Table of Contents

error.html

Appendix C: Future Goals

Multiple Device Control
Expansion of Time Restrictions
Expansion of Customizable Messages
Expansion of Supported Applications

Overview
Brief overview of project. What it is, expected learning outcomes, etc.

The goal of this project is to provide a simple user interface for parents to implement stricter controls over a child's computer access to various computer applications. Parents should be able to login to a website called *You're Grounded!* where they are provided with various options they can implement through a settings page and logs they may view through a monitoring page.

The user's selections will initiate custom scripts to run to implement the selections. Settings options will include the ability to restrict access to supported applications by certain time periods. A final option is provided to display a custom popup warning message to the child if an attempt to access a certain application is made during restricted hours. For example, the popular game Minecraft will be able to be restricted each night at betime or blocked completely if, as the title suggests, the child has been grounded from playing the game.

Logs will be set up for monitoring attempts to access a restricted application to inform parents of when children are trying to use it. Successful and unsuccessful attempts are logged for each application selected and displayed on the monitoring page in an easy to read format.

Initial Setup
A Raspberry Pi was chosen as the device for the children to use. The latest version of Raspbian was installed on its SD card. The latest versions of Tomcat and the JDK were added to the Pi next.

Installing Tomcat and the JDK on a Raspberry Pi
Installing these programs is relatively simple. The commands below in Figure 2 must be entered in a terminal on the Pi.

Figure 1

```
#apt-get install tomcat7
#apt-get install oracle-java8-jdk
#apt-get update
#apt-get dist-upgrade
```

Installing Tomcat presented a minor problem. It generated the output below in Figure 2.

Figure 2

```
Creating config file /etc/logrotate.d/tomcat7 with new version
[FAIL] no JDK found - please set JAVA_HOME.... failed!
```

Setting JAVA_HOME required navigating to find the actual location of the newly installed JDK. This is shown in Figure 3. The actual location of the JDK may vary depending on the version or operating system used. For this project it was located as shown in the command below.

Figure 3

```
# export JAVA_HOME=/usr/lib/jvm/jdk-8-oracle-arm-vfp-hflt/jre/bin/java
```

This would need to be installed before this project could be completed. Because there are perfectly good instructions for how to install MySQL and the Workbench available online and in the Murach textbook, this will not be covered.

WEB PAGES

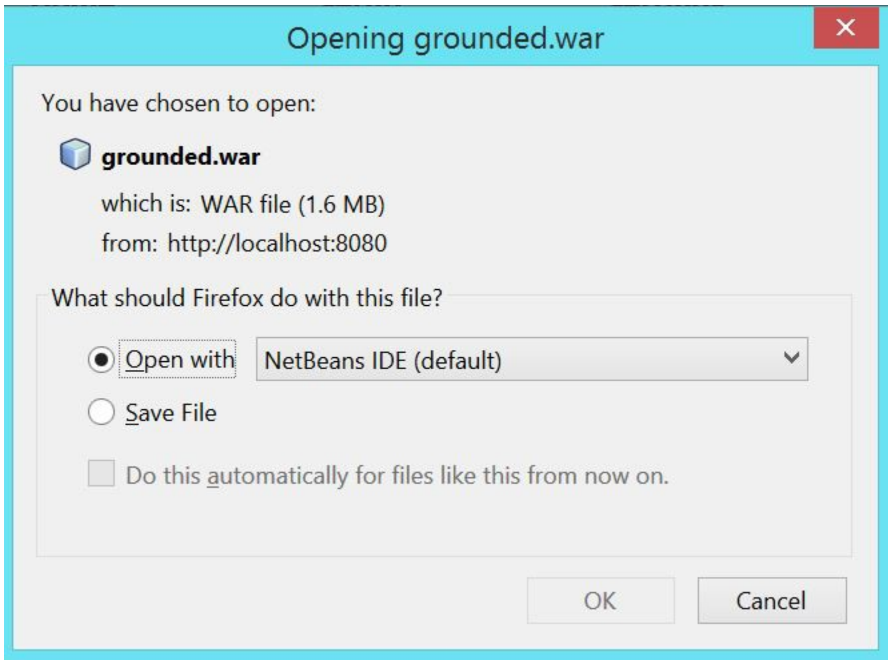
User Interface

There are six pages the user will see and may navigate to directly, including the welcome page. Four pages are displayed to the user when called from other pages, including an error page. There are also three HTML include files that are used to generate the headers, footers and navigation bars for each page to provide a consistent appearance throughout the application. Each of these pages will be discussed in detail in the pages to follow.

welcome.html
The welcome page was set in the web.xml file using a welcome-file tag. To enter the website the user simply clicks on the large red button located in the center of the screen.

index.jsp
The index page, although not serving any dynamic content to the user, was created as a JSP to allow three HTML include files to be used for a consistent header, navigation bars and footer. These are used throughout the site along with a CSS stylesheet to provide a consistent theme. The first option in the navigation bar, "Home", returns the user to this index page from anywhere in the site and a basic overview of the purpose of the site is provided.

setup.jsp
The next option in the navigation bar, "Setup", allows the user to download the WAR file for the project and provides the basic setup instructions for how to install this program on their own Rasberry Pi. Further help is available on the help page. The WAR file itself is stored in the same Web Pages directory as the setup page so that it either downloads directly to the user's device when clicked if using Chrome, or displays a window allowing the user to decide if they would like to save the file when using Firefox (Figure 222222222). **Figure 111111**


Opening grounded.war

You have chosen to open:

grounded.war

which is: WAR file (1.6 MB)

from: http://localhost:8080

What should Firefox do with this file?

● Open with NetBeans IDE (default)

○ Save File

☐ Do this automatically for files like this from now on.

settings.jsp

This is the heart of the website where the user is able to select the application they would like to restrict, time ranges and custom messages. Implementing this and various servlets was where I ran into the majority of my problems and setbacks. The current implementation will be covered here with the understanding that there are still errors and incomplete sections here that I am in the process of trying to fix.

The settings.jsp page is coded in HTML so that a form with the action set to "checkUser" is excuted when the user hits submit. Three input types are submitted, two radio buttons named selectApp and message and one set of checkbox type inputs that are named according to the range of time to which they correspond (graveyard, early, brunch, lunch, aftternoon, evening and night). Submitting this form sends the request to the CheckUserServlet to check if the user is logged in. If logged in, the user's selections are passed to the AppSettingsServlet and then sent to the updates page. If not, the user is sent to the account page to either login or create a new account.

The page is also set up to use EL to greet the user by username if they were logged in. This was done less for the user's benefit than for my own to test the functionality of the AppSettingsServlet and the checkUser method. It is not currently working due to unresolved issues.

account.jsp

The account page has two sections: one for logging in and another for creating a new account for new users. Out tags are used to help prevent against XSS attacks. A JavaScript function called pass is used to check on the client side if the passwords entered for creating a new accoun match. Regular expressions are used to ensure that the passwords used meet the criteria displayed in the message on the page. They must have at least 8 characters containing one uppercase, one lowercase, one number and one special character. Cookies are currently being displayed at the bottom of the page to assist with testing and troubleshooting.

updates.jsp

For users who are already logged in, after submitting the desired settings they are redirected to the updates page which shows which updates were made for the user to review. This is accomplished using a series of JSTL if tags for the times selected and EL for the application type and custom message selected.

monitor.jsp

The monitor page is designed to allow the user to easily view the log for each application. This page provides radio buttons for each supported application. It then passes the user's selection in the request to to a servlet called LogServlet.java. This servlet in its current configuration notes the selection and forwards the user to a new JSP page, logs.jsp.

logs.jsp

This page is set up for the user to view the log selected on the previous monitor page. It is not currently functional but will be able to display the printed output of the actual logs generated on the Raspberry Pi.

help.jsp

The help page is where the documentation can be found.. It is designed to be a somewhat more user-friendly version of this documentation. This is another page that functions as an HTML page but was created as a JSP to allow for the use of the includes files for header, navigations and footer.

Other Files
error.html
A custom error page was created as a template for other error pages to be quickly generated as needed. A simple message is displayed and the CSS stylesheet is used to provide a look that is thematically consistent for the website.

Includes
Each of the includes was given its own div class in the CSS sheet. For the navigation bars, two div classes were created, one each for the top and one for the side navigation bars. These are labelled header.html, navs.html, footer.html.

SOURCE PACKAGES

Business Layer
These three classes are JavaBeans with no args constructor and getters and setters for each of their private fields.

Log.java
There are three private fields: minecraftLog, scratchLog and browserLog. The log object will be used to retrieve and print the logs generated on the Rasbian OS for the user to view on the logs.jsp page.

Settings.java
The settings object has nine private fields: appType, customMessage, graveyard, early, brunch, lunch, afternoon, evening and night. The appType field is to select which application the settings object refers to. The customMessage field specifies the message to display and the last seven fields refer to 3 or 6 hour time ranges. This object is used in the appSettingsServlet and stored in the request to allow the settings to be saved for the user.

User.java
The user object has private fields: userID, email and password. This object is used in every servlet and forms the basis for how a user is identified during their session. In the relational database, the user object's userID is made the primary key for the user table and the foreign key of the remaining tables.

Controller Layer
AccountServlet.java

AppSettingsServlet.java

CheckUserServlet.java

LogServlet.java

ScriptServlet.java

Directory Structure

web.xml
This file contains the servlet mappings for all servlets used, as discussed earlier. It also has a welcome file specified. A session-config tag provides a 30 minute timeout for the session. A cookie tracking mode is specified by adding `<tracking-mode>COOKIE</tracking-mode>` to web.xml in the session-config tag. Security realms do not need to be ued here because logins are provided for each user and stored in a database. There are essentially no specialized administrators because the user's status is already designed to have full authority.

context.xml
This is the same context.xml used in Chapter 12 of the Murach text. I changed the Context path, Resource name, url, username and password.

Libraries
The libraries required for this project as shown in the diagram below are as follows:

- JSTL 1.2.2 - jstl-impl.jar
- JSTL 1.2.2 - jstl-api.jar
- MySQL JDBC Driver - mysql-connector-java-5.1.23-bin.jar
- tomcat-dbcp.jar
- JDK 1.8 (Default)
- Apache Tomcat or TomEE

Database

diagram
SQL script
Set up database:
Using the create_databases.sql script provided by Murach as a guide, the grounded database was created. Tables were set up to coincide with the fields of the Settings, Log and User objects.

Table	**Columns**
User	UserID: VARCHAR Password: VARCHAR
Log	LogID: VARCHAR UserID: VARCHAR LogFile: VARCHAR
Script	ScriptID: VARCHAR UserID: VARCHAR AppID: VARCHAR Graveyard: BIT Early: BIT Brunch: BIT Lunch: BIT Afternoon: BIT Evening: BIT Night: BIT ScriptFile: VARCHAR

Other Classes
These five utility classes in three packages were mostly borrowed from the Murach text and modified as needed. They were organized into source packages in similar ways to the Murach example as well.

Grounded.data
This source package contains three classes: DBUtil.java, SQLUtil.java and UserDB.java. The first two were used without modification. The last class, UserDB.java, was altered so that the fields used corresponded to those in the user table in the database which are UserID, Password and Email. Further modification is likely needed in this class since the insert and selectUser methods are not currently functioning as intended, despite many hours of troubleshooting.

Grounded.sql
Setting up a connection pool required the floowing steps:
1. Added MySQL JDBC Driver by adding a new library
2. Added tomcat-dbcp.jar by adding a new Jar file
3. A new source package called grounded.sql and a new Java class in that package called ConnectionPool.java were created. This was modeled after the Murach text example in chapter 12. Updates were made so the correct names were used. This included renaming the package and data source.

Grounded.util
This package consists of the CookieUtil.java class which was also borrowed from Murach without alteration.

Summary
What was accomplished doing the project. Lessons learned, etc.

Working on this project was both challenging and rewarding. There was more work under the hood, so to speak, than I had originally anticipated. For example, I did not realize at the beginning of this project just how many servlets would be required to accomplish the tasks I had envisioned. I believe I also veered significantly away from the sample project in the textbook which resulted in a good deal more writing of customized code than I had thought would be required.

Although I was not able to produce a working version of the website I had intended to create, I gained valuable experience in several areas. I now have a much better idea of how many different working parts are required to create what appears to the user to be a relatively simple design. I increased my skills in programming and troubleshooting in Java, HTML5, CSS, servlets, JSP, EL, MySQL and JDBC.

Appendix A: Code
WEB PAGES
User Interface
welcome.html

```
<!DOCTYPE>
<html>
      <head>
            <title>Don't Push Me!</title>
            <link rel="stylesheet" href="styles/style.css" type="text/css"/>
            <meta charset="UTF-8">
      </head>

  <body class = "wrapper">
            <div class="header">
              <h1>Don't Push Me or...</h1>
            </div>

            <div class="welcome">
                    <a href="index.jsp">
                        <img src="styles/big-red-button.png" alt="big red button"
></a>
            </div>

      </body>
</html>
```

index.jsp

```
<%--
    Document   : index
    Created on : Jul 18, 2015, 12:58:23 PM
    Author     : Primary
--%>

<!DOCTYPE html>
<%@ include file="includes/header.html" %>
<%@ include file="includes/navs.html" %>

<div id="mainBody">
    <br>
    Limiting a child's screen time is important. Whether you want to set up
    stricter policies to schedule access or revoke certain privileges
    temporarily, You're Grounded is here to help. And best of all, it's
    completely free!
    <br><br>
    If this is your first time here, begin with our easy setup page.
    Once complete you can change settings and monitor usage of devices and
    specific apps as much or as little as you need.
</div>


<%@ include file="includes/footer.html" %>
```

setup.jsp

```
<%--
    Document   : setup
    Created on : Jul 18, 2015, 1:36:38 PM
    Author     : Primary
--%>
<!DOCTYPE html>
<%@ include file="includes/header.html" %>
<%@ include file="includes/navs.html" %>

<div id="mainBody">
    <br>
    instructions to install tomcat, jdk, set JAVA_HOME <br><br>
        <a href="grounded.war">Download WAR file</a>
```

```
</div>


<%@ include file="includes/footer.html" %>
```

settings.jsp

```
<%@ include file="includes/header.html" %>
<%@ include file="includes/navs.html" %>

<div id="mainBody">
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<h4>Hello <!-- ${user.userID} -->, Select which services you would like to
limit:</h4>
    <form class action="checkUser" method="post">
        <input type="hidden" name="action" value="checkUser">
        <input type="radio" name="selectApp" value="minecraft"> minecraft<br>
        <input type="radio" name="selectApp" value="scratch"> scratch<br>
      <input type="radio" name="selectApp" value="browser"> browser<br>

    <h4>Select the times to restrict access</h4>
            <table style="width:100%">
            <tr>
                <td><input type="checkbox" name="graveyard" value="yes"> Midnight
- 6am</td>
                <td><input type="checkbox" name="afternoon" value="yes"> 3pm -
6pm</td>
            </tr>
            <tr>
                <td><input type="checkbox" name="early" value="yes"> 6am - 9am
</td>
                <td><input type="checkbox" name="evening" value="yes"> 6pm -
9pm</td>
            </tr>
            <tr>
                <td><input type="checkbox" name="brunch" value="yes"> 9am - noon
</td>
                <td><input type="checkbox" name="night" value="yes"> 9pm -
midnight</td>
            </tr>
            <tr><td><input type="checkbox" name="lunch" value="yes"> noon -
3pm</td></tr>
            </table>

    <h4>Set a custom message</h4>
        <input type="radio" name="message" value="null"> none<br>
        <input type="radio" name="message" value="restricted"> Access
restricted.<br>
        <input type="radio" name="message" value="bedtime"> Go to bed!<br>
        <input type="radio" name="message" value="warning"> This attempt has been
logged.
                    After 3 attempts you will be grounded!<br>

        <input type="checkbox" name="time" value="yes"> Read message using text to
speech <br><br>
        <input type="submit" value="Submit" >
    </form>
</div>
```

```
<%@ include file="includes/footer.html" %>
```

account.jsp

```
<%@ include file="includes/header.html" %>
<%@ include file="includes/navs.html" %>

    <script>
    function pass() {
        if (document.forms[0].pwd1.value != document.forms[0].pwd2.value) {
            alert('Your passwords did not match');
            return false;
        }
        return true;
    }
    </script>
    <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
    <br><strong>Login</strong><br>
    <form action="login" method="post">
        Username <br>
        <input type="text" name="username" value="<c:out value='$
{user.userID}'/>"><br>
        Password <br>
        <input type="password" name="pwd1"
               required pattern="(?=^.{8,}$)((?=.*\d)|(?=.*\W+))(?![.\n])(?=.*[A-
Z])(?=.*[a-z]).*$"
               value="<c:out value='${user.password}'/>"><br>
        <input type="submit" value="Login" >
    </form>
    <br><strong>Create New Account</strong><br>
    <form action="newUser" onSubmit="pass()" method="post">
            <input type="hidden" name="action" value="newUser">
            Username <br>
            <input type="text" name="userID" value="<c:out value='$
{user.userID}'/>" ><br/>
            <!--<input type="text" name="UserID" value="<c:out value='$
{user.userID}'/>"><br>-->
            Email<br>
            <input type="email" placeholder="user@email.com" name="email"
            value="<c:out value='${user.email}'/>" ><br/>
            Password <br>
            <input type="password" name="pwd1"
                   required pattern="(?=^.{8,}$)((?=.*\d)|(?=.*\W+))(?![.\n])(?
=.*[A-Z])(?=.*[a-z]).*$"
                   value="<c:out value='${user.password}'/>" ><br/>
            Confirm Password<br>
            <input type="password" name="pwd2"
                   required pattern="(?=^.{8,}$)((?=.*\d)|(?=.*\W+))(?![.\n])(?
=.*[A-Z])(?=.*[a-z]).*$"><br>
            <input type="submit" value="Create Account" >
    </form>
    <br>
    Password requires at least 8 characters containing one uppercase,<br>
    one lowercase, one number and one special character)<br>


    <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
    <table>
        <tr>
```

```
            <th>Name</th>
            <th>Value</th>
        </tr>
        <c:forEach var="c" items="${cookie}">
        <tr>
            <td>${c.value.name}</td>
            <td>${c.value.value}</td>
        </c:forEach>
    </table>

<%@ include file="includes/footer.html" %>
```

updates.jsp

```
<%@ include file="includes/header.html" %>
<%@ include file="includes/navs.html" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

 the ${settings.appType} settings were updated:
    <br><br>

        <c:if test="${settings.graveyard == 'yes'}">
            <label>12am - 6am: Restricted</label><br>
        </c:if>
        <c:if test="${settings.early == 'yes'}">
            <label>6am - 9am: &nbsp; Restricted</label><br>
        </c:if>
        <c:if test="${settings.brunch == 'yes'}">
            <label>9am - 12am: Restricted</label><br>
        </c:if>
        <c:if test="${settings.lunch  == 'yes'}">
            <label>12pm - 3pm: Restricted</label><br>
        </c:if>
        <c:if test="${settings.afternoon == 'yes'}">
            <label>3pm - 6pm: &nbsp;  Restricted</label><br>
        </c:if>
        <c:if test="${settings.evening == 'yes'}">
            <label>6pm - 9pm: &nbsp;  Restricted</label><br>
        </c:if>
        <c:if test="${settings.night == 'yes'}">
            <label>9pm - 12am: Restricted</label><br>
        </c:if>
    <br>

    <label>custom message:</label><br>
    <span>${settings.customMessage}</span><br>

<%@ include file="includes/footer.html" %>
```

monitor.jsp

```
<%@ include file="includes/header.html" %>
<%@ include file="includes/navs.html" %>

<div id="mainBody">
<h4>Select the log you would like to view: </h4>

    <form class action="showLog" method="post">
        <input type="radio" name="selectLog" value="minecraft"> minecraft log<br>
```

```
      <input type="radio" name="selectLog" value="scratch"> scratch log<br>
        <input type="radio" name="selectLog" value="browser"> browser log<br><br>

        <input type="submit" value="Submit" class="margin_left">
    </form>
</div>

<%@ include file="includes/footer.html" %>
```

logs.jsp

```
<%@ include file="includes/header.html" %>
<%@ include file="includes/navs.html" %>

<p>
    Here is the log you requested...

</p>

<%@ include file="includes/footer.html" %>
```

help.jsp

Other Files

error.html

Includes

header.html

navs.html

footer.html

SOURCE PACKAGES

Business Layer

Log.java

Settings.java

User.java

Controller Layer

AccountServlet.java

AppSettingsServlet.java

CheckUserServlet.java

LogServlet.java

ScriptServlet.java

Directory Structure

web.xml

context.xml

SQL script: create_databases.sql

Other Classes

Grounded.data

DBUtil.java

SQLUtil.java

UserDB.java

Grounded.sql

ConnectionPool.java

Grounded.util

CookieUtil.java

Appendix B: Screenshots

welcome.html
index.jsp
setup.jsp
settings.jsp
account.jsp
updates.jsp
monitor.jsp
logs.jsp
help.jsp
error.html

Appendix C: Future Goals

While working on the planning and development stages of this project, I had a lot of ideas that had to be set aside due to the complexity and time required to complete them. A partial list of some of these goals is included here to demonstrate the range of usefulness this project could provide its users as well as my own learning opportnities in creating a web application that provides highly customizable functionality.

Multiple Device Control
Multiple devices that children may be able to access willl be supported. The server will be run on a dedicated machine which only parents can access. Devices would be identified by MAC addresses which would be stored in a table in the database to ensure that each device was only able to be controlled by one user account, thus avoiding potential conflicts.

Expansion of Time Restrictions
Existing options for restricting access would be expanded. Rather than being bound to only restrict access during the available ranges of times, times could be entered manually by the user to restrict usage during any time of the day or night for as little as one minute. Additionally, days of the week and specific dates options would be provided for future scheduling that varied by date.

Expansion of Customizable Messages
Another existing feature that could benefit from increased functionality is the setting of custom messages. A future goal for this feature is to allow the user to create their own messages. Messages could be stored in the database and used for specific applications, times or dates.

Expansion of Supported Applications
Ideally, users will want to be able to restrict access to a much broader range of applications on a child's device. This ability to add new applications to the existing list would be easy to implement once a method was devised to determine the name that would need to be used in the script to limit its use. I will need to research this further or perhaps just leave it as an advanced user feature for those skilled enough at finding the name of the application themselves.