Building an LLM-powered mobile app using Hotwire Native

Mike Dalton

Who am I?

- Mike Dalton
- Live in Philadelphia
- Day Job: Software Engineer at Triumph
- Night Job: Building my own apps

my social media pic

Z. Cioccolato
GBC INTERNATIONAL BANK
8 Fisherman's Wharf
MUNI
PostHog
WHO'S YOUR
DADD AI ?!

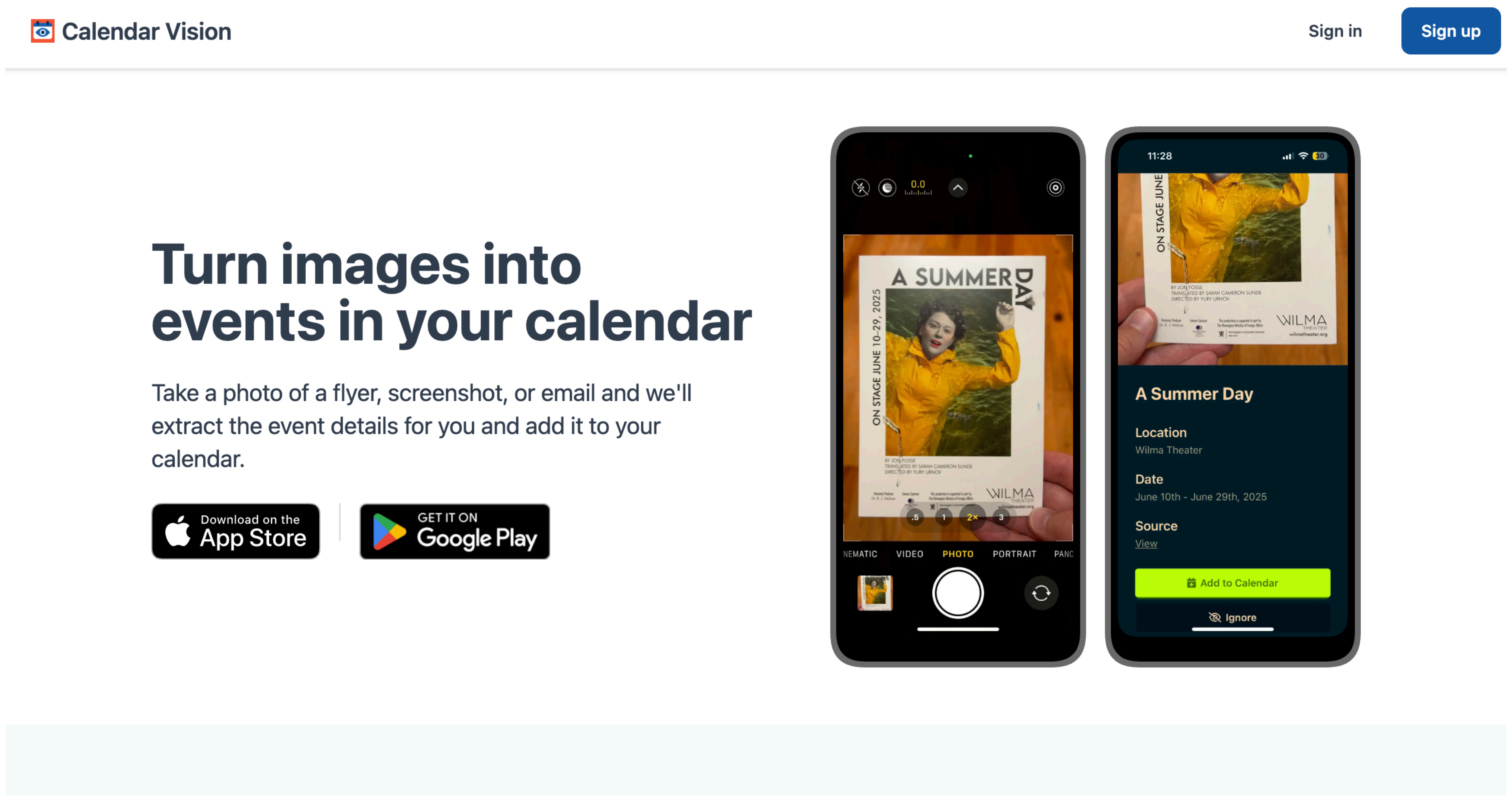
I'M JAWN MORGAN
MORGAN & MORGAN
AMERICA'S LARGEST INJURY LAW FIRM
DIAL #LAW (#529)
ThePeople.com
57 WHITMAN PLAZA

I'M JAWN MORGAN
MORGAN & MORGAN
AMERICA'S LARGEST INJURY LAW FIRM
ForThePeople.com
DIAL #LAW (#529)
8048
THIS VEHICLE MAKES WIDE TURNS

MAYO ADRIAN!
OFFICIAL MAYO AND MUSTARD
MIKE'S AMAZING REAL MAYONNAISE
OUTFRONT/
524

Let's Go Clubbing
Wawa
CLEARCHANNEL
100133

Calendar Vision



Calendar Vision

Sign in

Sign up

Turn images into events in your calendar

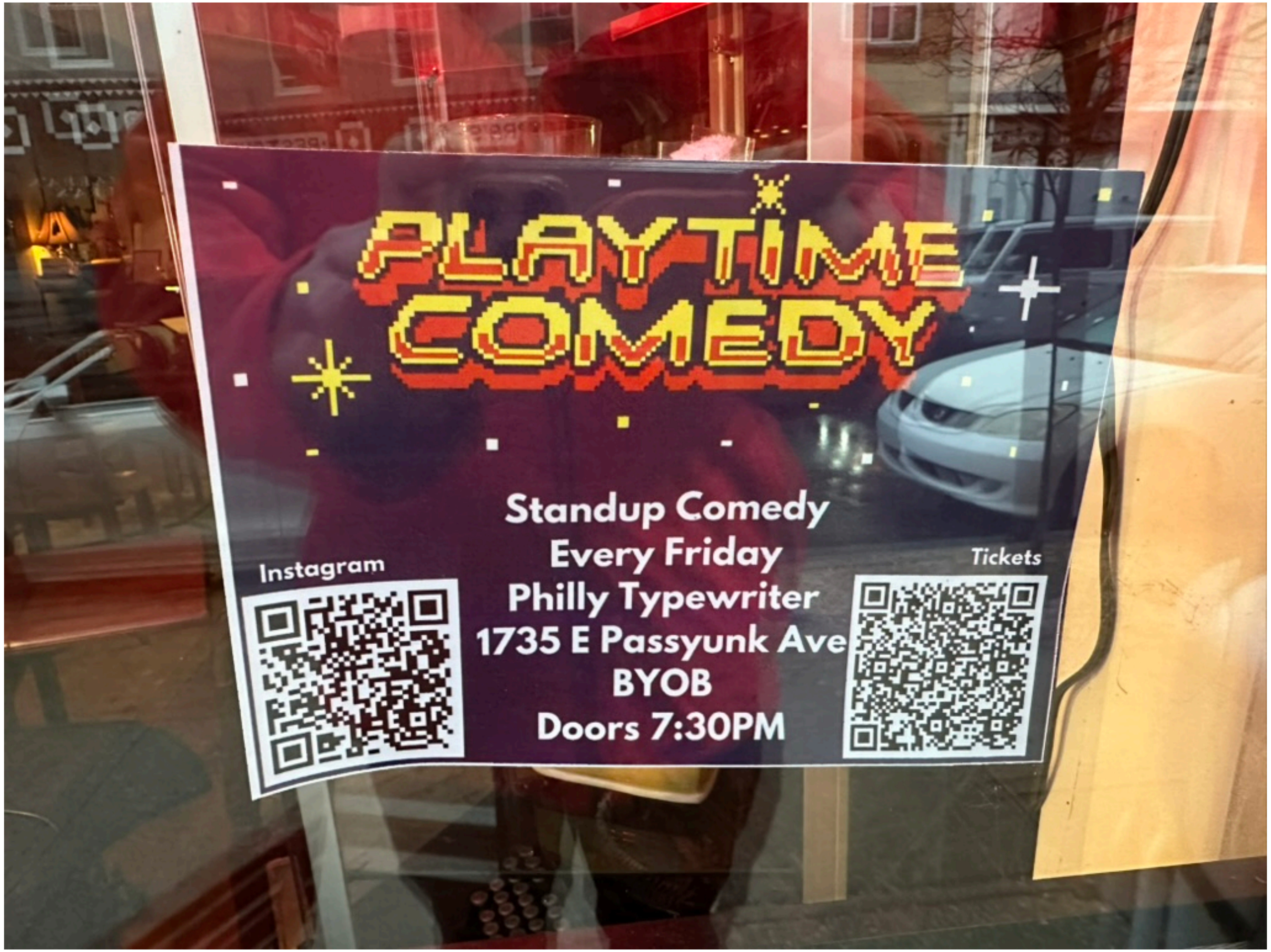
Take a photo of a flyer, screenshot, or email and we'll extract the event details for you and add it to your calendar.

A Summer Day

Location
Wilma Theater

Date
June 10th - June 29th, 2025

Source
View



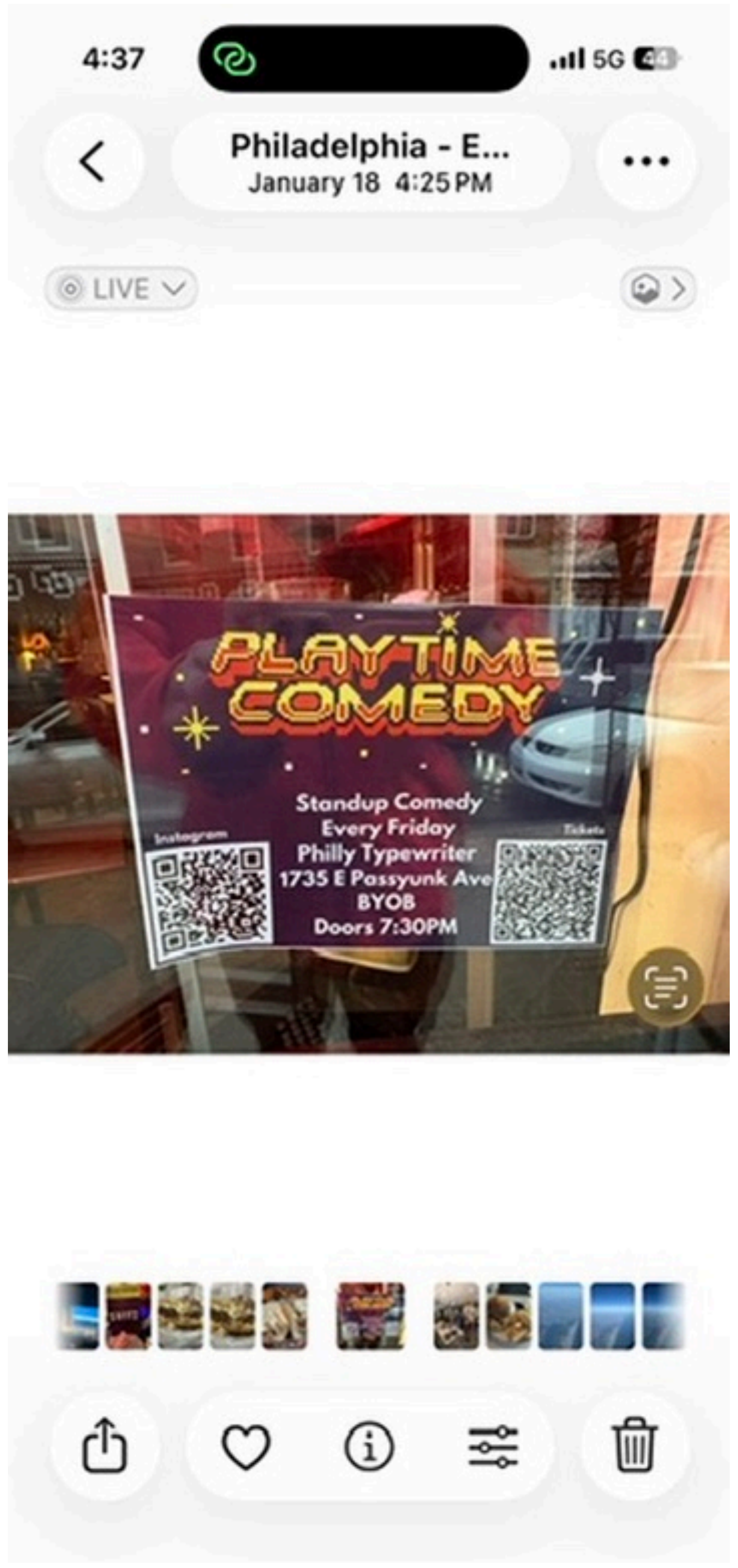
PLAYTIME COMEDY

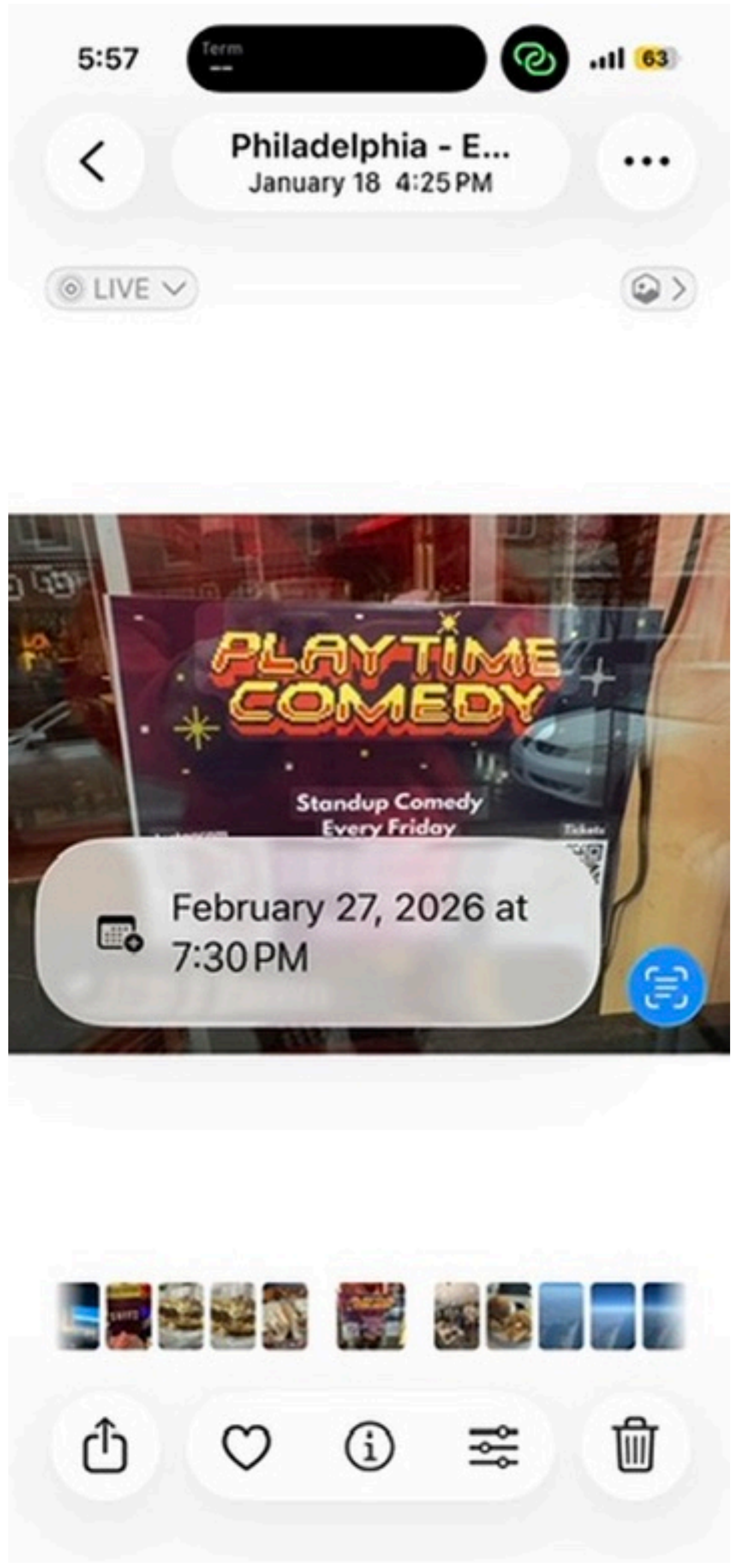
Standup Comedy
Every Friday
Philly Typewriter
1735 E Passyunk Ave
BYOB
Doors 7:30PM

Instagram

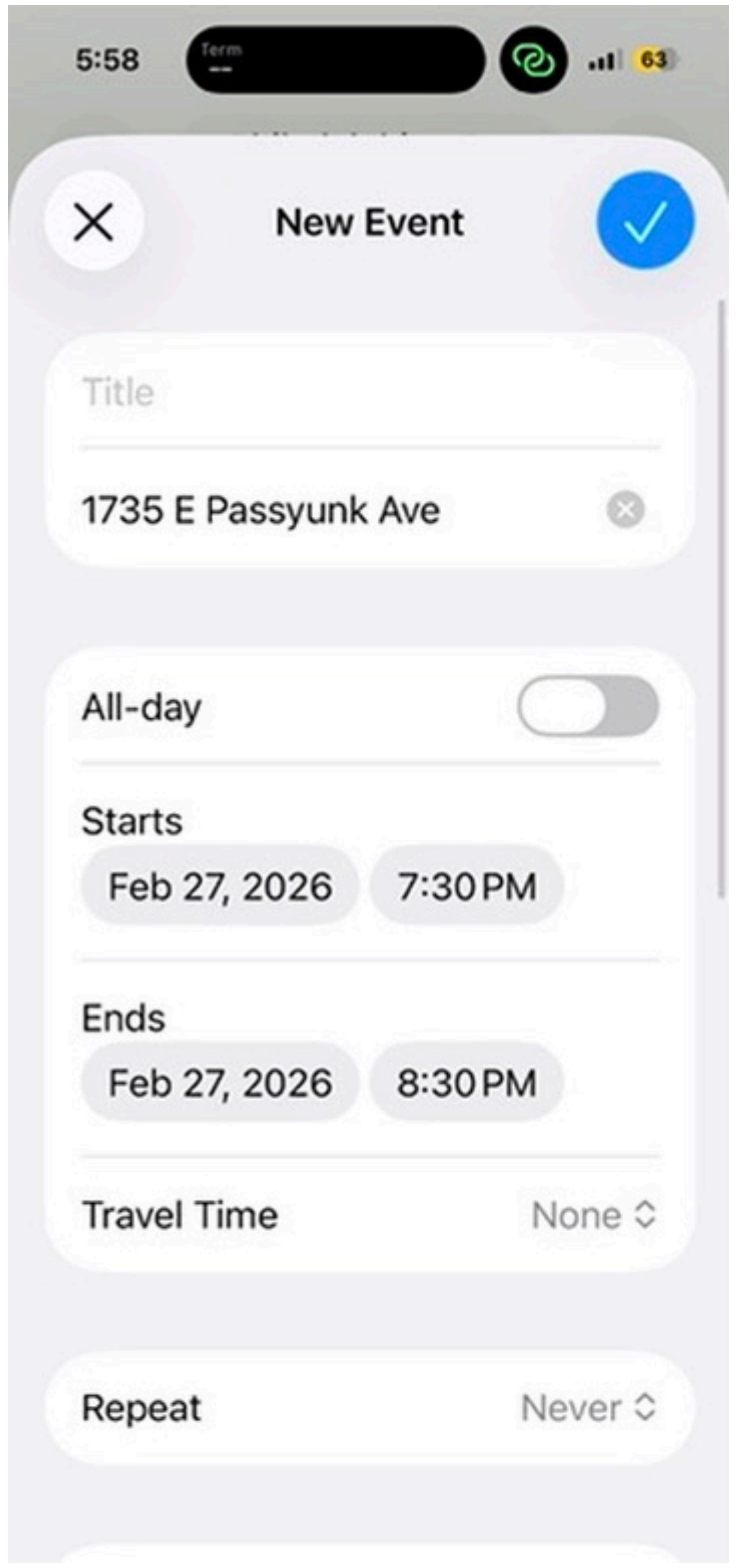
Tickets

iOS Photos App


4:37
Philadelphia - E...
January 18 4:25 PM


5:57
Philadelphia - E...
January 18 4:25 PM

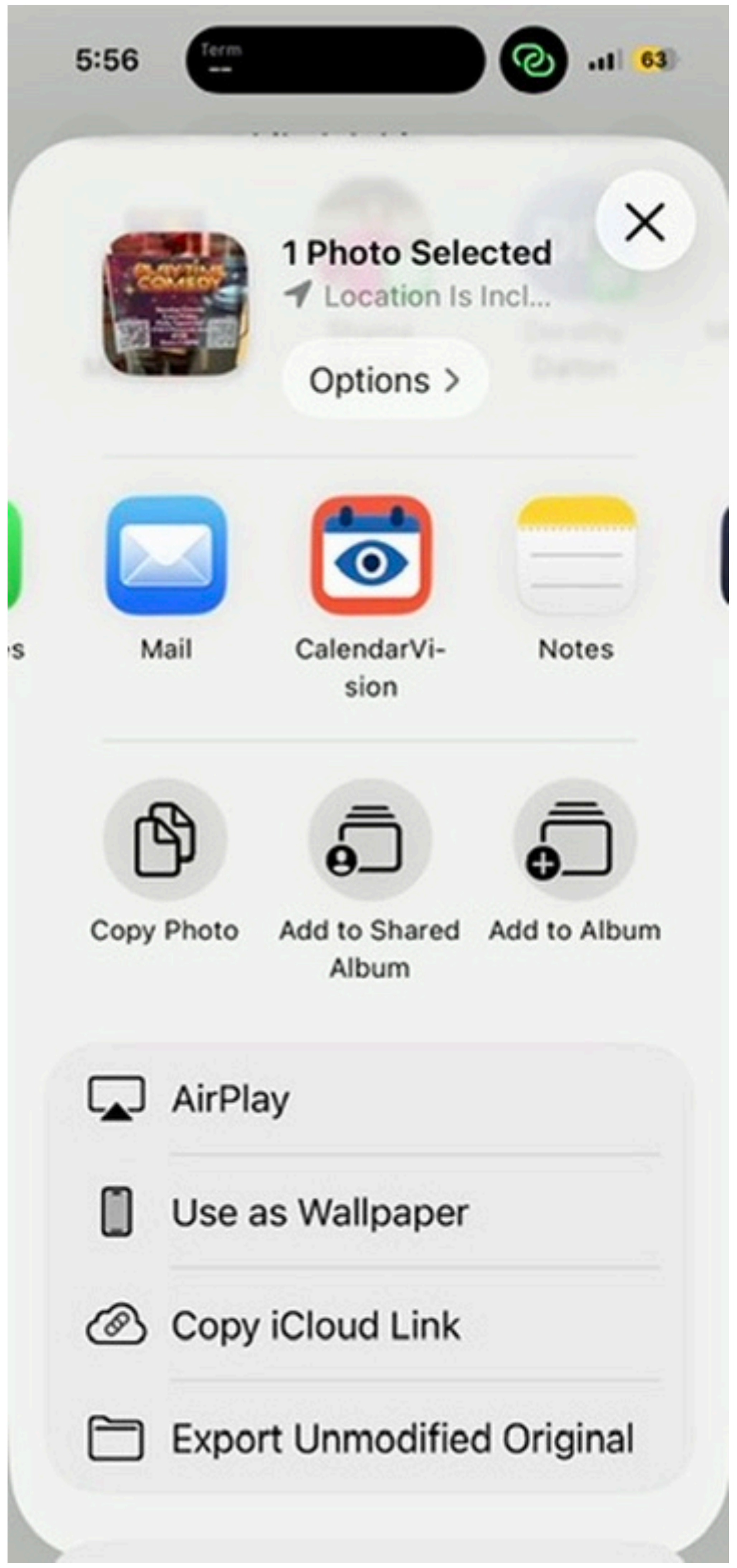
February 27, 2026 at 7:30 PM

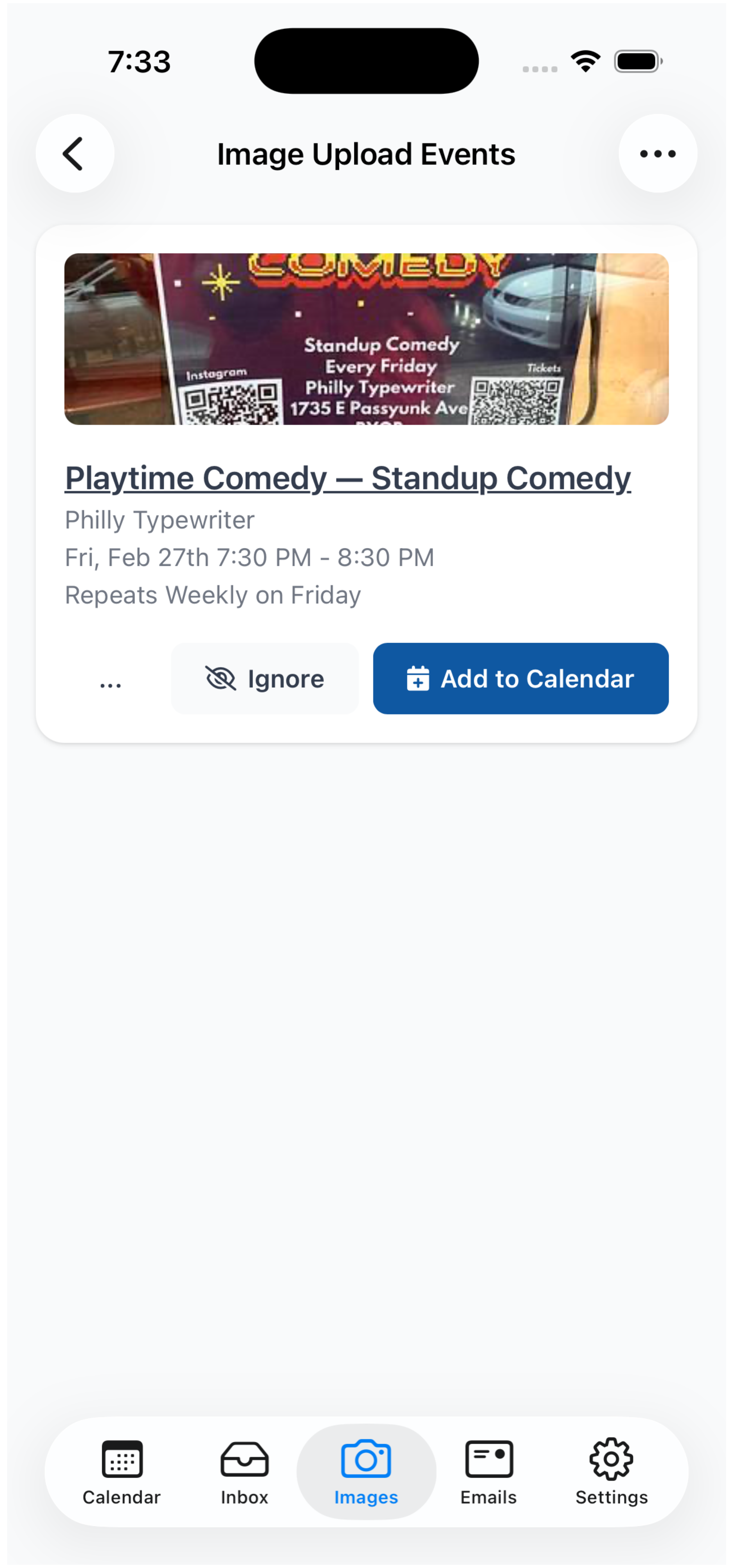

5:58
New Event

Title
1735 E Passyunk Ave

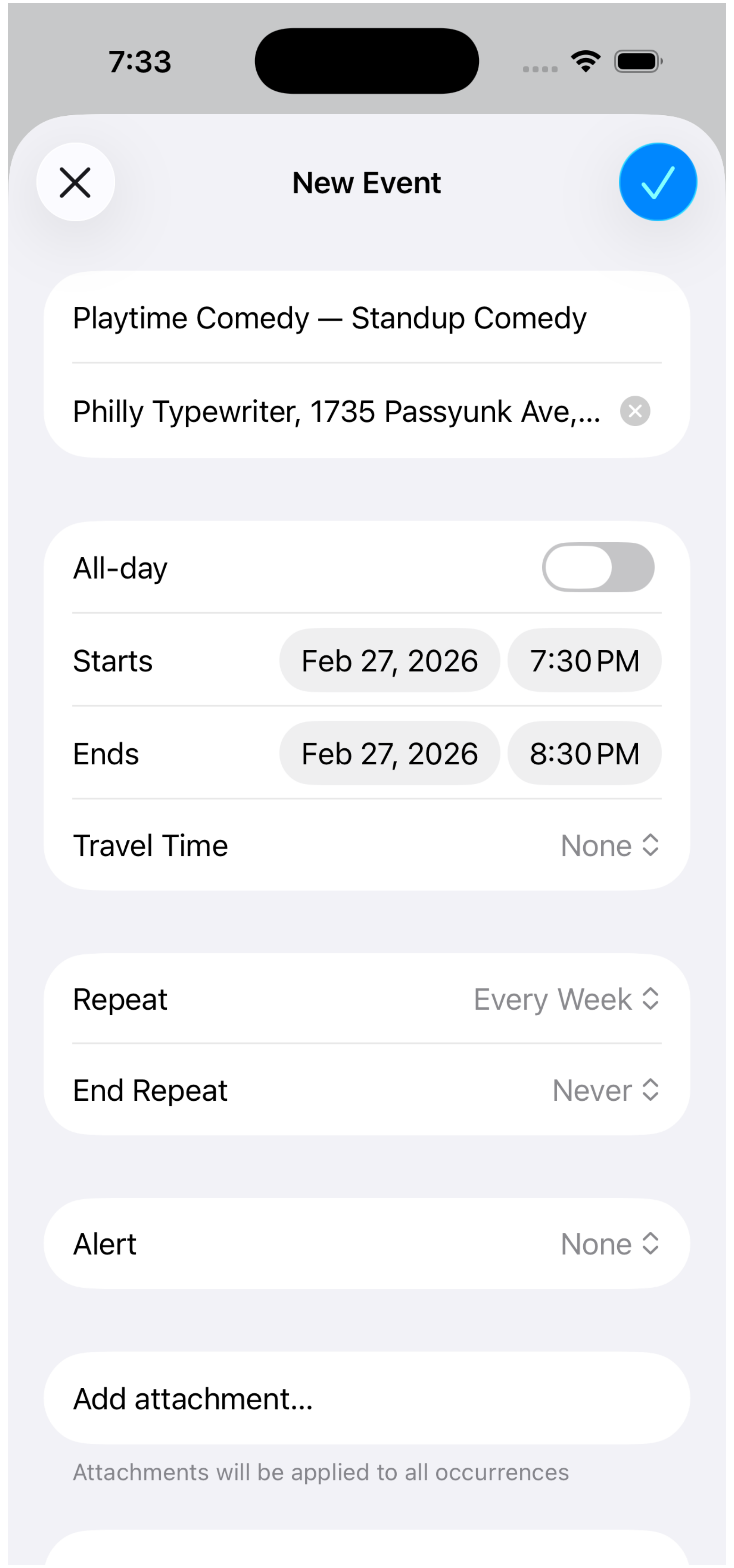
All-day	
Starts	Feb 27, 2026 7:30 PM
Ends	Feb 27, 2026 8:30 PM
Travel Time	None
Repeat	Never

Calendar Vision





Playtime Comedy — Standup Comedy
Philly Typewriter
Fri, Feb 27th 7:30 PM - 8:30 PM
Repeats Weekly on Friday



Playtime Comedy — Standup Comedy
Philly Typewriter, 1735 Passyunk Ave,...

All-day		
Starts	Feb 27, 2026	7:30 PM
Ends	Feb 27, 2026	8:30 PM
Travel Time	None	
Repeat	Every Week	
End Repeat	Never	
Alert	None	

Add attachment...

Attachments will be applied to all occurrences

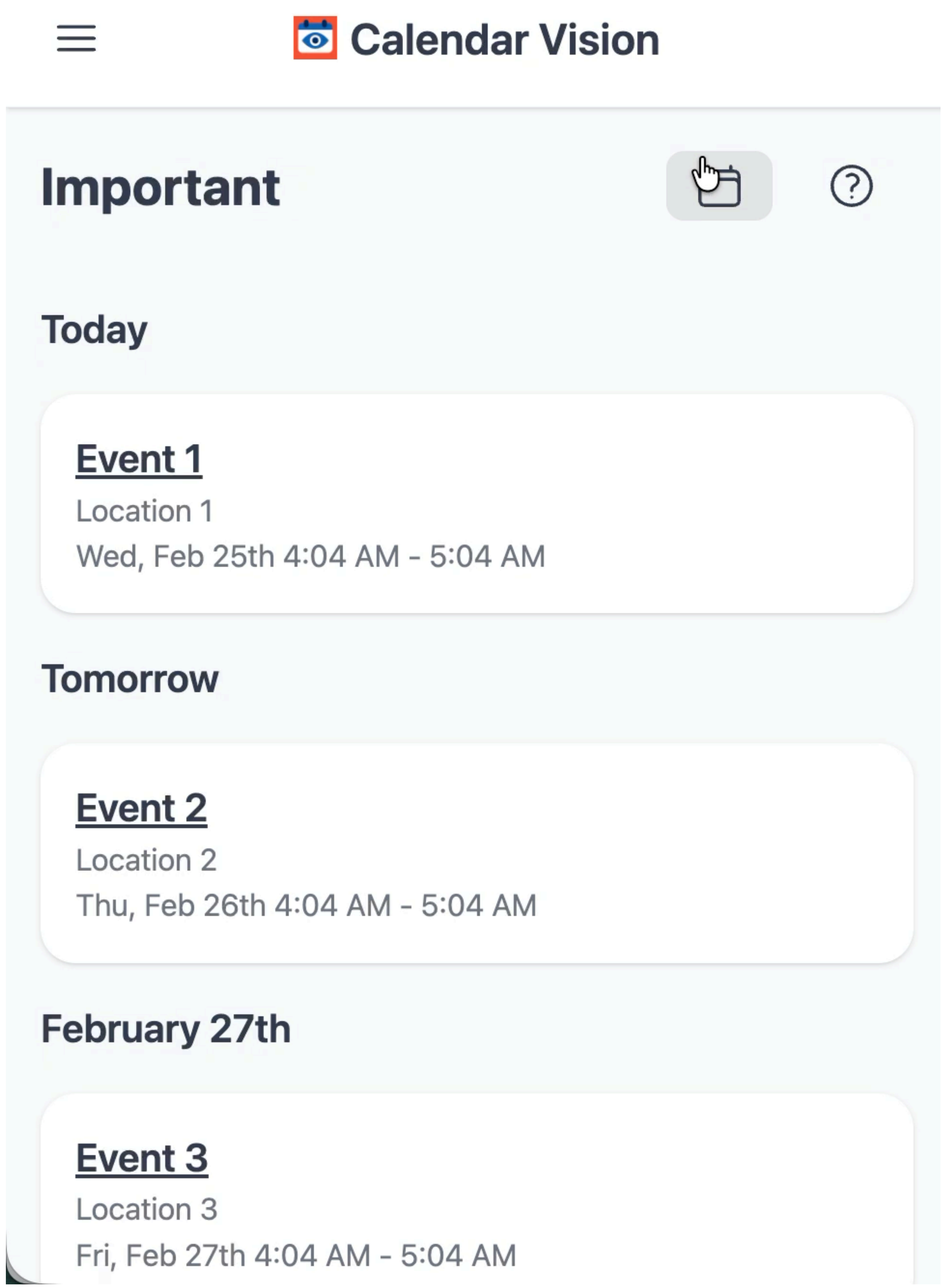
Building a mobile app with Rails

Hotwire Native

- Part of Hotwire
- Alternative native iOS, Kotlin and React Native
- Re-use most of your Rails app
- Native navigation with Rails views in web view

Calendar Switcher

Web



Calendar Vision

Important

Today

Event 1
Location 1
Wed, Feb 25th 4:04 AM - 5:04 AM

Tomorrow

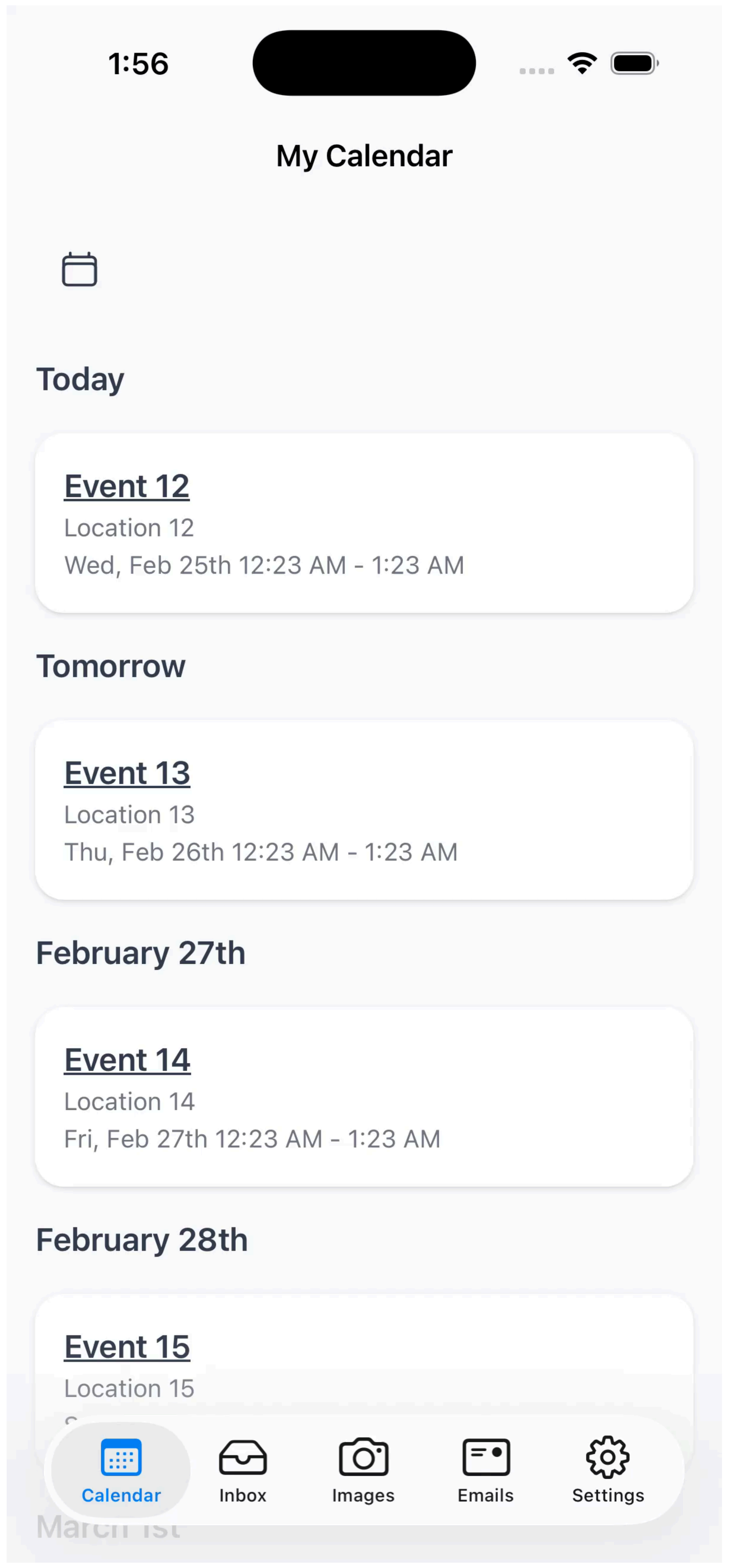
Event 2
Location 2
Thu, Feb 26th 4:04 AM - 5:04 AM

February 27th

Event 3
Location 3
Fri, Feb 27th 4:04 AM - 5:04 AM

Calendar Switcher

iOS App (HTML Dialog)



Today

Event 12
Location 12
Wed, Feb 25th 12:23 AM - 1:23 AM

Tomorrow

Event 13
Location 13
Thu, Feb 26th 12:23 AM - 1:23 AM

February 27th

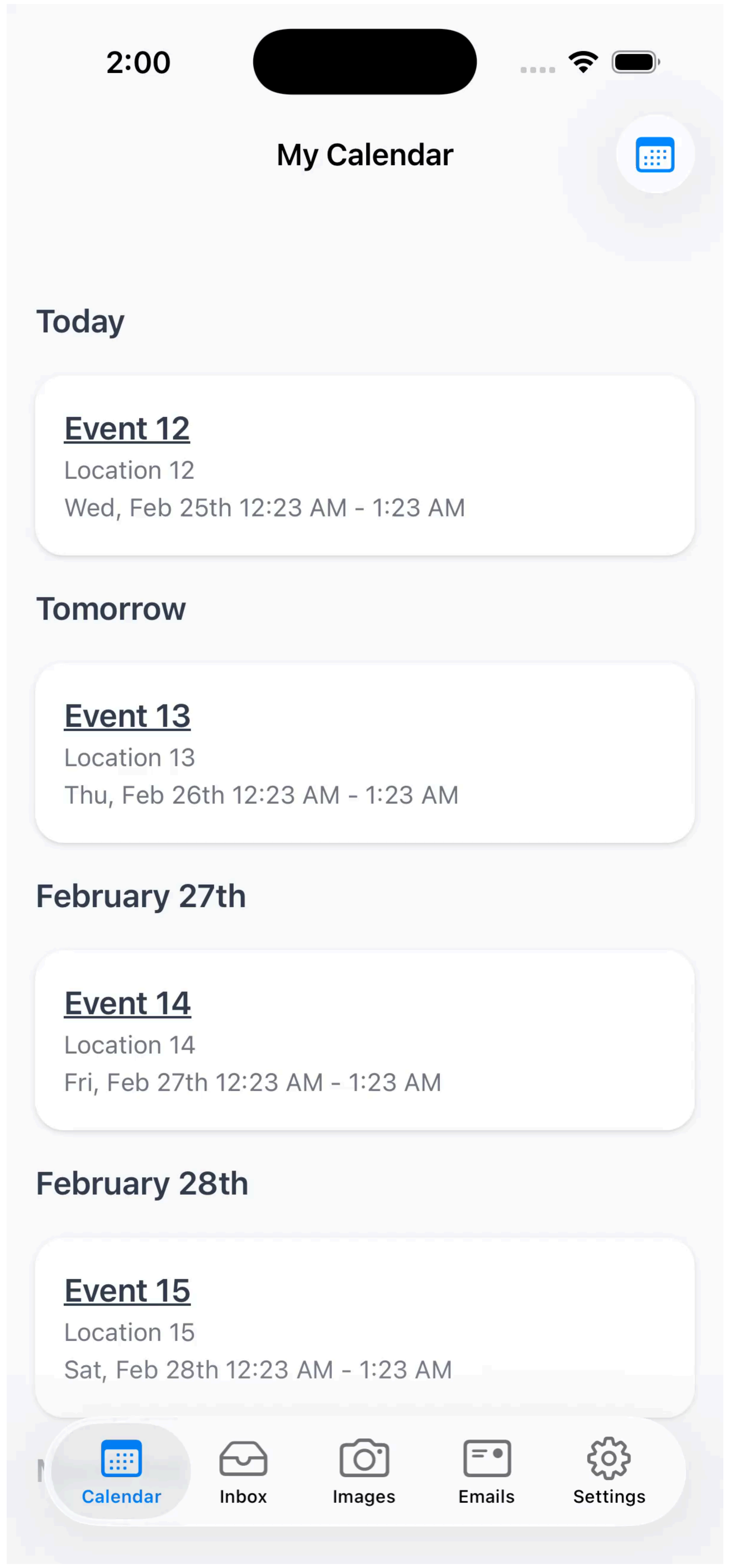
Event 14
Location 14
Fri, Feb 27th 12:23 AM - 1:23 AM

February 28th

Event 15
Location 15

Calendar Switcher

iOS App (Native Modal)



Today

Event 12
Location 12
Wed, Feb 25th 12:23 AM - 1:23 AM

Tomorrow

Event 13
Location 13
Thu, Feb 26th 12:23 AM - 1:23 AM

February 27th

Event 14
Location 14
Fri, Feb 27th 12:23 AM - 1:23 AM

February 28th

Event 15
Location 15
Sat, Feb 28th 12:23 AM - 1:23 AM

Web Implementation

```ruby
class CalendarsController < ApplicationController
  def index
    @calendars = current_user.calendars
    respond_to do |format|
      format.turbo_stream
    end
  end
end
```

app/controllers/calendars_controller.rb

```erb
<%= link_to calendars_path,
    data: {
      turbo_stream: true
    } do %>
  <%= render "heroicons/arrows_up_down" %>
<% end %>
```

app/views/calendars/show.html.erb

```erb
<%= turbo_stream.replace "dialog-container" do %>
  <div id="dialog-container">
    <dialog class="modal" open>
      <%= render "index", calendars: @calendars %>
    </dialog>
 </div>
<% end %>
```

app/views/calendars/index.turbo_stream.erb

Native Implementation

```erb
<%= link_to calendars_path,
    data: {
      turbo_stream: !hotwire_native_app?
    } do %>
  <%= render "heroicons/arrows_up_down" %>
<% end %>
```

app/views/calendars/show.html.erb

```ruby
class ConfigurationsController < ApplicationController
  def ios_v1
    render json: {
     rules: [
        {
          patterns: [
           "#{calendars_path}$"
          ],
          properties: {
            context: "modal",
            presentation: "default",
            pull_to_refresh_enabled: false
          }
        }
      ]
    }
  end
end
```

app/controllers/configurations_controller.rb

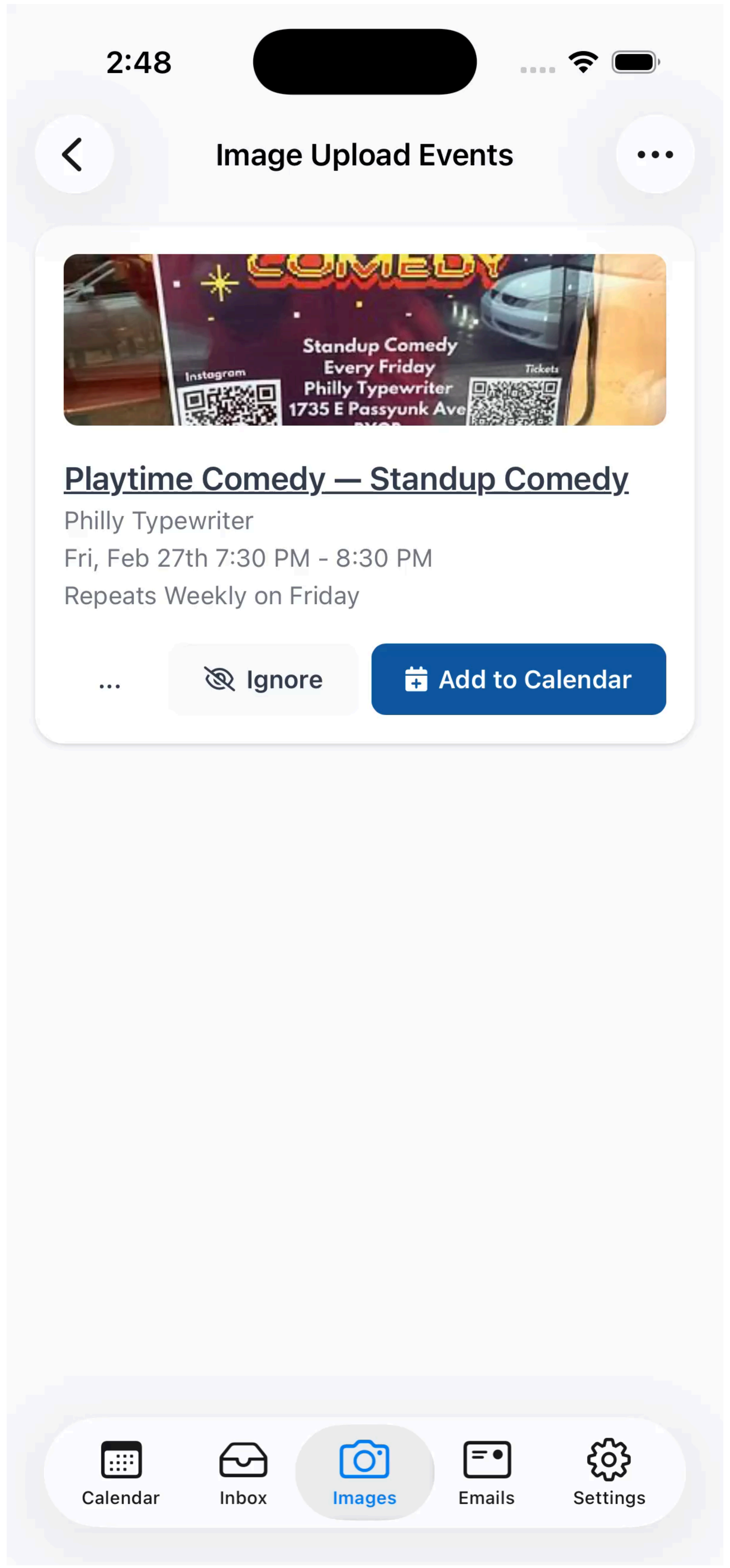
```ruby
class CalendarsController < ApplicationController
  def index
    @calendars = current_user.calendars
    respond_to do |format|
      format.html
      format.turbo_stream
    end
  end
end
```

app/controllers/calendars_controller.rb

```erb
<%= render "index", calendars: @calendars %>
```

app/views/calendars/index.html.erb

Add to Calendar



2:48

Image Upload Events

Playtime Comedy — Standup Comedy

Philly Typewriter

Fri, Feb 27th 7:30 PM - 8:30 PM

Repeats Weekly on Friday

Handle URL with a native component

```swift
class SceneController: UIResponder {
    private var addToCalendarController: AddToCalendarController?
}

extension SceneController: NavigatorDelegate {
    func handle(proposal: VisitProposal, from navigator: Navigator) -> ProposalResult {
        switch proposal.viewController {
        case "add_to_calendar":
            let idRegex = /(?<id>\d+)\/add_to_calendar/
            let url = proposal.url.absoluteString
            if let match = url.firstMatch(of: idRegex) {
                let calendarEventId = Int(match.id)!
                Task { await addToCalendarController?.addToCalendar(calendarEventId: calendarEventId) }
            }
            return .reject
        default:
            return .accept
        }
    }
}
```

SceneController.swift

Show Event View

- Get event data from Rails app
- Launch native event view

```swift
class AddToCalendarController: NSObject {
    private weak var window: UIWindow?

    init(window: UIWindow?) {
        self.window = window
        super.init()
    }

    func addToCalendar(calendarEventId: Int) async {
        let url = baseUrl.appendingPathComponent("calendar_events/\(calendarEventId)")
        let viewModel = CalendarEventViewModel(url: url)

        await viewModel.fetchCalendarEvent()

        guard let calendarEvent = viewModel.calendarEvent else {
            print("Error: Could not fetch calendar event")
            return
        }

        let event = EKEvent(eventStore: CalendarEventStore.shared)
        event.calendar = CalendarEventStore.shared.defaultCalendarForNewEvents
        event.title = calendarEvent.name
        event.location = calendarEvent.location
        event.startDate = calendarEvent.startsAt
        event.endDate = calendarEvent.endsAt

        await MainActor.run {
            let eventViewController = AddToCalendarEKEventEditViewController()
            eventViewController.calendarEventId = calendarEventId
            eventViewController.event = event
            eventViewController.eventStore = CalendarEventStore.shared
            eventViewController.editViewDelegate = self

            window?.rootViewController?.present(eventViewController, animated: true)
        }
    }
}
```

AddToCalendarController.swift

Dismiss Event View

- Event gets added to their calendar
- Dismiss event view
- Show success toast message

```swift
extension AddToCalendarController: EKEventEditViewDelegate {
    func eventEditViewController(
        _ controller: EKEventEditViewController,
        didCompleteWith action: EKEventEditViewAction
    ) {
        controller.dismiss(animated: true)

        switch action {
        case .saved:
            if let rootViewController = window?.rootViewController {
                ToastController(rootViewController).showToast("Event added!")
            }
            print("User saved the event to their calendar")
        default:
            print("User did not save the event to their calendar")
        }
    }
}
```

AddToCalendarController.swift

Configure URL to use native component

```ruby
class ConfigurationsController < ApplicationController
  def ios_v1
    render json: {
      rules: [
        {
          patterns: [
            "/calendar_events/[0-9]+/add_to_calendar"
          ],
          properties: {
            view_controller: "add_to_calendar"
          }
        }
      ]
    }
  end
end
```

app/controllers/configurations_controller.rb

Add link to open native component

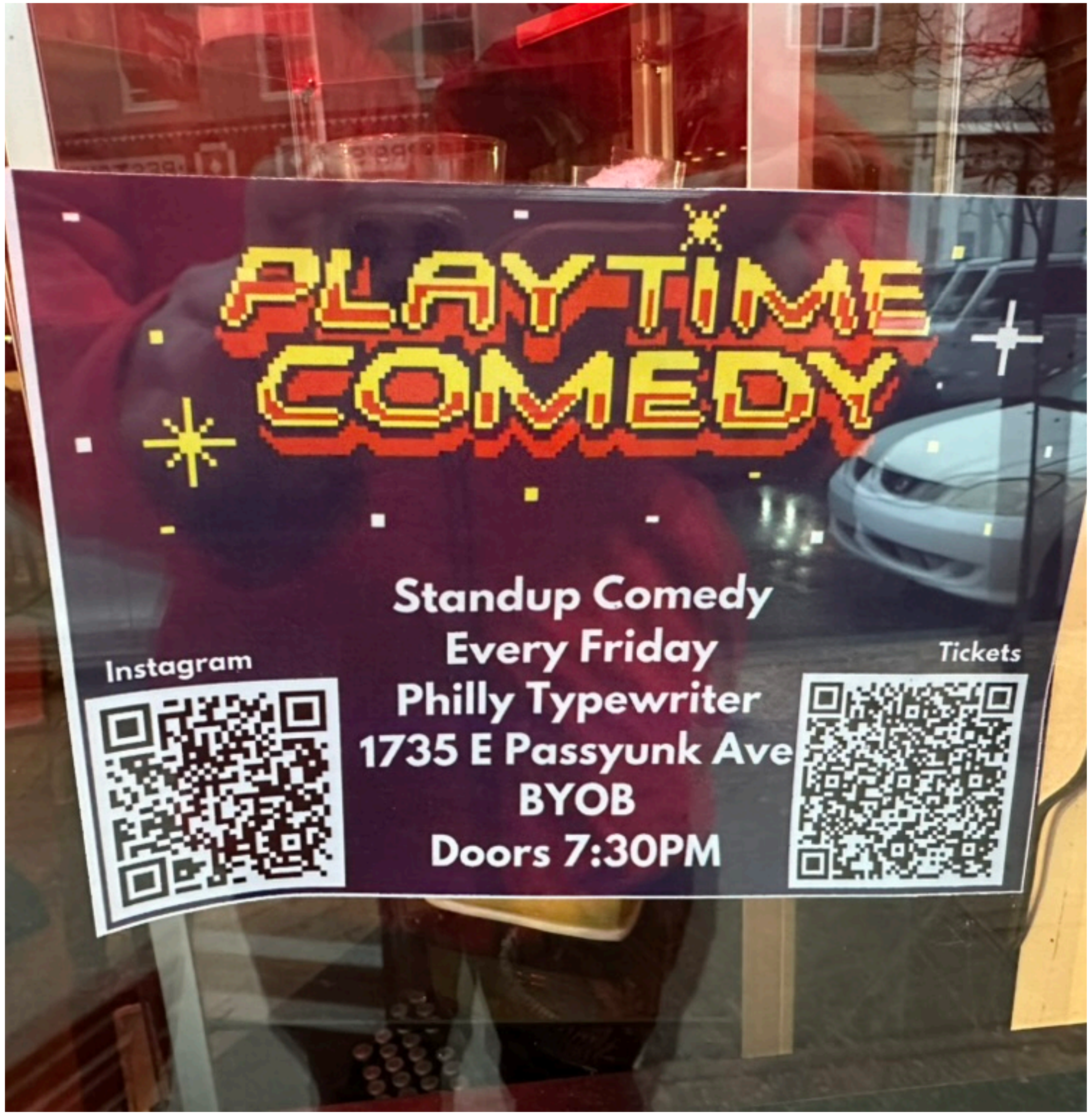
```erb
<%= link_to “/calendar_events/#{calendar_event.id}/add_to_calendar" do %>
  Add to Calendar
<% end %>
```

_add_to_calendar.html.erb

Extracting events from images

Extracting events from images

- User uploads an image (photo or screenshot)
- Prompt the LLM with the image and a question about the image
- Ask the LLM to give an answer with a JSON schema we define



PLAYTIME COMEDY

Standup Comedy
Every Friday
Philly Typewriter
1735 E Passyunk Ave
BYOB
Doors 7:30PM

Instagram

Tickets

Configure RubyLLM

```ruby
RubyLLM.configure do |config|
  config.openai_api_key = Rails.application.credentials.dig(:openai, :access_token)
  config.default_model = "gpt-5-mini"
end
```

config/initializers/ruby_llm.rb

```ruby
class Llm::RubyLlm::Image
  def initialize(image, active_support_time_zone:, today: nil)
    @image =image
    @active_support_time_zone = active_support_time_zone
    @today = today || ActiveSupport::TimeZone[@active_support_time_zone].now
  end

  def events
    message.content["events"]
  end

  private

  def message
    @message ||= RubyLLM.chat
      .with_schema(Llm::CalendarEventsSchema.schema)
      .ask(prompt, with: @image)
  end

  def formatted_today
    @today.strftime("%A, %B %d, %Y")
  end

  def prompt
    "What events are described in this image? Today's date is #{formatted_today}."
  end
end
```

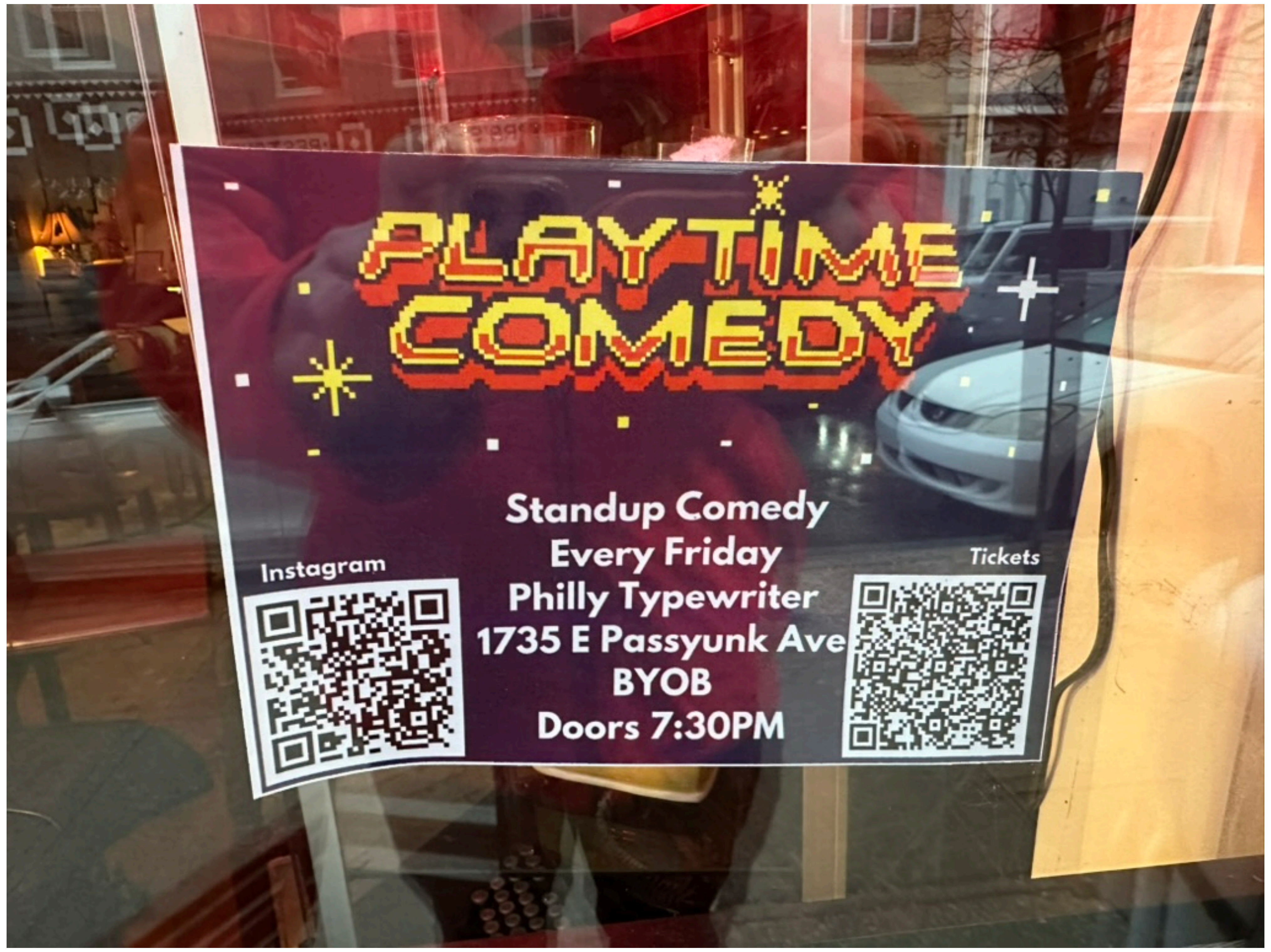
app/models/llm/ruby_llm/image.rb

```ruby
module Llm::CalendarEventsSchema
  extend T::Sig

  sig { returns(T::Hash[Symbol, T.untyped]) }
  def self.schema
    {
      type: "object",
      properties: {
        "events": {
          "type": "array",
          "description": "",
          "items": {
            type: "object",
            properties: schema_properties,
            required: schema_properties.keys,
            additionalProperties: false
          }
        }
      },
      required: [ "events" ],
      additionalProperties: false
    }
  end

  ...
end
```

```ruby
module Llm::CalendarEventsSchema
  ...

  sig { returns(T::Hash[Symbol, T.untyped]) }
  def self.schema_properties
    {
      "event_name": {
        "type": "string",
        "description": "The name of the event."
      },
      "event_location": {
        "type": [ "string", "null" ],
        "description": "The location of the event."
      },
      "event_description": {
        "type": [ "string", "null" ],
        "description": "The description of the event."
      },
      "event_date_starts_at": {
        "type": "string",
        "format": "date",
        "description": "The start date of the event."
      },
      "event_date_ends_at": {
        "type": [ "string", "null" ],
        "format": "date",
        "description": "The end date of the event."
      },
      "event_time_starts_at": {
        "type": [ "string", "null" ],
        "format": "time",
        "description": "The start time of the event in 24-hour format."
      },
      "event_time_ends_at": {
        "type": [ "string", "null" ],
        "format": "time",
        "description": "The end time of the event in 24-hour format.”
      }
    }
  end
end
```

```ruby
module Llm::CalendarEventsSchema

  sig { returns(T::Hash[Symbol, T.untyped]) }
  def self.schema_properties
    {
      "event_is_recurring": {
        "type": "boolean",
        "description": "If the event occurs multiple days, this is true. If there is a single event that spans multiple days, this is false."
      },
      "event_recurrence_frequency": {
        "type": [ "string", "null" ],
        "enum": CalendarEvent.recurrence_frequencies.keys,
        "description": "If the event is recurring, this is how often the event recurs. If the event is not recurring, set this to null"
      },
      "event_date_recurrence_starts_at": {
        "type": [ "string", "null" ],
        "format": "date",
        "description": "If the event is recurring, this is the first date of the recurring event."
      },
      "event_date_recurrence_ends_at": {
        "type": [ "string", "null" ],
        "format": "date",
        "description": "If the event is recurring, this is the last date of the recurring event."
      },
      "event_recurrence_days_of_the_week": {
        "type": [ "array", "null" ],
        "items": {
          "type": "string",
          "enum": CalendarEvent::DAYS_OF_THE_WEEK,
          "description": "The specific day of the week the recurring event occurs."
        }
      },
      "event_recurrence_weeks_of_the_month": {
        "type": [ "array", "null" ],
        "items": {
          "type": "string",
          "enum": CalendarEvent::WEEKS_OF_THE_MONTH,
          "description": "If the event occurs each month and on specific weeks of the month, these are those weeks."
        }
      }
    }
  end
end
```

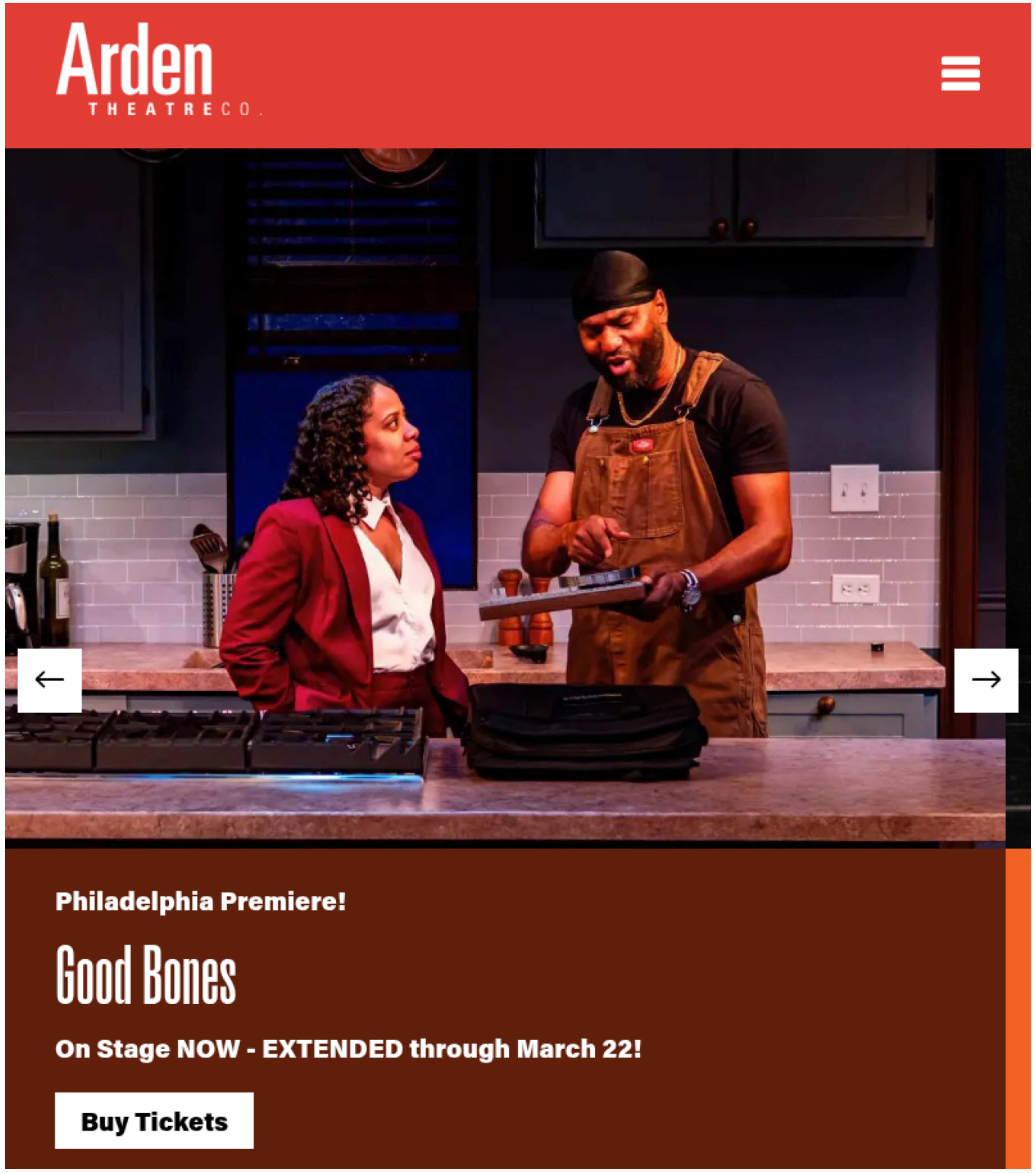
```
> Llm::RubyLlm::Image.new(ImageUpload.last.image, active_support_time_zone: Time.zone.name).events
=>
  [{"event_name" => "Playtime Comedy (Standup Comedy)",
    "event_location" => "Philly Typewriter – 1735 E Passyunk Ave",
    "event_description" =>
      "Weekly stand-up comedy show (BYOB)...",
    "event_date_starts_at" => "2026-02-27",
    "event_date_ends_at" => nil,
    "event_time_starts_at" => "19:30:00Z",
    "event_time_ends_at" => "21:30:00Z",
    "event_is_recurring" => true,
    "event_recurrence_frequency" => "weekly",
    "event_date_recurrence_starts_at" => "2026-02-27",
    "event_date_recurrence_ends_at" => nil,
    "event_recurrence_days_of_the_week" => ["friday"],
    "event_recurrence_weeks_of_the_month" => nil}]
```



Extracting events from websites

Extracting events from websites

- User provides a URL
- Take a screenshot with Selenium
- Prompt the LLM with the screenshot and a question about the screenshot
- Ask the LLM to give an answer with a JSON schema we define



Arden THEATRE CO.

Philadelphia Premiere!

Good Bones

On Stage NOW - EXTENDED through March 22!

Buy Tickets

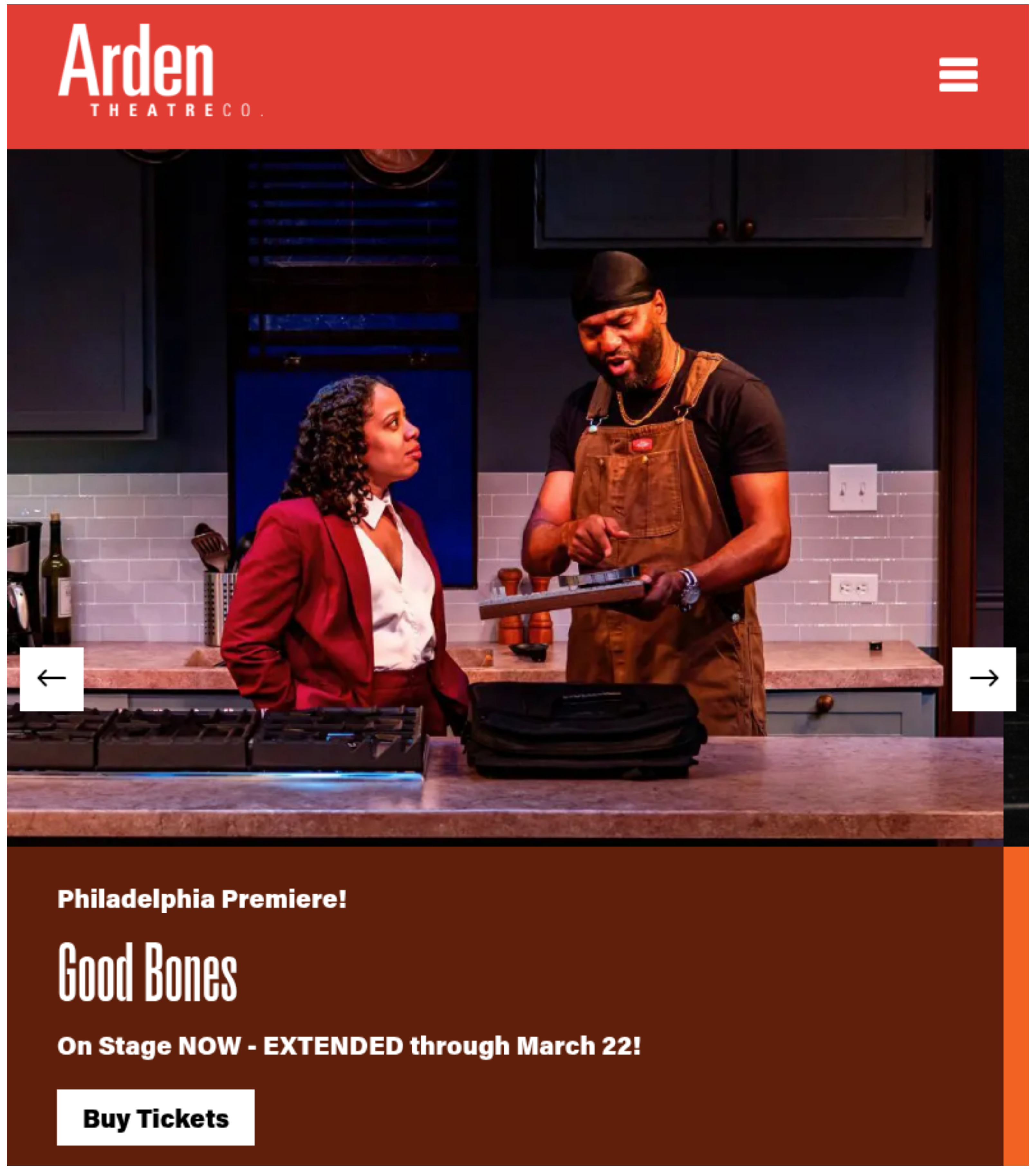
```ruby
screenshot = Screenshot.new(T.must(web_page.url))
screenshot.take do |file|
  web_page.screenshot.attach(
    io: file,
    filename: "#{web_page.id}-screenshot.png"
  )
end
```

```ruby
class Screenshot
  def initialize(url)
    @url = url
  end

  def take(&block)
    options = Selenium::WebDriver::Options.chrome(
      args: [
        "--headless=new",
        "--no-sandbox",
        "--disable-dev-shm-usage",
        "--disable-gpu",
        "--hide-scrollbars"
      ]
    )
    driver = Selenium::WebDriver.for(:chrome, options: options)

    driver.navigate.to @url

    with_temporary_file do |file|
      driver.save_screenshot(file.path, full_page: false)

      block.call(file)
    end

    driver.quit

    nil
  end
end
```

```ruby
class Llm::RubyLlm::WebPage
  def initialize(image, active_support_time_zone:)
    @image = image
    @active_support_time_zone = active_support_time_zone
  end

  def events
    message.content["events"]
  end

  private

 def message
    @message ||= RubyLLM.chat
      .with_schema(Llm::CalendarEventsSchema.schema)
      .ask(prompt, with: @image)
  end

 def prompt
    "This is an image of a web page. What events are described in this image? Today's date is #{formatted_today}."
  end

 def formatted_today
    ActiveSupport::TimeZone[@active_support_time_zone].now.strftime("%A, %B %d, %Y")
  end
end
```

app/models/llm/ruby_llm/web_page.rb

```
> Llm::RubyLlm::WebPage.new(WebPage.last.screenshot, active_support_time_zone: Time.zone.name).events
=>
  [{"event_name" => "Good Bones",
    "event_location" => "Arden Theatre Company — 40 N. 2nd Street, Philadelphia, PA 19106",
    "event_description" => "Philadelphia Premiere — On stage NOW. Run extended through March 22.",
    "event_date_starts_at" => "2026-02-25",
    "event_date_ends_at" => "2026-03-22",
    "event_time_starts_at" => nil,
    "event_time_ends_at" => nil,
    "event_is_recurring" => false,
    "event_recurrence_frequency" => "monthly",
    "event_date_recurrence_starts_at" => nil,
    "event_date_recurrence_ends_at" => nil,
    "event_recurrence_days_of_the_week" => nil,
    "event_recurrence_weeks_of_the_month" => nil}]
```



Philadelphia Premiere!

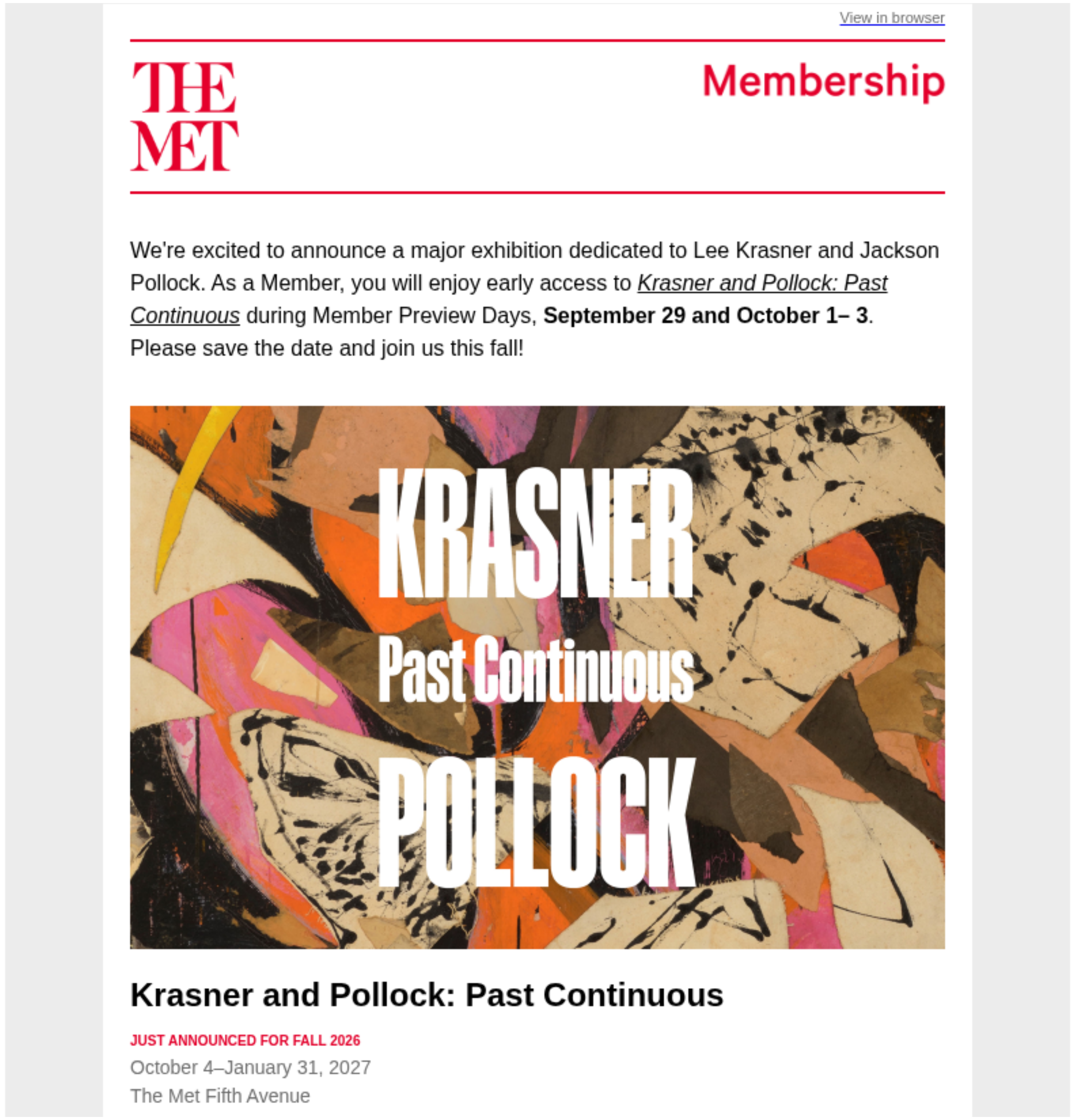
Good Bones

On Stage NOW - EXTENDED through March 22!

Extracting events from emails

Extracting events from emails

- Receive email from user using Postmark and Action Mailbox
- Take a screenshot using Selenium
- Prompt the LLM with the image and a question about the image
- Ask the LLM to give an answer with a JSON schema we define



View in browser

THE MET

Membership

We're excited to announce a major exhibition dedicated to Lee Krasner and Jackson Pollock. As a Member, you will enjoy early access to *Krasner and Pollock: Past Continuous* during Member Preview Days, **September 29 and October 1– 3**. Please save the date and join us this fall!

Krasner and Pollock: Past Continuous

JUST ANNOUNCED FOR FALL 2026

October 4–January 31, 2027

The Met Fifth Avenue

```ruby
class InboundEmailScreenshot
  def initialize(inbound_email)
    @inbound_email = T.let(inbound_email, InboundEmail)
  end

  def take(&block)
    Screenshot.new(internal_url).take do |file|
      block.call(file)
    end
  end

  private

  sig { returns(String) }
  def internal_url
    username = Rails.application.credentials.inbound_email_preview.username
    password = Rails.application.credentials.inbound_email_preview.password
    host_part "http://#{username}:#{password}@#{Rails.configuration.app_hostname}"
    path_part "/inbound_email_previews/#{@inbound_email.id}"
    "#{host_part}#{path_part}"
  end
end
```

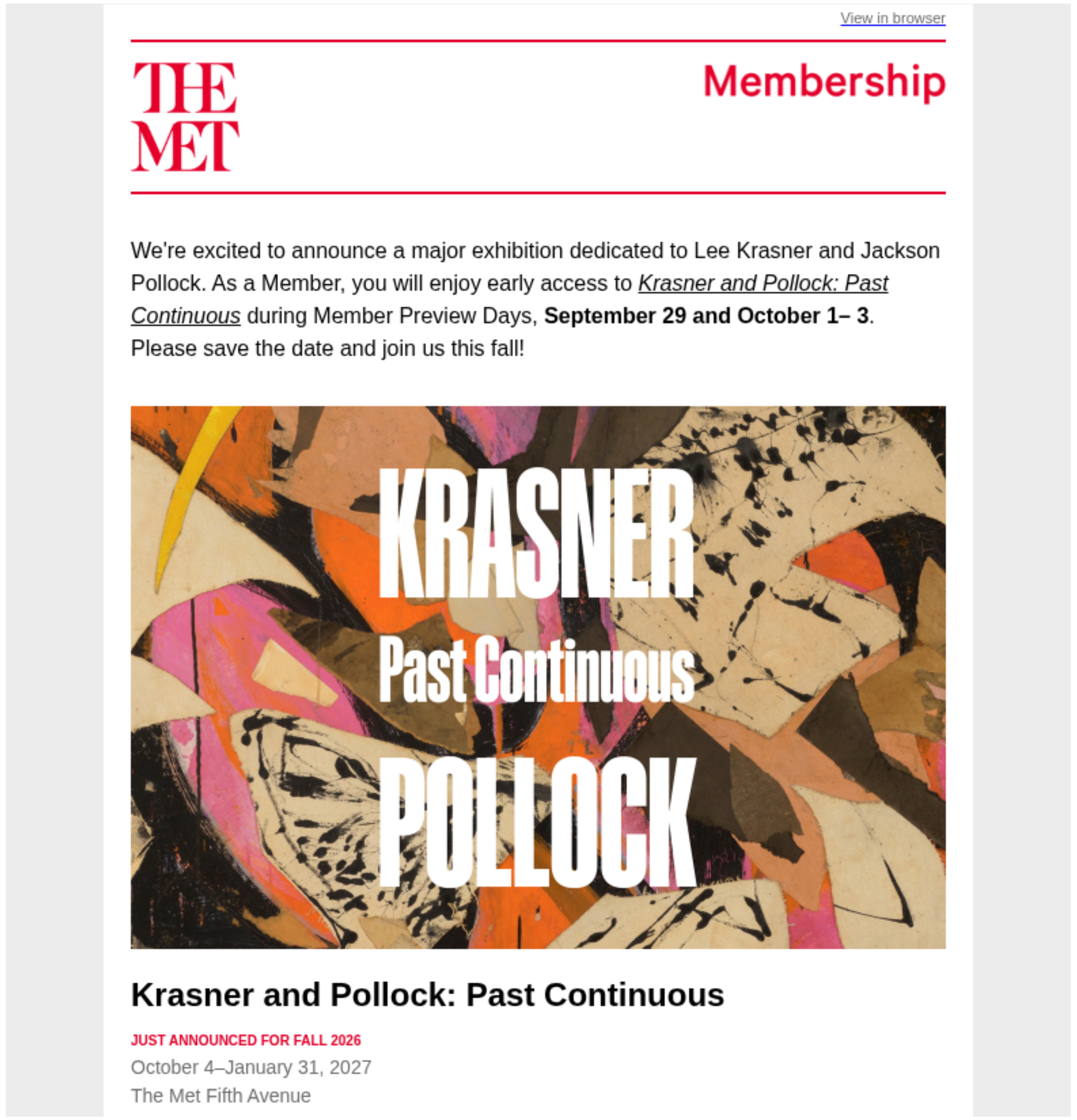
```ruby
class InboundEmailPreviewsController < ApplicationController
  http_basic_authenticate_with(
    name: Rails.application.credentials.inbound_email_preview.username,
    password: Rails.application.credentials.inbound_email_preview.password,
  )

  content_security_policy do |policy|
    policy.style_src :self, :https, :unsafe_inline
  end

  layout false

  def show
    @inbound_email = InboundEmail.find(params[:id])
  end
end
```

```erb
<!DOCTYPE html>
<html>
<head>
  <title>Inbound Email Preview</title>
  <meta name="rails-env" content="<%= Rails.env %>">
  <%= csp_meta_tag %>
  <%= stylesheet_link_tag "application", "data-turbo-track": "reload" %>
</head>
<body>
  <div>
    <% if @inbound_email.unsanitized_original_html.present? %>
      <%== @inbound_email.unsanitized_original_html %>
    <% else %>
      <div class="whitespace-pre-wrap">
        <%= @inbound_email.text %>
      </div>
    <% end %>
  </div>
</body>
</html>
```

```ruby
class Llm::RubyLlm::EmailImage
  def initialize(image, active_support_time_zone:)
    @image = image
    @active_support_time_zone = active_support_time_zone
  end

  def events
    message.content["events"]
  end

  private

  def message
    @message ||= RubyLLM.chat
      .with_schema(Llm::CalendarEventsSchema.schema)
      .ask(prompt, with: @image)
  end

  def prompt
    "This is an image of an email. What events are described in this image? Today's date is #{formatted_today}."
  end

  def formatted_today
    ActiveSupport::TimeZone[@active_support_time_zone].now.strftime("%A, %B %d, %Y")
  end
end
```

app/models/llm/ruby_llm/email.rb

```
> Llm::RubyLlm::EmailImage.new(InboundEmail.last.image, active_support_time_zone: Time.zone.name).events
=>
[{"event_name" => "Krasner and Pollock: Past Continuous",
  "event_location" => "The Met Fifth Avenue",
  "event_description" =>
   "Major exhibition devoted to Lee Krasner and Jackson Pollock (first major NYC presentation in over 20 years).",
  "event_date_starts_at" => "2026-10-04",
  "event_date_ends_at" => "2027-01-31",
  "event_time_starts_at" => nil,
  "event_time_ends_at" => nil,
  "event_is_recurring" => false,
  "event_recurrence_frequency" => "monthly",
  "event_date_recurrence_starts_at" => nil,
  "event_date_recurrence_ends_at" => nil,
  "event_recurrence_days_of_the_week" => nil,
  "event_recurrence_weeks_of_the_month" => nil},
 {"event_name" => "Member Preview Days for Krasner and Pollock (Members
  "event_location" => "The Met Fifth Avenue",
  "event_description" => "Early access member preview days for the Kras
  "event_date_starts_at" => "2026-09-29",
  "event_date_ends_at" => "2026-10-03",
  "event_time_starts_at" => nil,
  "event_time_ends_at" => nil,
  "event_is_recurring" => true,
  "event_recurrence_frequency" => "weekly",
  "event_date_recurrence_starts_at" => nil,
  "event_date_recurrence_ends_at" => nil,
  "event_recurrence_days_of_the_week" => nil,
  "event_recurrence_weeks_of_the_month" => nil}]
```



View in browser

THE MET

Membership

We're excited to announce a major exhibition dedicated to Lee Krasner and Jackson Pollock. As a Member, you will enjoy early access to *Krasner and Pollock: Past Continuous* during Member Preview Days, **September 29 and October 1– 3**. Please save the date and join us this fall!

Krasner and Pollock: Past Continuous

JUST ANNOUNCED FOR FALL 2026

October 4–January 31, 2027

The Met Fifth Avenue

What’s next?

- Get more people using it
- Show the user where the event is in the image
- Find events that users will be interested in

Let’s talk!

- Love my app?
- Hate my app?
- Think “I can do this myself with Open Claw”
- Lets talk!
- https://mikedalton.co/socials/
- https://calendarvision.app/