Table of Contents

Abstract Code w/ SQL

Login

Abstract Code

- Show **Log In** form
- User populates *username* ('$username'), *password* ('$password') input fields
- If data validation is successful for both *username* and *password* input fields:
    - When ***Log In*** button is clicked:

```
SELECT User.username,User.password
FROM User
WHERE User.username= '$username';
```

- If User record is not found:
    - Go back to **Log In** form, with error message.
- Else If User.*password* != '$password':
    - Go back to **Log In** form, with error message.
- Else:
    - Store login information as session variable '$username'.

Return to Table of Contents

    ◦ Go to **Main Menu** form.
- Else If both *username* and *password* input fields are invalid, then go show **Log In** form with error message

Main Menu

Abstract Code

- Query for information about the user and their profile where $username is the ID of the current user using the system from the HTTP Session/Cookie.

```
SELECT User.name, User.username
FROM User
WHERE User.username= '$username';
```

- Show **Main Menu** form
    - Find the current User using the User.username and Display User.name

```
SELECT User.name, User.username,
    Municipality.municipality_category,
    Government_Agency.agency_name_and_local_office,
  Company.location_of_headquarters,
  Company.num_employees
FROM User
LEFT JOIN Municipality ON User.username = Municipality.username
LEFT JOIN Government_Agency
   ON User.username = Government_Agency.username
LEFT JOIN Company ON User.username = Company.username
WHERE User.username = '$username`;
```

    - If Municipality.municipality_category != NULL, display municipality_category
    - Else If Government_Agency.agency_name_and_local_office != NULL, display Government_Agency.agency_name_and_local_office
    - Else If Company.location_of_headquarters != NULL, display both Company.location_of_headquarters and Company.number_of_employees.
- Upon:
    - Click ***Add Resource*** button - Jump to the ***Add Resource*** task.
    - Click ***Add Emergency Incident*** button - Jump to the ***Add Emergency Incident*** task.
    - Click ***Search Resources*** button - Jump to the ***Search Resources*** task.
    - Click ***Resource Status*** button - Jump to the ***Resource Status*** task.
    - Click ***Resource Report*** button - Jump to the ***Resource Report*** task.
    - Click ***Exit*** button - Logs user out of system and displays the **Login** form.

Return to Table of Contents

Add Resource

Abstract Code

- User clicked on ***Add Resource*** button from **Main Menu**
- Query for information about the user and their profile where $Username is the ID of the current user using the system from the HTTP Session/Cookie.

```
SELECT User.name, User.username
FROM User
WHERE User.username= '$username';
```

- Show **Add New Resource** form
  - Display User.name as Owner
  - Create unique numeric Resource ID ($ResourceID) **and** Display Resource ID
  - Display list of cost_options in $cost_per dropdown field.

```
SELECT Cost_Per.cost_option
FROM Cost_Per;
```

  - Display list of selectable ESF numbers along with their descriptions in the $primary_esf dropdown field.

```
SELECT Allowable_ESFs.esf_number, Allowable_ESFs.esf_descriptio
FROM Allowable_ESFs;
```

  - When User selects *Primary ESF* ($primary_esf), display list of selectable ESF numbers along with their descriptions in the $additional_esf dropdown field except for the selected primary ESF.

```
SELECT Allowable_ESFs.esf_number, Allowable_ESFs.esf_descriptio
FROM Allowable_ESFs
WHERE Allowable_ESFs.esf_number != $primary_esf;
```

- User enters *Resource Name* ($name), selects *Primary ESF* ($primary_esf), selects *Additional ESFs* ($additional_esfs), enters *Home Location* ($latitude and $longitude), enters *Cost* ($cost), selects *Cost/per* ($costper), optionally enters Model ($model), enters Capabilities ($capabilities), and enters *Maximum Distance* ($maximum_distance).
- Upon click **Cancel button** - Jump to the **Main Menu** form
- If data validation is successful for all fields, then:
  - Upon click **Save** button:

        - Store resource information as row in Resources and set Resource.res_status as “Available”

```
INSERT INTO Resource
VALUES ($username, $ResourceID, $name, $model,
    $capabilities, $latitude, $longitude, $cost,
    $maximum_distance, “Available”, $Costper);
```

        - Store primary ESF information as row in ESFs with a “Primary” esf_type.

```
INSERT INTO ESFs
VALUES ($ResourceID, $primary_esf, “Primary”);
```

        - For each user selected additional ESF, as $additional_esf, in the list $additional_esfs, store information as row in ESFs with an “Additional” esf_type.

```
INSERT INTO ESFs
VALUES ($ResourceID, $additional_esf, “Additional”);
```

- Else if required fields not selected or any input fields invalid, display **Add New Resource** form with error message

Add Emergency Incident

Abstract Code

- User clicked on ***Add Emergency Incident*** button from **Main Menu**
- Query for information about the user and their profile where $Username is the ID of the current user using the system from the HTTP Session/Cookie.

```
SELECT User.name, User.username
FROM User
WHERE User.username= '$username';
```

- Show **New Incident** form
    - Display list of declarations in dropdown.

```
SELECT Incident_Declarations.declaration,
   Incident_Declarations.abbreviation
FROM Incident_Declarations;
```

Return to Table of Contents

    - User selects *Declaration* ($declaration), enters *Date* ($date), enters *Description* ($description), enters *Location* ($lat and $long).
- Upon click **Cancel button** - Jump to the **Main Menu** form
- If data validation is successful for all fields, then:
    - Upon click **Save** button:
        - Store incident information as row in Incident
        - Create unique *IncidentID* ($incidentid)
            - Find *declaration* ($declaration) input by user and concatenate its abbreviation ($abbreviation) with autogenerated numeric unique ID
            - Store as *incidentid* ($incidentid)

```
INSERT INTO Incident
VALUES ($incidentid, $username, $date, $description,
    $Latitude, $Longitude, $abbreviation);
```

- Else if required fields not selected or any input fields invalid, display **New Incident** form with error message

Search Resources

Abstract Code

- User clicked on ***Search Resources*** button from **Main Menu** form
- Query for information about the user and their profile where $username is the ID of the current user using the system from the HTTP Session/Cookie.

```
SELECT User.name, User.username
FROM User
WHERE User.username= '$username';
```

- Show **Search** form
    - Displays text input for search keywords
    - Displays dropdown of ESF functions
        - Find all unique ESF functions from Look-up on Allowable_ESFs.esf_number and Allowable_ESFs.esf_description and display

```
SELECT Allowable_ESFs.esf_number,
   Allowable_ESFs.esf_description
FROM Allowable_ESFs;
```

Return to Table of Contents

- Displays text input for proximity to incident field
- Displays dropdown of User-owned Incidents
  - Find the current User using the User.username and find all unique incidents that the User owns by look-up on Incidents table on Incident.ID
  - Look up Incident.ID and Incident.Description, concatenate, and display in dropdown

```
SELECT User.name, User.username,
      Incident.incident_id,
    Incident.inc_description
FROM User
LEFT JOIN Incident ON User.username = Incident.username
WHERE User.username = '$username`;
```

- Show ***Cancel*** and ***Search*** buttons

- User optionally enters keywords, optionally selects ESF function, optionally selects proximity to emergency incident by entering a distance and selecting an incident from incident list dropdown
- Upon
  - Click ***Cancel*** button - Jump to the **Main Menu** form
  - Click ***Search* button** - Jump to the ***Search Results for Incident*** subtask

Search Results for Incident

Abstract Code

- User clicked on ***Search*** button from **Search Resources** and query was successful
  - Return query based on search criteria, all search criteria must be matched using "and" conditions:
    - If all fields are empty, read-only on resources table and return all resources
    - If User inputs keywords, Find matching substrings in Resource.model, Resource.capabilities, and Resource.name

    - If User selects ESF, Find matching ESF_Num in ESFs.
    - If User selects Incident:
      - If User selected distance, then find resources that are within the selected distance of incident by finding Resource.lat and Resource.lon and calculating distance of each resource from Incident.Location

```
SET @Lat1 := Incident.loc_lat FROM Incident
   WHERE Incident.incident_id = $Incident;
```

```
SET @Lon1 := Incident.loc_long FROM Incident
      WHERE Incident.incident_id = $Incident;

SELECT DISTINCT Resource.resource_id, Resource.name, Resource.username,
   Resource.cost, Resource.status, Resource.max_dist,
   IFNULL(Requests.return_by, “NOW”)
   @Lat2 := Resource.home_loc_lat,
   @Lon2 := Resource.home_loc_long,
   @dLat := RADIANS( @Lat2 ) - RADIANS ( @Lat1 ),
   @dLon := RADIANS( @Lon2 ) - RADIANS ( @Lon1 ),
   @A := POW(SIN( @dLat / 2),2) + POW(SIN( @dLon / 2),2) *
      COS( RADIANS( @Lat1 ) ) * COS(RADIANS ( @Lat2 ) ),
   @C := 2 * ATAN2( SQRT( @A ), SQRT(1-@A ) ),
   @distance := 6371 * @C
FROM Resource
LEFT JOIN ESFs ON Resource.resource_id = ESF.resource_id
LEFT JOIN Requests ON Resource.resource_id = Requests.resource_id
WHERE ($keywords is NULL OR $keywords LIKE Resource.model OR
   $keywords LIKE Resource.capabilities OR $keywords LIKE Resource.name)
   AND ($ESF is NULL OR $ESF LIKE ESFs.esf_number)
   AND (Requests.req_status = “Deployed”)
   AND (@distance < $within AND (Resource.max_dist is NULL OR
      @distance < Resource.max_dist) )
ORDER BY @distance, Resource.status, Resource.name
```

- If distance is not input, then distance is not calculated

```
SELECT DISTINCT Resource.resource_id, Resource.name, User.username,
   Resource.cost, Resource.res_status, ISNULL(Requests.return_by, “NOW”)
FROM Resource
LEFT JOIN ESFs ON Resource.resource_id = ESF.resource_id
LEFT JOIN Requests ON Resource.resource_id = Requests.resource_id
WHERE ($keywords is NULL OR $keywords LIKE Resource.model OR
   $keywords LIKE Resource.capabilities OR $keywords LIKE Resource.name)
   AND ($ESF is NULL OR $ESF LIKE ESF.esf_number)
   AND (Requests.req_status = “Deployed”)
ORDER BY Resource.status, Resource.name
```

- Display **Search Results for Incident** form displaying incident name, Table of results, and **close** button
  - Incident Name
    - If User had selected Incident in **Search Resources** form, Display Incident.inc_description and Incident.incident_id

```
SELECT Incident.incident_id, Incident.inc_description
```

Return to Table of Contents

```
FROM Incident
WHERE Incident.incident_id= '$Incident';
```

        - Else leave blank

    - Table of results - Present the query results from the ***Search for Resources*** Task in a tabular format. Sort results by distance, availability, and then name.
        - Always display ID (Resource.resource_id), Name (Resource.name), Owner (User.username who owns Resource), Cost (Resource.cost), Status (Resource.res_status), Next Available (Requests.return_by) columns
        - If Resource.res_status is Available:
            - Display text "NOW" in Next Available Column
        - If Resource.res_status is not Available:
            - Display Requests.return_by
        - If User had input incident and distance in **Search Resources** form, also Display Distance columns
            - Display Distance of each resource, present query results from the ***Search for Resources*** Task
        - If User had input incident in **Search Resources** form, also Display Action columns

            - In Action column, Display **Request**, **Deploy**, or no button
                - If Resource.res_status is available:
                    - If the resource owner $user_id and incident owner $user_id match:
                        - Display a **Deploy** button under the Action column.
                    - Else if no request is found in query results for specific resource by current User:
                        - Display a **Request** button under the Action column.
                    - Else:
                        - Display an empty cell under the Action column.
    - Upon click **Close button** - Jump to the ***Search Resources*** task
    - Upon click **Request button** - Jump to the ***Request Resource*** task
    - Upon click **Deploy button** - Jump to the ***Deploy Resource*** task

Request Resource

Abstract Code

Return to Table of Contents

- Upon click **Request** button for a table record:
    - Prompt user for an expected return date
        - If User enters return date and clicks “enter”:
            - Store return date information as a new record in Requests.return_by
            - Store return date information as a new record in Requests
            - Leave start date blank
            - Store Requests.req_status as “pending”
            - Return to **Search Results for Incident** form and Remove **Request** button for the table record

```
INSERT INTO Requests
VALUES ($resource_id, $incident_id, $request_date, $deployed_date, $return_by, “Pending” );
```

        - Else if User closes prompt, return to **Search Results for Incident** form

Deploy Resource

Abstract Code

- Upon click **Deploy** button for a table record:
    - Prompt user for an expected return date
        - If User enters return date and clicks “enter”:
            - Store return date information as a new record in Requests.return_by
            - Store start date as Requests.deployed_date as the current date
            - Store Requests.req_status as “deployed”
            - Return to **Search Results for Incident** form and Remove **Deploy** button for the table record

```
UPDATE Requests
SET
   Requests.return_by = $return_by,
   Requests.req_status = “Deployed”,
   Requests.deployed_date = CURDATE()
WHERE Requests.resource_id = $resource_id AND Requests.incident_id = $incident_id;
```

        - Else if User closes prompt, return to **Search Results for Incident** form

Return to Table of Contents

Resource Status

Abstract Code

- User clicked on ***Resource Status*** button from **Main Menu**
- Display Resources in Use table, Resources Requested by Me table, and Resource Requests Received by Me table
- Show **Resources in Use** Table
    - Find all resources that User had requested and is using (where Resource.res_status is "In-Use"):
        - Display ID (Resource.resource_id), name (Resource.name), incident (Incident.name), start date (Requests.request_date), and return date (Requests.deployed_date)

```
  SELECT Resource.resource_id, Resource.name, Incident.inc_description,
           User.name, Requests.deployed_date, Requests.return_by
  FROM Incident
LEFT JOIN Requests ON Requests.incident_id = Incident.incident_id
LEFT JOIN Resource ON Resource.resource_id = Requests.resource_id
LEFT JOIN User ON User.username = Resource.username
  WHERE Incident.username = '$username' AND
  Requests.req_status = 'Deployed';
```

-
    -
        - Under Action column - Display **Return** button
    - Upon click **Return button** - Jump to the ***Return Resource*** task
- Show **Resources Requested by me** Table
    - Find all resources in Resources table where Requests.status is "Pending" and Requests.username matches User.username
        - Display ID (Resource.ID), name (Resource.name), related incident (Incident.inc_description), owner (User.username) and return date (Requests.return_by)
        - Under Action column - Display **Cancel** button

```
  SELECT Resource.resource_id, Resource.name, User.name,
  Incident.inc_description, Request.return_by
  FROM Incident
LEFT JOIN Requests ON Requests.incident_id = Incident.incident_id
LEFT JOIN Resource ON Resource.resource_id = Requests.resource_id
LEFT JOIN User ON User.username = Resource.username
  WHERE Incident.username = '$username' AND
  Requests.req_status = 'PENDING';
```

-
    - Upon click **Cancel button** - Jump to the ***Cancel Resource Request*** task

- Show **Resource Requests Received by Me** Table

Return to Table of Contents

- Find all resources that User owns in Resource table where Requests.req_status is “Pending”
- Display ID (Resource.resource_id), name (Resource.name), related incident (Incident.name), owner (User.username), return date (Requests.return_by) and Action columns.

```
  SELECT Resource.resource_id, Resource.name, User.name,
  Incident.inc_description, Request.return_by
  FROM Requests
  LEFT JOIN Resource ON Resource.resource_id = Requests.resource_id
LEFT JOIN Incident ON Incident.incident_id = Requests.incident_id
LEFT JOIN User ON Incident.username = User.username
  WHERE Resource.username = '$username' AND
  Requests.req_status = 'PENDING';
```

    - If Resource.res_status is “Available”
        - Display “**Deploy**” and “**Reject**” buttons under Action Columns
    - Else if Resource.res_status is “In Use”
        - Display “**Reject**” buttons under Action Columns
    - Upon click **Cancel button** - Jump to the ***Cancel Resource Request*** task
    - Upon click **Deploy** button - Jump to the ***Deploy Resource*** task
    - Upon click **Reject button** - Jump to the ***Reject Resource Request*** task

Return Resource

Abstract Code

- Upon click **Return** button
    - Update Requests.req_status to “Returned”

```
UPDATE Requests
SET Requests.req_status='Returned'
WHERE Resource.resource_id = '$Resource.ID'
AND Incident.incident_id = '$Incident.ID' ;

UPDATE Resource
SET Resource.res_status='Available'
WHERE Resource.resource_id = '$Resource.ID' ;
```

    - Update Resource.res_status to “Available”
    - Remove resource from displaying on **Resource in Use** table

Cancel Resource Request

Abstract Code

- Upon click **Cancel** button
    - Delete record from Requests table
    - Remove display of record from **Resources Requested by me** Table

```
DELETE FROM Requests
WHERE Resource.resource_id = '$Resource.ID'
AND Incident.incident_id = '$Incident.ID' ;
```

Return to Table of Contents

Reject Resource Request

Abstract Code

- Upon click **Reject** button:
    - Delete record from Requests table
    - Remove display of record from **Resources Received by Me** Table

```
DELETE FROM Requests
WHERE Resource.ID = '$Resource.ID'
   AND Incident.ID = '$Incident.ID' ;
```

Resource Report

Abstract Code

- User clicked on ***Resource Report*** button from **Main Menu**
- Query for information about the user and their resources where $UserID is the ID of the current user using the system from the HTTP Session/Cookie
- Display **Resource Status** table with columns ESF#, Primary ESF, Total Resources, Resources in Use
    - Display ESF # and Primary ESF columns
    - Display Total Resources and Resources In Use column
        - Sum number of resources If User owns Resource, Display sum by ESF# in table row

```
SELECT Allowable_ESFs.esf_number AS ESF_Number
   Allowable_ESFs.esf_description AS Primary_ESF,
   COUNT(Resource.resource_id) AS Total_Resources,
   COUNT ( IF (Resource.res_status="In-Use", 1, null) ) AS Resource_In_Use
FROM Allowable_ESFs
LEFT JOIN ESFs ON Allowable_ESFs.ESF_number = ESFs.esf_number
LEFT JOIN Resource ON Resource.resource_id = ESFs.resource_id
WHERE ESFs.esf_type = "Primary" AND
   Resource.username = $username
  GROUP BY Allowable_ESFs.esf_number, Allowable_ESFs.esf_description
  ORDER BY Allowable_ESFs.esf_number;
```

- Display last row of Table as Total Row
    - Display Sum of "Total Resources" column and Sum of "Resources in Use" column

Return to Table of Contents