

Boy Scout Medical Record System for Blue Ridge Mountain Council



Team 4

John Kurlak
Pat Whelan
Zack Greer
Mauricio De La Barra

Client

Gregory W. Harmon
Blue Ridge Mountain Council
Boy Scouts of America

CS 4624

Multimedia, Hypertext, & Information Access
Virginia Tech

May 2, 2012

Executive Summary

For this semester project, our team decided to partner with the Boy Scouts of America in Pulaski County. Our coordinator, Gregory W. Harmon, works for the Boy Scouts and manages all of their camping facilities. Since they serve over 120,000 users per day, they were looking for ways to improve their medical recording procedures for filing injuries and accidents. For them, currently everything is written by hand into a log book and supplemented with various forms. Our project is basically a web-based digitalization of this recording procedure. This system has one main form that goes into a database. This main form has the ability to create arbitrary reports with electronic signatures (for legal reasons) as well as the ability to auto populate other form fields.

The technologies we used for this project include object-oriented PHP, MySQL, JavaScript, jQuery, phpass, CSS, and HTML5 (appcache/localStorage). The website that we developed has a home login page. After the user has successfully logged in with his or her user account information, there are multiple things he or she can do. The user can create a new user account with user information, delete an existing user, change the password of the currently logged in user, file an injury report (and upload photos of the injury), view previous injury reports, search reports (which can be downloaded and printed), manage backups (manually and automatically), access forms offline, and contact support for help. Some of the other features of the website include automatic output minification (for CSS, HTML, JavaScript, and fonts), client- and server-side input validation, and robust error handling.

Our final website solution ended up being 7,141 lines (158 pages) of code long. Our website is divided up into nine directories (root, backend, backups, css, fonts, form-templates, images, js, and photos), and the code is split up across 55 files. The root folder contains all of the website views and controllers. The backend folder contains all of the website models. The backups folder stores all manual and automatic backups in gzip format. The css folder stores all CSS. The fonts folder stores all custom web fonts. The form-templates folder stores RTF templates for each of the output forms. A user can easily modify these RTF templates, which have variable placeholders, to change the way the report forms look. The images folder contains all of the icons and images used by the website. The js folder stores all of the front-end JavaScript and jQuery code. The photos folder contains all of the photos that users have uploaded with injury forms.

Our database stores user account information and injury forms. We developed and normalized the database design in MySQL Workbench. We ended up with seventeen tables. Each injury form is broken up across a series of tables. A report table stores foreign keys to each of these injury tables. We managed our tables in phpMyAdmin, a web control panel. We perform database backups using mysqldump, a binary executable that comes with MySQL.

To make the website secure, we used the phpass library, which effectively combats rainbow tables and password crackers by using salted, per-user bcrypt password hashes. We also prepared SQL queries to prevent SQL injections. Finally, we sanitized output to prevent cross-site scripting (XSS) attacks.

Overall, the website we developed provides a nice alternative to the current paper solution that the Blue Ridge Mountain Council is using. It is our hope that the Blue Ridge Mountain Council can continue to use and modify our system for the years to come.

Table of Contents

- Executive Summary ii
- 1. Description of Problem / Proposed Solution 1
- 2. User Manual 2
 - 2.1. Account Management..... 2
 - 2.1.1. Login..... 2
 - 2.1.2. Create User 2
 - 2.1.3. Delete User..... 3
 - 2.1.4. Change Password 4
 - 2.1.5. Logout..... 5
 - 2.2. Data Access..... 6
 - 2.2.1. File an injury 6
 - 2.2.2. View Reports..... 6
 - 2.2.3. Search Reports 8
 - 2.3. Advanced 10
 - 2.3.1. Manage Backups 10
 - 2.4. Offline Access..... 11
 - 2.4.1. How to Submit a Form Offline 11
 - 2.4.2. Activate Offline Forms..... 11
 - 2.4.3. Submitting an Example Offline Form..... 12
- 3. Developer Manual 14
 - 3.1. Feature List..... 14
 - 3.2. Configuration 15
 - 3.2.1. Setting Up the Website 15
 - 3.2.2. Importing the Database..... 15
 - 3.2.3. Formatting RTF Reports 15
 - 3.2.4. Setting Up Automated Backups 15
 - 3.3. Modifying Offline Forms..... 17
 - 3.4. Inventory of All Data/Program Files 19
- 4. Lessons Learned..... 21
 - 4.1. Problems We Faced / Solutions 21
 - 4.2. Reflections..... 22
- 5. Our Process..... 23
 - 5.1. Timeline / Schedule 23

5.2. Contributions of Each Team Member.....	25
6. Current Running Instance.....	26
Acknowledgements	27
References	28

1. Description of Problem / Proposed Solution

The Boy Scouts of America in Pulaski County serves over 120,000 users per day. Any time an injury occurs at one of their facilities or during one of their activities, they must file it by hand on paper. Up to three injury forms may be filed for a single injury. Many of these forms have repeated information across them that gets tedious to enter. Furthermore, the patient care report form's assessment of injury page is very cumbersome. Report filers have to indicate with different symbols on a picture with different parts of the body the exact location and type of every injury. The client expressed that it would be very helpful if we replaced that aspect of the patient care report form with a section for uploading pictures of the injuries. Due to the tediousness of filing injury forms, the Boy Scouts of Pulaski County were looking for a way to improve their medical recording procedures. They decided that an electronic system would solve many of their problems. However, they wanted an option that would allow reports to be saved even if Internet access was not available.

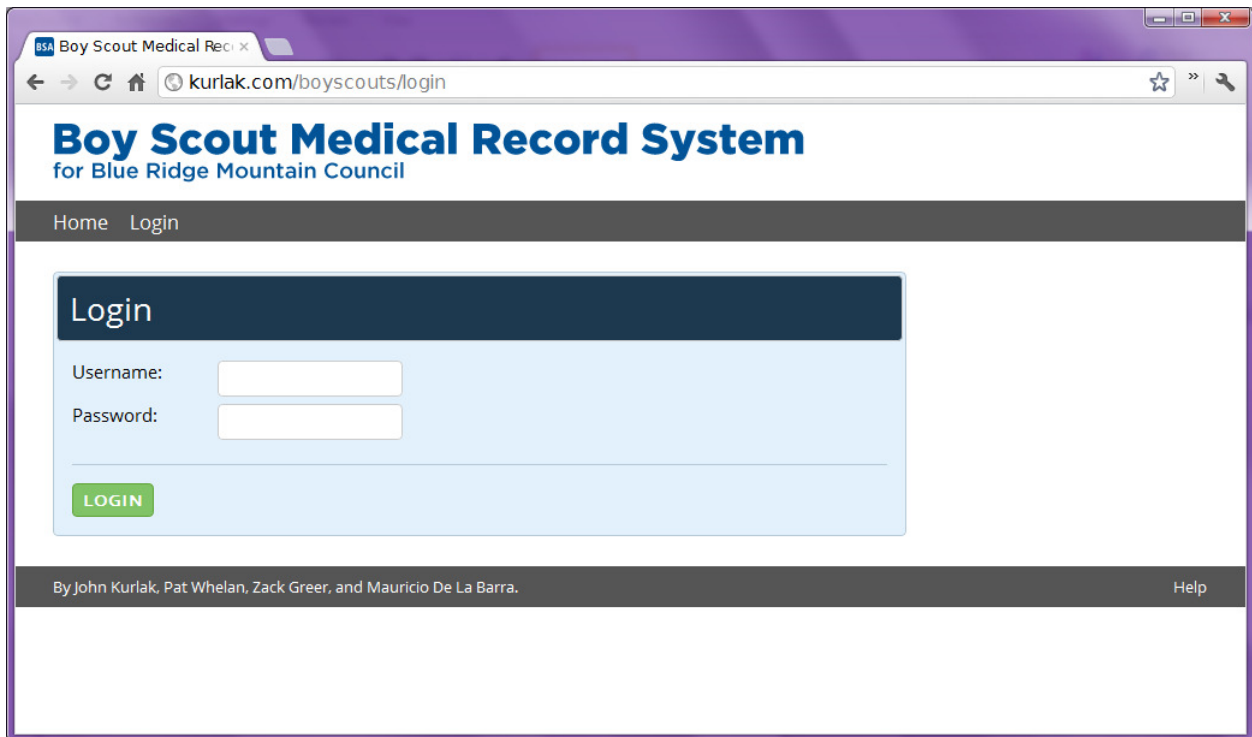
Our proposed solution was to develop a web-based digitalization of their recording procedure. We decided to make one main form with collapsible sections. Each section represents a different report form in the original system. If the user does not want to file one of the report forms, he or she can collapse that section. Any fields that are repeated in the forms automatically mirror each other. For example, if there are two fields for the patient's first name, typing the patient's first name in the first field will automatically copy it over to the second field, and vice-versa. In order to solve the offline access problem, we proposed to have forms savable when the user was offline and unloadable the next time Internet access was working again.

2. User Manual

2.1. Account Management

2.1.1. Login

Visit the site. Click “Login.” You will be directed to the login page:



The screenshot shows a web browser window with the address bar displaying "kurlak.com/boyscouts/login". The page title is "Boy Scout Medical Record System for Blue Ridge Mountain Council". The navigation menu includes "Home" and "Login". The main content area features a "Login" form with the following fields:

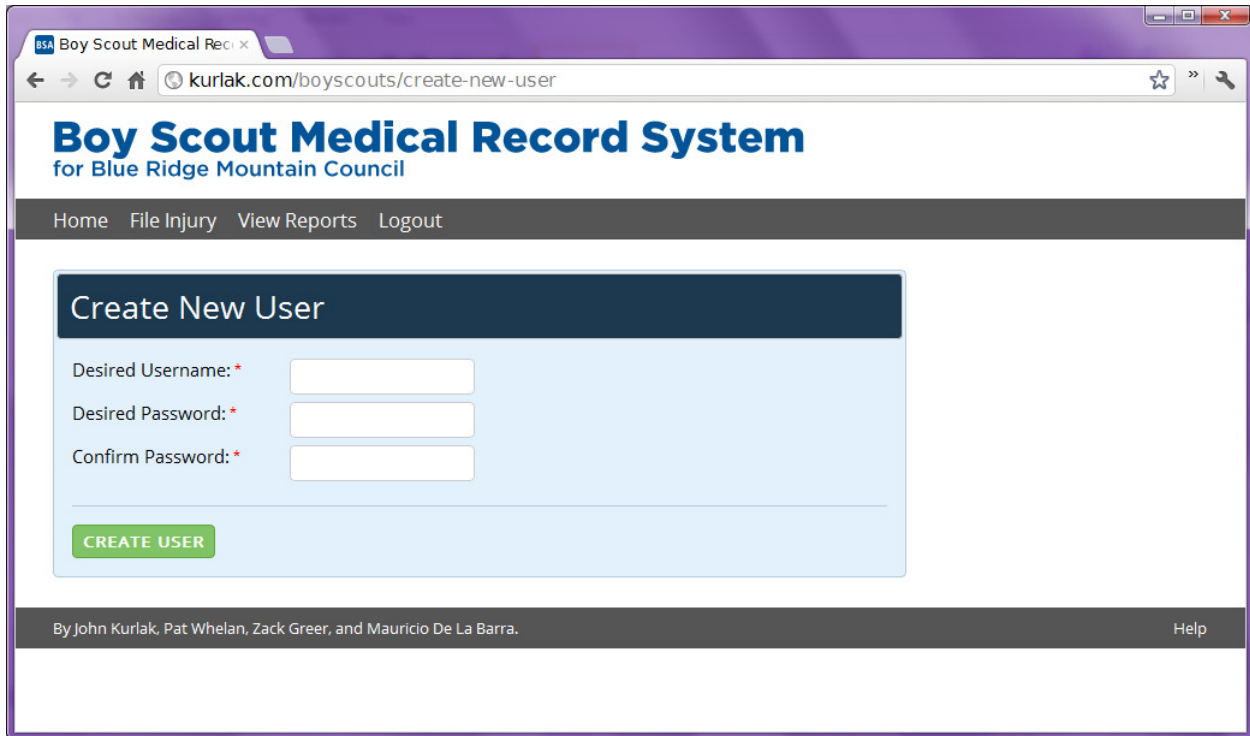
- Username:
- Password:

Below the password field is a green "LOGIN" button. At the bottom of the page, there is a footer with the text "By John Kurlak, Pat Whelan, Zack Greer, and Mauricio De La Barra." and a "Help" link.

Enter your username in the “Username:” field. Enter your password in the “Password:” field. Click “LOGIN.” This will take you to the Index.

2.1.2. Create User

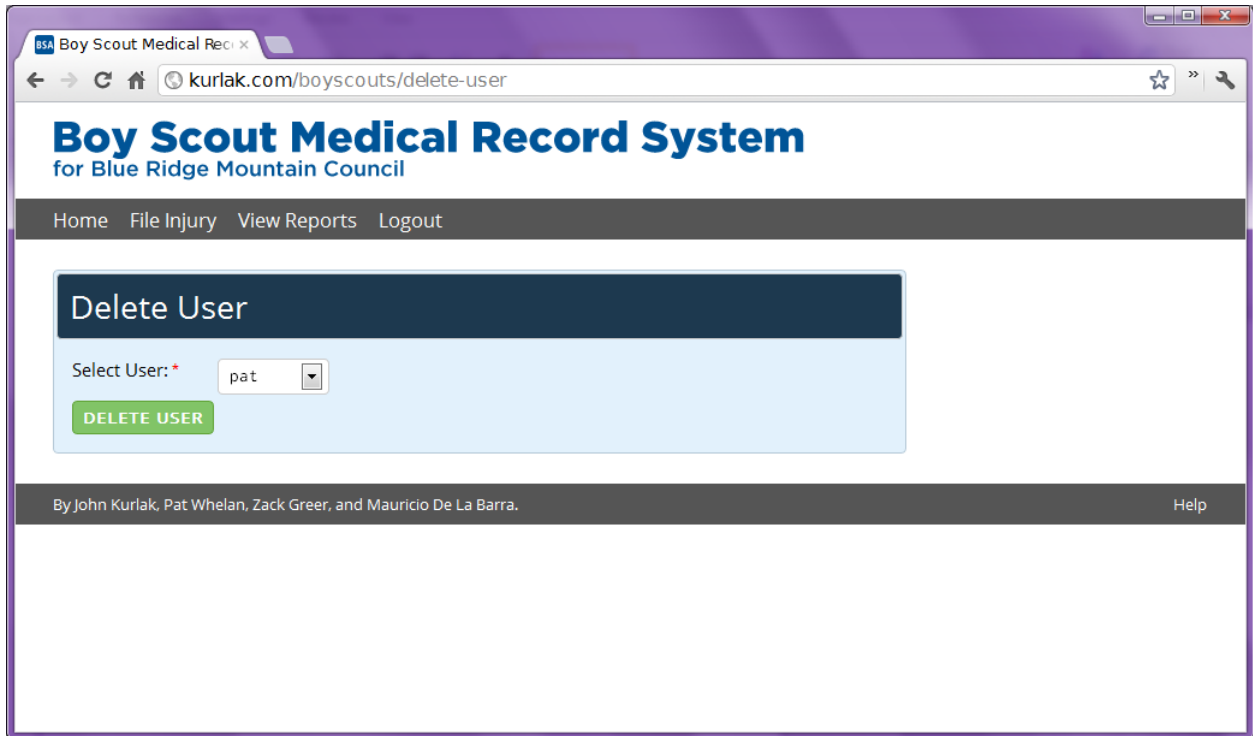
From the Index, click “Create New User.” You will be directed to the user creation page:



Enter the desired username in the “Desired Username:” field. Enter the desired password in the “Desired Password:” field. Re-enter the password in the “Confirm Password:” field. Finally, click “CREATE USER.” Note that usernames must be between 3 and 20 characters in length, and that passwords must be between 4 and 30 characters in length, and that industry-standard password security, while not required, is strongly recommended.

2.1.3 Delete User

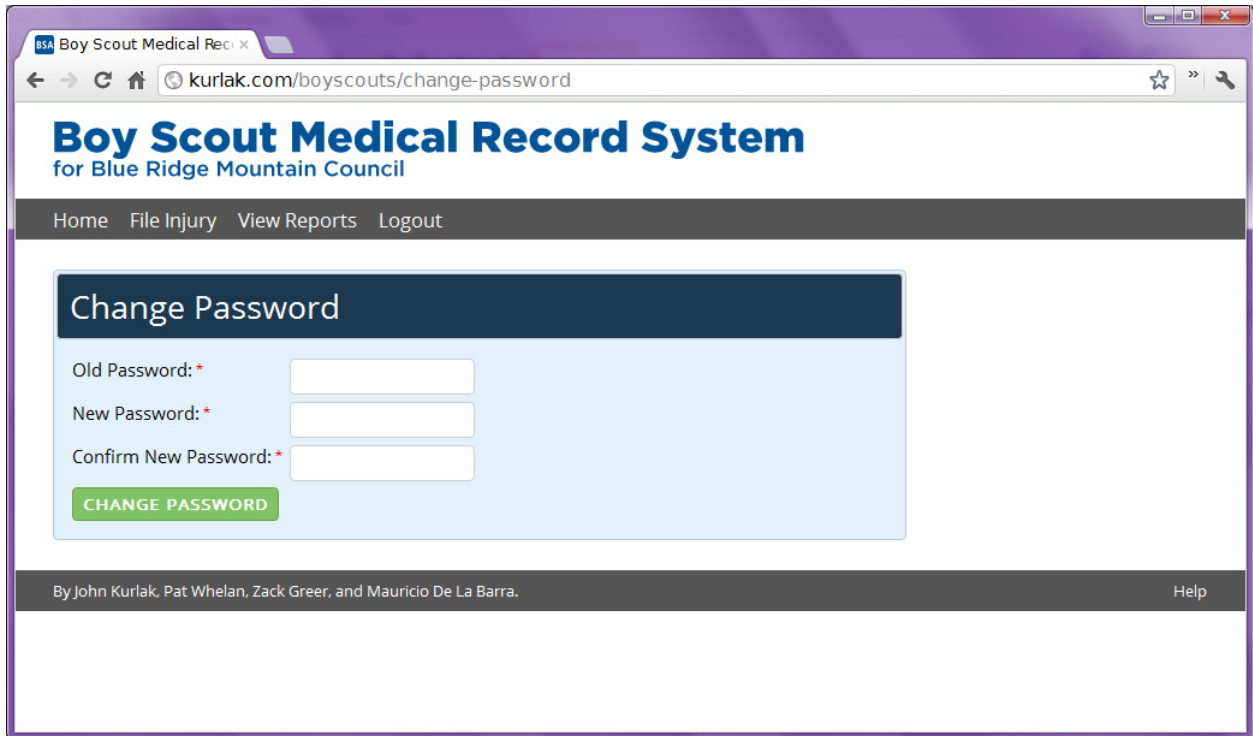
From the Index, click “Delete User.” You will be directed to the user deletion page:



Select the user to delete from the drop-down list in the “Select User:” field. Then click “DELETE USER.”

2.1.4. Change Password

From the Index, click “Change Password.” You will be directed to the change-password page:



Enter the previous password in the “Old Password:” field. Enter the desired password in the “New Password:” field. Re-enter the password in the “Confirm New Password:” field. Finally, click “CHANGE PASSWORD.” Note the passwords must be between 4 and 30 characters in length, and that industry-standard password security, while not required, is strongly recommended.

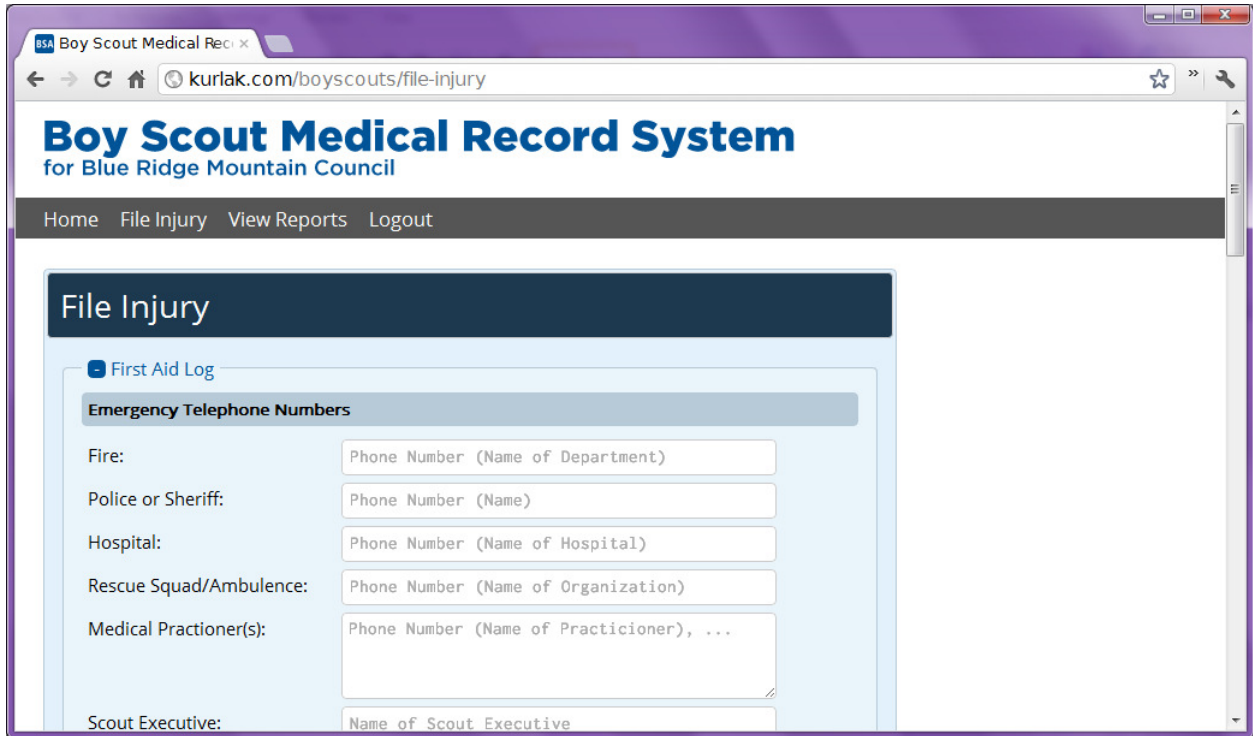
2.1.5. Logout

From any page, click “Logout,” located on the ribbon.

2.2. Data Access

2.2.1. File an injury

From any page, click “File Injury,” located on the ribbon. From the Index, you may click the button labeled “File Injury.” You will be directed to the injury filing page:

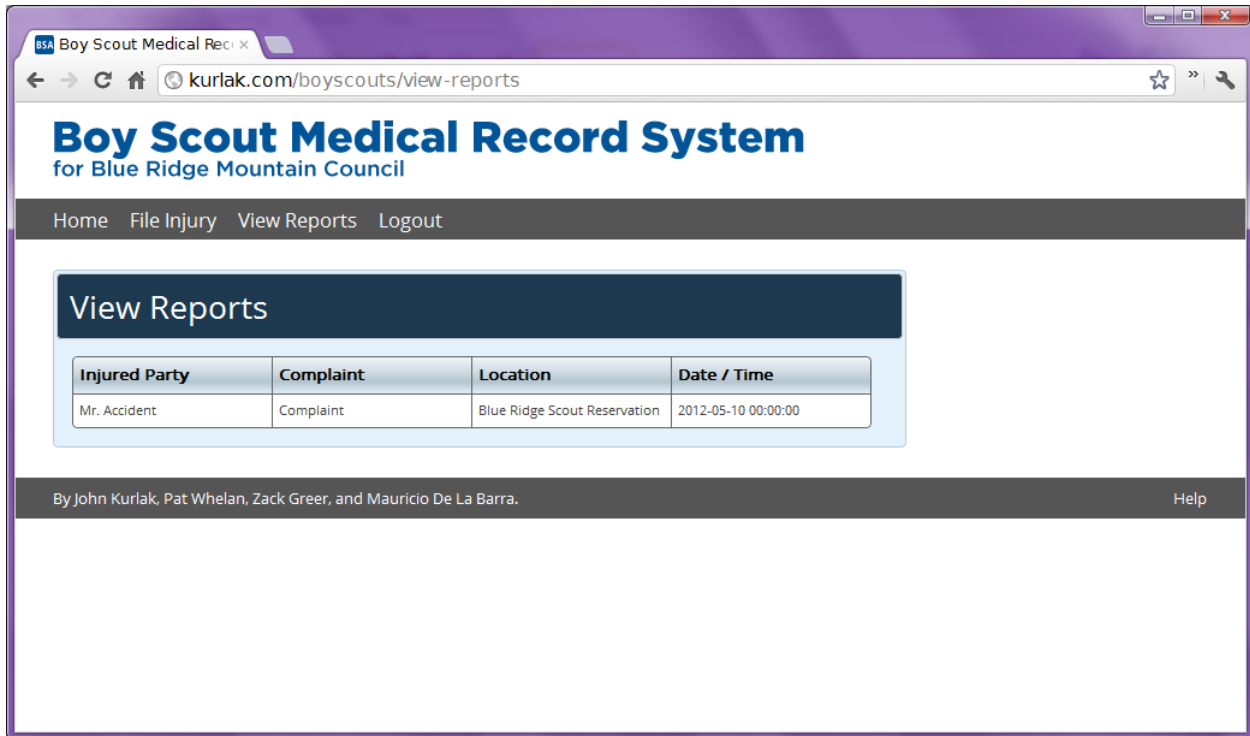


The screenshot shows a web browser window with the URL kurlak.com/boyscouts/file-injury. The page title is "Boy Scout Medical Record System for Blue Ridge Mountain Council". The navigation menu includes "Home", "File Injury", "View Reports", and "Logout". The main content area is titled "File Injury" and contains a "First Aid Log" section. Under "Emergency Telephone Numbers", there are several input fields: "Fire:" (Phone Number (Name of Department)), "Police or Sheriff:" (Phone Number (Name)), "Hospital:" (Phone Number (Name of Hospital)), "Rescue Squad/Ambulance:" (Phone Number (Name of Organization)), "Medical Practitioner(s):" (Phone Number (Name of Practitioner), ...), and "Scout Executive:" (Name of Scout Executive).

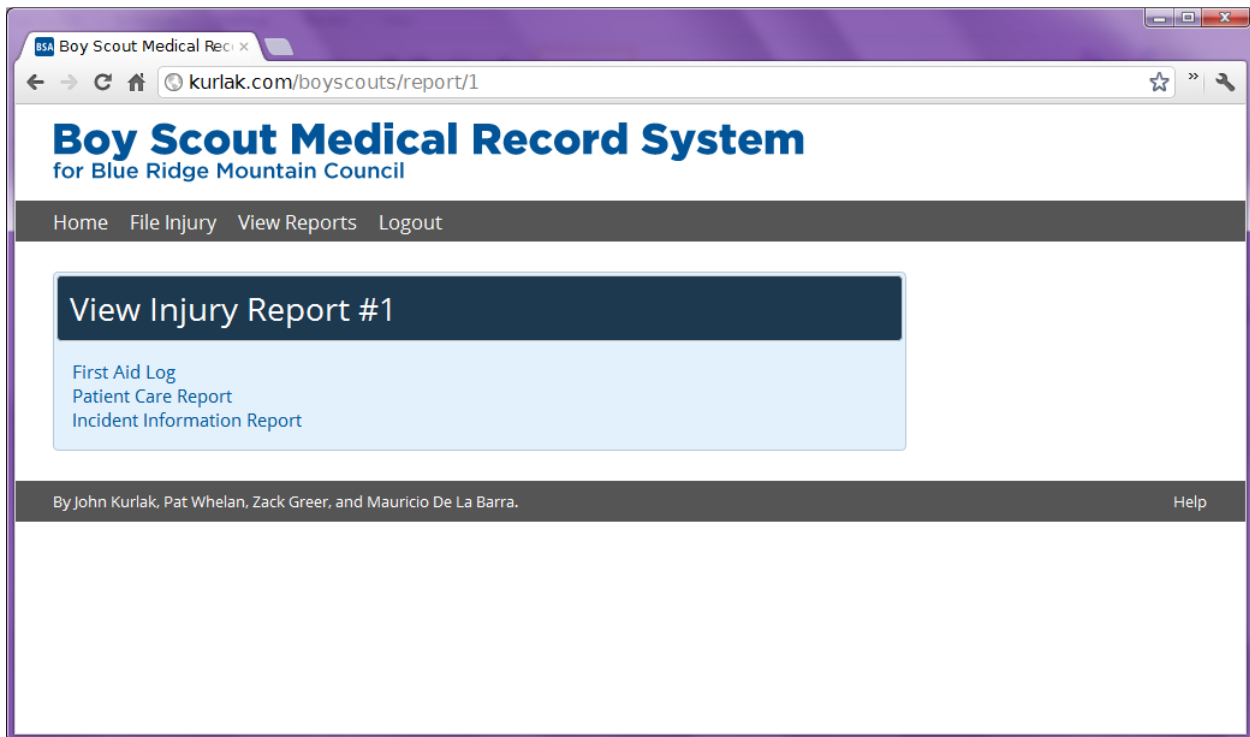
From here, you may add a patient care report or an incident information report by clicking the plusses located next to the appropriate headings. If there is a section that allows for multiple entries, there is an “Add Another” button on the top right of the section that you can click to add another entry. Fill out all relevant fields in the forms, and then click “FILE INJURY.” If there are errors, they will be reported, and you will be given the opportunity to go back and fix them.

2.2.2. View Reports

From any page, click “View Reports,” located on the ribbon. From the Index, you may click the button labeled “View Reports.” You will be directed to the report viewing page:

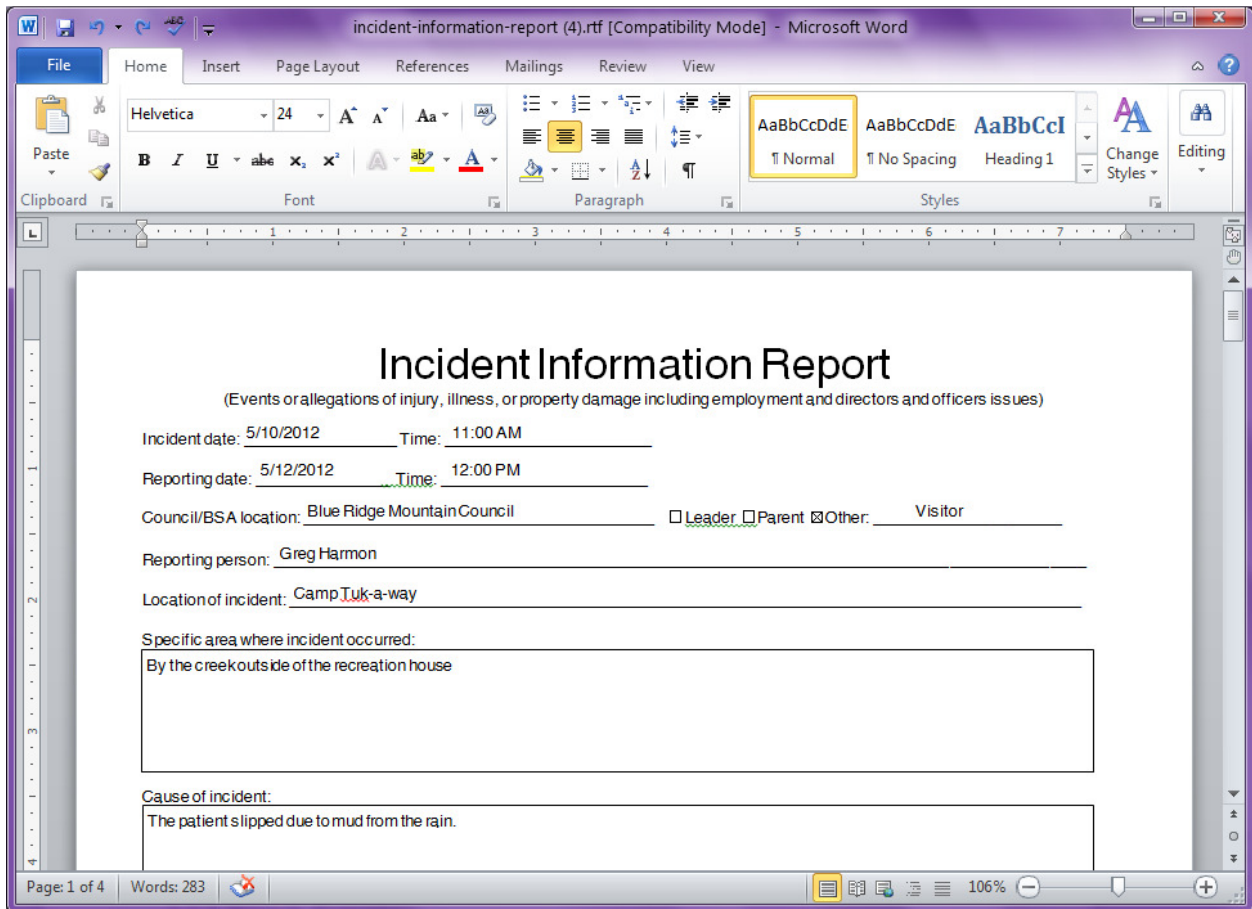


From here, you may select a report to view:



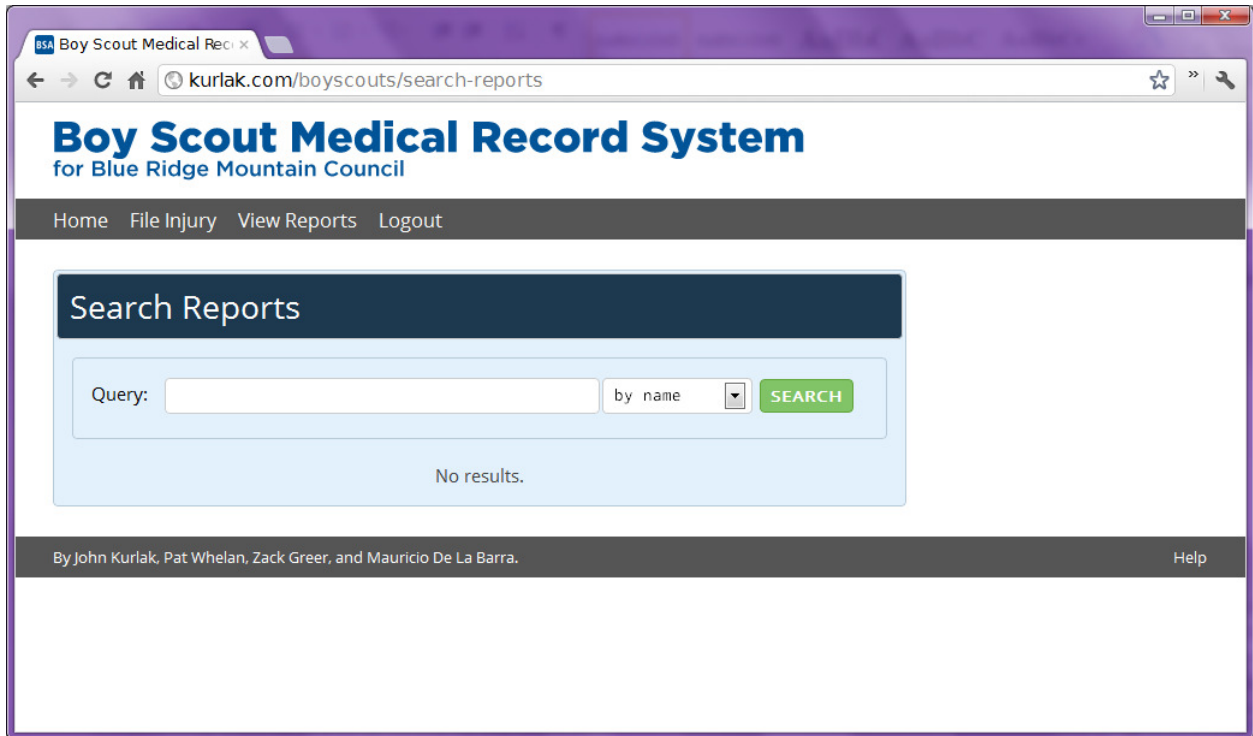
Note that reports are ordered chronologically. Click on the desired report, and you will be directed to the table of contents for that report. From there, you may download any part of that report as an rtf by clicking the relevant links.

For example, an Incident Information Report might look like:



2.2.3. Search Reports

From the Index, click "Search Reports." You will be directed to the report search page:

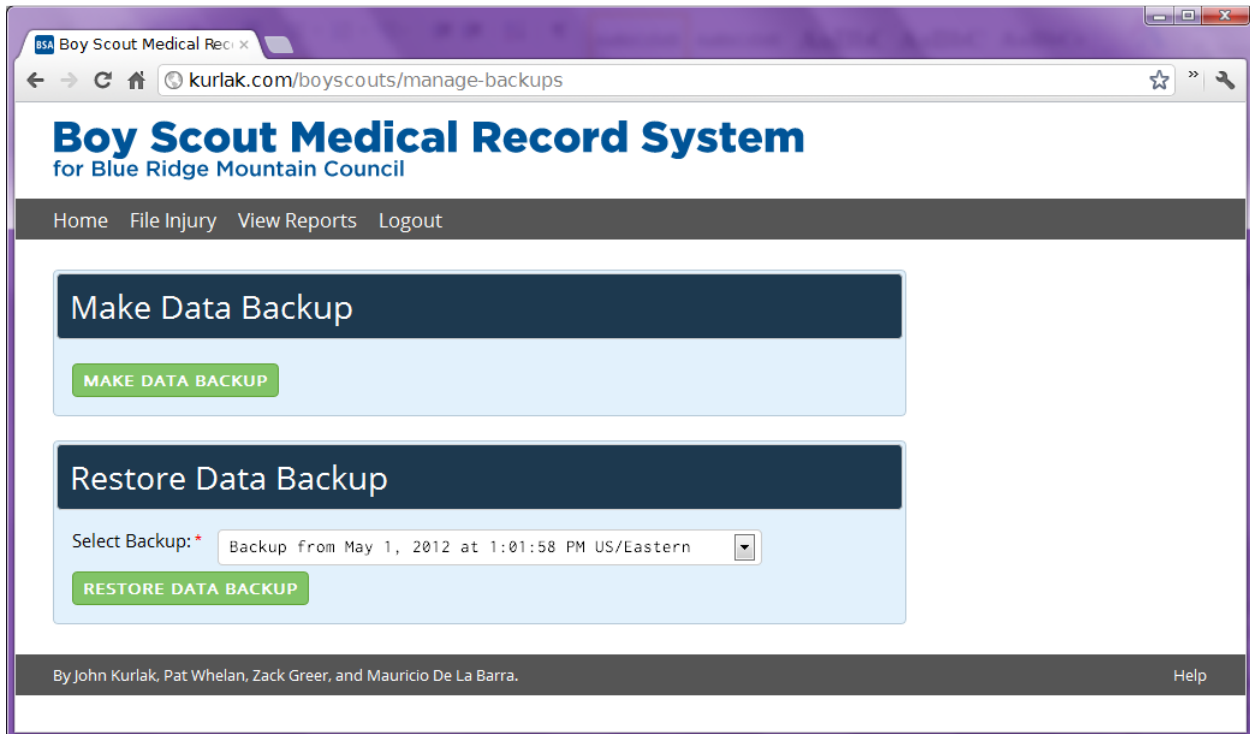


From here, you may search for reports. Enter a query and select a field to query by from the drop-down menu. Click “SEARCH.” The results will appear below. Now you may select a report to view. Note that reports are ordered by relevance. Click on the desired report, and you will be directed to the table of contents for that report. From there, you may download any injury form belonging to the report as an RTF by clicking the relevant links.

2.3. Advanced

2.3.1. Manage Backups

From the Index, click “Manage Backups.” You will be directed to the backup management page:



From here, you may make a data backup or restore to an earlier backup. Note that restoring to an earlier backup creates a backup of the current database, so there is no concern of wiping the database. Data is backed up automatically monthly.

If you would like to manually make a backup, click, “MAKE DATA BACKUP.” If you would like to restore a backup that has previously been made, select the backup from the dropdown and click, “RESTORE DATA BACKUP.”

2.4. Offline Access

2.4.1. How to Submit a Form Offline

BEFORE YOU BEGIN:

Offline website access requires an Internet browser that supports HTML5:

- Internet Explorer 8
- Firefox 12
- Chrome 18
- Safari 5

To find out what version of your internet browser on your computer, you can go to the “About” section of the browser. For an Apple handheld device (iPad, iPhone), the version of Safari is automatically updated with the iOS updates.

2.4.2. Activate Offline Forms

In order to activate offline forms, you must first visit the offline forms page by clicking the icon from the landing plan (displayed on the right) or visiting:
<host page>.com/offline-forms.php



Once the page finishes loading, it automatically updates your device with the files necessary to access the website without an internet connection.

It is recommended that you bookmark this page by right clicking the screen and click “Add Bookmark” or by typing CTRL+D on the keyboard.

NOTE:

Some browsers require you to confirm offline data storage. You will be notified at the top of the screen, as displayed below:

A screenshot of a browser window. At the top, a warning bar says "This website (www.kurlak.com) is asking to store data on your computer for offline use." with buttons for "Allow", "Never for This Site", and "Not Now". Below the warning is a form titled "Offline File Injury". The form has a "First Aid Log" section and an "Emergency Telephone Numbers" section. The "Emergency Telephone Numbers" section contains several input fields: "Fire:" (Phone Number (Name of Department)), "Police or Sheriff:" (Phone Number (Name)), "Hospital:" (Phone Number (Name of Hospital)), "Rescue Squad/Ambulance:" (Phone Number (Name of Organization)), "Medical Practioner(s):" (Phone Number (Name of Practitioner), ...), "Scout Executive:" (Name of Scout Executive), "Office:" (Phone Number), and "Home:" (Phone Number).

By clicking “Allow,” offline forms will download to your computer for offline use. If you do not see this warning, then your computer has already downloaded them.

2.4.3. Submitting an Example Offline Form

Step 1: Activate Offline forms as per the instructions in the previous section.

Step 2: Disconnect from the internet by removing the Ethernet cable from the computer, disabling the internet adapter on the bottom right corner of your screen, or configuring your browser to “Work Offline” by going to File -> Web Developer -> Work Offline.

Step 3: Navigate to the offline-forms page by entering it directly into your browser, or clicking on the previously saved bookmark.

NOTE: If the page doesn’t load, please see the section “Activate Offline Forms” above. If the problem persists, please contact us.

Step 4: Manually enter in all information into the form as if you were online, as shown below:

Health Officers / First-Aid Providers Add Another

First Name:

Last Name:

Scouting Position:

Address:

City:

State:

Zip Code:

Telephone:

E-mail Address:

Incident Information

Time: :

Date:

First Name:

Last Name:

Date of Birth:

Unit Number:

Complaint and Examination:

Step 5: Submit the form using the green “File Injury” button on the bottom. Since you are not connected to the internet, this file will not automatically be sent. Instead, it will be automatically saved to your computer.

Step 6: Reconnect to the internet.

Step 7: Navigate to the File Injury page by clicking on the File Injury Icon displayed below or visiting:

<host>.com/file-injury.php



Step 8: Once the page finishes loading, you will be prompted to automatically fill out a form with your previously entered information as displayed below. After viewing the contents, making any last minute changes, you can submit it by clicking the green “File Injury” once again. This will automatically remove the file from your computer and send it off to the server.

Boy Scout Medical Record System for Blue Ridge Mountain Council

Home File Injury View Reports Logout

File Injury

First Aid Log

Emergency Telephone Numbers

Fire:	7243160553
Police or Sheriff:	4342342342 (Police)
Hospital:	2342342343 (Hospital)
Rescue Squad/Ambulance:	9823424234 (Rescue)
Medical Practioner(s):	2984234324 (Medical)

You have 1 saved form that is waiting to be uploaded
Would you like to complete the submission now?

OK Cancel

3. Developer Manual

3.1. Feature List

- User
 - Login
 - Register
 - Change Password
 - Remove User
 - Logout
- File injury
 - Forms
 - First-Aid Log Form
 - Patient Care Report Form
 - Incident Information Report Form
 - Input validation
 - Picture uploader
- View reports
 - RTF file generator
- Search reports
 - By name
 - By date
 - By location
- Manage backups
- Automatic backups
- Offline injury report saving
- Automatic offline report uploads
- Contact support
 - CAPTCHA
- Automatic CSS/JS/HTML/font minification

3.2. Configuration

3.2.1. Setting Up the Website

To configure our site, a developer needs an Apache web server with `mod_rewrite`, PHP 4, and MySQL.

The first step to setup the website is to create a database and add a new user to it. This can often be done through a web host's control panel. Once this is done, the developer should update the settings in `backend/MySQL.php`.

Next, the developer should upload all of the files provided to a web server. He should then use his or her FTP program to `chmod` the `photos` directory to `777`.

3.2.2. Importing the Database

Once the files are uploaded, the developer should import `database.tar.gz` into the MySQL database. This can be done by going to the website that you have uploaded. Then, click "Login." Login with the following credentials:

Username: admin
Password: changeme

Once you have logged in, click "Manage Backups." Then, click "Restore Data Backup." The database should now be imported. If it did not work, you can try manually importing `database.sql` into phpMyAdmin or with the `mysql` binary file from the command line.

Now, the database should be setup for use.

3.2.3. Formatting RTF Reports

If you need to change the format of the report files, you can go into `/form-templates/` and change the RTF files.

Variables placeholders are written like `$name$`.

Section/namespaces are surrounded by two variable placeholders with the same name, like:

```
$section$  
RTF stuff  
$section$
```

3.2.4. Setting Up Automated Backups

To setup automated backups, you need to know the file path for the automated backup script. To get the file path, go to: <website URL>/make-backup.

The “path” value in the JSON response will be the path to the automated backup script. Copy that value.

Then, use your control panel or add an entry to the crontab to run the command:

```
php /home/directory/public_html/make-backup.php
```

You can setup your cronjob to run once a month with the following values:

```
0 0 1 * *
```

Once the cronjob is setup, automated backups should work!

3.3. Modifying Offline Forms

In order to change offline forms, the required files are needed:

- offline-forms.php
- offline.appcache (the Cache Manifest)
- offline.js (currently located in /js)

The dependencies for Offline Forms are:

- injury-forms.php
- images/favicon.png
- js/file-injury.js
- js/global.js
- js/jquery-ui-1.8.18.custom.min.js
- js/jquery.maskedinput-1.3.min.js
- css/styles.css
- css/datepicker/jquery-ui-1.8.18.custom.css
- <http://fonts.googleapis.com/css?family=Open+Sans>
- <http://fonts.googleapis.com/css?family=Inconsolata>
- <http://code.jquery.com/jquery-1.7.min.js>
- offline-forms.php

The website is accessible without an internet connection through HTML5's application caching (appcache). It loads any file with the manifest HTML attribute into the cache, as well as any file in the appcache file.

To Enable Appcache:

Configure the webserver to server the text/cache-manifest mime type.

The .appcache File:

The appcache file is currently offline.appcache. If CACHE MANIFEST is uncommented, then the cached files will not update unless the appcache file changes. In order to further develop the website, temporarily comment CACHE MANIFEST out in order to force the website to refuse HTML5 caching.

Any file listed in the CACHE section will be explicitly cached at all times.

Any file listed in the NETWORK section will be whitelisted, requiring that these files need an active internet connection, bypassing the cache.

Any file listed in the FALLBACK section will act as a safety net for all other files when there is no internet connection.

NOTE: In order to force browsers to update the cached files, you need to make a change to the .appcache file.

The Offline Forms:

The UI of the offline forms resides on offline-forms.php. It dynamically pulls the form fields from injury-forms.php.

The backend of the Offline Form all reside in `offline.js`. It uses HTML5's `localStorage` to store each value of the submitted form into a JSON string, and submits that to the `localStorage` database. Then, when you reconnect to the internet, it checks to see if you have files in `localStorage` and prompts the user to

3.4. Inventory of All Data/Program Files

[backend]	Stores models and data classes
Backup.php	Handles database backups
Captcha.php	Creates CAPTCHA image for contact form
CompressFonts.php	Minifies font files
CompressHTML.php	Minifies HTML files
CompressJavascript.php	Minifies JavaScript files
CompressStyles.php	Minifies CSS files
Config.php	Basic website configuration
Email.php	Handles e-mail sending
ErrorHandler.php	Manages server-side errors
FileUploader.php	Handles file uploads
Host.php	Gets reference to the page URL
Init.php	Initializes website, user state, error list, and database connection
Injury.php	Saves and retrieves injury form information
MySQL.php	Database wrapper for secure, easy queries
PasswordHash.php	phpass library file
RTF.php	Handles putting injury form data into Rich Text Format files
Rimouski.ttf	Font for CAPTCHA
Search.php	Handles searches for reports
SegoeUI.ttf	Font for CAPTCHA
User.php	Handles user actions
Validate.php	Validates user input
[css]	Website styles
[datepicker]	jQuery UI CSS
styles.css	Main website styles
[fonts]	Fonts used by website
[form-templates]	Templates for RTF injury forms
first-aid-log.rtf	First-Aid Log RTF form
incident-information-report.rtf	Incident Information Report RTF form
patient-care-report.rtf	Patient Care Report RTF form
[images]	Images and icons
[js]	Website JavaScript
datepicker.js	jQuery date picker code
file-injury.js	JavaScript file that handles injuries and automatic data propagation
global.js	Prevents accidentally browsing backwards on a page by pressing the delete key
jquery-ui-1.8.18.custom.min.js	jQuery UI JavaScript
jquery.maskedinput-1.3.min.js	jQuery Masked Input Plugin
offline.js	Handles offline access
search.js	Handles AJAX search requests
.htaccess	Creates pretty URLs; specifies 404 page; allows cache manifest to work
ajax-search.php	Returns search data as HTML
change-password.php	Change password form

contact.php	Support contact form
create-new-user.php	Create new user form
delete-user.php	Delete user form
favicon.ico	Favicon file for address bar / bookmarks
file-injury.php	Injury filing form
footer.php	Footer for all pages
header.php	Header for all pages
index.php	Home page for logged in and logged out users
injury-forms.php	Injury forms that are included by online and offline access form pages
login.php	Login form
logout.php	Logs user out
make-backup.php	File for automatic backup cronjob
manage-backups.php	Manage backup page
not-found.php	404 Page
offline-forms.php	Forms for filing an injury when offline
offline.appcache	Cache manifest for specifying which files can be accessed offline
php.ini	PHP settings
robots.txt	Robots file for search engines
search-reports.php	Search report page
view-first-aid.php	Generates First-Aid Log RTF files
view-incident-information.php	Generates Incident Information Report RTF files
view-patient-care.php	Generates Patient Care Report RTF files
view-report.php	Shows links to each injury form filed with a report
view-reports.php	Shows list of all the injury reports that have been filed
database.sql	Uncompressed database dump

4. Lessons Learned

4.1. Problems We Faced / Solutions

The first problem we faced was the learning curve regarding the domain knowledge. We needed to know which fields (concerning Boy Scout terms) in the injury forms represented the same values so that we would only have to ask for them once and only store them once in the database. In order to gain the domain knowledge, we searched the Internet for “boy scout hierarchy” and got back a number of helpful resources. Those resources are listed in our references section.

Next, we had the problem of being indecisive about architectural decisions. For example, we did not know whether to use MySQL or PostgreSQL for our relational database management system. Part of that decision was determining which options we could install or were already installed on the client’s server. Since our technical contact never replied, we port scanned their server to find out whether it supported MySQL or PostgreSQL. We found that the port for MySQL was open, so we assumed MySQL was a valid option. Next, we looked up licensing information for MySQL and learned that it would be all right for us to use it.

Another problem we faced was the sheer tediousness of dealing with all of the forms. For every form field (of which there are over 200), we had to create a database entry, create an HTML input element, provide client-side validation, provide server-side validation, save its value to the database on form submission, show its input value when a form submission fails, retrieve its value from the database, and insert its value into an RTF document. That means we had to work with over 1,600 operations without making mistakes. To overcome this problem, we tried to make as many methods as possible that we could reuse in as many ways as possible. However, everything was still very tedious.

A small problem we had was that the technical contact never replied to our e-mails. To temporarily solve this problem, we began putting our work on John’s web server. Eventually, we hope to talk about the problem with our client, Greg Harmon, to see what we can do to resolve it. Hopefully, we can setup our system on the client’s machines at a later date.

Finally, a slight problem we had was dealing with the brokenness of the HTML5 application cache technology. The problem we faced was that whenever we tried to access a page that was cached for offline use while online, our browsers would use the cached version of the page instead of the current version of the page. This made development difficult. To solve this problem, we disabled the appcache feature during development. Furthermore, the FALLBACK feature of the cache manifest will redirect browsers to the cached page instead of a 404 page when users try to go to pages that do not exist. We could not solve this problem, so our hope is that users do not go to a page that does not exist.

4.2. Reflections

When we first e-mailed our client, Greg Harmon, we asked about the problem and started to formulate what kind of solution he would need. Next, we called Mr. Harmon and began to determine the requirements for our final project. These requirements, when formalized, became our formal contract. However, when we first approached these requirements, we only asked a few questions. When we actually got into the material, we found that many questions arose along the way. We learned that it is extremely important to be in constant contact with your client in order to make sure that you not only understand what he wants, but also so that you make sure your solution is solving his needs. At every step of the way, it is important to verify and validate our actions.

We also learned that understanding domain knowledge before starting to work on a project is extremely important. It can save you time and poor design decisions if you know exactly what you are dealing with. For example, we did not know much about the hierarchy system for the Boy Scouts of America. One of the problems we encountered was that we did not know which form fields were the same and which were separate. Is troop the same as a unit number? Is council the same as district? Is camp the same as location? Once we had an understanding of the basic domain knowledge, working on the problem was a lot easier.

Another thing we learned was a lesson in project management. We did not get a lot of work done on the project until the week or two before the midterm. This set us behind significantly. It was after our midterm presentation that we learned that we needed to have a plan in our to stay on track. Setting milestones and tasks for each milestone was extremely helpful in ensuring that we could deliver our product on time.

Furthermore, we learned that self-organizing teams work really well. For example, John likes to work on front-end HTML, CSS, and JavaScript as well as back-end PHP. However, Pat and Zack like to work on back-end database development and design as well as back-end algorithm development. Mauricio enjoys writing database queries, interfacing with the client, and creating report forms. By letting everyone work on what they enjoyed the most, we were able to keep up team morale and work more effectively.

Another important lesson that we learned is that writing modular code is extremely beneficial. We have always heard that it is important, but it is not every day that we get the opportunity to actually see that for ourselves. Multiple times during the development of our project, we had to add, remove, modify, and reorganize various aspects of our system. Having modular code made that extremely easy. We do not even want to imagine what making the system would be like if we did not use modular code. We maintained modularity by attempting to use the MVC architecture as much as possible.

Finally, we learned that good documentation is mandatory. There are so many aspects of our system, and not everyone is familiar with each part. If someone had to work with someone else's code, seeing documentation is helpful.

In summary, we learned so many things from this project that we never would have anticipated. It was nice that while learned we could also help out members of our community!

5. Our Process

5.1. Timeline / Schedule

This is our timeline for the development of our solution. We tried to provide milestones at least twice per month.

February 6: Client contract and requirements due

We contacted our clients to let him know that we were going to offer our services for their website, as it is our capstone semester project. We contacted them by phone and email.

February 20: Architecture specification and preliminary research due

We gather the requirements that our client needed. These included: issues regarding webhosting and domain hosting, whether the client wanted a website or an application, use of an iPad to take photos of injuries, and the forms that we needed to digitalize.

March 5: Database design due

For this, we decided to use PostgreSQL.

March 26: Injury report form due

We made it possible for the website to file injury reports.

April 8: Automatic form data copying due

We made it possible for our website to automatically propagate data across the different injury report forms.

April 22: Reports and remaining work due; client review begins

We finished up what was left to do including, being able to manage backups and search forms within the website. We contacted our client to get some feedback.

May 1: Final project, including changes client wants, due

We made our final presentation for the semester project. We modified correspondingly according to the client's feedback

These are the deliverables we scheduled with our client:

March 26: Website with username/password authentication

We set up the website so that new users can be created and existing users can be deleted.

March 26: Injury form with optional fields depending on type and severity of injury

We customized the forms so that display different input information for the different types of forms.

April 22: Search functionality with ability to track types of incidents

We made it possible for the website to have search and sorting features for the different types of incidents.

April 22: Reports based on searches that can be printed

Injury reports can be printed in rich text format.

May 1: Automatic data backups

Backups to existing injury reports files can now be done automatically.

May 1: Deferred report upload option

We made it possible so that the user can delay submission of an injury report if he/she doesn't have internet access, or in the internet connection is slow.

April 8: (Optional, time permitting: Automatic propagation of data to existing PDF forms)

Unfortunately, we did not have time to complete this additional deliverable. This is because of issues that we had with the search functionality of the website.

May 1: (Optional, time permitting: Employee injury forms and reporting)

Unfortunately, we did not have time to complete this additional deliverable. This is because of issues that we had with automatic data backups.

5.2. Contributions of Each Team Member

The following table summarizes the contributions of each group member:

	Front-end	Back-end
John	<ul style="list-style-type: none"> • Design • Layout • File injury <ul style="list-style-type: none"> ◦ Forms ◦ Data propagation ◦ Collapsible forms and repeatable content ◦ Input masks • View reports • Search • Contact support • Manage backups • 404 page • Client interfacing • Documentation • Testing 	<ul style="list-style-type: none"> • User accounts <ul style="list-style-type: none"> ◦ Create user ◦ Delete user ◦ Change password ◦ Login ◦ Logout • Report generation • File injury • Database management wrapper • Input validation • E-mail handler • Minification scripts • File uploader • RTF report generator <ul style="list-style-type: none"> ◦ First-aid log form ◦ Patient care report form ◦ Incident information report form • Manual backups • Automatic backups • CAPTCHA • .htaccess file / URL Rewriting • Testing
Pat	<ul style="list-style-type: none"> • Offline report saving • Documentation 	<ul style="list-style-type: none"> • Initial database design with around twenty tables • Offline report saving • Automatic offline report uploads • Search
Mauricio	<ul style="list-style-type: none"> • Incident information report form • Client interfacing • Documentation 	<ul style="list-style-type: none"> • Database queries for filing injuries <ul style="list-style-type: none"> ◦ First-aid log form ◦ Patient care report form ◦ Incident information report form
Zack	<ul style="list-style-type: none"> • Patient care report form • First-aid log form • Documentation • Testing 	<ul style="list-style-type: none"> • Initial search concepts • Method for report generation • Normalization of database • Testing

6. Current Running Instance

Our project is current running on John's server at:

<http://www.kurlak.com/boyscouts/>

You can login to the system with:

Username: admin

Password: changeme

Acknowledgements

First, we would like to acknowledge our client Greg Harmon, for giving up some of his valuable time to e-mail and speak with us. Greg Harmon is the Director of Camping for the Blue Ridge Mountain Council of the Boy Scouts of America. He helped us to understand the problem, motivated us, and provided us with answers to any questions that we had. Greg can be contacted at (540) 529-5985. His address is 212 E. Court Street, Rocky Mount, VA 24151.

Next, we would like to thank our professor, Dr. Fox, and our graduate teaching assistant, Panagiotis Apostolellis. Dr. Fox gave us a number of suggestions for usability and modifiability. Both he and the GTA were instrumental in asking us important questions early on in our project, helping us to think about and sort through many of our architectural and design decisions.

Our sincerest thanks to all parties!

References

1. <http://www.html5rocks.com/en/tutorials/appcache/beginner/>
2. <http://diveintohtml5.info/storage.html>
3. http://en.wikipedia.org/wiki/Cache_manifest_in_HTML5
4. http://en.wikipedia.org/wiki/Rich_Text_Format
5. http://www.biblioscape.com/rtf15_spec.htm
6. <http://www.mysql.com/about/legal/licensing/index.html>
7. <http://www.openwall.com/phpass/>
8. <http://www.boysscouttrail.com/boy-scouts/boy-scout-organization.asp>
9. <https://sites.google.com/site/bsatroop0555/organization-hierarchy>
10. <http://stackoverflow.com/questions/225987/can-someone-explain-mysqls-license-and-what-it-means-to-closed-source-developme>