Phase 1 Report

Table of Contents

Data types

Users

Display Name	Varchar(20)	Not Null
Username	Varchar(12)	Not Null
Password	Varchar(12)	Not Null
Role	Varchar(20)	Not Null

CIMT

Username	Varchar(12)	Not Null
Phone Number	Varchar(10)	

Back to Top

Resource Provider

Username	Varchar(12)	Not Null
Address	Varchar(150)	

Admin

Username	Varchar(12)	
Email Address	Varchar(20)	

Incidents

ID	Varchar(20)	Not Null
Description	Varchar(30)	Not Null
Date	Date	Not Null
Cat_ID	Varchar(2)	Not Null
User	Varchar(20)	Not Null

Category

ID	Varchar(5)	Not Null
Category Type	Varchar(50)	Not Null

Back to Top

Resource

ID	Varchar(20)	Not Null
Name	Varchar(20)	Not Null
Primary Function	Varchar(20)	Not Null
Secondary Function	Varchar(20)	
Descriptions	Varchar(30)	
Capabilities	Varchar(20)	
Distance	Double(10,1)	
Owner	Varchar(20)	Not Null
Price	Decimal(7,2)	Not Null
Unit ID	VarChar(20)	Not Null

Unit

ID	Varchar(20)	Not Null
name	Decimal(2,2)	Not Null

Function

Number	Int	Not Null
Name	VARCHAR(20)	Not Null

Back to Top

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 circumstances should the system allow a resource that is currently in use be deployed to respond to another incident.
- The system should not allow the modification of resource information while it is in use.
- A resource should not be added if it is outside state lines or in a foreign country.
- All available resources should and will be dispatched and/or be in use within 24 hours.
- Incidents reported on CIMT should be either on campus or within a 1 mile radius of campus.
- All money values entered into the system will be considered in US Dollars such as the cost of resources.
- All possible emergencies should have at least one resource, regardless if it's available or not.
- Users residing outside the country cannot offer their resources unless approved by admin.

Markup Annotations

Bold Underline: Form.

Bold Italics: Buttons / Input Fields.

Back to Top

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:

SELECT COUNT(1) = 1 AS valid_login FROM `User` WHERE username = ‘$Username’ AND password = ‘$Password’;

If *Username* exists but User.password != ‘$Password’, or *Username* is not found as indicated from the above query result (namely, valid_login shows 0): Go back to **Login** form, with error message

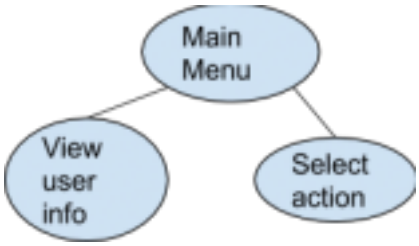
Else, log user in and go to **Main Menu** form.

Else, *Username* and *Password* input fields are invalid, display **Login** form, with error message.

Main Menu

Task Decomposition:

Back to Top



Abstract Code:

User successfully logged in from the **Login**

Run the **view user info** task by querying the user to display the user's '$Name' and user role ($name will stored in state):

Note: *...* indicates when a JS variable is being used in the context of SQL queries

Note: *Username* is the username that the user used to log in

displayName = SELECT display_name FROM user WHERE username = *Username*;

Show displayName on top of the page

Then, identify the role of the user

role = SELECT user_role FROM user WHERE username = *Username*;

Show phone number on top of the page if user is an resource provider

If role = "cimt", show $name and then , select cimt table, show $phoneNumber on top of the page.

phoneNumber = SELECT phone_num FROM cimt WHERE username = *Username*;

Show address on top of the page if user is a resource provider

If role = "resource_provider", show $name and then , select resource_provider table, $address

address = SELECT strt_addr FROM resource_provider WHERE username = *Username*;

Back to Top

Show email on top of the page if user is an admin

If role = "admin", show '$name and then , select admin table,$email

email = SELECT email FROM admin_user WHERE username = *username*;

Show email on top of the page

Run the **select action** task to display "Add Resource", "Add Emergency Incident", "Search Resources", "Resource Status", and "Resource Report" links.

Upon clicking:

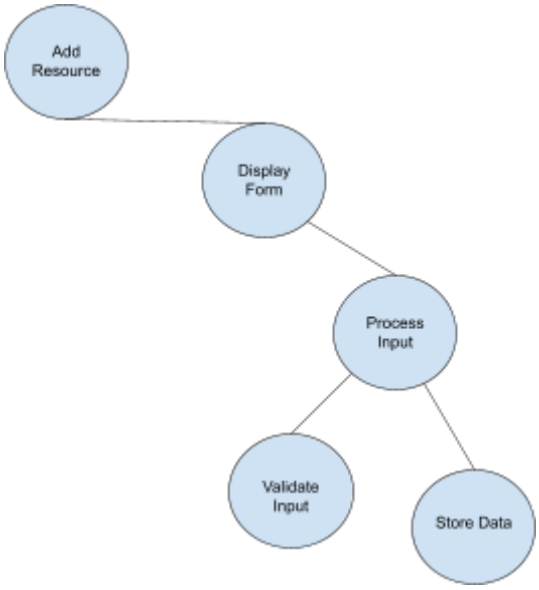
Add Resource button - Jump to **Add Resource** task.
Add Emergency Incident button - Jump to **Add Emergency Incident** task
Search Resources button - Jump to **Search Resources** task.
~~**Resource Status** button - Jump to **Resource Status** task.~~
Resource Report button - Jump to **Generate Resource 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 takes user back to the **Login** form.

Add Resource Task

Task Decomposition:

Back to Top



Abstract Code:

Note: *...* indicates when a JS variable is being used in the context of SQL queries

Note: *Username* is the username that the user used to log in

User clicked Add Resource from the **Main Menu**

Run **Add Resource task** via HTTP GET request to run **Display Form task**

Add Resource Form will include the following input fields, buttons, and drop down menu

- ***Plus Button*** - discards current form and runs **Add Resource Task** again
- Resource ID($id) - field is blank, empty on load
- Owner - display user's $name
- ***Resource Name***($name) **-** required field
- ***Primary Function***($primary_function) **,** required field, a pull down menu with the following menu options (that are queried from the database table):

  Get all the functions

  options = SELECT func_name from func;

  Loop thru options and display each option. Store the selected function in

Back to Top

“selectedFunction” variable

- #1: transportation (this is the default selected item in the pull down menu)
- #2: communications
- #3: engineering
- #4: search and rescue
- #5: education
- #6: energy
- #7: firefighting
- #8: human services

- ***Secondary Function*** ($secondary_function)**-** optional, displays same drop down menu from previous “Primary Function” minus the function already chosen (stored in function state)

Loop through options array and display all the options minus the selectedFunction

- ***Description***($description) **-** optional, user-entered text field
- ***Capabilities***($capabilities) **-** optional, user-entered text field
  - **Plus** Button - when pressed, saves the Capabilities input in a state (ARRAY), will display the previous input dynamically
- ***Distance***($distance)**:** optional field, user input field (numbers only)
- ***Cost / Unit:*** required
  - Cost($cost), User inputted field that is numbers only
  - Units($units), dropdown menu containing “day”, “hour”, “use”, will be queried from database table

unitsArray = SELECT DISTINCT unit FROM cost;
Loop through unitsArray and display each unit

- ***Cancel button*****:** this allows the user to exit the current view, returning to the Main Menu.
- ***Save button*****:** when the user clicks on this button, application will begin process input task

Process Input

- Starts with Validate Input task
  - Checks that required fields are not blank, display error message if they are
  - Checks that input fields are the appropriate data (cost and distance are numbers)
- After validation, begin store data task

Back to Top

Store Data

- Run query to store new resource into Resources table
    - Resource ID is auto-generated and stored in $id
    - Resource name is stored in $name
    - Primary Function - $primary_function (also increment the primary functions table)
    - Secondary Function - $secondary_function
    - Description - $description
    - Capabilities - $capabilities (store in capabilities table with primary key being the resource ID)
    - Distance - $distance
    - Cost - $cost
    - Unit - $unit

(React will check whether the optional inputs are filled in)

Note: *...* indicates when a JS variable is being used in the context of SQL queries

Note: *Username* is the username that the user used to log in

INSERT INTO emergency_resource (res_owner, res_name, prim_func_num, descript, capabilities, distance) VALUES (*owner*, *res_name*, *prim_func_num*, *descript*, *capabilities*, *distance*)

Submits request via HTTP POST

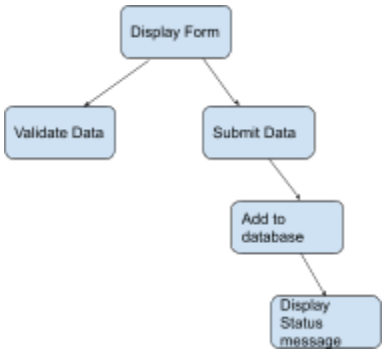
Returns database generated Resource ID

If save data task is successful, display “Resource Saved” message and shows unique resource ID at top of form

Back to Top

Add Emergency Incident Task

Task Decomposition:



Note: *...* indicates when a JS variable is being used in the context of SQL queries

Note: *Username* is the username that the user used to log in

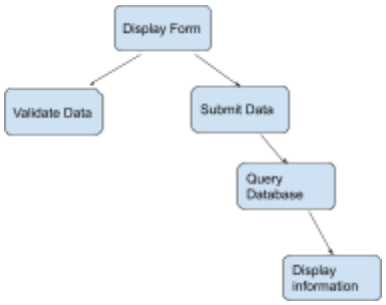
- Display Page
- New Incident Information as header
- ***Plus Button*** clears form/ jumps to a new add incident form
- Display following **Add Emergency Task Form**
    - **Category** (required) $category
        - Query database (category table) for all categories creates a dropdown with categories
            - SELECT cat_type FROM category
            - Drop down menu displays messages like
                - "must evac, secure lockdown"
                - "may evac, secure lockdown"
                - "no evac, limited lockdown"
                - "no evac, no lockdown"
    - **Date** (required mm/dd/yyyy) $date
        - User provided date
        - Valid Format: accepts
        - Invalid Format: alerts user to try again

Back to Top

        - **Description** (required) $description
            - User provided text describing incident
- On ***Save button*** click
    - Checks that date and description have valid inputs
    - Generates number for ID
        - +1 one of last added into database
    - Adds to the database
        - INSERT INTO incident (inc_date,descript, inc_owner, cat_id) VALUES (*inc_date*,*descript*, *inc_owner*, *cat_id*);
        - Category
        - Incident ID
        - Date
        - Description
        - Owner
    - Submit request via HTTP POST
    - If incident is successfully added, displays “Incident Saved” message and display unique incident ID from the database at the top of form
- On ***Cancel button*** click, brings back to Main Menu webpage

Search Resources

Task Decomposition



Abstract Code

- Display page
- “Search Resource” as heading
- ***Plus button*** clears form (basically the same as opening a new form)
- Form displayed next

Back to Top

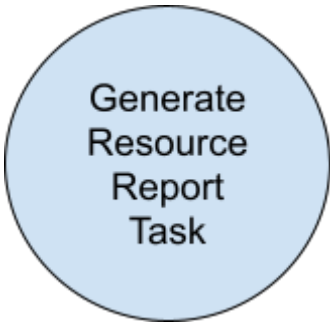
    - **Keyword:** (optional) $keyword
        - User inputs text
    - **Primary Function** (required) $primary_function
        - comes with a pull down menu with the following menu options (that are queried from the database table):
            - #1: transportation (this is the default selected item in the pull down menu)
            - #2: communications
            - #3: engineering
            - #4: search and rescue
            - #5: education
            - #6: energy
            - #7: firefighting
            - #8: human services
    - **Distance** (optional) $distance
        - User inputs numbers
        - Trimmed to .1 of a mile
- On click of cancel button return to main menu
- On click of search button, queries database using given criteria
    - Search database using substrings from $ResourceName, $Description, $Capabilities
        - SELECT emergency_resource.id AS 'Resource ID', emergency_resource.res_name AS 'Resource Name', emergency_resource.res_owner AS 'Owner', CONCAT('$',emergency_resource.price,'/',unit.name) AS 'cost/unit', emergency_resource.distance AS 'Distance'
        - FROM emergency_resource
        - JOIN unit ON emergency_resource.unit_id = unit.id
        - WHERE emergency_resource.res_name LIKE '%*keyword*%' OR emergency_resource.capabilities LIKE '%*keyword*%' OR emergency_resource.descript LIKE '%*keyword*%''
        - ORDER BY distance;
    - If distance is given, returns resources within the given range.
        - SELECT *
        - FROM emergency_resource
        - WHERE distance < $distance
        - ORDER BY distance;
    - If no fields given, displays all resources
        - SELECT *
        - FROM emergency_resource
        - ORDER BY distance;
    - Sort by distance from PCC

Back to Top

        - ORDER BY distance
- Displays returned table under form

Generate Resource Report Task

Task Decomposition:



Abstract Code:

Generate Resource Report Task

Note: *...* indicates when a JS variable is being used in the context of SQL queries

Note: *Username*/*CURRENT_USER* is the username that the user used to log in

- Using functions table, adds a row for each row in database
    - Select func_num, func_name
    - FROM func;
- Query all resources where $Owner = current user
    - SELECT *
    - FROM emergency_resource
    - WHERE res_owner = *CURRENT_USER*;
- Query to count all resources grouped by function number
    - SELECT prim_func_num, COUNT(*) AS 'Total Resources'
    - FROM emergency_resource
    - WHERE res_owner = *CURRENT_USER*
    - GROUP BY prim_func_num;
- Query to count all rows to get total resource count

Back to Top

        - SELECT COUNT(*)
        - FROM emergency_resource
        - WHERE res_owner = *CURRENT_USER*;
- Display table on page with information provided by resource report table
- Upon exit, returns user to main menu

Back to Top