Stories by Text: Building A Generalized System for Sending Message Sequences Across Popular Messaging Platforms

a CPSC 490 proposal

Abstract

I am working on an early literacy program called StoryTime to send illustrated children’s books by text message and messaging apps to families without books at home. It uses a curriculum intervention developed to improve learning outcomes for children ages 2-6 on the program. Parents may be sent digital books through many different platforms technologies: SMS, MMS, Facebook Messenger, iMessage, WhatsApp, and GroupMe, among others. The goal of this project is to create a generalizable system that can integrate these various messaging platforms into a single, simple interface to send digital stories.

Introduction

Research shows that text messaging programs for mothers of young children can translate into significant early literacy gains.\(^1\) StoryTime builds upon this type of program in two ways: first, through a story-based learning curriculum; and second, by extending the technology to the most popular internet-based messaging apps.

Mobile messaging apps are getting big.\(^2\) Falling data prices and cheaper devices enable tens of millions of Americans to use messaging platforms like Facebook Messenger and Apple’s iMessage. These chat platforms have now opened to developers, empowering thousands of engineers to create text-based applications to provide richer and more fluid app experiences for their users.

---

Facebook and Apple were some of the first to open their messaging platforms to developers, but I predict that more messaging apps will soon follow. So I want to design a system that will scale to these new messaging platforms without requiring a total redesign.

Goals

The basic functionality is simple: sending ordered sequences of text and picture messages to users, scheduling reading reminders and notifications, and handling user preferences like home language and send time. Each messaging platform has unique interfaces and requirements; but fundamentally, they share the most basic function: messaging. I would like to create a single system that provides a generalized interface for all available messaging platforms. This system would unite the basic functionality in a clean, modular design. It would provide an easy interface for developers to write platform-agnostic code to send message sequences to users. And it would make adding new platforms to the system much easier.

Challenges

The first version of this system will send stories through SMS, iMessage, and Facebook Messenger. Each of these platforms have different requirements in the form of dependencies, inputs, and outputs. For example, text messages require a user’s phone number, the message body, and an access token to the Twilio API; Facebook, on the other hand, needs a separate running server, a web hook, hashed signatures, and complex data objects as inputs for sending messages.

The second major challenge lies in the difficulty of using antiquated technology like SMS to send ordered sequences of text and images. To create the best reading experience for parents on SMS, text messages must arrive in a reasonable amount of time (I define reasonableness as minimizing the amount of idle time between consecutive messages). Text messages must also arrive in a specific order. However, cell phone carriers promise neither the time nor the arrival order of messages (see Twilio documentation)—texts are completely independent of one another once sent to the carrier. Thus, StoryTime’s application of SMS and MMS technology pushes beyond their standard use case.

3 https://support.twilio.com/hc/en-us/articles/223134247-Can-my-SMS-messages-arrive-in-order-
Right now, the standard way to send a sequence of text messages is to send them one at a time in a specific order and essentially pause for five to fifteen seconds before sending the next one. The wait-time is arbitrary, but if you pause for too little time, your recipient might receive his messages out-of-order; if you pause for too long, the recipient must wait full minutes to receive longer text sequences. There is no gold standard wait-time because timing is inconsistent for both the carrier and the user’s handset phone. My goal is to move past this naive solution by designing a system that leverages SMS status data (the timestamp for when a message has been enqueued by the carrier, for example) to intelligently determine the proper send-time of the next message in a sequence. In so doing, I hope to minimize user idle time and enforce the text messages’ order of receipt.

Last: I want the ability to quickly adapt the learning curriculum. This would be difficult using the most common libraries: sending even a simple text requires 6 lines of Ruby code. Sending a Facebook message requires even more. So I will devise a domain-specific language for developers to easily write scripts for the StoryTime curriculum. This is important because eventually we want to have hundreds of days’ worth of messaging content in the curriculum, and we need to easily adapt them.

The main learning concepts are:
1. Meta-programming techniques to create a domain specific language
2. Object-oriented design patterns to create the shared messaging interface
3. Web-hooks, parallel background jobs, and other web development practices to create a scalable, dynamic backend.

**Deliverables**

1. A web backend to send SMS and Facebook Messenger stories on schedule
   - a single interface for sending text and picture messages (platform-agnostic)
   - implementations for SMS and Facebook messaging functions
   - modules for message scheduling, user preferences, and notifications
2. A domain-specific language utilizing meta-programming techniques to write the StoryTime curriculum scripts (concept image below).
3. The algorithm and implementation to enforce order and timeliness in SMS sequences
My initial idea for a curriculum script DSL:

```ruby
# works across messaging apps: SMS, Facebook Messenger

# curriculum day 1:
send image('intro.jpg')
send 'Hi <parent>! This is StoryTime.'
send "We'll be sending you free books by __APP__."
end

curriculum day 2:
send 'Hi <parent>! Here's another story for you and <child_name>.'
send "When you see orange bubbles, try asking your child what's happening in the picture."
end

request feedback('How did you like this story?')
```

**Technology Stack**

1. Ruby + Sinatra for the web backend
2. Twilio for sending SMS and MMS
3. Facebook Messenger platform
4. Sidekiq for background worker processes (concurrent messaging)
5. I18n Ruby Gem for translations

**Conclusion**

By the end of CS490, I would like to have built a robust story messaging system, which can easily incorporate new messaging platforms, overcome the constraints of SMS, and allow developers to quickly adapt our learning curriculum. I believe this project could eventually make an impact on children’s lives by reaching their parents through the technology they already use.