

TEAM 49: REPORT

(CS6400 PROJECT)



OCTOBER 16, 2016

TEAM 049

BY:

Seyedmohammad Razavi (srazavi32)

Maziar Motahari (mmotahari3)

Andrew Keith (akeith30)

Zhenghang Gu (zgu64)

Contents

[1- Login Task 2](#_Toc464360133)

[2- MAIN MENU SELECTION TASK 3](#_Toc464360135)

[3- Adding Resources to ERMS 4](#_Toc464360137)

[4- Adding Emergency Incidents to ERMS 6](#_Toc464360139)

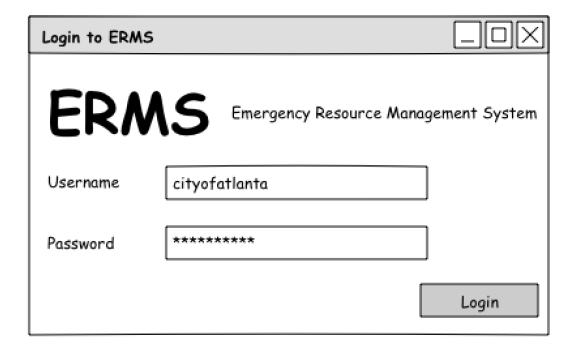
[5- Searching for Resources 7](#_Toc464360141)

[6- Resource Status 10](#_Toc464360143)

[7- Actions 12](#_Toc464360145)

[8- Report Generator 13](#_Toc464360146)

# Login Task



## ABSTRACT CODE AND CONSTRAINTS:

* User enters username(@username1) and password(@pass1)
* If data validation is successful for both email and password input fields, then:
  + When Login button is clicked:

select \* from users where USERNAME=@username1 and PASSWORD=@pass1;

* If username and password did not match any data in the database, display the error message form (“Username and Password Combination Does Not Exist!”) and return to the login menu.
* If username and password found in the database, store user information as session variable **@currentuser** and show ERMS menu form

# MAIN MENU SELECTION TASK

|  |
| --- |
|  |
| MAIN MENU FORM |

## ABSTRACT CODE AND CONSTRAINTS:

Find the current user using the username

Find name and display it on the specified location on the form. Assume @currentuser is passed to the form.

select NAME from users where USERNAME=**@currentuser** and PASSWORD=@pass1;

Display the following information depend on the type of user.

The population size, if the user is a municipality. If following query result is not empty show population.

select population from municipality\_users where USERNAME=**@currentuser**;

The jurisdiction, if the user is a government agency.

select jurisdiction from govagency\_users where USERNAME=**@currentuser**;

The location of the headquarters, if the user is a company.

select LOC\_OF\_HQ from company\_users where USERNAME=**@currentuser**;

If “Add Resource” link is clicked, close current form and open “Add Resource” form

If “Add Emergency Incident” link is clicked, close current form and open “Add Incident” form

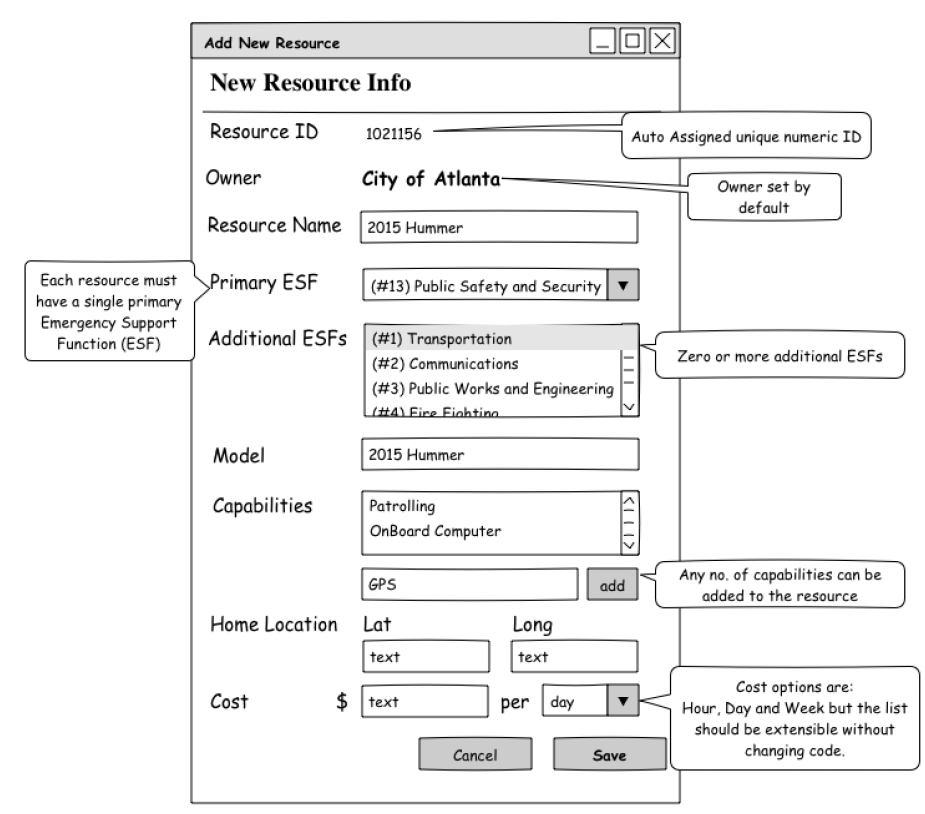
If “Search Resources” link is clicked, close current form and open “Search Resources” form

If “Resource Status” link is clicked, close current form and open “Resource Status” form

If “Resource Report” link is clicked, close current form and open “Resource Report” form

If “Exit” link is clicked, close current form and logout

# Adding Resources to ERMS



New resource form

## ABSTRACT CODE AND CONSTRAINTS:

* Fill the primary ESF dropdown. UI should allow for unique selection.
* Fill the additional ESF dropdown. UI should allow for multiple selection.
  + For both above steps the following SQL query is used:

select ESF\_ID, DESCRIPTION from ESF;

* Selected Primary ESF number is stored an integer in @P\_ESF1
* Selected Additional ESF numbers are stored in an integer list @A\_ESFs1
* The primary ESF of a resource cannot appear as an additional ESF
* Fill the Cost\_Per dropdown using below Query.

select COSTPER from COST\_PER;

* Find name and display it on the specified location on the form. Assume @currentuser is passed to the form.

select NAME from users where USERNAME=**@currentuser** and PASSWORD=@pass1;

* If data validation is successful for all input fields, then:
* When Save button is clicked.
* Read and store all input information in variables below:

(@Name1, @P\_ESF1, @M\_NAME1, @AMOUNT1, @COST\_PER1, @LATITUDE1, @LONGITUDE1, @CAPABILITIES1, @A\_ESF1)

* Resource ID will be automatically generated.
* Default Date\_AV value is the current date.
* Default status is ‘Available’.

insert into resource (USERNAME,NAME,P\_ESF,M\_NAME,AMOUNT,COST\_PER,STATUS,DATE\_AV,LATITUDE,LONGITUDE)

values(@currentuser, @Name1, @P\_ESF1, @M\_NAME1, @AMOUNT1, @COST\_PER1, 'Available',CURDATE(),@LATITUDE1, @LONGITUDE1);

* After numerical ID is assigned show it on the form and stored as @Res\_ID1.
* For each capability in @CAPABILTY1 list:

insert into resource\_capabilty(RES\_ID,CAPABILITY)

values(@Res\_ID1,@Capability1);

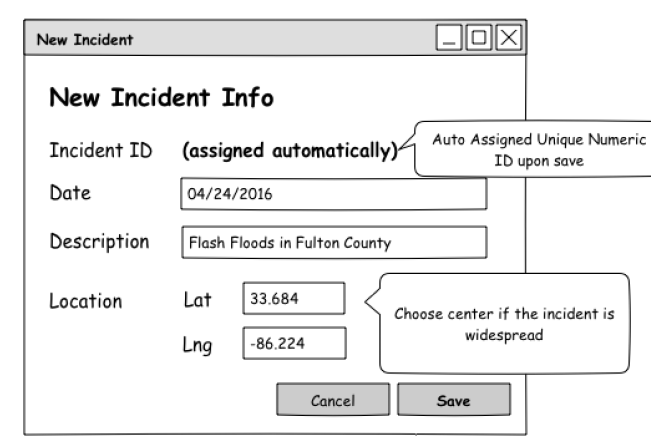
* For each capability in @CAPABILTY1 list:

insert into resource\_capabilty(RES\_ID,CAPABILITY)

values(@Res\_ID1,@Capability1);

* Owner is set automatically to the logged in user
* Primary ESF should be selected from one of the 15 predefined ESFs
* Each resource has only one Primary ESF
* Resources can have zero, one, or multiple additional ESFs
* The primary ESF of a resource cannot appear as an additional ESF
* Model field is optional
* Capabilities: zero or more
* Home Location: Latitude and Longitude
* The cost is stored in dollars per hour/day/week/each. This list should be extensible without requiring code changes
* Latitude and Longitude: Sign decimal degrees
* On clicking add capacities, add the new capability to the capabilities list and refresh the capability list on the display.
* On clicking cancel button
  + Close the add resources form
  + Show the main menu
* On save button
  + Check if all information is in correct data format.
  + Save record to the resource table
  + Close the add resource form
  + Show the main menu

# Adding Emergency Incidents to ERMS



ADD NEW INCIDENT FORM

## ABSTRACT CODE AND CONSTRAINTS:

* Each incident is automatically assigned a **unique numerical** **ID** on save
* The **owner** of the incident is automatically set to the current user
* All fields are required and should be validated before saving the incident to the database
* All incidents are private to the current user
* On clicking cancel button
  + Close the add incident form
  + Show the main menu
* On save button
  + Check if all information is in correct data format.
  + Read and store all input information in variables below:

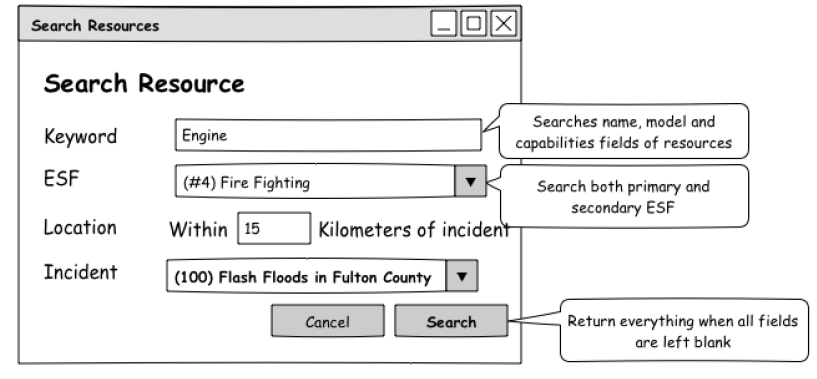
(@DESCRIPTION1,@LATITUDE1,@LONGITUDE1)

insert into incidents (USERNAME,DESCRIPTION,LATITUDE,LONGITUDE)

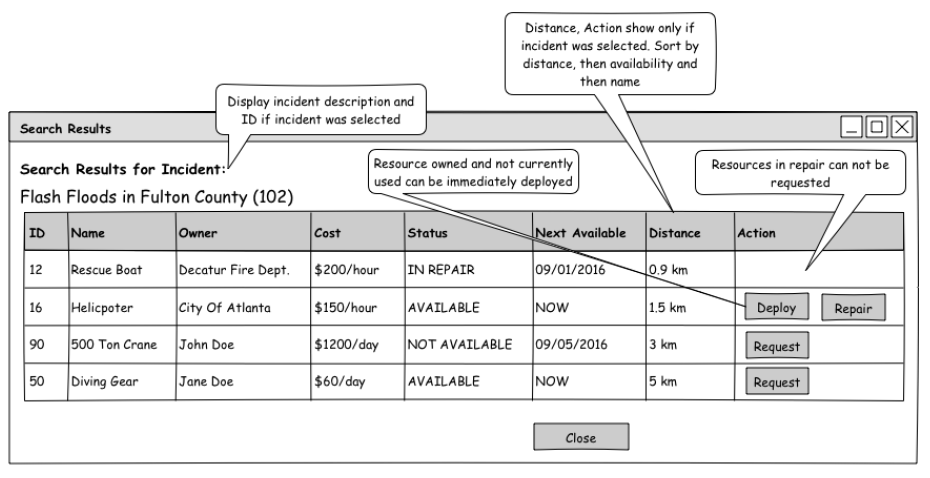
values(@currentuser, @ DESCRIPTION1, @LATITUDE1, @LONGITUDE1);

* + Close the add incident form
  + Show the main menu

# Searching for Resources



Search for Resources



Search Results

## ABSTRACT CODE AND CONSTRAINTS:

* User may search by keyword (which includes the resource name, model, and capabilities)
* User may search by Emergency Support Function (which includes both primary and additional ESFs),
* User may search by proximity to an emergency incident.
* It is possible that the user will leave all search fields blank
* Read incidents form incident table and show them in a dropdown selection list

select INC\_ID, DESCRIPTION from INCIDENTS where username=@currentuser;

* If cancel button is pushed, close window and display main menu
* If the search button is pushed do the following steps
* Show ESF list in the dropdown selection list

select ESF\_ID, DESCRIPTION from ESF;

* When Search button is clicked.
* Read and store all input information in variables below:

(@Keyword, @ESF1, @Distance1, @INC\_ID1)

* If an incident is selected read location of the incident to calculate distance of the incident to resources.

select LATITUDE, LONGITUDE from INCIDENTS where INC\_ID=@INC\_ID1;

This values are stored as @IncLat, @IncLong.

* If no field is selected

select a.ID,a.NAME,b.NAME,a.AMOUNT,a.COST\_PER,a.STATUS,a.DATE\_AV from resource a

inner join users b on a.USERNAME=b.USERNAME

* Search tables by KEYWORD

select a.ID,a.NAME,b.NAME,a.AMOUNT,a.COST\_PER,a.STATUS,a.DATE\_AV from resource a

inner join users b on a.USERNAME=b.USERNAME

left outer join resource\_capabilty c on a.ID=c.RES\_ID

where a.name like @keyword

or a.M\_NAME like @keyword

or c.CAPABILITY like @keyword

* Search resource table by ESF assume the input ESF\_ID is stored as ESFid1

select a.ID,a.NAME,b.NAME,a.AMOUNT,a.COST\_PER,a.STATUS,a.DATE\_AV from resource a

inner join users b on a.USERNAME=b.USERNAME

left outer join additional\_esf c on a.ID=c.RES\_ID

where a.P\_ESF = @ESFid1 or c.ESF\_ID=@ESFid1;

* IF incident is selected search resource table by LOCATION will be activated
* Assume function Distance (lat1, long1, lat2, long2) can calculate the distance between two locations 1 and 2 based on their latitude and longitude using *haversine* formul

select a.ID,a.NAME,b.NAME,a.AMOUNT,a.COST\_PER,a.STATUS,a.DATE\_AV, Distance(a.LATITUDE,a.LONGITUDE, @IncLat, @IncLong) from resource a

inner join users b on a.USERNAME=b.USERNAME

where Distance< @Distance1

order by Distance asc

* Depend on how many information fields are provided by the user the intersection of above queries will be displayed as the search results
* Display results in show results from using queries provided above
* Display incident description if an incident was selected

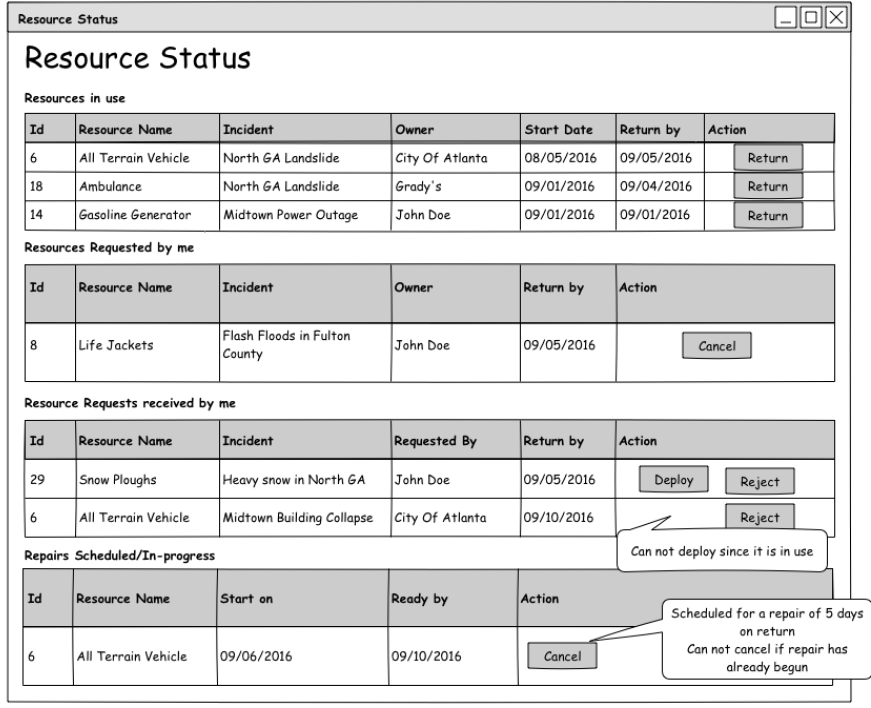
select INC\_ID, DESCRIPTION from INCIDENTS where INC\_ID=@INC\_ID1;

* Display search results for resource ID, Name, Owner, Cost, Status, Next Available Date
* Display distances only if an incident was selected
* Depends on the status of resource the action options will change

select status from RESOURCES where RES\_ID=@RES\_ID1;

* + Resources in repair have no action
  + Resources in use or available can be requested
  + Resources owned and available can be deployed or be scheduled for repair

# Resource Status



## ABSTRACT CODE AND CONSTRAINTS:

* Search and list all resources that are currently in use responding to any incidents owned by the current user. For each resource display the **ID**, **name**, **incident responding to**, and **owner.**

select a.RES\_ID as 'ID',b.NAME as 'Resource Name',c.DESCRIPTION as 'Incident',d.NAME as 'Owner', a.APP\_DATE as 'Start Date', a.RET\_DATE as 'Return By'

from requests a

join resource b

on a.RES\_ID=b.ID

join incidents c

on a.INC\_ID=c.INC\_ID

join users d

on b.USERNAME=d.USERNAME

where c.USERNAME=@currentuser and a.status='Approved';

* Search and list resource requests that have been sent by the current user (to another user) but have not yet been responded to

select a.RES\_ID as 'ID',b.NAME as 'Resource Name',c.DESCRIPTION as 'Incident',d.NAME as 'Owner', a.RET\_DATE as 'Return By'

from requests a

join resource b

on a.RES\_ID=b.ID

join incidents c

on a.INC\_ID=c.INC\_ID

join users d

on b.USERNAME=d.USERNAME

where c.USERNAME=@currentuser and a.status='Pending';

* Search and list resource requests received by the current user that are awaiting the user’s response

select a.RES\_ID as 'ID',b.NAME as 'Resource Name',c.DESCRIPTION as 'Incident',d.NAME as 'Requested By', a.RET\_DATE as 'Return By'

from requests a

join resource b

on a.RES\_ID=b.ID

join incidents c

on a.INC\_ID=c.INC\_ID

join users d

on c.USERNAME=d.USERNAME

where b.USERNAME=@currentuser and a.status='Pending';

* Search and list all resources that have a repair schedule request or are under repair

Select ID as 'ID',NULL as 'START\_DATE',DATE\_AV as'Ready by',STATUS from resource where status="In-Repair" and USERNAME=@currentuser

union

select RES\_ID as 'ID',START\_DATE,READY\_DATE as'Ready by',"REPAIR\_SCHEDULED" as 'STATUS' from rep\_requests;

# Actions

* A given resource cannot be used to respond to multiple incidences at the same time
* Repairs can be scheduled by the owner.
* A resource can be requested directly from the search results screen only when the user selected an incident on the search criteria form
* Resource requests are always made within the context of an incident
* Both available and in use resources may be requested
* Multiple requests may queue up for the same resource
* A resource may only be deployed to respond to one incident at a time
* Resources currently being repaired cannot be requested
* When Resources are requested, an expected return date must be added
* Repair requests should be of the form “for N days after return” for *In Use* resources and “for next N days” for *available* resources
* The *Next Available* date should take into account the requested repair
* If the repair duration has already begun, the repair cannot be cancelled and hence the resource cannot be deployed/requested.
* If the duration is yet to begin, the owner can accept requests for it but will have to explicitly cancel the repair request
* Once a resource has been returned back to available status, the system should prevent the same resource from being requested again for the same incident.
* The system should allow the user to *cancel* any pending resource requests.
* If a resource is currently in use, the deploy button should be disabled
* ***Deploy***the resource: When Button Deploy is pressed. Resource **status** will be set to in use. The request will be removed from the requests Table.

DELETE from requests where RES\_ID=@RES\_ID1 and INC\_ID=@INC\_ID1;

update resource

set STATUS='IN-USE' and DATE\_AV=$DATE\_AV where ID=@RES\_ID1;

* **Return** Resource: If there is a Repair request the status will change to In-Repair and Next available time will be calculated accordingly; otherwise, it changes the status of the resource to available change next available date to current date.

select START\_DATE,RETURN\_DATE from rep\_request where RES\_ID=@RES\_ID1;

If above query returns results, then

update resource

set STATUS='IN-REPAIR' and DATE\_AV=@RETURN\_DATE1 where ID=@RES\_ID1;

Else

update resource

set STATUS='AVAILABLE' and DATE\_AV=$curdate() where ID=@RES\_ID1;

* *Cancel* any pending resource requests: the request will disappear from the request table.

DELETE from requests where RES\_ID=@RES\_ID1 and INC\_ID=@INC\_ID1;

* Reject a request: the pending request will be removed from requests table.

DELETE from requests where RES\_ID=@RES\_ID1 and INC\_ID=@INC\_ID1;

* Repair: Is requested by owner. A Repair request form is displayed and user enters the expected number of days for repair. If the status is “AVAILABLE” it will change to “IN-REPAIR” and the request will not be recorded. Otherwise, it will be recorded in the rep\_request table.
* If the status is “IN-USE”

INSERT INTO rep\_request values (@RES\_ID1,@start\_date1,@return\_date1)

* If the status is “AVAILABLE”

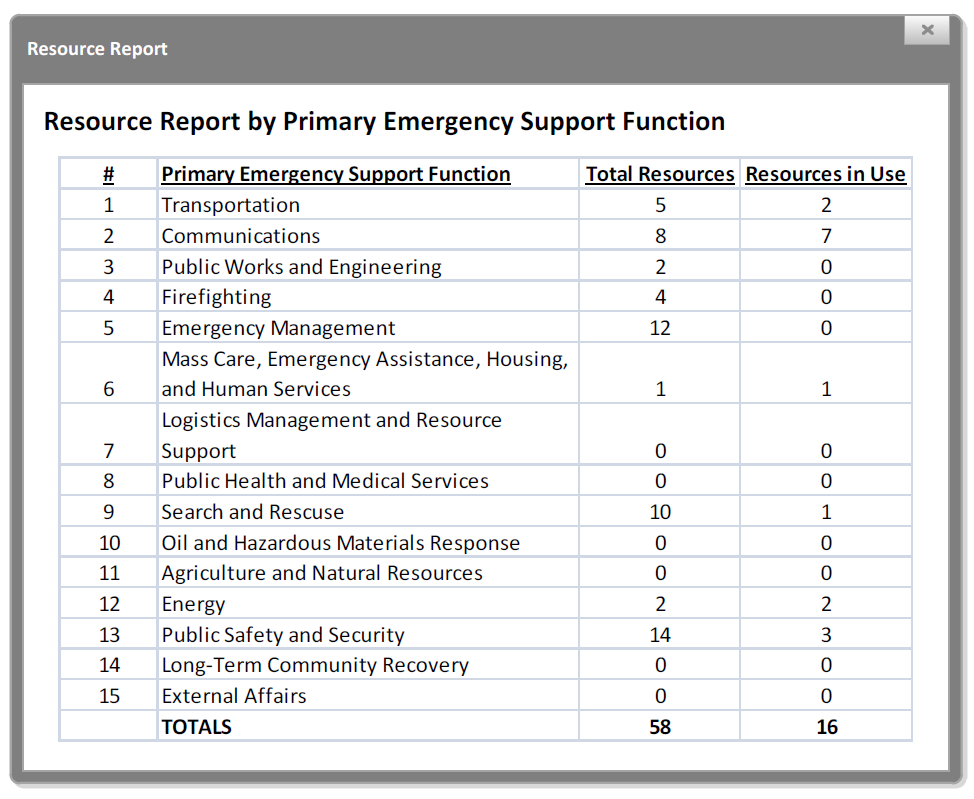
update resource

set STATUS='IN-REPAIR' and DATE\_AV=@RETURN\_DATE1 where ID=@RES\_ID1;

* Request:

INSERT INTO requests values (@RES\_ID1,@ INC\_ID1, @REQ\_DATE1, @RET\_DATE1)

# Report Generator



This report should only consider the primary ESF for each resource and ignore the additional ESFs field

Only resources owned by the current user should be counted for the Total Resources column.

The Resources in Use column counts the total number of resources with the given ESF that are currently in use

All ESFs should be shown, even if the user owns no resources for that ESF.

Select a.ESF\_ID,a.DESCRIPTION, count(b.ID) as 'Total Resources',count(c.ID) as 'Resources in use'

from ESF a

left outer join resource b

on a.ESF\_ID=b.P\_ESF and b.USERNAME=@currentuser

left outer join resource c

on a.ESF\_ID=c.P\_ESF and c.status='In-USE' and c.USERNAME=@currentuser

group by a.ESF\_ID,a.DESCRIPTION;