Table of Contents

## Login:

Abstract Code

- 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 username, name
FROM User
WHERE User.username = '$username' AND User.password='$password';
```

- If User record is found but password does not match or username does not match:
    - Return to **Login** form with error message.
- Else:
    - Store username and name information as session variables '$username' and '$name' respectively.
    - Go to **Main Menu** form.

- Else *username* or *password* input fields are invalid, display **Login** form, with error message.

## Main Menu:

Abstract Code

- Run the **Main Menu** task: query for information about the user where $username is the ID of the current user.
    - Show User name ($name).
- The following are a set of queries that will find what type of subclass the current user is. Although this is a "case 1" type of relationship from the lectures, in order to support better data consistency "case 4" was used to avoid duplication of data and better login querying.
- Find the Municipality category of username as applicable.
    - Show Municipality category as applicable.

```
SELECT category FROM Municipality WHERE username = '$username';
```

- Find the Government Agency agency name of username as applicable.
    - Show Government Agency agency name and local office as applicable.

```
SELECT local_office, agency_name
FROM GovernmentAgency WHERE username = '$username';
```

- Find the Companies headquarters and number of employees of username as applicable
    - Show Companies headquarters and number of employees as applicable.

```
SELECT headquarters, number_of_employees FROM Company
WHERE username = '$username';
```

- Click **Add a Resource** hyperlink - Jump to **Add new resource** form.
- Click **Add Emergency Incident** hyperlink - Jump to **Add new incident** form.
- Click **Search Resources** hyperlink - Jump to **Search Resources** form.
- Click **Resource Status** hyperlink - Jump to **Resource Status** form.
- Click **Resource Report** hyperlink - Jump to **Resource Report** form.
- Click ***Exit*** button - Invalidate login session and go to **Login** form.

## Add Resource:

Abstract Code

- User clicked on **Add Resource** hyperlink from **Main Menu**:
- Run the **Add a Resource** task: query for information about next Resource ID and name of current logged in user.
    - Find maximum resource ID value, increment it by one, and display it on form. Locally saved as $resource_id.

```
Declare @ResourceID INT;
SET @ResourceID = (SELECT top 1 @ResourceID := id FROM Resource ORDER by id desc);
SET @ResourceID = @ResourceID + 1;
```

    - Display the name of the owner which has been saved as a session variable $name.
        - Display Owner ($name).
- User enters *resource_name* ($resourceName) input field.
- Find list of all ESF numbers and Descriptions.

```
SELECT ESF_designation, ESF_description FROM ESF;
```

- Concatenate to lists into single line of text formatted as (#'ESF_designation') 'ESF_Description'
    - User selects primary_esf input field
    - Program backs out ESF_Designations and sets it to local variable ($primary_ESF).
- Display ESF list minus primary_esf

```
SELECT ESF_designation FROM ESF WHERE ESF.ESF_designation != '$primary_esf';
```

    - User selects secondary_esf input field as applicable.
        - Multi Select available
    - Program backs out ESF_Designation number and creates list ($secondary_ESF).
- User enters model input field as applicable ($model).
- User enters capabilities list input field as applicable ($capabilities).
- User enters the home_location_latitude input field ($home_location_latitude).
- User enters the home_location_longitude input field ($home_location_longitude).
- User enters the max_distance input field as applicable ($max_distance).
- User enters the cost input field as a positive value in US dollars ($amount).
- Find and display Resource cost_per list

```
SELECT unit_of_measure FROM CostPer;
```

- User selects cost_per input field ($unit_of_measure).
- User clicks ***Save*** button - User submitted data is validated. Data is then inserted into Resource, Cost, ESF, and Has Secondary ESF Designation, as applicable.
    - Data Validation
        - All required fields are filled in.
        - The dollar amount is not negative
        - Latitude and Longitude fields contain valid coordinates
    - Owner is saved as current logged in user
        - $owner = $username
    - Cost is saved to entity table

```
INSERT INTO Cost (amount, unit_of_measure)
VALUES (‘$amount’,’$unit_of_measure’);
```

- Determine cost_id for use in resource table ($cost_id).

```
Declare @cost_id INT;
SET @cost_id = (SELECT top 1 @cost_id := cost_id FROM Cost ORDER by id desc);
```

- Insert record into Resource table.

```
INSERT INTO Resource
VALUES
(‘$resource_id’,’$owner’,’$resourceName’,’Available’,’$max_distance’,’$model’,’home_location_latitude’,’home_location_longitude,’$cost_id’,’$primary_ESF’);
```

- If applicable, iterate through list of Secondary ESF Designations ($secondary_ESF) and insert into Has Secondary ESF Designation table.
    - Insert query below may have to be executed multiple times.

```
INSERT INTO HasSecondaryESFDesignation
VALUES (‘$resource_id’, ‘$secondary_ESF’);
```

- If applicable, iterate through list of Capabilities ($capabilities) and insert into ResourceCapability table.
    - Insert query below may have to be executed multiple times.

```
INSERT INTO ResourceCapability
VALUES (‘$resource_id,‘$capabilities’);
```

  - Jump to **Main Menu**.
- Click **Cancel** button - Jump to **Main Menu** form.

## Add Emergency Incident:

Abstract Code

- User clicked on **Add Incident** hyperlink from **Main Menu**:
- Run the **Add an Incident** task: assign Incident owner the value of the logged in user
- Find and display Incident Declaration Descriptions.

```
SELECT declaration_description FROM Declaration;
```

  - User selects the Incident declaration from list ($declaration).
- User enters the date into input fields ($date).
- User enters a description of the incident into input fields ($description).
- User enters the incident location latitude ($location_latitude)
- User enters the incident location longitude ($location_longitude).
- When ***Save*** button is clicked run **Add an Incident** task.

```
Declare @IdNumber INT;
Declare @IdPhrase TINYTEXT;
Declare @DeclationType INT;

SET @DeclarationType = (SELECT declaration_type FROM Declaration WHERE
Declaration.declaration_desicrption = '$declaration')

SET @IdNumber = (SELECT top 1 @IdNumber := id from Incident WHERE Incident.declaration
= @DeclarationType ORDER BY id DESC);

SET @IdNumber = @IdNumber + 1;
SET @IdPhrase = CONCAT(@DeclarationType, " ", @IdNumber;

INSERT INTO Incident (id, description, date, owner, location_latitude, location_longitude,
declaration_type)
VALUES (@IdPhrase, '$description',  '$date', '$username', '$location_latitude',
'$location_longitude', @DeclarationType);
```

- Click **Cancel** button - Jump to **Main Menu** form.

## Search Resources:

Abstract Code

- User clicked on **Search Resources** hyperlink from **Main Menu**:
- Run the **Search Resources** task: populate drop down with ESF values and lists of incidents.
    - Find list of all ESF numbers and Descriptions.

```
SELECT ESF_designation, ESF_description FROM ESF;
```

- 
    - Concatenate to lists into single line of text formetted as (#'ESF_designation') 'ESF_Description'
        - User selects esf input field
    - Find list of all incident ids and names.

```
SELECT id, description FROM Incident;
```

- 
    - Concatenate to lists into single line of text formatted as "('id) 'description'"
- User leaves blank or enters a Keyword for name, model, or capability of Resource in input field ($keyword).
- User selects none or selects an ESF ($ESF).
- User leaves blank or selects an Incident ($incident).
    - If user selects an Incident, user selects a Kilometer value for the location. ($distance)
- User clicks on ***Search*** button Jump to **Search Results** form. Variables from this form are persisted for use in the search results code.

## Search Resources Results:

Abstract Code

- User clicked on **Search** button from **Search Resources**:
- Run the **Search Results** task
- If User chooses the ***Close*** button, close the form and go back to the **Main Menu:**
- Variables from this form are persisted for use in the search results code.
- If all fields are left blank:
    - Run **Search for Resources by all** task

```
SELECT id, name, owner, cost_id, status, Request.expected_return_date FROM Resource
LEFT OUTER JOIN Request ON Resource.id = Request.resource_id;
```

    - Return all ERMS resources currently in the system
- Else if only Keyword is chosen:
    - Run **Search for Resources by keyword** task
    - Return resources containing the keyword(s) taken from resource name, model, and capabilities attributes.

```
SELECT id, name, owner, cost_id, status, Request.expected_return_date FROM Resource
LEFT OUTER JOIN Request ON Resource.id = Request.id_resource WHERE Resource.name =
'$keyword' OR Resource.model='$keyword' OR Resource.id = (SELECT id FROM
ResourceCapability
 WHERE capability = '$keyword');
```

- Else if only ESF is chosen:
    - Run **Search for Resources by ESF** task:
    - Return resources by ESF

```
SELECT id, name, owner, cost_id, status, Request.expected_return_date FROM Resource
LEFT OUTER JOIN Request ON Resource.id = Request.id_resource WHERE
Resource.ESF_primary_designation = '$ESF' OR Resource.id = (SELECT id FROM
HasSecondaryESFDesignation WHERE ESF_Designation = '$ESF');
```

- Else If only an incident value is chosen (with Location value):
    - Run **Search for Resources by proximity** task
    - Calculate distance and return resources within the desired radius from the Incident. ($user_selected_distance)
- Else if multiple search criteria:
    - Task results should be ANDed together.

- If **Search by proximity** task was run:
    - Display Incident Name above the Results output table
- For each Resource found from the Search:
    - Display the **ID, name, owner, cost** and whether or not the resource is currently **in use**.
    - If **Search by keyword** task was run:
        - Display Resources where a matching substring was found from the name, model or capabilities fields.
    - If **Search by ESF** task was run:
        - Display Resources where a matching Primary ESF or Additional ESF was found.

    - If **Search by proximity** task was run:
        - Lookup Incident Location Latitude and Longitude values
        - Lookup Resources home location Latitude and Longitude values
        - Calculate the distances from the Incident to Resources

```
SELECT @lat2 := r.home_location_latitude, @lon2 := r.home_location_longitude, @lat1 :=
i.location_latitude, @lon1 := i.location_longitude,  r.id, r.name, r.owner, c.amount + ‘/’ +
c.unit_of_measure as cost, r.status, r.return_date,
@dlat = @lat2 - @lat1, @dlon = @lon2 - @lon1, @a = POW(SIN(@dlat/2),2) +
(COS(@lat1)*COS(@lat2)*POW(SIN(@dlon/2),2)), @c =
2*POW(ATAN2(SQRT(@a),SQRT(1-@a)),2), @d = 6371*RADIANS(@c) AS distance,
req.expected_return_date
 FROM Resource r,
Incident i,
Request req
JOIN Cost c ON c.cost_id = r.cost_id
JOIN ESF esf ON esf.ESF_ID = r.primary_esf
JOIN ESF esf2 ON esf2.ESF_ID = r.additional_esf
 WHERE
 (r.name = ‘$keyword’ OR r.model = ‘$keyword’ OR r.capability = ‘$keyword’)
AND
 (r.ESF = ‘$PrimaryESF_ID’ OR r.ESF = ‘$SecondaryESF_ID’)
AND
r.id IN (SELECT Resource.Id, @dlat = @lat2 - @lat1, @dlon = @lon2 - @lon1, @a =
POW(SIN(@dlat/2),2) + (COS(@lat1)*COS(@lat2)*POW(SIN(@dlon/2),2)), @c =
2*POW(ATAN2(SQRT(@a),SQRT(1-@a)),2), @d = 6371*RADIANS(@c) FROM Resource,
Incident WHERE @d <= ‘$user_selected_distance’)
Order by distance asc, req.expected_return_date, r.name;
```

- If any combination of tasks were run display output of all tasks that were run
- Lookup Deployed state of Resource

        - If Resource is owned by the current user AND the Resource is NOT currently in use, then ***Deploy*** button is displayed in the Action column.
        - Else, ***Request*** button will be displayed in the Action column.
        - If Resource is NOT currently in use, then display AVAILABLE in the Status column and NOW in Next Available column.
        - Else, display IN USE in the Status column.
            - Lookup Expected Return date list
            - Determine the latest date from that list
            - Display the latest Expected Return date in the Next Available column.

- Sort Resources, first by distance from shortest to longest, then alphabetically.
- Display Resources that have calculated distances equal to or less than the Requested Location distance with the calculated distance value in the Distance column of the results table.
- For any ***Deploy*** button shown in the Action column:
    - Do nothing
- For any ***Request*** button shown in the Action column:
    - Do nothing
- Upon ***Deploy*** button being chosen by the user:
    - Display **Resource Deployed Form**
        - User Input: Resource Expected Return Date($expected_return_date)
        - Display Resource Status
        - Display Resource name
        - Display Incident description
    - If User chooses the ***Cancel*** button, close the form and go back to the **Search Resource Results:**
    - Validate the current user owns the Resource
    - Validate the Resource status is Available
    - If User chooses the ***Deploy Resource to Incident*** button, close the form and go back to the **Search Resource Results**:
        - Run the **Resources currently in use** task.
            - Store Resource Status value as IN USE
            - Store Resource Expected Return Date value provided by user
            - Store Resource Start Date value as today's date

```
INSERT INTO Resource (status)
SELECT ‘IN USE’
WHERE Resource.status = ‘Available’ AND Resource.owner=’$username’ AND Resource.Id =
‘$resource_id’;

INSERT INTO Deployed (incident_id, resource_id, date_deployed) SELECT Incident.id,
Resource.id, CURDATE() WHERE Deployed.incident_id = Incident.id AND
Deployed.resource_id=Resource.id;

INSERT INTO Request (expected_return_date)
VALUES('$expected_return_date');
```

        - Display updated status value in Status column for the requested resource
        - Display updated Expected Return date value in Next Available column for the requested resource
        - Run the **Resource Requests received by me** task.
- Upon ***Request*** button being chosen by the user,  jump to **Request Resource** form.

## Resource Status:

Abstract Code

- User clicked on **Resource Status** hyperlink from **Main Menu**:
- If User chooses the ***Close*** button, close the form and go back to the **Main Menu:**
- Run **Resources currently in use** task.

```
SELECT r.id, r.name, i.name, r.owner, dep.date_deployed, req.expected_return_date
FROM Deployed dep
LEFT JOIN Resource r ON r.id = dep.id_resource
LEFT JOIN Incident i ON i.id = dep.id_resource
LEFT JOIN Request req
ON req.id_resource = dep.id_resource AND req.id_incident = dep.id_incident
WHERE r.status = 'IN USE' AND i.owner = '$username';
```

- Run **Resources Requested** task.

```
SELECT r.id, r.name, i.description, r.owner, req.expected_return_date
FROM Request req
LEFT JOIN Resource r ON r.id = req.resource_id
LEFT JOIN Incident i ON i.id = req.incident_id
WHERE req.username = '$username';
```

- Run **Resource Requests received** task.

```
SELECT r.id, r.name, i.description, req.username, r.status
FROM Request req
LEFT JOIN Resource r ON r.id = req.id_resource
LEFT JOIN Incident i ON i.id = req.id_incident
LEFT JOIN User u ON u.username = req.username
WHERE r.owner = '$username';
```

- Run **Users Response to Resource Request** task.
- For each Resource found from **Resources currently in use**:
  - For each Incidence owned by the current user:
    - Display the **ID, name, incident responding to**, and **owner**.
    - Display ***Return*** button in the Action column.
- For each Requested Resource found from **Resources Requested**:
  - If Resource Request has NOT been responded to:
    - Display the **ID, name**, related **incident** and **owner** for the current user only.

        - Display ***Cancel*** button in the Action column.
- For each Requested Resource found from **Resource Requests received**:
    - If user has NOT responded to Resource Request:
        - Display the **ID, name**, related **incident** and **requesting user**.
        - Display ***Reject*** button in the Action column.
        - If Requested Resource has Available Status
            - Display ***Deploy*** button in the Action column
- For any ***Return*** button shown in the Action column:
    - Do nothing
- For any ***Cancel*** button shown in the Action column:
    - Do nothing
- For any ***Deploy*** button shown in the Action column:
    - Do nothing
- For any ***Reject*** button shown in the Action column:
    - Do nothing
- Upon ***Deploy*** button being chosen by the user:
    - Display Resource Deploy Form
        - User Input: Resource Expected Return Date
        - Display Resource Status as Available
        - Display Resource name
        - Display Incident description
    - If User chooses the ***Cancel*** button, close the form and go back to the **Resource Status:**
    - Validate the current user owns the Resource
    - Validate the Resource status is Available
    - If User chooses the ***Deploy Resource to Incident*** button, close the form and go back to the **Resource Status**:
        - Run the **Resources currently in use** task.
            - Store Resource Status value as IN USE
            - Store Resource Expected Return Date value provided by user
            - Store Resource Start Date value as today's date
    - Lookup and Display Resources in use
    - Run **Resource Requests Received** task:
        - Lookup and Display Resource Requests Received.
- Upon ***Reject*** button being chosen by the user:
    - Delete Resource Requests from Requests table.

```
DELETE from Request req WHERE req.id_resource = '$resource_id' AND req.id_incident =
'$incident_id';
```

    - Run **Resource Requests received** task:
        - Delete Requested Resource

- Upon ***Cancel*** button being chosen by the user:
    - Delete Resource Requests from Requests table.:
        - Delete Requested Resource

```
DELETE from Request req WHERE req.id_resource = ‘$resource_id’ AND req.id_incident =
‘$incident_id’
```

- Upon ***Return*** button being chosen by the user:
    - Run **Return Resource to available status** task:
        - Modify Resource Status to Available

```
UPDATE Resource
SET status = ‘AVAILABLE’
WHERE id = ‘$resource_id’;
```

- Modify Deploy date returned to current date.

```
UPDATE Deploy
SET actual_return_date = CURDATE()
WHERE id_resource = ‘$resource_id’ AND  actual_return_date = NULL;
```

- Delete request row from request table.

```
DELETE from Request req WHERE req.id_resource = ‘$resource_id’ AND req.id_incident =
‘$incident_id’;
```

## Resource Request:

Abstract Code

- If User chooses the ***Cancel*** button, close the form and go back to the **Search Resources Results** form.
- Lookup all existing Resource Requests for this resource

```
SELECT r.id, r.name, r.owner FROM Request req
JOIN Resource r ON r.id = req.id_resource
 WHERE req.id = '$resource_id';
```

- Display all existing Resource Requests for this resource
- User Input: Resource Expected Return Date
- Display Resource name
- Display Incident description

- If User chooses the ***Request Resource to Incident*** button:
    - Run the **Resources Requested** task.

```
INSERT INTO Request (username, id, expected_return_date,)
VALUES ('$username', '$resource_id', '$expectedreturndate');
```

    - Display popup window "Resource has been requested"
        - User chooses "OK" to close the popup window
    - Resource Request Form is closed.
    - go back to the **Search Resource Results** form:

## Resource Deployed:

Abstract Code

- If User chooses the ***Cancel*** button, close the form and go back to the **Resource Status** form or **Search Resources Results** form**:**
- Lookup all existing Resource Requests for this resource
- Display all existing Resource Requests for this resource
- User Input: Resource Expected Return Date
- Display Resource Status
- Display Resource name
- Display Incident description

```
SELECT Resource.status, Resource.name, Incident.description FROM Resource, Incident
WHERE Resource.id='$resource_id';
```

- Validate the current user owns the Resource
- Validate the Resource status is Available
- If User chooses the ***Deploy Resource to Incident*** button, close the form and go back to the **Resource Status** form or **Search Resources Results** form:
    - Run the **Resources currently in use** task.
        - Store Resource Status value as IN USE
        - Store Resource Expected Return Date value provided by user
        - Store Resource Start Date value as today's date

## Resource Report Summary:

Abstract Code

- If User chooses the ***Close*** button, close the form and go back to the Main Menu.
- Lookup ESF list
- Lookup all resources owned by the current user
- Display list of all ESFs.
- Display total resources for each ESF
- Display total resources in use for each ESF
- Calculate and Display total resources for all ESFs
- Calculate and Display total resources in use for all ESFs

```
SELECT Resource.ESF_primary_designation,ESF.ESF_description,
        SUM(CASE when Resource.owner='$username' then 1 else 0 end) “Total Resources”,
        SUM(CASE when Resource.status='IN USE' then 1 else 0 end) “Resources In Use”,
FROM Resource
JOIN ESF ON ESF.designation = Resource.ESF_primary_designation
GROUP BY Resource.ESF_primary_designation;
```

Note: The totals at the bottom of the report will be calculated from the returned dataset in code.