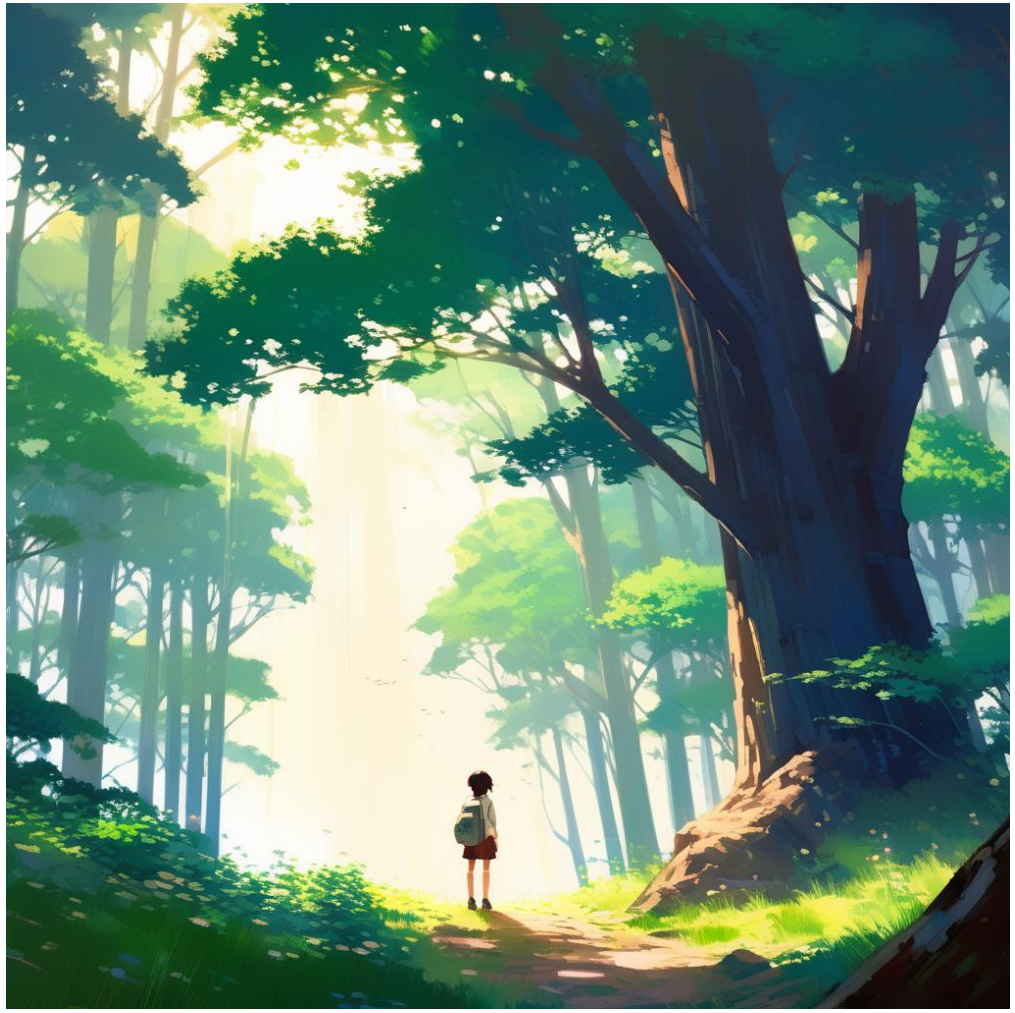


Canopy

Display a View Hierarchy from a description

R&D AI Challenge - Antony Costa

Agenda

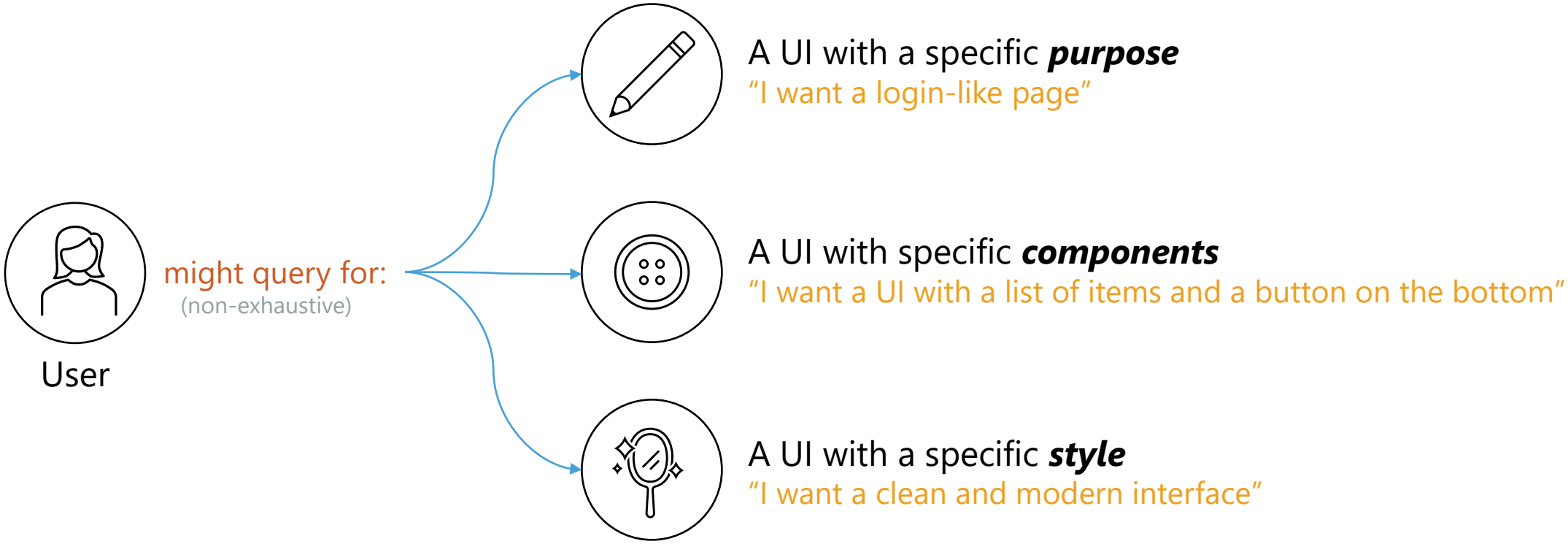
- The Problem
- The Data
- The Solution
- Display a UI from a catalogue of pre-existing UIs from a description
    - Technical Approach, Results & future work
- Display newly generated UIs that match a Description
    - Technical Approach & Results & future work

The Problem

Generate a View Hierarchy from a description

What is a Description?

A prompt a user using the platform would write to generate a view hierarchy structure.

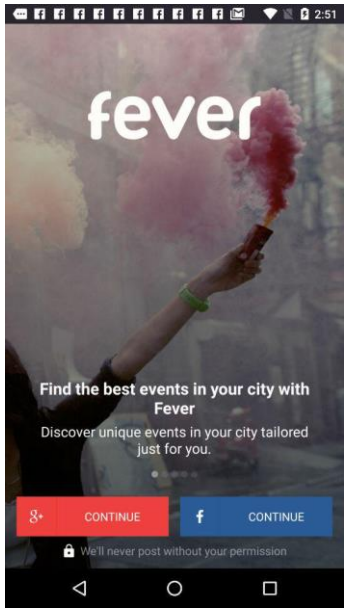


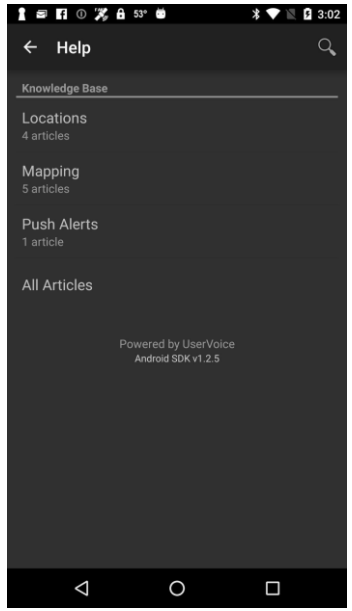
The Problem

Generate a View Hierarchy from a description

What is a View Hierarchy?

A graphical representation of the parent-child relationships among UI components.





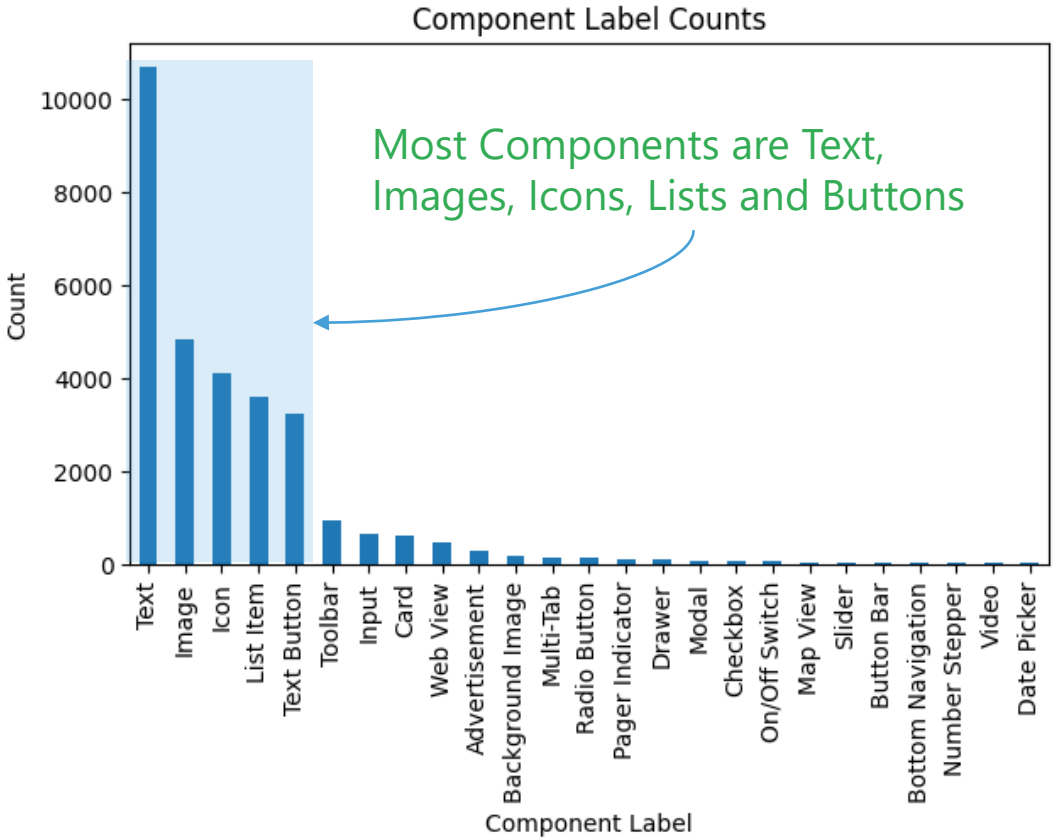

```
1 'children': [{'children': [{'text': 'Help', 'componentLabel': 'Text'},
    {'iconClass': 'arrow_backward', 'componentLabel': 'Icon'},
    {'iconClass': 'search', 'componentLabel': 'Icon'}],
   'componentLabel': 'Toolbar'},
  {'children': 2 'text': 'Knowledge Base', 'componentLabel': 'Text'}],
   'componentLabel': 'List Item'},
  {'children': [{'text': 'Locations', 'componentLabel': 'Text'},
    {'text': '4 articles', 'componentLabel': 'Text'}],
   'componentLabel': 'List Item'},
  {'children': [{'text': 'Mapping', 'componentLabel': 'Text'},
    {'text': '5 articles', 'componentLabel': 'Text'}],
   'componentLabel': 'List Item'},
  {'children': [{'text': 'Push Alerts 3 'componentLabel': 'Text'},
    {'text': '1 article', 'componentLabel': 'Text'}],
   'componentLabel': 'List Item'},
  {'children': [{'text': 'All Articles', 'componentLabel': 'Text'}],
   'componentLabel': 'List Item'},
  {'children': [{'text': 'Powered by UserVoice', 'componentLabel': 'Text'},
    {'text': 'Android SDK v1.2.5', 'componentLabel': 'Text'}],
   'componentLabel': 'List Item'}]}
```

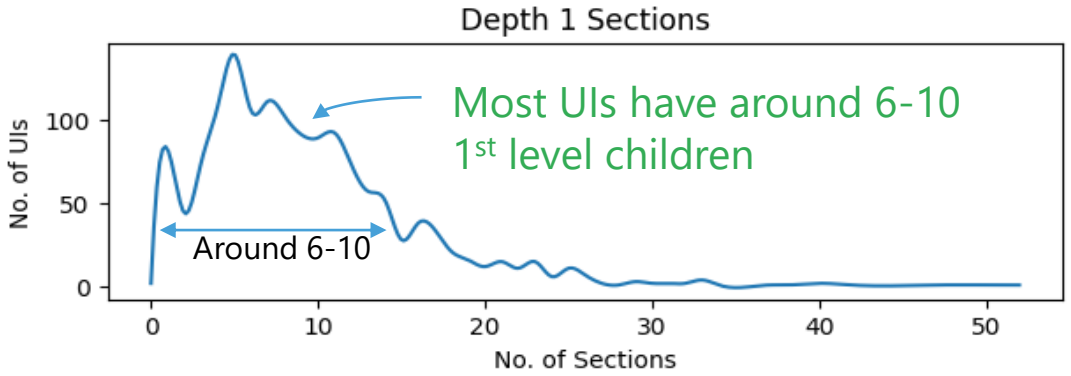
Tree composed by:

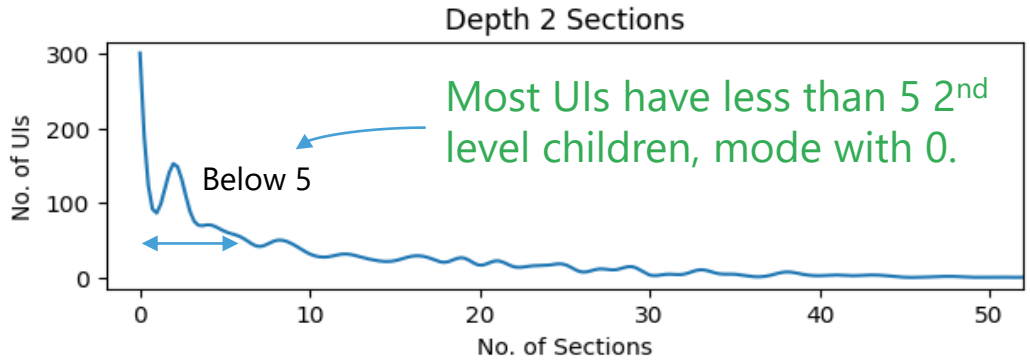
1. Nested Sections
2. Components
3. Component Attributes

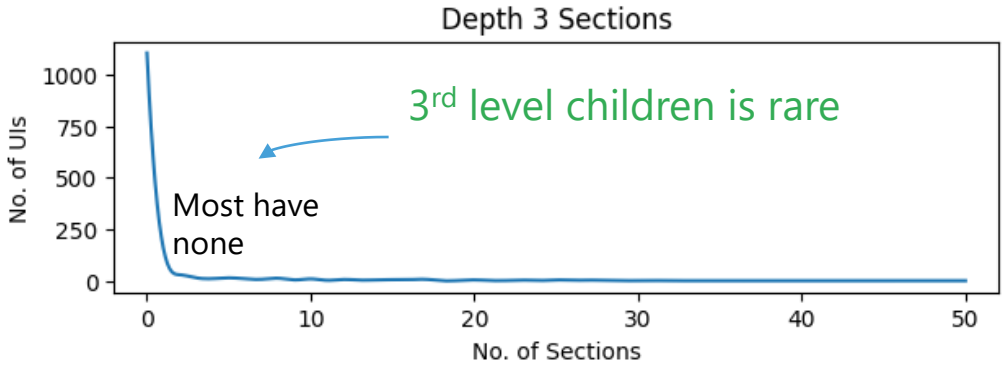
The Data

1460 View Hierarchy Files

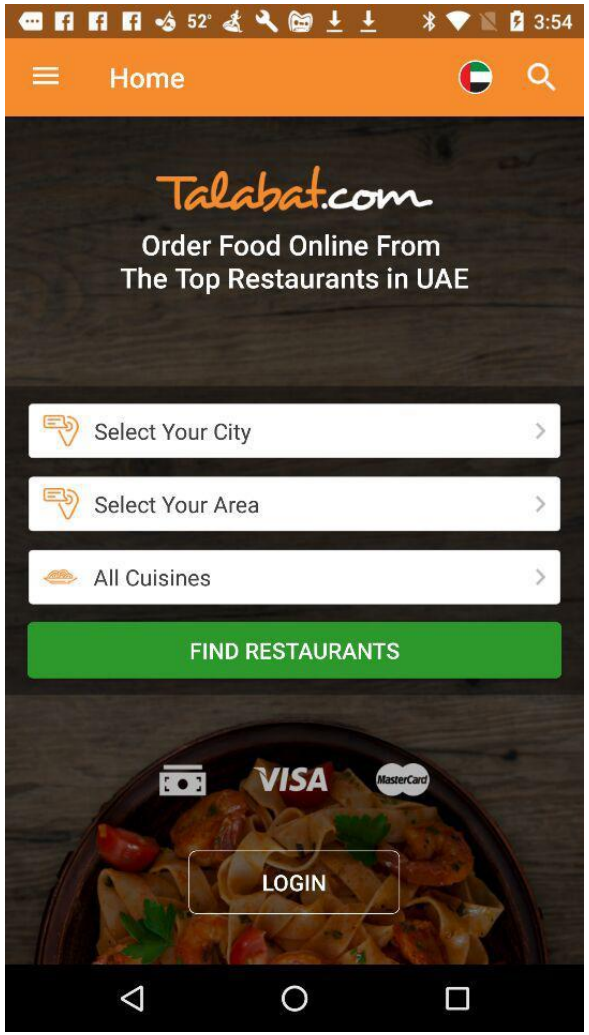




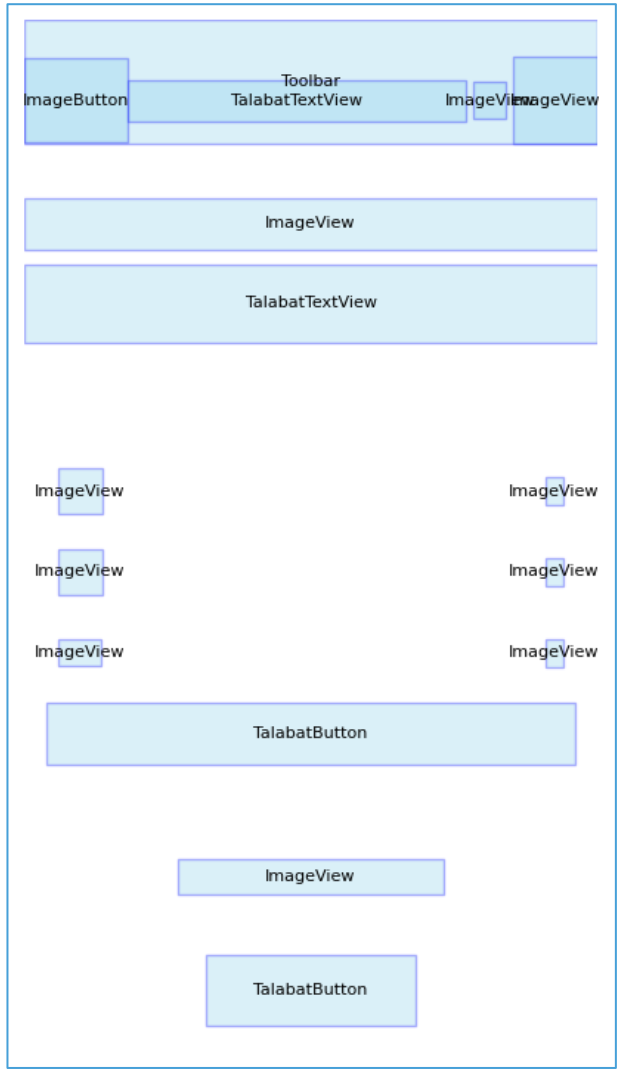




The Data



The output of the tree structure would look like this



The Data

Additional Data

Topic Database:
Useful to improve the understanding of a UI Tree Structure

	screen_id	topic
0	50245	tutorial
1	320	list
2	39729	login
3	8555	form
4	39305	list
...	...	...
1455	40928	login
1456	51913	settings

Caption Database:
Useful for performance evaluation (proxy of a description)

	image_id	caption
0	58658	The screen is a tutorial screen having a large text element located at the top part of the screen.
1	59376	A search app with a large background image ubicated at the center part.
2	23112	A gallery app with a large image element placed at the top-left part.
3	10693	This app is a gallery screen having an image situated at the top-left area of the screen.
4	13895	The app looks like a gallery app with an image component ubicated at the right area.
...	...	...
264	501	A modal app having a text button element placed at the center area.
265	60374	A tutorial app with an image component situated at the top part.

The Solution

What makes a good solution?

Alternative Solutions		Expected Output
1	Display sample UIs (from an existing database) that match a Design Topic (fixed list is provided to the user)	Simple Filter from Topic 1:Many UI
2	Display sample UIs (from an existing database) that match a Description (open text field but the user must select from a fixed list of suggestions)	Suggesting the most similar description to the text provided by the user
3	Display sample UIs (from an existing database) that match a Description (open text field)	Provide the most accurate UI based on the description (or no UI if an accuracy threshold is not met)
4	Display UIs (newly generated) that match a Description (open text field)	Provide accurate and functioning UIs based on the description (within a timeframe)

My Scope

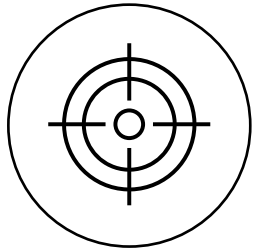
The Solution

What makes a good solution?

The solution's performance should maximize four criteria*

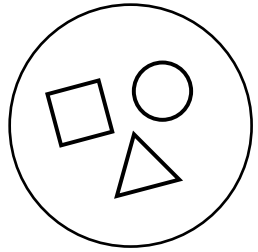
Structurally Working

The output UI should have a functioning tree structure



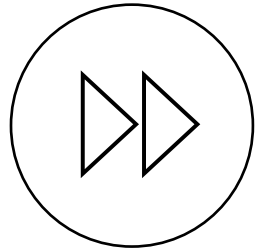
Accurate

The output should return an acceptable UI within the least #examples (and without requiring a lot of post-processing by the user)



Easy to use

Any restriction on the description input should not require a lot of effort



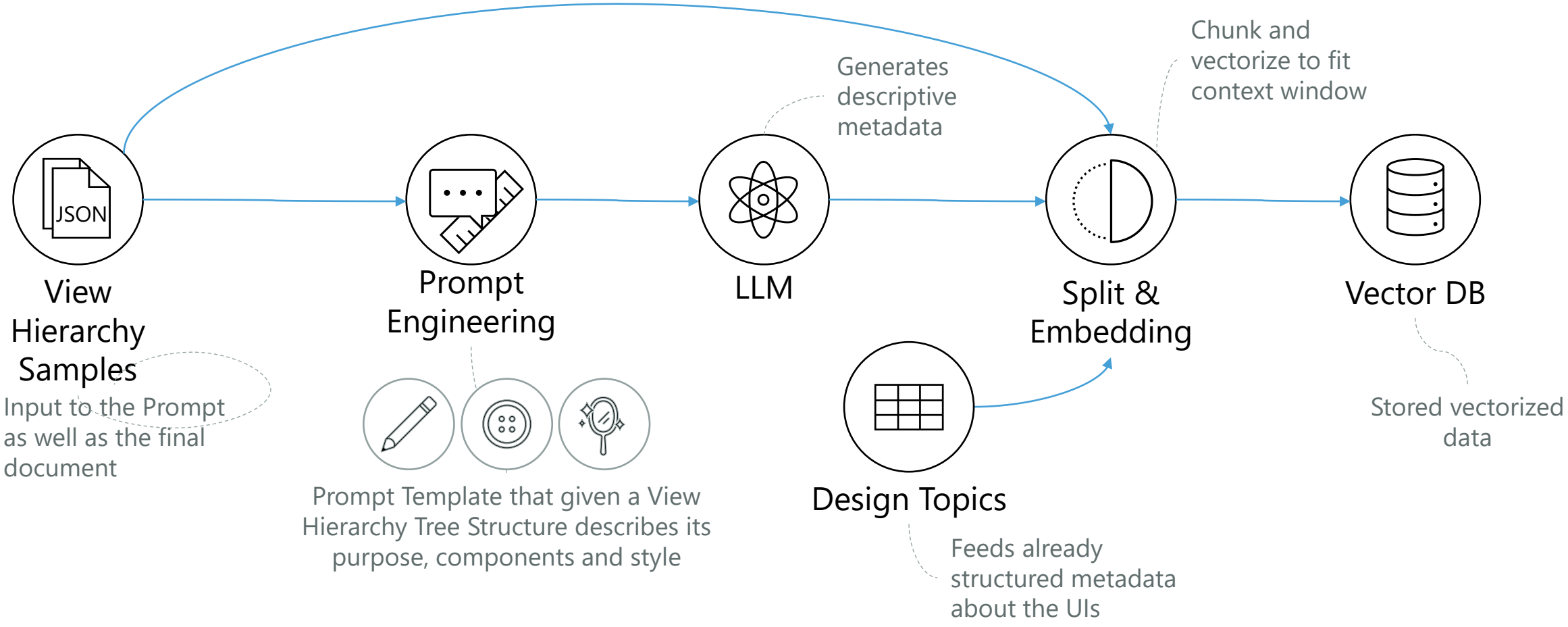
Fast

Generating examples from the input should not take too long

*Other performance criteria: Cheap? - would need to know more about the economics of Outsystems

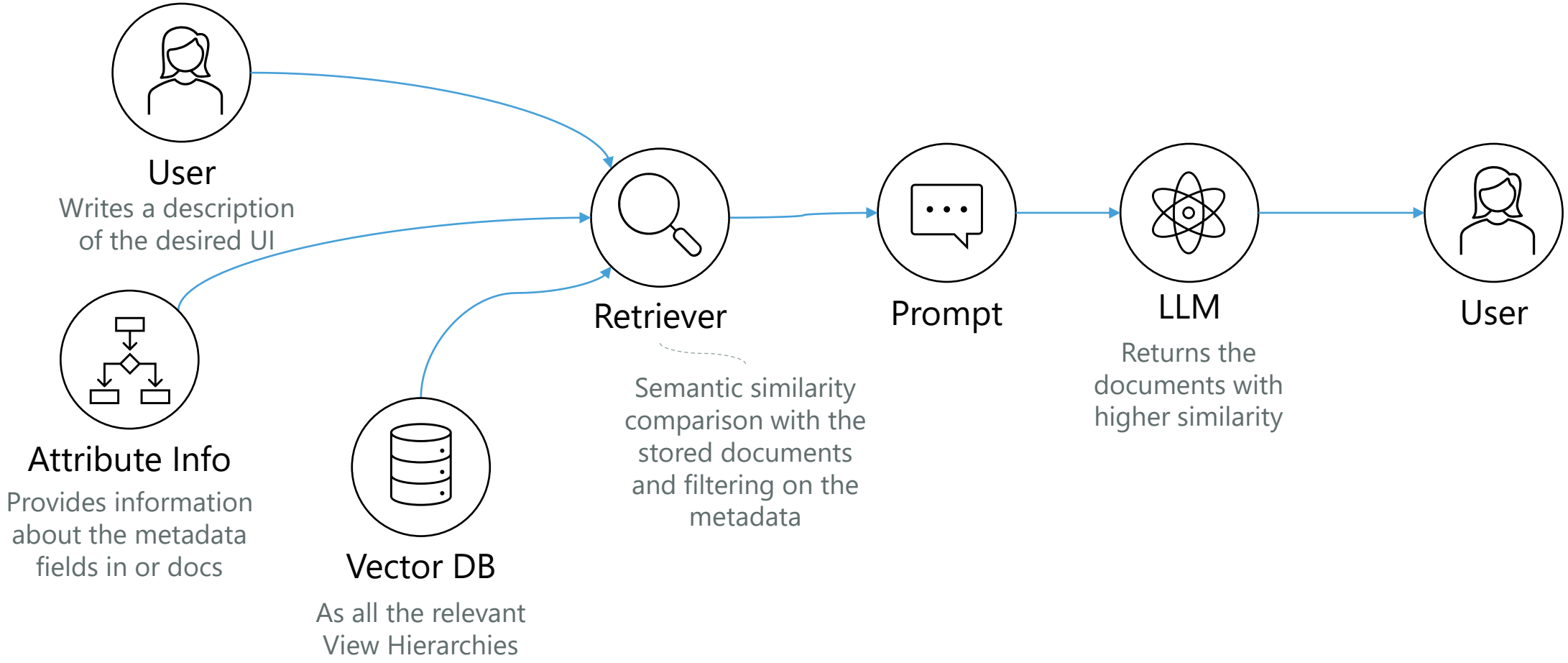
Technical Approach

Generate a High-Performing Vector Store (common to both approaches)



Technical Approach

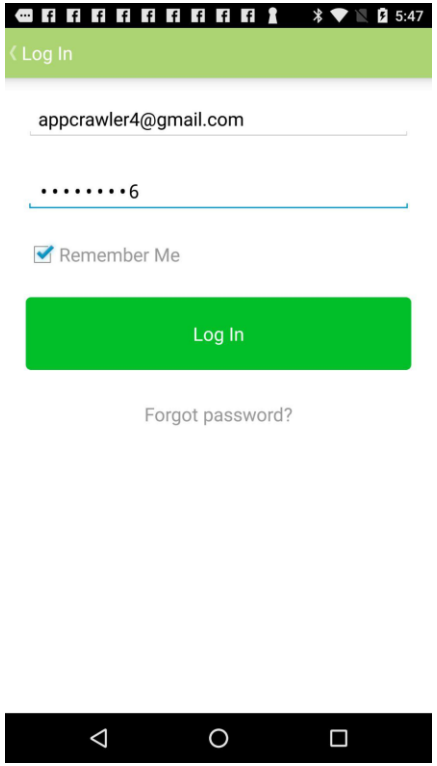
3 Display a UI from a catalogue of pre-existing UIs from a description



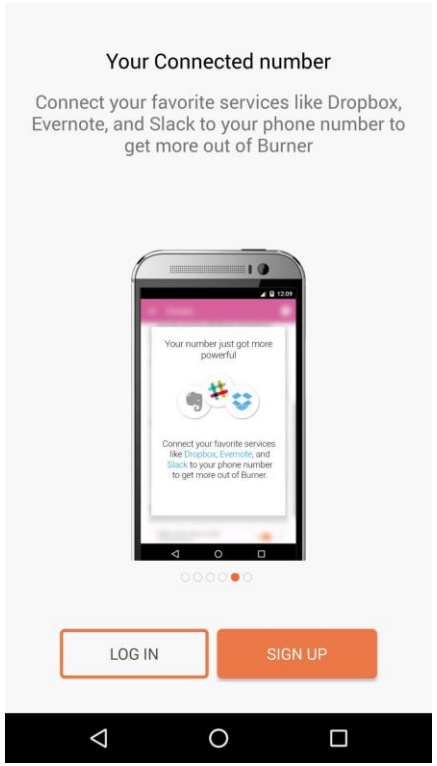
Results

3 Display a UI from a catalogue of pre-existing UIs from a description

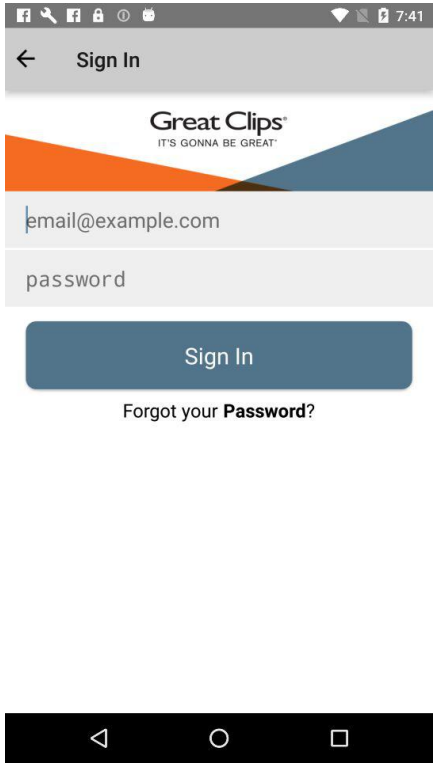
Top 5 Results of `retriever.invoke("I want a Login-like UI")`



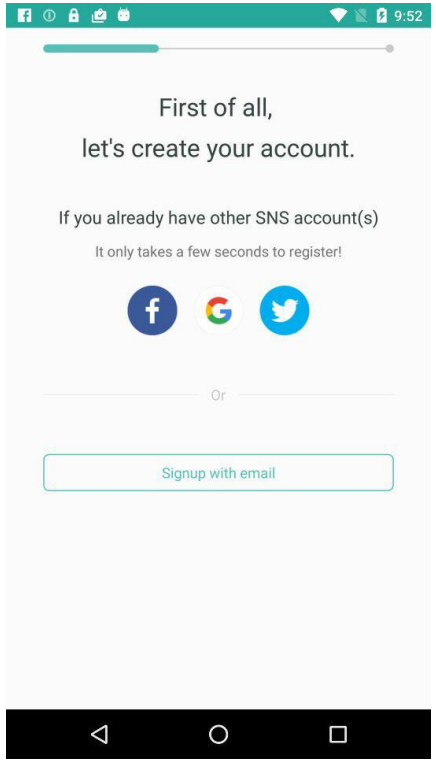
✓ login



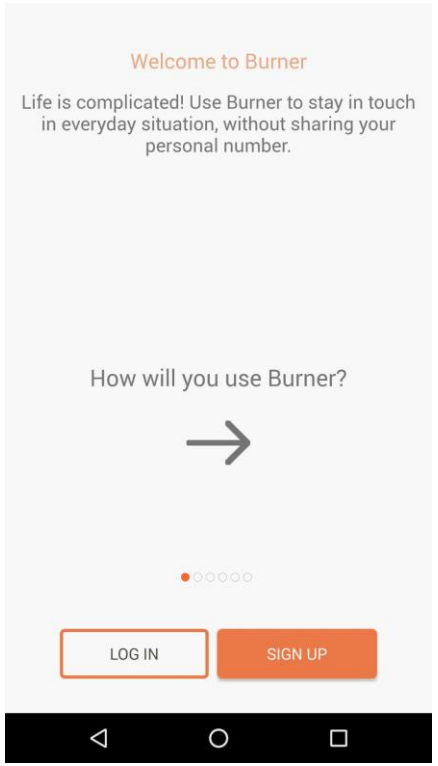
✓ login



✓ login



✓ login



? tutorial

Results

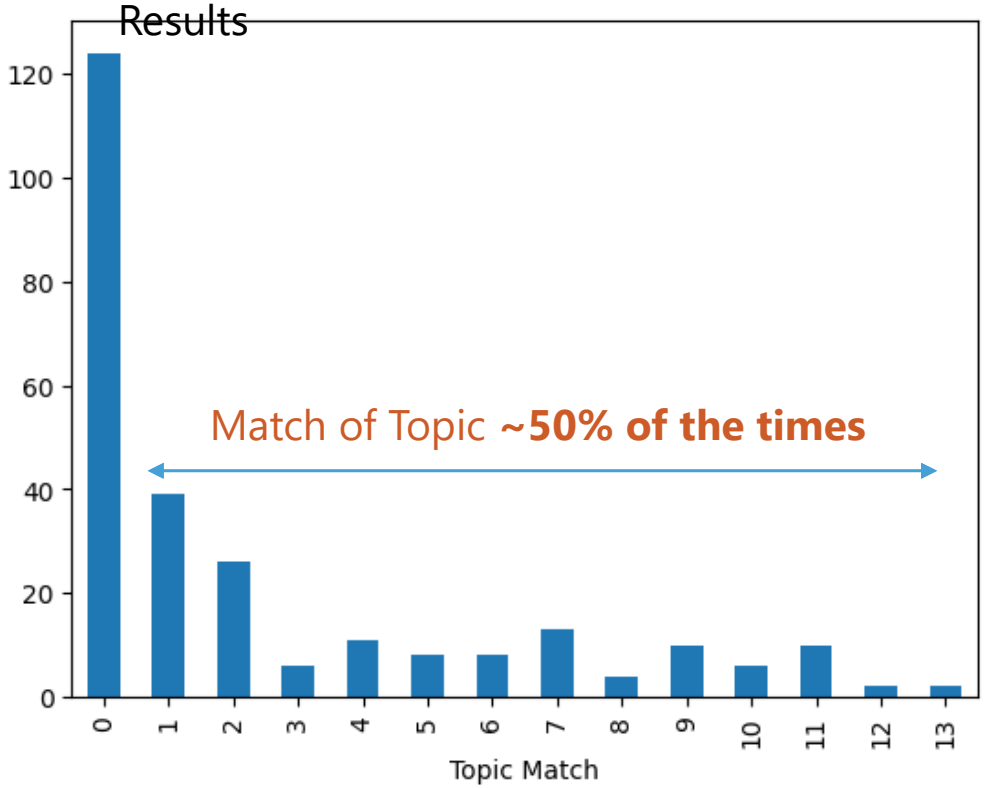
3 Display a UI from a catalogue of pre-existing UIs from a description

Results of feeding the Caption dataset as a query

caption
The screen is a tutorial screen having a large text element located at the top part of the screen.
A search app with a large background image ubicated at the center part.
A gallery app with a large image element placed at the top-left part.
This app is a gallery screen having an image situated at the top-left area of the screen.
The app looks like a gallery app with an image component ubicated at the right area.
...
A modal app having a text button element placed at the center area.
A tutorial app with an image component situated at the top part.

~16/250 Exact Match of the expected JSON file

Number of Topic Matches in the Top 20 Results



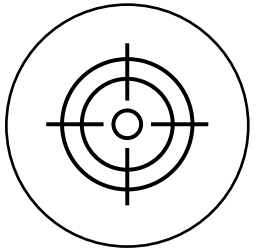
Results

3 Display a UI from a catalogue of pre-existing UIs from a description

Structurally Working

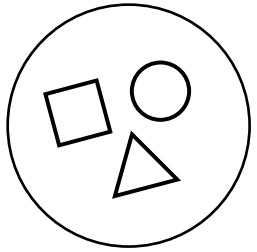
✓

Simply returns existing View Hierarchies



Accurate

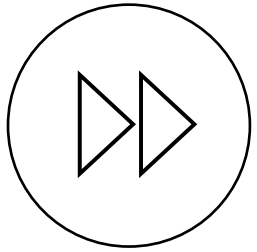
?



Easy to use

?

If the Description is very specific to the content and existing metadata the model performs well, but when the user is given too much freedom the results are bad.



Fast

✓

1-2 seconds
Must have vector store loaded in the background and run the engine in real-time

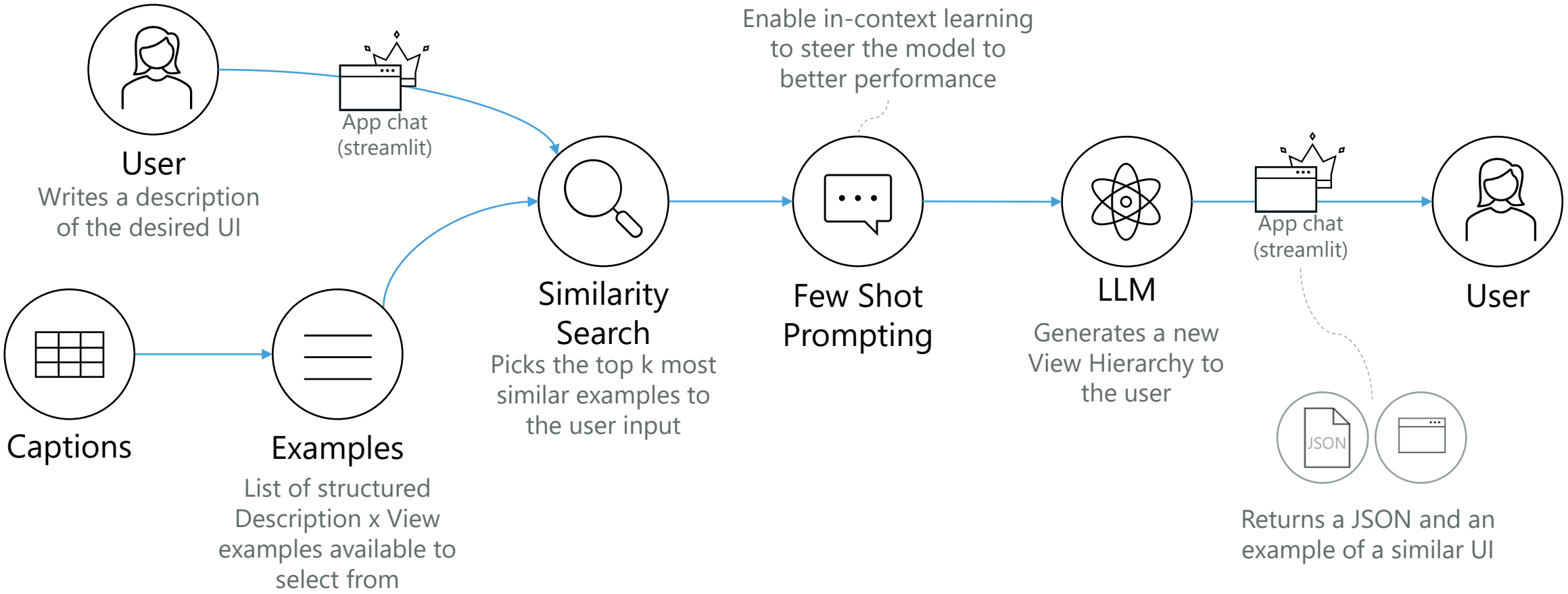
Future work

3 Display a UI from a catalogue of pre-existing UIs from a description

- **Evaluate performance on more realistic queries**
- **Reinforcement Learning –** if deployed, this solution could be improved by rewarding the model when there's a match between a user queries <> selected UI.
    - Display relevant and less relevant UIs
    - Store User queries and decisions
    - If similar descriptions lead to similar UI selections, that UI should be prioritized to the user

Thecnical Approach

4 Display newly generated UIs that match a Description

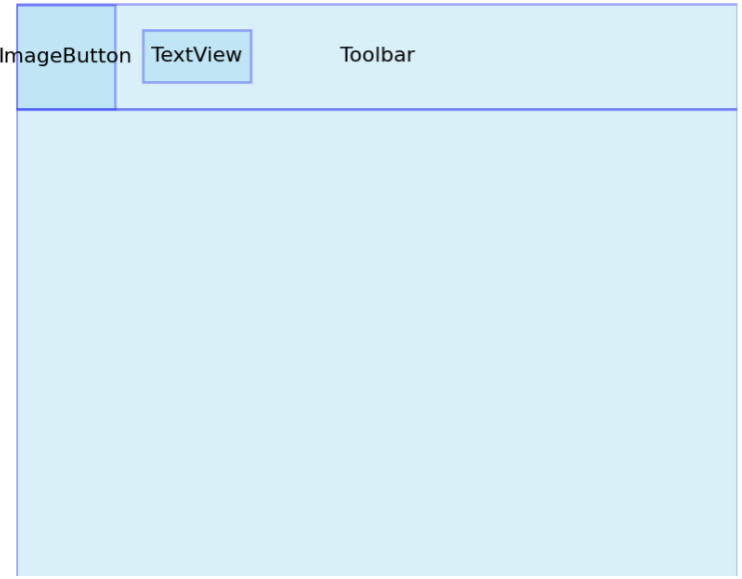


Results

4 Display newly generated UIs that match a Description

```
I want a screen that has an image
```

Example UI



Check the demo!

JSON Screenshot

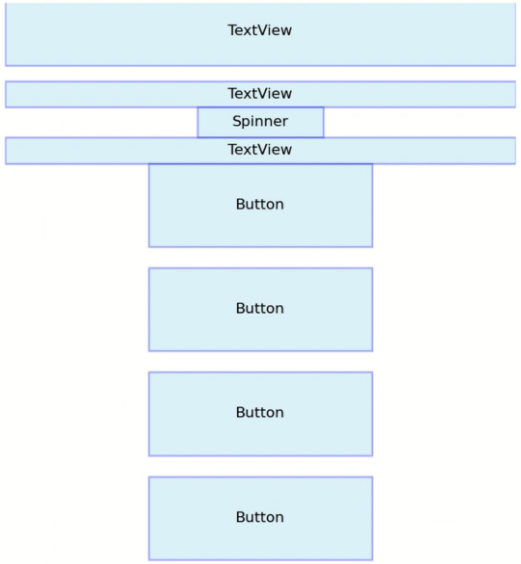

```
{
  "ancestors" : [
    0 : "FrameLayout"
    1 : "ViewGroup"
    2 : "View"
    3 : "Object"
  ]
  "class" : "PhoneWindow$DecorView"
  "children" : [
    0 : {
      "children" : [
        0 : {
          "iconClass" : "arrow_backward"
          "ancestors" : [
            0 : "ImageButton"
            1 : "ImageView"
            2 : "View"
            3 : "Object"
          ]
          "class" : "ImageButton"
          "Label" : "Icon"
```

Results

4 Display newly generated UIs that match a Description

```
I want a screen that has at least one
button at the bottom part
```

Example UI



Check the demo!

JSON Screenshot


```
"ancestors" : [
  0 : "android.widget.TextView"
  1 : "android.view.View"
  2 : "java.lang.Object"
]
"class" : "bitsie.playmee.musicplayer.free.ui.TypefaceTextView"
"componentLabel" : "Text"
}
1 : {
"text" : "Next"
"ancestors" : [
  0 : "android.widget.TextView"
  1 : "android.view.View"
  2 : "java.lang.Object"
]
"class" : "bitsie.playmee.musicplayer.free.ui.TypefaceTextView"
"componentLabel" : "Text Button"
"textButtonClass" : "next"
}
2 : {
"children" : [
  0 : {
    "text" : "English"
```

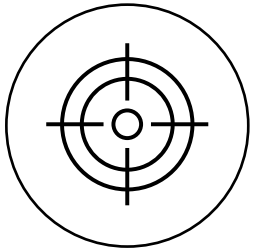
Results

4 Display newly generated UIs that match a Description

Structurally Working

✓

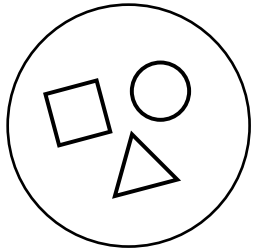
Seem to be working due to the few shot templates



Accurate

✓

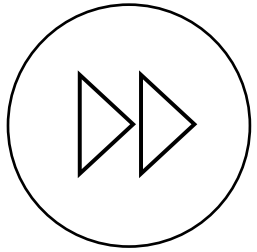
Seems to be accurate further testing and visualization would help have a more precise metric



Easy to use

✓

Simple text Input



Fast

✗

~20 seconds
Needs to be optimized

Future work

4 Display newly generated UIs that match a Description

- **Supervised model on top of a labelled dataset –** build a supervised model that knows if a Tree Structure has a set of components (i.e., buttons, toolbar, list with K items) and can later detect what component of the tree structure should be present if that label appears in a description.
- **Guide View Hierarchy when receiving explicit commands –** if a user says he wants “two buttons”, there should be a set of rules that automatically detect the request and add the elements to the tree structure


Thank you