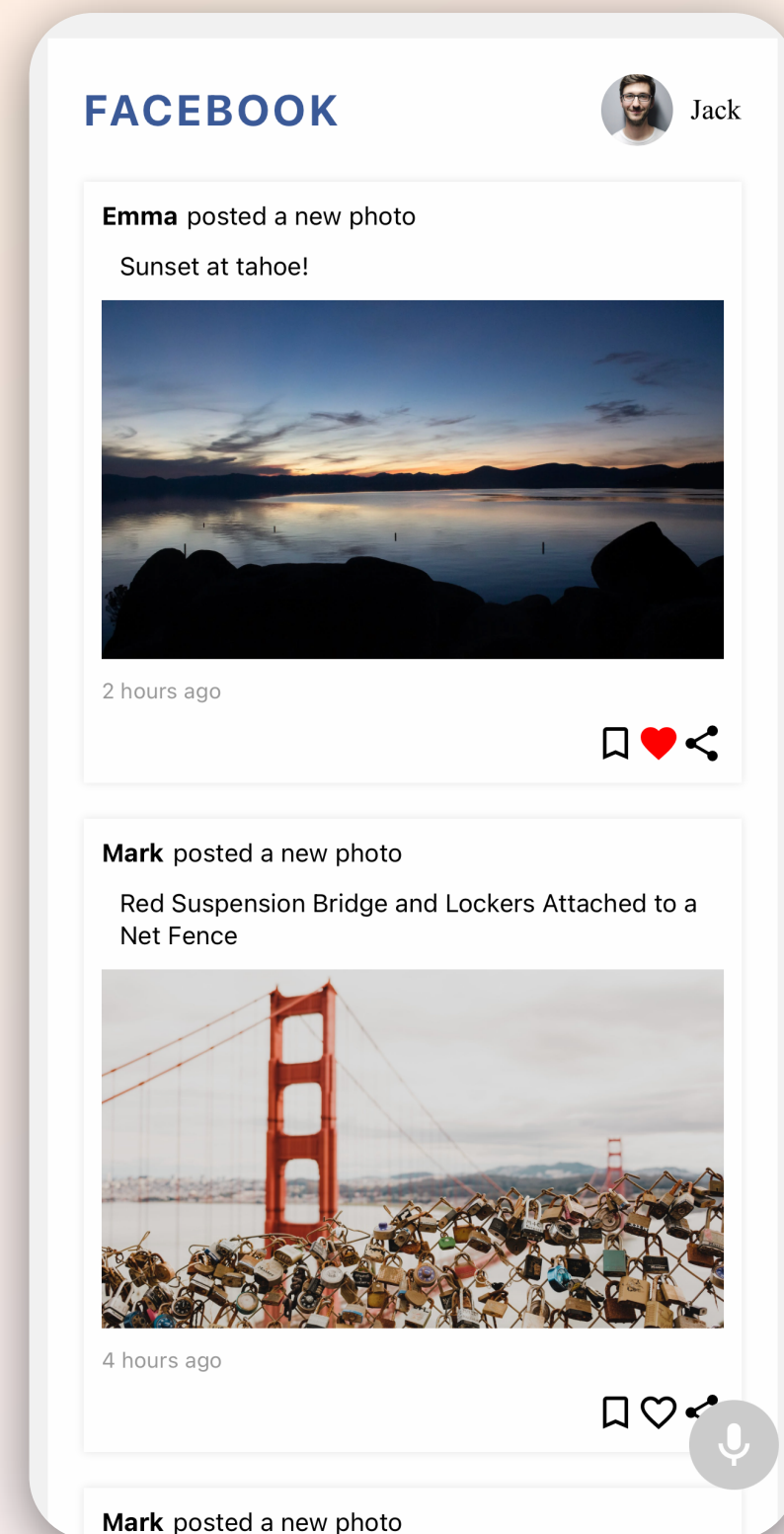
- For developers:
 - D-RQ1: Assessing the expressiveness of ReactGenie
 - D-RQ2: Development time for expert developers
 - D-RQ3: Ease of learning and usability for novice developers

We built three apps to demonstrate expressiveness (F-RQ1)



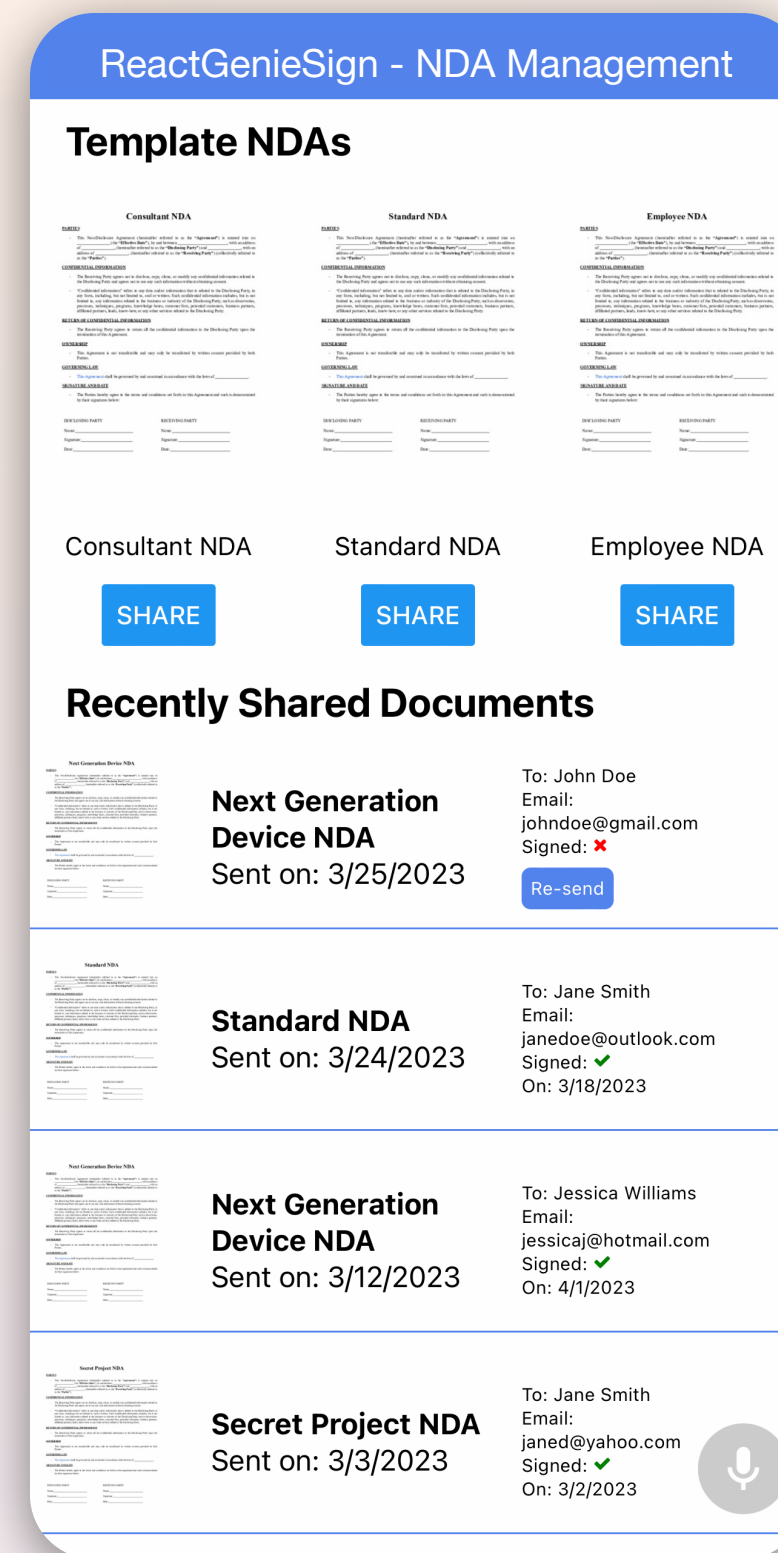
ReactGenieFoodOrdering

Add three of this to my cart



ReactGenieSocial

Show me posts from Mark that I have liked before.

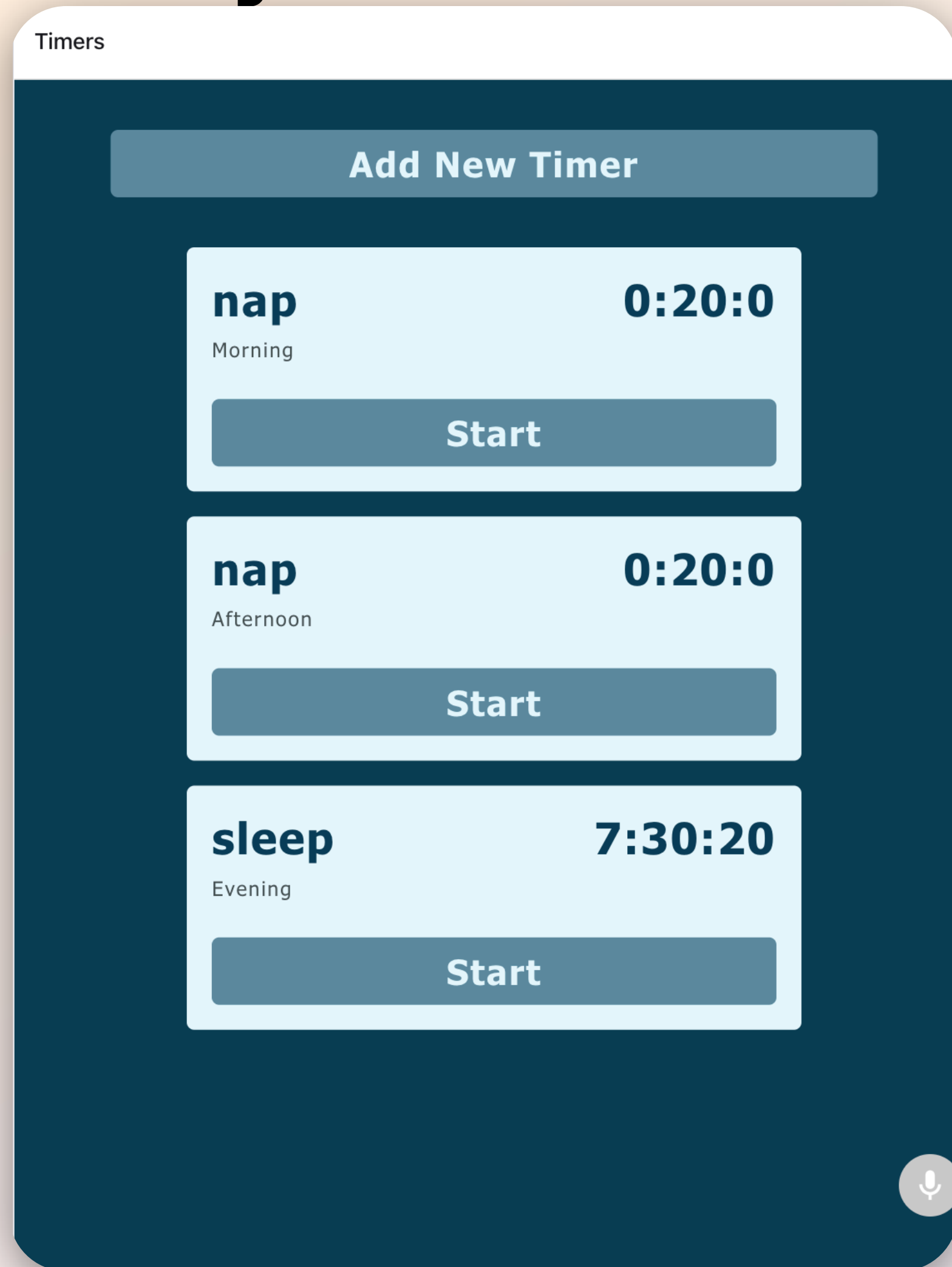


ReactGenieSign

Only show me request through this email

- Only 5% of the code (annotations) must be written to handle multimodal interactions.

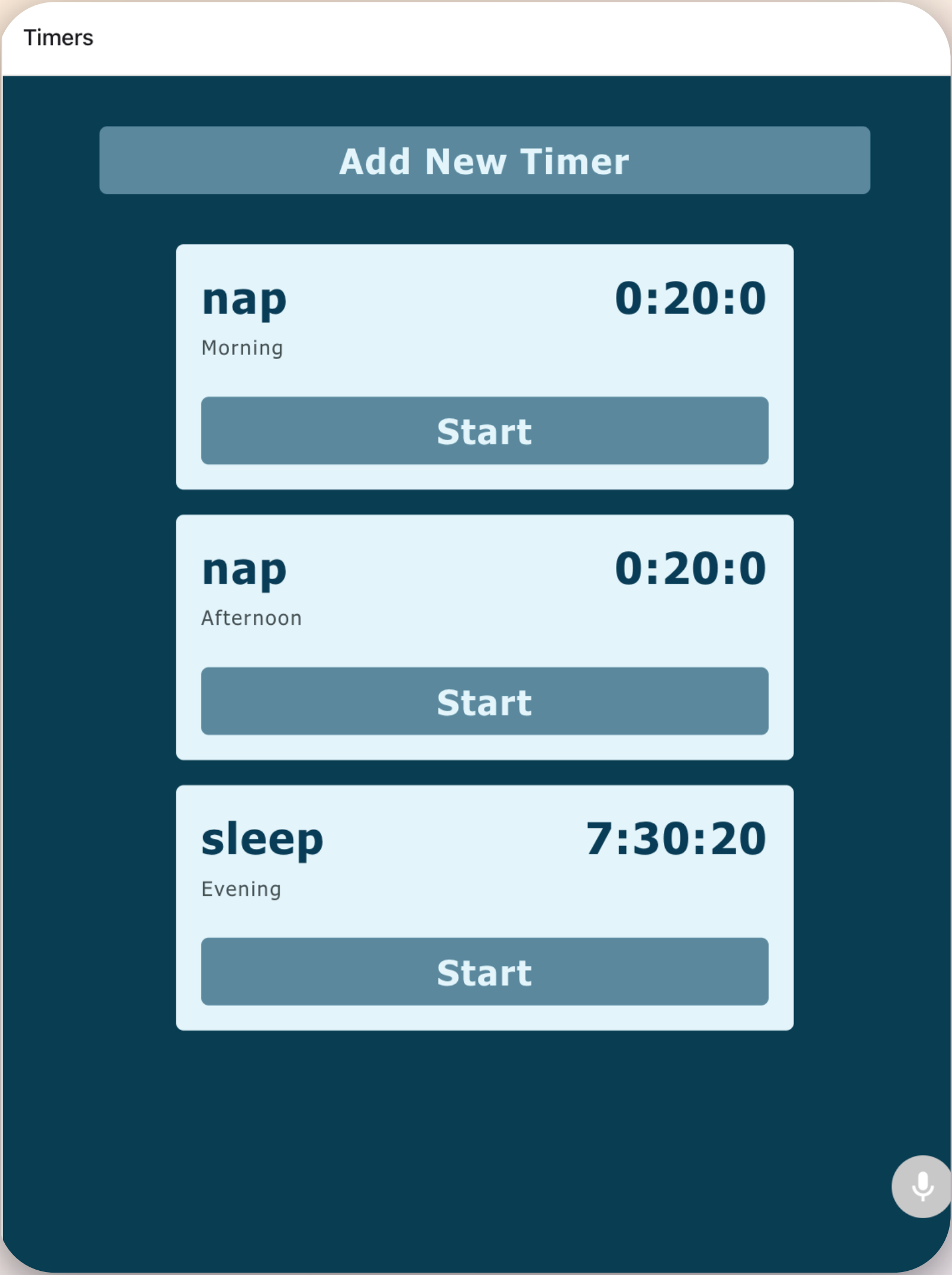
A small ReactGenie demo app that can be built within a lab study: ReactGenie Timer



ReactGenieTimer

- User can:
 - Create, start, and pause timer with voice.
 - Start/stop timer of a certain category
 - Filter timer by remaining time
 - ...

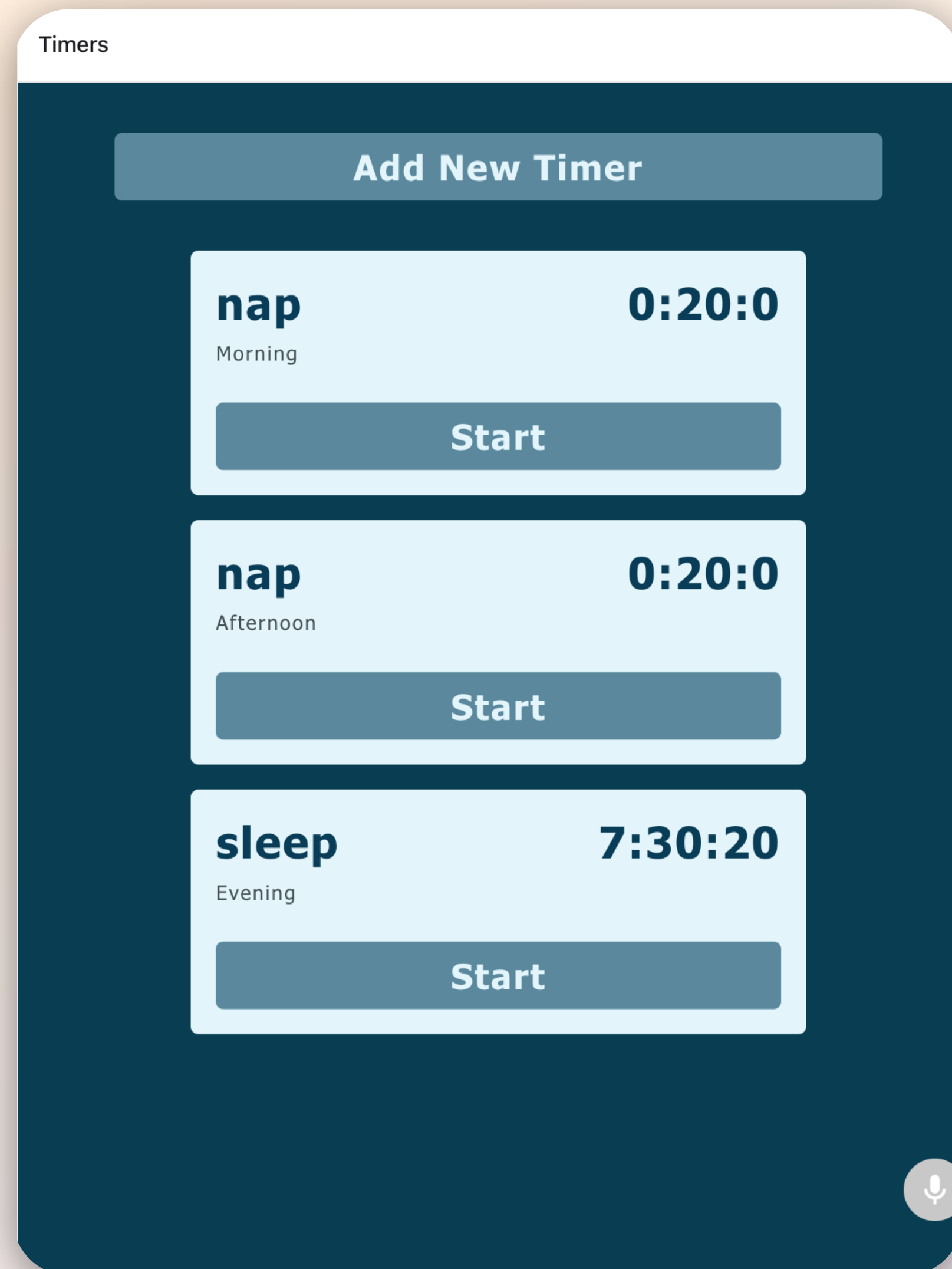
We asked an expert developer to build an App in ReactGenie and GPT Function Calling (F-RQ2)



ReactGenieTimer

Metric	ReactGenie	GPT-3 Function Calling
	Less time	
Time to Develop (minutes)	45 Less code	177
Additional Lines of Code	159 More features	523
Features Supported	Touch, Complex Commands, Navigation	Limited Support

Novice Developer Study on ReactGenie Usability (F-RQ3)



ReactGenieTimer

- Study Design:
Learn with tutorial ->
Construct timer app on GUI boilerplate
- High comprehension of framework functionality (99% of questions correct)
- Fast completion time: 67.3 minutes
- Positive feedback on ease of use
- Many participants asked to use ReactGenie in real-life applications

How good is ReactGenie as a framework?

- For users:
 - U-RQ1: Parser performance with natural language commands
 - U-RQ2: Usability and efficiency of multimodal UIs generated by ReactGenie

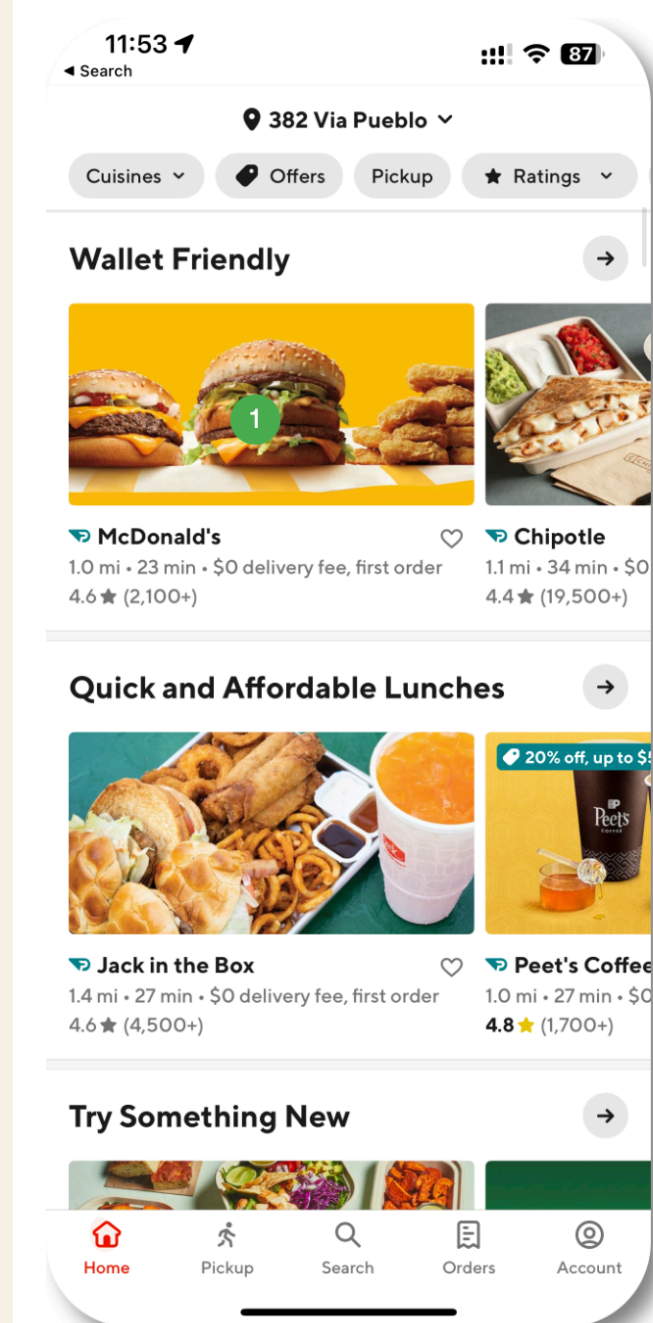
Evaluate parser's effectiveness with commands from crowd workers (U-RQ1)

How would you interact with this future food-ordering app? (1/6)

What would you say to the app? (type complete sentence)

What have I ordered from this restaurant last time?

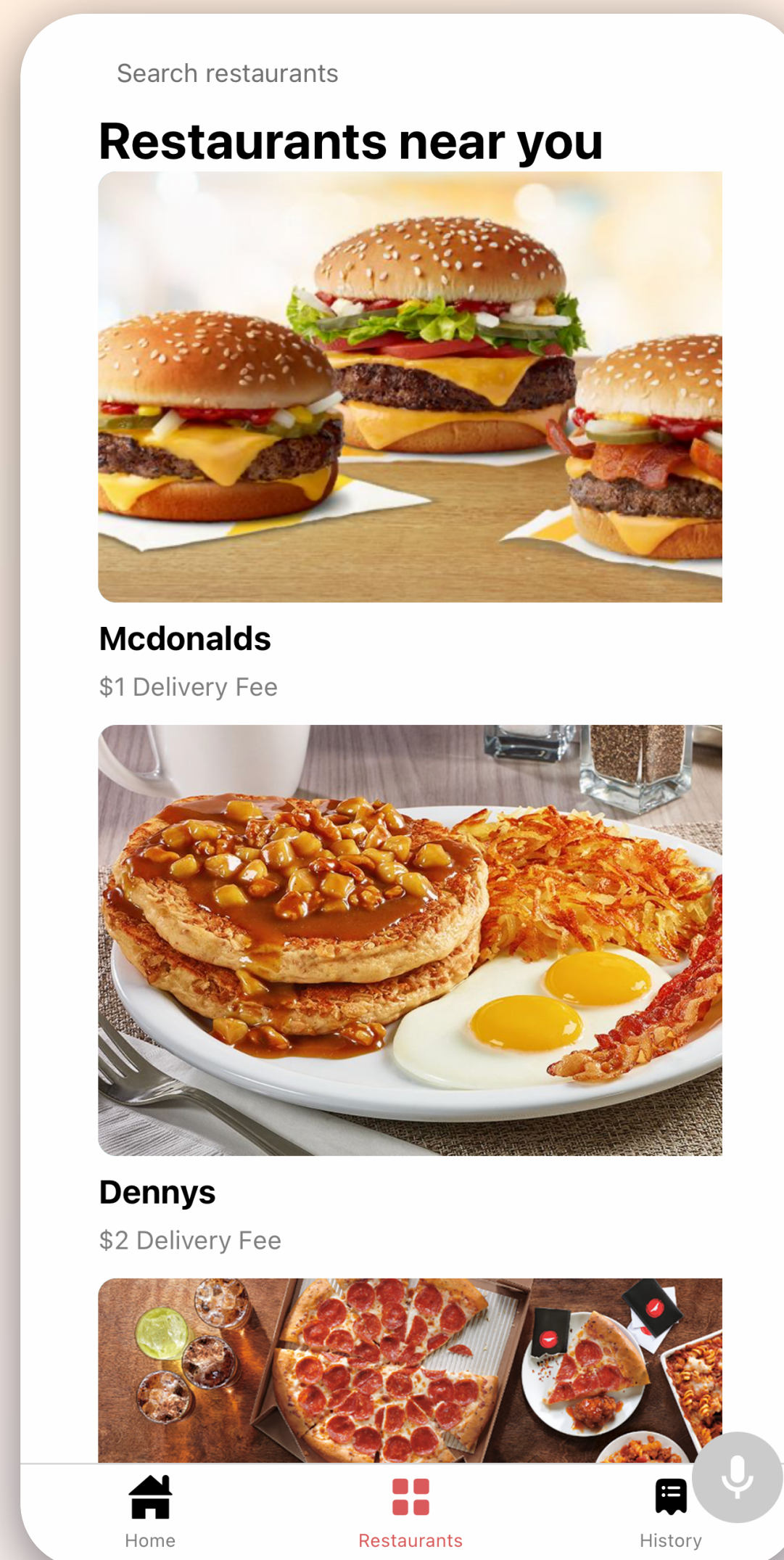
If you want to point to the screen, where would you point? (If you choose not to, just click next.)



Clear Submission

- Participants provided with app screenshots and videos to prompt commands
- Results:
 - 172 rich multimodal commands
 - Parser Accuracy:
 - 101 supported commands:
parsed correctly **91%**
 - 71 unsupported commands:
generated sensible NLPL **53%**

User Experience with ReactGenie-Generated UIs (U-RQ2)



ReactGenieFoodOrdering

- Compare user performance and experience using multimodal UIs vs. GUI-only
- Study Design:
 - Within-subject design with 16 participants
 - Multimodal UI vs. GUI-only
- Result: ReactGenie
 - Saved 40% time ($p=0.0004$)
 - Lower Cognitive Load ($p=0.013$)
 - Higher Usability ($p=0.031$)
- Participant Preferences:
 - Strong preference on MMI: 11/16

Lecture Goals

- Why do we need multimodal interactions?
- Three problems for multimodal app development
- ReactGenie: a multimodal app development framework
- **GenieWizard: addressing the multimodal app's feature discovery problem**

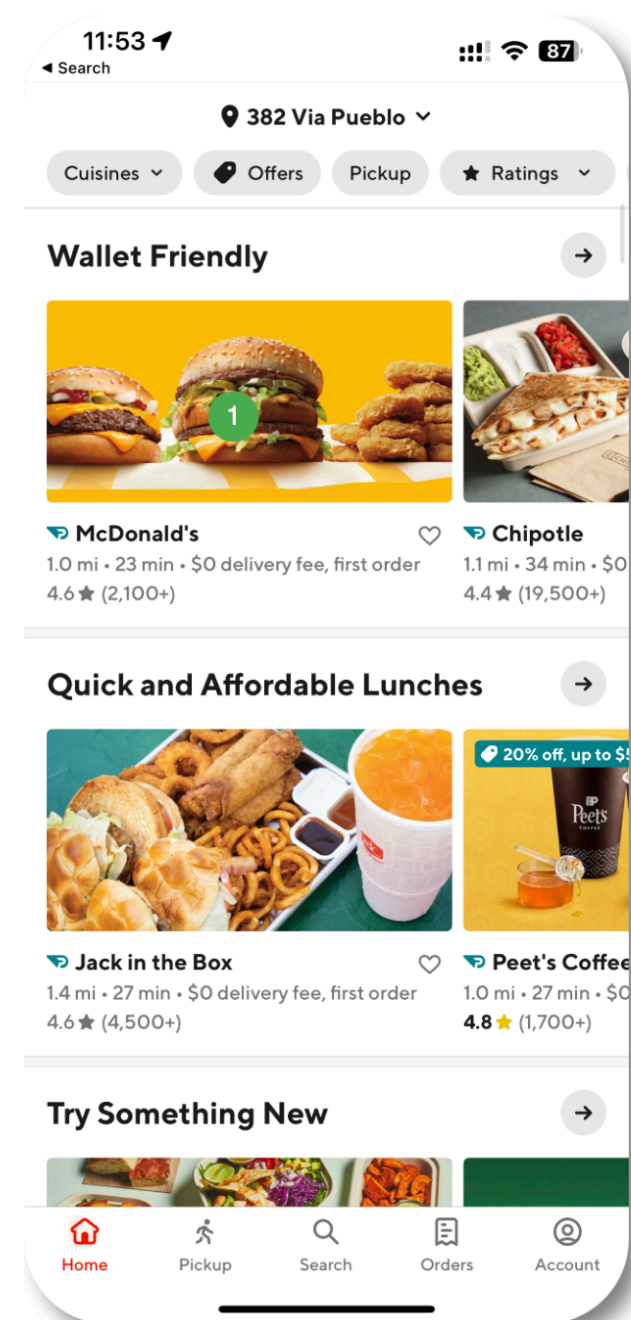
ReactGenie study result: Amazing on supported commands, How about unsupported commands?

How would you interact with this future food-ordering app? (1/6)

What would you say to the app? (type complete sentence)

What have I ordered from this restaurant last time?

If you want to point to the screen, where would you point? (If you choose not to, just click next.)



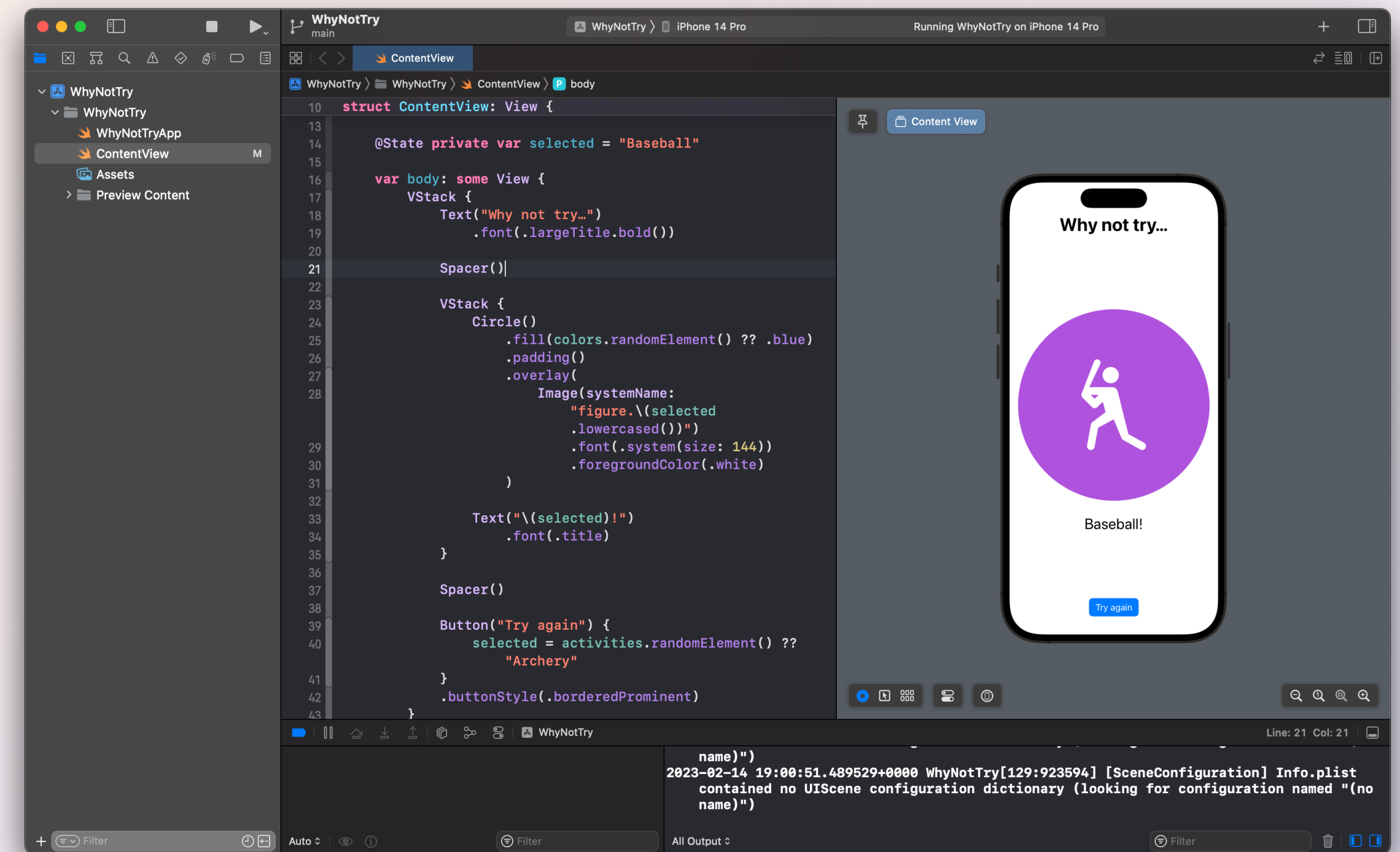
- Participants provided with app screenshots and videos to prompt commands
- Results:
 - 172 rich multimodal commands
 - Parser Accuracy:
 - 101 supported commands: Works Great
parsed correctly **91%**
 - 71 unsupported commands: 41% Commands
generated sensible NLPL **53%** Unsupported

Sensible: used non-existent functions in a reasonable way

Can we guess what is needed leveraging “hallucination” of LLMs

GenieWizard is and IDE to help developers

- GUI IDE helps developers:
In the same time frame
 - Follow GUI best practices
 - Visualize visual designs



IDE to support feature suggestions early in the dev process

Early Stage State Code

Newly Implemented Features

TS FoodItem.ts M

genie > TS FoodItem.ts > FoodItem > vegetarian

```
11 @GenieProperty("the name of the
12 public name: string;
13 public desc: string;
14 public imageUrl: string;
15 @GenieProperty("the price of th
16 public price: float;
17 @GenieProperty("ingredient of
18 public ingredient: string[];
19 @GenieProperty()
20 public vegetarian: boolean;
21
22 @GenieProperty("only supports a
23 public menuSection: MenuSection
24
25 constructor({id, name, desc, in
```

TS OrderItem.ts M

genie > TS OrderItem.ts > OrderItem > constructor

```
18 this.foodItem = foodItem;
19 this.status = "pending";
20 console.log(foodItem)
21 }
22
23 @GenieFunction("Add topping to
24 addTopping({topping}: {topping:
25 this.topping.push(topping);
26 return this;
27 }
28
29 @GenieFunction("Create a new Or
30 static CreateOrderItem({foodIt
31 const id = OrderItem.All().
32 return OrderItem.CreateOhi
```

GenieWizard

Guidances

DEBUGS

COMMANDS

Generate App Description

Generate Suggested Functions

PROGRESS

Implemented: 55.76 + 20.5%

Will Support: 23.38%

Won't Support: 20.86%

SUGGESTED FUNCTIONS

FoodItem.ingredients (21.14% / 21.14%)

FoodItem.vegetarian (13.01% / 13.01%)

FoodItem.picture (0% / 11.38%)

FoodItem.GetMenuSection() (0% / 9.76%)

FoodItem.contains (0% / 7.32%)

FoodItem.addToppings() (3.25% / 4.88%)

Reservation (0% / 4.07%)

FoodItem.customize() (4.07% / 4.07%)

FoodItem.GetFoodItem(name: "pasta").ingredient

OrderItem.CreateOrderItem(foodItem: FoodItem.All().mat...

Customize my Margherita pizza. Add extra cheese and olives.

FoodItem.GetFoodItem(name: "pasta").ingredient

GenieWizard

Guidances

Progress Tracker

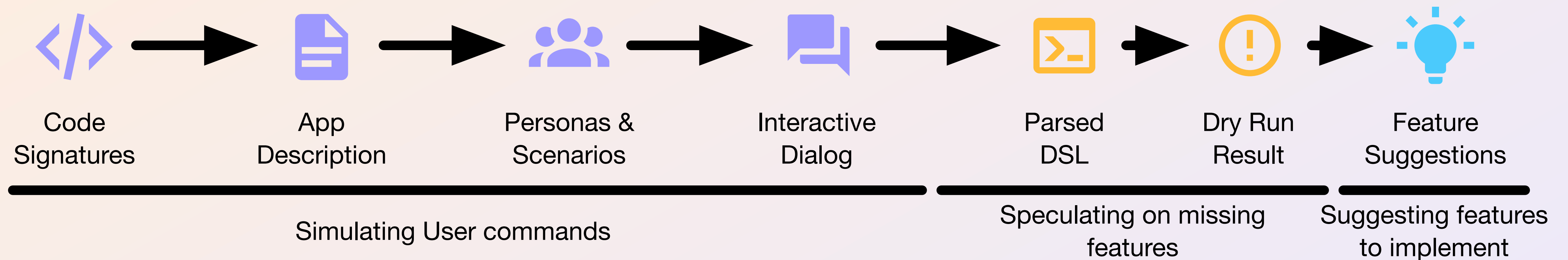
Suggested Features

Unique Features

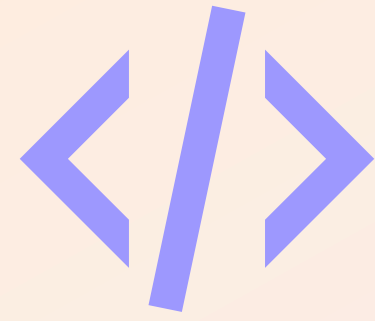
Generated Parsed Commands

Generated User Utterances

GenieWizard Pipeline: Provide feature suggestions matching real user's requests

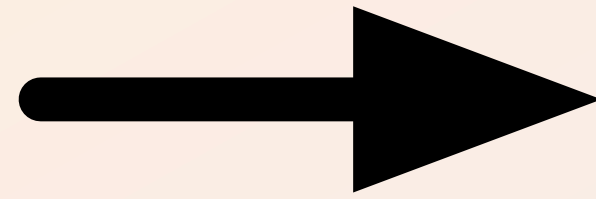


Step 1: Simulating User commands



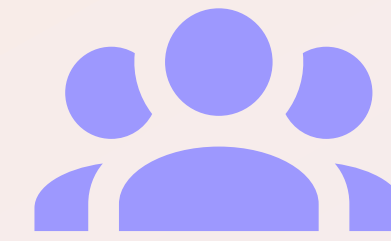
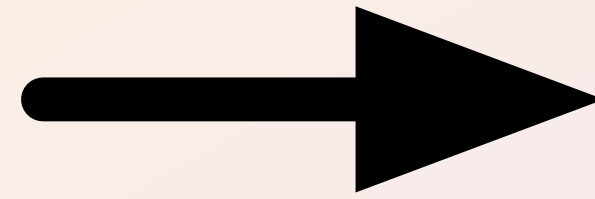
Code
Signatures

```
// A food item
class FoodItem {
    string name;
    float price;
    // the menu section of
    the food
    MenuSection menuSection;
    static FoodItem
    GetFoodItem(name: string);
    static FoodItem[] All();
    static FoodItem
    Current();
    static FoodItem[]
    AllCurrent();
}
```



App
Description

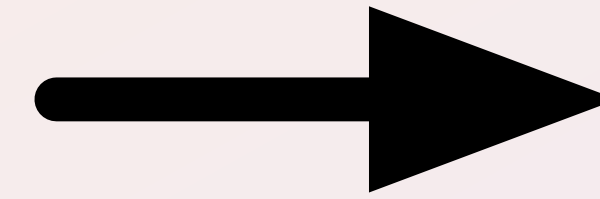
This app is a digital menu designed
for dining in an Italian restaurant.



Personas &
Scenarios

Landon Holt, a 22-year-old male
from Johnson County, Missouri, is a
personal financial advisor. ...

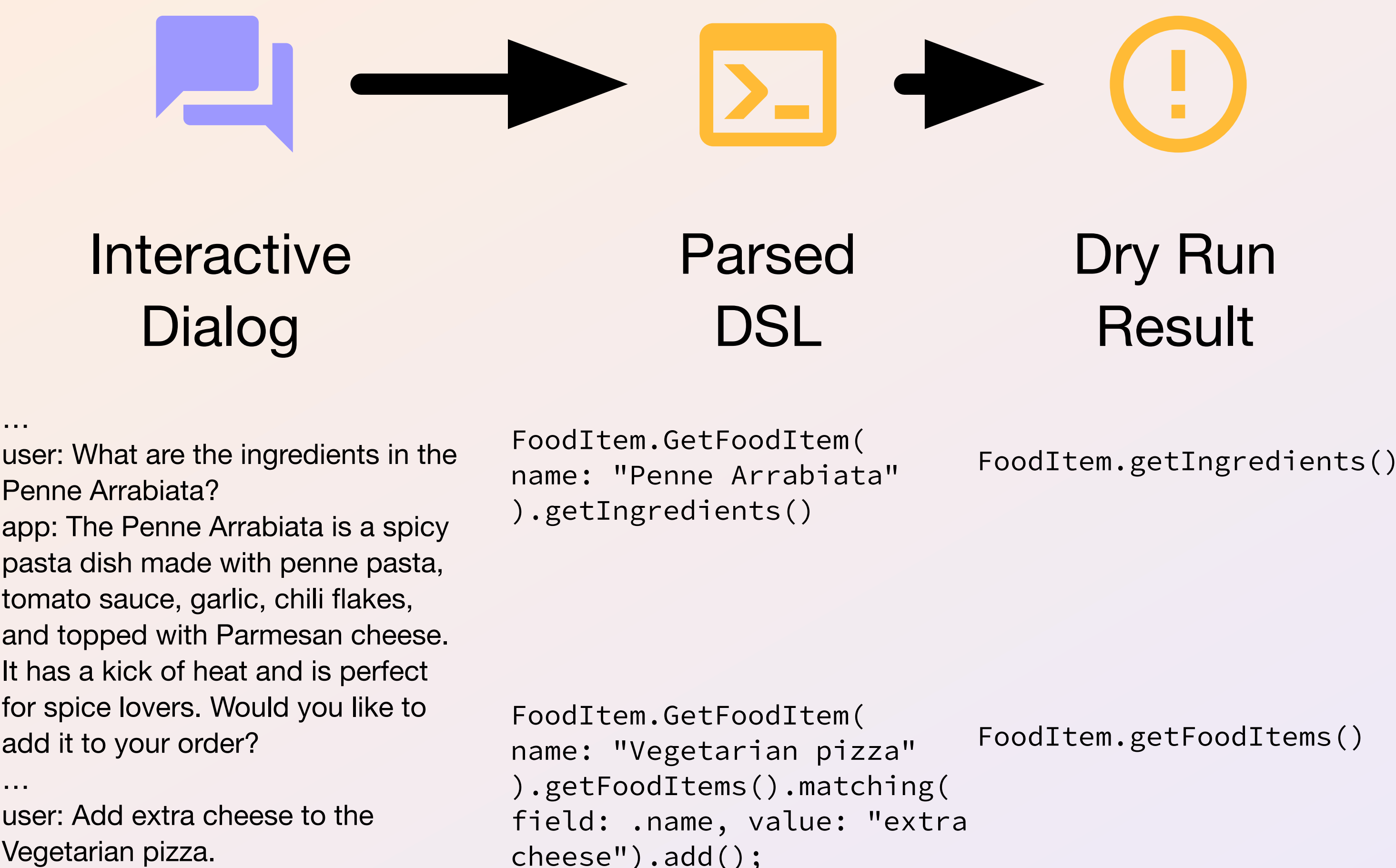
Landon wants to use the digital
menu app in an Italian restaurant to
enhance his dining experience. He
is looking forward to exploring the
menu options and discovering new
dishes. Landon's task is to browse
through the digital menu, select his
preferred dishes, and customize
them according to his dietary
preferences. ...



Interactive
Dialog

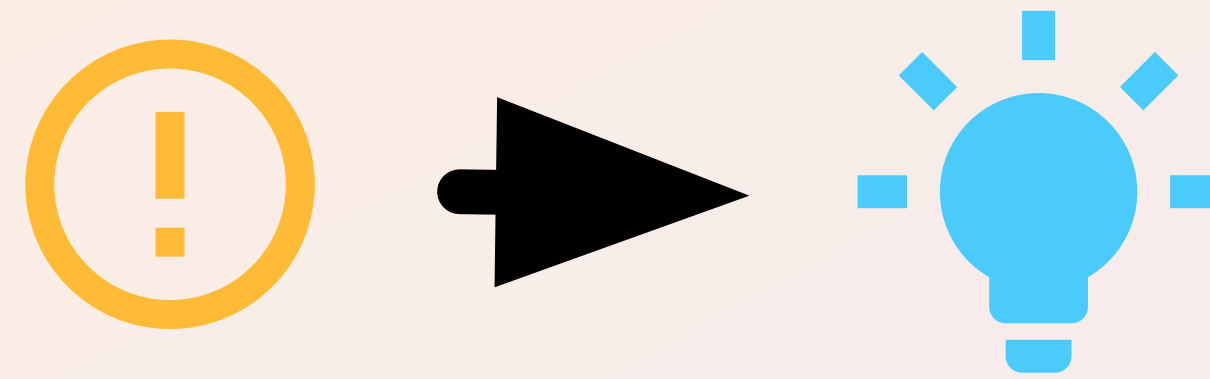
...
user: What are the ingredients in the
Penne Arrabiata?
app: The Penne Arrabiata is a spicy
pasta dish made with penne pasta,
tomato sauce, garlic, chili flakes,
and topped with Parmesan cheese.
It has a kick of heat and is perfect
for spice lovers. Would you like to
add it to your order?
...
user: Add extra cheese to the
Vegetarian pizza.

Step 2: Speculating on missing features



Speculative parsing based on LLM hallucination *Abstract interpretation*

Step 3: Suggesting features to implement



Dry Run
Result

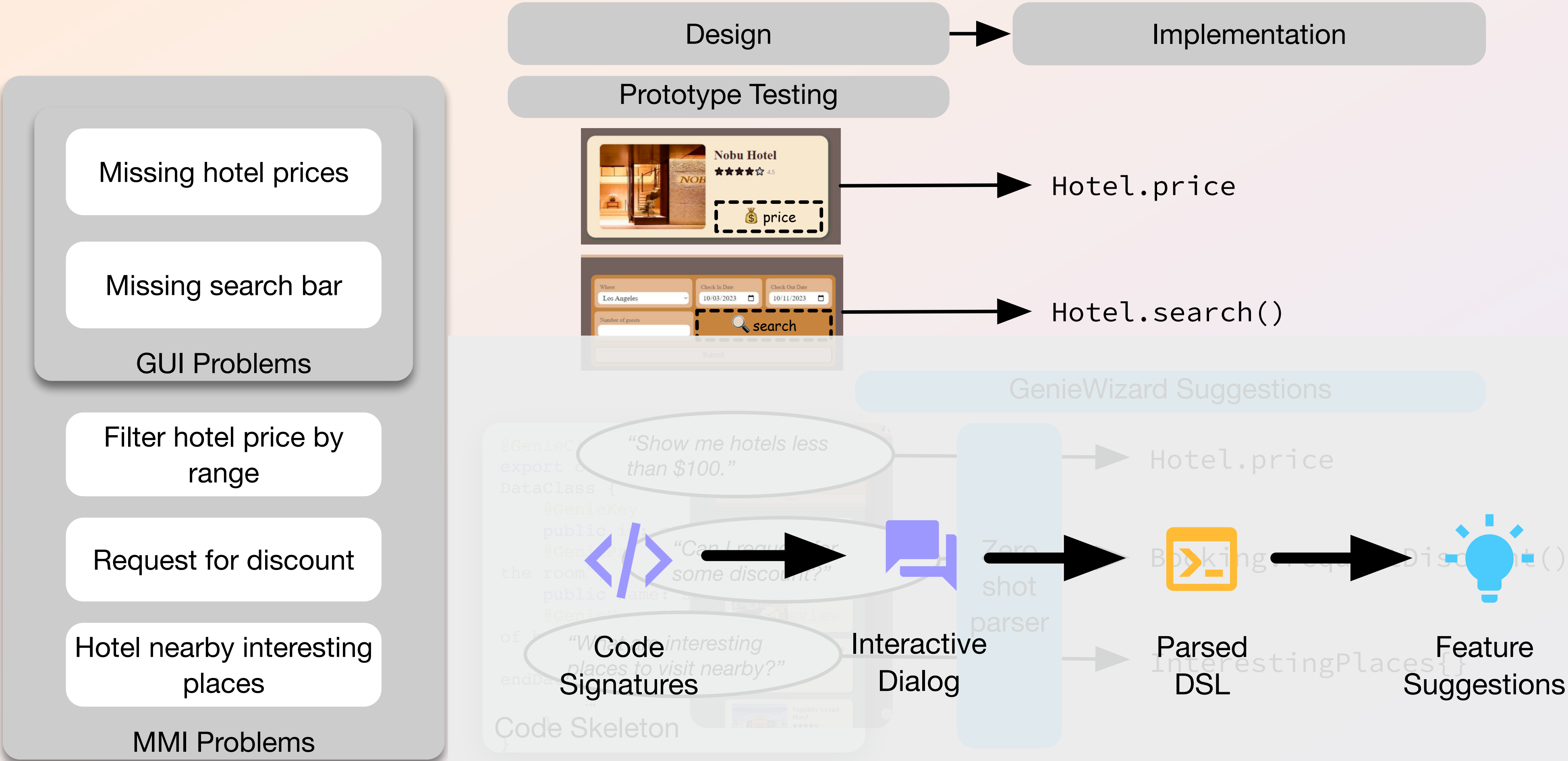
Feature
Suggestions

```
FoodItem.getIngredients() FoodItem.getIngredients()  
// Returns a list of  
ingredients for a specific  
food item.
```

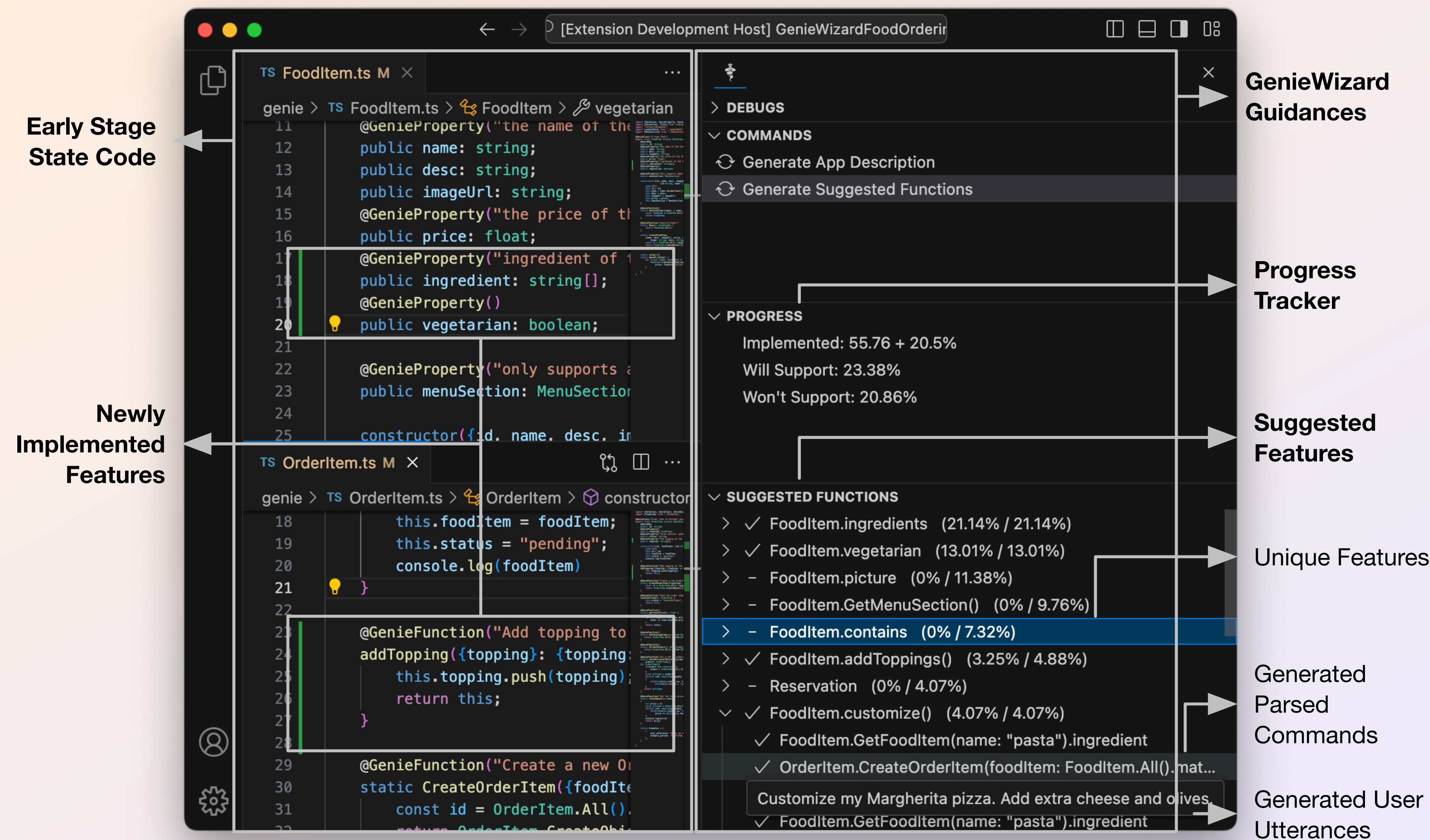
```
FoodItem.getFoodItems() FoodItem.addToppings()  
// Adds toppings to a  
specific food item.
```

*Agglomerative clustering
on content embedding*

GenieWizard is based on simulated user testing



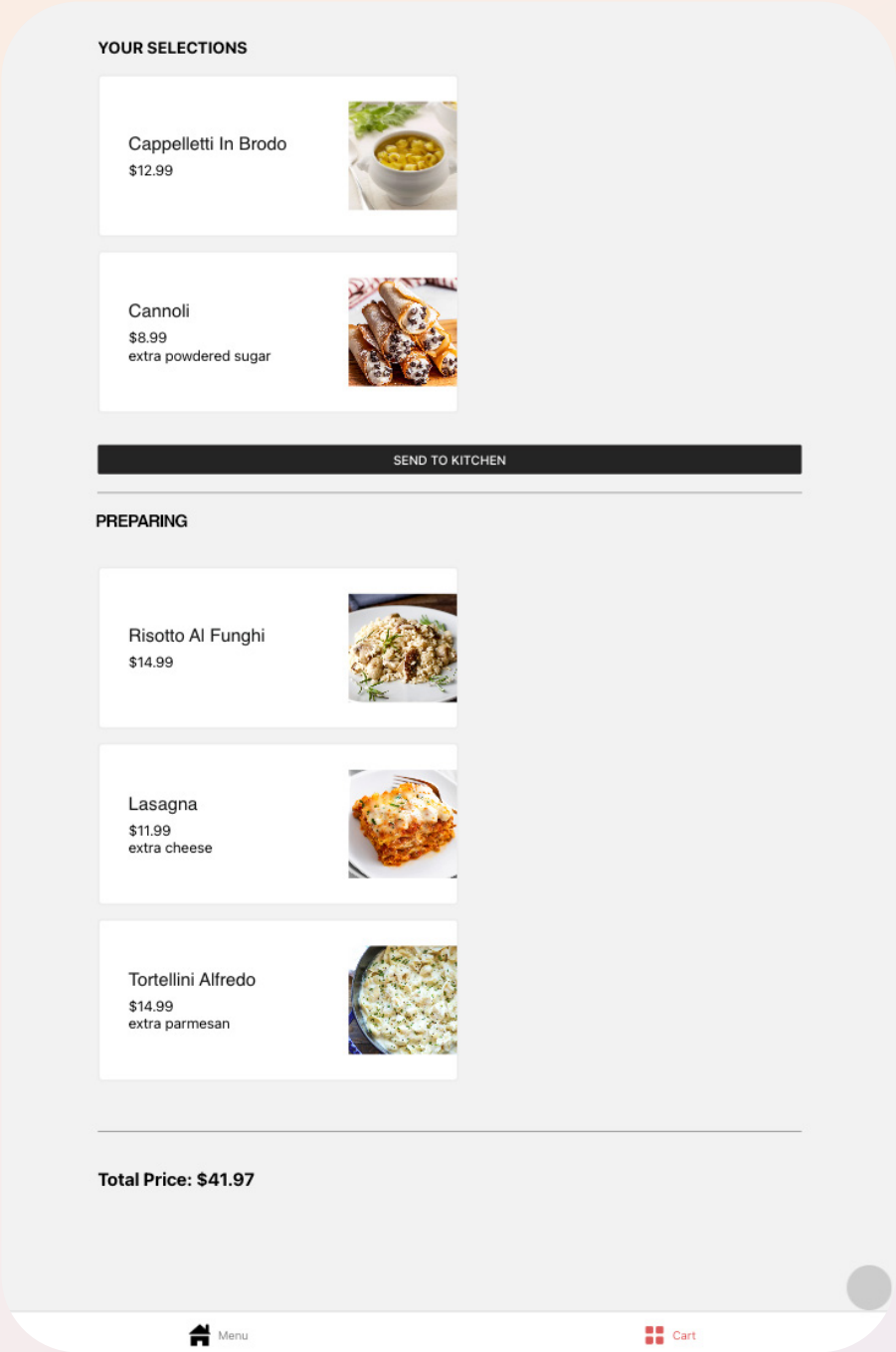
GenieWizard User Interface



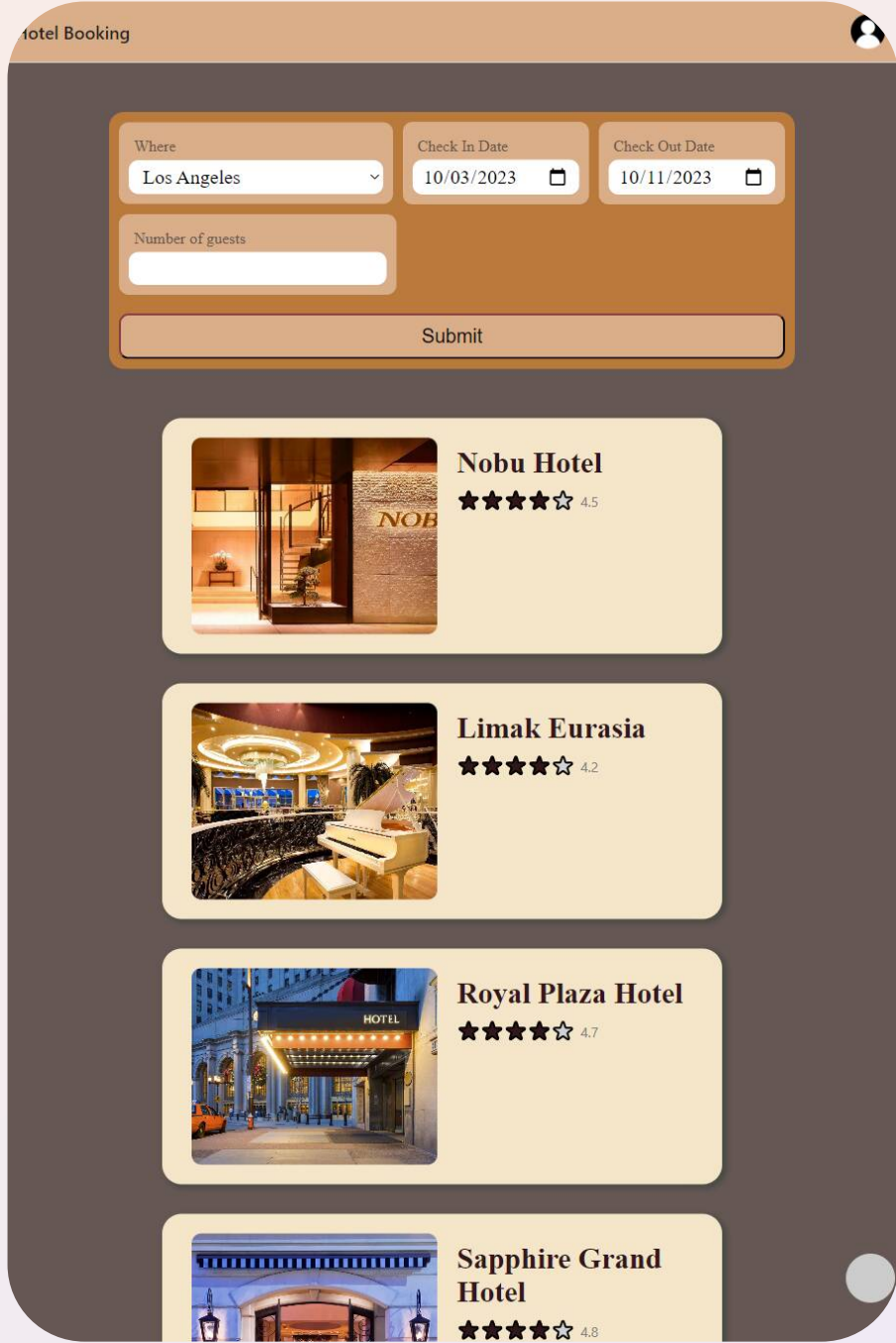
Two demo apps for testing GenieWizard

12 Developers
(Within subject)

Improved



Food Menu App



Hotel Booking App

GenieWizard

using

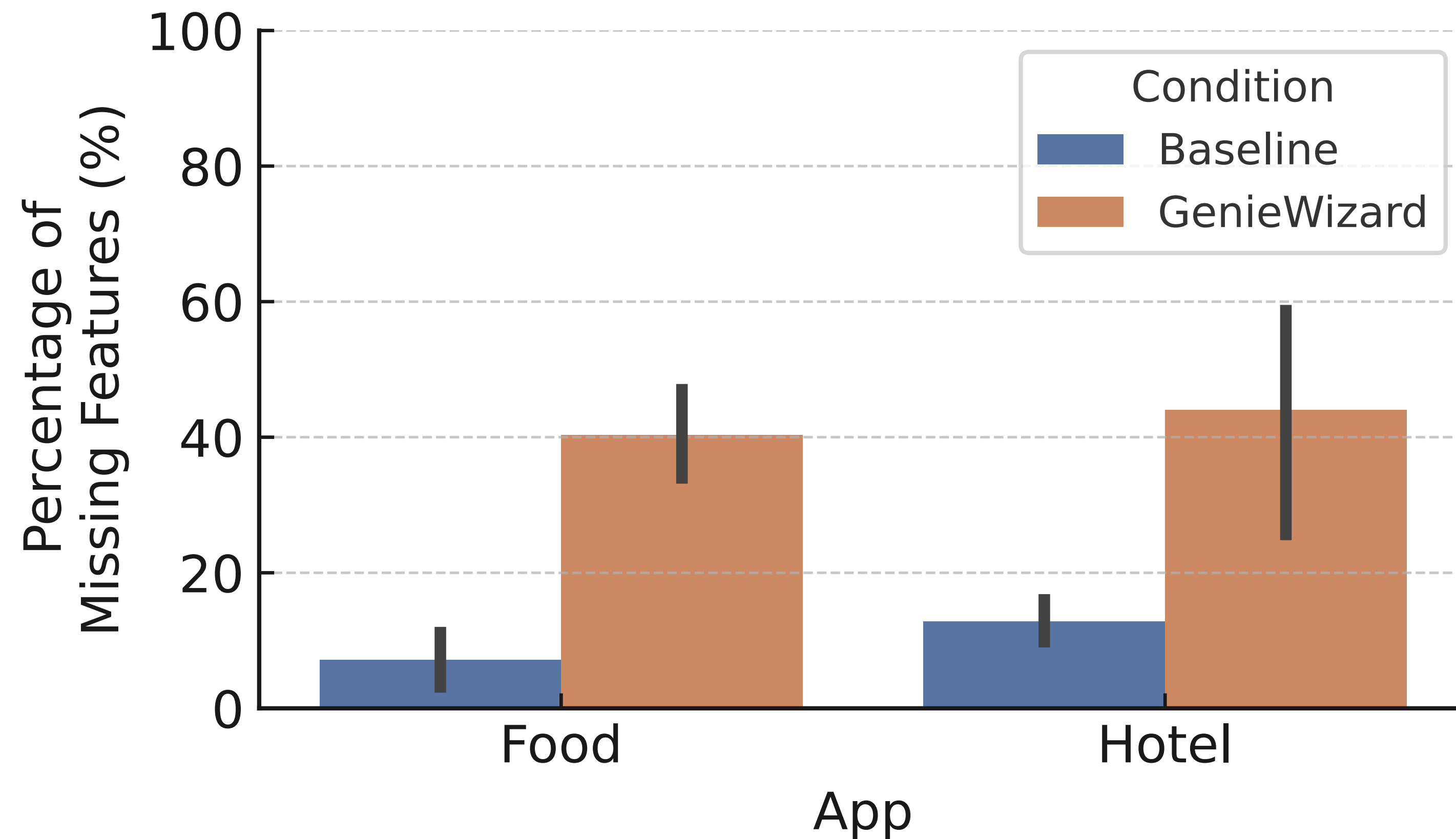
VS Code

Participants

Two Apps

Two Tools

GenieWizard helps developers identify more missing features

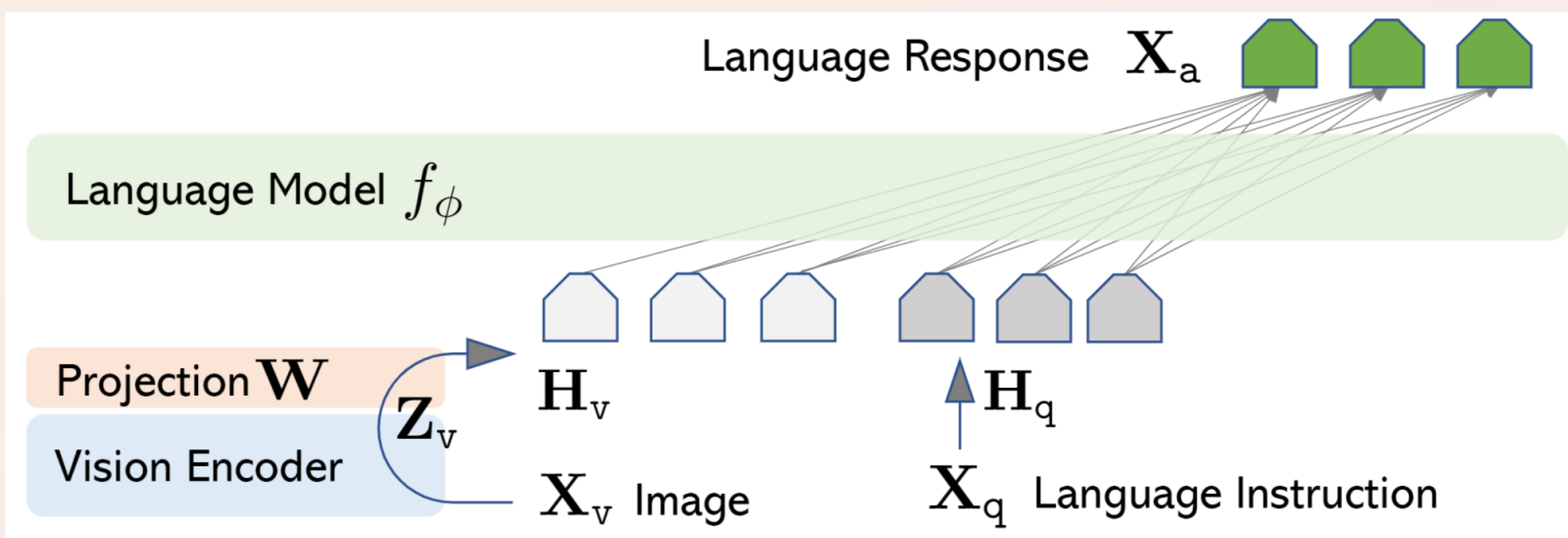


GenieWizard helps developer identify 42% instead of 10% of the missing features

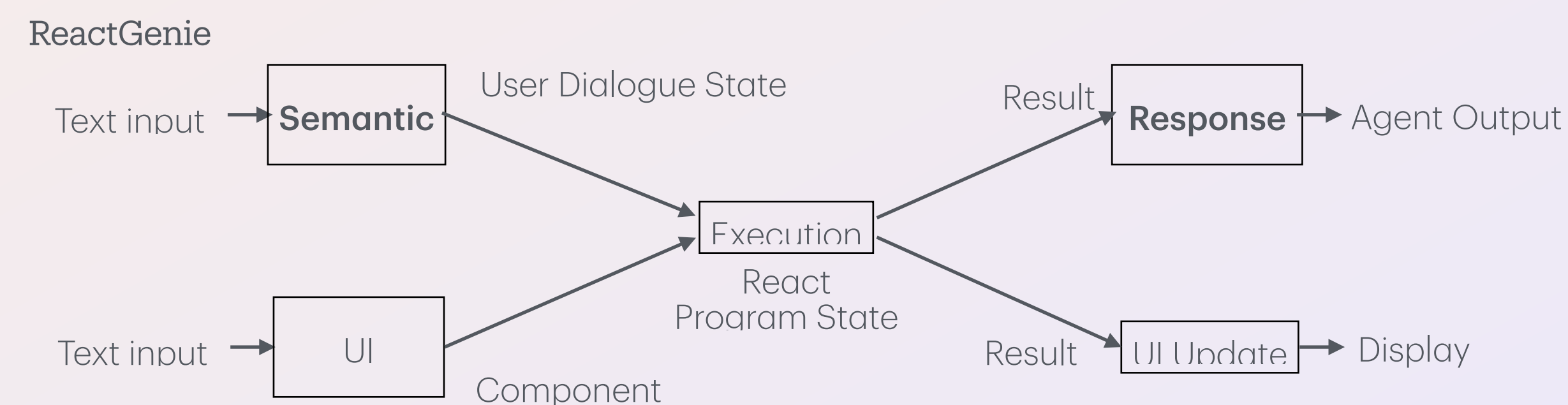
Discussion: New kinds of systems research in the era of AI

- System research areas:
 - Operating systems, Compilers, Networkings, Software Engineering
 - System research about AI today are mostly on accelerating training and serving
- Projects in today's talk (other parts of this course as well) used system research techniques for AI-HCI problems
 - ReactGenie
 - Interpreters, UI frameworks
 - GenieWizard
 - Speculative parsing, IDEs
- Future: Uses systems research techniques for AI systems

Discussion: Multimodal interactions vs multimodal machine learning



Multimodal models
Picture -> tokens



Multimodal interactions
UI + Interactions -> tokens

It's all tokenization problems

Recap: Multimodal Interactions

- Multimodal interaction history is long, but adoption is limited due to implementation costs.
- Compared to voice interfaces, multimodal ones are flexible, efficient, clearer, and less error-prone.

Recap: ReactGenie

- ReactGenie aims to foster multimodal interaction adoption.
 - Merges modern app features and multimodal interface flexibility, ensuring easy development.
 - Utilizes object-oriented state abstraction and declarative UI for modality synchronization.
 - Employs LLMs to expose the app's entire state, rather than limiting it to individual APIs for voice interfaces.
- GenieWizard helps developers build usable multimodal apps in the first iteration
 - Uses LLMs to simulate multimodal user interactions
 - Speculates missing features using LLM hallucination and abstract interpretation
 - Helps developers to help end users

Recap: GenieWizard

- GenieWizard helps developers build usable multimodal apps in the first iteration
 - Uses LLMs to simulate multimodal user interactions
 - Speculaes missing features using LLM hallucination and abstract interpretation
 - Helps developers to help end users

I hope that you will...

<https://jya.ng/reactgenie>

- Build with it 😎
- Break it 😈
- Fix it! 🥳

Recap

- Multimodal interaction history is long, but adoption is limited due to implementation costs.
- Compared to voice interfaces, multimodal ones are flexible, efficient, clearer, and less error-prone.
- ReactGenie aims to foster multimodal interaction adoption.
 - Merges modern app features and multimodal interface flexibility, ensuring easy development.
 - Utilizes object-oriented state abstraction and declarative UI for modality synchronization.
 - Employs LLMs to expose the app's entire state, rather than limiting it to individual APIs for voice interfaces.
- GenieWizard helps developers build usable multimodal apps in the first iteration
 - Uses LLMs to simulate multimodal user interactions
 - Speculates missing features using LLM hallucination and abstract interpretation
 - Helps developers to help end users