

Server Driven Mobile UIs

Mengda “JJ” Qi

GOTO Conference: Chicago 2020

About Me



Arizona Native / UIUC Grad

Software Engineer @ Capital One

3 years on web backend / frontend / devops

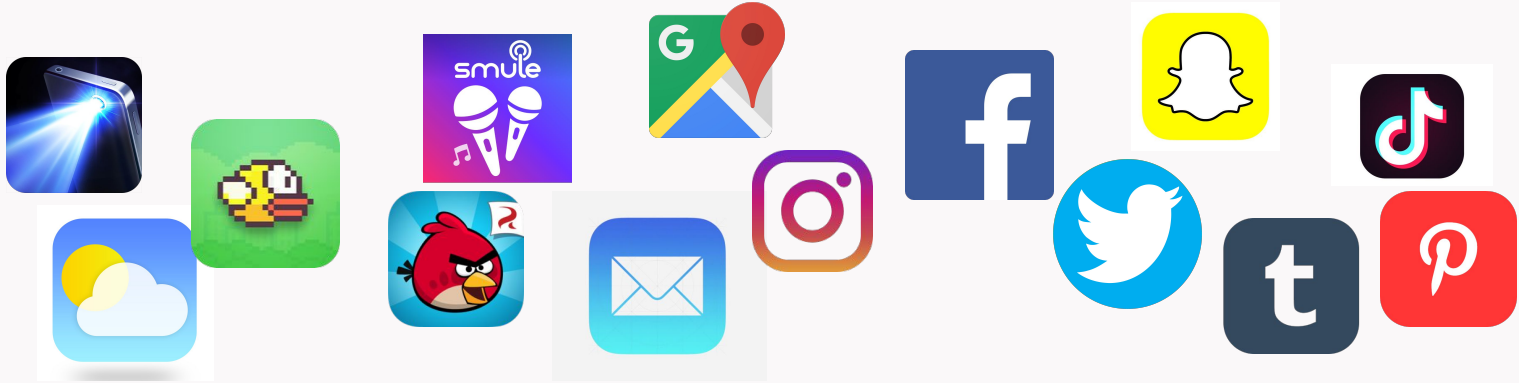
1 year on mobile (iOS)

Currently focused on bringing transactions insights to customers

Overview

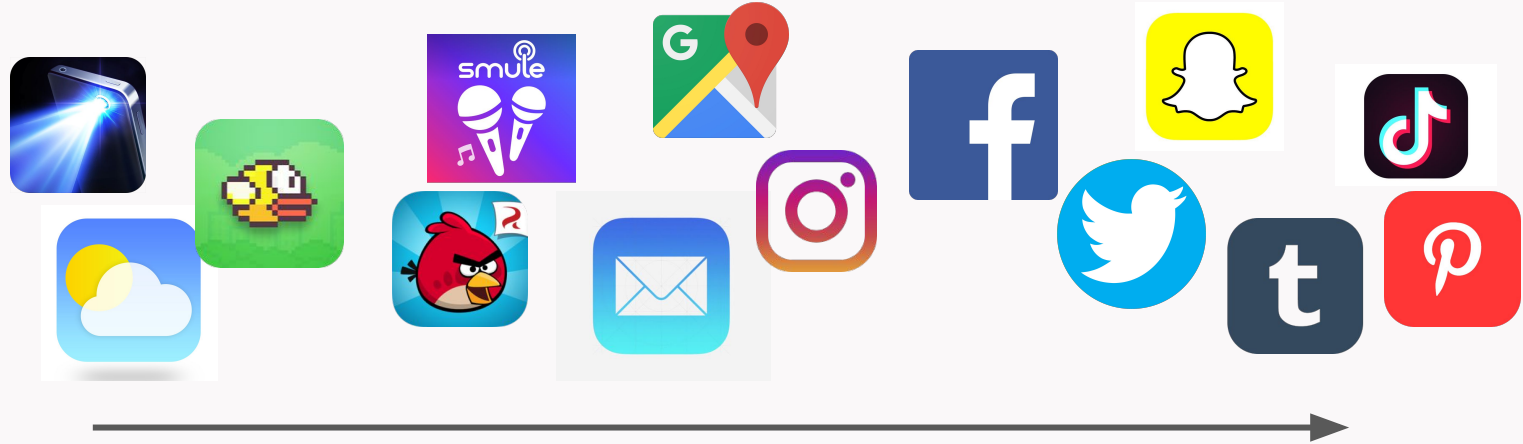
- A brief history of apps
- What does a server driven UI look like?
- What are some valid use cases?
- Getting started

App Evolution



Apps have come a long way

App Evolution



Personalized
Smart
Dynamic
Complex

Apps have come a long way

Challenges

Context Driven

- Apps behave differently depending on its current context
- ML enables hyper-personalization of content and experiences depending on the customer

More Features

- Apps have to support more features, screen real estate is the same
- Apps need to actually be useful

Not this

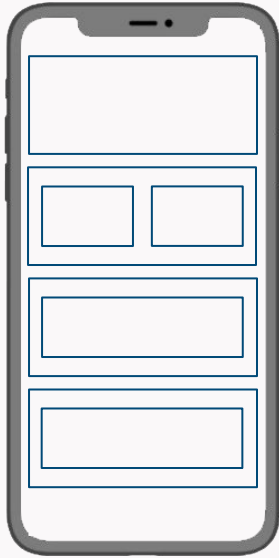
```
if(product == Products.CreditCard) {  
    if(customerType == Customer.Primary) {  
        if(customerStatus == Status.Active) {  
            if(customerAccounts.length > 2) {  
                if(featureToggleOn) {  
                    displayIcon()  
                }  
            }  
        }  
    }  
}
```

Front-ends should be dumb

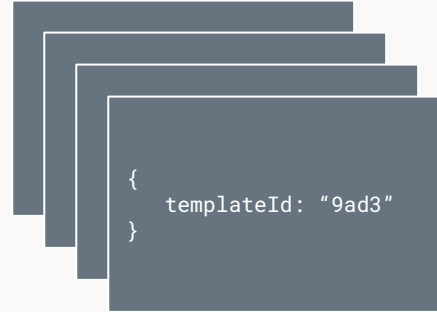
What goes into a server driven UI?



What does a server driven UI look like?



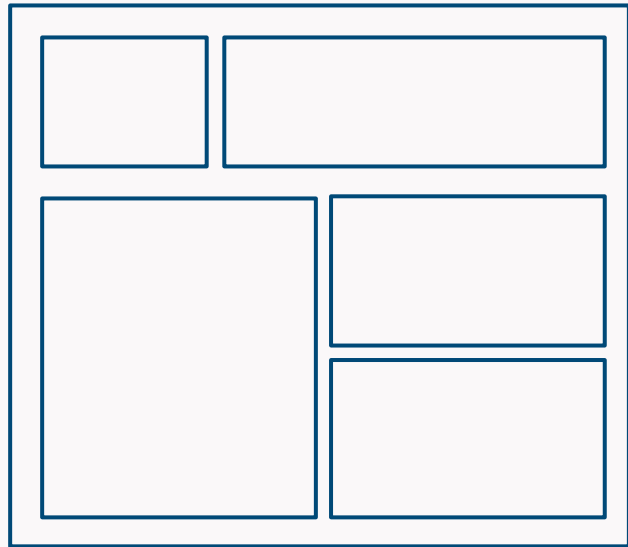
Frontend



Backend

Frontend Templates

- Templates are predefined UI layouts
- Templates have static layout, but dynamic content
- Templates can contain
 - Text labels
 - Images
 - Buttons
 - Input fields
 - Toggles
- Templates have their own implementation per platform
 - iOS - Android - Web



Example - Notifications Widget

- Notifications can be turned on and off from a toggle switch
- Icon changes when toggled on
- Additional text element appears to signal device state

Turn on Notifications



Turning on notifications helps you stay on top of the latest updates to your account. Updates are automatically sent to your device.

By turning on notifications, you accept our [Terms and Conditions](#)



Turn on Notifications



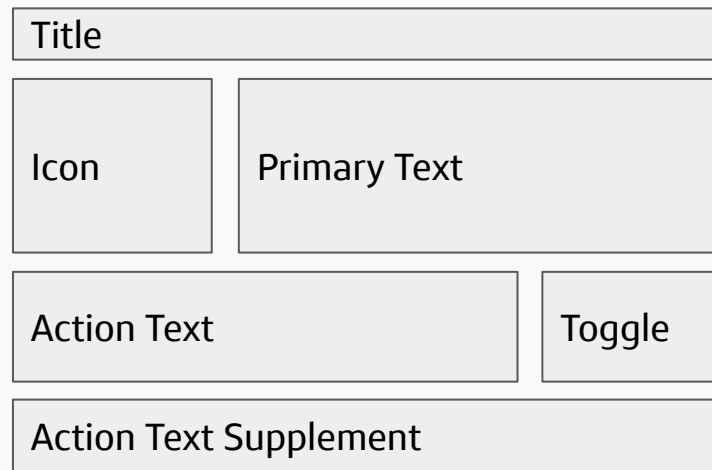
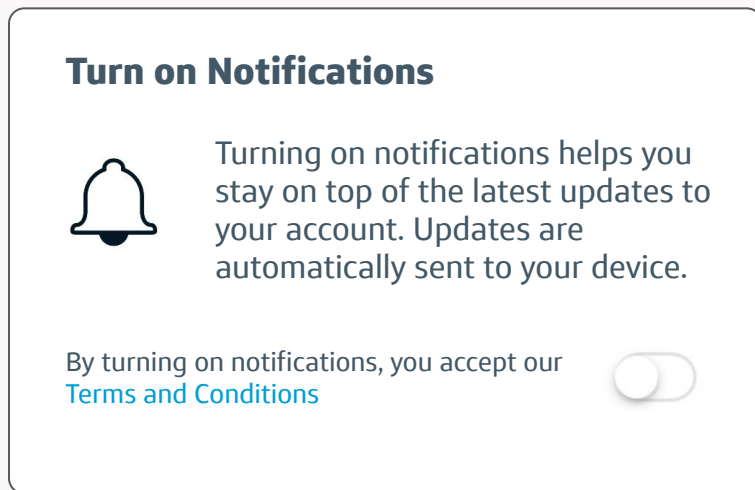
Turning on notifications helps you stay on top of the latest updates to your account. Updates are automatically sent to your device.

By turning on notifications, you accept our [Terms and Conditions](#)



Notifications have been enabled for this device.

Example



Example

Turn on Notifications



Turning on notifications helps you stay on top of the latest updates to your account. Updates are automatically sent to your device.

By turning on notifications, you accept our [Terms and Conditions](#)



Notifications have been enabled for this device.

Title

Icon

Primary Text

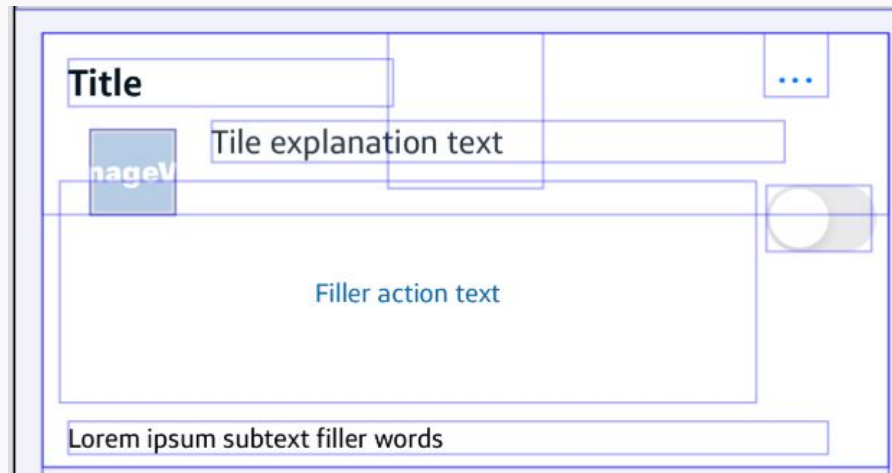
Action Text

Toggle

Action Text Supplement

UIView

```
struct ToggleView: UIView {  
    @IBOutlet weak var title: UILabel!  
    @IBOutlet weak var primaryText: UILabel!  
    @IBOutlet weak var icon: UIImageView!  
    @IBOutlet weak var actionText: UILabel!  
    @IBOutlet weak var toggle: UISwitch!  
    ...  
}
```



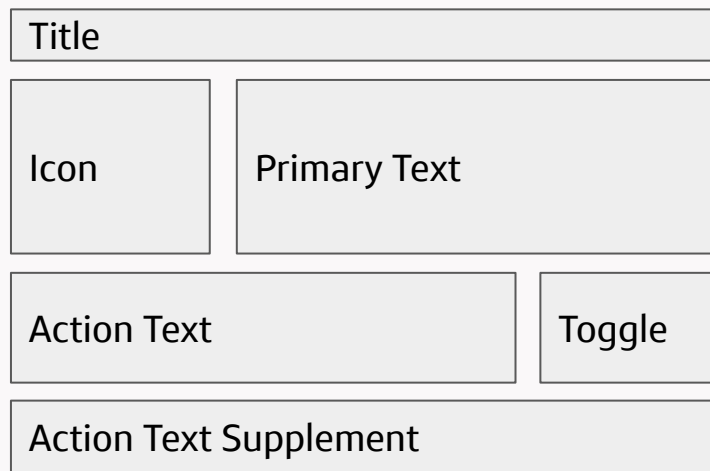
Backend

- Backend consists of an API server that serves JSON
- JSON follows standard template contract and is consistent across platforms
- JSON specifies content which includes
 - Text
 - Images
 - Actions
 - Conditional logic
 - ... and more
- A single data source means that any update to content and page flow only needs to happen once

```
{
  "type": "BASIC_TOGGLE",
  "id": String,
  "labels": [
    {
      key: ...
    }
  ],
  "actions": [
    {
      key: ...
    }
  ],
  "images": [...],
  "options": [...],
  "templates": [{
    "key": ...
  }],
}
```

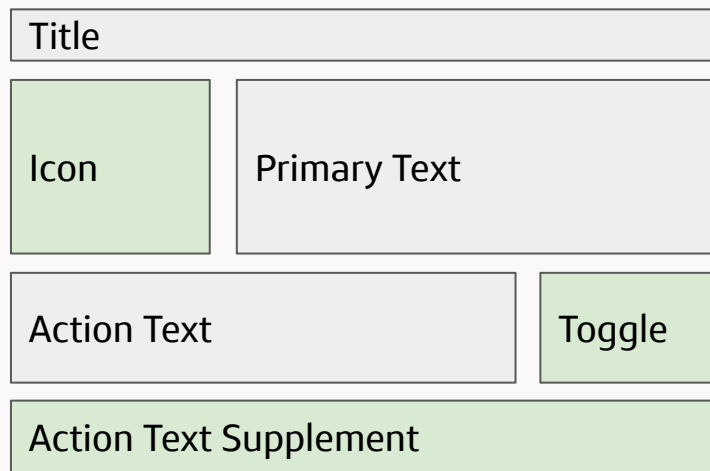

Tying it together

```
{
  "type": "BASIC_TOGGLE",
  "id": "NOTIFICATIONS",
  "labels": [{
    "key": "primary",
    "text": "Turning on notifications to
receive the latest updates to your device.",
  }, {
    "key": "title",
    "text": "Turn on Notifications",
  }, {
    "key": "actionText",
    "text": "By turning on notifications, you
accept our Terms and Conditions",
  }],
  ...
  "images": [{
    "key": "iconPrimary",
    "url": "https://contentserver.storage.."
  }],
}
```



JSON - Actions

```
"actions": [{
  "key": "primaryAction",
  "type": "localUpdate",
  "signals": [{
    "key": "SUCCESS",
    "signalId": "notificationsToggleSuccess",
    "adjustments": [{
      "type": "REPLACE",
      "relativeIconId": "primaryIcon"
    }]
  }],
  {
    "key": "FAILURE",
    "signalId": "notificationsToggleFailure",
    "adjustments": [{
      "type": "REPLACE",
      "relativeLabelId": "actionTextSupplment"
    }]
  },
  ...
}]
```



JSON Spec - API Call

```
"actions": [
  {
    "key": "primaryAction",
    "type": "triggerExternalUpdate",
    "signals": [
      {
        "key": "API_UPDATE",
        "signalId": "callSettingsAPI",
        "adjustments": [
          {
            "type": "API_CALL",
            "relativeTileId": "notifications",
            "apiSpec": {
              "method": "POST",
              "path": "/notifications/update",
              "body": "{...}"
            }
          }
        ]
      }
    ]
  }
]
...
```

Template Examples

A variety of common UI elements can be templated


- Menu items
- Modals
- Forms
- Toggles
- Charts

Success!

Your action was successful

[Acknowledge](#)

Update Your Email



Update your email to get the latest and greatest and latest

Rewards Bonus

You're 500 points away from earning your next rewards bonus!

Policy Updated

We've made some new updates to your account policy that may affect your status.

[View Policy](#)

Update Your Address

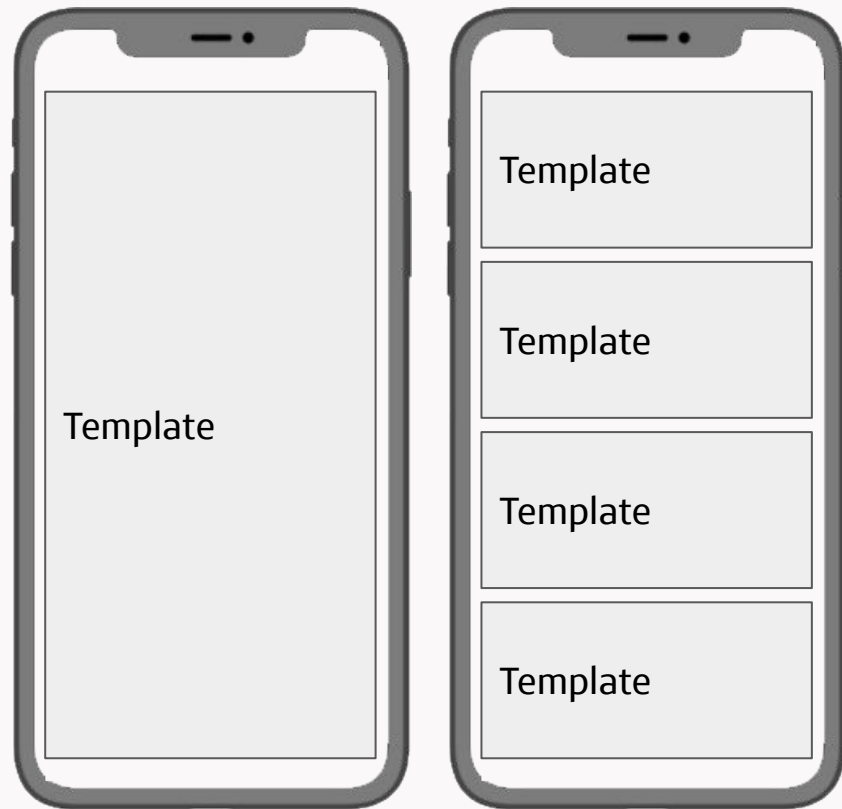
Name

Street

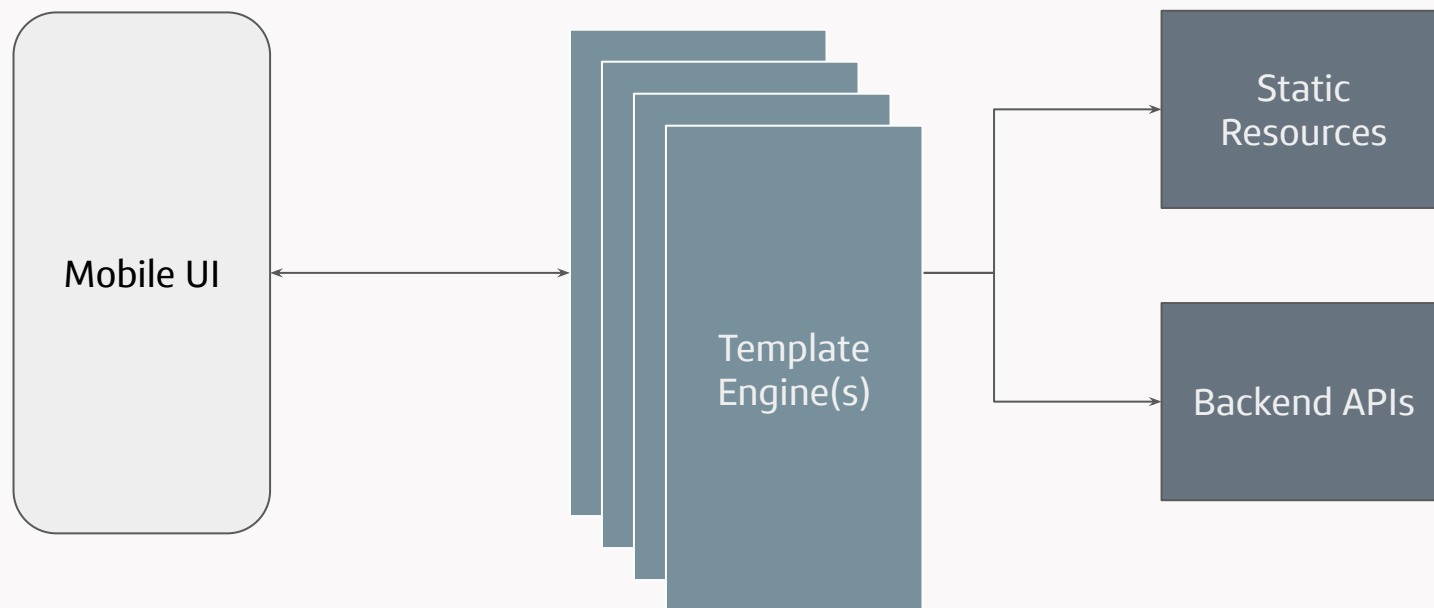
City

Nested Templates

- Templates should generally be self contained
- Templates can live in a parent view
 - E.g. TableView or CollectionView
- Templates can also have other nested templates as part of the specification

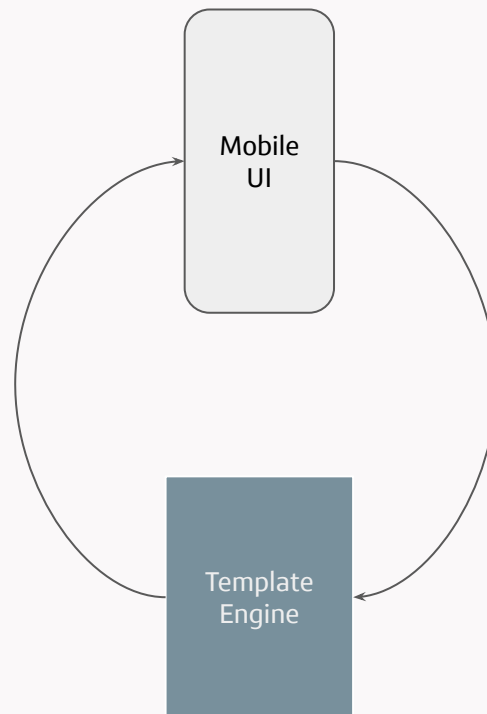


Data Flow



Data Flow - Example

1. Mobile UI initiates call to Template Engine
2. Template Engine returns initial JSON
3. Mobile UI triggers local/external actions
4. Mobile UI calls Template Engine
5. Template Engine returns next set of JSON
6. Repeat until finished



Use cases



Use Case - A/B Testing

Enable Notifications



Enabling notifications helps you stay on top your latest alerts. We'll send you updates as they come.

By enabling notifications, you accept our
[Terms and Conditions](#)



Turn on Notifications



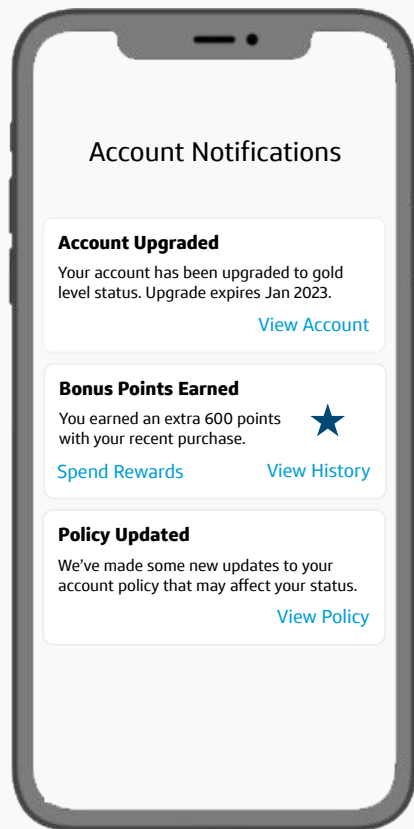
Turning on notifications helps you stay on top of the latest. Updates are automatically sent to your device

By turning on notifications, you accept our
[Terms and Conditions](#)



- Dynamic layouts are great for A/B Testing
- Server determines which assets to load based on your own criteria
- Iterate through different prototypes quickly

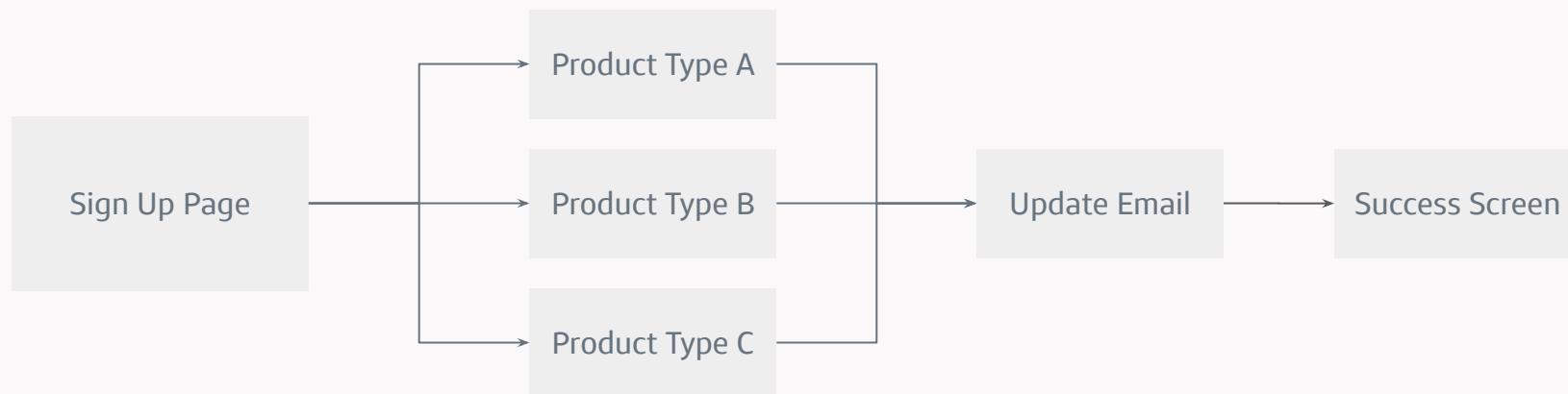
Use Case - Dynamic Messaging



- Publish important messaging through templates
- Having a variety of template types can help provide more specific details per user
- No need for app updates to provide critical messaging

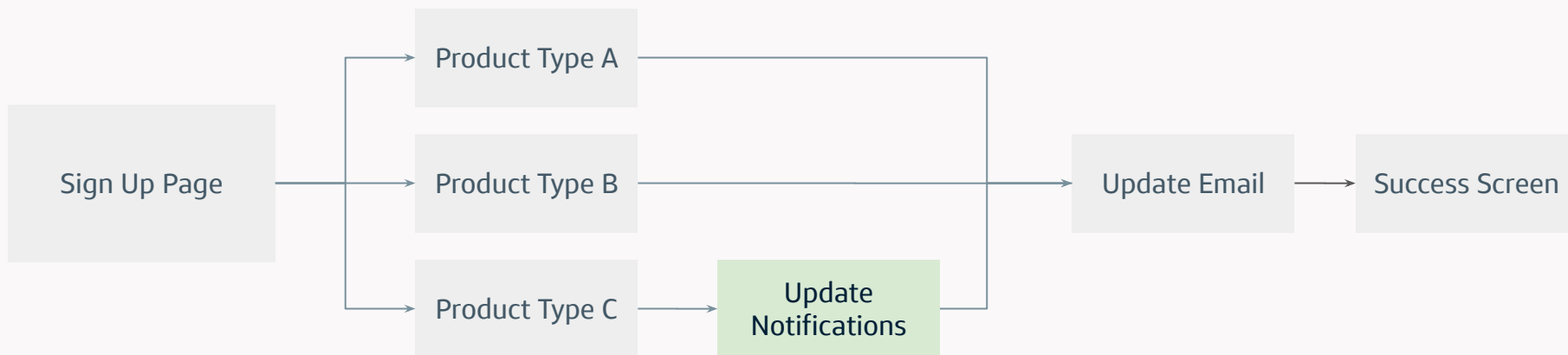
Use Case - User Setup

- User setup can be customized based on product type
- You can easily create non-linear flows



Use Case - User Setup

- You can easily add any new screen without UI updates
- Isolation of view & business logic enables better testing
 - Testing of flows can be done on backend!



Getting started



Tips & Tricks

1. Define a good use case first
 - a. Dynamic and simple content is the best way to start
 - b. Find duplicate views and features that are consistent across platforms
2. Define your template
3. Define your template contract
 - a. Determine what are the necessary components that can populate your template
4. Start with static JSON
 - a. Host it on the app to start
5. Incrementally build out backend as needed

Other Considerations

- Is this worth the extra work?
- How can I test my UI?
- How does this scale?
- Are front-end developers deprecated?

Thank you!

