iNotes

Overview

This document outlines in detail how the app looks, how it is used, and how it communicates with the content server.

Table of Contents

User Interface

This section outlines the user interface components of iNotes. It has four main views: **Splash Screen**, **Home Screen**, **Slide View**, **Glossary View**. Each are described in detail in the following sections. In this document, main views are notated in **bold**, and user interface elements are in *italics*.

Splash Screen

The Splash Screen is a static image that contains the following elements:

1. sponsor recognition
2. splash background image, which is loaded from the iNotes server.

Home Screen

This is the first view the user interacts with. It is modeled similar to a document library where each piece is a document.

1. *Program Notes* Button
    a. This is a button that when pressed shows a dropdown through which the user can explore a pdf of the program notes. These are dynamically loaded from the iNotes server for each concert.
    b. See Appendix A for the location of this XML file.

2. *Concert Listing*: The app lists each piece as a document. Users can swipe to scroll through documents if there are more available than fit on the screen.
    a. Movement listing: Each *super-documen*t when tapped displays the pieces movements, or *sub-documents*. All movements are stored as individual pieces. The app dynamically groups movements with similar titles.
    b. When pieces are they are tapped, the application navigates to the **Slide View** and shows the first annotation slide of the selected movement.
        i. On startup, the app calls an HTTP request to: `your.server.org/getPieces.php`
        ii. The script returns list of current pieces formatted as: `Composer, Piece: Mvt. #` with multiple pieces and movements separated by semi-colons.
        iii. The app then contacts server to get XML files related to current pieces and loads it as well as loads stored documents that contain previously downloaded content. This script is: `your.server.org/getData.php`
        iv. See Appendix A for use of PHP and and Appendix B for XML.

    c. "live" indicator
        i. This shows which piece and movement is live.
        ii. It is also a button that navigates to the live position similar to the *View Live* Button. This takes the (See *View Live* button)
    d. deleting concerts

   i. Concerts are kept as documents. Need the ability to delete stored documents.
   ii. Gesture swipe up to delete or best UX for delete.
3. Home Interaction
   a. *Glossary*: The *glossary* button takes the user to the **glossary view**. (See Glossary)
   b. *Brightness*: The *brightness* button allows the user to adjust the screen brightness. By default, the app forces the phone to have a 50% brightness setting. The concert hall is dark, and this prevents really bright phones from distracting other concert goers. Slider object to control brightness preferred.
   c. *help*: The help button presents an overlay above controls showing their functionality. This is dynamically loaded. See Appendix A for more information.

   d. *Info*: The *info* button presents a view that shows sponsors and affiliated parties. The `your.server.org` address in the bottom right is a *clickable link* that will open in the devices web browser.

4. The *navigation bar* should say “Home”.
   a. *Refresh* button: The *refresh* button appears on the left side of the navigation bar. It spins when content is loading, when content is done loading, it stops spinning. The refresh button performs the following functions:
      i. re-downloads all content
      ii. re-connects to multicast stream
   b. *View Live*: On the right side of the navigation bar is a button that says “View Live.” When tapped, the app navigates to the slide view and shows the current annotation.
      i. This button navigates to the current position being sent via the multicast network in the **Slide View**.
      ii. This button exists in both the **Home Screen** and the **Slide View**.
      iii. It also re-connects to the multicast stream each time it is pressed. This ensures a good connection, always.
      iv. This functionality is duplicated in the *live indicator*.

Slide View

The **Slide View** is the second main view of the app. It shows annotation content about the piece as it becomes contextually relevant in real-time.

1. *Navigation Bar*:
   a. *Concert Title*: This is the title text in the navigation bar. It should be formatted as: `Composer, Piece: Mvt. #`
   b. *Back Button*: A simple *back button* in the navigation bar takes the user back to the **Home Screen**.

c. *View Live*: This is a button in the navigation bar that is red and says "Live" when users are not currently viewing the live position or green and says "Live" if they are viewing the live position.
    i. This button navigates to the current position being sent via the multicast network.
    ii. It exists in both the **Home Screen** and the **Slide View**.
    iii. It also re-connects to multicast stream each time it is pressed. This ensures a good connection, always.

2. *Slide View Content*
    a. Downloaded content is received on startup via an XML string received through a php script.
    b. The app deconstructs the XML, and inserts the content into a set of HTML template *slides*.
    c. Each *Slide* is shown in a simple, stripped down webview.
    d. This allows pictures and other dynamic content to be displayed.
    e. *Glossary Links:* The text is compared to words available in the glossary. Words found as matches are colored yellow and are links to their definition in the **Glossary View**.
3. Content Interaction
    a. *Slide Number*: A simple indicator shows what *slide* the user is viewing on in the current track. Example: **5 of 17**
    b. *Track Selection*: when the callout icon, or *Track Selection* button, pressed, a pop out appears showing all possible annotation tracks/channels.
        i. Users can switch between which track they want to be viewing.
        ii. Tracks can be things like "History", "Composers Notes", "Theory Analysis", "Narration", etc.

    c. *Font Size*: users can adjust the font size desired.
    d. *Brightness*: users can toggle brightness
    e. *Help*: When pressed an overlay appears, showing the general functionality of all controls in the slide view.

4. *Slide Timeline*
    a. *Waveform*: placed in the visible part of the structure drawer with the color analysis as it's background. *Note: slides are liner via measure number. Measure number is not linear via waveform. an upcoming infrastructure update will enable the waveform to be drawn based on intensity measurements available in the iNotes CMS database.
    b. *Playhead Bar*: A simple bar scrolls across the visible part of the structure drawer with the color analysis as it's background.
    c. *Slide Milestone*: At the edge of the drawer is a bar that shows dots over time that indicate where the slides transition.
    d. When touched, it expands to show the *Structure Drawer*.

5. *Structure Drawer*
    a. When the *Slide Timeline* is touched, it expands to show the color analysis in a larger view.
    b. Each section of the piece is color coded, and when expanded, the *Structure Drawer* shows a text label of that section as well movement/analysis description.
    c. Similar sections are similar colors, giving a simple view of where similar themes occur in the piece.
    d. key signature
    e. Color, key signature and structure are defined in the iNotes CMS. *Note: Key signature will require an upcoming update to the CMS.

Glossary View

The **Glossary View** is the fourth main view of the app. It shows a list of words that might be useful for users to understand. The glossary words and their definitions are loaded dynamically through `glossary.xml`. This is outlined in Appendix A and Appendix B.

1. *Words TableView*: On the left there is a tableview displaying words loaded from the XML.
    a. The table view has subheadings that correspond to the “category” tag in the XML.
    b. When tapped the words definition appears in the *Definition Viewer* on the right. This Includes the word and possible subheadings, like “History”.
    c. The currently selected word is highlighted in light blue.
2. *Definition Viewer*: This view displays the definition of the selected word.
3. *Link Clicked:* When a link is clicked in the **Slide View**. The **Glossary View** is pushed and the tapped word is highlighted in the *Words TableView* and the definition is shown in the *Definition Viewer*.

Modes of Operation

Online Mode: iNotes SSID

When the app connected to the “livenote” SSID, it performs a unicast download of content as an xml string from a php script. A set of scripts is called to retrieve the content. Appendix A describes these php scripts.

1. Appendix B outlines this XML string.
2. The each XML string is a *Concert* that consists of *Pieces*.
    a. `Name`: name of piece
    b. `Number of Measures`
    c. `Track`: contain the annotation information. A `piece` can contain more than one `track`.
        i. `Name`: name of annotation track
        ii. `Page`: page of the annotation track
            1. `Time`: time of presentation in relation to a reference recording
            2. `Measure`: time of presentation based on a measure number
            3. `Text`: text associated with the annotation.

When on the “livenote” SSID the app joins a multicast group in order to receive the Multicast Measure Stream.

1. Joining Multicast Address
    a. This is done when the app launches.
    b. Everytime *View Live* or *Refresh* is pressed, the app reconnects
    c. When the multicast is joined, present a message that says “Live Stream Connected” on success, or “Live Stream Error”
    d. Current Measure Packet: This tells the app what the current “live” measure is.
    e. Push Notification Packet: This tells the app to show the audience a notification.
    f. Appendix C outlines the IP and Port of the multicast group and the packets received from the multicast stream.

Offline Mode: Wifi or 4G connected

Users can view all locally stored concert documents once the concert is over.

1. location of locally stored docs
    a. Downloaded content is sandboxed in the application for later viewing. In iOS it exists in the documents directory.
    b. Each piece is a folder named `POAComposerPieceMvt#`. This is basically the same structure as the piece name used previously, but with no non-alphanumeric characters, and a POA tacked onto the front. This folder includes the following.
        i. data.xml - xml annotation data of piece downloaded previously from php script
        ii. associated image files

iii. images.txt - a concatenated list of image names, separated by a ‘;’
iv. name.txt - the actual full name of the piece.
v. The Don Juan folder looks as follows: POAStraussDonJuanDonJuan
    1. data.xml
    2. images.txt
    3. name.txt
    4. DJ1.png
    5. DJ2.png
    6. DJ3.png
    7. DJ4.png
    8. DJ5.png
    9. DJ6.png
    10. DJ7.png

c. These documents get saved when the app content is downloaded at the concert.

2. android file system
3. *Deleting Content*: currently, the app allows users to view downloaded saved content through iTunes document sharing. They can delete it easily through there, but most people don’t know how to do that.

Modals and Alerting

Offline Mode Modal - Displayed when user IS NOT connected to the iNotes SSID. Also displays a message when the user tries to “view live”, when not at at live concert.

Online Connected Modal - Displayed when user IS connected to iNotes SSID and is receiving the multicast stream. When user presses “View Live” and reconnects, a simplified version of this message appears

CMS Custom Modal- In the CMS a push notification packet can be sent.

1. This packet includes a Title, Message, and Button Text.
2. A simple custom pop up message appears when this packet is received. Its analogous to a one way chat.
3. The messages may include all components (title, message, button text), or only a subset. If there is no button text, present the pop-up without a button and dismiss it after a set time. The iOS app dismisses it after 4 seconds.
4. The packet definition can be found in Appendix C.

CMS Custom LINK Modal: this is similar to the CMS generated modal alert.

1. The title will be “Link”.
2. However, an external web link is placed where the button text should be.
3. Buttons to ignore or navigate to that link are presented in the alert pop-up.
4. When they press the “take me there” button, they are taken out of the app to their phone’s default web browser that navigates to the link.
5. Easy way to implement more info, surveys, etc.
6. The packet definition can be found in Appendix C.

Appendix A

Content Retrieval

This appendix outlines the usage of the php scripts and retrieving content from the server.

Dynamic Image Assets

A large number of the assets in the app are dynamically loaded from the content server. This allows things like the Program Notes pdf, the glossary, and certain images to be customized by the orchestra for each performance. These images are located at the following url:

```
http://your.server.org/images/AppAssets/
```

The images include:

`ProgramNotes.pdf`	- Updated program notes file.(program notes button)
`AlbumHelp32.png`	- iPhone 4 Home Screen help overlay (help button)
`AlbumHelp169.png`	- iPhone 5 Home Screen help overlay (help button)
`ConductorFit32.png`	- iPhone 4 Home Screen Background
`ConductorFit169.png`	- iPhone 5 Home Screen Background
`InfoScreen32.png`	- iPhone 4 Home Screen info view (info button)
`InfoScreen169.png`	- iPhone 5 Home Screen info view (info button)
`SlideViewHelp32.png`	- iPhone 4 Slide View help overlay (help button)
`SlideViewHelp169.png`	- iPhone 5 Slide View help overlay (help button)
`Splash32.png`	- iPhone 4 splash screen
`Splash169.png`	- iPhone 5 splash screen
`SplashCut32.png`	- iPhone 4 splash screen trimmed for title bar
`SplashCut169.png`	- iPhone 4 splash screen trimmed for title bar
`glossary.xml`	- XML of glossary Definitions

Loading App Content From the Server

`getPieces.php`

The `getPieces.php` function retrieves the pieces that are made available for the current concert. It is called using a URL request:

```
http://your.server.org/getPieces.php
```

This returns the names of the pieces being performed, separated by a semicolon. An example of this output is shown below:

```
Higdon, Concerto For Orchestra: Mvt. 1;Higdon, Concerto For Orchestra: Mvt. 2;Higdon,
Concerto For Orchestra: Mvt. 3;Higdon, Concerto For Orchestra: Mvt. 4;Higdon,
Concerto For Orchestra: Mvt. 5;Strauss, Don Juan: Don Juan
```

getData.php

The `getData.php` function retrieves the XML data described in Appendix B associated with the pieces that are made available for the current concert. The script is passed the piece name as an instance variable. It is called using a URL request:

```
http://your.server.org/getData.php?database1=Strauss,Don%20Juan:%20Don%20Juan
```

You can also access more than one database at a time:

```
http://your.server.org/getData.php?database1=Higdon,%20Concerto%20For%20Orchestra:%20
Mvt.%201&database2=Higdon,%20Concerto%20For%20Orchestra:%20Mvt.%202&database3=Higdon,
%20Concerto%20For%20Orchestra:%20Mvt.%203&database4=Higdon,%20Concerto%20For%20Orches
tra:%20Mvt.%204&database5=Higdon,%20Concerto%20For%20Orchestra:%20Mvt.%205&database6=
Strauss,%20Don%20Juan:%20Don%20Juan
```

image.png

In the XML received, all images are denoted by `$` and a caption by a comma, similar to `$image.png, image caption.` These images are retrieved by accessing their location directly on the server.

```
http://your.server.org/images/image.png
```

Appendix B

Content XML Definition

In this appendix, we outline the definition of the XML file downloaded from `getData.php`. There is a complete example included as an attachment with this document. We also outline the XML for the glossary data.

Content from `getData.php`

```
<?xml version="1.0"?>
<concert>
    <piece>
        <name>{The Name of the Piece}</name>
        <measures>{number of total measures}</measures>
        <track>
            <name>NumSeconds</name><!--List Measure #s and timings in a 'track'-->
            <page>
                <time>{time of first measure in reference recording}</time>
                <measure>1</measure>
                <text>{time of first measure in reference recording}</text>
            </page>
            <page>
                <time>{time of second measure in reference recording}</time>
                <measure>2</measure>
                <text>{time of second measure in reference recording}</text>
            </page>
            <!--   . . .    -->
            <page>
                <time>{time of last measure in reference recording}</time>
                <measure>{Last Measure #}</measure>
                <text>{time of last measure in reference recording}</text>
            </page>
        </track>

        <!--One of possibly multiple annotation tracks-->
        <track>
            <name>The Music</name>
            <page>
                <time>{time of annotation location in reference recording}</time>
                <measure>{measure the annotation corresponds to}</measure>
                <text>Actual annotation text.</text>
            </page>
            <page>
                <time>{time of annotation location in reference recording}</time>
                <measure>{measure the annotation corresponds to}</measure>
                <text>This is an annotation with an image. $image.jpg, image caption </text>
            </page>
            <!--Now an example-->
            <page>
                <time>77</time>
                <measure>49</measure>
```

```
            <text>A return of the Don's themes creates a transition to the secondary theme. DJ2.png,
            Don Juan Portrait. </text>
        </page>
        <!--   . . .    -->
        <page>
            <time>{time of annotation location in reference recording}</time>
            <measure>{measure the annotation corresponds to}</measure>
            <text>Actual annotation text.</text>
        </page>
    </track>
    <track>
        <name>Structure</name> <!--Structure annotations for use in the structure drawer view-->
        <page>
            <time>{time of structure location in reference recording}</time>
            <measure>{measure the structure corresponds to}</measure>
            <text>{Structure Text},{structure color tag}</text><!--accepts color tag-->
        </page>
        <!--   . . .    -->
        <page>
            <time>{time of structure location in reference recording}</time>
            <measure>{measure the structure corresponds to}</measure>
            <text>{Structure Text},{Red},{Green},{Blue}</text><!--also accepts RGB vals,{0-255}-->
        </page>
        <!--Now an example -->
        <page>
            <time>{420}</time>
            <measure>{345}</measure>
            <text>The Main Theme,255,127,50</text><!--Orange-->
        </page>
        <page>
            <time>{time of piece end}</time>
            <measure>{measure of piece end}</measure>
            <text>END</text><!--Needed to tell the view where last structure block terminates-->
        </page>
    </track>
</piece>
</concert>
```

Definition Information from `glossary.xml`

```
<?xml version="1.0"?>
<glossary>
<!--Outline of format-->
    <item name="{Word to define}" category="{Category of word}">
        <description>
            {Definition of word}
        </description>
        <history>
            {historical context of word (optional)}
        </history>
        <image/>
    </item>
    <!--Now a few examples-->
    <item name="allegro" category="Tempo">
        <description>
                In a quick, lively tempo, usually considered to be faster than allegretto but slower than
                presto. Used chiefly as a direction.
```

```
        </description>
        <history/>
        <image/>
    </item>
    <item name="arco pizzicato" category="Style">
        <description>
                In music notation, a composer will normally indicate the performer should use pizzicato
                with the abbreviation pizz. A return to bowing is indicated by the Italian term arco.
        </description>
        <history/>
        <image/>
    </item>
    <item name="attacca" category="Music Theory">
        <description>
                Attack at once; -- a direction at the end of a movement to show that the next is to follow
                immediately, without any pause.
        </description>
        <history/>
        <image/>
    </item>
    <item name="bassoon" category="Instruments">
        <description>
                A low-pitched woodwind instrument with a double reed, having a long wooden body attached
                to a U-shaped lateral tube that leads to the mouthpiece. The range of this instrument is
                typically two octaves lower than that of the oboe.
        </description>
        <history>
                The bassoon developed from the older curtal (or dulzian) in the 17th century. An agile
                instrument with a mild tone, it has a range of 3 1/2 octaves, starting at B-flat two
                octaves below middle C.
        </history>
        <image/>
    </item>
<glossary>
```

Appendix C

Multicast Definition

The multicast network group address is: `239.255.255.251`
The port number is: `12345`

There are two types of multicast packets. The first is a general measure update packet. It is formatted as follows:

Measure Update Packet

The Measure update packet has a listing of the piece name, the current 'live' measure number, and a tag representing its count in the packets sent by the server. The piece name is formatted exactly the same as the one obtained by `getPieces.php` and used by `getData.xml`. This packet definition is outlined below:

```
{Composer}, {Piece}: {Mvt}; {MeasureNumber} : {sentPacketCount}
```

Below are a couple example packets:

```
Strauss, Don Juan: Don Juan; 35 : 568
```

```
Higdon, Concerto for Orchestra: Mvt 3; 23 : 1351
```

Push Message Packet

One out of every 10 or so packets sent is the push notification packet. This packet outlines the information to be presented in the current push notification. Once this is received by the client, it is their responsibility to not show a duplicate. There are also two packet types. "BUTTON" messages show a button containing the text that follows. "AUTO" shows no button and dismisses after a set duration. The message is outlined below:

```
PUSH|{MessageBoxTitle}|{MessageBoxBodyText}|{Button Style}|{MessageBoxButtonText}|
```

If the title of the message is "Link", the message contains a link to an webpage outside of the app. The link URL is placed where the button text would normally be in the packet. Three example packets denoting "AUTO", "BUTTON", and "Link" messages are shown below:

```
PUSH|Welcome|Please silence your cell Phones. Enjoy the concert.|BUTTON|Will do!|
```

```
PUSH|iNotes|The live stream has begun!|AUTO||
```

```
PUSH|Link|Please take a brief survey.|Button|http://www.survey.com/philorch|
```

Appendix D

Network Description

1. A **LAMP Server** hosts all the content in a MYSQL Database that contains the tables for the iNotes CMS. This server also runs the multicast server program that sends the current measure updates to the phones.
2. When users enter the concert hall. They open the app, and it immediately connects to the content server. Through a series of **http requests** and **php scripts**, it gets information about the pieces for the current concert and the annotations associated with them. If there are images in the annotations, they are sourced from the server as well.
3. Externally, there is **server control program** that updates the current live measure position. This can be done manually through a web interface, or automatically via a custom audio analysis program that follows along with the music.
4. The server sends this current position to all of the devices that joined the multicast group address. The displayed **content is dynamically updated** based on this current position. The live position is notated in both the “Home” view as well as the “Slide View”. When the app receives a push message packet, an alert message is generated and displayed immediately.