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

Who am I?

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

my social media pic

Calendar Vision

Sign in

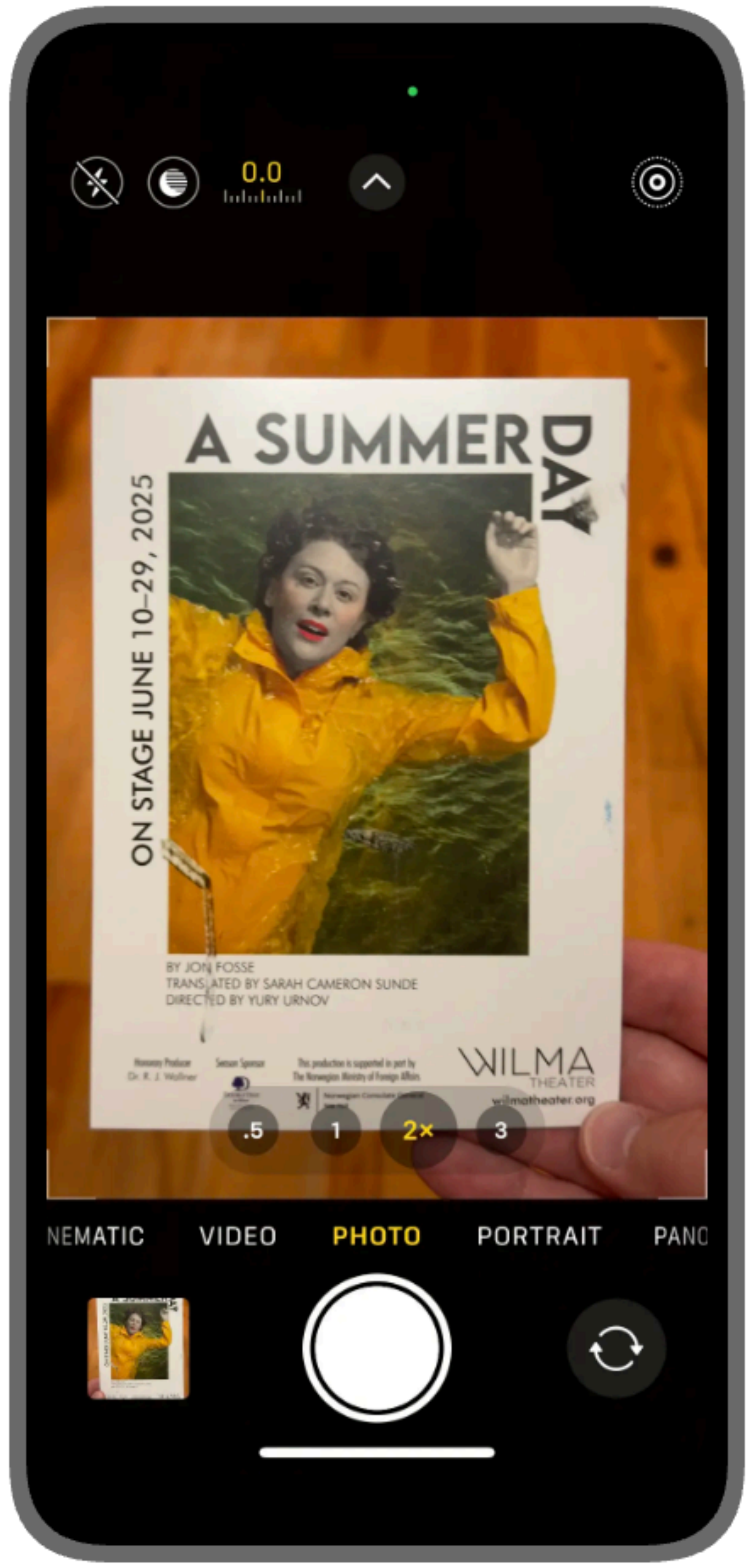
Sign up

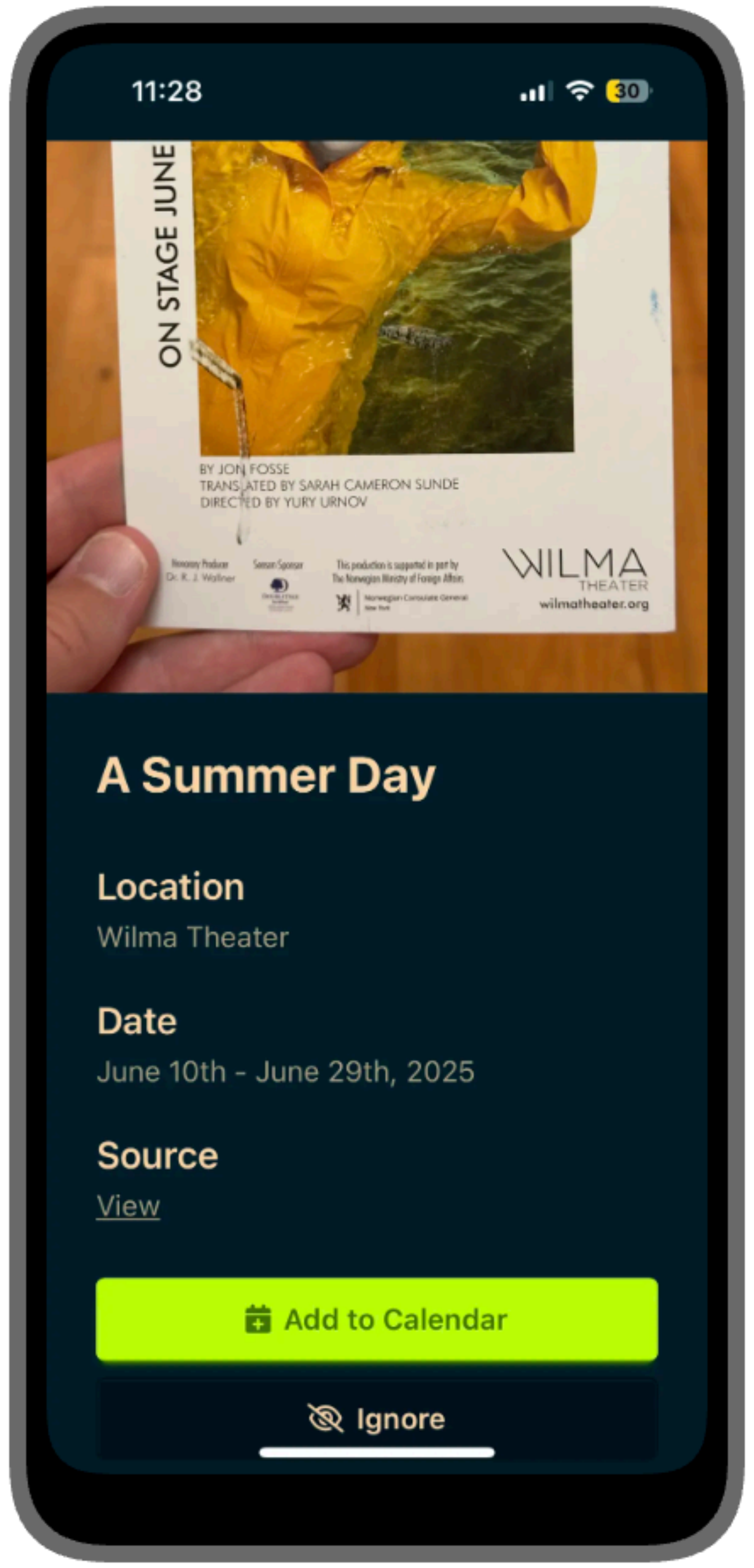
Turn images into events in your calendar

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

Download on the App Store

GET IT ON Google Play



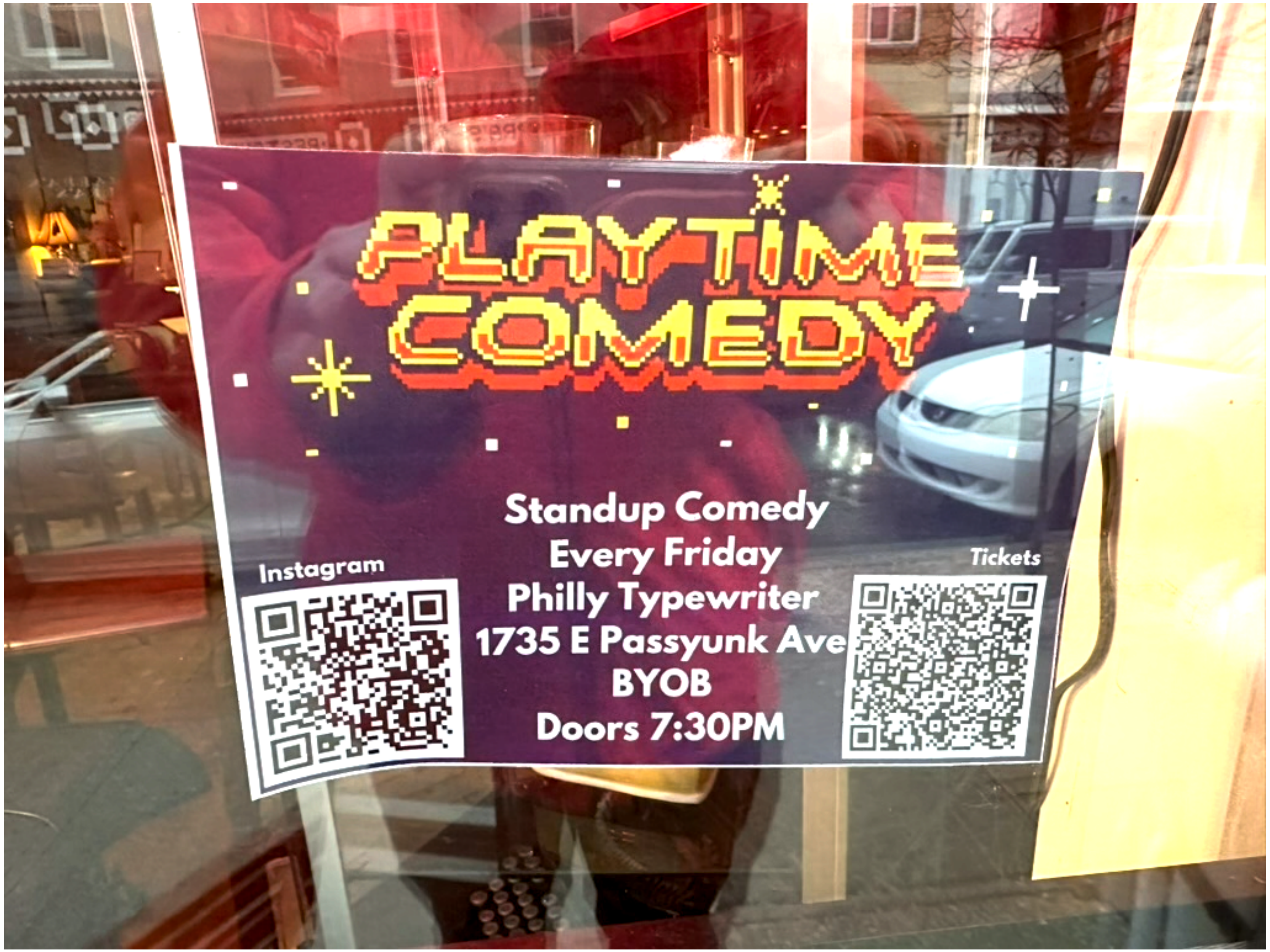


A Summer Day

Location
Wilma Theater

Date
June 10th - June 29th, 2025

Source
View



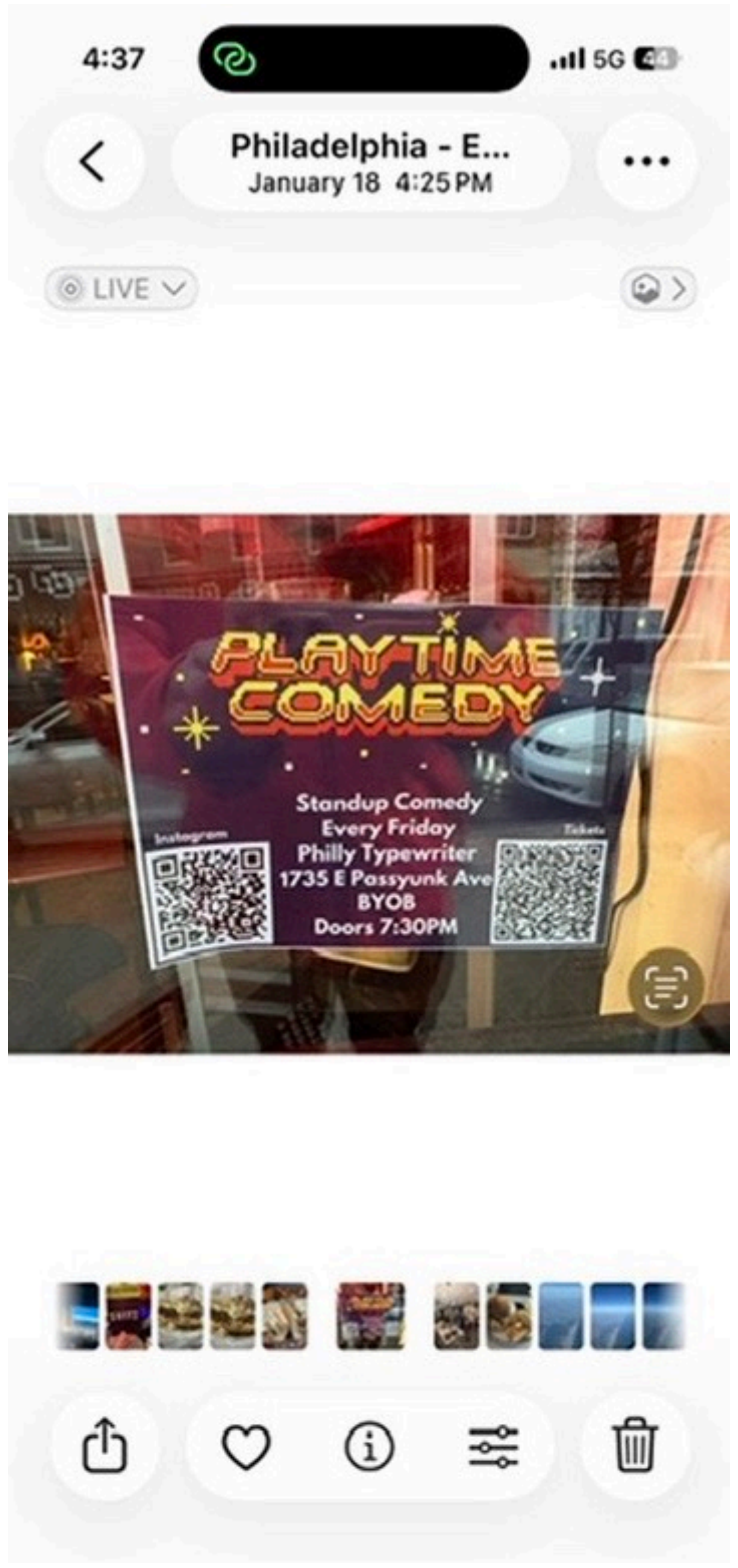
PLAYTIME COMEDY

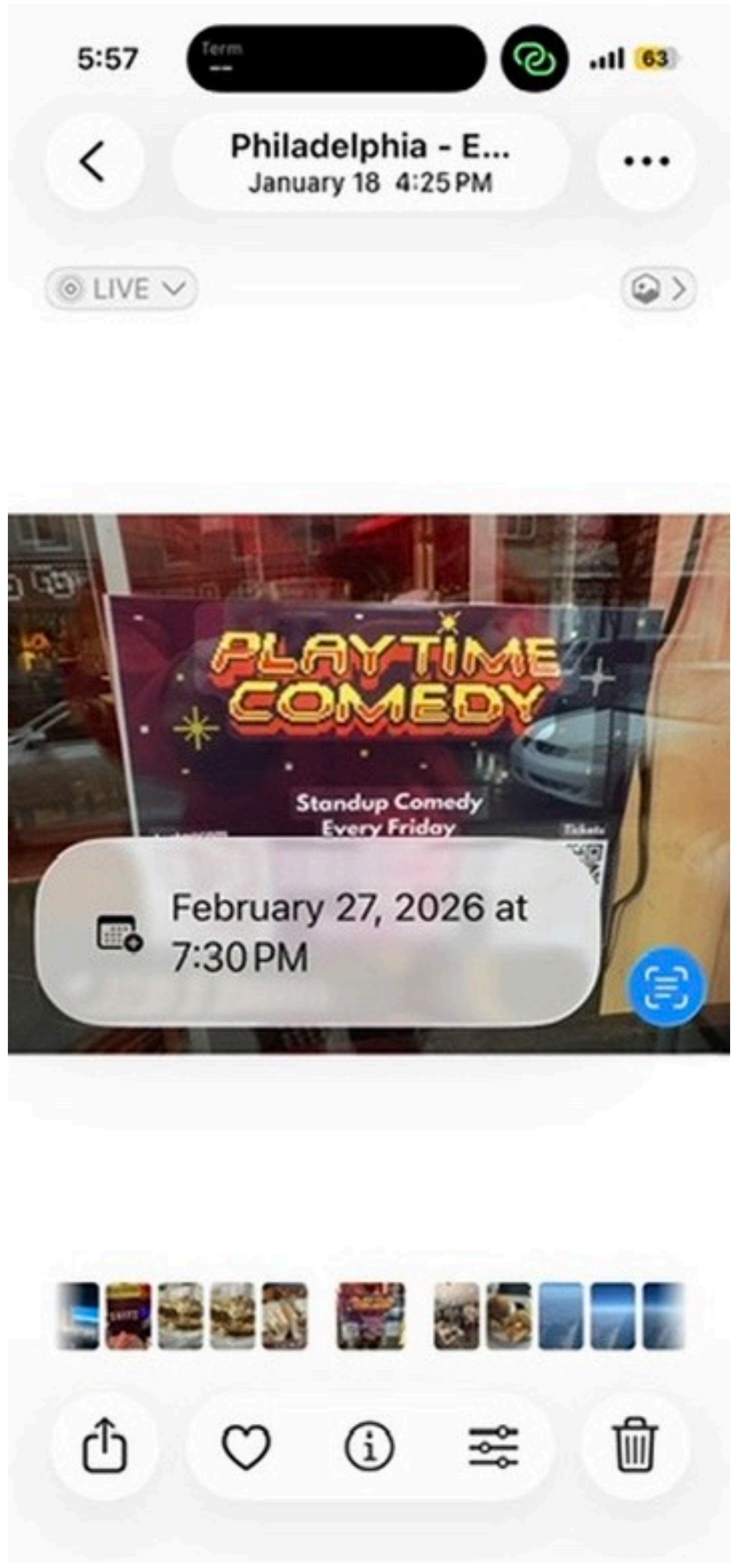
Standup Comedy
Every Friday
Philly Typewriter
1735 E Passyunk Ave
BYOB
Doors 7:30PM

Instagram

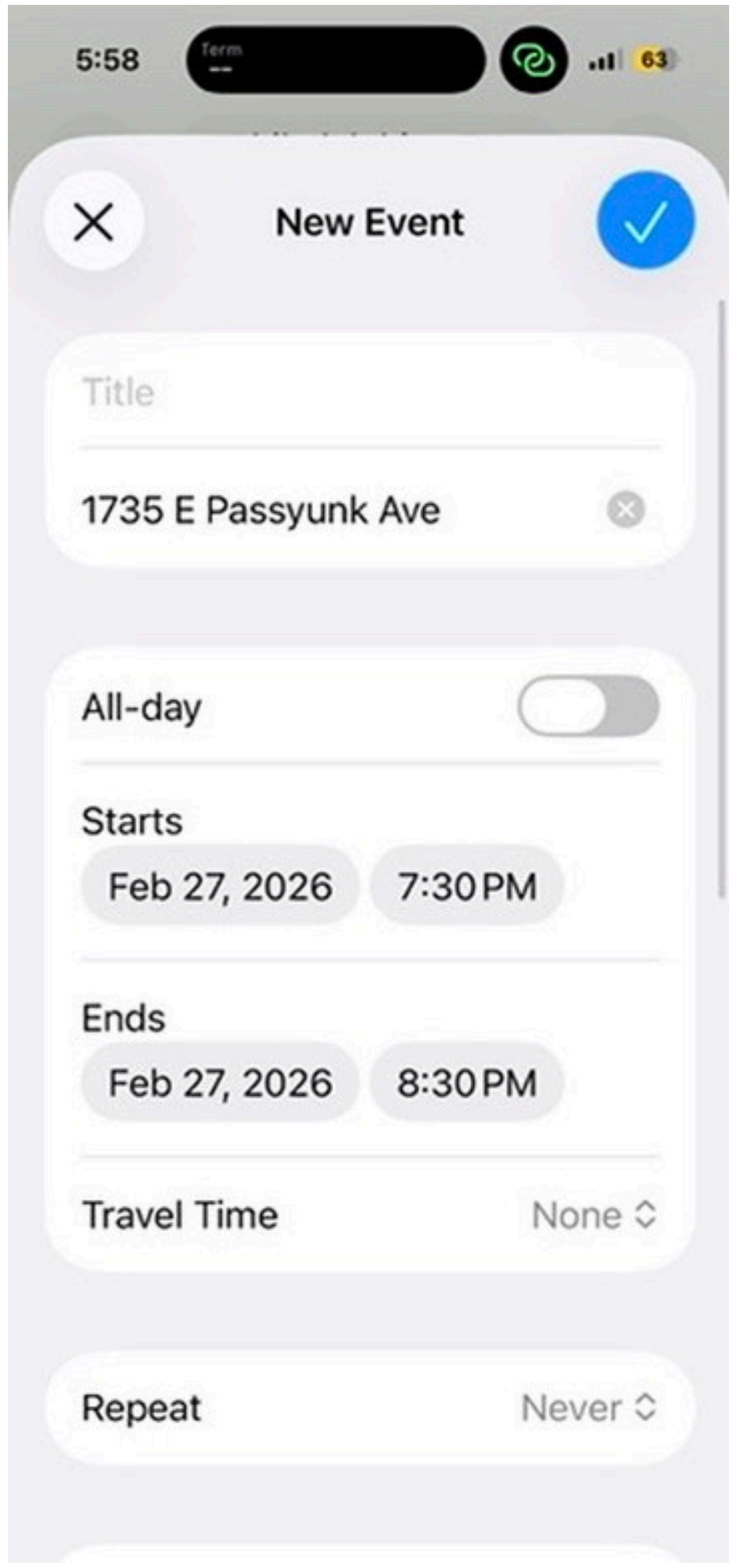
Tickets

iOS Photos App





February 27, 2026 at 7:30 PM



Title

1735 E Passyunk Ave

All-day

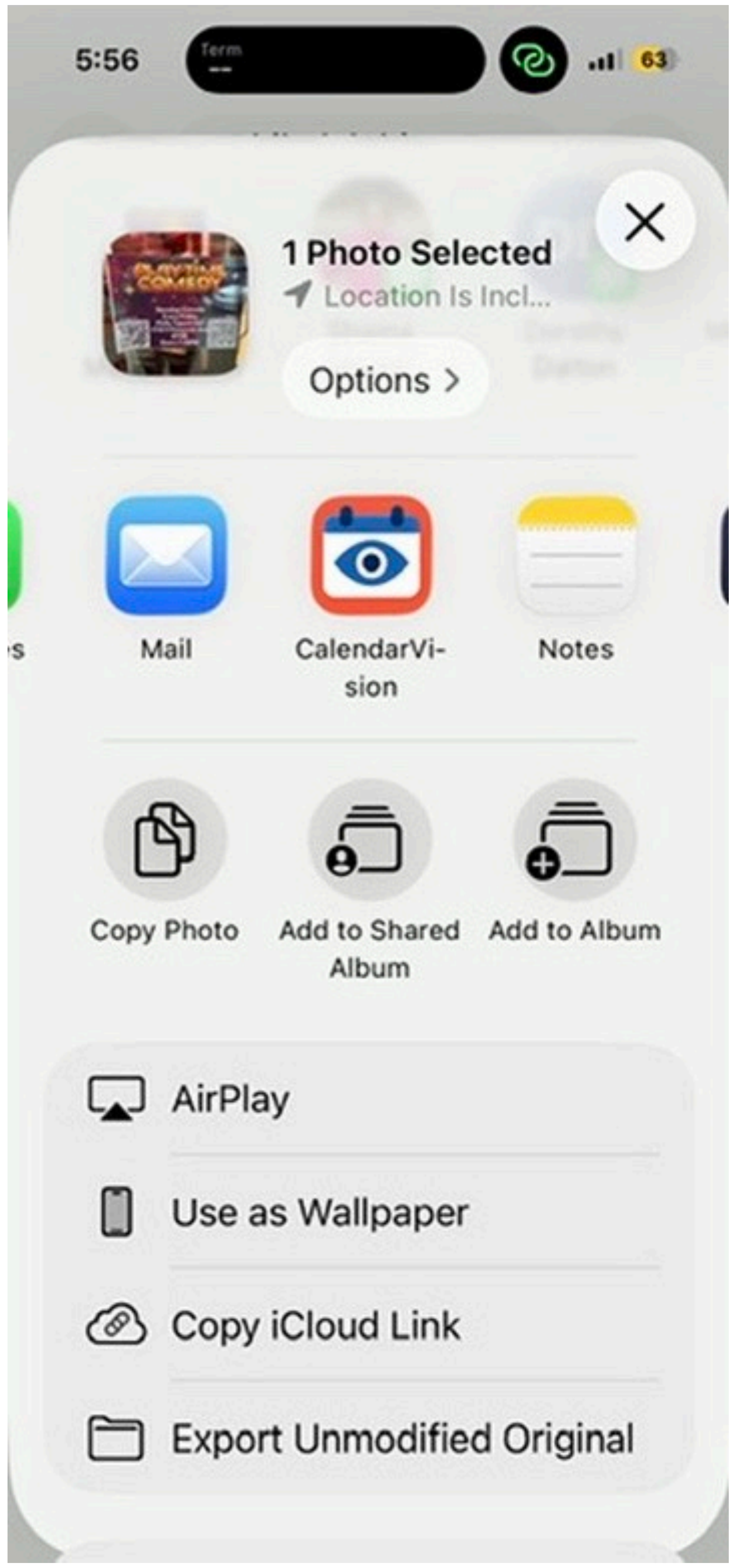
Starts Feb 27, 2026 7:30 PM

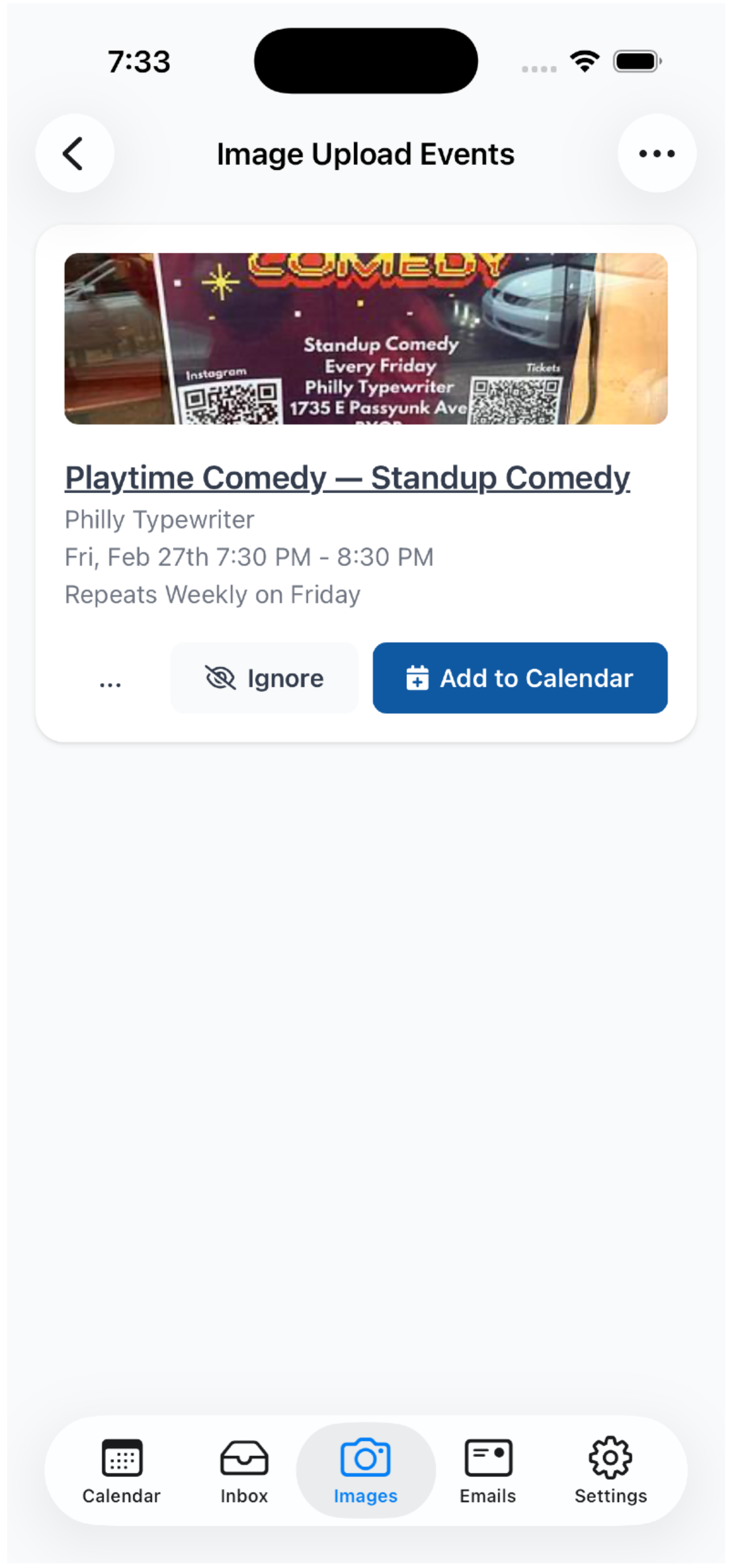
Ends Feb 27, 2026 8:30 PM

Travel Time None

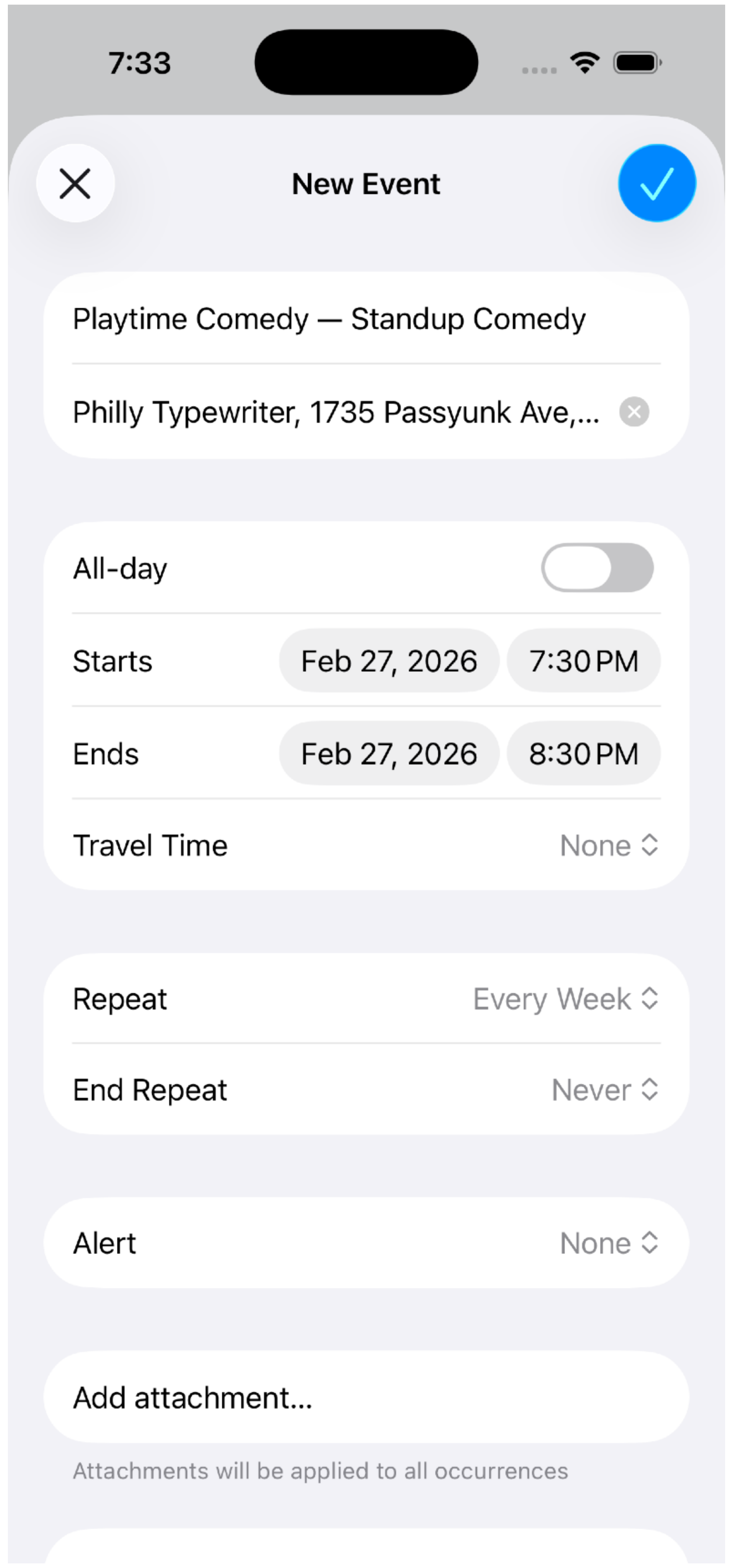
Repeat Never

Calendar Vision





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



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

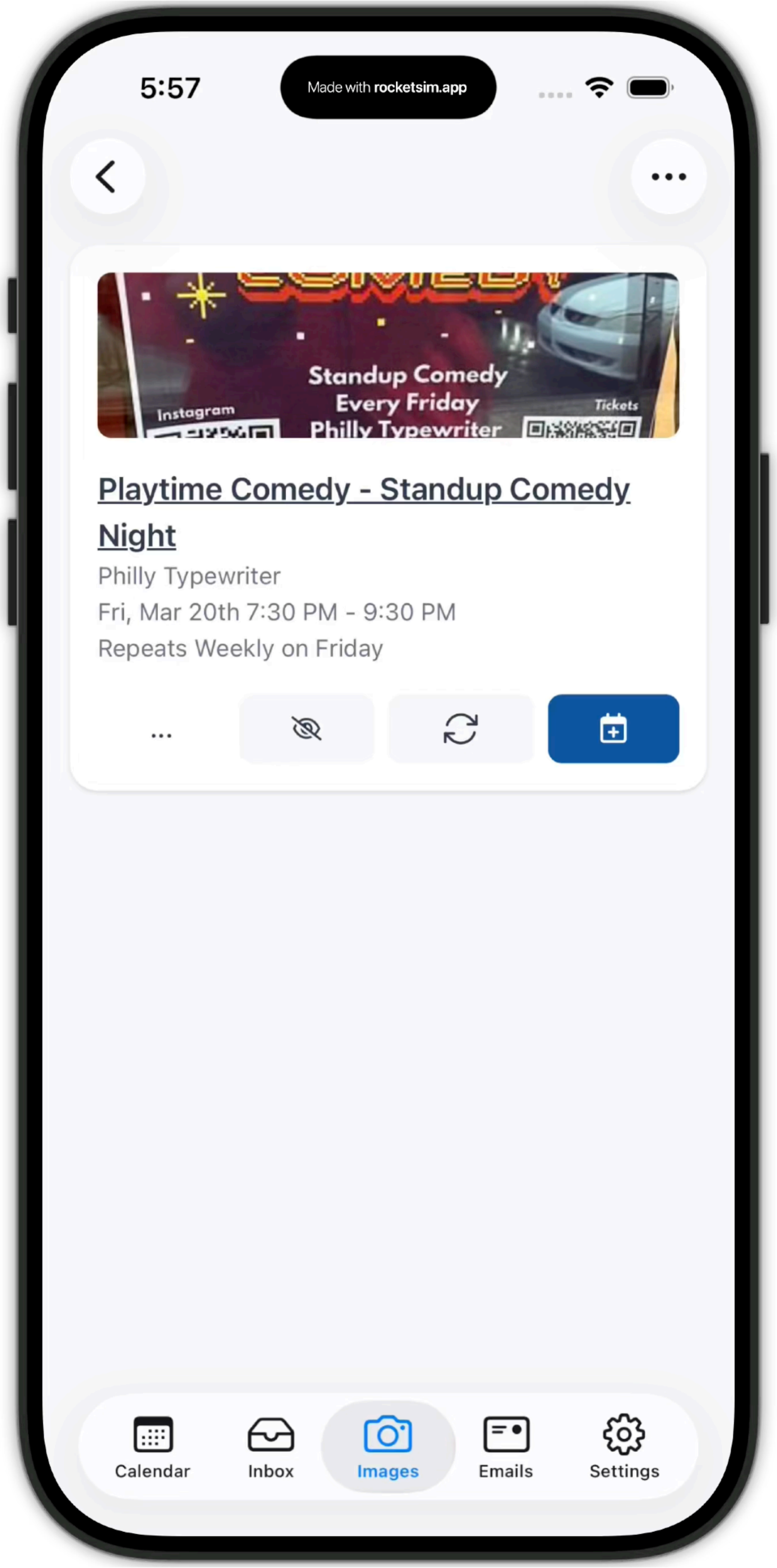
- Alternative native iOS, Kotlin and React Native
- Build native apps with Rails
- Re-use most of your Rails app
- Native navigation with Rails views in web view

Hotwire Native

Bridge Components

- Add native behavior
- Triggered with a Stimulus controller
- Requires custom Swift and Kotlin code

Add to Calendar


5:57

Made with rocketsim.app

Playtime Comedy - Standup Comedy Night

Philly Typewriter

Fri, Mar 20th 7:30 PM - 9:30 PM

Repeats Weekly on Friday

Add to Calendar

Rails

```javascript
import { BridgeComponent } from "@hotwired/hotwire-native-bridge"

export default class extends BridgeComponent {
  static component = "add-to-calendar"
  static values = {
    eventPayload: Object,
  }

  add() {
    this.send("add", this.eventPayloadValue)
  }
}
```

app/javascript/controllers/bridge/add_to_calendar_controller.js

Add to Calendar

Rails (continued)

```erb
<%# locals: (calendar_event:) %>
<% payload = calendar_event_json_payload(calendar_event) %>
<%= tag.div data: {
  controller: "bridge--add-to-calendar",
  bridge__add_to_calendar_event_payload_value: payload.to_json
} do %>
  <%= button_tag data: {
    action: "bridge--add-to-calendar#add:prevent",
  } do %>
    <i class="fa-regular fa-calendar-plus fa-lg"></i>
  <% end %>
<% end %>
```

app/views/calendar_events/_add_to_calendar.html.erb

Add to Calendar

iOS

```swift
final class AddToCalendarComponent: BridgeComponent {
    private var viewController: UIViewController? {
        delegate?.destination as? UIViewController
    }

    override func onReceive(message: Message) {
        guard message.event == "add",
              let calendarEvent: CalendarEvent = message.data() else {
            return
        }

        Task { await presentEventEditor(calendarEvent: calendarEvent, message: message) }
    }

    ...
}
```

AddToCalendarComponent.swift

Add to Calendar

iOS (continued)

```swift
final class AddToCalendarComponent: BridgeComponent {
    ...

    private func presentEventEditor(calendarEvent: CalendarEvent, message: Message) async {
        let event = EKEventBuilder.makeEvent(from: calendarEvent, eventStore: CalendarEventStore.shared)
        event.calendar = eventStore.defaultCalendarForNewEvents
        event.title = calendarEvent.name
        event.location = calendarEvent.location
        // ...

        await MainActor.run {
            let editDelegate = EventEditDelegate()
            let eventViewController = EKEventEditViewController()
            eventViewController.event = event
            eventViewController.eventStore = CalendarEventStore.shared
            eventViewController.editViewDelegate = editDelegate
            viewController?.present(eventViewController, animated: true)
        }
    }
}

private final class EventEditDelegate: NSObject, EKEventEditViewDelegate {
    func eventEditViewController(_ controller: EKEventEditViewController, didCompleteWith action: EKEventEditViewAction) {
        controller.dismiss(animated: true)
    }
}
```

AddToCalendarComponent.swift

Add to Calendar

Android

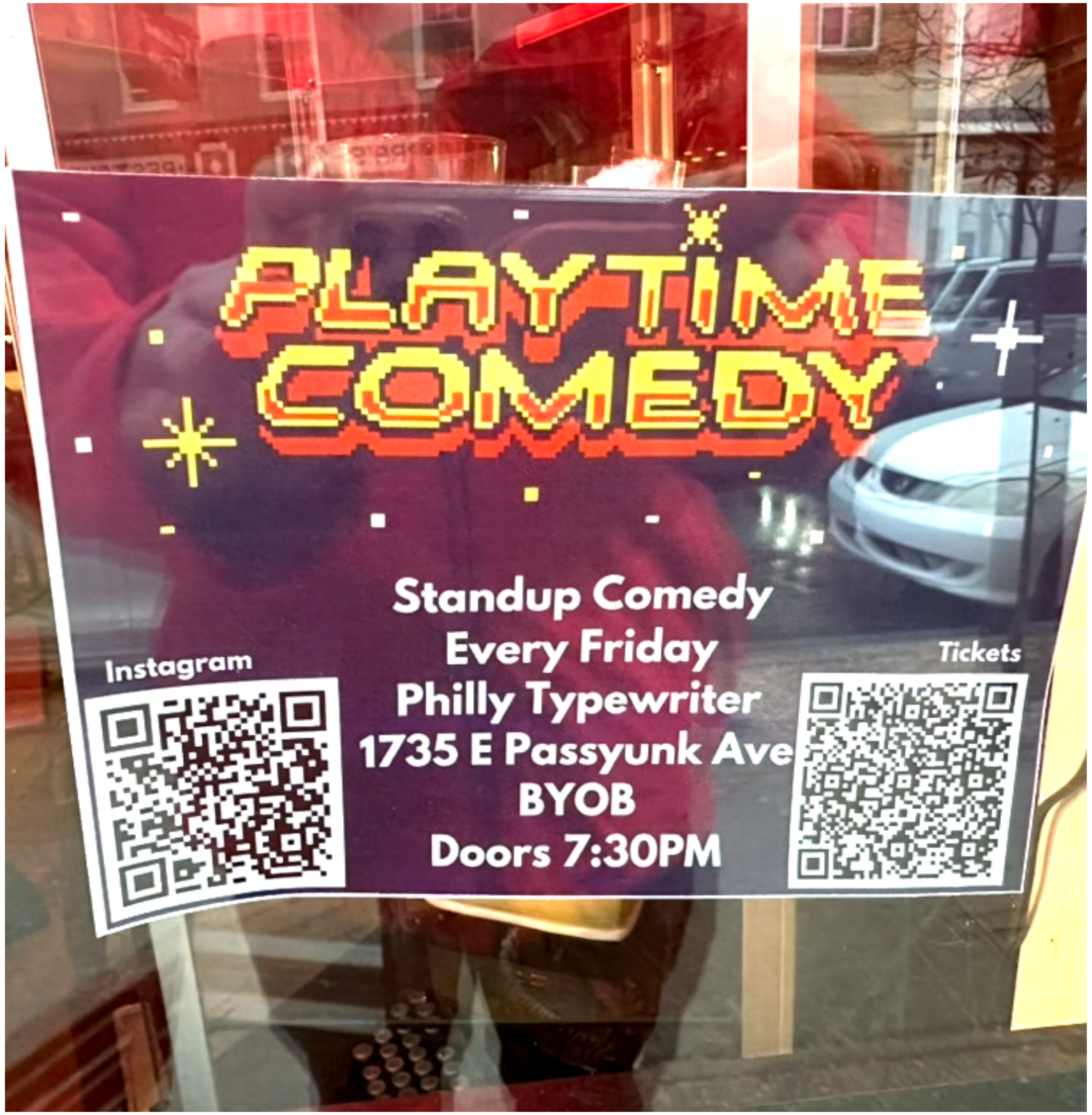
```kotlin
class AddToCalendarComponent(
    name: String,
    private val bridgeDelegate: BridgeDelegate<HotwireDestination>
) : BridgeComponent<HotwireDestination>(name, bridgeDelegate) {
    override fun onReceive(message: Message) {
        when (message.event) {
            "add" -> handleAddToCalendar(message)
        }
    }

    private fun handleAddToCalendar(message: Message) {
        val calendarEvent = message.data<CalendarEvent>()
        val activity = bridgeDelegate.destination.fragment.requireActivity()
        openCalendarEventInIntent(calendarEvent, activity)
    }

    private fun openCalendarEventInIntent(
        calendarEvent: CalendarEvent,
        activity: android.app.Activity
    ) {
        var intent = Intent(Intent.ACTION_INSERT)
            .setData(CalendarContract.Events.CONTENT_URI)
            .putExtra(CalendarContract.Events.TITLE, calendarEvent.name)
            .putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME, calendarEvent.startsAtInMillisecondsSinceEpoch)
            .putExtra(CalendarContract.EXTRA_EVENT_END_TIME, calendarEvent.endsAtInMillisecondsSinceEpoch)
        activity.startActivity(intent)
    }
}
```

AddToCalendarComponent.kt

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

Instagram

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

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
```

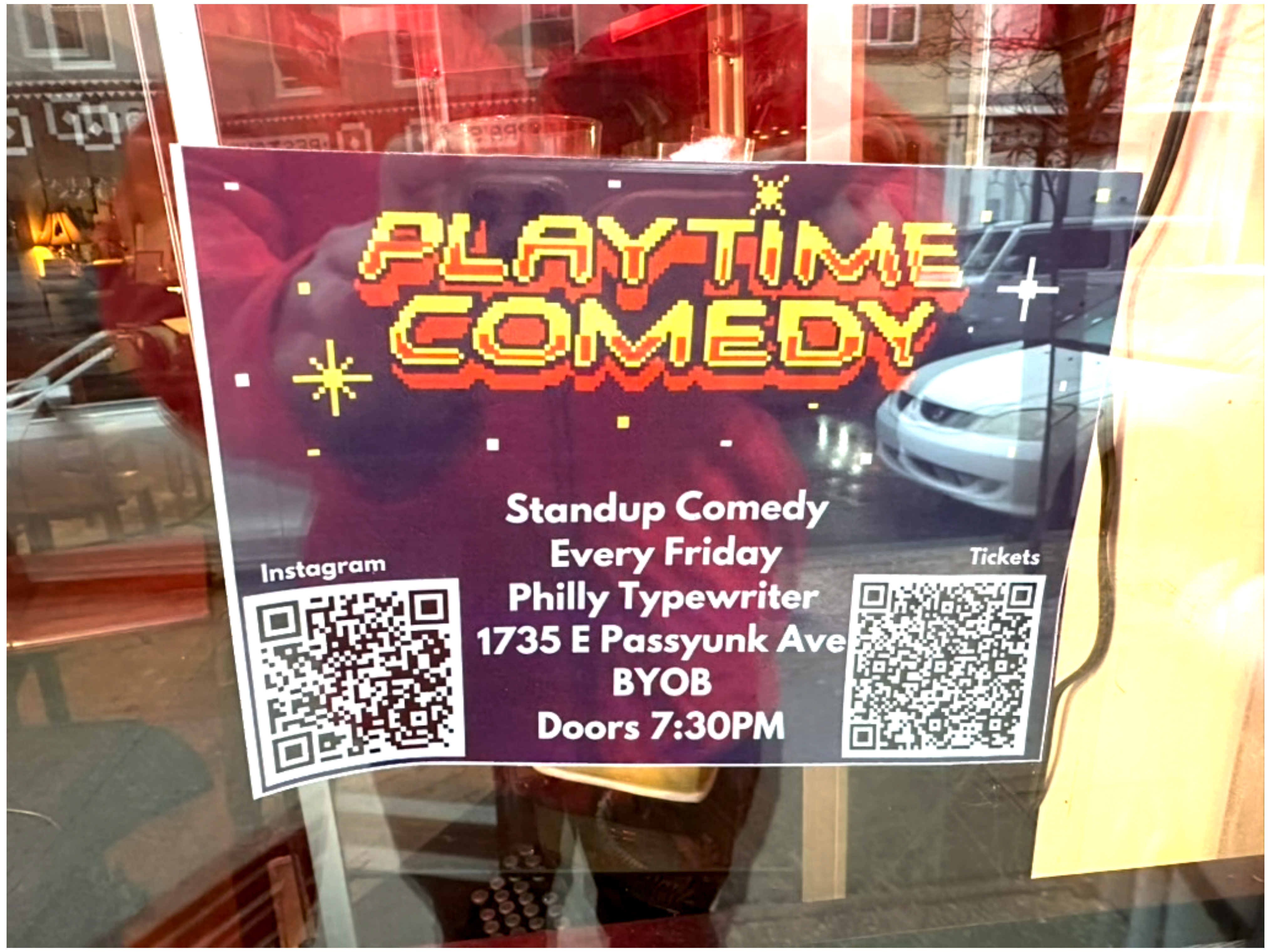
app/models/llm/ruby_llm/image.rb

```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}]
```



Thanks!

https://calendarvision.app/

https://mikedalton.co/socials/