Stanford CS224v Course

Conversational Virtual Assistants with Deep Learning

Lecture 13

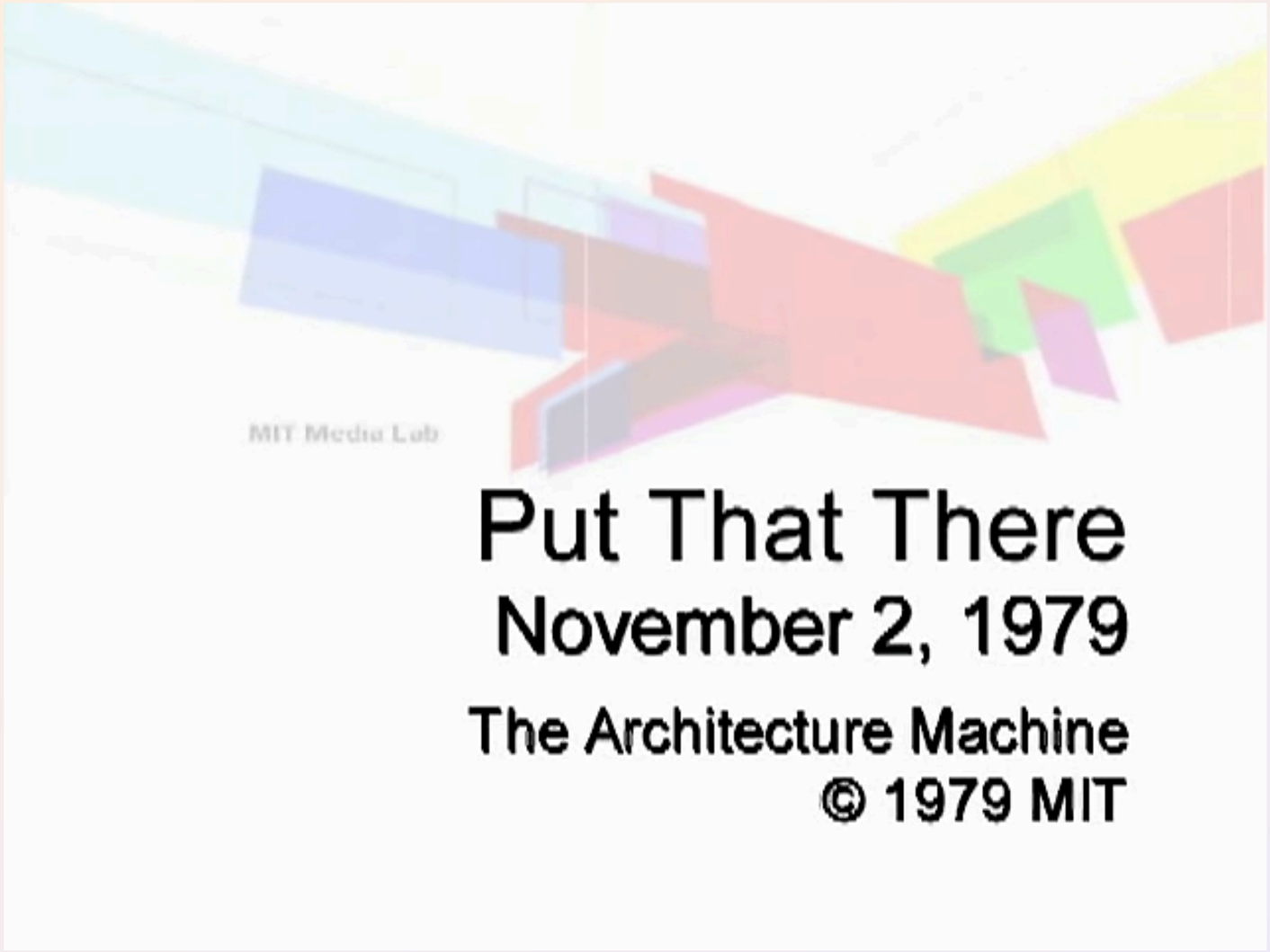
Multimodal Applications

Jackie (Junrui) Yang & Monica Lam

Lecture Goals

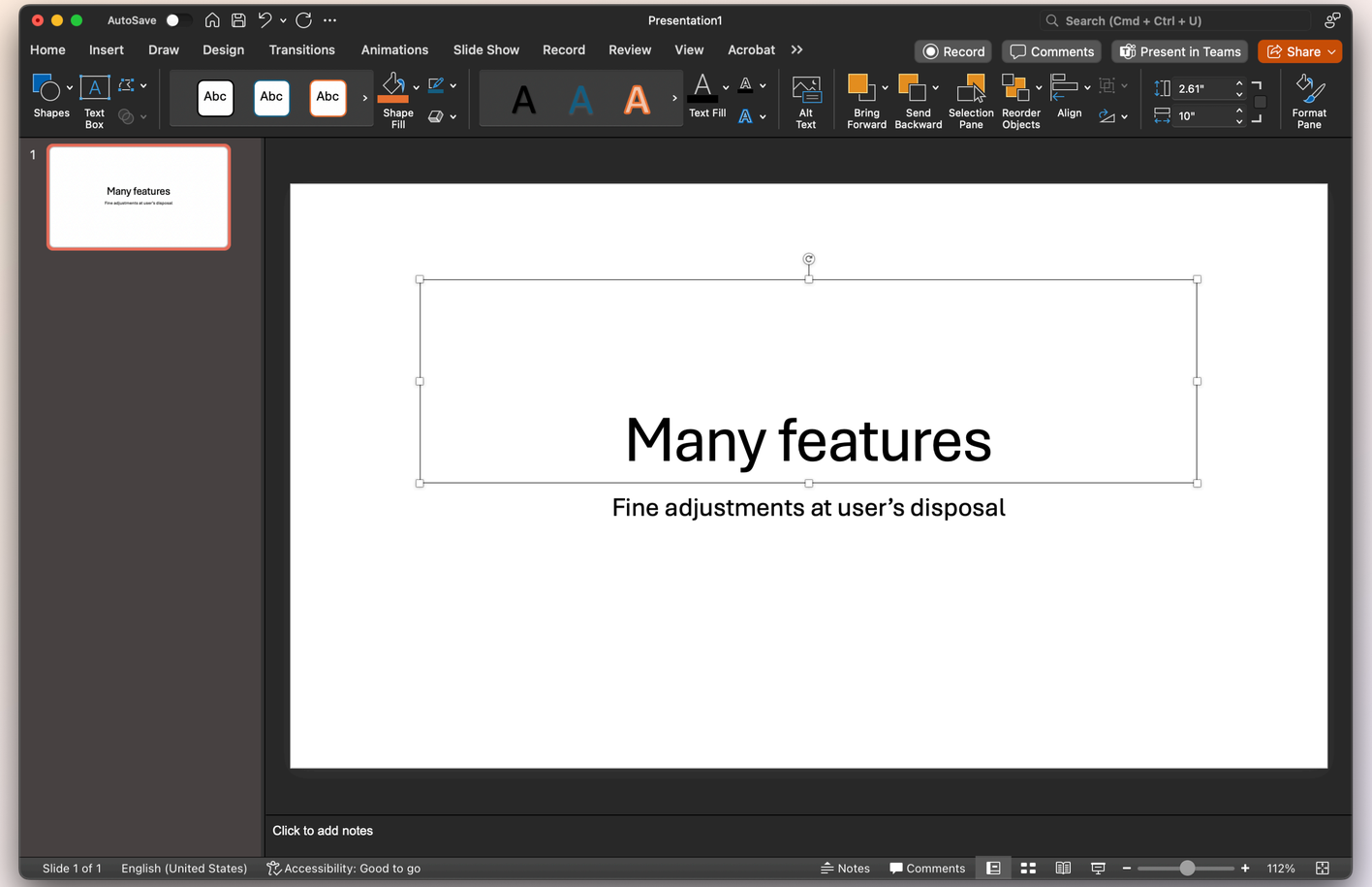
- **Why do we need multimodal interactions?**
- Three problems for multimodal app development
- ReactGenie: a multimodal app development framework

Proposed in 1979



Why?

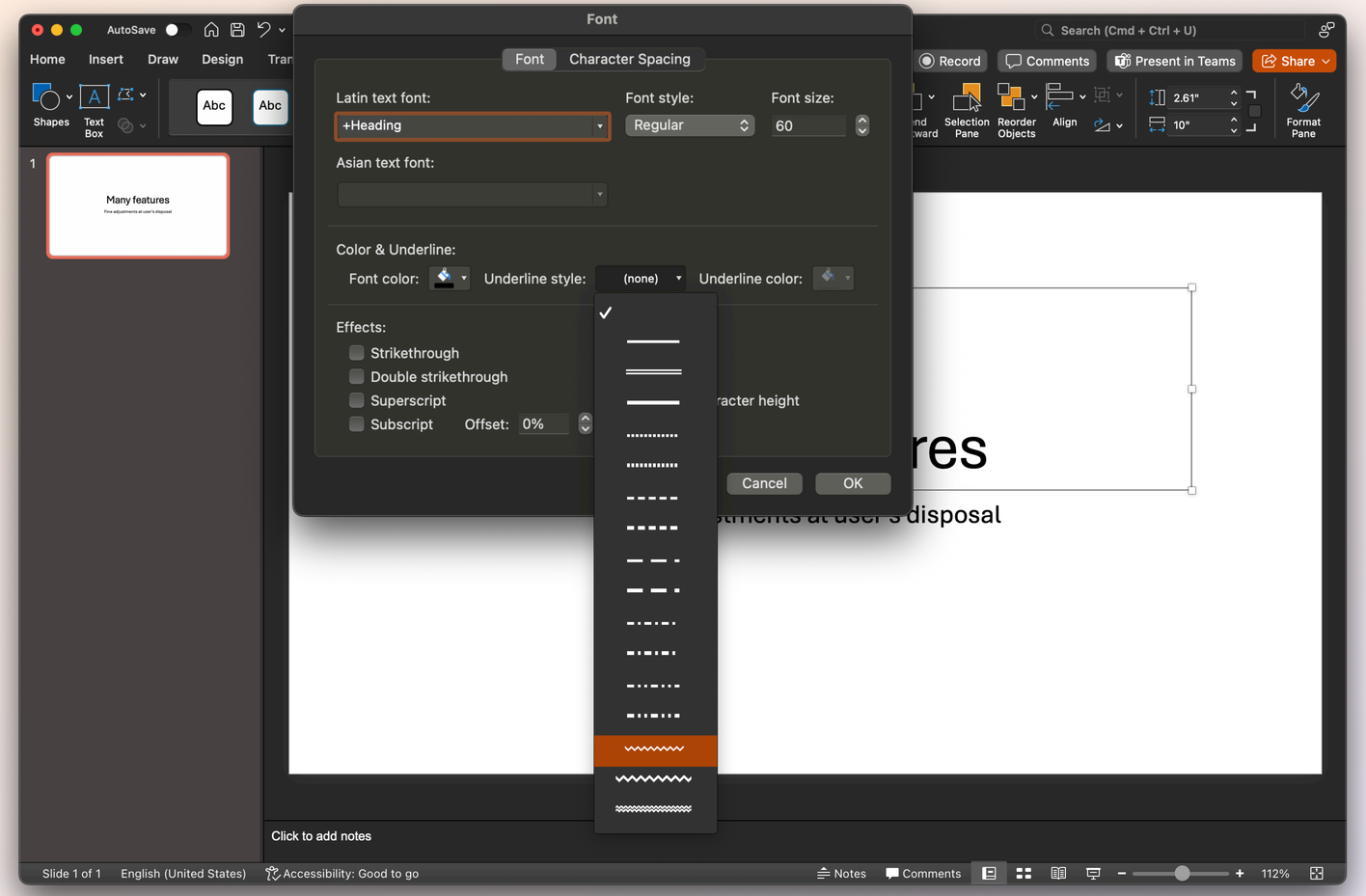
Today: Interacting with Touch + Graphics: Powerful and accurate



PowerPoint:

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

Interacting with Touch + Graphics: But sometimes inefficient and repetitive

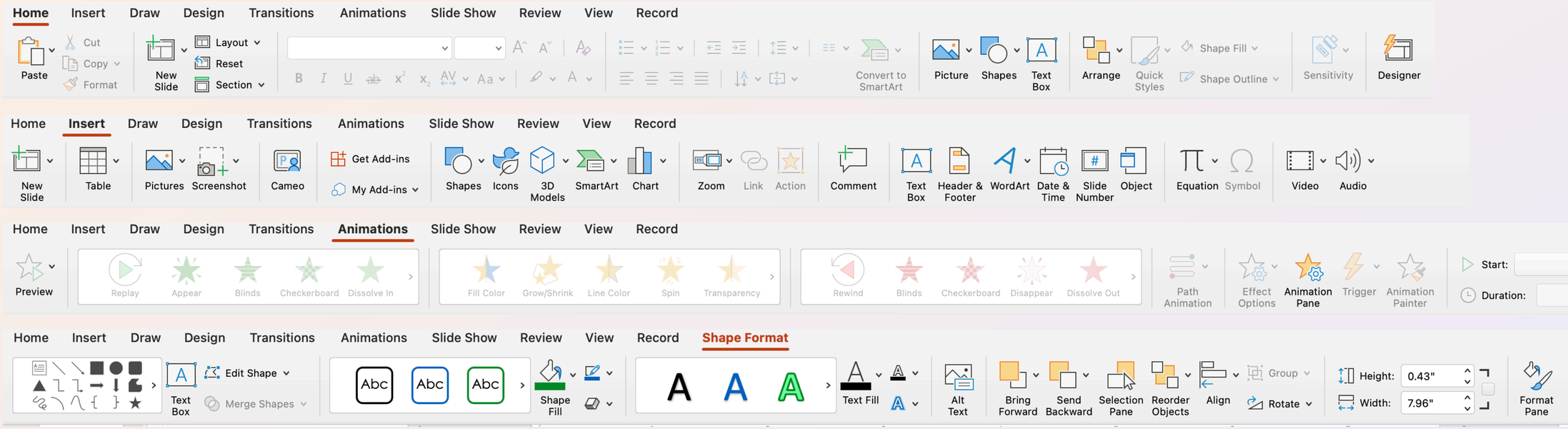


Slow and tedious for

- Apply actions to multiple objects
- Less common features

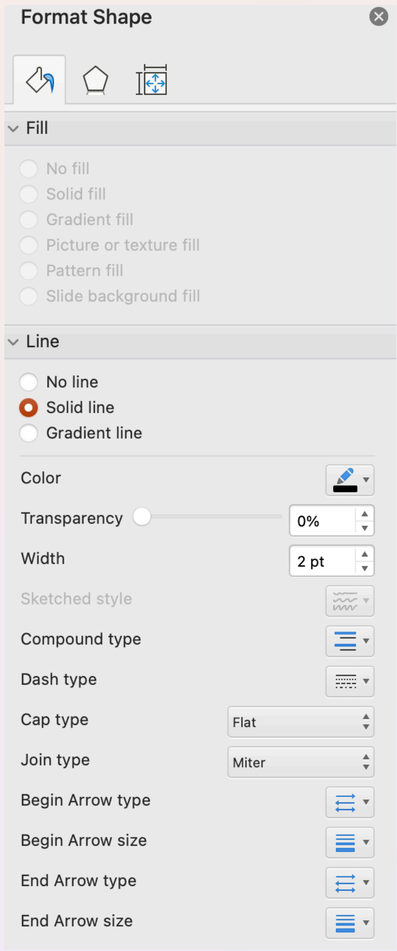
Example: Powerpoint

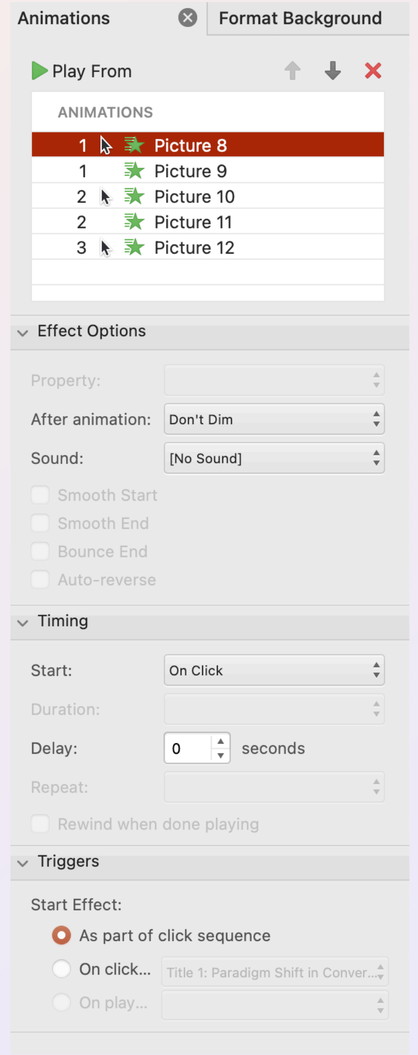
- Lots and lots of nested menus



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)

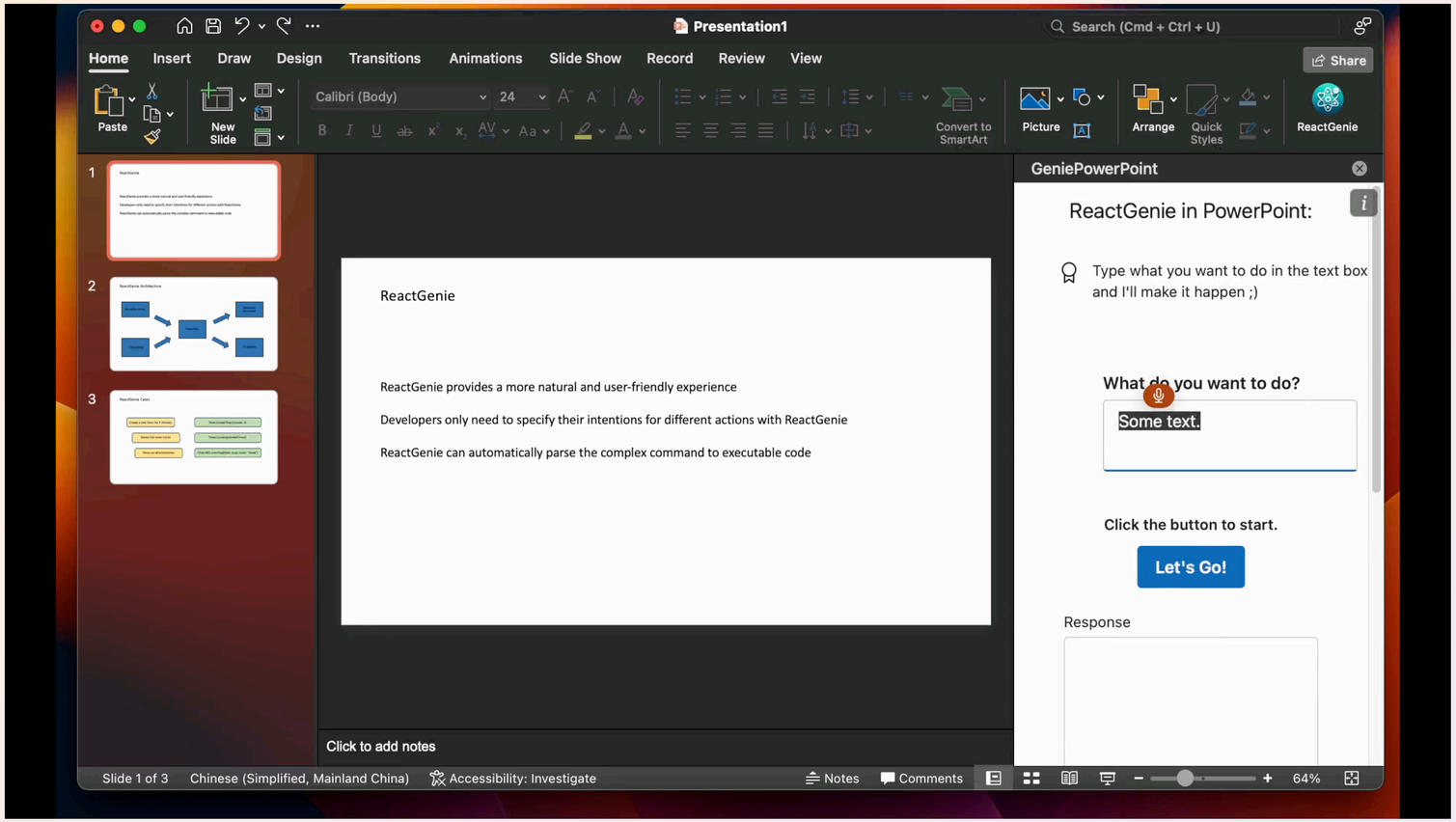




Here are Some Examples:

- 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

How to Create Sophisticated Multimodal Apps?

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

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: Limitation of Function Calling

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")

How many functions do we need for function calling — this is not scalable!

Solution 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(horizontalAligment:"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

Design of NLPL (Natural Language Programming Language)

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

Delete Shapes above this

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

Automatically distributed to each element

Easy to generate
Versatile query
Fewer errors
No Lambda Expression

NLPL

Expressiveness of NLPL

Change background color for all the yellow shapes to orange

Verb | Object Modifier | Object | Verb 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)

Examples of Compound Commands

Presentation1

Search (Cmd + Ctrl + U)

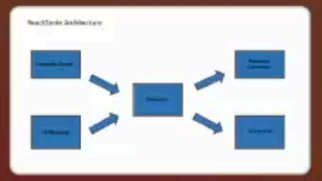
Home Insert Draw Design Transitions Animations Slide Show Record Review View

Share

Paste New Slide Calibri (Body) 24 Convert to SmartArt Picture Arrange Quick Styles ReactGenie

1

2



3

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

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%

Making PowerPoint Multimodal with Gen

- Built on MS PowerPoint Javascript API
  - 110 APIs: Slide, Shape, SlideMaster, TextFrame, TextRange, …
  - Yet, it does not include common APIs like Animation, Font color, etc.
  - It is super hard to expose API outside of the development cycle of GUI
- Powerpoint today has a human “GUI” interface
- How do we create a multimodal interface?

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: How to Expose APIs to Multimodal Interace

- Can we do it with minimal engineering effort?
- Solution: Just add annotations to existing code to expose desired features
  - GenieClass: indicates it has a multimodal interface
  - GenieKey: each instance has an ID
  - GenieProperty: exposed properties (variables)
  - GenieFunction: exposed functions

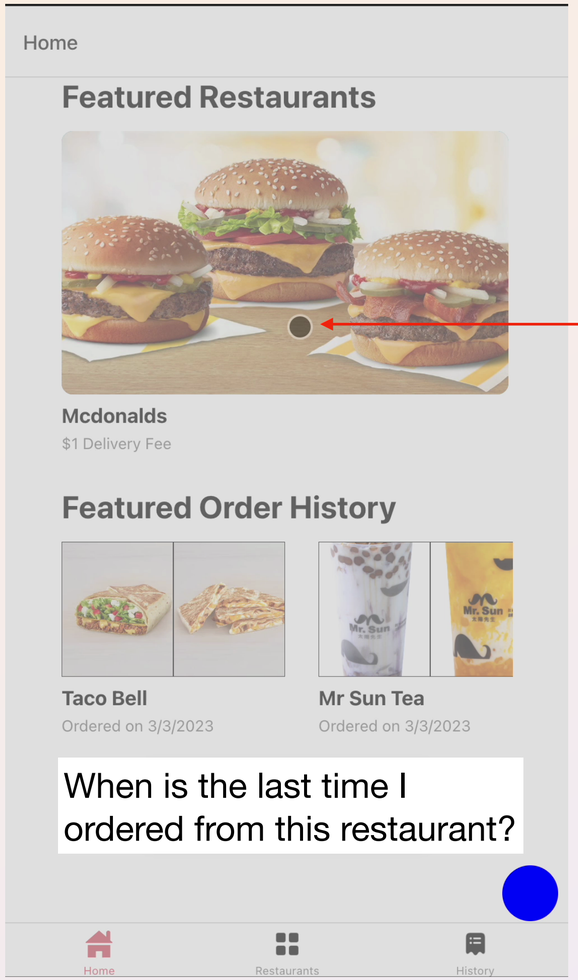
Solution 2: Annotate exposed APIs

```
@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();
    }
}
```

ReactGenie Annotations

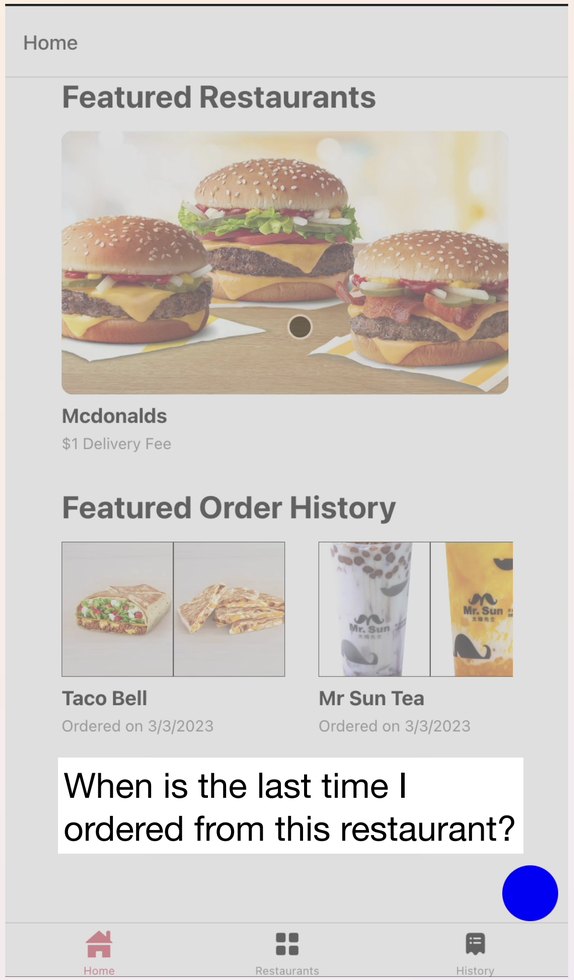
React App Logic Code

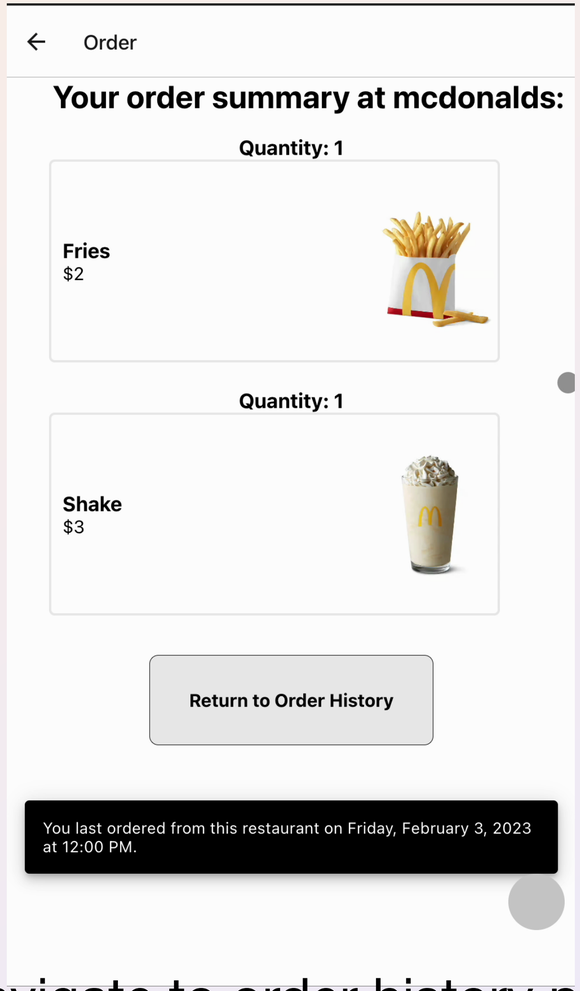
Problem 3: How to tell the user what happened? 🤷



User touching this picture

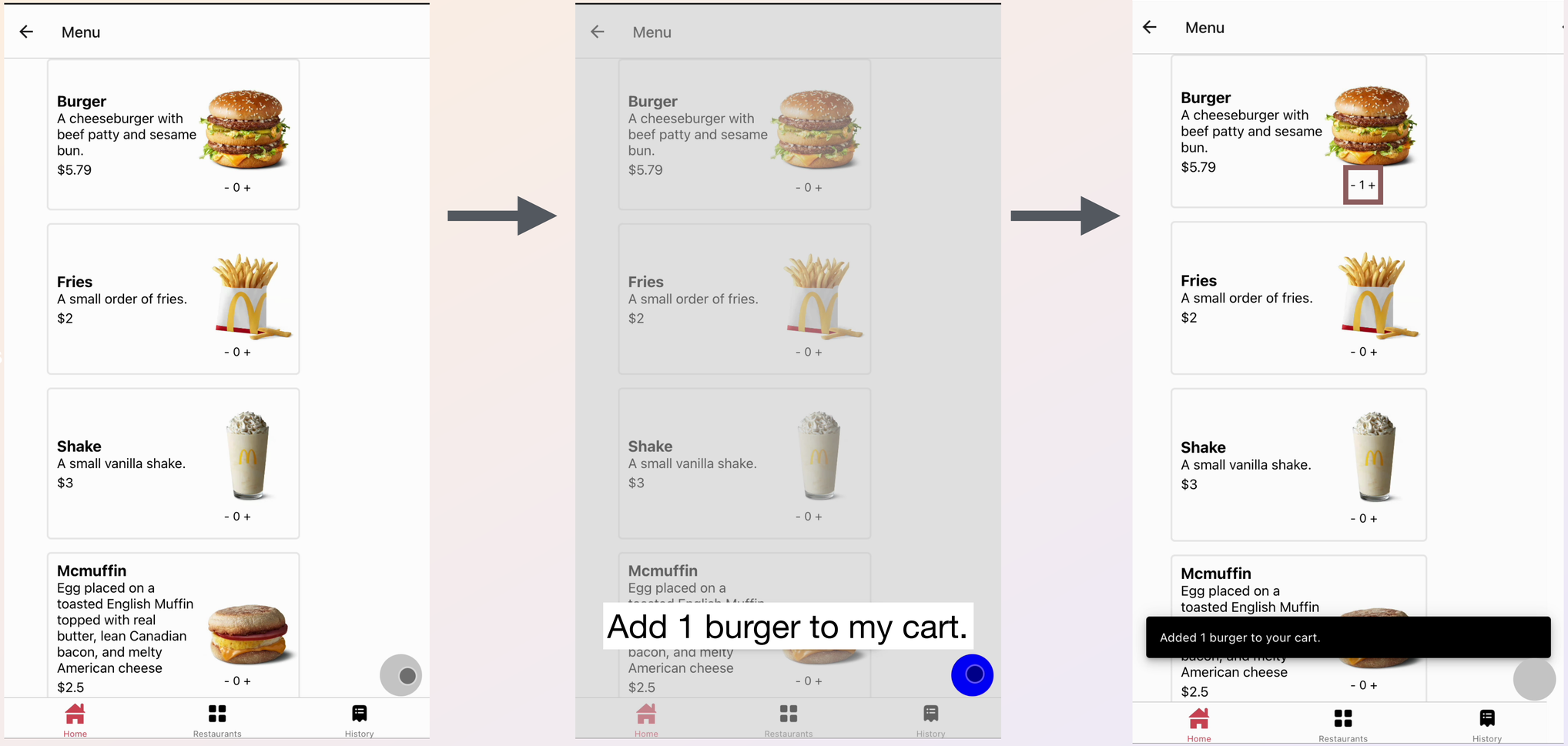
Solution 3: Interchangeable & simultaneous multimodal I/O





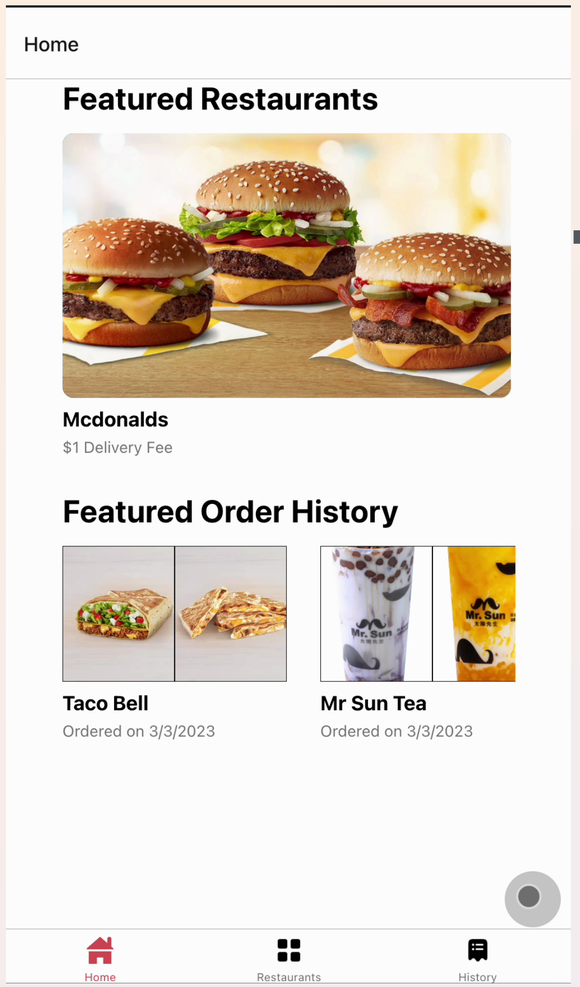
Navigate to order history page for McDonald's

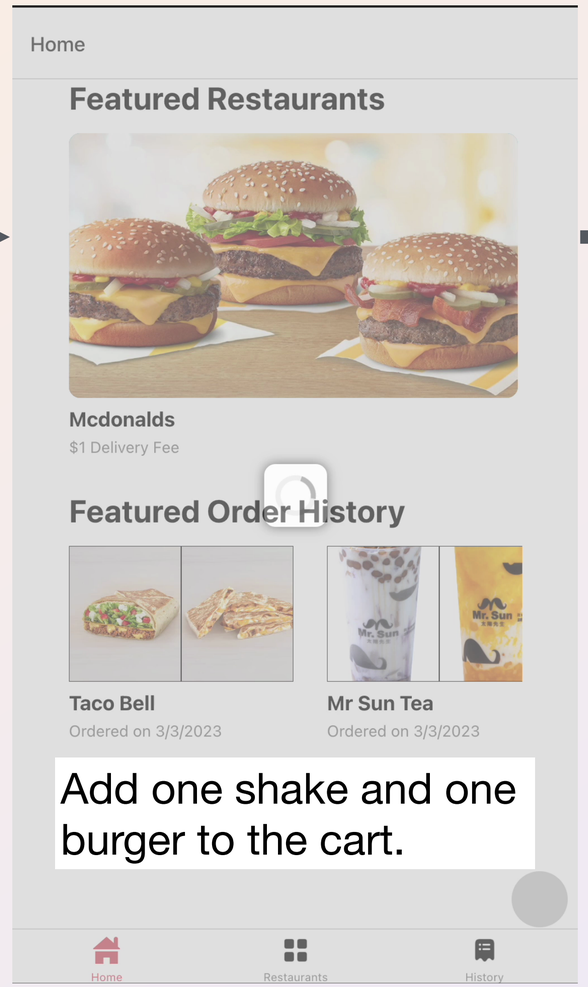
Example 2

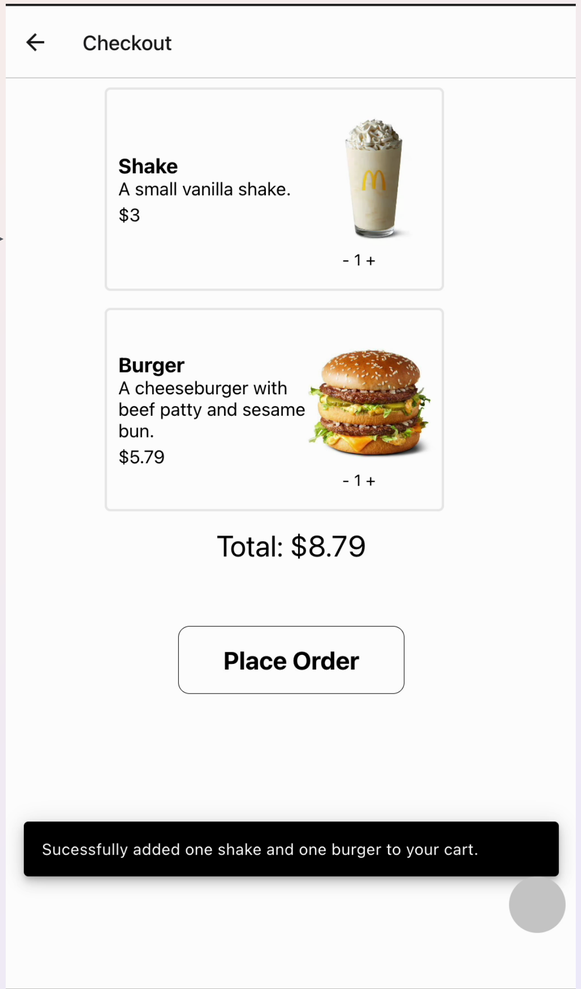


Perform action and Update page

Example 3







Perform action and Navigate to another page

Solution 3: Interchangeable & simultaneous multimodal I/O

- 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

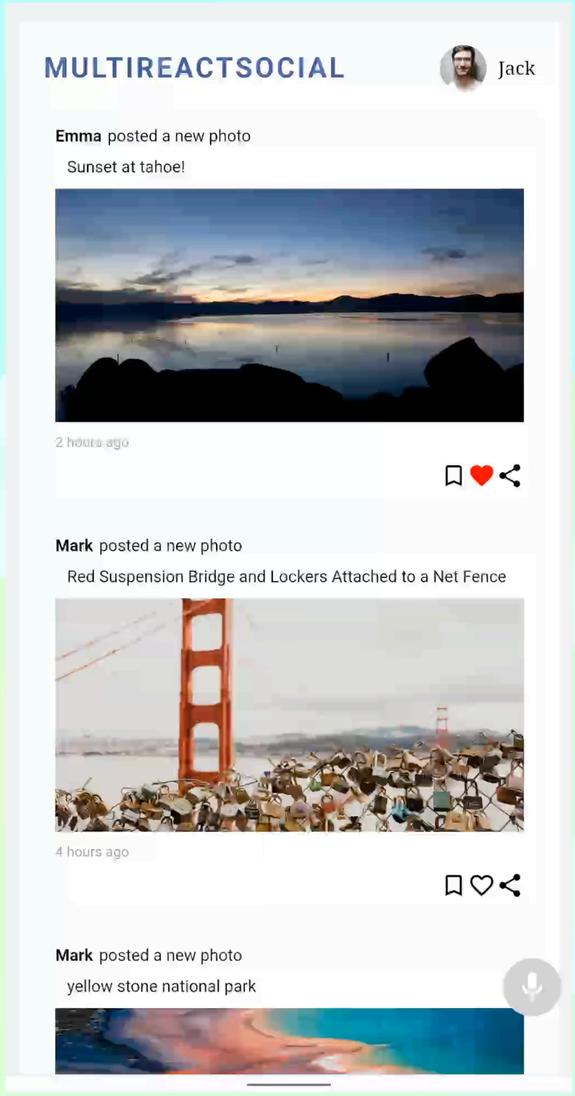
- Why do we need multimodal interactions?
- Three problems for multimodal app development
- **ReactGenie: a multimodal app development framework**

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

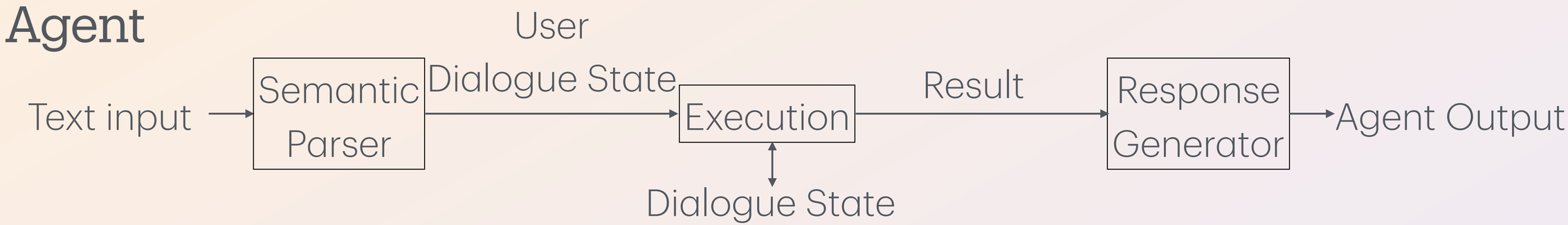
```
@DataClass()
class Recipe: GenieClass {
    @GenieProperty()
  name: String;

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

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

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

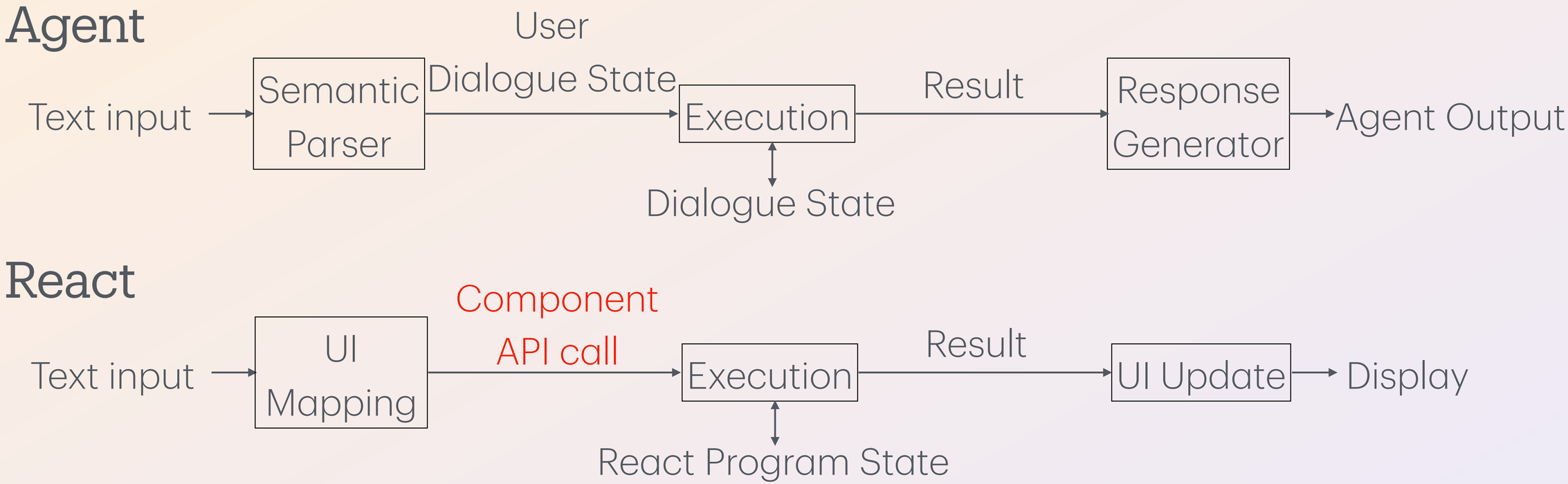
```
RecipeViewImpl = (recipe: Recipe) => {
  return (
    <div>
      <img image={recipe.img}/>
      <love loved={recipe.loved}
       onClick(()=>recipe.love())/>
      <div> {recipe.name} </div>
    </div>
  )
}

export RecipeView = GenieInterface(“Recipe”,
  RecipeViewImpl)
```

Components Annotations:
Which components represent which state classes

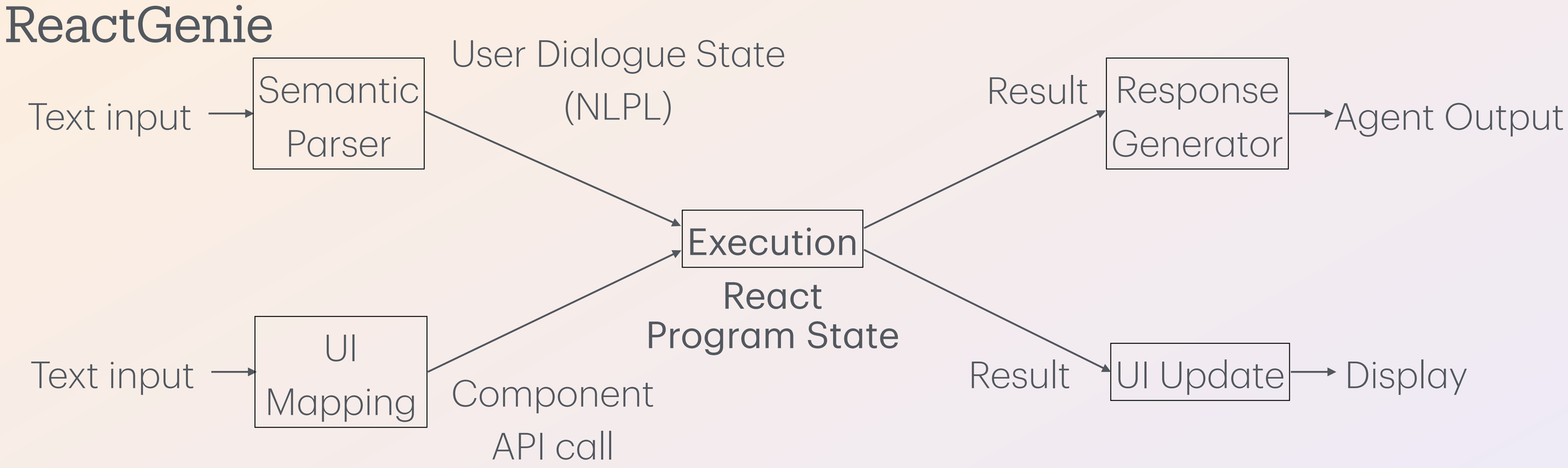
Recall the Agent Architecture



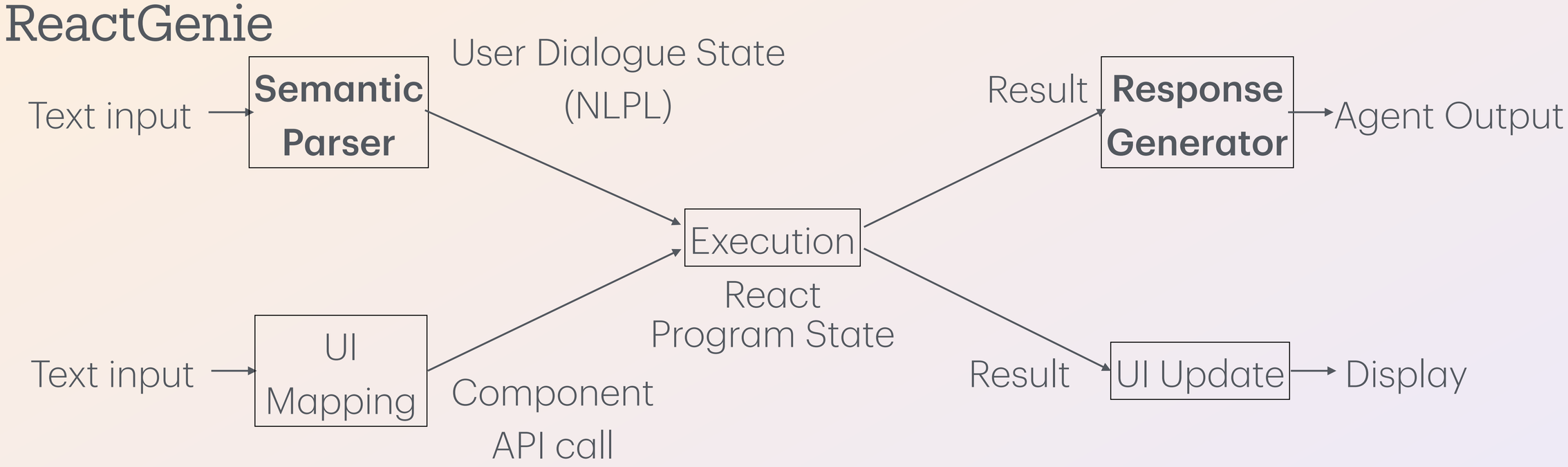
React has a Similar Architecture



Multimodal Agent Architecture



Semantic Parser + Response Generator



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(dashSyle:"RoundDot")
```

Make everything right aligned on this slide.

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

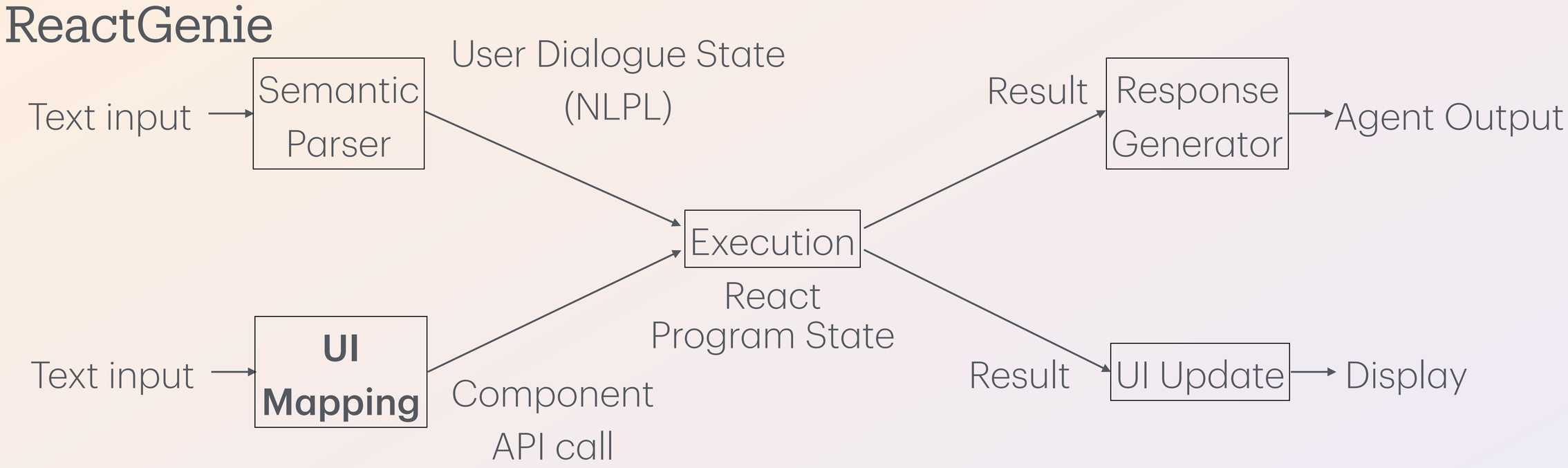
// Examples: user: get me the best restaurant in palo alto agent: Restaurant.all().matching(field: .address, value: "palo alto")…	Few-shot examples
// User interaction user: order the same burger that I ordered at mcDonald last time	Current interaction
parsed: Order.current.addFoods(foods: Order.all().matching…	Parsed result

LLM-Based Response Generator

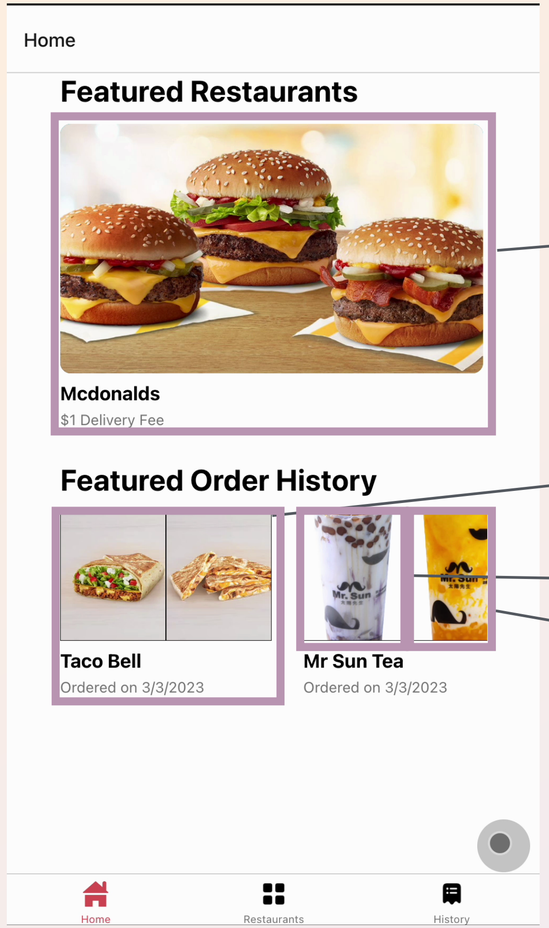
// Here are all the functions that we have
 	
class Restaurant {
  string name;
  string address;
  string cuisine;
  float rating;
 	
  // All active restaurants
  static Restaurant[] all();
 	
  // The current restaurants
  static Restaurant current();
 	
  // Get a list of foods representing the menu from a restaurant
  Food[] menu;
 	
  // Book reservations on date
  Reservation get_reservation(date: DateTime)
}
	Declaration

// Generate concise voice feedback for the user's command	Instructions
// 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"}]}	Current interaction
response: Your order with a hamburger and fries has been placed.	Generated Response

How to Handle Hybrid Inputs



UI Mapping



Restaurant
(name: “Mcdonald")

Order(date: “3/3/2023”)

FoodItem
(name: “boba tea”)

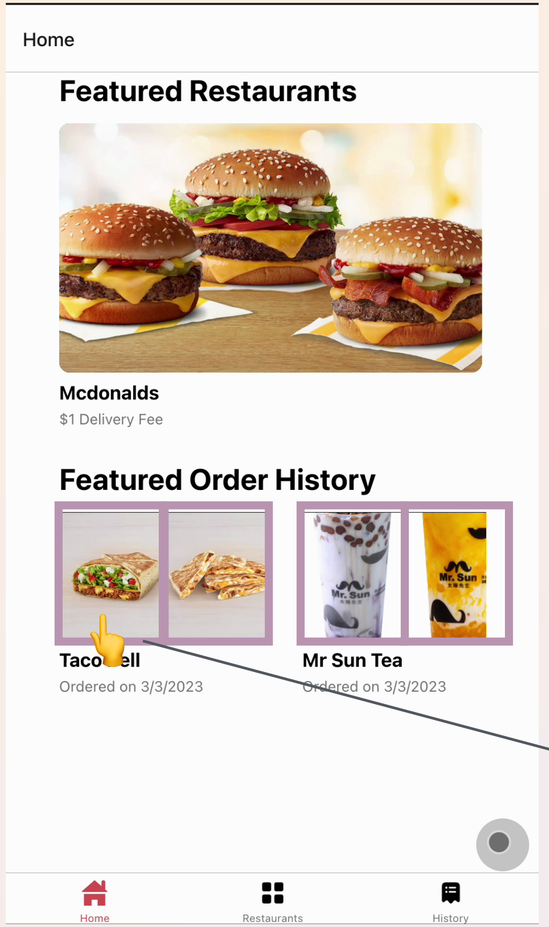
FoodItem
(name: “mango drink”)

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

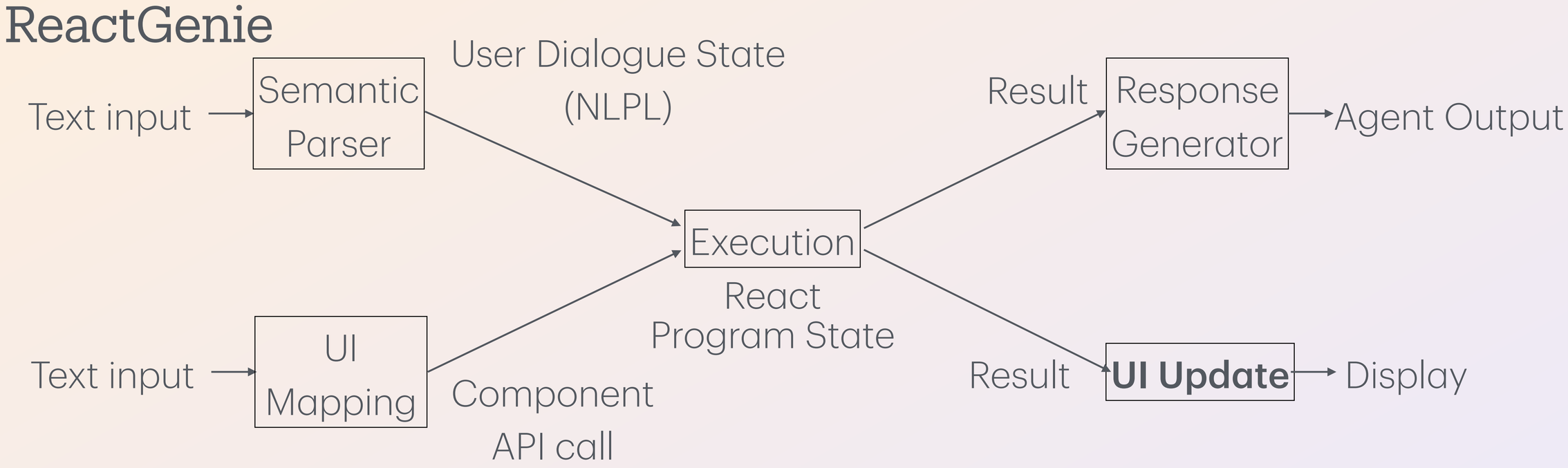
↓

FoodItem(name:"CrunchWrap")

↓

Continue execution

How to Navigate with Voice Commands



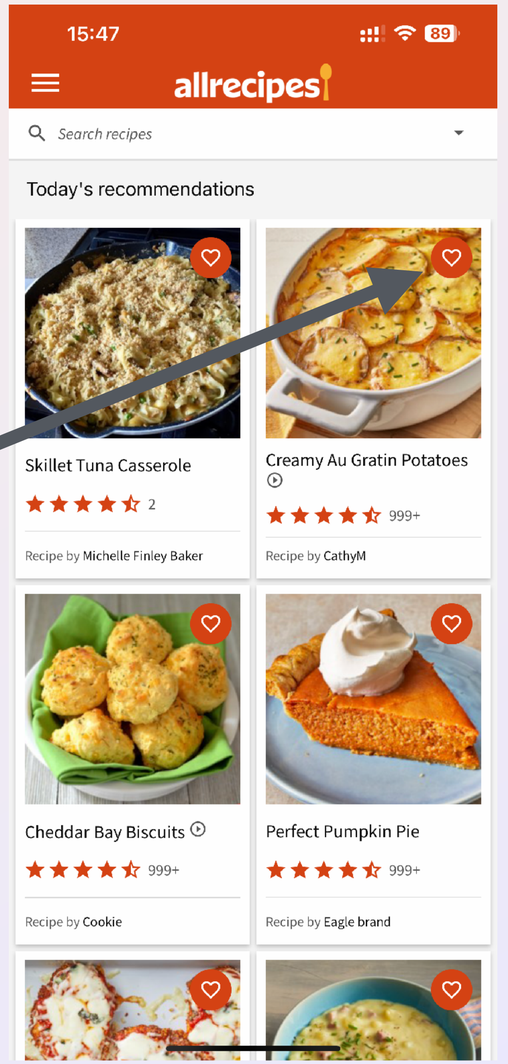
UI Update 1: Object is on UI → React takes care of it

"I love the Creamy Potatoes recipe!"

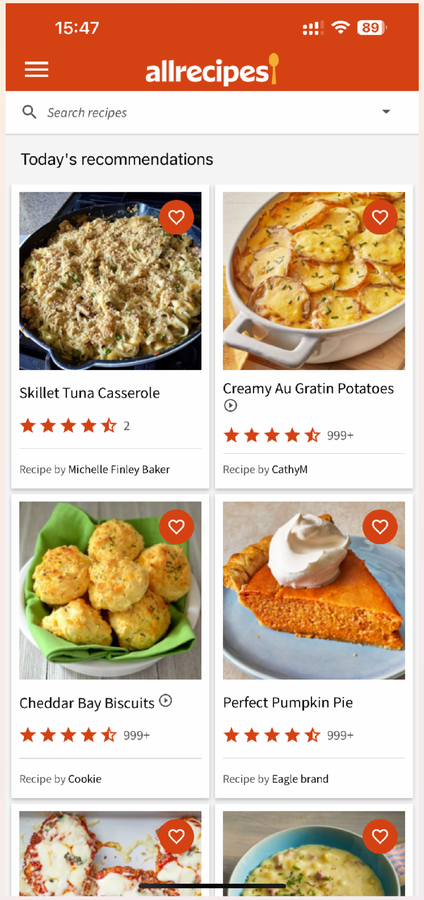
Recipe(name: "Creamy"). love()

recipe.loved = true

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

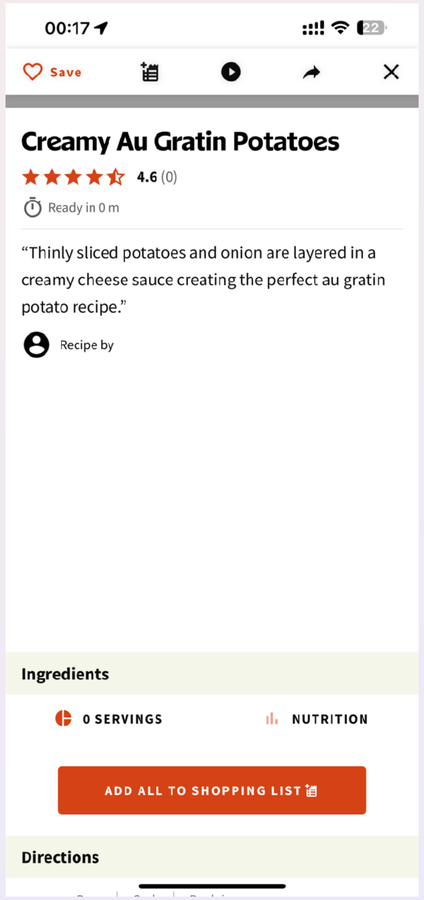


UI Update 2: Object not on UI → Navigate to the page



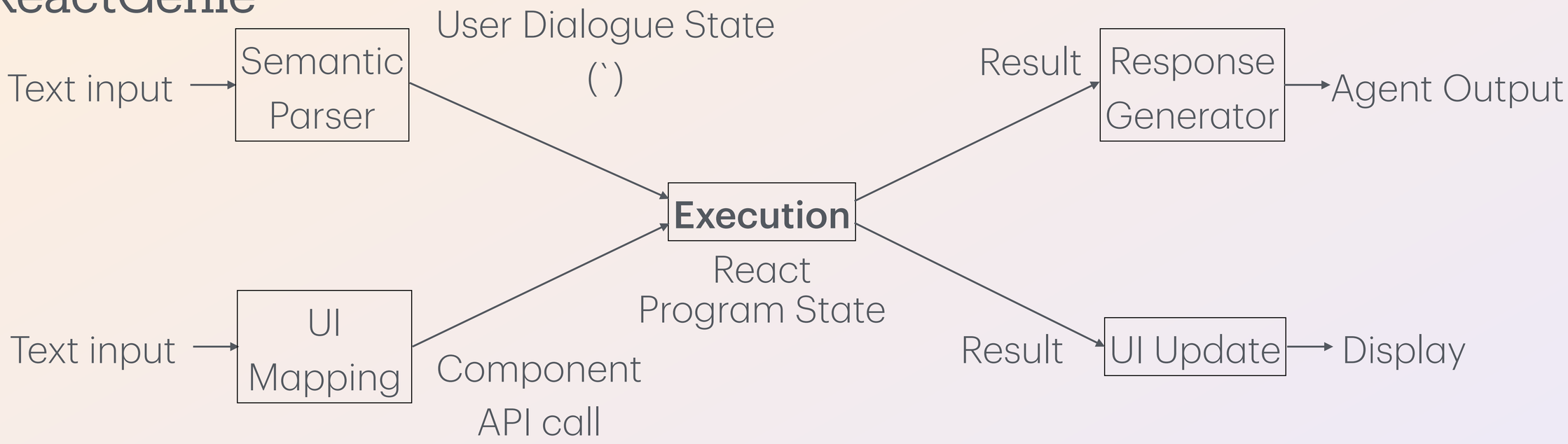
"Show me the Creamy Potato recipe!"

Recipe(name: "Creamy Potato")



Execution

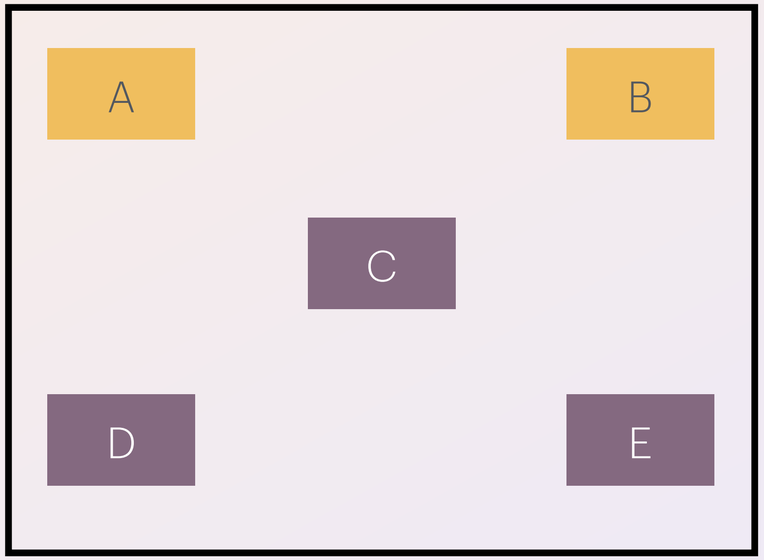
ReactGenie



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)
  - [Shape(text: "A"), Shape(text: "B")]
- Slide.Current().getShapes().between(field:.top, to:Shape.Current().top).fill.setForeGroundColor(color:”yellow")
  - [Fill(), Fill()]

Summary
ReactGenie = React + Annotations

Semantic Parser | Execution | Response Generator | UI Mapping | UI Update

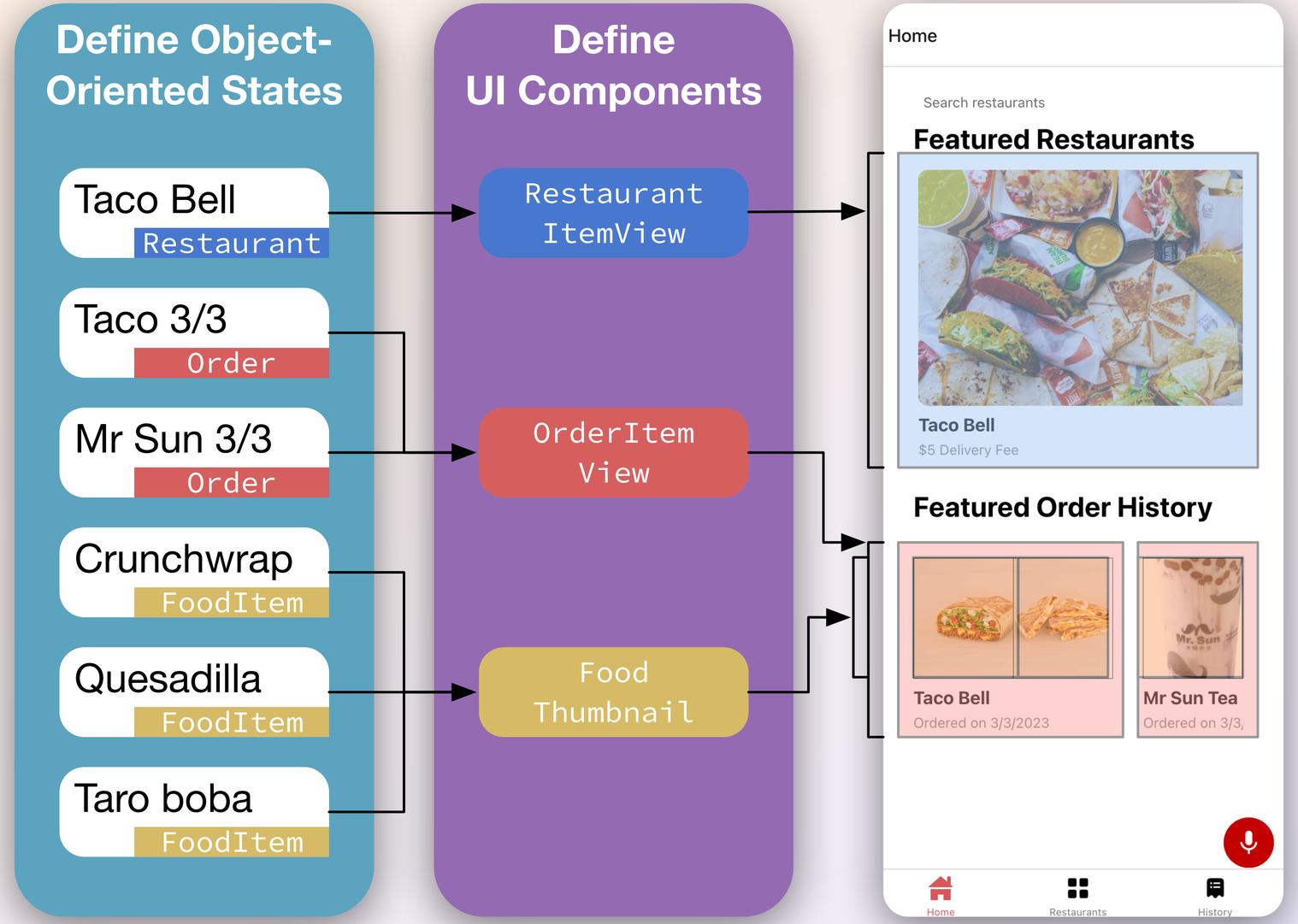
```
@DataClass()
class Recipe: GenieClass {
    @GenieProperty()
  name: String;
  img: Image;
    @GenieProperty()
  loved: boolean;

  @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

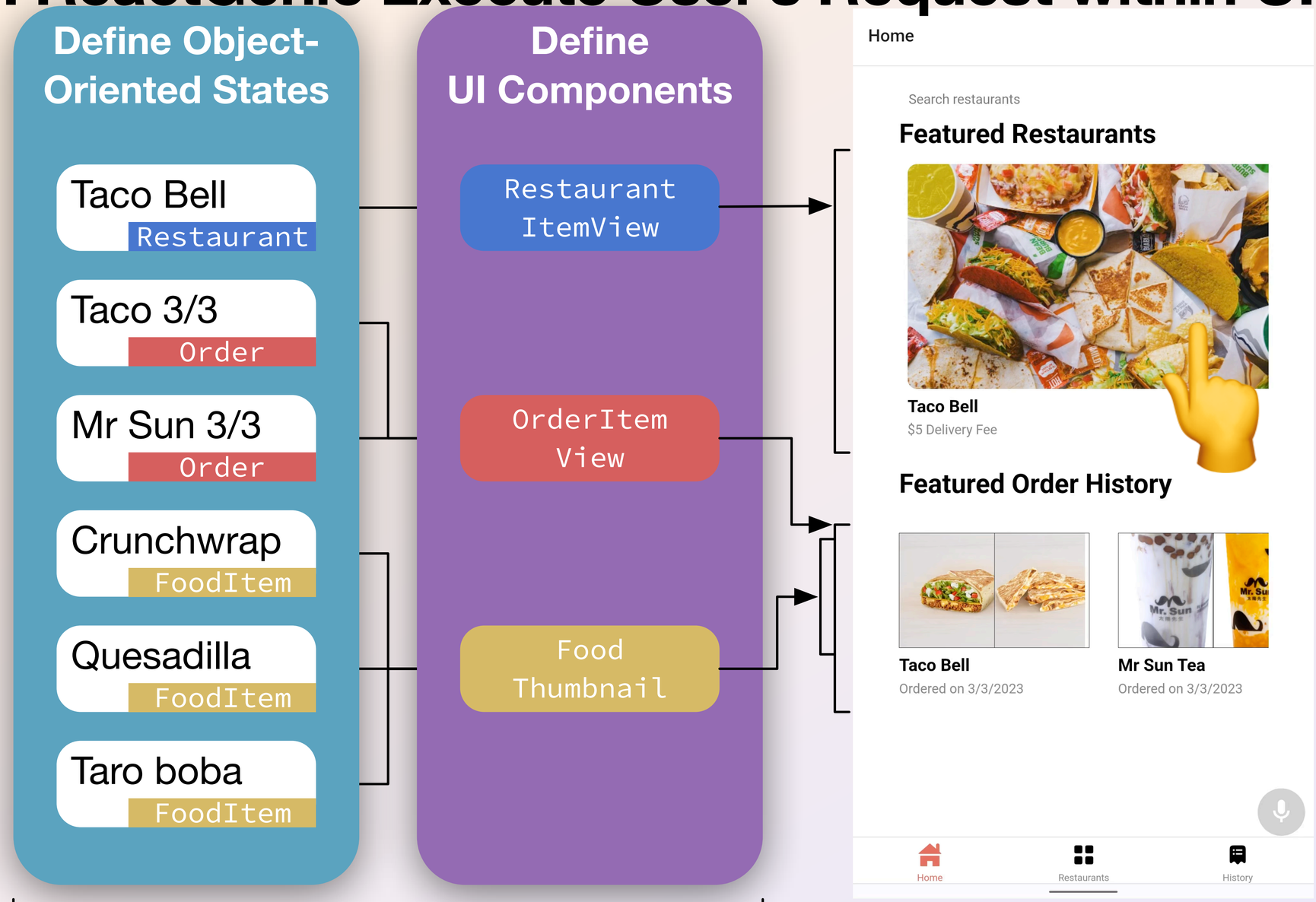
Recap: ReactGenie Uses Declarative UI Architecture for Ease of Development



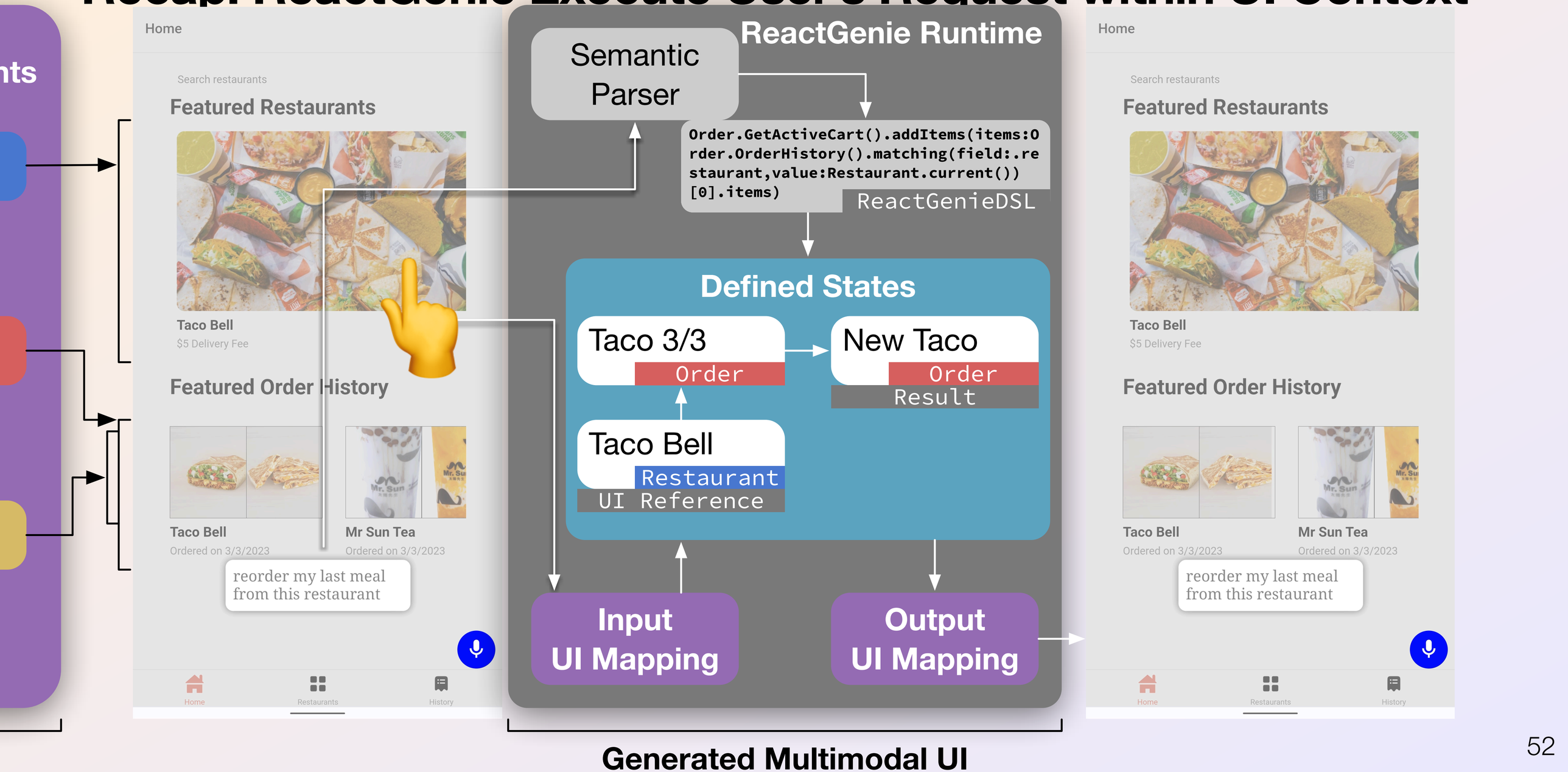
Recap: ReactGenie Use Annotations for Multimodal Integration

```
@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



Recap: ReactGenie Execute User's Request within UI Context

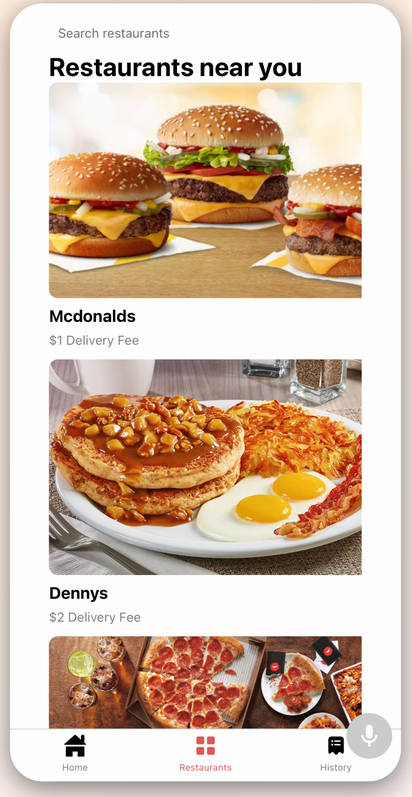


Generated Multimodal UI

How good is ReactGenie as a framework?

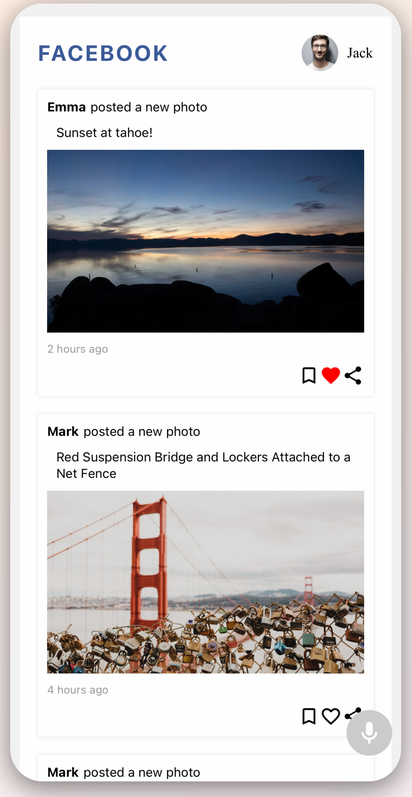
- For developers:
  - D-RQ1: Assessing the **expressiveness** of ReactGenie
  - D-RQ2: Development time for **expert** developers
  - D-RQ3: Ease of learning and usability for **novice** developers

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



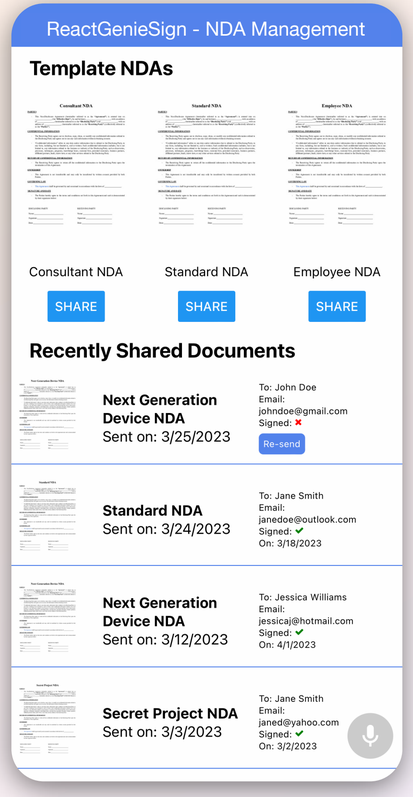
ReactGenieFoodOrdering

Add three of this to my cart



ReactGenieSocial

Show me posts from Mark that I have liked before.

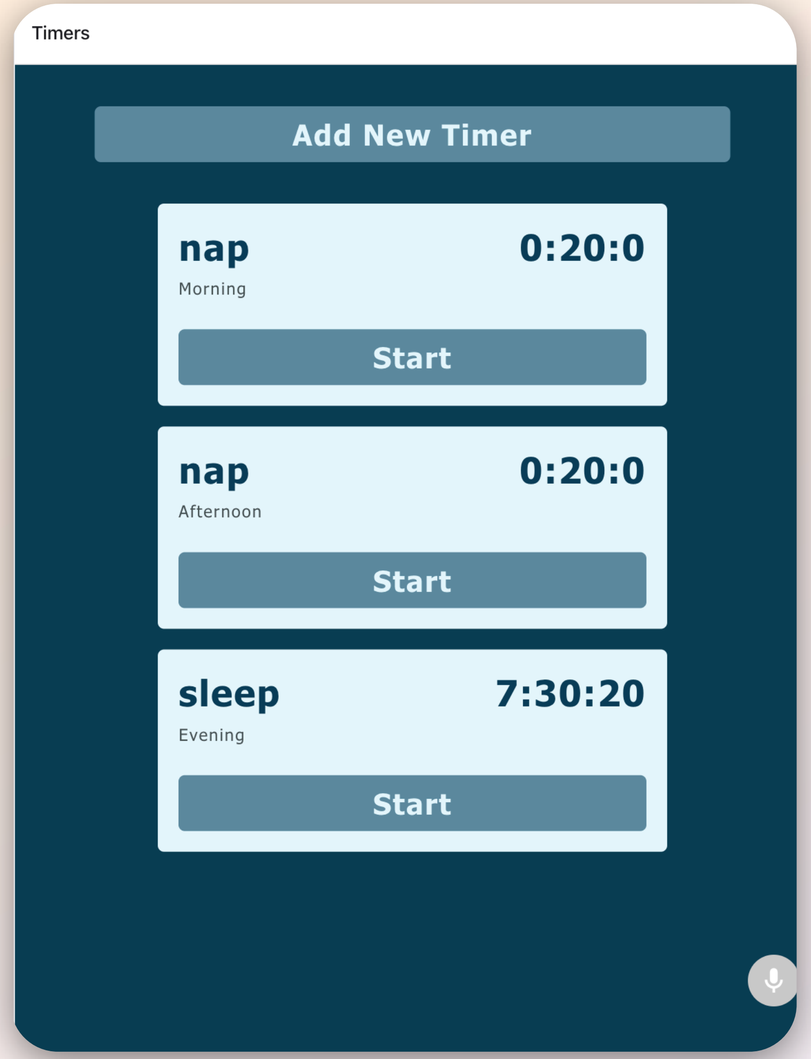


ReactGenieSign

Only show me request through this email

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

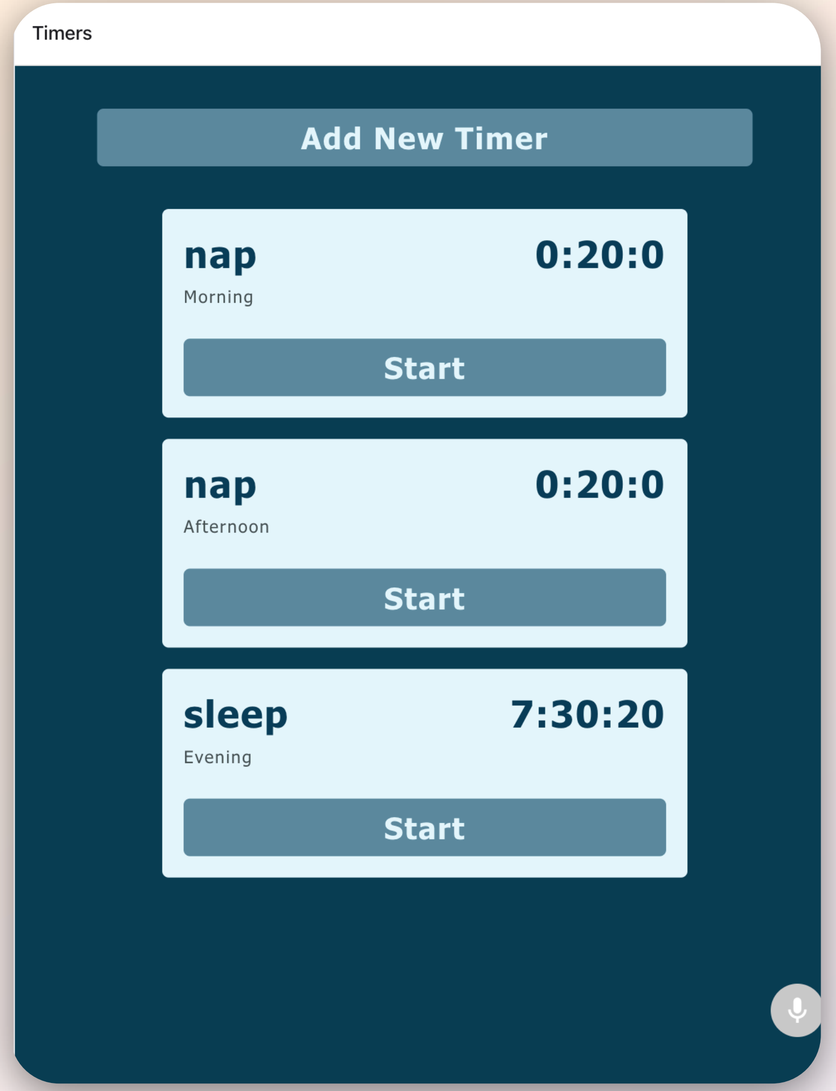
F-RQ2: Expert building an app: a Timer



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

ReactGenieTimer

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



ReactGenieTimer

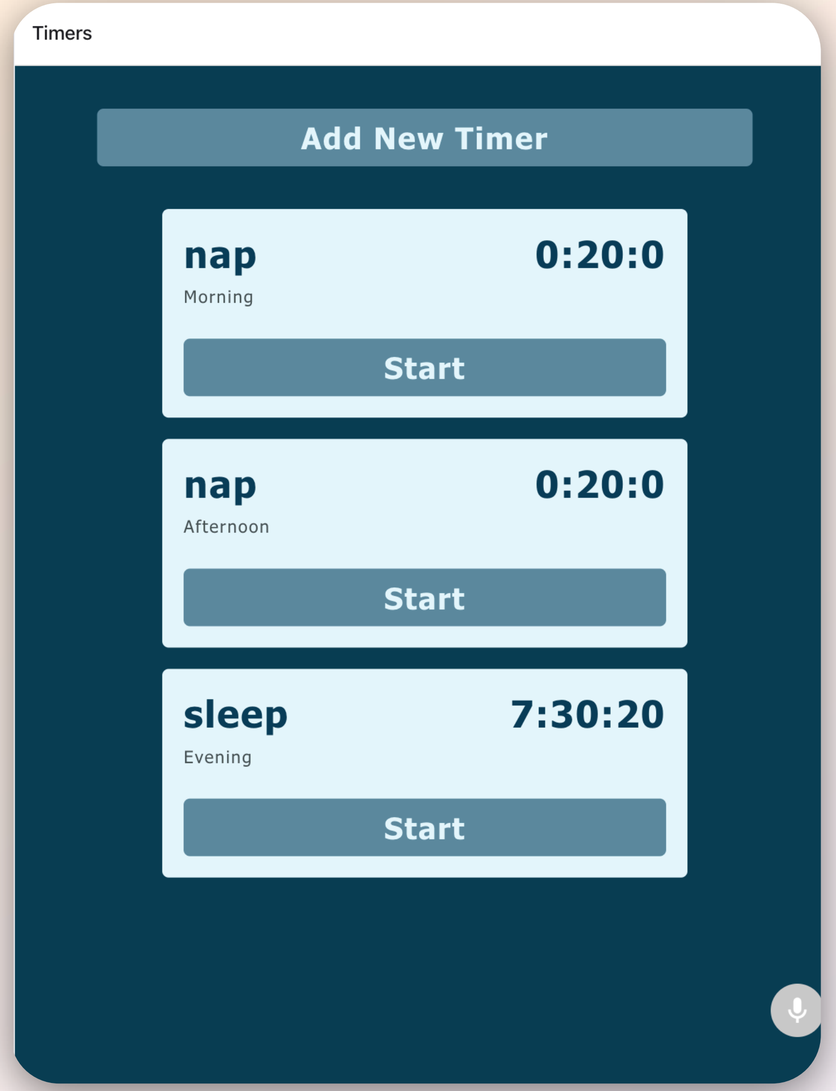
Metric	ReactGenie	GPT-3 Function Calling
Time to Develop (minutes)	45	177
Additional Lines of Code	119	523
Features Supported	Touch, Complex Commands, Navigation	Limited Support

Less time

Less code

More features

Novice Developer Study on ReactGenie Usability (F-RQ3)



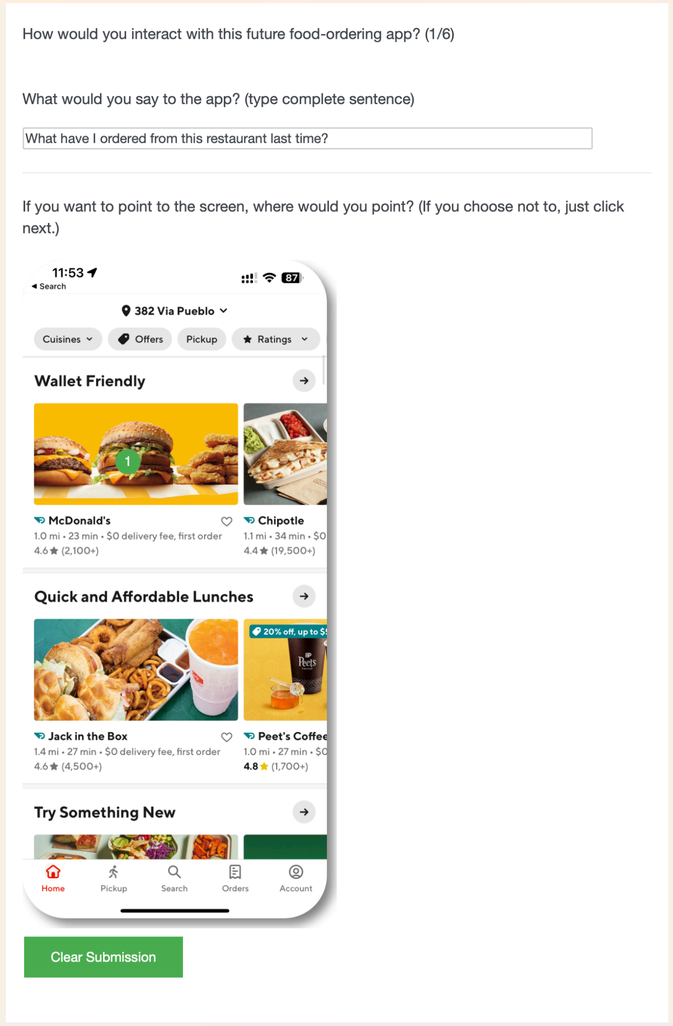
ReactGenieTimer

- 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?

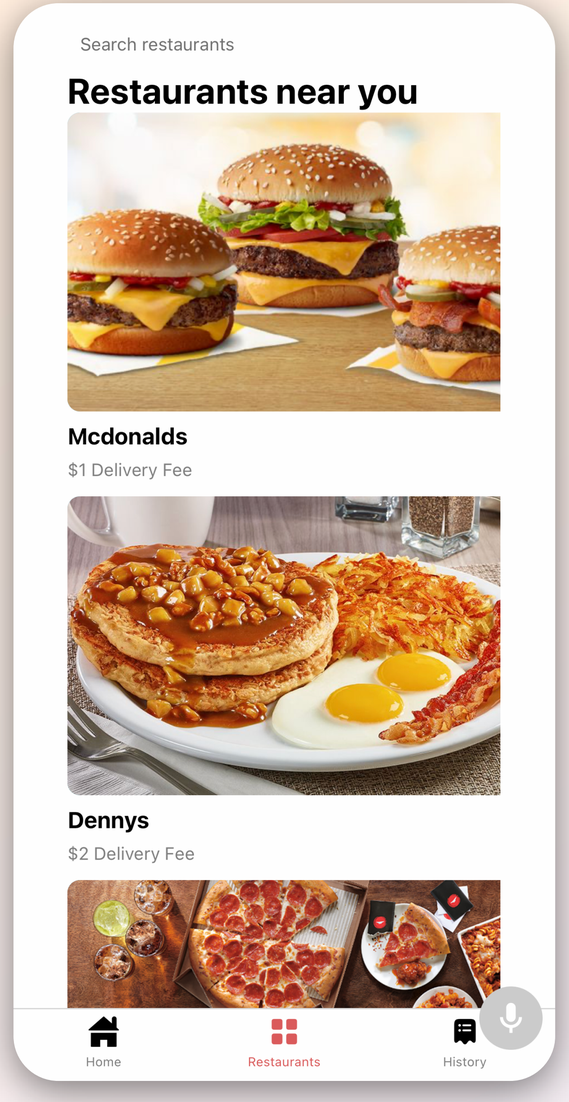
- For users:
  - U-RQ1: **Parser performance** with natural language commands
  - U-RQ2: **Usability and efficiency** of multimodal UIs generated by ReactGenie

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



- 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

Conclusions

- 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.