Table of Contents

Data types

- User

Attribute	Data Type	Nullable
UserName	Varchar(15)	Not Null
DisplayName	Varchar(30)	Not Null
Password	Varchar(30)	Not Null

- UserCIMT

Attribute	Data Type	Nullable
UserName	Varchar(15)	Not Null
PhoneNumber	Varchar(15)	Not Null

- UserRP

Attribute	Data Type	Nullable
UserName	Varchar(15)	Not Null
Address	Varchar(100)	Not Null

- UserSA

Attribute	Data Type	Nullable
UserName	Varchar(15)	Not Null
Email	Varchar(255)	Not Null

- Resource

Attribute	Data Type	Nullable
ResourceId	Int Unsigned	Not Null
UserName	Varchar(15)	Not Null
ResourceName	Varchar(30)	Not Null
PrimaryFunction	SmallInt Unsigned	Not Null
SecondaryFunction	SmallInt Unsigned	Null
Description	Varchar(255)	Null
Distance	Decimal(5,1)	Null
Cost	Decimal(10,2)	Not Null
UnitId	SmallInt Unsigned	Not Null

- ResourceFunctions

Attribute	Data Type	Nullable
FunctionId	SmallInt Unsigned	Not Null
Description	Varchar(255)	Not Null

- ResourceCapability

Attribute	Data Type	Nullable
ResourceId	Int Unsigned	Not Null
Capability	Varchar(20)	Not Null

- Unit

Attribute	Data Type	Nullable
UnitId	SmallInt Unsigned	Not Null
DisplayName	Varchar(10)	Not Null

- Incident

Attribute	Data Type	Nullable
IncidentId	Varchar(10)	Not Null
UserName	Varchar(15)	Not Null
CategoryId	Varchar(5)	Not Null
IncidentDate	Date	Not Null
Description	Varchar(255)	Not Null

- Category

Attribute	Data Type	Nullable
CategoryId	Varchar(5)	Not Null
CategoryType	Varchar(30)	Not Null
IncidentCount	Int Unsigned	Not Null

Business constraints

- A resource is available if it is not currently being used to respond to an incident.
- New resources entered into the system are available by default.
- In no circumstance should the system allow a resource that is currently in use be deployed to respond to another incident.
- A resource may be used for any incident regardless of its primary and secondary function.
- Once an incident is resolved, the status of any resource being used is set to available.
- The system must prevent users from creating multiple incidents for the same incident.

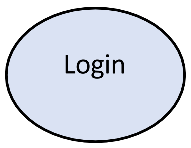
Markup Annotations

- **Bold Underline: Form.**
- ***Bold Italics: Buttons / Input Fields.***
- **Bold*:* Task.**
- *Italics:* Form Input fields / Column names in tabulated form.
- $XYZ: Database field/column named 'XYZ'.

Task decomposition with abstract code:

Login

Task Decomposition:



Abstract Code:

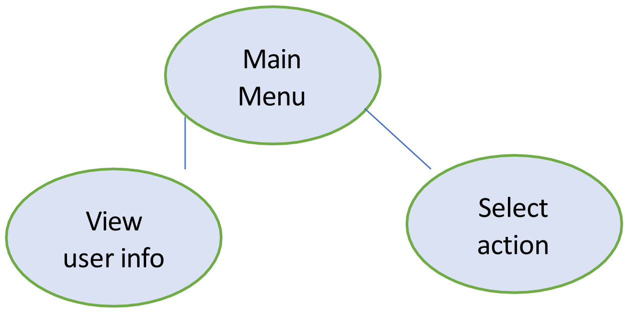
- User enters *Username* ('$Username') and *Password* ('$Password') input fields.
- If data validation is successful for both *Username* and *Password* input fields, then:
- When ***Login*** button is clicked:
    - If *Username* exists but the password does not match, do not proceed to the main menu:
        - Go back to **Login** form, with error message
    - Else, log user in and go to **Main Menu** form.
    - Else, *Username* and *Password* input fields are invalid, display **Login** form, with error message.

SQL Queries:

Select from User table the first record that matches the Username and Password parameters entered

Main Menu

Task Decomposition:



Abstract Code:

- User successfully logged in from the Login

- Run the view user info task by querying the user to display the user's '$Name' and type:
    - If the user is cimt_user, show the user's phone number.
    - If the user is resource_provider, show the user's address.
    - If the user is system administrator, show the user's email address.
- Run the **select action** task to display "Add Resource", "Add Emergency Incident", "Search Resources", "Resource Status", and "Resource Report" links.

Upon clicking:

- ***Add Resource*** button - Jump to **Add Resource** task.
- ***Add Emergency Incident*** button - Jump to **Add Emergency Incident** task
- ***Search Resources*** button - Jump to **Search Resources** task.
- ***Resource Status*** button - Jump to **Resource Status** task.
- ***Resource Report*** button - Jump to **Resource Report** task.
- ***Exit*** button - Invalidate login session and go back to **Login** form.

● Upon clicking "x" cross sign on **Main Menu** form:

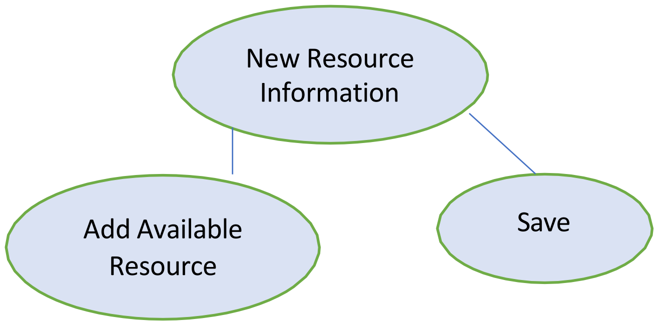
- Exit the current form and take user back to the **Login** form.

SQL Queries:

Select user displayName from User table where it matches username
Select from appropriate User sub-table for phone number/address/email address

New Resource Information

Task Decomposition:



<u>Abstract Code:</u>

- User enters *Resource Name* ('$resource_name'), *Primary Function* ('$primary_function'), Secondary Functions ('$secondary_function'), Description ('$description'), Capabilities ('$capability'), Distance from PCC ('$distance'), Cost ('$cost'), Per Unit ('$unit') input fields.
    - GET ('$username') from database server and display next to *Owner*.
    - When pages is called, GET ('$function') for *Primary* and *Secondary Functions* from database server.
    - When ('$primary_function') number and description is selected, the option is removed from the ('$function') list for secondary function.
    - A blank ('') is added to the ('$function) list for the ('$seondary_function').
- When ***Save*** button is clicked:
    - Check that *Max Distance* and *Cost* are non-negative numbers.
    - Run the Save Resource task, get response from database to display the user's ('$resource_id') in the Resource ID field, and display the "Resource saved".
    - POST ('$resource_id') to database server.
    - GET ('$username') from database server and display next to *Owner*.
- When ***Cancel*** button is clicked:
    - Exit the current form and take the user back to the **Main Menu** form.
- Upon clicking "x" cross sign on Add Available Resource form:
    - discard the current form and take user to another **Add Available Resource** form.

<u>SQL Queries:</u>

//GET primary functions:
Select all rows from ResourceFunction table

//GET units:
SELECT all rows from Unit table

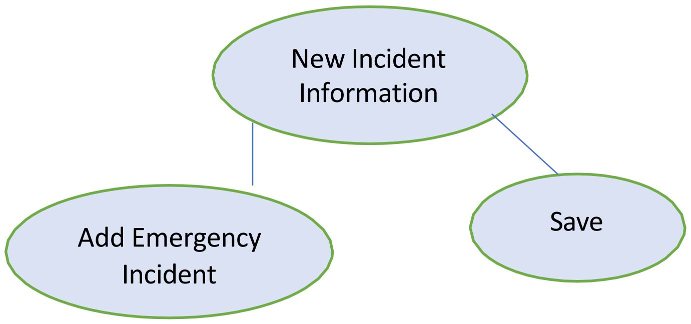
//INSERT resources:

Insert into Resource table all needed values

//INSERT Resource Capabilities
After the Resource ID has been set, insert into ResourceCapability table all user-defined capabilities for new resource

New Incident Information

<u>Task Decomposition:</u>



<u>Abstract Code:</u>

- User enters *Category* (‘$category’), *Date* (‘$date’), Description (‘$description’) input fields.
    - When pages is called, GET (‘$category) for *Category* from database server.
- When ***Save*** button is clicked:
    - Run the **Save Incident** task to display the user’s (‘$incident_id’) in the Incident ID field and display the “Incident saved.”
    - POST (‘$incident_id) to database server.

    - GET (‘$incident_id) from database server and display next to *Incident ID*.
- When ***Cancel*** button is clicked:
    - Exit the current form and take the user back to the **Main Menu** form.
- Upon clicking “x” cross sign on Add Emergency Incident form:
    - discard the current form and take the user to another **Add Emergency Incident** form.

<u>SQL queries:</u>

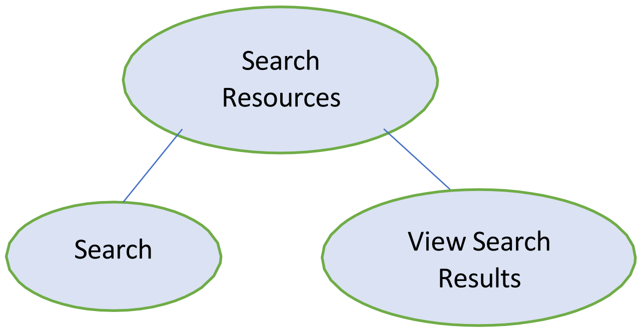
//GET categories:
Select CategoryType from Category table

//GET Incident ID
Select CategoryID and IncidentCount from Category table based on the user-selected category

//INSERT incidents:
Insert into Incident table information passed in by user along with the incident ID generated from Category table information

Search Results

<u>Task Decomposition:</u>



Abstract Code:

- User enters *Keyword* (‘$keyword’), Primary Function (‘$primary_function’), Incident (‘$incident’), Distance (‘$distance’) input fields.
    - When page loads, GET (‘$primary_function’) from database server and display in *Primary Function* dropdown.
    - When page loads, GET (‘$incident_id’) from database server and display in *Incident* dropdown.
    - Confirm (‘$distance’) > 0 and input allows for one decimal point.
- When ***Search*** button is clicked:
    - Run the **Search Resources** task to display the user’s (‘$resource_id’), (‘$resource_name’), (‘$owner’), (‘$cost’), (‘$unit’), (‘$distance’).
    - If Keyword != Blank, return matching resources containing keyword substrings in the ‘$ResourceName’, ‘$Description’, and ‘$Capabilities’ of Resource entity.
    - If Incident != Blank, return matching resources containing Incident.
    - If Distance != Blank, return resources with (‘$distance’) <= Distance.
    - User is taken to *Search Results* page.
    - GET (‘$resource_id’), (‘$resource_name’), (‘$owner’), (‘$cost’), (‘$unit’), (‘$distance’) from database server.
    - SORT by (‘$distance’).
- When ***Cancel*** button is clicked:
    - Exit the current form and take the user back to the **Main Menu** form.
    - Upon clicking “x” cross sign on **Search Resources** form:
    - Discard both Search Resources and Search Results and take user to a new Search Resources form.

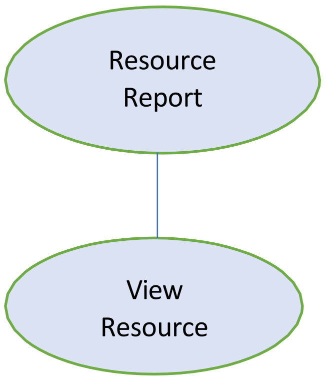
SQL query:

Select all from Resource table and order by Distance ascending

If Keywords, Primary Function, or Distance are given as parameter then add Where clause and add filters to SQL statement as needed

Resource Report

<u>Task Decomposition:</u>



<u>Abstract Code:</u>

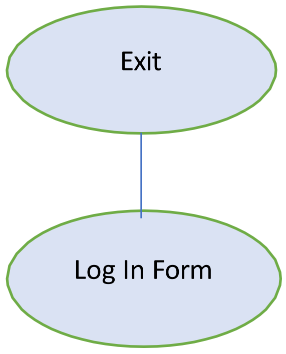
- Run the view resource report task to display ('$primary_function_#'), ('$primary_function'), ('$total_resources')
    - GET Resource information from Resource table where ('$username') === *Username*.
    - Loop through resource results and tally the number of resources per Primary Function
    - Display in table view a list of primary functions and the total number of resources owned by the user that correspond to that primary function
    - Display total number of resources returned from original query to resource table

<u>SQL query:</u>

//GET resources report page:
Select all from Resource table where username is the username of logged in user

Exit

Task Decomposition:



Abstract Code:

- ***Exit*** button resides in the navigation pane.
- When ***Exit*** button is clicked:
    - Logging the user out of the system, taking the user back to the **Login** form.