Georgia Tech

Master of Science in Analytics

Emergency Resource Management System

Phase 2 Abstract Code w/SQL | CS 6400 | Team 05

Table of Contents

Abstract Code (AC)

Login

Abstract Code

- Let '*$username*' represent the input from the Username field
- Let '*$password*' represent the input from the Password field
- User enters values into the Username and Password fields
- User selects the ***Login*** button
- Validate all fields
    - All required fields are filled in
- If all fields are valid:

```
SELECT password FROM users WHERE username = '$username';
```

    - If user record is found but user.password != '*$password*':
        - Display a "wrong password" error message and return the user to the **Login** form
    - Else if no record is found:
        - Display a "username does not exist" error message and return the user to the **Login** form:
    - Else:
        - Store '*$username*' as session variable
        - Go to **Main Menu** form
- Else:
    - Display a "Missing username and/or password" error message and return the user to the **Login** form

Main Menu/Navigation Bar

Abstract Code

- Show **"*Add Resource*"**, **"*Add Emergency Incident*"**, **"*Search Resources*"**, **"*Resource Status*"**, **"*Resource Report*"**, **"*Exit*"** buttons
- Lookup information about the user name and other details depending on user type from the HTTP Session/Cookie

```
SELECT A.name, B.top_line, B.bottom_line FROM
        users A INNER JOIN (
                SELECT username AS username, hq_location AS top_line, num_employees AS bottom_line
                        FROM companies UNION
                SELECT username AS username, category AS top_line, NULL
                        FROM municipalities UNION
                SELECT username AS username, job_title AS top_line, hire_date AS bottom_line
                        FROM individuals UNION
                SELECT username AS username, agency_name_local_office, NULL
                        FROM government_agencies) B
        ON A.username = B.username WHERE A.username = '$username';
```

- Display the name of the logged in user on the 1st line
- Display the hq_location/category/job_title/agency_name_local_office on the 2nd line
- Display the num_employees/hire_date/{blank if '*B.bottom_line*' is NULL} on the 3rd line
- Upon:
    - Click ***Add Resource*** button - Jump to the **Add Resource** task
    - Click ***Add Emergency Incident*** button - Jump to the **Add Incident** task
    - Click ***Search Resources*** button - Jump to the **Search for Resources** task
    - Click ***Resource Status*** button - Jump to the **View Resource Status** task
    - Click ***Resource Report*** button - Jump to the **Generate Resource Report** task
    - Click ***Exit*** button - close session and return the User to the **Login** form

Add Resource

Abstract Code

- User clicks on ***Add Resource*** button from **Main Menu**
- Show *Owner, Resource Name, Primary ESF, Additional ESFs, Model, Capabilities, Home Location, Max Distance, and Cost* input fields
- Show ***Add, Cancel,*** and ***Save*** buttons
- Run the **Populate Add Resource Form ESF** subtask:
    - Lookup information to populate the *Primary ESF* field, a dropdown box that includes a list of preloaded ESF values allowing for selection of 1 value

```
SELECT esf_id, description FROM esfs;
```

    - Use the results from the query to populate the *Additional ESFs* field, a dropdown box that includes a list of preloaded ESF values allowing for selection of 0 or many values
    - Format both dropdown values
    - Set a default selection value of "(#1) Transportation" for the *Primary ESF* field
    - Let *'$primary_esf_id'* represent the ID from the selected Primary ESF field value
    - Remove the selected *Primary ESF* value as an option from the *Additional ESFs* field through application code
    - Let *'$secondary_esf_ids'* represent a list of IDs from the selected Additional ESFs field
- Run the **Populate Add Resource Form Cost Pers** subtask
    - Lookup information to populate the *Cost Per* field, a dropdown box that includes a list of preloaded selection for units of time

```
SELECT cost_per FROM cost_pers;
```

    - Set a default selection value of "Hour"
- Let *'$cost_per'* represent the selection from the *Cost Per* field
- Let *'$resource_capabilities'* represent a list of added capability description values from the *Capabilities* field
- Upon:
    - Click ***Add*** button
        - Add input from the *Capabilities* field into '*$resource_capabilities*'
            - Allows for multiple text inputs
    - Click ***Cancel*** button
        - Exit out of **Resource Add** form and go back to **Main Menu**
    - Click ***Save*** button
        - Validate all fields before inserting new resource to the database:
            - All required fields are filled in
            - Dollar amount in *Cost* is not negative
            - *Max Distance* is not negative
            - Latitude and Longitude for *Home Location* contain valid coordinates

            ○ Latitude is in range [-90, 90], and Longitude is in range [-180, 180]
    ■ If all fields are valid:
        ● Let *'$resource_id'* represent the *Resource ID* field, calculated below
        ● Let *'$resource_name'* represent the *Resource Name* field
        ● Let *'$model'* represent the *Model* field
        ● Let *'$latitude'* represent the *Latitude* field
        ● Let *'$longitude'* represent the the *Latitude* field
        ● Let *'$max_distance'* represent the *Maximum Distance* field
        ● Let *'$cost'* represent the *Cost* field
        ● Run **Add New Resource** subtask to insert and save resource

```
INSERT INTO resources (owner, name,latitude, longitude, model, cost, cost_per, maximum_distance,
primary_esf_id)
        VALUES ('$username', '$resource_name', '$latitude', '$longitude', '$model', '$cost', '$cost_per',
        '$max_distance', '$primary_esf_id);
```

- **Note**: The resource_id field will be set to the next increment value by default
- **Note**: The availability_status field will be set to 'Available' by default

```
'$resource_id' = SELECT MAX(resource_id) FROM resources;
```

Loop through each value in the '*$secondary_esf_ids*' list represented by '$secondary_esf_id' and run the following:

```
INSERT INTO resource_secondary_esfs (resource_id, esf_id) VALUES ('$resource_id', '$secondary_esf_id');
```

Loop through all the values in the '$resource_capabilities' list represented by '$resource_capability' and run the following:

```
INSERT INTO resource_capabilities (resource_id, capability) VALUES ('$resource_id', '$resource_capability');
```

    ■ Else:
        ● Display warning message to User

Add Incident

Abstract Code

- User clicked on ***Add Emergency Incident*** button from **Main Menu**:
- Show *Declaration, Date, Description, Location* input fields.
- Show ***Cancel*** and ***Save*** buttons.
- Run the **Populate Add Incident Declaration Form** subtask
  - Lookup information to populate the *Declaration* field, a dropdown box that includes a list of incident type values allowing for selection of 1 value

```
SELECT abbreviation, description FROM incident_types;
```

  - Set a default selection value of "Emergency"
  - Let *'$abbreviation'* represent the ID from the selected Declaration field value
- Upon:
  - Click ***Cancel*** button
    - Exit out of **Add Incident** form and go back to **Main Menu**
  - Click ***Save*** button
    - Validate all fields before inserting new incident to the database:
      - All required fields are filled in
      - *Date* is valid date
      - Latitude and Longitude for *Home Location* contain valid coordinates
        - Latitude is in range [-90, 90], and Longitude is in range [-180, 180]
    - If all fields are valid:
      - Let *'$incident_id'* represent the Incident ID field, calculated below
      - Let *'$incident_date'* represent the Date field
      - Let *'$description'* represent the Description field
      - Let *'$latitude'* represent the *Latitude* field
      - Let *'$longitude'* represent the the *Latitude* field
      - Run **Add New Incident** subtask to insert and save incident

```
'$incident_id' = SELECT CASE WHEN MAX(incident_id) IS NULL THEN 1 ELSE MAX(incident_id) + 1 END
FROM incidents WHERE abbreviation = '$abbreviation';

    ● Note: Case statement used in case there isn't an existing incident with the abbreviation to set a default
      value of 1

INSERT INTO incidents (abbreviation, incident_id, owner, incident_date, description, latitude, longitude) VALUES
        ('$abbreviation', '$incident_id', $username', '$incident_date', '$description', '$latitude', '$longitude');
```

    - Else:
      - Display warning message to User

Search for Resources

Abstract Code

- User clicked on ***Search Resources*** button from **<u>Main Menu</u>:**
- Show *Keyword, ESF, Location, Incident* input fields.
- Show ***Cancel*** and ***Search*** buttons.
- For the *Location field* show **Up** and **Down** toggle buttons that increase or decrease the field value by 1
- Set a default selection value of 15 for the *Location* field
- Run the **Populate Search ESF Form** subtask
    - Lookup information to populate the *ESF* field, a dropdown box that includes a list of preloaded ESF values allowing for selection of 1 value

```
SELECT esf_id, description FROM esfs;
```

- Format dropdown values
- Run the **Populate Search Incident Form** subtask
    - Lookup information to populate the *Incident* field, a dropdown box that includes a list of incident abbreviation, id and description values allowing for selection of 1 value

```
SELECT abbreviation, incident_id, description FROM incidents;
```

- Format dropdown values
- Upon:
    - Click ***Search*** button
        - If all search criteria is blank:
            - Run **Display Search Results** subtask, return all potential results and only include the following columns:
                - *ID*
                - *Name*
                - Owner
                - *Cost*
                - *Status*

```
SELECT A.resource_id, A.name, A.owner, A.cost, A.availability_status, B.expected_return_date FROM
        resources A LEFT JOIN (
                SELECT resource_id, expected_return_date FROM resource_requests WHERE
                        request_status = 'Deployed') B ON
        A.resource_id = B.resource_id;
```

        - Else:
            - For fields that have input, validate and verify fields before querying the database:
                - *Location* is an integer

- Search behavior should include the following:
  - Populating multiple search criteria should treat each search criteria as a required parameter (AND)
  - Matching substrings with *Keyword* (resource name, model or capabilities
  - Exact match with *ESF* (primary and additional ESF)
  - Less than or equal to *Location*
    - Haversine formula used to calculate distance between two points defined by latitude and longitude coordinates as follows:
      - $\Delta lat = lat2 - lat1$ $\Delta lon = lon2 - lon1$
      - $a = \sin2\ (\Delta lat\ /\ 2) + \cos(lat1) * \cos(lat2) * \sin2\ (\ \Delta lon\ /\ 2)$
      - $c = 2 * \text{atan2}\ (\sqrt{a}, \sqrt{(1 - a)})$
      - $d = R * c$
  - Equal to *Incident*
- If the *Incident* field is not populated run the **Display Search Results** subtask and only include the following columns:
  - *ID*
  - *Name*
  - Owner
  - *Cost*
  - *Status*

Base

```
SELECT A.resource_id, A.name, A.owner, A.cost, A.availability_status, B.expected_return_date FROM
        resources A LEFT JOIN (
                SELECT resource_id, expected_return_date FROM resource_requests WHERE
                        request_status = 'Deployed') B ON
        A.resource_id = B.resource_id;
```

If Keyword Is Populated Add Clause:

```
WHERE resource_id IN (
        SELECT A.resource_id FROM resources A LEFT JOIN
                resource_capabilities B ON
                        A.resource_id = B.resource_id
                WHERE A.name LIKE '%$keyword%' OR A.model LIKE '%$keyword%' OR B.capability LIKE
                        '%$keyword%');
```

If ESF Is Populated Add Clause

```
WHERE/AND resource_id IN (
        SELECT resource_id FROM resources
                WHERE primary_esf_id = '$esf_id' UNION
                        SELECT resource_id FROM resource_secondary_esfs WHERE esf_id = '$esf_id');
```

- **Note**: The where/and in the ESF clause represents the potential of not having the keyword populated (where) or having the keyword populated (and)

- **Note**: The columns and input values will all be converted to uppercase to provide a more robust search feature for phase 3

- Else:
    - Run the **Display Search Results** subtask and include additional columns
        - *Distance*
        - *Action*

Base

SELECT FINAL.resource_id, name, owner, cost, availability_status, RETURN_DATE.expected_return_date, distance FROM (SELECT A.resource_id, A.name, A.owner, A.cost, A.availability_status, B.distance FROM resources A INNER JOIN (SELECT resource_id, (2 * ATAN2(SQRT(a), SQRT(1-a))) * 6373 AS distance FROM (SELECT POWER(SIN(dlat / 2),2) + COS(lat1) * COS(lat2) * POWER(sin(dlon / 2), 2) AS a, dlat, dlon, lat1, lat2, lon1, lon2, resource_id FROM (select lat2 - lat1 AS dlat, lon2 - lon1 AS dlon, lat1, lat2, lon1, lon2, resource_id FROM (SELECT RADIANS(A.latitude) AS lat1, RADIANS(B.latitude) AS lat2, RADIANS(A.longitude) AS lon1, RADIANS(B.longitude) AS lon2, B.resource_id AS resource_id FROM incidents A, resources B WHERE A.incident_id = '$incident_id' AND A.abbreviation = '$abbreviation') X) Y) Z) B ON A.resource_id = B.resource_id WHERE B.distance < '$location') FINAL LEFT JOIN (select resource_id, expected_return_date FROM resource_requests WHERE request_status = 'Deployed') RETURN_DATE ON FINAL.resource_id = RETURN_DATE.resource_id ORDER BY distance ASC;

- **Note**: This will be cleaned up and turned into a function to make cleaner during phase 3

When ESF, Keyword and Location Populated

Select CORE.resource_id, CORE.name, CORE.owner, CORE.cost, CORE.availability_status, RETURN_DATE.expected_return_date, DISTANCE.distance FROM (SELECT resource_id, name, owner, cost, availability_status FROM resources WHERE resource_id IN (SELECT A.resource_id FROM resources A LEFT JOIN resource_capabilities B ON A.resource_id = B.resource_id WHERE A.name LIKE '%keyword%' OR A.model LIKE '%keyword%' OR B.capability LIKE '%keyword%') AND resource_id IN (SELECT resource_id FROM resources WHERE primary_esf_id = '$esf' UNION SELECT resource_id FROM resource_secondary_esfs WHERE esf_id = '$esf')) CORE INNER JOIN (SELECT resource_id, name, owner, cost, availability_status, distance FROM (SELECT A.resource_id, A.name, A.owner, A.cost, A.availability_status, B.distance FROM resources A INNER JOIN (SELECT resource_id, (2 * ATAN2(SQRT(a), SQRT(1-a))) * 6373 AS distance FROM (SELECT POWER(SIN(dlat / 2),2) + COS(lat1) * COS(lat2) * POWER(sin(dlon / 2), 2) AS a, dlat, dlon, lat1, lat2, lon1, lon2, resource_id FROM (select lat2 - lat1 AS dlat, lon2 - lon1 AS dlon, lat1, lat2, lon1, lon2, resource_id FROM (SELECT RADIANS(A.latitude) AS lat1, RADIANS(B.latitude) AS lat2, RADIANS(A.longitude) AS lon1, RADIANS(B.longitude) AS lon2, B.resource_id AS resource_id FROM incidents A, resources B WHERE A.incident_id = '$incident_id' AND A.abbreviation = '$abbreviation') X) Y) Z) B ON A.resource_id = B.resource_id WHERE B.distance < '$location') Z) DISTANCE on CORE.resource_id = DISTANCE.resource_id LEFT JOIN (select resource_id, expected_return_date FROM resource_requests WHERE request_status = 'Deployed') RETURN_DATE ON CORE.resource_id = RETURN_DATE.resource_id ORDER BY distance ASC;

- **Note**: You can add/remove clauses based on what is populated
- **Note**: The columns and input values will all be converted to uppercase to provide a more robust search feature for phase 3

Pending Further Requirements Clarification

Should the user be able to see resources for incidents that have already had the resources deployed there with the following 3 potential behaviors:

- Display resources & request button, however, display an error message after checking that the resource has been used
- Display resources & no request button (potentially confusing to the user)
- Don't display resources

- **Note**: Option #1 selected for this implementation

- If owner = *'$currentuser'* and availability_status = 'Available':
  - Display **Deploy** button in the Action column
  - If selected:
    - Run the **Search Resources Deploy** subtask

```
UPDATE resources SET availability_status = 'In Use' WHERE resource_id = '$resource_id';

INSERT INTO resource_requests (resource_id, abbreviation, incident_id, request_start_date,
expected_return_date, request_accepted_deploy_date, request_status) values ('$resource_id', '$abbreviation',
'$incident_id', NOW(), '$expected_return_date', NOW(), 'Deployed');
```

    - Remove button after selection
- Else:
  - Display **Request** button
  - If selected:
    - Run **Check Previous Request Resource** subtask to create a request for resource

```
SELECT COUNT(*) FROM resource_requests WHERE
        request_status IN ('Pending', 'Deployed', 'Completed') AND abbreviation = '$abbreviation' AND
                incident_id = '$incident_id' AND resource_id = '$resource_id');
```

- **Note**: The requirements did not mention omitting the data from the user's view and only stated that the system should not allow them to request it again

    - If the count is greater than 0:
      - Don't allow user to request the resource
      - Display error message to user
    - Else:
      - Run **Request Resource** subtask to create a request for resource

```
INSERT INTO resource_requests(resource_id, abbreviation, incident_id, requested_start_date,
expected_return_date) VALUES ('$resource_id', '$abbreviation', '$incident_id', '$requested_start_date',
'$expected_return_date');
```

- **Note:** The request_id will be set by auto increment default
- **Note**: The deploy_date field will be set to NULL by default
- **Note**: The request_status field will be set to 'Pending' by default

    - Remove button after selection

- Click ***Cancel*** button

- Exits out of **Search for Resources** form and goes back to **Main Menu**

View Resource Status

Abstract Code

- User clicks on ***Resource Status*** button from **Main Menu**
- Show ***Cancel*** button
- Display a grid labeled Resources In Use associated to resources owned by the current User with the following data elements:
    - *ID*
    - *Resource Name*
    - *Incident*
    - *Owner*
    - *Start Date*
    - *Return By*
    - *Action*
    - Run **View Resource Status In Use** subtask
        - Lookup information to populate the *table* field
        - Let '$username' be current user's username

```
SELECT A.request_id, A.resource_id, B.description, C.owner, A.requested_start_date, A.expected_return_date
        FROM resource_requests A
        INNER JOIN incidents B
                ON A.abbreviation = B.abbreviation and A.incident_id = B.incident_id
        INNER JOIN resources C
                ON A.resource_id = C.resource_id
        WHERE C.availability_status = 'In Use'
                AND B.owner = '$username';
```

- Show ***Return*** button in *Action* column
    - If selected:
        - Run **Return Resources In Use** subtask
        - Let *'$request_id'* represent the ID from the selected request
        - Let *'$resource_id'* represent the ID from the selected resource

```
UPDATE resource_requests
        SET request_status = 'Completed'
                WHERE request_id = '$request_id';

UPDATE resources
        SET availability_status = 'Available' WHERE
                resource_id = '$resource_id';
```

- Remove item from grid

- Display a grid labeled Resources Requested By Me associated to resources that have been requested by the current User with the following data elements:
    - *ID*

        - *Resource Name*
        - *Incident*
        - *Owner*
        - *Return By*
        - *Action*
        - Run **View Resource Status My Requests** subtask
            - Lookup information to populate the table
            - Let *'$username'* be current user's username

```
SELECT A.request_id, A.resource_id, B.description, C.owner, A.expected_return_date
        FROM resource_requests A
        INNER JOIN incidents B
                ON A.abbreviation = B.abbreviation and A.incident_id = B.incident_id
        INNER JOIN resources C
                ON A.resource_id = C.resource_id
        WHERE A.request_status = 'Pending'
                AND B.owner = '$username';
```

            - Show ***Cancel*** button in *Action* column
                - If selected:
                    - Run **Cancel My Requests** subtask
                    - Let '$resource_id' represent the ID from the selected resource

```
UPDATE resource_requests
        SET request_status = 'Cancelled'
                WHERE request_id = '$request_id';
```

                - Remove item from grid
    - Display a grid labeled Resource Requests Received By Me associated to resources that have been requested by other Users owned by the current User with the following data elements:
        - *ID*
        - *Resource Name*
        - *Incident*
        - *Owner*
        - *Return By*
        - *Action*
        - Run **View Resource Status Requests To Me** subtask
            - Lookup information to populate the table
            - Let *'$username'* be current user's username

```
SELECT A.request_id, A.resource_id, B.description, C.owner, A.expected_return_date, B.abbreviation,
B.incident_id, C.availability_status
        FROM resource_requests A
        INNER JOIN incidents B
                ON A.abbreviation = B.abbreviation and A.incident_id = B.incident_id
        INNER JOIN resources C
                ON A.resource_id = C.resource_id
        WHERE A.request_status = 'Pending'
                AND C.owner = '$username';
```

            - Let '$request_id' represent the ID from the selected request
            - Let '$resource_id' represent the ID from the selected resource
            - Let '$abbreviation' represent the abbreviation from the requesting incident
            - Let '$incident_id' represent the incident number from the requesting incident
            - Let '$availability_status' represent the availability status of the resource
            - Show ***Reject*** button in *Action* column
                - If ***Reject*** button selected:
                    - Run **Reject Request** subtask

```
UPDATE resource_requests
        SET request_status = 'Rejected'
                WHERE  request_id = '$request_id';
```

                    - Remove item from grid
            - If '$availability_status' = 'Available'
                - Show ***Display*** button in *Action* column
                    - If ***Display*** button selected:
                        - Run ***Resource Status*** **Deploy Resource** subtask

```
UPDATE resources SET availability_status = 'In Use' WHERE resource_id = '$resource_id';

UPDATE resource_requests SET request_accepted_deploy_date = NOW(), request_status = 'Deployed' WHERE
request_id = '$request_id';
```

                    - Remove item from grid

- Upon:
    - Click ***Cancel*** button
        - Exit out of **Resource Status** form and go back to **Main Menu**

Resource Report

Abstract Code

- User clicks on ***Resource Report*** button from **Main Menu**
- Show ***Cancel*** button
- Display a grid labeled Resource Report By Primary Emergency Support Function with the following data elements :
    - *ESF #*
    - *Primary Emergency Support Function*
    - *Total Resources*
    - *Resources In Use*
- Run **Generate Resource Report** subtask
    - Lookup information to populate the table
        - Only consider the primary ESF for each resource and ignore the additional ESF fields
        - For *Total Resources*, lookup total count of resources owned by current user for each Primary ESF
        - For *Resources In Use*, lookup total count of resources owned by current user “In Use” for each Primary ESF
        - All ESFs should be shown, even if the user owns no resources for that ESF (display 0 as total count)
    - Return total aggregation row on bottom that sums the column values for:
        - Total Resources
        - Resources In Use

```
SELECT A.esf_id, A.description, B.total_count, C.used_count FROM
        esfs A LEFT JOIN (
                SELECT primary_esf_id, COUNT(*) AS total_count FROM resources
                        WHERE owner = '$username' GROUP BY primary_esf_id) B
        ON A.esf_id = B.primary_esf_id LEFT JOIN (
                SELECT primary_esf_id, COUNT(*) AS used_count FROM resources
                        WHERE owner = '$username' AND availability_status = 'In Use'
                        GROUP BY primary_esf_id) C
        ON A.esf_id = C.primary_esf_id order by A.esf_id ASC;

SELECT COUNT(*) AS total_count FROM resources WHERE owner = '$username';

SELECT COUNT(*) AS used_count FROM resources WHERE owner = '$username' AND
        availability_status = 'In Use';
```

- Upon:
    - Click ***Cancel*** button
        - Exit out of **Resource Report** form and go back to **Main Menu**

Appendix (SQL)

Check Password	SELECT password FROM users WHERE username = '*$username*';
Main Menu	SELECT A.name, B.top_line, B.bottom_line FROM users A INNER JOIN (SELECT username AS username, hq_location AS top_line, num_employees AS bottom_line FROM companies UNION SELECT username AS username, category AS top_line, NULL FROM municipalities UNION SELECT username AS username, job_title AS top_line, hire_date AS bottom_line FROM individuals UNION SELECT username AS username, agency_name_local_office, NULL FROM government_agencies) B ON A.username = B.username WHERE A.username = '$username';
ESF Dropdown	SELECT esf_id, description FROM esfs;
Cost Dropdown	SELECT cost_per FROM cost_pers;

Add Resource	INSERT INTO resources (owner, name,latitude, longitude, model, cost, cost_per, maximum_distance, primary_esf_id) VALUES ('$username', '$resource_name', '$latitude', '$longitude', '$model', '$cost', '$cost_per', '$max_distance', '$primary_esf_id');
	
	'$resource_id' = SELECT MAX(resource_id) FROM resources;
	
	INSERT INTO resource_secondary_esfs (resource_id, esf_id) VALUES ('$resource_id', '$secondary_esf_id');
	
	INSERT INTO resource_capabilities (resource_id, capability) VALUES ('$resource_id', '$resource_capability');

Declaration Dropdown	SELECT abbreviation, description FROM incident_types;
Add Incident	'$incident_id' = SELECT CASE WHEN MAX(incident_id) IS NULL THEN 1 ELSE MAX(incident_id) + 1 END FROM incidents WHERE abbreviation = '$abbreviation'; INSERT INTO incidents (abbreviation, incident_id, owner, incident_date, description, latitude, longitude) VALUES ('$abbreviation', '$incident_id', $username', '$incident_date', '$description', '$latitude', '$longitude');
Incidents Dropdown	SELECT abbreviation, incident_id, description FROM incidents;
Display Resources No Criteria	SELECT A.resource_id, A.name, A.owner, A.cost, A.availability_status, B.expected_return_date FROM resources A LEFT JOIN (SELECT resource_id, expected_return_date FROM resource_requests WHERE request_status = 'Deployed') B ON A.resource_id = B.resource_id;

Display Resources No Location Criteria	SELECT A.resource_id, A.name, A.owner, A.cost, A.availability_status, B.expected_return_date FROM resources A LEFT JOIN (SELECT resource_id, expected_return_date FROM resource_requests WHERE request_status = 'Deployed') B ON A.resource_id = B.resource_id
	
	--If Keyword Is Populated Add Clause:
	
	WHERE resource_id IN (SELECT A.resource_id FROM resources A LEFT JOIN resource_capabilities B ON A.resource_id = B.resource_id WHERE A.name LIKE '%$keyword%' OR A.model LIKE '%$keyword%' OR B.capability LIKE

	'%$keyword%');
	
	 --If ESF Is Populated Add Clause
	
	 WHERE/AND resource_id IN (SELECT resource_id FROM resources WHERE primary_esf_id = '$esf_id' UNION SELECT resource_id FROM resource_secondary_esfs WHERE esf_id = '$esf_id');

Display Resources Location Criteria	SELECT FINAL.resource_id, name, owner, cost, availability_status, RETURN_DATE.expected_return_date, distance FROM (SELECT A.resource_id, A.name, A.owner, A.cost, A.availability_status, B.distance FROM resources A INNER JOIN (SELECT resource_id, (2 * ATAN2(SQRT(a), SQRT(1-a))) * 6373 AS distance FROM (SELECT POWER(SIN(dlat / 2),2) + COS(lat1) * COS(lat2) * POWER(sin(dlon / 2), 2) AS a, dlat, dlon, lat1, lat2, lon1, lon2, resource_id FROM (select lat2 - lat1 AS dlat, lon2 - lon1 AS dlon, lat1, lat2, lon1, lon2, resource_id FROM (SELECT RADIANS(A.latitude) AS lat1, RADIANS(B.latitude) AS lat2, RADIANS(A.longitude) AS lon1, RADIANS(B.longitude) AS lon2, B.resource_id AS resource_id FROM incidents A, resources B WHERE A.incident_id = '$incident_id' AND A.abbreviation = '$abbreviation') X) Y) Z) B ON A.resource_id = B.resource_id WHERE B.distance < '$location') FINAL LEFT JOIN (select resource_id, expected_return_date FROM resource_requests WHERE request_status = 'Deployed') RETURN_DATE ON FINAL.resource_id = RETURN_DATE.resource_id ORDER BY distance ASC;
Display Resources All Criteria	Select CORE.resource_id, CORE.name, CORE.owner, CORE.cost, CORE.availability_status, RETURN_DATE.expected_return_date, DISTANCE.distance FROM (SELECT resource_id, name, owner, cost, availability_status FROM resources WHERE resource_id IN (SELECT A.resource_id FROM resources A LEFT JOIN resource_capabilities B ON A.resource_id = B.resource_id WHERE A.name LIKE '%keyword%' OR A.model LIKE '%keyword%' OR B.capability LIKE '%keyword%') AND resource_id IN (SELECT resource_id FROM resources WHERE primary_esf_id = '$esf' UNION SELECT resource_id FROM resource_secondary_esfs WHERE esf_id = '$esf')) CORE INNER JOIN (SELECT resource_id, name, owner, cost, availability_status, distance FROM (SELECT A.resource_id, A.name, A.owner, A.cost, A.availability_status, B.distance FROM resources A INNER JOIN (SELECT resource_id, (2 * ATAN2(SQRT(a), SQRT(1-a))) * 6373 AS distance FROM (SELECT POWER(SIN(dlat / 2),2) + COS(lat1) * COS(lat2) * POWER(sin(dlon / 2), 2) AS a, dlat, dlon, lat1, lat2, lon1, lon2, resource_id FROM (select lat2 - lat1 AS dlat, lon2 - lon1 AS dlon, lat1, lat2, lon1, lon2, resource_id FROM (SELECT RADIANS(A.latitude) AS lat1, RADIANS(B.latitude) AS lat2, RADIANS(A.longitude) AS lon1, RADIANS(B.longitude) AS lon2, B.resource_id AS resource_id FROM incidents A, resources B WHERE A.incident_id = '$incident_id' AND A.abbreviation = '$abbreviation') X) Y) Z) B ON A.resource_id = B.resource_id WHERE B.distance < '$location') Z) DISTANCE on CORE.resource_id = DISTANCE.resource_id LEFT JOIN (select resource_id, expected_return_date FROM resource_requests WHERE request_status = 'Deployed') RETURN_DATE ON CORE.resource_id = RETURN_DATE.resource_id ORDER BY distance ASC;

Search Resources Deploy	UPDATE resources SET availability_status = 'In Use' WHERE resource_id = '$resource_id';
	
	 INSERT INTO resource_requests (resource_id, abbreviation, incident_id, request_start_date, expected_return_date, request_accepted_deploy_date, request_status) values ('$resource_id', '$abbreviation', '$incident_id', NOW(), '$expected_return_date', NOW(), 'Deployed');

Check Previous Resource Request	SELECT COUNT(*) FROM resource_requests WHERE request_status IN ('Pending', 'Deployed', 'Completed') AND abbreviation = '$abbreviation' AND incident_id = '$incident_id' AND resource_id = '$resource_id');
Request Resource	INSERT INTO resource_requests(resource_id, abbreviation, incident_id,

	requested_start_date, expected_return_date) VALUES ('$resource_id', '$abbreviation', '$incident_id', '$requested_start_date', '$expected_return_date')
View Resource Status In Use	SELECT A.request_id, A.resource_id, B.description, C.owner, A.requested_start_date, A.expected_return_date FROM resource_requests A INNER JOIN incidents B ON A.abbreviation = B.abbreviation AND A.incident_id = B.incident_id INNER JOIN resources C ON A.resource_id = C.resource_id WHERE C.availability_status = 'In Use' AND B.owner = '$user';

Return Resources In Use	UPDATE resource_requests SET request_status = 'Completed' WHERE request_id = '$request_id';
	
	UPDATE resources SET availability_status = 'Available' WHERE resource_id = '$resource_id';

View Resource Status My Requests	SELECT A.request_id, A.resource_id, B.description, C.owner, A.expected_return_date FROM resource_requests A INNER JOIN incidents B ON A.abbreviation = B.abbreviation and A.incident_id = B.incident_id INNER JOIN resources C ON A.resource_id = C.resource_id WHERE A.request_status = 'Pending' AND B.owner = '$username';
Cancel My Requests	UPDATE resource_requests SET request_status = 'Cancelled' WHERE request_id = '$request_id';
View Resource Status Requests To Me	SELECT A.request_id, A.resource_id, B.description, C.owner, A.expected_return_date, B.abbreviation, B.incident_id, C.availability_status FROM resource_requests A INNER JOIN incidents B ON A.abbreviation = B.abbreviation and A.incident_id = B.incident_id INNER JOIN resources C ON A.resource_id = C.resource_id WHERE A.request_status = 'Pending' AND C.owner = '$username';
Reject Request	UPDATE resource_requests SET request_status = 'Rejected' WHERE request_id = = '$request_id';

Generate Resource Report	SELECT A.esf_id, A.description, B.total_count, C.used_count FROM esfs A LEFT JOIN (SELECT primary_esf_id, COUNT(*) AS total_count FROM resources WHERE owner = '$username' GROUP BY primary_esf_id) B ON A.esf_id = B.primary_esf_id LEFT JOIN (SELECT primary_esf_id, COUNT(*) AS used_count FROM resources WHERE owner = '$username' AND availability_status = 'In Use' GROUP BY primary_esf_id) C ON A.esf_id = C.primary_esf_id order by A.esf_id ASC;
	
	SELECT COUNT(*) AS total_count FROM resources WHERE owner = '$username';
	
	SELECT COUNT(*) AS used_count FROM resources WHERE owner = '$username' AND availability_status = 'In Use';