Emergency Resource Management System Documentation

Abstract Code and SQL

Login to ERMS

Display username and password fields

- User enters *username* ('$Username'), *password* ('$Password') input fields.
- If data validation is successful for both *username* and *password* input fields, then:
    - When ***Login*** button is clicked:

```
SELECT Password FROM `User` WHERE Username = '$Username';
```

        - If User record is found but User.password != '($Password)':
            - Go back to **Login to ERMS** form, with error message.
        - Else:
            - Store login information as session variable '($Username)'.
            - Go to **Main Menu**.
- Else *email* and *password* input fields are invalid, display **Login to ERMS** form, with error message.

Main Menu

- Lookup and display different User Tables for User.username and type

```
SELECT Username, Category, null, null, null, null, null FROM Municipalities WHERE
Username = '$Username'
UNION
SELECT Username, null, Location, NumberofEmployees, null, null, null FROM
Companies WHERE Username = '$Username'
UNION
SELECT Username, null, null, null, AgencyNameLocationOffice, null, null FROM
GovAgencies WHERE Username = '$Username'
UNION
SELECT Username, null, null, null, null, JobTitle, DateHired FROM Individuals WHERE
Username = '$Username';
```

- Show "***Add Resource***", "***New Incident***", *"**Search Resources***", "***Resources Status***" and "***Resource Report***" tabs.
- Upon:
    - Click ***Add Resource*** button- Jump to the **Add Resource** task.
    - Click ***New Incident*** button- Jump to the **New Incident** task.
    - Click ***Search Resources*** button- Jump to the **Search Resources** task.
    - Click ***Resources Status*** button- Jump to the **Resources Status** task.
    - Click ***Resource Report*** button- Jump to the **Resource Report** task.
    - Click ***Exit*** button- Invalidate login session and go back to the **Login to ERMS** form.

Add New Resource

- Generate and display a unique numeric ID ('$ID')
- Read the ESF Table and display dropdown menu for *Primary ESF* and multichoice list for *Additional ESF*

```
SELECT Number, Description FROM ESF;
```

- Read the TimeUnit Table and display the dropdown menu for *Time Unit*

```
SELECT Name FROM TimeUnit;
```

- Find the logged in user; display the logged in user's own User.username ('$Username') as the resource's owner

- The user fills out the **New Resource** Form for the new resource, including recording *Resource Name ($Name)*, *Primary ESF ($PrimaryESFNumber)*, *Additional ESFs ($ESFNumber)*, *Model ($Model)*, *Capabilities ($CapabilityName)*, *Home Location's Latitude ($Latitude) and Longitude ($Longitude)*, *Cost ($Cost), Cost Unit ($UnitName)* and *Maximum Distance ($MaxDistance)*.
- When the user clicks ***Save*** button:
    - The resource is assigned the unique numerical ID ('$ID') automatically
    - The resource's owner is also set automatically to the logged in user ('$Username')
    - *Resource Name*, *Primary ESF*, *Additional ESFs*, *Model*, *Capabilities*, *Home Location*, *Cost/Cost Per* and *Maximum Distance* are saved

```
INSERT INTO `Resources`
VALUES ('$ID', '$Name', '$Latitude', '$Longitude', '$Model', '$MaxDistance',
'$PrimaryESFNumber', '$Cost', '$UnitName', '$Username');
```

    - For each *Addtional ESF* chosen by the user:

```
INSERT INTO `AdditionalESF`
VALUES ('$ResourceID', '$ESFNumber');
```

    - For each *Capability* input by the user:

```
INSERT INTO `Capabilities`
VALUES ('$ResourceID', '$CapabilityName');
```

- When the user clicks ***Cancel*** button:
    - Go back to the **Main Menu**

New Incident

- Read the Declarations Table and display dropdown menu for *Declaration*

```
SELECT Abbreviation, Name FROM Declarations;
```

- The user fills out the **New Incident** Form for the new emergency incident, including choosing a incident *declaration ($Abbreviation)*, recording the *date ($Date)* of the incident, a brief *description ($Description)*, *latitude ($Latitude), and longitude ($Longitude)* coordinates of the incident.

- When the user clicks ***Save*** button:
    - The incident *declaration*, the *date* of the incident, a brief *description*, and the *latitude/longitude* coordinates of the incident are saved into Incidents table
    - The incident is automatically assigned a unique ID that combines the abbreviation of the incident declaration abbreviation with an automatically generated number ($Number) unique to that incident declaration
    - the owner of the incident is automatically set to the current user ($Username)

```
INSERT INTO 'Incidents' VALUES (`$Abbreviation`, `$Number`, `$Date`, `$Description`, `$Latitude`, `$Longitude`, `$Username`);
```

- When the user clicks ***Cancel*** button:
    - Go back to the **Main Menu**

Search Resources

- Get the ESF Table and display them in *ESF* dropdown menu

```
SELECT Number, Description FROM ESF;
```

- Find a list of incidents that is owned by current user. That is to say Incidents.owner match current user's User.username. Display them in *Incident* dropdown menu

```
SELECT Abbreviation, Number, Description FROM Incidents WHERE Username = `$Username`;
```

- User enters *keyword* ('$Keyword'), chooses *ESF* ('$ESF') from ESF dropdown menu, enters *radius* ('$Radius') and chooses emergency *Incident*('$Incident') from Incident dropdown menu.
- Upon:
    - Click ***Search*** button-
        - If all the dropdown selection and Location meet the requirement (can be NULL):
            - Jump to the **Search Results** task.
    - Click ***Cancel*** button- jump back to the **Main Menu**.

Search Results

- Display incident description and ID if Incidents was selected
- Find a list of resources that *keyword*('$Keyword') is matched in Resources.name Resources.model or Resources.Capabilities, proximity to incident is within *Location*('$Location'), and *ESF*('$ESFNumber') is matched in Resources.PrimaryESF or AdditionalESF. Note that Proximity is calculated using haversine formula. If any of the filters are NULL, do not validate it.
- Derive availability (resource status) from the InUse Table
- Display resource ID, resource name, resource owner, resource cost and cost per unit, resource status, next available date, and distance

```
SELECT r.Name, Cost, UnitName, u.Name, i.ReturnDate,
( 6371 * acos( cos( radians(37) ) * cos( radians( lat ) ) * cos( radians( lng ) - radians(-122) ) +
sin( radians(37) ) * sin(radians(lat)) AS Distance
FROM Resources r
JOIN User u ON r.Username =u.Username
LEFT JOIN InUse i ON r.ResourceID = i.ResourceID
WHERE r.Username = `$Username` AND
(r.PrimaryESFNumber = `$ESFNumber` OR `$ESFNumber` IN (SELECT ESFNumber FROM
AdditionalESF))
ORDER BY Distance;
```

- If *Incident* was selected
    - For all resources row:
        - If Resources.owner is current user:
            - If resource is available, display ***Deploy*** button.
            - Else do not display button
        - Else display ***Request*** button
- Upon:
    - Click ***Request*** Button (if there exists such a button)-
        - Popup a dialog to get an Return Date
        - Update Requests Table with Incident ID ($Abbreviation) ($Number), Resource ID ($ResourceID), Return Date ($ReturnDate), Request Date ($RequestDate). Note underlined part is the key for table.

```
INSERT INTO `Requests` VALUES (`$ResourceID`, `$Abbreviation`,
`$Number`, `$RequestDate`, `$ReturnDate`);
```

        - Gray shade the clicked button and make it unclickable
    - Click ***Deploy*** buttons (if there exists such a button)-
        - Popup a dialog to get an Return Date

        - Update InUse Table with Resource ID, Incident ID, Return date. Note underlined part is the key for table. For every resource, it is guaranteed there is no more than one Incident taking it.

```
INSERT INTO `InUse` VALUES (`$ResourceID`, `$Abbreviation`, `$Number`,
`$StartDate`, `$ReturnDate`);
DELETE FROM Requests WHERE ResourceID = `$ResourecID` AND
Abbreviation = `$Abbreviation` AND Number = `$Number`;
```

        - Gray shade the clicked button and make it unclickable
    - User can click ***Close*** to go back to **Main Menu**

Resource Status

- Find all resources that are in InUse Table and being used by me, display Resources.Name, Incidents.Description, resource owner`s User.Name, Start Date, Return Date from InUse table and ***Return*** button.

```
SELECT Resources.Name, Incidents.Description, Resources.Username,
InUse.StartDate, InUse.ReturnDate FROM `InUse`
INNER JOIN `Resources` ON InUse.ResourceID = Resources.ID
INNER JOIN `Incidents` ON InUse.Abbreviation = Incidents.Abbreviation AND
InUse.Number = Incidents.Number
WHERE Incidents.Username = `$Username`
```

- Find all resources that are requested by me and display Resources.Name, Incidents.Description, resource owner's User.Name, Return Date from Requests table and ***Cancel*** button.

```
SELECT Resources.Name, Incidents.Description, Resources.Username,
Requests.ReturnDate FROM `Requests`
INNER JOIN `Incidents` ON Requests.Abbreviation = Incidents.Abbreviation AND
Requests.Number = Incidents.Number
INNER JOIN `Resources` ON Requests.ResourceID = Resources.ID
WHERE Incidents.Username = `$Username`;
```

- Find all resource requests that are received by me and display Resources.Name, Incidents.Description, resource owner's User.Name, and Return Date from Requests table.

```
SELECT Resources.Name, Incidents.Description, Resources.Username,
Requests.ReturnDate FROM `Requests` INNER JOIN `Incidents`
```

```
ON Requests.Abbreviation = Incidents.Abbreviation AND Requests.Number =
Incidents.Number
INNER JOIN `Resources` ON Requests.ResourceID = Resources.ID
WHERE Resources.Username = `$Username`;
```

- If resource is in use, display ***Reject*** button.
- Else, display ***Deploy*** and ***Reject*** buttons

```
SELECT Resources.ID FROM `Requests` INNER JOIN `Incidents`
ON Requests.Abbreviation = Incidents.Abbreviation AND Requests.Number =
Incidents.Number
INNER JOIN `Resources` ON Requests.ResourceID = Resources.ID
INNER JOIN `InUse` ON Requests.ResourceID = InUse.RequestID
WHERE Resources.ID= `$ResourceID`;
```

- If the user clicks ***Return*** in the Resource in Use list:
  - Update the LastUsed table with this returned resource

```
UPDATE `LastUsed` SET ResourceID = `ResourceID`,
Abbreviation = (SELECT Abbreviation FROM `InUse` WHERE ResourceID =
`$ResourceID`),
Number = (SELECT Number FROM `InUse` WHERE ResourceID =
`$ResourceID`);
```

  - **Return Resource**: Remove the resource from the Resource InUse Table

```
DELETE FROM `InUse` WHERE ResourceID = $`ResourceID`;
```

  - Grayshade ***Return*** and make it unclickable
- If the user clicks ***Cancel*** in the Resources Requested by Me list:
  - **Cancel Request**: Remove the request from the Requests Table

```
DELETE FROM `Requests` WHERE ResourceID = $`ResourceID`;
```

  - Grayshade ***Cancel*** and make it unclickable
- If the user clicks ***Deploy*** in the Resource Requests Received by Me list:
  - **Deploy Resource**: Move the resource from Resource Requests Table to Resource in Use Table and record the current date ($CurrentDate) as Start Date

```
INSERT INTO `InUse` VALUES ( ResourceID,
```

```
(SELECT Abbreviation FROM `Requests` WHERE ResourceID =
$`ResourceID`) AS Abbreviation,
(SELECT Number FROM `Requests` WHERE ResourceID = $`ResourceID`)
AS Number, `$CurrentDate`);

DELETE FROM `Requests` WHERE ResourceID = $`ResourceID`;
```

    - Grayshade ***Deploy*** and make it unclickable
- If the user clicks ***Reject*** in the Resource Requests Received by Me list:
    - **Reject Resource**: Remove the resource from Resource Requests Table

```
DELETE FROM `Requests` WHERE ResourceID = $`ResourceID`;
```

    - Grayshade ***Reject*** and make it unclickable

Resource Report

- Display *ESF#* and *Primary Emergency Support Function* columns according to ESF Table
- Generate a Resources table grouped by ESF# ('$ESF_number') with Primary Emergency Support ('$ESF') Function, Total Resources

```
SELECT e.Description FROM `Resources` r GROUP BY PrimaryESFNumber
JOIN `ESF` e ON r.PrimaryESFNumber = e.Number;
```

- Query the number of resources in each ESF group from the table obtained in step 2 and display in Total Resources column
- Select resources that are in InUse table from the table obtained in step 2
- Query the number of resources in each ESF group, and display *Resources in Use* column accordingly.

```
SELECT e.Description FROM `Resources` r GROUP BY PrimaryESFNumber
JOIN `ESF` e ON r.PrimaryESFNumber = e.Number
JOIN `InUse` i ON r.ResourceID = i.ResourceID;
```

- Click [X](the close window button) to jump back to **Main Menu**.