Bolt

June 18, 2013

DoubleEncore

Johnathan Aspinwall
Matthew Jordan
Derek Schissler

I. INTRODUCTION

The Bolt App will allow firefighters to quickly and accurately navigate to the exact addresses of fires or other emergencies. The app will show an intractable, gridded view of the entire city, from which a firefighter would have the ability to click a grid cell to zoom in on any specific grid section.

CLIENT DESCRIPTION

The Bolt App will be built for Double Encore, a Denver-based company that specializes in mobile development projects. Upon completion, the app will be delivered to the true client of this project: the Golden Fire Department. The Golden Fire Department is very important to the protection of the people and assets within the bounds of Golden and other surrounding areas. It is important for the fire department to be able to accurately and easily navigate to emergencies. Currently, the Golden Fire Department uses extremely large binders that consists of a grid-map system of the City of Golden and surrounding areas. While navigating to emergencies, firefighters must turn from page to page in this large binder in an attempt to find to the exact location of the emergency. This can be very difficult while the fire engine is moving and as well this system is slow, tedious, and takes up a lot of room. Since all of the fire engines in the Golden Fire Department are equipped with iPads, the idea formed that they could be used to solve this problem.

PRODUCT VISION

The Bolt App will be targeted by fire departments and will be used by firefighters while navigating to emergencies. The app will consist of a friendly user-interface that displays the map of the entire city, separated into grid cells. These maps will include both exact, accurate addresses of structures and the locations of the various fire hydrants located within the city. Additionally, there are some locations, which the client calls “plates”, which have the ability to be zoomed in to show an extra level of detail. From any given cell view, the firefighters will be able to quickly swipe in any of the four directions to show adjacent grid cells in order to aid in determining the best route to emergencies. This will allow for easy access to detailed information that will be important when navigating to emergencies. To start, the Bolt App will only be used by the Golden Fire Departments. If the app is successful within their organization, then there may be the possibility that the app can spread to other fire departments across the nation. This product will be different than other similar apps on the market because it will be built specifically for fire department use. There are no other mobile apps that can be used to create a grid-mapping system using PDF documents. Since Bolt accomplishes this, and does so in an elegant manner, it should be very useful to fire departments in emergency situations.

RISKS

The risks involved with this app are mostly limited to the interactions with Dropbox and with the user who creates the pdf files needing to follow the proper specifications. Minimizing these risks is crucial because we are developing a product that will be used for emergency services. The failures could include the program crashing, failing to sync with Dropbox, or not loading the files properly.

Specifically regarding Dropbox, working with the API could lead to issues related to syncing the files, especially due to the required file structure. Additionally, if Dropbox changes any of its protocols currently used in the app, significant changes could be required to the code in order to maintain functionality. Another risk here is Apple allowing communication with the Dropbox API, since Apple is known to change policies abruptly if there are any perceived security threats.

The risk involving the user should be avoidable since there is a specification document supplied to those who are creating the pdf files to work with this app. However, if Adobe changed their format for the creation of these documents, that would also require changes to the code of the app, posing another risk, though less immediate since this would only apply to newly created documents.

II. REQUIREMENTS

FUNCTIONAL REQUIREMENTS

The main functional requirements for the application to work are, first, the Dropbox integration to download the maps onto the devices, and second, the screens that will display and allow interaction with the maps. Additional requirements include a search function that allows the firefighters to immediately find the location of an address on the map itself, as well as an ability to view informational documents, called plates, when further detail not included on the maps is needed.

The detailed functional requirements for the Firehouse Grid Search app includes:

- Dropbox
    - Allow the user to manage the Dropbox credentials that will be used for the app
    - Integrate with the Dropbox API to download pdf maps into local storage
    - Upload png renderings of the pdf files back to Dropbox
    - Update the local files if new ones are added to, or changed in, Dropbox
- Map Display

- Have a Home display that will show all of the available grids in a “Grid Map Overlay” (shown in Figure 1 below) that is interactive such that tapping the grid cell on the screen brings up a detailed view of that cell.
- Gestures
  - When looking at the grid, touching any grid cell will open a more detailed map of that area
  - While displaying a grid cell, swiping up/down/left/right will display the respective grid cell
  - Pressing the up/down/left/right/diagonal buttons will display the respective grid cell
- Searching for house numbers while looking at a specific map will highlight the location
- Go-to feature that allows users to view a list of the grids and plates, then allowing the user to simply tap the desired file to have it displayed on screen.
- Notes feature will allow users to add notes to files for pertinent information that may be lacking or changes that need to be made (ex. Fire hydrant info, master key locations, etc)
- Ability to attach info to a specific location within the map

NON-FUNCTIONAL REQUIREMENTS

The components of the application that will be necessary for the application to run correctly and consistently are the maps themselves. The maps will need to be pre-built and inserted into a Dropbox account that the application can use to retrieve the files. They will also be cached on the iPad for faster access, requiring occasional syncing with Dropbox.

The detailed non-functional requirements for the Firehouse Grid Search app include:

- Maps
  - All files must be in pdf format
  - Will need to be separated into some type of grid-like structure
    - Files must have some type of naming convention or file structure to determine their order and level of detail
  - Files will be saved in a Dropbox account and may be occasionally updated

Figure II-1

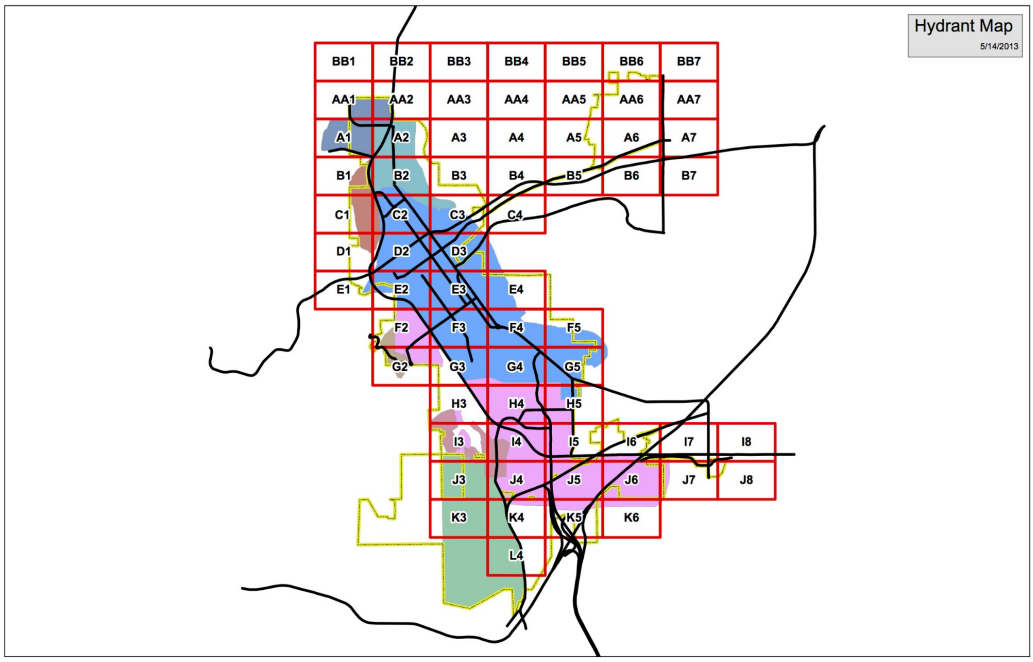


Figure II-1 shows the pdf of the main city grid layout, the first view the user will be shown after starting the app. This will be shown with buttons overlayed where files exist.

Figure II-2

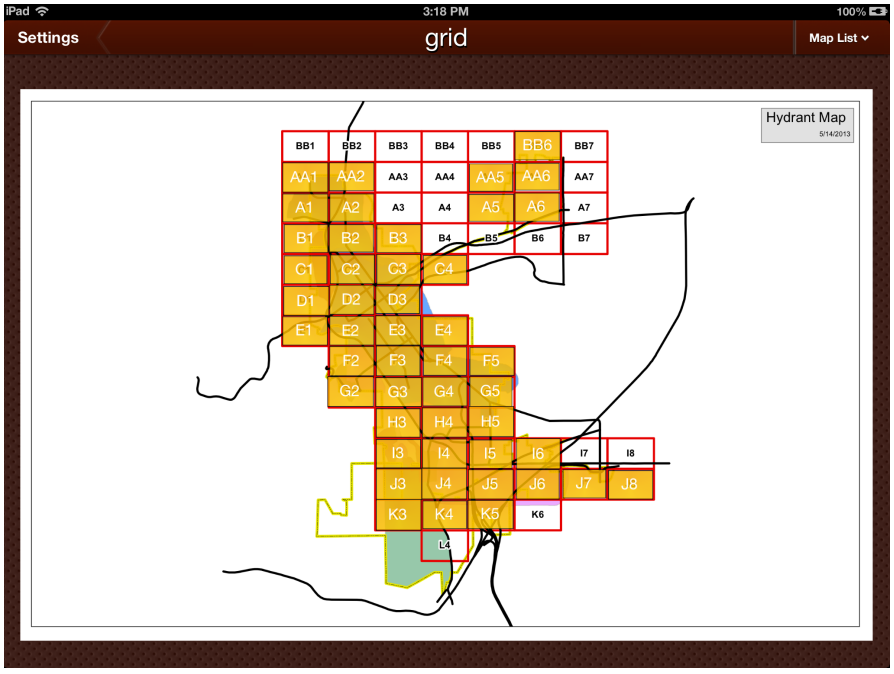


Figure II-2 shows the actual implementation of how the main city grid layout becomes incorporated into the app, along with the buttons overlayed where the files exist, noting that the grid cells without yellow buttons do not yet have files associated with them.

III. System Architecture

Figure III-1

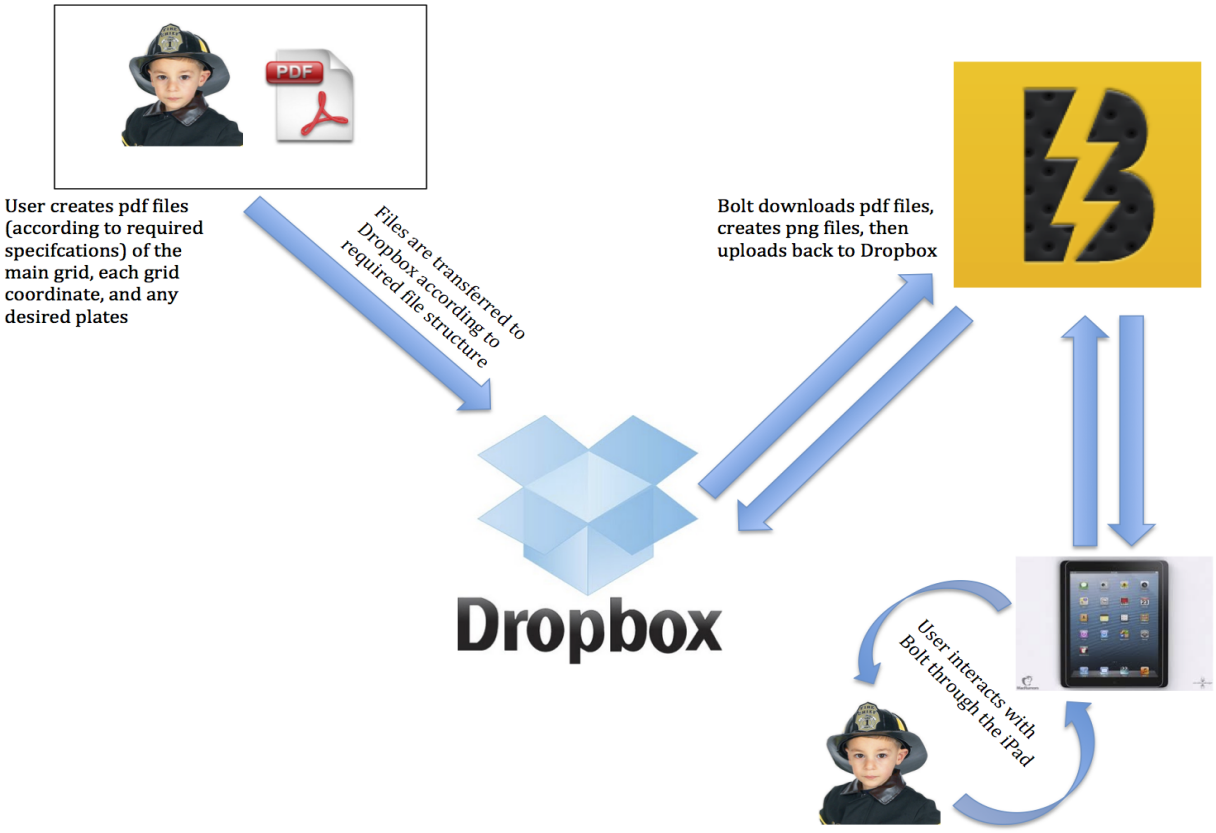


Figure III-1 illustrates the system architecture for what the app requires to function. The first step is for the user to create the pdf documents according to the specification document attached in the appendices. Next, the user must download and install Bolt, then login to Dropbox using the interface provided in Bolt. This creates the necessary file structure. Then the user can load all of the pdf files into Dropbox, reload Bolt, which then downloads all the files onto the iPad. As the user begins to use the app on the iPad, the pdf files will be rendered into png files, which are then uploaded back to Dropbox, so this process only needs to happen once on any iPad. This allows all other iPads to download the already rendered png files for faster operation.

Figure III-2

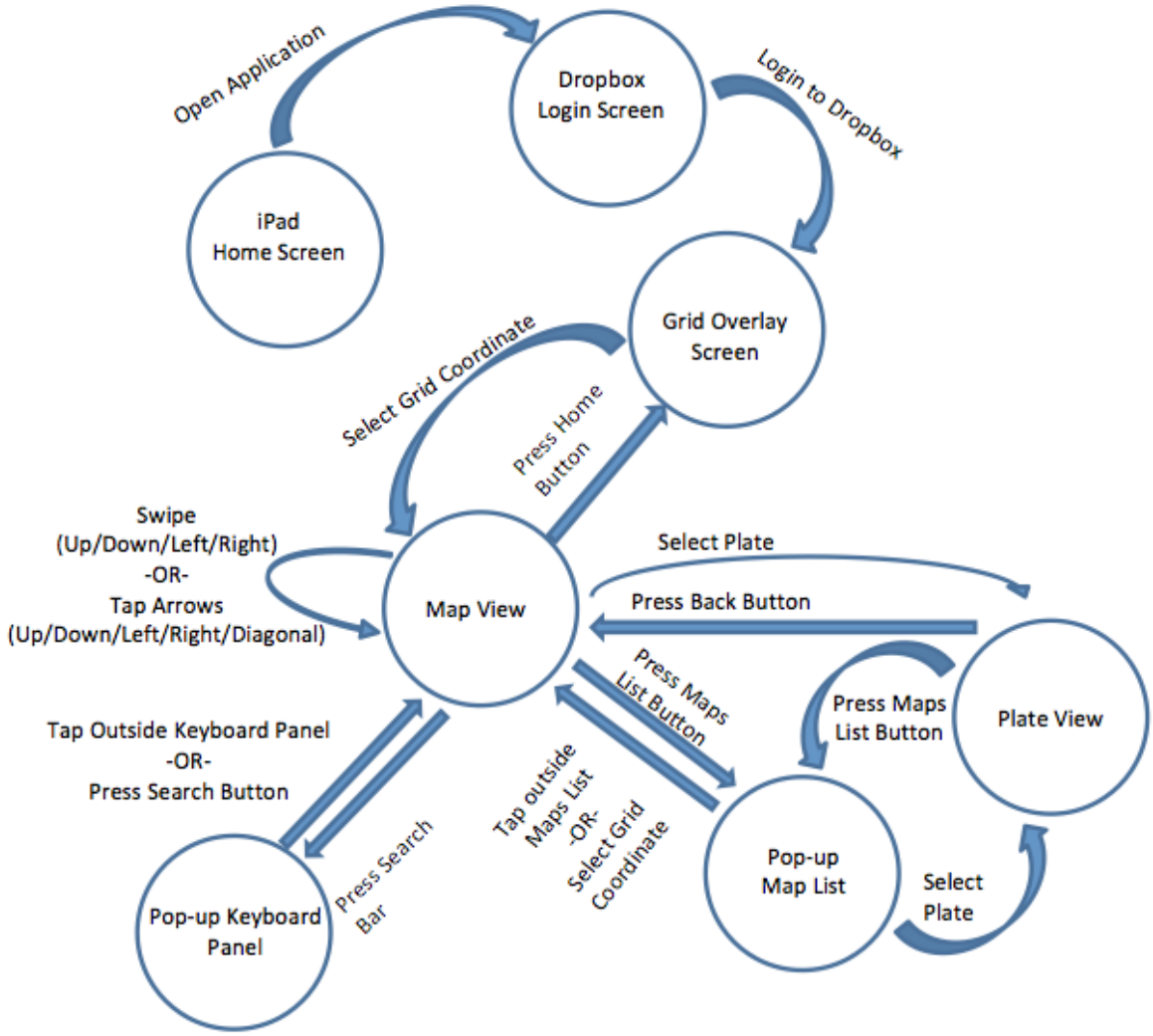


Figure III-2 shows how Bolt will operate after its initial setup has already taken place. In the event the user has not logged out of Dropbox, that login screen will be bypassed (not shown since no action is required) moving directly into the Grid Overlay Screen. From this main screen, the user can select a grid coordinate, which will move into the Map View, providing options for searching, moving into a plate view, or pulling up the maps list and navigating from there.

IV. Technical Design

PARSING PDF DOCUMENTS

Parsing a PDF document for specific information is a very involved process. Our first parsing task was to parse out links contained within the document. These links allow for an area of the document to be clicked in order to open a different document. These links begin their life in the Annotation dictionary, which can be parsed using the "Annots" entry. This dictionary contains a list of annotations, each with a several dozen properties. We iterate through each of the annotations and we begin by parsing a few properties as shown in the tables below which has been taken from the Adobe PDF Reference Document.

Figure IV-1

Rect	rectangle	*(Required)* The *annotation rectangle*, defining the location of the annotation on the page in default user space units.

Figure IV-1: A snippet from the Adobe PDF Reference Document [1, 490]. A row from a table that contains properties that can be found within an Annotation dictionary.

Figure IV-2

A	dictionary	*(Optional; PDF 1.1)* An action to be performed when the annotation is activated (see Section 8.5, "Actions"). ***Note:*** *This entry is not permitted in link annotations if a* **Dest** *entry is present (see "Link Annotations" on page 501). Also note that the* **A** *entry in movie annotations has a different meaning (see "Movie Annotations" on page 510).*

Figure IV-2: A snippet from the Adobe PDF Reference Document [1, 492]. A row from a table that contains properties that can be found within an Annotation dictionary.

The "Rect" property defines a rectangle that the annotation is confined within. We parse this rectangle object farther for properties such as coordinates, height, and width. We then use this information to translate a size and position on the page in which to draw an intractable button.
The "A" property contains another dictionary which contains properties related to the Action of the annotation (if applicable). Since we only want to deal with annotations with actions in this case, if the Action dictionary is not present, then we ignore the annotation and continue our iteration. Once we have the Action dictionary, we parse the Action type property from it as listed in the table below.

Figure IV-3

S	name	*(Required)* The type of action that this dictionary describes; see Table 8.34 on page 518 for specific values.

Figure IV-3: A snippet from the Adobe PDF Reference Document [1, 514]. A row from a table that contains properties that can be found within an Action dictionary.

The "S" property determines what kind of Action is to be performed. The table below lists the Action types. In our case, we only want to deal with Launch type Actions.

Figure IV-4

TABLE 8.34 Action types

ACTION TYPE	DESCRIPTION
GoTo	Go to a destination in the current document.
GoToR	("Go-to remote") Go to a destination in another document.
Launch	Launch an application, usually to open a file.
Thread	Begin reading an article thread.
URI	Resolve a uniform resource identifier.
Sound	*(PDF 1.2)* Play a sound.
Movie	*(PDF 1.2)* Play a movie.
Hide	*(PDF 1.2)* Set an annotation's Hidden flag.
Named	*(PDF 1.2)* Execute an action predefined by the viewer application.
SubmitForm	*(PDF 1.2)* Send data to a uniform resource locator.
ResetForm	*(PDF 1.2)* Set fields to their default values.
ImportData	*(PDF 1.2)* Import field values from a file.
JavaScript	*(PDF 1.3)* Execute a JavaScript script.

Figure IV-4: A snippet from the Adobe PDF Reference Document [1, 518]. A table containing the list of possible values that can be used for the Action type.

The purpose of the Action type property is to determine what other properties are contained within the

Action dictionary. We only continue parsing the Annotation if the Action is a Launch type Action. We then parse the Action dictionary for the property listed below.

Figure IV-5

F	file specification	*(Required if none of the entries **Win**, **Mac**, or **Unix** is present)* The application to be launched or the document to be opened or printed. If this entry is absent and the viewer application does not understand any of the alternative entries, it should do nothing.

Figure IV-5: A snippet from the Adobe PDF Reference Document [1, 521]. A row from a table that contains properties that can be found within an Action dictionary if the Action is a Launch type Action.

The "F" property is a dictionary containing File Specification information. This dictionary contains information about the file that the Launch type Action is instructed to open or print. We parse this dictionary for the file string property as shown in the table below.

Figure IV-6

F	string	*(Required if the **DOS**, **Mac**, and **Unix** entries are all absent)* A file specification string of the form described in Section 3.10.1, "File Specification Strings," or (if the file system is **URL**) a uniform resource locator, as described in Section 3.10.4, "URL Specifications."

Figure IV-6: A snippet from the Adobe PDF Reference Document [1, 122]. A row from a table that contains properties that can be found within a File Specification dictionary.

This "F" property returns a string that is a relative path to the file to be opened. In our case, it may return a string such as "K5.pdf" if parsed from the grid page or "Plate1.pdf" if parsed from a cell page. This string is used to determine the PDF that the button should open when it is pressed.

If the iteration completes, then we now have all of the required information (location, size, and file path) to draw a button on the screen that the user can interact with.

VIEWING PDF DOCUMENTS

We have spent a lot of time optimizing the viewing of PDF files for the best user experience. Originally, we tried using a UIWebView to display PDF files. The purpose of this approach was simplicity. However, we moved away from this approach because of a lack of usability which included

functions like zooming and scrolling. We then found the Quartz framework which allows for 2D objects like PDFs to be rendered on the screen. After implementing this approach, we noticed that the performance of the app greatly suffered; rendering PDFs could take in upwards of 10 seconds to display on the screen. After some thought, we decided to use the Quartz framework to render the PDF only once. After the PDF is rendered, then it can be saved to an image file such as a PNG. This decision was made because rendering an image file can be done exponentially faster than parsing and rendering a PDF file. Now, viewing a PDF file requires first checking if a PNG version of that file exists. If not, then it uses the Quartz framework to render and create the PNG version of that PDF. After the PNG is created, or if it already existed, then we use a UIImageView to render the PNG version of the PDF onto the screen. This can be done in less than a second. Additionally, after the PNG version has been created, we send a request to upload the file to Dropbox. This means that a PDF only has to be rendered one time on one iPad, then it will be downloaded in a PNG version to all other iPads which can then display the file very quickly.

V. Design and Implementation Decisions

TECHNICAL DESIGN DECISIONS

- The use of Dropbox for file storage and synchronization
    - This decision was made for ease of use on the client's side. The client can simply upload one set of files to one central location which can then be used on every iPad that they need it on. As well, different accounts can be used by different people or departments if necessary which can show a completely different set of maps.

- The use of PDF files for maps
    - This was a client-made decision because it is the easiest media for the client to export the maps into. The map-creation software has the option to export to PDF with just a few clicks of the mouse. As well, PDF files can contain additional information that files like images cannot contain.

- The use of .xib files rather than .storyboard files for UI design
    - This decision was made to help facilitate some level of backwards compatibility. Storyboards are not compatible on devices containing a version of iOS earlier than 5. This decision required little work by us now but may have great reward if in the future

people want to use the app on earlier versions of iOS.

- Using a UIScrollView to contain all of the content related to displaying PDFs rather than using a UIWebView
    - This decision was made first because the client wanted the maps to be zoomable. Using this approach better allowed for zooming and panning than a UIWebView could. The second reason was that by creating our own objects for rendering and displaying the PDFs inside the UIScrollView meant that we had greater control over how the content is displayed than a UIWebView could give us.

- Rendering PDF map files as images then saving them to disk before loading them into a UIImageView
    - This decision was made for performance reasons. When rendering a PDF file, it can take in upwards of 10 seconds. Because the app is designed to be used in emergency situations, it is important that it reacts quickly to user interaction. Waiting 10 seconds to view a grid or a grid cell was not feasible. By rendering the files as PNGs, this wait time was improved to under one second. The PDF files are still parsed in the background for information such as text for searching and links for button creation.

- Decision to parse the content from the links within the PDF file to create overlay buttons as opposed to drawing a custom grid
    - One reason that this decision was made because it allows the user to see which grid locations are available and which grid locations are not. Another reason is that parsing links brings additional functionality: it allows the client to create their own grid file which can contain any background that they may want to use, and it also allows the client to create a link wherever they want which can take them to a file that contains whatever they want. This also meant that we did not need additional code for the grid file and for plates. This same parsing works in both situations.

- The use of buttons to navigate between maps in addition to swipe gestures
    - This was decided for ease of use for the UI and to give users multiple options so that they can determine which functionality they like better.

- Active searching for faster operation
    - Because searching the document takes less than a second to perform, active searching (searching while typing) is used to speed up the display of the address location to make the app more responsive to this user interaction.

VI. Results

The purpose of this project was to design an iPad app for the Golden Fire Department to replace the large binders used to find the exact location of an address and the closest fire hydrant. The main requirement for the app was to establish a faster, more efficient method of navigating through the maps used by the firefighters. To accomplish this, the app needed a main grid-layout page with clickable buttons for each grid location that can then pull up a more detailed map of the grid. It also needed the ability to display a "plate" document by touching the name of the plate on the screen. A search feature was also desired, such that when the number of a house is typed in the search bar, the location of that number is highlighted for faster locating. All of these requirements have been met. When opening the app (after its initial setup), the app opens to the main city grid page with a layout of all the cells. The app then displays buttons in each cell where a file exists, thereby preventing the illusion of any buttons if no file has been loaded into Dropbox. This main page also includes a "Settings" button to log in/out of Dropbox and another "Maps List" button that shows the full list of available maps and plates. Tapping any cell where a button is shown will cause the screen to display the detailed map associated with that grid location. When viewing a map, a search bar becomes visible that the user can tap on to display the keyboard, then they can type in the number of the address they are looking for and have it instantly highlighted on screen. Additionally, buttons are added for easier navigation, such as a "Back" button as well as directional buttons to move to adjacent grid cells. There was one last feature that was not implemented due to time and complexity constraints. This was the ability to add notes to certain grids and having those notes sync across the fire department's iPads. While this feature may not have been implemented as desired, the methods for parsing the pdf documents allows the user to create their own links inside of any pdf document, then attach that to any other pdf document, effectively creating the same ability to add notes, even if through a slightly more complicated process. Ultimately, the apps works better than the client had expected upon final delivery and has implemented more subtle features than originally requested, all to the delight of the client.

VII. Appendices

Appendix A: Bolt Installation Instructions

Installation Instructions

1) Install iPad™ Application

First step is to download and install the “Bolt” application onto 1 iPad.
Open the application on that iPad and log into your Dropbox™ account you plan on using with Bolt.
This will automatically create the correct folder structure in Dropbox that Bolt needs to locate your maps (/Apps/Bolt/Maps).

2) Prepare your Map Files

For Bolt to work properly, PDF files need to be prepared in advance by naming them in a specific way and creating links in the PDF files that the Bolt app will use to create the navigation.

Naming Files

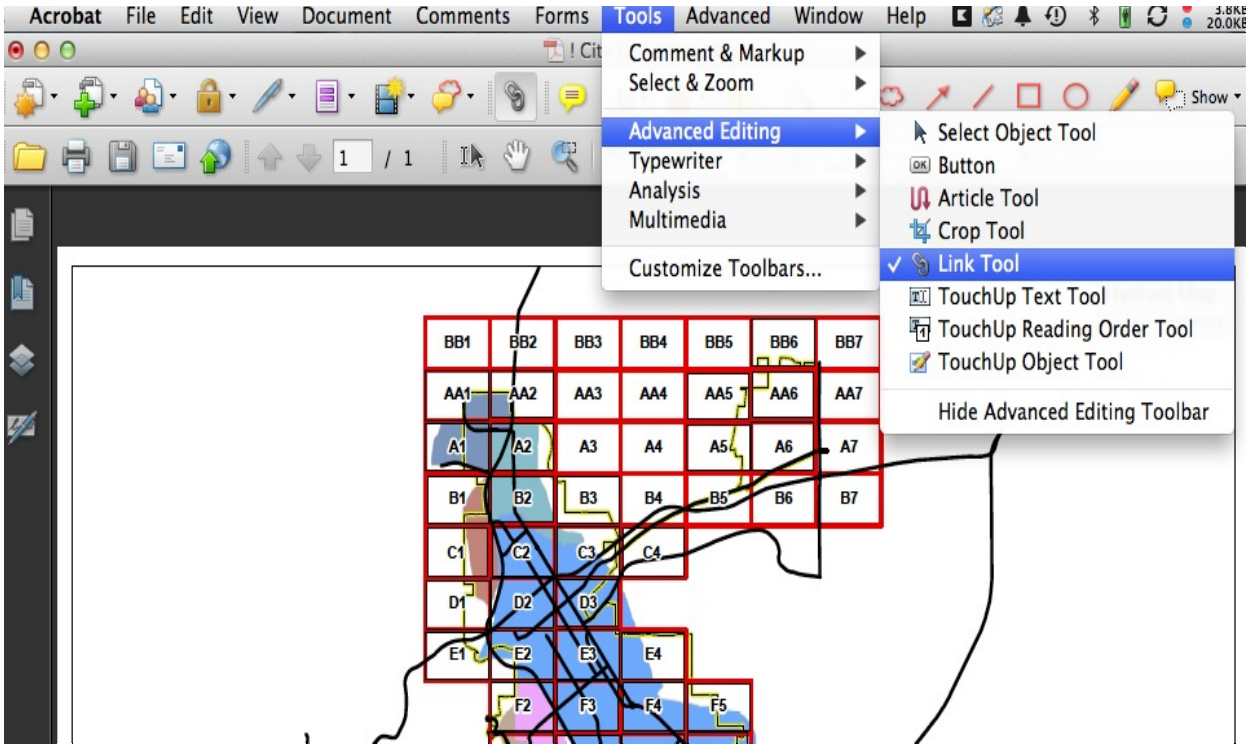
Bolt makes an assumption the map pages follow a “grid” name sequence (eg: A1, A2, B1, B2, etc).
Naming files in this type of grid sequence will allow them to fall logically in the file list (eg: A1.pdf B2.pdf).
Start by organizing PDFs in a single folder called “Maps” and naming them in a similar manner to the example above. The letter needs to be before the number (ie: D5.pdf will work but 5D.pdf will not). The application assumes low numbers are to the left of high numbers and the alphabet goes from top to bottom. The exception here is AA will be above A. Numbers less than 0 (zero) will not work.

Creating Links

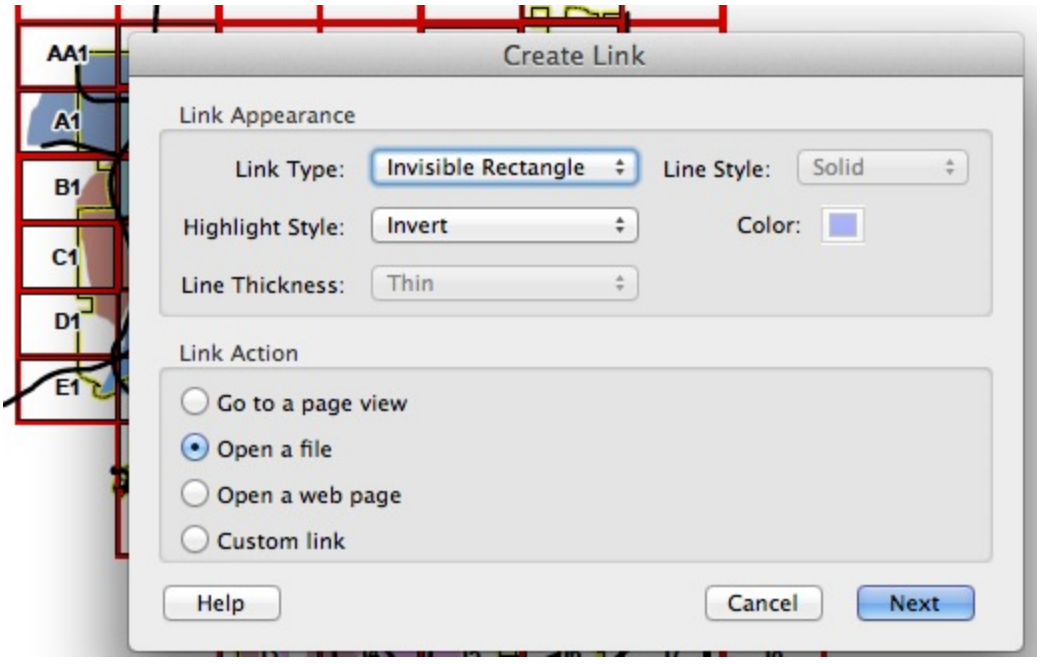
Next, hyperlinks need to be created in the PDF files. This is what creates the navigation logic that Bolt will use.

In this example, we start with a grid overlay of the city. You see a grid that makes up the cities map pages. This grid file needs to be called “**grid.pdf**” when you save it and import it into Dropbox. Adobe Acrobat Pro is used for this example to build links from this grid overlay to the specific map pages.

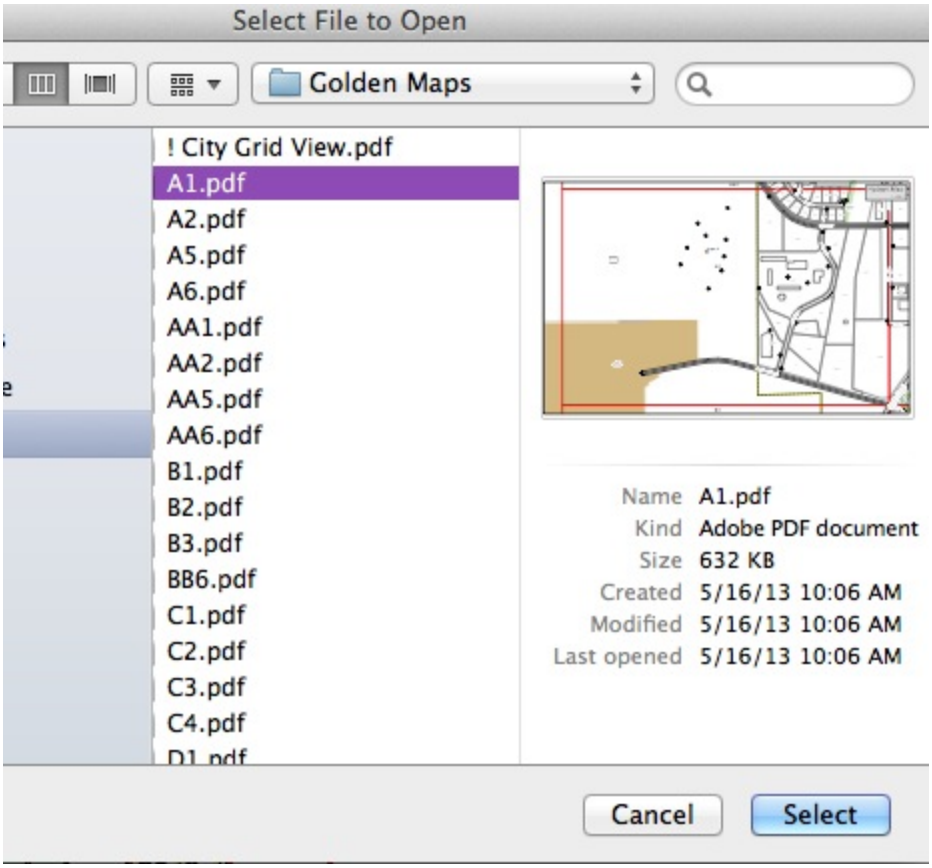
Open the grid file in Adobe Acrobat Pro and select the “Link” tool:



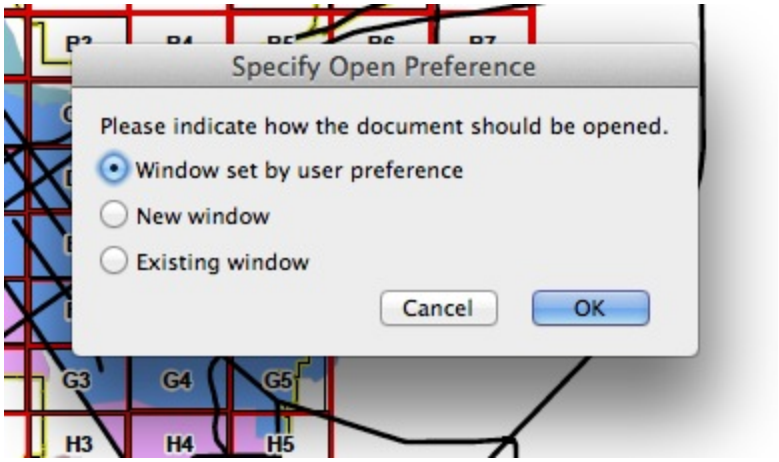
With the Link Tool active, draw a box around one of your grid squares. We'll use A1 for this example:



As soon as you complete your box, this dialogue will appear. Make sure "Open a file" is selected. Click "Next" and navigate into your "Maps" folder to select the correct map detail page:



Select the correct PDF file.



You may use any of these options and then click OK.

You just built a hyperlink in a PDF file. Repeat these steps for every map grid page you need to. Remember to save this file as "**grid.pdf**" when you are done.

Additional Map Detail Pages

Let's say you have additional map pages with additional details. Perhaps a building, a golf course, or a

section of the city that is too dense with buildings, etc for the larger map view to show. You can use a similar process to the steps above to link to further details.

Let's say that map page C5 has an office park in it with lots of smaller buildings, each with their own address. The C5 map page is too wide to view these details so we have a smaller map page detailing just that office park. In our example, we call that smaller map a "Plate" though the name does not matter (you may call it C5A for example).

Open up C5 in Adobe Acrobat Pro. Select the Link Tool and draw a box around the office park. Link that to your detail file (eg: Plate1.pdf) just as you did with the map pages above. Make sure all your map pages are saved in the root "Maps" folder.

In Bolt, this will now create a clickable link on the map page to that detail page. Repeat those steps for all the additional detail pages you have.

Getting files into Bolt

Once you have completed these steps for all your map pages and all your PDF files are saved in a folder on your computer, you are ready to upload them to your Dropbox account and sync them to Bolt.

Open your Dropbox account with a browser. Navigate into: Apps/Bolt/Maps.
Upload all of the map PDF files you have created into this "Maps" folder in Dropbox.

Now when you re-load the Bolt application on your iPad, it will look for any changes, new maps, etc in this folder and sync them to the iPad.

Note: The first time you open each map page on the iPad, it will take slightly longer while it automatically creates an image file of that PDF. Subsequent loads of that same map page will be very quick.

Adding Additional iPads

On any additional iPads you want to add Bolt, simply download the application and enter your Dropbox credentials. All the map files will automatically sync.

Updating or adding map pages

To update any existing map pages, simply upload the new version to the Dropbox "Maps" folder. Make sure it keeps the same exact name as the file you are replacing. This new file will automatically sync to all your iPads next time you launch Bolt.

To add new map pages, first create the link to the page on the "grid.pdf" page and then upload your new map page along with the updated grid.pdf.

To add new detailed pages, create a link from the map page to the detail page. Upload the new detail page along with the updated map page to Dropbox. It will automatically sync changes next time you launch Bolt on each iPad.

Appendix B: Bibliography

[1] *PDF Reference*. 3rd ed. Addison-Wesley, 2001. Print. <http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/pdf_reference_archives/PDFReference.pdf>.