**Technical Documentation**

**Author: Ben Cresitello-Dittmar**

**This documentation specifies the features implemented by Ben Cresitello-Dittmar in the 3 different apps written for Save The Children. The first is the sponsor app and is found in the SaveTheChildrenMobile repo under children. The second is the country office manager / admin site and located under / country\_office of the SaveTheChildrenMobile app. The third is the backend server that handles push notifications and some database queries.**

**Admin / Country Office Site:**

**Feature: Multiple User Classes**

Architecture:

The architecture of multiple user classes is as follows. The admin / country office app asks the server for the index.html file. There are 3 possible responses based on an authentication token stored in request cookies.

1. If an authentication cookie is not present, the server responds with the index.html file in the /public/auth folder. This is the folder that contains all routes and JavaScript necessary for authentication and account creation.
2. If the token is set, it determines if the token is authentic using the firebase.auth().verifyIdToken() function. If it is authentic, it queries the firebase database for the users UID and the corresponding permissions.
   1. A permission level of ‘admin’ renders the /public/app-admin index.html file.
   2. A permission level of ‘office’ renders the /public/app-office index.html file.
   3. If no authentication level is found, the user is routed back to the /public/auth directory to log in
3. If the token is set but determined to be unauthentic, the user is routed to /public/auth to authenticate.

This checks are performed any time the page is refreshed, therefore, if the token expires the user will be routed back to the authentication page.

List of Functions:

isAuthentic() : /routes/routes.js

This function determines if a supplied firebase token is authenticated. It uses the firebase.auth().verifyIdToken(TOKEN) function in the firebase admin package.

AuthLevel() : /routes/routes.js

This function determines the authentication level for a given UID. This UID is the account identifier within the firebase user object. It queries the /Accounts reference in the database for all accounts. This is a list of UID’s and then their associated permission levels. The current auth levels are ‘admin’ and ‘office’. Admins have the ability to send push notifications. Offices have the ability to view and print messages for a given child (associated with their office location).

External Modules / Services:

Node Modules:

cookie-parser

firebase-admin

Firebase accounts in the database as follows:

Accounts

UID

permissions: PERMISSION\_LEVEL

Improvements / Vulnerabilities:

This feature is currently susceptible to XSRF (cross site request forgery). A XSRF token needs to be implemented.

Environment:

A properly formatted firebase database of accounts with permission levels.

**Feature: Send Push Notifications**

Architecture:

This feature is front-end. It allows a user with admin credentials to send a push notification through firebase cloud messaging. The notifications are send using the following ‘data’ object

{

token: AUTHENTICATION\_TOKEN,

message: MESSAGE\_CONTENT,

regions: [REGION\_1, REGION\_2],

title: MESSAGE\_TITLE,

body: MESSAGE\_BODY

}

\*\* NOTE: message and body are both currently the same data. In the push body is never \*\*

The user is required to enter the ‘Title’ ‘Notification Message’ and ‘Enter Region’ fields. Upon clicking ‘SEND MESSAGE TO SERVER’. The controller located in /public/components/pushNotifications and creates the specified ‘data’ object specified above and makes a POST request to <https://save-the-children.herokuapp.com/post_notification>.

Upon receiving a response from the POST, the page will generate a dialog box notifying the user of if the message was successfully sent or not.

List of Functions:

$scope.emitMessage : /public/components/pushNotifications/pushNotificationsController.js

This function posts a message to the backend server. The $scope.message, $scope.push\_region, and $scope.title variables must be set. The push region currently supports only one option, however, the server can handle multiple regions. Changing the single region in the array to multiple will send the message to multiple regions.

$showAlert : /public/components/pushNotifications/pushNotificationsController.js

This function shows an alert message on the page. It uses the angular-material module to display a popup with the given title and content.

External Modules / Servies:

Node Modules:

angular-material

Improvements / Vulnerabilities:

NONE

Environment:

NONE

**Feature: PDF Preview**

Architecture:

Upon selecting a PDF file in the file selector, the user can click ‘PREVIEW PDF’. This will call a set of functions that renders the PDF to the canvas HTML5 element. It uses the Mozilla PDF.js module to do this.

List of Functions:

base64ToUint8Array(BASE64) : /public/components/pdfUpload/pdfUploadController.js

This is a helper function to preview the pdf. It converts the base64 encoded pdf data to a uint8 array. This is needed for the canvas to render the pdf file.

$scope.renderNext() : /public/components/pdfUpload/pdfUploadController.js

This is a helper function that renders the next page of the pdf.

$scope.renderPrev() : /public/components/pdfUpload/pdfUploadController.js

This is a helper function that renders the previous page of the pdf.

$scope.renderPage(NUM) : /public/components/pdfUpload/pdfUploadController.js

This function renders a given page of the pdf. The pdf must be loaded into the variable $scope.pdf. This var must be intitialized using PDFJS.

$scope.previewFile() : /public/components/pdfUpload/pdfUploadController.js

This function previews the selected pdf. It is called when the user clicks ‘PREVIEW PDF’ button from the country office site.

External Modules / Services:

PDFJS Module: https://mozilla.github.io/pdf.js/

Improvements / Vulnerabilities:

Move code so its not cluttering up the controller. Possibly implement an angular directive.

Environment:

NONE

**Server**

Firebase Routes

Push Notifications

**Client App**

Push Notifications