

Multimodal Applications

CS 224V

Monica Lam, Jackie (Junrui) Yang

Lecture Goals

- Why do we need multimodal interactions?
- Introduction to existing multimodal interactions
- Three problems for multimodal app development
- Introduction to ReactGenie: a multimodal app development framework
- Evaluation of ReactGenie

Lecture Goals

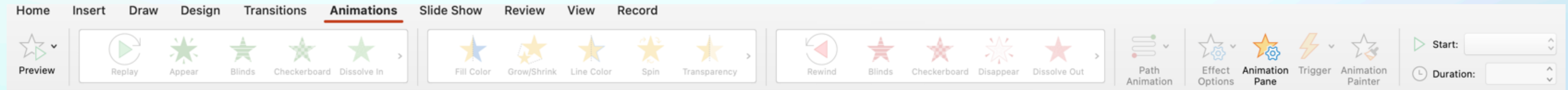
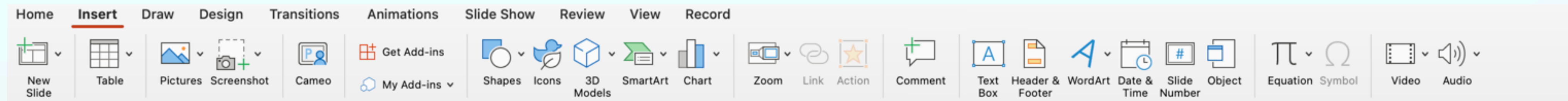
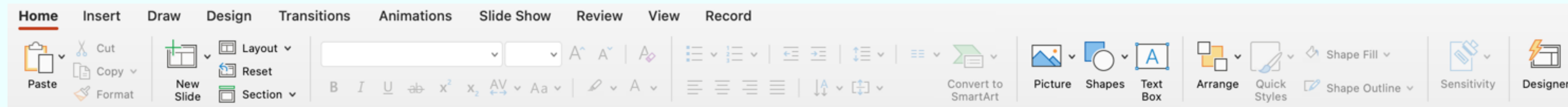
- **Why do we need multimodal interactions?**
- Introduction to existing multimodal interactions
- Three problems for multimodal app development
- Introduction to ReactGenie: a multimodal app development framework
- Evaluation of ReactGenie

Why Multimodal Interaction

- Limitations of voice assistants
 - Lack of discovery
 - Slow for long text
 - Cannot show graphics
 - Interactive analog adjustments
- Can we combine the best of both worlds?
- Advantages of GUI
 - Explore the possible functions in the menus
 - Glance over long lists and paragraphs
 - Images, videos, PowerPoint
 - CAD tools, PhotoShop, WYSIWYG

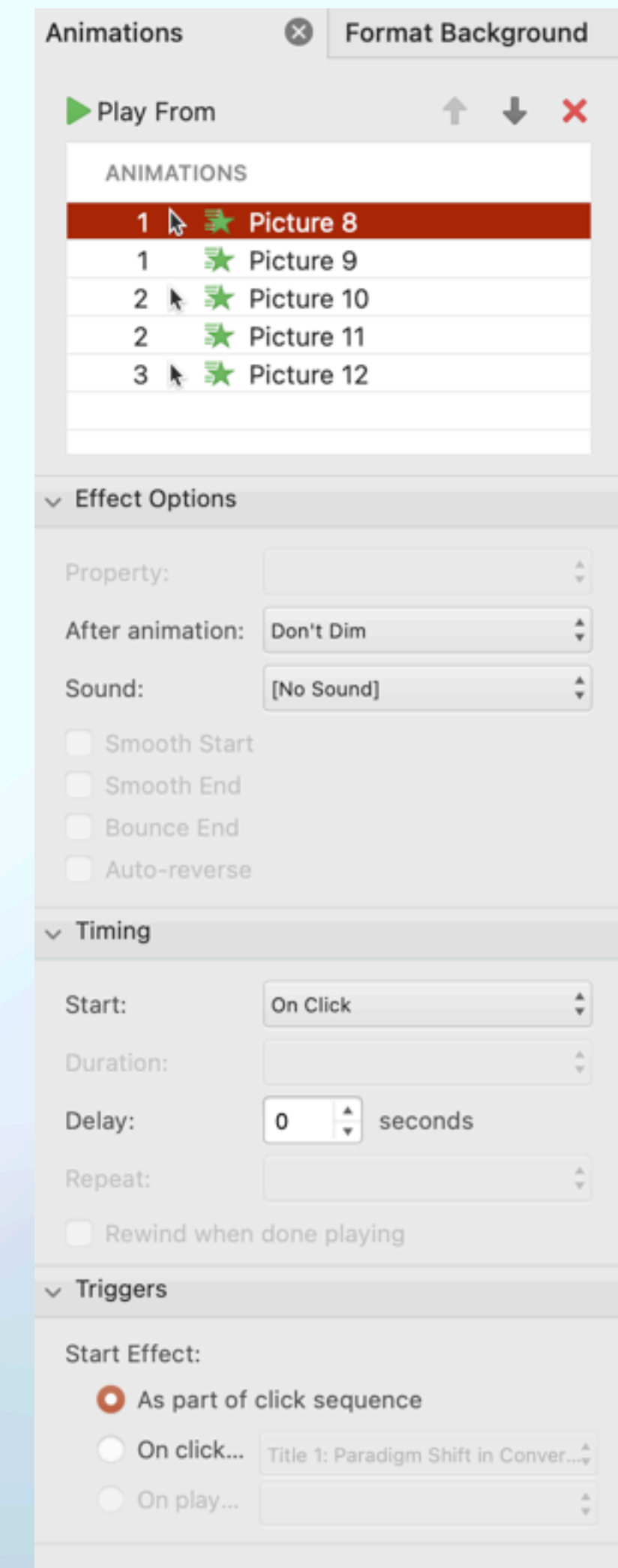
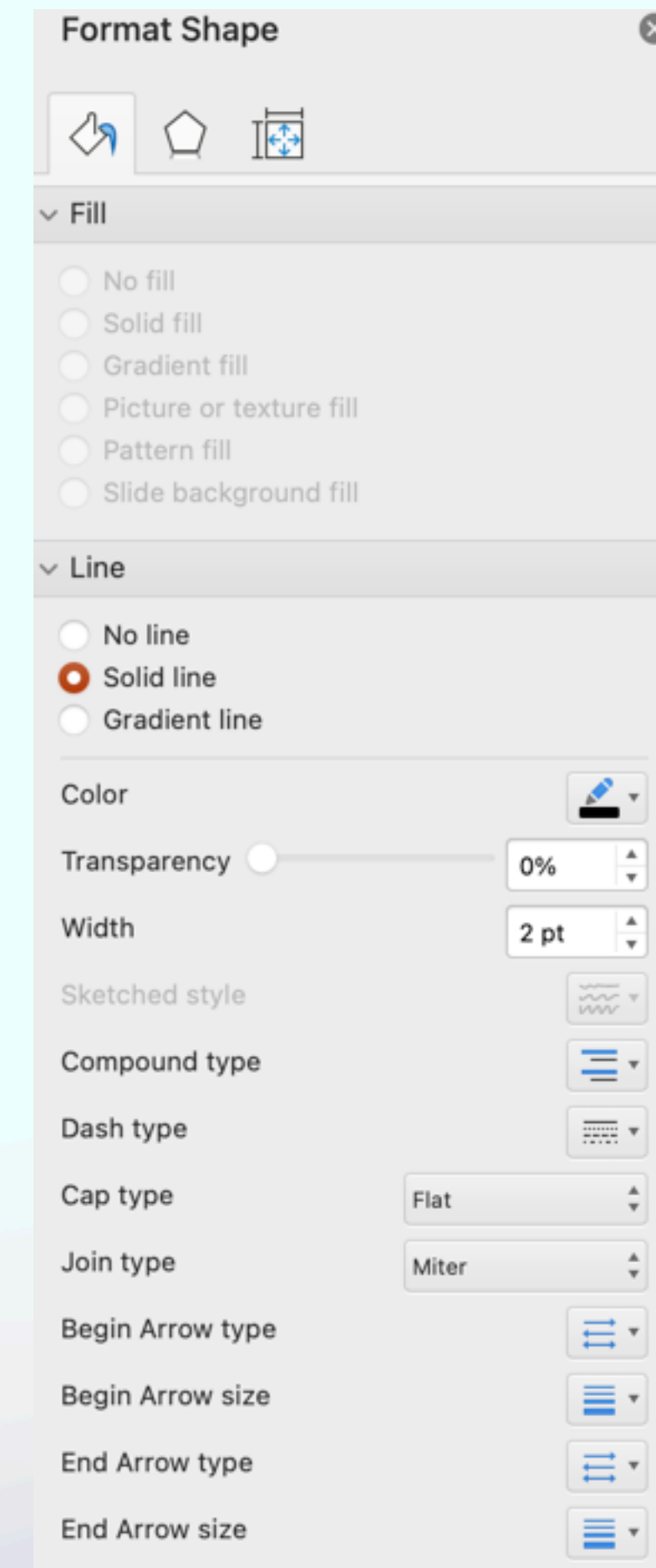
Example: Powerpoint

- 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.

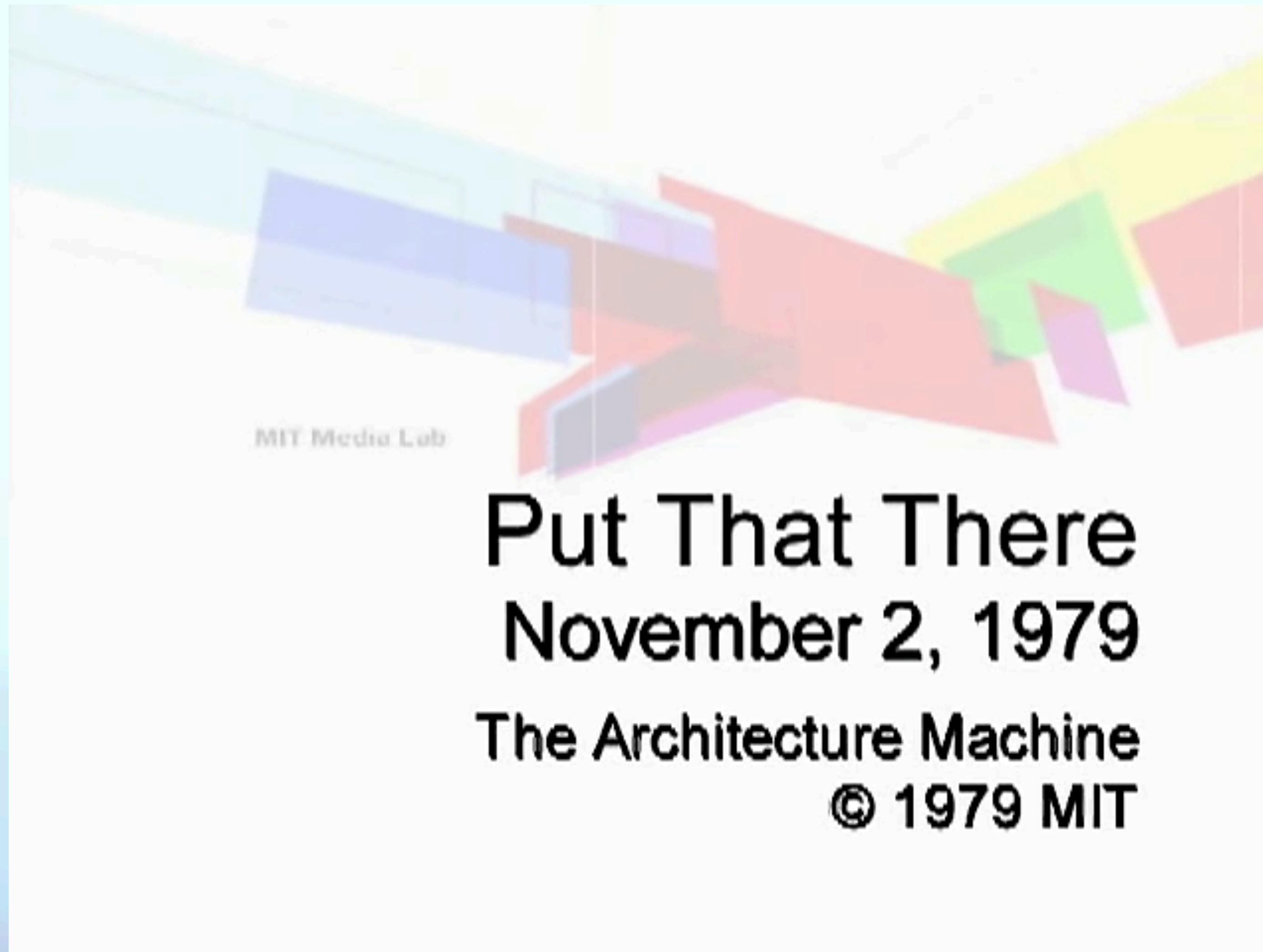
Solution: Multimodal Applications

- User: Much faster if we can give the high-level intent
 - Don't have to figure out which menus to open and where the features are listed
 - Don't have to learn the exact feature names
 - Don't have to do repetitive work on each slide/bullet point, etc.

Lecture Goals

- Why do we need multimodal interactions?
- **Introduction to existing multimodal interactions**
- Three problems for multimodal app development
- Introduction to ReactGenie: a multimodal app development framework
- Evaluation of ReactGenie

Multimodal Interaction: Voice + Gesture



Put-That-There

1982 Bolt et al.

Multimodal Interaction: Voice + Gesture



QuickSet

1997 Cohen et al.

Multimodal interfaces includes:

- **Input**
 - **Visual (Face location, Gaze, Facial expression, Lipreading, Sign language)**
 - **Auditory (Speech input)**
 - **Touch (Pressure, Location and selection, Gesture)**
 - **Other sensors (Sensor-based motion capture)**

Multimodal input: Gaze + Gesture

Example Applications

Gaze + Gesture

2015 Xiao et al.

Multimodal interfaces includes:

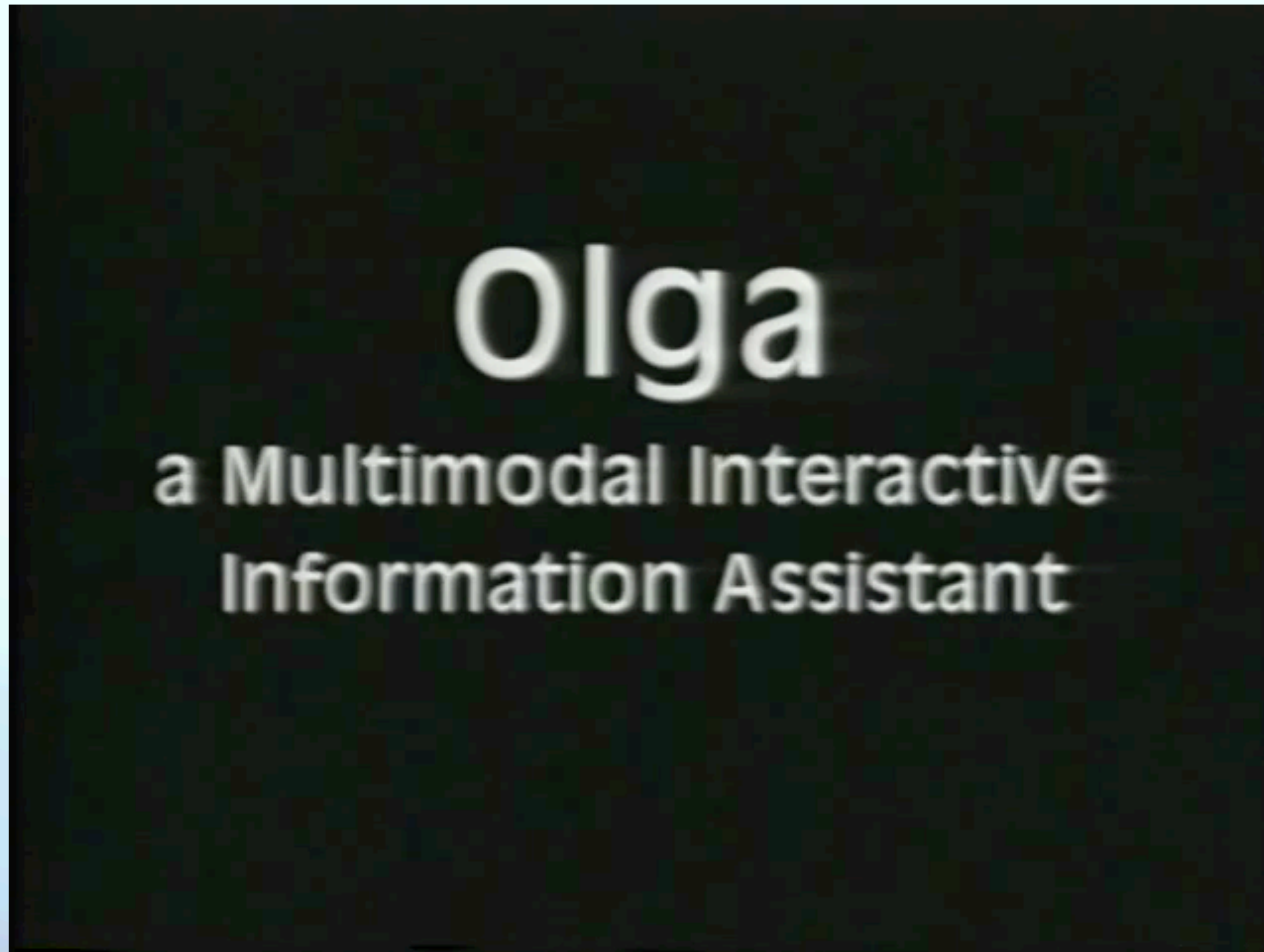
- Input

- Visual (Face location, Gaze, Facial expression, Lipreading, Sign language)
- Auditory (Speech input)
- Touch (Pressure, Location and selection, Gesture)
- Other sensors (Sensor-based motion capture)

- **Output**

- **Graphics (Lists, Buttons, Charts, Pictures)**
- **Sound (Speech, Music, Sound Effects)**
- **Touch (Haptics)**
- **Other outputs (Temperatures, AR/VR)**

Multimodal Output: Graphics + Sound



OLGA

1998 Sundblad et al.

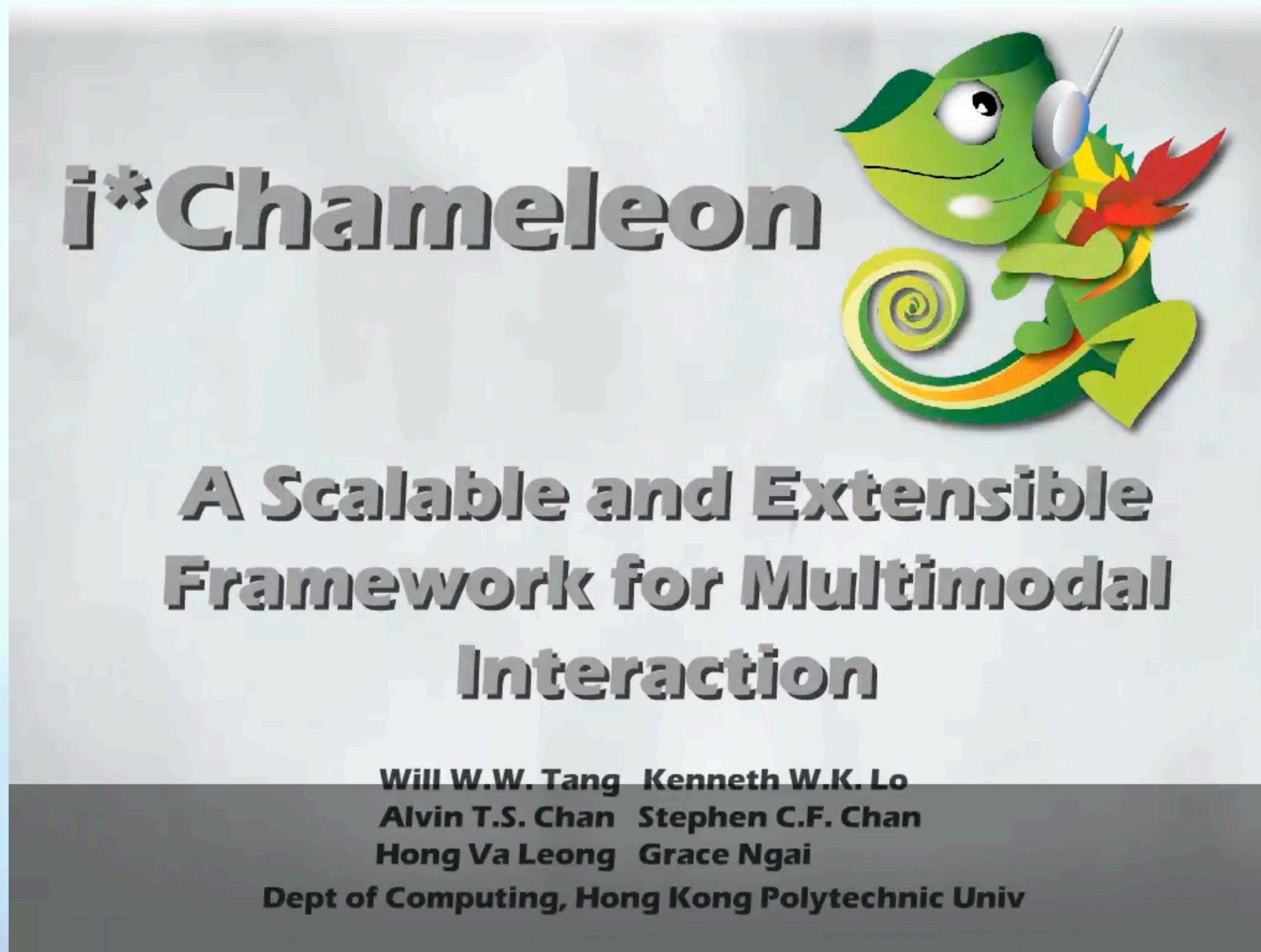
Multimodal interfaces can (Oviatt 2000):

- Flexibility: Permit combination and alternating input modes.
- Efficiency: Support effective manipulation, especially graphical information.
- Clarity: Simplify and improve speech interfaces.
- Precision: Offer accurate spatial information.
- Options: Provide multiple interaction techniques.
- Error Reduction: Enhance error avoidance and resolution.
- Inclusivity: Accommodate diverse users and tasks.
- Adaptability: Resilient to changing environmental conditions.
- Accessibility: Cater to individual needs, including handicaps.
- Balance: Help prevent overuse of any single mode.

Multimodal interfaces can (Oviatt 2000):

- Flexibility: Permit combination and alternating input modes.
- Efficiency: Support effective manipulation, especially graphical information.
- Clarity: Simplify and improve speech interfaces.
- Error Reduction: Enhance error avoidance and resolution.

Flexibility of Multimodal Interfaces



i*Chameleon

2013 Tang et al.

Efficiency of Multimodal Interfaces



PixelTone

2013 Laput et al.

Clarity of Multimodal Interfaces

ReMap: Lowering the Barrier to Help-Seeking with Multimodal Search

C. Ailie Fraser, Julia M. Markel, N. James Basa, Mira Dontcheva, Scott Klemmer



UC San Diego
The Design Lab

ReMap

2020 Fraser et al.

Error Reduction of Multimodal Interfaces

Voice and Touch Based Error-tolerant Multimodal Text Editing and Correction for Smartphones

Maozheng Zhao¹ , Wenzhe Cui¹ , I.V. Ramakrishnan¹, Shumin Zhai², Xiaojun Bi¹

¹*Stony Brook University*, ²*Google LLC*



Text input
error-correction

2021 Zhao et al.

Lecture Goals

- Why do we need multimodal interactions?
- Introduction to existing 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
- Introduction to ReactGenie: a multimodal app development framework
- Evaluation of ReactGenie

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 feasible with simple function calling!!!

Compositionally of multimodal commands

ReactGenieDSL
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")
```

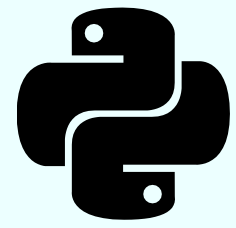
- Make everything right aligned on this slide.

```
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")
```


Solution to compositionally: ReactGenieDSL



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

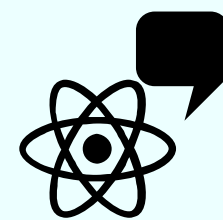


SQL

```
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

ReactGenieDSL

Delete Shapes above this

Automatically distributed
to each element

Expressiveness of ReactGenieDSL

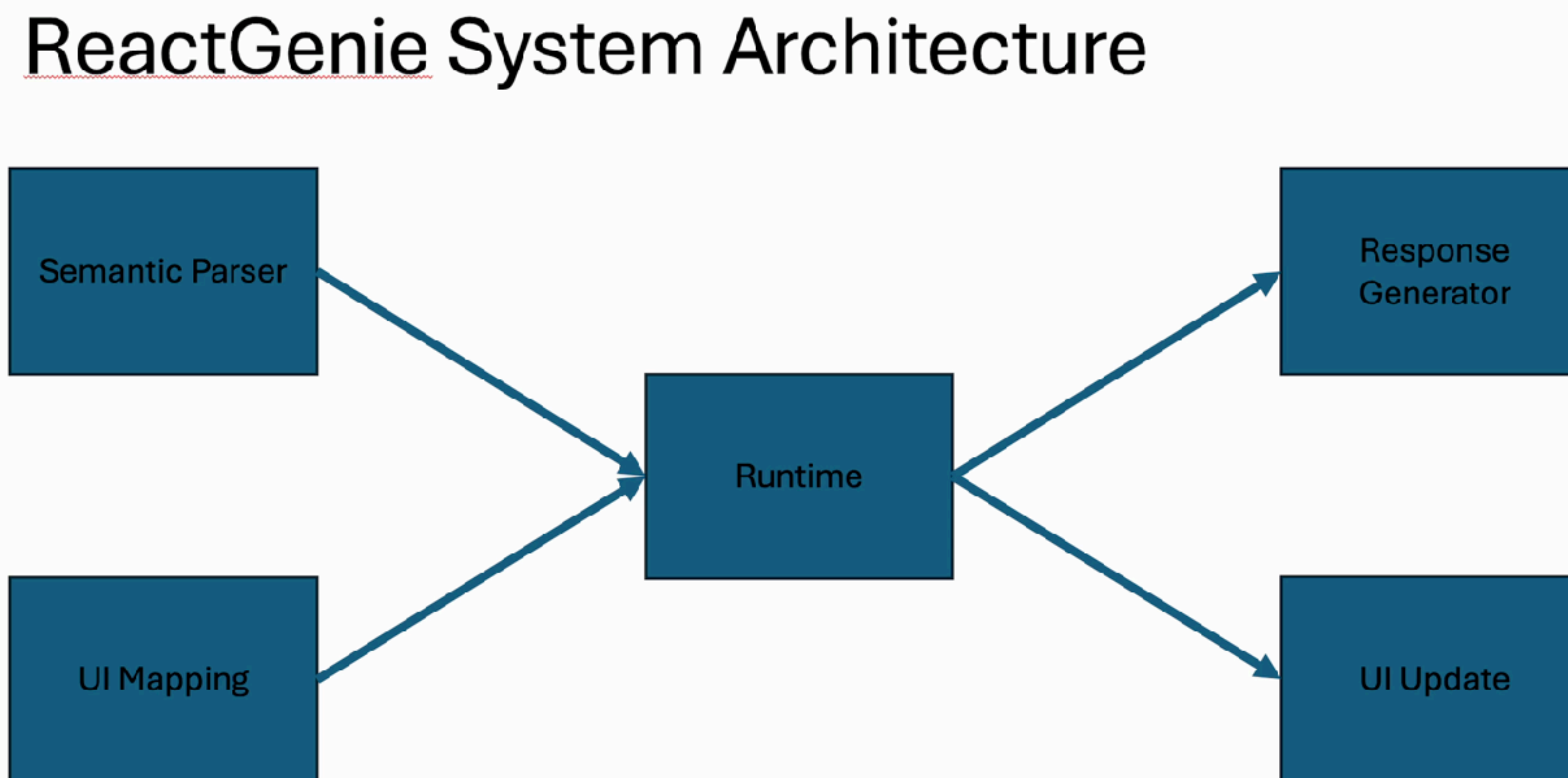
In a series of apps

Change background color for all the yellow shapes to orange

VerbObject ModifierObjectVerb Modifier

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

Feature of RGDSL	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)



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

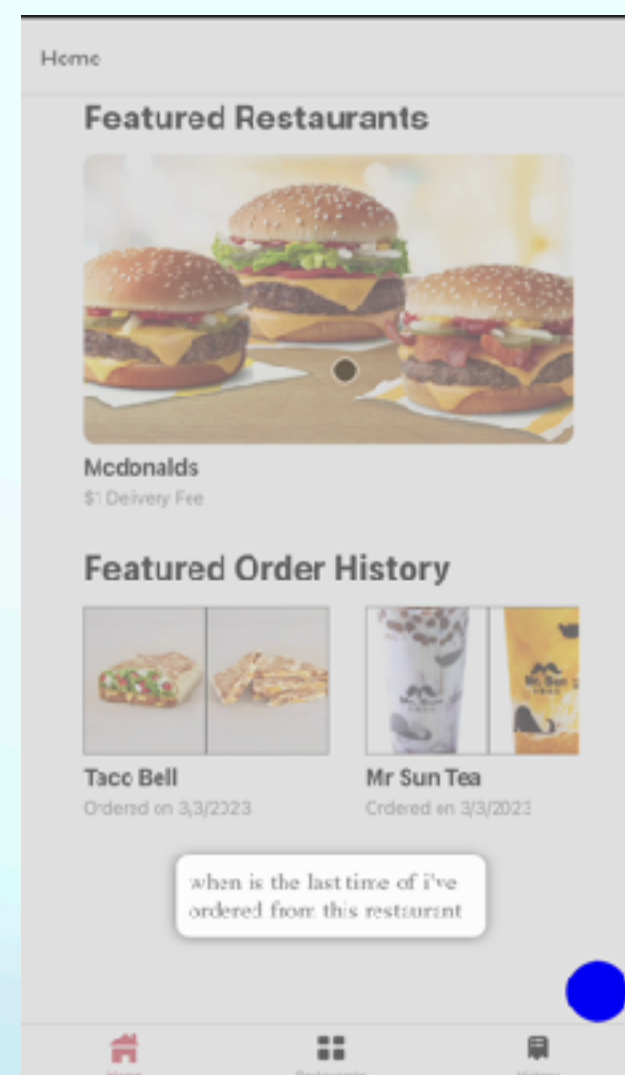
ReactGenie
Annotations

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

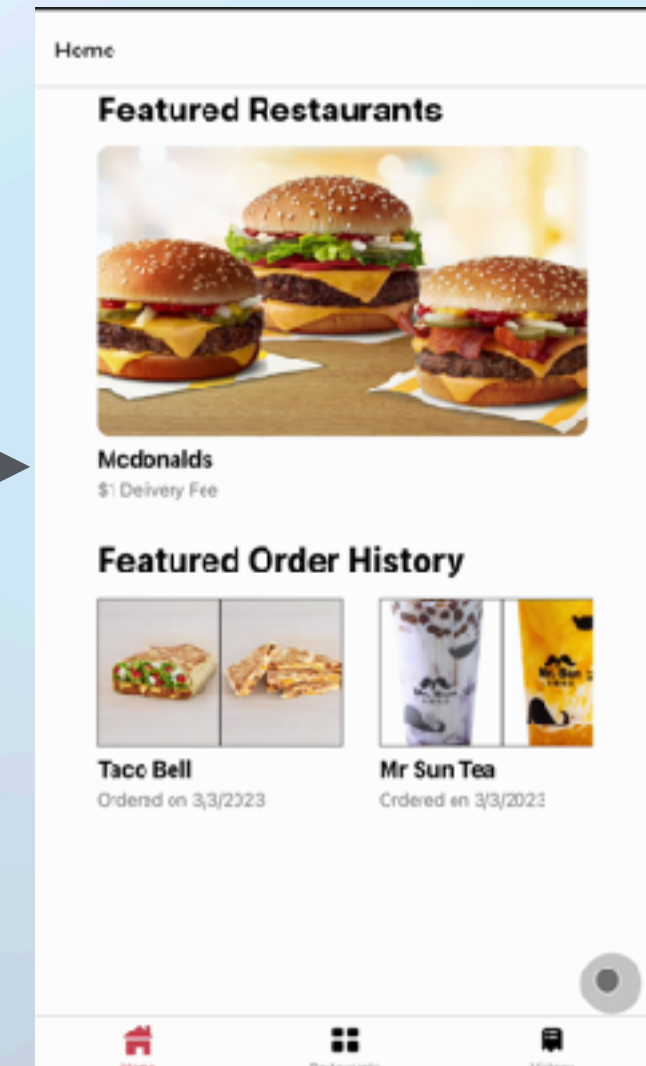
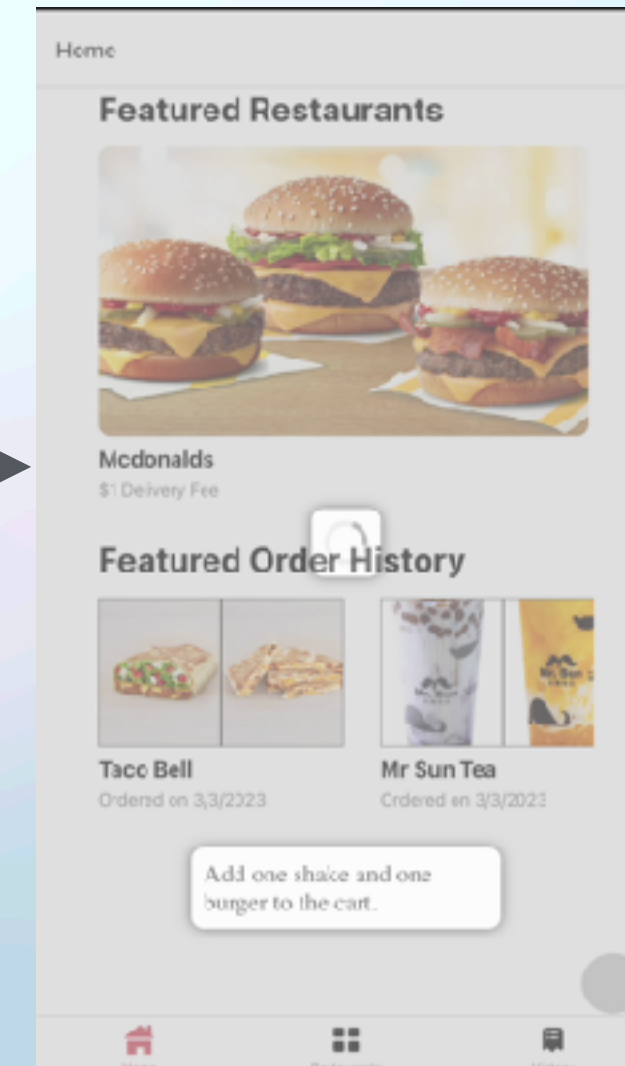
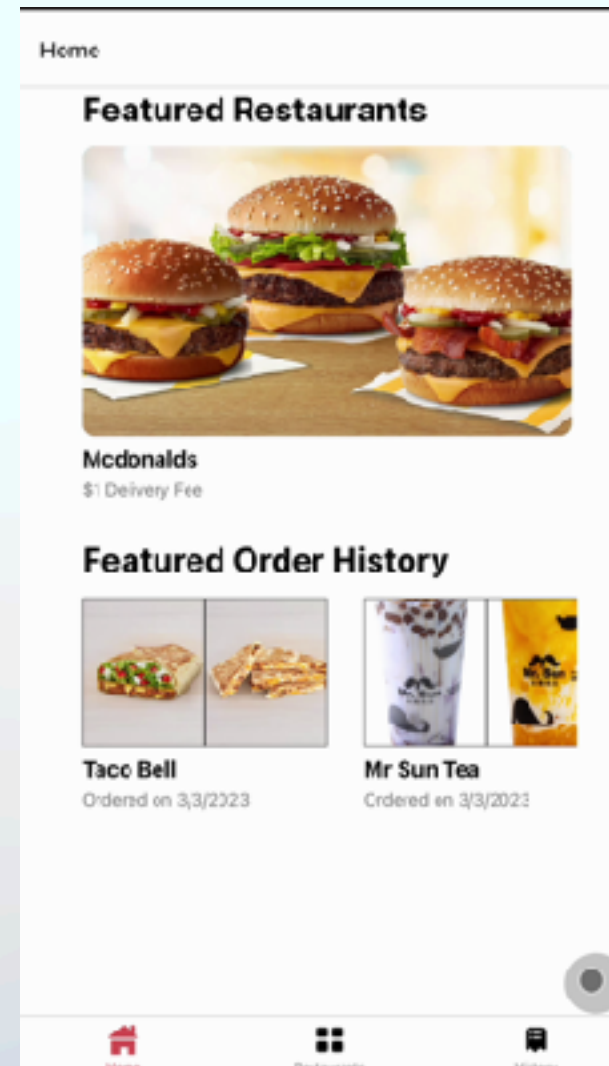
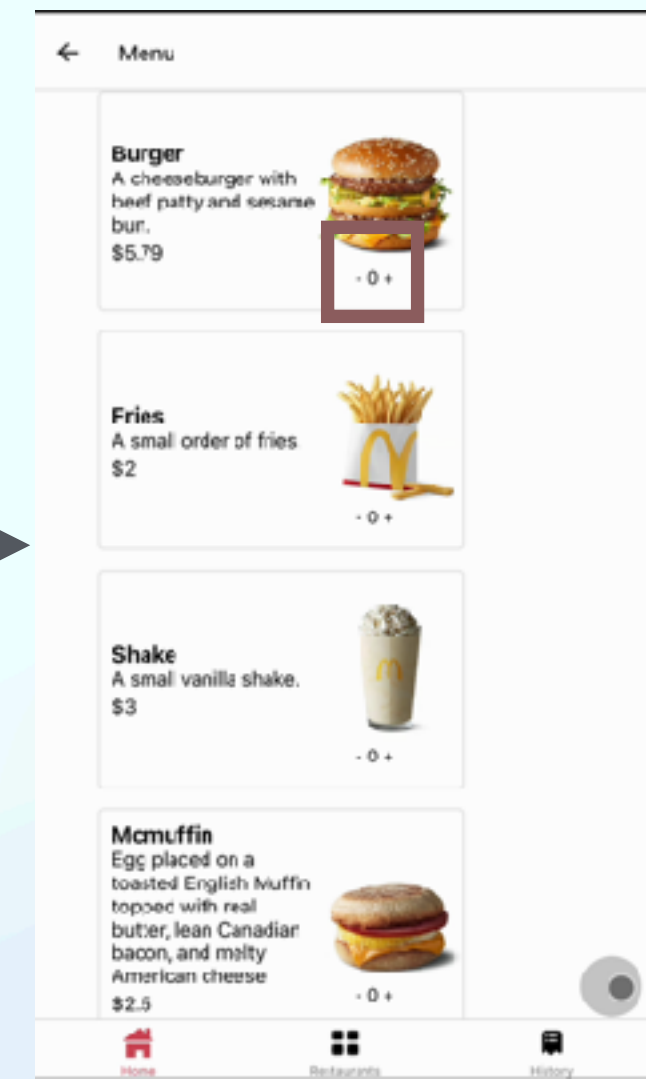
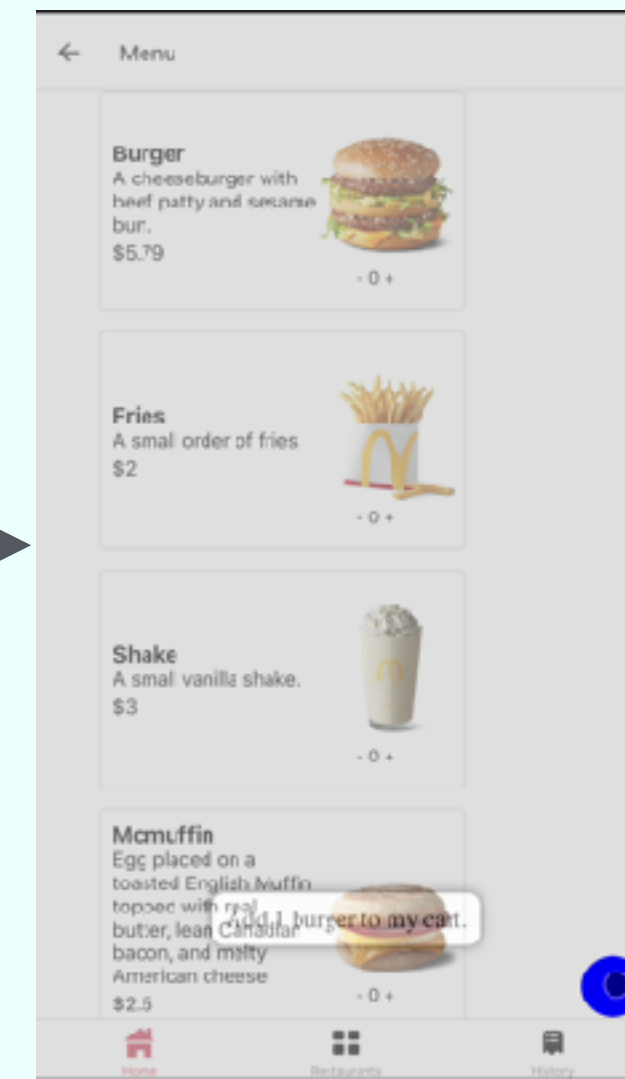
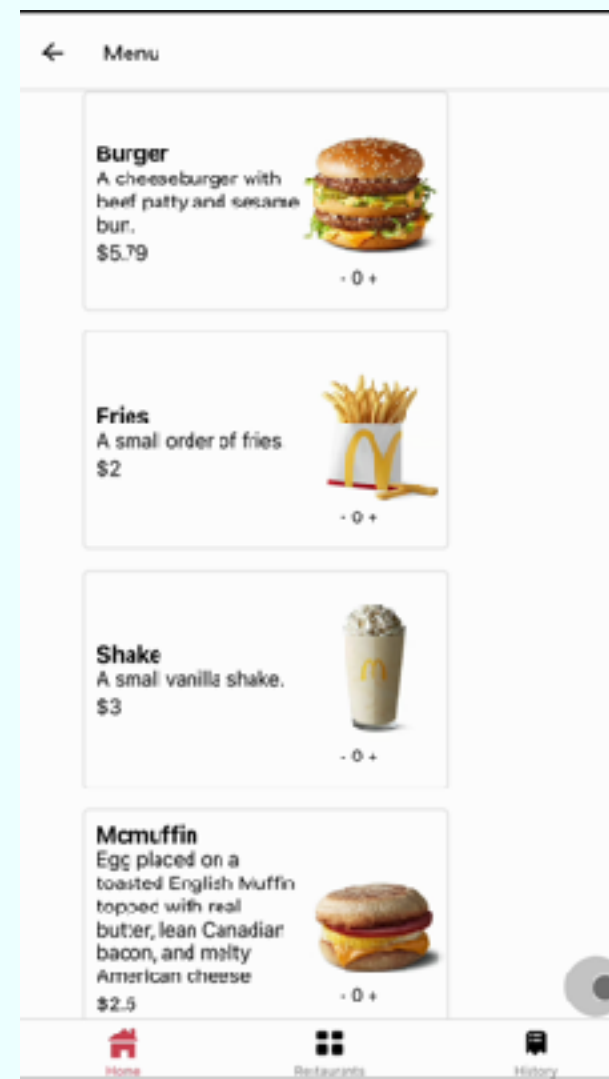
React App Logic Code

When using a ReactGenie app

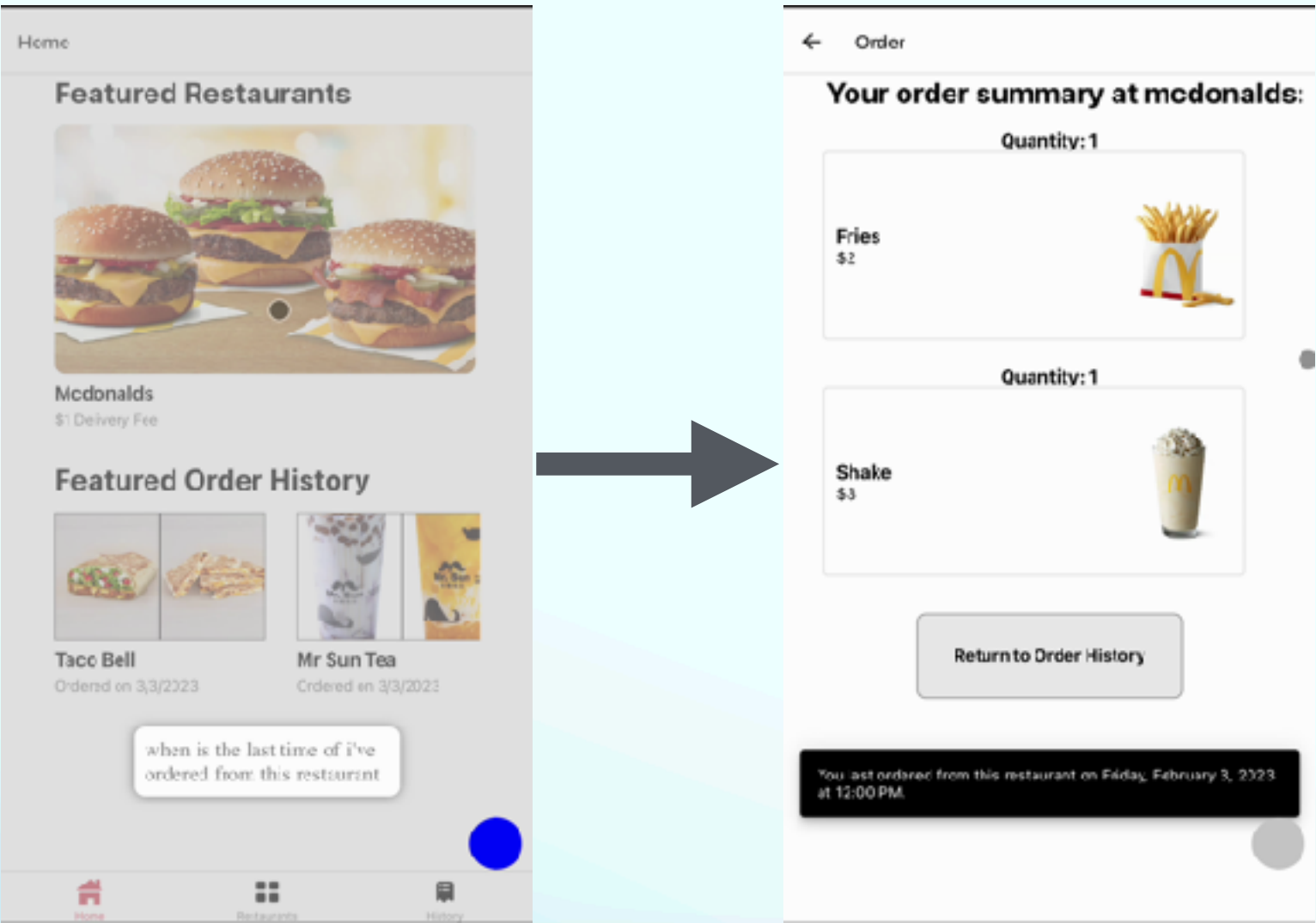
How can the user know what happened? 🙋



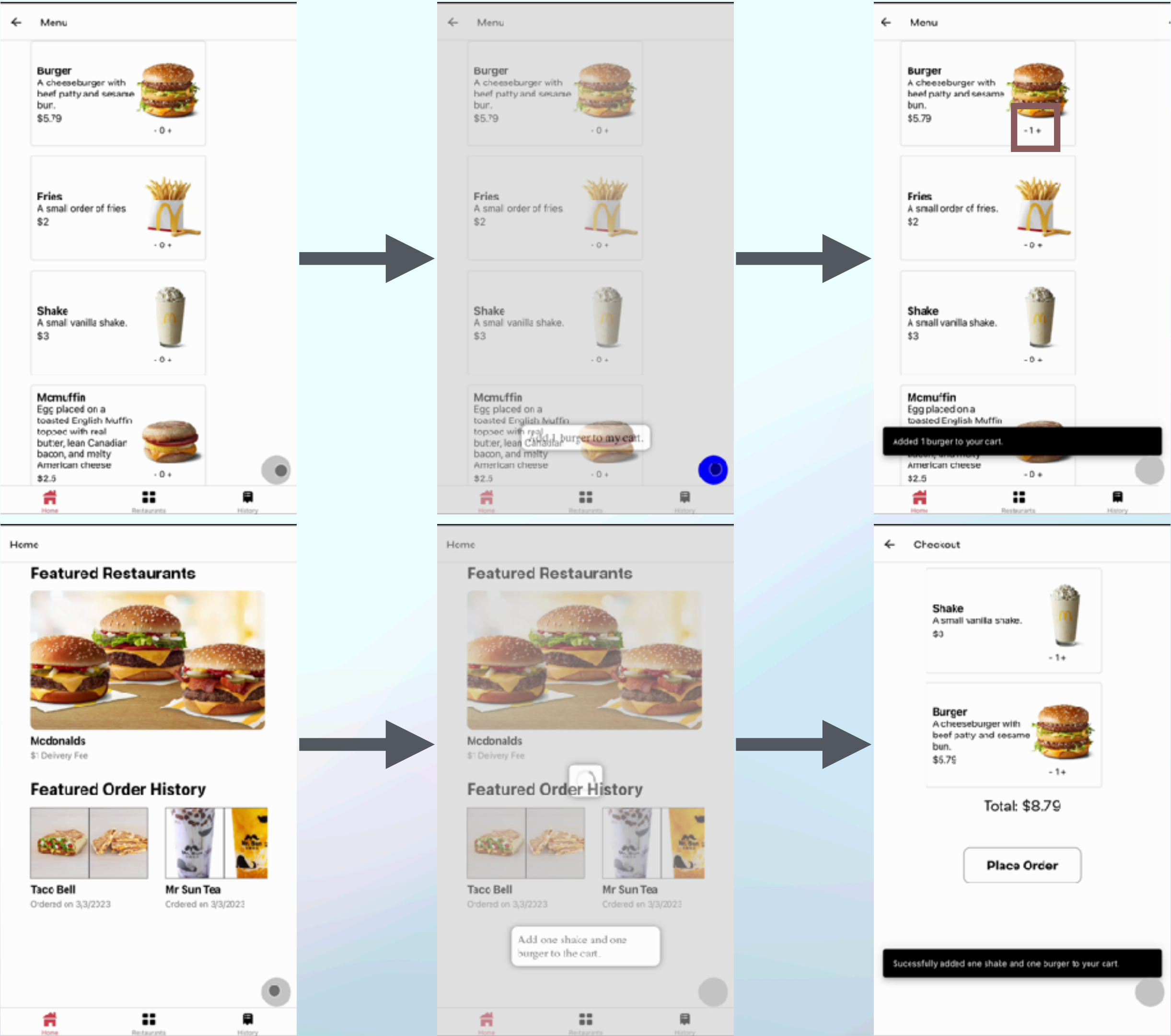
Which restaurant?



Problem 3: Allows for interchangeable and simultaneous multimodal interactions



UI input mapping



Navigation

UI updates

Problem 3: Allows for interchangeable and simultaneous multimodal interactions

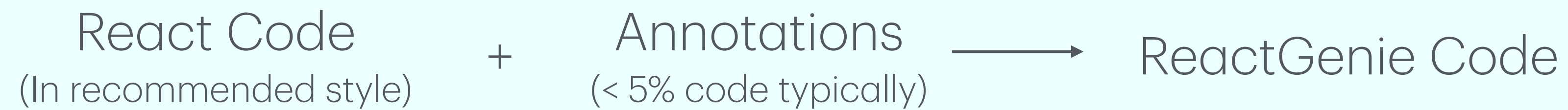
- Multimodal Input
 - UI Input Mapping: Getting what object that I'm touching
- Multimodal Output
 - Resulting objects are on-screen — UI Updates: Update the values on screen
 - Resulting objects are off-screen — Navigation: Navigate to the page with results

Lecture Goals

- Why do we need multimodal interactions?
- Introduction to existing multimodal interactions?
- Three problems for multimodal app development
- **Introduction to ReactGenie: a multimodal app development framework**
- Evaluation of ReactGenie

What is ReactGenie?

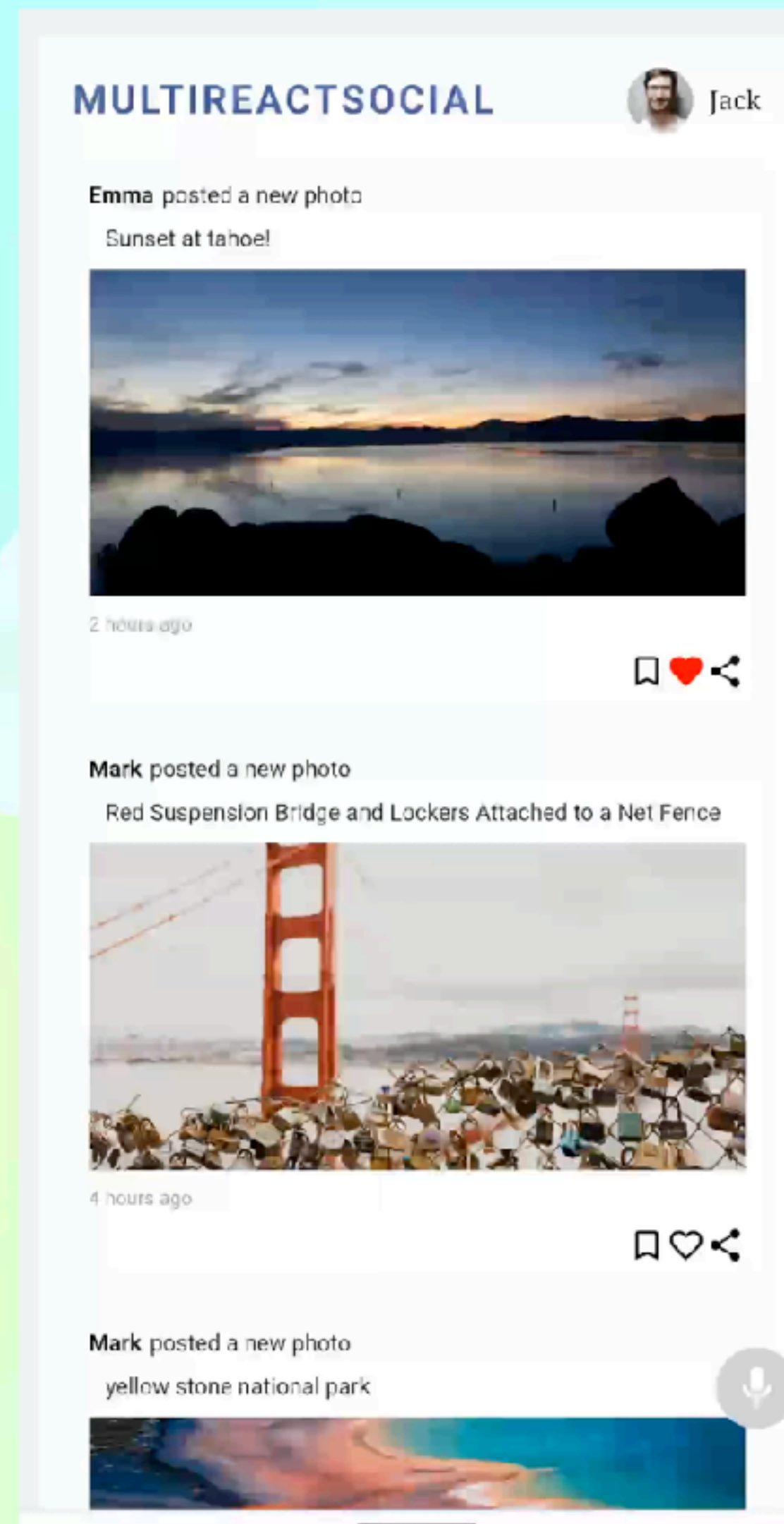
Ease of Development



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

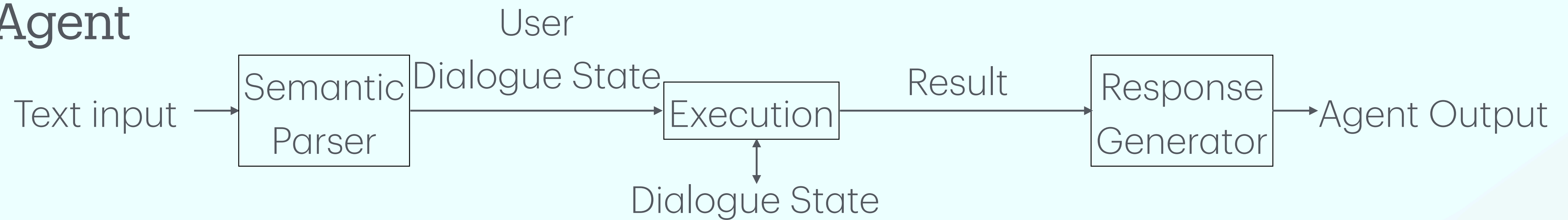
```
RecipeViewImpl = (recipe: Recipe) => {
    return (
        <div>
            <img image={recipe.img}/>
            <love loved={recipe.loved}
                onClick={()=>recipe.love()}/>
            <div> {recipe.name} </div>
        </div>
    )
}

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

Components Annotations:
Which components represent
which state classes

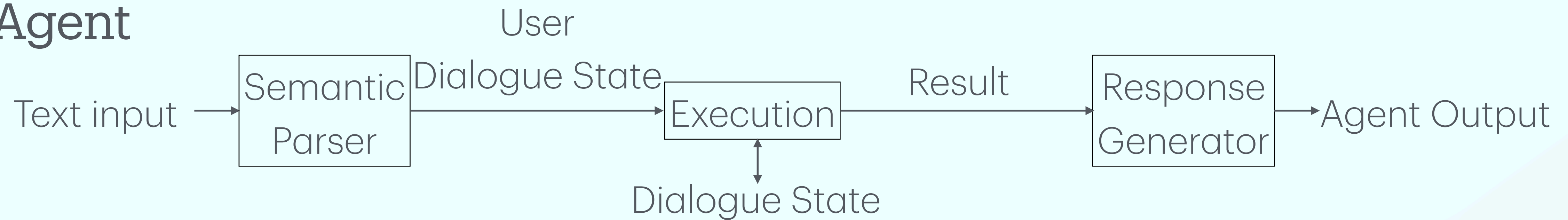
Recall the Agent Architecture (No Policies)

Agent

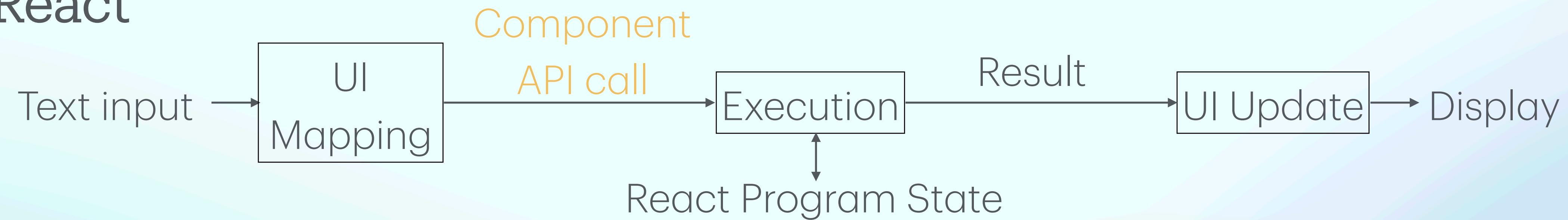


React has a Similar Architecture

Agent



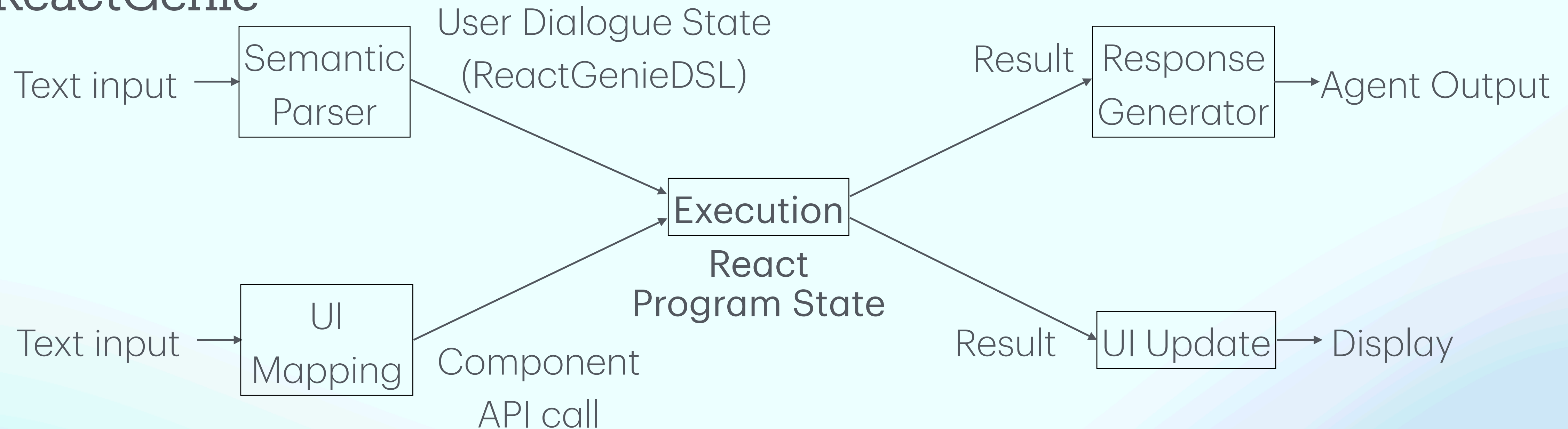
React



- Execution of React Programs
 - Updates the state of the variables
 - Navigates to a display page (all current values are displayed)

Multimodal Agent Architecture

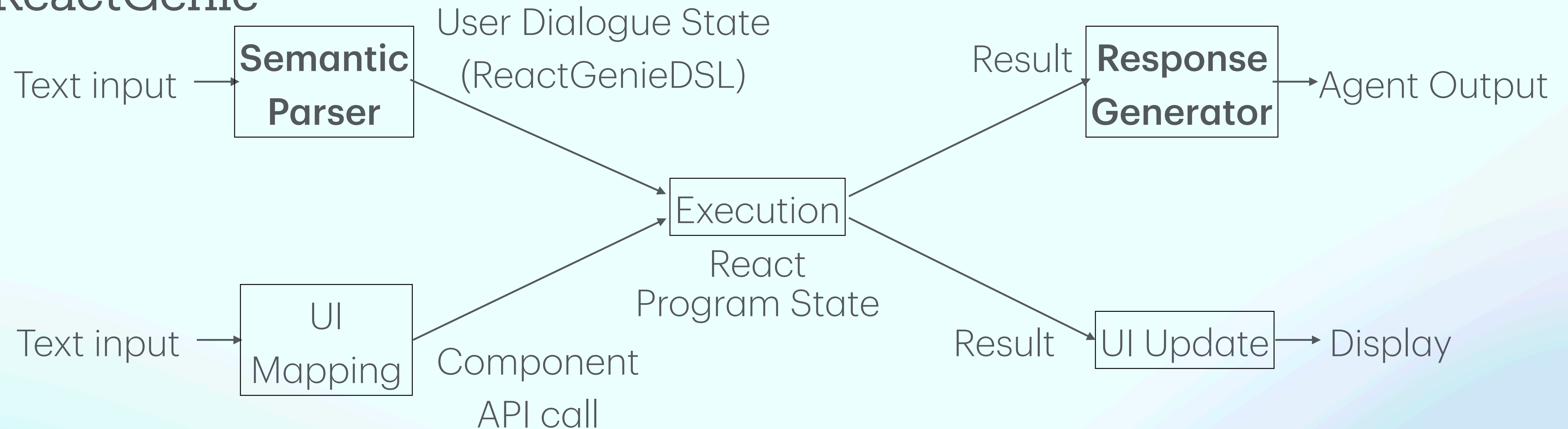
ReactGenie



- Execution of ReactGenie Programs
 - Updates the state of the variables
 - Navigates to a display page (all current values are displayed)
 - Generates the text result

Semantic Parser + Response Generator

ReactGenie



Revisiting ReactGenieDSL

ReactGenieDSL
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

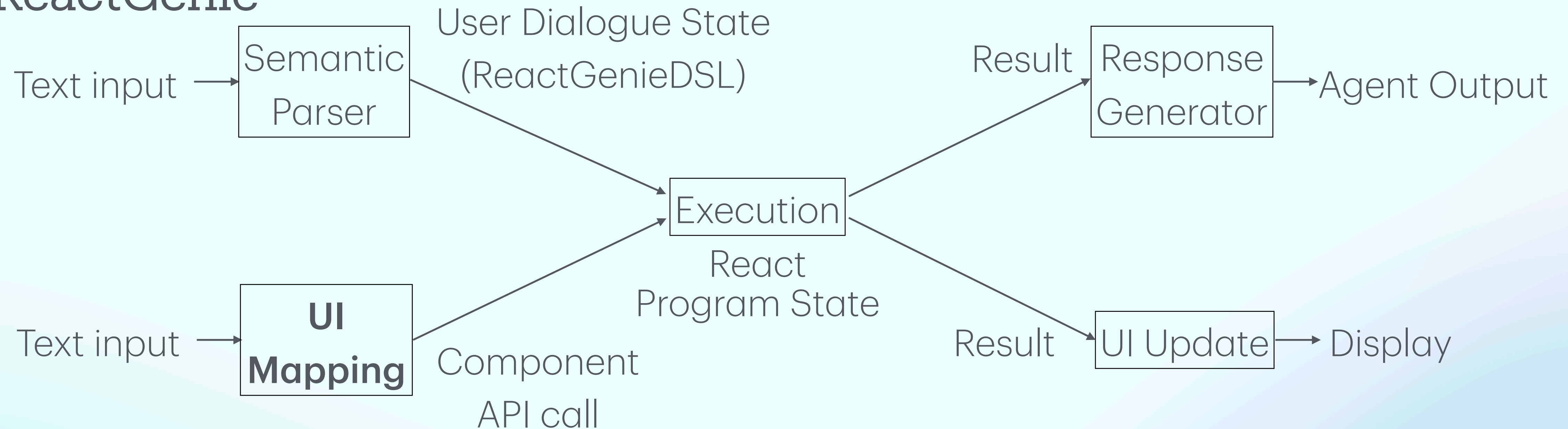
<pre>// Here are all the functions that we have class Restaurant { string name; string address; string cuisine; float rating; // All active restaurants static Restaurant[] all(); // The current restaurants static Restaurant current(); // Get a list of foods representing the menu from a restaurant Food[] menu; // Book reservations on date Reservation get_reservation(date: DateTime) }</pre>	Declaration
<pre>// Examples: user: get me the best restaurant in palo alto agent: Restaurant.all().matching(field: .address, value: "palo alto")...</pre>	Few-shot examples
<pre>// User interaction user: order the same burger that I ordered at mcDonald last time</pre>	Current interaction
<pre>parsed: Order.current.addFoods(foods: Order.all().matching...</pre>	Parsed result

LLM-Based Response Generator

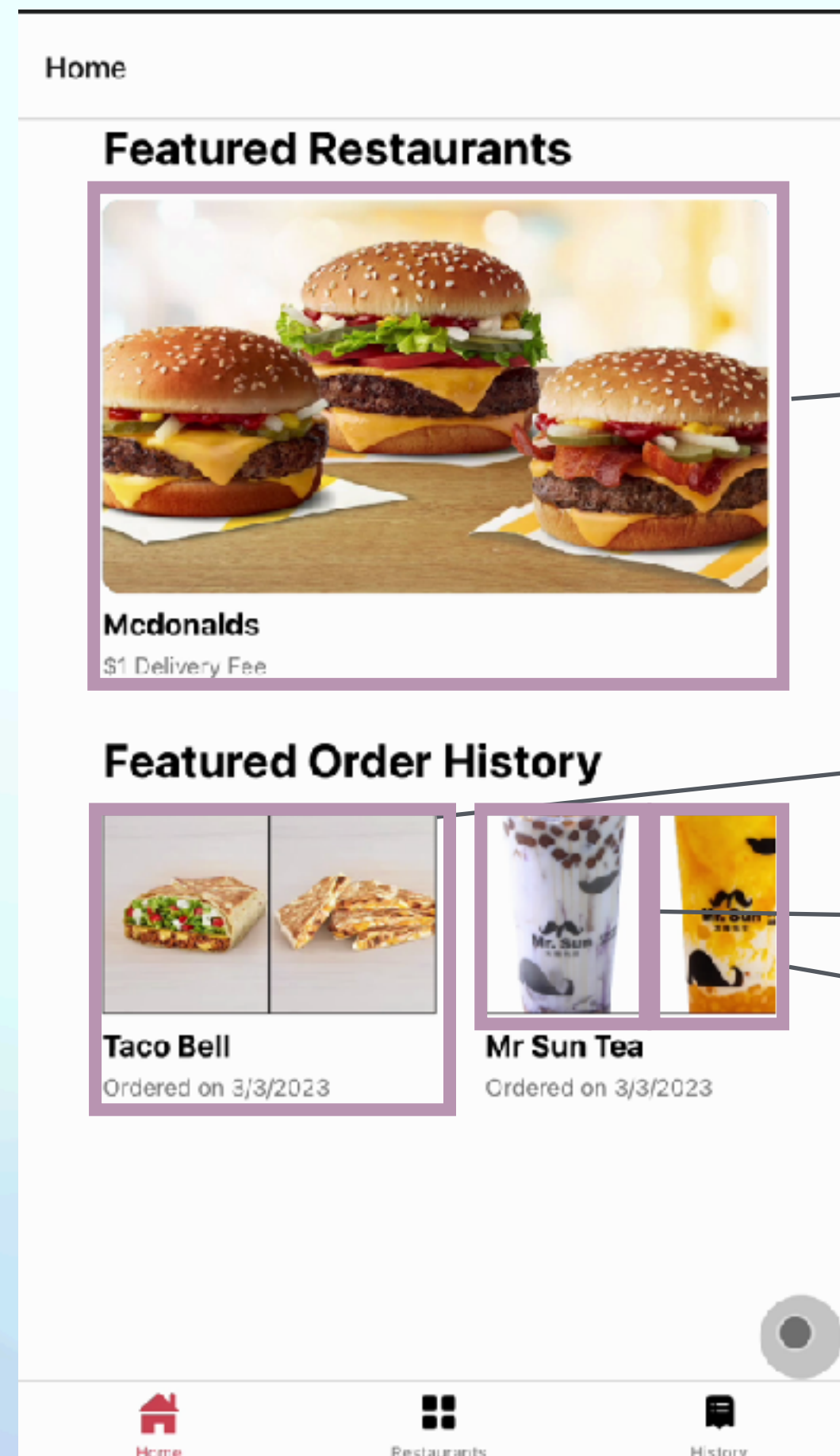
<pre>// Here are all the functions that we have class Restaurant { string name; string address; string cuisine; float rating; // All active restaurants static Restaurant[] all(); // The current restaurants static Restaurant current(); // Get a list of foods representing the menu from a restaurant Food[] 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")

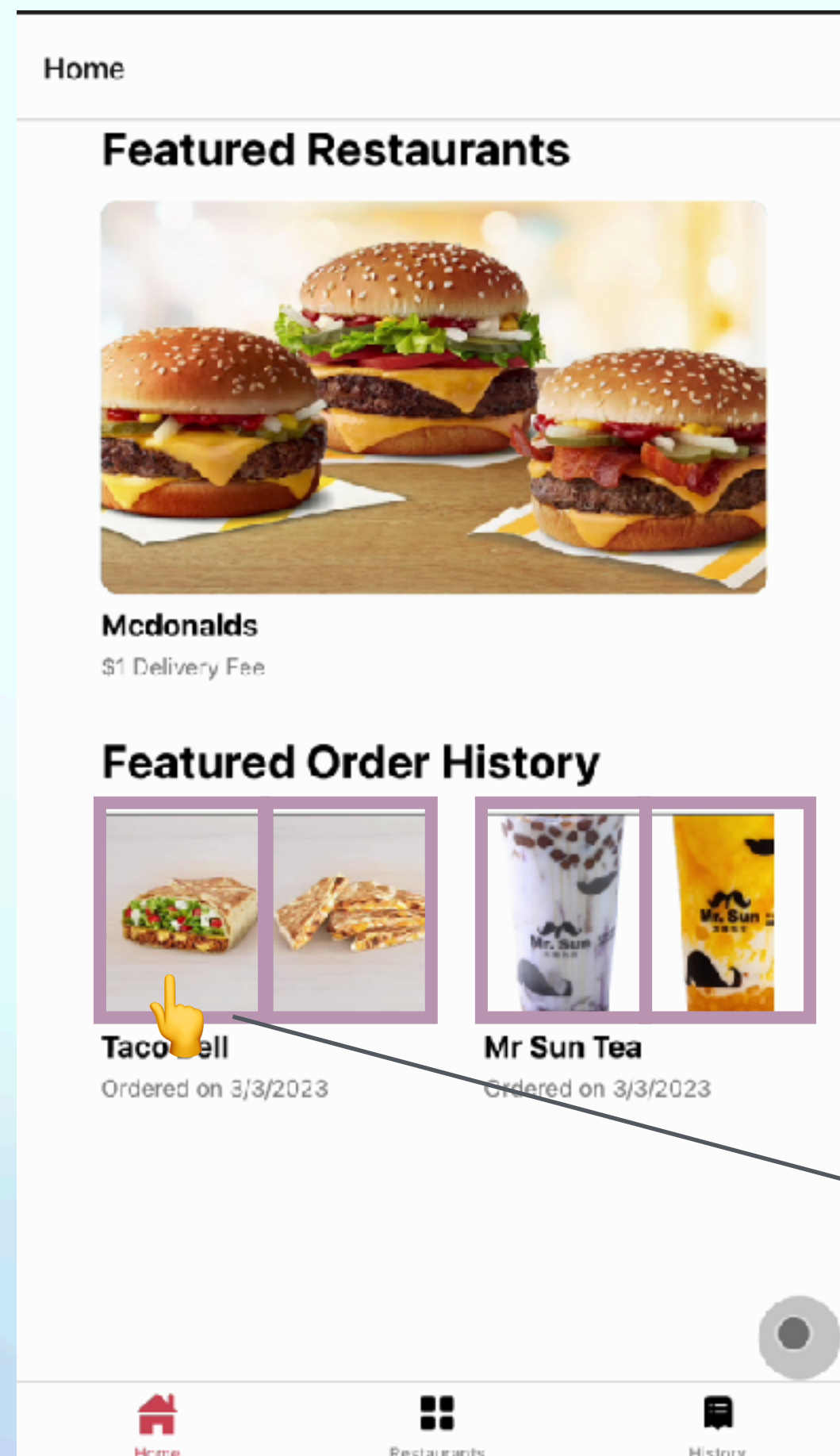
FoodItem
(name: "mango drink")

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



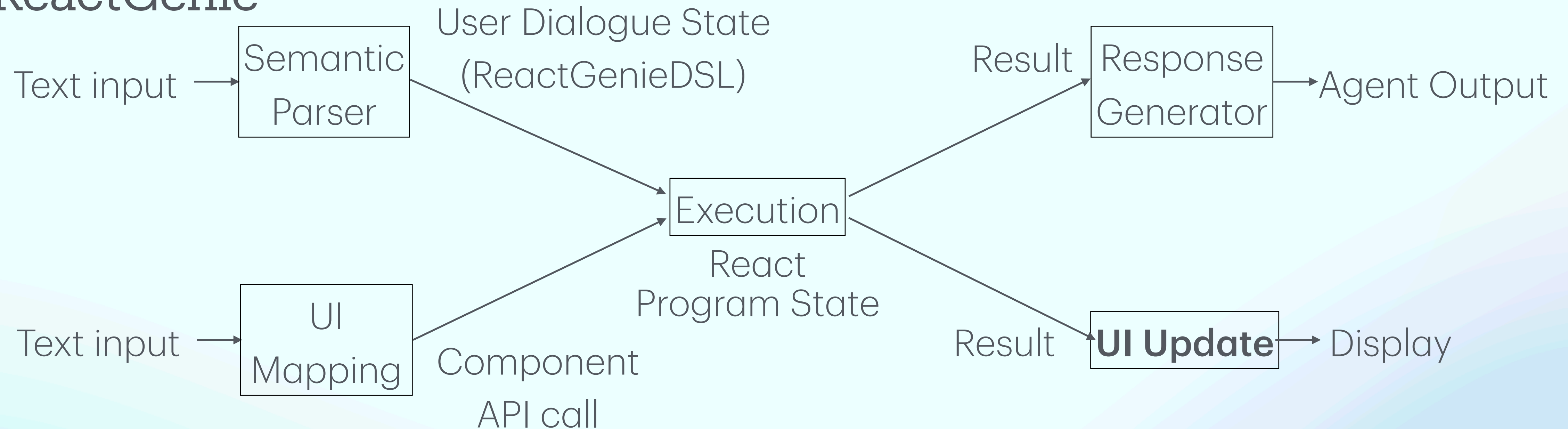
```
FoodItem(name: "CrunchWrap")
```



Continue execution

How to Navigate with Voice Commands

ReactGenie



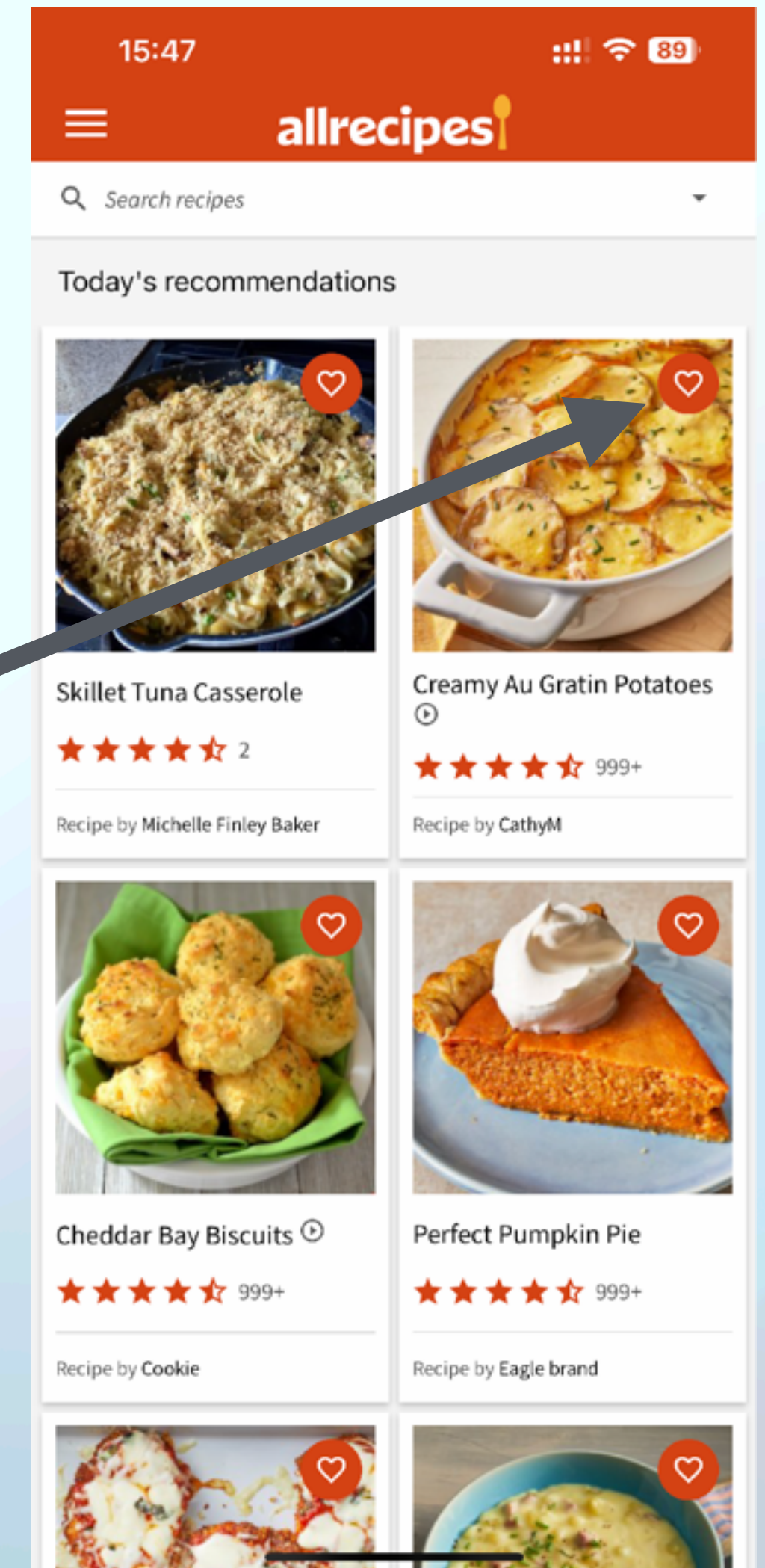
UI Update 1: Object is on UI → React takes care of it

"I love the Creamy Potatoes recipe!"

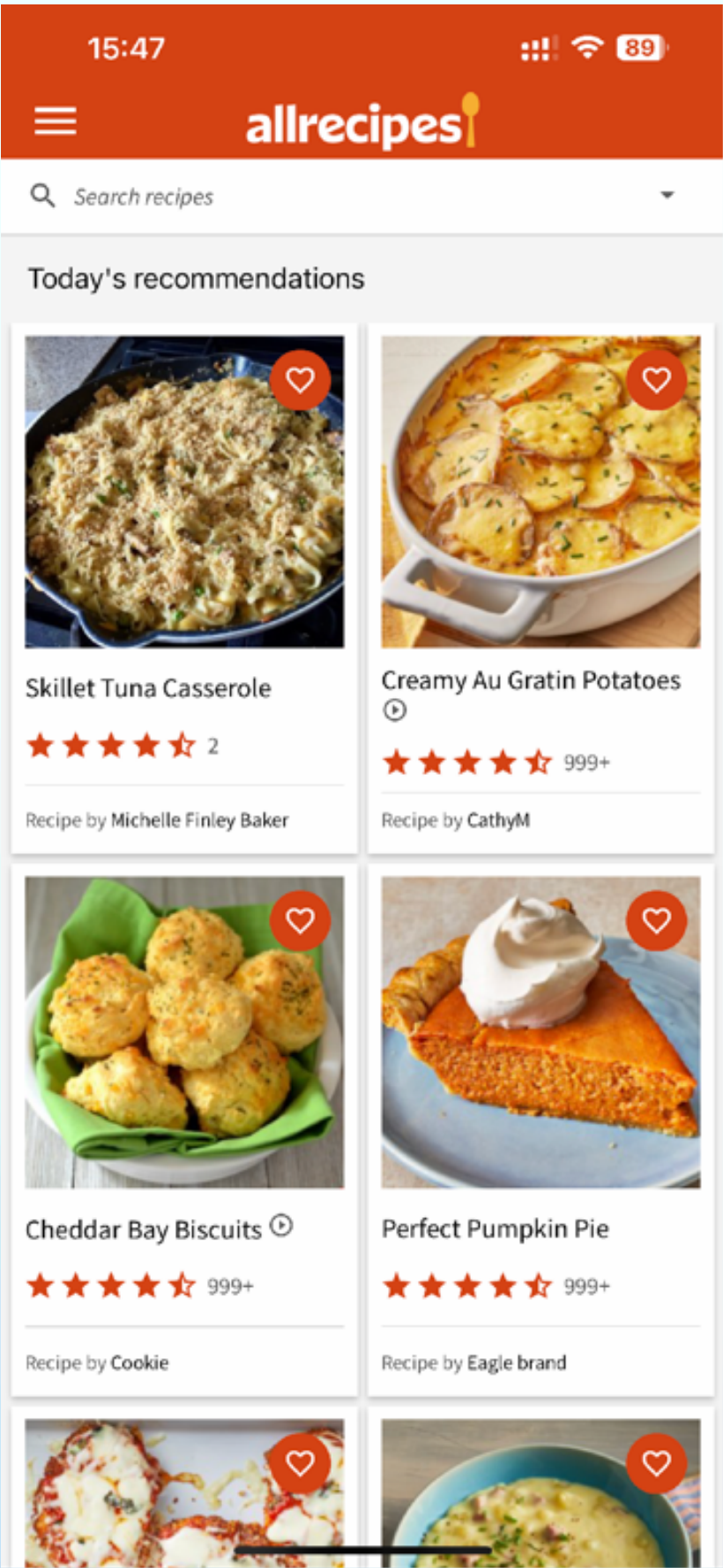
Recipe(name: "Creamy"). love()

recipe.loved = true

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

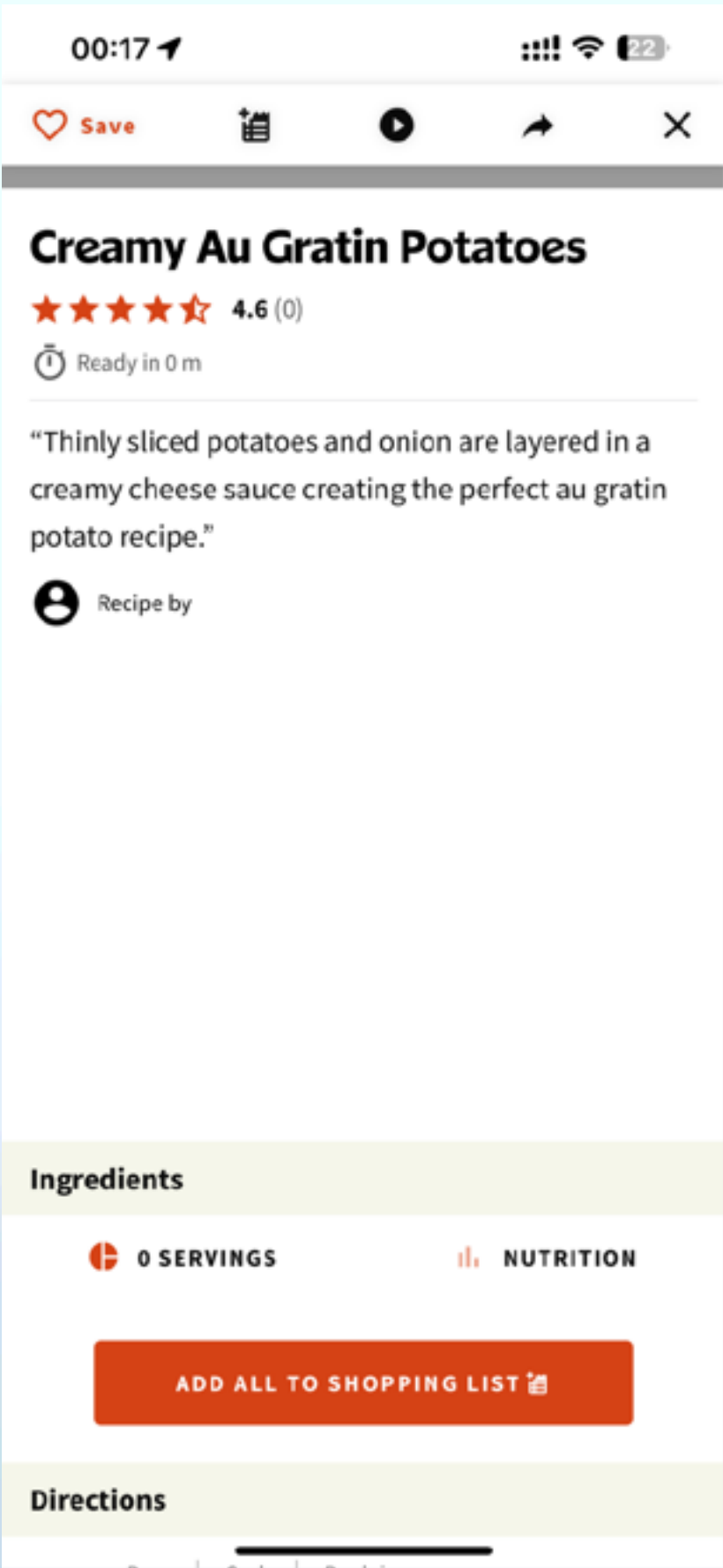


UI Update 2: Object not on UI → Navigate to the page



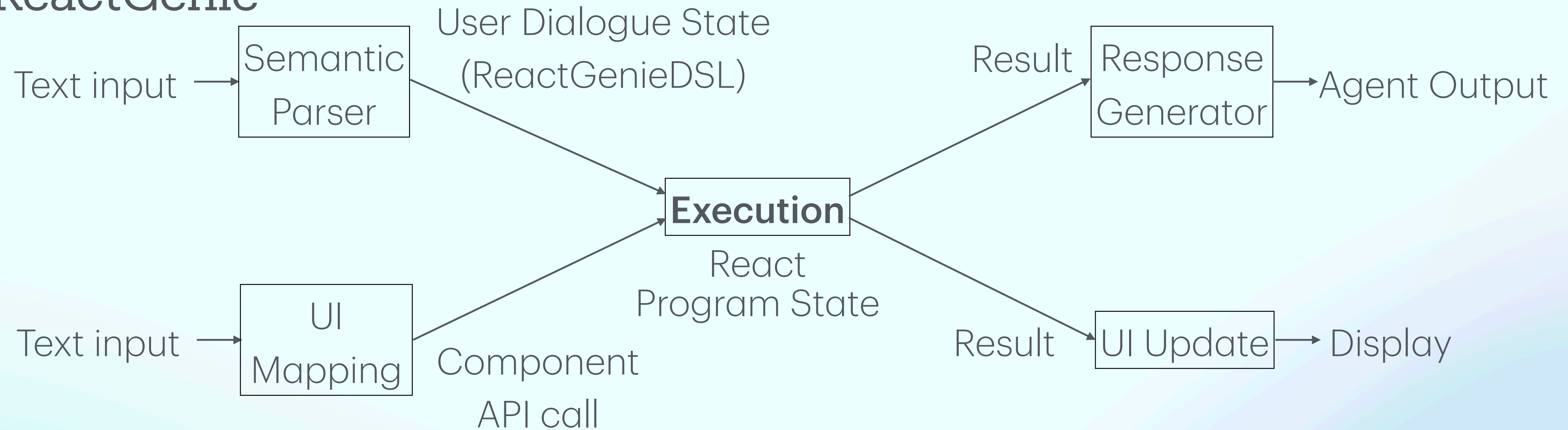
"Show me the Creamy Potato recipe!"

Recipe(name: "Creamy Potato")



Execution

ReactGenie

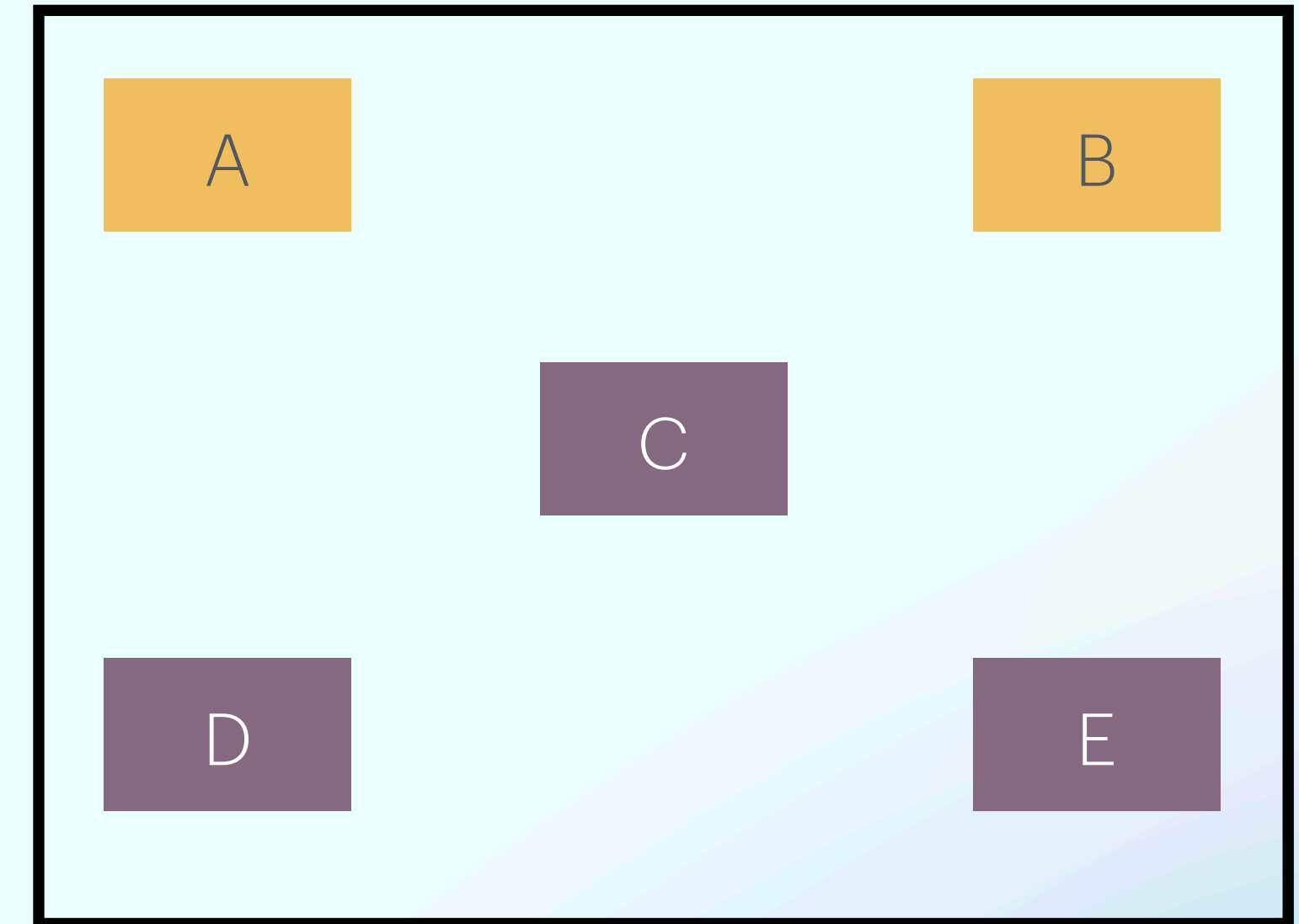


Execution

“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;
    }
}
```

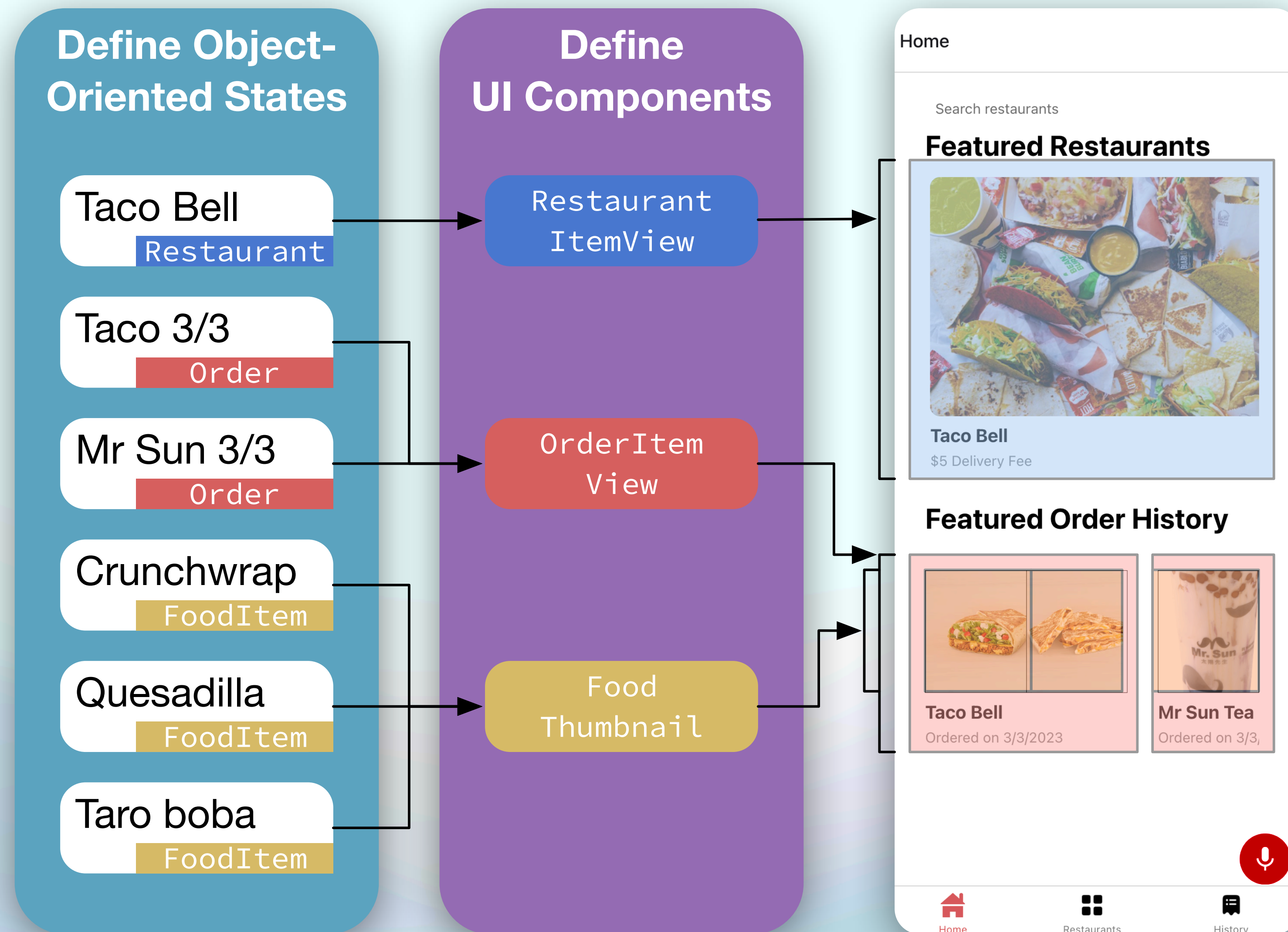
State Annotations:
Which class/property/function
can be accessed with voice

```
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

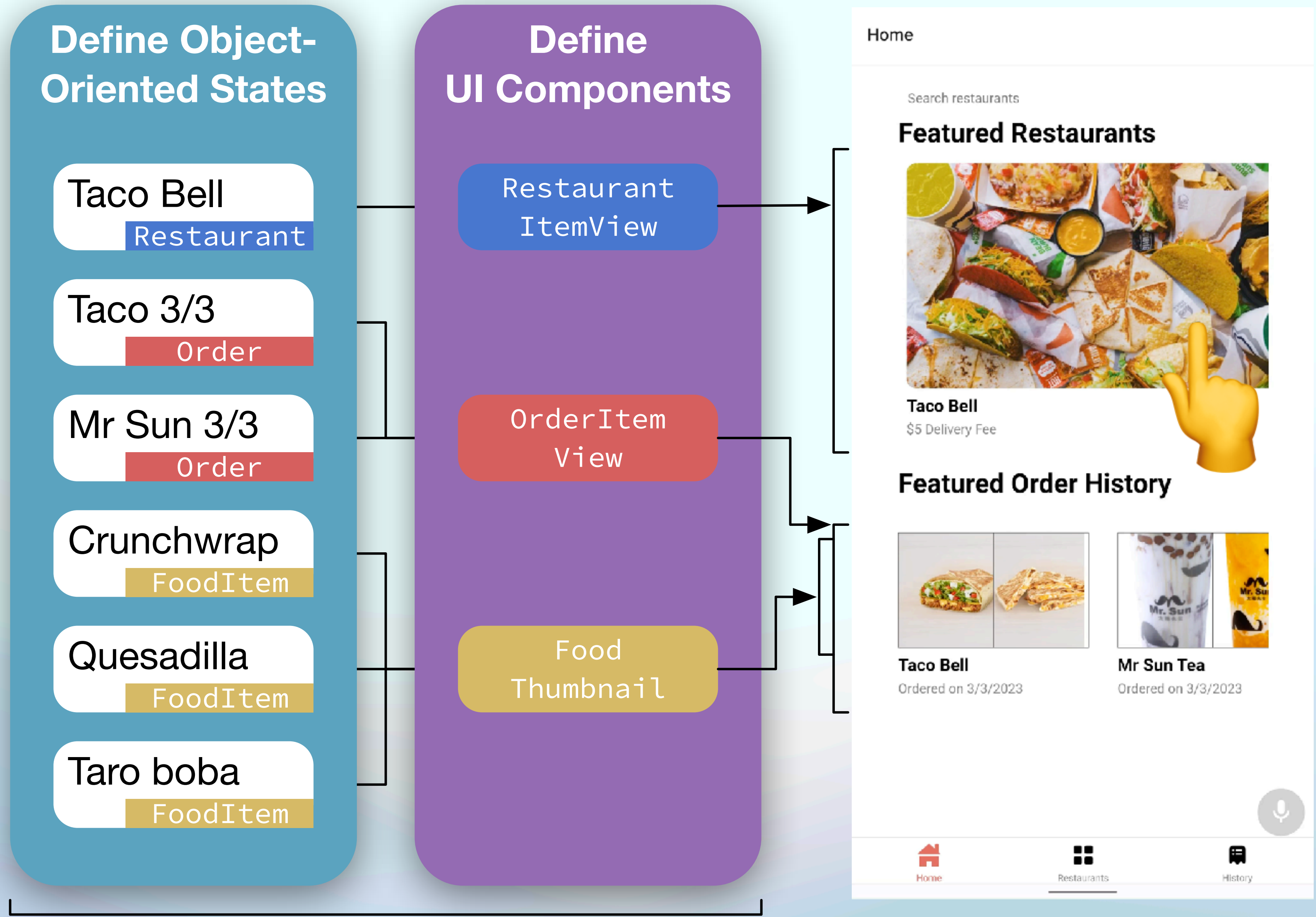
Recap: ReactGenie Uses Declarative UI Architecture for Ease of Development



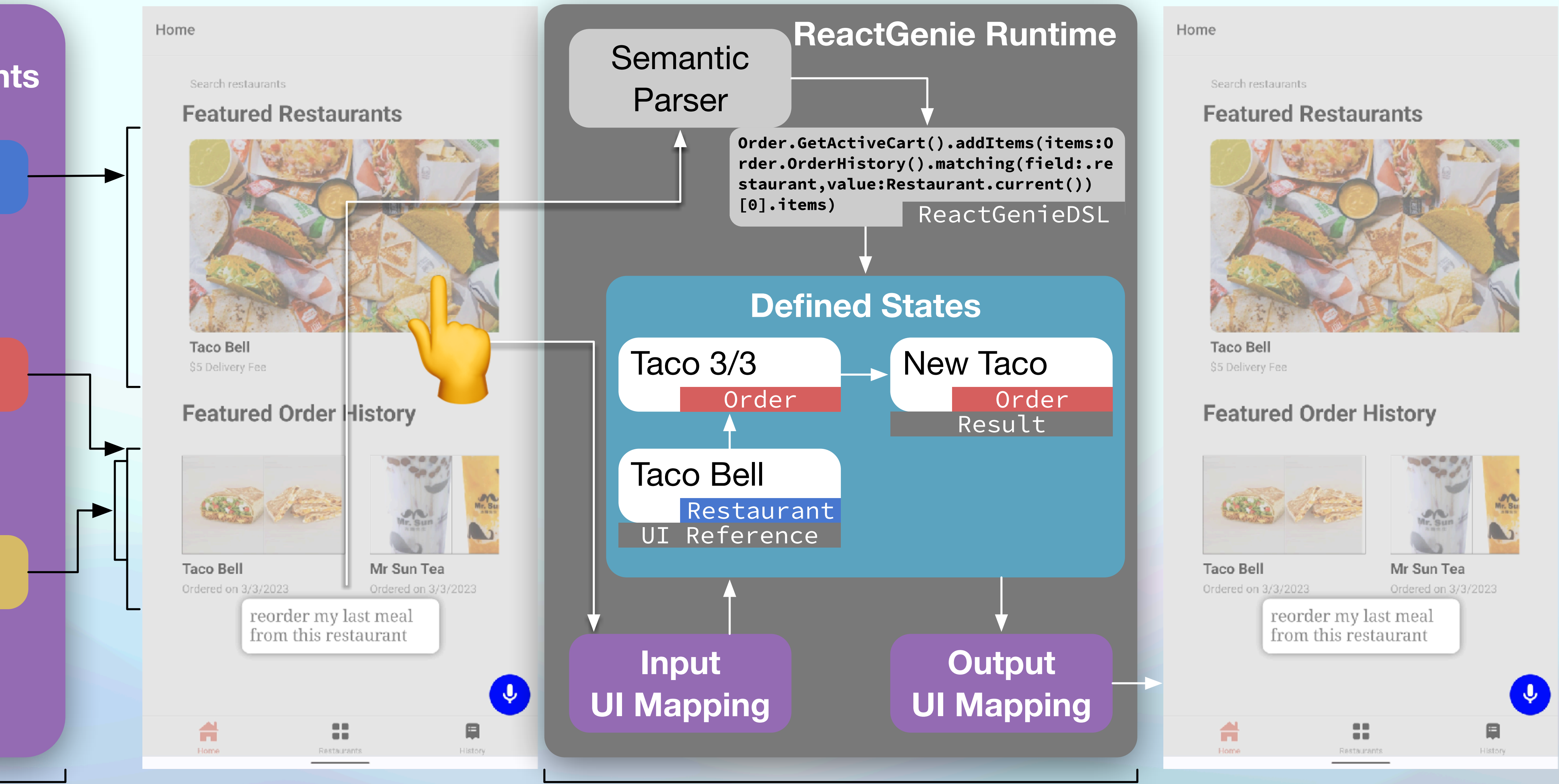
Recap: ReactGenie Use Annotations for Multimodal Integration

```
@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



Recap: ReactGenie Execute User's Request within UI Context



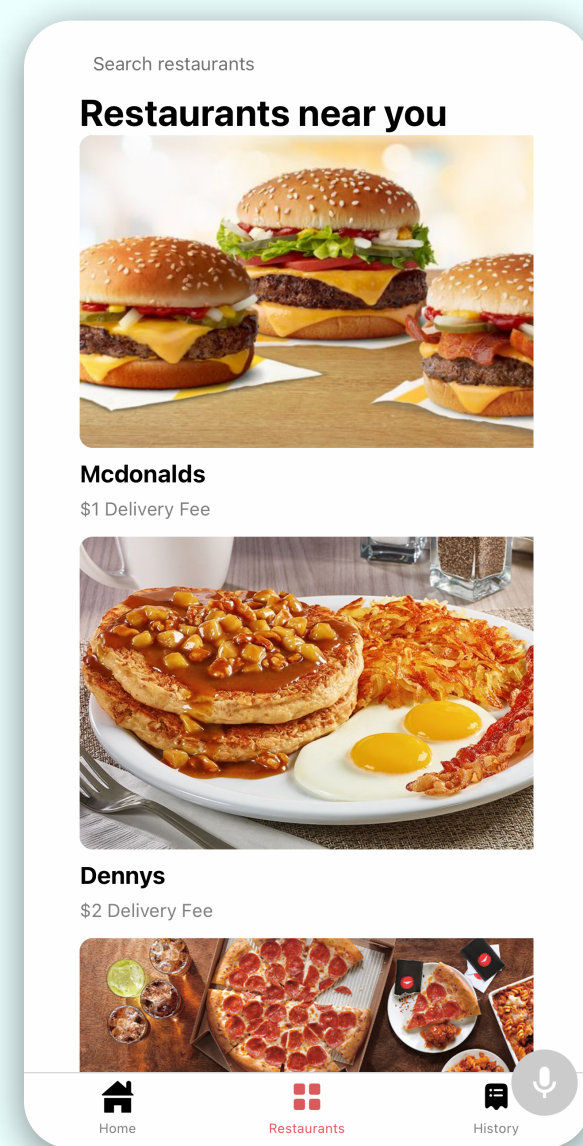
Lecture Goals

- Why do we need multimodal interactions?
- Introduction to existing multimodal interactions?
- Three problems for multimodal app development
- Introduction to ReactGenie: a multimodal app development framework
- **Evaluation of ReactGenie**

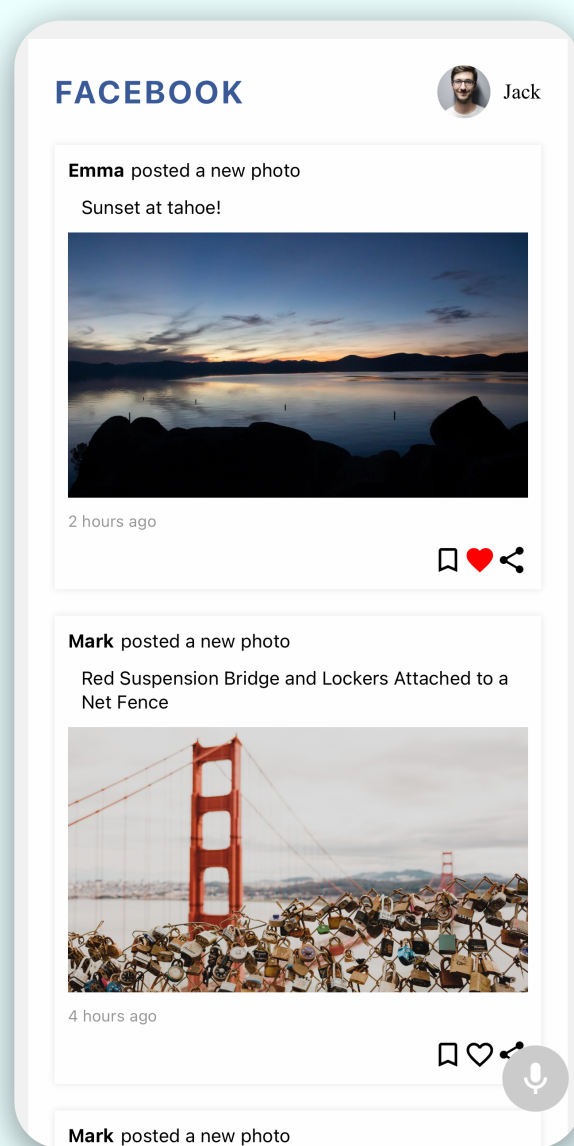
How good is ReactGenie as a framework?

- For developers:
 - D-RQ1: Assessing the expressiveness of ReactGenie
 - D-RQ2: Cost comparison for expert developers using ReactGenie vs. existing
 - D-RQ3: Ease of learning and usability for novice developers
- For users:
 - U-RQ1: Parser performance with natural language commands
 - U-RQ2: Usability and efficiency of multimodal UIs generated by ReactGenie

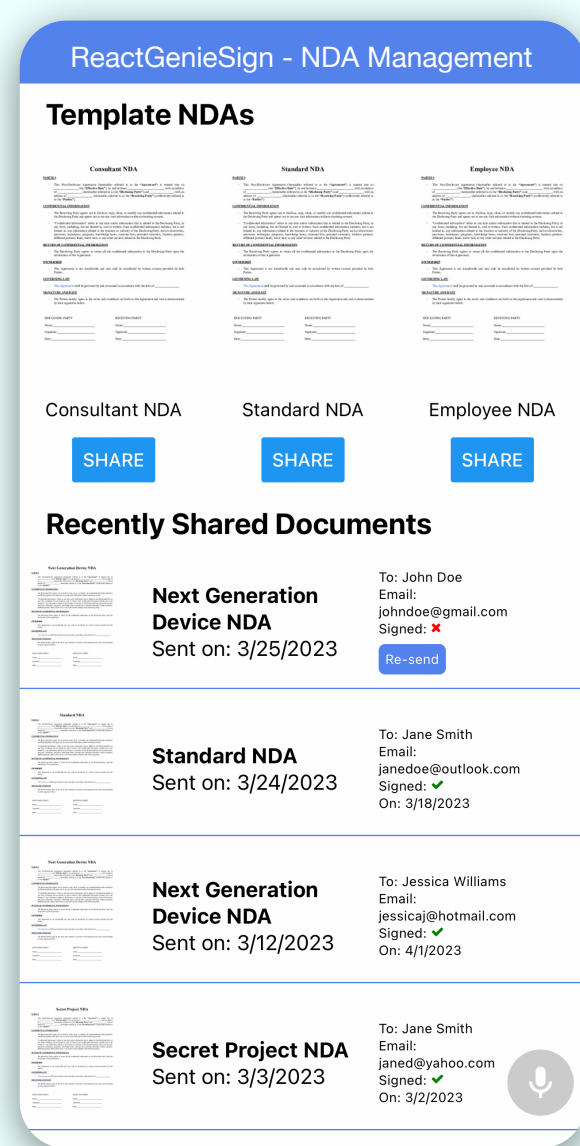
We built three apps to demonstrate expressiveness (F-RQ1)



ReactGenieFoodOrdering



ReactGenieSocial



ReactGenieSign

Add three of this to my cart

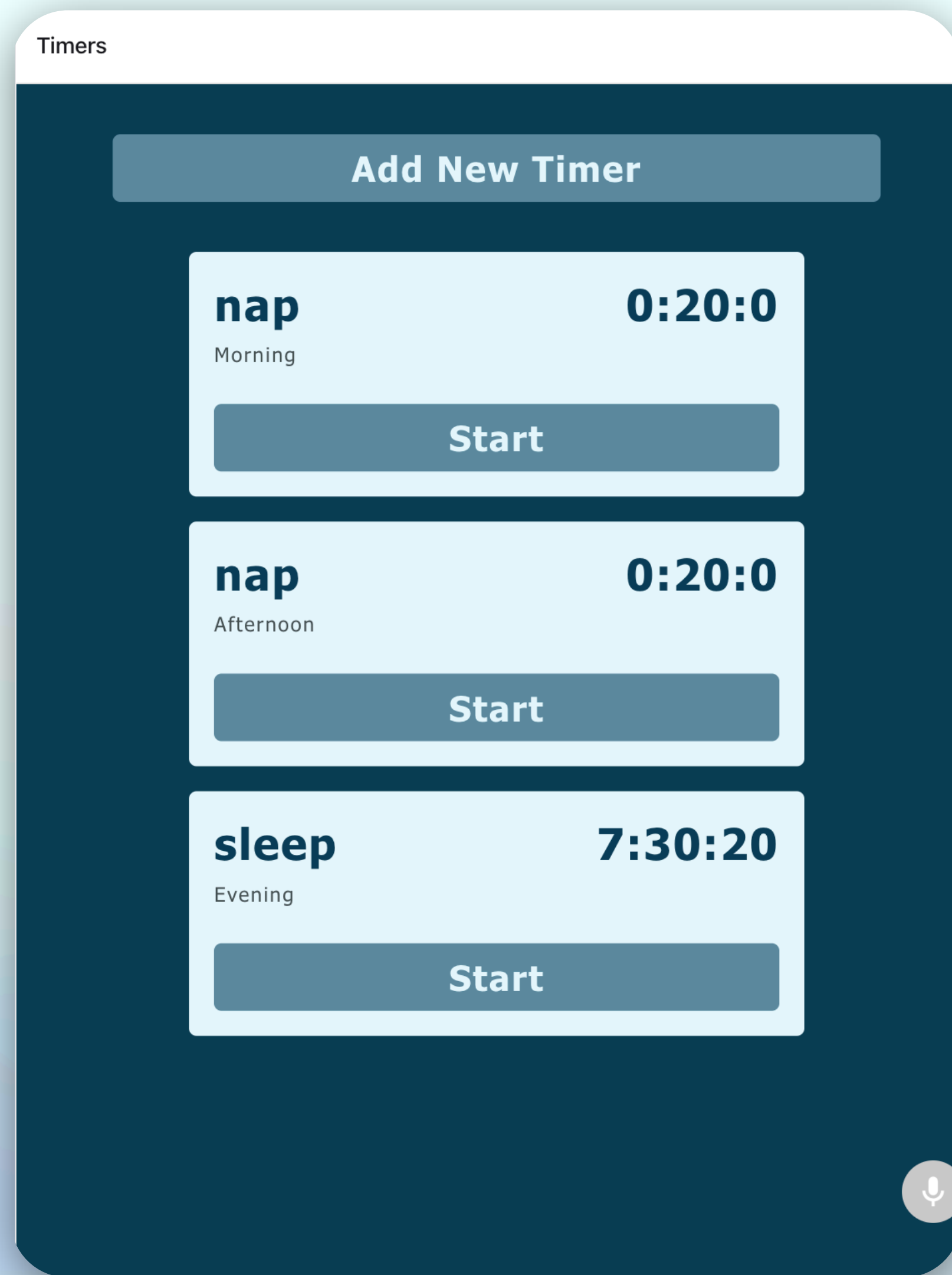
Show me posts from Mark that I have liked before.

Only show me request through this email

- Three apps and a wide range of multimodal commands demonstrate the expressiveness of ReactGenie.
- Different UI naturally decomposed to components representing different ReactGenie state classes.
- Only 5% of the code (annotations) must be written to handle multimodal interactions.

App Name	FoodOrdering	Social	Sign
Classes	Order, FoodItem, Restaurant, OrderItem	Post, User, Message	User, Document, SignatureRequest, EmailAddress
GenieComponent	11	7	6
OtherComponents	8	2	3
GenieFunction	22	8	11
GenieProperty	18	10	16
State Code (lines)	835	449	421
Component Code (lines)	1854	585	446
Examples (count)	11	6	6

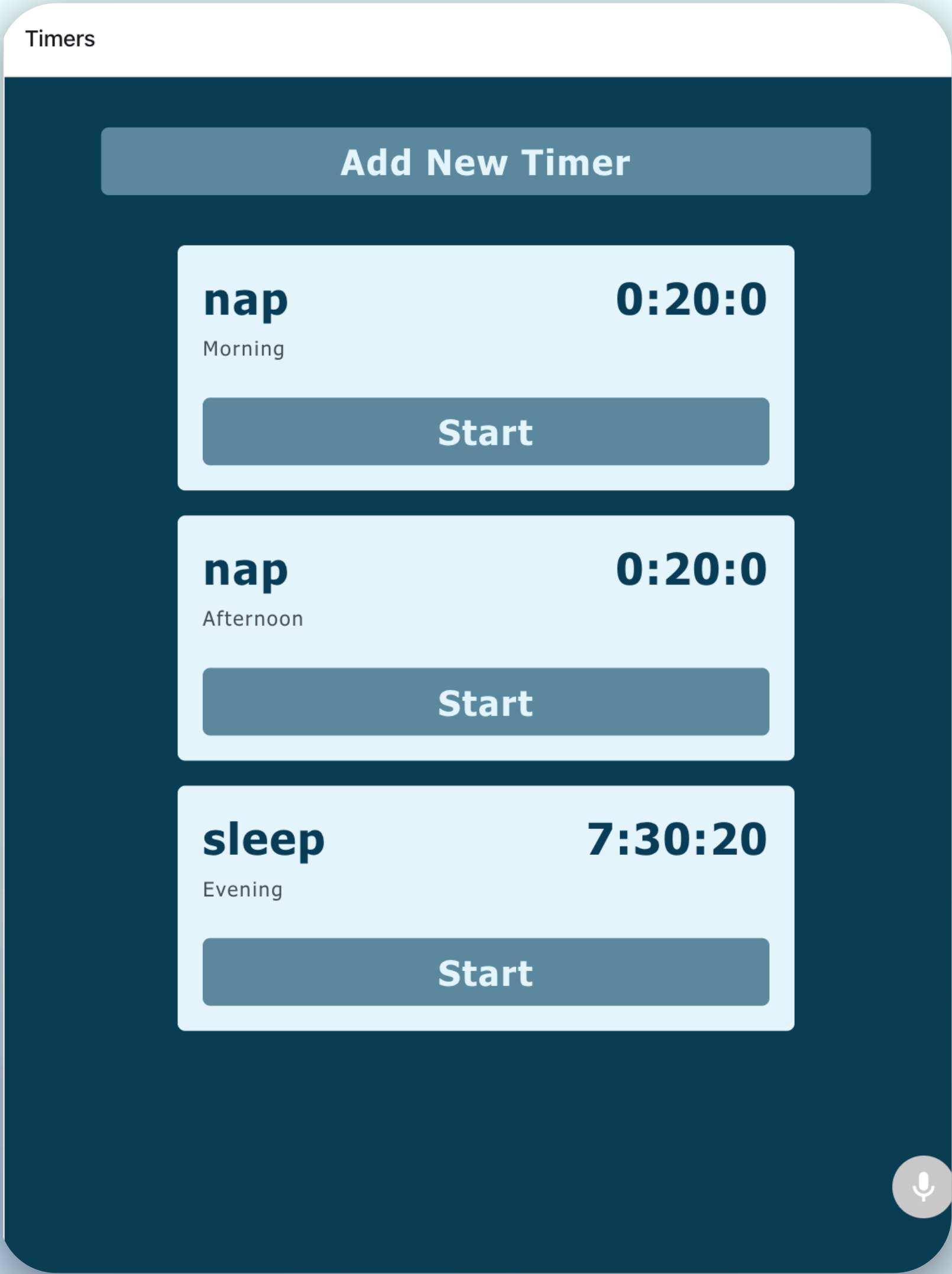
A small ReactGenie demo app that can be built within a lab study: ReactGenie Timer



ReactGenieTimer

- Classes: Timer, TimeDelta
- Functions: CreateTimer, StartTimer, PauseTimer, ClearTimer, ...
- 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 FC (F-RQ2)

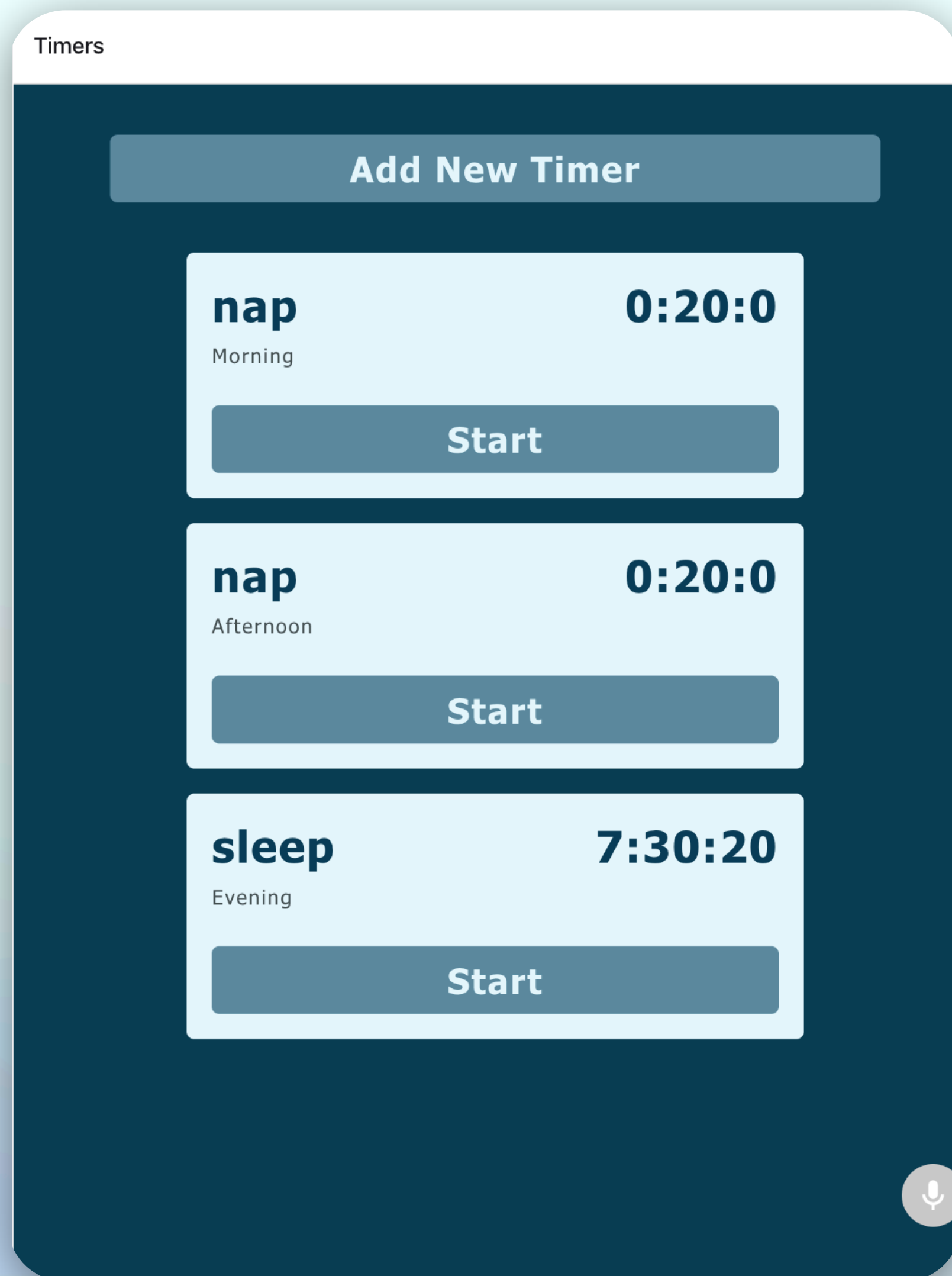


ReactGenieTimer

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

- ReactGenie demonstrates significant advantages in terms of time efficiency and code complexity.
- ReactGenie offers comprehensive multimodal support, enhancing user convenience.
- GPT Function Calling requires extensive code and lacks essential features like UI mapping.
- ReactGenie is a powerful tool for multimodal app development, streamlining the process and improving user experiences.

Novice Developer Study on ReactGenie Usability (F-RQ3)



ReactGenieTimer

- Conducted an IRB-approved user study with novice developers
- Developers tasked with building multimodal applications using ReactGenie
- Study Design:
 - Two-part process: Learning phase with tutorial and independent app construction
 - Provided boilerplate code for focus on multimodal features
- High comprehension of framework functionality
- Every developer finished the app in just a bit more than one hour!
- Positive feedback on ease of use and willingness to use ReactGenie in real-life applications

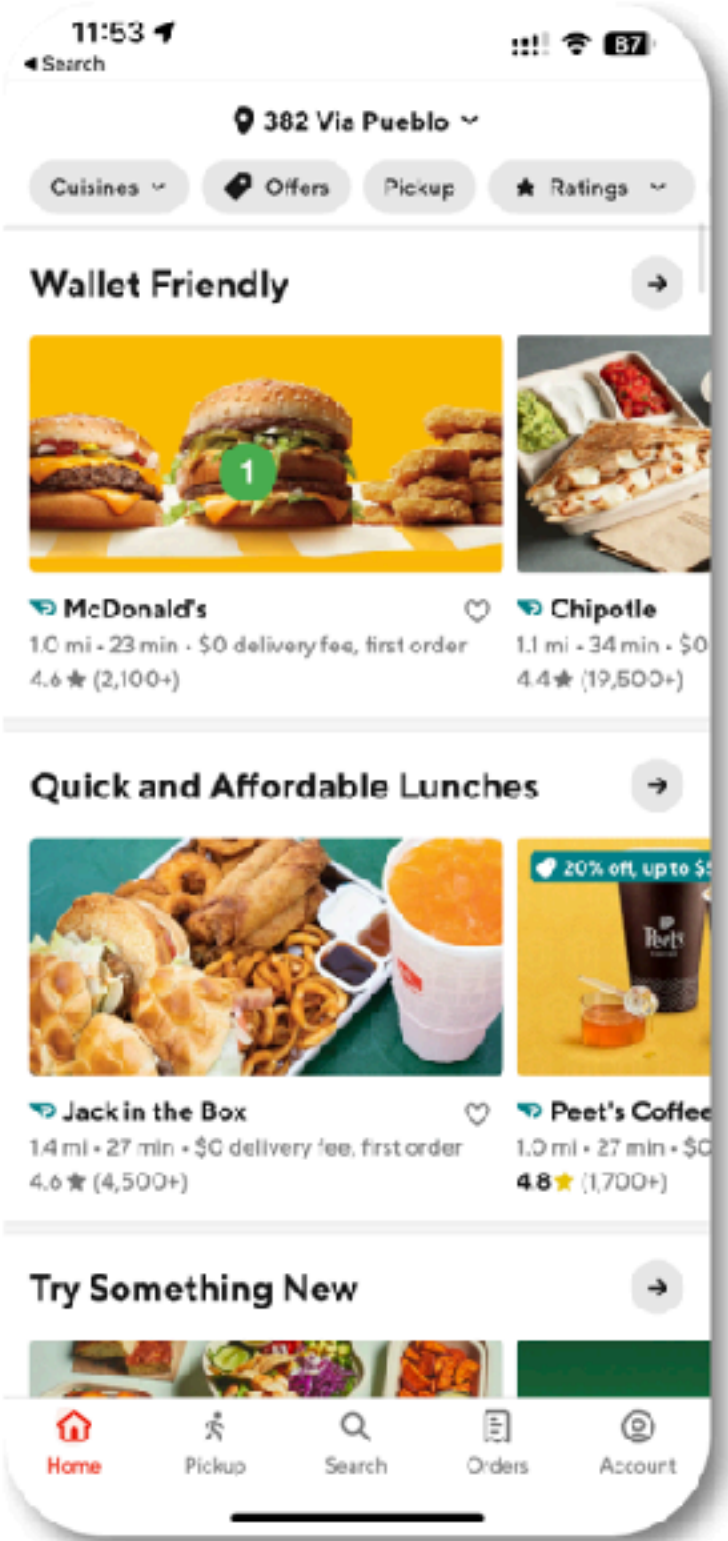
Metric	Result
Completion Time	Average: 109.67 minutes (std = 24.19 minutes)
Learning Phase	Average: 42.33 minutes (std = 14.90 minutes)
Independent Building Phase	Average: 67.33 minutes (std = 17.52 minutes)
Comprehension Quiz	11/12 participants scored 7/7 , 1 participant scored 6/7
SUS Usability Score	Average: 6.08/7 (Ease of Use)
NASA-TLX Cognitive Load	Average: 21.99/100 (Lower indicates better)
Willingness to Use	All developers indicated affirmation to ReactGenie; 6/12 participants asked us about using ReactGenie in real-life applications

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)

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



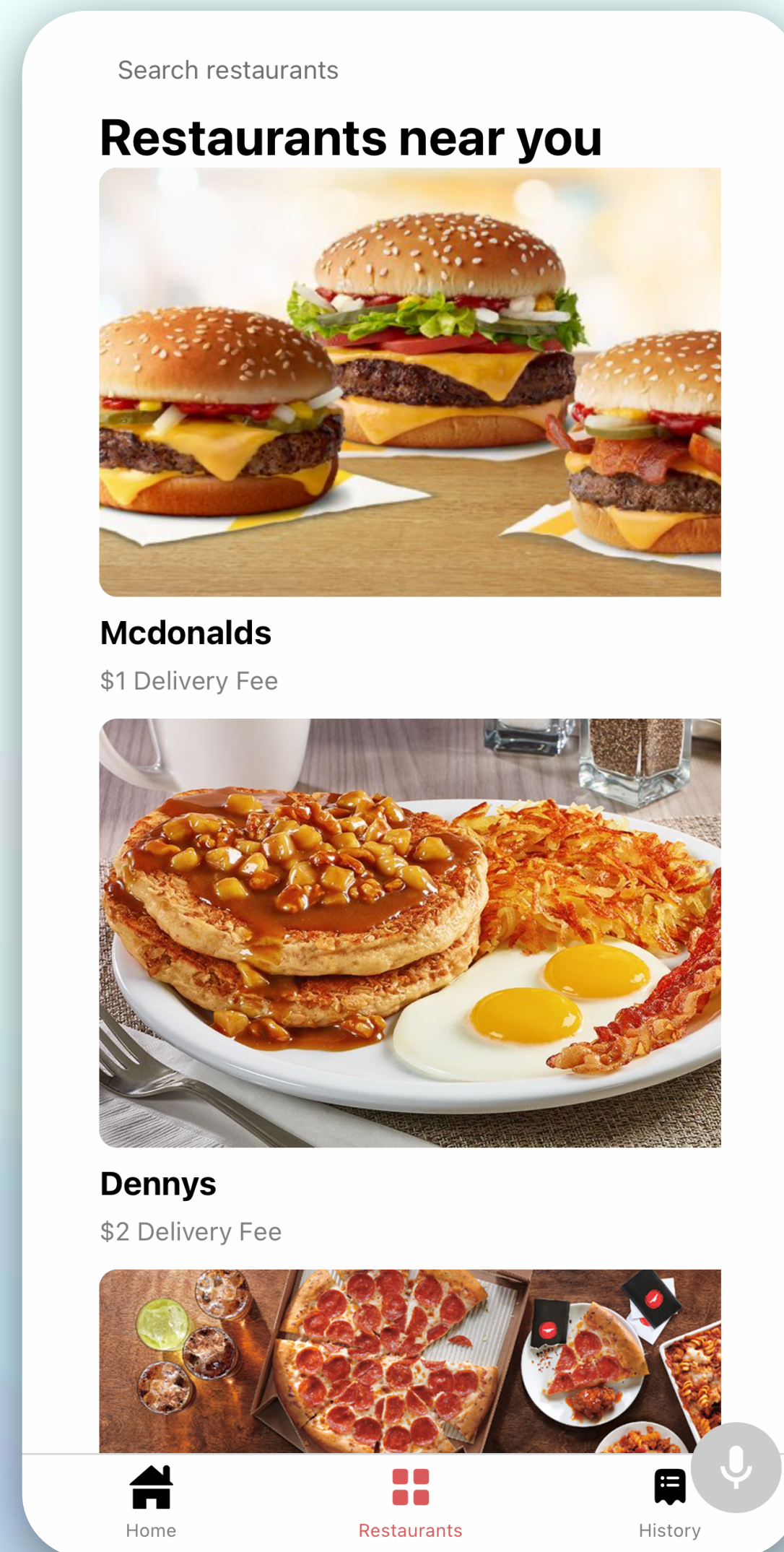
Clear Submission

- Used Cloudlicit method for real-world scenario commands
 - Participants provided with app screenshots and videos to prompt commands
- Results:
 - 288 commands analyzed after filtering unclear responses
 - Categories of Commands:
 - Simple UI interaction: 100 commands
 - Targeted interactions (complex tasks): 172 commands
 - Out of scope for ReactGenie: 16 commands
 - Parser Accuracy:
 - 101 supported commands: parsed correctly **91%**
 - 71 unsupported commands: generated sensible ReactGenieDSL **53%**

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

Compare user performance and experience using multimodal UIs vs. GUI-only

- Study Design:
 - Within-subject design with 16 participants
 - Tasks performed on both multimodal UI and GUI-only versions of ReactGenie Food Ordering app
- Measured task completion time, cognitive load (NASA-TLX), and usability (SUS)
 - Cognitive Load: Significantly lower with multimodal UI (NASA-TLX score: 24.6 vs. 34.5)
 - Usability: Higher with multimodal UI (SUS score: 73.0 vs. 63.3)
- Participant Preferences:
 - 11/16 preferred multimodal UI
 - Reasons for preference: Ease of use, efficiency, and clarity in task execution
 - Reason for alternative choice: Poor voice recognition



ReactGenieFoodOrdering

Recap

- Multimodal interaction history is long, but adoption is limited due to implementation costs.
- Multimodal interactions entail human-computer communication via multiple input/output modes.
- 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.

I hope that you will...

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