

AudioteBook

Class of 2013

R. Raman and R. Walport

Advised by E. Bell

© Copyleft
R. Walport and R. Raman
2013

Application and code:

The application comes in three distinct parts:

1. An iPhone application
 - a. Code found at <https://github.com/RWalport/AudiotebookApp>
 - b. Built solely in Xcode using iOS and its frameworks. The only external code used was from Simon at <http://maybelost.com/> for core data access. This code is clearly labelled as such.
2. A REST API built with Django and PostgreSQL
 - a. Code found at <https://github.com/rashmi-raman/audiotebook-api>
 - b. Simply put, a REST based API allows an application to retrieve database objects via the HTTP protocol's GET method, and can allow for users to submit data to the database as well via the POST method.
 - c. This application is available on Heroku, a cloud based platform that allows applications to run
 - d. The application is available at
<https://fast-dusk-7046.herokuapp.com/api/reportinghistory/?format=json>
3. A viewer built with Backbone.JS, HTML and CSS
 - a. Code found at <https://github.com/rashmi-raman/audiotebook-viewer>
 - b. This viewer allows the user to search and filter her reporting data as well as sort it by reporting date.
 - c. Prototype, with current data, found at www.audiotebook.com

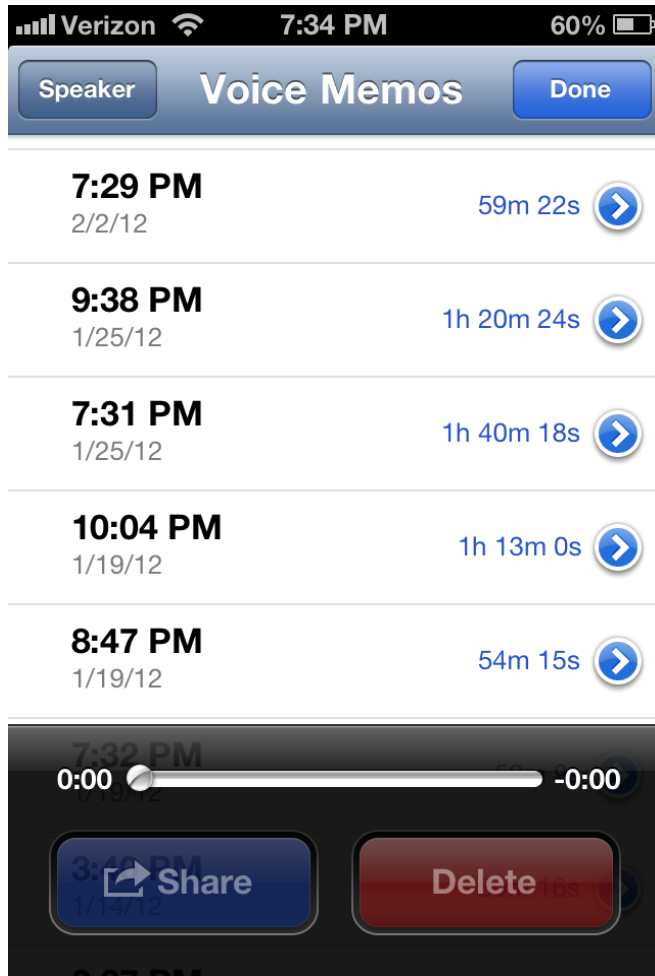
Development Blog

I saved that where...?

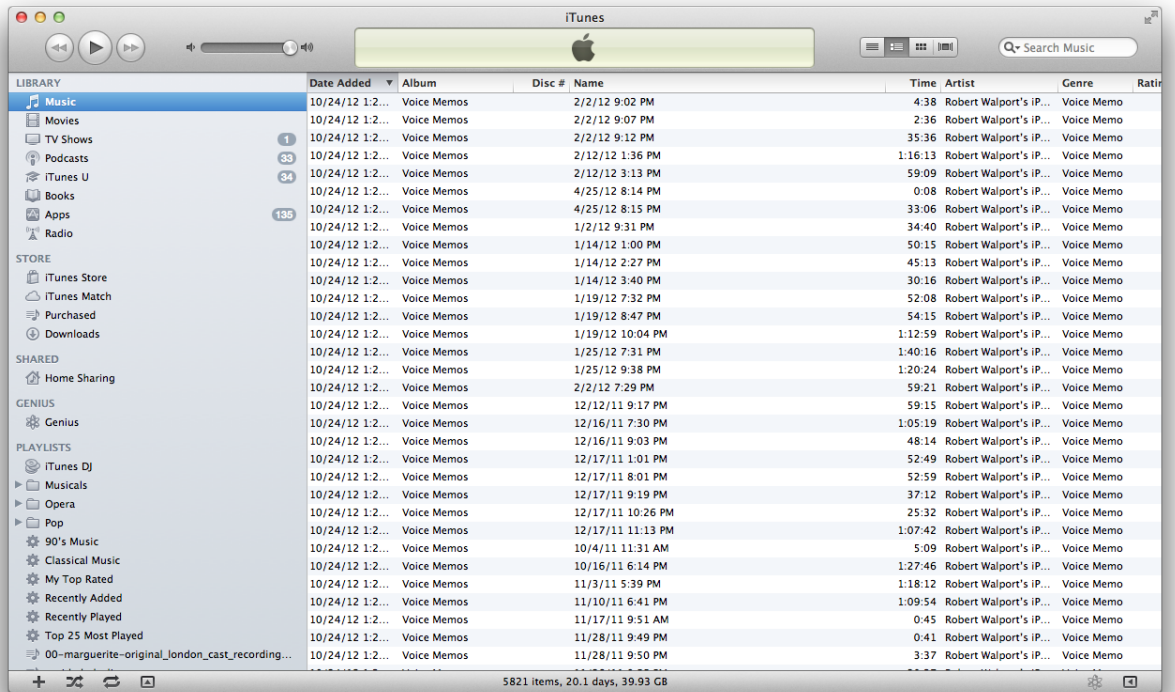
Posted by [rpw2114](#) | Thursday, 11 November 2012 | [Developer's Blog](#)

Journalists are quite spectacular accumulators of information. For every interview they might have a notebook full of poorly scrawled notes, a full audio recording, photos and even video. Factor in that they might be speaking to upwards of a dozen people a day and the fact that most journalists aren't the best archivers is hardly surprising.

Take recording an interview on an iPhone. The in-built Voice Memo app is a reliable and easy piece of software for audio recording but here's a look at what you end up with (see below). Yes it's great that they are indexed by date and time but wouldn't it be helpful if I could label them with who I was talking to, or where I was, or even what they said? Sure I can put a custom label on them (that's a couple of words at most) and the downside of that is that I no longer get to see the date and time... (and the label isn't searchable or orderable either!)



6 Hours of interviews “helpfully” indexed by the time I took the interview
The problem gets worse not better once you migrate the recordings onto your computer. First and foremost Apple does not make the transfer the easiest of tasks, and once you’ve got it onto your computer you’re presented with...



Which is if anything more confusing and less helpful than what you had on the phone.

At least now however the tagging process can begin properly.

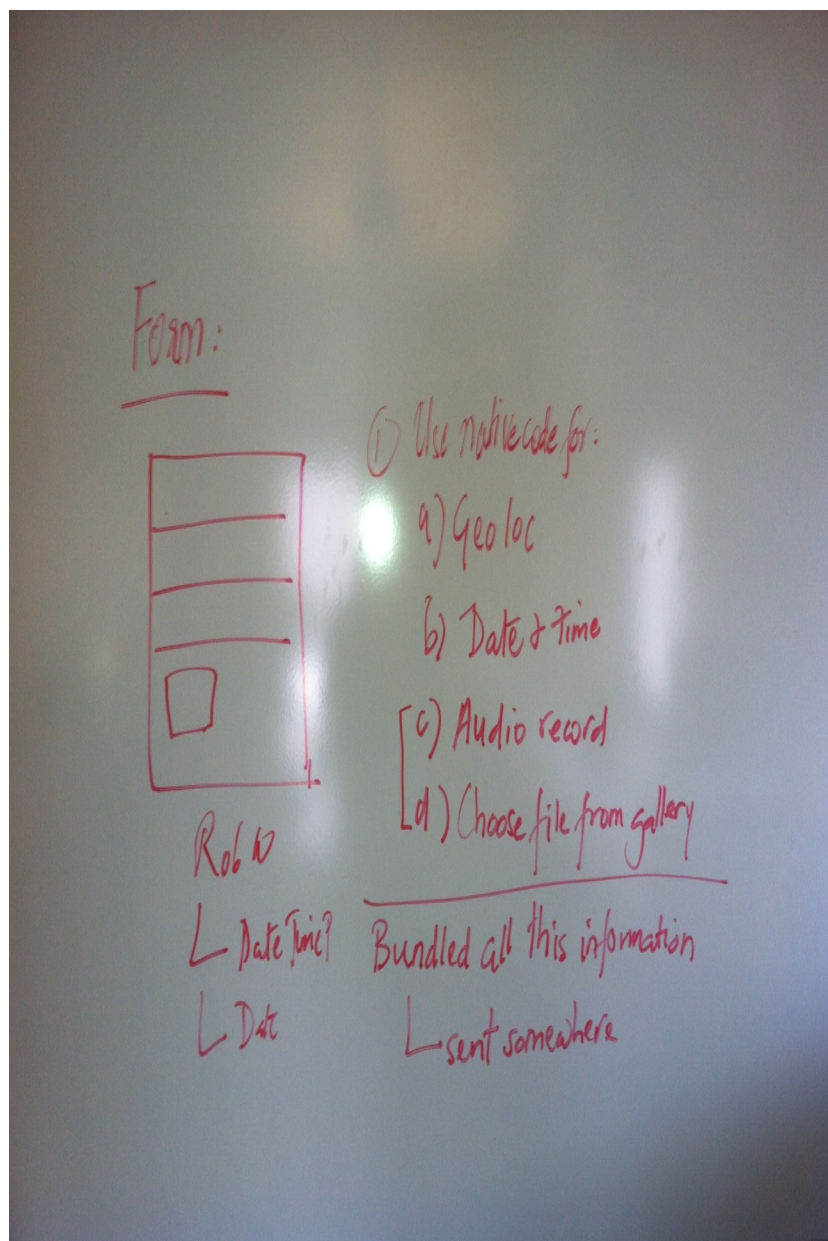
The trouble arises here that many people aren't all that fussed about organizing things for the future. If you know where the files are right now, and you won't regularly need them in the future, then why put the effort into organizing them fully? What's more we haven't even touched the fact that all the other media attached to this interview will live in disparate places probably never to be united completely.

Is this a good state of affairs? We don't think so.

Proof of Concept thought process

Limiting our ambitions has not been a very easy task – we'd like to add bells and whistles of every form and function to our initial proof of concept (PoC). But then, our latent editorial skills emerged.

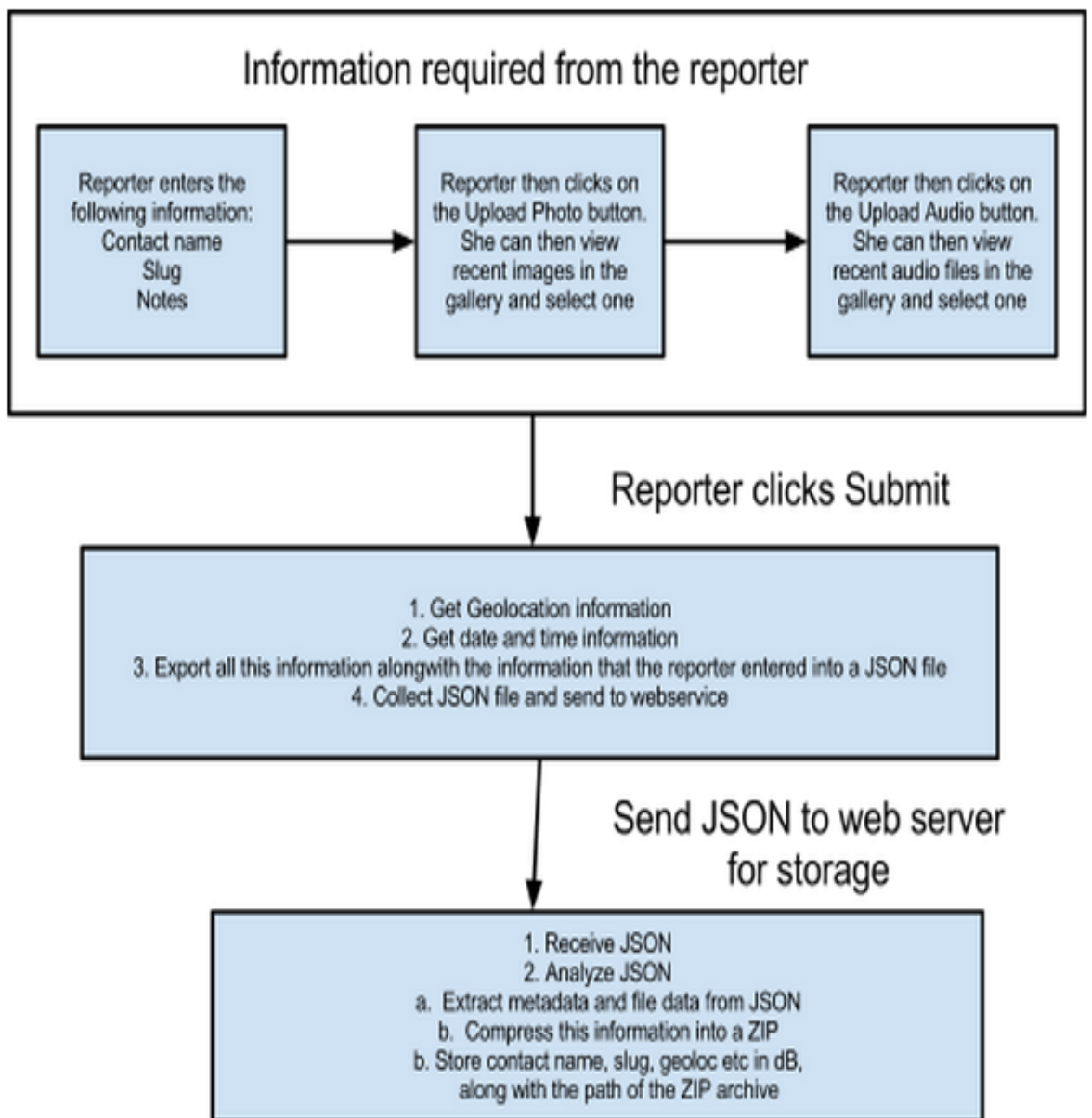
Here's a glimpse into our thought process.



Jotting down our thought process

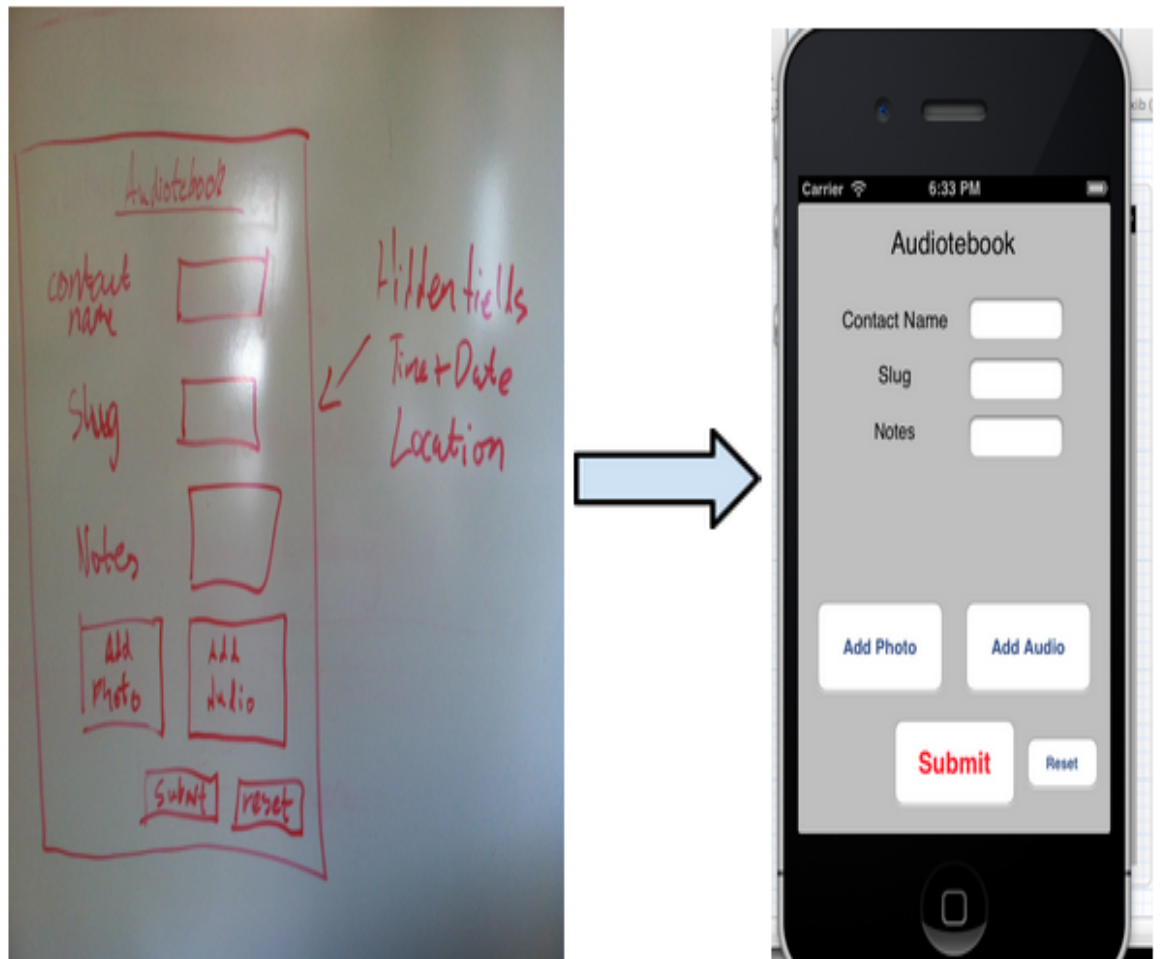
We've now ruthlessly scoped on what needs to be included in order to prove that the mechanism actually works. Some features were (regretfully, we may add) either postponed, cancelled or put on the wishlist for various reasons.

First, we needed to outline our basic process. And here's what we came up with.



Process flow

Now that we had this process flow in place, we next decided to create a mockup of the screen. Whiteboards have never been more useful. And finally, we designed an initial version of the screen in XCode.



Making an actual iPhone screen!

Following this, we came up with the idea of creating a simple viewer in Java to see a tabulated version of all the structured reporting data that had been submitted so far.

The need for Audiotebook

The audiotobook is our attempt at evolving the journalism profession but before we move on to what we are creating, it's worth considering where we are now and why this is the case.

In practice the current reporting and writing process roughly breaks down into three phases.

The first stage is the shortest and hardest, and depending on the reporter and can overlap variably with the second stage. This stage is all about finding a story. There are many ways to go about this but for many it will be a combination of influences; knowledge of a given area, a journalist's "beat"; conversations with sources; past stories; friends and family or even just little things that catch a reporter's eye.

The second stage is what we might call the collection phase. A reporter first and foremost will go out and talk to people¹. They will commonly do this with an audio recorder and a physical notepad, as well as possibly a camera and even a video recorder. A reporter might also do document research, whether making freedom of information requests to the government, scanning documents in Libraries or searching on the internet. These pieces of information will then coexist, but do so across multiple platforms and pieces of software. It is the reporter's job to track when, where and how all this is recorded.

The third and final stage is the writing and publishing phase. The reporter brings all their data together: interviews, documents, photographs etc. and writes a story. This

¹ http://www.thenewsmanual.net/Manuals%20Volume%201/volume1_16.htm

can vary in length from a few hundred words to many thousands and contain any combination of media such as text, audio, video and graphics. The story might be passed between editor and reporter several times, gradually honed to its finished product and then published. Traditionally this would have been in a newspaper but increasingly this might be online: one study in New Zealand showed that 65% of the material coming out of the main newsrooms was exclusively written for online publication².

What we propose is to create a reporter's notebook 2.0, we've dubbed the audiotobook. The notebook 2.0 is much lauded concept, but rarely well defined. Ideally, in our vision, it would be an application, across multiple platforms, that enhances the process of acquiring a story and as a result enriches the end product, whilst also storing and curating the data such that it can be used for future stories.

We propose a new reporter's life-cycle built around this new tool, one with similarities at almost every stage but changes that fundamentally alter the potential product without drastically altering the day to day activities of the reporter.

Stage One: Find a story. This stage doesn't change all that dramatically initially. Finding stories is still a matter of all the influences mentioned in the current cycle. What changes is how this phase can be thought of over time. Traditionally, even if covering a topic or geographical beat, stories, roughly speaking, exist as individual entities. As we move through the altered process below you will hopefully see how I propose molding this process into a cyclical one through which "stories" are born out of other "stories".

² <http://fuseworksmmedia.com/blog/if-its-important-its-online-fuseworks-research>

The second phase is where the notebook 2.0 tool comes in. It's a mobile app which allows the user to make audio recordings that, via a simple interface, the reporter can annotate with structured information. Some it can be generated automatically: start and end times, date, geolocation. Other elements must be entered manually: whom the interviewee is (though this is a one time heavy manual entry, i.e. entering all the details of the person, and then simply requires picking a name from a list on every other occasion), the story slug (a journalism shorthand form of story labelling), photographs etc. This data is then transmitted to the web app with all the media attached.

The Web app then contains every interview that the reporter has conducted in a structured database. The reporter can continue to add information through the app, whether these are pdfs associated with a given interview or a publication related to a given person. Essentially the app becomes a research library with each item considered as an entity linked to every other. This isn't by computer science standards a revolutionary idea, but it's a step that journalists simply aren't taking currently. Writing the narrative is then a process that involves far less trawling through material and far more time analysing the important elements.

Our work has been principally on getting a prototype mobile app working with a simple web interface for interacting with the recorded data. The first stage app will allow the user to take audio, photo, notes as well as geolocation and time data and transfer it from a simple iphone app to a web database. The data is only minimally interactive online but displayed in a clear tabular format.

In the long term the goal would be to create a web app that works as both a sophisticated database but also potentially a CMS. By tying the writing process to the

actual raw materials the reporter can now include many of their full research documents in their entirety. A freedom of information request document for example can sit alongside the narrative on its own page. A news website becomes more than just a transient, up to the minute, content supplier, a genuine resource for wider learning.

The benefits of such a system are not only seen at the reporter level. The data really starts to shine on a newsroom scale. An editor now has the pooled data of all their reporters (albeit with certain restrictions in the case of certain types of confidential sources), a potentially incredible resource. Through this they will have extensive data on who is talking to whom, where and why. Are all geographic and topic areas being properly covered? Are two reporters talking to the same source without being aware of the fact (by no means an impossibility)? Are separate teams' research overlapping in unexpected ways?

Audiotobook Proof of Concept implementation

Posted by [rr2779](#) | Thursday, 1 January 2013 | [Developer's Blog](#)

iPhone app

The iPhone app was built using the XCode IDE provided by Apple. The following features were added to the app :

- a. Input boxes for contact name, slug and any additional notes
- b. Image picker widget for selecting pictures from the camera roll
- c. File picker to select audio files associated with the reporting

d. Automatic geolocation and date/time recognition

Once the submit button is clicked, all this information was serialized into a JSON stream and sent to the webservice using the PUT method of the REST API that we created

Web Service

We designed a webservice that implements the REST API. The API exposes two methods.

PUT

The PUT method is used to submit data to the webservice. Once the JSON stream is received from the iPhone app, the webservice :

- a. Separates the metadata and file information from the stream, from the actual file data
- b. Creates a metadata file with the information gathered
- c. Creates an image and audio file using the raw file data
- d. Compresses these files in a single archive
- e. Stores information related to this in a mySQL database

GET

The GET method is used to retrieve data from the database. The GET method returns an array of JSON objects containing information from the database. At the moment, the GET method returns all entries, but a future development would include adding filters to the search results.

Web Viewer

The Web Viewer can be used to retrieve information and the archive created containing data from the reportage. It invokes the GET method of the API, retrieves the JSON, analyzes it and then displays it in a tabular format.

Further, the geolocational data is sent to the [GeoNames API](#) for reverse geocoding, in order to get the nearest postal address.

The archive file is also available in a downloadable format from this viewer.

Feature set for Audiotobook v0.2

Posted by [rr2779](#) | Thursday, 2 February 2013 | [Developer's Blog](#)

iPhone:

1. Date format: No special characters
2. Audio form to bit stream base64
3. Multiple photos
4. Contact functionality – add more fields to get a richer dataset of sources
5. **Improve the GUI / look and feel!**

Web Application / REST API :

1. Ditch Java, adopt Django
2. Hosted online
3. Searchable (contact name, dates and slug)
4. Sortable
5. Filterable

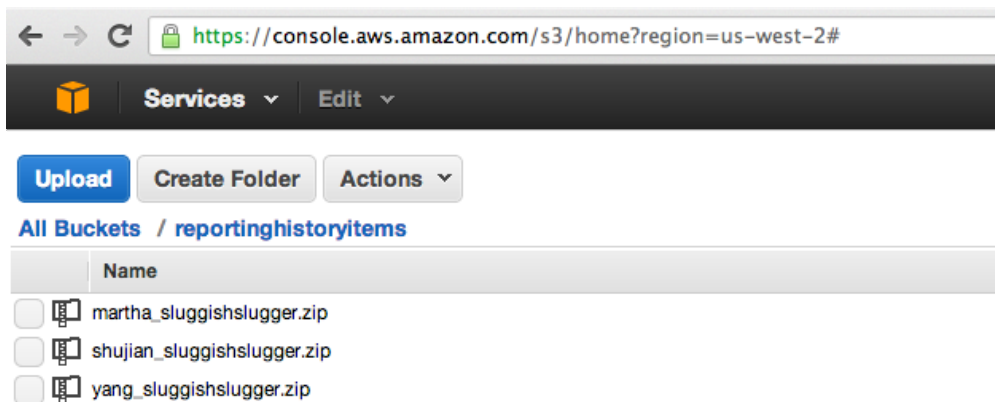
Create a data dictionary

Audiotobook v1 implementation

Posted by [rr2779](#) | Tuesday, 12 February 2013 | [Developer's Blog](#)

Static files

All our archive files will now be stored in a “bucket” provided by Amazon’s S3 service. Amazon S3 (Simple Storage Service)³ is an online storage web service offered by Amazon Web Services. So, now we have a permanent and accessible storage system for the archives created during reporting.



Web Service

We redesigned the web service that implements the REST API. And there were mainly two reasons for rewriting the web service.

³ <http://aws.amazon.com/s3/>

In the middle of January (while we were developing our PoC), the US Government issued a severe warning⁴ about a security related bug in Java and advised all users to disable Java on their browsers. The second reason is that while there are several platform as a service (PaaS)⁵ options which support Java applications, deploying Java applications can get to be a cumbersome and unwieldy process.

Hence, after we completed the PoC, we took a decision to rewrite the REST API as well as the web viewer in a different platform. We had three options for implementation - Ruby on Rails⁶, Node.js⁷ or Django⁸. Each platform has its own benefits and disadvantages in terms of difficulty in coding, support in various platform services in the cloud etc.

We decided to go with Django because firstly, we LOVE Python (who doesn't?) and more importantly, we love journalist-coders. The Django web framework was built by journalist-coders from Kansas including Adrian Holovaty, the founder of the now defunct EveryBlock⁹. And, yes, our decision might seem arbitrary but ultimately, it would have come down to such criteria even after creating a pros and cons chart¹⁰. This, unfortunately, is a truism in the software business - you want scalability, performance, support from the community, ease in coding and quick time to market.

Our API is now hosted on Heroku at this link¹¹. The API exposes two methods - GET and POST (similar to our previous blog post). The big difference is that it's now out there! In a public link, no less! So, that means - unlike our PoC where an iPhone simulator on a Macbook was communicating to a Java API running on the same

⁴ <http://www.reuters.com/article/2013/01/11/us-java-security-idUSBRE90A0S320130111>

⁵ http://en.wikipedia.org/wiki/Platform_as_a_service

⁶ <http://suarism.com/2011/04/01/how-to-write-a-ruby-rails-3-rest-api/>

⁷ <http://mcavage.github.com/node-restify/>

⁸ <http://tastypieapi.org/>

⁹ <http://everyblock.com/>

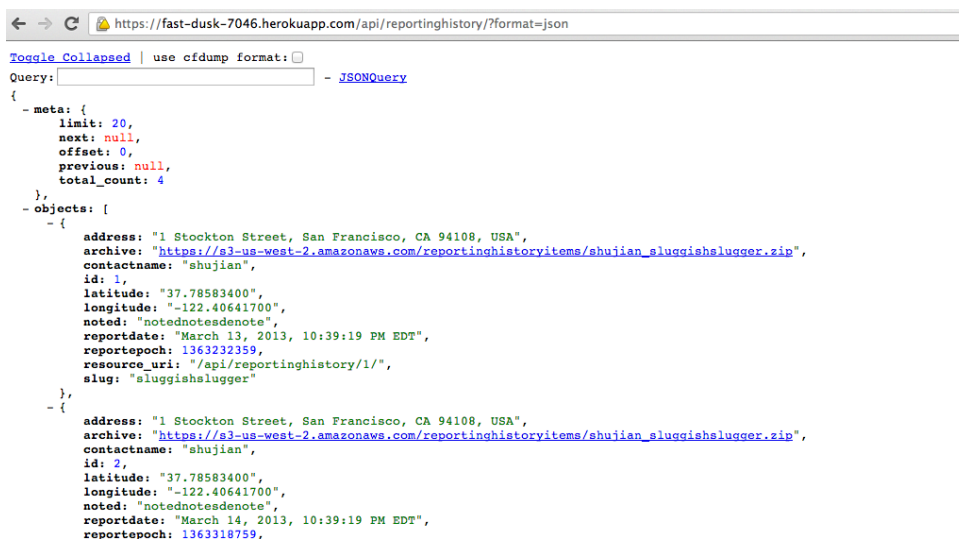
¹⁰ <https://www.udemy.com/blog/modern-language-wars/>

¹¹ <https://fast-dusk-7046.herokuapp.com/api/reportinghistory/?format=json>

Macbook, we can now submit information to this API from a copy of the app on an actual iPhone! Woot!

The geolocational data is sent to the Google Reverse Geocoding API for reverse geocoding¹², in order to get the nearest postal address.

And, we can now write a web viewer which can pull information from this API.



The screenshot shows a web browser window with the URL `https://fast-dusk-7046.herokuapp.com/api/reportinghistory/?format=json`. Below the address bar, there is a "Toggle Collapsed" button and a "use cfdump format:" checkbox. A "Query:" input field is present, followed by a "- JSONQuery" label. The main content area displays a JSON response with the following structure:

```
{
  - meta: {
    limit: 20,
    next: null,
    offset: 0,
    previous: null,
    total_count: 4
  },
  - objects: [
    - {
      address: "1 Stockton Street, San Francisco, CA 94108, USA",
      archive: "https://s3-us-west-2.amazonaws.com/reportinghistoryitems/shujian_sluggishslugger.zip",
      contactname: "shujian",
      id: 1,
      latitude: "37.78583400",
      longitude: "-122.40641700",
      noted: "notednotesdenote",
      reportdate: "March 13, 2013, 10:39:19 PM EDT",
      reportepoch: 1363232359,
      resource_uri: "/api/reportinghistory/1/",
      slug: "sluggishslugger"
    },
    - {
      address: "1 Stockton Street, San Francisco, CA 94108, USA",
      archive: "https://s3-us-west-2.amazonaws.com/reportinghistoryitems/shujian_sluggishslugger.zip",
      contactname: "shujian",
      id: 2,
      latitude: "37.78583400",
      longitude: "-122.40641700",
      noted: "notednotesdenote",
      reportdate: "March 14, 2013, 10:39:19 PM EDT",
      reportepoch: 1363318759,
    }
  ]
}
```

PUT

The PUT method is used to submit data to the webservice. Once the JSON stream is received from the iPhone app, the webservice :

- a. Separates the metadata and file information from the stream, from the actual file data
- b. Creates a metadata file with the information gathered
- c. Creates an image and audio file using the raw file data
- d. Compresses these files in a single archive and stores it in the S3 bucket
- e. Stores information related to this in the PostgreSQL database provided by Heroku

¹² <https://developers.google.com/maps/documentation/geocoding/#ReverseGeocoding>

GET

The GET method is used to retrieve data from the PostgreSQL database. The GET method returns an array of JSON objects containing information from the database. The GET method can provide results filtered on the basis of contact name and slug and can be sorted by the report date.

Web Viewer

Another implementation decision we needed to make involved rebuilding the web viewer. Initially, this was built on Java and was running on a local machine, but we wanted to make this available online as well.

Hence, we decided to build this in simple HTML using the Backbone JS¹³ framework. Backbone is a lightweight, javascript framework built especially to communicate with APIs and handle the lifecycle and display of these data objects. And Backbone was designed alongside the DocumentCloud¹⁴ project - so another great framework designed by journalist-coders!

Our web viewer can be accessed from this link¹⁵, once again using Amazon's S3 service to host a static website.

¹³ <http://backbonejs.org/>

¹⁴ <https://www.documentcloud.org/home>

¹⁵ <https://s3-us-west-2.amazonaws.com/audiotebook-viewer/index.html>

Your reporting history

Search Reporting History [Go](#)

[Sort by date](#) | [Reverse sort by date](#) | [Clear filter](#) |

shujian for story sluggishslugger
March 13, 2013, 10:39:19 PM EDT at 1 Stockton Street, San Francisco, CA 94108, USA
[notednotesdenote](#)
[Download archive](#)

shujian for story sluggishslugger
March 14, 2013, 10:39:19 PM EDT at 1 Stockton Street, San Francisco, CA 94108, USA
[notednotesdenote](#)
[Download archive](#)

The Web Viewer can be used to retrieve information and the archive created containing data from the reportage. It invokes the GET method of the API, retrieves the JSON, analyzes it and then displays it in a list format.

The archive file is also available in a downloadable format from this viewer.

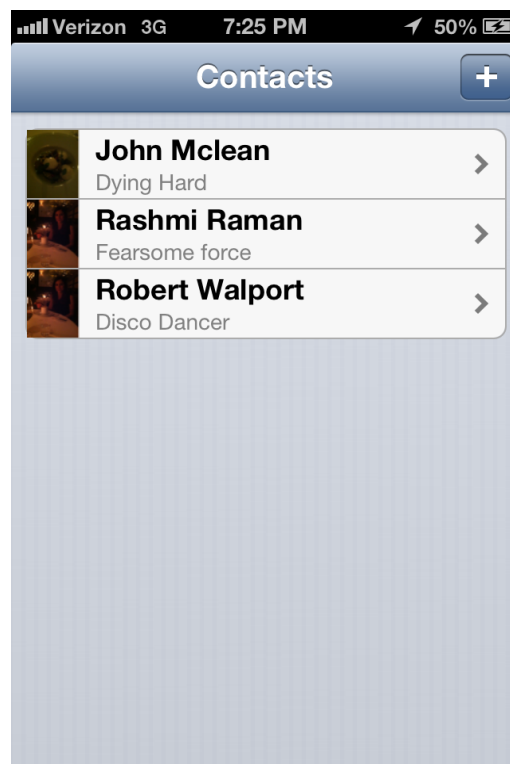
Contacts on the iphone

Posted by [rpw2114](#) | Wednesday, 3 March 2013 | [Developer's Blog](#)

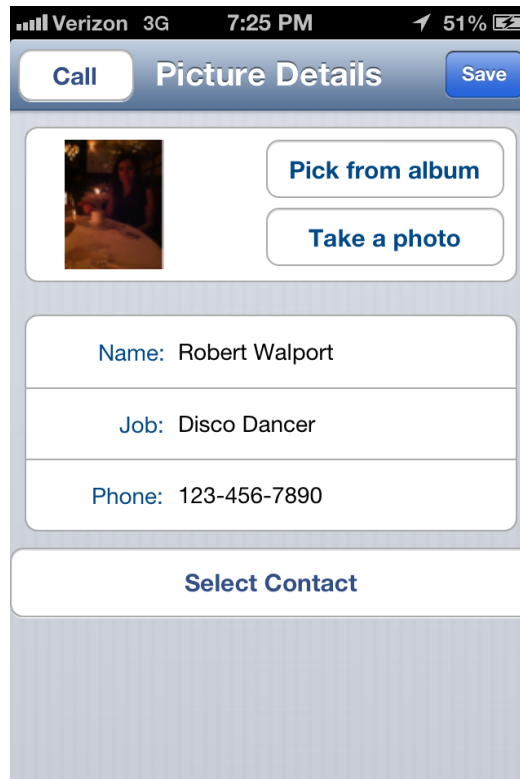
As part of our development of the phone app we considered how and why a reporter might store data locally. One major facet of any interview is the person you're interviewing and information on them is clearly crucial. What is more, over the course of a journalist's professional life, the same sources might be interviewed/contacted many times. How then to create a system that quickly and efficiently allows access to

these contacts, and what isn't broken doesn't much need fixing and the standard phone contacts was our model as we progressed.

A first crucial point is that we decided to consider a separation between phone contacts and our app sources. You don't want all the people from your standard contacts list in your source database, and there are plenty of potential fields in a source database you don't want in your contacts (e.g. associated beat, employer etc.). We decided therefore to build a sources database from the ground up within the application, one that can grow over time and allow the reporter to enter the details once and then pull up the source by just their name (or potentially some other key). On the flip side of this, we don't want people having to enter information twice (once in Audiotobook and another in contacts) so we have attempted to provide some of the phone functionality within our contacts pages (you can make a phone call straight from within the app!).



The Contacts List



The Contacts Edit Page

At this point the deliberation over what fields to include continues and to be honest we'll never settle on a perfect few. Down the road we would like include the option to allow the user to add custom fields which would eliminate some of these problems altogether. Another factor is instant messaging which is currently unsupported. In an ideal app texting could be done within the app and these messages then included in the web app as a potential research/information source (privacy considerations clearly an issue here though). Another future advance would be to add the ability to import select phone contacts to the application database.

Corridor Testing

Posted by [rpw2114](#) | Thursday, 21 March 2013 | [Developer's Blog](#)

As part of the final stages of our prototype development we decided to take the app to the corridors of the J school to see what people think about it, what they think we're getting right and what we're getting wrong. Most importantly it was about finding out where we should be taking the app in the future, if we have the opportunity to develop it further.

Pierce Gibson Crosby is currently in the MS program. He liked the idea of the app and could see himself using a product like this but thought the interface needed to be simplified (and found our black, grey and white color scheme difficult). He liked the overall ability to structure data but thought more flexibility was needed. Our audio currently comes in one format, mp3 (though the iphone is capable of recording in higher quality wav) and ideally he would want the ability to export in different formats depending on the end use of that audio. He also raised the idea of allowing for more flexibility in the "interview" format. Could our app also allow for "audio notes", i.e. personal recordings that are stored in the database as well?

The contacts section worked well but he instantly raised (and this is something discussed in one of our blog posts) that you should be able to import contacts from your phone's native contact section. In line with how we've expanded contacts beyond the single interview, he also raised the issue that sometimes a reporter might take photos of an event without associating them with a particular interview or source. This is a natural (and perhaps important) expansion from our current app though beyond our scope for this prototype.

Finally he was concerned about the lack of history/activity logs on the phone. Though the contacts live as a database on the phone - all data relating to interviews is sent to

the remote server and then accessible only from the web. Pierce suggested that perhaps there should at least be some sort of local history (even if it lacked the actual audio data) to provide reminders out in the field.

Pierce didn't much like the idea that an editor might have access to his interview history. This is clearly a line that would have to be tread lightly, though from our development point of view we would still want to provide maximum flexibility to allow different newsrooms to deal with this issue according to their own best practices.

Kenan Davis is the digital media coordinator for the J school and also a graduate of the MS program. He found our UI design wanting, finding it confusing that one cannot access the contacts section without first going through the interview page. He is quite correct, it's totally non-ideal as it currently stands, and this is in some ways an artifact of our original intentions vs the app in execution. Our original idea was for the app to purely be about recording interviews and structuring that data. As we expanded the contacts section it has taken on a full fledged role (for example with the calling functionality), that needs to be accessible from some sort of menu. This is definitely something we should alter in the near to short term. He also raised the issue of what happens when the user has many contacts, we need a level of search functionality to get around this scaling problem.

He was also concerned about the reliability and how we ensure that every interview gets saved. For purely practical reasons this is an element we haven't explored in our development thus far and we do need some backup copies saved locally until the remote server can confirm that it has received all the data. Further to this, he wondered why he wasn't able to get the historic information on his phone. For hard drive capacity reasons we are unable to store significant audio locally but some sort

of system not unlike how dropbox deals with storage might work. This would mean the app would get the required data as and when it is asked for (much like browsing the web). A great idea but a huge expansion in scope

He didn't much like the name "audiotebook" either and nor did Pierce. In reality we don't really like it either. We should change this!

Major talking points then were:

- Reliability - making sure things get saved
- Flow - navigation between contacts, interviewing and potentially photos
- History - exploring the database through the phone app
- Flexibility - allowing different users to utilise the same functionality in different ways.

Final Thoughts

Posted by [rpw2114](#) | Wednesday, 20 March 2013 | [Developer's Blog](#)

In drawing to a close what has been months of development, discussion and research we find ourselves with a working prototype albeit one with substantial holes in functionality. What we have achieved we believe proves the concept (check out the corridor testing for direct feedback), while demonstrating the need for a more coherent set of tools to make the product worthwhile.

Our starting point came from the lack of applications devoted principally to journalistic

reporting and some six months on, the marketplace hasn't much changed. Though many tools exist that have been co-opted by journalists - like soundcloud¹⁶, dropbox¹⁷ and audioboo¹⁸ - few if any of these products was actually created with the reporting process as the principal target.

What we have built serves its purpose to an extent but falls down because it lacks sufficient verification of storage. Data can all too easily be lost, a situation that is completely unfeasible for practical journalism. Security is one area we also did not explore in broader terms. Information security is an area of considerable debate and many journalism organizations, such as CPJ¹⁹, are working hard to raise awareness. Our app and platform could potentially place us at the center of this, we'll be sitting on vast amounts of private data with potentially many foes who want illicit access.

Not only that, there are the more commonplace users to consider. A key element of the design was to facilitate information flow across the newsroom but how much data should be shared, how much can editors see? We want to encourage collaboration and sharing, but there are limits and we haven't delineated them clearly at this stage. Secondly on more political issues of the rights of the journalist and their sources to anonymity, could we be subpoenaed to reveal a journalists interviews?

From a technical standpoint the ios platform and xcode IDE were new to the whole team and this provided plenty of headaches along the way. The end product works but lacks polish, purely for times reasons, and we are computer scientists/journalists not designers and that also really shows. UX is a major element in driving adoption of a product and ours is not there yet.

¹⁶ <http://soundcloud.com/>

¹⁷ <http://www.dropbox.com/>

¹⁸ <http://audioboo.fm/>

¹⁹ <http://www.cpj.org/>

We hoped you've enjoyed reading this blog over the last four months and hopefully this won't be the last you hear from us. From everything we've done and everyone we've spoken to, there's real promise here, it's just a question of getting all the pieces to fit together.